**摘要**

嵌入式系统技术经过几十年的快速变革。现在已经深入到我们生活中的方方面面。从Inter所制作到第一个8051芯片，到现在的Crotex-A9，Crotex-A53。嵌入式系统的CPU主频和内存大小已经提升了几十倍。人们因为嵌入式系统的发展使得生活质量变得更好。城市的自取自行车系统，指纹门锁和倒车雷达预测等等。通过在嵌入式芯片中运行系统和软件算法，从而实现人们日常生活中的一些需求。

本次系统使用Crotex-A9平台的三星Exynos 4412处理器，使用低成本USB摄像头作为采集前端，对A9开发平台移植嵌入式Linux操作系统。完成对摄像头视频采集和使用网络对视频的传输。首先采集出摄像头的每一帧图像数据，图像数据为YUV或MJPEG格式。在对图像作相关处理后，把每一帧的图像数据通过网络输送到每一个需要的用户中。当用户打开浏览器，登录相关的服务器地址，就可以看到视频连贯的帧数据。

在本次的软件设计中，系统软件选择开源的嵌入式Linux操作系统，系统内核和配置会根据使用的外设情况进行相应的裁剪。以此来缩减系统的复杂度。在程序软件方面，使用C语言来调用相关的系统调用API和库函数，完成对系统设计的需求。在采集层，使用符合UVC协议的摄像头并使用Linux中的v4l2 API进行开发。在传输层，使用Linux系统中的socket套接字进行基础的TCP/IP连接。同时，搭建http服务器。把每一帧数据放入服务器等待用户读取。

**关键字**： 嵌入式； 视频采集； 数据传输； Linux系统；

**Abstract**

Embedded system technology after decades of rapid change. Has now penetrated into all aspects of our lives. From the Inter to the first 8051 chip, to the current Crotex-A9, Crotex-A53. Embedded system CPU frequency and memory size has been improved several times. People because of the development of embedded systems make the quality of life better. Urban self-extracting bicycle system, fingerprint lock and reversing radar forecasting and so on. Through the embedded chip running system and software algorithms, in order to achieve some of the needs of everyday life.

The system uses Crotex-A9 platform Samsung Exynos 4412 processor, using low-cost USB camera as a collection front-end, the A9 development platform for transplanted embedded Linux operating system. The completion of the camera video capture and use of network-to-video transmission. Firstly, the image data of each frame of the camera is collected, and the image data is YUV or MJPEG format. After the correlation processing of the image, the image data of each frame is transmitted to each required user through the network. When the user opens the browser, login the relevant server address, you can see the video frame data.

In this software design, the system software selection open-source embedded Linux operating system, the system kernel and configuration will be based on the use of peripherals accordingly cut. So as to reduce the complexity of the system. In the program software, the use of C language to call the relevant system call API and library functions, the completion of the system design requirements. In the acquisition layer, using the camera in line with the UVC protocol and use the Linux v4l2 API for development. At the transport level, the underlying TCP / IP connection is made using socket sockets in the Linux system. At the same time, build http server. Put each frame of data into the server waiting for the user to read.

**Key words**: embedded; video capture; data transmission; Linux system;

**目 录**

摘要

Abstract

第一章 绪论

1.1 嵌入式视频发展现状

1.2 研究背景及意义

1.3 设计研究的主要内容

1.4 本文的组织结构

第二章 设计方案分析

2.1 系统整体结构分析

2.1.1 系统需求分析

2.1.2 系统设计构思

2.1.3 系统结构框图

2.2 图像采集方案分析

2.2.1 摄像头输出分析

2.2.2 程序采集结构

2.3 网络传输方案分析

2.3.1 设计要求

2.3.2 设计方案

2.4 本章小结

第三章 系统设计

3.1 系统硬件结构组成

3.1.1 摄像头

3.1.2 Cortex-A9 4412嵌入式平台

3.2 系统软件平台

3.2.1 嵌入式Linux系统

3.3 服务端硬件系统连接

3.4 服务端网络连接

3.5 服务端系统结构组织划分

3.6 本章小结

第四章 程序设计和系统实现

4.1 开发流程介绍

4.2 开发环境搭建

4.2.1 Linux系统内核配置

4.3 采集层控制

4.3.1 v4L2视频采集接口

4.3.2 采集流程与实现

4.4 视频传输

4.4.1 视频传输流程

4.4.2 socket套接字

4.4.3 建立http服务器

4.5 本章小结

第五章 系统测试与扩展

5.1 测试条件和环境

5.2 性能测试

5.3 系统可扩展性分析

5.4 本章小结

第六章 总结

致谢

参考文献

附录

**第一章 绪论**

**1.1 嵌入式视频发展现状**

嵌入式系统体积小、功能强大、方便灵活、其系统稳定。已经应用于工业、农业、教育、国防、科研和生活中的各个领域。推动了自动化进程，是自动化生产和提高效率的一把推手。

自从人们进入信息时代，嵌入式视频就开始了飞速的发展。嵌入视频应用于人们生活之中的方方面面。交通灯的视频监控，安防的摄像，以及人们所使用的视频通话等等。随着科技和技术水平的不断提高，嵌入式视频将会变得更加智能化、网络化、规范化和集成化。并且嵌入式视频还将与其他各个行业的知识和内容结合起来。比如，最近十分流行的大数据、人工智能等等。人脸识别锁就是其作品之一。

它的出现，目的是要把生活变得更加简单、更加方便、更加舒适。嵌入式视频系统就像是一个专用电脑。实现专业、专用化计算。因此这也是未来世界的“数字基因”。嵌入式视频系统将为我们提供无限的想象空间。它就像是一个黑洞，会把当下世界的很多技术和成果吸引进来，同时再加以结合，来打造一个未来的数字时代。

**1.2 研究背景及意义**

嵌入式视频是各种视频监控系统的重要组成部分，它以直观、丰富的内容信息而广泛的应用于各种场所。嵌入式视频系统在面积很小的板上完成视频的采集、控制、和网络传输。相当于实现了一个远程的视频监控系统。但是又有区别于传统的监控系。传统的监控系统要求使用PC端或使用专用的客户端来查看，而本次的系统设计只需要使用浏览器即可观看，大大降低了产品的成本。

本系统使用使用Web网页来展示视频信息，只需要用户和采集端处于同一个物理连接下。如果对系统进行拓展，那么可以把数据传输到互联网，只要用户端可以使用互联网。那么就可以观看到视频数据。这时，系统就变成了直播系统。大家可以互相观看其他人的视频。

另一方面，可以在采集后端对图像进行处理和检测。比如使用opencv对图形进行人脸识别。或对图像做分割处理。这样系统就成为了一个智能识别的签到系统。就研究的意义来讲，本次设计的系统可扩展性很强。

**1.3 设计研究的主要内容**

本次的系统设计使用成本较低的USB摄像头作为视频采集的前端，嵌入式处理器Exynos 4412来完成视频处理，数据传输等功能。同时，系统使用免费且开放源代码的Linux嵌入式操作系统来管理系统的硬件和模块。这样，系统的成本会降低很多。

在采集方面，主要涉及到Linux系统下的摄像头视频采集和相关的文件操作。在数据传输方面，主要涉及到网络编程等相关内容。需要使用socket套接字来建立一套和用户端的连接。系统开发时，在运行Linux的PC机上对程序进行交叉编译，生成的二进制可运行文件即可复制到嵌入式开发板上运行。

**1.4 本文的组织结构**

本论文总体分为六个章节：

第一章为绪论，说明嵌入式视频的发展现状，系统的研究背景和研究意义。最后介绍本次设计主要涉及哪些内容。

第二章为系统设计，首先对总体系统进行需求分析和研究，总结出大的系统框架和结构。之后分析作为服务处理端所需要完成的工作，并分为采集和传输两大部分来分别设计。

第三章为服务处理端系统设计，先说明系统所采用的硬件结构和它们之间的连接关系，再说明系统所采用的软件框架。最后对系统内容进行整理和总结。

第四章为系统的具体实现过程。从硬件的具体连接，到软件环境的搭建。再针对具体的需求使用程序来实现和完成。最后对操作的过程进行总结。

第五章为系统测试和扩展。主要对系统的功能和性能做一个科学严谨的测试。再对其可扩展性进行说明和分析。

第六章为总结。是对本论文的全文总结。

**第二章 设计方案分析**

**2.1 系统整体结构分析**

**2.1.1 系统需求分析**

本次系统需要完成对视频的采集和传输，要求在用户端可以流畅的观看各个节点的视频。在采集方面，视频分辨率不能太低，至少需要320\*240及以上。想要无卡顿的看到播出的视频数据，那么视频的帧率则至少在5帧及其以上。在视频数据传输方面，要使用TCP/IP进行网络传输，那么对系统传输的带宽则有相应的要求。如果传输帧率或分辨率较高，那么对所需要的网络带宽就会有较高的要求。

**2.1.2 系统设计构思**

通过系统需求分析，嵌入式系统处理器每秒至少需要采集320\*240\*2\*5个字节的数据。而且，在采集数据的同时，需要向网络中发送视频数据。那么在处理器选择方面，就直接排除掉ARM9平台以下的所有处理器。如果使用ARM9平台的处理器，在之后的功能扩展方面就会收到一些限制。正好在我手里有一块Crotex-A9平台的开发板，本次设计就使用这个块开发板。

当系统和用户端进行网络视频数据传输时，将使用网络的service-client模式。显然，作为用户端不可能实时主动去联系服务端。那么在嵌入式平台中的网络程序就作为Service程序，在用户端连接至Service时，程序打开网络接口和采集接口，对视频数据进行采集和传输。

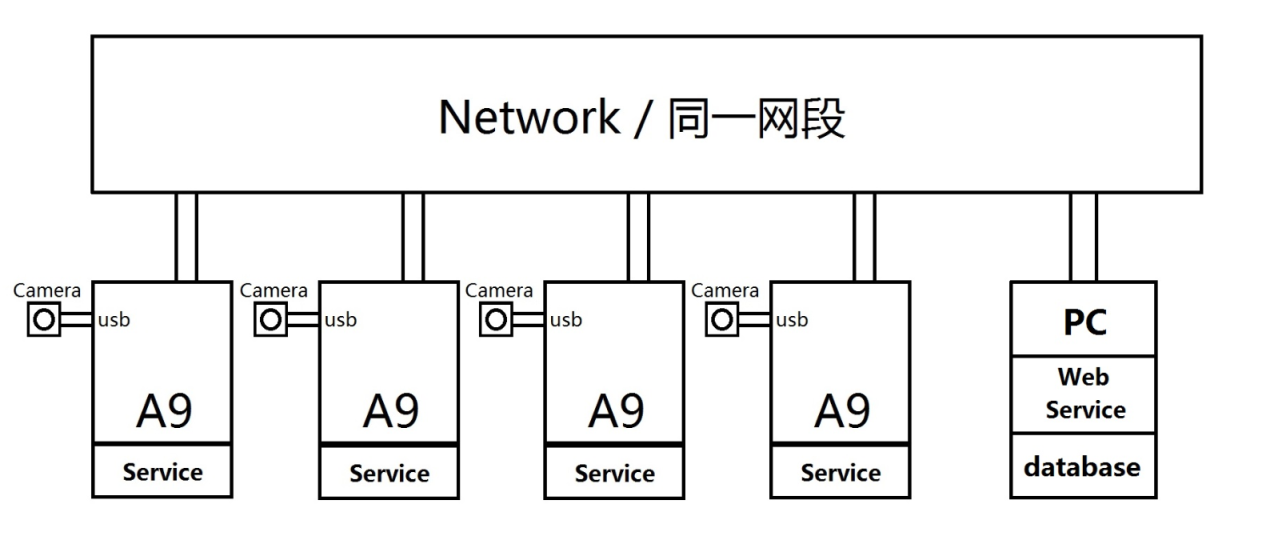
考虑到系统的可扩展性，在实际的应用场景中，系统中不可能只有一个视频采集、传输的节点。那么考虑到多节点的扩展和兼容性，使用交换器或路由器连接各个节点系统，让每个节点的物理层都互相连接。之后利用网络配置，就可以使其在同一网段内。

在使用上位机（Web用户端）对服务端进行访问时，相当于用户端做为client去请求Service中的内容。同时，在用户端PC上可创建一个用于管理整个系统的数据库。在数据库中记录每个视频节点的位置信息。

作为Web用户端，其本身可作为一个http服务器。只要其他用户与其在同于物理连接层内，就可以做为一名用户，访问http服务器。http服务器再去请求用户所想要访问的节点中服务器的数据。整个系统就是一个分布式的系统。

**2.1.3 系统结构框图**

经过系统需求、设计构思和分析。系统的结构框图如下图2-1所示，所有系统节点、服务器、用户同在一个物理连接层内，可以通过交换机连接。没个单独视频节点拥有独立的IP地址，通过用户选择不同的节点来选择播放不同视频。

****图2-1 系统框图

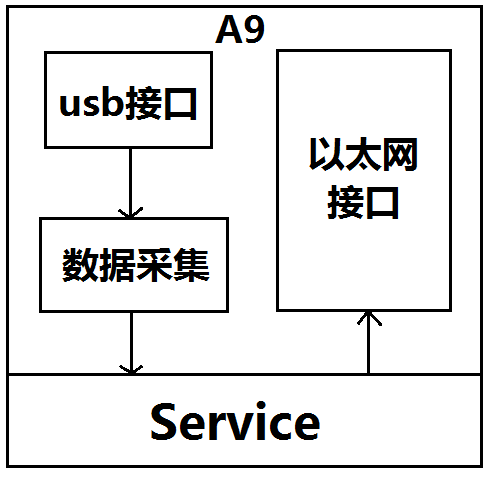
本次的服务端设计的系统框图如下图2-2。使用程序采集USB摄像头所输出的视频数据。当有用户连接时，使用网络接口，利用服务器把数据发送给用户端，用户在Web界面上就可以看到当前的视频信息。

图2-2 服务端系统框图

**2.2 图像采集方案分析**

**2.2.1 摄像头输出分析**

本次设计使用的摄像头是通用的USB接口摄像头，该USB摄像头符合标准的UVC（Linux）驱动。在从摄像头中读取帧数据时，有两种数据格式可以选择。一是YUV格式，二是MJPEG格式。这是市面上经常会有的两种格式，YUV格式较多，MJPEG格式较少。YUV格式是无压缩的，数据量大。MJPEG格式是压缩的格式，数据量较小。早期的计算机由于处理器性能不够，USB摄像头多采用MJPEG格式输出数据。现代的计算器处理器性能较好，USB摄像头多采用YUV格式输出数据。

YUV格式是颜色的一种编码方法，经常被欧洲彩色电视所使用。在YUV的三个字母中，分别代表三个分量。其中，Y代表着明亮度，也就是灰度值。而剩下的UV则代表着色彩和饱和度，是用来描述指定像素的颜色。YUV格式把亮度信息Y和色彩信息UV互相分离，也就是说，如果只有Y分量，那么图像就是一幅没有色彩的黑白（灰度）图像。这样就可以兼容早期的黑白电视。现在市面的USB摄像头大多支持这种格式的输出。

MJPEG是一种视频编码格式。中文翻译为“技术即运动静止图像（或逐帧）压缩技术”。是把运动的视频序列当做连续的静止图像来处理。它是压缩视频中的每一帧，并没有采用帧间压缩。MJPEG压缩源于JPEG压缩。所以，把MJPEG压缩中单独一帧数据提取处理，就可以保存为JPEG格式的图片。

表2-1 YUV和MJPEG对比

|  |  |  |
| --- | --- | --- |
|  | YUV | MJPEG |
| 是否压缩 | 否 | 是 |
| 占用带宽 | 大 | 小 |
| 三方工具打开 | 否 | 是 |
| 数据大小 | 大 | 小 |

上表2-1为YUV和MJPEG格式的对比，如要要在嵌入式系统中选择这两种格式的数据。那么最好选择MJPEG格式，这样就可以给CPU节省很大的负担。因为嵌入式系统的CPU采用ARM架构，使用精简指令集。而我们平时使用的PC则使用的是复杂指令集。

**2.2.2 程序采集结构**

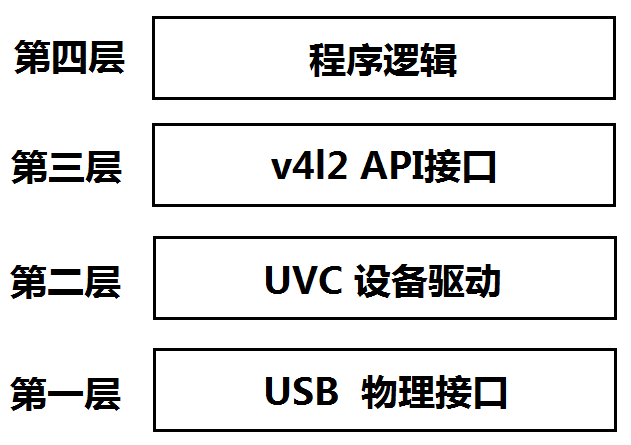
视频采集的接口框架是基于Linux系统的。我把它分为了四层，如下图2-3所示。

图2-3 程序采集结构

第一层是物理接口。使用USB主机模式连接使用USB模式的摄像头。通过USB的物理接口来传输数据。

第二层是Linux设备驱动层，要让运行Linux系统的开发板识别USB摄像头，那么在Linux内核中需要拥有该设备点的驱动程序。本次设计使用标准的UVC设备驱动，只需要在编译内核时添加相关的驱动选项即可。而不需要自己去写Linux驱动程序。

第三层是程序API接口，Linux系统给我们提供了一个标准的视频采集接口v4l2，全称为video 4 linux 2。使用v4l2来设置摄像头的相关参数，比如分辨率，帧数，图像参数格式等。

第四层是程序逻辑层，也就是程序的主体。在这一层实现如何把采集到到每一帧视频数据传输到用户端，而且要做到流畅和稳定。其就包括时序和资源的处理。总体来讲就是需求用程序实践的逻辑。

**2.3 网络传输方案分析**

**2.3.1 设计要求**

服务端网络的传输，本质上是建立一个网络服务器。作为一个网络服务器，需要给众多的用户同时提供服务。比如，多个用户端可以同时观看同一个节点的数据。从程序的设计角度来看，就需要程序满足并发性的设计。在和一个用户通信时，还可以和另外一个用户进行通信。这样，就需要使用到计算机中的多进程，或多线程技术。以此来满足系统的设计需要。

**2.3.2 设计方案**

在使用程序来完成所需要的设计要求时，要选择合适的网络结构模型。常用的网络结构模型有以下几种：

1. 一问一答的服务器/客户端模型
2. 多线程服务器模型
3. 多进程服务器模型
4. IO多路复用模型

在第一种模型中，是一种一对一的同步阻塞模型。也就是说，如果没有用户连接的话，程序会一直阻塞住（accpte()函数），直到有用户连接。而且是一对一，如果已经有一个用户在连接，那么第二个用户将不能再连接。并且对于阻塞模型来说，会很占用系统资源。因为cpu会一直等待阻塞，会浪费资源。

第二种模型中，是一种一对多的模型。每当一个新用户连接至服务器时，程序会给其创建一个新的线程来完成其所需要的任务。在程序做资源管理的时候，可以使用线程池来管理众多的连接线程。但是，它同样也有它的缺点。使用线程的稳定性相对较差，一个线程崩溃可能会导致程序的整个崩溃。第二，因为线程有共同的程序空间，在对临界资源访问时，需要加锁解锁。否则可能会造成死锁的现象。但是这样一来，又会降低系统的性能。最后，这种模型也是阻塞模型，还是会等待用户来连接。

第三中模型和第二种模型很相似，不同的是使用了进程来代替线程来完成并发的操作。在Linux系统中，父进程和子进程使用不同地址空间。所以不用考虑临界资源的操作。它的缺点也是进程的缺点，fork一个进程会耗费时间和资源。一个进程占用资源会比一个线程所占用的资源大。进程在销毁时，要符合系统的标准流程，否则会造成僵尸进程或主进程一直等待子进程。最后，它也是一个阻塞的模型。

在第四中模型中，使用了异步非阻塞的方式。通过调用Linux中的select函数或epoll函数来实现。那么阻塞和非阻塞有什么区别呢，就好比烧一壶水。如果是阻塞方式，当你开始烧水时，那你就阻塞在烧水这里，一直等到水烧开，整个活动结束。但是如果是非阻塞方式的话，当你烧水时，你可以在烧水的同时干其他事情。当水烧开后，你听到水汽的声音，然后过去处理，整个活动结束。这是非阻塞方式。当同时连接到服务器的用户较多时，使用非阻塞的连接方式会使系统的性能提高很多。一般此模型用于并发连接较多的情况。

在此次系统设计中，由于不需要很大的并发连接。综合考虑后使用第二种多线程连接模型。

**2.4 本章小结**

在本章节，介绍了整个系统的框架和结构。结合系统框图展现了如何去实现系统的需求。并在服务端，把采集系统和传输系统两个系统分开设计和介绍。在采集端，使用USB摄像头连接A9开发板。利用Linux的相关驱动和API函数来完成采集任务。在数据传输方面，介绍了服务器设计所常用的几种模型。并说明了阻塞模型和非阻塞模型的区别。在具体的任务中需要来选择所需要的模型。

**第三章 系统设计**

**3.1 系统硬件结构组成**

**3.1.1 摄像头**

摄像头按数据模式大致可分为模拟摄像头和数字摄像头。模拟摄像头现在基本以已经被淘汰，随着数字技术和通信技术的发展。数字摄像头在信噪比，灵敏度和图像清晰度等各个方面都遥遥领先于模拟摄像头。

在数字摄像头中，分为CCD摄像头和CMOS摄像头。CCD摄像头一般应用于高端的设备中，成像效果比CMOS摄像头较好。CMOS应用于普通的数字设备中。虽然效果比不上CCD设备。但是CMOS集成度高，低功耗，成本低。CMOS摄像头顾名思义，是使用CMOS图像传感器组成的摄像头。一般由CMOS传感器和镜头组成。

在选择摄像头是，还需要考虑摄像头的视角和焦距。一般情况下，如果需要观察远景的话应选择长焦镜头，反之亦然。在选择摄像头视角时，如果需要摄入的图像多，则应该选择广角镜头。视角越广，纳入镜头的图像越多，但是图像也会相应的变形。所以在选择摄像头时，应综合考虑。

本次设计所采用的摄像头为深圳锐尔威视科技开发的摄像模组。该摄像头使用USB2.0接口。可录制视频的最大分别率为1280\*720。可输出3种格式的视频数据，分别为H.264、YUYV、MJPEG。其中YUYV和MJPEG是一个通道，H.264是另外一个通道。也就是说，该摄像头可以支持双通道的数据输出。详细的参数如下表3-1。

表3-1

|  |  |
| --- | --- |
| 产品型号 | RER-USB100W04H |
| Sensor | OV9712 |
| 最高有效像素 | 1280(H)X800(V) |
| 信噪比 | 40dB |
| 动态范围 | 69dB |
| 灵敏度 | 3.7V/lux-sec@550nm |
| 供电方式 | USB BUS POWER 4P-2.0mm插座 |
| 工作电压 | DC5V |
| 工作电流 | 120mA~220mA |
| 工作温度 | 0～70℃ |
| 板机重量 | 30g |

摄像头可以输出不同大小分变率的视频。如320\*240、640\*480、800\*600、1280\*720。通过在程序中使用ioctl函数来控制摄像头输出大小。

**3.1.2 Cortex-A9 4412嵌入式平台**

**Cortex-A9**是ARM公司所突出的架构，使用ARMv7指令集。**Cortex-A9**结构有如下特点：

1. 具有高性能的浮点单元（FPU），具有单精度双精度的浮点指令。
2. 具有图形处理单元ARM Mali™可以整合出高整合度的系统。
3. 提供低功耗、高性能应用的标准单元库和存储器，可以实现功耗控制等。
4. 具有更好的可调试技术，是开发人员可以更好使用。

ARM处理器一直受到智能终端的青睐。从ARM7，ARM9到ARM11。后来又有了**Crotex-A8**，**A9**，**A12**。现在，使用新一代ARMv8指令集的64位处理器**Crotex-A53**，A57，和A73也已经登录市场。

从目前的嵌入式处理器的商用使用情况来看，**Crotex-A9**经常被用于互联网电视、智能电视盒子、路由器等许多嵌入式产品。**Crotex-A9**首次进入人们眼前是三星的Galaxy S3智能手机。三星的**Galaxy S3**采用**Exynos 4412**四核处理器，运行安卓4.0操作系统，主频为1.4GHz。在2012年，是三星的旗舰手机。现在，虽然该处理器已经淡出智能手机的市场。但是它依然活跃在嵌入式领域。

本次系统设计使用的是讯为公司开发的4412开发板，4412属于Crotex-A9结构。开发板上集成了处理器所需要运行的其他模组，如网络模块，USB接口，LVDS接口，HDMI接口等等。关于处理器的详细参数见下表3-2。

表3-2

|  |  |
| --- | --- |
|  | Exynos 4412 |
| 公司 | 三星 |
| 工艺 | 32纳米 |
| 架构 | Cortex-A9 |
| 典型主频（Hz） | 四核/1.6G |
| 一级缓存 | 32KB+32KB(per Core) |
| 二级缓存 | 1MB |
| GPU | Mail-400 MP4 |
| 内存宽度 | 双通道64位 |
| Flash存储 | EMMC |
| PIN引脚数 | 786-PIN |
| 价格（美金） | 10 |

本次所使用的开发板配备1GB的内存，4GB的EMMC硬盘存储。同时还有以太网、声卡、JTAG、GPIO、AD转换、USB HOST、USB OTG、TF卡、串口等一系列接口。在硬件的系统配置上，完成本次系统开发应该说是足够了。

**3.2 系统软件平台**

**3.2.1 嵌入式Linux系统**

在早期的嵌入式产品中，并没有可在片上运行的操作系统。所有的程序都是运行在裸机上。整个程序只有一个main函数，通过一个while（1）来死循环执行命令。因为早期的嵌入式处理器运行速度很慢，内存和ROM也都很小，而且也是单核没有协处理器，很难运行较大运算和程序。在这种情况下，嵌入式工程师使用汇编语言对产品进行开发和调试，因为汇编最接近机器语言。但是，使用汇编语言开发周期长，而且不利于理解。

随着技术的不断进步，摩尔定律的验证和半导体工艺越来越好，处理器的主频越来越高，其中有一家科技公司开始崭露头角，当时Inter已经霸占了PC处理器的半壁江山。由于Inter的处理器主要适用于X86结构的计算机。这种结构使用的是复杂指令集，也就是说，使用一条指令就可以完成很多个操作。虽然这个看起来提高了很大的效率，但是经过统计，在用户的普通操作中最常用的也不超过百分之五十。ARM公司决定反其道而行，他们创造了出了精简指令集。精简指令集如其名，每一条指令只实现一个简单的操作。这样整个系统就会简洁许多。

同时，使用哈佛结构，把数据存储器和程序存储器分开，并创建多级流水线。嵌入式处理器比之前的性能提高了十几倍，主频可以达到上百兆赫兹。这样，处理器可以运行一些简单的操作系统，如ucos，rt-os等等。这类操作系统提供多任务的处理方法，可以使得程序的逻辑更加清楚。但是这类系统并没有内存管理和网络等其他内容的管理，核心的内容是进程上下文如何进行切换。在这种处理器上，是无法运行大型的操作系统如windows或Linux的，但是可以进行一些简单的数据运算和处理。典型的处理器如意法半导体的stm32或是美国德州仪器公司的TIVA M4。

为了适应时代的发展，智能设备终端对数据和信息的处理越来越多。ARM公司后来又推出了Crotex A系列，该系列处理器相比于Crotex-M系列处理器的性能又提升了十几倍，处理器的主频也已经超过1GHz。并且也有很多的协处理器和单元，比如FPU和MMU。这时，为了管理这么庞大的硬件系统。那也就需要相应强大的系统软件，WinCE、VxWorks、Linux就成为了嵌入式所常用的系统。但是Linux系统是其中应用最广泛的，这和它的特点有关。

嵌入式Linux系统的特点如下：

* 开放的源代码
* Linux可以定制还裁剪，所需容量小。最小只需2MB。
* 使用Linux系统不需要版权费用。
* Linux经过几十年的发展，其系统内核是十分稳定的。
* Linux有很多的开源社区，各个工具和软件都有良好的支持。

Linux的源代码使用GPL开源协议，所有的Linux源代码都是开源的，这样可以更有利于开发人员对系统进行开发。不仅如此，Linux系统可以进行深度的定制和裁剪。可以针对不同的需求来裁剪系统，把不需要的的模块去除掉，只把需要的模块和其相关的部分留下。比如，Linux的桌面发行版大小接近1个G左右。在经过裁剪和定制后，把不需要的部分去掉，比如GUI和其他驱动，最后可能就只身下几十兆的大小，这十分适合体积较小的嵌入式系统，因为嵌入式系统所完成的工作逻辑是相对固定的。正所谓，麻雀虽小，五脏俱全。

在嵌入式Linux系统流行之前，在嵌入式系统中较为流行VxWorks和WinCE，但是却要付出高额的版权费。而Linux系统则不需要考虑这些问题，这使得越来越多的人们开始考虑时把Linux作为嵌入式的首选操作系统。正是由于这样，linux由许许多多的人们共同使用和开发，共同弥补和修复它的漏洞。在今天，Linux是一个非常稳定的操作系统。

因此，本次设计使用Linux操作系统。首先定制系统内核，把Linux系统编译和安装到开发板上。这时，开发板就如同一台装有Linux系统的PC机。在开发程序时，使用PC机上的Linux系统进行交叉编译。把生成的可运行文件拷贝到开发板上，之后就可以通过命令让开发板运行程序了。

**3.3 服务端硬件系统连接**

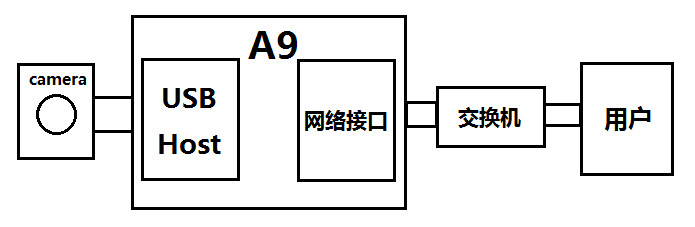
在前两节中，介绍了系统中所需要最基本的硬件摄像头和嵌入式开发板。整个系统的硬件系统连接如下图3-1所示。

图3-1 系统硬件连接

嵌入式开发板具有USB Host接口和以太网网络接口，使用USB Host接口连接USB摄像头，使用网线连接开发板和交换机。那么，用户只要连接到交换机上，就完成了系统的硬件连接。

**3.4 服务端网络连接**

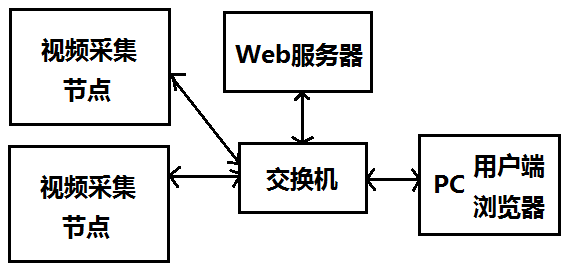
用户端观看视频节点所采集的视频时，首先使用浏览器登录在同一网段中的Web服务器。在网页中添加视频采集节点所需要的IP地址等信息，即可出现所添加节点的视频信息。在网络连接方面用户端可以使用多种方式连接至网络。下图3-2为第一种连接方式。

图3-2

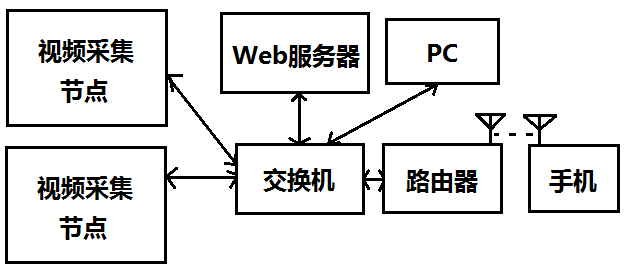
这种方式使用电脑通过网线直接连接交换机。在同一网段中的各个节点通过交换机的交换互相通信。在这种模式下，系统的最大带宽取决于单个节点的网卡带宽和交换机的交换速度。一般来讲，网卡的带宽并不会成为瓶颈。主要的因素在与交换机的速度。

图3-3

上图3-3为网络的第二种连接方式。在上图的方式中，增加了手机的连接方式，通过在交换机后添加路由器。把网络通过WIFI的方式辐射出去，这样，只要连接到该WIFI网络，就可以访问视频节点获取信息。

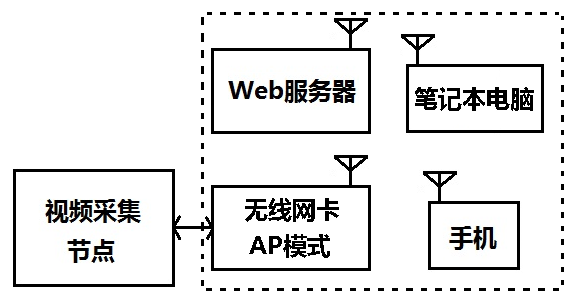
如果不使用交换机，可以在嵌入式采集节点上安装无线网卡，并将其配置为AP模式。那么这时，系统就成为了一个简单的WIFI摄像头，用户使用笔记本或者手机作为sta模式连接至无线网卡。就相当于使用了交换机连接进入了网络，唯一的区别是，一个是有线连接，一个是无线连接。系统的网络带宽取决于无线网卡的所采用的传输协议和传输速度。具体如下图3-4所示。

图3-4

**3.5服务端系统结构组织划分**

本系统结构可以划分为硬件和软件两大部分。通过把系统进行细分，把各个功能进行模块化。知道了哪些模块已经完成，而哪些模块未能完成，根据完成情况做具体的调整。这样更有利于对系统进行开发，同时加快开发的进度。首先，硬件部分的模块如下：

1. 嵌入式核心板：核心板主要为处理器芯片Exynos 4412，以及芯片所需的外面电路和模块。比如缓存和内存单元。
2. 外围电路：外围电路主要为接口电路，有RS-232串口，用于系统的调试。有USB HOST电路，用于连接摄像头。有USB OTG电路，用于烧写系统内核和启动文件。还有以太网接口电路，用于连接网络。
3. 摄像头：摄像头使用USB摄像头，具体内容之前已经介绍。

本次的软件从程序可分为操作系统和应用层程序。而操作系统又分为系统引导和系统内核以及文件系统。具体内容如下所示：

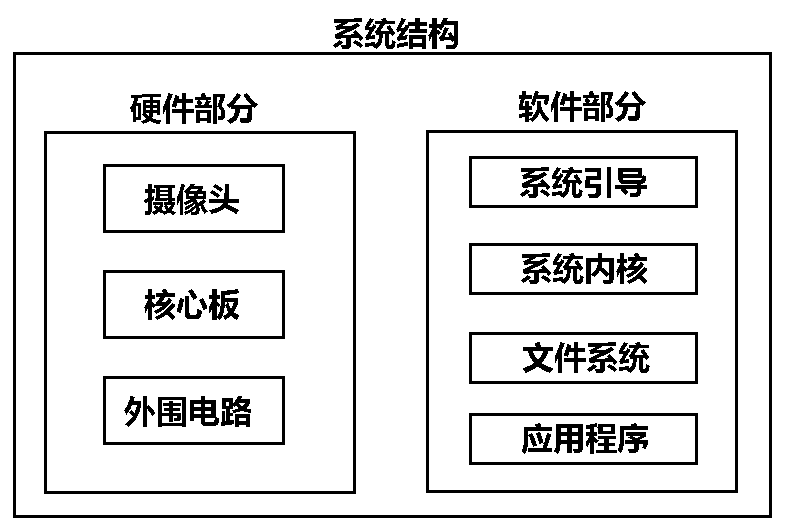
1) 系统引导：嵌入式系统的引导程序为Bootloader。它在操作系统之前运行，检查硬件坏境和引导系统内核。

2) 系统内核：嵌入式系统需要对内核进行定制和裁剪，把不需要的部分去除掉，可以更好的利用储存和内存空间。

3) 文件系统：和系统内核类似，通过软件生成所需要的大部命令即可。使用烧录工具烧录进开发板。

4) 应用层程序：该程序运行于嵌入式Linux环境下，完成系统所需求的逻辑要求。

通过以上的分析，系统的结构组织如下图3-5：

图3-5

**3.6 本章小结**

在本章节中，介绍了系统的硬件结构和软件解结构。通过对需求的分析来对硬件进行选型。同时介绍硬件相关的部分知识，让读者了解为什么选择。在软件方面，介绍了Linux系统的特性以及选择的原因。之后，对整个系统的连接方式进行阐述，说明了各个模块之间如何进行组合连接。最后，对服务端系统结构组织进行的划分和讲解。

**第四章 程序设计和系统实现**

**4.1 开发流程介绍**

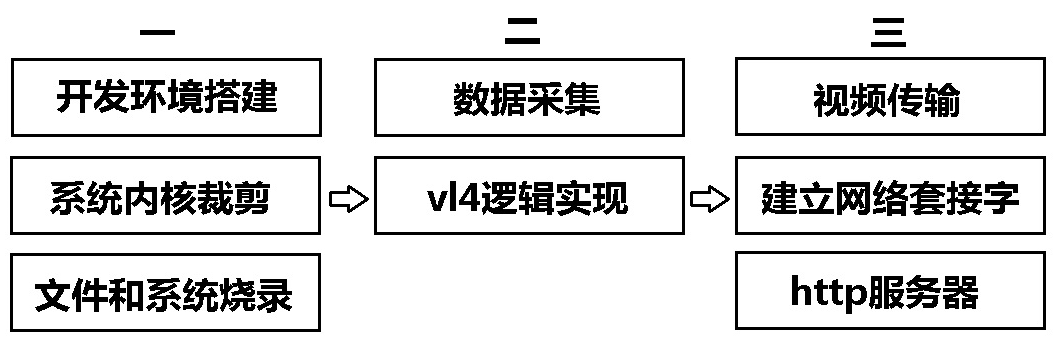
首先，在对系统进行开发之前，需要有一个总体的开发流程。通过对系统的分析。总结出如下图4-1所示的开发流程图。

图4-1

在对系统进行程序开发时，先要对其搭建相关的开发环境。否则，后续的程序功能将不能在开发板上进行实验。在开发板的环境准备好之后，就可以进行应用程序的测试。

接下来，把摄像头和开发板相连接后，就可以使用程序来采集摄像头的数据了。在这里，使用到了Linux中的v4l2 API接口。在程序中，需要考虑的是如何通过API接口来实现系统的逻辑功能。

之后，就是网络节点编程了。需要把视频的每一帧数据传输到用户的网络节点。在Linux系统下使用socket建立网络套接字。并使用网络套接字建立http服务器。把采集到的视频信息放到服务器中，用户连接至服务器后，就可以获取到视频信息。

**4.2 开发环境搭建**

**4.2.1 Linux系统内核配置**

对系统内核进行配置和裁剪，实际上是对Linux源代码的配置和编译。在配置内核时，需要启用相应的设备驱动。本次设计的USB摄像头使用UVC驱动，所以应在配置内核时启用该驱动。

UVC，全称为：USB Video Class，是一种为USB视频捕获设备定义的协议标准，为微软和其他几家设备厂商所联合推出。只要是符合UVC规格的硬件设备在不需要安装任何额外的驱动程序下即可在主机电脑中使用，因为驱动已被包含在内核中。使用UVC技术的包括摄像头、数码相机、类比影像转换器、电视棒及静态影像相机等设备。

在一般情况下，我们通过编写makefile来编译程序。那如何去配置编译十分庞大的Linux系统内核呢，使用make menuconfig可以很好的完成这一任务。它是图像化的界面，可以更加方便简洁的配置内核。

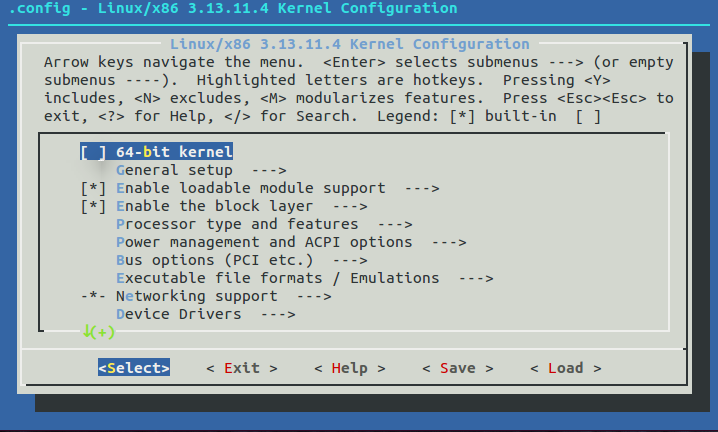
在linux的源码目录中，执行 make menuconfig 命令。出现如下图4-2所示。

图4-2

根据如下步骤进行选择配置。

Device Drivers --->

[\*]Multimedia support --->

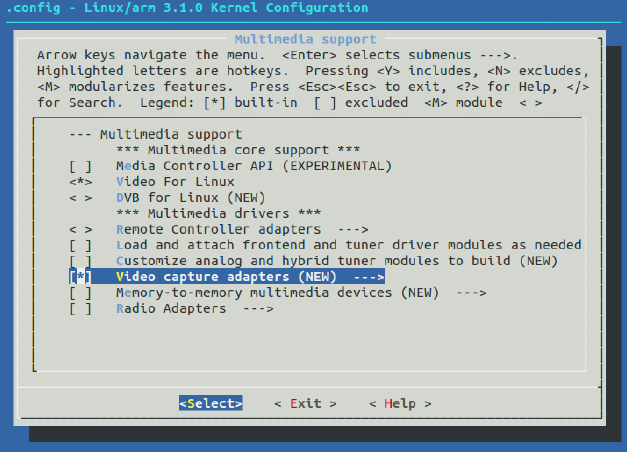
<\*>Video For Linux

图4-3

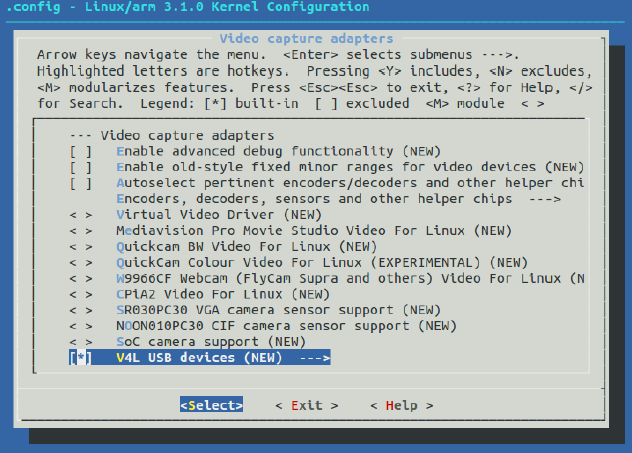
[\*] Video capture adapters --->

图4-4

[\*]V4L USB Devices --->

<\*>USB Video Class(UVC)

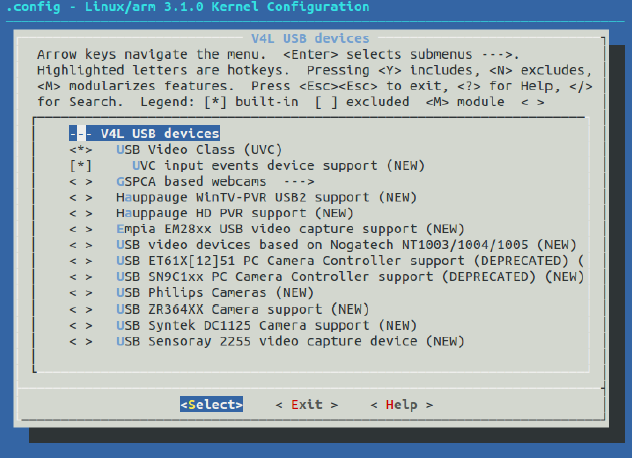
[\*]UVC input event devices support’

图4-5

保存配置之后，就可以吧UVC编译进内核生成img文件，或者也可以把UVC编译成模块，名称为uvcvideo.ko。使用insmod就可以把模块加载进内核。当插入USB摄像头后，内核会打印uvcvideo: Found UVC 1.00 device。同时会在/dev目录下生成设备节点video0。如果已有节点，则数字标号递增，如video1、video2等。

**4.3 采集层控制**

**4.3.1 v4L2视频采集接口**

v4L的全称为Video4Linux，是一个视频截取和设备输出的API，它支持很多的USB摄像头，电视调谐卡和其他的一些设备。V4L2是V4L的第二个版本，提供更好的扩展性和灵活性，支持的硬件设备更多。

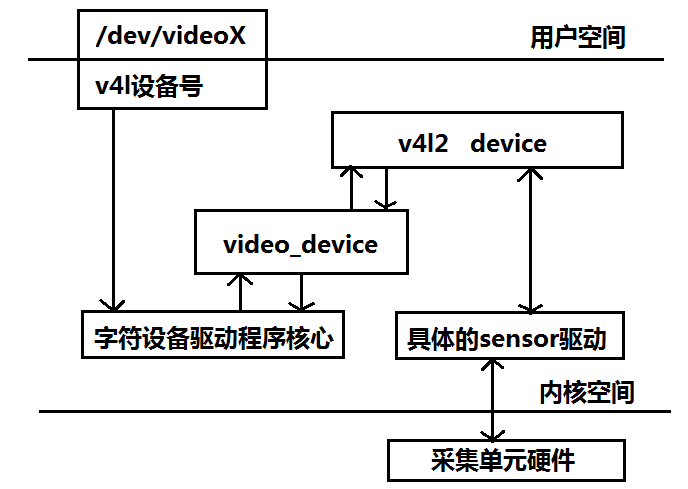
v4l2在Linux系统中的结构图如下：

图4-6

在Linux中，驱动可大致分为块设备，字符设备和网络设备。从v4l2的框图中可以看到，v4l2是一个字符设备。V4l2的大部分功能都是通过设备文件的ioctl导出的。详细的功能如下：

1. 查询设备所支持的功能。
2. 提供视频流一系列操作方法。如枚举设备所支持的所有数据格式和读取一帧的视频数据等。
3. 兼容TV视频标准，可搭配电视调谐卡观看电视节目。
4. 设置输入输出端口。
5. 对设备的一些特定控制，如设置视频的对比度，亮度等等。需要所接入的设备支持。

在本次的设计内容当中，使用的是UVC协议的USB摄像头。在使用v4l2时，所使用的功能主要为上第二点。需要对视频格式进行获取和设置，之后向设备请求视频缓冲区，并初始化视频缓冲区。之后就可以根据时间节点读取视频中的每一帧数据。

**4.3.2 采集流程与实现**

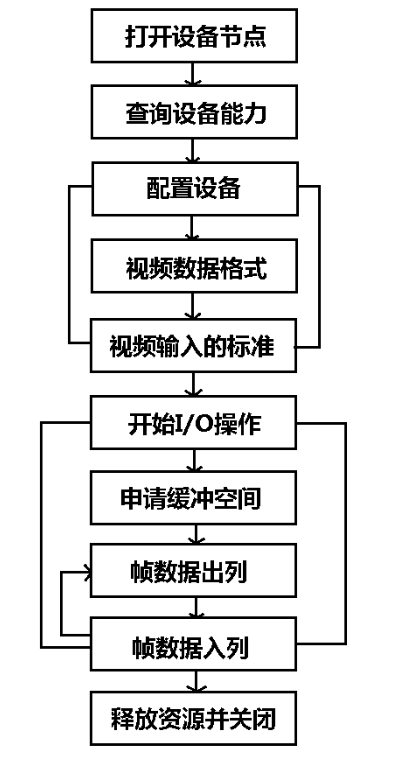
在介绍完v4l2的内容后之后，就是使用v4l2对数据采集进行实现了。使用v4l2采集的流程图如下图4-7。

图4-7

1. 要使用linux中的open函数来打开驱动所创建的设备节点。比如，节点名为video0。那么打开方法如下：

int fd = open（“/dev/video0″, O\_RDWR | O\_NONBLOCK, 0）；

它打开了位于dev目录下的video0节点，返回的文件描述符为fd。通过对fd进行其他的配置操作。

1. 在打开驱动节点之后。我们需要对设备的信息进行检索和设置。也就是说，我们需要知道摄像头能够输出什么格式的内容。其中我们想要的格式在不在其中，如何设定我们需要的格式。在这里，我们通过ioctl来对设备进行配置和操作。

配置时，必须使用到的结构体为v4l2\_format。在结构体中使用宏来配置相关的信息。比如视频输出的格式是YUV还是MJPEG，视频的高度和宽度等等。最后ioctl写入设备。具体操作如下：

struct v4l2\_format fmt;

memset ( &fmt, 0, sizeof(fmt) );

fmt.type = V4L2\_BUF\_TYPE\_VIDEO\_CAPTURE;

fmt.fmt.pix.width = 720;

fmt.fmt.pix.height = 576;

fmt.fmt.pix.pixelformat = V4L2\_PIX\_FMT\_YUYV;

fmt.fmt.pix.field = V4L2\_FIELD\_INTERLACED;

if (ioctl(fd, VIDIOC\_S\_FMT, &fmt) == -1) {

return -1;

}

1. 在配置完成之后，就可以对设备进行读取操作了。首先向设备申请缓冲区，之后从缓冲区队列里依次读取数据。同样，申请缓冲区时使用ioctl函数。原型如下：

int ioctl(int fd, int request, struct v4l2\_requestbuffers \*argp);

使用程序申请一个拥有四个缓冲帧的缓冲区：

struct v4l2\_requestbuffers req;

req.count=4; req.type=V4L2\_BUF\_TYPE\_VIDEO\_CAPTURE;

req.memory=V4L2\_MEMORY\_MMAP;

ioctl(fd,VIDIOC\_REQBUFS,&req);

之后使用内存映射MMAP读取数据，使用的结构体为：

struct buffer

{

void\* start;

unsigned int length;

}\*buffers;

使用的函数为：

#include <sys/mman.h>

void \*mmap(void \*addr, size\_t length, int prot, int flags, int fd, off\_t offset)

其中，addr为映射的起始地址，一般为NULL，让内核自动选择。Length为被映射内存块的长度。Prot标志映射后能否被读写。Flags确定此内存映射能否被其他进程共享。现将四个已经申请到的缓冲帧映射到应用程序，用buffers指针记录。

buffers = (buffer\*)calloc (req.count, sizeof (\*buffers));

if (!buffers) {

// 映射

fprintf (stderr, "Out of memory/n");

exit (EXIT\_FAILURE);

}

for (unsigned int n\_buffers = 0; n\_buffers < req.count; ++n\_buffers)

{

struct v4l2\_buffer buf;

memset(&buf,0,sizeof(buf));

buf.type = V4L2\_BUF\_TYPE\_VIDEO\_CAPTURE;

buf.memory = V4L2\_MEMORY\_MMAP;

buf.index = n\_buffers;

// 查询序号为n\_buffers 的缓冲区，得到其起始物理地址和大小

if (-1 == ioctl (fd, VIDIOC\_QUERYBUF, &buf))

exit(-1);

buffers[n\_buffers].length = buf.length;

// 映射内存

buffers[n\_buffers].start =mmap (NULL,buf.length,PROT\_READ | PROT\_WRITE ,MAP\_SHARED,fd, buf.m.offset);

if (MAP\_FAILED == buffers[n\_buffers].start)

exit(-1);

}

内存映射之后就可以获取帧数据了，具体代码如下：

struct v4l2\_buffer buf; CLEAR (buf);

buf.type = V4L2\_BUF\_TYPE\_VIDEO\_CAPTURE;

buf.memory = V4L2\_MEMORY\_MMAP;

ioctl (fd, VIDIOC\_DQBUF, &buf); // 从缓冲区取出一个缓冲帧

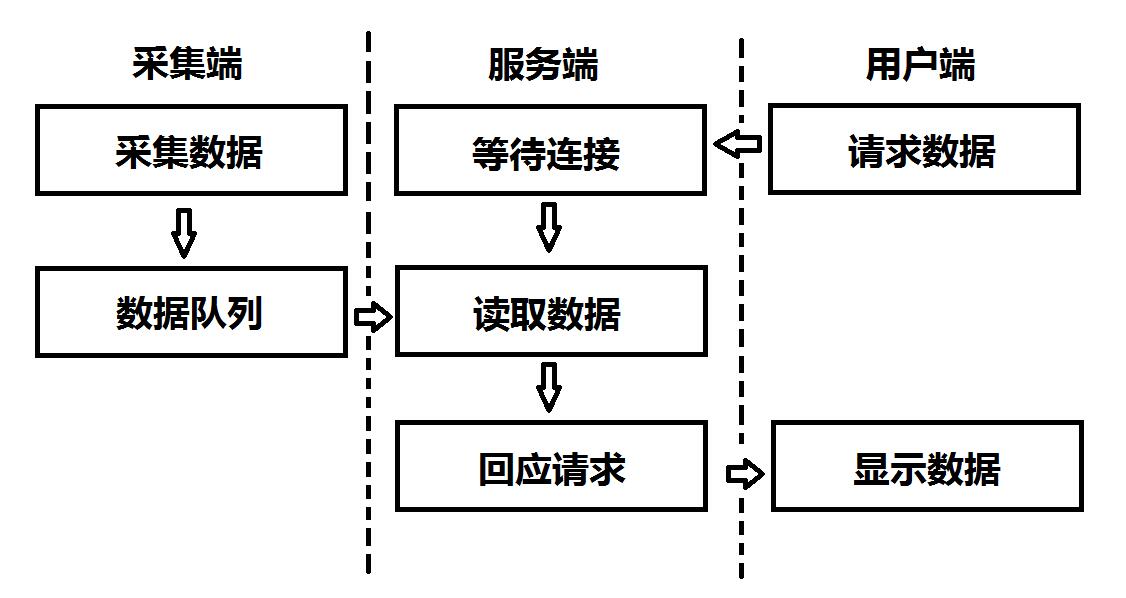
process\_image (buffers[buf.index.]start); //

ioctl (fdVIDIOC\_QBUF&buf); //

1. 在数据读取结束之后，就可以关闭所打开的所有文件，释放所占用的系统资源。整个流程到此结束。

**4.4 视频传输**

**4.4.1 视频传输流程**

****图4-8

如上图4-8所示，在整个视频传输的流程中，嵌入式采集端一直对视频数据进行采集，并将采集到的数据保存到一个循环队列中，当队列中的数据储存满时。就会对最早的那一帧数据进行覆盖操作。

当用户端想要查看视频数据时，那么他会想板上的节点发送一个请求信息。请求查看当前点的数据，板上的服务器收到请求之后，会读取数据队列中的数据，如果队列为空，那么本次的读取失败。如果成功读取，那么服务端就会源源不断的把每一帧的信息传送到用户端，用户端接收到数据之后。通过刷帧的方式，把整个视频显示出来。

**4.4.2 socket套接字**

视频传输过程中需要对服务器搭建，那么服务器底层就需要对网络通信进行基础的操作。在Linux系统下，网络传输的的基础就是使用socket套接字。通过对socket文件的读写，来完成两个程序的通信。

首先，需要创建一个socket文件。具体程序如下：

sock = socket(AF\_INET,SOCK\_STREAM,0);

这一句，使用了socket函数创建一个套接字。在形参中AF\_INET代表了使用32位的ipv4协议和16位的端口号。SOCK\_STREAM代表了使用流来传输socket。最后的0表示自动选择适配的协议类型。值得注意的是，当我们创建一个socket时，描述它的字存放于协议族空间中，但并没有一个具体的地址，接下来，我们需要给它一个具体的地址。

使用bind函数把一个地址族中的特定地址给socket。比如是ipv4地址还是ipv6地址。具体的代码如下：

struct sockaddr addr;

bind(sock, &addr, sizeof(struct sockaddr));

其中，第一个sock表示使用socket创建的空间。addr指向要绑定给socket的协议地址。这个地址根据创建socket时所选择的协议族的不同而不同。最后一个表示结构的长度。

如果该socket作为一个服务器来使用，那么还需要调用listen函数。这时，如果客户端程序调用了connect函数。那么服务器就会收到这个请求。listen函数和connect函数的程序如下：

listen(sock, 1); // listen函数

connect(sock, &addr, sizeof(struct sockaddr)); // connect函数

服务器端依次调用了sock函数、bind函数、listen函数后，就会监听指定的socket地址。那么客户端依次调用socket函数、connect函数之后就会向服务器发送一个连接的请求。在服务器端监听到这个请求之后，再调用accept函数老接收请求，如此一来，连接就建立成功，可以进行接下来的I/O操作了，这个I/O操作和普通文件的操作是类似的。具体的accept函数如下：

New\_sock = accepet(sock, &addr, sizeof(struct sockaddr));

第一个形参sock是创建socket是生成的sock，而accept函数返回的是一个已连接的socket描述字。之后使用I/O操作对这个描述字进行读写就完成的网络的传输。值得注意的是，accept函数是一个阻塞的函数，如果没有多线程或者多进程去执行它的话，一旦没有客户端连接，程序就和一直阻塞在accept函数。

最后，在传输结束后，需要使用close函数来关闭网络连接。具体的操作如下：

close(New\_sock);

**4.4.3 建立http服务器**

在完成基本的socket创建后，接下来就是要制作http服务器了，因为用户端使用浏览器对视频进行观看，所以在网络层协议就选择了http协议。http协议是一个基于TCP/IP的一个网络协议，广泛应用于网页的浏览和一些实时的信令传输。

一般而言，http运行于默认的80端口，但是在程序端也可以自行制定，只要两边的端口号一致，就可以进行通信。通常，由http客户端发送一个请求，创建一个到服务器指定端口的TCP连接。http服务器一旦收到请求，服务器会向客户端返回一个状态，比如“HTTP/1.0 200 OK”发送的内容是基于TCP的。同时返回的还有请求的文件、错误消息等等。

发送的请求消息包括：请求行、请求头、空行、消息体。其中请求行比如“GET /project/logo.jpg HTTP/1.1”表示想要得到project目录下的logo.jpg这个文件。请求头比如“Accept-Language：en”表示使用英文表示，当然，还有其他的比如表示消息文本大小的等等很多请求头，服务器只有收到这些信息，才能做出正确的判断。之后的空行是为了区别请求信息和消息信息。比如我们需要传输的视频帧就在消息体中。

Http有很多的请求方法。在这里，主要介绍Get方法。因为我们视频帧的传输就使用Get方法。Get方法通过在请求头里增加内容即可。不过请求的方法和内容都会显示在浏览器的地址栏中，一般而言，需要对信息做加密处理，比如base64。不要让用户看到明文。但是，我们本次请求的内容不含隐秘信息。所以就不加密了。请求的信息和地址用‘？’间隔。每个信息之间用‘&’来间隔。服务器收到请求信息后根据格式进行解析。之后再根据内容来对客户端进行回复。完成一次的http连接。

本次的http使用了8080端口，当客户端使用浏览器Get请求时，服务端会把视频帧信息源源不断的传输过去。接收端可以使用java script来对这些数据进行处理，从而把一帧一帧的图片信息链接成一个完整的视频流。

**4.5 本章小结**

本章主要介绍了程序开发的具体过程，从底层的视频采集流程，到上层到搭建http进行网络视频传输。在底层中，使用v4l2框架对usb的摄像头进行识别，同时配置输出格式，分辨率等等的信息。之后，对数据进行处理。同时搭建服务器，每当有用户请求来临，就从处理好的数据中读取一帧的数据。接收端通过http接收到信息后，使用java script处理数据。然后就可以把视频内容显示在浏览器中。

**第五章 系统测试与扩展**

**5.1 测试条件和环境**

**5.2 性能测试**

**5.3 系统可扩展性分析**

**5.4 本章小结**