Android Application Dev Example
GPIO control LED light

1. Initialize GPIO pins at Linux kernel configue file.

Step 1:

Take a look at sys_config.fex file, check which GPIO pins have been initialized.

```
cyx@Server-102:~/a20_box/A20-420-V12$
cyx@Server-102:~/a20_box/A20-420-V12$
cyx@Server-102:~/a20_box/A20-420-V12$ vi lichee/tools/pack/chips/sun7i/configs/android/wing-mbox203/sys_config.fex
```

```
#cyx@Server-102:~/a20_box/A20-420-V12$ cat lichee/tools/pack/chips/sun7i/configs/android/ cubiexxx /sys_config.fex
```

(According to your device name and path. Or type adb shell on your PC)

(device name)

Step 2: Find [gpio_para] config section:

```
;------------------------------------------------------------------------------------
;gpio configuration
;gpio_pin_5 ---> 3g_onoff_pin
;gpio_pin_6 ---> 3g_vbat_pin
;------------------------------------------------------------------------------------
[gpio_para]
gpio_used               = 1
gpio_num                = 8
gpio_pin_1              = port:PH20<1><default><default><1>
gpio_pin_2              = port:PH10<1><default><default><0>
gpio_pin_3              = port:PB03<1><default><default><1>
gpio_pin_4              = port:PI00<1><default><default><0>
gpio_pin_5              = port:PG10<1><default><default><1>
gpio_pin_6              = port:PG11<1><default><default><1>
gpio_pin_7              = port:PI00<1><default><default><1>
gpio_pin_8              = port:PI01<1><default><default><1>
;gpio_pin_9              = port:PI02<1><default><default><1>
```

We can see there are 8 GPIO pins been initialized.

Let me give an example here with the first initialized pin (gpio_pin_1(PH20)), making it as LED blinking light.

Note: For the definition of GPIO configuration, please view the page http://linux-sunxi.org/GPIO

Alright, based on the configuration of GPIO initialization, next we open the Eclipse develop program, then import javalib.jar file according to your platform.

1. The location of javalib.jar:

Android4.2/out/host/common/obj/JAVA_LIBRARIES/layoutlib_intermediates/javalib.jar

2. Import

For example, your project name is “hello”, then go to “hello” - > Build Path -> Configure Build Path -> Libraries -> Add External JARs, then choose javalib.jar -> OK

3. Add source code into your APK, please refer to the following example.

JAVA source code:

```
package com.smdt.test.led;

import android.app.Activity;
import com.smdt.test.led.R;
import android.os.Bundle;
import android.util.Log;
import android.os.Gpio;

public class LED_Test extends Activity {

    private boolean ThreadExit = true;
    private static int LedTime = 500;
    Thread browseThread = null;

    private void ctrlLedLight(boolean enable) {
        try {
            if (enable) {
                Log.d(TAG, "ctrlLedLight open led light");
                Gpio.writeGpio('h', 20, 1);//
            } else {
                Log.d(TAG, "ctrlLedLight close led light");
                Gpio.writeGpio('h', 20, 0);//
            }

    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

```
private void startCtrlLedThread() {
    browseThread = new Thread() {
        public void run() {
            try {
                while (ThreadExit) {
                    if (LedTime <= 0) {
                        LedTime = 500;
                    }
                    ctrlLedLight(false);

                    Thread.sleep(LedTime);
                    ctrlLedLight(true);
                    Thread.sleep(LedTime);

                    }
                } catch (Exception e) {
                    e.printStackTrace();
                }
            }
        };
        browseThread.start();
    }

    public String TAG = "Led Test";

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        startCtrlLedThread();
    }
```

```
    protected void onDestroy() {
        // mHandler.removeCallbacks(mRunnable);
        System.out.println("----------onDestroy------");
        ThreadExit = false;
        super.onDestroy();
    };
}
```

AndroidManifest.xml 布局文件

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.smdt.test.led"
    android:versionCode="1"
    android:versionName="1.0" >
    <uses-sdk
        android:minSdkVersion="10"
        android:targetSdkVersion="10" />
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
    <uses-permission
android:name="android.permission.RECEIVE_BOOT_COMPLETED"></uses-permission>
    <application
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name">

        <activity
            android:name="com.smdt.test.led.LED_Test"
            android:label="@string/app_name" >

            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```

Note:

a. import android.os.Gpio; is for import the supported controller GPIO pins.

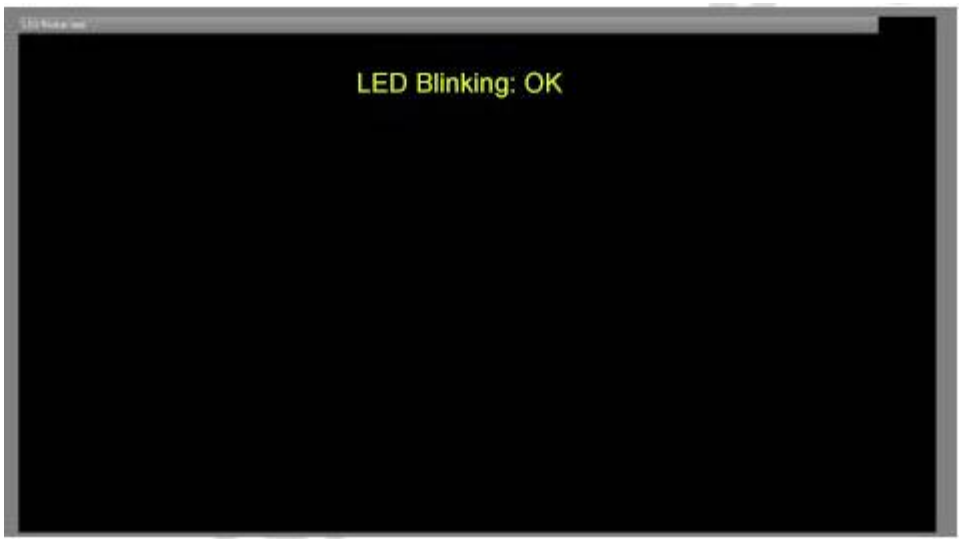
b. Gpio.writeGpio('h', 20, 1);

There are three parameters been set,

1. **'h'** means the GPIO pins group.

2. **20** means the exact IO which the first parameter point to.
3. **1** means this PIN is set to HIGH. If **0, it** means PIN is set to LOW.

4. The GPIO pin that Application is trying to control, must be initialized by sys_config.fex, so that this pin is able to be used.



Application UI

5. Conclusion

This is just a simple demo of how to control GPIO pin, you can make more functions based on the example above.