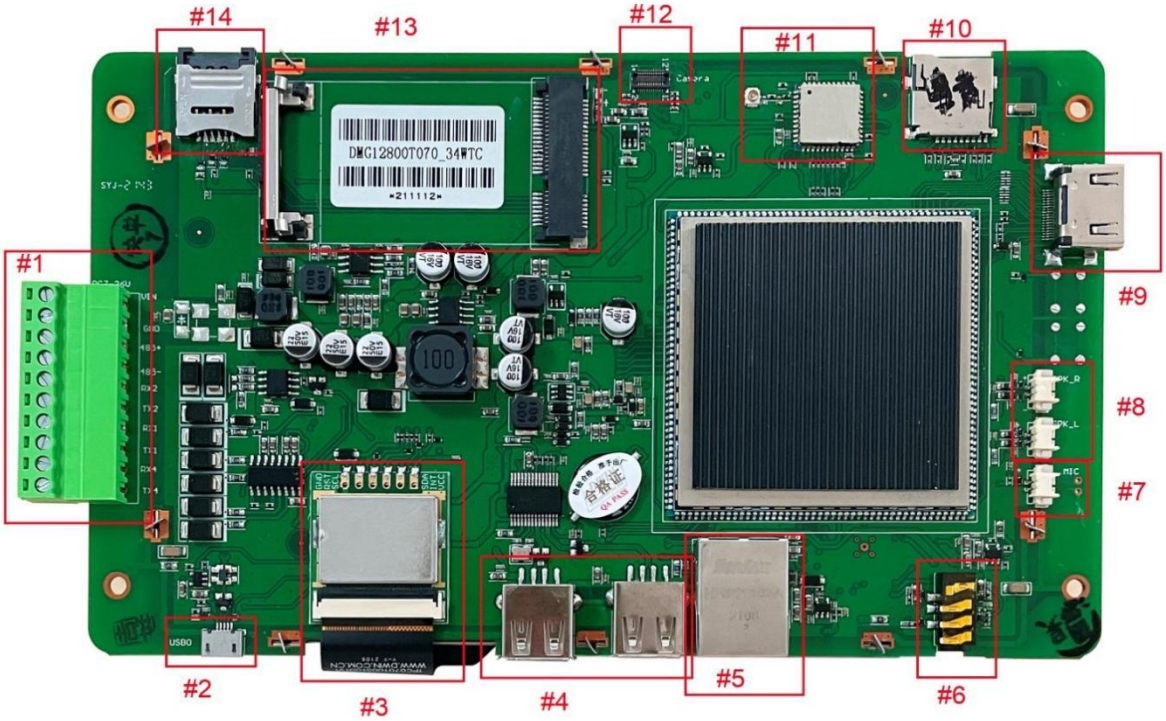
Development Guide for DWIN Android Screen

Contents

1 Introduction of hardware



Number	Function Description
#1	Power supply and UART
#2	USB debugging interface
#3	TP interface
#4	USB HOST interface
#5	Ethernet interface
#6	Headphone jack
#7	Microphone interface
#8	Speaker interface
#9	HDMI interface
#10	SD card slot
#11	WIFI and antenna
#12	MIPI camera interface
#13	4G module slot
#14	SIM card slot

2 Android Studio development environment

2.1 Tools

android-studio-ide-191.5977832-windows.exe

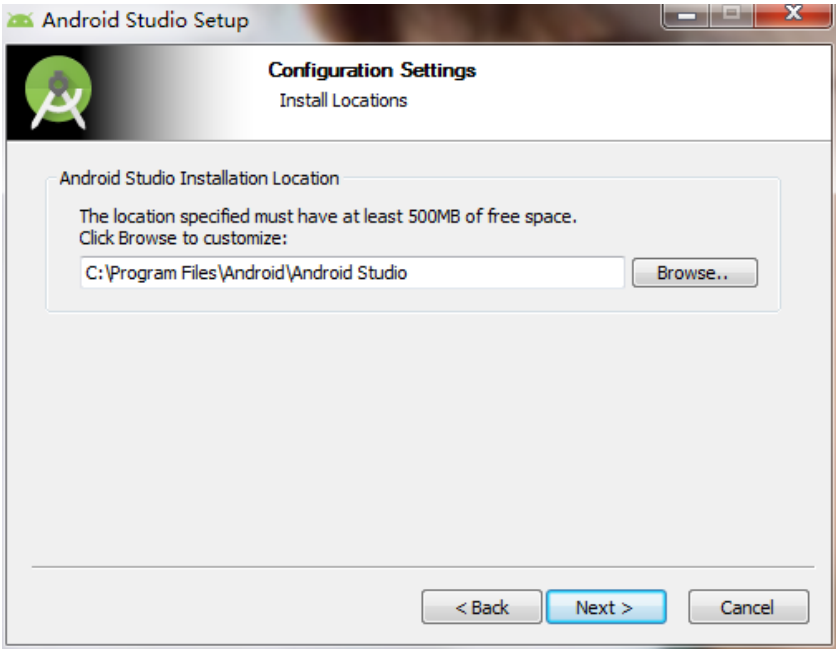
The above APK can be downloaded from https://developer.android.google.cn/studio/archive.

(Android Studio 3.6 or later versions)

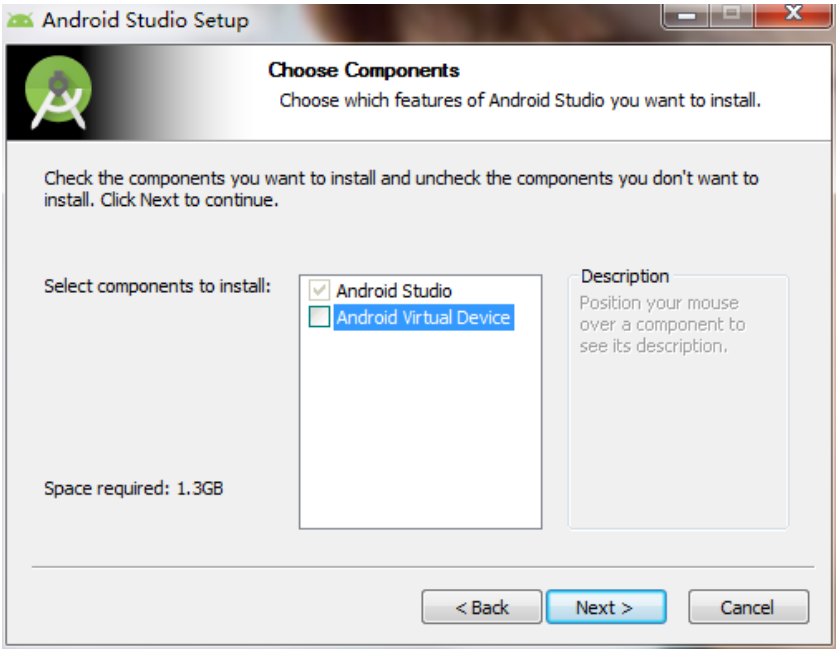
2.2 Software installation

a) Double-click the .exe file to install Android Studio.

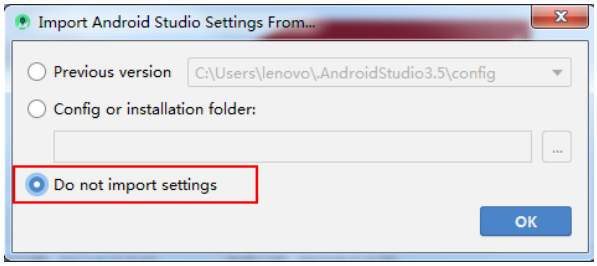
Set the installation location.



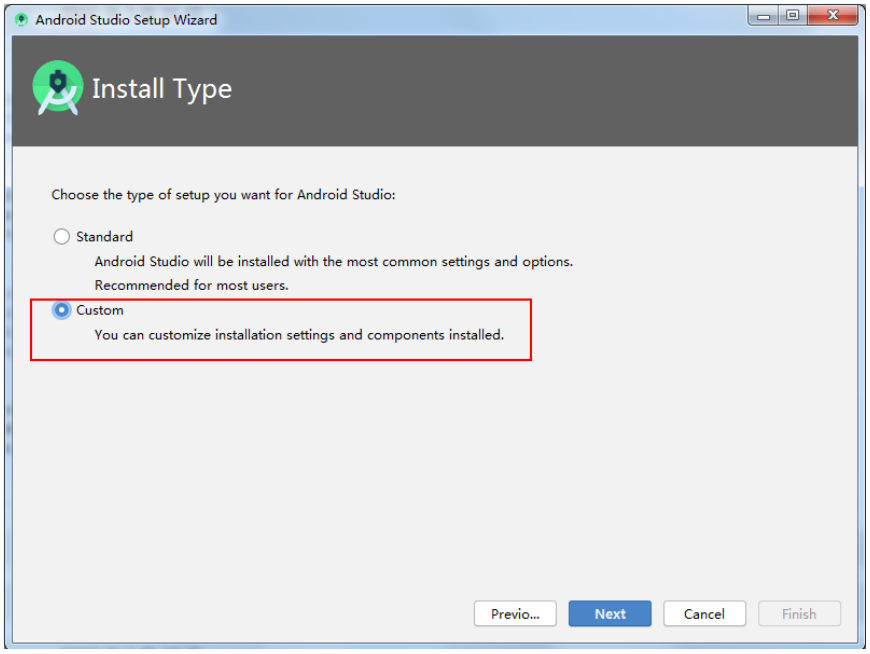
Choose whether you need AVD (Android Virtual Device).



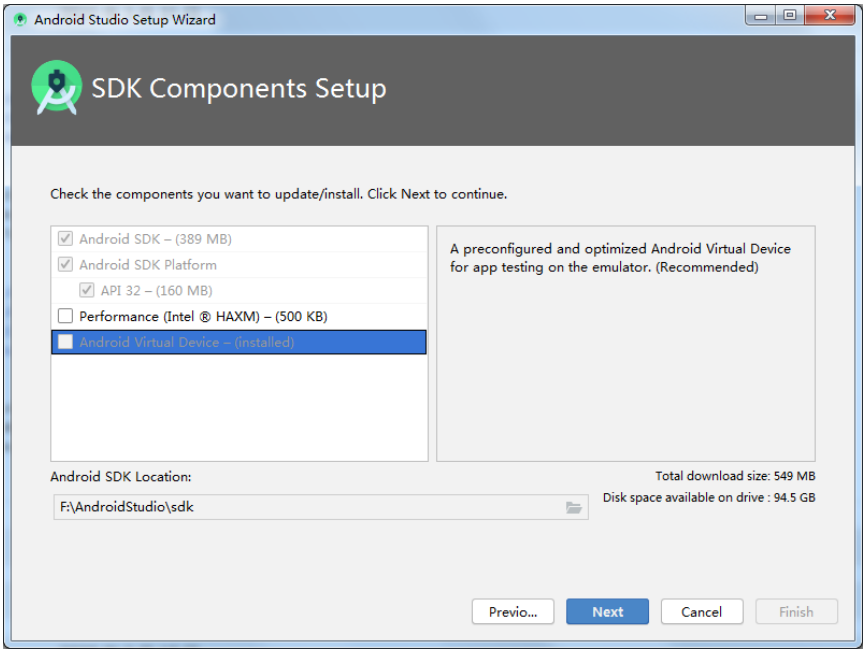
It is recommended to choose not to import settings for the first time to use Android Studio.



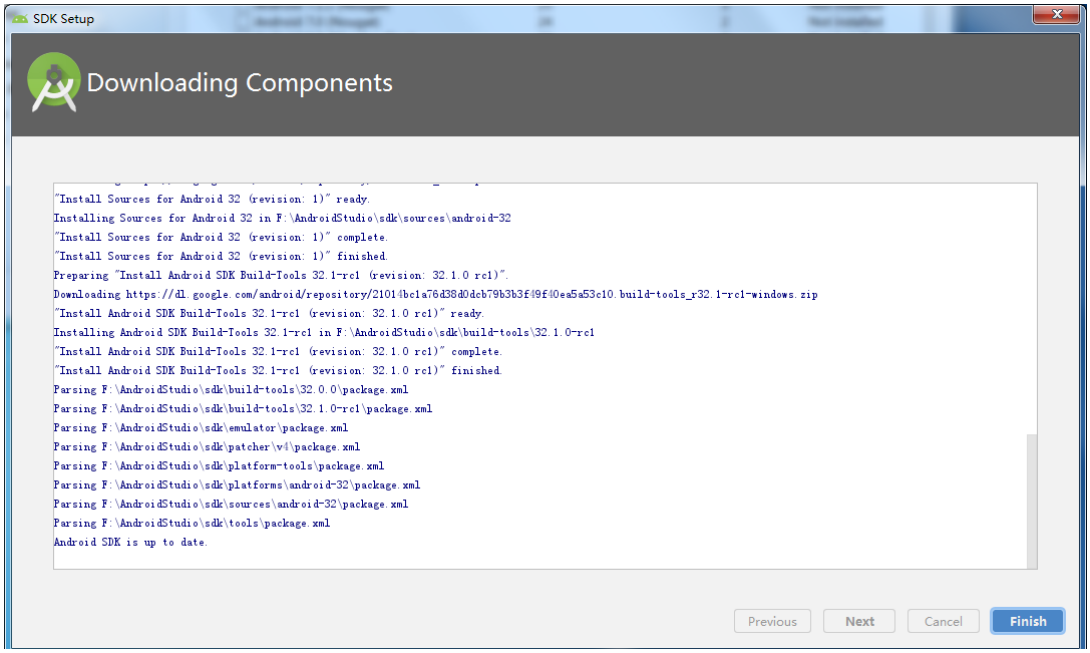
Select custom for install type.



Set SDK location.

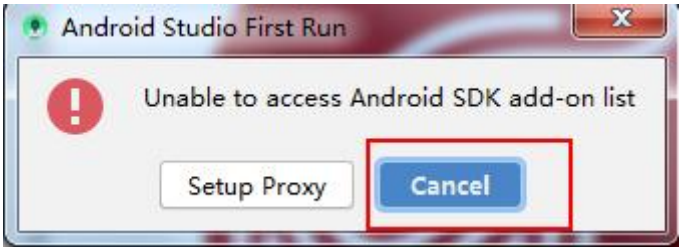


Wait for installation to complete



b) Open the installed Android Studio and create a new project

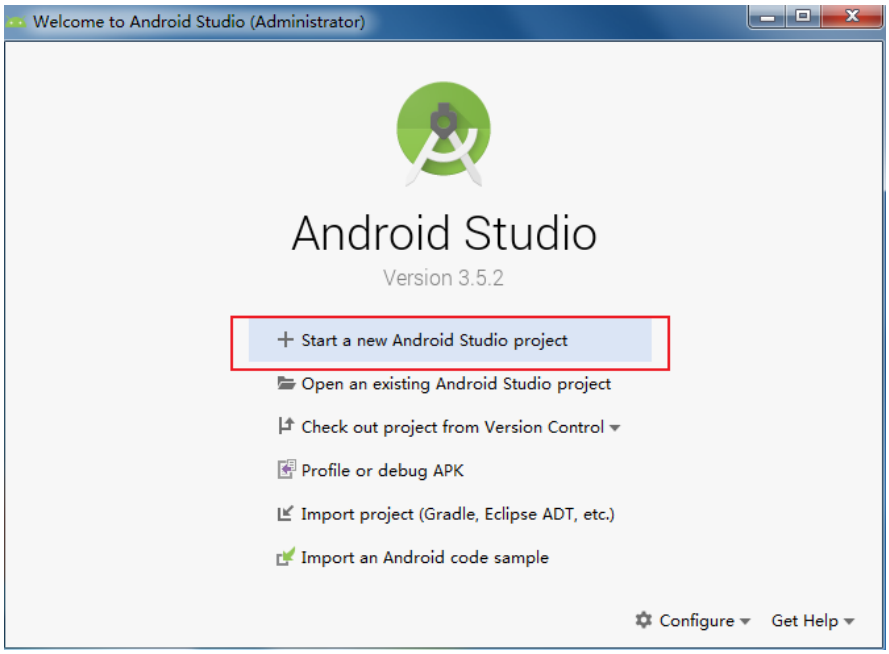
The first time you open it, the following error will appear, select [Cancel] to close it.



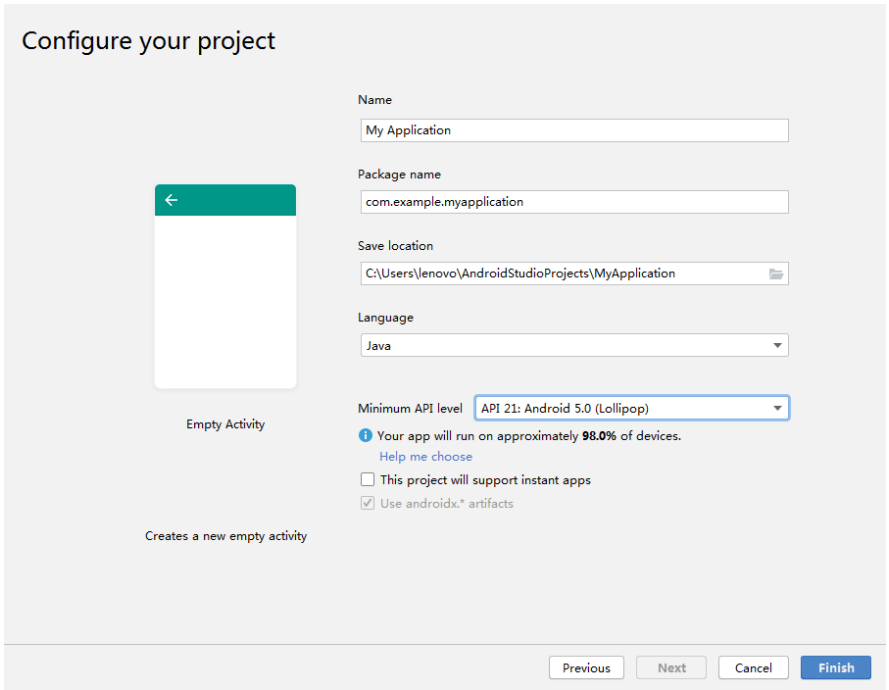
For the first run, you need wait for downloading and automatic configuration of the environment.

2.3 Create a new project

a) Select [Start a new Android Studio project]

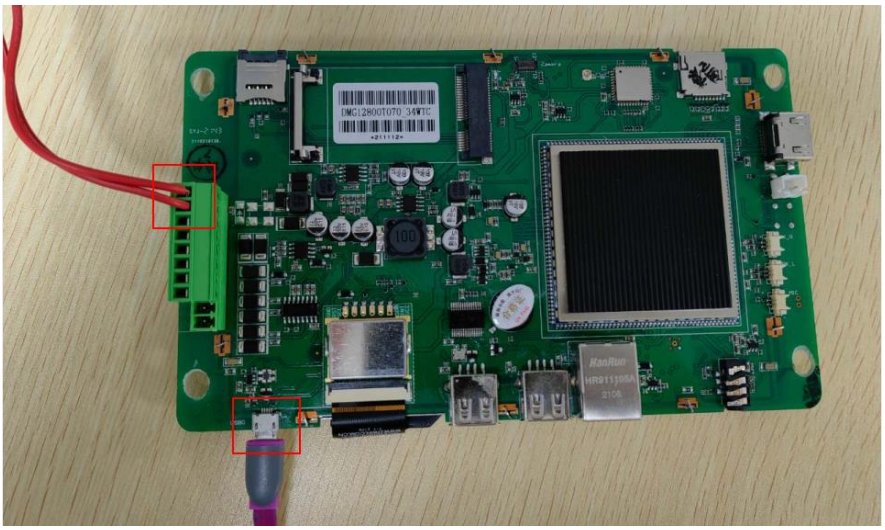


b) Fill in the project name, package name, set the save location, programming language (Java is used in the example). Set the minimum API level (in this case we choose 5.0).



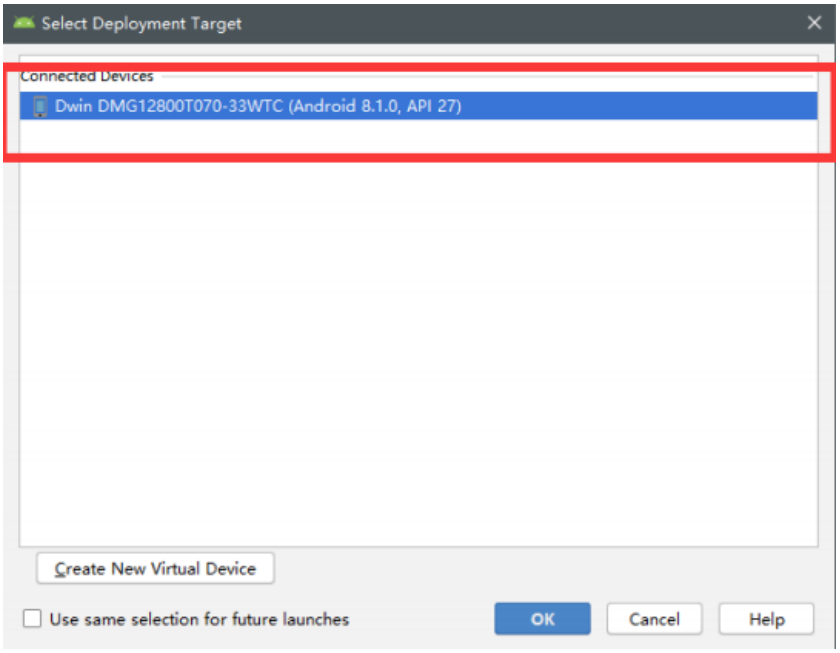
c) Compile and run the first project.

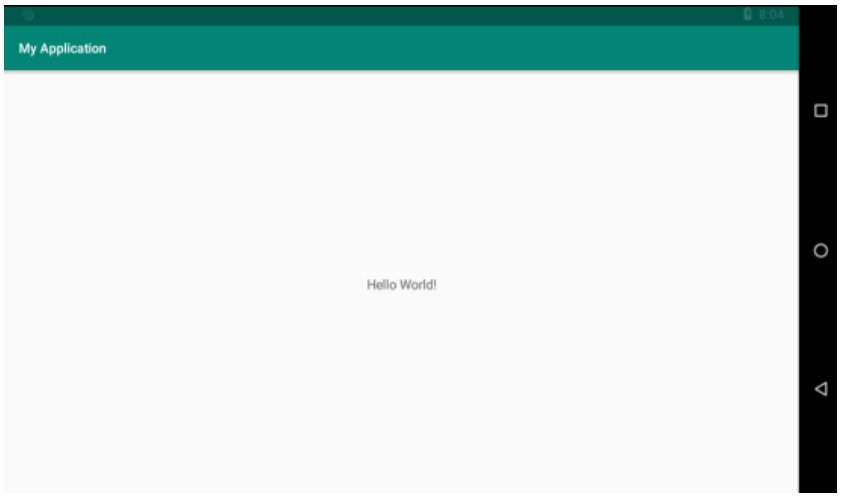
You need to connect the Android screen and PC in advance as shown below and then select debug operation.



As the figure shows, the Android screen is connected to the PC by a USB cable, and the Android screen is powered on. (The layout of each Android screen may not be the same, the figure is for reference only).

Then select DWIN screen, click OK, and the program demo will run as follows.

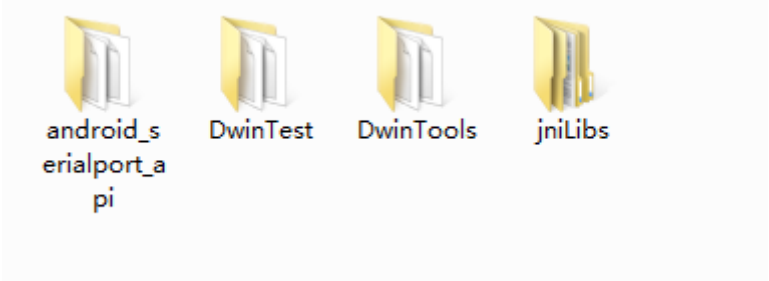



My Application
Hello World!

3 DWIN Android tools

3.1 DWIN android toolkit

DWIN provides android test software source code and serial java library files as follows.



jniLibs: encapsulated so library

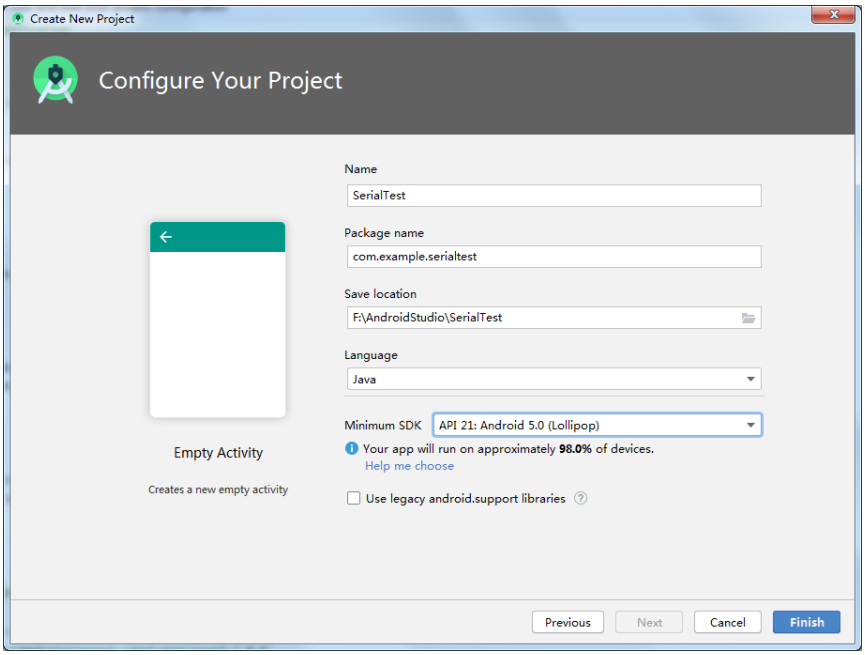
android_serialport_api: call library tool

DwinTools: complete source code of the DWIN toolkit project

DwinTest: DWIN test software

3.2 Create a new project by DWIN Java serial libs

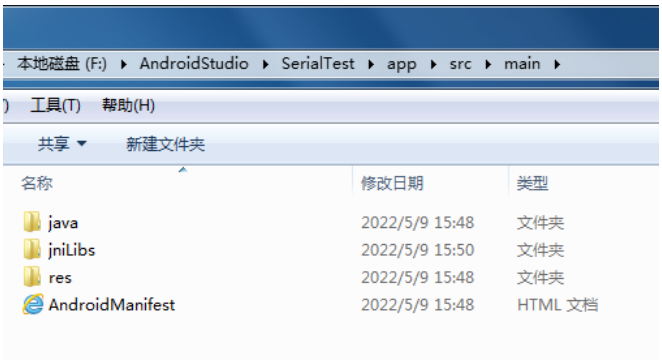
a) Create a new project. For example, name it as SerialTest, save it at “F:\AndroidStudio\SerialTest” and set the programming language as Java.

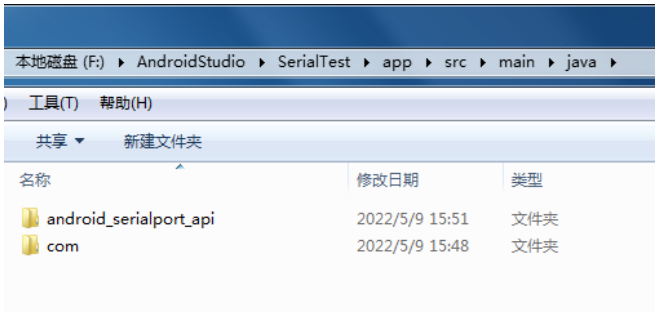


b) Copy jniLibs in the android toolkit to “F:\AndroidStudio\SerialTest\app\src\main”.

Copy the android_serialport_api folder to “F:\AndroidStudio\SerialTest\app\src\main\java”.

Note: The directory must be named as android_serialport_api and cannot be changed as shown below.





c) After completing the above steps, you can operate the UART. The specific operation can be done according to each Activity of DwinTools\app\src\main\java\com\dwin\DwinTools\serial of the DWINTools project source code in the code package for reference and then modify and use according to your own project.

Application.java is integrated with android.app.Application, mainly used to generate SharedPreferences, by which the baud rate and serial device are configured. In general, no need to modify.

SerialPortActivity.java is used to create input and output streams and threads for operating serial port read and write. Users can use the Activity inheritance to read and write.

SerialPortPreferences.java is used to scan the device for available serial devices ttyS*, and then write to SharedPreferences for configuration, which can be rewritten by users according to their app.

ConsoleActivity.java is integrated with SerialPortActivity for serial port character type send/receive operation.

HexConsoleActivity.java integrated from SerialPortActivity, used for serial port Hex type send/receive operation.

SerialMainMenu.java main directory Activity for jumping to the specified Activity.

Sending01010101Activity.java is used to keep sending 0101 to test the serial port waveform.

LoopbackActivity.java used to test whether the serial port sending and receiving function is normal.

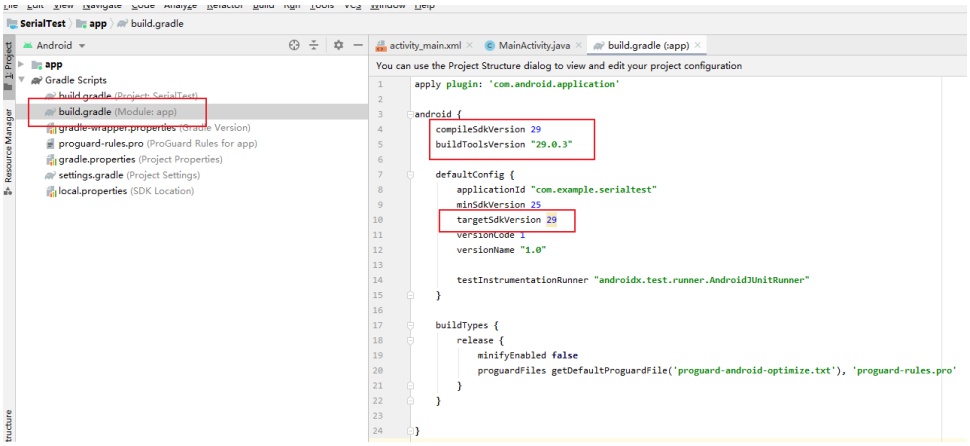
HexHelper.java a series of tool classes for converting between characters and hex.

Before use, you need to change the configuration in build.gradle (Module:app) to the following.

compileSdkVersion 29

buildToolsVersion "29.0.3"

targetSdkVersion 29



3.3 Common functions of DWIN Android screen

Most of the special functions of the DWIN Android screen are operated in the form of sending broadcasts.

3.3.1 Dynamic restart

The dynamic restart can be realized by sending the broadcast of android.intent.action.DWIN_reboot

```
Intent sr = new
Intent("android.intent.action.dwin_reboot");
sendBroadcast(sr);
```

3.3.2 Dynamic return/ return to home page

The dynamic return can be realized by send broadcast `android.intent.action.dwin_input_back_key`

The return to home page can be realized by broadcast

android.intent.action.dwin_input_home_key

```
Intent sr = new
Intent("android.intent.action.dwin_input_back_key");
sendBroadcast(sr);

Intent sr = new
Intent("android.intent.action.dwin_input_home_key");
sendBroadcast(sr);
```

3.3.3 Read chip_id

By the getChipId() method in the tool class ChipUtil in the given DWINTools source code, the value of chip_id can be acquired.

```
    break;
case R.id.btn_get_chipid:
    // Obtain Chin ID
    String chipId = ChipUtil.getInstance().getChipId();
    mTextViewChipId.setText("Chip ID : " + chipId);
    break;
```

3.3.4 To rotate screen by app

This can also be realized by sending a broadcast. The related code is in the RotationActivity class in DWINTools, and the system will reboot after a successful broadcast, and the screen will be rotated.

```
@Override
public void onClick(View v) {
    switch (v.getId()){
        case R.id.btn_rotation_0:
            setRotation("0");
            break;
        case R.id.btn_rotation_90:
            setRotation("90");
            break;
        case R.id.btn_rotation_180:
            setRotation("180");
            break;
        case R.id.btn_rotation_270:
            setRotation("270");
            break;
    }
}

/**
 *  By sending a broadcast, the system will hide/show the
 * @param value Rotation value
 */
private void setRotation(String value){
    Log.v(TAG, msg: "Dwin test sendBroadcast!");
    Intent sr = new Intent(DWIN_ROTATION);
    sr.putExtra( name: "message", value);
    sendBroadcast(sr);
}
```

3.3.5 To set navigation bar permanently hidden by app

This can also be realized by sending a broadcast.

The relevant code is in the NavigationBarActivity class in DWINTools, and the system will reboot after a successful broadcast, and the status of the navigation bar will be modified.

```
// Send"0"to display navingation bar,and"1"to hide navigation bar
public static final String DWIN_NAVIGATION_BAR_SHOW_VALUE = "0";
public static final String DWIN_NAVIGATION_BAR_HIDE_VALUE = "1";

/**
 *  By sending the broadcast, the system will rotate the screen
 * @param value Whether to display the value of the guide bar
 */
private void setNavigationBar(String value){
    Log.v(TAG, msg: "Dwin test sendBroadcast!");
    Intent sr = new Intent(DWIN_NAVIGATION_BAR);
    sr.putExtra( name: "message", value);
    sendBroadcast(sr);
}
```

3.3.6 Automatic startup after booting

There are generally two ways to start automatically after booting: one is to start after receiving the boot

broadcast, the other is to set your own APK to Launcher.

In the former situation, the APK is started after the system desktop is entered. While in the latter situation, the APK is directly started without the system desktop being entered.

a) APK started after entering the system desktop

Background knowledge: When Android boots, a system broadcast will be sent with the content of ACTION_BOOT_COMPLETED, and the string constant of Android.intent.action.BOOT_COMPLETED. You should find this message in the APK and then start it. So the essence of this way is to realize the BroadcastReceiver.

① Interface Activity, MainActivity .java file

```
public class MainActivity extendsActivity {
/** Called when the activity is first created. */
@Override
public void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
// without title
requestWindowFeature(Window. FEATURE_NO_TITLE );
//  full screen
getWindow().setFlags(WindowManager.LayoutParams. FLAG_FULLSCREEN ,
WindowManager.LayoutParams. FLAG_FULLSCREEN );
setContentView (R.layout. activity_main );
}
}
```

This code is very simple. When the Activity starts, it will display the TextView. So you can use it to display words as you want.

② To receive broadcast messages: BootBroadcastReceiver.java

```
public class BootBroadcastReceiver extends BroadcastReceiver {
static final String action_boot = "android.intent.action.BOOT_COMPLETED";
```

```
@Override
public void onReceive(Context context, Intent intent) {
if (intent.getAction().equals( action_boot )) {
Intent ootStartIntent = new Intent(context, MainActivity. class);
ootStartIntent.addFlags(Intent. FLAG_ACTIVITY_NEW_TASK );
context.startActivity(ootStartIntent);
}
}
}
```

This class inherits from BroadcastReceiver. In the overriding method onReceive, it detects whether the received Intent meets BOOT_COMPLETED, and if it does, then it will start the MainActivity Activity.

③ To configure AndroidManifest.xml file

```
<?xml version= "1.0" encoding= "utf-8" ?>
<manifest xmlns:android= "http://schemas.android.com/apk/res/android"
package= "com.ajie.bootstartdemo"
android:versionCode= "1"
android:versionName= "1.0" >
<uses-sdk
android:minSdkVersion= "8"
android:targetSdkVersion= "17" />
<application
android:allowBackup= "true"
android:icon= "@drawable/ic_launcher"
android:label= "@string/app_name"
android:theme= "@style/AppTheme" >
<activity
android:name= "com.dwin.bootstartdemo.MainActivity"
android:label= "@string/app_name" >
```

```
<intent-filter>
<action android:name= "android.intent.action.MAIN" />
<category android:name= "android.intent.category.LAUNCHER" />
</intent-filter>
</activity>
<receiver android:name= "com.dwin.bootstartdemo.BootBroadcastReceiver" >
<intent-filter>
<action android:name= "android.intent.action.BOOT_COMPLETED" />
<category android:name= "android.intent.category.HOME" />
</intent-filter>
</receiver>
</application>
<uses-permission android:name= "android.permission.RECEIVE_BOOT_COMPLETED" >
</uses-permission>
</manifest>
```

Note: In the part marked with red, the node has registered a receiver with the system, and the child node intent-filter means receiving Android.intent.action.BOOT_COMPLETED news and configuring Android.permission.RECEIVE_BOOT_COMPLETED permission.

After the configuration is completed, compile the APK and install it on the Android screen. Shut down and reboot, the page of the MainActivity activity will be displayed.

DWIN Android broadcast reception is operated by the default interface of Android system, using the standard Android API function library. For specific usage, please refer to the Android official website API Guides.

b) You can choose to either enter the android interface or the application after booting.

The essence of realizing automatic start after booting is to modify the application to have the launcher authority.

Replace the `<intent-filter>` of the first `activity` started in the configuration file. `AndroidManifest.xml`:

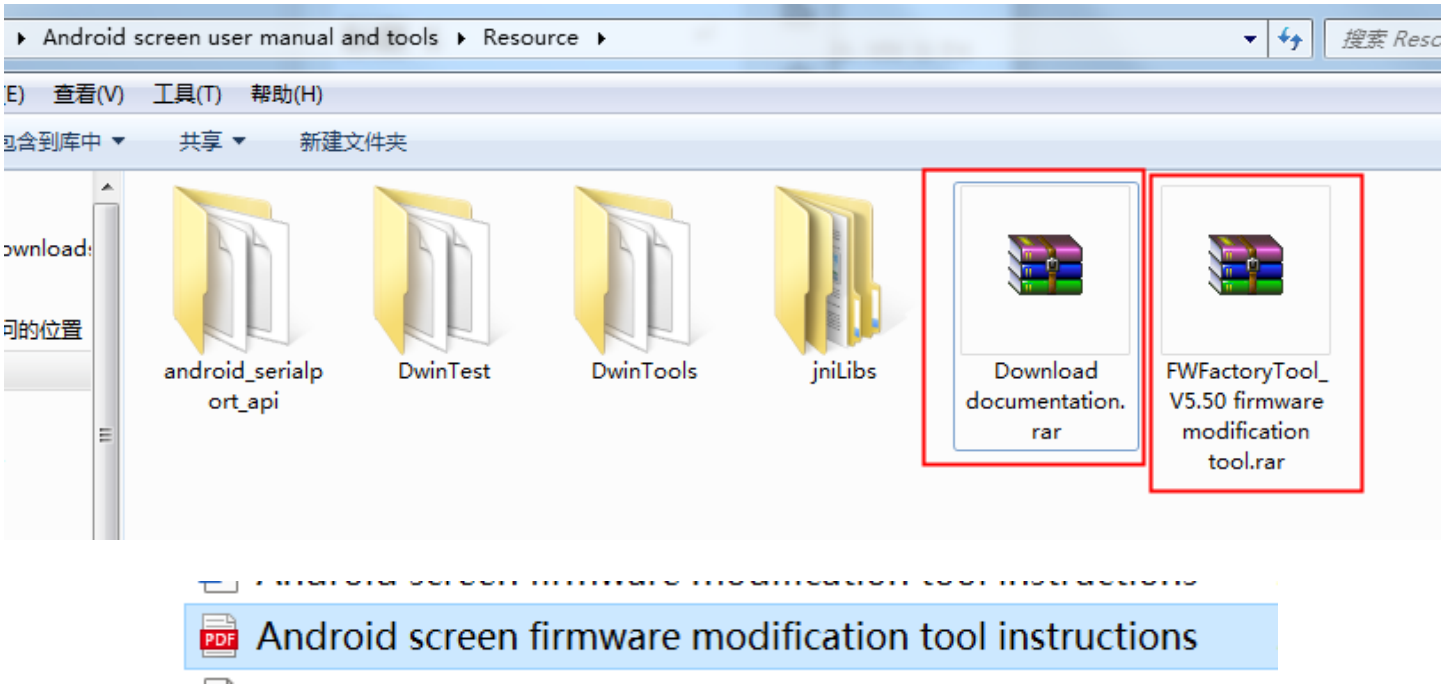
```
<?xml version= "1.0" encoding= "utf-8" ?>
<manifest xmlns:android= "http://schemas.android.com/apk/res/android"
package= "com.ajie.bootstartdemo"
android:versionCode= "1"
android:versionName= "1.0" >
<uses-sdk
android:minSdkVersion= "8"
android:targetSdkVersion= "17" />
<application
android:allowBackup= "true"
android:icon= "@drawable/ic_launcher"
android:label= "@string/app_name"
android:theme= "@style/AppTheme" >
<activity
android:name= "com.dwin.bootstartdemo.MainActivity"
android:label= "@string/app_name" >
<intent-filter>
<action android:name= "android.intent.action.MAIN" />
<category android:name= "android.intent.category.HOME" />
<category android:name= "android.intent.category.DEFAULT" />
<category android:name= "android.intent.category.MONKEY" />
</intent-filter>
</activity>
</application>
</manifest>
```

Click [Home] after running the program, and then the [Launcher] selection box will pop up. You can select the APK you have developed and click [Always].

4 Download and firmware modification

Download instructions and tools are provided in the Download documentation.

For firmware modification, refer to the Android screen firmware modification tool instructions document; and for tools, refer to the FWFactory.



Revision records

Rev	Revise Date	Content	Editor
00	2022-10-27	First Edition	Lvzhi Chen

Please contact us if you have any questions about the use of this document or our products, or if you would like to know the latest information about our products:

Customer service Tel: 400 018 9008

Customer service QQ: 400 018 9008

Customer service email: dwinhmi@dwin.com.cn

DWIN Developer Forum: http://forum.dwin.com.cn

Thank you all for continuous support of DWIN, and your approval is the driving force of our progress!