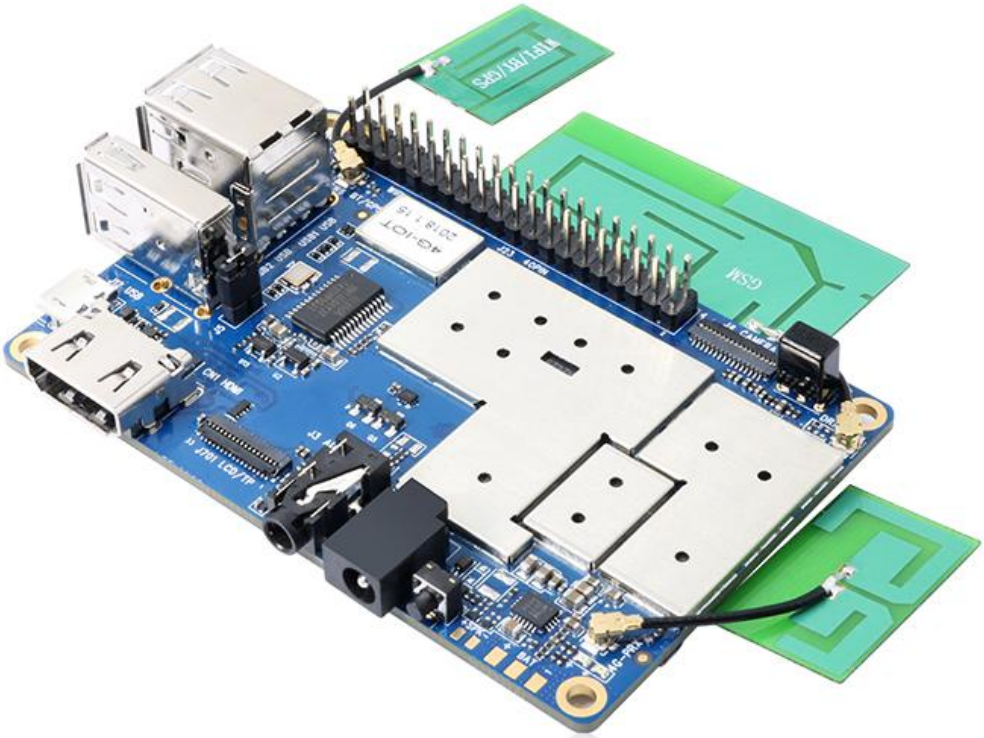
# Orange Pi 4G-IoT
# User Manual



# History

| Ver | Data | Author | Brief | Publish | Me |
|---|---|---|---|---|---|
| 1.1 | 2018-01-26 | Younix | Create File | 2018-3-27 | |
| 1.2 | 2018-05-10 | Engineer Pan | Add Android8.1 | 2018-5-15 | |
| 1.3 | 2018-10-18 | Engineer Pan | Add Usage of GPIO | 2018-10-19 | |
| 1.4 | 2019-06-24 | Csy | Linux source code compilation and serial port debugging | 2019-06-25 | |
| 1.5 | 2020-03-18 | csy | Linux Firmware Flashing | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |

# Contents

**I. Orange Pi Introduction....1**

1. What is Orange Pi 4G-IOT?....1
2. What can I do with Orange Pi 4G-IOT?....1
3. Who is it for?....1
4. Orange Pi 4G-IOT Hardware Specification....1
5. GPIO Specification....3

Interface instructions:....5

**II. Using Method....6**

1. Prepare the Hardware and Software....6
2. Power Methods....6
3. Before Usage....6

**III. Android Compilation Environment Construction....8**

1. Download SDK compression package....8
2. Construct Compilation Environment....8
3. Compilation of SDK Source Code....9
4. Generated Firmware....11

**IV. Android Firmware Flashing....13**

1. Flash Tool Introduction....14
2. Method for Image Flashing....15
3. FAQ....18

**V. Linux Compilation Environment Construction....21**

1. Get Linux Source Code....21
2. Compilation of Linux Source Code....23

**VI. Serial Debugging Tool....25.**

1. Windows-based Use....26
2. Based on the use of the Linux platform....30

**VII. Usage of GPIO....32**

1. /sys/class/gpio....32
2. Modify and Display the GPIO Status under ADB Mode....33

# I. Orange Pi Introduction

## 1. What is Orange Pi 4G-IOT?

It's an open-source single-board computer. It can run Android 6.0 Image. It uses the MTK serial MT6737 SoC, and has 1GB DDR3 SDRAM.

## 2. What can I do with Orange Pi 4G-IOT?

You can use it to build…

- A computer
- A wireless server
- Games
- Music and sounds
- HD video
- A speaker
- Android
- Scratch

Pretty much anything else, because Orange Pi 4G-IOT is open source.

## 3. Who is it for?

Orange Pi 4G-IOT is for anyone who wants to start creating with technology – not just consuming it. It's a simple, fun, useful tool that you can use to start taking control of the world around you.

## 4. Orange Pi 4G-IOT Hardware Specification

| Orange Pi 4G-IoT Specification | |
|---|---|
| Processor | MT6737 |

| | | |
|---|---|---|
| CPU | | Quad core ARM® Cortex-A53, Main frequency up to 1.25GHz |
| GPU | | ARM Mali-T720 MP1 |
| Memory | | 1GB DDR3 |
| Emmc | | 8GB EMMC Flash |
| Wireless | | WIFI / BT / FM / GPS Four in one |
| Radio frequency | GSM | 900/1800 (850/1900 optional) |
| | WCDMA | B1/B8 (B2/B4/B5 optional) |
| | TD-CDMA | / |
| | CDMA2000 | / |
| | FDD-LTE | B1/B3/B7/B20 (B2/B4/B17optional) |
| | TDD-LTE | B38/40/41B |
| Display | | HD |
| Capacitance touch | | Support |
| Camera | | 13M (25pin ZIF Connector) |
| Accelerometer Sensor | | Support |
| IR Control | | Support (Adapted iDroid remote controller) |
| Fingerprint Identification | | Support |
| SIM Card | | mini Single SIM Card |
| TF Card | | Support hot-plugging |
| Audio | Earphone | For audio input / output |
| | Mic | For audio input |
| USB | USB Host × 3 | Support OTG |
| | Micro USB × 1 | Only for writing image |
| LED | Power Indicator LED | Red |
| | Status Indicator LED | Green |
| Key | | Power |
| HDMI | | Support |

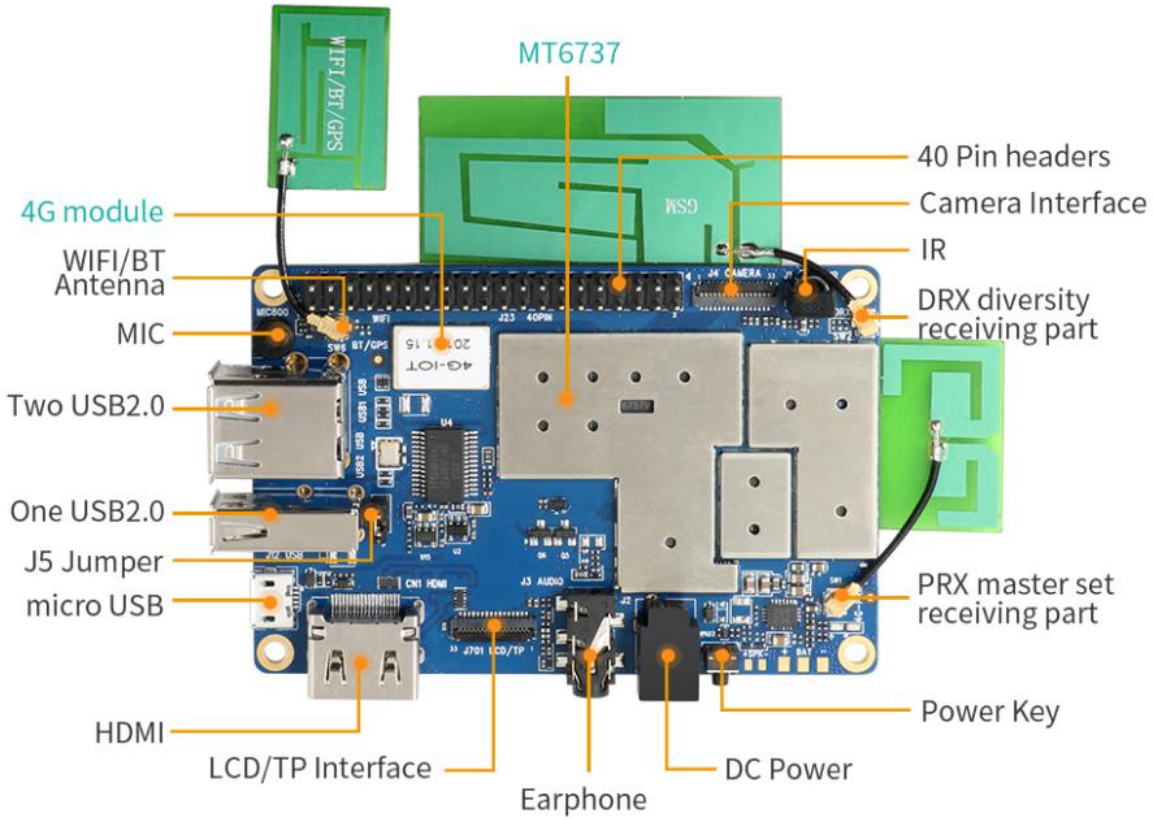
| | | |
|---|---|---|
| Low-level peripherals | 40pin Headers | 1.8V, SPI × 2 , I2C × 3, UART × 2 |
| Power | DC | 5V 2A |
| | Battery | Connection through a weld plate |
| OS/Software | | |
| OS | | Android 6.0 |
| Programming support | | C、C++、Kotlin、Java、Shell、Pyhon |
| Interface definition | | |
| Size | | 55mm*85mm |
| Weight | | 43g |
| Orange Pi™ is trademark of Shenzhen Xunlong Software CO., Limited | | |

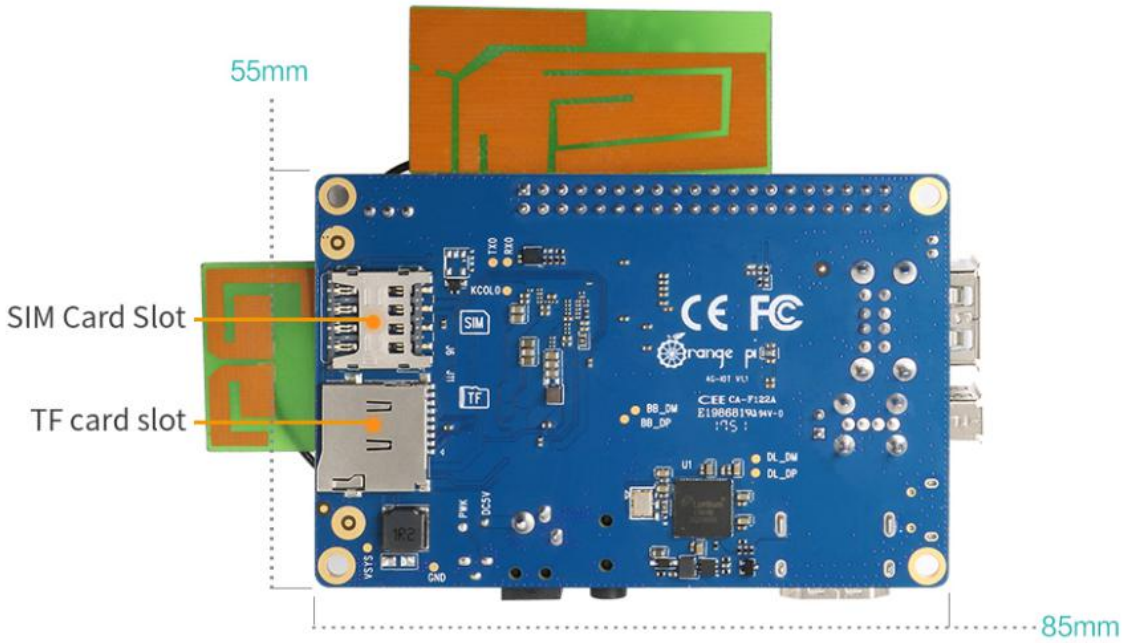
## 5. Orange Pi 4G-IOT 40pins GPIO Specification

| OrangePi(4G-IOT) | | |
|---|---|---|
| P01 | VIO28_PMU | |
| P02 | DC5V | |
| P03 | SDA1 | GPIO49 |
| P04 | DC5V | |
| P05 | SCL1 | GPIO50 |
| P06 | GND | |
| P07 | EINT87 | GPIO87 |
| P08 | UTXD2 | |
| P09 | GND | |
| P10 | URXD2 | GPIO57 |
| P11 | URXD1 | |
| P12 | EINT8 | EINT8 |

| | | |
|---|---|---|
| P13 | UTXD1 | |
| P14 | GND | |
| P15 | UCTS1 | |
| P16 | EINT9 | EINT9 |
| P17 | VIO28_PMU | |
| P18 | EINT11 | EINT11 |
| P19 | SPI2_MI | EINT3 |
| P20 | GND | |
| P21 | SPI2_MO | EINT4 |
| P22 | URTS1 | |
| P23 | SPI2_CK | EINT6 |
| P24 | SPI2_CS | EINT5 |
| P25 | GND | |
| P26 | EINT12 | |
| P27 | SDA2 | GPIO51 |
| P28 | SCL2 | GPIO52 |
| P29 | SPI_CS | GPIO65 |
| P30 | GND | |
| P31 | SPI_CK | GPIO66 |
| P32 | URTS2 | GPIO60 |
| P33 | SPI_MO | PA9 |
| P34 | GND | |
| P35 | SPI_MI | GPIO67 |
| P36 | UCTS2 | GPIO59 |
| P37 | EINT_123 | GPIO123 |
| P38 | SCL3 | GPIO54 |
| P39 | GND | |
| P40 | SDA3 | GPIO53 |

## Interface instructions:



# II. Using Method

## 1. Prepare the Hardware and Software

Hardware Requirement:

- Orange Pi 4G-IoT Development Board
- A PC for compilation with following specs:

  64bit CPU

  Up to 16GB RAM

  UP to 40GB spare disk space

  Operation system should up to Ubuntu12.04, it would be better if it is Ubuntu16.04

You could refer to Google file for more details: https://source.android.com/source/building

Software Requirement:

- Orange Pi 4G-IoT SDK
- Orange Pi 4G-IoT Firmware
- Android-image-flash-tool

## 2. Power Methods

There are two methods for power supply:

- DC （5V 2A）in for power:
- Battery in for power:

Usually use 3.7V battery to solder on the back side of the development board.

## 3. Before Usage

After receiving the product, please put the antennas of the product from the position of Pic 1 to the position of Pic 2 (or to the outside of the board), which can not be attached to the board so as not to affect the signal.


range pi
Regulatory Compli
Safety Informatio
range pi
A new generation of minipc
Getting Started

Pic 1


range pi
Regulatory Compli
Safety Informatio
range pi
A new generation of minipc
Getting Started

Pic 2

# III. Android Compilation Environment Construction

## 1. Download SDK compression package

- **Android 6.0**

Download the compression packages, corresponding to OrangePi_4G-IoT_Android 6.0_V1.0.tar.gz.00, OrangePi_4G-IoT_Android 6.0_V1.0.tar.gz.01, OrangePi_4G-IoT_Android 6.0_V1.0.tar.gz.14, a total of 15 volume compression packages. After obtaining the volume compressed packages, place all the compressed packages in the same directory, such as

```
Create directory:
mkdir OrangePi_4G-IOT_Android6.0
Copy Volume Compression Packet:
cp -rf OrangePi_4G-IoT_Android6.0_V1.0.tar.gz.*
OrangePi_4G-IOT_Android6.0/
Merge Compression Packet:
cat OrangePi_4G-IoT_Android6.0_V1.0.tar.gz.* >
OrangePi_4G-IOT_Android6.0.tar.gz
Unzip:
tar xzvf OrangePi_4G-IOT_Android6.0.tar.gz
```

- **Android 8.1**

After downloaded compression package, you will have 11packages named x00, x01, x02, x03 ...x10.Put packages on the same directory like the following:

```
Create directory
mkdir OrangePi_4G-IOT_Android8.1
Copy compress package
cp -rf x00 x01 x02 ... x10
Merge compression package
cat x*>OrangePi_4G-IOT_Android8.1.tar.gz
Decompression
tar xzvf OrangePi_4G-IOT_Android8.tar.gz
```

## 2. Construct Compilation Environment

It could also refer to Google file: http://source.android.com/source/initializing.html

- **Install JDK**

Compilation of Android6.0 is based on JAVA7, it needs to first install OpenJDK before compilation.

Command for installing:

```
sudo apt-get install openjdk-7-jdk
```

Configure environment variable of JAVA, here is the path for installation:

/usr/lib/jvm/java-7-openjdk-amd64

It could configure on the terminal with the following command:

```
export JAVA_HOME=/usr/lib/jvm/java-7-openjdk-amd64
export PATH=$JAVA_HOME/bin:$PATH
export CLASSPATH=.:$JAVA_HOME/lib:$JAVA_HOME/lib/tools.jar
```

- **Install Software Package**

For Ubuntu12.04:

```
sudo apt-get update
sudo apt-get install git-core gnupg flex bison ccache gperf libsdl1.2-dev
libesd0-dev libwxgtk2.6-dev build-essential zip curl libncurses5-dev
zlib1g-dev valgrind libc6-dev lib32ncurses5-dev x11proto-core-dev
libx11-dev lib32readline-gplv2-dev lib32z1-dev libgl1-mesa-dev gcc-4.4
g++-4.4 g++-4.4-multilib g++-multilib mingw32 tofrodos python-markdown
libxml2-utils xsltproc wine
```

For Ubuntu14.04：

```
sudo apt-get update
sudo apt-get install git-core gnupg flex bison ccache gperf libsdl1.2-dev
libesd0-dev libwxgtk2.8-dev build-essential zip curl libncurses5-dev
zlib1g-dev valgrind libc6-dev lib32ncurses5-dev x11proto-core-dev
libx11-dev lib32readline-gplv2-dev lib32z1-dev libgl1-mesa-dev
g++-multilib g++-4.8-multilib mingw32 tofrodos python-markdown
libxml2-utils xsltproc libc6-dev-i386 lib32z1 lib32ncurses5 lib32bz2-1.0
lib32readline-gplv2-dev wine
```

We could process to SDK compilation after finished the above.

# 3. Compilation of SDK Source Code

There are many compilation shell scripts for development.

- **Android 6.0**

The directory would be: SDK/code/orangepi/scripts

```
$ cd code/orangepi/scripts
$ ls
anr_LM.sh  auto.sh  clean.sh  codegen.sh  init_project.sh  tar_img.sh
```

auto.sh is automatically compilation script

clean.sh automatically scavenging the compiled result script

On the directory of code/orangepi/scripts, execute the automatically compilation script:

```
$ ./auto.sh IoT_bd6737m_35g_b_m0_op_smt_hd720_pcb_v2 v00 eng
```

The meaning of the parameter is:

#$1 project_info [eg: IoT_bd6737m_35g_b_m0_op_smt_hd720_pcb_v2]

#$2 version_info [eg: v00 v01 ...]

#$3 compile_mode [eng:user userdebug eng]

- **Android 8.1**

The directory would be: SDK/code/orangepi/scripts

```
$ cd code/orangepi/scripts
$ ls
anr_LM.sh  auto.sh  clean.sh  codegen.sh  init_project.sh  tar_img.sh
```

auto.sh--scripts for automatic compilation

clean.sh--scripts for automatically cleaning the compiled result

On the directory of code/orangepi/scripts, execute automatic compilation scripts:

```
$ ./auto.sh IoT_k37mv1_bsp_ry_smt_hd720_pcb_v2 v00 eng
```

Definition of the three parameters:

#$1 project_info [eg: IoT_k37mv1_bsp_ry_smt_hd720_pcb_v2]

#$2 version_info [eg: v00 v01 ...]

#$3 compile_mode [eng:user userdebug eng]

Execute command to compile:

source build/envsetup.sh

luncher ------>full_k37mv1_bsp-eng

make -j4

## Module compilation

Here would take an example of only compilation launcer:

mm packages/apps/Launcher3/ or enter into directory of packages/apps/Launcher3/, execute mm

Please note that some modules depend on the relationship of package, you need to run mma.

# 4. Generated Firmware

- **Android6.0**

After compiled, the firmware will gather in the directory of: code/IoT_op_smt_hd720_pcb_v2, pack it and name it like the following: IoT_op_smt_hd720_pcb_v2_v00_eng_20180126140300.tar.gz

```
$ tree IoT_op_smt_hd720_pcb_v2
IoT_op_smt_hd720_pcb_v2
├──── images
│       ├──── boot.img
│       ├──── cache.img
│       ├──── lk.bin
│       ├──── logo.bin
│       ├──── MT6737M_Android_scatter.txt
│       ├──── preloader_bd6737m_35g_b_m0.bin
│       ├──── recovery.img
│       ├──── secro.img
│       ├──── system.img
│       ├──── trustzone.bin
│       └──── userdata.img
└──── modem
     ├──── APDB_MT6735_S01_alps-mp-m0.mp1_W17.21
     ├──── _APDB_MT6735_S01_alps-mp-m0.mp1_W17.21.check
     └──── APDB_MT6735_S01_alps-mp-m0.mp1_W17.21_ENUM
```

Except the above method, it could also be generated into update.image via Linux_Pack_Firmware.

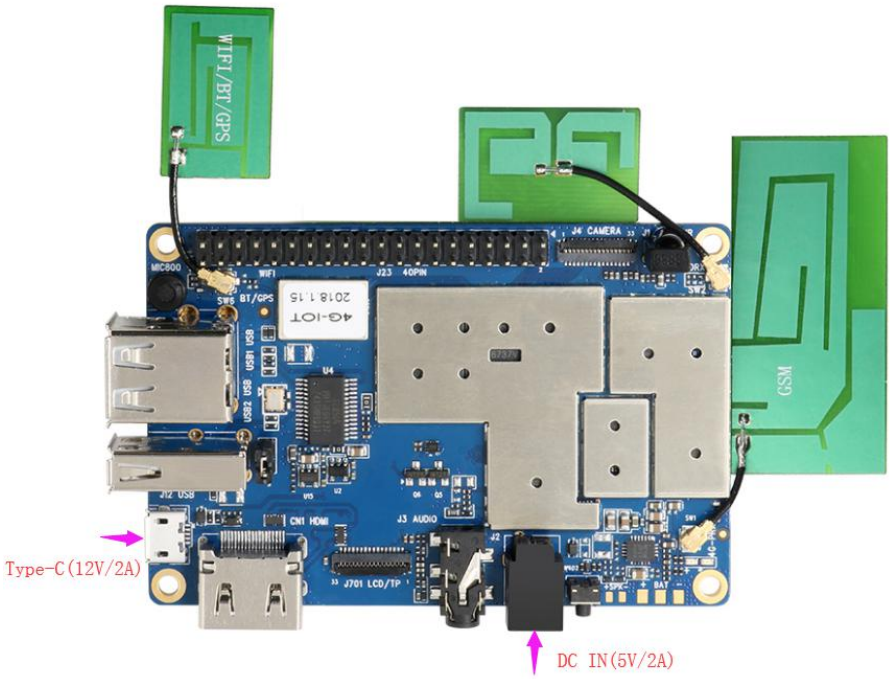
- **Android8.1**

After compiled, the firmware will gather in the directory of: code/IoT_ry_smt_hd720_pcb_v2, pack it and name it like the following: IoT_ry_smt_hd720_pcb_v2_v09_eng_20180504163653.tar.gz

```
$ tree IoT_op_smt_hd720_pcb_v2
IoT_op_smt_hd720_pcb_v2
├──── images
│       ├──── boot.img
│       ├──── cache.img
│       ├──── lk.bin
│       ├──── logo.bin
│       ├──── MT6737M_Android_scatter.txt
│       ├──── preloader_k37mv1_bsp.bin
│       ├──── recovery.img
│       ├──── secro.img
│       ├──── system.img
│       ├──── trustzone.bin
│       └──── userdata.img
└──── modem
     ├──── APDB_MT6735_S01_alps-mp-m0.mp1_W18.04
     ├──── _APDB_MT6735_S01_alps-mp-m0.mp1_W18.04.check
     └──── APDB_MT6735_S01_alps-mp-m0.mp1_W18.04_ENUM
```

# IV.Android Firmware Flashing

Relevant keys and connectors for firmware flashing of **Orange Pi 4G-IoT:**




**List of generated firmwares:**

IoT_op_smt_hd720_pcb_v2

```
├─── images
|    ├─── boot.img
|    ├─── cache.img
|    ├─── lk.bin
|    ├─── logo.bin
|    ├─── MT6737M_Android_scatter.txt
|    ├─── preloader_bd6737m_35g_b_m0.bin
|    ├─── recovery.img
|    ├─── secro.img
|    ├─── system.img
|    ├─── trustzone.bin
|    └─── userdata.img
└─── modem
     ├─── APDB_MT6735_S01_alps-mp-m0.mp1_W17.21
```

```
├── _APDB_MT6735_S01_alps-mp-m0.mp1_W17.21.check
└── APDB_MT6735_S01_alps-mp-m0.mp1_W17.21_ENUM
```

You could download the packed image partition files from the official website: http://www.orangepi.org/downloadresources/

Unzip the file with the following command:

```
$ tar zxvf IoT_op_smt_hd720_pcb_v2_v00_eng_20180126140300.tar.gz
```

You could get the file which mentioned on the list of generated files, or you could also compile it by yourself with reference of to the part of Android Compilation Environment Construction.

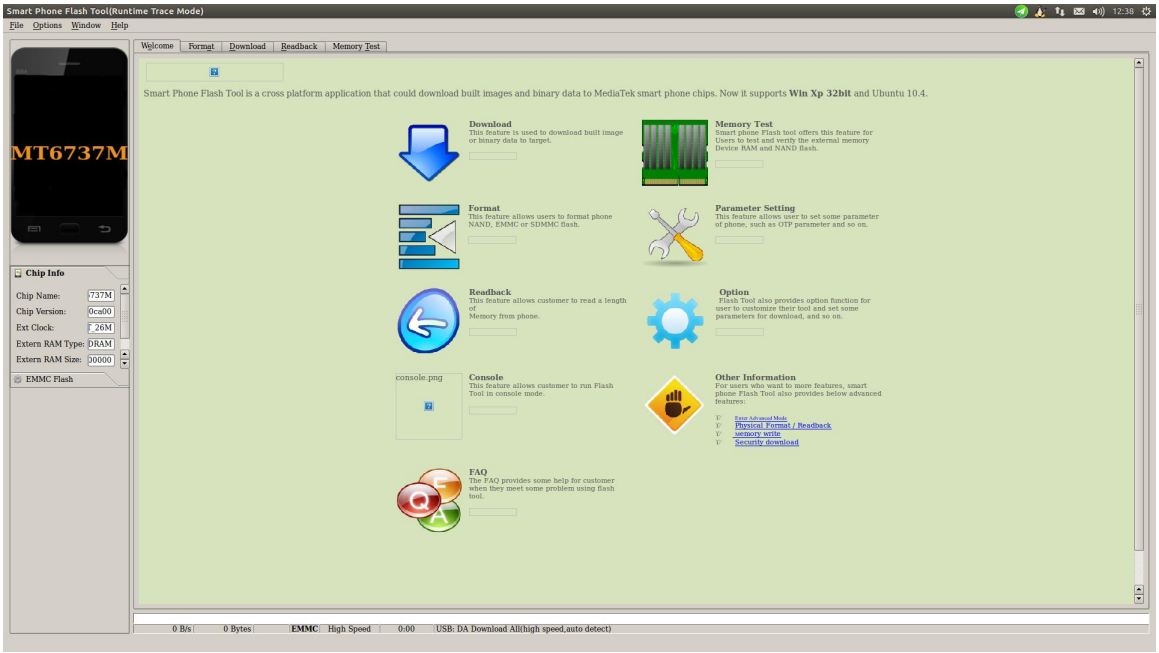
**Supporting OS of PC:**

- Windows 10
- Windows 7 (32/64 bit)
- Windows 8 (32/64 bit)
- Ubuntu10.04 / 12.04 / 14.04 (32/64bit)

# 1. Flash Tool Introduction

You could download the **Smart Phone Flash Tool** on the download page of Orange Pi 4G-IoT part. There are tools for Windows and Linux version, you could select a suitable version according to your PC environment.

Interface like the following:



Using method for both Windows and Linux versions are same, here will illustrate with Linux version.

## 2. Method for Image Flashing

```
If you cannot connect the Orange Pi 4G-IOT to the computer, pls try:
$sudo apt-get remove modemmanager
$sudo /etc/init.d/udev retstart
And then reboot the computer

Download MTK driver according to the corresponding system under Windows
Unzip Driver_Auto_Installer_EXE_v5.1453.03.rar
And then install it。
```

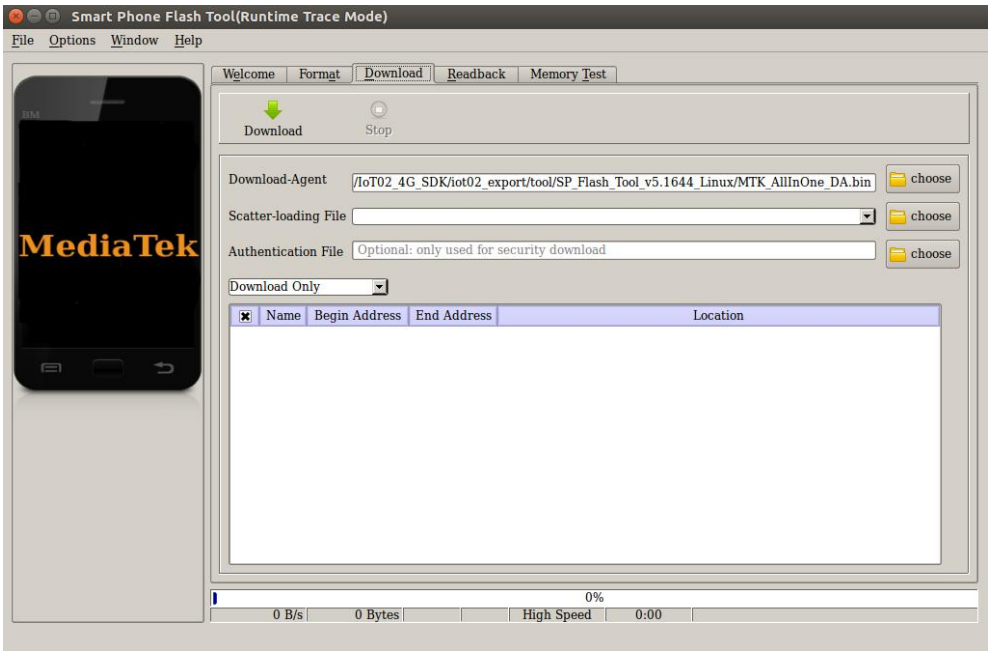
- **Unzip and open flash tool**

```
$ unzip SP_Flash_Tool_v5.1644_Linux.zip
$ cd SP_Flash_Tool_v5.1644_Linux
$ sudo ./flash_tool.sh
```

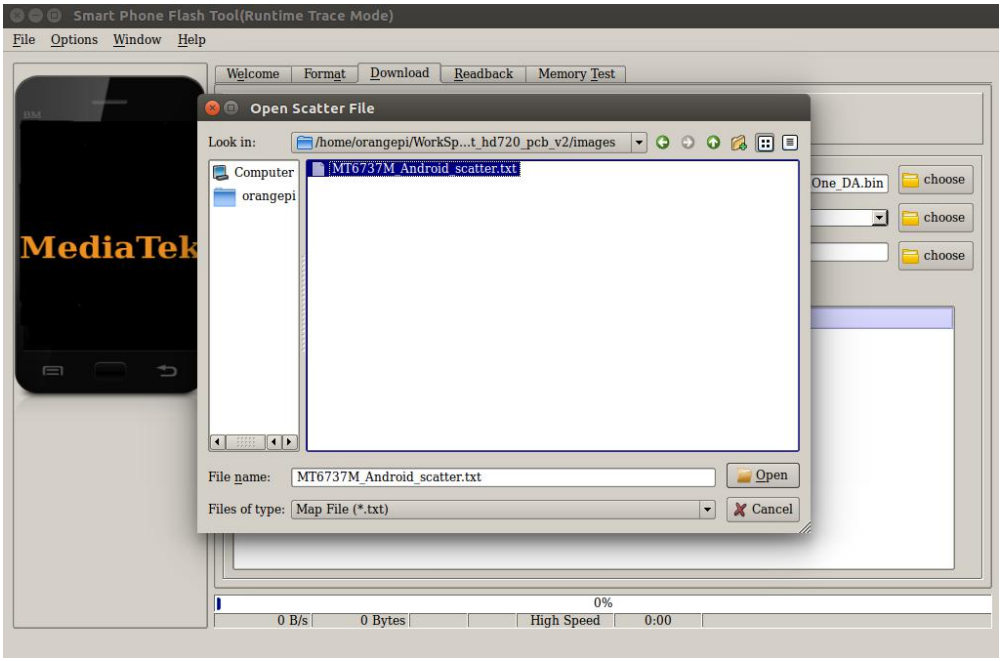
If it is the first time you use this software, you might receive the warn like the following. It is normal to receive this, you could click OK enter into the software. In the future you could manually specified the path of Scatter File.


- **Enter into flash mode**

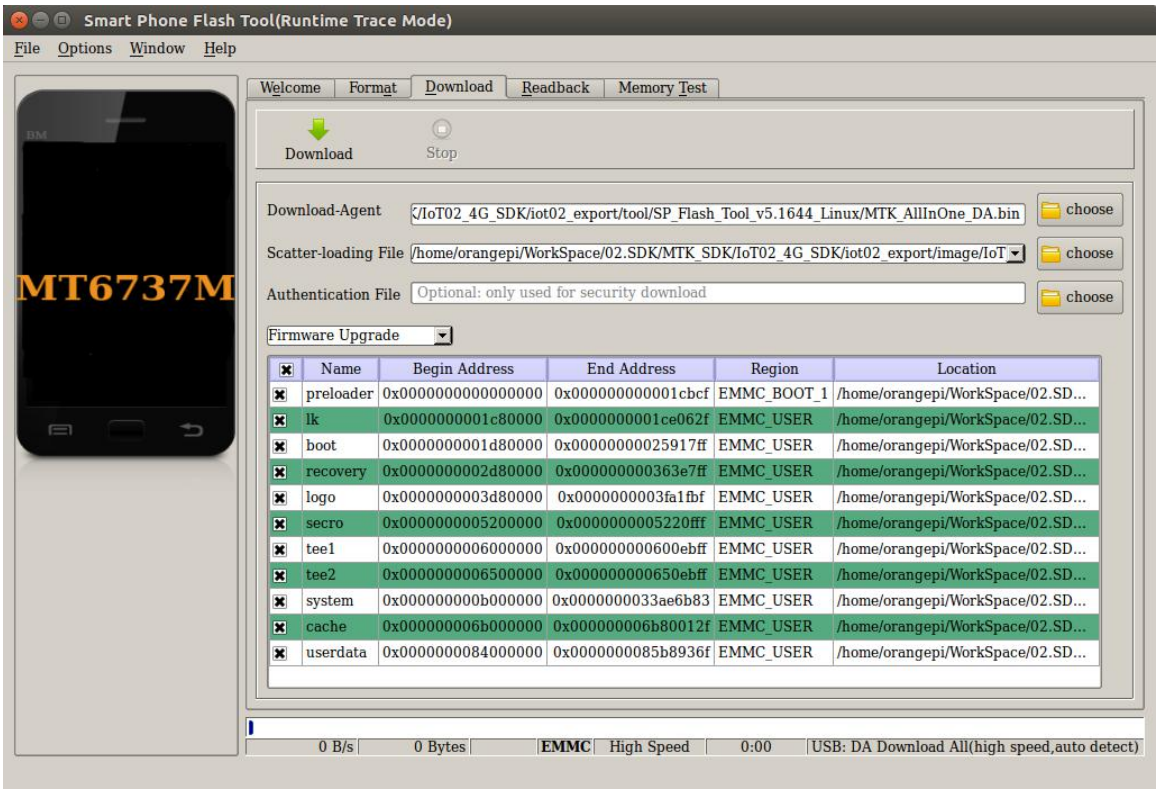
a. Switch into Download page like the following:




b. Click choose on the right side of Scatter-loading File and select the path of Scatter File like the following:



c. After double click the selection, the **partition information display section** will automatically fill the path of each partition file and the absolute starting address to which they are to be flashed.




d. In the top left corner of **partition information display section**, there would be a drop-down menu. Three of this options:

Format All + Download // Format all information on the partitions and re-download the selected partition

Firmware Upgrade // Update the difference on the selected partition

Download Only // Re-download no matter there is difference or not

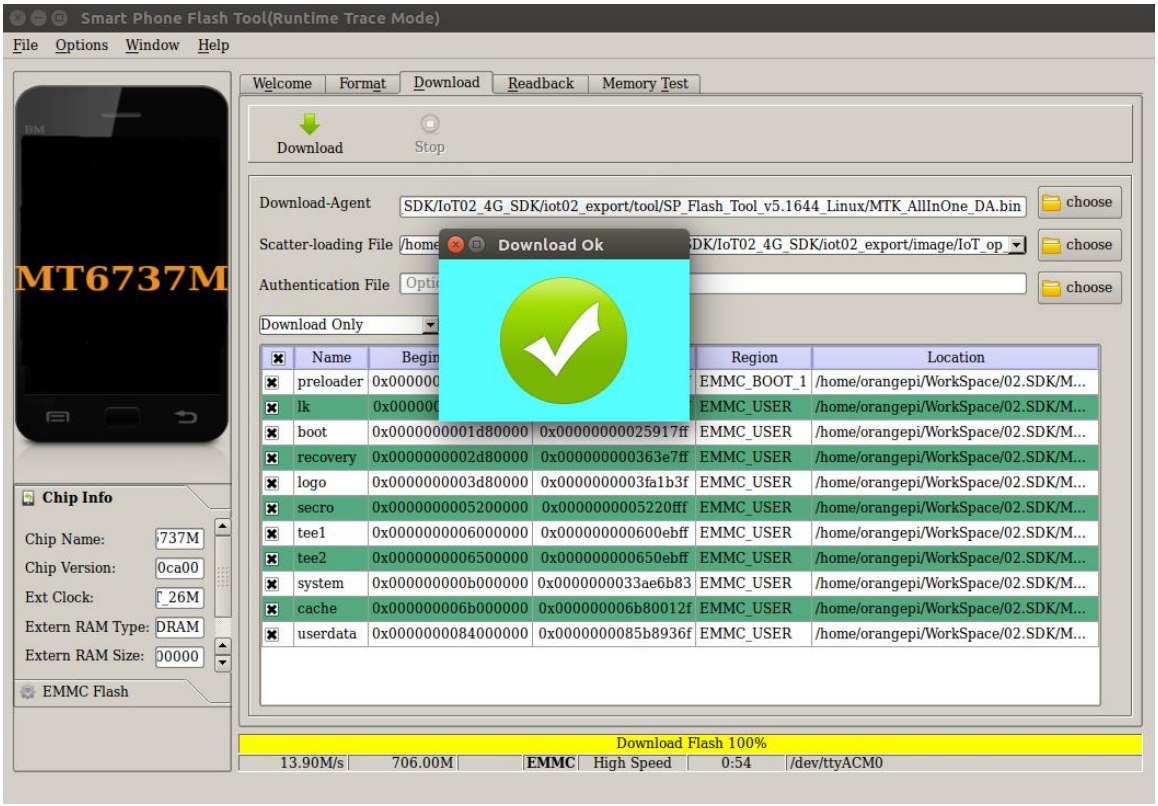
Please note it: Usually update firmware you only need to select **Firmware Upgrade**, please **do not** select **Format All + Download**

It you select Format All, you will lose the calibration information which we worked before sending out products. If this situation is inadvertent, please contact the Orange Pi service and obtain the calibration parameters through the machine code, and re-flash the calibration parameters.

e. Use USB data cable to connect PC and Orange Pi, the right side of Orange Pi will be in red LED. In this case do not need to connect DC power supply.

f. Click Download button 


g. The interface would show like the following after downloaded:




h. Take of the USB cable and insert DC power supply

Wait around 5 seconds, it will display the charging interface of shutdown

When the Power button is loosened after 5 seconds, the system will start to enter the system

When the updated partition is more, the first boot will take a long time (the full partition update needs 8min), please be patient.

# 3. FAQ

Android8.1 compilation tool chain uses the new Jack server to replace the old compilation tool chain.

- **Configure Jack server before compilation**

①Modify .jack file on the path of $HOME:

```
# Server settings
SERVER=false
SERVER_PORT_SERVICE=8072
SERVER_PORT_ADMIN=8073
#SERVER_PORT_SERVICE=8096
#SERVER_PORT_ADMIN=8097
SERVER_COUNT=1
SERVER_NB_COMPILE=4
SERVER_TIMEOUT=60
SERVER_LOG=${SERVER_LOG:=$SERVER_DIR/jack-$SERVER_PORT_SERVICE.log}
JACK_VM_COMMAND=${JACK_VM_COMMAND:=java}
# Internal, do not touch
SETTING_VERSION=2
~
~
```

②Modify .jack-settings file on the path of $HOME:

```
# Server settings
SERVER_HOST=127.0.0.1
SERVER_PORT_SERVICE=8096
SERVER_PORT_ADMIN=8097

# Internal, do not touch
SETTING_VERSION=4
~
~
~
~
~
```



(These two files TCP port should not be used before, and these two files and ports should be with same configure.)

③code/prebuilts/sdk/tools/ directory, execute /jack-admin kill-server and./jack-admin restart-server

- **Failed to contact Jack server**

If you meet the following error when compilation:

FAILED: /bin/bash -c "(prebuilts/sdk/tools/jack-admin install-server prebuilts/sdk/tools/jack-launcher.jar prebuilts/sdk/tools/jack-server-4.8.ALPHA.jar 2>&1 || (exit 0) ) && (JACK_SERVER_VM_ARGUMENTS=\"-Dfile.encoding=UTF-8 -XX:+TieredCompilation\" prebuilts/sdk/tools/jack-admin start-server 2>&1 || exit 0 ) && (prebuilts/sdk/tools/jack-admin update server prebuilts/sdk/tools/jack-server-4.8.ALPHA.jar 4.8.ALPHA 2>&1 || exit 0 ) && (prebuilts/sdk/tools/jack-admin update jack prebuilts/sdk/tools/jacks/jack-2.28.RELEASE.jar 2.28.RELEASE || exit 47; prebuilts/sdk/tools/jack-admin update jack

prebuilts/sdk/tools/jacks/jack-3.36.CANDIDATE.jar 3.36.CANDIDATE || exit 47; prebuilts/sdk/tools/jack-admin update jack prebuilts/sdk/tools/jacks/jack-4.7.BETA.jar 4.7.BETA || exit 47 )"
Writing client settings in /home/user3/.jack-settings
Installing jack server in "/home/user3/.jack-server"

Modify: On directory of code/prebuilts/sdk/tools/, execute ./jack-admin kill-server and ./jack-admin restart-server, then re-compilation

- **Out of memory error**

First stop running jack server,
Then execute jack-admin on the directory of kill-server prebuilts/sdk/tools to kill Jack server
Then open jack-admin file, search JACK_SERVER_COMMAND on the file, then change JACK_SERVER_COMMAND="java -XX:MaxJavaStackTraceDepth=-1 -Djava.io.tmpdir=$TMPDIR $JACK_SERVER_VM_ARGUMENTS -cp $LAUNCHER_JAR $LAUNCHER_NAME" into JACK_SERVER_COMMAND="java -Xmx3g -XX:MaxJavaStackTraceDepth=-1 -Djava.io.tmpdir=$TMPDIR $JACK_SERVER_VM_ARGUMENTS -cp $LAUNCHER_JAR $LAUNCHER_NAME". Which is add option of -Xmx3g.

- **FAILED: setup-jack-server**

FAILED: setup-jack-server
Jack server installation not found

Solution: Execute the following command on the directory of prebuilts/sdk/tools: ./jack-admin install-server jack-launcher.jar jack-server-4.11.ALPHA.jar
jack-launcher.jar, jack-server-4.11.ALPHA.jar is up to the file name on the directory of prebuilts/sdk/tools.

- **Test Issue**

When the sim card cannot be recognized, you need to format SD card when it could not recognize it, and need to formatting flash.

# V. Linux Compilation Environment Construction

## 1. Get Linux Source Code

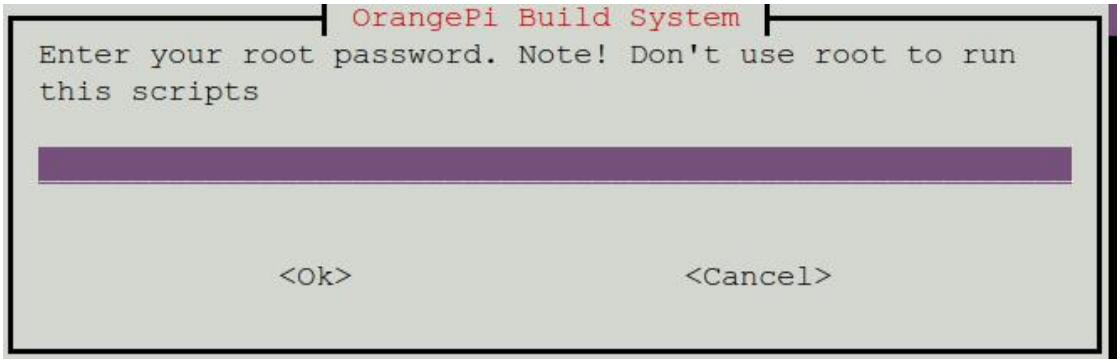
- **Orange Pi Source Code Downloader**

```
$ sudo apt-get install git
$ git clone https://github.com/orangepi-xunlong/OrangePi_Build.git
$ cd OrangePi_Build
$ ls
Build_OrangePi.sh lib README.md
```

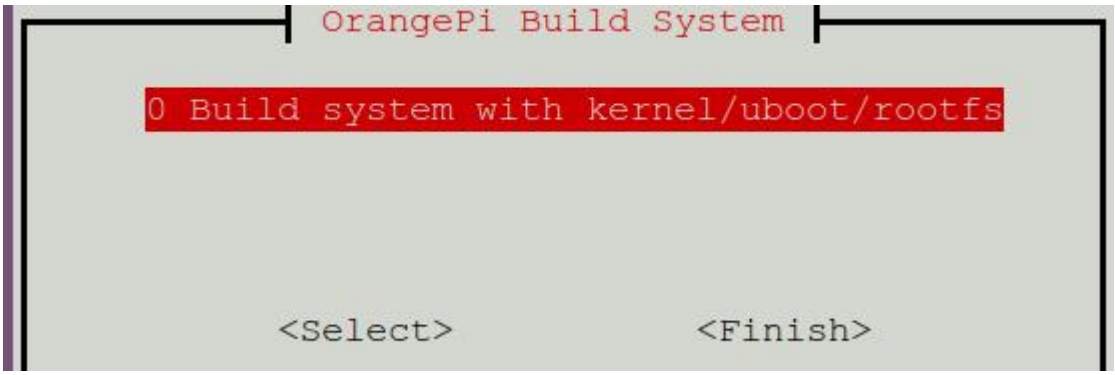
- **Run the Downloader**

```
$ ./Build_OrangePi.sh
```

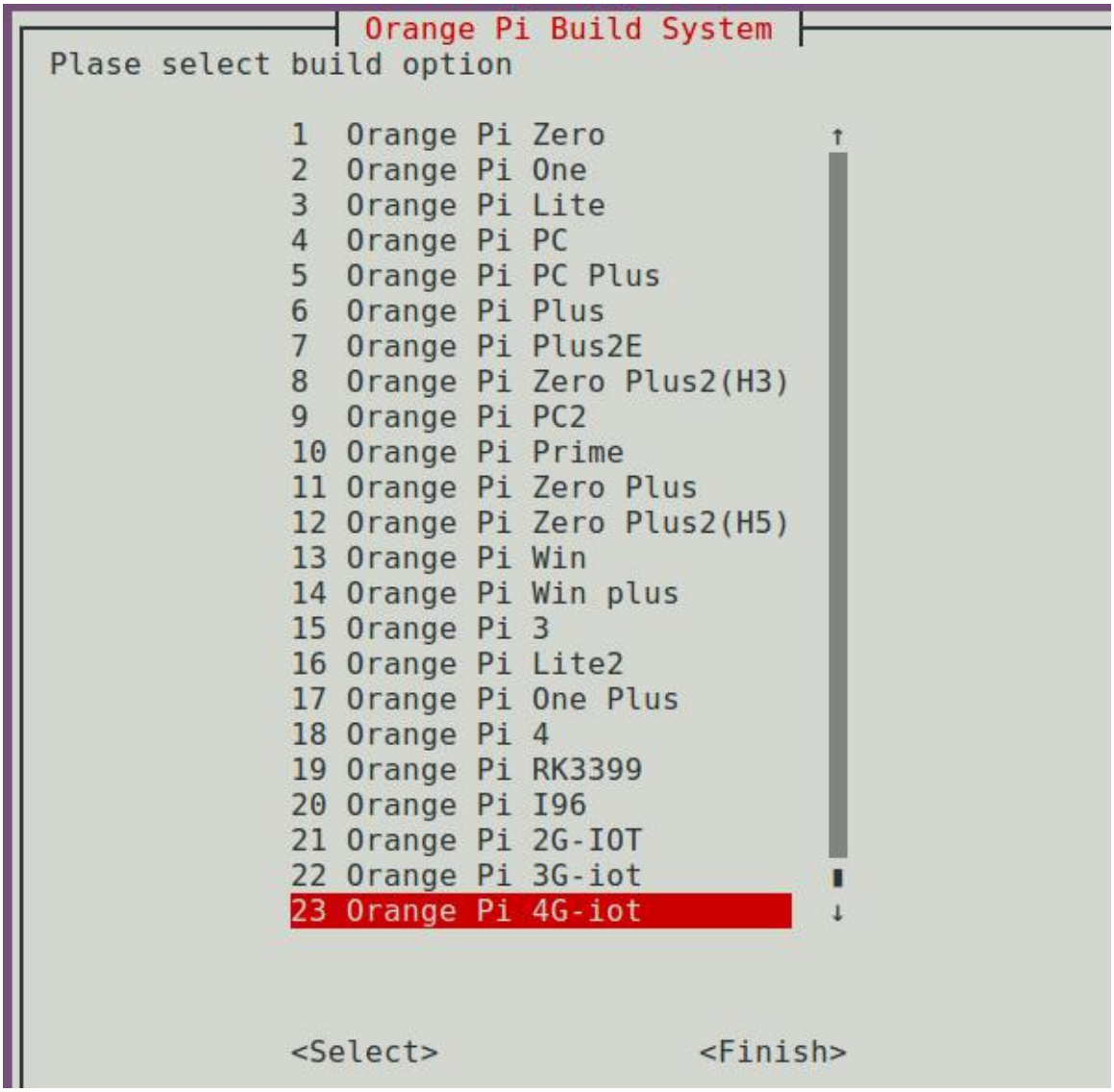
Enter the root password and press Enter




Choose 0 Build system with kernel/uboot/rootfs, enter the development board model selection interface.




Choose 23 orangepi 4G-iot，Press Enter and start download Orange Pi 4G-iot Linux Source Code.



The downloaded source code will be stored in the same directory as OrangePi_Build.

Orange Pi 4G-IOT Linux source code directory structure is as follows

```
.
├──── bootloader
├──   build.sh -> scripts/build.sh                                  Compile the
startup script
├──── external                                                      Store
additional configuration files and partial code
├──── kernel
├──── output      Store the output file, generate it after compiling the
source code
├──── scripts
└──── toolchain

6 directories, 1 file
```

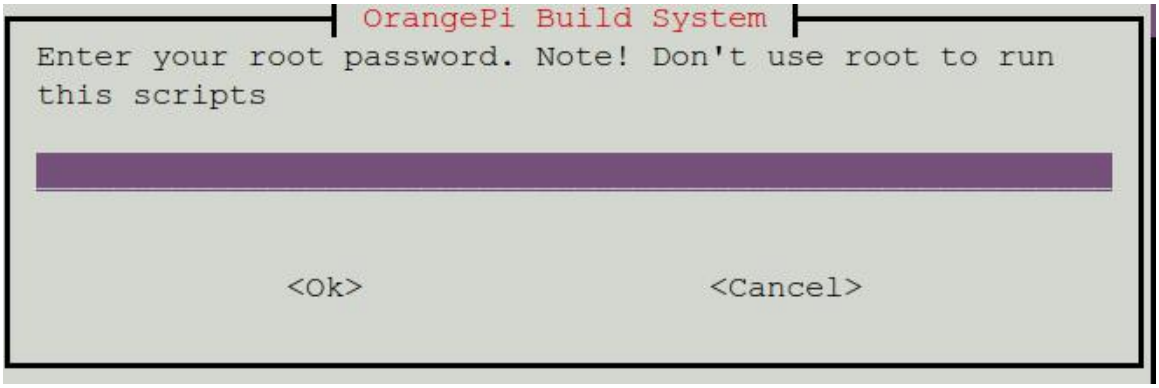
## 2. Compilation of Linux Source Code

- **Execute the compilation start up script**

```
$ cd OrangePi4G-iot
$ ./build.sh
```

According to the board model to choose and press Enter.

Enter the root password and press Enter, then select the function needed to execute.




The functions of each option are as follows:

- **0 Build Release Image** **Compile full image**
- **1 Build Rootfs** **Compile Rootfs**
- **2 Build Uboot** **Compile preloader and lk**
- **2 Build Linux** **Compile kernel source code**

Choose 0 Build Release Image, generate the following complete firmware package

```
output/images/
├── OrangePi_4g-iot_ubuntu_xenial_server_linux3.18.19_v1.1
│   ├── boot.img
│   ├── lk.bin
│   ├── lk_emmc.bin
│   ├── lk_sd.bin
│   ├── logo.bin
│   ├── MT6737M_Android_scatter.txt
│   ├── preloader_bd6737m_35g_b_m0.bin
│   ├── rootfs.img
│   └── trustzone.bin
└── OrangePi_4g-iot_ubuntu_xenial_server_linux3.18.19_v1.1.tar.gz

1 directory, 10 files
```

Android burning tool can be used to burn the above image file into flash, please refer to Android firmware burning chapter.

Also can execute 3 Install Image into EMMC option to open the burning tool for burning

# Linux Firmware Flashing

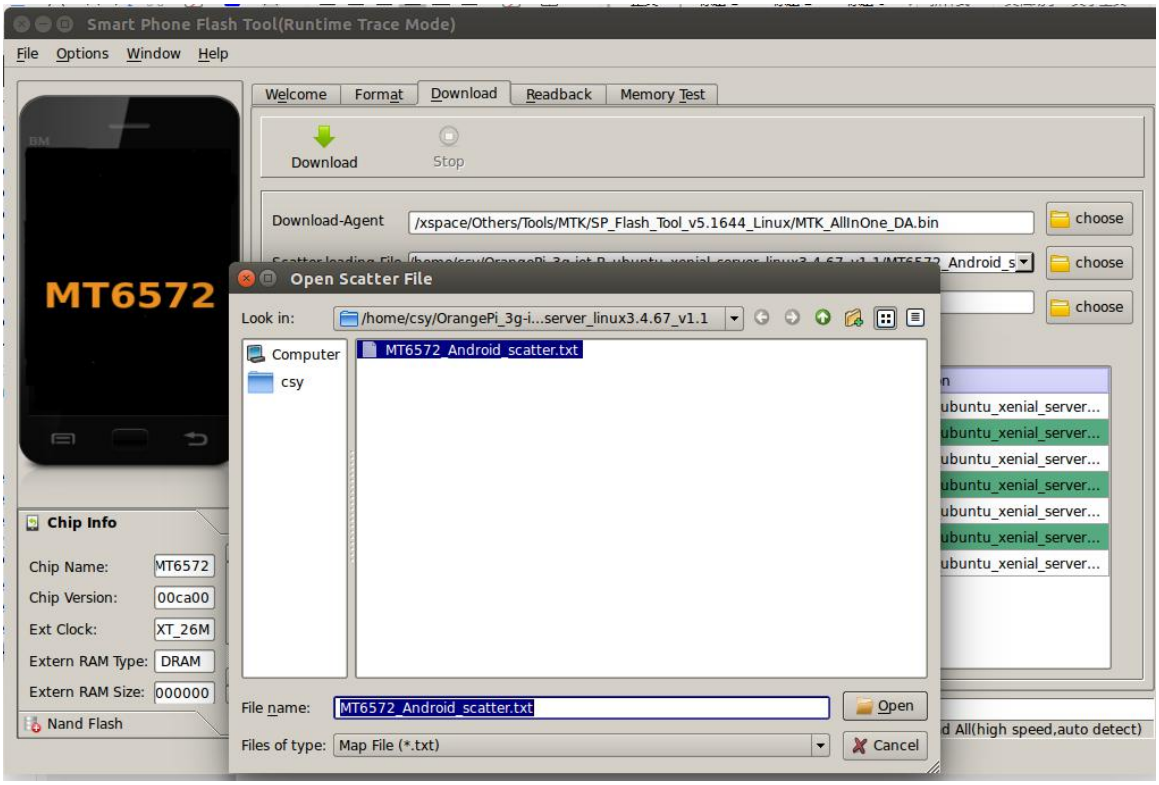
Currently, Linux systems do not support modems and LCD screens.

You can only log in to the system through the serial port. After connecting to wifi, you can log in through ssh.

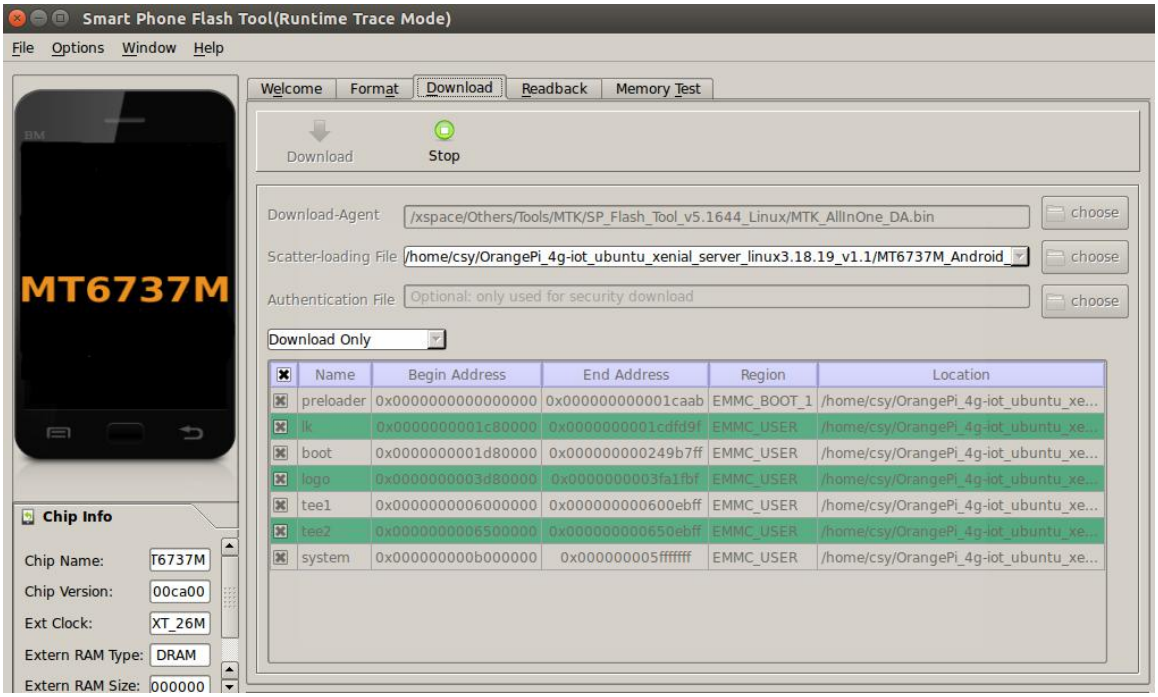
The previous "Android Firmware Burning" has introduced how to use the burning tool, so I won't go into details here, just open the burning tool directly.

We click on the far right of the Scatter-loading File column

And select the path of Scatter File, as shown below

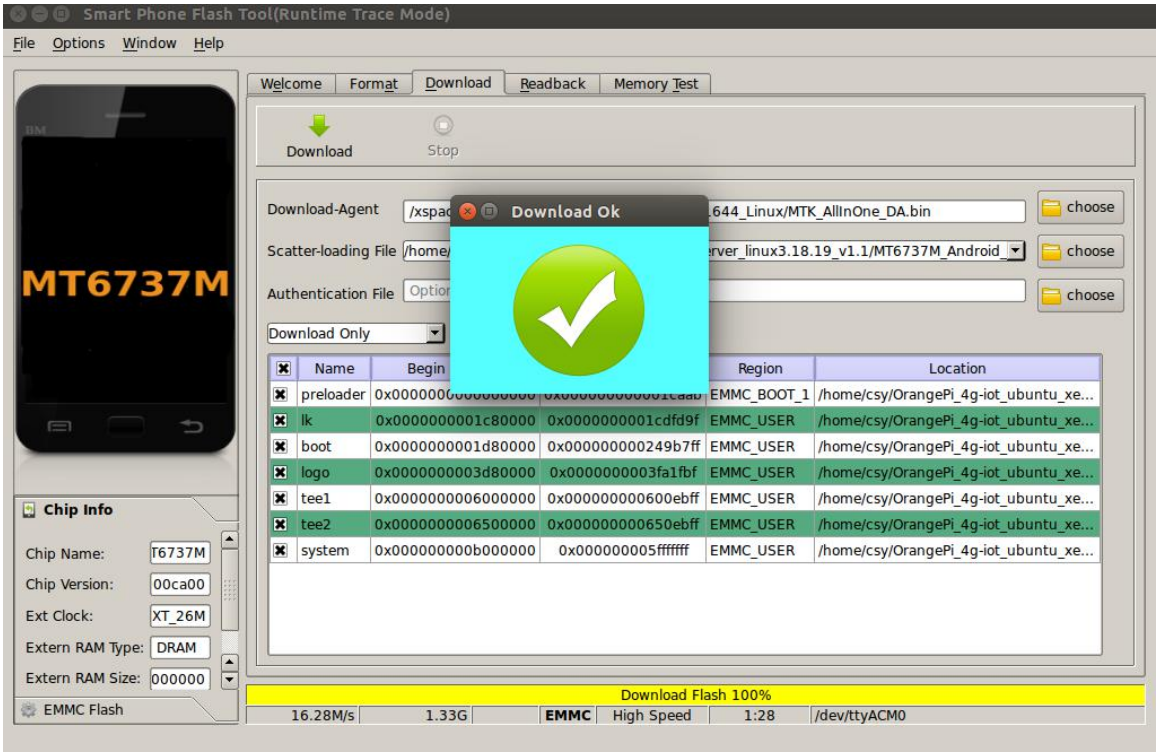


First disconnect the board from the PC and click the Download button




Next, use a USB download cable to connect the host's USB to the machine's MircoUSB interface.

Burning completed



Start the system.

Connect the board to a 5V 3A power adapter. Connect the serial port to see the system startup.

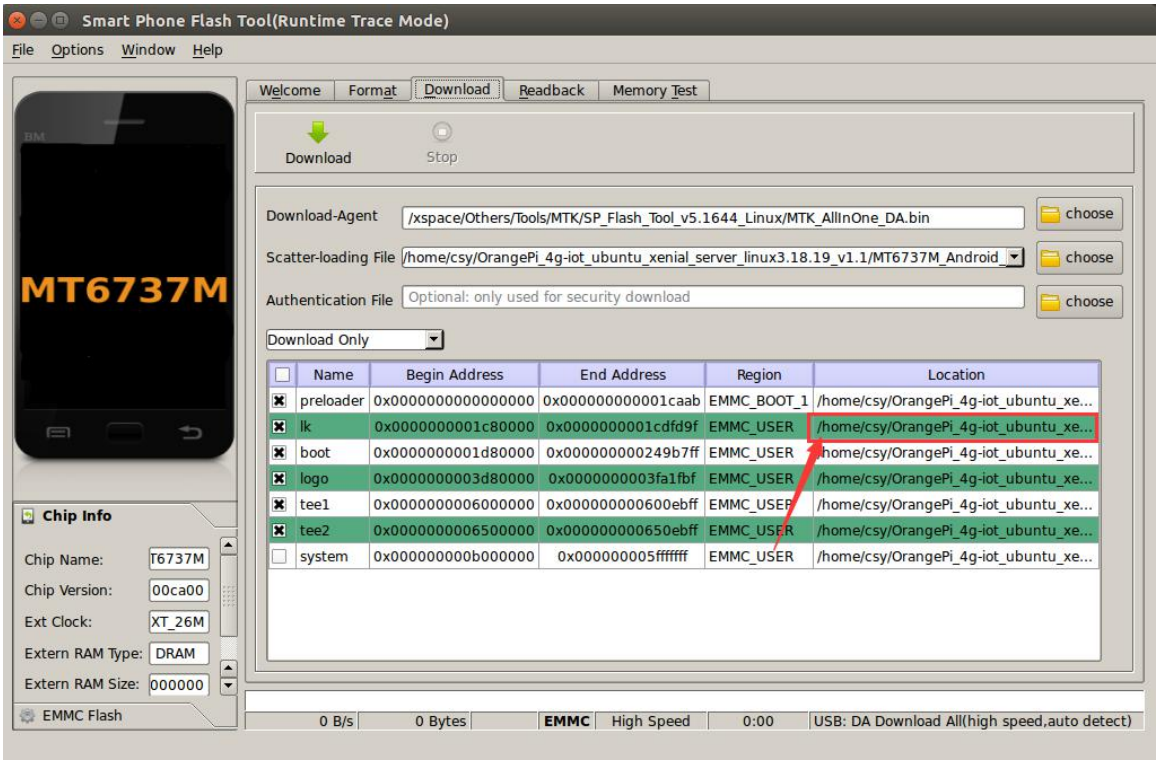
```
Ubuntu 16.04.1 LTS orangepi4g-iot ttyMT0

orangepi4g-iot login: root
Password:
Welcome to Ubuntu 16.04.1 LTS (GNU/Linux 3.18.19+ armv7l)
```

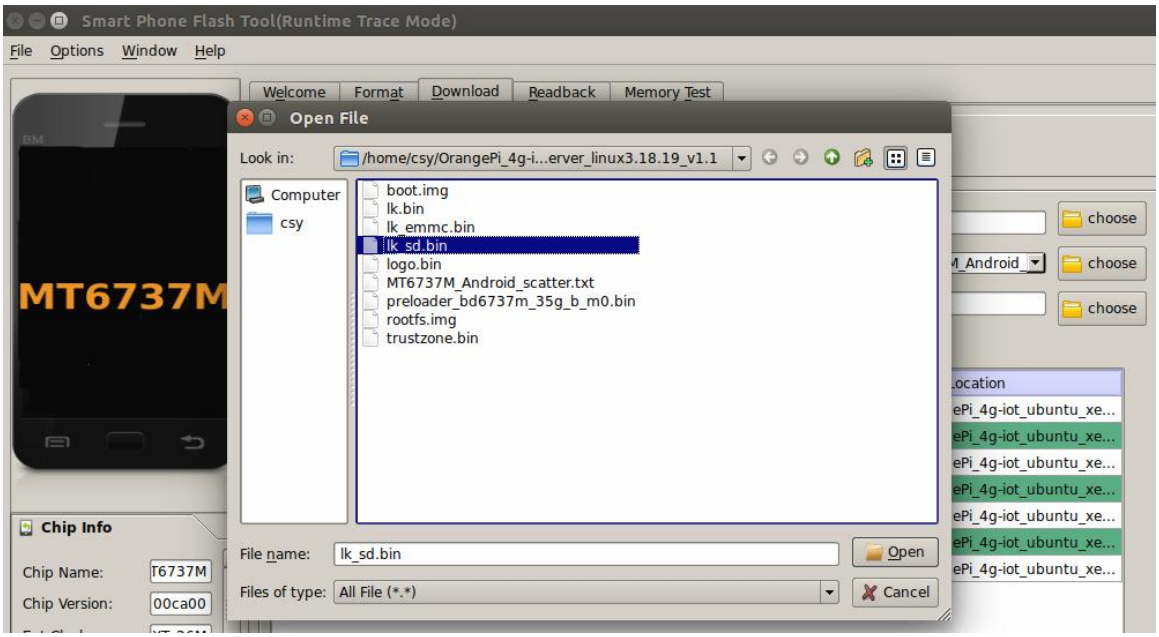
If the onboard space is not enough, you can burn the rootfs to the TF card. Proceed as follows.烧录支

Hold the lk of the mounted TF card rootfs

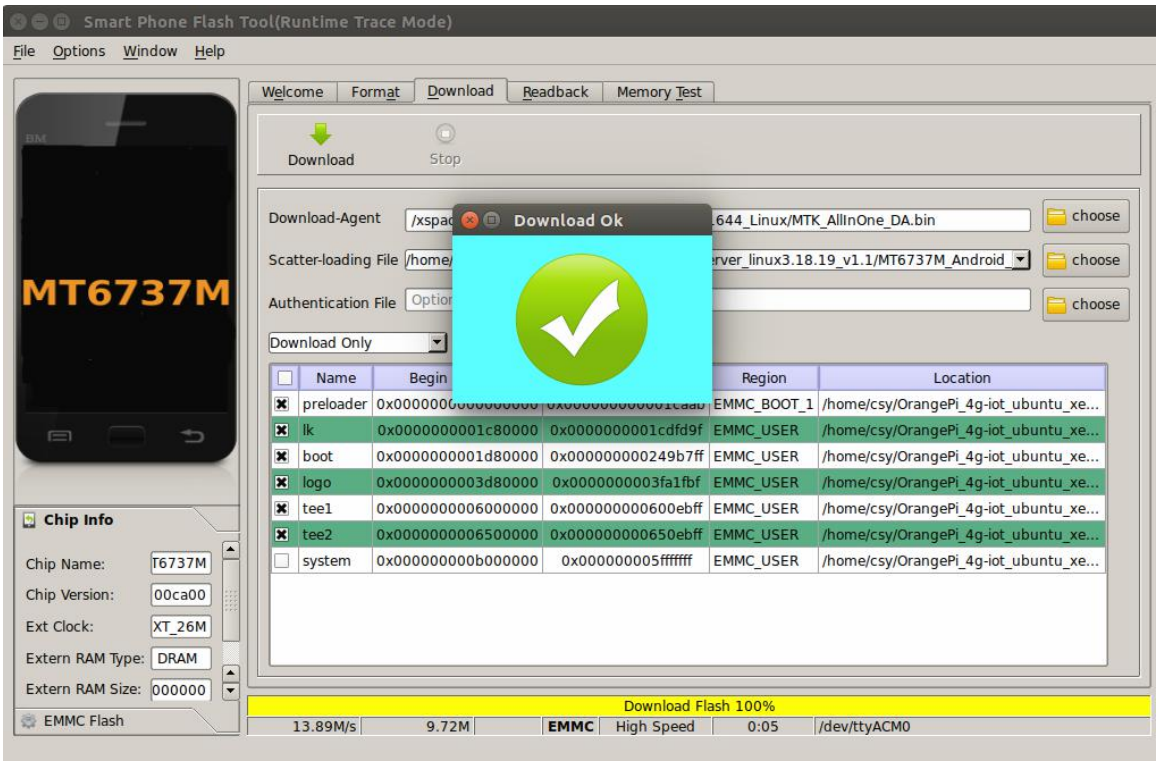
Click on the content in the box below



Select lk_sd.bin in the popup window.




Click download, download is complete



Burn rootfs to TF card

Prepare a TF card above 8G and class10. After formatting. Burn roofs.img as follows

```
pv rootfs.img | sudo dd of=/dev/sdb bs=1M
```

sdb is the device file corresponding to the TF card

If it is a windows system, you can also use Win32DiskImager software to burn.

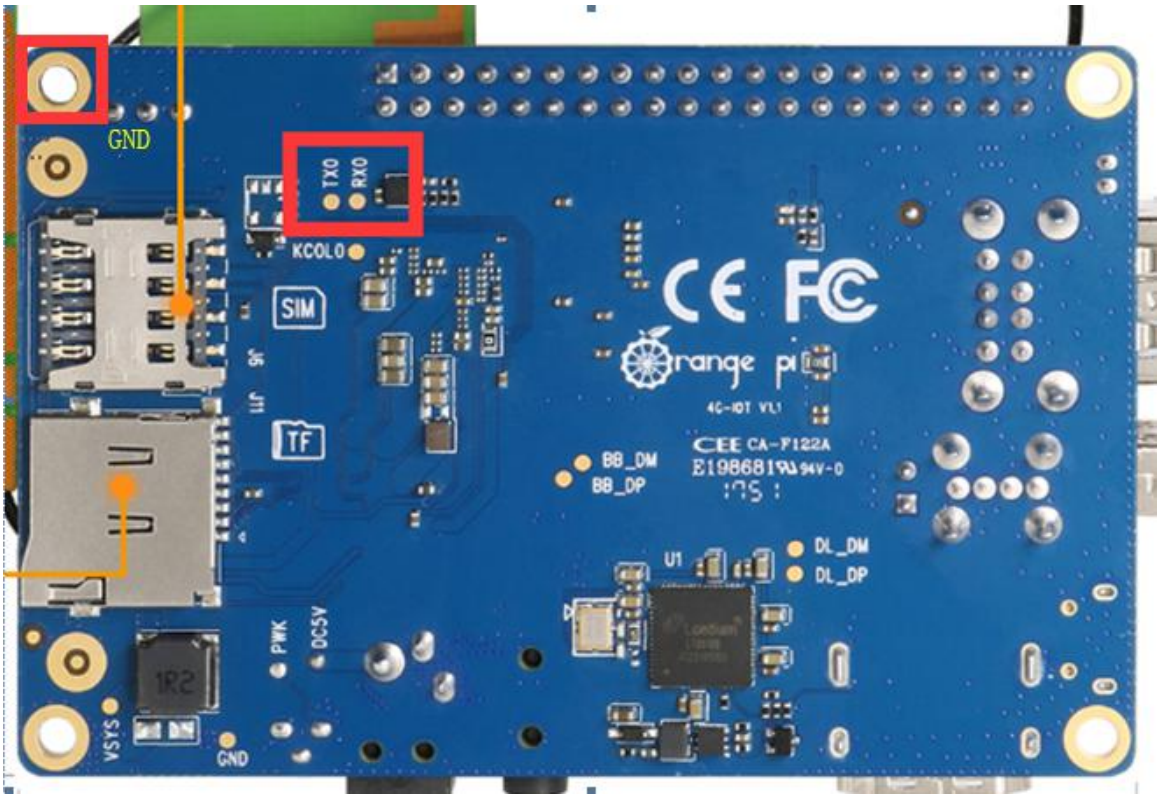
Start the system

Insert the tf card into the board and start it after power on.

# VI. Serial Debugging Tool

First need to prepare a USB to TTL serial cable, need to support 921600 baud rate

The debugging serial port of 4G-iot is not brought out, so users need to fly their own wires. The solder joints of the following figure boxes are TXD and RXD of the serial port. GND, RXD and TXD need to be brought out.




Connection mode:

Board TXD Connects USB to TTL RXD

Board RXD Connecting USB to TTL TXD

Board GND Connecting USB to TTL GND

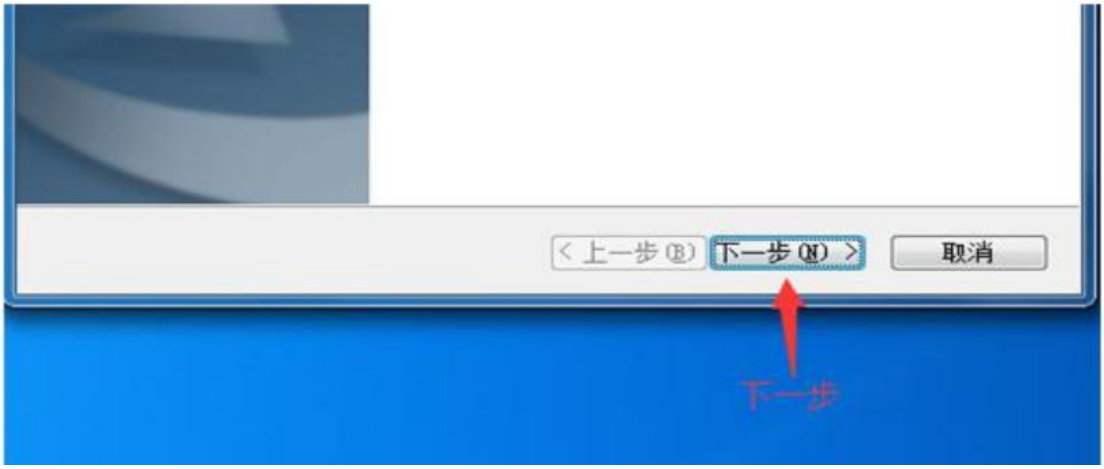
# 1. Usage based on Windows platform

In the process of using Orange Pi for project development, in order to get more debugging information, Orange Pi defaults support serial port information debugging. For developers, just prepare the materials mentioned above and can easily get serial debugging information. The serial port debugging tools used by different upper computers are similar, basically refer to the following ways to deploy. There are many tools for serial port debugging using the Windows platform. The commonly used tools are Putty. This section uses putty as an example for deployment.

- **Install USB driver**

Download the latest version of the driver PL2303_Prolific_DriverInstaller_v130.zip, download and extract.


Select application installation as an administrator



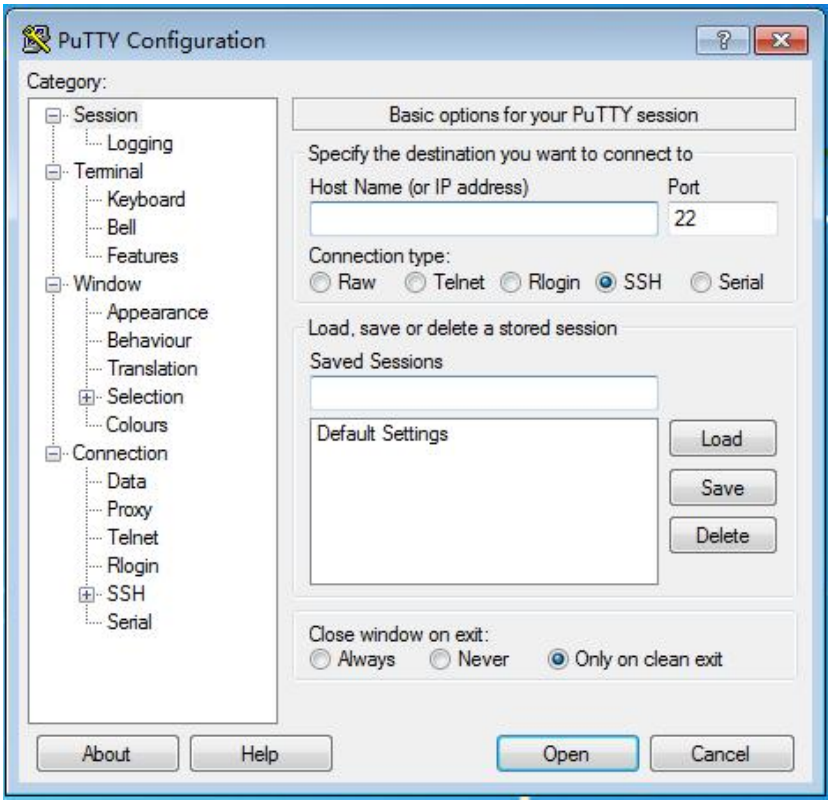
Waiting for installation and click.


- **Download and install Putty**

Putty can be downloaded from the address below, please choose the version that suits your development environment.

```
https://www.chiark.greenend.org.uk/~sgtatham/putty/latest.html
```

Double-click the downloaded putty.exe to open putty. The software interface is shown below.

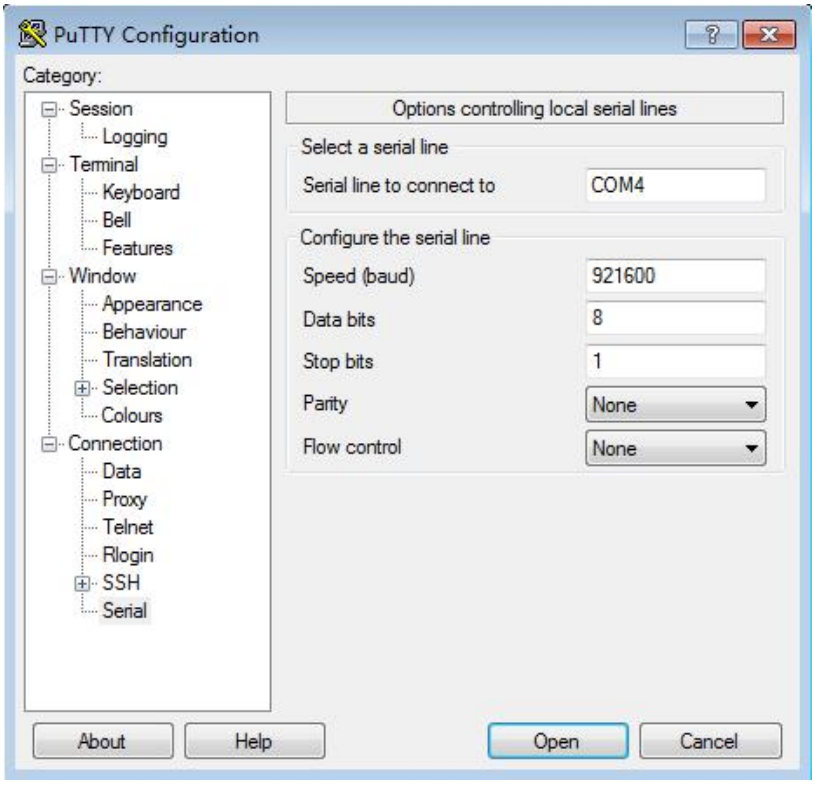


- **Acquisition of device information**

In Windows 7, we can check whether the serial port connection is normal and the serial port device No. through Device Manager. If the device is not recognized properly, please check if the driver is installed successfully. If there is a problem with the driver installation, try using the 360 Driver Master to scan the installation driver.


- **Putty configuration**

Set the serial port to the corresponding port number (COM4), turn off the flow control, and set the speed to 921600.




- **Start debugging serial output**

When the Orange Pi is powered on, putty will automatically print the serial port log information.

# 2. Usage based on Linux platform

There is a little difference using putty between Windows platform and Linux platform. The following mainly explains the differences. All operations are based on the Ubuntu 14.04 system.

- **Install and start Putty**

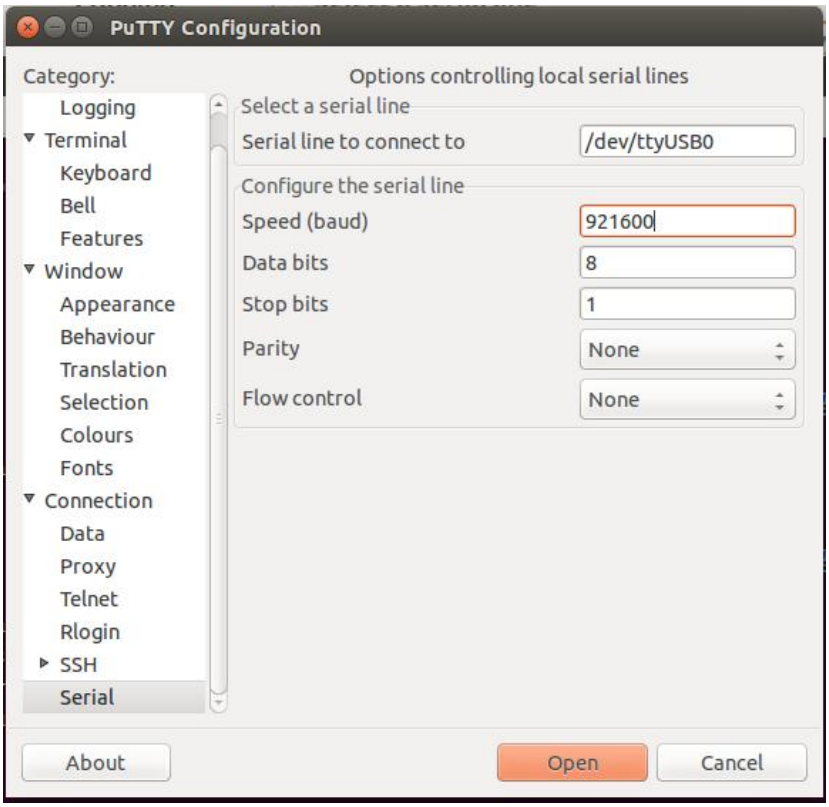
```
$ sudo apt-get install putty
$ sudo putty
```

- **Configuration Putty**

The serial port number can be viewed via ls /dev/ttyUSB*

The baud rate needs to be set to 921600

And turn off flow control


PuTTY Configuration
Category:
Logging
Terminal
Keyboard
Bell
Features
Window
Appearance
Behaviour
Translation
Selection
Colours
Fonts
Connection
Data
Proxy
Telnet
Rlogin
SSH
Serial
Options controlling local serial lines
Select a serial line
Serial line to connect to
/dev/ttyUSB0
Configure the serial line
Speed (baud)
921600
Data bits
8
Stop bits
1
Parity
None
Flow control
None
About
Open
Cancel

# VII. Usage of GPIO

There are two methods for using GPIO:

**1. There is a general GPIO Operation Interface under Linux, “/sys/class/gpio” 。**

You could find the configuration file under the directory of “/sys/class/gpio”, the control program can be divided into four steps:

①、Configuring GPIO：

There is a file ‘export’ under the directory of “/sys/class/gpio”, ， you could invoke it to achieve configuration. This file have numbered the GPIO, or you could also download the schematic or get form this manual.

For example, the 37th pin on 40pins is GPIO123,you could enter：# echo 123 > /sys/class/gpio/export，and come back to the directory “/sys/class/gpio”， there will have a new directory “./gpio123”,which include the configuration files of in and out of the IO port.

Note： export file have the permission of root write only. You have to execute as root the above command or executable file written in C.

②、Configuring the direction of GPIO (input and output):

Enter the command on the terminal：# echo "out" > /sys/class/gpio/gpio38/direction，which is set this GPIO as output.

③、Configuring the output level of GPIO：

Enter the command on the terminal：#echo 1 > /sys/class/gpio/gpio123/value，which is set this GPIO output high level， enter: echo "0" > /sys/class/gpio/gpio123/value for setting this GPIO output low level。

④、Shutting Down GPIO：

Enter the command on the terminal：#echo "38" > /sys/class/gpio/unexport， which could delete the GPIO configuration file，the directory ‘gpio38’ have been deleted.

# 2. Modify and Display the GPIO Status under ADB Mode

**Get the Open/Close Status of GPIO : cat /sys/devices/virtual/misc/mtgpio/pin**

Enter the command under ADB mode: cat /sys/devices/virtual/misc/mtgpio/pin , then it will show：

pin: [mode] [pull_sel] [din] [dout] [pull en] [dir] [ies] [smt]

0:11101010

1:01101010

、 、 、 、 、 、

22:1-100-10-1-1

、 、 、 、 、 、 、

42：00000110

**Corresponding meaning per row：**

IO Number：mode，pull select，input value，output value，pull enable，direction，ies

**Modify the Status of GPIO：**

You could get the 40pin GPIO specifications from this manual or from schematic which have been uploaded to our official website: http://www.orangepi.org/downloadresources/. For example the 37th pin on 40pins is GPIO123,

echo -wdout123 1 > pin -This is Set the GPIO to output high level

echo -wdout123 0> pin -This is Set the GPIO to output high level