乐行天下激光雷达sdk使用简介

# 简介

乐行天下激光雷达可以同时提供windows，linux，android三个版本的sdk。由于绝大多部分的客户使用的是android系统，所以本文档着重以android接口为讲解，win平台下提供demo程序，如果有需要android平台外其他平台的接口，请联系文档作者：[guankang@imscv.com](mailto:guankang@imscv.com)。

接口文档以及demo程序可以从github上进行下载，下载链接：<https://github.com/navipack/host-sdk>。

# 雷达的基本使用与配置

## 启动需求

雷达启动需要供电口提供800mA以上电流方可正常启动，启动的时候可以观察算法板上两个指示灯，两个指示灯会以先后顺序启动，如果两个指示灯皆正常点亮，且雷达转动，则基本正常。



图2-1 程序正常启动的指示灯状态

## 启动现象

将雷达插入到任何主机设备上都将会虚拟出两个节点，一个是虚拟串口节点，一个是android device设备节点。

在windows平台下表现为设备管理器出现两个节点，一个android设备节点，一个是usb串行口节点。如下图所示：



图2-2 雷达与windows连接端口

在linux平台下或者android平台下，将会在 /dev节点下生成一个ttyACMX的设备节点。执行adb devices可以看到一个android设备，则表示启动成功。

## 网络配置

如果需要雷达的网络配置，即让雷达能够联网实现通过wifi进行开发，需要对雷达进行wifi配置。需要说明的是，暂时雷达模块仅支持指定的网卡进行连接。后续也会更新优化。当前情况下支持的网卡型号为**8192cu**芯片的wifi模块，通过测试的模块链接：[https://item.taobao.com/item.htm?spm=a230r.1.14.58.Z1QTOX&id=7915403646&ns=1&abbucket=17#detail](https://item.taobao.com/item.htm?spm=a230r.1.14.58.Z1QTOX&id=7915403646&ns=1&abbucket=17" \l "detail)。

### 通过sdk对wifi进行配置

通过sdk对wifi配置需要使用串口与雷达模块连接，调用setWifiParam函数（方法）对wifi进行配置。但是此方法配置成功后暂时没有办法直接获取设备ip地址。后续会增加获取接口，所以暂时不推荐使用。

### 通过shell命令直接配置wifi

通过shell命令配置wifi是最常见且最稳定的配置方式。配置方式如下：

1. 确保雷达设备已经连接到了系统上，执行adb devices能够发现设备节点。

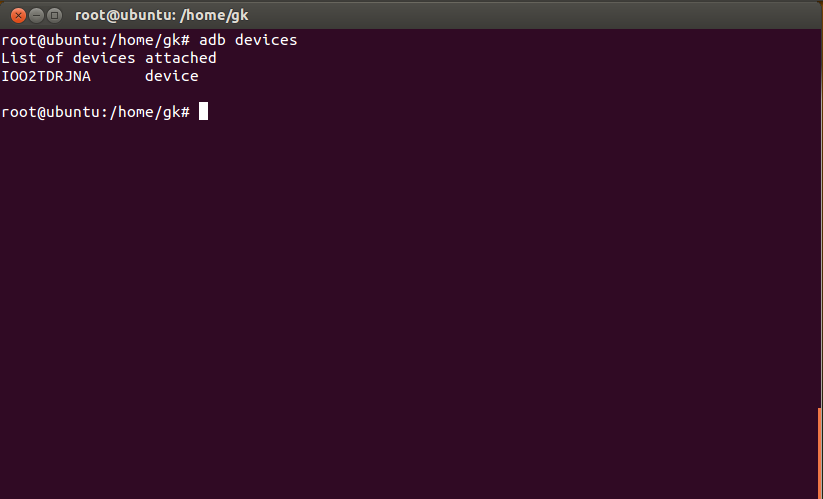


图2-3 发现雷达设备列表

1. 执行adb shell进入系统

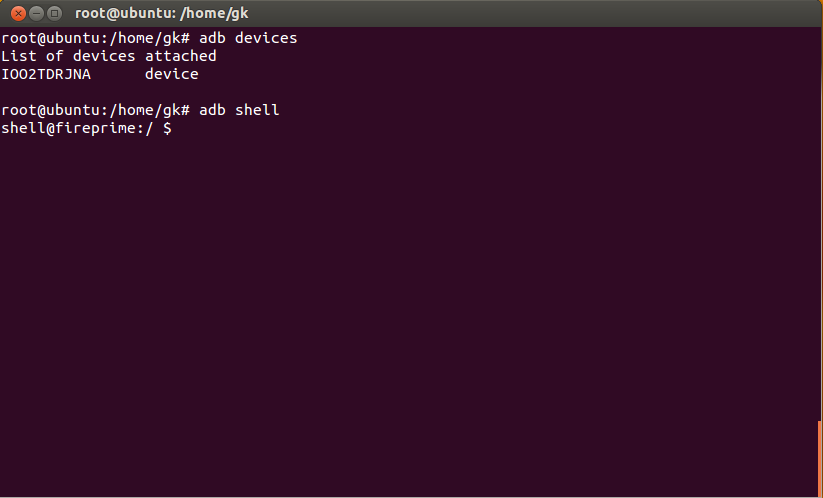


图2-4 进入雷达设备

1. 切换到超级用户 执行 su

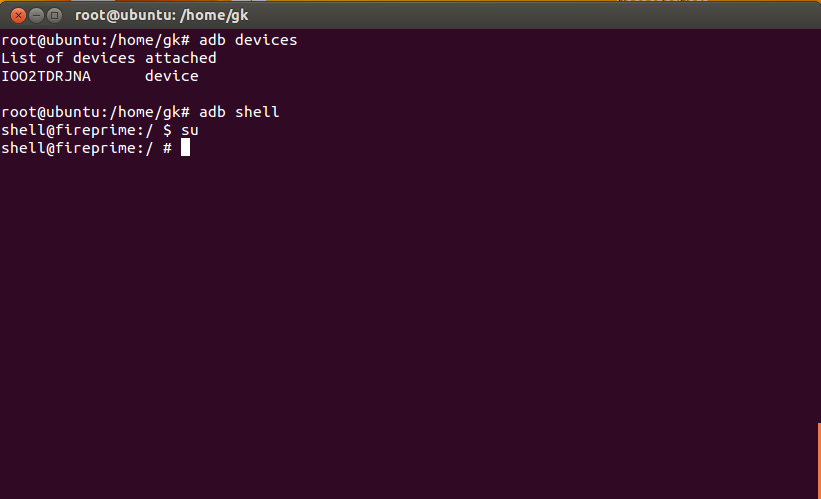


图2-5 切换到超级用户权限

1. 配置wifi 执行 wifi\_start.out ssid password（其中ssid指的 是wifi名字，password为wifi密码）

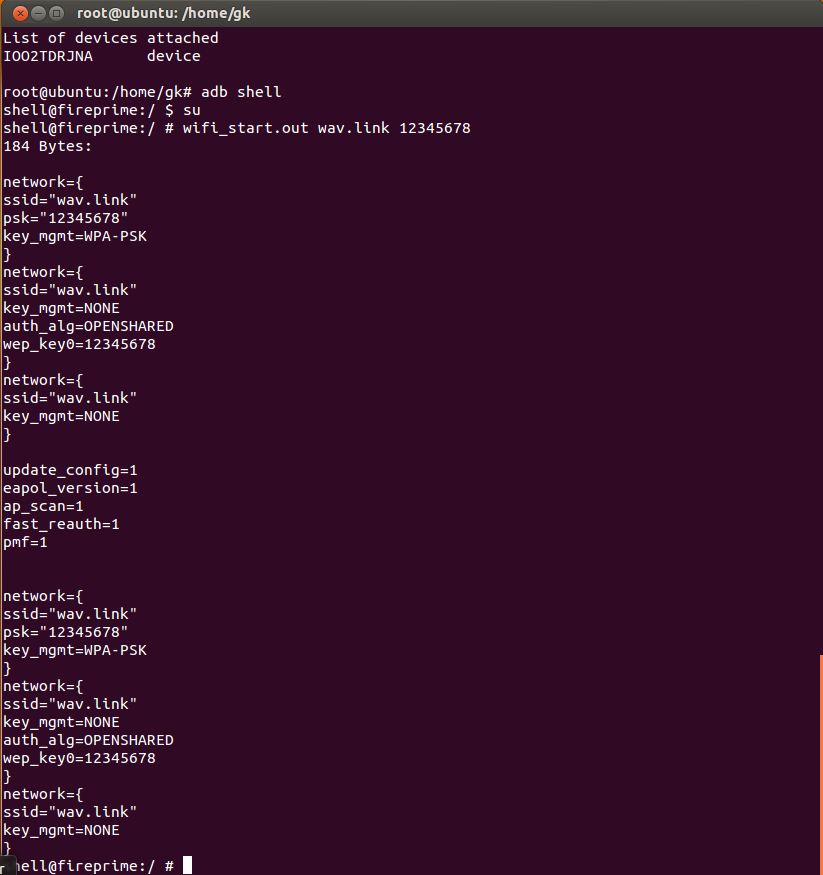


图2-6 网络配置及其正常输出

1. 等待10秒左右，使用netcfg命令查看当前的ip,如果结果为下图：

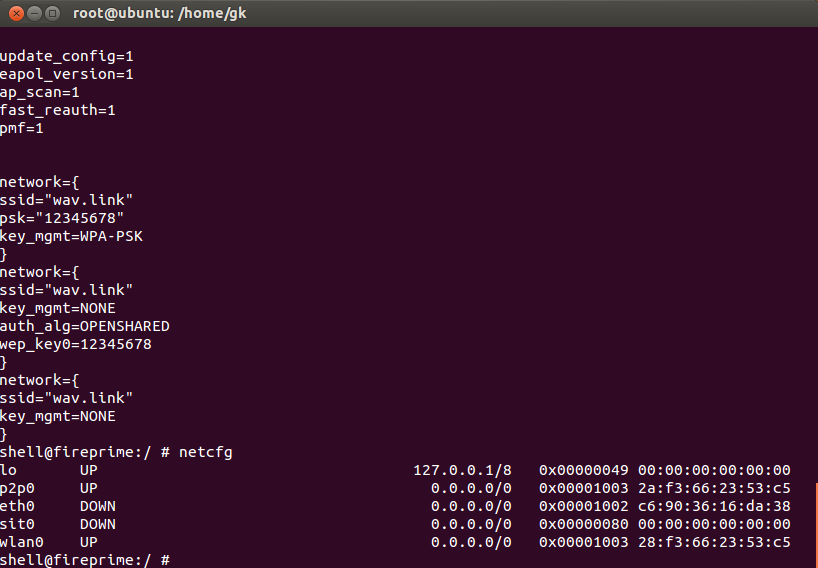


图2-7 未正常配置网络的信息

从图中可以看出wlan的ip并没有出现，说明没有配置成功，此时需要确定ssid以及password是否设置正确，如果都正确则重新执行步骤4.直到出现下图:

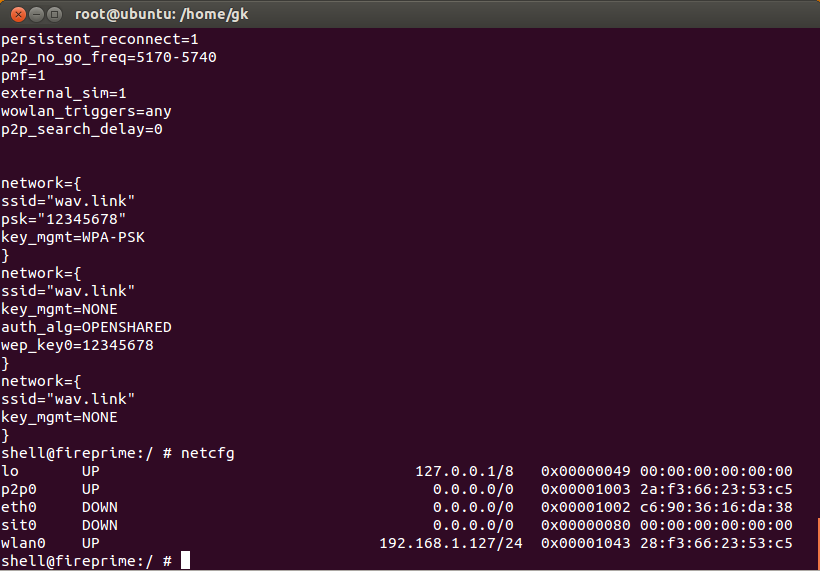


图2-8 正常配置网络的信息

此时，雷达的ip配置为192.168.1.127.一般情况下，雷达只要在wifi的覆盖范围内，即使重启路由器，该ip也不会改变了。如果要确保IP不变，可以使用路由器给该网卡分配静态IP。

1. 到此，wifi配置工作完毕。如果需要重新配置wifi信息，只需要重复上述步骤即可。

## 运行程序的统一性

### 版本统一性

要让程序正确运行与通讯，需要使用配套版本进行开发测试，即本地sdk版本与雷达navipack版本需要保持一致。本地sdk版本可由接口getSdkVersion获得。雷达navipack的版本可以通过接口setGetNaviPackVersion来向navipack程序拉取，并通过解析navipack消息获取。如果两者程序不一致，将会导致意想不到的事情发生，比如地图错位等。

### 更新navipack程序

为保证程序的统一性。在github中，有NaviPackExe文件夹中保存和当前sdk对应的NaviPack版本。将对应的navipack版本更新到雷达中也可以通过接口以及shell两种方式更新。

通过接口，则需要上位机与雷达以网络或者串口的方式连接，调用setUpdateNaviPackFile接口来发送升级文件进行升级，升级消息通过解析设备消息获得。如果升级成功，则会有对应的消息放出，并且在下次重启雷达时生效。

通过shell升级，下面讲述通过有线方式传输（无线需要使用adb connect命令，与之类似不做过多说明），即把NaviPack程序直接push到系统板中。需要确保自己电脑或者其他上位机能够通过adb和雷达连接。即执行 adb devices 的时候可以获取到android设备，并且设备状态的device，如果设备状态为offline则需要重新拔插设备。下面的流程基于已经能够看到非offline的一个android设备：

1. 执行adb root
2. 执行 adb push NaviPack /data/shelly\_robot ;其中NaviPack为github上NaviPackExe文件夹中的可执行程序 。

## 雷达配置参数说明

雷达放到不同的底盘上应当有不同的配置信息方可正常使用，配置信息主要包含底盘的传感器信息以及雷达的安装参数等。而这个配置信息也是需要上传到雷达中去的。

## 配置信息文件

### 配置文件内容如下：标红为重要参数（后续可能改动以最新改动版本为准）

;本配置文件包含了激光雷达套件所需要配置的信息

;背景知识：载体的质点默认在两个驱动轮中心，载体位姿即是质点位姿；

;载体坐标系零点在质点，正前方为X方向，Y方向符合右手法则；

;各个传感器参数都是相对载体坐标系。

;useDefaultConfig:

; 0表示不使用默认配置，此时，其他配置没有意义

; 1表示要使用默认配置，此时，配置信息为后面参数

[ChassisParam]

useDefaultConfig=0

;车的形状参数 可以是三边形、四边形，及多边形。不建议采用多边形表示

chassisShapeParamNum=4

;车子的形状参数X(相对轮轴中心)

chassisShapeParamX=-285,285,285,-285

;车子的形状参数Y(相对轮轴中心)

chassisShapeParamY=-285,-285,285,285

**;是否有IMU传感器**

**bhasIMUSensor=1**

**;没有IMU情况下，需要该轮间距参数**

**WheelDistance=338**

[UltraSensorParam]

useDefaultConfig=1

;超声波传感器个数

ultrasoundSensorNum=8

;相对轮轴中心安装位置

ultrasoundSensorX=222,178,62,0,0,0,62,178

ultrasoundSensorY=0,116,164,0,0,0,-164,-116

;超声波朝向

ultrasoundSensorOrientationAngle=0,45,90,0,0,0,270,315

;超声波用于NaviPack的最小视距（不建议修改）

ultrasoundSensorMinMeasureDistance=30

;超声波用于NaviPack的最大视距（不建议修改）

ultrasoundSensorMaxMeasureDistance=600

;超声波视场角

ultrasoundSensorFOV=15

[DropSensorParam]

useDefaultConfig=1

;跌落传感器个数

dropSensorNum=4

;跌落传感器相对轮轴中心安装位置

dropSensorX=216,127,127,216

dropSensorY=56,152,-152,-56

[CollisionSensorParam]

useDefaultConfig=1

;碰撞传感器开关量个数

collisionSensorNum=0

;碰撞传感器相对轮轴中心安装位置

collisionSensorX=

collisionSensorY=

[LidarSensorParam]

useDefaultConfig=1

**;激光雷达传感器相对轮轴中心安装位置**

**lidarSensorX=-100**

**lidarSensorY=0**

**lidarSensorOrientationAngle=0**

;盲区距离

BlindRadius=285

;最远测距

ValidDistance=8000

[NaviPackParam]

useDefaultConfig=1

maxLineVelocity=500

minLineVelocity=100

maxAngularVelocity=45

minAngularVelocity=0

**;使能超声波传感器用于Navipack**

**enableUltrasound=0**

**;使能开关量传感器（跌落、碰撞）用于Navipack**

**enableSwitchSensor=0**

;使能多传感器数据融合用于导航

enableUsingTotalMapForNavigation=0

# sdk使用

## Android使用

android使用在github上有完整的demo，具体可以参考javadoc文档说明，这里只说工程配置。

### Android工程配置

1. 建立空工程

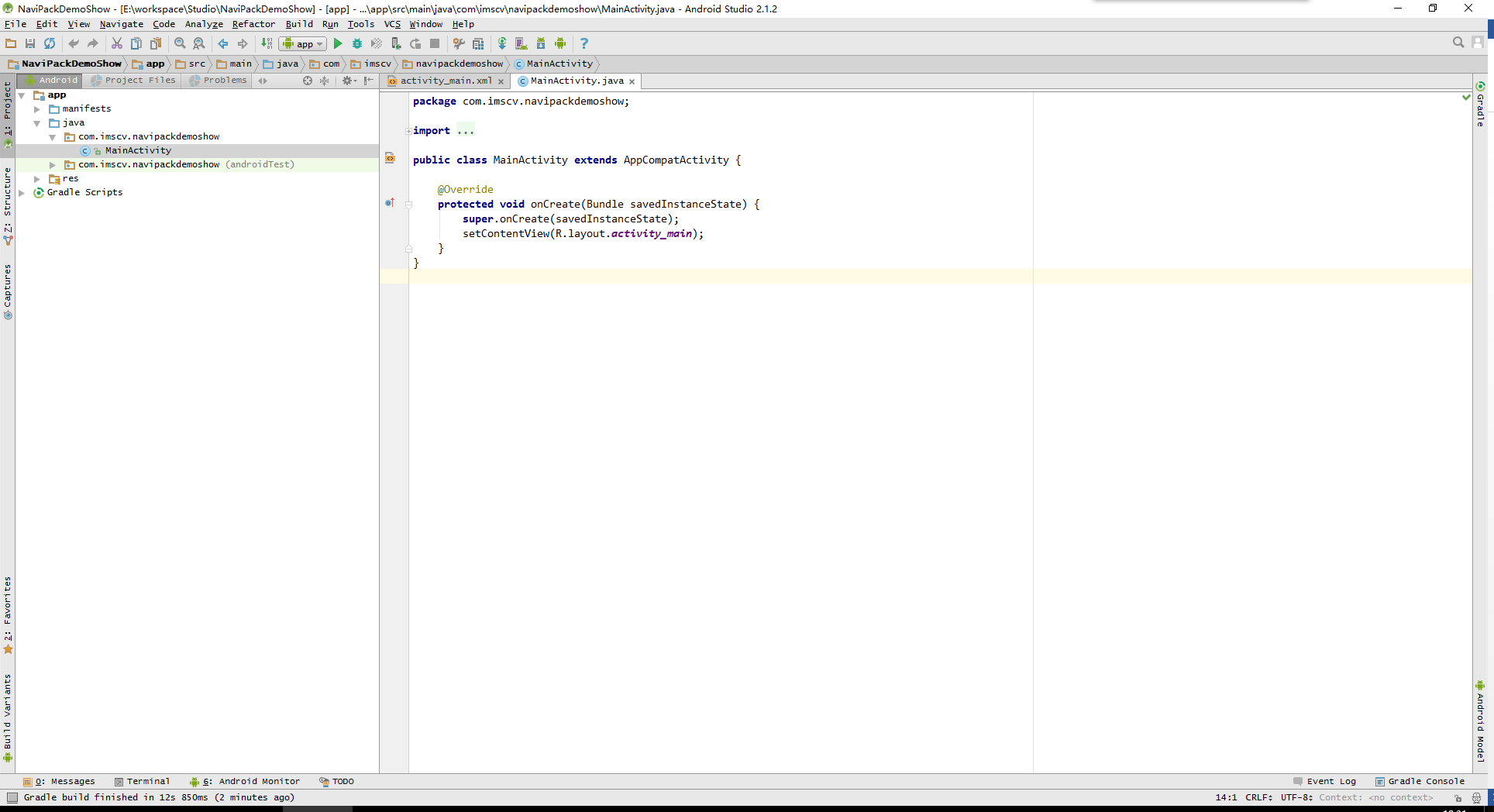


图3-1 android空工程的建立

1. 添加navipacksdk module. File—New—Import Module… 选择navipacksdk所在的文件夹

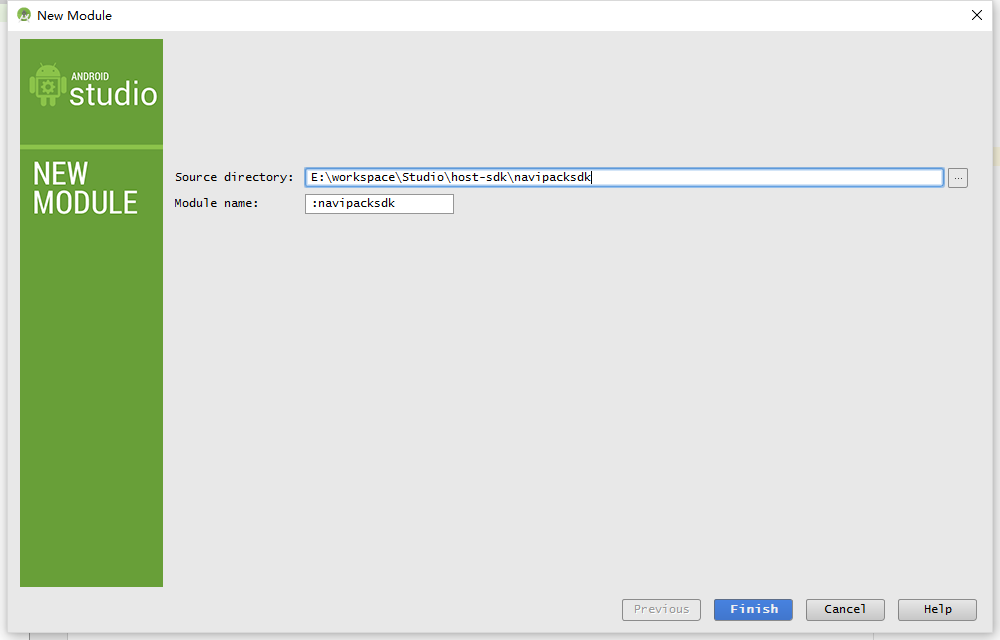


图3-2 android工程添加module

1. 此时文件结构如下：

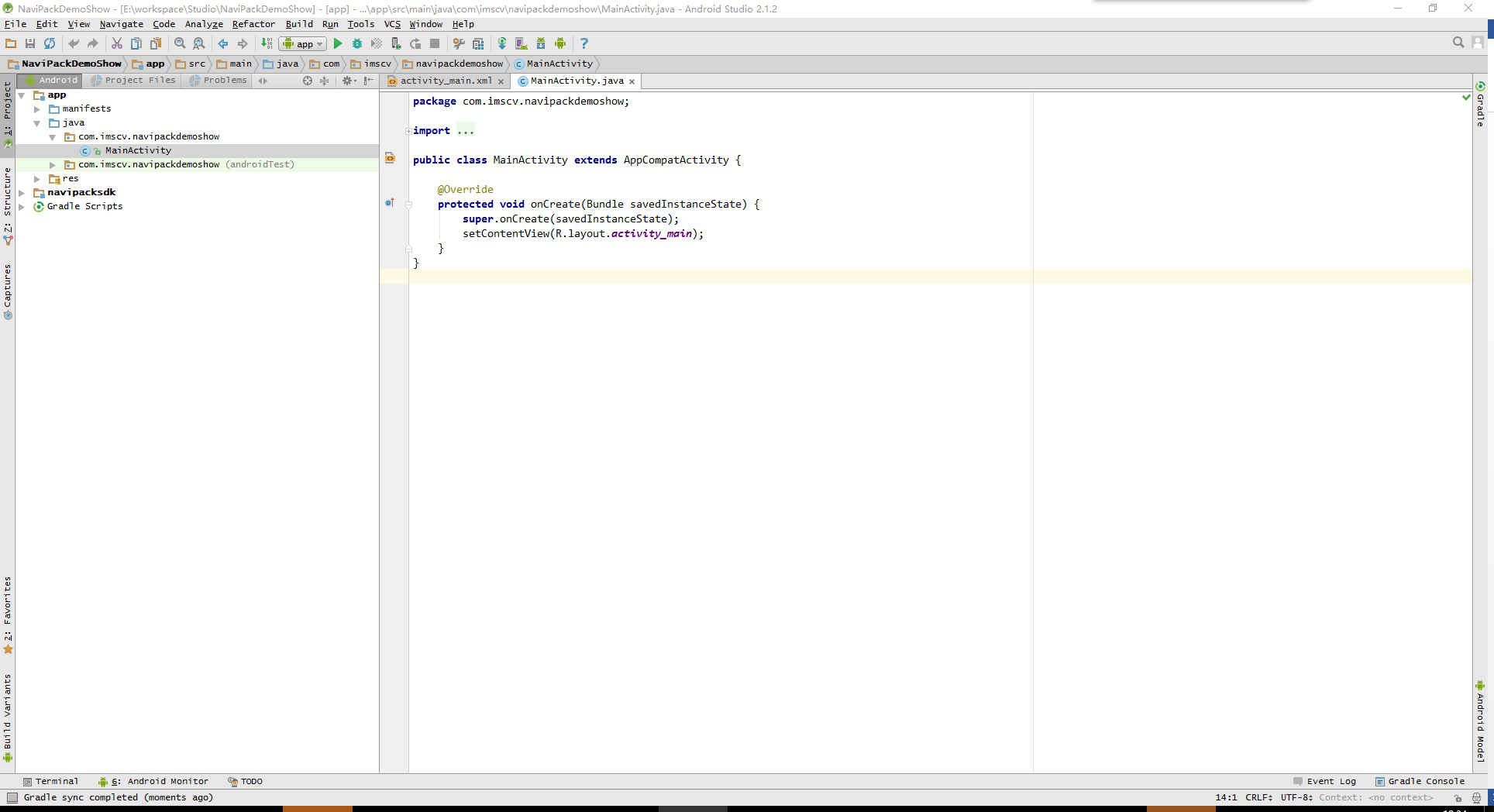


图3-3 android 添加module后的结构

到此，空工程的配置结束。

### Android与NaviPack的连接

1. 创建NaviPackSdk单例,mNaviPackSdk = NaviPackSdk.getInstance();
2. 创建NaviPack对象ID，id = mNaviPackSdk.createHandler(conType);
3. 进行连接 mNaviPack.open(id,IP\_OR\_DEVICE\_POINT,0,listener);
4. 如下图：

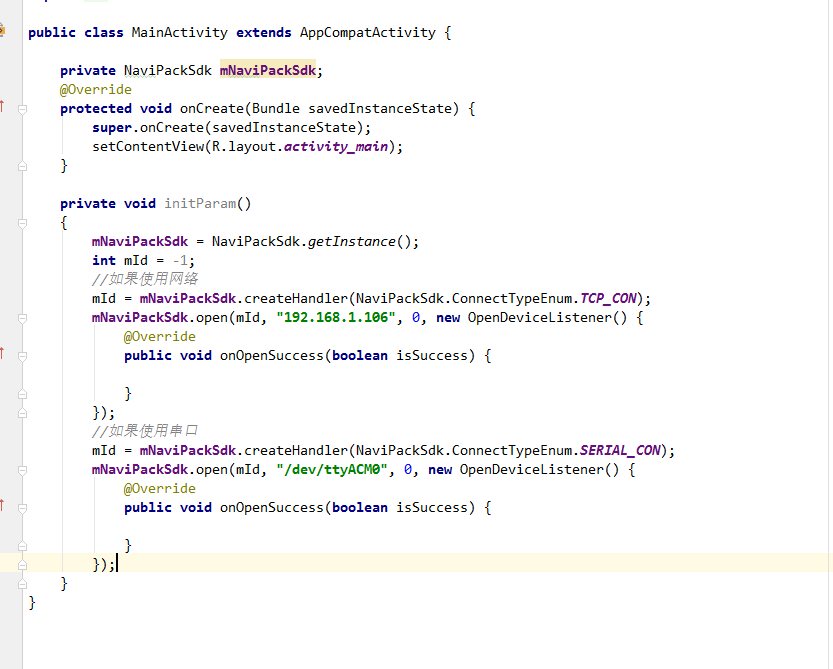


图3-4 android Sdk的最基本的连接操作

连接成功后所有的通讯可以参考demo程序。

## Sdk重要回调方法

Sdk中有两个回调需要特别说明，其包括了几乎所有的通讯包信息。

### DeviceMsgListener

该方法回调navipack的消息。包括且不限于地图更新数据，版本信息，传感器数据，寄存器数据等。其回调的msgType的值在NaviPackType中有声明。其功能码也在javaDoc中有指出。

### DeviceErrorMsgListener

该方法回调navipack中的比较有用的消息，其中这些错误消息大多为提示性的消息。如果有需要使用，则捕捉相关的错误码即可。