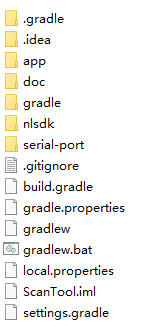
**ScanTool使用手册**

1. 简介

ScanTool是基于一个用于Android系统的SDK+demo程序包，工程采用[Android Studio](http://www.android-studio.org/)编译。

开发包的目录结构和软件层次如下图所示：

开发包目录结构 软件层次

serial-port、nlsdk是SDK相关源码目录，app是demo相关源码目录。

ScanTool app采用了以下第三方开发包：

implementation 'com.jakewharton:butterknife:10.2.1'  
annotationProcessor 'com.jakewharton:butterknife-compiler:10.2.1'  
implementation '[io.reactivex.rxjava2:rxjava:2.0.1](https://github.com/ReactiveX/RxJava/)'  
implementation '[io.reactivex.rxjava2:rxandroid:2.0.1](https://github.com/ReactiveX/RxJava/)'

注意：

1. 目前仅支持在Android主机上打开一台我司设备。
2. 在首次编译工程时需确保主机能够连接外网，gradle会自动更新所需的第三方工具，时间较长请保持耐心。
3. 请注意第三方库的版权声明。
4. 适用产品：可以使用Newland Easyset软件的设备亦可使用本工具
5. SDK接口简要使用说明

基本使用流程为：创建设备—打开设备—使用设备—关闭设备（可选）

1. 创建通讯接口的流实例

NLDevice构造函数参数指定为需要使用的通讯类型

DEV\_CDC usb虚拟串口  
 DEV\_POS USB pos接口  
 DEV\_COMPOSITE USB键盘和POS的复合设备  
 DEV\_SUREPOS IBM SurePos接口  
 DEV\_UART 物理串口

NLDeviceStream ds = new NLDevice(NLDeviceStream.DevClass.*DEV\_UART*);

1. 打开流实例
2. 打开物理串口实例

这里对物理串口的访问要求设备具备root权限来操作/dev/目录下设备节点。

if (!ds.open("/dev/ttyAMA0", 115200, new NLDeviceStream.NLUartListener() {  
 @Override  
 public void actionRecv(byte[] recvBuff, int len) {  
 barcodeLen = len;  
 if (usbOpenChecked) {  
 System.*arraycopy*(recvBuff, 0, barcodeBuff, 0, len);  
 observable.subscribeOn(Schedulers.*newThread*())  
 .observeOn(AndroidSchedulers.*mainThread*())  
 .subscribe(usbRecvObserver);  
 }  
 }  
  
})) {  
 bnOpenDevice.setText(R.string.*TextOpen*);  
 usbOpenChecked = false;  
 return;  
}

UART设备流有1个监听接口用于接收UART收到的数据：

interface NLUartListener {  
 void actionRecv(byte [] RecvBuff, int len);  
}

1. 打开USB通讯口实例（包含CDC/ POS/COMPOSITE）

if (!ds.open(this, new NLDeviceStream.NLUsbListener() {  
 @Override  
 public void actionUsbPlug(int event) {  
 if (event == 1) {  
 MainActivity.this.ShowToast("设备插入");  
 } else {  
 MainActivity.this.ShowToast("设备拔出");  
 observable.subscribeOn(Schedulers.*newThread*())  
 .observeOn(AndroidSchedulers.*mainThread*())  
 .subscribe(usbPlugObserver);  
 }  
 }  
  
 @Override  
 public void actionUsbRecv(byte[] recvBuff, int len) {  
 barcodeLen = len;  
 if (usbOpenChecked) {  
 System.*arraycopy*(recvBuff, 0, barcodeBuff, 0, len);  
 observable.subscribeOn(Schedulers.*newThread*())  
 .observeOn(AndroidSchedulers.*mainThread*())  
 .subscribe(usbRecvObserver);  
 }  
 }  
  
})) {  
 bnOpenDevice.setText(R.string.*TextOpen*);  
 usbOpenChecked = false;  
 return;  
}

USB设备流有2个监听接口：

interface NLUsbListener {  
 /\*\*  
 \* 当检测到USB设备拔插动作是通知应用  
 \* @param event 1:USB设备插入， 0:USB设备拔出  
 \*/  
 void actionUsbPlug(int event);  
 void actionUsbRecv(byte [] RecvBuff, int len);  
}

actionUsbPlug用来监听USB的拔插动作

actionUsbRecv用来监听接收数据

1. 对打开的设备流进行操作
2. 获取图片：
3. 获取图片分辨率（第一次调用时）

int[] wh = ds.getImgSize();

1. 获取图像数据



1. 更新固件：

更新进度通过设置监听来完成



注：

1. 对于采用USB通讯方式更新mcu类产品的固件，由于该类产品在固件更新的过程中会重启进入boot，导致android设备会检测到USB设备的拔插动作，从而导致要求用户重新进行USB 访问授权。对此有两种处理方式：一对于集成了SDK的应用程序给与系统签名，这么做可以避免拔插重新要求授权；二是SDK内部代码增加了2s延时，这就要求在设备弹出授权框的时候必须及时确认授权，否则授权失败。



如上述代码所示，对于已经获取系统签名的程序，该处延时可以缩减到100ms以提高更新效率。

1. 固件在更新过程中必须保持设备不能断电，务必在确认设备已经重启后再执行断电操作！
2. 其他操作：

略。

1. SDK接口说明

ScanTool\nlsdk\src\main\java\com\nlscan\nlsdk\NLDeviceStream.java

|  |  |
| --- | --- |
| 内部枚举类型 | |
| enum DevClass USB通信接口类型， | |
| DEV\_CDC, | usb虚拟串口 |
| DEV\_POS, | USB pos接口 |
| DEV\_COMPOSITE, | USB键盘和POS的复合设备 |
| DEV\_SUREPOS, | BM SurePos接口（目前还未支） |
| DEV\_UART | 物理串口 |
| enum NLUpdateState 固件更新状态 | |
| STATE\_ENTER\_UPDATE, |  |
| STATE\_SET\_PARAM, |  |
| STATE\_SEND\_DATA, |  |
| STATE\_WAIT\_UPDATE, |  |
| STATE\_UPDATE\_COMPLETE |  |
| 内部回调接口 | |
| NLUsbListener调用open方式时该接口作为系统USB事件的监听器接口，用于监听USB拔插事件 | |
| void actionUsbPlug(int event); | 当检测到USB设备拔插动作时通知应用  1:USB设备插入， 0:USB设备拔出 |
| void actionUsbRecv(byte [] RecvBuff, int len); | 当通讯口接收到数据通知应用  RecvBuff 接收缓冲  len 缓冲大小 |
| NLUartListener 调用open方式时，该接口作为串口的接收数据的接口，当串口接收到数据的时候会回调该接口的方法 | |
| void actionRecv(byte [] RecvBuff, int len); | 当通讯口接收到数据通知应用  RecvBuff 接收缓冲  len 缓冲大小 |
| transImgListner 调用获取图像时监听传输图像进度 | |
| void curProgress(int percent); | percent 传输进度 |
| updateListner 调用更新固件时监听更新进度信息 | |
| void curProgress(String type, NLUpdateState state, int percent); | type boot:引导程序 kernel：内核代码 flash：其他配置文件  state 升级状态指示，由NLUpdateState中定义  percent 指示每个state状态下的完成百分比 |
| 接口方法 | |
| boolean open(Context context, NLUsbListener listener); | USB设备打开接口，调用前必须保证拥有对USB设备节点的读写权限。当需要打开的设备是DEV\_COMPOSITE时,会通知设备进行通讯口的切换。  参数说明：  context: 安卓 context 用于枚举设备  listener: 系统USB事件监听器  返回值：  true：成功 false：失败 |
| boolean open(String pathName, int baudrate, NLUartListener listener); | 串口设备打开接口  **参数说明：**  pathName 串口设备名 如：/dev/ttys0  baudrate 串口波特率  listener 串口接收事件监听  **返回值：**  true：成功 false：失败 |
| NLCommStream getDevObj(); | 应用通过返回流对象进行分析打开的设备流类型  参数说明：  无  返回值：  返回流创建的流对象 |
| String GetSdkVersion(); | 获取SDK版本号  参数说明：  无  返回值：  SDK版本号 |
| void close(); | 关闭设备  参数说明：无  返回值：无 |
| boolean isOpen(); | 判断设备是否打开  参数说明：无  返回值：  true：打开 false：关闭 |
| boolean checkHealth(); | 判断设备是否正常（特定设备支持，详情参见用户手册）  参数说明：无  返回值：  true：工作正常 false：工作异常 |
| String getDeviceInformation(); | 获取设备信息  参数说明：无  返回值：  统一指令集QRYSYS指令的返回结果 |
| boolean startScan(); | 发送默认自定义触发指令（0x10 0x54 0x04）开始读码，由于该指令须启用配置，在使用前请确认串行触发指令已打开（SCNTCE1）。触发模式下有效。  参数说明：无  返回值：  true：读码指令发送成功 false：指令发送失败 |
| boolean stopScan(); | 停止读码，触发模式下有效。当设备接收到触发读码指令在超时时间内没有达到码词，可以通过该接口停止设备解码。  参数说明：  返回值：  true：指令发送成功 false：指令发送失败 |
| boolean restartDevice(); | 重启设备  参数说明：无  返回值：  true：指令发送成功 false：指令发送失败 |
| boolean setConfig(String command); | 发送单条设置指令 例子： setConfig("128ENA1") 该设置码掉电不保存；如果需要设置码掉电后依然生效，可以在命令前增加字符“@”，比如setConfig("@128ENA1")  参数说明：  command 统一指令集的设置指令  返回值：  true：指令发送成功 false：指令发送失败 |
| String getConfig(String command); | 查询统一指令的当前设置，仅支持单条指令的查询。比如 SCNMOD\* 查询返回结果为 SCNMOD0  参数说明：  command 统一指令集的获取指令  返回值：  返回当前查询指令的应答 |
| int updateFirmware(byte[] fireware, updateListner listner); | 更新模组头固件，固件升级包会根据客户的要求包含不同的内容。  参数说明：  fireware 固件内容缓冲  listner 监听更新固件的进度监听器  返回值：  {class NLError} 中描述的错误类型 |
| int updateConfig(File f); | 更新模组头配置，设备的配置文件通常包含多条配置信息，配置在发送到设备后需要较长的执行时间  参数说明：  f xml格式的批量配置文件句柄  返回值：  >0:更新成功；<0:更新失败 ; =0 更新成功，且进行了端口切换 |
| int[] getImgSize(); | 获取设备当前图像的长度信息（长、宽）  参数说明：无  返回值：  设备当前图像的宽（int[0]）高（int[1]） |
| boolean getImgBuff(byte[] ImgBuff, int imgSize, transImgListner listner); | 遵循统一指令集获取图像的方法，获取设备当前图像，目前该接口仅支持获取原始大小、bmp格式的图片  参数说明：  ImgBuff 接收获取到的图像缓存  imgSize 图像缓存的大小  返回值：  true：图像获取成功 false：图像获取失败 |