

Embedded System

#Term Project - Web streaming cctv solution

날짜 : 12 22 2019

조 : 9조

제출 : 201410935 조현종

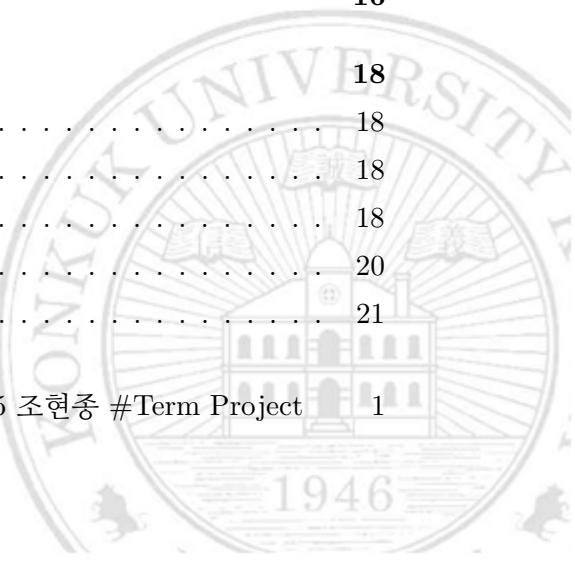
201410524 김기홍

201412235 김지웅



Contents

1 Abstract	3
2 개요	3
2.1 구현 기능	3
2.2 구현 과정	4
2.2.1 Motion jpeg	4
2.2.2 Socket programing	4
2.2.3 Libjpeg 8a	4
2.2.4 디바이스 드라이버를 통한 기능제어	5
3 code	5
3.1 code hierarchy	5
3.2 main.c	5
3.3 text.h	7
3.4 text.c	7
3.5 camera.h	8
3.6 camera.c	9
3.7 jpegComp.h	10
3.8 jpegComp.c	11
3.9 mjpegServer.h	11
3.10 mjpegServer.c	12
3.11 build	13
3.12 클라이언트 뷰어	14
3.13 build	14
4 cross compile	16
4.1 libjpeg 8a	16
4.2 apache httpd	16
5 result	16
6 discussion	18
6.1 device control	18
6.1.1 state machine by dipsw	18
6.1.2 textlcd	18
6.2 camera to jpeg	20
6.3 클라이언트 프로그램	21



6.4 jpeg compress	22
6.5 mjpeg server	22
7 Conclusion	23



1 Abstract

저희는 EMPOSIII 임베디드 보드를 이용하여 CCTV 방범 솔루션을 제작하였습니다. 최근 1인 가정이 급속도로 늘어나면서 많은 범죄가 일어나고 있습니다. 실제로 제 친구도 자취를 하는동안 도둑이 들어서 곤란했던적이 있었습니다. 또 아이를 키우거나, 반려동물과 함께하는 사람들은, 아이나 반려동물이 집에서 사고가 나지 않을까 항상 걱정하게 됩니다. 이런 경우 cctv를 설치하는게 하나의 방안이 될수있지만 시중에서 판매하는 CCTV등을 사서 사용하기엔 설치도 매우 복잡하고 많은 비용이 듭니다. 그래서 저희는 하나의 임베디드 보드를 이용해 실시간으로 영상을 찍어 인터넷으로 언제 어디서든 볼 수 있는, 누구나 사용할 수 있는 cctv를 목표 하였으며, 또한 클라이언트 프로그램을 제작하여 컴퓨터를 통해 얼굴 감지, 움직임감지, 동영상 녹화등의 기능이 가능하게 하였습니다.

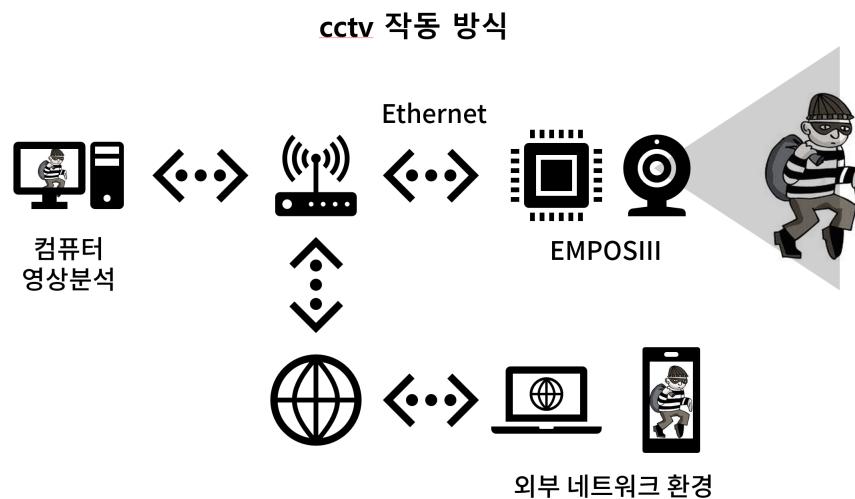


Figure 1: cctv 개요

2 개요

2.1 구현 기능

- 웹을 이용한 실시간 cctv 영상 송출 기능
- 외부 네트워크에서 웹 브라우저를 통한 실시간 감시
- EMPOSIII 보드에 있는 lcd와 text lcd를 이용한 현 동작 상태 확인
- 클라이언트 프로그램을 통한 동영상 녹화 기능, 얼굴 감지 기능, 움직임 감지기능
- 현 상태 알림, 카메라 송출 정지기능

2.2 구현 과정

먼저, 카메라 화상을 웹상으로 보내기위해 가장 먼저 정해야 할 것은 영상 포맷입니다. 여러가지 영상 포맷이 있지만, 성능이 한정적인 10년전 임베디드 보드에서 실제로 사용가능한 영상 포맷은 많지 않았으며, cctv 특성상 실시간 프레임이 보장 가능해야 했습니다. 그래서 저희는 영상 포맷들 중 하드웨어 스펙과 구현의 용이성, 딜레이 등을 고려하여 Motion jpeg(mjpeg)를 사용하였습니다. 영상 포맷을 정한뒤, 이를 구현하기위한 jpeg compression은 오픈소스 프로젝트인 libjpeg 라이브러리를 cross compile 하여 사용하기로 하였고, 컨테이너는 http를 통해 직접 socket programing을 이용해 구현 한뒤 apache httpd를 이용해 웹 뷰어를 구현하였습니다.

2.2.1 Motion jpeg

Motion jpeg (mjpeg)란, 각각의 프레임을 jpeg로 압축하여 이를 하나의 컨테이너로 묶은 데이터입니다. 일반적인 dvix, h264등과 다르게 각 프레임 별로 압축할뿐 시간축으로는 압축을 고려하지 않기 때문에 데이터 사이즈가 더 큰 반면 상대적으로 저사양으로 동작시킬 수 있습니다. 실시간으로 mjpeg를 웹에 스트리밍 하기위해서, c로 소켓 프로그램을 만들어 MIME 포맷에 맞추어 jpeg 프레임을 순차적으로 send 하였습니다.

2.2.2 Socket programing

표준 웹브라우저 어디서든 볼수 있게 하는게 목표였기 때문에 MIME를 준수하여 Mjpeg 스트리밍 서버를 구현하였습니다. 이때, MIME는 다음과 같은 규격을 따라야 합니다.

```

1 HTTP/1.0 200 OK
2 Server: EMPOSI
3 Connection: close
4 Content-Type: multipart/x-mixed-replace; boundary=myboundary
5
6 --myboundary
7 Content-Type: image/jpeg
8 Content-Length: <image size>
9
10 <frame1 jpeg Data>
11 --myboundary
12 Content-Type: image/jpeg
13 Content-Length: <image size>
14
15 <frame2 jpeg Data>
16 --myboundary
17 //continue

```

줄바꿈은 \r\n으로 구현 됩니다.

2.2.3 Libjpeg 8a

수업시간에 사용했던 EMPOSI를 opencv 1.x 를 이용하면 카메라로부터 jpeg 데이터를 만들 수 있지만, 이 경우 NAND flash에 파일로 저장되어 메모리에 버퍼 형태로 저장하는 것 보다 필연적인 속도 저연이 발생하게 됩니다. 실시간 cctv 영상 처리를 위해서는 낸드플래시를 거치지 않고, 메모리 버퍼에 jpeg 데이터를 상주시키고 소켓으로 뿐려주어야 합니다. 이를 위해 메모리버퍼에 이

미지 저장이 지원되는 libjpeg 8a 라이브러리를 직접 크로스 컴파일 하여 emposIII에 포팅하였고, 응용프로그램은 동적 라이브러리를 사용해 cross compile하여 보드에서 구동시켰습니다.

2.2.4 디바이스 드라이버를 통한 기능제어

cctv로써, 동작을 제어하고 현 상태를 표시하는 부분이 필요했습니다. 사용자는 사생활을 위해 카메라를 잠시 끈다면가, 지금 cctv가 예리없이 잘 돌고 있는지 등을 확인하여야 합니다. 이를 위해 lcd 와 textlcd에 현 상태를 표시하도록 하였으며, dip switch를 이용해 카메라 동작을 잠시 멈추거나, 아예 프로그램을 끌 수 있게 만들었습니다.

3 code

3.1 code hierarchy

Main.c

- text.c
- camera.c
- jpegComp.c
- mjpegServer.c

3.2 main.c

```

1 #include "camera.h"
2 #include "mjpegServer.h"
3 #include "jpegComp.h"
4 #include <stdio.h>
5 #include <string.h>
6 #include <stdlib.h>
7 #include <unistd.h>
8 #include <sys/mman.h>
9 #include <fcntl.h>
10 #include <signal.h>
11 #include <fcntl.h> /* for O_RDONLY */
12 #include <linux/fb.h> /* for fb_var_screeninfo, FBIOGET_VSCREENINFO*/
13 #include <termios.h>
14 #include <ctype.h>
15 #include <assert.h>
16 #include <limits.h>
17 #include <time.h>
18 #include <float.h>
19 #include <math.h>
20 #include <sys/ioctl.h>
21 #include "text.h"
22 #define DEBUG
23 #define FBDEV_FILE "/dev/fb0" //cam
24
25 int main()
26 {
27     int dev;
28     int f_loop = 1;
29     unsigned short vkey_org[2], vkey_new[2], vkey_run[2];
30     unsigned char * imageBuffer = (unsigned char *)malloc(IMAGE_WIDTH*IMAGE_HEIGHT*3);
31     unsigned char * jpegBuffer[1];
32     unsigned char * camCloseImg[1];
33     unsigned long jpegSize;
34     unsigned long camCloseImgSize;
35     int err;
```



```

36
37
38     FILE * fp = fopen("camClose.jpeg","rb");
39     fseek(fp, 0, SEEK_END);
40     camCloseImgSize =ftell(fp);
41     fseek(fp, 0, SEEK_SET);
42
43     camCloseImg[0] = (char *)malloc((camCloseImgSize+1) * sizeof(char));
44     fread(camCloseImg[0],camCloseImgSize,1,fp);
45     fclose(fp);
46
47 //open dip sw
48 if ((dev = open("/dev/dipsw", O_RDONLY)) < 0)
49 {
50     perror("DIPSW open fail\n");
51     return -1;
52 }
53
54 cameraInit();
55 initSocket(8080);
56 createConnection();
57 textLCDInit();
58 display1();
59 printf("Set dipsw 1 to run cam\nSet dipsw 9 to quit\n");
60 display_set();
61 while (f_loop) //dip_sw 확인
62 { //웹서버는 계속 둘게끔
63     read(dev, &vkey_new[0], 4);
64     printf("\t> (SW1) : %02X\n", vkey_new[0]);
65     if (vkey_new[0] == 1 || vkey_new[0] == 9){
66         display2();
67         display_init();
68         camera(imageBuffer);
69
70 #ifdef DEBUG
71     printf("camera");
72 #endif //Debug
73     jpegCompress(imageBuffer, jpegBuffer, &jpegSize);
74
75 #ifdef DEBUG
76     printf("compress");
77 #endif //Debug
78     err = sendFrame(*jpegBuffer, jpegSize);
79 #ifdef DEBUG
80     printf("sendFrame");
81 #endif //Debug
82     if (err == -1){
83         closeConnection();
84         createConnection();
85     }
86 }else{
87     sendFrame(*camCloseImg, camCloseImgSize);
88     sleep(1);
89     display1();
90     display_set();
91 }
92
93 if (vkey_new[0] == 9){
94     f_loop = 0;
95     cameraClean();
96     display1();
97 }
98
99 }
100
101 }
102 close(dev); //dipsw파일을닫는다.
103 closeConnection();
104 closeSocket();
105 free(imageBuffer);
106 free(jpegBuffer[0]);
107 //free(jpegBuffer);
108 return 0;

```



109 }

3.3 text.h

```

1 void display1();
2 void display2();
3 int textlcdInit();
```

3.4 text.c

```

1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <unistd.h>
4 #include <sys/types.h>
5 #include <sys/stat.h>
6 #include <fcntl.h>
7 #include <string.h>
8 #include <sys/ioctl.h>
9 #include "text.h"
10
11 #define TEXTLCD_BASE 0xbc
12 #define TEXT_LCD_COMMAND_SET _IOW(TEXTLCD_BASE, 0, int)
13 #define TEXT_LCD_FUNCTION_SET _IOW(TEXTLCD_BASE, 1, int)
14 #define TEXT_LCD_DISPLAY_CONTROL _IOW(TEXTLCD_BASE, 2, int)
15 #define TEXT_LCD_CURSOR_SHIFT _IOW(TEXTLCD_BASE, 3, int)
16 #define TEXT_LCD_ENTRY_MODE_SET _IOW(TEXTLCD_BASE, 4, int)
17 #define TEXT_LCD_RETURN_HOME _IOW(TEXTLCD_BASE, 5, int)
18 #define TEXT_LCD_CLEAR _IOW(TEXTLCD_BASE, 6, int)
19 #define TEXT_LCD_DD_ADDRESS _IOW(TEXTLCD_BASE, 7, int)
20 #define TEXT_LCD_WRITE_BYTE _IOW(TEXTLCD_BASE, 8, int)
21
22 void *fb_mapped;
23 int mem_size;
24
25 typedef struct strcommand_variable
26 {
27     char rows;
28     char nffonts;
29     char display_enable;
30     char curwor_enable;
31     char nblink;
32     char set_screen;
33     char set_rightshift;
34     char increase;
35     char nshift;
36     char pos;
37     char command;
38     char strlength;
39     char buf[16];
40 } strc;
41
42 int i, dev;
43
44 strc strcommand;
45
46 char buf0[16] = "CCTV INITIALIZE";
47 char buf1[16] = "    CCTV Run    ";
48 char buf2[16] = "    CCTV PAUSE   ";
49 char buf3[16] = "WHAT ARE YOU?";
50
51 void display1()
52 {
53     strcommand.pos = 0;
54     ioctl(dev, TEXT_LCD_DD_ADDRESS, &strcommand, 32);
55     ioctl(dev, TEXT_LCD_CLEAR, &strcommand, 32);
56     for (i = 0; i < 16; i++)
57     {
```



```

58         strcommand.buf[0] = buf0[i];
59         ioctl(dev, TEXT_LCD_WRITE_BYTE, &strcommand, 32);
60     }
61 }
62
63 void display2()
64 {
65     strcommand.pos = 0;
66     ioctl(dev, TEXT_LCD_DD_ADDRESS, &strcommand, 32);
67     ioctl(dev, TEXT_LCD_CLEAR, &strcommand, 32);
68     for (i = 0; i < 16; i++)
69     {
70         strcommand.buf[0] = buf1[i];
71         ioctl(dev, TEXT_LCD_WRITE_BYTE, &strcommand, 32);
72     }
73 }
74
75 int textlcdInit()
76 {
77     int cnt = 0;
78
79     strcommand.rows = 0;
80     strcommand.nfonts = 0;
81     strcommand.display_enable = 1;
82     strcommand.nblink = 0;
83     strcommand.set_screen = 0;
84     strcommand.set_rightshift = 1;
85     strcommand.increase = 1;
86     strcommand.nshift = 0;
87     strcommand.pos = 10;
88     strcommand.command = 1;
89     strcommand.strlength = 16;
90     dev = open("/dev/textlcd", O_RDONLY | O_NDELAY);
91     if (dev != -1)
92     {
93
94         write(dev, buf0, 16);
95         strcommand.pos = 40;
96         ioctl(dev, TEXT_LCD_DD_ADDRESS, &strcommand, 32);
97         write(dev, buf1, 16);
98         sleep(1);
99
100        ioctl(dev, TEXT_LCD_CLEAR, &strcommand, 32);
101    }
102    else
103    {
104        printf("can't run\n");
105        exit(1);
106    }
107    return 0;
108 }

```

3.5 camera.h

```

1 #include <stdio.h>
2 #include <string.h>
3 #include <stdlib.h>
4 #include <unistd.h>
5 #include <sys/mman.h>
6 #include <fcntl.h>
7 #include <signal.h>
8 #include <fcntl.h> /* for O_RDONLY */
9 #include <linux/fb.h> /* for fb_var_screeninfo, FBIOGET_VSCREENINFO*/
10 #include <termios.h>
11 #include <ctype.h>
12 #include <assert.h>
13 #include <limits.h>
14 #include <time.h>
15 #include <float.h>
16 #include <math.h>
17 #include <sys/ioctl.h>

```



```

18 void cameraInit();
19 void cameraClean();
20 void camera(char *image_buffer);
21 void display_set();
22 void display_init();
23 #define FBDEV_FILE "/dev/fb0"
24
25 #define CIS_IMAGE_WIDTH 320
26 #define CIS_IMAGE_HEIGHT 240
27 #define CIS_IMAGE_SIZE CIS_IMAGE_WIDTH*CIS_IMAGE_HEIGHT*2
28

```

3.6 camera.c

```

1 #include "camera.h"
2
3 struct fb_var_screeninfo fbvar;
4 unsigned short cisdata[CIS_IMAGE_SIZE];
5
6 int isitFirst = 0;
7 int dev_fd, fbfd;
8 int screen_width;
9 int screen_height;
10 int line_length;
11 struct fb_var_screeninfo fbvar;
12 struct fb_fix_screeninfo fbfix;
13 void *fb_mapped;
14 int mem_size;
15
16 void display_init()
17 {
18     FILE *fp = fopen("running", "rb");
19     unsigned short tmp[800*480];
20     fread(tmp, 800*480*2, 1, fp);
21     //unsigned short tmp2[800*480];
22     unsigned short *ptr;
23     int coor_x, coor_y;
24
25     for (coor_y = 0; coor_y < 480; coor_y++)
26     {
27         ptr = (unsigned short *)fb_mapped + screen_width * coor_y;
28         for (coor_x = 0; coor_x < 800; coor_x++)
29         {
30             *ptr++ = tmp[coor_y * 800 + coor_x];
31         }
32     }
33 }
34
35 //bg = background (800*480 16bit image)
36 void display_set()
37 {
38     FILE *fp = fopen("initialize", "rb");
39     unsigned short tmp[800*480];
40     fread(tmp, 800*480*2, 1, fp);
41     //unsigned short tmp2[800*480];
42     unsigned short *ptr;
43     int coor_x, coor_y;
44     for (coor_y = 0; coor_y < 480; coor_y++)
45     {
46         ptr = (unsigned short *)fb_mapped + screen_width * coor_y;
47         for (coor_x = 0; coor_x < 800; coor_x++)
48         {
49             *ptr++ = tmp[coor_y * 800 + coor_x];
50         }
51     }
52 }
53
54 void cameraInit(){
55     if ((dev_fd = open("/dev/camera", O_RDWR | O_SYNC)) < 0)
56     {
57         perror("camera open");

```



```

58         exit(1);
59     }
60     if (access(FBDEV_FILE, F_OK)) //Is it possible to R/W?
61     {
62         printf("%s: access error\n", FBDEV_FILE);
63         exit(1);
64     }
65     if ((fbfd = open(FBDEV_FILE, O_RDWR)) < 0)
66     {
67         printf("%s: open error\n", FBDEV_FILE);
68         exit(1);
69     }
70     if (ioctl(fbfd, FBIOGET_VSCREENINFO, &fbvar) < 0)
71     {
72         printf("%s: ioctl error - FBIOGET_VSCREENINFO \n", FBDEV_FILE);
73         exit(1);
74     }
75     if (ioctl(fbfd, FBIOGET_FSCREENINFO, &fbfix))
76     {
77         printf("%s: ioctl error - FBIOGET_FSCREENINFO \n", FBDEV_FILE);
78         exit(1);
79     }
80     screen_width = fbvar.xres;
81     screen_height = fbvar.yres;
82     line_length = fbfix.line_length;
83     mem_size = line_length * screen_height;
84     fb_mapped = (void *)mmap(0, mem_size, PROT_READ | PROT_WRITE, MAP_SHARED, fbfid, 0);
85     display_init(); //init once
86
87     printf("1.3M CIS Camera Application (320x240)\n");
88
89 }
90
91 void cameraClean(){
92     munmap(fb_mapped, mem_size);
93     close(dev_fd);
94     close(fbfd);
95 }
96 void camera(char *image_buffer)
97 {
98
99     write(dev_fd, NULL, 1);
100    read(dev_fd, cisdata, CIS_IMAGE_SIZE);
101    int x = 0, y = 0;
102    for (y = 0; y < 240; y++)
103    {
104        for (x = 0; x < 320; x++)
105        {
106            unsigned int pixelIdx = ((y * 320) + x) * 3;
107            image_buffer[pixelIdx + 0] = ((cisdata[y * 320 + x] & 0xF800) >> 11) << 3; //r
108            image_buffer[pixelIdx + 1] = ((cisdata[y * 320 + x] & 0x07E0) >> 5) << 2; //g
109            image_buffer[pixelIdx + 2] = (cisdata[y * 320 + x] & 0x1F) << 3; //b
110        }
111    }
112 }
113
114
115 }
```

3.7 jpegComp.h

```

1 #include <stdio.h>
2
3 #define IMAGE_WIDTH 320
4 #define IMAGE_HEIGHT 240
5
6
7 int jpegCompress(unsigned char * inputBuffer, unsigned char ** outImgBuffer, unsigned long *
→ outImgSize);
```



3.8 jpegComp.c

```

1 #include "jpegComp.h"
2 #include <jpeglib.h>
3 #include <cderror.h>
4 #include <cdjpeg.h>
5 #include <jversion.h>
6
7
8 int jpegCompress( unsigned char * inputBuffer, unsigned char ** outImgBuffer, unsigned long *
9   → outImgSize)
10 {
11     //300kb = 300 * 1024
12     //img size는 원만하면 300 안넘음
13     outImgBuffer[0] = (unsigned char *)calloc(300 * 1024, 1);
14     *outImgSize = 300 * 1024;
15
16     struct jpeg_compress_struct cinfo;
17     struct jpeg_error_mgr jerr;
18
19
20     cjpeg_source_ptr src_mgr;
21     JDIMENSION num_scanlines;
22
23     cinfo.err = jpeg_std_error(&jerr);
24     //jerr.addon_message_table = cdjpeg_message_table;
25     jerr.first_addon_message = JMSG_FIRSTADDONCODE;
26     jerr.last_addon_message = JMSG_LASTADDONCODE;
27
28
29     jpeg_create_compress(&cinfo);
30     jpeg_mem_dest(&cinfo, outImgBuffer, outImgSize);
31
32
33     cinfo.image_width = IMAGE_WIDTH;
34     cinfo.image_height = IMAGE_HEIGHT;
35     cinfo.input_components = 3;
36     cinfo.in_color_space = JCS_RGB;
37     jpeg_set_defaults(&cinfo);
38     jpeg_set_quality(&cinfo, 70, TRUE );
39
40     jpeg_start_compress(&cinfo, TRUE);
41
42
43     unsigned char *row_pointer[1];
44
45     int row_stride = IMAGE_WIDTH * 3;
46     while (cinfo.next_scanline < cinfo.image_height)
47     {
48         row_pointer[0] = &inputBuffer[cinfo.next_scanline * row_stride];
49         (void)jpeg_write_scanlines(&cinfo, row_pointer, 1);
50     }
51
52     jpeg_finish_compress(&cinfo);
53     jpeg_destroy_compress(&cinfo);
54
55
56 }
```

3.9 mjpegServer.h

```

1 #include <unistd.h>
2 #include <stdio.h>
3 #include <stdlib.h>
4 #include <string.h>
5 #include <arpa/inet.h>
6 #include <sys/types.h>
7 #include <sys/stat.h>
8 #include <sys/socket.h>
9 #include <ifaddrs.h>
```



```

10 #include <netdb.h>
11 #include <netinet/in.h>
12 #include <math.h>
13 #include <signal.h>
14
15 void initSocket(int port);
16 void createConnnection();
17 void closeConnection();
18 void closeSocket();
19 int sendFrame();

```

3.10 mjpegServer.c

```

1 #include "mjpegServer.h"
2
3 // string for socket
4 const char* header_packet = "HTTP/1.0 200 OK\r\nServer: EMPOIII\r\nConnection:
5 →   close\r\nContent-Type: multipart/x-mixed-replace; boundary=myboundary\r\n\r\n\r\n";
6 const char* mjpegHeaderPacket = "--myboundary\r\nContent-Type: image/jpeg\r\nContent-Length: ";
7 FILE* fp;
8 //char* jpegData;
9 int on = 1;
10 char buffer[1024] = { 0, };
11 size_t read_len;
12 //int jpegSize;
13 int client;
14 int sock;
15 struct sockaddr_in addr;
16 struct sockaddr_in client_addr;
17 socklen_t addr_len = sizeof(addr);
18
19 #define DEBUG
20
21
22 void createConnnection(){
23
24     client = accept(sock, (struct sockaddr*)&client_addr, &addr_len);
25     if (client < 0) {
26         perror("accept() error");
27         return;
28     }
29     read_len = recv(client, buffer, sizeof(buffer), 0);
30
31     #ifdef DEBUG
32     if (read_len > 0) {
33         printf("RECV: %s", buffer);
34     }
35     #endif //DEBUG
36     send(client, header_packet, strlen(header_packet), 0);
37     printf("header send\n");
38 }
39
40 void closeConnection(){
41     close(client);
42 #ifdef DEBUG
43     printf("closeConnection\n");
44 #endif //DEBUG
45 }
46
47 void closeSocket(){
48     close(sock);
49 #ifdef DEBUG
50     printf("closeSocket\n");
51 #endif //DEBUG
52 }
53
54 void initSocket(int port){
55     int bind_port = port;
56     sock = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP);

```



```

58     if (sock < 0) {
59         perror("socket() error");
60     return;
61 }
62
63     if (setsockopt(sock, SOL_SOCKET, SO_REUSEADDR, (void*)&on, sizeof(on)) != 0) {
64         perror("setsockopt() error");
65     return;
66 }
67
68     addr.sin_family = AF_INET;
69     addr.sin_addr.s_addr = htonl(INADDR_ANY);
70     addr.sin_port = htons(bind_port);
71     if (bind(sock, (struct sockaddr*)&addr, addr_len) != 0) {
72         perror("bind() error");
73     return;
74 }
75
76     if (listen(sock, 5) != 0) {
77         perror("listen() error");
78     return;
79 }
80     printf("listening... %d\n", bind_port);
81 }
82
83
84
85
86 int sendFrame(char * jpegData, int jpegSize){
87     int err;
88
89     err = send(client, mjpegHeaderPacket, strlen(mjpegHeaderPacket), MSG_NOSIGNAL);
90     if (err < 0){
91         return -1;
92     }
93     char size[20];
94     sprintf(size,"%d", jpegSize);
95     send(client, size, strlen(size), MSG_NOSIGNAL);
96     //printf(size);
97     //printf("%d", sizelen);
98     send(client, "\r\n\r\n", strlen("\r\n\r\n"), MSG_NOSIGNAL);
99
100    err = send(client, jpegData, jpegSize, MSG_NOSIGNAL);
101    if (err < 0){
102        return -1;
103    }
104    //send(client, "\r\n\r\n", strlen("\r\n\r\n"), MSG_NOSIGNAL);
105    err = send(client, "\r\n", strlen("\r\n"), MSG_NOSIGNAL);
106    if (err < 0){
107        return -1;
108    }
109
110 }
111 }
```

3.11 build

```

1 arm-linux-gcc -c mjpegServer.c
2 arm-linux-gcc -c main.c
3 arm-linux-gcc -c camera.c
4 arm-linux-gcc -c text.c
5 arm-linux-gcc -c jpegComp.c -L../../jpeg-8a/buildlib/buildlib/lib -ljpeg -L../../jpeg-8a
6 arm-linux-gcc -o cameraStreamServer -O2 -L../../jpeg-8a/buildlib/buildlib/lib -ljpeg -L../../jpeg-8a
→ camera.o jpegComp.o main.o mjpegServer.o text.o
```



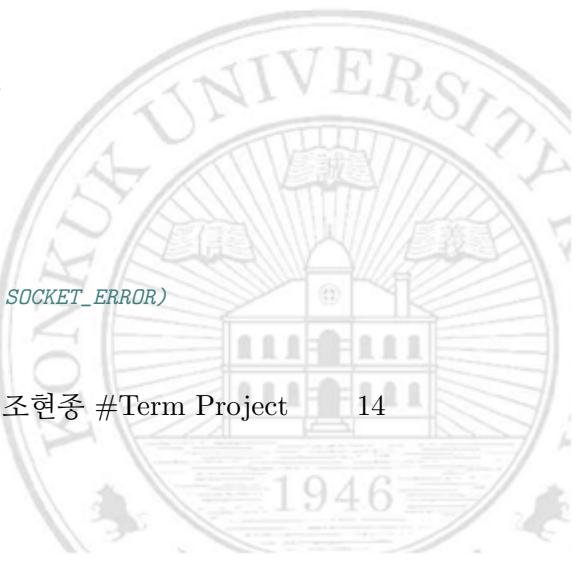
3.12 클라이언트 뷰어

3.13 build

```

1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <string.h>
4 #include <vector>
5 #include <iostream>
6 #include <fstream>
7 #include <opencv2/opencv.hpp>
8 #include <opencv2/objdetect/objdetect.hpp>
9 #include <opencv2/video/tracking.hpp>
10 #include "opencv2/core/core.hpp"
11 #include <opencv2/core/ocl.hpp>
12 #include <time.h>
13 #include <winsock2.h>
14 #include <sys/types.h>
15 #include <WinBase.h>
16
17
18 #pragma comment (lib , "ws2_32.lib")
19 #pragma comment(lib, "vfw32.lib")
20 #pragma comment( lib, "comctl32.lib" )
21
22
23 using namespace std;
24 using namespace cv;
25
26 const std::string currentDateTime() {
27     time_t now = time(0); //현재 시간을 time_t 타입으로 저장
28     struct tm tstruct;
29     char buf[80];
30     tstruct = *localtime(&now);
31     strftime(buf, sizeof(buf), "%Y-%m-%d.%X", &tstruct); // YYYY-MM-DD.HH:mm:ss 형태의 스트링
32     return buf;
33 }
34
35 void showError(const char * msg)
36 {
37     std::cout << "에러 : " << msg << std::endl;
38     exit(-1);
39 }
40
41
42 int main() {
43
44     VideoCapture capture;
45     string videoStreamAddress;
46     char Tosocket[50];
47     cout << "Put Your IP Camera Address : ";
48     cin >> videoStreamAddress;
49     string A = "http://";
50     string B = "/mjpg/video.mjpg";
51     strcpy(Tosocket, videoStreamAddress.c_str());
52     videoStreamAddress = A + videoStreamAddress + B;
53
54     capture.open(videoStreamAddress);
55
56     WSADATA data;
57     ::WSAStartup(MAKEWORD(2, 2), &data);
58
59     SOCKET client = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP);
60
61 //if (client == INVALID_SOCKET)
62 //showError("클라이언트 생성 실패");
63
64     sockaddr_in addr = { 0 };
65     addr.sin_family = AF_INET;
66     addr.sin_addr.s_addr = inet_addr(Tosocket);
67     addr.sin_port = htons(23000);
68 //if (connect(client, (sockaddr *)&addr, sizeof(addr)) == SOCKET_ERROR)
69 //    showError("연결 실패");

```




```

140         writer.open(motion, VideoWriter::fourcc('M', 'J', 'P', 'G'), fps, size,
141             → true);
142         save = 1;
143         if (!writer.isOpened())
144         {
145             cout << "save avi error" << endl;
146             return 1;
147         }
148         if (save){
149             isitfirst = 0;
150             if (cnt == 30) {
151                 printf("done\n");
152                 writer.release();
153                 isitfirst = 1;
154             }
155             writer << frame;
156         }
157
158         cnt++;
159         resize(frame, frame, Size(frame.cols * 3/2, frame.rows * 3/2));
160
161         imshow("frame", frame);
162         if (face)
163         {
164             resize(face_img, face_img, Size(face_img.cols * 3 / 2, face_img.rows * 3
165             → / 2));
166             imshow("face", face_img);
167             face = 0;
168         }
169         if (waitKey(5) == 27) break;
170     }
171
172     closesocket(client);
173     ::WSACleanup();
174     return 0;
175 }
```

4 cross compile

4.1 libjpeg 8a

```

1 wget http://www.ijg.org/files/jpegsr8a.tar.gz
2 #압축풀고이동
3 ./configure --host=arm-none-linux-gnueabi CC=arm-none-linux-gnueabi-gcc
   → AR=arm-none-linux-gnueabi-ar STRIP=arm-none-linux-gnueabi-strip
   → RANLIB=arm-none-linux-gnueabi-ranlib --prefix=<own_your_abs_libjpeg_build_path>
4 make
5 make install
6 #0이후 지정한 <빌드 패스>/lib/ 내의 파일들을 전부 보드의 /lib 디렉토리로 옮길것
```

이후에 cross compile 할때 jpeg 라이브러리를 사용하려면 gcc 옵션으로 -L'빌드폴더'/lib, -ljpeg, -I'빌드폴더'/include를 사용한다.

4.2 apache httpd

부록으로 별첨하겠습니다.

5 result





Figure 2: camera off



Figure 3: camera on



Figure 4: text lcd when camera init



Figure 5: text lcd when camera run



Figure 6: stream on smartphone



6 discussion

6.1 device control

6.1.1 state machine by dipsw

보드의 동작 상태를 물리적으로 가장 직관적으로 판단할 수 있게 하기 위해서 dip_sw를 통해 동작을 제어한다.

```

1  if ((dev = open("/dev/dipsw", O_RDONLY)) < 0)
2  {
3      perror("DIPSW open fail\n");
4      return -1;
5 }
```

시스템 콜 함수를 통해 커널 모드로 진입하여 시스템 콜 인터페이스에서 파일을 읽고 쓸 수 있다. dipsw를 사용하기 위해 open() 시스템 호출을 이용하여 파일을 열어준다.

```
1  read(dev, &vkey_new[0], 4);
```

device를 열어서 vkey[0]의 값을 읽어온다. vkey_new[0]은 dipsw 1번의 값이다. dip_sw 1번의 값이 일 경우에 어플리케이션을 실행하고 값이 일 경우에 중단한다 그 외의 경우에는 카메라 꺼짐의 상태를 유지한다.

6.1.2 textlcd

```

1  if (vkey_new[0] == 1 || vkey_new[0] == 9){
2      display2();
3      display_init();
4      camera(imageBuffer);
5
6      #ifdef DEBUG
7      printf("camera");
8      #endif //Debug
9      jpegCompress(imageBuffer, jpegBuffer, &jpegSize);
10     #ifdef DEBUG
11     printf("compress");
12     #endif //Debug
13     cameradisplay();
14     err = sendFrame(*jpegBuffer, jpegSize);
15     #ifdef DEBUG
16     printf("sendFrame");
17     #endif //Debug
18     if (err == -1){
19         closeConnection();
20         createConnetion();
21     }
22 }else{
23     sendFrame(*camCloseImg, camCloseImgSize);
24     sleep(1);
25     display1();
26     display_set();
27 }
28 if (vkey_new[0] == 9){
29     f_loop = 0;
30     cameraClean();
31     display1();
32 }
```

Text lcd를 사용하기 위해 text.c 파일과 text.h 파일을 만들어주었다. 디바이스 커널을 만든 후 시스템 콜 함수를 사용하여 디바이스를 조절할 수 있다, .

```

1 #define TEXTLCD_BASE 0xb8
2 #define TEXT_LCD_COMMAND_SET _IOW(TEXTLCD_BASE, 0, int)
```

```

3 #define TEXT_LCD_FUNCTION_SET _IOW(TEXTLCD_BASE, 1, int)
4 #define TEXT_LCD_DISPLAY_CONTROL _IOW(TEXTLCD_BASE, 2, int)
5 #define TEXT_LCD_CURSOR_SHIFT _IOW(TEXTLCD_BASE, 3, int)
6 #define TEXT_LCD_ENTRY_MODE_SET _IOW(TEXTLCD_BASE, 4, int)
7 #define TEXT_LCD_RETURN_HOME _IOW(TEXTLCD_BASE, 5, int)
8 #define TEXT_LCD_CLEAR _IOW(TEXTLCD_BASE, 6, int)
9 #define TEXT_LCD_WRITE_BYTE _IOW(TEXTLCD_BASE, 7, int) #define TEXT_LCD_DD_ADDRESS _IOW(TEXTLCD_BASE, 8, int) void *fb_mapped; int mem_size; typedef struct strcommand_variable
10 {
11     char rows;
12     char nfonts;
13     char display_enable;
14     char curwor_enable;
15     char nblink;
16     char set_screen;
17     char set_rightshift;
18     char increase;
19     char nshift;
20     char pos;
21     char command;
22     char strlenth;
23     char buf[16];
24 } strc;

```

ioctl 함수를 사용하여 디바이스 드라이버에 접근한 경우 호출되는 함수와, textlcd에 표시되는 값을 조절하기 위한 정보 저장구조체를 설정한다.

```

1 char buf0[16] = "CCTV INITIALIZE";
2 char buf1[16] = "      CCTV Run    ";

```

textlcd의 크기에 맞추어 문자형 배열을 만들어 준 후 상황에 맞는 문자열을 표시한다,

```

1 void display1()
2 {
3     strcommand.pos = 0;
4     ioctl(dev, TEXT_LCD_DD_ADDRESS, &strcommand, 32);
5     ioctl(dev, TEXT_LCD_CLEAR, &strcommand, 32);
6     for (i = 0; i < 16; i++)
7     {
8         strcommand.buf[0] = buf0[i];
9         ioctl(dev, TEXT_LCD_WRITE_BYTE, &strcommand, 32);
10    }
11 }

```

strcommand.pos에 을 주어서 0 textlcd의 표시 위치를 윗 줄로 설정한다. 디바이스 제어 함수를 사용해서 text_lcd의 주소에 접근하고, 먼저 textlcd를 지워준 다음에 buf0의 내용을 써준다. display1() → CCTV INITIALIZE display2() → CCTV Run

// jpeg 24bit를 16bit rgb로 변환 sv210 보드의 디스플레이에 사진을 넣기 위해, jpeg 24bit로 되어 있는 파일을 16bit rgb 파일로 변환하여 전송하였다.

```

1 //FILE* pFile = fopen("running", "wb"); FILE* fp = fopen("first_initialize", "wb");
2 unsigned char Red, Blue, Green;
3 unsigned short pixel = 0;
4 for (int y = 0; y < height; y++)
5 {
6     for (int x = 0; x < width; x++)
7     {
8         Red = img_color.at<Vec3b>(y, x)[2] >> 3;//R8
9         Green = img_color.at<Vec3b>(y, x)[1] >> 2;//G8 Blue = img_color.at<Vec3b>(y, x)[0] >>
10        3;//B8
11        pixel = ((unsigned short)Red << 11 | (unsigned short)Green
12        << 5 | (unsigned short)Blue); fwrite(&pixel, 2, 1, fp);
13        //fprintf(fp, "\n");
14    }
15    printf("\n");
16 }
fclose(fp);

```

unsigned short 형으로 pixel 변수를 선언한 후, red 부분의 상위 비트5 , green 부분의 6 비트, blue 부분의 비트를 분리 한 후에 상위 5 5,6,5비트를 각각 r,g,b로 or 연산하여 합쳐 준 뒤에 binary 파일로 저장한 후, 보드 내에서 open해서 출력해준다.

6.2 camera to jpeg

HBE - EMPOIII - SV210에는 CIS(CMOS Image Sensor) camera를 사용하여 외부로부터 RGB Data를 입력 받는다. camera module에서 socket을 통해 웹으로 이미지 데이터를 보내기 위해선 파일 크기가 작은 포맷을 사용해야 한다 이미지가 한 장이면 전송 속도가 큰 비중을 갖지 않지만 실시간, MJPEG를 구현하기 위해선 보드의 이미지 전송 속도가 무엇보다 중요하다.

해당 보드에서 cv함수를 통해 입력받은 CIS data를 전송할 수도 있었지만 보드 내에서, cv 함수들을 사용하기에는 속도의 저하가 예상되어 이미지 품질은 떨어지지만 파일 크기가 작은 JPEG Compress 방식을 사용했다. JPEG Compress는 row별로 담긴 r / g / b를 읽어 파일에 쓰기 때문에 CIS data의 r / g / b value를 분리해야 한다. cameraInit() 함수로 camera와 display screen을 open() 하고, display_Init() 을 한다.

```

1   void cameraInit()
2 {
3     if ((dev_fd = open("/dev/camera", O_RDWR | O_SYNC)) < 0)
4     {
5       perror("camera open");
6       exit(1);
7     }
8     if (access(FBDEV_FILE, F_OK)) //Is it possible to R/W?
9     {
10       printf("%s: access error\n", FBDEV_FILE);
11       exit(1);
12     }
13     if ((fbfd = open(FBDEV_FILE, O_RDWR)) < 0)
14     {
15       printf("%s: open error\n", FBDEV_FILE);
16       exit(1);
17     }
18     if (ioctl(fbfd, FBIOGET_VSCREENINFO, &fbvar) < 0)
19     {
20       printf("%s: ioctl error - FBIOGET_VSCREENINFO \n", FBDEV_FILE);
21       exit(1);
22     }
23     if (ioctl(fbfd, FBIOGET_FSCREENINFO, &fbfix))
24     {
25       printf("%s: ioctl error - FBIOGET_FSCREENINFO \n", FBDEV_FILE);
26       exit(1);
27     }
28     screen_width = fbvar.xres;
29     screen_height = fbvar.yres;
30     line_length = fbfix.line_length;
31     mem_size = line_length * screen_height;
32     fb_mapped = (void *)mmap(0, mem_size, PROT_READ | PROT_WRITE, MAP_SHARED, fbfid, 0);
33     display_init(fb_mapped, screen_width); ///init once      printf("1.3M CIS Camera Application
34   }                                     ↪ (320x240)\n");

```

dip_sw의 값(vkey_new[0])이 1이 들어오면 1 CIS camera의 값을 16bit의 cisdata로 받아 rgb 값을 분리해 image_buffer에 담는다.

```

1   void camera(char *image_buffer)
2 {
3   write(dev_fd, NULL, 1);
4   read(dev_fd, cisdata, CIS_IMAGE_SIZE);
5   int x = 0, y = 0;
6   for (y = 0; y < 240; y++)
7   {
8     for (x = 0; x < 320; x++)

```



```

9
10    {
11        unsigned int pixelIdx = ((y * 320) + x) * 3;
12        image_buffer[pixelIdx + 0] = ((cisdata[y * 320 + x] & 0xF800) >> 11) << 3; //r
13        → image_buffer[pixelIdx + 1] = ((cisdata[y * 320 + x] & 0x07E0) >> 5) << 2; //g
14        → image_buffer[pixelIdx + 2] = (cisdata[y * 320 + x] & 0x1F) << 3; //b
15    }
16 }

```

Red value는 상위 5bit(15 11)으로 cisdata에 0xF800을 masking해 추출한 후 unsigned char 형에 맞게 3 right shifting 한다. JPEG Compress는 row별로 rgb를 한 번에 읽어오기 때문에 image_buffer의 Index인 pixelIdx가 * 3이 된다. 현 이미지의 cisdata가 imge.buffer에 입력이 끝나면 jpegCompress() 가 호출되고 IMAGE_WIDTH * 3의 row별로 wirte가 되는 걸 볼 수 있다.

```

1 unsigned char *row_pointer[1];
2 int row_stride = IMAGE_WIDTH * 3; /* JSAMPLEs per row in image_buffer */
3 while (cinfo.next_scanline < cinfo.image_height)
4 {
5     row_pointer[0] = &inputBuffer[cinfo.next_scanline * row_stride];
6     (void)jpeg_write_scanlines(&cinfo, row_pointer, 1);
7 }

```

dip_sw의 값(vkey_new[0]) 이 9가 들어오면 들어온 시점의 이미지까지 send() 한다. 이 후 다음 조건문으로 cameraClean() 이 호출되고 할당받은 가상주소 해제 munmap() 와 디바이스들을 close() 한다.

```

1 void cameraClean()
2 {
3     munmap(fb_mapped, mem_size);
4     close(dev_fd);
5     close(fbfd);
6 }

```

6.3 클라이언트 프로그램

socket을 통해 전송된 웹서버의 MJPEG를 실시간을 읽어와 움직임과 얼굴을 감지한다. 시중의 IP Camera들은 모든 프레임을 파일에 저장하기 때문에 데이터 용량과 저장에 어려움이 있다. 이번 TermProject에서는 저장 파일의 용량을 개선하고자 움직임이 detect되면 해당 시 점부터 움직임이 사라진 후 초까지의 영상을 계속해서 분리 저장하는 방법을 고안했다. 현 프레임과 전 프레임의 배열을 absdiff() 를 통해 절대적 차이를 diffImage에 저장한 다. threshold() 에 diffImage에 threshold value를 주고 masking된 이미지에 noise와 같은 작은 object들을 dilate() 로 제거해준다. 노이즈 제거된 이미지 mask에서 findContours() 로 윤곽을 찾는다.

```

1 cvtColor(frame, grayImage, COLOR_BGR2GRAY);
2 cvtColor(last_frame, grayImage2, COLOR_BGR2GRAY);
3 GaussianBlur(grayImage, grayImage, Size(5, 5), 0.5);
4 avgImage = grayImage2;
5 absdiff(grayImage, avgImage, diffImage);
6 threshold(diffImage, mask, 50, 255, THRESH_BINARY);
7 dilate(mask, mask, Mat(), Point(-1, -1), 2);
8 findContours(mask, cnts, RETR_EXTERNAL, CHAIN_APPROX_SIMPLE);
9 putText(frame, currentDate(), Point(frame.cols - frame.cols / 3 * 2, frame.rows - frame.rows
→ / 8), FONT_HERSHEY_SIMPLEX, 0.55, Scalar(255, 255, 255), 2);
    찾은 윤곽의 수가 일정값 이하면 움직임이 없다고 가정하고 continue 한다 다음 프레임을 받는다.

```

```

1     for (int i = 0; i < cnts.size(); i++)
2     {
3         if (contourArea(cnts[i]) < 500)
4         {
5             continue;
6         }
7         putText(frame, "Motion Detected", Point(10, 20), FONT_HERSHEY_SIMPLEX,

```

```

8     0.75, Scalar(0, 0, 255), 2);
9     printf("Motion Detected!!\n"); //char motionDetect[] = "Detect!";
10    //send(client, motionDetect, strlen(motionDetect), 0); detect = 1; cnt = 0;
11 }

```

매 프레임에서 facedetect는 lbp기반의 lbpcascade_frontalface를 사용한다. detect된 faces vector의 Point를 이용해 rectangle() 함수로 직사각형을 그린다.

```

1   for (int y = 0; y < faces.size(); y++)
2 {
3     Point lb(faces[y].x + faces[y].width, faces[y].y + faces[y].height);
4     Point tr(faces[y].x, faces[y].y);
5     Rect rect(faces[y].x, faces[y].y, faces[y].width, faces[y].height);
6     rectangle(frame, lb, tr, Scalar(0, 255, 0), 2, 8, 0);
7     face_img = frame(rect);
8     face = 1;
9 }

```

motion detect가 되면 VideoWriter open() 을 통해 매 프레임이 저장된다. motion이 사라지면 release() 해주고 다음 motion detect가 detect되면 motiondetect % 2d의 새로운 파일에 video가 쓰인다.

```

1 if (isitfirst && detect)
2 {
3     char motion[100];
4     sprintf(motion, "motiondetect%2d.avi", numb++);
5     writer.open(motion, VideoWriter::fourcc('M', 'J', 'P', 'G'), fps, size, true);
6     save = 1;
7     if (!writer.isOpened())
8     {
9         cout << "save avi error" << endl;
10        return 1;
11    }
12 }
13 if (save)
14 {
15     isitfirst = 0;
16     if (cnt == 30)
17     {
18         printf("done\n");
19         writer.release();
20         isitfirst = 1;
21     }
22     writer << frame;
23 }

```

6.4 jpeg compress

위에서 24비트 이미지로 변환한 버퍼를 가지고, libjpeg 함수를 이용해 변환한다. jpeg 이미지가 들어갈 적당한 크기, 여기서는 300kb 버퍼를 동적 할당 한 뒤 cinfo 스트럭트에 이미지 정보를 기록해준다. 일반적으로는 파일포인터를 열고 파일을 스트럭트에 지정해 주지만, 우리는 메모리에 버퍼를 두고 사용해야 하므로 jpeg_mem_dest 함수를 사용해 아웃풋 버퍼를 지정해 준다. jpeg_write_scanlines 함수에 한 col씩 포인터를 건내주면 jpeg 생성은 완료된다.

6.5 mjpeg server

리눅스의 소켓 함수를 사용하여 구성하였다. mjpeg는 위에서 설명했듯이 mime를 사용, jpeg 이미지를 border로 구분하여 전송해 만들었다. 실제로, 파이썬으로 구현된 소스는 많았지만 c로 구현되어 있는 소스는 웹상에서도 찾기 매우 힘들었기에 직접 mime를 공부해가며 코딩하고, wireshark를

이용해 패킷을 하나하나 분석해가며 만들었다. 먼저 initSocket 함수에서 포트를 정하고 소켓을 열어준다. createConnection 함수에서 헤더 패킷을 보낸 뒤에 sendFrame 함수를 통해 바운더리패킷을 보내준다. contents type 은 image/jpeg로 지정하고 contents-length를 jpeg의 바이너리 사이즈로 지정한다.

7 Conclusion

EMPOSIII 보드를 사용하여 웹 스트리밍 cctv 솔루션을 제작하여 보았다. 수업시간에 배웠던 디바이스 드라이버를 이용해 딥스위치와 카메라, textlcd를 조작하고 mmap을 통해 lcd를 조작하면서 수업시간에 배운 것들을 충실히 복습하였다. 수업시간에 배운것 외에도 직접 라이브러리를 크로스 컴파일 하여 jpeg 파일을 만들며 이미지 파일 구조와 웹 소켓 대해 학습하였으며, 이를 socket을 통해 전송하여 원격지의 웹상에서 동영상 스트림을 볼수있게 구성했다. 더 나아가, x86 대상의 클라이언트 프로그램을 만들어 보드의 한정된 리소스에서 벗어나 강력한 computer vision을 활용해 최근 cctv기능들을 구현해 보았다.

