Embedded System Software 프로젝트

Webcam control App & Device

(결과 보고서)

과목명: [CSE4116] 임베디드시스템소프트웨어

담당교수: 서강대학교 컴퓨터공학과 박 성 용

학번 및 이름: 120230200, 김지섭

개발기간: 2023. 06. 01. - 2023. 06. 26.

I. 개발 목표

한 학기 동안 습득한 리눅스 시스템 프로그램 및 안드로이드 프로그래밍 기법을 이용해 실습 보드 상에서 자유 주제로 임베디드 소프트웨어를 개발한다.

Project 목표:

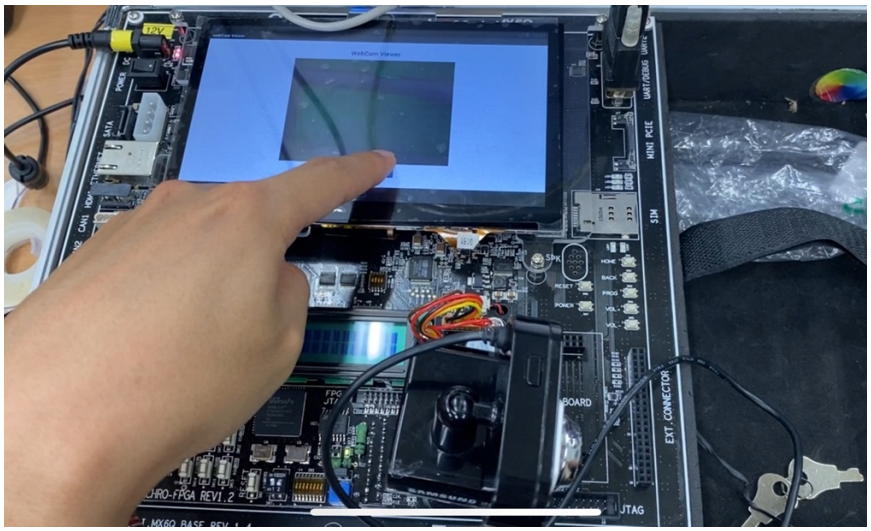
- Embedded Board 에 USB webcam 을 연결하여, Android App 에 해당 영상을 송출한다.
- Webcam 이 원하는 위치를 촬영할 수 있도록 방향을 조절할 수 있도록 한다.
- 사진 촬영 및 저장 후 Android 의 갤러리에서 해당 이미지를 확인할 수 있도록 한다.

II. 개발 범위 및 내용

가. 개발 범위

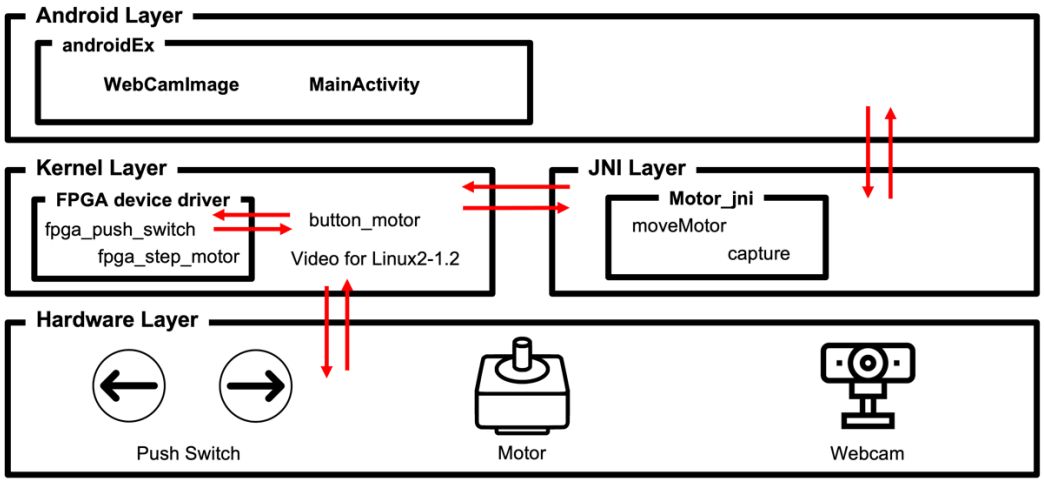
Webcam 과 Motor 를 JNI 를 활용하여 Control 하는 Android Application 구현
Motor Device driver 와 Video for Linux2 를 활용한 JNI 구현
Push switch 와 motor 의 device driver 를 활용한 Motor control application 구현

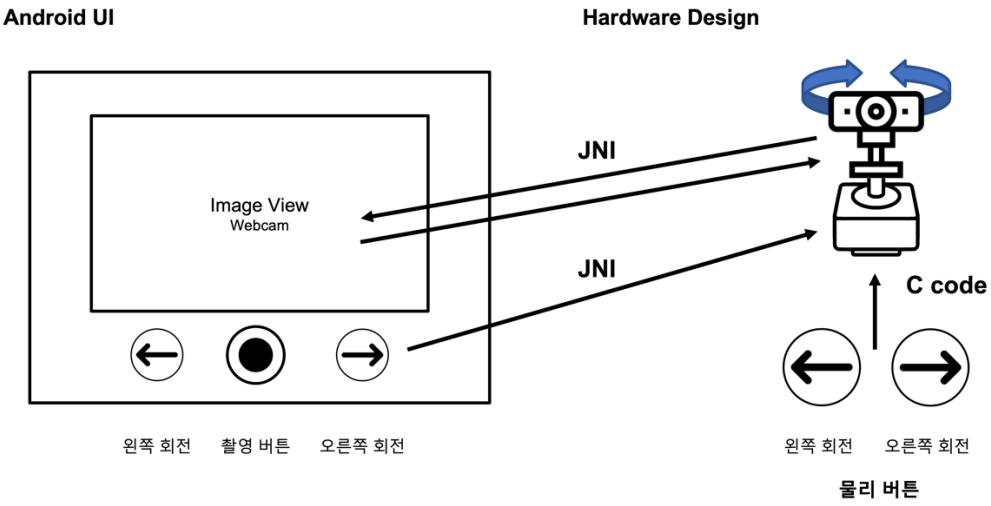
1. Android Application 예시



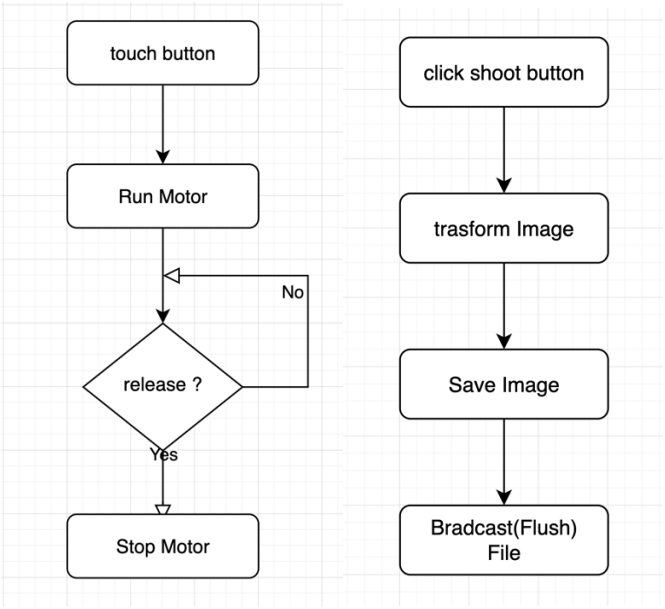
나. 개발 내용

1. System Architecture





2. Flow chart



대부분의 Code 가 순차적으로 수행되기 때문에 OnTouch 와 OnClick 의 차이점에 대해 보여주는 것으로 Flow chart 를 그렸습니다.

OnTouch 의 경우 Release 된 경우 Motor 를 멈추는 JNI 함수를 수행하지만, 사진을 촬영하는 onClick 의 경우 최초 click 이 확인되면, 이후의 touch 여부는 중요하지 않기때문에 기능적 차이와 flow 적 차이가 존재합니다.

3. Android Application 구현

```
public class MainActivity extends Activity implements OnTouchListener, OnClickListener {
	private static final String TAG = "WebCam";
	private native int moveMotorRight();
	private native int moveMotorLeft();
	private native int stopMotor();
	private native byte[] capture();
	ImageView webCam;
	Bitmap bitmap = null;
	byte[] source = null;

	Thread thread;
	static{
	System.loadLibrary("motor_jni");
	};
}
```

Main Activity 선언부 입니다. JNI 를 사용하기 위해 System.loadLibaray("motor_jni")를 활용하여, libmotor_jni.so 파일을 load 합니다.

```
@Override
 protected void onCreate(Bundle savedInstanceState) {
	super.onCreate(savedInstanceState);
	Log.d(TAG, "Start Application");
	setContentView(R.layout.fragment_main);
	View leftButton = findViewById(R.id.left_button);
	leftButton.setOnTouchListener(this);
	View rightButton = findViewById(R.id.right_button);
	rightButton.setOnTouchListener(this);
	View shootButton = findViewById(R.id.shoot_button);
	shootButton.setOnClickListener(this);
	webCam = (ImageView) findViewById(R.id.web_cam);

	if(thread == null) {
		thread = new Thread(new WebCamImage());
	}
	thread.start();

 }
```

onCreate 입니다. ImageView 를 지속적으로 갱신하기 위해 uiThread 를 사용했고, 이때, 별도의 thread 를 구현하기 위햇 thread 를 생성한 후 thread 를 실행시켰습니다.

```
@Override
 public boolean onTouch(View v, MotionEvent event) {
      // TODO Auto-generated method stub
      switch(v.getId()) {
           case R.id.right_button :
                switch(event.getAction()) {
                     case MotionEvent.ACTION_DOWN:
                          // PRESSED
                      Log.d(TAG, "Pressed Right");
                      Log.d(TAG, String.valueOf(moveMotorRight()));
                          return true; // if you want to handle the touch event
                     case MotionEvent.ACTION_UP:
                          // RELEASED
                      Log.d(TAG, "Released Right");
                      Log.d(TAG, String.valueOf(stopMotor()));
                      return true; // if you want to handle the touch event
                }
           case R.id.left_button :
                switch(event.getAction()) {
                     case MotionEvent.ACTION_DOWN:
                          // PRESSED
                      Log.d(TAG, "Pressed Left");
                      Log.d(TAG, String.valueOf(moveMotorLeft()));
                          return true; // if you want to handle the touch event
                     case MotionEvent.ACTION_UP:
                          // RELEASED
                      Log.d(TAG, "Released Left");
                      Log.d(TAG, String.valueOf(stopMotor()));
                          return true; // if you want to handle the touch event
                }
      }
      return false;
 }
```

onTouch 함수입니다. SW 버튼을 누르는 동안 모터가 돌아가고, 손을 떼는 순간 모터가 멈추는 것을 구현하기 위해 강의 시간에 배운 onClickListener 가 아닌 onTouchListener 를 활용하였습니다. 이를 통해, 손을 버튼에서 떼는 것 또한 event 로 인식하여, 해당 event 발생시 motor 를 멈추는 function 을 실행할 수 있었습니다.

```
public void onClick(View v) {
	// TODO Auto-generated method stub
	switch(v.getId()){
		case R.id.shoot_button:
			SimpleDateFormat formatter = new SimpleDateFormat("yyyy-MM-dd-hh-mm-ss");
			Log.d(TAG, formatter.format(new Date()));
			OutputStream out = null;
		try {
			File file = new File("/sdcard/DCIM", formatter.format(new Date()) + ".png");
			file.createNewFile();
			out = new FileOutputStream(file);
			Context c = this.getBaseContext();
			c.sendBroadcast(new Intent(Intent.ACTION_MEDIA_SCANNER_SCAN_FILE, Uri.fromFile(file)));
			bitmap.compress(Bitmap.CompressFormat.PNG,   80, out);
		} catch (FileNotFoundException e) {
			// TODO Auto-generated catch block
			e.printStackTrace();
		} catch (IOException e) {
			// TODO Auto-generated catch block
			e.printStackTrace();
		}
	}
}
```

onClick 함수입니다. 촬영버튼을 누를시, 이미지를 저장하는 로직을 만들었습니다.
저장된 이미지를 갤러리에서 확인할 수 있도록 카메라 이미지가 저장되는 경로인 DCIM 에 이미지를 저장하였습니다.

이때, application 실행 중에는 이미지가 disk 에 저장되지 않는 문제를 발견했고, 해당 문제가 이미지 파일을 memory 에 저장하고, disk 로 flush 하지 않기 때문이라는 것을 파악하였습니다.

이를 해결하기 위해 context 를 bradcast 하는 code 를 추가하였습니다.

```
Context c = this.getBaseContext();
c.sendBroadcast(new Intent(Intent.ACTION_MEDIA_SCANNER_SCAN_FILE, Uri.fromFile(file)));
```

```
class WebCamImage implements Runnable{
    Handler handler = new Handler();
    boolean hasImageAlready = (webCam.getDrawable() != null);
    @Override
    public void run() {
        // TODO Auto-generated method stub
        while(true) {
            // get Image
            Log.d(TAG, "Get Image");
            source = capture();
            if(source != null) {
                Log.d(TAG, "Image Not NULL : ");
                // decode Image
                ByteArrayOutputStream out = new ByteArrayOutputStream();
                YuvImage yuvImage = new YuvImage(source, ImageFormat.YUY2, 640, 480, null);
                yuvImage.compressToJpeg(new Rect(0, 0, 480, 360), 100, out);
                byte[] imageBytes = out.toByteArray();

                bitmap = BitmapFactory.decodeByteArray(imageBytes, 0, imageBytes.length);

                Log.d(TAG, bitmap.getConfig().name());

                if(bitmap != null) {
                    Log.d(TAG, "Bitmap Not NULL");
                    runOnUiThread(new Runnable() {
                        @Override
                        public void run() {
                            // TODO Auto-generated method stub
                            webCam.setImageBitmap(bitmap);
                            Log.d(TAG, "Set Image");
                        }
                    });
                }
            }
        }
    }
}
```

4. JNI 구현

```
#define LOG_TAG "jniTag"
#define LOGD(...)     __android_log_print(ANDROID_LOG_VERBOSE, LOG_TAG, __VA_ARGS__)
#define FPGA_STEP_MOTOR_DEVICE "/dev/fpga_step_motor"
#define VIDEO_DEVICE "/dev/video2"
#define VIDEO_WIDTH 640
#define VIDEO_HEIGHT 480
#define VIDEO_FORMAT V4L2_PIX_FMT_YUYV

jint JNICALL Java_com_example_androidex_MainActivity_moveMotorRight(JNIEnv *env, jobject this){
    int dev;
    unsigned char motor_state[3] = {1, 0, 50};

    LOGD("Start SUCCESS");
    dev = open(FPGA_STEP_MOTOR_DEVICE, O_WRONLY);
    LOGD("DEV : %d", dev);

    if (dev<0) {
        return -1;
    }

    write(dev,motor_state,3);
    close(dev);

    return 1;
}
```

Motor 를 구동하는 JNI 코드입니다. Right, left, stop 총 3 가지 함수가 존재하며, motr_stat 를 각 기능별로 다르게 하여, 모터가 지정된 동작을 수행할 수 있도록 하였습니다.

```
JNIEXPORT jbyteArray JNICALL Java_com_example_androidex_MainActivity_capture(JNIEnv *env, jobject
this){
    int video_fd;
    struct v4l2_format format;
    struct v4l2_requestbuffers reqbuf;
    struct v4l2_buffer buf;
    video_fd = open(VIDEO_DEVICE, O_RDWR);
    if (video_fd == -1) {
        LOGD("Failed to open video device");
        return NULL;
    }

    // Set video format
    format.type = V4L2_BUF_TYPE_VIDEO_CAPTURE;
    format.fmt.pix.width = VIDEO_WIDTH;
    format.fmt.pix.height = VIDEO_HEIGHT;
    format.fmt.pix.pixelformat = VIDEO_FORMAT;
    format.fmt.pix.field = V4L2_FIELD_NONE;
    if (ioctl(video_fd, VIDIOC_S_FMT, &format) == -1) {
        LOGD("Failed to set video format");
        close(video_fd);
        return NULL;
    }

    // Request video buffers
    reqbuf.type = V4L2_BUF_TYPE_VIDEO_CAPTURE;
    reqbuf.memory = V4L2_MEMORY_MMAP;
    reqbuf.count = 1;
    if (ioctl(video_fd, VIDIOC_REQBUFS, &reqbuf) == -1) {
        LOGD("Failed to request video buffers");
        close(video_fd);
        return NULL;
    }

    // Map video buffers
    struct v4l2_buffer buffer_info;
    buffer_info.type = V4L2_BUF_TYPE_VIDEO_CAPTURE;
    buffer_info.memory = V4L2_MEMORY_MMAP;
    buffer_info.index = 0;
    if (ioctl(video_fd, VIDIOC_QUERYBUF, &buffer_info) == -1) {
        LOGD("Failed to query video buffer");
        close(video_fd);
        return NULL;
    }
```

```
    void *buffer_start = mmap(NULL, buffer_info.length, PROT_READ | PROT_WRITE, MAP_SHARED,
video_fd, buffer_info.m.offset);
    if (buffer_start == MAP_FAILED) {
        LOGD("Failed to map video buffer");
        close(video_fd);
        return NULL;
    }

    // Start video streaming
    enum v4l2_buf_type type = V4L2_BUF_TYPE_VIDEO_CAPTURE;
    if (ioctl(video_fd, VIDIOC_STREAMON, &type) == -1) {
        LOGD("Failed to start video streaming");
        close(video_fd);
        return NULL;
    }

    // Capture image
    if (ioctl(video_fd, VIDIOC_QBUF, &buffer_info) == -1) {
        LOGD("Failed to enqueue video buffer");
        munmap(buffer_start, buffer_info.length);
        close(video_fd);
        return NULL;
    }

    if (ioctl(video_fd, VIDIOC_DQBUF, &buffer_info) == -1) {
        perror("Failed to dequeue video buffer");
        munmap(buffer_start, buffer_info.length);
        close(video_fd);
        return NULL;
    }

    jbyteArray bytes = (*env)->NewByteArray(env, buffer_info.length);
    (*env)->SetByteArrayRegion(env, bytes, 0, buffer_info.length, buffer_start);
     munmap(buffer_start, buffer_info.length);
     close(video_fd);
     return bytes;
}
```

Webcam 의 이미지를 java 로 보내기 위한 JNI 입니다. 삼성 Webcam 으로, General 한 환경에선 삼성에서 자체적으로 개발한 device driver 가 제공되기 때문에, 이미지 처리 format 이나 다른 문서가 존재하지 않았습니다. 관련 정보를 찾던 중 Video 4 Linux 를 지원한다는 자료를 확인하여, 해당 driver 로 구현하였습니다.

5. C Code 구현

```
#define MAX_BUTTON 9
#define MOTOR_ATTRIBUTE_ERROR_RANGE 4
#define FPGA_STEP_MOTOR_DEVICE "/dev/fpga_step_motor"

int move(int move, int right_left)
{
    int dev;
    unsigned char motor_state[3] = {move, right_left, 50};

    dev = open(FPGA_STEP_MOTOR_DEVICE, O_WRONLY);

    if (dev<0) {
        return -1;
    }

    write(dev,motor_state,3);
    close(dev);

    return 1;
}
```

Motor 를 움직이는 코드로, push switch 버튼의 입력 유무에 따라 동작되는 함수입니다.

```
int main(void)
{
	int i;
	int dev;
	int buff_size;
	int moved = 0;
	unsigned char push_sw_buff[MAX_BUTTON];
	dev = open("/dev/fpga_push_switch", O_RDWR);

	if (dev<0){
		printf("Device Open Error₩n");
		close(dev);
		return -1;
	}

	(void)signal(SIGINT, user_signal1);

	buff_size=sizeof(push_sw_buff);
	printf("Press <ctrl+c> to quit. ₩n");
	while(!quit){
		usleep(400000);
		read(dev, &push_sw_buff, buff_size);
		if(push_sw_buff[3] == 1 && push_sw_buff[5] == 0)
		{
			moved = 1;
			move(1, 1);
		}
		else if(push_sw_buff[3] == 0 && push_sw_buff[5] == 1)
		{
			moved = 1;
			move(1, 0);
		}
		else if(moved ==1)
		{
			move(0,0);
			moved = 0;
		}
	}
	close(dev);
}
```

Push switch 버튼의 입력을 받아 모터를 조작하는 코드입니다.

지속적으로 push switch 의 stat 를 읽어오고, 4,6 번 버튼에서 손이 떨어지는 경우 모터를 멈추도록 구현했습니다.

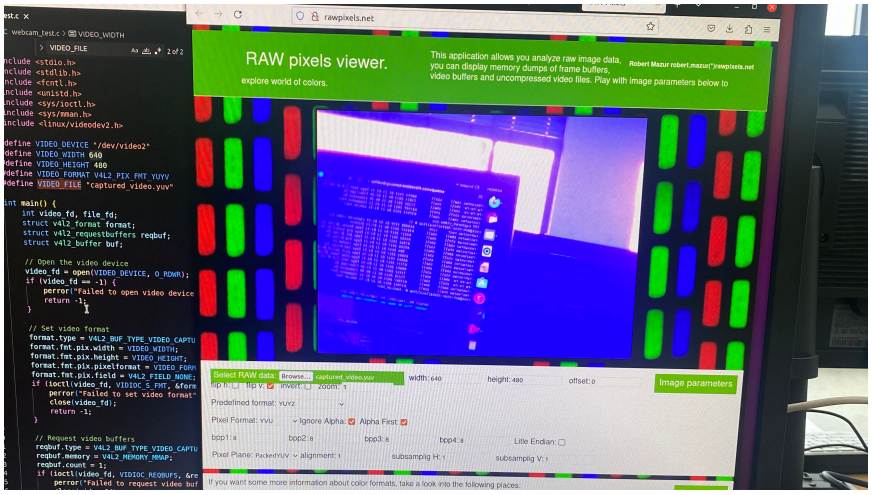
III. 추진 일정 및 개발 방법

가. 추진 일정

구분	기능	1주차		2주차		3주차	
test application	webcam test						
application	android mock-up						
	Motor JNI						
	Button motor control						
	Webcam JNI						

나. 개발 방법

개발해야 하는 기능들을 구분하여 waterfall 방식으로 개발하였습니다. 우선 프로젝트의 가능여부 판단을 위해 webcam 에서 image 를 캡쳐하는 code 를 구현한 후, 해당 이미지 파일을 여러 format 에 맞춰 확인했습니다.



이를 통해 해당 webcam 에서 이미지를 처리하는 format 을 확인할 수 있었습니다.

이후 Android Application 의 목업을 구현하고, JNI 와 물리버튼을 통해 Motor 를 조작하는 코드를 구현하였습니다. 마지막으로 웹캠 이미지를 해당 format 에 맞춰 android 에서 제공하는 JPG 형태로 변환하였고, 이를 android 의 imageView 로 보내 화면이 나오도록 구현하였습니다.

IV. 연구 결과

Linux 의 기본 Device driver 인 Video for Linux2 를 활용하여, Webcam 영상을 화면에 송출하고, 수업시간에 배운 내용들을 활용하여, 모터와 버튼을 통해 Webcam 을 조작하는 Application 과 Embedded Device 를 구현하였습니다.

Webcam Device 의 Memory 구조, Data 처리 방식 등 Device 에 대한 정보가 없어, 정확한 Format 을 찾고, 해당 format 을 android 에서 지원하는 format 으로 변환하는 과정에서 많은 Overhead 가 있었습니다.

향후 외부 Device 에서의 Control 이나, Device Driver Custom 등 다양한 것들을 추가적으로 진행하면 좋은 공부가 될 것 같습니다.