**视频监控系统分析与设计**

刘启明（12330210） 郑雅迪（123302）

刘 悦（12330219） 杨宇灏（12330374）

目录

[第一章问题陈述 3](#_Toc422127396)

[第二章需求分析 3](#_Toc422127397)

[1 功能描述 3](#_Toc422127398)

[1.1 服务器端 3](#_Toc422127399)

[1.2客户端 4](#_Toc422127400)

[2 用例模型 4](#_Toc422127401)

[2.1 用例图 4](#_Toc422127402)

[2.2用例归约 5](#_Toc422127403)

[流程图 6](#_Toc422127404)

[3 补充归约 12](#_Toc422127405)

[3.1 兼容性 12](#_Toc422127406)

[3.2 性能 12](#_Toc422127407)

[3.3 可用性 12](#_Toc422127408)

[3.4 设计约束 12](#_Toc422127409)

[3.5 经济性 12](#_Toc422127410)

[4 术语表 12](#_Toc422127411)

[第三章视频监控与回放系统设计 14](#_Toc422127412)

[1 系统框架 14](#_Toc422127413)

[1.1 服务器端框架 14](#_Toc422127414)

[1.2 客户端框架 16](#_Toc422127415)

[2 系统关键抽象 18](#_Toc422127416)

[2.1 服务器端关键抽象 18](#_Toc422127417)

[2.2 客户端关键抽象 19](#_Toc422127418)

[3 用例分析 19](#_Toc422127419)

[3.1用例功能描述 19](#_Toc422127420)

[3.2用例时序图及协作图 19](#_Toc422127421)

[4 用例类的分析和设计 24](#_Toc422127422)

[5 总类图 25](#_Toc422127423)

[6 分析类到分析机制映射 26](#_Toc422127424)

[7 子系统以及接口设计 27](#_Toc422127425)

[7.1图像采集子系统设计 27](#_Toc422127426)

[7.2 视频编码子系统设计 28](#_Toc422127427)

[7.3 视频解码子系统设计 29](#_Toc422127428)

[7.4 视频传输子系统设计 30](#_Toc422127429)

[8部署图 31](#_Toc422127430)

[9构件图 32](#_Toc422127431)

# 第一章问题陈述

现今，视频监控技术应用越来越广泛。在视频监控系统中，服务器端依赖于搭载摄像头的嵌入式设备，客户端可以通过网络接入在线的监控设备，进行对监控区域的查看。

在本系统中，我们开发了一套嵌入式Linux的视频监控系统，采用c/s（client/server）模型。 Linux服务器端和客户端都为搭载Linux操作系统的Tiny6410开发板。Tiny6410开发板主要由S3C6410芯片，CMOS摄像头，以太网接口，LCD显示屏构成。

在系统的服务器端，我们设置了一个简易的用户界面。用户可以选择录制一定时长的视频，回放查看最近一次录制的视频。用户也可以选择将采集的视频发送到系统的客户端，从而在客户端的LCD显示屏上同步显示。

在系统的客户端，在与服务器端创建会话后，负责接收来自服务器端的视频数据，并且解码显示在LCD屏幕上。客户端没有用户界面，通过串口直接连接到PC端，直接输入相关参数。

# 第二章需求分析

## 1 功能描述

本系统分为服务器端和客户端两部分。这里分别对服务器端和客户端的功能进行描述。

### 1.1 服务器端

服务器端的用户界面有4个功能，分别为录像，回放，开始传输，停止传输。

#### 1.1.1录像

用户在点击“record”按钮后，可以通过开发板上的CMOS摄像头录制一段视频。录像时长可以通过终端输入。因为开发板内存有限，这里限制录制时长不大于60秒。

#### 1.1.2回放

用户在点击“playback”按钮后，LCD屏幕会全屏显示回放最近录制的一段视频。回放结束后会自动回到用户界面。

#### 1.1.3开始传输

在客户端已经创建与服务器端socket会话的前提下，用户通过点击“start”按钮，在终端输入客户端IP地址，端口号，指定本地端口号后，服务器端开始将CMOS摄像头采集到的视频传输给客户端。此时客户端的LCD屏幕上会同步显示服务器端CMOS摄像头采集到的影像。

#### 1.1.4停止传输

在服务器端传输视频到客户端的过程中，用户可以通过点击“stop”按钮停止传输。此时客户端将不再显示服务器端采集的影像。

### 1.2客户端

用户通过终端运行客户端的监控程序，输入服务器端的IP地址，端口号，本地端口号之后即可创建会话。当服务器端有H264视频数据发送到客户端的时候，客户端会在LCD屏幕上全屏显示。

## 2 用例模型

### 2.1 用例图



图2.2.1 视频监控系统用例图

### 2.2用例归约

#### 2.2.1 简要说明

这里将系统的服务器端和客户端的用例归约分开介绍。服务器端的4个功能又可以划分为本地功能（即录像与回放）和网络功能（视频传输）两部分。客户端依赖于服务器端传输的数据，负责接受视频数据并解码显示。

在网络传输中，我们规定视频数据为H.264编码的数据，采用RTP协议传输。

#### 2.2.2 事件流

##### 2.2.2.1 服务器端事件流

用户打开服务器端应用后，用户界面中除了“stop”按钮不可点击外，其他3个按钮都可点击。点击按钮进入对应事件流：

1）点击“record”按钮，进入录像子事件流。

2）点击“playback”按钮，进入回放子事件流。

3）点击“start”按钮，进入开始传输子事件流。

4）点击“stop”按钮，进入停止传输子事件流。

(1) 录像

用户界面的4个按钮都被置为不可点击。用户在终端输入希望录制的时长（小于60秒）后，摄像头会开始录制视频。到了设定时间后会自动停止录制，用户界面的4个按钮被置为可点击。视频录制主要分为视频采集，视频编码，写入文件3个流程。

(2) 回放

用户界面的4个按钮都被置为不可点击，LCD屏幕将全屏显示最近录制的一段录像，回放结束后自动返回用户界面，用户界面的4个按钮被置为可点击。

(3) 开始传输

用户界面中“start”按钮置为可点击，其他3个按钮均被置为不可点击。用户在终端输入客户端的IP地址，端口号，本地端口号之后即可开始传输视频至客户端。视频传输主要分为视频采集，视频编码，分包发送3个流程。

(4) 停止传输

用户点击“stop”按钮后，停止视频传输。此时界面中“stop”按钮再次被置为不可点击，其他3个按钮可点击。

##### 2.2.2.2客户端事件流

用户在终端执行客户端程序后进入实时监控事件流。

（1）用户输入服务器端IP地址和端口号以及本机端口号，然后创建RTP会话，等待从服务器端传来的H.264的视频数据。

（2）一旦接受到了H.264的视频数据，客户端开始解码并把视频图像全屏显示在LCD屏幕上。

#### 2.2.3 特殊要求

1)服务器端的所有事件流均要在摄像头能正常工作的前提下进行，网络状态要良好。

2)服务器端使用的摄像头需要支持YUV420的采集格式。

3)服务器端和客户端需要有JRTPlib的动态链接库。

4)服务器端要同时支持MFC硬解码和硬编码。

#### 2.2.4 前置条件

在使用服务器端网络视频传输功能时，前提是客户端首先要建立与服务器端的网络连接,监听socket是否有数据包到达。

#### 2.2.5 后置条件

在服务器端停止传输后，客户端程序需要在终端退出。

### 流程图



图2.2.2 视频监控系统服务器端活动图



图2.2.3 视频监控系统服务器端录像功能活动图



图2.2.4视频监控系统服务器端回放功能活动图



图2.2.5视频监控系统服务器端视频传输活动图



图2.2.6视频监控系统客户端视实时监控活动图

## 3 补充归约

### 3.1 兼容性

由于是针对嵌入式设备开发的系统，使用到了硬编码和硬解码的技术。所以对硬件平台要求比较严格，这里要求服务器端和客户端包含S3C6410芯片，Linux2.6.38内核。其中服务器端使用ov9650摄像头。

### 3.2 性能

系统服务器端与客户端进行视频传输的时候，视频质量受网络环境影响，可能由于丢包，乱序问题造成画面质量下降。

MFC硬编码和硬解码使得视频编解码速率很快。

### 3.3 可用性

系统服务器端与客户端进行视频传输的时候，视频质量受网络环境影响，可能由于丢包，乱序问题造成画面质量下降。

### 3.4 设计约束

本系统已规定所依赖的服务器端基于S3C6410开发板实现。视频数据传输使用无线网络，RTP/RTCP协议。视频编码格式为H.264。

本系统客户端主要使用C++和C语言实现。

开发工具为QtCreater，使用开源的Qt类库完成用户图形界面的开发。

使用三星的MFC硬编码，硬解码API进行视频数据的编解码。

网络传输部分利用了JRTPlib库封装的函数和类。

### 3.5 经济性

Tiny6410开发板是很便宜的一款基础开饭板，开发平台为Linux，开发工具为开源的Qt Creator（非商业版）。因此成本较低。

## 4 术语表

|  |  |
| --- | --- |
| **视频监控系统** | 由两块搭载Linux操作系统的Tiny6410开发板分别作为服务器端和客户端构成的一个通过网络视频传输实现视频监控的系统。 |
| **视频监控系统服务器端** | 一块Tiny6410开发板，负责视频采集和编码。可以本地录制回放，也可以通过网络传输到客户端进行显示。 |
| **视频监控系统客户端** | 一块Tiny6410开发板，负责从服务器端接受视频数据，解码并显示。 |
| **S3C6410芯片** | S3C6410X是由三星公司开发的基于ARM1176JZF-S核的用于手持、移动等终端设备的通用处理器。 |
| **Qt Creator** | 一个主要由C++语言编写的跨平台的集成开发环境 |
| **YUV420格式** | YUV是被欧洲电视系统所采用的一种颜色编码方法（属于PAL），是PAL和SECAM模拟[彩色电视](http://baike.baidu.com/view/103469.htm)[制式](http://baike.baidu.com/view/735528.htm)采用的[颜色空间](http://baike.baidu.com/view/3427413.htm)。  在YUV420中，一个像素点对应一个Y，一个2X2的小方块对应一个U和V。 |
| **RGB565格式** | RGB565使用16位表示一个像素，这16位中的5位用于R，6位用于G，5位用于B。程序中通常使用一个字（WORD，一个字等于两个字节）来操作一个像素。 |
| **CMOS ov9650摄像头** | 是一款支持YUV420格式图像输出的摄像头。 |
| **LCD屏幕** | Liquid Crystal Display 的简称，这里指Tiny6410开发板的4.3寸[液晶](http://baike.baidu.com/view/6262.htm)显示器。 |
| **RTP协议** | 实时传输协议RTP（Real-time Transport Protocol）是一个[网络传输协议](http://baike.baidu.com/view/16807.htm)，它是由IETF的多媒体传输工作小组1996年在RFC 1889中公布的，后在RFC3550中进行更新。 |
| **JRTPlib** | JRTPLIB 是一个用C++语言实现的RTP库，包括UDP通讯。 |
| **MFC硬编码API** | 三星提供的用于视频硬件编码的一系列接口函数，可以支持MPEG-4 ,H.264,H.263三种格式。 |
| **MFC硬解码API** | 三星提供的用于视频硬件解码的一系列接口函数，可以支持MPEG-4 ,H.264,H.263三种格式。 |
| **H.264** | 一种视频编码格式，是由ITU-T[视频编码](http://baike.baidu.com/view/746807.htm)专家组（VCEG）和ISO/IEC[动态图像专家组](http://baike.baidu.com/view/3589231.htm)（MPEG）联合组成的联合视频组（JVT，Joint Video Team）提出的高度压缩数字[视频编解码器](http://baike.baidu.com/view/89060.htm)标准。 |
| **SPS** | 序列参数集（Sequence parameter set）包括一个图像序列的所有信息，即两个关键帧图像间的所有图像信息 |
| **PPS** | 图像参数集（Picture parameter set）包括一个图像的所有分片的所有相关信息，包括图像类型、序列号等，解码时某些序列号的丢失可用来检验信息包的丢失与否。 |
| **关键帧** | 又称IDR图像，I帧。  在H264协议里定义了三种帧，完整编码的帧叫关键帧。**H.264 引入关键帧是为了解码的重同步，当解码器解码到关键帧时，立即将参考帧队列清空，将已解码的数据全部输出或抛弃，重新查找参数集，开始一个新的序列。** |

表2.4.1 术语表

# 第三章视频监控与回放系统设计

## 1 系统框架

### 1.1 服务器端框架

本系统服务器端采用四层框架，从上到下依次为表示层，控制层，数据层，通信层。

**表示层：**

表示层使用Qt的库完成用户UI界面显示，完成与用户的交互。表示层主要负责获取用户选择的功能，然后在改变界面逻辑。

**控制层：**

控制层由MFC的H.264硬编解码API，S3C6410芯片的post processor构成。负责对摄像头采集的原始YUV420格式的数据的编码，对H.264格式的录制视频的解码，以及显示图像时YUV420格式的图像数据到RGB565格式的转换。在录像与回放功能中对H.264视频文件的打开和关闭。在视频传输的功能中负责网络传输。

**数据层：**

数据层主要由CMOS摄像头采集图像产生YUV420格式的图像数据，H.264编码格式压缩的视频数据，以及封装H.264格式的RTP数据报组成。

**通信层：**

通信层提供与客户端的连接，在视频监控的功能中负责通过RTP协议传输视频数据到客户端。

层次结构之间的依赖关系如图所示：

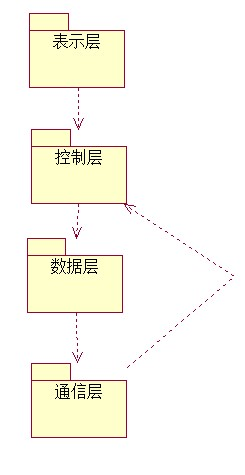


图3.1.1 视频监控系统服务器端层次结构依赖关系图

服务器端各层各模块之间的关系用包描述：



图3.1.2 服务器端模块间关系包

### 1.2 客户端框架

本系统客户端采用四层框架，从上到下依次为表示层，控制层，数据层，通信层。

**表示层：**

表示层是Tiny6410开发板的LCD显示屏显示的RGB565格式的视频图像。

**控制层：**

控制层由MFC的H.264硬解码API，S3C6410芯片的post processor，处理H.264的RTP数据包构成。负责对接收到的H.264格式的视频数据进行重组，解码，格式转换。

**数据层：**

数据层是从服务器端接收到的H.264的RTP数据报

**通信层：**

通信层提供与服务器的连接，负责请求服务器端传输视频数据和接收来自服务器端的H.264的RTP数据报。

层次结构之间的依赖关系如图所示：

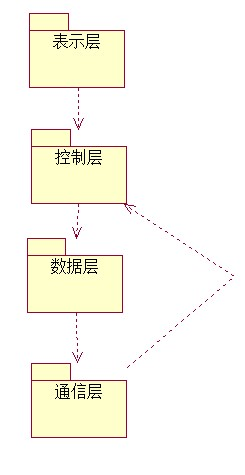


图3.1.3 视频监控系统客户端层次结构依赖关系图

客户端各层各模块之间的关系用包描述：



图3.1.4客户端模块间关系包

## 2 系统关键抽象

系统关键抽象即系统实体类图，系统实体类描述了系统中的类及其相互之间的各种关系，它反映了系统中包含的各种对象的类型以及对象间的各种静态关系。主要描述了系统实体层中各实体类的属性及其相互的关系。是对实体层中各模块的描述。

### 2.1 服务器端关键抽象



图3.2.1 视频监控系统服务器端关键抽象

### 2.2 客户端关键抽象



图3.2.2视频监控系统客户端关键抽象

## 3 用例分析

### 3.1用例功能描述

系统服务器端录像用例描述用户录制一段视频的过程；回放用例描述用户回放查看最近一段录制的视频的过程；视频传输用例描述服务器端采集编码并发送视频数据到客户端的过程。

系统客户端实时监控用例描述从服务器端接收视频数据并且同步解码显示在LCD屏幕上的过程。

### 3.2用例时序图及协作图

用例时序图及协作图反映了两个用例过程，系统中各个类的交互过程。

在服务器端录像时序图中，用户先点击“record”按钮，然后在终端输入需要录制视频的时长。此时用户界面的所有按钮都被置为不可点击。之后会通过开发板上的CMOS摄像头录制视频，到了设定时长后会自动停止录制。此时用户界面的所有按钮再次置为可点击。

在服务器端回放时序图中，用户点击“playback”按钮，此时LCD屏幕将全屏显示最近一段时间录制的视频，用户界面将会隐藏。视频播放完毕后会自动回到用户界面。

在服务器端视频传输时序图中，用户点击“start”按钮，此时用户界面的4个按钮除了“stop”可点击，其他都不可点击。用户在终端输入客户端的socket参数，本地端口号后，服务器端开始使用CMOS采集视频图像并发送给客户端。用户点击“stop”按钮后，服务器端关闭视频传输，此时用户界面除了“stop”按钮外，其余3个按钮均被再次置为可点击。

在客户端实时监控时序图中，用户在终端执行客户端程序，然后输入服务器端的socket参数和本地端口号即可创建RTP会话循环监听是否有H.264的数据包到达。如果有数据包到达，重组数据包中的视频数据并进行解码，全屏显示到LCD屏幕上。



图3.3.1 服务器端录像时序图



图3.3.2 服务器端录像协作图



图3.3.3 服务器端回放时序图



图3.3.4服务器端回放协作图



图3.3.5 服务器端视频传输时序图



图3.3.6 服务器端视频传输协作图



图3.3.7客户端实时监控时序图



图3.3.8客户端实时监控协作图

## 4 用例类的分析和设计

因为系统服务器端采用面向对象设计，客户端采用结构化设计。所以这里只讨论服务器端的用例类。

* 边界类：服务器用户控界面类Widget，运行时相关信息输出类LogMsg。
* 控制类：图像捕获类VideoCapture，图像编码类MFCH264Encoder，图像解码类SsbSipH264Decode，解码视频使用帧提取类FrameExtractor，读取视频文件使用的类FileRead，检测视频播放时间的类perfomance，视频数据网络传输类RTPSend。
* 实体类：H264Frame，主要用于解码时对H.264的数据进行缓存。



图3.4.1 服务器端录像VOPC图



图3.4.2服务器端回放VOPC图



图3.4.3服务器端视频传输VOPC图

## 5 总类图

因为系统服务器端采用面向对象设计，客户端采用结构化设计。所以这里只讨论服务器端的总类图。



图3.5.1 视频监控系统服务器端总类图

## 6 分析类到分析机制映射

|  |  |
| --- | --- |
| 分析类 | 分析机制 |
| Widget | Persistence,Security，Legacy Interface |
| VideoCapture | Distribution |
| MFCH264Encoder | Distribution |
| RTPsend | Distribution |
| SsbSipH264Decode | Distribution |
| FrameExtractor | Distribution |
| FileRead | Distribution |

表3.6.1 分析类到分析机制的映射表

## 7 子系统以及接口设计

本节在本章第一节系统框架和第二节系统关键抽象的基础上，将服务器端划分成四个逻辑上相独立，功能上相依赖的子系统，分别是图像采集子系统、视频编码子系统、视频解码子系统和视频传输子系统。四个子系统的相互关系如图3.7.1所示。



图3.7.1 服务器端子系统相互依赖关系图

### 7.1图像采集子系统设计

图像采集子系统用于使用摄像头采集YUV420格式的原始图像，为视频编码子系统提供原始数据。在此子系统，我们使用VideoCapture类和来实现上述功能。



图3.7.2 图像采集子系统图

### 7.2 视频编码子系统设计

视频编码子系统用于对原始YUV420格式图像进行编码，编码格式为H.264。



图3.7.3 视频编码子系统图

### 7.3 视频解码子系统设计

视频解码子系统用于视频回放时将H.264的视频数据解码为YUV420的图像数据。



图3.7.4 视频解码子系统图

### 7.4 视频传输子系统设计

视频传输子系统用于服务器端与客户端之间的视频传输，使得客户端能够获取H.264编码格式的视频数据。



图3.7.5 RTP数据通信子系统图

## 8部署图



图3.8.1 部署图

## 9构件图



图3.9.1 视频监控系统服务器端构件图