标识：AEX-ANDROIDEX-SDK-GUIDE

版本：V1.1

**AndroidEx SDK**

**开发指南**



深圳市安卓工控设备有限公司

2016年10月

# 引言

## 标识

项目名称：ANDROIDEX SDK开发指南

文档名称：ANDROIDEX SDK开发指南

文档简称：**开发指南**

文档标识号：AEX-KK3X-SDK-GUIDE-V11

## AndroidEx SDK简介

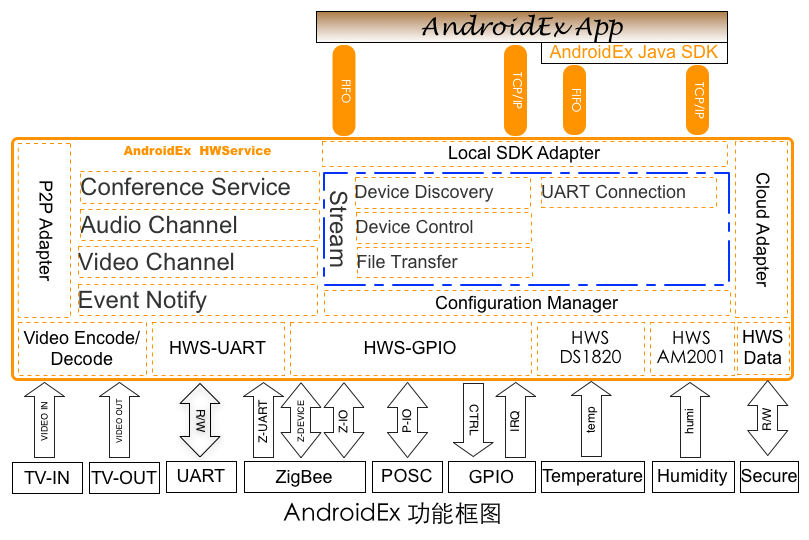
**AndroidEx SDK (简称AexSDK)是应用于KK3X系列主板的软件开发包。**

**KK3X系列主板是建立在ARM Cortex系列CPU内核的基础上，目前有使用A10、A20、ATM7059、S500、AM335X等主控芯片的主板。使用这些主控芯片大幅提升网络高清SOC 的应用水平，使得电子多媒体产品的用户体验得到大幅提升和改善。高度集成的新一代处理器将以领先的网络视频性能和高性价比特性，广泛应用于平板电脑、家庭智能高清网络播放器、汽车多媒体控制器、安防监控网络、工业控制以及各种随身多媒体娱乐产品等新兴市场。**

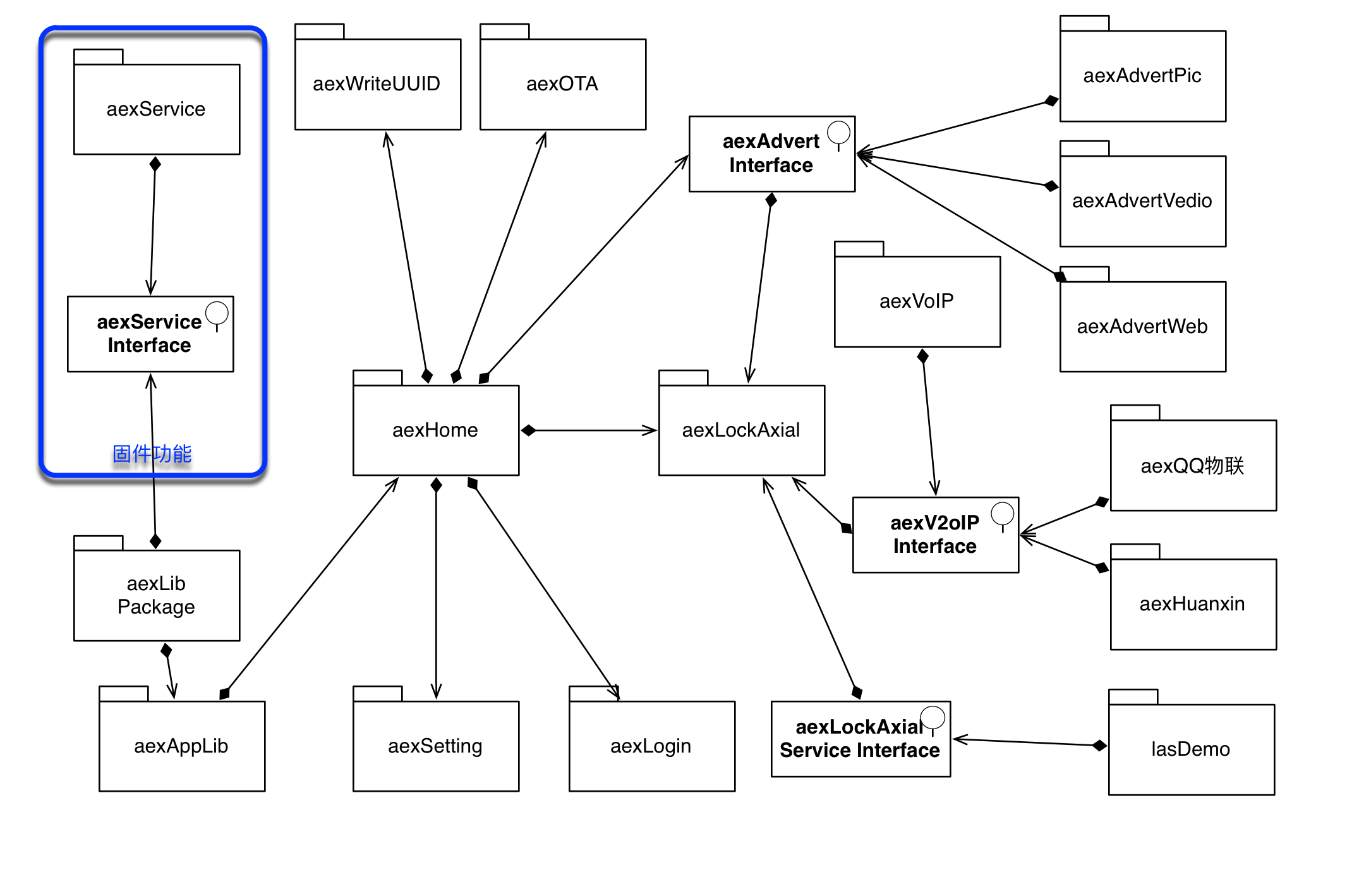
## KK3X 应用介绍

**KK3X结构紧凑，功能全面，品质稳定、使用灵活。可以用于5-55寸的工业控制、机器人、智能装备、可穿戴装备、广告机、查询机、自助智能终端、排队机、POS机、智能门禁对讲等多种领域。它支持串口、GPIO、I2C、I2S等多种通讯接口，可以支持的外设有：热敏打印机、发票打印机、ZigBee、UHF、NFC、身份证阅读、密码键盘、银行刷卡器等等。我们可以提供AndroidEx SDK开发包和全程技术支持，开发包包括内核层驱动、物联网组网、Java层和JavaScript层的设备驱动和物联网通讯的使用接口，并提供演示程序源码，以方便客户专注于功能和UI的开发，快速形成终端产品为客户带来收益。**

## AexSDK系统结构

****

**软件包关系：**

****

* **aexlib：负责上层应用与系统固件的通讯和调用。有些功能是通过修改固件来完成的，但是固件层修改后增加的api的调用一般只能通过底层SDK编译来生成APK。aexlib库就是将固件增加的api调用通过此库封装以便于在Android Studio和Eclipse里调用；**
* **aexAppLibs：提供应用APK编写所需的公用的功能，如：全屏显示、日志、配置等功能。该项目生成的是aar的库，可以在Android Studio上直接使用，如果要在Eclipse上使用需要做处理；**
* **aexHome：这是一个代替Android桌面的应用，演示了如何实现系统自启动、全屏显示等常用应用功能;**
* **aexWriteUuid：KK3X系列主板具有一个唯一的识别码GUID，系统默认情况下是没有配置此识别码的，该应用是为了初始化识别吗的。同时也会初始化LAN的Mac地址，Wlan的Mac地址等。当系统启动时发现没有初始化这个识别码则会自动启动此应用初始化系统参数；**
* **aexLogin：系统登录应用，当用户需要修改系统参数，配置系统时，可以选择校验用户密码，验证通过后才可以管理。默认情况下密码为空，会提示用户设置密码或者跳过密码设置和检查。只有设置了密码时才会要求校验密码成功才可以配置系统；**
* **aexOTA：系统在线升级的应用，当系统、配置、应用有升级时可以通过此应用程序完整在线监测和升级；**
* **aexSetting：系统配置应用，可以配置系统运行参数、网络参数、外设配置参数等等；**
* **aexAdvert：广告展示、更新的应用程序；**
* **aexLockAxial：锁相智能可视门禁对讲应用。**

# 硬件固化参数

KK3X主板内置一个EEROM的存储器，可以存储一些系统固化的参数，在系统重新烧录、格式化等操作时，这些参数都不会变化。这些参数用户层使用虚拟文件的方式存取。

| **参数** | **类型** | **地址** | **说明** |
| --- | --- | --- | --- |
| Flag0 | 2字节整数 | /sys/class/androidex\_parameters/androidex/flag0 | **系统版本标识0x3X00，主板硬件版本号。** |
| Flag1 | 1字节整数 | /sys/class/androidex\_parameters/androidex/flag1 | **驱动启用标识** |
| Uuid | 16字节GUID | /sys/class/androidex\_parameters/androidex/uuid | **硬件唯一序列号** |
| Serial | 8字节序列号 | /sys/class/androidex\_parameters/androidex/serial | **为uuid的低8字节** |
| Lan Mac | 以太网地址 | /sys/class/androidex\_parameters/androidex/lan\_mac | **以太网的网络地址** |
| BT Mac | 蓝牙地址 | /sys/class/androidex\_parameters/androidex/bt\_mac | **蓝牙的网络地址** |
| Wlan Mac | 无线地址 | /sys/class/androidex\_parameters/androidex/wlan\_mac | 无线网的网络地址 |
| Password | 16字节MD5 | /sys/class/androidex\_parameters/androidex/password | 密码的哈希值 |
| BT UART | 30字节 | /sys/class/androidex\_parameters/androidex/bd\_uart | 北斗的串口配置 |
| User Data |  | /sys/class/androidex\_parameters/androidex/userinfo | **用户数据** |

硬件驱动包含：

* TCA9535的IO键盘驱动：flag1&0x01 == 0x01启用该驱动；
* FM188回音消除芯片驱动：flag1&0x02 == 0x02启用该驱动；
* RNS110 NFC芯片驱动：flag1&0x04 == 0x04启用该驱动；
* HYM8563实时时钟芯片驱动：flag1&0x08 == 0x08启用该驱动；
* USB3G 模块驱动：flag1&0x10 == 0x10启用该驱动；
* AEXGPIO驱动：flag1&0x20 == 0x20启用该驱动。
* HwServer与KDC的串口通讯：flag1&0x40==0x40启动HwServer与KDC的串口通讯。

驱动控制是需要重启后才生效的。

参数可以按照文件的方式存取，参数存取的目录为：

/sys/class/androidex\_parameters/androidex

主要参数有：

bt\_mac flag1 uuid bd\_uart

lan\_mac serial uevent wlan\_mac

flag0 password userinfo

读取参数的方式：

cat /sys/class/androidex\_parameters/androidex/uuid

写入参数的方式：

echo “0x34” > /sys/class/androidex\_parameters/androidex/flag0

如果写入的字符串中有引号的要使用单引号。如’{“A”}’。

手工设置参数示例：

echo “0x34”>/sys/class/androidex\_parameters/androidex/flag0

echo “d2bd1b63-f186-4b01-8325-e0f014fbf411” > /sys/class/androidex\_parameters/androidex/uuid

echo “” > /sys/class/androidex\_parameters/androidex/userinfo //这一句很重要，否则腾讯的QQ物联会出错

查看软件通过ANDROID\_ID读取到的序列号的方法：

cat /sys/class/androidex\_parameters/androidex/serial；echo “\n”

硬件固化参数主要使用aexLib库来调用存取。

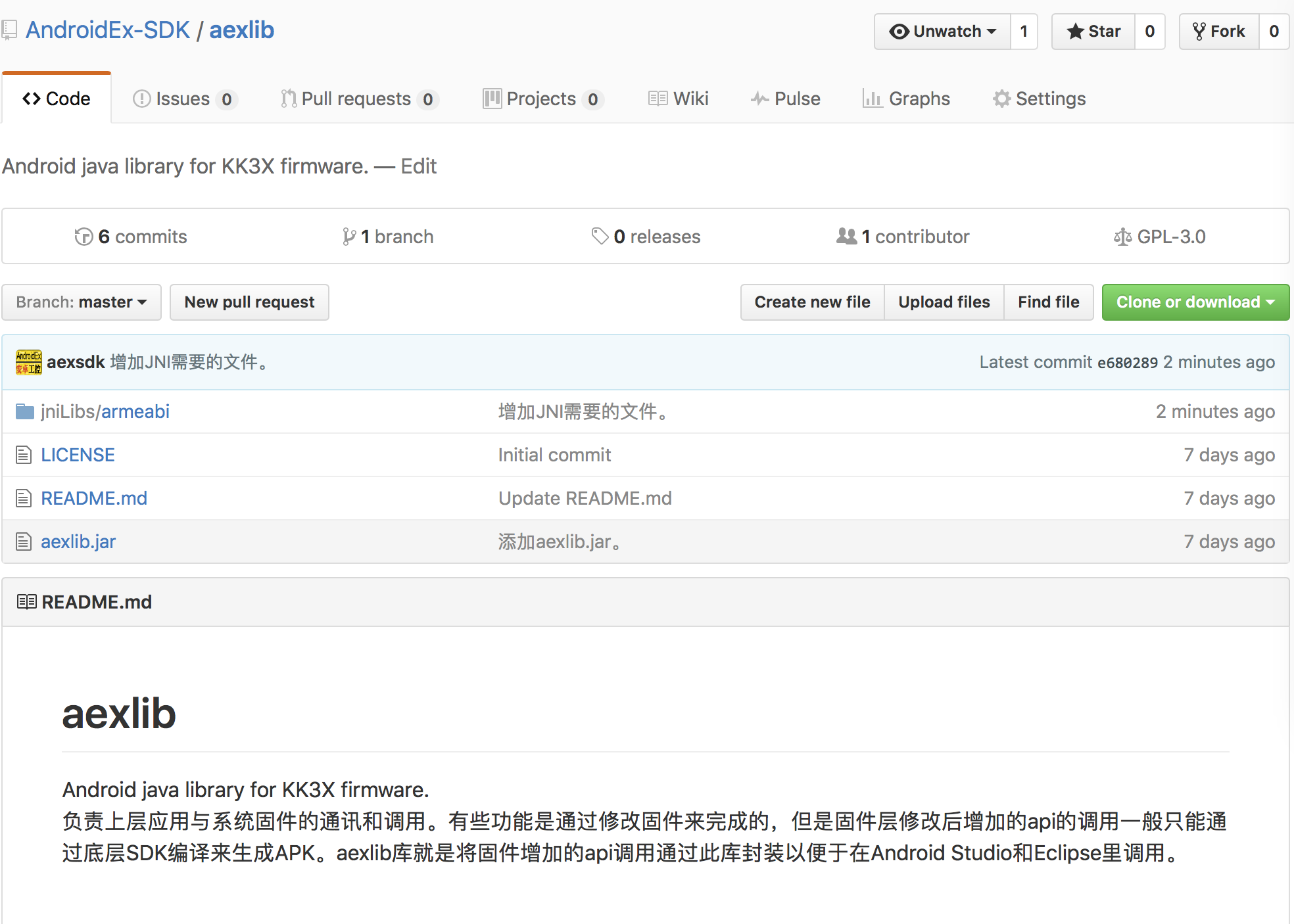
# SDK详解

## AexLib库

aexLib库由操作硬件固化参数的相关类和一些常用JNI组成。这个库与固件联系密切，也会调用framework层的一些扩展服务和接口，因此是在固件编译时生成的。上层应用可以直接使用编译好的jar库和so来开发。

编译好的aexLib库的下载地址为：

<https://aexsdk@github.com/AndroidEx-SDK/aexlib.git>



对于Android Studio：下载后把aexlib.kar放入libs目录，jniLibs放入src/main/jniLibs目录。并在build.gradle增加如下代码：

/\*\* 导入so文件 \*/  
task nativeLibsToJar(type: Zip, description: "create a jar archive of the native libs") {  
 destinationDir file("$projectDir/libs")  
 baseName "jniaexlib"  
 extension "jar"  
 from fileTree(dir: "src/main/jniLibs", include: "\*\*/\*.so")  
 into "lib"  
}  
tasks.withType(JavaCompile) {  
 compileTask -> compileTask.dependsOn(nativeLibsToJar)  
}  
repositories {  
 flatDir {  
 dirs 'libs'  
 }  
}

aexlib包含的类：

| **类名** | **说明** |
| --- | --- |
| com.androidex.aexlibs.hwService | 硬件固化参数操作类 |
| com.androidex.aexlibs.WebJavaBridge | 实现Web调用Java的桥接功能 |
| com.androidex.hwuitls.hwFile | 实现文件目录路径的获取方法 |
| com.androidex.hwuitls.hwLog | 实现Log日志的通用类 |
| com.androidex.logger.Log | 可以在LogFragment显示的Log |

类：com.androidex.aexlibs.hwService

方法概要

| **返回类型** | **方法和说明** | **说明** |
| --- | --- | --- |
| String | execShellCommand(String cmd) | 执行命令行命令 |
| void | runSetAlarm(long wakeupTime) | **设置闹钟** |
| void | runReboot() | **安全重启系统** |
| void | runRebootToAdfu() | **重启到升级固件状态** |
| void | runShutdown() | 安全关机 |
| void | setPlugedShutdown(boolean value) | 设置DC断电后是否关机 |
| int | getOtgState() | **获得USB0的状态0: HOST,1: DEVICE** |
| void | setOtgToHost() | **设置USB0为HOST** |
| void | setOtgToDevice() | **设置USB0为DEVICE** |
| void | setAdbToNet(int port) | **设置ADB使用网络端口PORT** |
| int | writeHex(String file, String hexValue) | **向文件file写入十六进制编码字符串** |
| String | getAndroidExParameter(String addr) | **获得硬件固化参数** |
| void | setAndroidExParameter(String addr,String value) | **设置硬件固化参数** |
| String | get\_uuid() | **获得格式化、升级均不改变的硬件唯一标识** |
| String | get\_serial() | **安卓的ID，此值为UUID的后半部分** |
| String | getSdkVersion() | **获得固件的编译版本号** |
| void | EnterFullScreen() | **进入全屏状态** |
| void | ExitFullScreen() | **退出全屏状态** |

类：com.androidex.aexlibs.WebJavaBridge

com.androidex.aexlibs.WebJavaBridge.OnJavaBridgePlugin

| **方法和说明** |
| --- |
| public JSONObject onExecute(final String action,final JSONObject args,String callbackId);  当JavaScript调用Java时首先进入此接口函数，Plugin需实现此接口来具体实现调用的功能。 |

com.androidex.aexlibs.WebJavaBridge. OnJavaBridgeListener

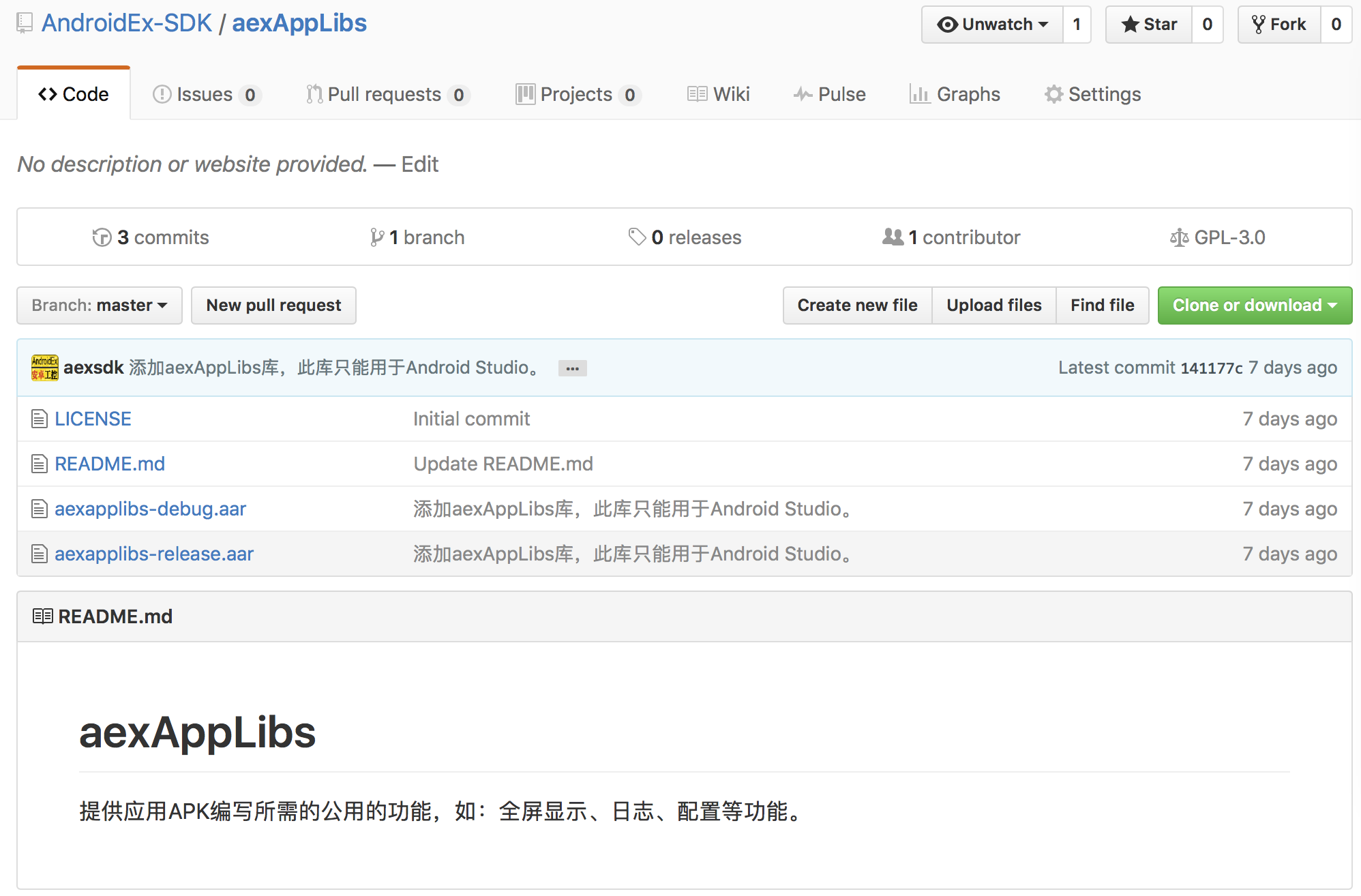
| **方法和说明** |
| --- |
| public void onDeviceEvent(int code,String args);  设备事件以及外设事件，JS只需要设置一个事件，通过事件Code和args来确定具体的事件操作。 |
| public void onSendJavaScript(String jscode);  当需要让Web执行JS代码 |

## aexAppLibs库

提供应**用APK编写所需的公用的功能，如：全屏显示、日志、配置等功能。该项目生成的是aar的库，可以在Android Studio上直接使用，如果要在Eclipse上使用需要做处理。**

编译好的aexLib库的下载地址为：

<https://github.com/AndroidEx-SDK/aexAppLibs.git>



## 系统设置

系统设置主要是针对设备硬件的初始化、硬件配置、网络配置、在线升级等功能。

### 初始化UUID：aexWriteUuid

当设备安装好以后首先要设置的就是UUID，设备管理及应用系统多数都会根据此标识来工作。此UUID在系统格式化、升级等操作后也不会改变的唯一标识。有些用户喜欢使用Android的标准API获得ANDROIDID来作为设备ID，我们的硬件对Android的这个标准API的实现进行了更改，通过此方法获得的是UUID的后半部分（16字节大写十六进制字符串）。



### 系统登录：aexLogin

系统登录Activity，当硬件固化参数没有设置密码的时候提示设置密码，当设置密码后则验证密码，登录成功后才可以继续使用其他系统配置的功能。



### 系统在线升级：aexOTA

系统在线升级。

当系统第一次启动如果设置了UUID，则首先根据UUID到后台服务器下载系统预置信息：开机Logo、开机动画、开机运行参数、关键更新、开机运行程序等。

后台服务器的确定：系统预置了后台服务器的域名地址，初次之外系统会再局域网内广播查找网内服务器供用户选择。



### 系统配置：aexSettings

系统配置。









## 典型应用

典型应用包括：终端桌面程序、广告展示、智能门禁对讲终端。

### 终端桌面应用：aexHome

终端桌面程序主要演示如下功能：

* 可以做系统桌面，这样的自动启动实现可以避免启动后出现安卓桌面后才进入用户应用程序；
* 全屏显示的演示；
* Log日志的使用演示；
* Activity嵌入到View的显示方式；
* 启动可返回的系统设置界面。

### 广告展示应用：aexAdvert

广告展示应用，可以演示如下功能：

* 广告图片幻灯片播放；
* 视频播放；
* WEB页面播放及切换。

### 智能门禁应用：aexLockAxial

**锁相开源门禁机是互联网+安防门禁的应用实例。它是使用安卓系统的控制板结合腾讯QQ物联和微信设备实现安防门禁功能。**

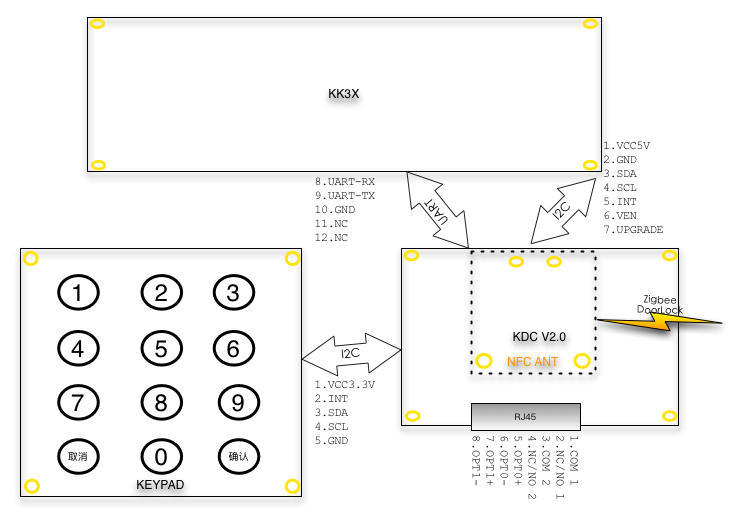
#### 使用场景

* **手机QQ远程开门；**
  1. **用户到门禁附近打开手机QQ点击开锁功能开门；**
  2. **访客在门禁机上按房号呼叫用户手机QQ，可视对讲开门。**
* **近距离蓝牙连接开门；**
  1. **用户到门禁附近通过手机QQ的连接门禁蓝牙模块开门；**
  2. **蓝牙智能穿戴设备开门。**
* **带NFC手机刷手机开门；**
  1. **NFC手机模拟门禁卡开门。**
* **各种NFC卡开门**
  1. **小区物业卡、停车卡；**
  2. **公交卡；**
  3. **银联闪付卡；**
  4. **电卡、燃气卡等。**

#### 锁相开源门禁的特点

* **开门的方式：多样、便捷、安全；**
* **实时反馈：下面的信息均会实时反馈到系统服务器和用户的手机上**
  + **谁、什么时间、通过什么方式开门？**
  + **开锁后门是否开启？门是否关好？**
* **使用QQ物联及微信设备做云通讯平台：安全、可靠、运维成本低；**
* **开放：客户可以定制硬件、软件、平台，打造自己的门禁机品牌。**
  + **门禁机终端硬件开源：客户可以定制设备的外观、添加和裁剪功能；**
  + **标准化配件：外壳、摄像头、按键板、门禁控制板、主控板等均提供标准化配件，降低客户定制成本；**
  + **门禁机终端软件开源：具备基础门禁功能的锁相门禁机终端软件向所有客户开源，客户可以根据自己的实际需要定制。**

#### 功能框图



锁相开源门禁机分为三部分组成：安卓控制板、按键板、门禁控制器。

* 安卓控制板：采用KK34V2.1，负责液晶显示、网络通讯、音视频采集&编解码&传输、以及与门禁控制板通讯完成按键接收、开门控制、状态事件报告等功能；
* 门禁控制板：采用KDC V2.0，负责转发按键信息，转发NFC读卡信息，控制门锁继电器，接收门磁输入状态；
* 按键板：采用KEYPAD V1.0，实现3X4的按键，并通过I2C与门禁控制板连接。

#### 开门

开门的实现是向/dev/rkey写入指定格式的数据即可，参考代码如下：

String **rkeyDev** = **"/dev/rkey"**;  
**int ident** = 0;

/\*\*  
 \* 开门指令  
 \* **@param index** 门的序号,主门=0,副门=1  
 \* **@param delay** 延迟关门的时间,0表示不启用延迟关门,大于0表示延迟时间,延迟时间为delay\*150ms  
 \* **@return** 大于0表示成功,实际上等于9表示真正的成功,因为返回值表示写入的数据,开门指令长度为9.  
 \*/  
**public int** openDoor(**int** index, **int** delay){  
 kkfile rkey = **new** kkfile();  
  
 **if**(index < 0 || index > 0xFE) index = 0;  
 **if**(**ident** < 0 || **ident** > 0xFE) **ident** = 0;  
 **if**(delay < 0 || delay > 0xFE) delay = 0;  
 String cmd = String.format(**"FB%02X2503%02X01%02X00FE"**,**ident**,index,delay);  
 **int** r = rkey.writeHex(**rkeyDev**,cmd);  
 **return** r > 0?1:0;  
}  
**public int** closeDoor(**int** index){  
 kkfile rkey = **new** kkfile();  
  
 **if**(index < 0 || index > 0xFE) index = 0;  
 **if**(**ident** < 0 || **ident** > 0xFE) **ident** = 0;  
 String cmd = String.format(**"FB%02X2503%02X000000FE"**,**ident**,index);  
 **int** r = rkey.writeHex(**rkeyDev**,cmd);  
 **return** r > 0 ? 1:0;  
}

#### 获取门状态

获取开门状态是通过监听广播来实现的，启动代码里添加

**mReceiver** = **new** NotifyReceiver();  
 IntentFilter filter = **new** IntentFilter();  
 filter.addAction(**"com.android.action.doorsensor"**);  
 ……  
 registerReceiver(**mReceiver**, filter);

接收代码如下：

if (intent.getAction().equals(mDoorSensorAction)){  
 String doorsensor = intent.getStringExtra("doorsensor"); //获取广播消息内容  
 UEventMap mds = new UEventMap(doorsensor);

Intent ds\_intent = new Intent();  
 ds\_intent.setAction(DoorLock.DoorLockStatusChange);  
 ds\_intent.putExtra("doorsensor",mds.get("doorsensor"));

*/\*\**

*\* doorsensor的格式为0xXXYY*

*\* XX ： 表示门的序号，00=主门，01=副门*

*\* YY ： 表示门的状态，默认设置00表示打开，01表示关闭，实际情况要根据门磁传感器接常开还是常闭来决定。*

*\*/* sendBroadcast(ds\_intent);  
 }

全部示例代码如下：

*/\*\*  
 \*   
 \* 锁相开源门禁机软件的主要服务类,DoorLock主要提供开门,关门指令以及上报门开和关闭的事件.  
 \*/***public class** DoorLock **extends** Service **implements** OnBackCall {  
  
 **public static final** String ***TAG*** = **"DoorLock"**;  
 **public static final** String ***mDoorSensorAction*** = **"com.android.action.doorsensor"**;  
  
 **private** DoorLockServiceBinder **mDoorLock**;  
  
 */\*\*  
 \* 当门的状态改变时的事件定义  
 \*/* **public static final** String ***DoorLockStatusChange*** = **"DoorLockStatusChange"**;  
 */\*\*  
 \* DoorLock通过DoorLockOpenDoor广播获得开门指令并发送给门禁控制器  
 \*/* **public static final** String ***DoorLockOpenDoor*** = **"DoorLockOpenDoor"**;  
 **private** NotifyReceiver **mReceiver**;  
 **private static** DoorLock *mServiceInstance* = **null**;  
  
 **public static** DoorLock getInstance()  
 {  
 **return** *mServiceInstance*;  
 }  
  
 @Override  
 **public void** onCreate() {  
 **super**.onCreate();  
 **mDoorLock** = **new** DoorLockServiceBinder();  
 **mReceiver** = **new** NotifyReceiver();  
 IntentFilter filter = **new** IntentFilter();  
 filter.addAction(***mDoorSensorAction***);  
 filter.addAction(***DoorLockOpenDoor***);  
 registerReceiver(**mReceiver**, filter);  
 **int** r = **mDoorLock**.openDoor(1,16);  
 **if**(r == 9)  
 Toast.*makeText*(DoorLock.**this**, String.*format*(**"Open %d,delay %ds close."**,1,16\*150/1000), Toast.***LENGTH\_SHORT***).show();  
 **else** Toast.*makeText*(DoorLock.**this**, String.*format*(**"Open door 1 fail return %d."**,r), Toast.***LENGTH\_SHORT***).show();  
  
 Log.*d*(***TAG***,String.*format*(**"open door %d"**,r));  
 }  
  
 @Override  
 **public void** onDestroy() {  
 unregisterReceiver(**mReceiver**);  
 **super**.onDestroy();  
 *mServiceInstance* = **null**;  
 **mDoorLock** = **null**;  
 }  
  
 @SuppressWarnings(**"deprecation"**)  
 @Override  
 **public void** onStart(Intent intent, **int** startId) {  
  
 **super**.onStart(intent, startId);  
 *mServiceInstance* = **this**;  
 }  
  
 @Override  
 **public int** onStartCommand(Intent intent, **int** flags, **int** startId) {  
 **return super**.onStartCommand(intent, flags, startId);  
 }  
  
 @Nullable  
 @Override  
 **public** IBinder onBind(Intent intent) {  
 **return mDoorLock**;  
 }  
  
 @Override  
 **public void** onBackCallEvent(**int** code, String args) {  
 Log.*v*(**"onBackCallEvent"**,args);  
 }  
  
 **public class** DoorLockServiceBinder **extends** IDoorLockInterface.Stub{  
 String **rkeyDev** = **"/dev/rkey"**;  
 **int ident** = 0;  
  
 */\*\*  
 \* 开门指令  
 \** ***@param index*** *门的序号,主门=0,副门=1  
 \** ***@param delay*** *延迟关门的时间,0表示不启用延迟关门,大于0表示延迟时间,延迟时间为delay\*150ms  
 \** ***@return*** *大于0表示成功,实际上等于9表示真正的成功,因为返回值表示写入的数据,开门指令长度为9.  
 \*/* **public int** openDoor(**int** index, **int** delay){  
 kkfile rkey = **new** kkfile();  
  
 **if**(index < 0 || index > 0xFE) index = 0;  
 **if**(**ident** < 0 || **ident** > 0xFE) **ident** = 0;  
 **if**(delay < 0 || delay > 0xFE) delay = 0;  
 String cmd = String.*format*(**"FB%02X2503%02X01%02X00FE"**,**ident**,index,delay);  
 **int** r = rkey.writeHex(**rkeyDev**,cmd);  
 **return** r > 0?1:0;  
 }  
 **public int** closeDoor(**int** index){  
 kkfile rkey = **new** kkfile();  
  
 **if**(index < 0 || index > 0xFE) index = 0;  
 **if**(**ident** < 0 || **ident** > 0xFE) **ident** = 0;  
 String cmd = String.*format*(**"FB%02X2503%02X000000FE"**,**ident**,index);  
 **int** r = rkey.writeHex(**rkeyDev**,cmd);  
 **return** r > 0 ? 1:0;  
 }  
 }  
  
 **public class** NotifyReceiver **extends** BroadcastReceiver {  
 @Override  
 **public void** onReceive(Context context, Intent intent) {  
 **if** (intent.getAction().equals(***mDoorSensorAction***)){  
 String doorsensor = intent.getStringExtra(**"doorsensor"**);  
 UEventMap mds = **new** UEventMap(doorsensor);  
  
 Log.*d*(***TAG***, String.*format*(**"%s\t Door sensor=%s\n"**,mds.get(**"doorsensor"**), mds.toString()));  
  
 Intent ds\_intent = **new** Intent();  
 ds\_intent.setAction(DoorLock.***DoorLockStatusChange***);  
 ds\_intent.putExtra(**"doorsensor"**,mds.get(**"doorsensor"**));  
 sendBroadcast(ds\_intent);  
 } **else if**(intent.getAction().equals(***DoorLockOpenDoor***)) {  
 **int** index = intent.getIntExtra(**"index"**, 0);  
 **int** status = intent.getIntExtra(**"status"**, 0);  
  
 **if** (status != 0){  
 **mDoorLock**.openDoor(index, 0x20);  
 }**else** {  
 **mDoorLock**.closeDoor(index);  
 }  
 }  
 }  
 }  
  
 **public static final class** UEventMap {  
 *// collection of key=value pairs parsed from the uevent message* **private final** HashMap<String,String> **mMap** = **new** HashMap<String,String>();  
  
 **public** UEventMap(String message) {  
 **int** offset = 0;  
 **int** length = message.length();  
  
 **if**(length == 0)**return**;  
 **if**(message.substring(0,1).equals(**"{"**)){  
 message = message.substring(1);  
 }  
 **if**(message.substring(message.length() - 1,message.length()).equals(**"}"**)){  
 message = message.substring(0,message.length() - 1);  
 }  
 length = message.length();  
 **while** (offset < length) {  
 **int** equals = message.indexOf(**'='**, offset);  
 **int** at = message.indexOf(**','**, offset);  
 **if** (at < 0) **break**;  
  
 **if** (equals > offset && equals < at) {  
 *// key is before the equals sign, and value is after* **mMap**.put(message.substring(offset, equals).trim(),  
 message.substring(equals + 1, at).trim());  
 }  
  
 offset = at + 1;  
 }  
 }  
  
 **public** String get(String key) {  
 **return mMap**.get(key);  
 }  
  
 **public** String get(String key, String defaultValue) {  
 String result = **mMap**.get(key);  
 **return** (result == **null** ? defaultValue : result);  
 }  
  
 **public** String toString() {  
 **return mMap**.toString();  
 }  
 }  
  
}

### 自助终端应用：aexSmartBiz

自助终端应用主要演示如何使用安卓工控主板来实现自助终端设备，如：填单机、排队机、收银机、自动售货机、自助服务终端等等。主要演示功能：

* 设备配置方式：如何配置各种外设的连接端口和使用参数；
* NFC读卡器的使用演示；
* CPU读卡器的使用演示；
* 银行卡读卡器的使用演示；
* 密码键盘的使用演示；
* 二代身份证读取的使用演示；
* 热敏打印机的使用演示。





深圳市安卓工控设备有限公司

ShenZhen AndroidEx Co., Ltd.

地址：深圳市龙岗布吉龙景工业园E栋二楼东

主页：<http://www.androidex.cn>

电话：+86-755-28435491