ARM11 平台上 H.264 的视频监控系统实现

Jobs 在三番的 Apple 全球新产品发布会上展示了 iPhone 那刻起，iPhone 彻底的改变整个无线产业，甚至有分析师称，自贝尔发明第一部电话以来，iPhone 是业界期望值最高的一款手机。iPhone 华丽的 UI 界面，友好独特的用户体验，使这款手机立即成为时尚的代名词。然而很少人知道，支持 iPhone 华美舞姿的心脏正是这颗三星的多媒体应用处理器——S3C6400。该处理器是韩国三星电子基于 ARM1176JZF 内核构建的高性能多媒体应用处理器，她不仅具有强大的硬件编解码单元，完善的外设，而且拥有高达 667MHz 的运行频率，保证了 Mac OS 在 iPhone 上演绎绝美的华章。

目前三星在中国国内推出了 S3C6410 处理器——S3C6400 的升级版本。处理器最高支持 800MHz 的主频，且批量的价格仅为 12.5$（USD）。强悍的 S3C6410 处理器也引起数字电视、机顶盒、游戏机以及手机在内的消费及无线产品、军用 PDA、GPS 导航、视频监控等多应用领域的关注。

摘要：介绍基于华恒科技 HHS3C6410 平台，将摄像头采集的数据通过硬件编码后通过网络发送，客户端(Linux)通过网络实时接收并显示的具体实现；主要介绍 S3C6410 的 H264 硬件编码特性和 V4L2 编程。

关键词：S3C6410 H.264 视频监控 V4L2

0 引言

HHS3C6410 是华恒科技推出的一款针对高性能手持设备和通用视频处理应用的低功率,高性能的嵌入式开发板，采用三星 S3C6410 ARM1176JZF-S 处理器，最高主频可达 667MHz；S3C6410 内部集成的多媒体编解码器(MFC)支持 mpeg4/h.263/h.264 的编码与解码，并支持 VC1 解码，性能可以达到全双工 30fps@640x480 同时编解码和半双工 30fps@720x480 或 25fps@720x576 编解码。

H.264/AVC 标准是一套兼顾广播和电信，覆盖从低码率通信到高清晰电视的广域标准，相比以前的标准，具有更高的压缩率，高质量图像，容错功能，并有很强的网络适应性。

1 总体设计

监控系统由监控前端，监控终端，网络三部分组成，监控前端是一个嵌入式 Linux 系统，它通过 S3C6410 的 Camera IF 接收摄像头(SAA7113)采集的数据，并传送给硬件编解码(MFC)模块，并把得到的经过 H264 压缩的数据打包发送到 IP 网络上，监控终端(Linux)通过网络接收数据包，经过解码实时播放。总体框架如图 1:

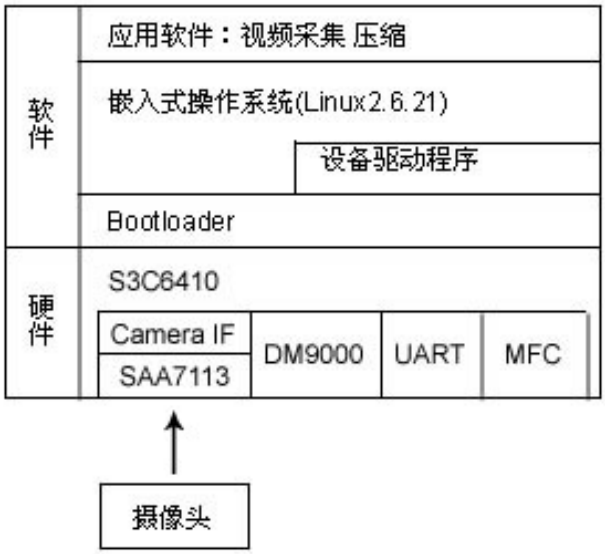

图 1. 总体框架图

整个嵌入式视频采集系统由软件和硬件两部分组成。硬件方面，以 S3C6410 处理器为核心通过 Camera IF 接口接收摄像头数据，经过 MFC 硬件编码后通过 DM9000 发送，UART 作为开发调试接口；软件方面，引导程序和 Linux 内核，设备驱动程序形成基本的嵌入式运行环境，应用层负责视频采集，压缩及传输。

2 软件设计

软件设计包括嵌入式系统构建和应用软件两部分，其中嵌入式系统部分按照华恒科技 S3C6410 用户手册构建，其中包括了 Bootloader，Linux 内核，交叉编译器，驱动等；下面介绍应用软件设计：

监控前端：

监控前端主要包括视频采集模块和视频压缩编码模块。

视频采集模块使用 V4L2 接口收集摄像头数据到缓冲区中，视频压缩模块调用 MFC 驱动把 YUV420 数据压缩编码，同时可以指定编码参数。示例代码如下：

```
cam_fp = open(cam_name, O_RDWR);       //打开 camera 设备
...
mfc_fd = open(MFC_DEV_NAME, O_RDWR|O_NDELAY);       //打开 MFC 设备
...
addr = (char *) mmap(0,BUF_SIZE,PROT_READ |
PROT_WRITE,MAP_SHARED,mfc_fd,0);          //mmap MFC
//设置编码参数
enc_init.in_width          =    out_width;
enc_init.in_height         =    out_height;
enc_init.in_frameRateRes       =    atoi(argv[2]);
enc_init.in_frameRateDiv       =    0;
enc_init.in_bitrate            =    atoi(argv[3]);
```

```
enc_init.in_gopNum          =    atoi(argv[4]);

ioctl(mfc_fd, IOCTL_MFC_H264_ENC_INIT, &enc_init);

frame_size   = (enc_init.in_width * enc_init.in_height * 3) >> 1;
//得到 MFC 输入缓冲区地址
get_buf_addr.in_usr_data = (int)addr;
ioctl(mfc_fd, IOCTL_MFC_GET_FRAM_BUF_ADDR, &get_buf_addr);
in_buf = (char *)get_buf_addr.out_buf_addr;
//得到 MFC 输出缓冲区地址
get_buf_addr.in_usr_data = (int)addr;
ioctl(mfc_fd, IOCTL_MFC_GET_LINE_BUF_ADDR, &get_buf_addr);
out_buf = (char *)get_buf_addr.out_buf_addr;
//V4L2 编程
ioctl(cam_fp, VIDIOC_QUERYCAP, &cap);
//选择输入/输出
ioctl(cam_fp, VIDIOC_S_INPUT, &index);
ioctl(cam_fp, VIDIOC_S_OUTPUT, &index);
//设置格式，注意必须设置输出为 YUV420 格式
ioctl(cam_fp, VIDIOC_G_FBUF, &fb);
fb.capability = cap.capabilities;
fb.fmt.width = out_width;
fb.fmt.height = out_height;
fb.fmt.pixelformat = V4L2_PIX_FMT_YUV420;
ioctl(cam_fp, VIDIOC_S_FBUF, &fb);

on = 1;
ioctl(cam_fp, VIDIOC_OVERLAY, &on);

while(1)
{
    ...
    //接收摄像头数据到 MFC 的输入缓冲区
    read(cam_fp,   in_buf,   (out_width * out_height * 3 / 2));
    //控制 MFC 开始编码
    ioctl(mfc_fd, IOCTL_MFC_H264_ENC_EXE, &enc_exe);
    //把编码后帧的大小及数据发送出去
    send(net_fd, &enc_exe.out_encoded_size, 2, 0);
    send(net_fd, out_buf, enc_exe.out_encoded_size, 0);
    ...
}
```

把摄像头的输出直接设为 MFC 的输入可以节省一次内存操作。

监控终端：

监控终端是一台运行 Linux 的 PC 机，主要负责从网络接收压缩包，使用 SDL 和 Avcodec 解码并显示，示例代码如下：

```
...
While(1)
{
    ...
    //接收压缩包
    recv(sock_fd, &size, 2, MSG_WAITALL);
    recv(sock_fd, &frame_buf[0], size, MSG_WAITALL);
    //初始化 avcodec
    if (!pCodecCtx)
    {
        avcodec_init();
        pCodecCtx = avcodec_alloc_context();
        avcodec_open(pCodecCtx, pCodec);
        //设置参数
        if (pCodec->capabilities & CODEC_CAP_TRUNCATED)
        {
            pCodecCtx->flags |=     CODEC_FLAG_TRUNCATED;
            pCodecCtx->height = 480;
            pCodecCtx->width    = 640;
        }
        pFrame = avcodec_alloc_frame();
        avcodec_decode_video(pCodecCtx, pFrame, &frame_finishd, sps_buf,
sizeof(sps_buf));
        //初始化 SDL
        SDL_Init(SDL_INIT_VIDEO);
        screen = SDL_SetVideoMode (pCodecCtx->width, pCodecCtx->height, 0, 0);
        yuv = SDL_CreateYUVOverlay(pCodecCtx->width, pCodecCtx->height,
SDL_YV12_OVERLAY, screen);

        SDL_WM_SetCaption(buf, "H264/TCP");
        rect.y = rect.x = 0;
        rect.w = pCodecCtx->width;
        rect.h = pCodecCtx->height;

        pict.data[0]        = yuv->pixels[0];
        pict.data[1]        = yuv->pixels[2];
        pict.data[2]        = yuv->pixels[1];

        pict.linesize[0] = yuv->pitches[0];
        pict.linesize[1] = yuv->pitches[2];
        pict.linesize[2] = yuv->pitches[1];
    }
```

```
if (pCodecCtx)
{
    //解码视频流
    avcodec_decode_video(pCodecCtx, pFrame, &frame_finishd, frame_buf, size);
    if (frame_finishd && yuv)
    {
        SDL_LockYUVOverlay(yuv);
        if ((!swsctx && !(swsctx = sws_getContext(pCodecCtx->width,
            pCodecCtx->height, pCodecCtx->pix_fmt,
            pCodecCtx->width,   pCodecCtx->height,
            PIX_FMT_YUV420P, SWS_BICUBIC, NULL, NULL, NULL))) ||
        sws_scale(swsctx, pFrame->data, pFrame->linesize, 0,
            pCodecCtx->height, pict.data, pict.linesize))
                perror("sws\n");

        SDL_UnlockYUVOverlay (yuv);
        //显示
        SDL_DisplayYUVOverlay(yuv, &rect);
    }
}
}
```

以上程序编译需要 libSDL, libavcodec, libswscale。

3 小结

本系统基于 S3C6410 实时采集，编码，传输数据，基本实现了视频监控的功能，测试采集，编码，传输 VGA 的图像可以达到 25fps，使用多线程优化后应该可以更高，VGA 分辨率图像经过 H.264 编码后平均每帧只有 12k 左右，占用网络带宽很少。

参考文献

[1] S3C6410X_UM_Rev1.10_080822.pdf

[2] Video for Linux Two API Specification Revision 0.23.pdf

[3] http://ffmpeg.mplayerhq.hu/