Mobile Programming

Final Project

32122608 Nam Ho Kang

32112192 Sang Woo Kim

Link to a video demonstration: https://www.youtube.com/watch?v=AHbng-iVVec

1. Project Objective

To build a program that runs on HBE-SM5-S4210. The program runs not only in the application framework level, but also in Linux kernel level. Among the features of HBE-SM5-S4210, we are going to use:

- Keypad
- Dip Switch
- OLED
- Full Color LED
- LED

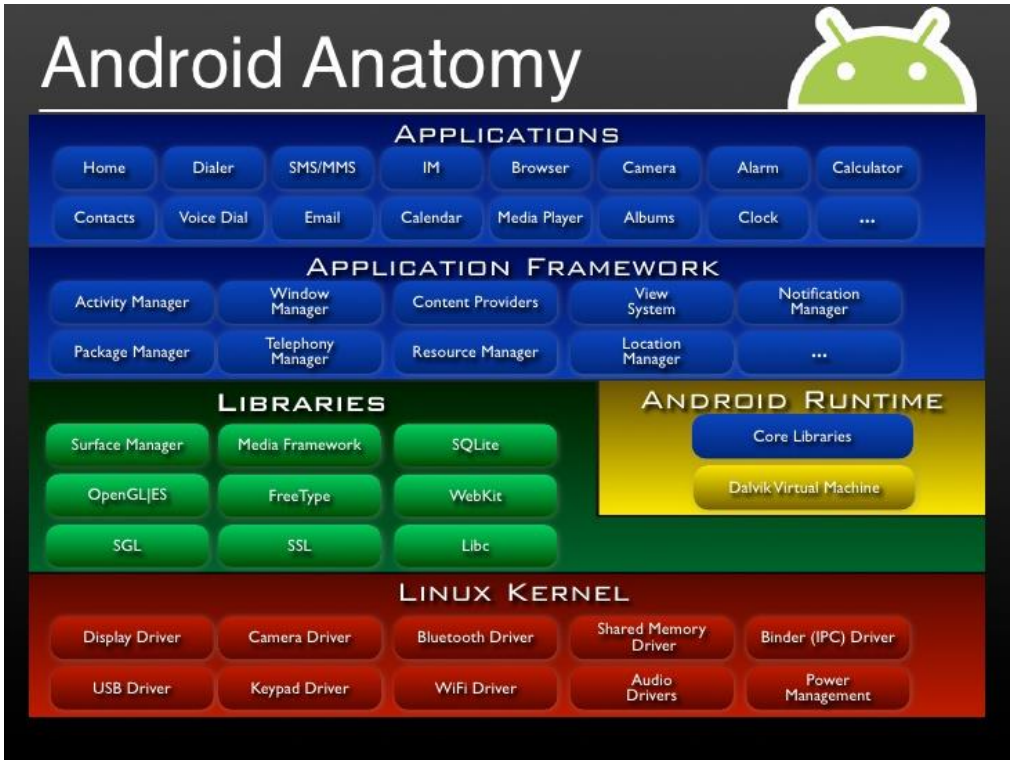
By using these features, we will be able to build a program that demonstrates and makes use of things that we will be unable to with an ordinary Android device.

2. Motivation

Our program's name is Launchpad. A original Launchpad is a popular musical device that can be used to make music with different beats and sounds. Our program basically copies that feature. The user will be able to choose different background music with touching button, then the user can add different beats with keypads. Dip Switch will be used to change volume of the music or beats. LED shows witch music is being played; upon pressing a button, a corresponding LED will turn on. OLED screen displays two kinds of pictures. One is when the music is not playing, and the another one is when it is playing. Full Color LED is also used in the same situation as OLED. When the music is on, the LEDs make different colors randomly, then when it is off, the LEDs turn off.

I was not able to make use of 7 Segment and textLCD. For some reason, the 7 Segment did not display correct numbers, but only displayed an inversed 7. The textLCD did work, but had errors of overlaying texts when I used a method to display certain texts.

3. Prior Knowledge



To learn about Android, these five layers must be known.

- Application Layer

  Application that users use. Users cannot access lower levels if the application does not allow it. Even if it does, direct access is not granted, the access goes through application framework.

- Application Framework

  In the application framework layer, all the managers are located. The managers include activity manager, package manager, etc. These managers allow the use of layers below.

- Android Runtime

  In Android Runtime layer, Core Libraries and Dalvik Virtual Machine (DVM) are included. DVM is what allows different applications to launch. Java programs are run on virtual machines, and one program needs a virtual machine, or it would be too slow and may cause different problems. One DVM is created upon turning the device on, and when different applications are launched, the DVM copies itself and runs the application. In this way the virtual machine does not need to be created every single time an application is launched

- Libraries

  Libraries are literally the libraries. It consists of different kinds of libraries that the Android device needs to operate.

- Linux Kernel

  A kernel is a set of drivers. Because there are different modules attached to different Android devices, different settings are needed. What indicates what modules are to be used, and allowing the modules to be used is a driver. Without a kernel, or even with a kernel that is for a different Android device, the Android cannot be launched.

Java Native Interface (JNI) is another important thing to know. Android applications are built based on Java. However, what controls the modules attached is not. It does not understand a language like java, but only C. Thus, the developers will have to work with C for the modules, and Java for the application. JNI is what allows this to happen. The developer has to make libraries written in C language, and declare the functions in Java as "native *returntype* function(*parameter*)". By using JNI, the developer is able to successfully integrate the features of the modules inside the application.

4. Program Structure

- MainActivity

```
private int BGMcheck[] = new int[bgmNum];

int LED = 0;
int m_index = -1;
@Override
public void onClick(View v) {
    switch (v.getId()){
        case R.id.bgm_button1:
            m_index = 0;
            if (BGMcheck[m_index] != 1) {
                tflag |= 0b10;
                LED |= 0x01;
                musicManager.play(R.raw.bgm_dark_hiphop, m_index);
                BGMcheck[m_index] = 1;
            }else{
                LED &= ~(0x01);
                tflag &= 0b01;
                musicManager.stop(m_index);
                BGMcheck[m_index] = 0;
            }
            break;
```

This is how different music are played. BGMcheck array is used for checking which button started the music. Tflag checks if music is being played or not, int LED is used to light on LED when a button is pressed. Keypads are also used in a similar way.

```
private class DotThread extends Thread{
    int dot = 1;
    public void run(){
        try{
            while(true){
                if (tflag != preflag) {
                    if (tflag > 0){
                        dot = 2;
                        preflag = tflag;
                    }
                    if (tflag < 1){
                        dot = 1;
                        preflag = 0;
                    }
                }
                JNI.displayDot(dot);
                Thread.sleep(500);
```

DotThread is what controls the Dot Matrix. It checks whether music is being played or not, and sends corresponding number to change text.

```
private class LEDThread extends Thread{
    int init = 0;
    public void run(){
        try{
            while(true) {
                if (musicOn()) {
                    for (int i = 6; i < 10; i++){
                        int color[] = getRandomRGB();
                        JNI.displayOLED(i, color[0], color[1], color[2]);
                    }
                    JNI.displayImage(2);
                    init = 1;
                }else {
                    if (init == 1)
                        JNI.displayOLED(5, 0, 0, 0);
                    JNI.displayImage(1);
                    init = 0;
                }
                Thread.sleep(500);
            }
        }
```

LEDthread contols full color LEDs. It also checks if the music is being played or not, and displays random colors or turns the LEDs off.

```
private class Mythread extends Thread{
    public void run(){
        try{
            while(true){
                int BGMvol = JNI.VolBGM();
                int Beatvol = JNI.VolBeat();
                if (BGMvol != preBGM)
                    musicManager.changeVolume(BGMvol, BGMcheck, bgmNum);
                if (Beatvol != preBeat)
                    beatManager.changeVolume(Beatvol,Beatcheck, beatNum);
                preBeat = Beatvol;
                preBGM = BGMvol;
                Thread.sleep(500);
```

Mythread is used for Dip Switch. It receives current Dip Switch value from VolBGM method, and if it is changed, the volume is changed.

- jnicombine

```
public class jniCombine {
    public DipSW dipSW;
    private LED led;
    private DotMatrix dotMatrix;
    private FullColor fullColor;
    private Image image;
    //private Segment segment;
    //private TextLCD textLCD;
```

jnicombine is what declares and uses JNI features.

All the methods are called through jnicombine.

- DotMatrix

```
void display(String data){
    int i, j, ch = 0;
    char buf[] = data.toCharArray();
    String result = "00000000000000000000";
    for (i = 0; i < data.length(); i++){
        ch = (int) buf[i];
        if (ch < 32 || ch > 126){
            break;
        }
        ch -= 0x20;
        for (j = 0; j < 5; j++) {
            String str = Integer.toHexString(font[ch][j]);
            if (str.length() < 2) {
                result += "0";
            }
            result += str;
        }
        result += "00";
    }
    result += "00000000000000000000";
    for (i = 0; i < (result.length() - 18) / 2; i++) {
        for (j = 0; j < 10; j++) {
            Control(result.substring(2 * i, 2 * i + 20));
        }
    }
    Control("00000000000000000000");

}
```

Display function receives a string as a parameter. According to each characters, which dots should be turned on is determined.

Font[][] array contains hexadecimal values for all ASCII codes from 32 ~ 126.

- DipSW

```
public int UpdateValue(){
    int newValue = GetValue();
    if ((newValue & 0x1) == 0x1) {
        return 1;
    }
    if ((newValue & 0x2) == 0x2) {
        return 2;
    }
    if ((newValue & 0x4) == 0x4) {
        return 3;
    }
    if ((newValue & 0x8) == 0x8) {
        return 4;
    }
    if ((newValue & 0x10) == 0x10)
        return 5;
    }
    if ((newValue & 0x20) == 0x20)
        return 6;
    }
    if ((newValue & 0x40) == 0x40)
        return 7;
    }
    if ((newValue & 0x80) == 0x80)
        return 8;
    }
    return 9;
}
```

GetValue reads from DipSW module and returns the value (i.e. which switch is on). The priority is given from the left, meaning that if the first switch is on, all the other switches are not taken into conditions.

- LED

```
JNIEXPORT jboolean JNICALL Java_jniActivity_LED_TurnOn
        (JNIEnv *env, jobject obj, jint data) {
    if (fd < 0)
        fd = open("/dev/fpga_led", O_RDWR);
    if (fd < 0)
        return -errno;
    data &= 0xff;
    write(fd, &data, 4);
    return 1;
}
```

LED is probably the simplest. It just writes a value to the module.

- FullColorLED

```
JNIEXPORT jint JNICALL Java_jniActivity_FullColor_Control
        (JNIEnv *env, jobject obj, jint led_num, jint val1, jint val2, jint val3){
    int ret;
    char buf[3];
    ret = (int) led_num;
    switch (ret){
        case FULL_LED1:
            ioctl(fd, FULL_LED1);
            break;
        case FULL_LED2:
            ioctl(fd, FULL_LED2);
            break;
        case FULL_LED3:
            ioctl(fd, FULL_LED3);
            break;
        case FULL_LED4:
            ioctl(fd, FULL_LED4);
            break;
        case ALL_LED:
            ioctl(fd, ALL_LED);
            break;
    }
    buf[0] = val1;
    buf[1] = val2;
    buf[2] = val3;

    if (fd < 0) fd = open("/dev/fpga_fullcolorled", O_WRONLY);
    if (fd < 0) return -1;
    write(fd, buf, 3);

    return ret;
}
```

This screenshot is from the Full Color LED's JNI library. Upon receiving 3 values and number of LED, the colors and LED number is written to the LED module.

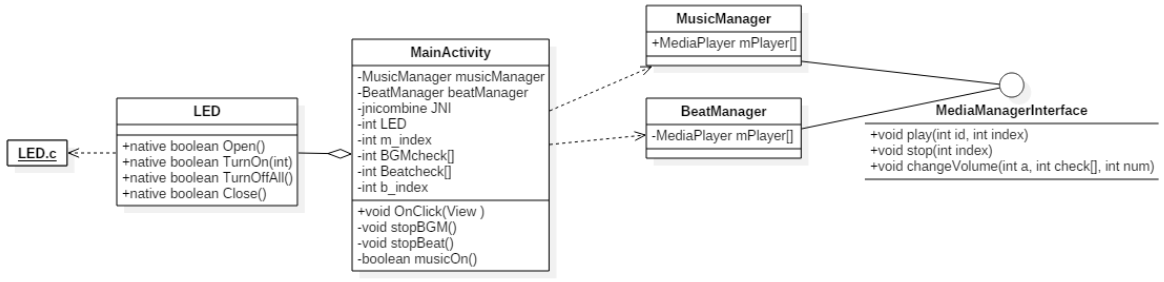
- OLED

```
public boolean UpdateValue(int n) {
    if (curimg == n)
        return false;
    java.io.InputStream is0 = context.getResources().openRawResource(R.raw.launchpad);
    java.io.InputStream is1 = context.getResources().openRawResource(R.raw.dropthebeat);

    switch (n) {
        case 1:
            bm = BitmapFactory.decodeStream(is0);
            break;
        case 2:
            bm = BitmapFactory.decodeStream(is1);
            break;
    }
    curimg = n;

    int width = bm.getWidth();
    int height = bm.getHeight();
    int[] pixels = new int[width * height];
    bm.getPixels(pixels, 0, width, 0, 0, width, height);
    Control(pixels);
    return true;
}
```

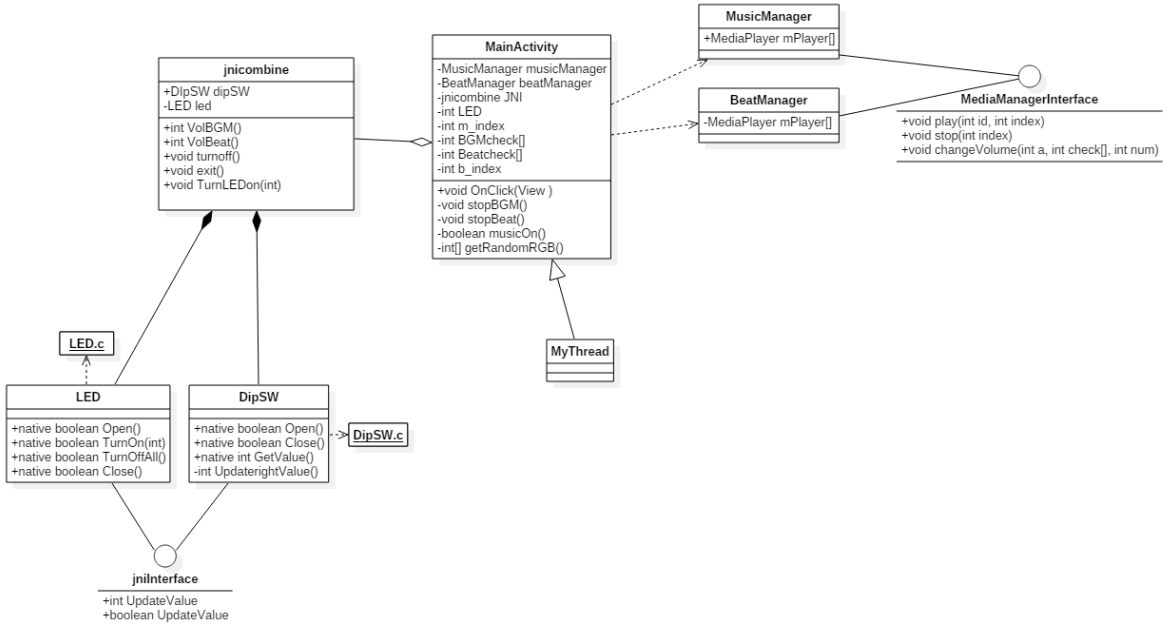
Bm is a bitmap variable. A number is used as a parameter in MainActivity, and a corresponding picture is chosen to be displayed.

5. UML

First Design



Second Design



Final Design