International Journal of Signal Processing, Image Processing and Pattern Recognition
Vol.9, No.1 (2016), pp.409-416
http://dx.doi.org/10.14257/ijsip.2016.9.1.39

Design and Implementation of Mobile Video Surveillance System based on V4L2

LI Hui-jun

Electronic and information engineering department, Shanxi University, Taiyuan 030013, China
Email address:lihuijuns@163.com

Abstract

Embedded technology, mobile streaming media technology, and Linux device driver technology have become a hot topic in the video surveillance field. In this paper, we designed a mobile video monitoring system on the basis of Android platform, accomplishing a camera device driver program, which complies with v4l2 standard on Linux kernel. The system consists of capture terminal, server and client. The capture terminal is used for video acquisition, video coding and transmission. The client connects the capture terminal through the mobile network by mobile telephone, pad and so on. We complete the tests of the capture and system function by building an experiment. It turns out that the monitoring function can be fulfilled by the mobile video surveillance system.

Keyword*: Linux device driver, V4L2, Mobile video surveillance*

1. Introduction

With the rapid development of mobile communication and the evolution of the mobile network, the mobile video surveillance has been used in many fields, such as home security, Emergency rescue and so on [1]. Currently, many telecommunications operator and equipment factories have been regarded mobile video surveillance business as a new growth point, and provided some increment service to users. Based on this situation, this article put forward one kind of design program based on the Android mobile video surveillance system, which has some characteristics, such as emergency, flexibility, mobility and so on. Therefore, there is a wide range of application value [2].

V4L2(Video for Linux 2) is related to programming interface of video in the Linux kernel, V4L2 can support a variety of equipment, such as, provide the video capture interface, video output interface, direct transmission video interface, video interval blanking interface [3].V4l2 is mainly to provide interfaces for the application, So that the application has the ability of finding and operating equipment, and some callback function is used to set the camera resolution, frame rate, video compression format and image parameters, and other functions.

2. System Design and Implementation

The mobile video surveillance system we designed is mainly used for the situations in which a good mobility is required, such as emergency command, scheduling site, emergency management, and monitoring of live. Therefore, the mobile streaming media technology, embedded technology, and mobile communication technology are adopted effective in the system. Besides, we proposed a designing scheme based on android

ISSN: 2005-4254 IJSIP
Copyright ⓒ 2016 SERSC

mobile video surveillance system [4]. The system consists of capture terminal, server and client, which is monitored through the mobile network, as shown in Figure 1.

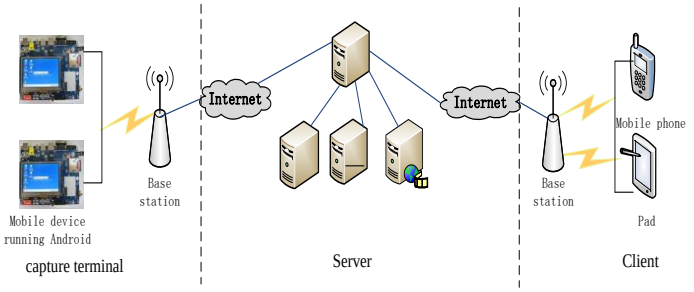

Figure 1. The System Architecture

Samsung s3c6410 is performed as the core processor, there is integrated an ARM1176 nucleus in this chip, and include the LCD controller, camera controller, and imagedma modules. The camera is OV2655, which supports two output format, *i.e.*, RGB and YUV [5]. Figure 2 shows the system software architecture. The system workflow is as follows:

(1) Camera begins to gather, acquired the video stream is placed into the application buffer to YUV442 format.

(2) The application calls the video code module and reads the video stream YUV buffer complying with the H.264 standard compression hardware encoding, then the encoded data is stored in the buffer.

(3) Extracted from the buffer h.264 video for RTP packet, eventually, through the socket communication will send out a data reported.

(4) Received video data in the client design software on the play after coming from the video stream decoding.

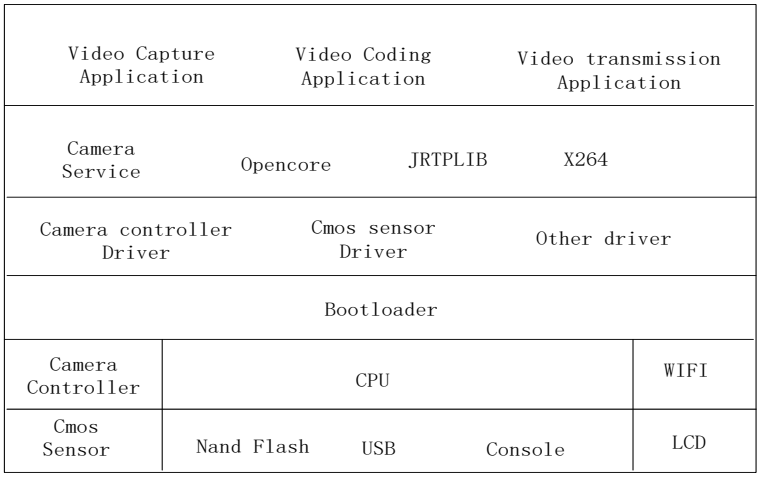

Figure 2. Mobile Video Monitoring System Software Framework

Copyright ⓒ 2016 SERSC

3. Software Design

To complete the video acquisition software design and implementation is the focus of article, the capture terminal is development of based on android platform, video capture module is typical Camera Equipment. So video capture driver module designed follow the V4L2 architecture.

3.1. The Programming Model of V4L2

The code of v4l2 exists under the directory of kernel\drivers\media\video, which is the video device driver in Linux kernel, and provides standard for each video equipment driver programming model and the unified interface for application. There are other codes under this directory, such as v4l2-dev.c, v4l2-common.c, v4l2-device.c, v4l2-mem2mem.c and head files. V4l2-dev.c provides video capture interface, fulfills the registration structure video_device. V4l2-device.c is major to complete the registration of v4l2_device. The standard of V4L2 device driver framework is shown in Figure 3.

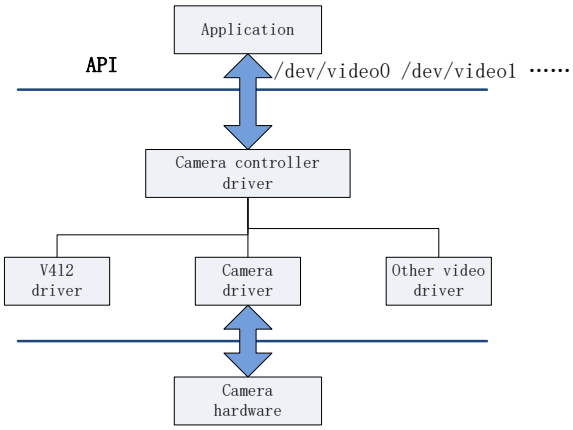

Figure. 3 The standard of V4L2 Device Driver Framework

The header file #include<linux/videodev2.h> is included in V4L2 drivers: which defines the important structure and related parameters within the driver program. Applying the variables is needed before device registration, which shows the video_device structure of the system. Finally, the appropriate function is used to register and unregister the device. Specific code is as follows:

```
// Dynamically allocate a structure variable
struct video_device *vdev=video_device_alloc()
// Registered video device , vdev is Need to register the device struct
video_register_device(struct video_device*vdev,int type,int nr)
// unregister the video device
video_unregister_device(struct video_device*vdev)
```

The v4l2_device and video_device are the two important struct in v4l2, v4l2_device struct is major complement of registering subdevs device in the v4l2 framework, it defines v4l2_file_operation*fops, release, v4l2_ioctl_ops*ioctl_ops function pointer in the video_device struct, which completed device driver registration, release, opening, closing and video equipment, formatting, data processing.

Copyright ⓒ 2016 SERSC

3.2. The Video Data Acquisition Process

The video capture program was designed by adopting the assembly line based on v4l2 [6]. The basic steps include opening the camera device, setting the image format, processing data, starting video capture, closing device device and so on [7]. The process of video capture is shown in Figure 4.

- The application called system function of open () to open the camera device, open the device file of video.

- When the camera as a device file is opened, we can get device information, select the video input and set the video format through calling the function of ioctl. The function is as follows.

```
static int s3c64_camera_set_fmt(struct camera_device*cam, struct v4l2_format*f)
{
    struct v4l2_pix_format*pix = &f->fmt.pix;
    pcdev->pix_width = pix->width; //set the width and height of each frame
    pcdev->pix_height = pix->height;
    switch(pix->pixelformat){ // supporting YUV, RGB video output format
    case  V4L2_PIX_FMT_YUV422:
    case  V4L2_PIX_FMT_YUV420:
    case  V4L2_PIX_FMT_RGB565:
    }
}
```

- Applying the memory on the user space, the command parameter of the VIDIOC_REQBUFS is set in the function of ioctl. Register the buffer to driver.

- The application starts video collection by calling the function of ioctl(cam_fd,VIDIOC_STREAMON,&type).

- The application stops video collection by calling the function of ioctl(cam_fd,VIDIOC_STREAMOFF,&type), and closing the camera of being used.

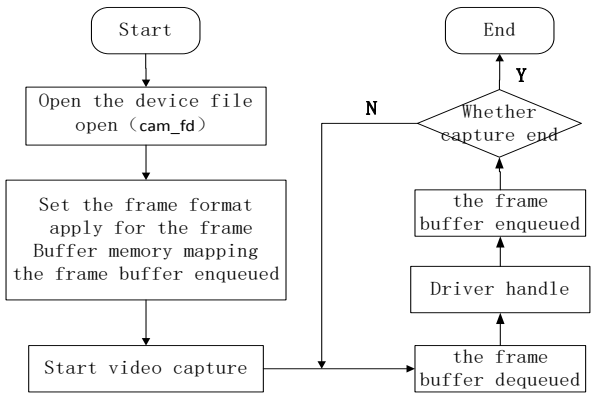

Figure 4. The Process of Video Capture

 Copyright ⓒ 2016 SERSC

3.3. The Implementation of Camera Driver Program

The camera driver framework based on v4l2 is shown in Fig. 5. During the development of video capture driver, it is used to achieve the module of video capture driver with the interface of providing by the v4l2 driver. The important interface and struct used in this paper are defined in some specific files, such as <Linux/videodev2.h>, <media/v4l2-common.h>, <media/V4L2-dev.h> and so on. These head files should be included during the implement of processing.

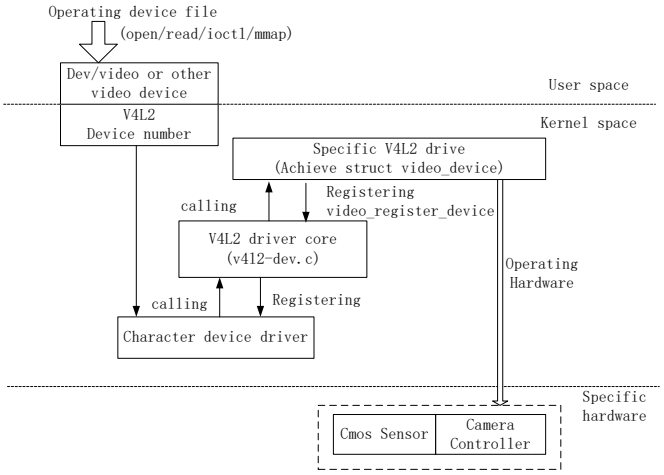

Figure 5. The Camera Driver Framework based on v4l2

The implement of camera driver is based on the subsystem framework of soc-camera, which is divided into the Host and Device at both ends [8]. Device is seen as some sensors which are mounted on i2c bus, the Host and Device connected by bus, soc-camera subsystem framework is shown in Figure 6. The biggest advantage of using soc-camera subsystem framework is that the same sensor could be controlled by many camera host, and could not make larger change to sensor driver, so due attention should be paid to the implement of camera host during program development.

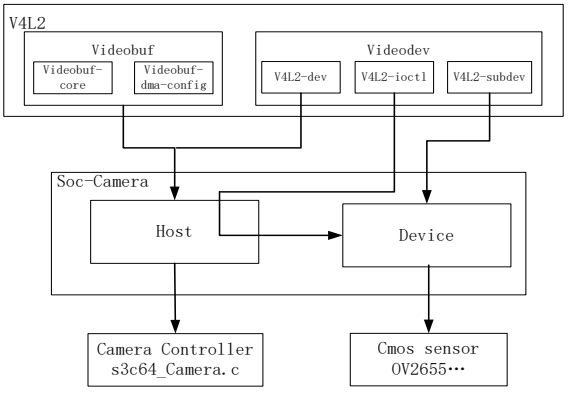

Figure 6. The soc-camera Subsystem Framework

In the process of implementation, first of all, the driver module included host and sensor driver should be initialized. Second, the program should call the function of probe after successful initialization [9]. The function of probe Used to register the host driver and fill the methods in the struct of soc_camera_host_ops. Some methods are

Copyright ⓒ 2016 SERSC

needed to be realized,such as .add, set_fmt, try_fmt. .add is mainly to complete the sensor to access to the host, the method of set_fmt is used to set the resolution format, and try_fmt is used to set the control commands of operation sensor. Some struct and parameters of development are needed to be defined in the struct of s3c64_camera_dev; the important struct and parameters are as follows:

```
struct s3c64_camera_dev{
        struct soc_camera_host soc_host;//the struct is defined in Soc Camera.h of v4l2
        soc_camera_device*icd;//the struct is defined the sensor related information
        int src_format;//the format of sensor data
        int pix_width;//the width of video frame
        int pix_height;//the Height of video frame
}
static int s3c64_camera_init(void)
{       //the host driver initialization, and call the function of probe successful
        return    platform_driver_probe(&s3c64_camera_driver, s3c64_camera_probe);
}
static int s3c64_camera_probe(struct platform_device*pdev)
{    //define the struct pointer of pcdev, The pointer point to Camera_dev device
     struct s3c64_camera_dev *pcdev=&s3c64_camera_hostdev;
     ImageDMA_Init();//initialize ImageDMA, use to move data and change image format
     // Define equipment operation function
     pcdev->soc_host.ops = &s3c64_soc_camera_host_ops;
     pcdev->soc_host.v4l2_dev.dev=&pdev->dev;
     return soc_camera_host_register(&pcdev->soc_host);// Registered soc_host driver
}
static struct soc_camara_host_ops s3c64_soc_camera_host_ops={
     // Fill the ops method
     .add         =    s3c64_camera_add_device,
     .set_fmt     = s3c64_camera_set_fmt,// Set the image format, such as YUV，RGB，JPEG
     .set_ctrl    = s3c64_camera_set_ctrl,
     .init_videobuf = s3c64_camera_init_videobuf,
}
```

4. Conclusion

We proposed a mobile video surveillance based on v4l2, and it is focused on designing video capture software in the capture terminal, and finally fulfilled video capture driver based on v4l2. Finally, we complete test, the test results is proved that the client can display correctly surveillance picture on the screen. Test results as shown in Figure 7. It is the biggest advantage that the capture terminal and client have the ability of strong mobility and accessibility in this paper, it is provided the best solution which used in some occasion where strong demand for mobility, so, it has well reference value.

 Copyright ⓒ 2016 SERSC

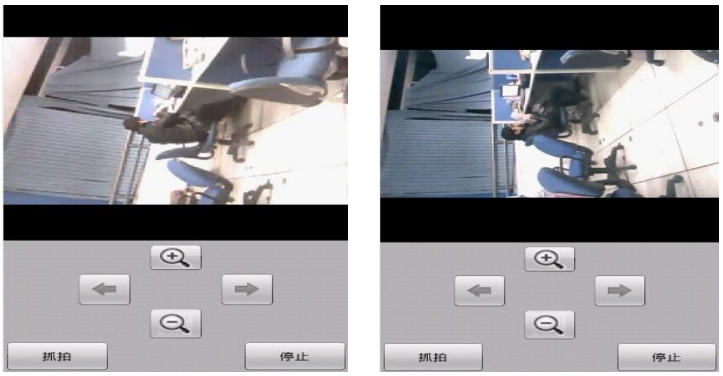

Figure 7. The Client Monitor Screen of Different Resolution

Reference

[1] C. T. Hsu and Y. C. Tsan, "Mosaics of video sequences with moving objects", Signal Processing: Image Communication, vol. 19, no. 1, **(2004)**, pp. 81-98.

[2] R. Cucchiara, A. Prati and R. Vezzani, "Advanced video surveillance with pan tilt zoom cameras", Proc. of the 6th IEEE International Workshop on Visual Surveillance, Graz, Austria, **(1995)**.

[3] L. J. Chen and K. S. Narendra, "Intelligent Control Using Multiple Neural net2 works", International journal of Adaptive Control and Signal Processing, vol. 17, no. 6, **(2003)**, pp. 417-430.

[4] B. S. Chen, C. S. Tsong and H. J. Uang, "Mixed H2/H∞ fuzzy output feedback control design for nonlinear dynamic systems: An LMI approach", IEEE Transaction on Fuzzy Systems, vol. 8, no. 3, **(2003)**, pp. 249-265.

[5] B. Dirks, "Video for Linux Two API Specification: Revision 0.24", Michael H-Schimek, **(2008)**.

[6] "S3C2440A 32-bit CMOS Microcontroller User's Manual", **(2004)**.

[7] G. Hewgil, "RC5 and Java toys", http://www.hewgill.com/re5/index.html, **(2009)**.

[8] I. Foster and C. Kesselman, "The grid: Blueprint for a new computing infrastructure", 2nd. San Mateo, CA: Morgan Kaufmann, **(2009)**.

[9] D. Janakiram, A. Gunnam, N. Suneetha, V. Rajani, K.Vinary and K. Reddy, "Object-oriented wrappers for the Linux kernel", Software: Practice and Experience, vol. 38, no. 13, **(2008)**, pp. 1411-1427.

Copyright ⓒ 2016 SERSC

 Copyright ⓒ 2016 SERSC