杰理 Usb Dongle 开发文档(Android)

珠海市杰理科技股份有限公司
Zhuhai Jieli Technology Co.,LTD

修改记录

版本	更新日期	描述
0.1.0	2022.05.20	首次发布

目录

1. 接入流程

1 准备环境

正文

开发环境	版本	备注
Android Studio	建议使用最新的版本	用于开发 Android 应用
杰理 Usb dongle	建议使用最新的固件	固件须支持杰理 Usb Dongle 协议
测试手机	建议不低于 Android5.0	SDK 不支持模拟器

1.1. 库导入

1.1.1. SDK 库

1、jl_usb_dongle_Vx.x.x.release.aar：SDK 库，提供了相关接口，方便二次开发，**必选**

注意：xxx 为版本号，请以最新发布版本为准

1.1.2. 导入库

1、将 SDK 库 aar 文件放入工程目录中的对应 moudle 的 libs 文件夹下

2、在 Android Studio 中 module 的 build.gradle 中添加：

```
implementation fileTree(include: ['*.aar'], dir: 'libs')
```

1.1.3. Android 清单要求

由于并非所有 Android 设备都保证支持 USB 主机 API，因此请添加 <uses-feature> 来声明 App 使用 *android.hardware.usb.host* 功能。

1.2. 初始化

1.2.1. 创建 Usb dongle 类对象

```
1.  dongleManager = DongleManager.getInstance();
```

这里的 DongleManger 是继承自 UsbClientImp。

1.2.2. 遍历 Usb 设备

```
1.  UsbDevice usbDevice = dongleManager.findDevice(vid, pid);
```

通过指定的 VID 和 PID，得到 `UsbDevice`。

1.2.3. 打开设备

先判断用户是否授权打开 USB 设备

```
1.  private void open(UsbDevice usbDevice) {
2.   UsbManager usbManager = (UsbManager) context.getSystemService(Context.USB_SERVICE);
3.   if (!usbManager.hasPermission(usbDevice)) {
4.    Jlog.w(tag, "Request permission now");
5.    PendingIntent permissionIntent = PendingIntent.getBroadcast(MainApplication.getContext(),
6.      0, new Intent(ACTION_USB_PERMISSION), 0);
7.    usbManager.requestPermission(usbDevice, permissionIntent);
8.    IntentFilter filter = new IntentFilter(ACTION_USB_PERMISSION);
9.    MainApplication.getContext().registerReceiver(mUsbReceiver, filter);
10.  } else {
11.   openDevice(usbDevice);
12.  }
13. }
```

打开 Usb 设备

```
1.  private void openDevice(UsbDevice usbDevice) {
2.   dongleManager.open(usbDevice, new UsbCallback() {
3.    @Override
4.    public void onSuccess() {
5.     Jlog.i(tag, "open success");
6.    }
7.
8.    @Override
9.    public void onFailure(ErrorCode errorCode) {
10.    Jlog.e(tag, "open failure");
11.   }
12.  });
13. }
```

1.2.4. 关闭设备

当不再使用或者退出 App 时，需要关闭 Usb 设备：

```
1.  dongleManager.close();
```

1.3. 发送命令

```
1.  dongleManager.send(Command, CommandCallback);
```

1、这里的 Command 是继承自 DongleCommand 类。如果 SDK 没有满足需求的命令，可以通过自定义命令实现，参考自定义命令章节。

2、这里的 CommandCallback，是一个抽象的接口类：

```
1.  public abstract class CommandCallback {
2.
3.      /**
4.       * 命令已发送时回调
5.       */
6.      public void onSent(BaseCommand command) {
7.      }
8.
9.      /**
10.      * 命令发送过程中出错时回调
11.      * @param code 错误信息
12.      */
13.     public abstract void onError(ErrorCode code);
14.
15.     /**
16.      * 当设备回复对应命令时回调
17.      * @param command 命令
18.      */
19.     public void onResponse(BaseCommand command) {
20.     }
21. }
```

1.4. 其他事件监听

用于监听设备事件回调的接口

```
1.  public abstract class OnDongleEventListener implements OnCommandListener
    {
2.      /**
3.       * 设备发送 RCSP 命令时回调
4.       * @param command RCSP 命令包
5.       */
```

```
    @Override
    public void onRcspNotify(RcspCmd command) {
    }

    /**
     * 设备主动发过来的命令时回调
     * @param command 命令
     */
    @Override
    public void onCommandNotify(BaseCommand command) {
    }

    /**
     * Usb dongle 通信过程中出现错误时回调
     * @param errorCode 错误信息
     */
    @Override
    public void onError(ErrorCode errorCode) {

    }
}
```

可以在打开设备成功时设置事件监听器：

```
dongleManager.setEventListener(onDongleEventListener);
```

2. 功能与接口

2.1. RCSP 数据

2.1.1. 功能描述

Usb Dongle 透传远端 HID 设备的 RCSP 数据，接收端无需回复。

●byte[] getRcspData()

设备端返回二进制 RCSP 数据数组。

●setRcspData(byte[] data)

App 设置要发送的 RCSP 数据数组。

2.1.2. 使用示例

```
RcspCmd cmd = new RcspCmd();
cmd.setRcspData(rcspData);
cmd.setChannel(channelID);//RCSP 专用频道
dongleManager.send(cmd, new CommandCallback() {
 @Override
 public void onSent(BaseCommand command) {
  // 发送成功的回调
 }

@Override
public void onError(ErrorCode code) {
 // 发送失败的回调
}

@Override
public void onResponse(BaseCommand command) {
 //  设备回复对应命令
}
});
```

2.2. Usb dongle 信息

2.2.1. 功能描述

请求 Usb dongle 设备信息，设备须要回复。

●int getVid()

得到 Dongle 的厂商 ID

●int getPid()

得到 Dongle 的产品 ID

●int getSdkVersion()

得到 Dongle SDK 版本信息(SDK 版本号: 例如 211)

●getVersion()

得到 Dongle 产品版本号

●getEnvStatus()

得到 Dongle 所处环境, (0 -- SDK， 1 -- loader)

2.2.2. 使用示例

```
DongleInfoCmd cmd = new DongleInfoCmd();
dongleManager.send(cmd, new CommandCallback() {
 @Override
 public void onSent(BaseCommand command) {
  // 发送成功的回调
 }

 @Override
 public void onError(ErrorCode code) {
  // 发送失败的回调
 }

 @Override
 public void onResponse(BaseCommand command) {
  //  设备回复对应命令
 }
});
```

2.3. 在线蓝牙设备

2.3.1. 功能描述

请求在线设备信息列表，设备须要回复。

●List<BluetoothDevicesCmd.Device> getBluetoothDevices()

获取在线蓝牙设备信息列表

●BluetoothDevicesCmd.Device

这个类是 BluetoothDevicesCmd 的内部静态类，信息包含：

1)对应远端 HID 的 channel

2)是否支持 OTA 升级

3)是否已通过设备认证

4)远端设备的 Mac 地址

5)设备名称

2.3.2. 使用示例

```
1.  BluetoothDevicesCmd cmd = new BluetoothDevicesCmd();
2.  dongleManager.send(cmd, new CommandCallback() {
3.   @Override
4.   public void onSent(BaseCommand command) {
5.    // 发送成功的回调
6.   }
7.
8.   @Override
9.   public void onError(ErrorCode code) {
10.   // 发送失败的回调
11.  }
12.
13.  @Override
14.  public void onResponse(BaseCommand command) {
15.   //  设备回复对应命令
16.   BluetoothDevicesCmd cmd = (BluetoothDevicesCmd) command;
17.         cmd.getBluetoothDevices();// 在线设备列表
18.  }
19. });
```

2.4. 设备状态通知

2.4.1. 功能描述

如果有设备上线或者下线，设备主动通知 App，此时 App 需要重新获取在线设备。

●List<BluetoothStateCmd.State> getDeviceStates()

得到上线/下线设备的信息列表

●BluetoothStateCmd.State

这个类是 BluetoothStateCmd 的内部静态类，信息包含：

1)对应设备的 channel

2)设备状态(下线/上线)

2.4.2. 使用示例

```
1.  private final OnDongleEventListener onDongleEventListener = new OnDongle
    EventListener() {
2.
3.   @Override
4.   public void onCommandNotify(BaseCommand command) {
5.    if (command.getCmdType() == Command.BLUETOOTH_STATE) { //通知App 设备有
    状态更新
6.     BluetoothStateCmd cmd = (BluetoothStateCmd) command;
7.     List<BluetoothStateCmd.State> states = cmd.getDeviceStates();// 设备
    状态列表
8.    }
9.   }
10. };
```

2.5. 连接远端设备

2.5.1. 功能描述

请求连接远端 HID 设备，设备须要回复。

●setRemoteChannel(int remoteChannel)

设置远端设备对应的 channel

●setMac(byte[] mac)

设置远端设备的 Mac 地址

●setUseNewConnectWay(boolean useNewConnectWay)

设置是否使用新的回连方式

●int getResult()

远端设备是否连接成功, 0 表示成功

2.5.2. 使用示例

```
1.  ConnectRemoteCmd cmd = new ConnectRemoteCmd();
2.  cmd.setRemoteChannel(channelID);
3.  cmd.setMac(mac);
```

```
4.  cmd.setUseNewConnectWay(false);
5.  dongleManager.send(cmd, new CommandCallback() {
6.   @Override
7.   public void onSent(BaseCommand command) {
8.    // 发送成功的回调
9.   }
10.
11.  @Override
12.  public void onError(ErrorCode code) {
13.   // 发送失败的回调
14.  }
15.
16.  @Override
17.  public void onResponse(BaseCommand command) {
18.   //  设备回复对应命令
19.  }
20. });
```

2.6. 断开远端设备

2.6.1. 功能描述

断开远端 HID 设备，设备须要回复。

●setRemoteChannel(int remoteChannel)

设置设备对应的通道号

●int getResult()

返回是否断开成功的设备状态, 0 表示成功

2.6.2. 使用示例

```
1.  DisconnectRemoteCmd cmd = new DisconnectRemoteCmd();
2.  cmd.setRemoteChannel(channel);
3.  dongleManager.send(cmd, new CommandCallback() {
4.   @Override
5.   public void onSent(BaseCommand command) {
6.    // 发送成功的回调
7.   }
8.
9.   @Override
10.  public void onError(ErrorCode code) {
11.   // 发送失败的回调
```

```
12.  }
13.
14.  @Override
15.  public void onResponse(BaseCommand command) {
16.   //  设备回复对应命令
17.   DisconnectRemoteCmd cmd = (DisconnectRemoteCmd)command;
18.   cmd.getResult();//0：成功，1：失败
19.  }
20. });
```

2.7. 自定义命令

2.7.1. 功能描述

当已有命令或接口不能满足当前需求时，可以通过自定义命令的接口来实现自己的命令和数据。自定义命令数据结构应该由 App 端和设备端的开发人员共同协商和实现。接收端须要回复。

●byte[] getData()
设备返回的自定义数据
●setData(byte[] data)
设置自定义数据

2.7.2. 使用示例

```
20. CustomCmd cmd = new CustomCmd();
21. cmd.setData(rcspData);
22. dongleManager.send(cmd, new CommandCallback() {
23.  @Override
24.  public void onSent(BaseCommand command) {
25.   // 发送成功的回调
26.  }
27.
28.  @Override
29.  public void onError(ErrorCode code) {
30.   // 发送失败的回调
31.  }
32.
33.  @Override
34.  public void onResponse(BaseCommand command) {
```

```
35.    //   设备回复对应命令
36.    CustomCmd customCmd = (CustomCmd) command;
37.    byte[] data = customCmd.getData();
38.   }
39. });
```

2.8. 设备认证标志

2.8.1. 功能描述

设置设备认证标志，设备须要回复。

●setDeviceChannel(int deviceChannel)

设置对应设备 channel

●setAuthed(boolean authed)

设置设备认证标志

●int getResult()

设备返回操作结果, 0 表示成功

2.8.2. 使用示例

```
1.  DeviceAuthFlagCmd cmd = new DeviceAuthFlagCmd();
2.  cmd.setAuthed(isAuthed);
3.  cmd.setDeviceChannel(channel);
4.  dongleManager.send(cmd, new CommandCallback() {
5.   @Override
6.   public void onSent(BaseCommand command) {
7.    // 发送成功的回调
8.   }
9.
10.  @Override
11.  public void onError(ErrorCode code) {
12.   // 发送失败的回调
13.  }
14.
15.  @Override
16.  public void onResponse(BaseCommand command) {
17.   //   设备回复对应命令
18.  }
19. });
```