**Embedded  System Software Project**

**(과제 수행 결과 보고서)**

**과목명: [CSE4116] 임베디드시스템소프트웨어**

**담당교수: 서강대학교 컴퓨터공학과 박 성 용**

**학번 및 이름: 20141591, 최기용**

**개발기간: 2020. 06. 22. -2020. 06. 28.**

**최 종 보 고 서**

**I. 개발 목표**

- Android SDK,NDK를 이용한 커널과 연결되는 간단한 Android Application 제작

**II. 개발 범위 및 내용**

- 자신들이 설계한 개발 목표를 달성하기 위하여 어떠한 내용의 개발을 수행할 지 그 범위와 개발 내용을 기술할 것.

**가. 개발 범위**

학기 전반적으로 배운 내용들을 이용해, 시간(분 단위)을 설정하면 해당 시간 후 알람이 울리게 되는 프로그램 제작. 이 때, Android NDK, SDK, Device Driver Kernel programming 등의 내용이 모두 포함된다.

**나. 개발 내용**

1) Device Driver

두 가지 Device driver를 제작했다. 하나는 App의 주 기능이되는 alarm device driver이다. Ioctl, write 두가지를 통해 호출 되는데 메인 thread에서는 ioctl을 통해 입력된 시간을 함께 정보로 넘겨준다. COMMAND가 넘어오게 되면 매초 timer를 등록시키며 FND에 경과한 시간을 띄운다. 백그라운드에서 돌아가는 thread는 write를 호출하게 되며 들어올 경우 정해진 시간이 지나 timer가 delete될 때까지 sleep하게 된다.

또 하나는, buzzer driver로 설정한 시간이 경과하게 되면 보드의 buzzer음이 울리도록 한다.

2) JNI

Java Android Application에서 C언어로 작성된 코드를 이용할 수 있도록 하는 JNI 코드를 제작하였다. 위에서 말한 두가지 DEVICE를 open write 등 할 수 있도록 만드는 함수들을 제작하였다.

jint JNICALL Java\_org\_example\_ndk\_NDKExam\_devopen(JNIEnv \* env,jobject this)

void JNICALL Java\_org\_example\_ndk\_NDKExam\_devwrite(JNIEnv \* env,jobject this,jint fd,jcharArray buf)

void JNICALL Java\_org\_example\_ndk\_NDKExam\_devioctl(JNIEnv \* env,jobject this,jint fd,jint minute)

void JNICALL Java\_org\_example\_ndk\_NDKExam\_devclose(JNIEnv\* env,jobject this,jint fd)

jint JNICALL Java\_org\_example\_ndk\_NDKExam\_buzopen(JNIEnv\* env, jobject this)

void JNICALL Java\_org\_example\_ndk\_NDKExam\_buzwrite(JNIEnv\* env, jobject this,jint fd)

void JNICALL Java\_org\_example\_ndk\_NDKExam\_buzclose(JNIEnv\* env, jobject this,jint fd)

3) Java Application

알람 어플리케이션이다. editText view에 분 단위의 시간을 입력할 수 있다. 입력 한 후 Setting 버튼을 클릭하면 알람 타이머는 시작하게된다. 이 때, FND에는 초단위로 시간이 표현된다. 설정 시간이 흐르면, Buzzer를 open하여, 알림이 울린다. Check 버튼을 누르게 되면 현재까지 앱 실행 후 설정했던 시간의 합과, file에 저장되어있는 지금까지 setting했던 총 시간을 표현한다.

**III. 추진 일정 및 개발 방법**

- 자신들이 설정한 개발 목표를 달성하기 위한 개발 일정을 설정하고, 각 요소 문제를 해결하기 위해서 어떤 방법을 사용할 지 기술할 것.

1. **추진 일정**

**06.22 ~ 06.23 : 아이디어 구체화**

**06.23 ~ 06.26 : 프로젝트 코드 복습 및 SDK, NDK 강의,실습자료 복습**

**06.26 ~ 06.28 : 코드 구현**

**06.28 ~ 06.29 : 보고서 및 발표자료 작성**

**나. 개발 방법**

1. Device Driver
2. Alarm driver

어플리케이션 메인 스레드는 ioctl을 통해 driver와 통신한다. 분 단위의 입력이 bytestream으로 들어오고, 이를 이용해 timer를 등록하고 주어진 시간에 도달하면 timer를 delete한다.

백그라운드 스레드는 write를 호출하여 kernel에 들어오게 되고, write에서는 이 thread를 wait queue에 넣는다. 해당 시간이 지나면 wake up 된다.

1. Buzzer driver

Background thread에서 open된다. Alarm device가 시간이 다 되어 wake up 되면 실행되는데, 이때 계속해서 buzzer음이 울리게 한다. 멈추기 위해서는 Interrupt Handler를 이용했다. Home button을 click 하게 되면, flag를 주어 멈추게 한다.

1. JNI

Java로 짜여진 application이 C로 짜여진 device driver를 호출 할 수 있게 하도록 한다.

이 때, dev\_write에서는 Java field 변수를 제어한다. Alarm device가 종료되면, 흘렀던 시간을 더하여 total 시간 변수를 update한다. buzwrite에서도 flag를 통해, Home 버튼 Interrupt Handler 입력이 들어오면, 알람음이 멈추게 한다.

1. Java Application

Java로 짜여진 application으로, 화면에 띄워질 xml과 textView 설정, 디바이스 드라이버 호출 등을 담당한다.

**IV. 연구 결과**

- 최종 연구 개발 결과를 자유롭게 기술할 것.

1) Device driver

1. Alarm driver

dev\_driver.h

#define DEVICE\_NAME "dev\_driver"

#define MAJOR\_NUM 242 //Major number

/\*

 \* FND PHYSICAL ADDRESS

 \*/

#define FND\_PHY\_ADDRESS 0x08000004

#define SET\_OPTION \_IOW(MAJOR\_NUM,0,unsigned int) //Encode SET\_OPTION by \_IOW MACRO

#define COMMAND \_IO(MAJOR\_NUM,1) //Encode COMMAND by \_IO MACRO

Major number 와 device name은 각각 242, dev\_driver

IOCTL위한 command setting

Dev\_driver.c

wait\_queue\_head\_t wq\_write;

DECLARE\_WAIT\_QUEUE\_HEAD(wq\_write);

Background thread를 wait 시키기 위한 구조

long dev\_ioctl(struct file \* file, unsigned int cmd, unsigned long arg){

    unsigned int byte\_stream;

    //execute by COMMAND

    switch(cmd){

        //first ioctl (SET\_OPTION)

        case SET\_OPTION:

            //Receive Argument(byte\_stream in jni)

            if (copy\_from\_user(&byte\_stream,(unsigned int\*)arg,sizeof(unsigned int))){

                return -1;

            }

            //Setting input time

            minute = byte\_stream & 0x000000FF;

            break;

        //SECOND IOCTL (COMMAND)

        case COMMAND:

            //setting timer device environment & add\_timer

            timer.expires = get\_jiffies\_64() + HZ;

            timer.function = stopwatch\_function;

            add\_timer(&timer);

            break;

        default:

            break;

    }

    return 0;

}

Ioctl 구조

Byte stream을 통해 minute를 설정하고, timer를 등록시킨다.

static void dev\_write(struct file \*filp, const char \*buf, size\_t count, loff\_t \*f\_pos){

    printk("Sleep On\n");

    //thread program will be slept(in wait queue)

    interruptible\_sleep\_on(&wq\_write);

}

dev\_write Background thread가 호출 하는 함수로, 호출 시 sleep queue에 들어가게된다.

void stopwatch\_function(unsigned long timeout){

    fnd\_execute();

    //add timer every sec

    timer.expires = get\_jiffies\_64() + HZ;

    timer.function = stopwatch\_function;

    add\_timer(&timer);

    if(fnd\_min == minute){

        del\_timer(&timer);

        outw(0,(unsigned int)iom\_fpga\_fnd\_addr);

        \_\_wake\_up(&wq\_write,1,1,NULL);

        fnd\_min =0;

        fnd\_sec = 0;

    }

}

Alarm timer function 분 단위 입력이 minute에 다다르면, fnd 초기화 하고, background queue를 wake up 시킨다.

2. Buzzer driver

#define IOM\_BUZZER\_MAJOR 264        // ioboard buzzer device major number

#define IOM\_BUZZER\_NAME "fpga\_buzzer"       // ioboard buzzer device name

#define IOM\_BUZZER\_ADDRESS 0x08000070 // pysical address

Major number와 device name은 각각 264 , fpga\_buzzer

irqreturn\_t home\_inter\_handler(int irq, void \* dev\_id,struct pt\_regs\* reg){

    printk(KERN\_ALERT "home interrupt!!! = %x\n", gpio\_get\_value(IMX\_GPIO\_NR(1, 11)));

    if(buz\_flag == 0){

        buz\_flag = 1;

    }

    return IRQ\_HANDLED;

}

Interrupt handler를 통해 Buzzer 제어. Home button key가 눌리게 되면 buz\_flag를 on

ssize\_t iom\_buzzer\_write(struct file \*inode, const char \*gdata, size\_t length, loff\_t \*off\_what)

{

    unsigned char value;

    unsigned short int \_s\_value;

    const char \*tmp = gdata;

    if (copy\_from\_user(&value, tmp, 1))

        return -EFAULT;

    \_s\_value = value & 0xF;

    outw(\_s\_value,(unsigned int)iom\_fpga\_buzzer\_addr);

    //if flag on

    //return 2 and device close

    if (buz\_flag == 1){

        buz\_flag =0;

        return 2;

    }

    return length;

}

Buzzer write 호출되면 buzzer음을 발생시키고, interrupt가 들어오기 전까지는 length = 1 return Home button interrupt가 발생하여 flag가 on되면 2를 return

버튼에 따라, Stopwatch를 통제하기 위한 Boolean type의 flag 들과, interrupt handler function 명시 fnd\_min, fnd\_sec 변수 통해, stopwatch에 표시할 시간 제어

2) JNI

void JNICALL Java\_org\_example\_ndk\_NDKExam\_devwrite(JNIEnv \* env,jobject this,jint fd,jcharArray buf){

    LOGV("check !");

    write(fd,buf,2);

    LOGV("check 2 !");

    jclass thisClass = (\*env)->GetObjectClass(env,this);

    jfieldID fidnumber = (\*env)->GetFieldID(env,thisClass,"total\_sleep","I");

    jint number = (\*env) -> GetIntField(env,this,fidnumber);

    number = total\_sleep;

    (\*env)->SetIntField(env,this,fidnumber,number);

}

dev\_write 함수로 Background thread가 호출한다.

Write()하여 sleep queue에 들어가게 되는데, 시간이 지나면 깨어나서 application field variable인 total\_sleep에 JNI file에 있는 total\_sleep을 넣어준다.

void JNICALL Java\_org\_example\_ndk\_NDKExam\_devioctl(JNIEnv \* env,jobject this,jint fd,jint minute){

    bytestream = 0;

    bytestream = bytestream | minute;

    total\_sleep += minute;

    ioctl(fd,SET\_OPTION,&bytestream);

    ioctl(fd,COMMAND);

}

Ioctl을 통해 alarm timer를 동작시키는 함수로, bytestream을 통해 설정한 시간을 전달하고, JNI file 변수인 total\_sleep에 설정 시간을 더해준다.

void JNICALL Java\_org\_example\_ndk\_NDKExam\_buzwrite(JNIEnv\* env, jobject this,jint fd){

    int retval;

    int state =0;

    int data;

    LOGV("Buzzer write correct!");

    while(retval != 2){

        if(state!=0){

            state = 0;

            data =1;

            retval = write(fd,&data,1);

            if(retval<0){

                LOGV("Buzzer write error!");

                exit(-1);

            }

        }else{

            state =1;

            data =0;

            retval = write(fd,&data,1);

            if(retval <0){

                LOGV("Buzzer write error!");

                exit(-1);

            }

        }

        LOGV("check ret : ");

        sleep(1);

    }

}

Buzzer device write함수로, 설정시간에 도달하면 동작하게 된다. While문 내에서 계속해서 beep 음이 울리도록 하며, Home button interrupt가 들어오게 되면 return 값으로 2를 받아 탈출하게 된다.

3)Java Android Application

public native int devopen();

    public native void devioctl(int fd,int minute);

    public native void devclose(int fd);

    public native void devwrite(int fd, char [] buf);

    public native int buzopen();

    public native void buzwrite(int fd);

    public native void buzclose(int fd);

native Function 선언

try{

            BufferedReader br = new BufferedReader(new FileReader(getFilesDir().toString()+"/minutes.txt"));

            String readStr ="";

            String str = null;

            str = br.readLine();

            readStr += str;

            total\_sleep2 = Integer.parseInt(readStr);

            br.close();

        }catch(FileNotFoundException e){

            total\_sleep2 =0;

            e.printStackTrace();

        }catch(IOException e){

            e.printStackTrace();

        }

처음 App 실행 시 file을 읽어와 앱 구현 이후 총 설정 시간 저장

ltn = new OnClickListener(){

            public void onClick(View v){

                String tmp = data.getText().toString();

                minute = Integer.parseInt(tmp); // receive editText content

                fd = devopen();

                devioctl(fd,minute); // ioctl communication

                mThread = new BackThread(mHandler);

                mThread.setDaemon(true);

                mThread.start(); //thread run

                tv.setText("Sleep During "+ minute+ "minutes..");

                devclose(fd);

            }

        };

        /\*

         \* If click this button,

         \* User will check How long alarm using

         \*/

        btn.setOnClickListener(ltn);

edit text에 값 입력 후 button 클릭 시 Alarm device 실행 과 종료 시점을 알기 위한 Background thread 실행

btn.setOnClickListener(ltn);

        ltn2 = new OnClickListener(){

            public void onClick(View v){

                tv.setText("You sleep today " + total\_sleep+" minutes!"+"total "+ total\_sleep2+"!");

            }

        };

        btn2.setOnClickListener(ltn2);

check 버튼 클릭 시 앱 실행 후 총 수면 시간과, 앱 구현 후 총 설정 시간 출력

Handler mHandler = new Handler(){

        public void handleMessage(Message msg){

            if(msg.what ==0){

                tv.setText("Wake Up!!");

            }

        }

    };

    /\*

     \* Thread

     \* Using dev\_write

     \* When Alarm driver executing, this Thread will sleep

     \* And, wake\_up later setting time, then buzzer driver will start

     \*/

    class BackThread extends Thread{

        int mBackValue = 0;

        Handler sHandler;

        BackThread(Handler handler){

            sHandler = handler;

        }

        public void run(){

            //tv.setText("Sleep During "+ minute+ "minutes..");

            try{

                devwrite(fd,buf);

                Message msg = Message.obtain();

                msg.what=0;

                sHandler.sendMessage(msg);

                buzfd = buzopen();

                buzwrite(buzfd);

                buzclose(buzfd);

                try{

                    total\_sleep2 += minute;

                    BufferedWriter bw = new BufferedWriter(new FileWriter(getFilesDir().toString()+"/minutes.txt",false));

                    bw.write(Integer.toString(total\_sleep2));

                    bw.close();

                }catch(IOException e){

                    e.printStackTrace();

                }

            }catch(Exception e){

                ;

            }

            //tv.setText("Wake Up!!");

        }

    }

Background Thread 구현 부분, Alarm device가 실행하게 되면 thread가 run 하기 시작하는데, write를 호출해 sleep 하게 된다. 시간이 다 되면 wake up 후 buzzer device를 실행 시키고, text에 Wake up을 띄우기 위해서 Handler를 사용했다. 또한, FileWriter를 이용해서 File에 총 설정 시간을 덮어씌우는 방식으로 update 시킨다.

**V. 기타**

-  본 설계 프로젝트를 수행하면서 느낀 점을 요약하여 기술하라. 내용은 어떤 것이든 상관이 없으며, 본 프로젝트에 대한 문제점 제시 및 제안을 포함하여 자유롭게 기술할 것.

본 프로젝트는 주제가 정해지지 않은 자유 주제 프로젝트이다. 새로운 기능을 구현하기 보다는 지금까지 구현해왔던 프로젝트와 실습 내용들을 다시 한번 사용해보고, 연결 시킬 수 있는 프로젝트 내용을 선택했다. 다만, 실습으로만 진행하고 과제로는 해보지 못한 Android Programming JNI,SDK 구현 부분이 다소 생소해서 강의 내용을 다시 읽고 따로 공부하는데 시간이 많이 걸렸다. Kernel timer 기능을 이용해서, Alarm device를 구현하였고, Interrupt Handler 기능을 이용해서 Buzzer device를 구현하였다. Device와 통신 하는 부분도 Project2에서 사용했던 ioctl 기능을 사용해보았다. 해당 device들을 APP에서 제어할 때 JNI를 사용하여 C코드를 호출할 수 있게했다. 이 외에도 Java Android application에서 File Read/Write기능과, Background Thread 기능 등 실습자료에 기술 되어있던 기능을 구현해보았다.

프로젝트를 진행하면서 자칫 까먹을 수 있었던 지난 프로젝트 내용을 복습할 수 있었고, 처음 경험해보는 android programming을 통해 프로젝트 진행 중에도 계속 새로운 아이디어가 생각나, 직접 짠 코드가 실생활에서 많이 사용되는 android application이 동작하는 것을 보니, 다양한 기능을 더 공부해보고 싶다는 생각이 들었다. 또한, 깊지는 않더라도, 이전 수강 과목들인 OS, JAVA 등의 과목을 한번에 복습하고, 실제 동작하는 것을 보니 와 닿아서 이해에도 도움이 되었다.