# Design and Implementation of the Embedded Based Web Camera System

Qinsheng Du [1,2]
1 College of Computer Science and Technology, Jilin University, Changchun, 130012, China
2 College of Computer Science and Technology, Changchun University, Changchun, 130022, China
Email: duqsh@sina.com

Baohua Jiang, Yonglin Tang
Tourism College, Changchun University, Changchun, 130122, China
Email: { jiangbaohua, tangyonglin}@126.com

Xiongfei Li†
College of Computer Science and Technology, Jilin University, Changchun, 130012, China
Email: lxf@jlu.edu.cn

***Abstract* —This system is composed of the frontal network camera and the remote monitoring client. Firstly, it introduces the whole system structure design and the definition of functions. This embedded web camera takes the powerful ARM9 chip as MPU. The camera captures the video through embedded multitask operating system and the digital video has been compressed by the JPEG algorithm. The general users can view video collected directly by the camera in internet explorer. The authorized users can also control the motion of the camera and configure the parameters of the embedded web camera straightly by Common Gateway Interface.**

***Index Terms* — ARM , Linux , JPEG .**

## I. Introduction

Along with the rapid development of computer technology and network technology, advanced embedded technology and video transmission technology has been effectively combined together [1]. It becomes the development trend of the video monitoring system. In the monitored site, the camera captures the video through the powerful ARM9 chip as MPU and the embedded multi-task operating system. Then the digital video has been compressed by the JPEG algorithm and directly sent to Ethernet. On the other place the general users can view video collected directly by the camera in internet explorer and the users who are authorized can also control the motion of the camera and the parameters configuration straightly by Common Gateway Interface.

Embedded network video monitoring technology effectively improved the video monitoring and transmission system based on PC. The problems existing on PC, such as a large amount of data spending a lot of resources for direct storage and transmission, can be settled.

## II. The Architecture And Main Functions

This system mainly consists of the acquisition terminal and the remote management. The system with the camera installed at the scene obtained from the original video. The video becomes digital signals from analog through the encoder and then is compressed into the JPEG format. The data is converted in streaming format through the streaming media server and is real-time transmitted into the network from the Ethernet interface [2, 3]. The remote client is connected into the network. The monitoring module obtains JPEG data directly from the browser and can watch the scene images. The user can also control the remote camera and set the system configuration.

## III. The Camera Design

### A. The Structure of Hardware

Figure 1 shows the structure of hardware.

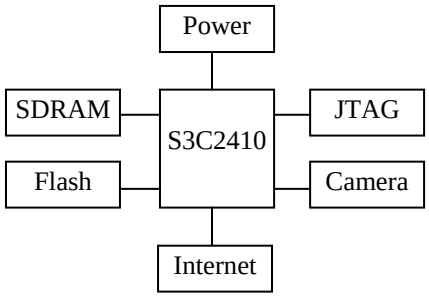


Figure1. The structure of hardware

In this design a 32 bit S3C2410 is chose, the core of which is ARM920T [4].

The boot program and the embedded operating system kernel is stored in 64 MB NAND Flash that is enough big to store application program and important data. SDRAM is used to run the operating system, applications as well as various types of data cache. SDRAM has a total


doi:10.4304/jsw.7.11.2560-2469

capacity of 64MB for the low cost, mass image data and complex image processing.

The microprocessor also integrates abundant resources, such as the LCD controller, USB Host, USB Slave, interrupt control, power control, UART, SPI, SDI/MMC, IIS, GPIO, RTC, TIMER/PWM, ADC. The microprocessor S3C2410 is the center of control and data processing. It controls video acquisition and compression [5].

### *B. System Software*

*1) Kernel Transplant*

In this system the embedded Linux is used as the operating system because it has many advantages, such as open source, without royalty, strong portability, strong support network, many kinds of application software, a tool chain, free, easy to cut [6,7]. After the transplant Linux operating system can run on ARM, POWERPC, M68K and other hardware platforms. In this system the Linux2.6 kernel is adopted and has been successfully transplanted. The method is as follows [8]:

(1) From the Linux official website download the necessary documents, the kernel package Linux-2.6.14.tar.bz2 and cross compiler arm-Linux-gcc-3.4.1.tar.bz2.

(2) Setup a cross compiler arm-linux-gcc-3.4.1 by the Linux command, such as MKDIR, tar, MV and export.

(3) Modify the Makefile files and the related hardware files. The Makefile shows the organization relationship of the kernel modules. The interrelation and dependencies among each module are recorded in the Makefile. Therefore the developers should modify the Makefile file under the Linux-2.6.14 root directory. The main task is to modify the target code type and specify a compiler for compiling the kernel.

(4) Using the command Make compile the kernel to create the kernel image file zImage. Download the zImage to the development board through the corresponding software. Restart the development board and you will see the Linux2.6.14 kernel boot messages. The Linux2.6.14 kernel has been successfully transplanted into the development board.

*2) BootLoader Transplant*

Bootloader is a program before the operating system kernel run. It is similar to the BIOS program in the PC. Through this program the hardware device is initialized. The memory space map function is set up. The hardware and software environment is brought to an appropriate condition and all are ready for the final system kernel call. In this system U-BOOT is adopted as BootLoader for its advantage, such as open source, supporting a variety of embedded operating system kernel and microprocessor series, high stability and reliability, highly flexible function settings. In this system U-BOOT is transplanted onto S3C2410 according to the follow method:

(1) Add the new configuration options in the top Makefile file for the development board.

S3C2410_config: unconfig; @. / mkconfig $ ( config = @: _ ) arm ARM920T s3c2410.

(2) Create the S3C2410 directory to store the relevant code about the development board and add the files, such as flash.c, s3c2410.c, Makefile.

(3) Add a new configuration file for the development board.

(4) Configure the development board with the command: $make s3c2410_config.

(5) Compile U-BOOT. Execute the Make command and get the U-BOOT image after the compilation success.

(6) Add the driver or the function options.

(7) Debug the U-BOOT source code until the U-BOOT is able to start on the development board normally.

*3) Root File System*

The Root File System is the core of the Linux operating system, including the system software and the library, the support structure and the application software to provide users, the area to store the read and write data results. When the Linux starts, install the kernel and initialize the environment, find a file system as the root file system and then load it. In the embedded system the root file system usually includes ROMFS, CRAMFS, RAMFS, JFFS2, EX2 etc.

CRAMFS is a compressed read-only file system. The content in a file system does not require to be decompressed into the memory at one-time. When the system needs access to the data on some location, calculate the CRAMFS position immediately, decompress the data into the memory in real time and get the data in the file system through the memory access. Because CRAMFS has such advantage, it is selected as the root file system in the system. Here we use the busybox tool to construct the embedded Linux root file system CRAMFS.

*4) Peripheral Driver*

For the embedded systems, because there are no general peripheral drivers, thus the peripheral drivers development is an essential part of the embedded system design process [9]. In this system the embedded operating system kernel is Linux 2.6.14 that contains most of the peripheral drivers, such as RS232, USB and LCD. It needs only to be initialized and recompiled. However the kernel does not include the driver of Ethernet chip CS8900 that is an essential peripheral. The users need design their own driver.

(1) Download CS8900.C and CS8900.H by use of the network tools and copy them to DRIVERS / NET directory of the kernel.

(2) Modify the configuration menu and add CS8900 configuration option that can be used when ARCH_SMDK2410 is configured.

(3) The NIC is initialized and the related files, such as smdk2410.h, mach-smdk2410.c, Makefile, are modified and recompiled by the command Make.

Then the CS8900 driver transplant succeeds.

### *C. Video Acquisition Module*

The video acquisition module is the core design of the network camera. It captures the scene through embedded Linux operating system calling V4L (video4linux) and imaging device drivers. V4L is the basis of image process in Linux system. It consists of a set of API supporting

image equipments in the Linux kernel. In cooperation with the proper video card and card drivers V4L can acquire images, AM/FM wireless broadcast, CODEC, channel change. At present V4L is mainly used in the image streaming system and the embedded video system. Its application is widespread, such as remote teaching, remote medical treatment, video conference, video monitor and video telephone. As shown as Figure 2, V4L is a 2-layer structure, the top layer for V4L driver and the lower layer for image device drivers [10].

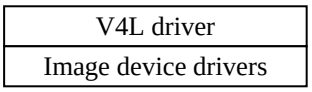


Figure2. 2-layer structures of V4L

In Linux operating system an external device is handled as device files. Therefore the operation becomes the operation of device files. Video files are in /dev/ directory, usually for video0. The camera connected to the video acquisition terminal through USB. V4L APIs are called in a program. The read operation for the device file video0 can realize the data acquisition.

Firstly the V4L header file videodev.h need be included, such as < Linux / videodev.h >. The corresponding API files are in the /usr/SRC/Linux 2.4/ Documentation/Video4Linux/API.html. For the communication with related equipments, some structures, functions and variables are necessary, such as < sys / types.h >, < sys / stat.h >, < sys / ioctl.h >, < sys / mman.h >, < Linux / videodev.h >, < fcntl.h > and < unistd.h >. Figure 3 shows the relationship among the Camera, V4L, the device drivers and the embedded Linux operating system [11].

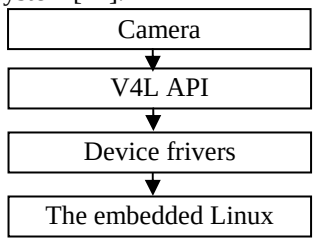


Figure3. The relationship diagram

The important data structure is as follows:

(1) Video_Capability

```
struct video_capability
{ char name[32];
  int maxwidth;      /*Supported width*/
  int minwidth;      /*Supported width*/
  int maxheight;     /* And height*/
  int minheight;     /*And height*/
  int type;
  int channels;      /*Num of channels*/
  int audio;         /*Num of audio devices*/
};
```

This structure consist the important information of the camera.

(2) Video_Window

```
struct video_window
{
  _u32 x, y;                /*Position of windows*/
  _u32 flags;
  _u32 width, height;       /*its size*/
  _u32 chromakey;
  struct video_clip *clips; /*Set only*/
  int clipcount;
}
```

(3) Video_Channel

```
struct video_channel
{ _u32 flags;
  _u32 type;
  _u16 norm;          /*Norm set by channel*/
  int channel;
  int tuners;
  char name[32];}
```

(4) Video_Picture

```
struct video_picture
{
  _u16 brightness;
  _u16 hue;
  _u16 color;
  _u16 contrast;
  _u16 whiteness;  /*Black and white only*/
  _u16 depth;      /*Capture depth*/
  _u16 palette;    /*Palette in use*/
}
```

(5) Video_Audio

```
struct video_audio
{
  int audio;         /*Audio channel*/
  char names[16];
  _u16 bass,treble;
  _u16 mode;
  _u16 volume;     /*If settable*/
  _u16 balance;    /*Strreo balance*/
  _u16 step;       /*Step actual volume uses*/
  _u32 flags;
}
```

(6) Video_Mmap

```
struct video_mmap
{
int height, width;
unsigned int frame;/*Frame(0-n) for double buffer*/
unsigned int format;  /* VIDEO_PALETTE_*   */
}
```

(7) Video_Mbuf

```
struct video_mbuf
{
  int size;              /*Total memory to map*/
  int frames;            /*Frames*/
  int offsets[VIDEO_MAX_FRAM];
}
```

The important functions are as follows:

(1) Open Device Files

```
int v4l_open ( char *dev, v4l_device *vd ) {};
```

It can open image source device files.

(2) Initialization of Pictures

```
int v4l_get_picture ( v4l_device *vd ) {};
```

It can obtain input image information.

(3) Initialization of Channels

```
int v4l_get_channels ( v4l_device *vd ) {};
```

It can obtain each channel of information.

(4) Norm Set of Each Channel

```
int v4l_set_norm ( v4l_device *vd, int norm ) {};
```

It can set the norm of all channels.

(5) Device Address Mapping

```
v4l_mmap_init (v4l_device *vd) {};
```

It returns the address of image data.

(6) Initialization of Mmap Buffer

```
int v4l_grab_init(v4l_device *vd, int width, int height){};
```

(7) Capturing Video Synchronously

```
int v4l_grab_sync ( v4l_device *vd ) {};
```

(8) Capturing Video

```
int device_grab_frame ( ) {}.
```

Here we mainly discussed the function of v4l_get_picture.

```
int v4l_get_picture ( v4l_device *vd )
{
  int ret;
  vd->frame_current = 0;
  ret =  get_grab_frame(vd,vd->current);
  if ( ret<0)
  return ret;
if(ioctl(vd->fd,VIDIOCSYNC,&(vd->frame_current))<0)
 {perror("v4l_grab_sync");
  return ERR_SYNC; }
  vd->frame_using[vd->frame_current]=0;
return 0; };
```

The specific process is as follows:

(1) Open device file;

(2) Inquiry and confirm the equipment performance;

(3) Set the width, the height and the color depth of the captured image;

(4) Establish the memory mapping;

(5) Read the image data;

(6) Close the device.

Figure 4 shows the process flow.

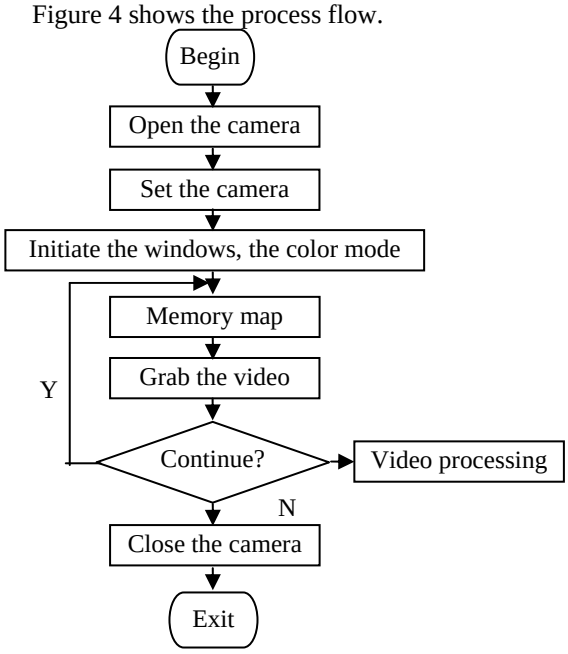


Figure4. The flow of the video acquisition

In the above progress the memory mapping is important. Firstly the display device address is mapped to the system address by the function mmap().The returns address of mmap() is the image data stored address. Every frame image offsets the fixed length. While the images the camera obtains will contain a number of frames. By this way the image data is captured.

The video acquisition is the following process.

After the device initialization, the video acquisition can begin. There are generally two methods. One is that the data is read directly by the read (), another is the memory map by the MMAP (). In the Read () method read the data through the kernel buffer, while in the MMAP () method the device files are mapped into the memory and bypass the kernel buffer. Usually the fastest disk access is slower than the slowest memory access. Therefore the MMAP () methods speed up the I/O access. In addition, the MMAP () system call can share the memory by mapping the same file between processes. The processes can access the files as they access the common memory. The pointers need only to be used without the file manipulation functions called. The access efficiency is much higher. Because the MMAP () method has more advantages, in the programming the memory mapping method is used.

The video acquisition process by the mmap () method is as follows:

(1) Get the frame information of the camera storage buffers by ioctl(vd->fd,VIDIOCGMBUF,&vd->mbuf) and then modify the video settings, such as the vertical and horizontal resolution, the color display format, and the current frame state.

The important statements are as follows.

```
vd->mmap.height=240;
vd->mmap.width=320;
vd->mmap.format=VIDEO_PALETTE_RCB24;
vd->framestat[0]=vd->framestat[1]=0;
```

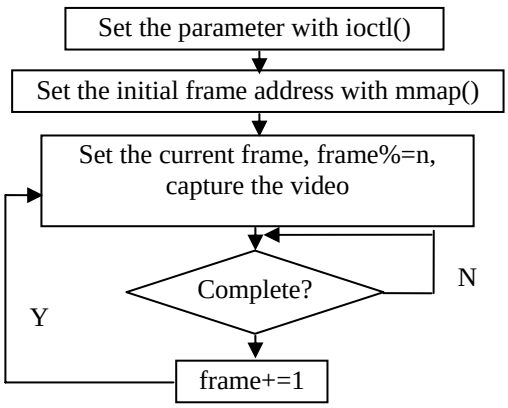


Figure5. The flow of the continuous video acquisition

```
vd->frame=0;
```

(2) Then the device files corresponding with the camera is mapped to the memory area by vd->map = (unsigned char*)mmap(0, vd->mbufsize, PROT_READ|PROT_WRITE, MAP_SHARED, vd->df, 0). This device file contents mapped to the memory area are readable and writable and they can also be shared among different processes. When this function succeeds,

the memory area pointer is returned. In the memory each frame address is determined by vd->map+vd->mbuf.offsets[vd->frame].

Here we introduce each parameter of the mmap function. The first parameter indicates the start address of the shared memory. Here 0 indicates that by the system can assign the address dynamically. The second parameter indicates the byte number mapped to the address space. The third parameter specifies the shared memory access authority. It can be set the value PROT_READ (readable), PROT_WRITE (write) and PROT_EXEC (executable). The fourth parameter specifies the shared memory attributes, generally set MAP_SHARED or MAP_PRIVATE.

(3) Video capture. After the video is mapped to the memory, it can be captured by ioctl(vd->fd, VIDEOCMCAPTURE, &(vd->mmap)). If the call is successful, a frame can be captured. The operation is non-blocking. VDIOCSYNC can judge if the capture ends.

(4) The VDIOCSYNC call wait until one frame capture ends.

```
if ( ioctl (vd->fd, VIDIOCSYNC, &frame)<0)
{
perror("VDIOCSYNC ERROR!");
return -1;
   }
```

If the function succeeds, one frame image has been captured and another frame image will be captured. In the above program the frame indicates the current frame sequence number. After the acquisition the munmap command is called to cancel the mapping: munmap(vd->map,vd->mbufsize).

At most 32 frames are collected once by the video4linux. For the single frame the current frame only need to be set as vd->frame=0, i.e., the first frame. If the function ioctl(vd->fd, VIDEOCMCAPTURE, &(vd->mmap)) succeeds, then the device is activated and a frame image is captured. The acquisition process is non-blocking. The function ioctl(vd->fd, VIDIOCSYNC, &frame) is used to judge if the frame capture ends. The successful return indicates the acquisition end. On the base of the single frame acquisition we can confirm the cycle number n about the frame buffer data after the acquisition completion by the most frame number. Then we capture the continuous frames. In the same way each frame is captured in the loop. Each collected frame is assigned a address by the statement map+vd->mbuf.offsets[vd->frame] and saved as the file format. If this progress continues, we can add the outer loop where frame= 0. Figure 5 shows the continuous video collection.

Through the above operations, video data will be acquired and stored to the camera memory. Video data can be stored as files and also be released to Internet after the compression. The latter process method is used in this design. The video data will be compressed by JPEG and then data flow will be generated and released to the Internet.

### *D. Video Compression Module*

Through the above collection procedure we can obtain the primary image data. According to the image format the video information will be stored into files. Through the network the data is transmitted to the server by the webserver and can be refreshed and displayed. A large amount of collected video data brings a great burden to handle and transmit on network so that the original image data is too large to be convenient for transmission over the network. Therefore it must be compressed. The three kinds of picture formats, such as BMP, JPG and GIF, are supported by the general web browsers. The JPEG compression method is used in this system. Here the high performance ARM9 processor is used to compress the collected data in this design.

JPEG (Joint Photographic Experts Group) is a widely used compression standard that is supported by the general operating system and applications. The file name suffix is ".JPG" or ".JPEG". It is the most commonly used image file format. It is drawn up by a software development union organization. Its main goal is to study with a continuous tone image including gray and color image. JPEG algorithm was identified as international standards of the static digital image compression. It is not only applicable to static image compression, but also suitable for frame image compression of television image sequence. JPEG algorithm is in accordance with full color video standards.

It is a lossy compression method. Its main process includes color model transformation, discrete cosine transform, rearrangement of DCT, quantization and coding of results, etc. By this method the image can be compressed into very little storage space.

JPEG is the most popular image format on the network. The compression technology is very advanced. The redundant image data is removed by the lossy compression method. The method obtains the higher compression ratio and at the same time can get very vivid images. It is very suitable for video transmission on the network. Here the raw images are stored as the JPEG format. The remote monitoring based on web browser is easily implemented with an embedded web server.

The JPEG standard is based on transform coding and integrates DCT and Huffman coding. The compression effect is good. The core content of the JPEG algorithm is the discrete cosine transform ( DCT ) coding method. Figure 6 shows the key steps.

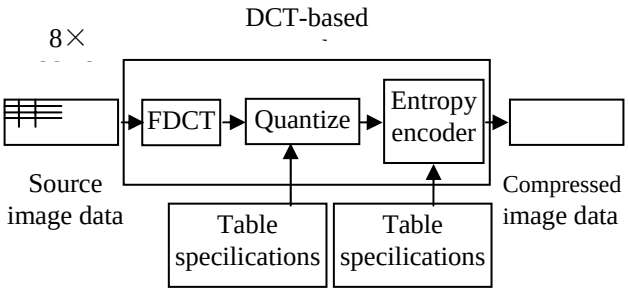


Figure 6. The DCT-based encoder

The encoding process includes three steps, the source image data input, the DCT-based coding and the compression image output. The DCT encoder comprises a forward converter, a Z quantizer and an entropy encoder, in addition to a quantization table and an

entropy coding table ( Huffman table ). Decoding is the inverse process of encoding. After the compressed image data flow arrives at the receiver by the channel, the images are restored and reconstructed by the DCT-based decoder. The DCT-based decoder is the inverse process of the DCT-based encoder. The quantization table and the Huffman table is the same as the sender.

Without the JPEG library in Linux, the jpegsrc.v6b.tar.gz need to be downloaded, decompressed and installed in /usr/src.

```
cd jpeg-6b
./configure
Make
Make install
```

In this way the jpeg function library will work in Linux. When the programs are compiled and linked with –l jpeg, the jpeg library will be linked.

For this a separate function is defined in the program. The function has five parameters. The first parameter is the image file name. The second parameter is the collected original image data. The third parameter defines the image width. The fourth parameter defines the image height. The last parameter is used to set the JPEG image compression quality. The specific codes are as follows.

```
void put_image_jpeg(FILE *out, char *image, int width, int height, int quality)
{  #ifdef HAVE_LIBJPEG
  int y, x, line_width;
  JSAMPROW row_ptr[1];
  struct jpeg_error_mgr jerr;
  struct jpeg_compress_struct cjpeg;
  char *line;
  line=malloc(width *3);
  if(!line)
    return;
  cjpeg.err=jpeg_std_error(&jerr);
  jpeg_create_compress(&cjpeg);
  cjpeg.image_height=height;
  cjpeg.image_width=width;
  cjpeg.input_components=3;
  cjpeg.in_color_space=JCS_RGB;
  jpeg_set_default(&cjpeg);
  jpeg_set_quality(&cjpeg, quality, TRUE);
  cjpeg.dct_method=JDCT_FASTEST;
  jpeg_stdio_dest(&cjpeg, out);
  jpeg_start_compress(&cjpeg,TRUE);
  row_ptr[0]=line;
  line_width=width*3;
  for(y=0;y<height;y++) {
    for(x=0;x<line_width;x+=3){
      line[x]=image[x+2];
      line[x+1]=image[x+1];
      line[x+2]=image[x];}
   jpeg_write_scanlines(&cjpeg, row_ptr, 1);
   image+=line_width;}
  jpeg_finish_compress(&cjpeg);
  jpeg_destroy_compress(&cjpeg);
  free(line);
  #else
  fprintf(stderr, "libjpeg not available – cannot write jpeg!\n");
  #endif}
```

After the original video image data is compressed into the JPEG format, a child process sends the JPEG image data when the client connects. The IP address of both PC and the camera system are set in the same network segment. When the system runs in this way, if the IP address, such as http: / /192.168.0.222: 81/, is entered in the PC browser, the JPEG format image will be displayed. The image may be refreshed and updated.

There is a video server application program in the Linux system on the target board. This program can support to play real time video files. Webcam is a common video application. From the network we can download the Linux version webcam_server, webcam_server-0.50.tar.gz, which is based on the GNU framework, completely free, an open source program [12]. After decompression the command . / configure is executed and then the Makefile file is created. There is a variable CC in the Makefile under the current directory and the SRC directory. The variable CC should be set as /usr/local/arm/3.4.3/bin/arm-linux-gcc. After the command make is executed, the webcam_server executable file is created. The application is loaded into the development board and can be used.

## IV. REMOTE MONITOR CLIENT

### *A. Web Server*

Resources are usually limited in embedded systems. There are lightweight Web Servers, such as HTTPD, THTTPD, boa [13]. Boa Web Server is used in this design. This server is open-source and can support CGI. Its main processes are as follows:

(1) Download the latest package from www.boa.org and unzipped into relevant directory;

(2) Set the default SERVER_ROOT path in the top of the defines.h file in the boa/src directory;

(3) Choose cross-compiling tools. In boa directory make boa configuration by . / configure -- host = i686 – pc - Linux – gnu -- target = arm – Linux;

(4) Generate executable file boa in src/ directory after the execution of make;

(5) Configuration of boa.conf files. Here set the socket of boa, Server root directory, log files, html, CGI, the attribute temp directory, etc.

### *B. Common Gateway Interface*

CGI ( Common Gateway Interface ) is the interaction standard between external applications and WWW server. According to the CGI standard the external program can handle the input data from the client browser and the interaction between the client and the server, realize the dynamic web technology. In this system when the user sends control command to the network camera through the browser, the server starts up the CGI module and then the CGI module will transmit the command. At last the camera will execute the action.

### C. Internet Explore Browser

In this system the main function of the web page is to show the remote dynamic video. The standard html pages can only display the words and pictures, so real time video can't be watched through the window added in the standard html pages. To solve the problem, the method is to embed the real time video monitor software into html pages.

### D. Play Video

The Linux operating system is used in the remote client. Java applet supports to play the video stream. The JDK environment should be installed in Linux.

(1) The JDK Linux 1.6version file, jdk-6u11-linux-i586-rpm.bin, may be downloaded from Sun website.

(2) The file need be added the execution permission. The command is as follows:

chmod a+x jdk-6u11-linux-i586-rpm.bin

(3) The file is decompressed and automatically installed. The command is as follows:

./ jdk-6u11-linux-i586-rpm.bin

The Java environment will be created in /usr/java.

(4) The related environment variable is set. The command is as follows:

export PATH=/usr/java/jdk1.6.0_11/bin:$PATH

(5) In order to allow the browser to find the Java Plug-in, the environment variable NPX_PLUGIN_PATH points to the javaplugin.so directory. The command is as follows:

export NPX_PLUGIN_PATH=/usr/java/ jdk1.6.0_11/jre/plugin/i386/ns7

(6) Then the video streams can be played by the Java applet program. The commands are as follows:

java –classpath applet.jar:/usr/java/ jdk1.6.0_11/lib /tool.jar

WebCam 192.168.0.123 8888

Then the host collects the video stream from the USB camera. The vivid images are displayed on the screen. The vivid images are displayed on the screen, as shown in figure 7. The width is 320 pixels and the height is 240 pixels. The speed is 8 frames per second. The images are very clear and can be refreshed dynamically.

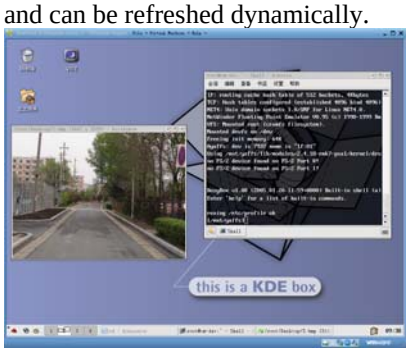


Figure7. The vivid images on the screen

## V. CONCLUSION

This paper presents an embedded network camera design based on ARM S3C2410 and the Linux system. This camera system can accomplish the acquisition, the compression and the display of video data. This system is a complete solution integrating with Web Server and CGI. Because the system adopts with the high-performance embedded processors to complete the main control, compression and web processing work, it is simple, inexpensive, stable and widespread.

## ACKNOWLEDGMENT

This work is partially supported by the science and technology research project of Jilin Provincial Education Department under Grant No. 2011223, 2011339 and GH11067, the Science and Technology Development Program of Jilin under Grant No. 2010036, 20102109, 201101088, 201105046 and 201201138.

## REFERENCES

[1] Tunhua Wu, Qinqin Shen, Changle Zhou and Ping Wang. Design and Implementation of a General Purpose 2D CAD System. Journal of Computer, Vol.7, No.3, pp.666-671, March 2012.
[2] Lain Bate, Steve Liu. Real-time Embedded System. Computing & Control Engineering Journal, 2002, 8.
[3] Bruce Powel Douglass. Real-Time Design Patterns. Embedded Systems Conference Papers, San Francisco, 2001.
[4] SAMSUNG's Digital World:http://www.samsung.com/
[5] David Seal. ARM Architecture Reference Manual. Addison-Wesley. Second Edition. Published 2001.
[6] Judy Democker. Three reasons why Linux will trounce the embedded market. IBM developer Works:Linux, 2001.
[7] Scott Maxwell. Linux Core Kernel Commentary. Coriolis, pp.112-113, 2000.
[8] Robert Love. Linux Kernel Development. Pearson Education, pp.263-265,2004.
[9] Alessandro Rubini, Jonathan Corbet. Linux Device Drivers, Third Edition. O'Reilly, 2006.
[10] Michael H Schimek, Bill Dirks, Hans Verkuil. Video for Linux Two API Specification Draft O.13. April 2006.
[11] Alan Cox. Video4Linux Programming. 2002.
[12] WebCam_Sever for Linux. http://freshmeat.net/projects/webcam_server/
[13] Jeremy Bentham. TCP/IP Learn Web Servers for Embedded System. China Machines Press, May, 2003.

**Qinsheng, Du** Jilin Province, China. Birthdate: July,1978. He is a lecturer of College of Computer Science and Technology, Changchun University. He is currently pursing his Ph.D at College of Computer Science and Technology, Jilin University. He is mainly engaged in the research of embedded system.

**Baohua, Jiang** Jilin Province, China. Birthdate: February,1977. He is lecturer of Tourism College, Changchun University. He is a graduate student at College of Computer Science and Technology, Jilin University. He is mainly engaged in the research of embedded system.

**Yonglin, Tang** Jilin Province, China. Birthdate: May, 1957. He is a professor of Tourism College, Changchun University. He is mainly engaged in intelligent control and embedded system.

**Xiongfei, Li** Jilin Province, China. Birthdate: January, 1963. He is a professor of College of Computer Science and Technology, Jilin University. He is mainly engaged in embedded database.