

分 类 号: TP39
研究生学号: 2011544030

单位代码: 10183
密 级: 公 开



吉 林 大 学

硕士学位论文

视频监控系统传输技术的研究与实现

**Research and Realization of Technology of
Transmission of Video Surveillance System**

作者姓名: 李斌来

专 业: 软件工程

研究方向: 智能控制与嵌入式系统

指导教师: 赵宏伟 教授

培养单位: 软件学院

2013 年 4 月

视频监控系统传输技术的研究与实现

**Research and Realization of Technology of
Transmission of Video Surveillance System**

作者姓名：李斌来

专业名称：软件工程

指导教师：赵宏伟 教授

学位类别：软件工程硕士

答辩日期：2013 年 5 月 25 日

未经本论文作者的书面授权，依法收存和保管本论文书面版本、电子版本的任何单位和个人，均不得对本论文的全部或部分内容进行任何形式的复制、修改、发行、出租、改编等有碍作者著作权的商业性使用（但纯学术性使用不在此限）。否则，应承担侵权的法律责任。

吉林大学硕士学位论文原创性声明

本人郑重声明：所呈交的硕士学位论文，是本人在指导教师的指导下，独立进行研究工作所取得的成果。除文中已经注明引用的内容外，本论文不包含任何其他个人或集体已经发表或撰写过的作品成果。对本文的研究做出重要贡献的个人和集体，均已在文中以明确方式标明。本人完全意识到本声明的法律结果由本人承担。

学位论文作者签名：李润来

日期：2013 年 5 月 25 日

摘 要

视频监控系统传输技术的研究与实现

视频监控系统是监控领域的热门研究内容,其应用领域十分广泛,在医疗事业、军工国防、工业控制以及各种街道、商店、码头等公共场所都能够见到该系统的身影。视频监控系统主要经历了传统模拟闭路视频监控系统、基于 PC 机的集成化数字视频监控系统、基于嵌入式的远程视频监控系统三个主要的发展时代。随着无线网络的发展,视频监控系统出现了无线视频监控这一新的分支。无线视频监控系统具有灵活性强、高性价比、使用方便等特点,在安全、军工、交通、安防、公共景区等领域得到了广泛的应用。无线视频监控系统有着无需布线、携带方便、使用灵活等优点已经成为了视频监控研究的热点。随着社会科技的不断进步,视频监控系统正向着集成化、数字化、网络化、便携化等方向发展。

本文所研究的无线视频监控系统属于第三代视频监控系统,具有远程图像采集、远程传输以及实时监控等功能。该系统基于 ARM9 开发,由一个监控端和多个采集端构成。采集端通过数字摄像头捕获图像数据,并使用无线网卡将图像信息传输至监控端。用户在监控端可以进行多路视频共同监控,也可以选择某一特定分路进行监控,并能够将该屏幕中想要保留的画面保存至本地硬盘。监控端还提供了文件管理以及工具箱等辅助功能,具有良好的用户交互界面,方便用户对画面捕捉与记录。将来还会研究实现语音信号的传输,能够监听到监控场地的声音或通过监控端向监控场所发送语音广播信号。

根据无线视频监控系统自身的特点,本文对该系统的功能以及需求进行了分析,并对硬件以及软件两个方面做了的整体结构设计。通过对视频监控系统以及硬件、软件市场的调研,考虑了性价比、普及度等因素后,决定使用 ARM9 S3C2440 作为核心处理器进行开发。S3C2440 能够支持多路实时分屏播放视频,并且具有体积小、易于携带等优点。软件方面搭建嵌入式 Linux 操作系统并配置内核,安装 USB 摄像头与无线网卡等硬件设备的驱动。采集端方面,在 S3C2440 基础上外接 ZC301P 芯片 USB 摄像头作为采集设备。视频采集程序基于 V4L 设计,通过摄像头采集视频数据后,使用 MPEG-4 标准的对采集到的数据进行编码。然后使用 RTP/RTCP 协议通过无线网卡完成数据从采集端到监控端的传输。在监控端同样选用 S3C2440 处理器,采用多线程技术接收多路信号并把接收到的视频数据进行编码处理,最后使用基于 QT 的视频显示模块进行视频的播放。最终完成了本系统的开发,实现了多路监控功能。

关键词：

嵌入式系统 LINUX 视频监控 多线程

Abstract

Research and Realization of Technology of Transmission Video Surveillance System

Video surveillance technology is the hot content in the monitoring field, and the application is very wide. In the medical career, military defense, industrial control and various streets, shop, wharf and other public places to be able to see the figure of the system. Video surveillance system mainly through three main development era, the traditional analog closed-circuit video monitoring system, based on the PC machine integrated digital video monitoring system, based on the embedded remote video monitoring system. With the development of wireless network, Video monitoring system appeared a new branch wireless video monitoring. Wireless video monitoring system has high adaptability, high performance, convenient etc, widely used in Transportation, traffic, fire control, security, scenic spots, etc. Wireless video monitoring system has no wiring, easy to carry, use agile etc has become a hot spot in the study of the video monitoring. Along with the social science and technology progress, video surveillance system is developing in intelligence, networking, digital, HD etc.

The wireless video surveillance system which the article talks about belongs to the third generation of video monitoring system, With remote image acquisition, remote transmission and real-time control functions. The system is based on ARM9 development, Made up of a monitoring terminal and several collecting terminals. Collecting terminal capture the video signal with camera. and then sending the data to the receiving terminal by the wireless network card. Users in the monitoring terminal can be surveillancing multiple video in common, Also users can choose a particular channel to surveillance, and recording the suspicious things. Monitoring terminal also provides The assistant functions of file management and the tool box, having good user interface, easy of user to capture and record. In the future, we will realize the transmission of audio signal, can listen to the voice from the monitoring site, or send the audio signal to the specific video collecting side from the monitoring terminal.

Based on the characteristics of the wireless video surveillance system, In this article we analysis the function and requirement of this system, and make a structure design for two aspects of hardware and software. Investigate and survey video

surveillance system, hardware and software market, consider the cost performance, popularity etc, we decide to use ARM9 S3C2440 for development as a core processor. S3C2440 can support multiple real-time play video, and it has advantages of small volume, easy to carry etc. On the software side, build embedded Linux operating system and configuring the kernel. Install USE camera WLAN card's and other hardware device drive. On collecting terminal, based on S3C2440 jion up ZC301P chip USB camera as collecting device. Video capture program design based on V4L, after capture the video data by the camera, using mpeg-4 standard to encoding the data. Then use RTP/RTCP protocol conveying date from collecting terminal to monitoring terminal by WLAN card. In the monitoring terminal we also choose S3C2440 processor, use multithread technology to receive multichannel signal and decoding the data, then use the display module which based on QT to play the video. Finally completed the development of this system, Realize the multiple monitoring function.

Keywords:

Embedded ,LINUX, Video surveillance, multithreaded

目 录

第 1 章 绪 论.....	1
1.1 视频监控系统的背景与意义	1
1.2 视频监控系统的发展与国内外现状	1
1.3 无线视频监控系统	3
1.4 本文的主要工作	4
1.4.1 本文的主要研究内容	4
1.4.2 论文结构安排	4
第 2 章 无线视频监控系统的体系结构.....	6
2.1 无线视频监控系统的总体结构设计	6
2.2 无线视频监控系统硬件体系结构	8
2.2.1 嵌入式处理器的选择	8
2.2.2 视频采集设备	9
2.2.3 无线传输设备	9
2.2.4 视频显示设备	9
2.2.5 系统硬件体系结构的设计	10
2.3 无线视频监控系统软件体系结构	11
2.3.1 嵌入式操作系统的选择	11
2.3.2 视频数据采集技术	13
2.3.3 视频数据编解码技术的选择	13
2.3.4 无线传输技术	15
2.3.5 系统软件体系结构的设计	15
第 3 章 视频采集与视频编解码的设计.....	17
3.1 LINUX 下驱动程序加载方法.....	17
3.2 基于 V4L 的视频采集模块的设计与实现	19
3.2.1 V4L 应用常用数据结构及编程接口	19
3.2.2 视频采集的实现	19
3.3 视频编解码模块的设计	22
3.3.1 MPEG-4 视频压缩	23

3.3.2 基于 FFMPEG 的视频解码	23
3.4 Linux 下的多线程技术的研究.....	26
3.4.1 多线程编程接口	26
3.4.2 多线程模块的工作流程	27
3.4.3 多线程在 RTP 传输中的应用	29
第 4 章 基于 RTP/RTCP 的无线传输设计.....	31
4.1 RTP/RTCP 协议分析.....	31
4.1.1 RTP 实时传输协议.....	31
4.1.2 RTCP 控制协议	33
4.2 无线视频监控系统传输模块的实现	33
4.2.1 JRTPLib 库的安装.....	33
4.2.2 无线视频系统发送端的设计	34
4.2.3 无线视频系统接收端的设计	36
第 5 章 系统的移植与测试.....	38
5.1 嵌入式 LINUX 系统的移植	38
5.1.1 建立交叉编译环境	38
5.1.2 BootLoader 的移植.....	39
5.1.3 系统内核的移植	40
5.1.4 安装根文件系统	41
5.2 系统测试	41
5.2.1 系统运行效率测试	42
5.2.2 播放帧率测试	43
5.2.3 传输测试	43
第 6 章 总结与展望.....	44
6.1 工作总结	44
6.2 工作展望	45
参考文献.....	46
作者简介及在学期间所取得的科研成果.....	48
致 谢.....	49

第 1 章 绪 论

1.1 视频监控系统的背景与意义

随着现代科技的不断发展,计算机技术、网络技术、多媒体技术等已经得到了广泛的应用。其中,视频监控作为一种防范能力较强的安全防范综合系统就是这些技术的综合体现。该系统有着远程图像采集、远程传输以及实时监控等功能。以方便、直观、实时性强等优点被广泛应用于许多领域,如:医疗事业、军工国防、工业控制以及各种街道、商店、码头等公共场所。借助视频监控系统,人们可以实现对一些重要地点布置采集设备,并利用远端的监控设备查看现场的情况,以方便人们快速的根据实际情况来做出相应的处理。由于计算机技术的增强、网络传输速率的加快、嵌入式技术的发展以及图像处理技术的进步,视频监控技术也得到了长足的发展。

视频监控系统的发展历程基本分为三代:从最初的传统模拟闭路视频监控系统;发展到第二代基于 PC 机的集成化数字视频监控系统,这一代监控系统以数字信号代替了模拟信号;最后发展到第三代基于嵌入式的远程视频监控系统^[1],利用嵌入式概念实现更加灵活、方便的远程视频监控。比如在智能家居领域,人们可以通过智能终端,如智能手机、掌上电脑、笔记本等设备看到家中的情况并对家电进行控制。近年来,视频监控的应用领域正不断的扩大。

传统的视频监控系统存在着诸多缺点,如成本高、布线复杂、施工周期长、移动不便等。无线视频监控系统具有传统监控系统所不具备的诸多优点,如:体积小、性价比高、便利性强、无需布线等^[2],正逐渐成为视频监控系统的主流。在此背景下,本文选用了基于 ARM 的嵌入式平台,同时利用目前流行的无线网络技术,设计了一种基于嵌入式的无线视频监控系统^[3]。

1.2 视频监控系统的发展与国内外现状

视频监控技术从最初出现到现在发展了不到三十年的时间,随着计算机技术、多媒体技术、网络技术等技术的发展,视频监控技术也同样有着高速的发展。已经从早期的模拟监控发展到了现在的网络监控。

视频监控技术的发展可以分为以下三个阶段:

第一代传统模拟闭路视频监控系统(CCTV)。它主要由摄像机、录像机、监控器等设备组成。这种系统捕获模拟信号并由电缆等有线介质,从采集端传输到监控端。这种监控方式优点为价格低廉,架设方便,操控简单等。但是模拟监控系统同样存在一些弊端:该系统由于要借助电缆传输,所以只能够监视本地地点。另外,当时的录像机无法记录下高质量的视频,复制录像会再次降低录像质量。而限制该系统发展的另一原因是该系统采集的是模拟信号,而图像处理技术无法对模拟信号进行编解码等工作。由于以上原因,这种传统的模拟视频监控技术逐渐被飞速发展的现代社会所淘汰^[3]。

第二代视频监控系统为基于 PC 机的集成化数字视频监控系统。19 世纪 90 年代,采集技术、编图像处理技术得到了飞速的发展、同时 PC 机的普及使这种监控系统得到了广泛的应用^[4]。它通过远处的摄像机或其他视频采集设备,采集视频信息并通过视频压缩卡等处理采集到的信息。再由通信网络将信息传到监控端。该系统的最大特点就是用数字信号代替了模拟信号。与第一代模拟监控系统相比,它的功能性、灵活性较强。同时视频的质量以及传输速度等方面都有了很大的提高^[4],但是,该系统也同样有着很多缺点:如稳定性不够高、结构复杂、可靠性低;同时,软件的开放性不高,传输距离受限,PC 机端需要专人管理。所以,这种监控方式也不是最为理想的方案。

第三代视频监控系统为基于嵌入式的远程视频监控系统。20 世纪 90 年代末,嵌入式技术、图像处理技术、网络技术、电子信息与通信技术得到了飞速的发展,这促使了新一代视频监控系统的兴起。第三代视频监控系统不但继承了第二代系统的数字图像采集功能,另外还具有上一代所没有的网络传输、图像处理等特点。该系统的终端采用嵌入式设备和嵌入式实时操作系统,将采集进来的图像进行编码等处理,并通过网络传输技术传网络中去,实现网络监控。嵌入式监控系统以成本低、携带方便、稳定性高、性能较强等优势,结合 TCP/IP 技术,可以直接连入以太网。

嵌入式网络视频监测系统具有如下优点:

1. 嵌入式网络视频监测系统可以直接连入网络,即插即看,加入新的监控端只需添加新的 IP 即可,做到降级成本、方便使用。
2. 嵌入式视频监测系统采用嵌入式实时操作系统,该系统大大提高了实时性、可靠性、稳定性,并且无需专人管理。

由于这种新型的监控系统优点明显,所以一经出现就得到了较高的关注,发展迅。

1.3 无线视频监控系统

目前,视频监控正向着便携化、数字化、以及集成化发展。在网络视频监控系统的基础上,结合无线网络技术,出现了一个新的分支:无线视频监控系统。无线视频监控系统有着更高的集成度、灵活度,同时有着更强的携带性,这种系统建立于嵌入式远程视频监控系统之上,整个系统不需要布线,各种信号都由网络传输。目前,这类系统已经成为视频监控中的热点研究对象。

无线视频监控系统的应用领域十分广泛。在某些特定的环境下,如:要搭建监控的系统不便于布线或不可能布线,例如在港口,旷野或山区等地区,布线的成本相当昂贵甚至无法实施,这时便可以采用无线视频监控系统,省去了布线所带来的困难,同时完成了视频信号的传输。可以看出无线视频监控系统的搭建相对有线监控要便捷很多,并且成本更低。

无线视频监控系统基本分为三个部分组成:采集端、传输系统、监控端。采集端将视频信息采集到嵌入式板中,利用数字图像处理技术将信息数字化并压缩后交由传输系统,通过无线网络技术将信息传导监控端,监控端可以为嵌入式设备或 PC 机等,接收无线信号后利用图像处理技术处理之后便能够在显示屏上播放监控画面。

在视频采集端,目前较为流行的是采用嵌入式处理器加数字摄像头的组合。而监控端方面,人们可是使用各种智能设备作为接收端,如掌上电脑、智能手机、笔记本等。在传输系统中,无线网络技术有很多种,如:WIFI、3G 通信技术、通用分组无线业务、蓝牙以及 Zigbee 技术^[4]。本文选用的是 WIFI 技术。WiFi 是 WirelessFidelity 的简称,指的是“无线相容性认证”,同时这也是一种属于无线联网范畴的技术,是一种与蓝牙技术一样的应用于家庭与办公室的短距离无线技术,在此之前的联网方式都是通过网线连接电脑,而现在则是通过无线电波来联网。从本质上来说 wifi 是一种商业性的认证,其产品符合 IEEE 802.11a/b/g/n 无线网络规范,它是当前全世界范围内应用最为广泛的无线局域网络标准,其工作频段一般都是 2.4Ghz。时至今日,IEEE 802.11 这个标准已经被业界统称为 WiFi。从实际应用的层面上来说来说,WIFI 技术可以帮助用户访问电子邮箱、网站和主流媒体,也就是说它为其使用用户提供了一种访问互联网的较为方便的无线途径。WIFI 的组建方法简单,没有布线的需要,特别适合需要移动办公的用户。相对于同样的短程无线技术作用半径更大。当前世界上一直最新的 WiFi 其作用半径最远可以达到三百米左右左右,而蓝牙的电波半径却只有区区十五米左右。Wifi 的最高传播速度可以达到 37.5Mbit /s,丝毫不逊于现有的无线网络,可以满足现在的绝大多数个人、家庭或则社会信息化的基本需求。

以上就是无线视频监控系统的的基本构成,随着嵌入式技术以及无线网络的不断进步,无线视频监控系统会取得进一步的发展^[5]。

1.4 本文的主要工作

本文主要对基于嵌入式无线视频监控系统中所涉及到的关键技术进行研究,其中包括:音视频采集、音视频编解码、无线传输技术、多线程技术等。本文通过对这些技术的比较进行了选择以及编程实现。另外对嵌入式系统的硬件体系结构、软件体系结构进行了设计。

1.4.1 本文的主要研究内容

本文主要研究的内容有:

- 1.根据无线视频监控系统的基本思想,对系统中所涉及的各种关键技术做了比较,提出了无线视频监控系统的硬件总体设计方案及软件设计方案。
- 2.比较了各种嵌入式系统,选择了 Linux 作为本文的开发环境,建立的交叉编译环境,完成了嵌入式 Linux 内核的剪裁、移植。
- 3.介绍视频压缩基本理论、MPEG-4 视频压缩标准和流程,使用 Video4Linux 编程,设计并实现了对 USB 摄像头控制及图像采集功能。采用 MPEG-4 标准对采集到的视频进行了压缩处理。
- 4.利用 RTP/RTCP 实时传输协议完成无线网络通信协议。
- 5.完成向 MINI2440 的 ffmpeg 视频编解码移植过程。
- 6.研究 Linux 下多线程原理,编程实现视频采集、传输、播放的多线程功能。

1.4.2 论文结构安排

根据本文所做的工作对全文做了如下安排:

第 1 章主要为视频监控技术的背景、意义的介绍,视频监控系统的的国内外发展状况以及无线视频监控系统的概要。

第 2 章主要为系统的总体设计,分为硬件体系结构和软件体系结构两个部分,对每个部分中的技术选择以及方案设计做了详细的讲解。对每个模块进行了划分。

第 3 章主要为系统的视频采集、视频编码模块以及多线程的实现。完成了

USB 摄像头的驱动移植，通过 V4L 进行了视频采集。并利用 MPEG-4 编码标准对采集到的数据进行压缩编码。最后研究了 LINUX 多线程技术。

第 4 章主要内容研究无线视频监控系统中所用到的无线传输技术 RTP/RTCP。

第 5 章对系统进行了整体的移植与环境搭建，并进行了测试工作。

第 6 章为总结部分，并对下一步的工作重点进行了展望。

第 2 章 无线视频监控系统的体系结构

2.1 无线视频监控系统的总体结构设计

无线视频监视系统总体结构主要分为图像采集端、视频图像接收端两部分，这两部分通过无线网络传输技术实现信息的互通，系统组成如图 2.1 所示。由 USB 摄像头采集视频数据后，将视频数据存放于 ARM 板上的存储器中，随后利用视频处理技术将视频数据进行编码压缩，压缩后能够更快的将数据在无线网络中传递到接收端。接收端接收到视频数据后将其进行解码处理后，便可以再 LCD 显示器上实现播放。

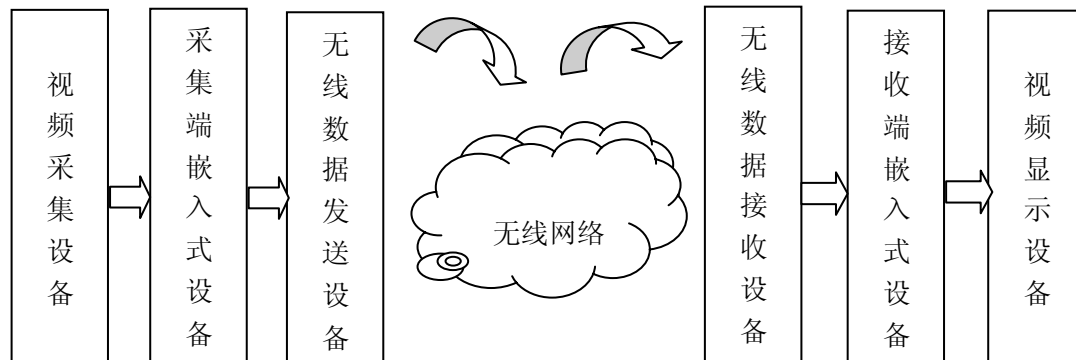


图 2.1 无线视频监视系统构成

Fig2.1 Architecture of video surveillance system

将无线视频监视系统按照功能实现来划分，可将系统划分为：视频图像压缩功能、视频流解压功能、视频图像采集功能、以及视频播放功能。如图 2.2 所示。

整个系统的工作流程与工作原理如图 2.3 所示，S3C2440 微处理器和 USB 摄像头构成了整个系统的视频采集终端。视频采集设备为 USB 摄像头，通过摄像头采集到视频数据之后传到嵌入式芯片处理。系统方案中选用 MPEG-4 压缩标准来完成视频的编解码。之后通过 RPT/RTCP 无线网络传输协议从采集端传到监控端。监控端由 LCD 显示器和 S3C2440 构成，监控端收到压缩过的视频数据之后，再次利用 MPEG-4 进行解码。最后，用户便能够随时通过 LCD 监视现场的情况。

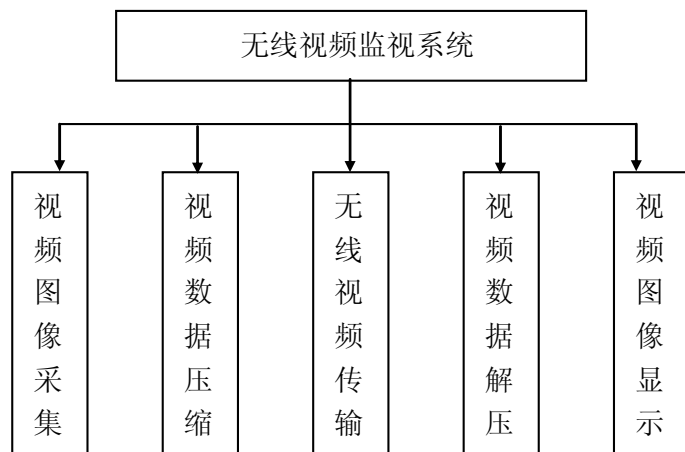


图 2.2 无线视频监视系统功能框图

Fig2.2 Functional diagram of wireless surveillance system

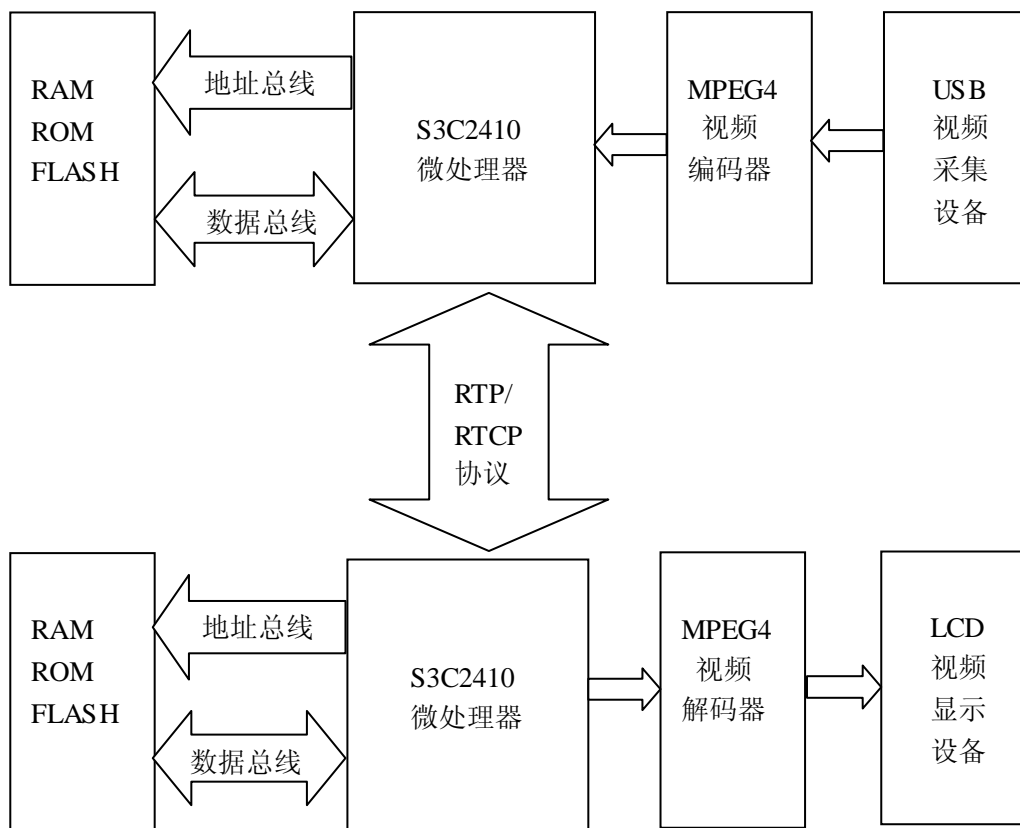


图 2.3 系统总体结构图

Fig2.3 Diagram of system Architecture

2.2 无线视频监控系统硬件体系结构

在硬件体系结构的设计之前,根据用户的应用需求,首先选择合适的嵌入式处理器以及扩展外围设备。我们需要对现有的硬件设备进行对比,从性能、成本等方面综合考虑,硬件设备的选型得当直接影响着系统的成败。根据上面所提到,我们需要为嵌入式处理器、视频采集设备、无线收发设备、视频显示设备等进行选择、设计。

2.2.1 嵌入式处理器的选择

近些年,嵌入式处理器发展飞快。目前比较知名的嵌入式处理器有:摩托罗拉的 68000/Coldfire、IBM 的 PowerPC、Sun 的 spare 和 ARM 的 ARM 系列等。表 2.1 列出了几种常见处理器及其价格和应用领域。

表 2.1 常见处理器特性比较

Table 2.1 processor characteristics

处理器类型	价格	主要应用领域
ARM	低	主要应用于工业计算机
X86	高	手持设备、可视电话、网络监控、医疗电子设备等
PowerPC	高	多媒体控制应用功能的通信与网络设备
Coldfile	高	通信设备、音频、工业控制等

无线视频监控系统的开发过程中,如何选择这些不同型号的产品十分重要。在整个系统中,嵌入式处理器相当于整个系统的核心,它需要参与到采集图像、编解码以及网络传输等多种工作中。因此从性能、稳定性、扩展性以及性价比多方面原因。目前,ARM 处理器的应用广泛,并且拥有庞大的客户群和完善的文档系统,因此本文选择 ARM 处理器中 SAMSUNG 的 S3C2440 作为本次开发的嵌入式处理器,同时选择友善之臂的 MINI2440 作为搭载此处理器的开发板。S3C2440 内核为 ARM920T,具有 16/32 位 RISC 结构,其 CPU 运行频率为 400MHz,最高可达 533Mhz,其性能完全可以实现图像采集、传输以及播放等工作需要。板载 64MB 的 SDRAM 内存,32bit 数据总线,SDRAM 时钟频率高达 100MHz,ARM920T 处理器内置 MMU(内存管理单元),能够对虚拟存储进行管理,支持 Windows CE、Linux 等嵌入式操作系统,该处理器采用的是 AMBA 总线结构,提高了硬件的性能。该处理器最大支持 256K 色的 TFT 液晶显示屏,完全能够支持无线视频监控系统的功能需求。该处理器还提供了 CMOS 摄像头

接口并可以外接 USB 设备。三星公司的 S3C2440 芯片市场价格只有 50 元左右，成本较低也是本文选择此芯片的一个原因。

2.2.2 视频采集设备

本文所选用的视频采集设备为 USB 摄像头，其芯片为 zc301，是目前与嵌入式 LINUX 系统兼容较好的一款高性能摄像头。支持 CMOS 感应技术，具有较高的 VGA(35 万像素)感应器，能够捕获 130 万像素图片，分辨率为 640X480DPI。此外 zc301 芯片还具有图像自动校正、自动曝光、自动白平衡等先进技术。采集到的图像变换平滑、层次变现实力强、动态画面流畅并且切价格低廉，能够完美兼容 S3C2440 处理。是本系统的一个较好选择。

2.2.3 无线传输设备

本系统采用多路传输，并且对于视频的质量有一定的要求，所以原来的基于 IEEE802.11b 的无线局域网其速度为 11Mbps，其速度不能满足本系统的要求。因此本文选择了 RT3070 芯片的无线网卡，此网卡采用 IEEE802.11n 传输标准，兼容 IEEE 802.11g 传输协议。802.11g 标准的为 54Mbps，是 IEEE802.11b 的五倍。该网卡有效覆盖面积为 200M，相对监控范围较为，若想实现远程监控，可将其连入 Internet。该网卡有两种两种工作方式：Infrastructure（集中控制式）、Ad-Hoc（对等式）。USB 接口完全兼容 S3C2440 嵌入式芯片，其传输速率可以达到 150Mbps。

2.2.4 视频显示设备

根据无线视频监控系统自身的特点，由于其便于携带、灵活的特性，视频播放设备不可以太大，而太小的屏幕也无法清晰的使用户观看到监控现场。所以本文选择了 NEC7.0 寸 TF 以及 T 电容式触摸屏，它可以较好的完成分屏播放功能，显示色彩为 256K，大小为 800X480。并且根据用户的需要，只需更改一些参数便可以更换成更为小巧的 NEC3.5 寸全新真彩 LCD，其大小为 240x320，以支持掌上设备。

2.2.5 系统硬件体系结构的设计

在核心嵌入式处理器以及外围设备选择完毕后，需要根据所选择硬件设备的特点以及工作方式对整个系统进行硬件体系结构的设计。无线视频监控系统硬件体系结构图如图 2.4 所示。

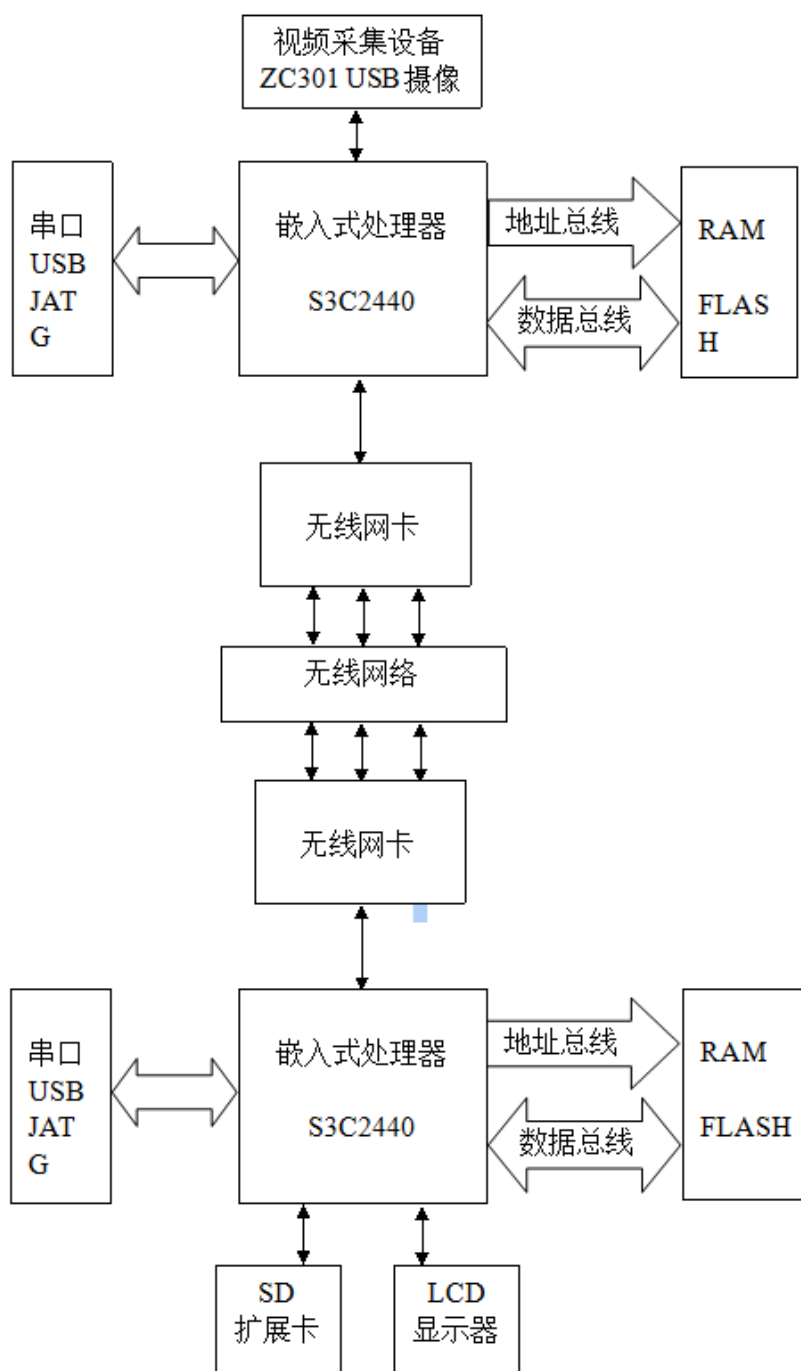


图 2.4 无线视频监控系统硬件体系结构图

Fig 2.4 Wireless video monitoring system hardware structure diagram

该系统的核心为 S3C2440 微处理器,并扩展了 64M SDRAM,以保证 LINUX 系统以及视频监控系统的正常运行。同时还为处理器扩展了 1GB 大小的 Nand Flash,它的特点为掉电不丢失,可以利用它来存储视频监控系统以及视频图像数据。此外,处理器还带有 2M 掉电不易失的 Nor Flans,更高为系统进行方面的更新操作。USB 摄像头通过 USB Device 连接到处理器,S3C2440 自带设备控制器与 USB 主机控制器,它支持低速与高速 USB 设备的传输,通过其引出 USB 连接口特别方便。在监控端,除摄像头外,无线网卡另需一 USB Device 来连接至处理器。摄像头为之前已经选择的 Zc301 芯片双飞燕 PK-635 摄像头,网卡为 RT3070 芯片的 FW50U 无线网卡。通过 JTAG 接口,可以对开发的系统的 Flash 进行更新修改,另外,JTAG 接口能够在调试程序时进行单步调试。监控端的方面,采用 NEC7.0 寸 TFT 真彩触摸式显示屏,它具有 256K 色以及 800x480 的分辨率。监控端同样配备了 1G 的 Nand Flash,以保证其能够保存监控到的视频数据。但考虑到有时会有长时间监控的情况,处理器另外可以扩展 SD 卡存储接口。用户可根据自己的需要自行添加 SD 卡。根据以上设备的选择和方案,本文选择了友善之臂 MINI2440 开发板作为无线视频监控系统的开发平台,并加以布设,完成了无线视频监控系统的硬件布置。

2.3 无线视频监控系统软件体系结构

无线视频监控系统之前,同样需要像选择硬件设备一样,为软件系统在稳定性、安全性、运行效率、容错性等方面做一系列的比较以及取舍。在软件体系结构中,操作系统是核心。操作系统是管理计算机硬件资源,控制其他程序运行并为用户提供交互操作界面的系统软件的集合。操作系统是计算机系统的关键组成部分,负责管理与配置内存、决定系统资源供需的优先次序、控制输入与输出设备、操作网络与管理文件系统等基本任务。之后,在操作系统以及硬件设备的支持上,选择合适的视频采集技术、视频编解码技术、无线网络传输技术、多线程技术等。

2.3.1 嵌入式操作系统的选择

嵌入式操作系统如其名字一样是指用于嵌入式系统的操作系统。嵌入式操作系统是一种用途广泛的系统软件,一般这种系统自身会带有硬件驱动、系统内核、接口驱动等,并提供图形化界面。嵌入式系统需要对系统本身的软件、硬件等资

源进行合理分配。同时还要负责系统的进程、线程调度以及任务执行等工作。它能够通过安装、删减某些模块来将系统的功能与资源调整为所在系统的特征^[6]。目前在嵌入式领域广泛使用的操作系统有：嵌入式 Linux、Windows CE、VxWorks 等，以及应用在智能手机和平板电脑的 Android、iOS 等。

S3C2440 具有内存管理单元（MMU），支持 Windows CE 与嵌入式 Linux 等嵌入式操作系统，我们会分别从几个方面对目前比较流行的几种嵌入式操作系统进行比较，并从中选出一个最适合的作为本系统的核心。嵌入式操作系统选择的基本特征为体积小、执行效率高、剪裁性和可移植性较好。并且要考虑其能否够得到硬件平台的支持，能否实现所要实现的功能。另外，嵌入式操作系统分为商用型与免费型。商用型有着诸多优点：如运行稳定、有较好的技术支持和售后服务等^[6]。其中一些更是用于航空航天、军事工业等高尖端技术领域。但其价格十分昂贵。免费型操作系统不但价格免费，它的核心源代码也是公开的，这使得系统的软件有更好的兼容性，而在视频监控系统上的表现也是分满意。

1. VxWorks 操作系统。该系统是美国 WindRiver 公司研发的，是一款稳定性、实时性较强的嵌入式操作系统。该体系基本支持所有目前流行的嵌入式处理器，具有良好的可持续开发潜力以及人性化的开发环境。该系统还支持 POSIX、ANSI C 等工业标准。由于此操作系统有着优秀的稳定性以及高实时性，在军事、航空航天、医疗等高尖端科技上都会看到其身影。但 VxWorks 并不是开源软件，开发费用十分昂贵。

2. Windows CE 操作系统。该系统为美国 Microsoft 公司开发，是一款专为掌上电子设备所使用的操作系统。这种操作系统能够将嵌入式技术与大型的 Windows 桌面技术相结合。Windows CE 将条码扫描与数据终端整合在了一起，该系统可离线操作并带有电池。具有自动存储、自动处理、实时采集、即时显示、自动传输等功能。提供了较高的可靠性、实时性数据。使用该系统的设备一般有着便携、灵巧、方便并适于手持等特点。目前 Windows CE 的最新版本为 Windows Embedded Compact 7，该版本的内核相比之前的版本有较大的提高，系统内全部的元件 EXE 都转换成了 DLL。在智能手机 Smart Phone 和掌上电脑 Pocket PC 上所使用的 Windows Mobile 系统便是这种内核^[7]。Windows CE 同样需要版权使用费用。

3. LINUX 操作系统。Linux 是一种开放源码的类 Unix 操作系统，Linux 有许多不同的版本，但各个版本所使用的内核基本是一样的。Linux 支持各种类型的硬件设备，如 PC 机、手机、服务器、超级计算机等。Linux 是一个领先的操作系统，世界上运算最快的 10 台超级计算机运行的都是 Linux 操作系统。其实，Linux 只是其内核的一种表述，并不是一个完整的操作系统。而人们现在经常用

Linux 这个词来默认表示带有数据库以及各种工具的整个操作系统了^[8]。嵌入式 LINUX 系统是其代码完全开放,因此在修改、剪裁等工作上操作更为简单。开发者能够根据系统的需要去掉不需要的功能,并优化所需要的功能来完成系统的二次开发,从而降低整个系统开销与能耗。另外一个优点为成本低,他不但本身系统免费,其系统上所使用的应用也同样多数为开源免费,相比其他商用系统,能够节省几百到几万美元不等。Linux 具有丰富的开源代码与开发工具。开发者可以根据需要获取现有的技术代码,在缩短了开发时间的同时又节省了开发成本。Linux 还具有十分强大的网络功能,支持多种网络协议并提供多种网络服务功能。

综上所述,在比较了各个操作系统的效率、成本、稳定性等因素之后,最终本文选择了 LINUX 系统。

2.3.2 视频数据采集技术

视频采集的过程就是,将采集设备所采集到的模拟信号转换为数字信号。本文所采用的视频采集技术为 Linux 内核所提供的 Video4Linux,简称 V4L。它是一些视频系统和音频系统的基础,常被使用在需要采集图像的场所,如视频监控、可视电话等,是基于嵌入式 Linux 系统开发中经常使用的系统接口。

V4L 是 Linux 中关于视频设备的内核驱动,它为了 Linux 下的视频采集设备驱动的编写提供统一的接口和一套规范。另外,因其是开源方案,所以又可以节省一部分开发成本。V4L 提供了多种标准与驱动,并提供了一套 API 接口,有利于开发与扩展。目前 V4L 已经有其升级版本 V4L2,它修复了 V4L 的一些 BUG 并提供了一些额外的操作函数。V4L2 一般兼容 V4L,所以很多程序可以用 V4L 接口。在本系统中 V4L 已经足够使用。

2.3.3 视频数据编解码技术的选择

在图像数据采集完毕后,其数据量是非常庞大的。而实际使用中,系统的存储容量以及传输带宽都是有限的。因此,必须利用视频压缩、编解码技术减小视频数据的规模,以便于视频的存放以及传输。选择一个好的视频压缩编码技术格外重要。

目前制订视频编解码标准的国际组织有 ITU-T 与 ISO/IEC 两大组织,其中 ITU-T 主要制订的编码标准有: H.261、H.263、H.264 等。而 ISO/IEC 的制定的视频编码标准有 MPJEG、MPEG-1、MPEG-2、MPEG-4 等,如表 2.2 所示。

其中 JPEG、MPEG-1、MPEG-2、H.261 以及 H.263 为第一代视频编解码技术。第一代视频编码技术是基于传统的信号处理理论,主要为除去数据的冗余,特点是算法可靠性高。但它并没有考虑到使用者的主观特性,如:视频信息的含义以及重要程度。而第二代视频编码技术目标在于如何除去视频内容的冗余,充分考虑了人眼的视觉特性,利用人的视觉系统的特点来提高编码效率,其特点为编码效率高,算法复杂。MPEG-4 就是采用的第二代视频编码技术。

表 2.2 常见视频编解码技术
Table2.2 Video codec technology

	码率	压缩比	应用领域
MPEG-1	0.8kbps-1.5Mbps	30-50:1	VCD、Internet
MPEG-2	3Mbps-100Mbps	50-200:1	数字广播、DVD、专业视频处理
MPEG-4	10Kbps-1200Mbps	100-300:1	Internet、视频剪辑
H.261	64Kbps-1.5Mbps	30-100:1	视频电话
H.263	10Kbps-384Kbps	100-300:1	视频电话
H.264	小于 1Mbps	>200:1	数字电视、卫星广播

根据图表,我们可以看出 MPEG-4 与 H.264 具有高压缩比,但是 H.264 需要较高的硬件运算。MPEG-4 算法具有高灵活性与扩展性,并且对硬件要求也相对较低。综合考虑本系统自身的特点、使用领域、以及硬件性能等原因,MPEG-4 标准是一个比较适合的选择。

目前,主要有三种较为流行的基于 MPEG-4 解码器,分别为 MS MPEG V3、Xvid 和 FFMPEG。MS MPEG V3 是微软开发的解码器,应用于 Windows 的默认组件,如 Windows Media 多媒体播放器,借助 V3 技术能够生成 ASF 格式的视频。此编码器性能较高,但其不是开源技术,能本较大。

Xvid 为开源解码技术,它只支持 MPEG-4 解码,是公认的 MPEG-4 最好的编解码技术之一。然而由于它的 VLC 解码优化并不是很理想,所以解码效率无法达到理想状态。另外,它对于硬件的要求也较高。

FFMPEG 同样是开源解码技术,支持多种编解码标准,并且由于编码模块的优化其编解码速度快于 Xvid。对于处理的要求相对较低,所以综合以上原因,本文选择了 FFMPEG 作为视频的编解码器。

2.3.4 无线传输技术

在本系统中对于监控的实时性有较高的要求,除了对采集到的时间进行压缩之外,还需要配合一个好的无线传输技术来满足系统的实时性。所以本文选择了 RTP (Real-time Transport Protocol, 实时传输协议)。RTP 是目前为解决流媒体实时传输的一个较好方法。过去的网络传输协议由于功能的需要,一直在可靠性与安全性方面受到较高的重视,而 RTP 协议则更加倾向于实时性。RTP 协议一般与 RTCP 控制协议一起使用。在嵌入式 Linux 系统环境中有一些开源的库可以使用,如 LIBRTP、JRTPLIB 等,我们在开发程序时可以方便的使用这些函数进行开发。

2.3.5 系统软件体系结构的设计

分析并确定了要使用的基本软件技术之后,像硬件体系结构一样,同样要设计出一套软件体系结构。本文以嵌入式 Linux 为整个软件系统核心,其他各个应用项目都是在 Linux 上进行开发的。在视频采集方面本文通过 V4L 技术,负责视频采集设备的控制。将采集到的视频数据模拟信号转化为数字信号;选择 MPEG-4 作为视频编解码标准,使用 FFmpeg 开源编码程序对采集到的视频数据进行编码;通过 RTP/RTCP 协议与多线程技术完成无线传输;最后将视频解码,使用 QT 完成播放功能。

根据所选择的软件技术、无线视频监控的特点、功能。我们能够绘制出一套数据流程图,如图 2.6 所示。从图中可以看出,本系统的采集端与监控端为多对一的关系。正如在日常使用监控系统时,能够在接收地点监控各个不同区域。采集端完成了视频数据的采集、编码与发送,其数据流程为:首先采集端与发送端同时初始化硬件设备并设置端口号;采集端发出视频捕获命令;USB 摄像头接到命令后开始视频捕捉;之后将视频数据传递到处理器,处理器再根据编码标准对数据进行编码;最后通过无线网卡将压缩后的数据传送到监控端。监控端的数据流程为:首先进行设备的初始化并设置本地接收端口,注意:本地接收端口并不是一个,因为采集端与监控端的多对一关系,端口数等采集端个数相等。无线网卡接收完数据之后,将其传送至处理器。监控端主程序判断当前是否为多路监控,如果是,将每一路采集到的视频信号数据交由不同的线程,之后在 LCD 显示器上对应的位置播放。

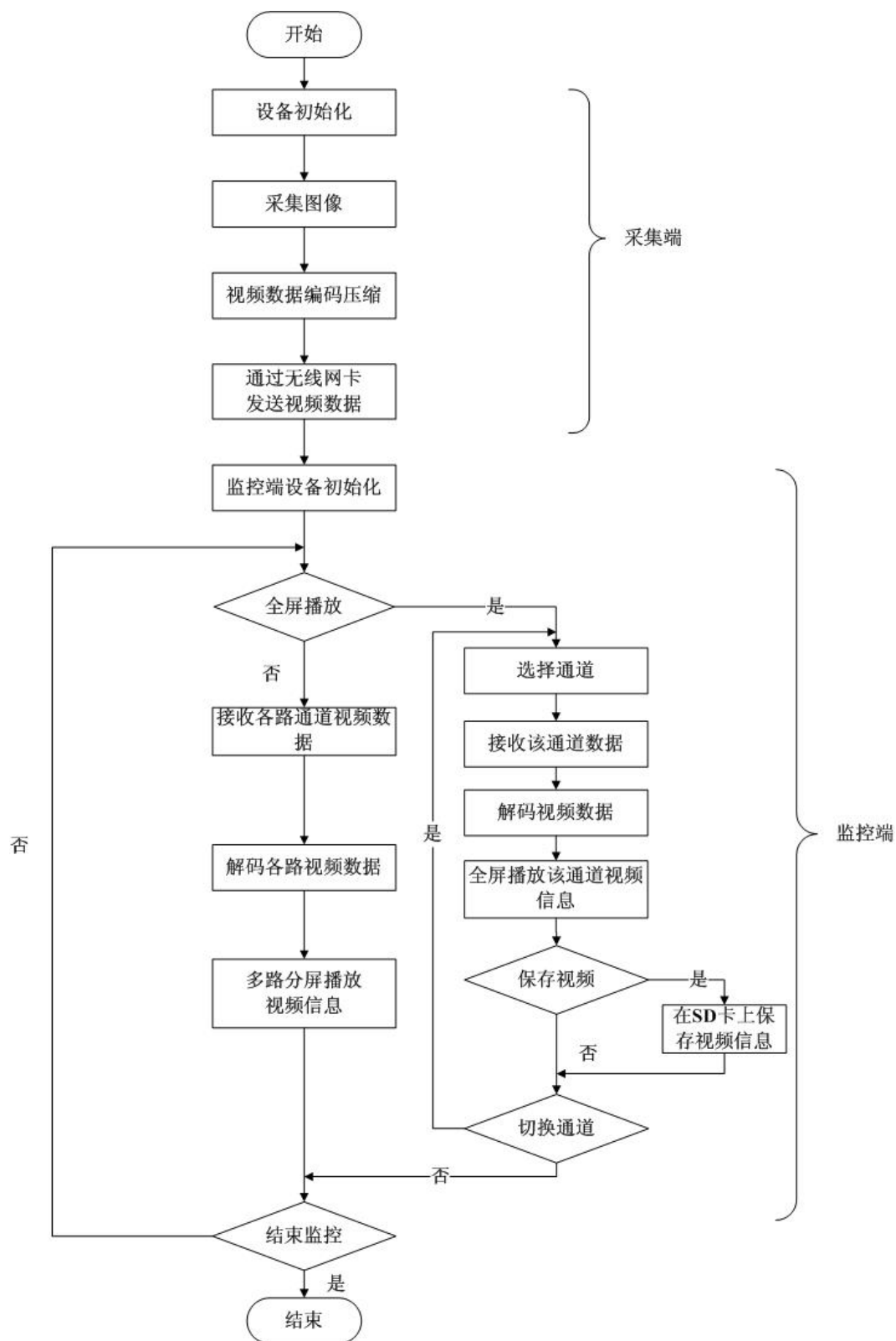


图 2.5 多路无线视频监控系统数据流程图

Figure 2.5 Data flow chart of wireless video monitoring system

第 3 章 视频采集与视频编解码的设计

3.1 LINUX 下驱动程序加载方法

一个完整的系统由硬件以及软件构成,而硬件与软件必须相互搭配才能使系统达到最大效率。硬件处于系统的最底层,用户通过应用软件的不同功能对硬件进行操作。而要是硬件与应用软件建立一个互通的桥梁,就需要驱动程序的帮助。

驱动程序是应用程序和硬件之间的软件层,它及能够对硬件进行控制又可以完成交互数据功能,同时它还为应用程序屏蔽了硬件设备的具体细节。操作系统可以通过它提供的统一化函数来实现对硬件的操控。驱动程序可以以静态方式编译入内核也可以动态加载到内核,前者一般用于开发阶段,后者则一般用于产品完成阶段。利用驱动程序,应用程序可以将硬件设备视为一个普通文件,并对硬件设备加以管理、操控^[9]。在一个嵌入式系统中,大多数硬件设备都需要配备一个驱动程序。

在嵌入式 LINUX 系统中,无法像一般的桌面 LINUX 一样使用 insmod/rmmod 来灵活的加载驱动。所以,通常以静态方式来加载内核。一般具有具体设备意义的文件称为设备文件,用来区别于传统意义上的文件,他们一般存放在/dev 目录中。当某一设备文件被上层应用打开时,Linux 内核就会使用 file 结构对其进行标识,内核会使用 file-operations 结构体来实现具体函数,所产生的函数便能够对设备实现具体操控。file-operations 结构体一般包括 open, close, ioctl, read, write 等函数。

如此,我们知道在 Linux 下编写驱动程序的原理和思想类似于其他的 Unix 系统,但它与 Windows 系统中的驱动程序有很大的区别,在 Linux 环境下设计驱动程序、思想简洁、操作方便、功能强大,但是支持函数较少,只能依赖 kernel 中的函数,有些操作需要自行编写。在 Linux 系统中,一个完整的 USB 驱动程序包括:USB core、USB 主控制器驱动、USB 设备驱动 3 个模块^[10]。USB core 是一个包含很多数据结构和函数的 API,它支持着 USB 主控制器和 USB 设备驱动。USB 主控制器的任务是分工主机对 USB 信息的传输,并提供了设备与 USB core 数据传输的通道,同时核对相关的数据结构和注册功能。

下面进行 USB 摄像头驱动的加载,本文选用的是双飞燕摄像头,其芯片为市面较为常见的中星微 ZC301,支持 spca5xx 驱动。本文 LINUX 内核为 2.6.32.2,内核中已经集成了本摄像头的驱动,在设计过程中只需适当的进行配置即可,节省了开发时间。图 3.1 中为本系统 USB 视频采集子系统组成。

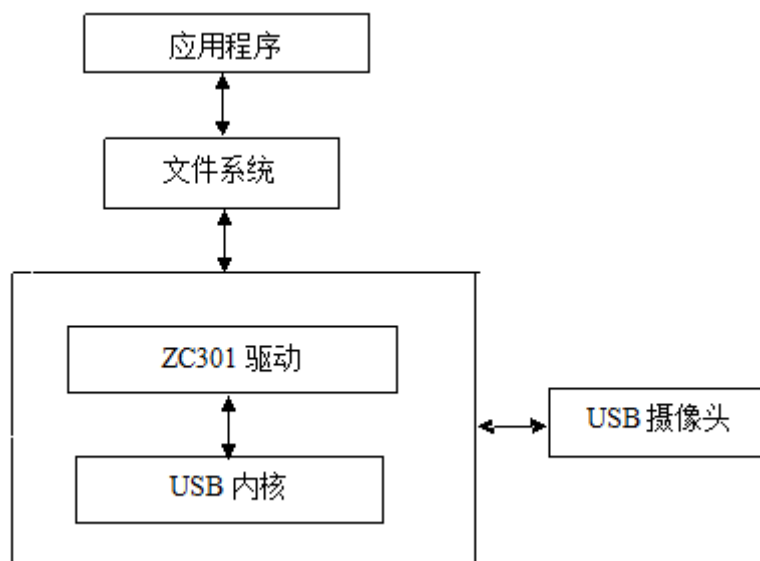


图 3.1 USB 视频采集系统功能框图

Fig 3.1 picture of USB video capture system

我们的 USB 摄像头驱动 `spca5xx_driver` 内容如下所示：

```
static struct usb_driver spca5xx_driver = {
    .owner = THIS_MODULE,
    .name = "gspca",
    .id_table = device_table,
    .probe = spca5xx_probe, .disconnect = spca5xx_disconnect
};
```

本系统所使用的摄像头属于外置设备，在系统启动之前首先需要加载摄像头相应的驱动模块。首先将相应的模块放入相应目录中，同时把加载命令输入进 `/etc/init.d/rcS` 启动脚本中，之后将视频设备文件与 `/dev/video0` 相连接。如此启动系统便会在启动时自动加载摄像头驱动并初始化。

USB 摄像头具体安装移植过程如下：

获取安装驱动模块。在 <http://mxhaard.free.fr/> 下载最新 `spca5xx` 系列驱动以及 `spca5xx-LE`，`spca5xx-LE` 是该网站为专嵌入式编程开发的摄像头补丁。把它们放到 `/kernel/driver/usb` 目录下进行解压，并打入嵌入式补丁。执行 `make menuconfig` 命令进入配置单，可以看到 `SPCA5XX` 选项将其选中，该部分属于内核静态模块，只要通过编译就可以实现内核内置驱动了。用此内核启动系统，便会识别 `zc301` 摄像头，并在 `/dev/` 生成摄像头文件以便于控制。这样摄像头驱动便成功安装完毕了。

3.2 基于 V4L 的视频采集模块的设计与实现

在 LINUX 环境下进行视频采集,系统通常是使用 Video4Linux 的 API 函数来完成视频数据的采集。Video4Linux 简称 V4L,提供了视频设备相关的 API 函数,并为设备驱动与应用程序之间提供了标准接口。开发人员能够通过这个接口所提供的 API 函数来对各种采集设备开发驱动接口,并实现不同硬件对音视频数据的采集工作^[1]。可以看出,Video4Linux 分为两层结构,下层为支持硬件的驱动接口;上层为系统调用要用的 API^[2],本系统主要将使用上层 API 来设计应用程序。下面将详细介绍其设计与实现过程。

3.2.1 V4L 应用常用数据结构及编程接口

Video4Linux 中与视频相关的常用数据结构如表 3.1 所示:

表 3.1 相关结构体

Table 3.1 Structures

名称	作用
video_capability	包含 name、audios、type 等成员变量,主要的作用是存储设备的基本情况以及各项参数信息。
Video_picture	包含的成员变量有 brightness、contrast、whiteness、palette 等,主要作用为保存设备采集到图像的属性。
Video_mmap	用于实现内存映射机制
Video_mbuf	获得视频采集设备存储图像所占内存大小
Video_channel	保存各信号源相关信息
Video_buffer	该结构体的作用是最底层的设备用来对内存中的缓冲区信息的描述

3.2.2 视频采集的实现

V4L 视频编程的基本流程为:首先打开视频设备,之后获取硬件信息,并对设备做以相应的设置。此时便能够读取视频数据,之后根据需要对视频进行压缩、显示等操作。最后关闭设备即可。

具有流程如图 3.2 所示。具体实现过程如下:

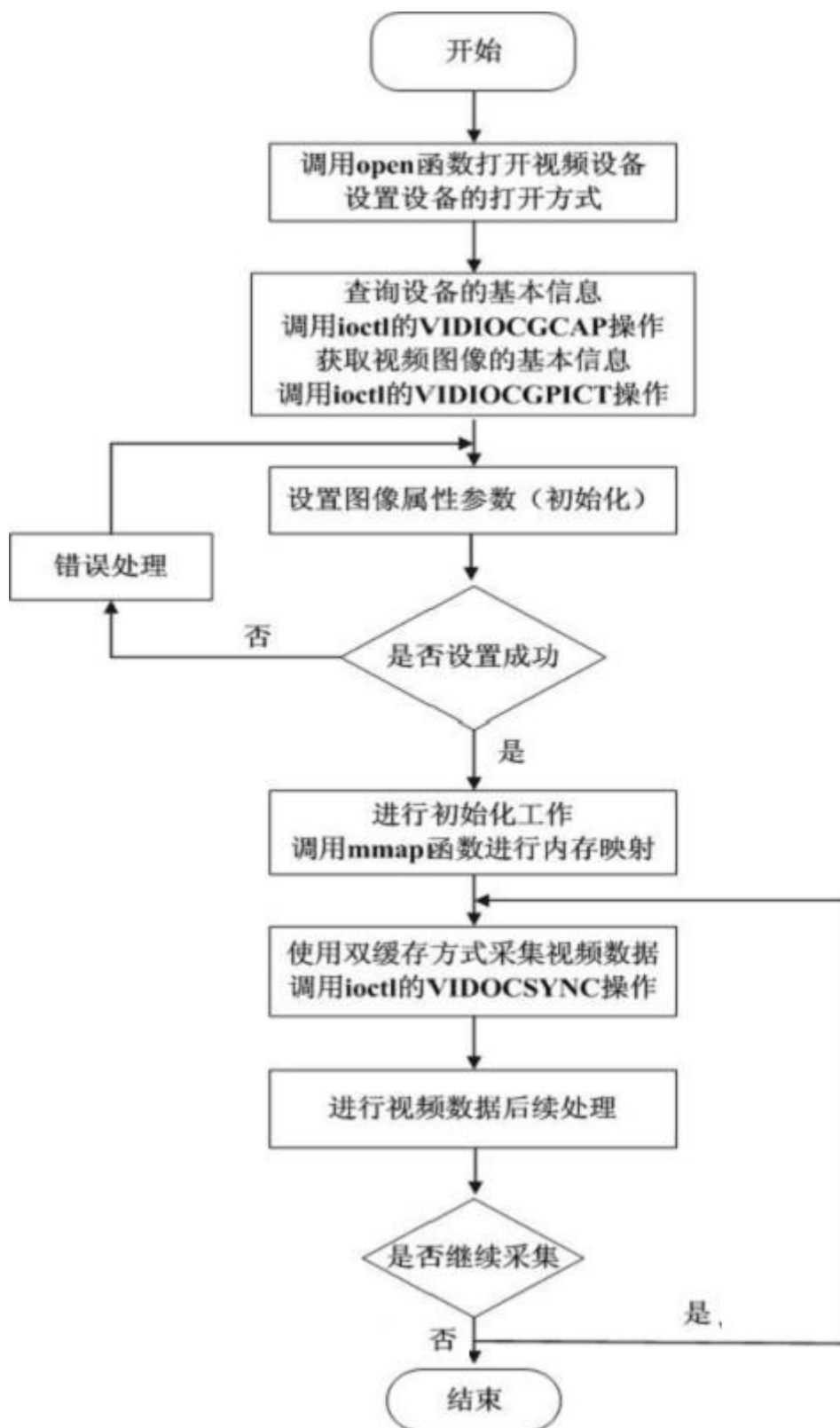


图 3.2 无线视频监控系统视频采集流程图

Fig 3.2 the precess wireless video surveillance system capture

1. 打开视频设备。调用 `int v4l_open()` 函数打开 USB 摄像头，在使用 `open`

函数时, 还要声明一个 `video_device` 类型的变量。执行 `Fd=open("/dev/video0", O_RDWR)` 便可得到设备描述符。其中 `video0` 为摄像头的默认名称。如果成功打开设备便会返回相应属性信息, 反之, 如果打开失败则返回-1^[13]。

2. 获得设备信息。要获取硬件设备与采集图像的信息, 如设备名称、型号、信号类型、分辨、颜色等, 需要调用 `int v4l_get_capability` 与 `int v4l_get_picture` 两个函数, 在这个两个函数中都用到了 `ioctl()` 函数。`ioctl` 函数是 LINUX 环境下对 I/O 通道进行管理操作的函数, 主要功能为对一些设备的特性进行控制。它为系统调用提供了一种执行设备特定命令的方法其, 函数原型为:

`int ioctl(int fd, int cmd, ...)` 其中 `fd` 与之前的函数一样为描述符, 开发人员可以通过 `cmd` 对设备进行控制, 还可以补充一些参数, 是否需要与 `cmd` 有关^[14]。由此可见, `ioctl` 系统调用的功能是使用文件描述符来对各种文件, 特别是字符设备文件进行控制, 实现指定的 I/O 操作。这里, 我们执行 `ioctl(vd->fd, VIDIOCGPICT, &(vd->picture))` 读取 `vd->capability` 来获取摄像头信息, 该函数返回成功后, 用户程序空间就从内核空间就拷贝到了这些信息。执行 `ioctl(vd->fd, VIDIOCGCAP, &(vd->capability))` 读取 `vd->picture` 结构体, 便获得了图片的亮度、对比度、调色板等信息。

3. 更改设备的相关设置。调用 `ioctl` 命令可以获取 `VIDIOCGWIN`、`VIDIOCGCAP`、`VIDIOCGMBUF`、`VIDIOCGPICT` 等常量, 成功获取摄像头基本信息之后, 使用 `ioctl` 命令 `VIDIOCSWIN`、`VIDIOCSPICT` 就可以更改摄像头基本信息, 具体命令如下:

```
ioctl(cam_fd, VIDIOCGPICT, &cam_cap); /*获取摄像头的基本信息, 如最大最小分辨率*/
```

```
ioctl(cam_记, VIDIOCGWIN, &win); /*设置摄像头图像大小信息, 如当前分辨率*/
```

```
ioctl(cam_fd, VIDIOCGMBUF, &cam_mbuf); /*获取摄像头存储缓冲区的帧信息*/
```

```
ioctl(cam_fd, VIDIOCSPICT, &cam_pic); /*设置摄像头缓冲中 video_picture 信息, 主要获取图像采集格式*/
```

4. 图像采集。V4L 提供的图像获取的方式有两种: 一直读取与内存映射。

直接读取就是通过 `read()` 函数对内核缓冲区直接获取图像。由于硬盘缓冲区的访问速度较内核缓冲区读取速度慢, 所以读取内核缓冲区这种方法速度较慢。

内存映射是利用 `mmap()` 函数把设备文件映射到内存中, 使用的时候只需要使用指针就可以进行访问, 不必再调用读取、写入等函数, 不同进程还能够共

享内存,节省了使用文件操作函数的时间开销,加快了 I/O 操作。不同进程共享内存是通过将同一块物理内存映射到不同进程的地址空间。不同的进程之间能够知道其他进程对内存的操作如何。本文采用内存映射的方法来获得图像数据^[15]。

在进行图像采集时,以下函数同样有着重要的作用,配合 `mmap` 函数,共同实现图像的采集。

`extern int v4l_mmap_init(v4l_device *);` 该函数的作用是把视频数据映射到内存中。

`extern int v4l_grab_init(v4l_device *, int, int);` 初始化函数。

`extern int v4l_grab_frame(v4l_device *, int);` 该函数的作用为采集图像数据。本系统设计了在处理一帧图像时,同时能够采集下一帧的图像的方法。

`extern int v4l_grab_sync(v4l_device *);` 该函数的作用是同步截取图像。当函数返回 0 时,表示图像帧截成功^[14]。

图像截取时,我们所调用的 `vd.map` 指针所指向位置便是采集到的图像所在处,通过设置参数可以实现第一帧与第二帧的转换。另外本系统还使用了双缓存的方式,在内存中另外在开辟一个存储空间。在系统对采集到的数据进行各种操作时,另一个存储空间可以用来保存采集到的图像,这样便加快的系统运行的效率^[14]。

5. 视频数据的后续处理。在完成了视频数据的捕获之后,本系统的下一个任务是将视频数据进行压缩编码,以及传输等操作。这些技术将在后续章节详细讲解。

6. 关闭视频设备。关闭之前,需释放使用过的共享内存空间,调用 `munmap()` 函数,之后使用 `close()` 函数关闭设备。

3.3 视频编解码模块的设计

无线视频监控系统本身对实时性以及流畅性具有较高的要求,另外监控视频的清晰度也是要考虑的一大问题。然而,对于无线视频监控系统自身的特点来说,其流畅度与清晰度是矛盾的。由于嵌入式处理器的性能在一定程度上存在局限性,所以二者无法同时满足很高的要求。另外无线网络的带宽也无法承受摄像头捕获到的高清图像所占的高带宽。**USB** 摄像头在刚刚采集到图像之后,视频数据由模拟信号转为数字信号,此时的数字信号在未经任何处理的时候所占有的带宽会达到 **2MB/S**,每分钟会产生 **120MB** 的信息量。对于如此大的信息量一台个人计算机都难以处理,对于本系统所选用的嵌入式处理器更是无法承受。因此,

无论是存储还是传输，都必须将采集到的视频数据进行编码压缩。

3.3.1 MPEG-4 视频压缩

对于一个视频的播放过程，我们可以将其看做为一幅一幅的图片连续播放而成。实际上，相邻的两帧图片之间存在某些联系与数据的冗余，视频压缩就是利用这些特点来实现的。

传统的压缩编码是以香农信息论为基础，用统计概率模型来描述信源，编码实体是像素或像素块。这种编码方式被称为第一代视频编码技术，其主要目的是消除视频数据相关冗余，JPEG、MPEG-1、MPEG-2、H.261 以及 H.263 等国际编码标准就是第一代编码技术的代表。对多媒体产业产生了巨大的影响。这些标准主要采用的技术有嫡编码、变换编码以及运动补偿等。第一代视频编码技术并没有考虑到使用者的自身特点，只是为了去除数据的冗余，属于低层次的编码技术。第二代视频编码技术，由原来基于块的图像编码转为去除视频内容的冗余。考虑了使用者自身的特点，充分考虑了人眼的视觉特性影响，这是目前视频编码最为活跃的一个领域。

MPEG-4 标准是一种基于对象的压缩编码方法，它能够将视频图像数据进行高效压缩，并且支持对多媒体信息的内容访问，同时还具有交互功能。另外，MPEG-4 还自带一些检测误码以及恢复误码的工具，使其能够在无线网络或 Internet 这种容易产生误码的环境中，提高抗信道误码^[16]。

MPEG-4 的基本思想是将视频中的每一帧图像都视为活动背景和视频对象的组合，首先将视频分割成在时间和空间上有关联的视频以及音频对象，然后对每个对象进行单独编码，之后传输到接收端，再对每个对象进行解码，最终汇成总的视频流。这种方式方便对不同的对象采用不同的编码方式，有利于不同类型数据的结合。在对每一帧图像的不同对象编码方式由：形状编码、运动信息编码以及纹理编码。

3.3.2 基于 FFMPEG 的视频解码

FFmpeg 是一款支持多平台的开源软件，它能够记录、转换数字音视频并将其流化。根据用户所选用的组件不同，它所采用的许可证也不同，分为 LGPL 和 GPL 两种。FFMPEG 进行编解码时，主要是依靠它所提供的 libavcodec 音视频编码库，该编码库能够保证编解码的质量和可移植性。此外，FFMPEG 还包含 libavformat 库，该库能够生成并解析各种音视频封装格式，包括读取音频、视频

帧和读取生成解码上下文结构所需的解码信息等功能。具体编程过程中会使用到这两个库所提供的各种编程接口。此外, libavutil 库中包含一些公共的工具函数, 简要介绍几个库中所包含的函数:

1. void avcodec_init(void); 用于初始化 libavcodec, 一般最先调用该函数, 引入头文件: #include "libavcodec/avcodec.h"。该函数必须在调用 libavcodec 里的其它函数前调用, 一般在程序启动或模块初始化时调用。该函数是非线程安全的。

2. void avcodec_register_all(void), 该函数用来初始化 libavformat 并注册多种音视频格式的编解码器以及各种文件的编解复用器, 系统会为后来运行的文件自动匹配编解码器。

3. avformat_alloc_context, 该函数负责申请一个 AVFormatContext 结构的内存, 并进行简单初始化, 设置成默认值, 如果失败则返回 NULL。

4. avcodec_find_decoder(enum CodecID id), 该函数通过 code ID 查找一个已经注册的音视频解码器, 查找解码器之前, 必须先调用 av_register_all 注册所有支持的解码器, 查找成功返回解码器指针, 否则返回 NULL。音视频解码器保存在一个链表中, 查找过程中, 函数从头到尾遍历链表, 通过比较解码器的 ID 来查找。

5. avcodec_open(), 该函数用于对 AVCodec、图像以及高度的初始化, 将其化为 AVCodecContext 数据结构。

6. parse_options(argc, argv, options), 函数的作用为初始化参数, 函数中 argc 代表参数的数量, argv 负责保存输入参数, options 为一个存储着 FFMPEG 所支持的参数类型的数组。

7. write_video_frame(AVFormatContext *oc, AVStream *st), 这个函数中做了真正的编解码工作, 用到的数据结构有 AVCodecContext *c, SwsContext *img_convert_ctx。其中 SwsContext 是用来变换图像格式的。

8. av_free(), 该函数可以将之前使用过的内存空间释放掉, 并执行一些其他的结束工作。

以上为 FFMPEG 在视频编解码是所涉及的几个主要函数, 其中包括一些数据结构, 他们用来存储图像信息以及某些结构体, 具体就不在此展开了。

FFMPEG 是在 LINUX 平台下开发的, 所以移植到嵌入式 LINUX 系统中十分方便, 只需通过修改相关编译参数即可, 首先下载 FFMPEG 源码包进行解压, 之后进入解压目录对 Makefile 文件进行修改, 然后对 configure 配置文件, 编辑 configure 文件内容。在配置生成可执行文件的选项时, 在 \$(CC) 后面加上参数 "static", 最后运行 make 和 make install, 完成编译和安装。此时, 将交叉编译成功的 FFMPEG 程序复制到目标开发板中便完成了 FFMPEG 的移植工作。

本系统分为采集模块与监控模块两个部分, 在两个部分 FFMPEG 的工作过

程区别就在编码与解码这一步骤，他们是一个相逆的过程。所以只要知道了编码便能够知道解码的工作方式，下面主要从编码方面介绍本系统的编码过程。

无线视频监控系统的视频编码过程如图 3.3 所示。分为以下几个步骤：

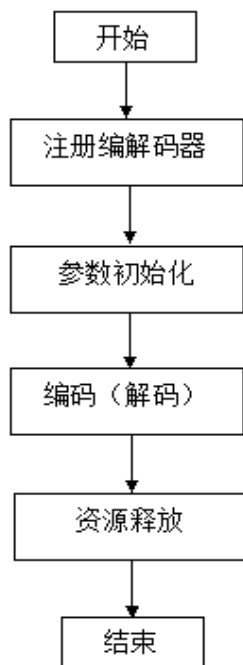


图 3.3 无线视频监控系统视频编码（解码）流程图

Figure 3.3 Video encode(decode) chat of wireless video monitoring system

1. 首先调用 `void avcodec_init(void)` 函数，用于初始化 libavcodec，在使用 avcodec 库前必须调用该函数，用于初始化一些静态数据。

2. 调用 `av_register_all()` 函数来初始化 libavformat 并注册多种的注册多种音视频格式的编解码器，之后系统就会自动为视频文件寻找匹配的编解码器。

参数初始化。调用 `parse_options()` 函数解析参数的作用的并检查所有输入参数；调用 `avcodec_alloc_context()` 与 `avcodec_alloc_frame()` 函数，为 AVCodecContext 与 AVFrame 数据结构设置初值。调用 `avcode_find_encoder` 函数选择编码标准；

3. 编码或解码过程，调用 `write_video_frame()` 函数并配合 `avcodec_encode_video()` 来完成视频的编码，或配合 `avcodec_decode_video` 来完成视频的解码。在此过程中，系统对视频进行了预测、变换、量化、熵编码等操作。完成编码之后系统就会把原始的视频数据转换成 MPEG-4 格式的压缩数据，解码过程正好相反。

4. 释放资源。在系统使用的过程中，资源会占用越来越多的内存，使用 `free`

() 函数来释放掉之前申请的内存工作空间以及关闭文件等操作

这样就完成了视频数据的编码(解码)的过程,系统可以视频数据进行下一步的处理。

3.4 Linux 下的多线程技术的研究

无线视频监控系统需要同时进行采集、压缩、传输、播放等功能。而实现这些功能所消耗的时间与资源都是不一样的,如果他们都在一个单线程中工作,必然会对整个系统产生影响,降低系统效率。为了提高系统效率,防止线程之间的冲突,本系统需要引进多线程的方法来使不同的程序“同时”完成不同的任务。线程也称为轻量级进程,通常为一个程序过程中的最小单位。线程一般由指令指针、线程 ID、堆栈与寄存器集合组成。一个进程可以创建多个线程,所创建的线程必须有父进程。进程内的线程能够使用该父进程的所有资源,而线程其本身一般是不拥有资源的,只是拥有一些运行时所必须的数据结构。在一个进程或程序中都会有最少一个线程,如果只有一个线程则就是该进程或程序自身。同属于一个进程中的线程之间是可以相互创建于销毁的。由于线程之间共享资源的特性,当他们使用同一个资源时或出现争夺现象,过程中便会出现就绪、运行、堵塞三种情况^[4]。线程分为用户级线程与内核级线程两种类型,内核态线程需内核参与,由内核来完成调度并提供系统调用。用户级线程在进行切换时不需要借助系统调用,也不需要内核的支持,具有高效性。在一个进程创建多个线程的情况下,使用用户级线程优势较大,本系统线程为用户态线程^[17]。

在本系统中,我们使用的是嵌入式处理器,在性能方面有一定的限制,使用线程比进程就会有较多的优势。在 Linux 环境中,创建进程时必须分配独立的地址空间与资源,由于嵌入式系统资源有限,并且同一个进程中的线程可以共同访问该进程的资源,因此多线程的方式能够节省大量的系统资源。在运行效率上线程也比进程快的多,一般来说,线程的创建或切换时间与资源开销大约是进程的 30 倍左右。另外,线程之间的通信机制也非常方便。

3.4.1 多线程编程接口

Linux 操作系统为多线程编程提供了一些函数库与编程接口,最为常用的多线程库为 pthread 库, pthread.h 头文件提供了接口声明。以下为一些主要的库中所包含的多线程函数。

1. `int pthread_create(pthread_t*restrict tidp,const pthread_attr_t *restrict_attr, void* (*start_rtn) (void*), void *restrict arg)`. 该函数的作用是创建线程。线程创建成功时新线程的 ID 会存放到 `tidp` 指针指向的内存空间；第二个参数 `attr` 可以设置线程的属性；`start_rtn` 函数所指向的程序便是新线程将要执行的程序^[18]。

2. `void pthread_exit(void *status)`；当线程希望被终止时便可调用该函数，该函数能够终止调用它的线程。

3. `int pthread_join(pthread_t tid,void **status)`,该函数作用为等待一个指定线程结束，指定的线程为函数的第一个参数，也就是线程创建时所分配的 ID 号。被等待线程的返回值一般存放在 `Status` 所指向的位置。

4. `int pthread_mutex_init (pthread_mutex_t *mutex, const pthread_mutexattr_t *restrict attr)`；该函数的作用是多线程编程中互斥锁的初始化。函数是以动态方式创建互斥锁的，参数 `attr` 指定了新建互斥锁的属性。要在创建互斥锁的时候便设置其属性，如果参数 `attr` 为空，则使用默认的互斥锁属性：快速互斥锁。函数调用成功后，完成互斥锁的初始化工作^[19]。

5. `int pthread_mutex_lock(pthread_mutex_t *mptr)`；

`int pthread_mutex_unlock(pthread_mutex_t *mptr)`；这两个函数的相对的，用于加锁和解锁操作。以便线程之间的资源共享，防止出现两个线程同时访问一个共享数据。

6. `int pthread_cond_wait` 函数的作用为在某个条件为真之前，阻塞线程。期间线程可以被 `pthread_cond_signal()`和函数 `pthread_cond_broadcast()`唤醒。一般与互斥锁函数一起使用来实现线程之间的同步。

3.4.2 多线程模块的工作流程

当一个进程启动时，他会自动创建一个线程并将该线程初始化，系统会调用 `pthread_initialize()` 函数完成初始化工作。之后便可以通过 `pthread_handle_create()` 创建线程函数来创建新的线程，有多个线程被创建时，如何管理调用这些线程，便有了多线程思想^[20]。

多线程模块的工作流程如图 3.4 所示，其具体流程如下：

首先当进程启动时，它会自动创建一个主线程，之后调用 `pthread_initialize()` 函数来初始化管理线程，此时线程机制便会开始运行，初始化工作包括定义线程变量、初始化互斥变量、初始化条件变量等^[20]。

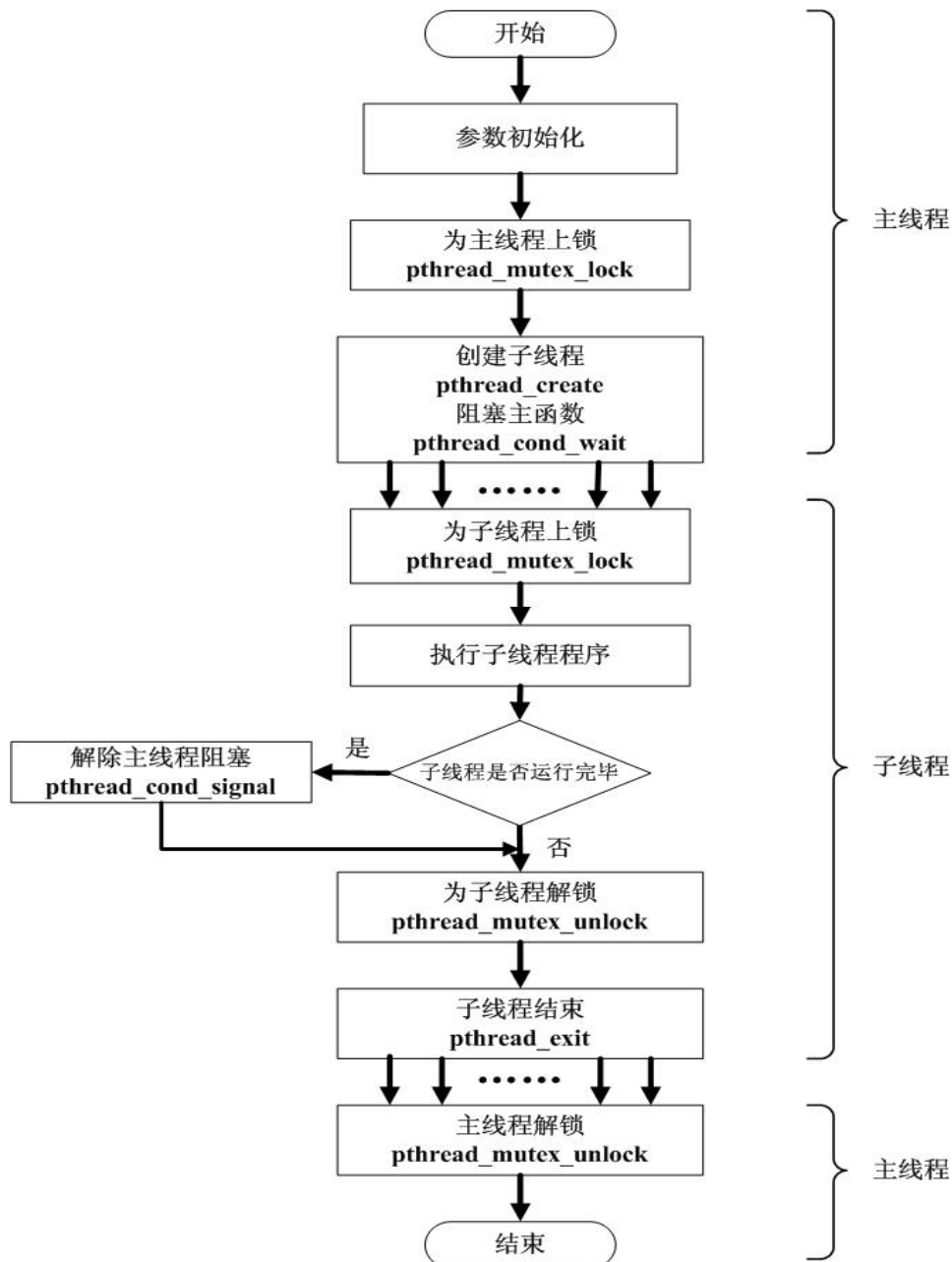


图 3.4 多线程模块工作流程图

Figure 3.4 Flow chat of multi-thread module

初始化工作完成之后，调用 `pthread_mutex_lock()` 函数为主线程加上互斥锁，以防止多个线程同时请求 `pthread_cond_wait` 的竞争条件，实现线程之间的同步。当需要创建新的线程时，管理线程会通过 `pthread_create()` 函数来创建子线程^[20]。为防止子线程运行过程中主线程就以及运行完毕的情况发生，子线程可以使用 `pthread_cond_wait()` 函数。不同线程对共享资源访问时需要用到互斥量这一概念，它是信号量的一个简化版。具有加锁和解锁两种状态，当一个线程要访问共享资源时，需要看当前互斥量的状态如何。如互斥量处于解锁状态，则线程便

可以访问该资源，同时为互斥量加锁。如互斥量处于加锁状态，线程则会被阻塞。当有多个线程同时遇到同一加锁互斥量时，哪一个线程获得访问权由当前调度机制决定。第一个变为可运行状态的线程再次对互斥量加锁，如此循环。由于互斥锁只有加锁与解锁两种状态，所以线程的同步工作需要借助条件变量来共同实现^[20]。

条件变量使得线程在对同一共享资源进行争夺时出现等待状态。利用全局变量，线程之间能够传递条件。当一个线程在争夺资源时，线程不会像之前一样无条件争夺，而是会等待条件变为真而处于挂起状态。另一个线程可以指定将条件变量变为某一个值来让某一特定的线程访问。同时，互斥量还能够对条件变量起到保护作用。其中会借助 `pthread_cond_wait()` 和 `pthread_cond_signal()` 两个函数来提醒对互斥量的再加锁以及唤醒特定条件的线程^[21]。

最后，通过 `pthread_mutex_unlock()` 和 `pthread_exit()` 等函数来解除线程占用的互斥锁并结束该线程。

3.4.3 多线程在 RTP 传输中的应用

多路无线视频监控系统有多个采集端，在监控端方面要接收从不同的采集端发来的数据。此时，要为每一个接收端口设置一个线程，如只采用单线程则在接收一个采集端的数据时，出现其他的数据无法接入的情况。每一个线程都有其各自的端口号，这些端口共享同一个接收端的 IP 地址，除端口号外的流程都是一样的，如图 3.5 所示。

以一个线程为例，首先用下面的函数创建一个线程：线程的创建方法上一节已经介绍过，使用 `int pthread_create()` 函数，创建一个新的线程，得到新线程的 ID 并设置线程的属性参数。

新线程建立好之后，要为其初始化。之后通过下面几个函数为线程设置时间戳、端口号、接收自定义的数据包等，并为其建立 RTP 会话：

```
RTPUDPV4TransmissionParams transparams1;
RTPSessionParams sessparams1;
sessparams1.SetOwnTimestampUnit(1.0/30.0); //设置时间戳
sessparams1.SetAcceptOwnPackets(true);      //设置接收自己定义数据包
transparams1.SetPortbase(portbase1);        //设置接收端口号
status1 = sess.Create(sessparams1,&transparams1);
```

这样，线程 1 的 RTP 会话初始化工作便完成了，其他的线程的初始化结构与其类似。

建立 RTP 会话之后，便准备进行数据的接收。不同的线程会通过端口号不同的特点来接收指定数据。RTP 的具体实现过程将在下一章介绍。

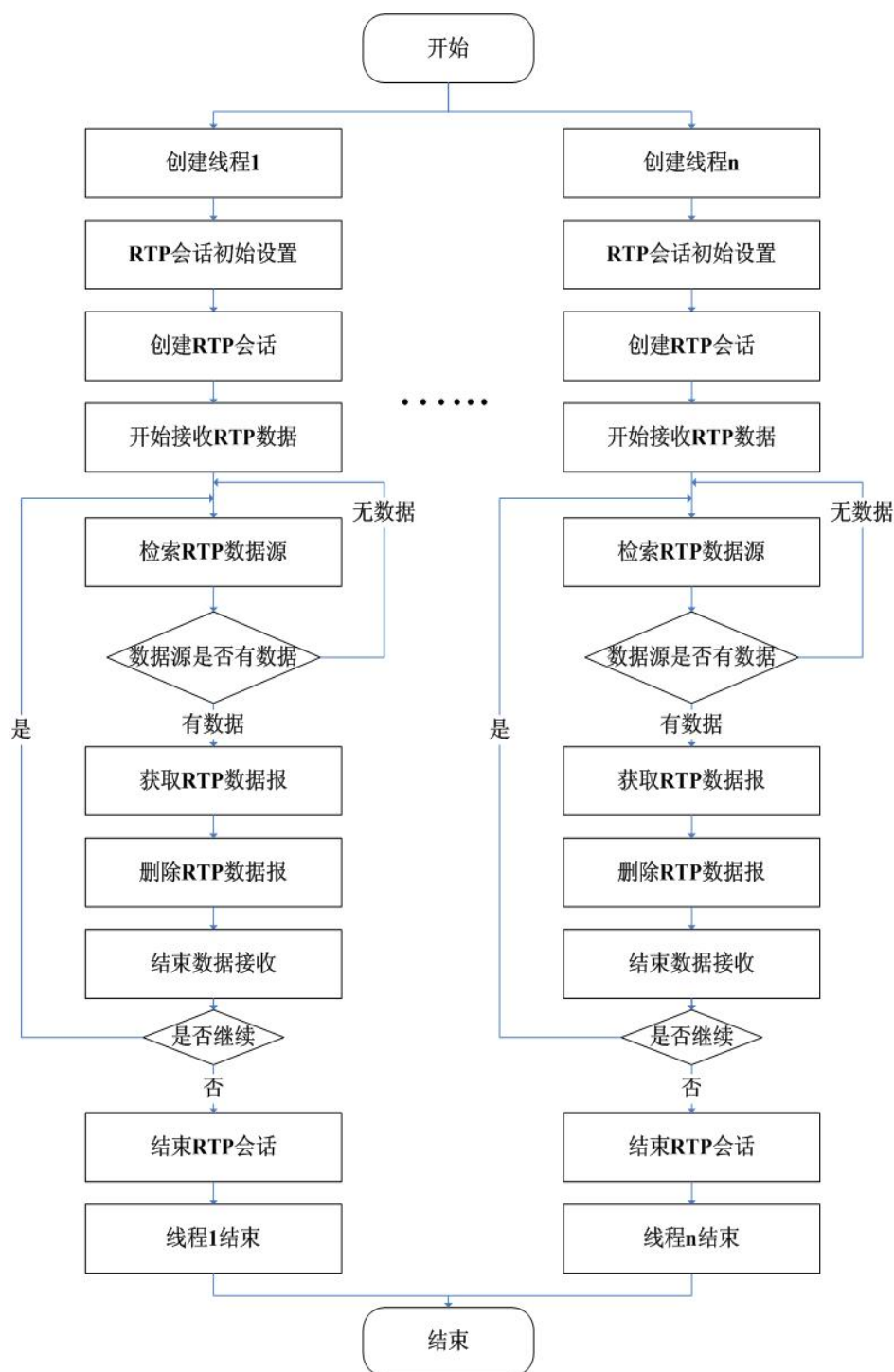


图 3.5 接收端数据接收流程图

Fig.3.5 The picture of process of data reception in acquisition side

第4章 基于 RTP/RTCP 的无线传输设计

通过上一章,我们已经完成了采集端摄像头的加载,并对采集到的数据进行了压缩编码。接下来要解决的任务就是把视频数据从采集端传输到监控端。本章着重讲述如何实现本系统的无线网络传输,并与多线程结合完成整个无线视频监控系统的的设计。

4.1 RTP/RTCP 协议分析

根据视频网络传输的特点吗,与文本数据传输相比,它对于传输的可靠性与安全性并不是相当高。但是对于传输的实时性有较高的要求。在传输文本性非实时传输来说,当网络堵塞时只是接受数据较慢但不影响使用。而视频传输如出现网络堵塞则会导致难以忍受的视频服务质量。因此, TCP 所带有的验证重发机制并不适用于本系统。RTP 实时传输协议的出现,很好的解决了这些问题。该协议是 IETF 工作组提出制订的^[16]。

RTP 主要用于针对多媒体应用的中的实时传输,它提供序列编号、时间戳、负载类型标识和传输控制等功能。它能够工作在一对一和一对多两种环境中,并切能够同步流媒体^[22]。RTP 属于传输层协议,而且没有定义网络层以下的功能,因此必须与其他网络协议一起使用来完成数据的传输^[23]。通常 RTP 使用 UDP 来完成数据的传输,而 UDP 本身不提供任何 Qos,因此, RTP 需要配合 Qos 服务来满足实时传输的时延和丢包要求。但如果有需要也能够在 TCP 等其他协议中工作。RTP 协议只有封装数据以及实时传输的功能,它本身并提供数据传输的可靠性的保障,而且不具有堵塞和流量的控制功能。因此通常会结合 RTCP (实时传输控制协议)来完成这些功能^[24]。在 RTP 开始会话时,它会使用两个端口:一个给 RTP,另一个给 RTCP。在会话期间,通常 RTCP 会向会话中的所有成员周期性地发送控制信息,包括数据包的数量、数据包丢失的数量、网络状况等反馈信息。通过这些信息能够有效的对传输质量进行控制。由此, RTP 与 RTCP 配合使用来完成本文的网络传输工作。

4.1.1 RTP 实时传输协议

由上一节可知, RTP 通常利用 UDP 来完成数据的传递,与 TCP 协议相同, RTP 数据报由头部和负载两个部分构成,其报文头为 12 个字节^[25]。RTP 数据报

的头部格式^[36]如图 4.1 所示。

V=2	P	X	CC	M	PT	序列号 (sq)
时间戳						
同步源标识 (SSRC)						
提供标识源 (CSRC)						
.....						

图 4.1 RTP 报文头格式

Fig4.1 Header Formation of RTP

以下为 RTP 报文中各个代号表达的意思：

Version (V)：版本号，2bit。识别 RTP 版本。

Padding (P)：填充位，1bit。当 P 位置位时，说明包的末端有附加字节，部分不属于有效载荷。

Extension (X)：扩展位，1bit。当需要在后面再接一个扩展时，将此位置 1 即可。

CSRC Count (CC)：CSRC 计数器，4bit。该标示用来计算固定头后 CSRC 标识符的数量并表示其来源。

Marker (M)：标志位，1bit。标志的解释由具体协议 Profile 文档规定。

Payload type (PT)：有效负载类型，7bit。表示 RTP 负载的格式。

Sequence number (SN)：序列号，16bit。用于标记包的编号，第一次产生为一个随机值，此后每发送一个包再次随机值上加 1。

Time Stamp：时间戳，32bit。标记了数据报中第一个字节的采样时间，该数据会根据时钟频率的大小而随着时间不断增加。在数据同步和消除抖动起着关键作用。

SSRC：同步源标识，32bit，该标识在会话开始时会随机产生并禁止重复，其作用为防止在一个会话中有相同的同步源标识。

从 RTP 数据报中能够看出，它包含了类型、序列号、时间戳等信息。在传输过程中，RTP 的报头与负载共同封装成 RTP 包，再与 UDP 层添加 UDP 报头封装成 UDP 包，最后与 IP 层 IP 层添加 IP 报头封装成为 IP 包，接收过程与发送过程互为逆过程。

4.1.2 RTCP 控制协议

RTCP 实时传输控制协议与 RTP 实时传输协议配合使用, 在 RTP 实时传输时, 由 RTCP 来完成流量控制、媒体同步、服务质量监视等功能^[38]。RTCP 在 UDP 协议层之上, 属于传输层协议, 因此也要像 RTP 一样进行数据封装成为 UDP 包。类似于 RTP 数据包, 前一部分是固定的。它通信的方式与数据报分配传递机制相同, 采用周期性通信。RTCP 报文具有五种类型:

Sender Report(SR): 发送端报告, 发送端所发送数据包的接收反馈信息、RTP 时间戳、包数统计等信息

Receiver Report (RR): 接收端报告, 包括接收端接收报文的丢包率、最大序列号、时延信息等。

Source Description Items (SDES): 源描述报文, 在对话中各个源的信息由此报文给出, 如用户名、邮件地址、电话号码等。

Indicates End of Participation (BYE): 结束报文。当此次会话需要挂断时, 发出此报文。

Application Specific Functions (APP): 应用程序报文。解决 RTCP 的扩展问题。

由此可见, RTCP 对于 RTP 的传输有着重要的作用, 它带有对于监控质量的重要信息, 能够对网络传输进行有效的调整。在会话中, 由于采用的是多播方式, 因此会话中所有成员都能够看到控制信息。

4.2 无线视频监控系統传输模块的实现

4.2.1 JRTPLib 库的安装

在 Linux 系统中, 我们可以使用一个开源的 RTP 库来完成实时流媒体的编程, 比如 LIBRTP、JRTPLIB 等。本文选用的是 JRTPLIB 库来开发实时传输的, 它使用 SOCKET 机制来实现网络传输^[26]。在很多场合下是一个非常不错的选择。

本文使用的是 JRTPLIB-3.8.2 版本, 首先在网上下下载源代码包 `jrtp-3.8.2.tar.bz2`, 将其放在系统根目录下。对其解压使用命令 `tar -xvzf jrtp-3.8.2 /jrtp-3.8.2`。之后进入 `/jrtp-3.8.2` 目录中, 运行如下指令可对该库进行配置安装:

```
[root@linux src/jrtp-3.8.2] # ./configure --host = arm-linux
```

```
[root@linux src/jrtplib-3.8.2] # make
```

```
[root@linux src/jrtplib-3.8.2] # make install
```

以上命令执行成功之后，Linux 上的 JRTPLIB 便安装完成了^[23]。

4.2.2 无线视频系统发送端的设计

本文所开发的系统为多路监控，有多个采集端，每个采集端中的 RTP 发送流程除了端口号不同之外完全相同。所以我们只分析一个发送端的发送流程，如图 4.2 所示。

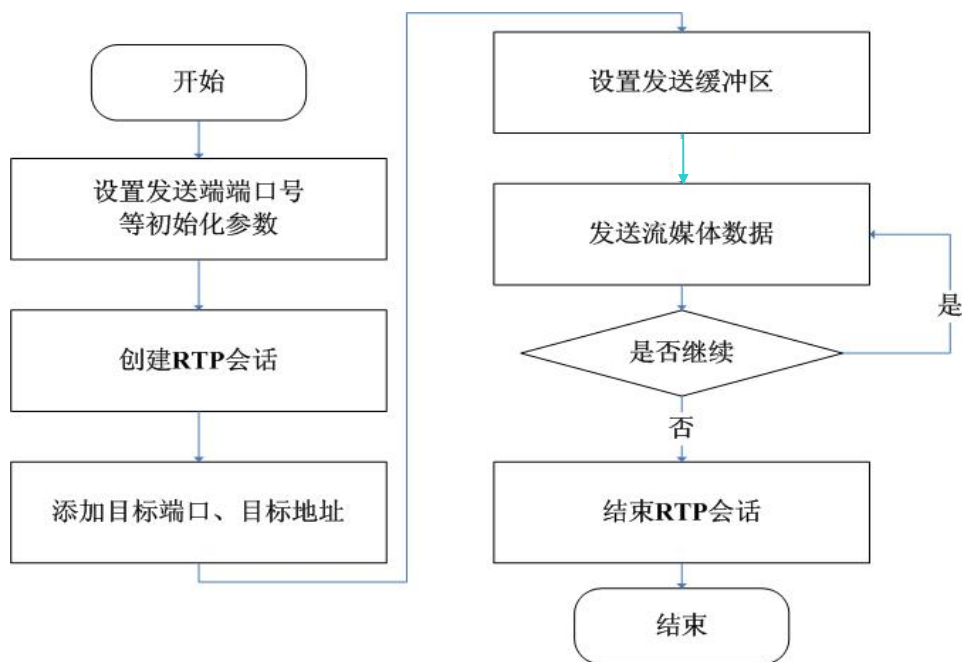


图 4.2 RTP 发送端流程

Fig.4.2 The process of RTP sending terminal

在建立 RTP 会话时，需要先对会话中需要的端口、参数等进行初始化工作。包括发送端以及接收端的端口号、要传输包的最大值以及接收端的 IP 地址等信息。初始化工作完成之后，便可以建立 RTP 会话了；同时设置好需要存放发送数据的发送缓冲区，并将数据放入该区中；最后由 RTP 协议将数据打包并发送。整个过程被设置为一个死循环，在需要时能够强行退出。以下为具体实现过程：

1. 建立 RTP 会话

在本系统中为了表示 RTP 会话，首先需要建立一个 RTPSession 类的对象。之后进行初始化工作并使用 Create()函数来建立 RTP 会话。之后便能够实现流媒

体的传输了。Create 函数为：

```
Create(sessparams, &transparams);
```

2. 初始化参数设置

首先定义 Create(sessparams, &transparams)中的两个参数

```
RTPUDIPv4TransmissionParams transparams;
```

```
RTPSessionParams sessparams;
```

第一参中 transparams.SetPortbase(portbase)函数用来设置本地通讯端口。

第二个参数 RTPSessionParams 的 sessparams 对象成员函数：

```
sessparams.SetOwnTimestampUnit(1.0/1000.0);
```

用于设置时间戳，1.0/1000.0 表示的是 1 秒钟采样 1000 次^[27]。

```
sessparams.SetMaximumPacketSize(MAX);
```

定义了发送包的数据最大值。

```
sessparams.SetAcceptOwnPackets(true);
```

用于自定义需要接收的包。

3. 设置目标端口与目标地址

建立好 RTP 会话之后，需要设置数据发送的目标地址。通常调用 AddDestination（）函数来完成，本文设置的地址如下：

```
destport=8000;
```

```
destip=inet_addr("192.168.1.100");
```

```
destip = ntohl(destip);
```

```
RTPIPv4Address addr(destip,destport);
```

```
status = sess.AddDestination(addr);
```

4. 设置发送缓冲区

在发送数据之前先设置一个发送缓冲区，使用 pBuf 指针变量指向发送缓冲区。调用语句 FILE *fp=fopen("1.jpg","r") 指向“1.jpg”流媒体数据。使用 fseek(fp,0,SEEK_END)将指针设置到流媒体数据的末尾。之后用 char* pBuf=new char[lLength+1]和 pBuf[lLength] 来申请缓冲区内存资源，再使用 fseek(fp, 0, SEEK_SET)将指针指向数据的第一帧位置，最后使用函数 fread（）从 fp 指针指向的数据中读取数据并放入缓冲区中，准备发送。

5. 数据发送

最后可以使用 RTPsession 类的成员函数 SendPacket()发送数据。它是一个重载函数有四种用法，最常见的为 sess.SendPacket((void *)pBuf,lLength,0,false,10); 其中 pBuf 是要被发送的数据，lLength 为发送的数据长度，这里 RTP 负载类型设置为 0，标识为 false，时间戳增量为 10。一般同一个 RTP 会话中后三个参数

是相同不变的。

4.2.3 无线视频系统接收端的设计

在接收端同样需要像发送端一样首先进行初始化工作，初始化方式与发送端类似。首先采用两层循环的 `PollDate()` 方法来接收数据，第一层循环中首先使用 `FIRST` 函数查找第一个有效的数据源，找到第一个数据源之后便可以使用 `NEXT` 函数查找第二个数据源了。然后继续如此循环地检查，这样遍历所有带有数据的源。在第二层循环中，系统会将获取第一层循环中所检测到数据源中的数据。当获得的数据报使用完毕后，将其释放即可。图 4.3 为数据接收流程图。本文所设计的接收端也同样带有缓冲区。通过 `outfile1=open()`，便可以打开存缓冲区中数据^[28]。

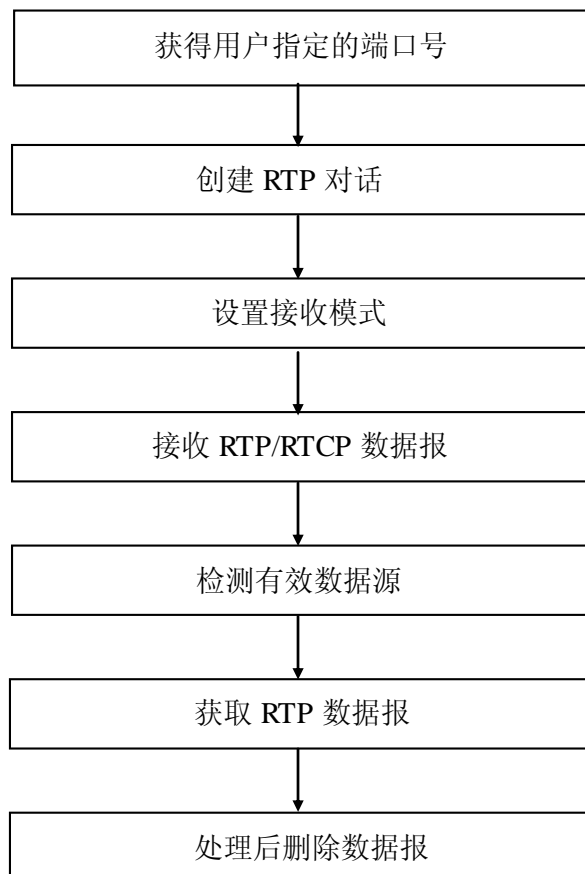


图 4.3 数据接收流程图

Fig4.3 Flow chart of receiving data packet

JRTPLIB 提供了三种 RTP 数据报接收模式，三种模式通过设置能够接受该

模式所指定的数据。设置接收模式可以通过调用 `RTPSession` 类的 `SetReceiveMode()` 方法, 三种模式分别为:

(1) `RECEIVEMODE_ALL`

默认模式, 接收任何的 RTP 数据报。

(2) `RECEIVEMODE_IGNORESOME`

首先通过调用 `AddToIgnoreList()` 函数来进行设置列表, 被设置的数据源将拒绝接收。

(3) `RECEIVEMODE_ACCEPTSOME`

同样使用 `AddToIgnoreList()` 函数来进行设置。但与上一种接受方式相反, 系统会接受被设置的数据源的数据报^[29]。

第 5 章 系统的移植与测试

5.1 嵌入式 LINUX 系统的移植

如前所述,本文选择了嵌入式 Linux 系统为开发程序的底层操作系统。因此在安装运行应用程序之前,需先将 Linux 系统移植到开发板中。所谓移植就是让一套软件在不同的硬件平台上正常工作的过程。这里软件可以为系统软件或应用软件。由于嵌入式系统本身的硬件资源有限,无法与 PC 等相比,所以不能将一套完整的 Linux 系统移植过来。必须先为系统进行合适的剪裁,并对内核进行重新配置,然后再移植到开发板上。另外由于嵌入式系统的硬件资源等原因,一般是不能够胜任代码的开发、编译与调试工作的。所以,需要在 PC 机上建立交叉编译环境。另外,嵌入式 Linux 系统与 PC 不同的是,在硬件设备刚启动时还需要一个小的引导程序,最后还要进行一套文件系统的配置。综上所述,嵌入式 Linux 操作系统的移植一般包括:建立交叉编译环境、移植 BootLoader、内核移植、创建根文件系统等几个部分。

5.1.1 建立交叉编译环境

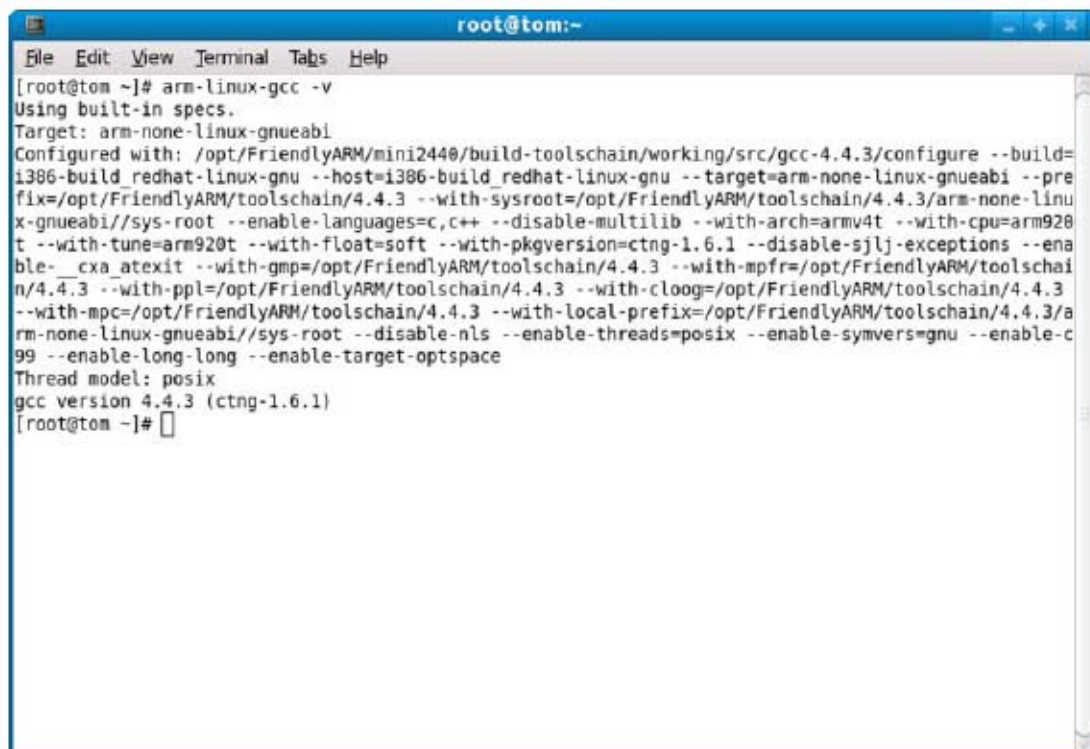
在开发 PC 机上的应用软件时,可以直接在 PC 机的操作系统上进行程序的编写、编译、调试,然后在 PC 机上直接运行。而嵌入式设备上只有一个空白的操作系统,没有编译调试的工具,所以必须要借助 PC 机,在 PC 机上进行编译之后烧写入嵌入式设备中。但 PC 机上自身使用的编译工具编译后的程序无法在嵌入式系统中运行,所以 PC 机必须使用一套适合嵌入式系统的编译工具。一种代码原本无法在另一种平台下工作,通过一种编译接收将其称为可能,这种技术就是交叉编译,交叉编译在如本文这种嵌入式开发中十分重要^[30]。

本文选用的交叉编译器为 arm-linux-gcc-4.4.3 版本,安装方法如下:首先将 arm-linux-gcc-4.4.3 复制到任意目录下,之后进入该目录并执行以下命令进行解压安装:

```
#cd /tmp
```

```
#tar xvfz arm-linux-gcc-4.4.3.tgz -C /
```

之后在命令行输入 arm-linux-gcc -v, 会出现如图 5.1 中信息,至此交叉编译环境便成功安装了^[30]。



```

root@tom:~# arm-linux-gcc -v
Using built-in specs.
Target: arm-none-linux-gnueabi
Configured with: /opt/FriendlyARM/mini2440/build-toolschain/working/src/gcc-4.4.3/configure --build=
i386-build_redhat-linux-gnu --host=i386-build_redhat-linux-gnu --target=arm-none-linux-gnueabi --pre
fix=/opt/FriendlyARM/toolschain/4.4.3 --with-sysroot=/opt/FriendlyARM/toolschain/4.4.3/arm-none-linu
x-gnueabi//sys-root --enable-languages=c,c++ --disable-multilib --with-arch=armv4t --with-cpu=arm920
t --with-tune=arm920t --with-float=soft --with-pkgversion=ctng-1.6.1 --disable-sjlj-exceptions --ena
ble-_cxa_atexit --with-gmp=/opt/FriendlyARM/toolschain/4.4.3 --with-mpfr=/opt/FriendlyARM/toolschain
/4.4.3 --with-ppl=/opt/FriendlyARM/toolschain/4.4.3 --with-cloog=/opt/FriendlyARM/toolschain/4.4.3
--with-mpc=/opt/FriendlyARM/toolschain/4.4.3 --with-local-prefix=/opt/FriendlyARM/toolschain/4.4.3/a
rm-none-linux-gnueabi//sys-root --disable-nls --enable-threads=posix --enable-symvers=gnu --enable-c
99 --enable-long-long --enable-target-optspace
Thread model: posix
gcc version 4.4.3 (ctng-1.6.1)
root@tom ~]#

```

图 5.1 交叉编译环境安装信息

Fig 5.1 Information of cross-compiling enviornment setup

5.1.2 BootLoader 的移植

不论是 PC 机还是嵌入式计算机,大多数设备在开机上电到操作系统系统需要一个引导程序。通过这个引导程序,可以初始化硬件设备,创建内核需要的一些信息并准备好软件环境,从而起到引导内核等作用。嵌入式 LINUX 系统中所使用的引导程序为 BootLoader。

本文所使用的开发板所自带的 BootLoader 是由三星的 vivi 优化而来,名为 Supervivi。该引导程序有两种菜单模式,可以根据需要而选择,能够烧写系统或调试。同时,由于它通过 USB 与设备相连,所以烧写与下载很快,并且在 PC 机上有良好的交互界面。它有两种操作模式:启动加载模式和下载模式。启动加载模式会自动加载 Linux 系统,其中不需要任何认为操作,开发完成的产品都会在这种模式下工作。下载模式中我们可以通过 Supervivi 提供的菜单或者直接使用命令的方式将内核映像和根文件系统映像等下载到 RAM 中并进行适当的调试^[31]。

Supervivi移植过程如下,移植时需要借助一个名为DNW的软件,该软件能够将引导程序快速的烧写到Nand Flash分区中。首先打开DNW软件,将开发板通

过USB与PC机相连。在PC上建立超级终端，超级终端中会出现相应选择，选择V之后超级终端开始等待DNW的同步传输。此时点击DNW中的传输选项，便开始自动的下载引导程序了。

通过以上几个步骤就完成了 supervivi 的移植过程，之后便能够通过 supervivi 初始化硬件并调用内核了。

5.1.3 系统内核的移植

针对不同的硬件平台，在进行内核移植之前要对内核进行相应的剪裁、配置等工作。

1. 首先要获取内核，本文使用的是 Linux-2.6.32.2 版本。从网上下载此内核的压缩包，放入任意目录下。并执行：`#tar -jxvf linux-2.6.32.2-rc4.tar.bz2`，之后进入内核目录。

2. 修改内核目录下根目录中的 Makefile，其中把“ARCH?=”和“CROSS-COMPILE?=”分别修改为“ARCH ?=arm”和“CROSS_COMPILE ?=arm-linux-”。如此便指明了平台并配置交叉编译器。

3. 接下来我们要配置内核，执行 `make menuconfig` 命令。我们会看到内核的配置单，如图 5.2 所示。

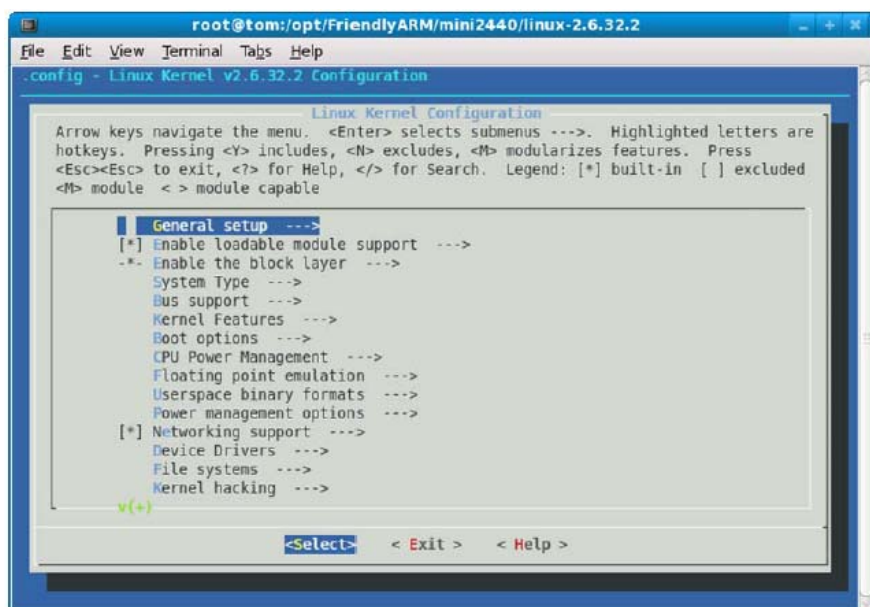


图 5.2 内核配置菜单

Fig 5.2 kernel configuration

在这里我们要对开发板的各个硬件支持做一正确的调整。以 CPU 为例，在

主菜单选择 System type, 我们可以看到很多 CPU 的型号, 选择适合本开发板的 S2C2440。利用此方法, 对显示屏、鼠标、键盘、网卡等进行配置。除硬件之外, 驱动如 nand flash 驱动支持、yaff2s 文件系统的支持也许在此配置。配置完毕之后, make zlmage 便可以得到我们需要的内核文件了, 之后将其烧入开发板即可。

5.1.4 安装根文件系统

文件系统就是把磁盘上的文件进行一个明确清晰的分区的方法。如 WINDOWS 中的 C、D、E 盘, Linux 有着一套自己的文件系统。Linux 以目录的方式来管理所有文件, 最上层目录为根目录, 其他的为子目录, 所有目录构成 Linux 的根文件系统。每个目录都有着自己的功能。目前比较常见的嵌入式文件系统有: **RAMFS、CRAMFS、ROWS、JFFS2、Yaffs。**

本文选择的是 Yaffs 文件系统。下载 YAFFS 压缩包, 之后与 bootloader 或者 Linux 内核一样, 对其进行修改编译器、修改配置, 然后执行 #make 命令生成根文件系统。使用超级终端配合之前使用过的 DNW 软件将其下载至开发板, 下载完毕之后自动将带有该文件系统的种内核安装到 Nand Flash 中。如此便完成了根文件系统的移植。

5.2 系统测试

首先在测试之前要搭建一个测试环境, 在测试的过程中能够方便的对系统进行调试。本系统的测试环境结构图如图 5.3 所示。

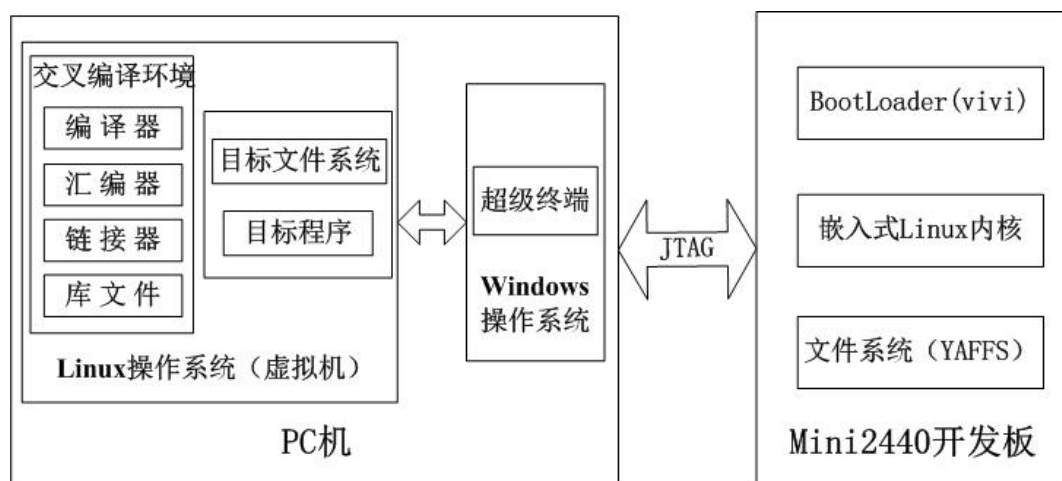


图 5.3 系统测试环境结构图

Fig 5.3 the picture of system testing

本文中实现嵌入式操作系统与 PC 机互通的方式为超级终端，配合开发板上 SUPERVIVI 能够很容易的实现传输文件的功能。为了方便的使用 LINUX 与 WINDOWS 实现文件共享，本文选择的搭建虚拟机并共享的方式。在虚拟机的 LINUX 中建立交叉编译环境，在传到 WINDOWS 中通过超级终端烧入开发板。

建立好测试环境之后，我们对该系统的系统运行效率情况、播放帧率以及传输帧率等方面进行测试。

5.2.1 系统运行效率测试

由系统的运行效率上，们能够看出我此系统的硬件性能是否满足软件的要求，或者软件的优化是否足够。本文主要对采集端的 CPU 与内存使用情况，监控端的单路以及多路的 CPU 与内存使用情况做了测试该测试过程是在 Linux 环境下调用 top 命令进行统计的。测试情况如表 5.1 所示。

表 5.1 系统运行效率测试表

Table 5.1 table of system efficiency testing

测试端	监控内容	第一次	第二次	第三次	第四次	第五次
采集端	CPU 占用率	11.2%	11.3%	11.9%	11.0%	11.6%
	内存占用率	6.4%	6.5%	6.4%	6.5%	6.6%
监控端 (单路全屏)	CPU 占用率	15.2%	17.4%	17.8%	16.2%	15.4%
	内存占用率	7.0%	7.1%	7.4%	6.9%	7.3%
监控端 (多路分屏)	CPU 占用率	22.0%	26.1%	25.9%	27.6%	29%
	内存占用率	7.0%	7.6%	7.4%	7.7%	7.4%

以看出，采集端的 CPU 平均使用率为 11.5%，内存平均使用率为 6.5%；监控端单路工作时 CPU 平均使用率为 16.5%，内存平均使用率为 7.2%；监控端多路工作时 CPU 使用率比单路要高一些，平均使用率为 26.3%，内存平均使用率为 7.3%。可以看出，该系统在 S3C2440 硬件支持下能够完美的运行。

5.2.2 播放帧率测试

多路视频监控系统的采集工作的相对独立的,每个采集端之间并不会相互影响,在计算帧率时将按平均值计算。监控端只有一个,在进行多路播放时把所有路的帧率求和再除以路数计算。测试的情况如表 5.2 所示。

表 5.2 系统帧率统计表

Table 5.2 table of system frame rate

功能端	30s	60s	300s	1200s	平均值
采集端	13.1	13	12.8	12.7	12.8
监控端单路	12.8	12.9	12.9	12.7	12.7
监控端二路	10.5	10.6	10.4	10.3	10.7

从表中可以看出,采集端在发送数据时其帧数在 13 帧上下浮动。监控端在单路工作时,基本与发送的帧率保持一致。而两路播放时帧率明显下降了平均 2 帧,播放画面基本流畅,满足该系统流畅监控的功能需求。在实际使用中,可能会要求更多的路数,播放帧率还会降低。但是对于本系统来说,基本完成了预期希望实现的效果,证实了无线多路视频监控系统播放的可行性。

5.2.3 传输测试

本文对传输距离进行了测试,从 1 到 80 米的每个 5 米进行一次帧率传输的测试,得出了单路接收端距离与帧率之间的关系。如表 5.3 所示

表 5.3 单路方式接收端帧率与距离的关系

Table 5.3 The relationship of frame rates for reception terminal with distance in one channel

距离	1 米	10 米	20 米	30 米	40 米	50 米	60 米	70 米	80 米
帧率(fps)	13.9	13.5	12.6	11.1	8.7	5.1	1.9	0.3	0

从表中我们可以发现,随着距离增加传输帧率下降。IEEE802.11g 无线局域网的传输范围的理论值为 200 米,但这个值是理想环境下。本系统测试是在教学楼中,有很多外界的干扰。所以在实际使用中本系统的最近范围在 30 米之内。

第 6 章 总结与展望

6.1 工作总结

当今这个计算机技术、网络技术、信息技术飞速发展的时代,视频监控系统已经发展到了以嵌入式为基础的第三代视频监控技术。无线视频监控系统作为第三代视频监控系统的一个分支,具有无需布线、携带方便、使用灵活等优点。在在医疗事业、军工国防、工业控制以及各种街道、商店、码头等公共场有着广泛的应用。

本文研究了基于 ARM9 的多路无线视频监控系统。根据该系统的特点,对系统的软件以及硬件做了整体结构设计。并对系统划分了各个模块,分析了各个模块之间的关系。通过对当前市场所流行的硬件以及软件的比较,从综合角度出发选择了较为合适的组合实现了各个模块的功能。成功移植了内核以及监控程序,并对系统进行了测试。实现了本文中所需要的采集、传输等功能。本文主要完成的工作如下:

1. 详细的研究了系统所涉及的技术和方法,并选择了较为合适本系统的相关技术加以开发。对系统涉及的视频采集技术、视频编解码技术、无线网络传输技术等作了比较详细的介绍与说明。

2. 对系统进行了功能分析,将系统分为采集端与监控端,完成了系统的模块化分,将系统分为采集模块、传输模块与监控模块。提出了各模块的设计方法与流程并实现。

3. 完成了 USB 摄像头驱动的移植工作,使用 VIDEO4LINUX 相关技术编写了视频采集程序,实现了 Z301P 芯片 USB 摄像头的视频数据采集。提出并设计了双针采集以及双缓冲采集方法,加快了视频采集速率。

4. 实现了 FFMPEG 在嵌入式视频监控系统上的应用,使用 MPEG-4 标准对采集到的视频数据进行编码压缩。提出了在接收端使用多线程技术实现对各个采集端发送的视频数据同时接收的方法,并使用 RTP/RTCP 作为传输协议完成了系统的视频数据的发送与接收。

5. 对系统内核做了剪裁与编译,为无线网卡、USB 摄像头做了系统内核配置。并将内核移植到开发板上。

6. 搭建了测试环境并对系统做了相关测试。包括系统运行效率测试、播放帧率测试、传输测试,并给出了具体测试结果以及测试分析。实现了多路视频同时播放,达到了 15 帧/s 的流畅程度。

6.2 工作展望

本文所研究的无线视频监控系统只是视频监控系统中的一个分支,笔者只是对该系统所涉及的几个关键技术做了研究。本无线视频监控的研究只是一个初步阶段,还存在的一些问题。在测试中能够发现,单路监控时能够较好的反应出监控地点的情况,但三路以上时便会出现帧数下降。在以后的研究中需要进一步的对线程调度、系统优化以及压缩算法上进行研究。本文的视频监控系统仅提供了视频图像的采集功能,在今后还可以进行音频信号的采集传输。随着嵌入式系统的进一步发展,可以在图像采集之后便在本机直接进行一些分析工作,通过分析来实现其他一些不同的功能,以此来预警或节省资源。还可以考虑在客户监控端的软件中加入识别或人体跟踪算法。

本文研究的无线多路视频监控系统还只是初步的一个只能够简单监控的系统,今后还有更多的功能项目等着我们去研究,可以将视频监控端软件成为一个功能更加齐全完善的客户端。由于作者水平所限,在文中难免会出现披露,希望各位老师和读者指正。

参考文献

- [1] 曹巧玲, 齐红强. 视频监控技术简介与未来技术发展趋势[J].2010-02-15.
- [2] 朱曙. 新一代数字监控系统[J]. 中国多买提视讯.2004(11)50-5.
- [3] 王伟岗. 青岛公安图像综合应用平台的研究及应用[D].2010.
- [4] 李勋. 无线视频监控系统关键技术研究[D].2012:2-10.
- [5] 王磊.基于多线程的嵌入式远程监控 WEB 服务器的设计与实现[D]. 2011.
- [6] 吴伟.嵌入式实时操作系统在 ARM 系列微处理器上的移植研究 .2007.
- [7] Window CE[OL]. <http://baike.baidu.com/view/41539.htm>.
- [8] 张锋涛.基于 i.MX255 的物联网网关的研究与设计[D].2012.
- [9] 杨磊.基于嵌入式 Linux 视频采集平台的开发. 2008.
- [10] 荣佳, 贺良华. 基于嵌入式 LINUX 的 USB 视频采集模块的驱动开发[J]. 计算机与数字工程, 2009-02-20.
- [11] d_south. video4linux(v4l) 使用摄像头的实例基础教程与体会 [OL]. <http://blog.csdn.net/dotphoenix/article/details/4929775> .
- [12] d_south. video4linux(v4l) 使用摄像头的实例基础教程与体会 [OL]. http://blog.csdn.net/d_south/archive/2009/06/22/4288836.aspx.
- [13] 邓斯佳. 基于 ARM 的虹膜图像采集系统[J].2011.
- [14] 杨海山.基于 S3C2440 的田间视频采集系统研究与开发[D] . 2009.
- [15] Gray'note. 基于 Video4Linux 的 USB 摄像头图像采集实现 [OL]. <http://blog.csdn.net/ipromiseu/article/details/4249193>
- [16] 荣锐. 基于 ARM9 的视频传输系统研究[D]. 2009.
- [17] 杨秋蔚. 基于 ARM9 无线图像采集系统的研究与开发[D].2009
- [18] Tommy_wxie. Linux 多线程函数解析 -Kernel Hacking- 博客频道 [OL]. http://blog.csdn.net/tommy_wxie/article/details/7209561
- [19] 刘国建. 无线电子终端关键技术研究及其应用开发[D] 2008-12-27.
- [20] 李明. 嵌入式智能视频监控系统的的设计[D]. 2009-06-30.
- [21] 柳亚东. 基于 S3C2440 的嵌入式视频网络监控系统[D]. 2009-01-01
- [22] 张玉民. 嵌入式环境中流媒体网络视频监控系统的研究与开发 [D].

2008-05-01

- [23]徐方艾. 基于 S3C2410 的无线视频监视系统[D]. 2009 5-10.
- [24]郑磊. 基于嵌入式 Linux 的网络视频监控系统研究[D]. 2009-03-01.
- [25]张鹏, 贾敏智. 基于嵌入式 Linux 的网络视频监控系统[J]. 2008-08-15.
- [26]W.RICHARD STEVENS, STEPHEN A.RAGO.ADVANCED PROGRAMMING
IN THE UNIX ENVIRONMENT[M]. 人民邮电出版社, 2006.5.
- [27]rtsp++网络视频传输[OL]. <http://blog.csdn.net/perfectpdl/article/details/5939541>
- [28]刘楠. 无线视频监控系统多路传输与控制技术研究[D]. 2012-04-01.
- [29]jrtpplib 介绍[OL]. <http://blog.csdn.net/scq2099yt/article/details/2219971>.
- [30]杜宝祯. 嵌入式无线网络化测控系统的设计与实现[D]. 2010-06-01.
- [31]张文涯. 基于嵌入式 Linux 的网络视频监控系统设计与实现[D]. 2009-05-01.

作者简介及在学期间所取得的科研成果

李斌来，男，1987 年 4 月 14 日出生于吉林省长春。2010 年 7 月 1 日获得北华大学工学学士学位，目前，攻读吉林大学工程硕士学位。

参加的项目：2011 年 9 月至今，参与无线视频监控系统项目的设计与实现，主要负责视频采集模块、传输模块的设计工作。

电话：

邮箱：

致 谢

本研究及学位论文是在我的导师赵宏伟老师的亲切关怀和悉心指导下完成的。他严肃的科学态度，严谨的治学精神，精益求精的工作作风，深深地感染和激励着我。论文的选题，项目研究方向的把握，以及论文撰写过程中的评审等等，赵老师结合自己多年来丰富的科研项目的经验为我们在攀登学术的高峰中引路，在学术上的每一次的进步都离不开赵老师的谆谆教导。赵老师不仅在学业上给我以精心指导，同时还在思想、生活上给我以无微不至的关怀，在此谨向赵老师致以诚挚的谢意和崇高的敬意。同时，我还要感谢在一起愉快学习、生活的实验室的同学们，正是由于你们的帮助和支持，我才能克服一个一个的困难和疑惑，直至本文的顺利完成。

另外，感谢我的父母，他们在对我培养的过程中付出很多很多，感谢父母对我的理解与包容，关心和爱护，正是由于你们的默默鼓励与支持，我才会取得今天的成绩。感谢我的朋友们，有了你们的陪伴，让我的人生变得更加的绚丽多彩。

在论文即将完成之际，我的心情无法平静，从开始进入课题到论文的顺利完成，有多少可敬的师长、同学、朋友给了我无言的帮助，在这里请接受我诚挚的谢意！最后我还要感谢培养我长大含辛茹苦的父母，谢谢你们！

最后，再次对关心、帮助我的老师和同学表示衷心地感谢！