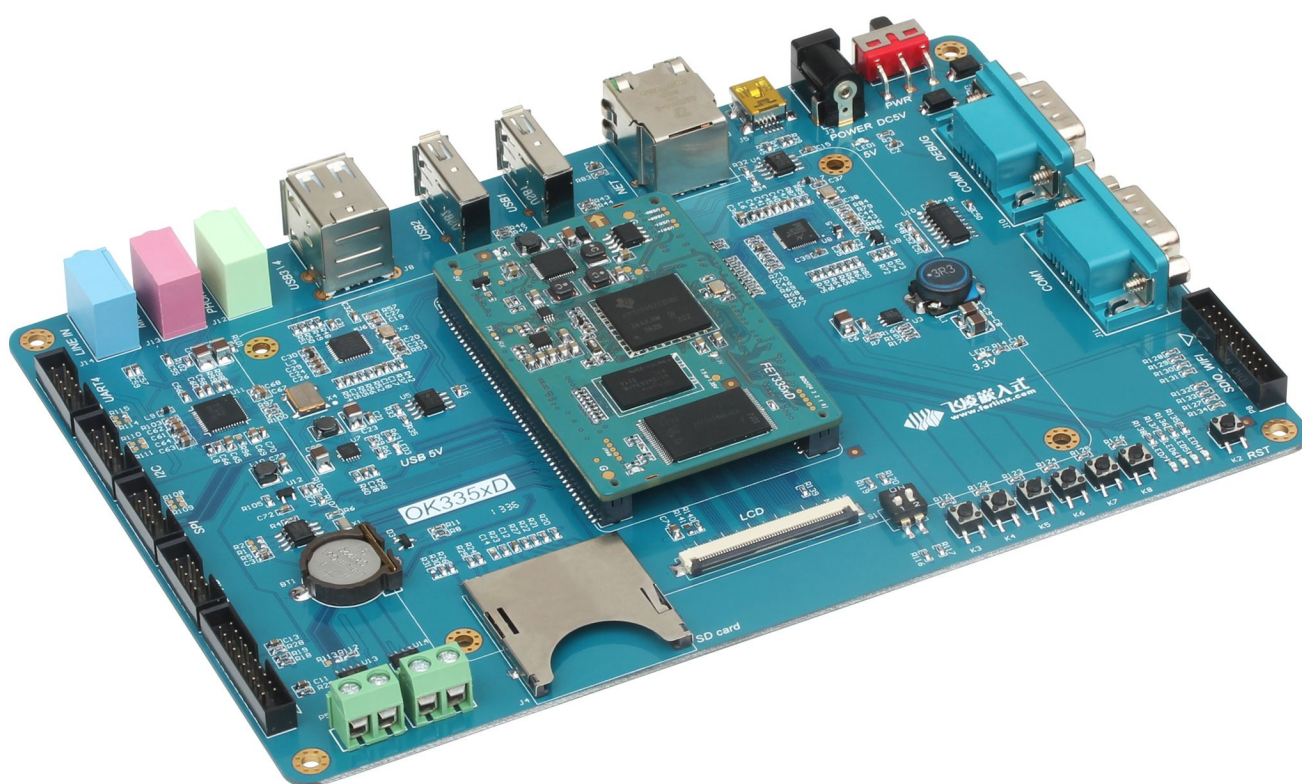


# OK335xD

Android 软件手册



Devoted to create the best embedded products

## 注意事项与维护

### 产品使用环境

供电电压: DC5V  $\pm$  10%  
运行温度: 商用级 0 – 80 °C  
工业级 -40 – 85 °C  
湿度: 10 – 90% (不结露)



### 注意事项

- 请勿带电插拔核心板及外围模块！
- 请遵循所有标注在产品上的警示和指引信息。
- 请保持本产品干燥。如果不慎被任何液体泼溅或浸润，请立刻断电并充分晾干。
- 使用中注意本产品的通风散热，避免温度过高造成元器件损坏。
- 请勿在多尘、脏乱的环境中使用或存放本产品。
- 请勿将本产品应用在冷热交替环境中，避免结露损坏元器件。
- 请勿粗暴对待本产品，跌落、敲打或剧烈晃动都可能损坏线路及元器件。
- 请勿使用有机溶剂或腐蚀性液体清洗本产品。
- 请勿自行修理、拆卸本公司产品，如产品出现故障请及时联系本公司进行维修。
- 擅自修改或使用未经授权的配件可能损坏本产品，由此造成的损坏将不予以保修。

如果产品出现故障，请联系飞凌技术服务部。

## 版权声明

本手册所有权由保定市飞凌嵌入式技术有限公司独家持有。未经本公司的书面许可，任何单位和个人无权以任何形式复制、传播、转载本手册的任何部分，否则一切后果由违者自负。

## 更新记录

日期	版本号	说明
2013.12	V1.3	<ol style="list-style-type: none"><li>1. Android2.3 支持 4.3 寸、10.4 寸 LCD 屏，同时支持电阻屏和电容屏</li><li>2. 增加 SPI、EEPROM、RS485、ADC、PWM 测试程序，提供硬件调用接口</li><li>3. 一键烧写时 LCD、LED 增加显示烧写进度功能</li><li>4. Android2.3 内核升级到 Linux3.2</li><li>5. Android2.3 下增加 USB、蓝牙支持</li><li>6. Android2.3 下增加对 U 盘存储的支持</li></ol>
2013.10	V1.2	<ol style="list-style-type: none"><li>1. 增加 Android2.3.4 的支持</li><li>2. 增加 Android 应用开发示例</li><li>3. Android2.3 支持 5 寸、7 寸、8 寸 LCD 屏，同时支持电阻屏和电容屏</li><li>4. 增加看门狗测试程序，提供硬件调用接口</li></ol>
2013.9	V1.1	<ol style="list-style-type: none"><li>1. Android4.2 增加 7 寸电容触摸屏的支持，暂不支持电阻屏</li><li>2. 增加 Line in 和 Mic 录音支持</li><li>3. 增加一键烧写 Android 系统功能</li></ol>
2013.8	V1.0	OK335xD-Android4.2 用户手册第一版

# 技术支持与更新

## 1 技术支持范围

- 1.1 本公司产品的软、硬件资源提供情况咨询；
- 1.2 本公司产品的软、硬件手册使用过程中遇到的问题；
- 1.3 本公司提供的 OEM、ODM 售后技术支持；
- 1.4 已购买本公司产品用户的资料丢失、更新后重新获取；
- 1.5 本公司产品的故障判断及售后维修服务；

## 2 技术讨论范围（非必解决问题）

- 2.1 源码的修改以及理解；
- 2.2 操作系统如何移植；
- 2.3 用户在自行修改以及开发中遇到的软硬件问题；

## 3 技术支持方式

- 3.1 电话（即时）：0312-3119192
- 3.2 邮箱（非即时）：
  - Linux 技术支持：linux@forlinx.com
  - Win CE 技术支持：wince@forlinx.com
  - Android 技术支持：android@forlinx.com
- 3.3 论坛（非即时）：<http://bbs.witech.com.cn>

## 4 技术支持时间

周一至周五：上午 9:00—11:30  
下午 13:30—17:00

公司按照国家法定节假日安排休息，在此期间无法提供技术支持，有问题请发邮箱或论坛技术支持区，我们会在工作日尽快给您回复。

## 5 资料下载方法

请登陆“[bbs.witech.com.cn](http://bbs.witech.com.cn)”找到“[开发板资料下载](#)”选择对应平台下载

# 目 录

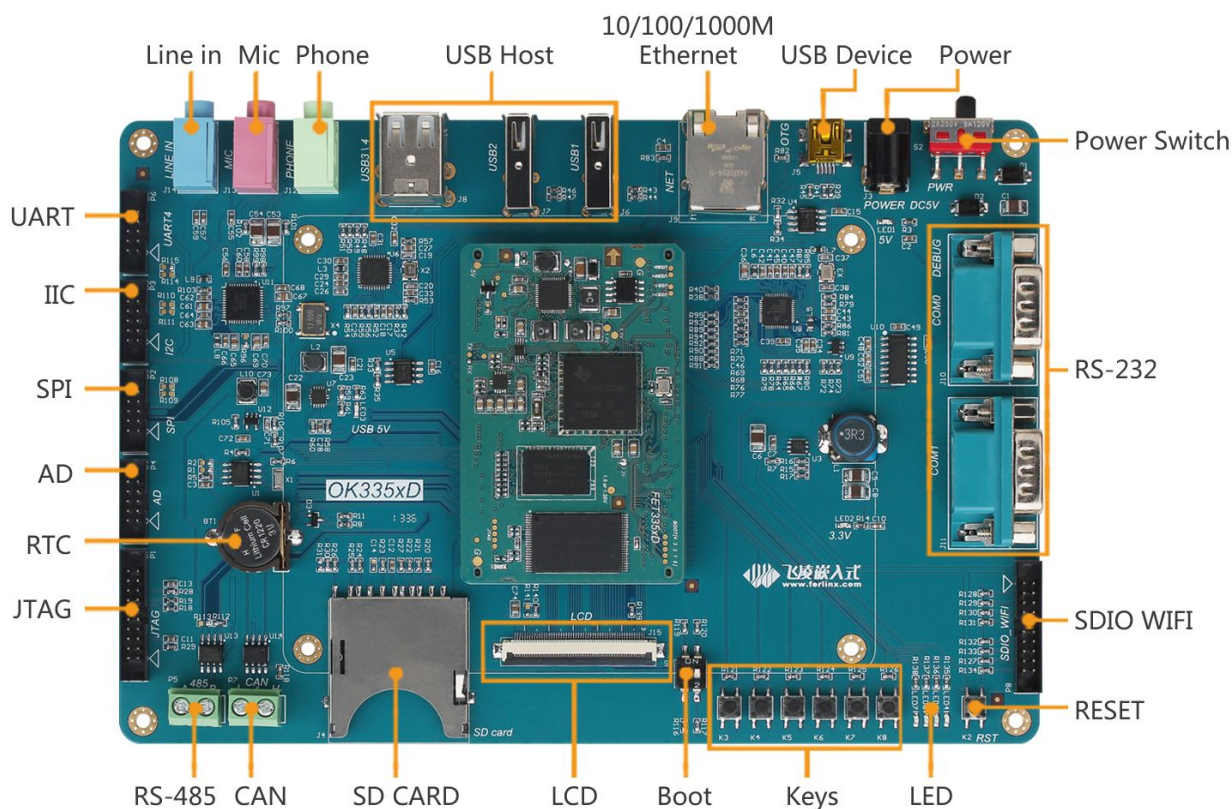
注意事项与维护	1
产品使用环境	1
注意事项	1
版权声明	2
更新记录	3
技术支持与更新	4
1 技术支持范围	4
2 技术讨论范围（非必解决问题）	4
3 技术支持方式	4
4 技术支持时间	4
5 资料下载方法	4
目 录	5
第一章 OK335xD 简介	7
第二章 Android 编译环境的搭建	8
2.1 安装 Ubuntu 12.04.2 及编译环境	8
2.2 安装编译 Android 系统所需要的库	8
2.2.1 安装编译 Android4.2 系统所需要的库	8
2.2.2 安装编译 Android2.3.4 系统所需要的库	8
2.3 Android 系统的编译	9
2.3.1 解压 Android 源码	9
2.3.2 Android 系统的编译	9
2.3.3 驱动代码路径	11
2.4 通过 Uboot 设置屏幕参数	12
2.4.1 更改屏幕大小	12
第三章 安装 Android 系统	14
3.1 制作用于一键烧写 Android 系统的 SD 卡	14
3.2 分步更新系统	17
第四章 Android 功能使用及测试	18
4.1 Android 主界面展示	18

4.2 Android 应用程序	19
4.3 Android 查看图片	20
4.4 Android 编辑图片	21
4.5 Android 播放视频	22
4.6 Android 播放音乐	23
4.7 Android 录音（支持 Line in 和 Mic 输入）	24
4.8 Android 调节音量	26
4.9 Android 背光控制	27
4.10 Android 设置时间（外部 RTC）	28
4.11 Android 有线连接（支持千兆以太网）	28
4.12 Android 按键	29
4.13 Android 2D、3D 测试	30
4.14 Android SD 卡测试	31
4.15 Android USB 鼠标测试	34
4.16 Android USB 存储测试	35
4.17 Android 看门狗测试	36
4.18 Android 串口测试	37
4.19 Android RS485 测试	39
4.20 Android SPI 测试	41
4.21 Android LED 测试	42
4.22 Android EEPROM 测试	43
第五章 Android 应用程序开发	44
5.1 建立 Android 应用开发环境	44
5.1.1 下载并安装 JDK（Java SE Development Kit）	44
5.1.2 安装 adt-bundle-windows	46
5.1.3 创建 helloworld 工程	48
5.1.4 使用 adb	55
5.2 在 Android 程序中访问硬件	57
附录 1: Ubuntu 的安装与设置	65
F1.1 Ubuntu 的安装	65
F1.2 Linux 终端	70
F1.3 Ubuntu12.04.02 root 用户登录设置	72
F1.4 设置 Ubuntu 网络参数	72

## 第一章 OK335xD 简介

OK335xD 开发板（工业级）是一款基于飞凌 FET335xD 核心板的高性能开发平台，采用 200Pin DIP 插针方式板对板连接器，插架采用高速 100Pin 镀金接插件，最大限度的保证稳定性和可靠性，整板引出了大部分调试接口与功能接口，可外接 4.3 寸、5 寸、7 寸、8 寸、10.4 寸液晶屏。出色的性能和工业级标准，为您产品保驾护航，缩短开发周期，提高开发效率。

下图为 OK335xD 接口简述，详细硬件介绍请参见硬件手册。



## 第二章 Android 编译环境的搭建

OK335xD 的所有软件都向用户开源, 用户可以通过光盘获取软件和硬件的文档及源码, Android4.2 的操作系统源码及相应二进制文件夹等由于文件太大, 存放于飞凌 OK335xD 配套光盘的 C 盘和 D 盘:

📁 用户光盘 C: Android4.2.part1.rar

📁 用户光盘 D: Android4.2.part2.rar

须将 C、D 盘的两个压缩包复制到电脑本地硬盘并放于同一文件夹路径下, 然后解压即可自动合并为一个文件夹。以下文档中提到的所有目录 (bin, src, filesystem, tools, doc 等) 都是以合并后的文件夹为根目录

### 2.1 安装 Ubuntu 12.04.2 及编译环境

在这里建议用户使用 Ubuntu 真机进行编译, ubuntu 的安装与配置见附录 1。

注: 推荐电脑配置处理器: Core(TM) i7 内存: 4G 以上

### 2.2 安装编译 Android 系统所需要的库

#### 2.2.1 安装编译 Android4.2 系统所需要的库

##### 1. 安装 libncurses5-dev:i386

```
#apt-get install libncurses5-dev:i386
```

##### 2. 安装编译 Android 所需要的库文件

```
#cd /work/tools  
#./install-devel-packages.sh
```

#### 2.2.2 安装编译 Android2.3.4 系统所需要的库

##### 1. 安装编译 Android 所需要的库文件

```
#sudo apt-get install git-core gnupg flex bison gperf build-essential zip curl zlib1g-dev  
libc6-dev lib32ncurses5-dev ia32-libs x11proto-core-dev libx11-dev lib32readline-gplv2-dev  
lib32z1-dev libgl1-mesa-dev gcc-multilib g++-multilib mingw32 tofrodos python-markdown  
libxml2-utils xsltproc
```

##### 2. 需要把编译器设置成 4.4

```
#sudo apt-get install gcc-4.4
```

```
#sudo apt-get install g++-4.4
#sudo rm -rf /usr/bin/gcc /usr/bin/g++
#sudo ln -s /usr/bin/gcc-4.4 /usr/bin/gcc
#sudo ln -s /usr/bin/g++-4.4 /usr/bin/g++
#sudo apt-get install g++-4.4-multilib
```

## 2.3 Android 系统的编译

Android 4.2 系统的源码包 Android4.2.2-source.tar.gz 位于 src/Android4.2.2 目录下，将它拷贝到 ubuntu 文件夹/work/forlinx 下；

Android 2.3.4 系统的源码包 Android2.3.4-source.tar.gz 位于 src/Android2.3.4 目录下，将它拷贝到 ubuntu 文件夹/root/work 下。

注：防止编译出现不必要的错误，请您也把代码解压到上面提到的目录下。

### 2.3.1 解压 Android 源码

解压命令如下：

Android 4.2 源码：

```
#cd /work/forlinx
#tar -zxvf Android4.2.2-source.tar.gz
```

Android 2.3.4 源码：

```
#cd /root/work
#tar -zxvf Android2.3.4-source.tar.gz
```

注：解压之后的文件夹 TI\_Android\_JB\_4.2.2\_DevKit\_4.1.1 中包括文件系统、kernel、bootloader，rowboat-android-gingerbread-am335xevm 中包括系统、kernel、bootloader。

### 2.3.2 Android 系统的编译

编译 android4.2 过程依赖于三个脚本文件，这些脚本文件包含在源码解压之后生成的目录下，并进入该目录，执行以下命令：

```
#cd /work/forlinx/TI_Android_JB_4.2.2_DevKit_4.1.1
#export PATH=$PWD/prebuilts/gcc/linux-x86/arm/arm-eabi-4.6/bin:$PATH
#./build_android_clean.sh（清理文件系统、内核、bootloader 中原先生成的文件）
#./build_android_ok335xd_compile.sh（自动编译文件系统、内核和 bootloader）
#./pack-ubi.sh（把所生成的文件系统制作成 ubi 文件，并把其他所需要的文件拷贝到 mkubifs 目录下）
```

编译 android2.3.4 过程依赖于三个脚本文件，这些脚本文件包含在源码解压之后生成的目录下，并进入该目录，执行以下命令：

```
#cd /root/work/rowboat-android-gingerbread-am335xevm
```

```
#export PATH=$PWD/prebuilt/linux-x86/toolchain/arm-eabi-4.4.0/bin:$PATH
#./build_android_clean.sh (清理文件系统、内核、bootloader 中原先生成的文件)
#./build_android_ok335xd_compile.sh (自动编译文件系统、内核和 bootloader)
#./pack-ubi.sh (把所生成的文件系统制作成 ubi 文件, 并把其他所需要的文件拷贝到 mkubifs 目录下)
```

注: 1.这个编译过程需要耗费几个小时的时间, 时间长短视电脑配置而定。

2.您也可以进入到 u-boot、kernel、system 的目录下单独编译 u-boot、kernel 和 system。

编译 u-boot:

```
#make CROSS_COMPILE=arm-eabi- distclean
#make CROSS_COMPILE=arm-eabi- ok335x_config
#make CROSS_COMPILE=arm-eabi- -j8
```

编译 kernel:

```
#make ARCH=arm CROSS_COMPILE=arm-eabi- distclean
#make ARCH=arm CROSS_COMPILE=arm-eabi- ok335xd_evm_android_defconfig
#make ARCH=arm CROSS_COMPILE=arm-eabi- uImage -j8
```

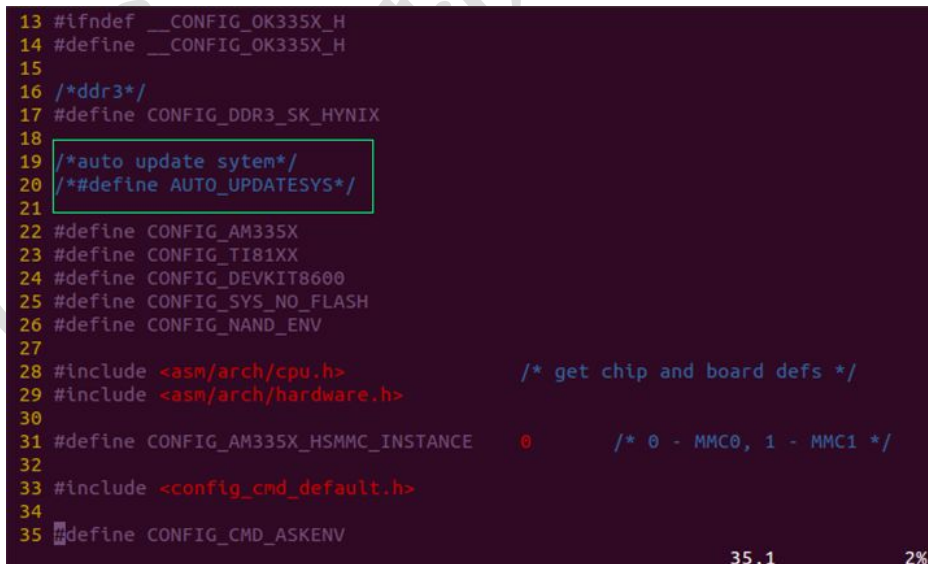
编译 system:

```
#./compile.sh
```

3.批量烧写版本的 uboot 编译

该版本的 uboot 需要打开条件编译宏 AUTO\_UPDATESYS (include/configs/ok335x.h), 然后像编译普通 uboot 一样进行编译即可。

编译宏打开前如下图:



```
13 #ifndef __CONFIG_OK335X_H
14 #define __CONFIG_OK335X_H
15
16 /*ddr3*/
17 #define CONFIG_DDR3_SK_HYNIX
18
19 /*auto update sytem*/
20 /*#define AUTO_UPDATESYS*/
21
22 #define CONFIG_AM335X
23 #define CONFIG_TI81XX
24 #define CONFIG_DEVKIT8600
25 #define CONFIG_SYS_NO_FLASH
26 #define CONFIG_NAND_ENV
27
28 #include <asm/arch/cpu.h> /* get chip and board defs */
29 #include <asm/arch/hardware.h>
30
31 #define CONFIG_AM335X_HSMC_INSTANCE 0 /* 0 - MMC0, 1 - MMC1 */
32
33 #include <config_cmd_default.h>
34
35 #define CONFIG_CMD_ASKENV
```

编译宏打开后如下图：

```

13 #ifndef __CONFIG_OK335X_H
14 #define __CONFIG_OK335X_H
15
16 /*ddr3*/
17 #define CONFIG_DDR3_SK_HYNIX
18
19 /*auto update sytem*/
20 #define AUTO_UPDATESYS
21
22 #define CONFIG_AM335X
23 #define CONFIG_TI81XX
24 #define CONFIG_DEVKIT8600
25 #define CONFIG_SYS_NO_FLASH
26 #define CONFIG_NAND_ENV
27
28 #include <asm/arch/cpu.h> /* get chip and board defs */
29 #include <asm/arch/hardware.h>
30
31 #define CONFIG_AM335X_HSMMC_INSTANCE 0 /* 0 - MMC0, 1 - MMC1 */
32
33 #include <config_cmd_default.h>
34
35 #define CONFIG_CMD_ASKEV

```

### 2.3.3 驱动代码路径

设备	驱动程序源代码在内核中的位置	对应的设备名
网卡驱动	drivers/net/ethernet/ti/cpsw.c	eth0
LED 驱动	drivers/misc/led_dev.c	/dev/led
LCD 背光驱动	drivers/video/backlight/pwm_bl.c	/sys/class/backlight
USB 接口 U 盘	drivers/usb/storage/	/dev/sdX
USB 鼠标	drivers/hid/usbhid/	/dev/input/mice
Flash ECC 校验	drivers/mtd/nand/nand_ecc.c	无
Nand Flash 驱动	drivers/mtd/nand/omap2.c	/dev/mtd/mtdX
UBI 文件系统	fs/ubifs/	无
SD 卡驱动	drivers/mmc/card/	/dev/block/mmcblk0pX
LCD FrameBuffer	drivers/video/da8xx_fb.c	/dev/graphics/fb0
电阻触摸驱动	drivers/input/touchscreen/ti_tsc.c	/dev/input/event0
电容触摸驱动	drivers/input/touchscreen/ft5x06_ts.c	/dev/input/event1
RTC 实时时钟驱动	drivers rtc/rtc-ds1307.c	/dev/rtc0
音频驱动(IIS 接口)	sound/soc/codec/tlv320aic3x.c	/dev/snd/controlC0, pcmC0D0c,pcmC0D0p, seq,timer
SPI 驱动	drivers/spi/spidev.c	/dev/spidev2.0
串口(含三个串口)	drivers/tty/serial/omap-serial.c	/dev/ttyO0,1,4
按键驱动	drivers/input/keyboard/gpio_keys.c	/dev/input/event2
看门狗驱动	drivers/watchdog/omap_wdt.c	/dev/watchdog

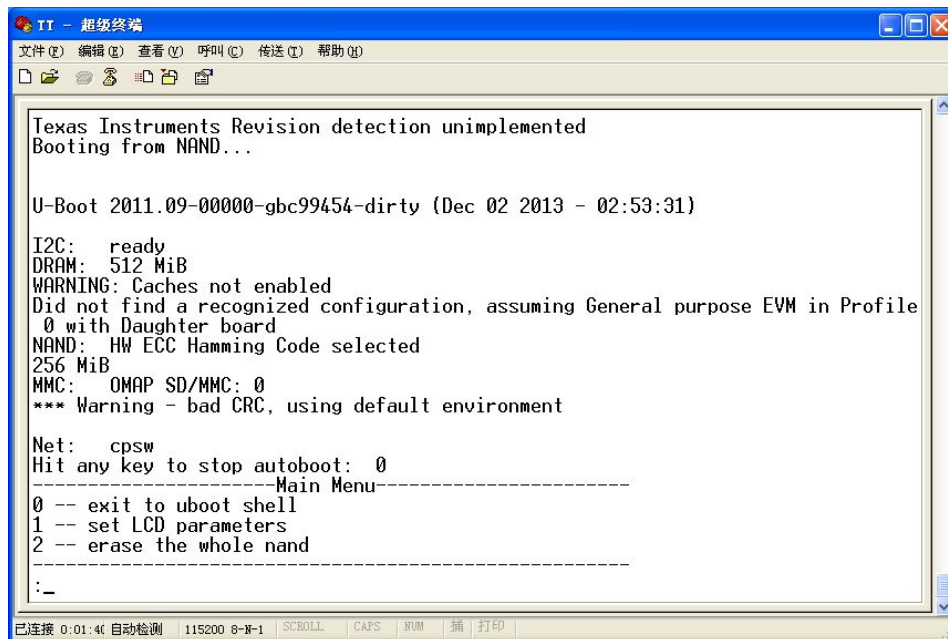
注：现在 android4.2 不支持电阻屏。

## 2.4 通过 Uboot 设置屏幕参数

### 2.4.1 更改屏幕大小

OK335xD 产品默认支持 7 寸屏，如果用户需要使用 4.3/5/8/10.4 寸屏幕可以通过 uboot 进行参数设置，方法如下：

1. Uboot 启动时按任意键进入如下菜单：



```

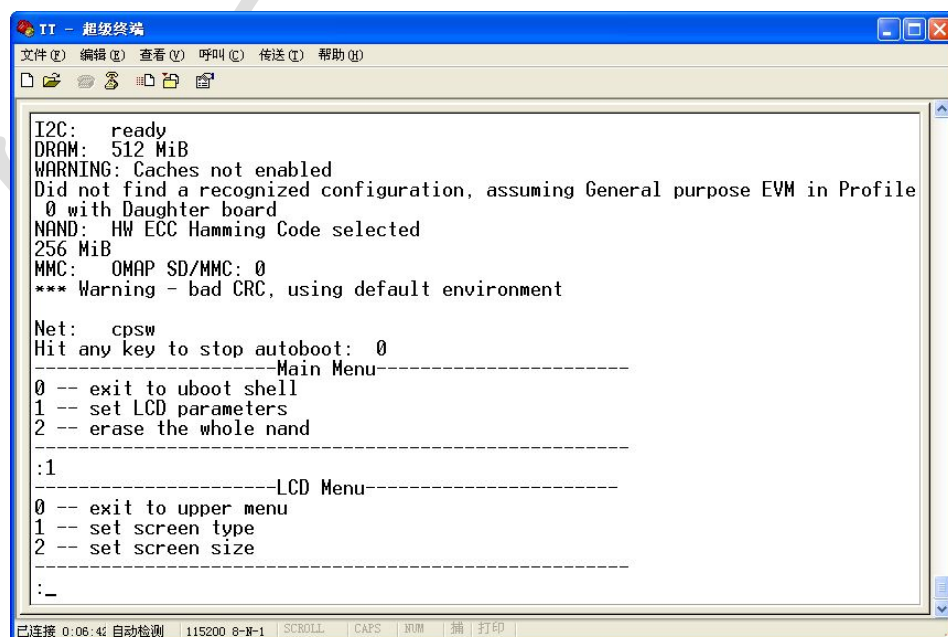
Texas Instruments Revision detection unimplemented
Booting from NAND...

U-Boot 2011.09-00000-gbc99454-dirty (Dec 02 2013 - 02:53:31)

I2C:  ready
DRAM: 512 MiB
WARNING: Caches not enabled
Did not find a recognized configuration, assuming General purpose EVM in Profile
0 with Daughter board
NAND:  HW ECC Hamming Code selected
256 MiB
MMC:  OMAP SD/MMC: 0
*** Warning - bad CRC, using default environment

Net:  cpsw
Hit any key to stop autoboot:  0
-----Main Menu-----
0 -- exit to uboot shell
1 -- set LCD parameters
2 -- erase the whole nand
:~
    
```

2. 输入 1 进入如下界面：

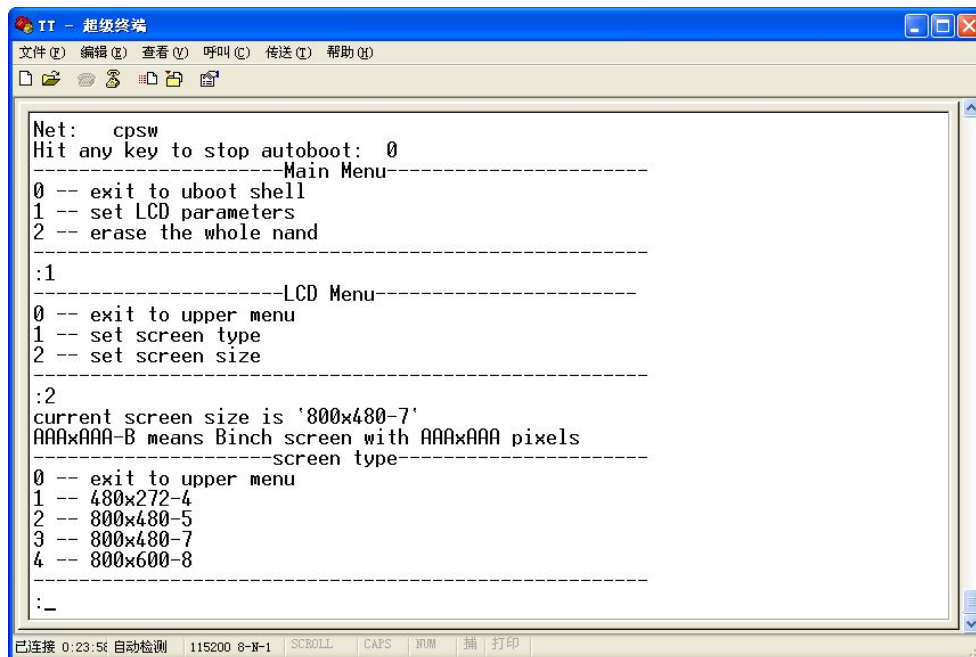


```

I2C:  ready
DRAM: 512 MiB
WARNING: Caches not enabled
Did not find a recognized configuration, assuming General purpose EVM in Profile
0 with Daughter board
NAND:  HW ECC Hamming Code selected
256 MiB
MMC:  OMAP SD/MMC: 0
*** Warning - bad CRC, using default environment

Net:  cpsw
Hit any key to stop autoboot:  0
-----Main Menu-----
0 -- exit to uboot shell
1 -- set LCD parameters
2 -- erase the whole nand
:1
-----LCD Menu-----
0 -- exit to upper menu
1 -- set screen type
2 -- set screen size
:~
    
```

3. 输入 2 进入屏幕大小设置菜单，程序打印出当前的屏幕大小和供设置的大小菜单：

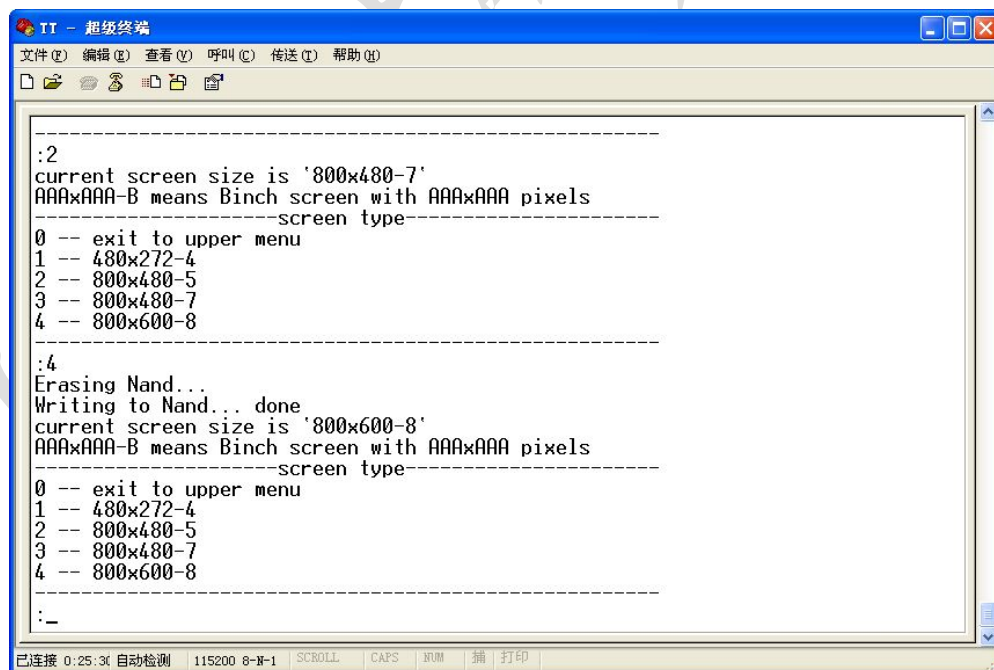


```

TI - 超级终端
文件(F) 编辑(E) 查看(V) 呼叫(C) 传送(T) 帮助(H)
[Icons]

Net: cpsw
Hit any key to stop autoboot: 0
-----Main Menu-----
0 -- exit to uboot shell
1 -- set LCD parameters
2 -- erase the whole nand
-----
:1
-----LCD Menu-----
0 -- exit to upper menu
1 -- set screen type
2 -- set screen size
-----
:2
current screen size is '800x480-7'
AAA×AAA-B means Binch screen with AAA×AAA pixels
-----screen type-----
0 -- exit to upper menu
1 -- 480x272-4
2 -- 800x480-5
3 -- 800x480-7
4 -- 800x600-8
-----
:_
已连接 0:23:56 自动检测 115200 8-N-1 SCROLL CAPS NUM 捕捉 打印
  
```

4. 假如用户要设置为 8 寸屏，则输入 4 会显示如下界面，表示设置 8 寸屏成功：



```

TI - 超级终端
文件(F) 编辑(E) 查看(V) 呼叫(C) 传送(T) 帮助(H)
[Icons]

:2
current screen size is '800x480-7'
AAA×AAA-B means Binch screen with AAA×AAA pixels
-----screen type-----
0 -- exit to upper menu
1 -- 480x272-4
2 -- 800x480-5
3 -- 800x480-7
4 -- 800x600-8
-----
:4
Erasing Nand...
Writing to Nand... done
current screen size is '800x600-8'
AAA×AAA-B means Binch screen with AAA×AAA pixels
-----screen type-----
0 -- exit to upper menu
1 -- 480x272-4
2 -- 800x480-5
3 -- 800x480-7
4 -- 800x600-8
-----
:_
已连接 0:25:30 自动检测 115200 8-N-1 SCROLL CAPS NUM 捕捉 打印
  
```

注：设置成功后重启系统，或者退回到 uboot 命令行执行 bootd 命令即可使用新参数启动系统。

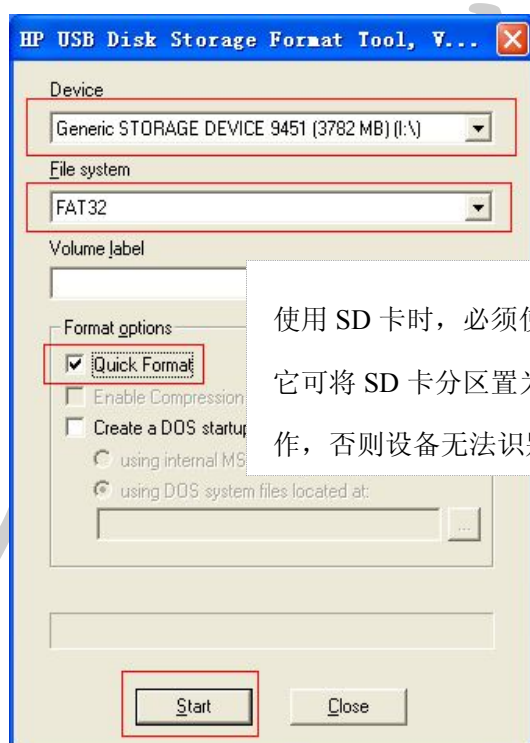
## 第三章 安装 Android 系统

### 3.1 制作用于一键烧写 Android 系统的 SD 卡

#### 1. SD 卡格式化

请使用光盘 tools 目录下的 HPUSBFW.EXE 安装到 PC 上（支持 XP、win7）格式化 SD 卡。

- 把 SD 卡通过读卡器接入 PC
- 打开您安装到 PC 的工具，出现类似提示如下
- 选择 “FAT32” 系统格式
- 点击 “Start”
- 等待格式化完成，点击 “OK”

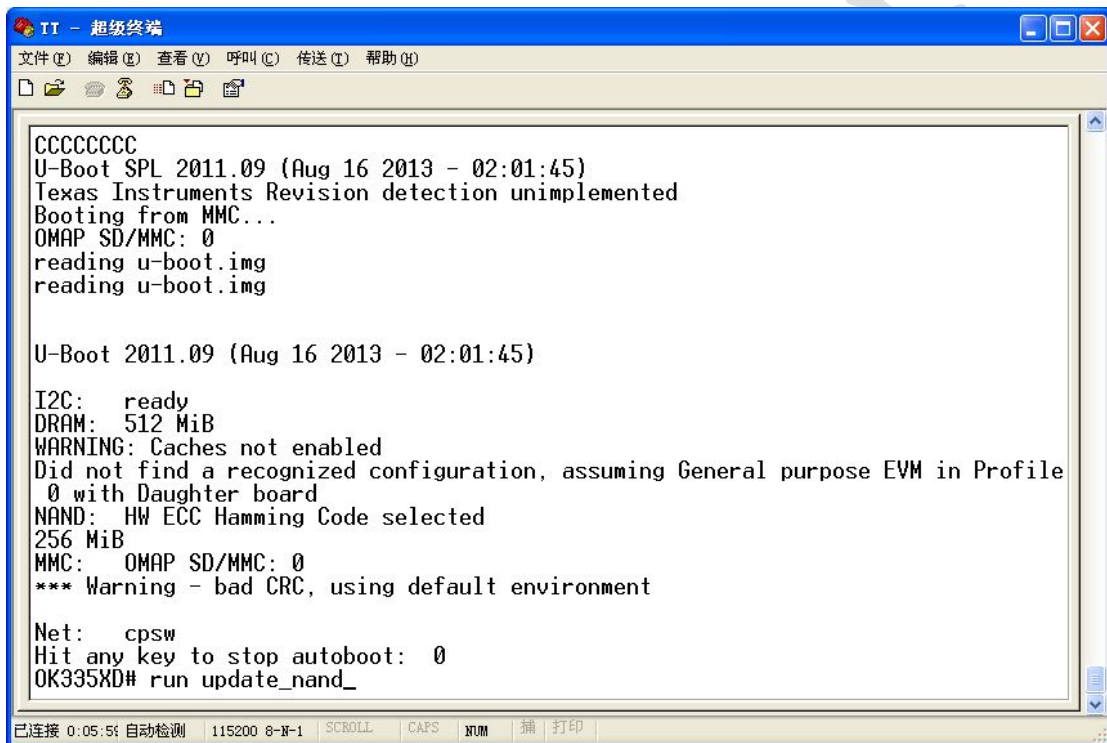


使用 SD 卡时，必须使用 HPUSBFW.EXE 工具或  
它可将 SD 卡分区置为活动分区的进行格式化操  
作，否则设备无法识别 SD 卡信息。



制作完成之后请把 Android 编译生成的文件 MLO、u-boot.img、uImage、ubi.img (生成的文件在 Android 源码目录下的 temp 文件中), 拷贝到 SD 内。或者您使用我们 bin/batch 目录下提供了 SD 卡批量烧写 NAND 的 uboot 版本, 使用该版本的 uboot 进行烧写, 用户无需敲入任何命令。

- 1) 将 bin/batch 目录下的 MLO、u-boot.img 以及 bin 目录下的 uImage、ubi.img 拷入 SD 卡 boot (FAT32) 分区。
- 2) 将拨码开关拨到 SD 卡启动项。然后把 SD 卡插入到开发板的 SD 卡插槽中上电启动。



```

CCCCCCCC
U-Boot SPL 2011.09 (Aug 16 2013 - 02:01:45)
Texas Instruments Revision detection unimplemented
Booting from MMC...
OMAP SD/MMC: 0
reading u-boot.img
reading u-boot.img

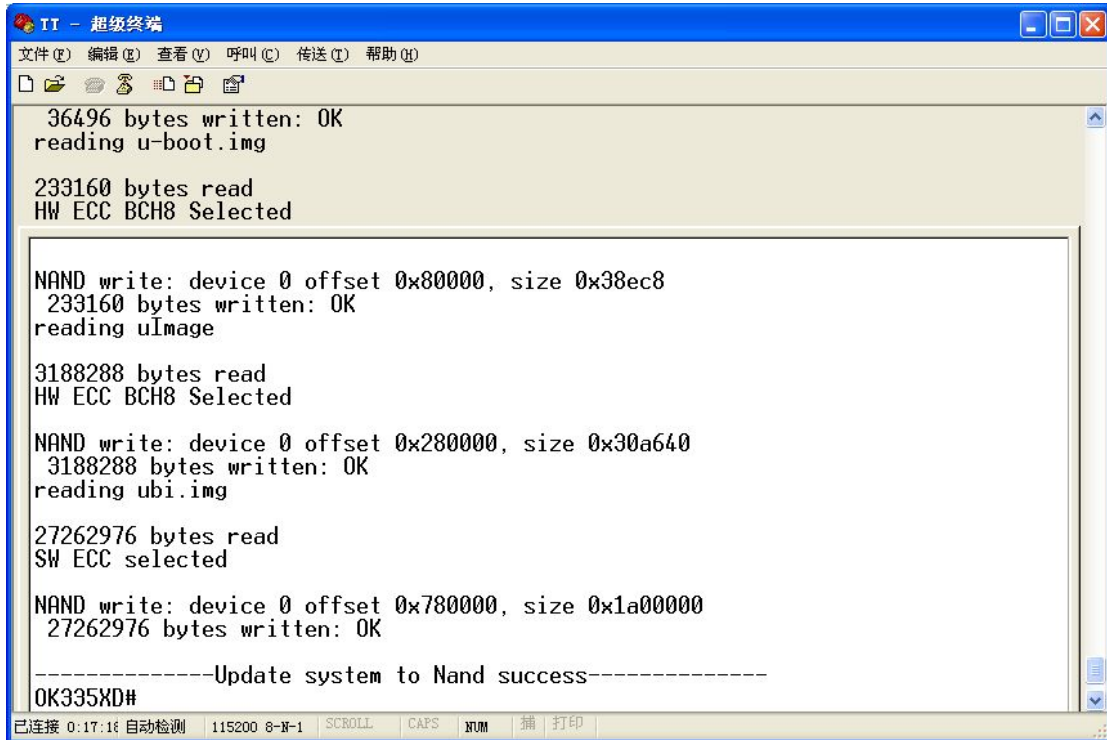
U-Boot 2011.09 (Aug 16 2013 - 02:01:45)

I2C:  ready
DRAM:  512 MiB
WARNING: Caches not enabled
Did not find a recognized configuration, assuming General purpose EVM in Profile
0 with Daughter board
NAND:  HW ECC Hamming Code selected
256 MiB
MMC:   OMAP SD/MMC: 0
*** Warning - bad CRC, using default environment

Net:   cpsw
Hit any key to stop autoboot:  0
OK335XD# run update_nand_
    
```

烧写过程中 led 灯会有流水效果，LCD 和串口会有进度显示，烧写成功后 led 长亮并且串口打印：

Update system to nand success.



```

II - 超级终端
文件(F) 编辑(E) 查看(V) 呼叫(C) 传送(T) 帮助(H)

36496 bytes written: OK
reading u-boot.img

233160 bytes read
HW ECC BCH8 Selected

NAND write: device 0 offset 0x80000, size 0x38ec8
233160 bytes written: OK
reading uImage

3188288 bytes read
HW ECC BCH8 Selected

NAND write: device 0 offset 0x280000, size 0x30a640
3188288 bytes written: OK
reading ubi.img

27262976 bytes read
SW ECC selected

NAND write: device 0 offset 0x780000, size 0x1a00000
27262976 bytes written: OK

-----Update system to Nand success-----
OK335XD#
已连接 0:17:16 自动检测 115200 8-N-1 SCROLL CAPS NUM 捕 打印
    
```

**注意：**该模式下的 **uboot** 在烧写成功后便处于死循环状态，用户可断电拔卡后进行另一台设备烧写。批量烧写方式适用于批量产品客户，使用此方法不需要人工干预，不需要连接 PC，一张 SD 卡即可把系统快速烧写到设备中。

## 3.2 分步更新系统

分步更新系统是指更新 nand 中的单个文件而不是整个系统，用户只需将要更新的文件拷入 SD 卡，然后设置开发板从 nand 启动，进入 uboot 命令行后执行以下操作（注更新完成后需要拔出 SD 卡）：

### 1. 更新 MLO, u-boot.img 文件

```
mmc rescan
fatload mmc 0 82000000 MLO
nand erase 0 80000
Nandecce hw 2
nand write.i 82000000 0 ${filesize}
```

```
mmc rescan
Fatload mmc 0 82000000 u-boot.img
nand erase 80000 200000
Nandecce hw 2
nand write.i 82000000 80000 ${filesize};
```

### 2. 更新内核文件

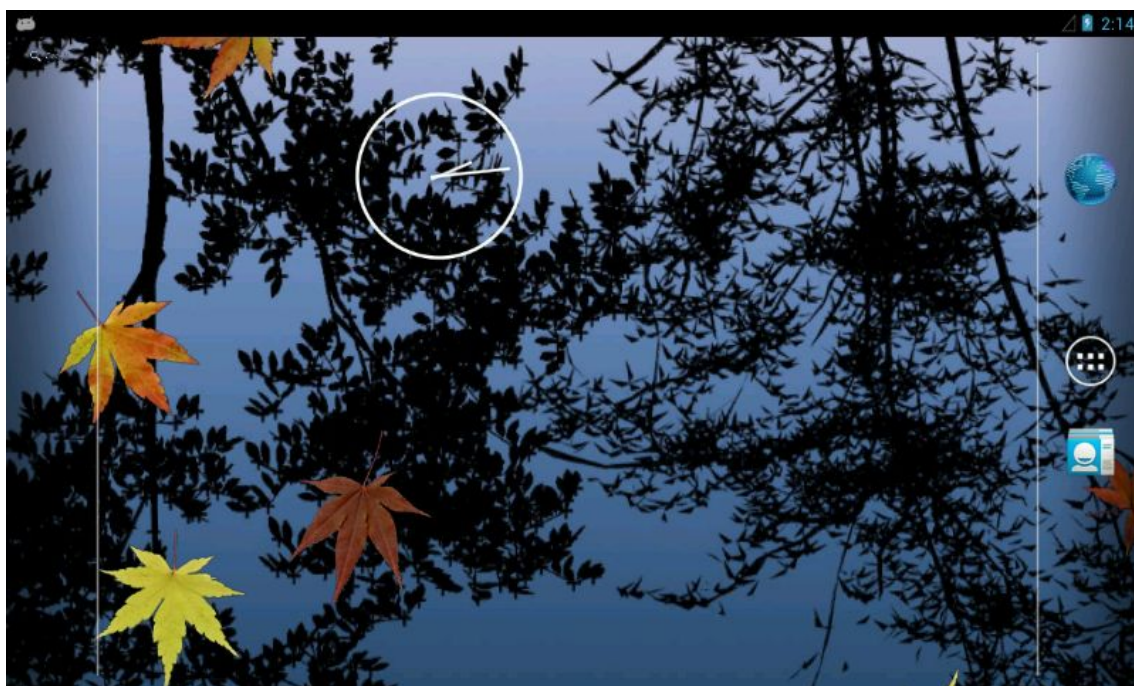
```
mmc rescan
fatload mmc 0 82000000 ulmage
nand erase 280000 500000
nandecce hw 2
nand write.i 82000000 280000 ${filesize}
```

### 3. 更新文件系统

```
mmc rescan
fatload mmc 0 82000000 ubi.img
nand erase 780000 1F880000
Nandecce sw
Nand write.i 82000000 780000 ${filesize};
```

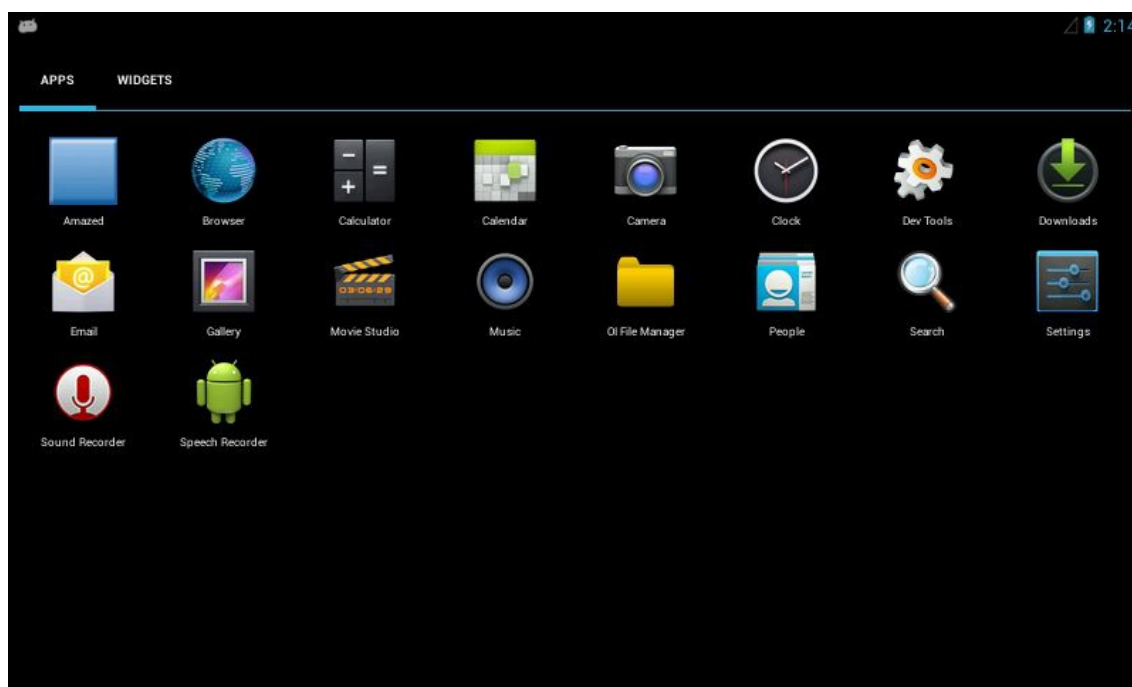
## 第四章 Android 功能使用及测试

### 4.1 Android 主界面展示




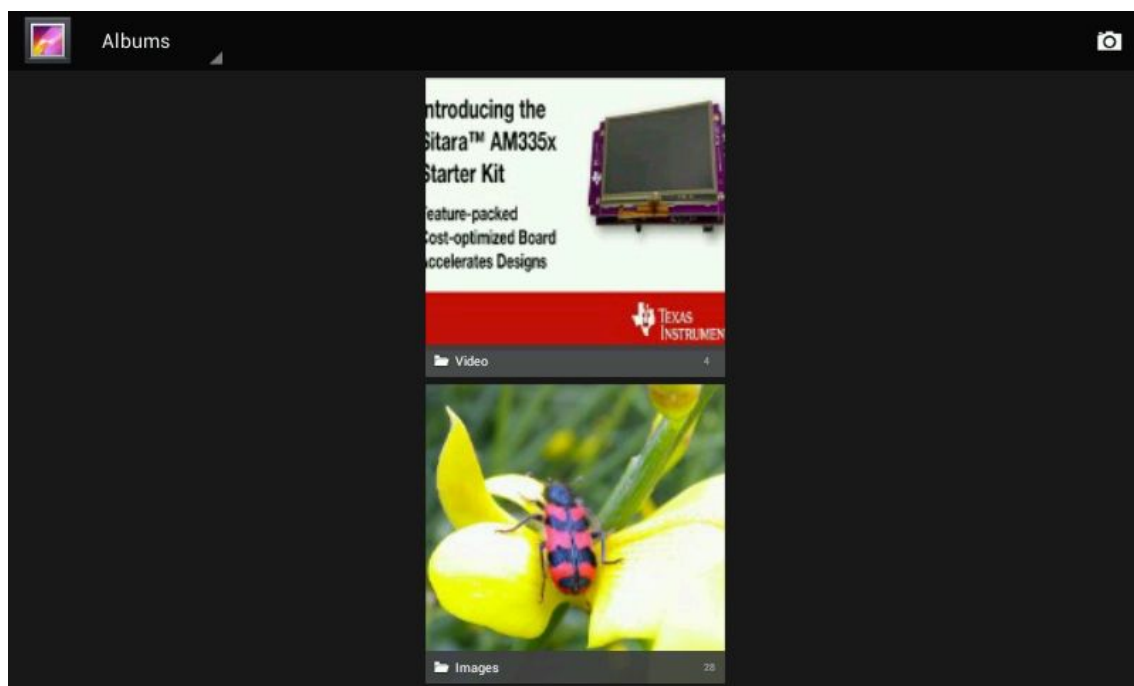
## 4.2 Android 应用程序

点击主界面上的“”图标，即可出现下面的界面：

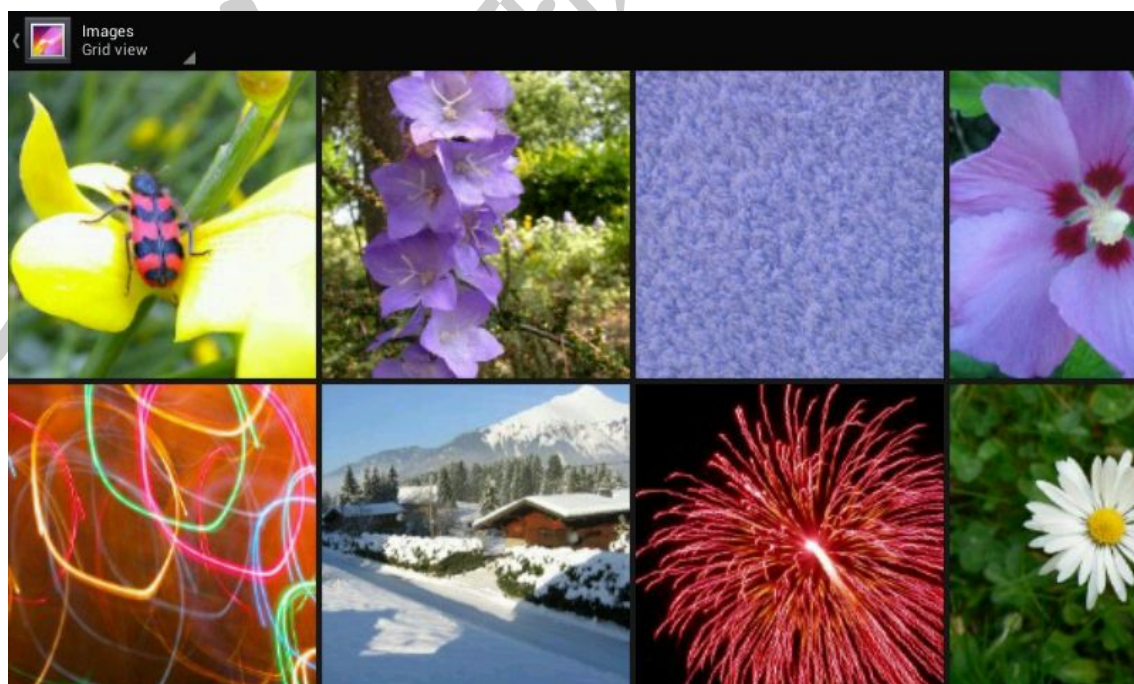


## 4.3 Android 查看图片



选择主界面上的“”图标 -> “图库”出现下图的界面：

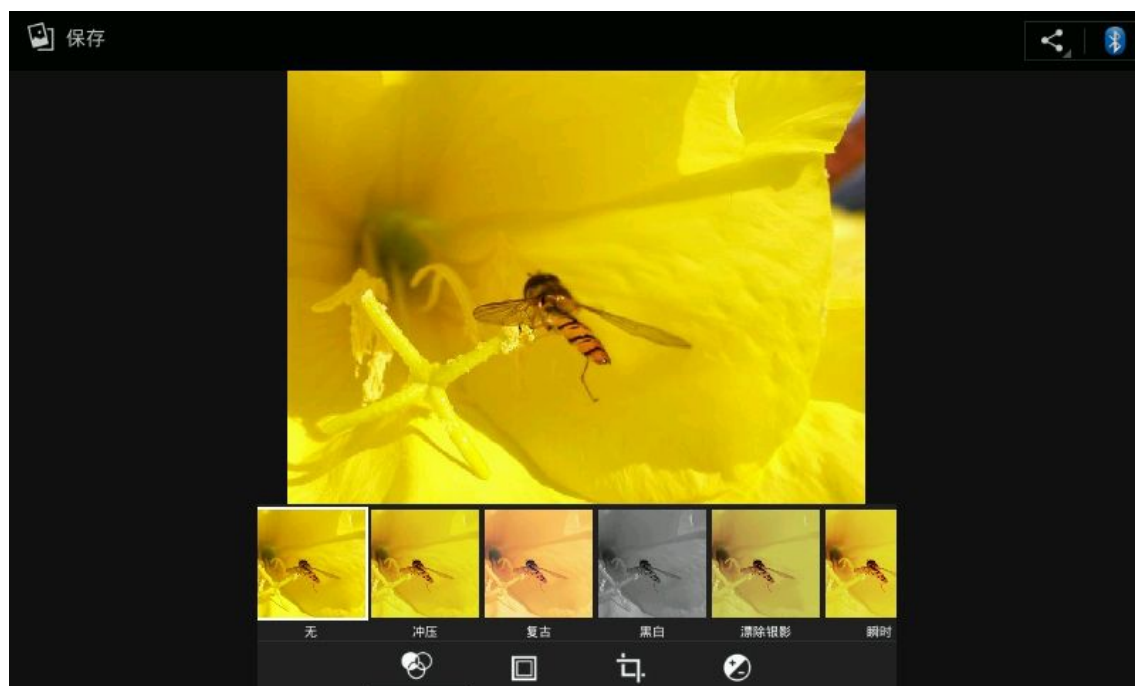


点击“Images”,查看系统图片预览：




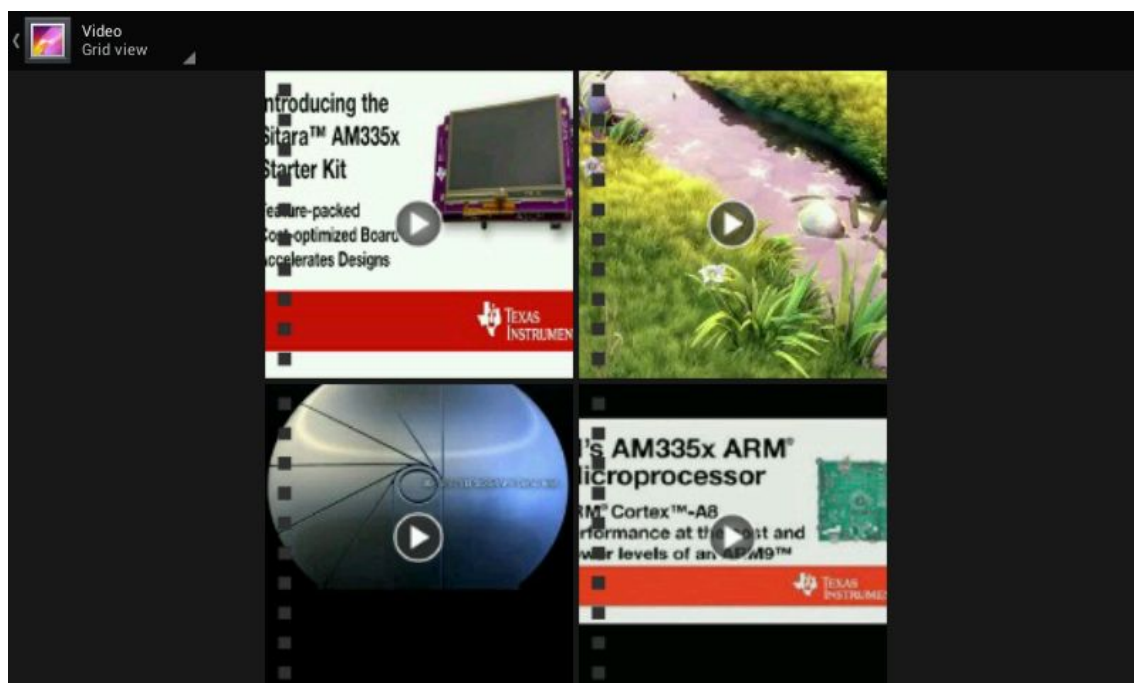
## 4.4 Android 编辑图片

选择 “” -> “图片” -> “Images” 打开图片预览之后，点击打开您要编辑的图片，之后点击 “” 按钮对图片进行编辑




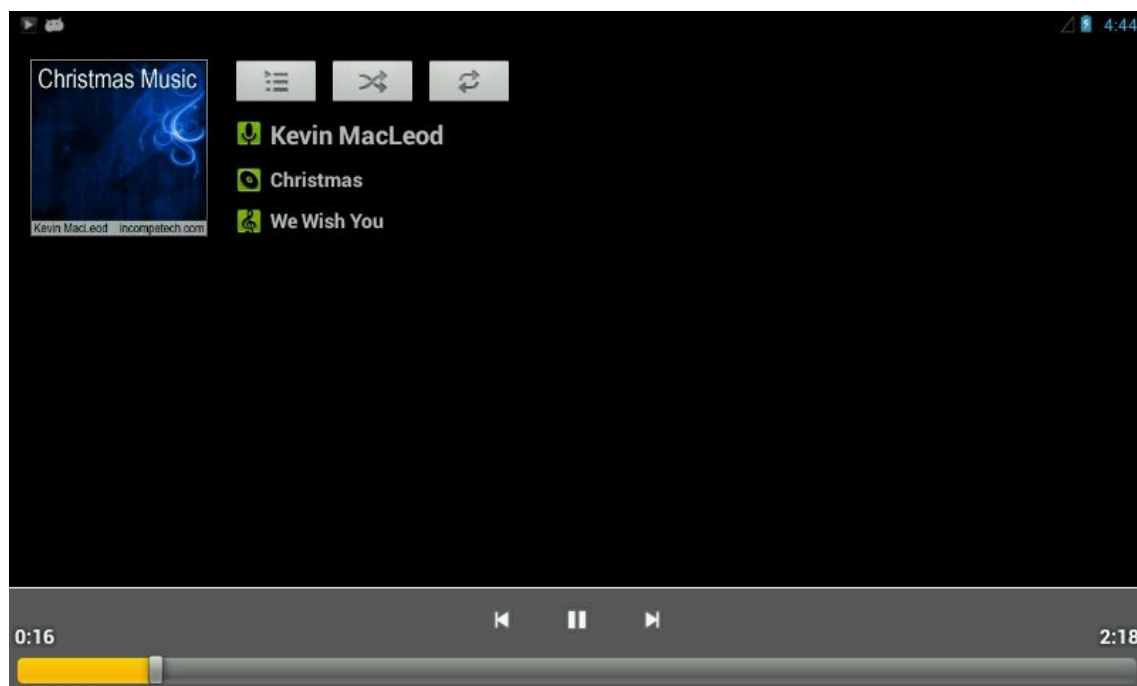
## 4.5 Android 播放视频

选择 “” -> “图库” -> “Video”，点击想要播放的录像的缩略图开始播放





## 4.6 Android 播放音乐


选择 “” -> “音乐” -> “艺术家”，然后选择想要播放的歌曲，点击播放：




## 4.7 Android 录音（支持 Line in 和 Mic 输入）

选择 “” -> “录音机” -> “”



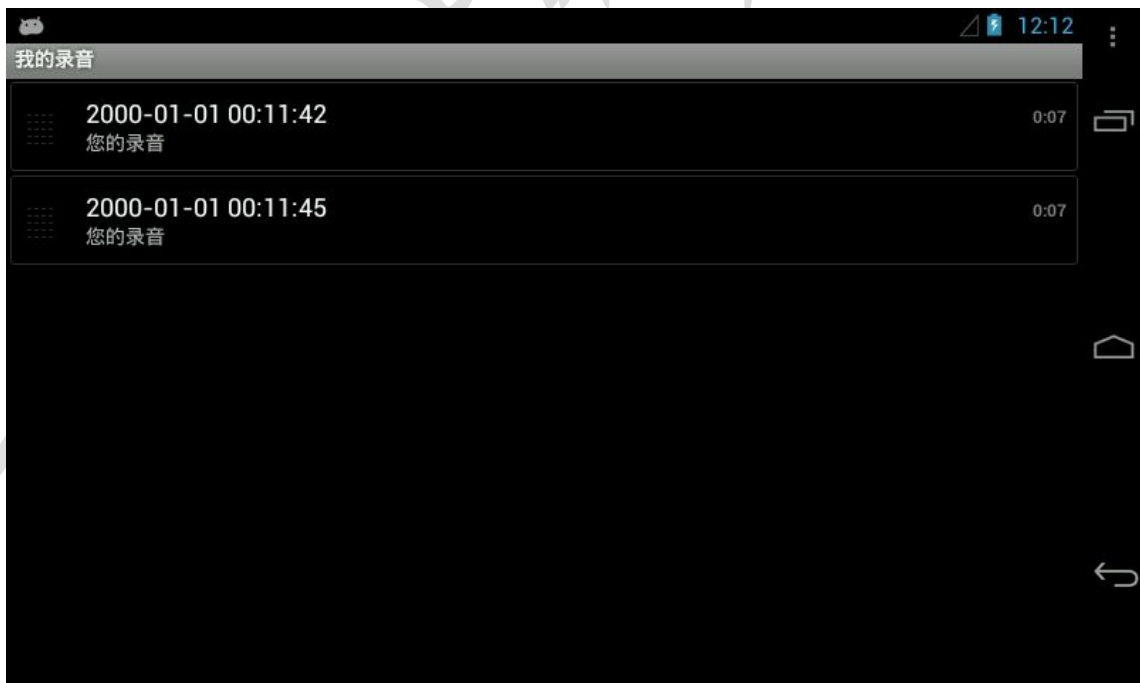
在录制完成之后按下 “” 选择完成



播放录音选择 “” -> “音乐” -> “播放列表” -> “我的录音”




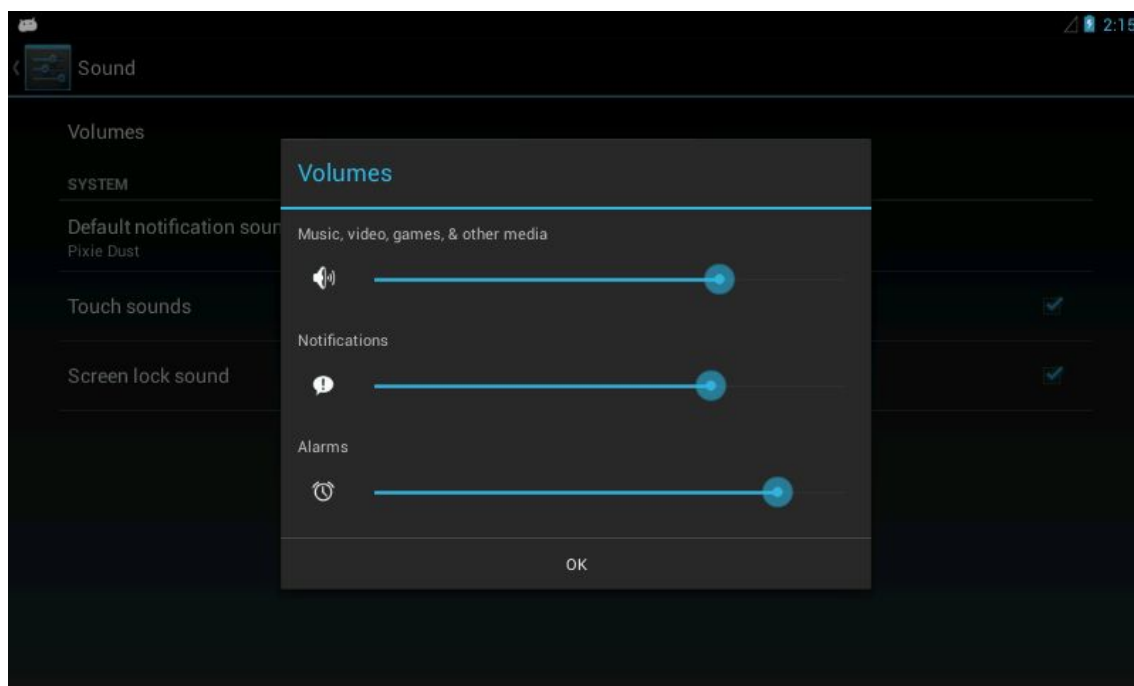
选择您刚才录制之后所保存的文件，点击播放




## 4.8 Android 调节音量

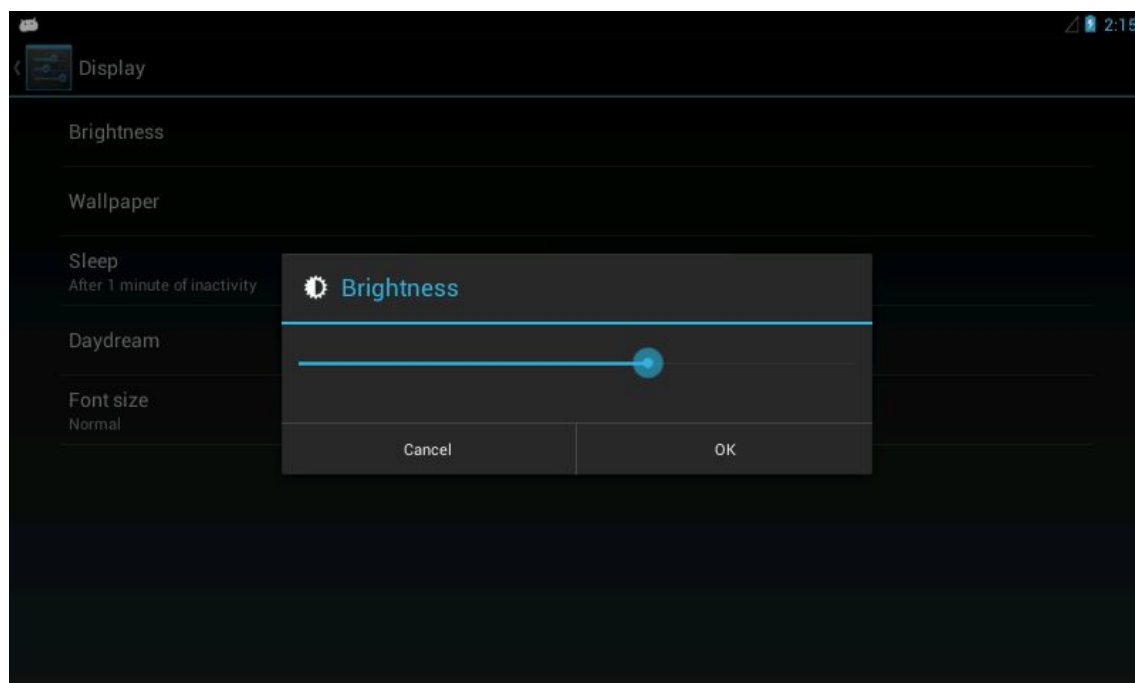


选择 “” -> “设置” -> “声音” -> “音量”，在下图中可以看到，在音量中有三个设置选项，从上到下分别是“音乐、视频、游戏及其他媒体”的音量调节，“通知”的音量调节和“闹铃”的音量调节。用户可根据自己的实际情况进行音量大小的调节。




## 4.9 Android 背光控制

选择 “” -> “设置” -> “显示” -> “亮度”，滑动界面上的圆圈来调节显示的亮度。调节范围为 19%-100%。



## 4.10 Android 设置时间（外部 RTC）



选择“”->“设置”->“日期和时间”，在这里可以更改日期和时间，并且在您断电之后时间仍可同步更新（确保板子上已经安装了纽扣电池供电）。




## 4.11 Android 有线连接（支持千兆以太网）

1. 插入网线后，超级终端会做如下显示，同样用 `#netcfg` 命令查看此时的网络状态，这是可以看到动态获取的 IP。此时您可以使用系统自带的浏览器浏览网页。

```
root@android:/ # [10445.733459] PHY: 0:00 - Link is Up - 1000/Full
[10445.739562] ADDRCONF(NETDEV_CHANGE): eth0: link becomes ready

root@android:/ # netcfg
lo                UP                127.0.0.1/8      0x00000049 00:00:00:
00:00:00
sit0              DOWN              0.0.0.0/0        0x00000080 00:00:00:
00:00:00
eth0              UP                192.168.1.107/24 0x00001043 78:c5:e5:
d0:c1:20
root@android:/ # ping www.baidu.com
PING www.baidu.com (119.75.217.56) 56(84) bytes of data:
64 bytes from 119.75.217.56: icmp_seq=1 ttl=56 time=3.57 ms
64 bytes from 119.75.217.56: icmp_seq=2 ttl=56 time=3.75 ms
64 bytes from 119.75.217.56: icmp_seq=3 ttl=56 time=3.20 ms
64 bytes from 119.75.217.56: icmp_seq=4 ttl=56 time=3.60 ms
^C
--- www.baidu.com ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3004ms
rtt min/avg/max/mdev = 3.204/3.532/3.754/0.206 ms
root@android:/ # _
```

2. 如果需要代理上网，需进行配置 Ethernet proxy settings。配置的步骤为：选择 “” -> “设置” -> “Ethernet proxy settings”



我们 android2.3 下支持静态 IP 的设置，您可以依次在相应的输入框中输入您的 IP 地址、子网掩码、DNS 地址和网关。（例如：IP：192.168.1.2 子网掩码：255.255.255.0 DNS 地址：8.8.8.8 网关地址：192.168.1.1）

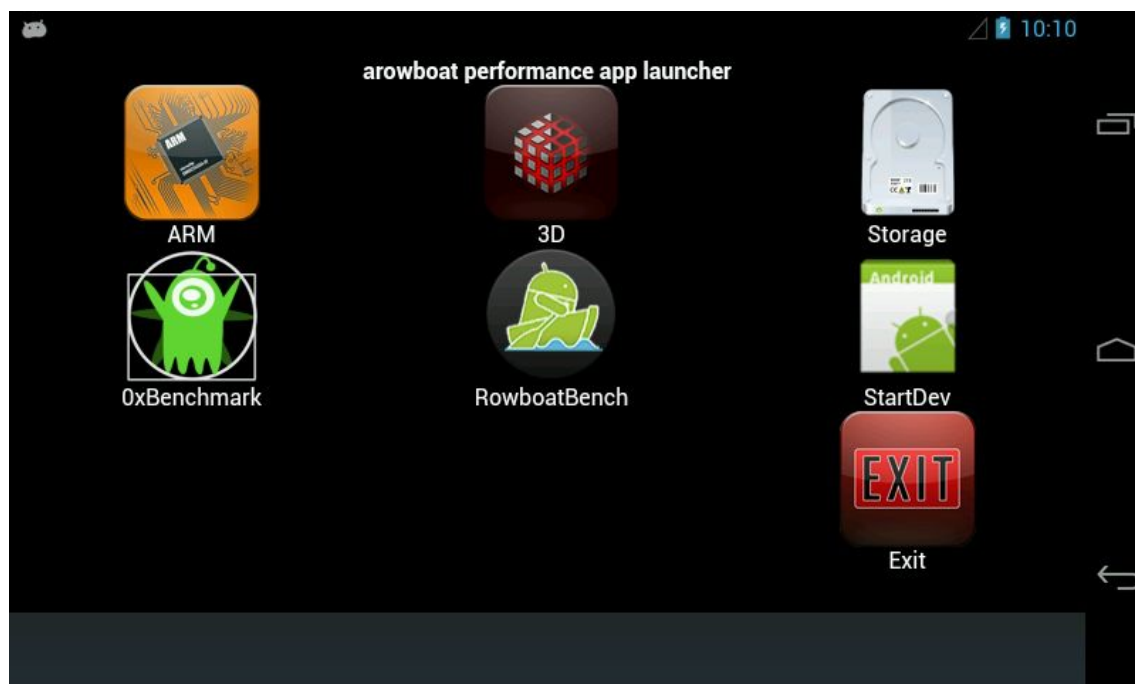
## 4.12 Android 按键

本开发板上有 K3 到 K8，6 个按键，我们在 android4.2 中配置了其中 3 个按键，分别为：K3—> 返回键、K4—> Home 键、K5—> 菜单键；android2.3 中配置了 6 个按键，分别为：K3—> 返回键、K4—> 菜单键、K5—> Home 键、K6—> 音量增大键，K7—> 音量减小键，K8—> 搜索键。

## 4.13 Android 2D、3D 测试

本测试为 Android4.2 下测试;

选择 “” -> “”



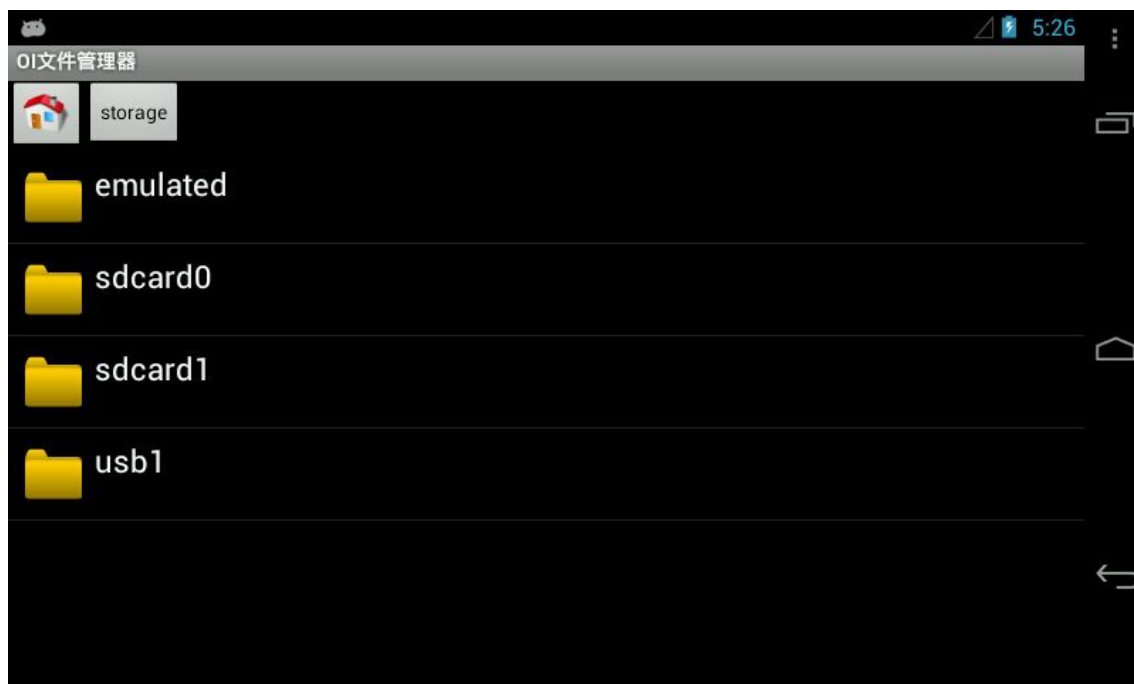
选择 “3D”，这下面有 4 个 3D 的样例，可以观察到 3D 的演示

选择 “RowboatBench” -> “2D Tests” -> 您可以观察到 2D 的演示

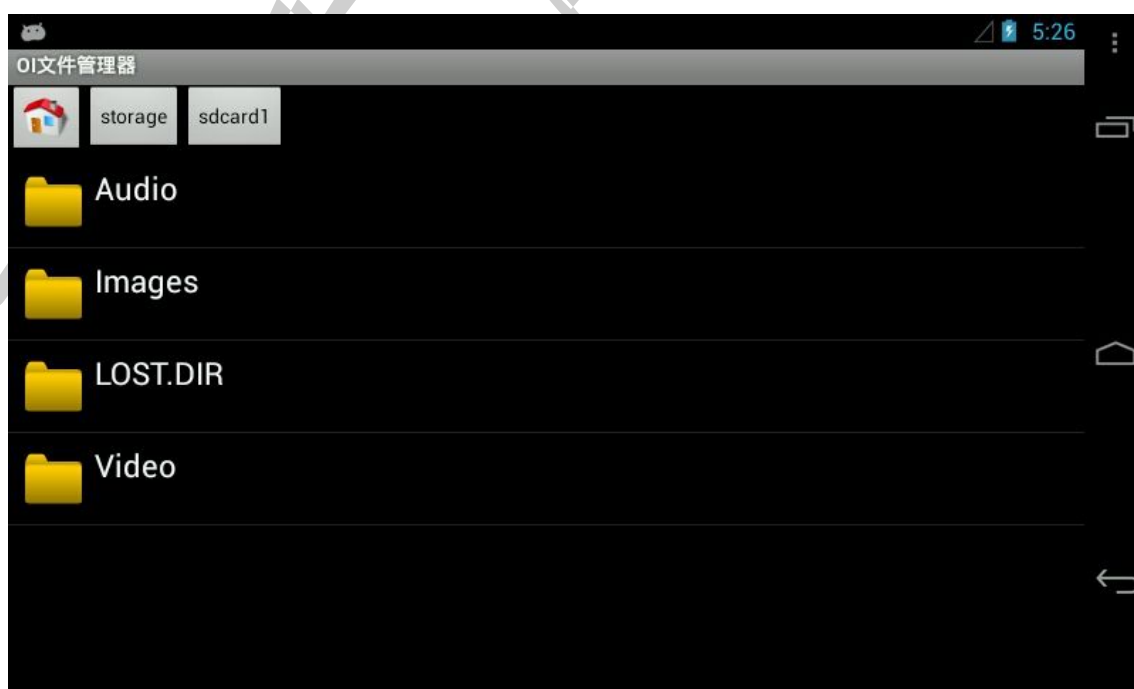
## 4.14 Android SD 卡测试

本测试为 android4.2 SD 卡的测试

选择 “” -> “”



选择 “sdcard1”



注意：我们 android2.3 下可以正常识别 fat32 的 SD 卡。

目前 android4.2 可以识别的 SD 为 ext4 格式的 SD 卡，具体的制作 SD 的方法如下：

1. 将准备好的 SD 卡（4GB 以上）正确的插入 SD 读卡器中，并与电脑相联；
2. 在终端中输入以下几个命令：

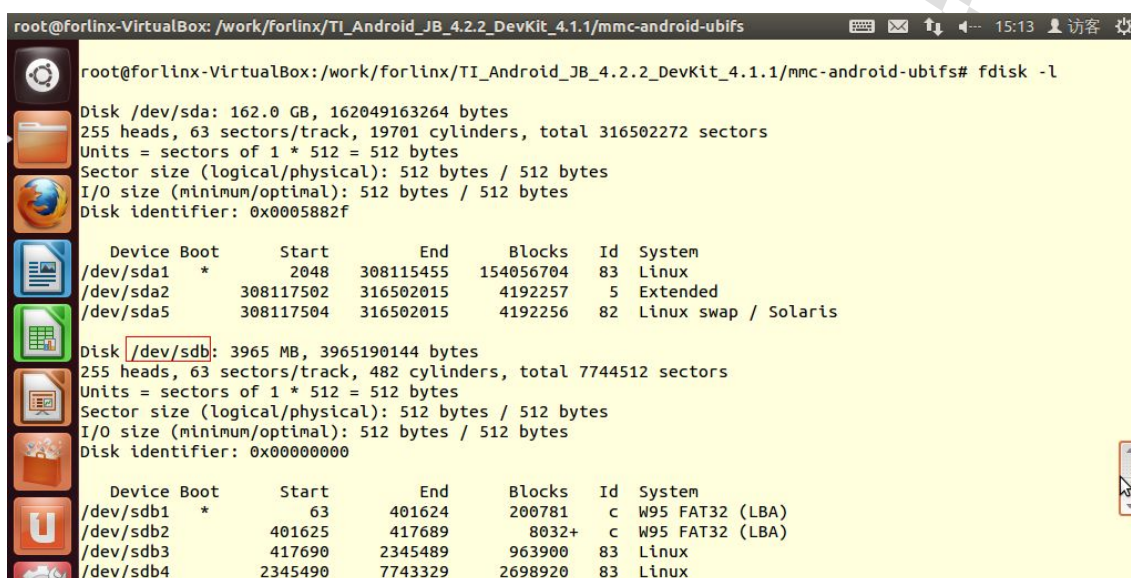
\$sudo fdisk -l （找到您所插入 SD 卡的挂载点）

\$cd /work/forlinx/TI\_Android\_JB\_4.2.2\_DevKit\_4.1.1/mmc-android-ubifs

\$sudo ./mkmmc-android-ubifs.sh /dev/<sd card mount-point>

注：<sd card mount-point>表示 SD 卡挂载点

下图为操作样图，找到所插入 SD 卡的挂载点为/dev/sdb



```

root@forlinx-VirtualBox: /work/forlinx/TI_Android_JB_4.2.2_DevKit_4.1.1/mmc-android-ubifs# fdisk -l

Disk /dev/sda: 162.0 GB, 162049163264 bytes
255 heads, 63 sectors/track, 19701 cylinders, total 316502272 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x0005882f

   Device Boot      Start         End      Blocks   Id  System
/dev/sda1  *           2048       308115455   154056704    83   Linux
/dev/sda2             308117502   316502015    4192257      5   Extended
/dev/sda5             308117504   316502015    4192256    82   Linux swap / Solaris

Disk /dev/sdb: 3965 MB, 3965190144 bytes
255 heads, 63 sectors/track, 482 cylinders, total 7744512 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x00000000

   Device Boot      Start         End      Blocks   Id  System
/dev/sdb1  *              63        401624       200781    c   W95 FAT32 (LBA)
/dev/sdb2             401625        417689         8032+    c   W95 FAT32 (LBA)
/dev/sdb3             417690       2345489       963900    83   Linux
/dev/sdb4             2345490       7743329      2698920    83   Linux
    
```

例：下图为制作 SD 卡的过程，输入命令：

\$sudo ./mkmmc-android-ubifs.sh /dev/sdb

```

root@forlinx-VirtualBox: /work/forlinx/TI_Android_JB_4.2.2_DevKit_4.1.1/mmc-android-ubifs
root@forlinx-VirtualBox:/work/forlinx/TI_Android_JB_4.2.2_DevKit_4.1.1/mmc-android-ubifs# ./mkmmc-android-ubifs.sh /dev/sdb
Assuming Default Locations for Prebuilt Images
All data on /dev/sdb now will be destroyed! Continue? [y/n]
y
[Unmounting all existing partitions on the device ]
umount: /dev/sdb 未挂载
[Partitioning /dev/sdb...]
记录了1024+0 的读入
记录了1024+0 的写出
1048576字节(1.0 MB)已复制, 2.87513 秒, 365 kB/秒
Disk /dev/sdb doesn't contain a valid partition table
DISK SIZE - 3965190144 bytes
CYLINDERS - 482
Checking that no-one is using this disk right now ...
OK
Disk /dev/sdb: 482 cylinders, 255 heads, 63 sectors/track
sfdisk: ERROR: sector 0 does not have an msdos signature
/dev/sdb: unrecognized partition table type
Old situation:
No partitions found
New situation:
Units = cylinders of 8225280 bytes, blocks of 1024 bytes, counting from 0


   Device Boot  Start      End  #cyls   #blocks   Id System
/dev/sdb1  *         0+        24      25-    200781    c  W95 FAT32 (LBA)
/dev/sdb2             25        25         1      8032+    c  W95 FAT32 (LBA)
/dev/sdb3             26       145       120     963900   83  Linux
/dev/sdb4           146       481       336    2698920   83  Linux
Successfully wrote the new partition table
Re-reading the partition table ...

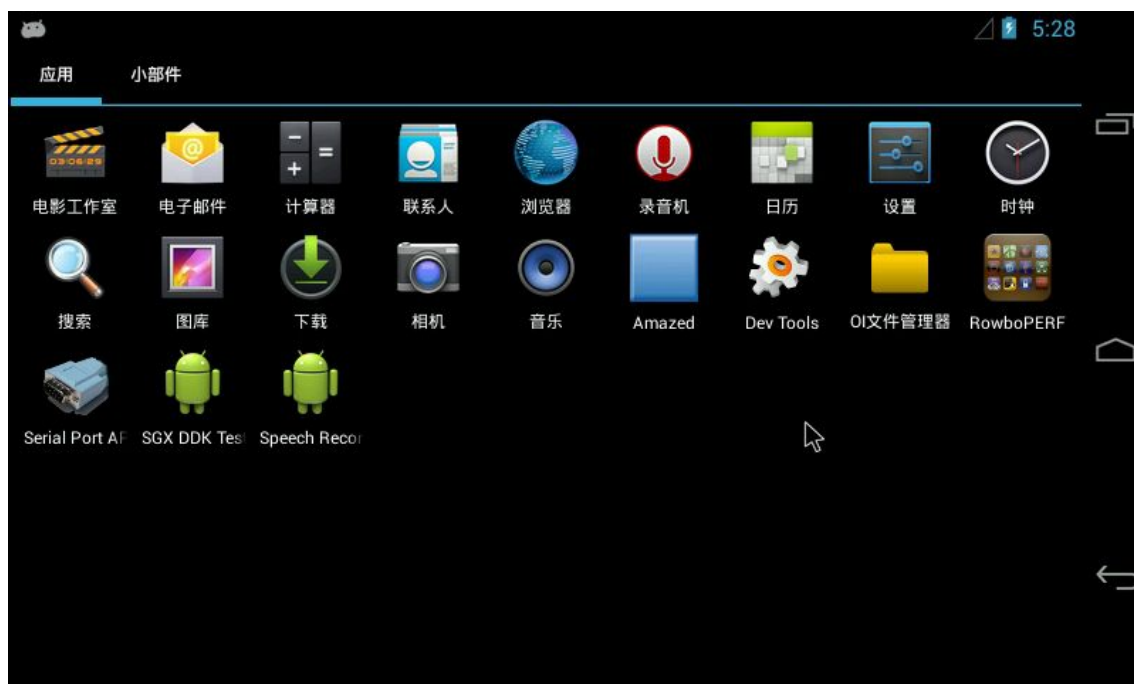
If you created or changed a DOS partition, /dev/foo7, say, then use dd(1)
to zero the first 512 bytes: dd if=/dev/zero of=/dev/foo7 bs=512 count=1
(See fdisk(8).)
[Making filesystems...]
[Copying files...]
[Copying boot files...]
[Copying ubifs image...]
[Copying start-here folder to boot partition...]
[syncing...]
[Copying all media clips to data partition...]
[Done]
root@forlinx-VirtualBox:/work/forlinx/TI_Android_JB_4.2.2_DevKit_4.1.1/mmc-android-ubifs#

```

把 SD 卡插入卡槽中，就可以把 SD 中第四个分区挂载到/storage/sdcard1 上

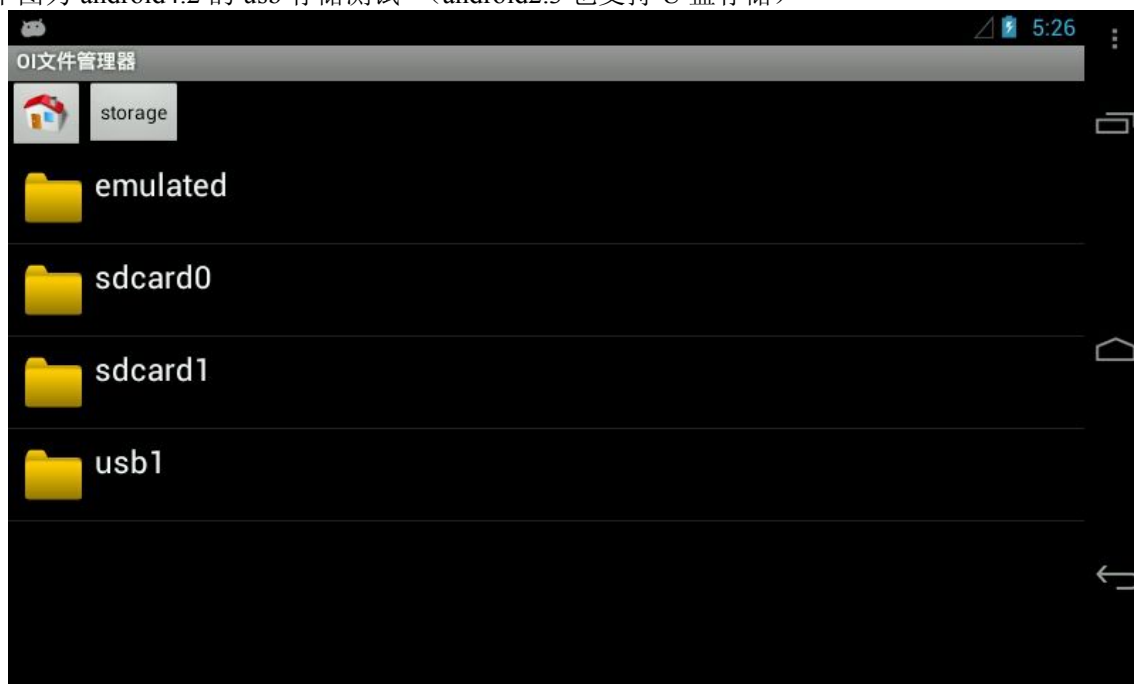
## 4.15 Android USB 鼠标测试

系统运行之后，在 USB host 上插入 USB 鼠标，您就会在界面内看到鼠标光标“”，您可以通过鼠标操作 Android 系统。



## 4.16 Android USB 存储测试

下图为 android4.2 的 usb 存储测试（android2.3 也支持 U 盘存储）

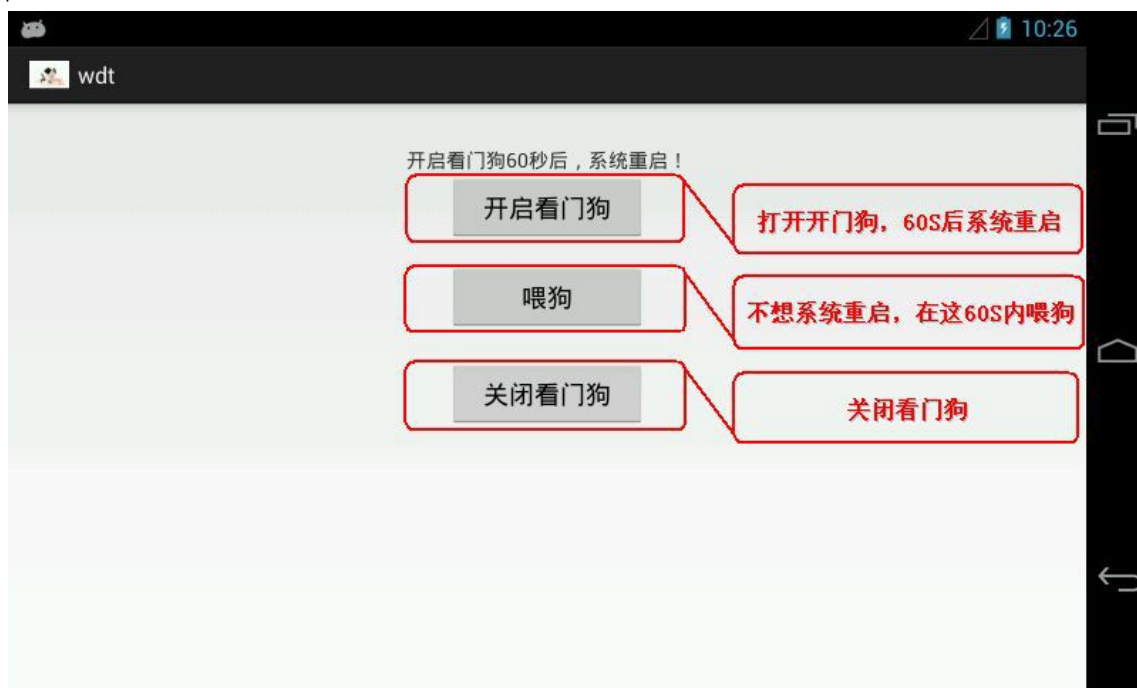


选择“usb1”您会下目录下看到您所插入的 usb 存储内的内容。



## 4.17 Android 看门狗测试

选择 “” -> “”



## 4.18 Android 串口测试

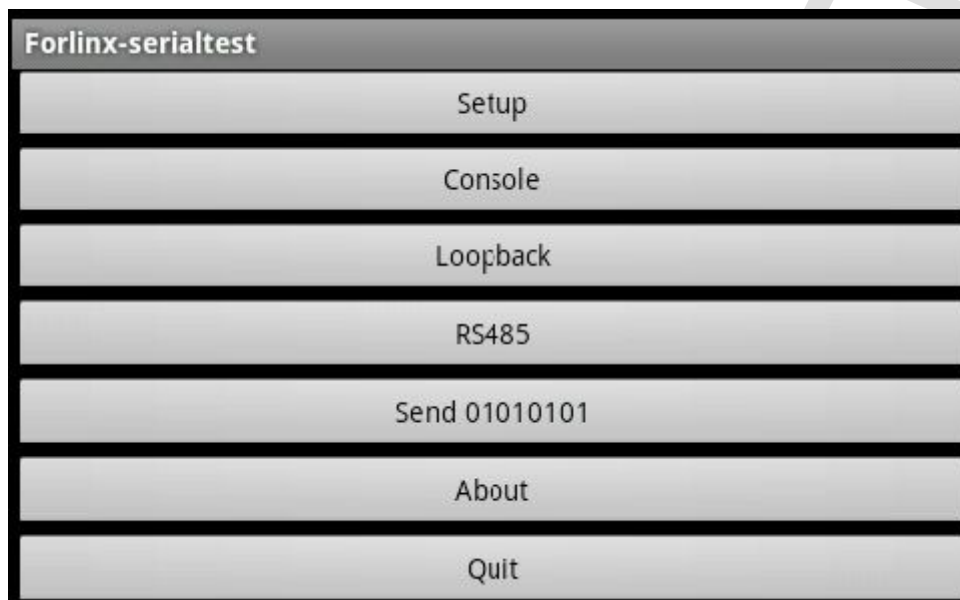
OK335xD 开发板中有三个串口，在开发板上的标识符分别为 COM0，COM1，UART。COM0 为调试口，不需要测试。

- 1) COM0 口，RS232 电平，用于调试口，设备名称： /dev/ttyO0;
- 2) COM1 口，RS232 电平，使用 DB9 公头，设备名称： /dev/ttyO1;
- 3) UART 口，TTL 电平，使用插针接口，设备名称： /dev/ttyO4;

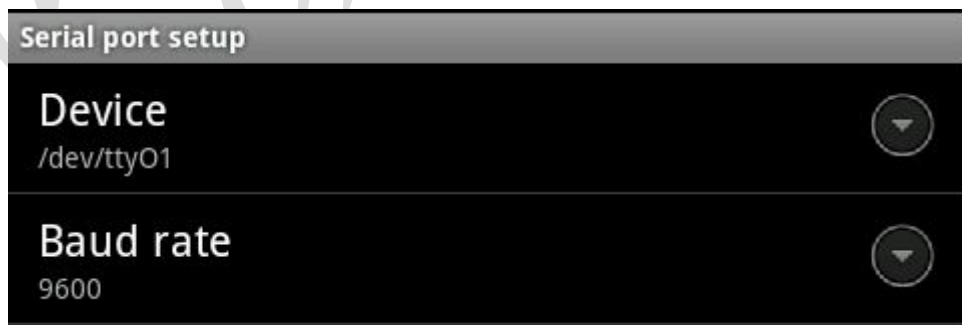
测试方法如下：

1. 关掉开发板电源，将相应串口的 RX 和 TX 管脚短接。

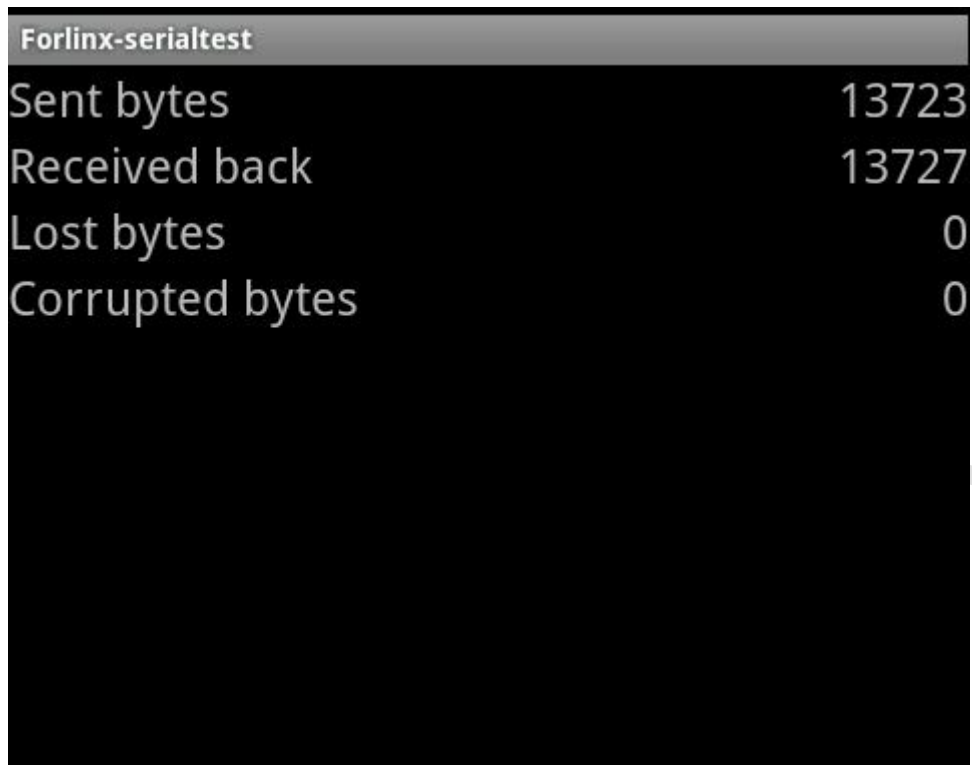
选择 “” -> “”



2. 选择“Setup”出现下图界面，设置您要测试的串口。下图为测试 ttyO1



3. 选择“Loopback”，把串口的接收和发送连接到一起。





The screenshot shows a black terminal window with white text. The title bar reads 'Forlinx-serialtest'. The content displays the following statistics:

Statistic	Value
Sent bytes	13723
Received back	13727
Lost bytes	0
Corrupted bytes	0

4. 在“Setup”中“Display format”下设置为 ASCII 显示（char）和十六进制（hex），不选择的话为 ASCII 显示。

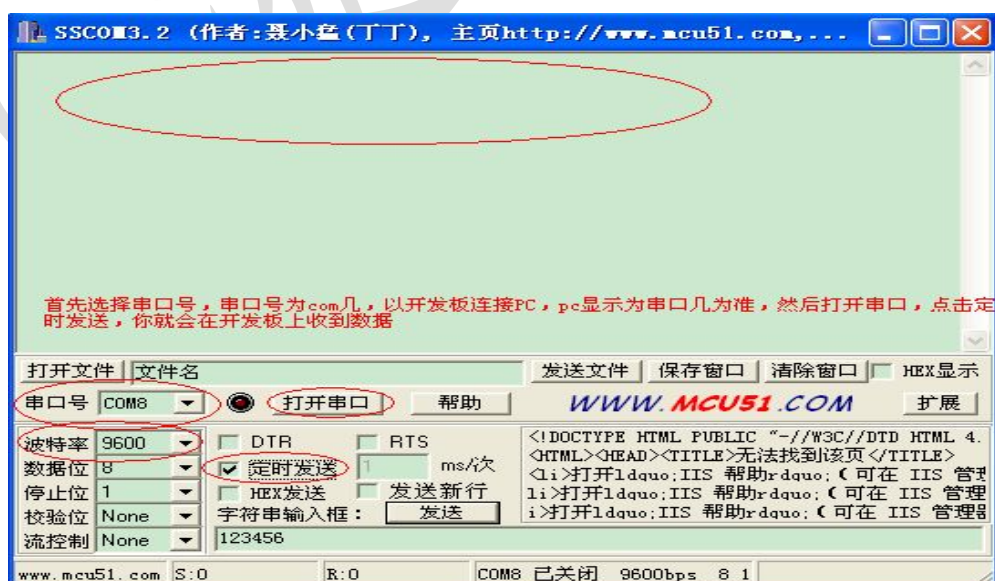
## 4.19 Android RS485 测试

将一个 RS232 转 485 模块和我们的开发板连接，如果是使用我们提供的转换模块，连接方式是 A+（转换模块）接 A（开发板）、B+（转换模块）接 B（开发板）。

选择 “” -> “”，RS485 对应的设备号为 ttyO2，在 “Setup” 中选择 ttyO2(OMAP-SERIAL)， “Baud rate” 下设置波特率， “Display format” 下设置为 ASCII 显示（char）和十六进制（hex），不选择的话为 ASCII 显示，点击 “RS485”，首先进入收数据模式



打开 pc 端的串口测试程序 “”，



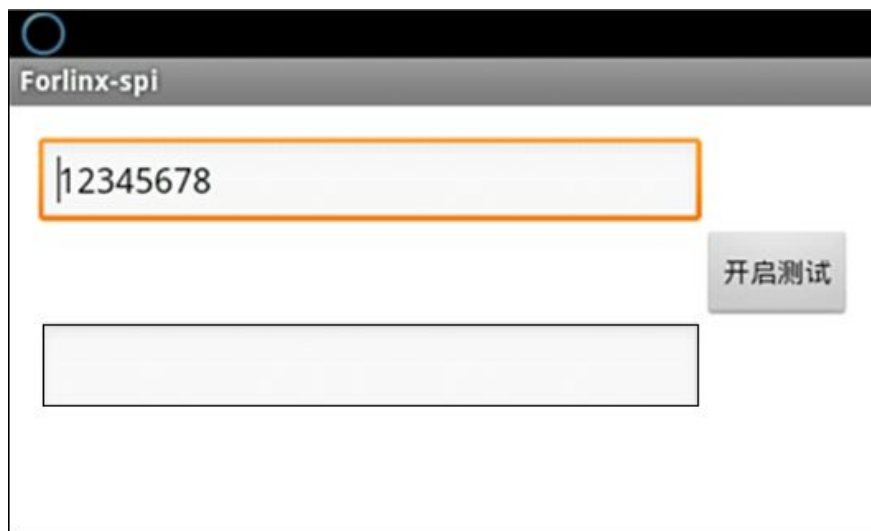
点击“Rs485 rec”,485 切换为发送模式



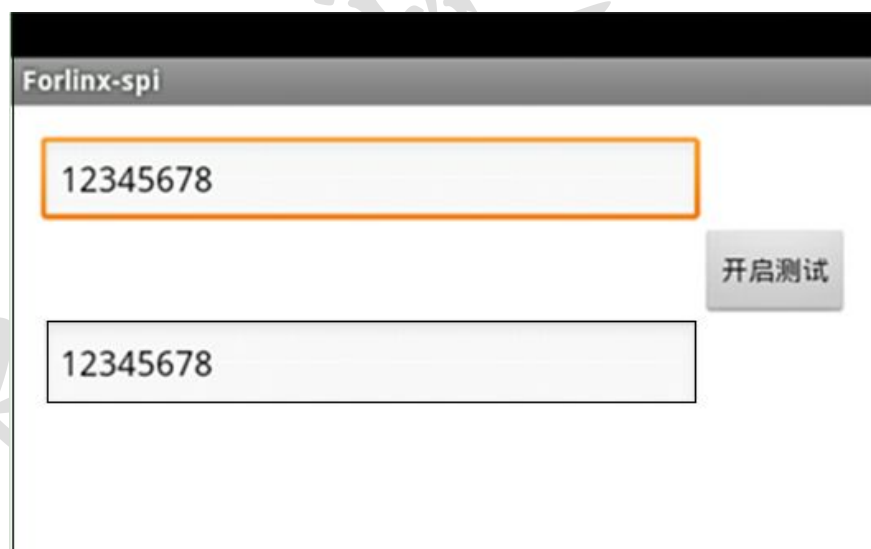
## 4.20 Android SPI 测试

OK335xD 开发板提供了一路 TTL 电平的插针 SPI 接口,将原理图中的 SPI1-D0 和 SPI1\_D1 引脚短接,查看原理图找到开发板上对应的引脚。

选择 “” -> “”



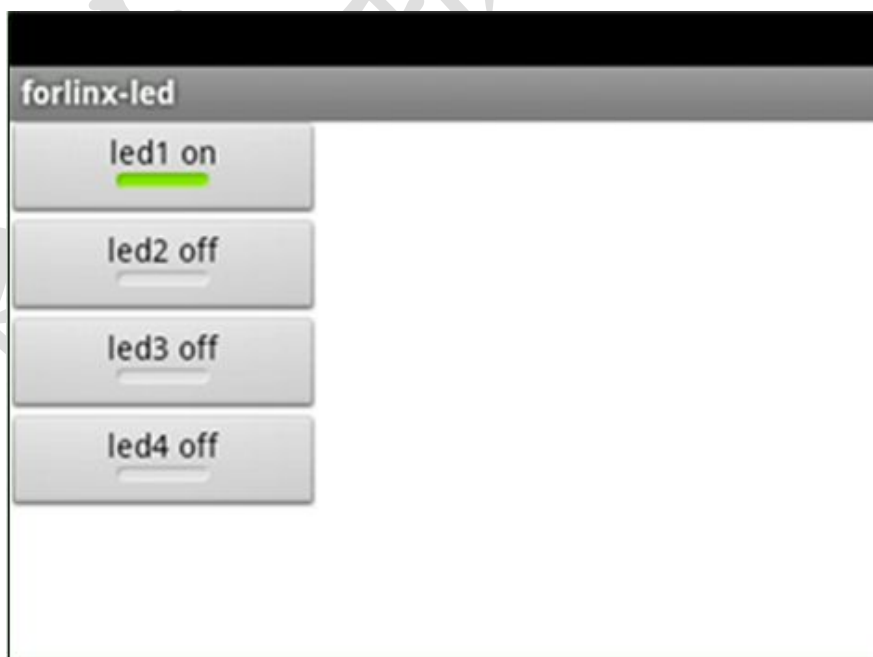
短接之后点击开启测试



## 4.21 Android LED 测试

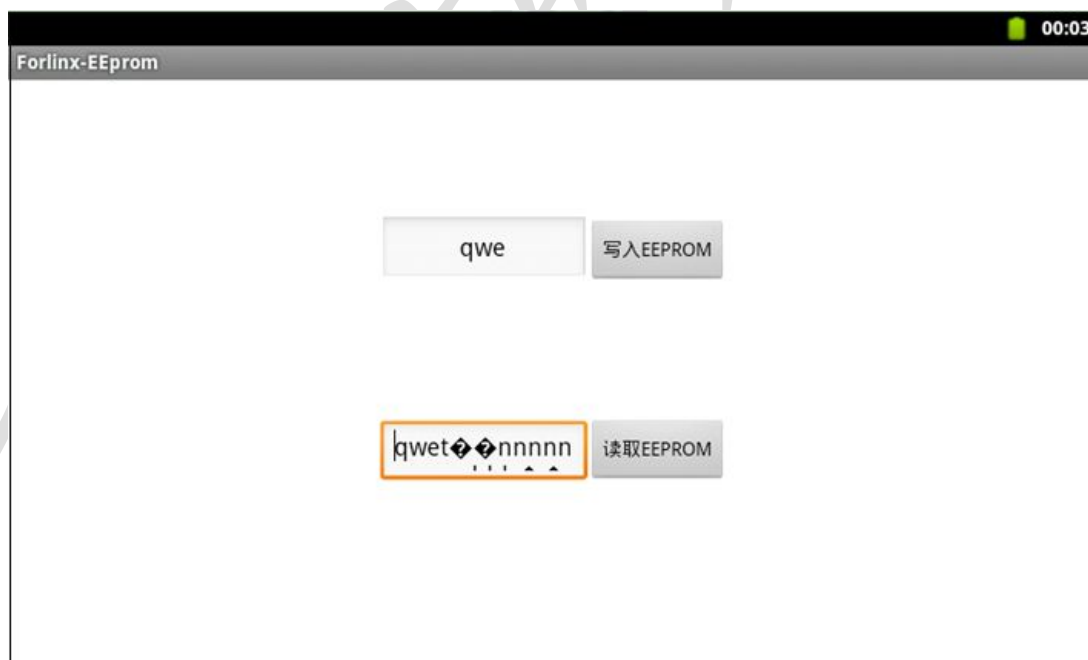
OK335xD 底板上的 LED4、LED5、LED6、LED7 为用户 LED 灯

选择 “” -> “”



## 4.22 Android EEPROM 测试

选择 “” -> “”



## 第五章 Android 应用程序开发

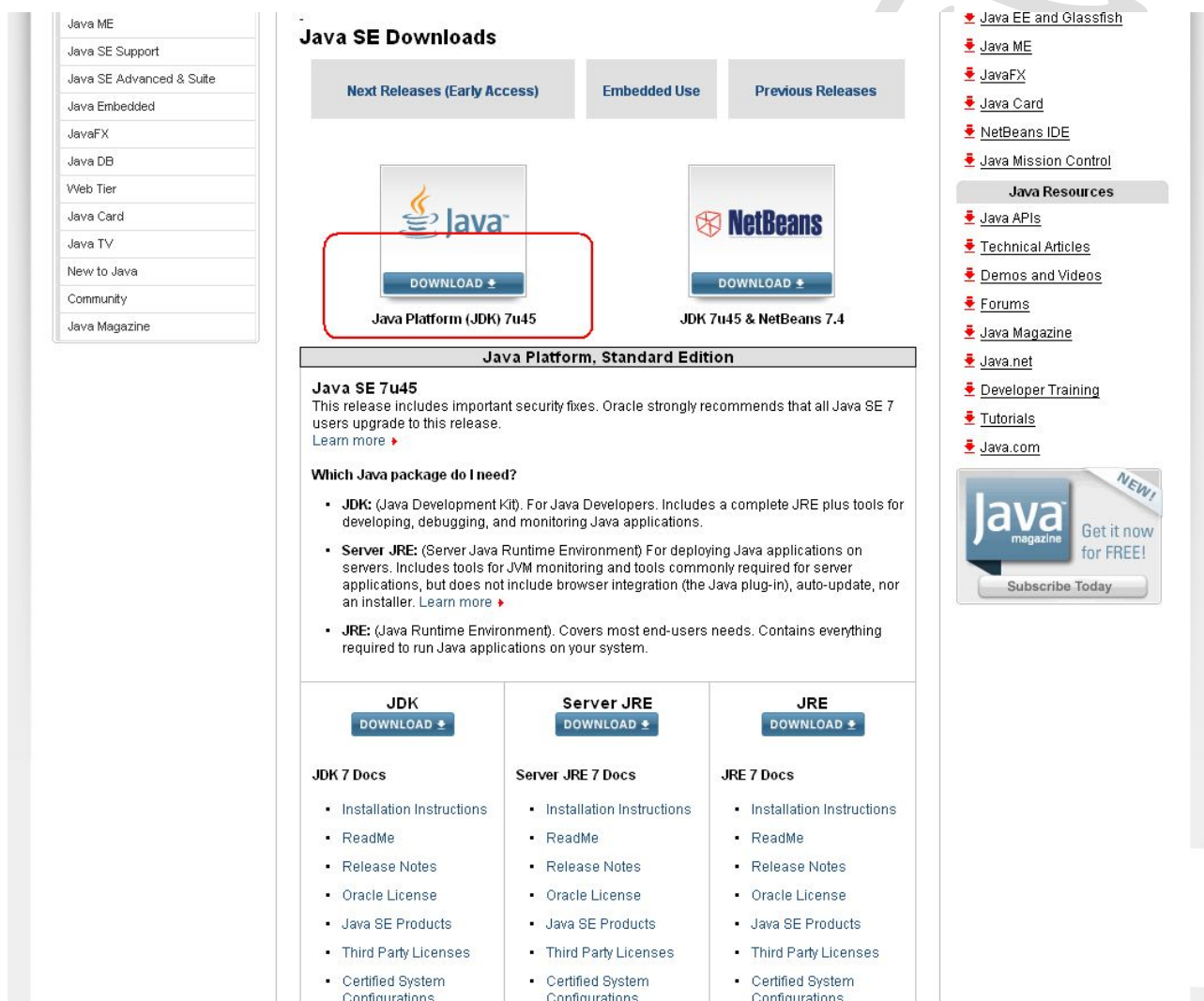
本章节讲解如何建立 Android 开发环境，包括 Android SDK 和 Eclipse 集成开发环境的下载及安装，以及如何使用 OK335xD 开发板作为真机调试程序，非常适合 Android 初学者学习和参考。

### 5.1 建立 Android 应用开发环境

#### 5.1.1 下载并安装 JDK (Java SE Development Kit)

由于 Android SDK 和 Eclipse 都是用 Java 编写的，因此需要先在 Windows 上安装 JDK，JDK 可按以下方法下载：

访问网站 <http://www.oracle.com/technetwork/java/javase/downloads/index.html>，在页面中点击 JDK



**Java SE Downloads**

Next Releases (Early Access) Embedded Use Previous Releases

**Java Platform (JDK) 7u45**

**JDK 7u45 & NetBeans 7.4**

**Java Platform, Standard Edition**

**Java SE 7u45**  
This release includes important security fixes. Oracle strongly recommends that all Java SE 7 users upgrade to this release.  
[Learn more](#)

**Which Java package do I need?**

- JDK:** (Java Development Kit). For Java Developers. Includes a complete JRE plus tools for developing, debugging, and monitoring Java applications.
- Server JRE:** (Server Java Runtime Environment) For deploying Java applications on servers. Includes tools for JVM monitoring and tools commonly required for server applications, but does not include browser integration (the Java plug-in), auto-update, nor an installer. [Learn more](#)
- JRE:** (Java Runtime Environment). Covers most end-users needs. Contains everything required to run Java applications on your system.

**JDK** **Server JRE** **JRE**

**JDK 7 Docs**

- Installation Instructions
- ReadMe
- Release Notes
- Oracle License
- Java SE Products
- Third Party Licenses
- Certified System Configurations

**Server JRE 7 Docs**

- Installation Instructions
- ReadMe
- Release Notes
- Oracle License
- Java SE Products
- Third Party Licenses
- Certified System Configurations

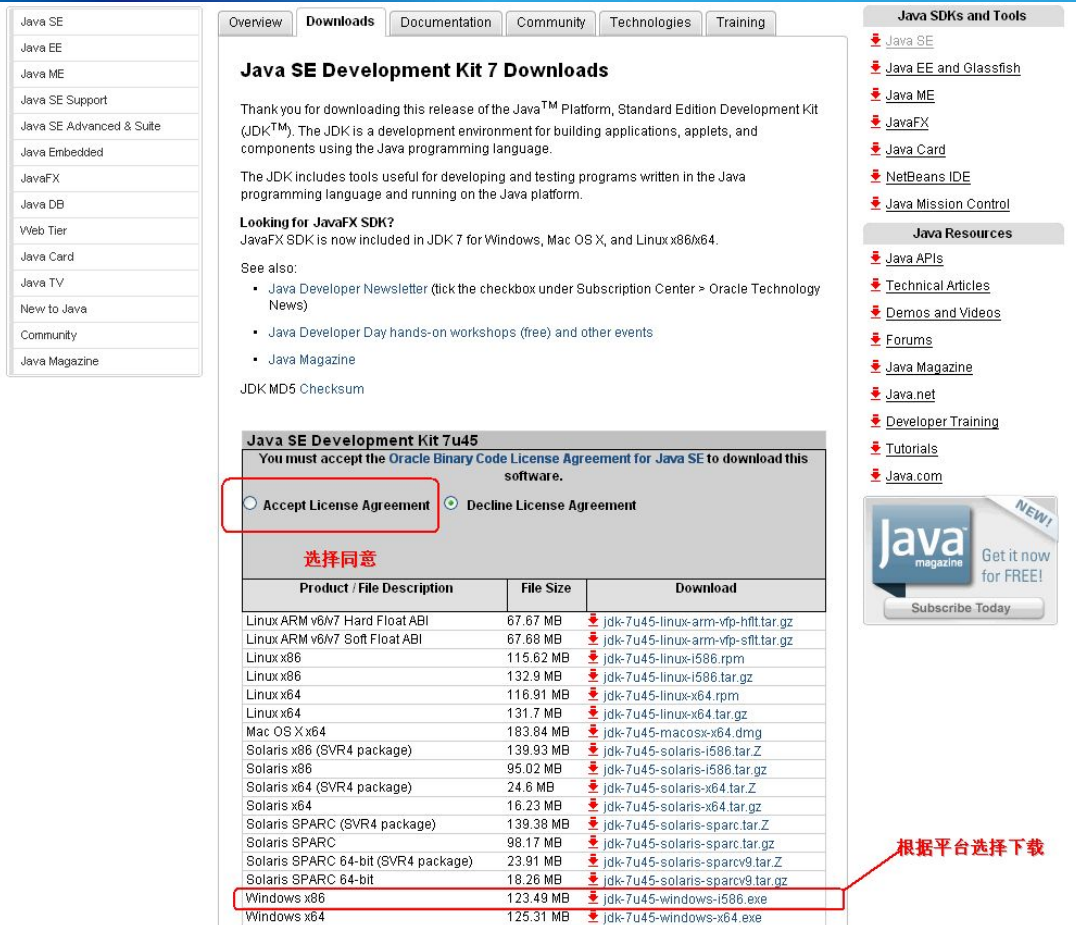
**JRE 7 Docs**

- Installation Instructions
- ReadMe
- Release Notes
- Oracle License
- Java SE Products
- Third Party Licenses
- Certified System Configurations

**Java Resources**

- Java APIs
- Technical Articles
- Demos and Videos
- Forums
- Java Magazine
- Java.net
- Developer Training
- Tutorials
- Java.com

**Subscribe Today**



**Java SE Development Kit 7 Downloads**

Thank you for downloading this release of the Java™ Platform, Standard Edition Development Kit (JDK™). The JDK is a development environment for building applications, applets, and components using the Java programming language.

The JDK includes tools useful for developing and testing programs written in the Java programming language and running on the Java platform.

**Looking for JavaFX SDK?**  
JavaFX SDK is now included in JDK 7 for Windows, Mac OS X, and Linux x86/x64.

See also:

- Java Developer Newsletter (tick the checkbox under Subscription Center > Oracle Technology News)
- Java Developer Day hands-on workshops (free) and other events
- Java Magazine

JDK MD5 Checksum

**Java SE Development Kit 7u45**  
You must accept the Oracle Binary Code License Agreement for Java SE to download this software.

☐ Accept License Agreement ☒ Decline License Agreement

**选择同意**

Product / File Description	File Size	Download
Linux ARM v6M7 Hard Float ABI	67.67 MB	jdk-7u45-linux-arm-vfp-hflt.tar.gz
Linux ARM v6M7 Soft Float ABI	67.68 MB	jdk-7u45-linux-arm-vfp-sflt.tar.gz
Linux x86	115.62 MB	jdk-7u45-linux-i586.rpm
Linux x86	132.9 MB	jdk-7u45-linux-i586.tar.gz
Linux x64	116.91 MB	jdk-7u45-linux-x64.rpm
Linux x64	131.7 MB	jdk-7u45-linux-x64.tar.gz
Mac OS X x64	183.84 MB	jdk-7u45-macosx-x64.dmg
Solaris x86 (SVR4 package)	139.93 MB	jdk-7u45-solaris-i586.tar.gz
Solaris x86	95.02 MB	jdk-7u45-solaris-i586.tar.gz
Solaris x64 (SVR4 package)	24.6 MB	jdk-7u45-solaris-x64.tar.gz
Solaris x64	16.23 MB	jdk-7u45-solaris-x64.tar.gz
Solaris SPARC (SVR4 package)	139.38 MB	jdk-7u45-solaris-sparc.tar.Z
Solaris SPARC	98.17 MB	jdk-7u45-solaris-sparc.tar.gz
Solaris SPARC 64-bit (SVR4 package)	23.91 MB	jdk-7u45-solaris-sparcv9.tar.Z
Solaris SPARC 64-bit	18.26 MB	jdk-7u45-solaris-sparcv9.tar.gz
Windows x86	123.49 MB	jdk-7u45-windows-i586.exe
Windows x64	125.31 MB	jdk-7u45-windows-x64.exe

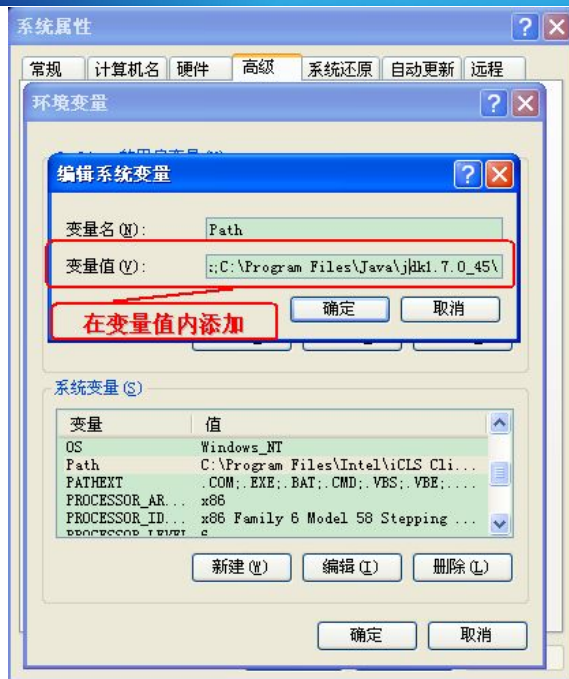
**根据平台选择下载**

下载完成后，双击安装程序，根据向导的提示完成安装即可。

安装完成后，需要将 JDK 命令添加到 Path 环境变量中，通过下面的方法将 JDK 命令所在的路径添加到 Path 环境变量中：

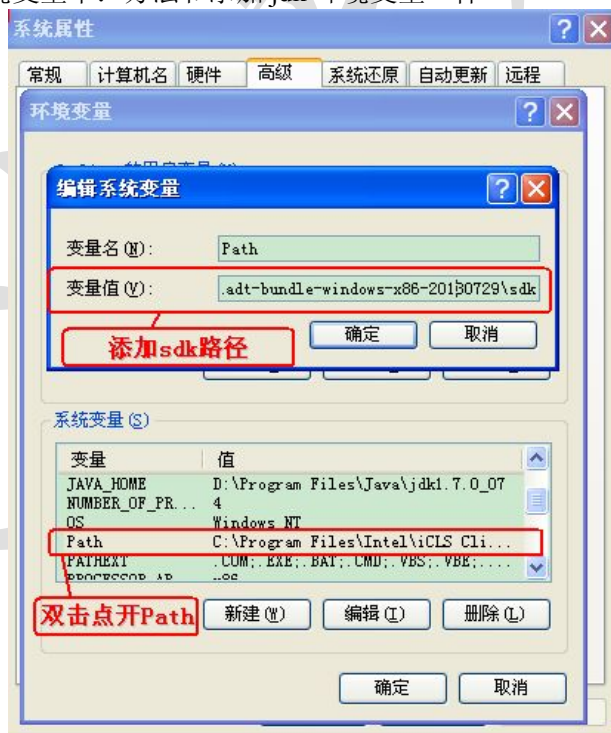
- 1) 右击“我的电脑” -> 属性，再选择左边导航的“高级系统设置”选项。
- 2) 点击右下角的“环境变量”选项。
- 3) 在“系统变量”中，找到 Path 环境变量，双击它，在变量值前面追加以下内容：“C:\Program Files\Java\jdk1.7.0\_45\”。
- 4) 点击“确定”完成环境变量设置。



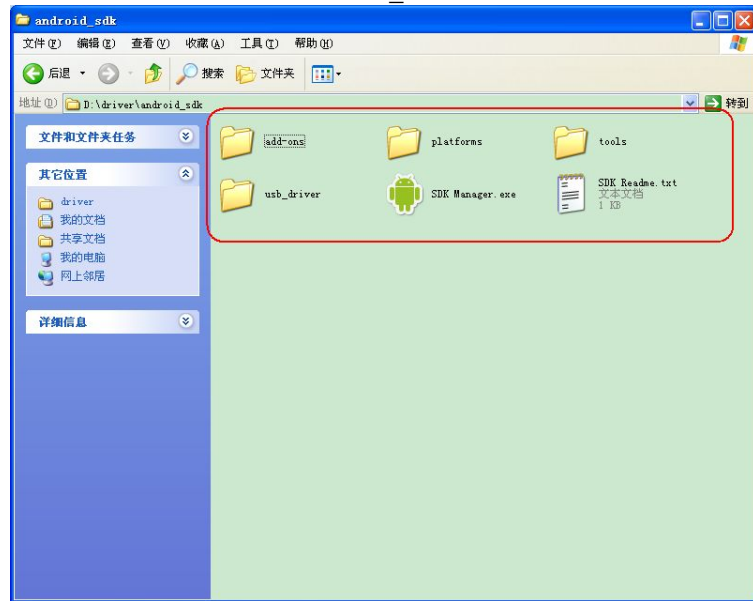


### 5.1.2 安装 adt-bundle-windows

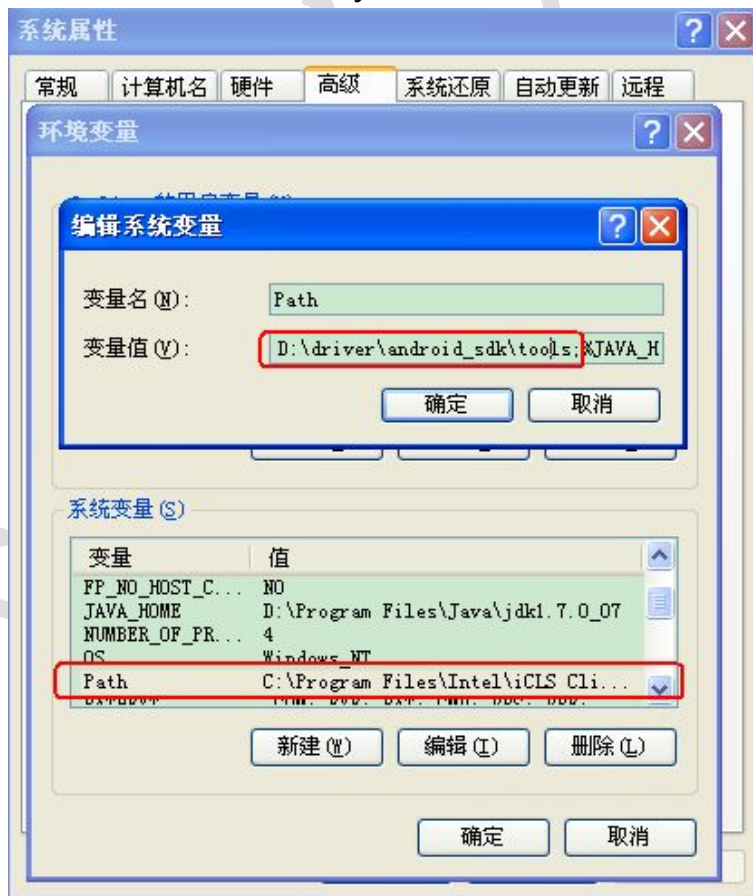
拷贝光盘下的 Windows 的 android 开发工具 adt-bundle-windows-x86-20130729.zip。解压到您的电脑上，在 adt-bundle-windows-x86-20130729 下会看到 eclipse、sdk 和 SDK Manager.exe。需要把 sdk 添加到 Path 环境变量中：方法和添加 jdk 环境变量一样



拷贝光盘下的 android\_sdk.rar 到电脑上，解压在 android\_sdk 下您会看到

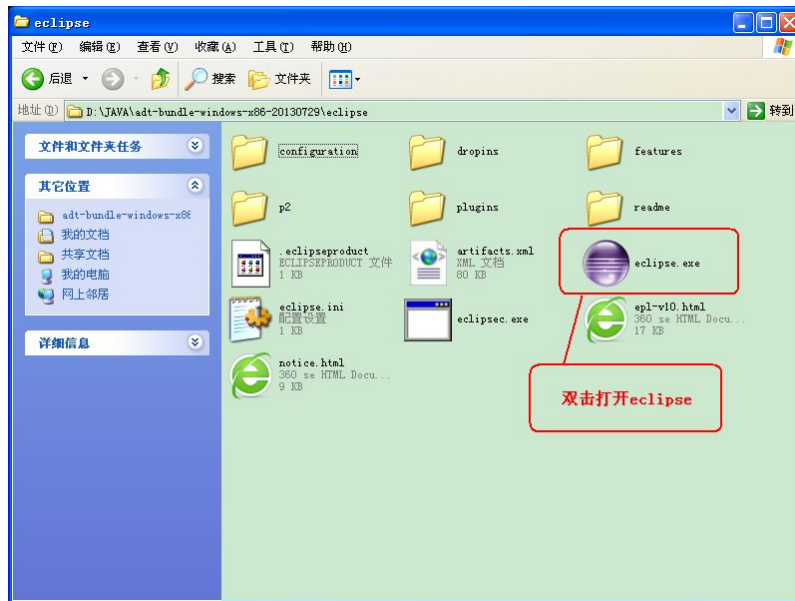


把 android\_sdk 添加到 Path 环境变量内：方法和添加 jdk 环境变量一样

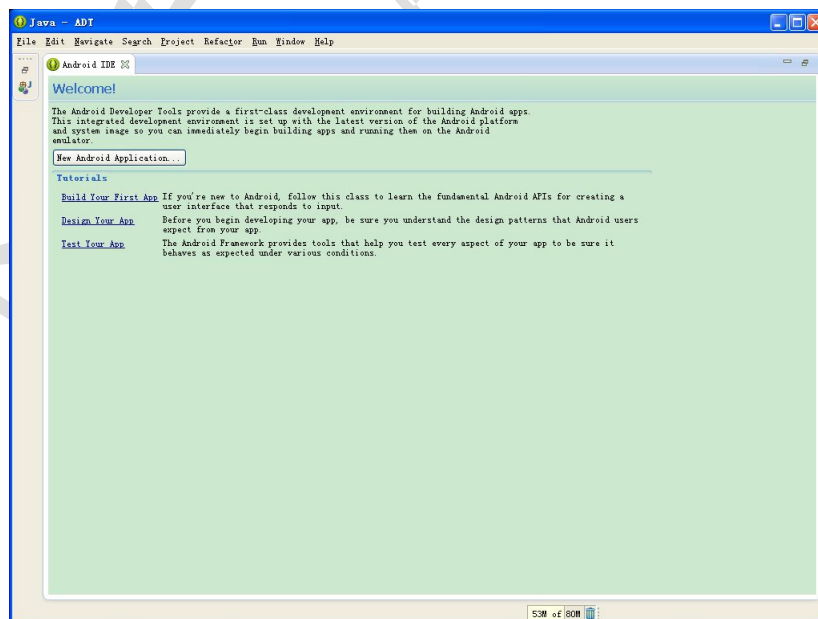
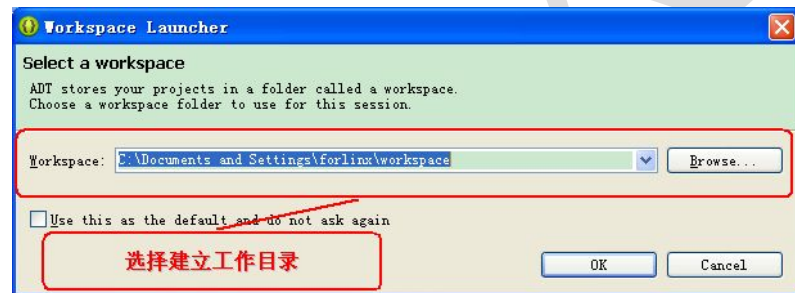


## 5.1.3 创建 helloworld 工程

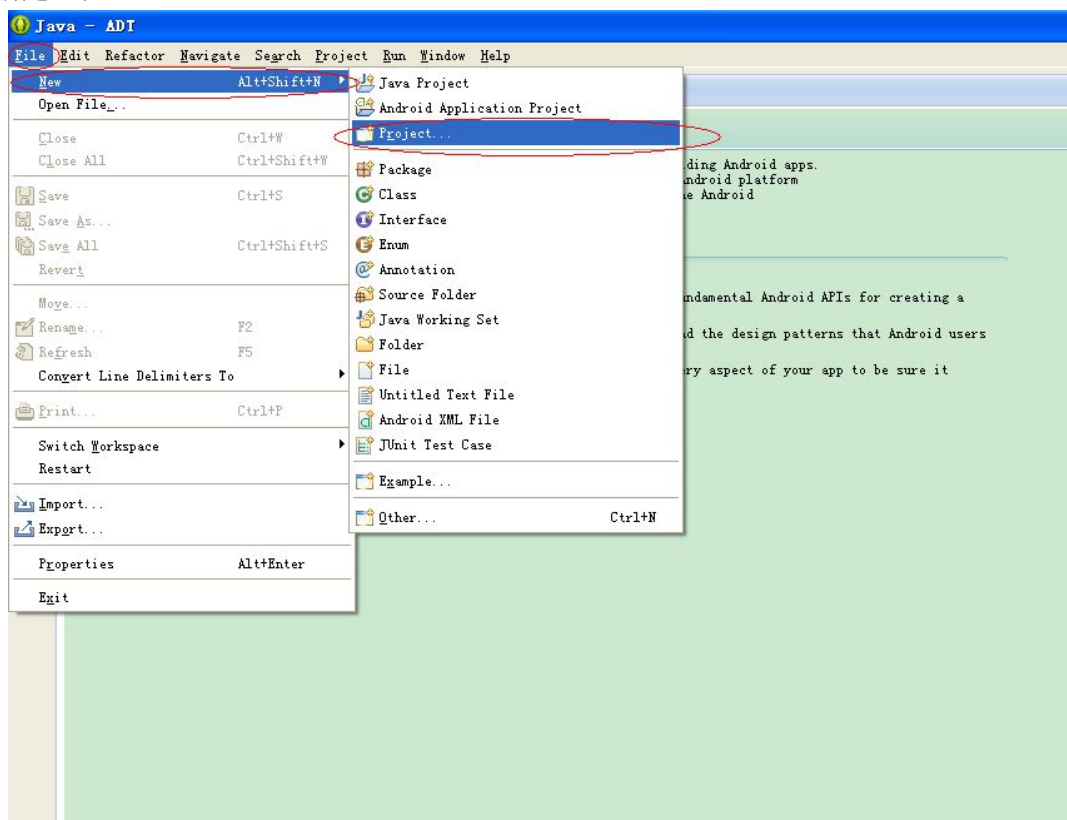
### 1. 打开 eclipse



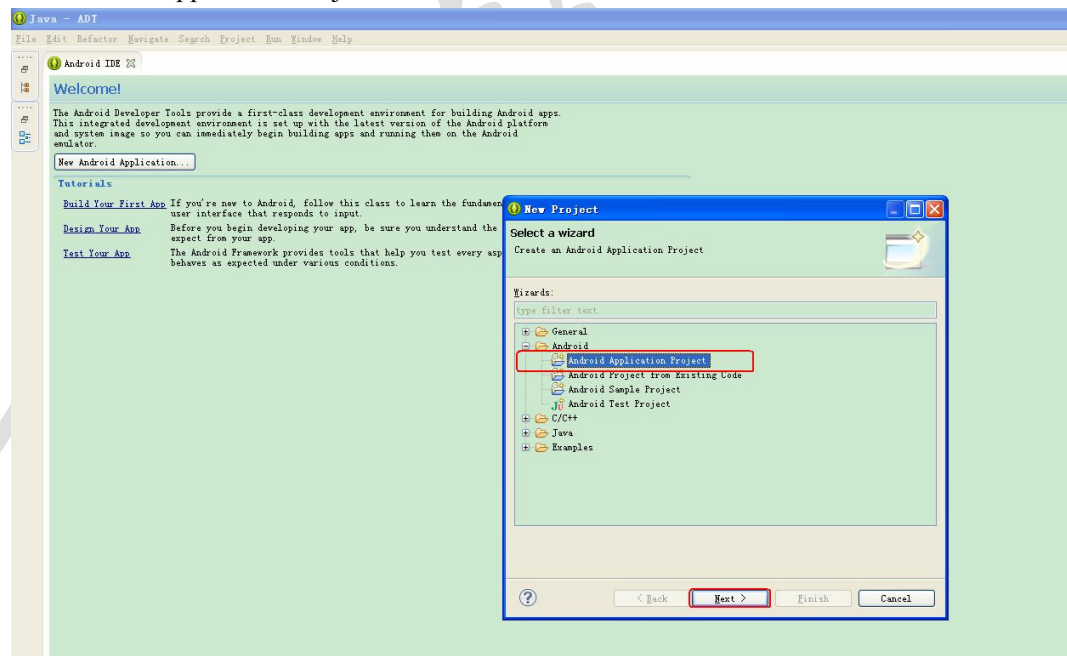
### 2. 创建工作目录



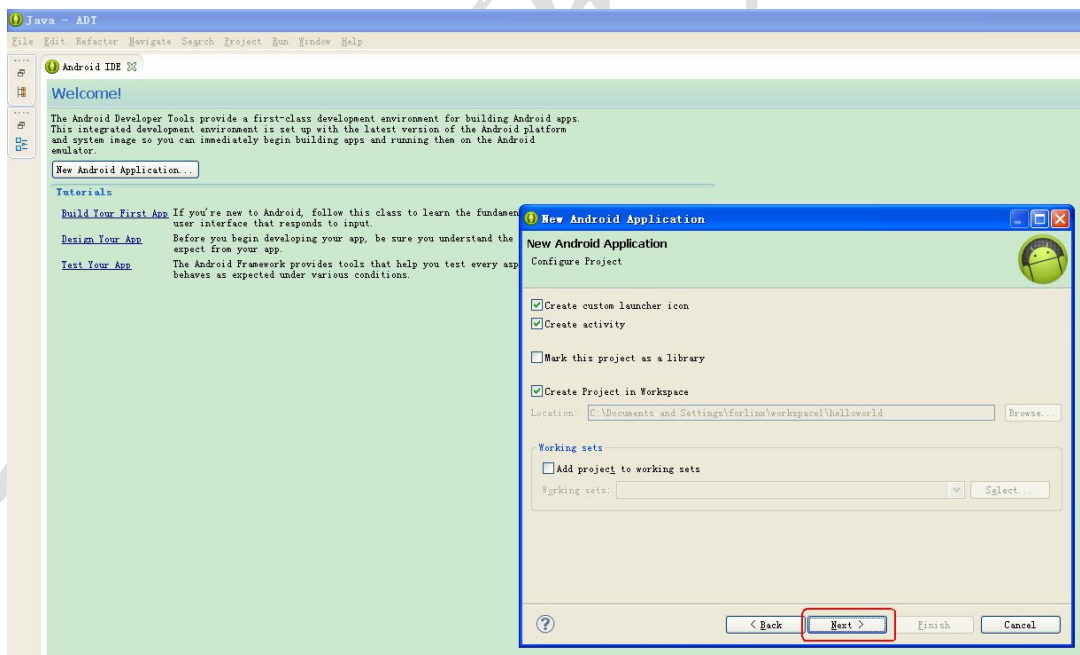
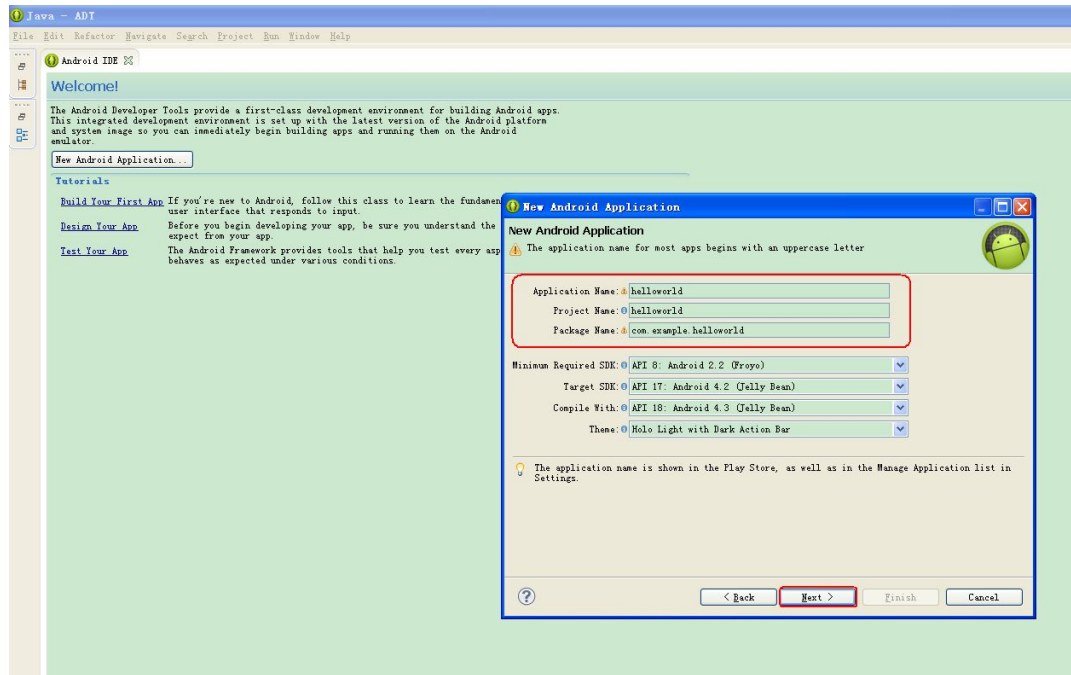
### 3. 新建工程

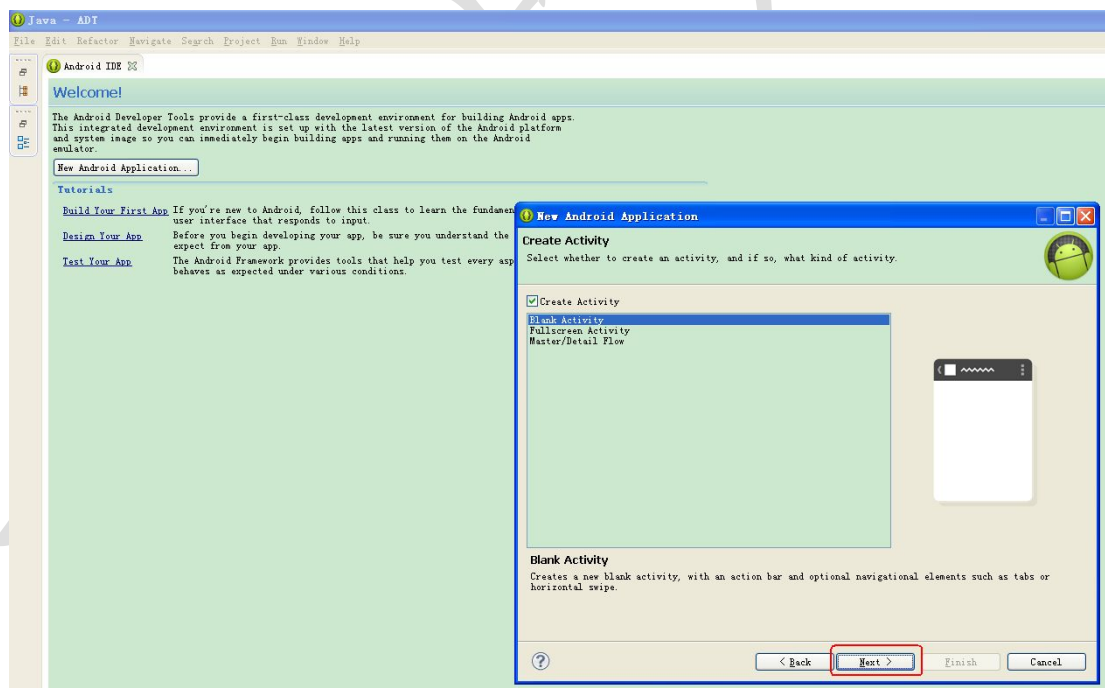
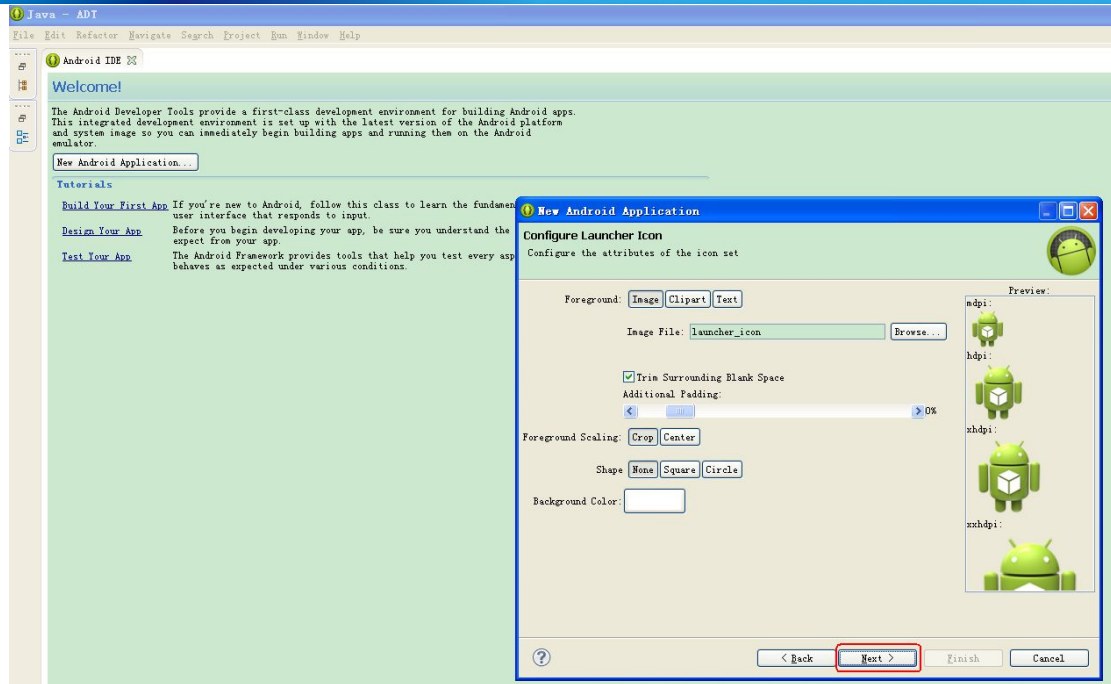


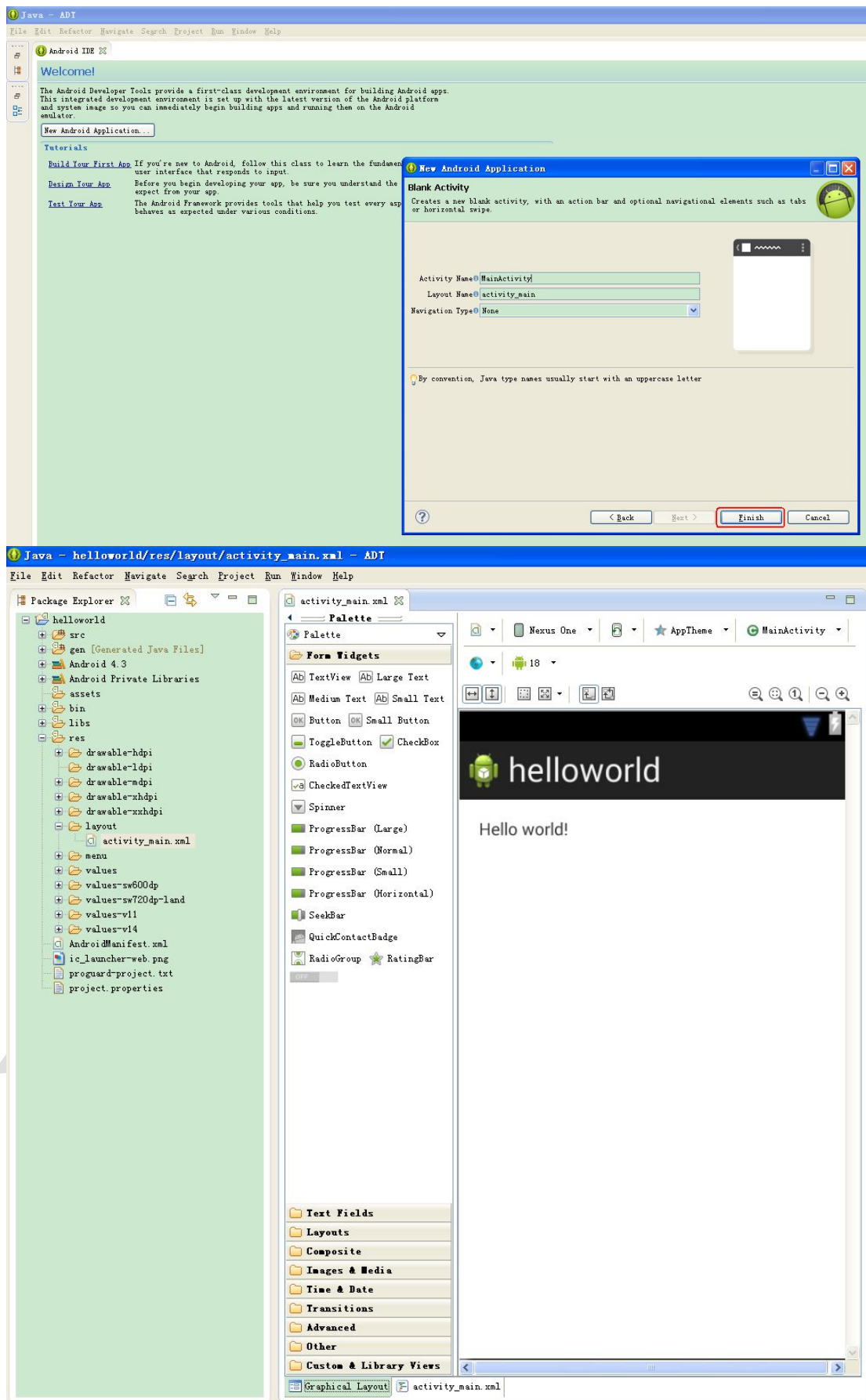
### 4. 选择 Android Application Project, 点击下一步;



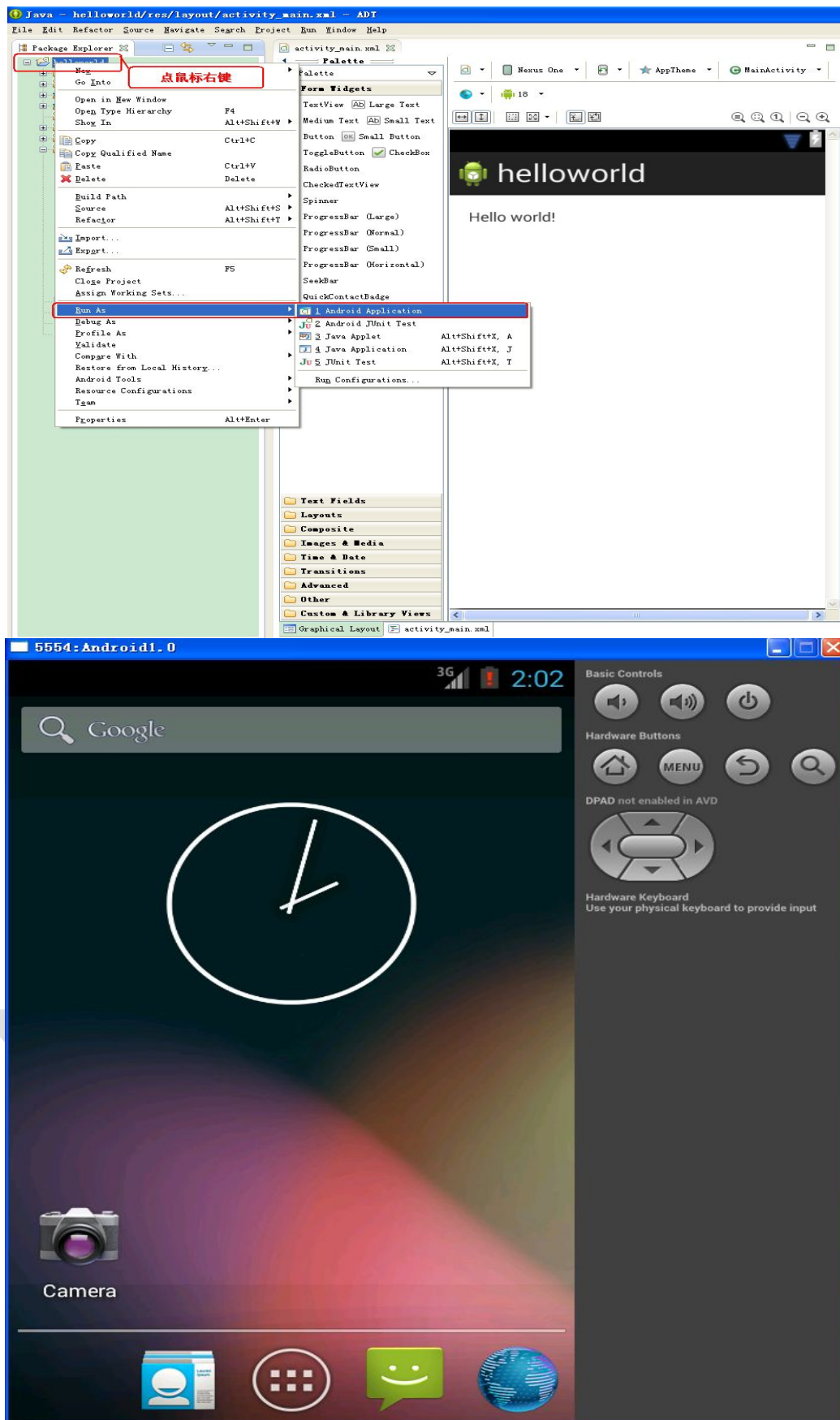
## 5. 应用工程名称全部默认，点击下一步；

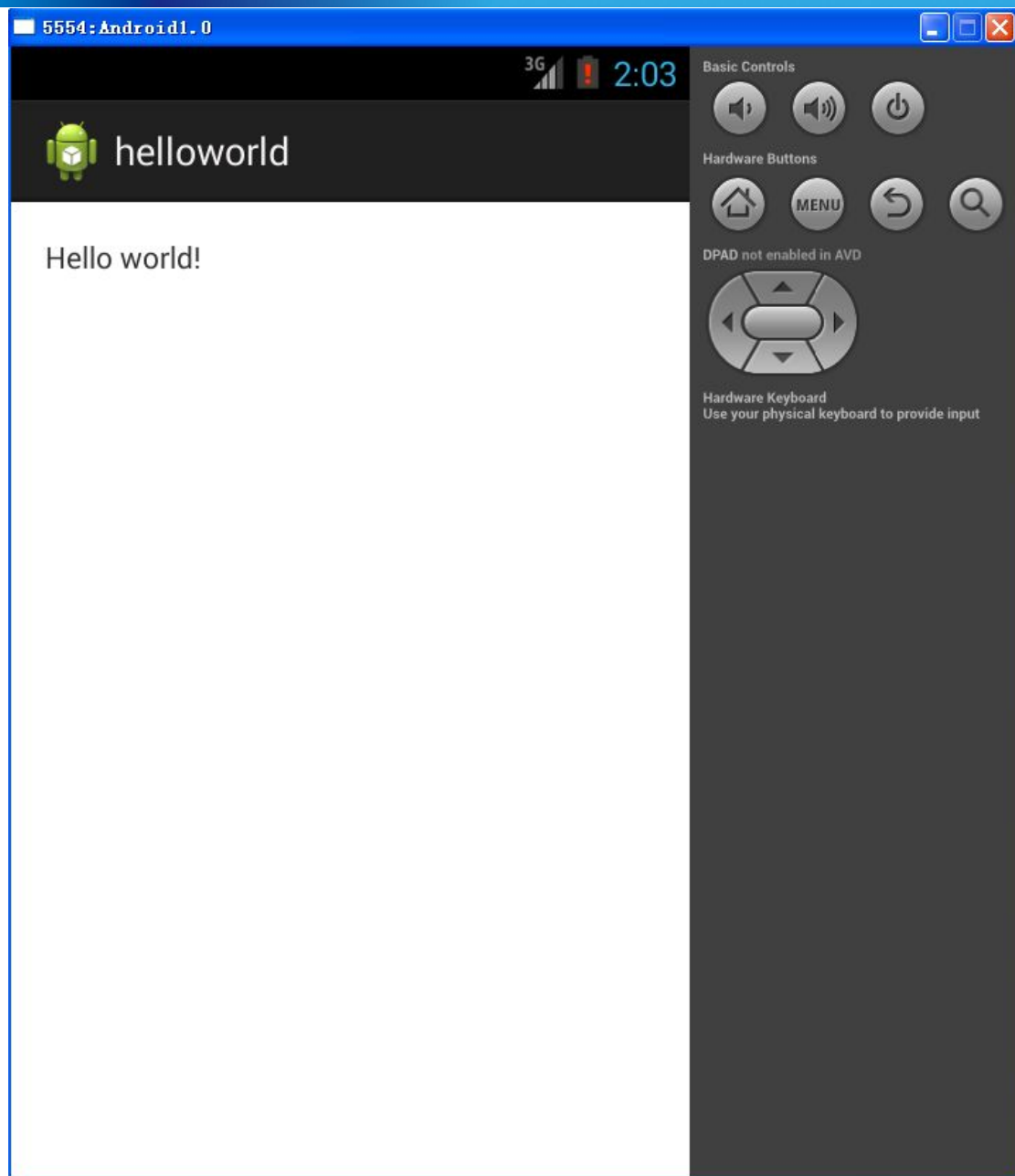






## 6. 在模拟器中运行 helloworld 程序



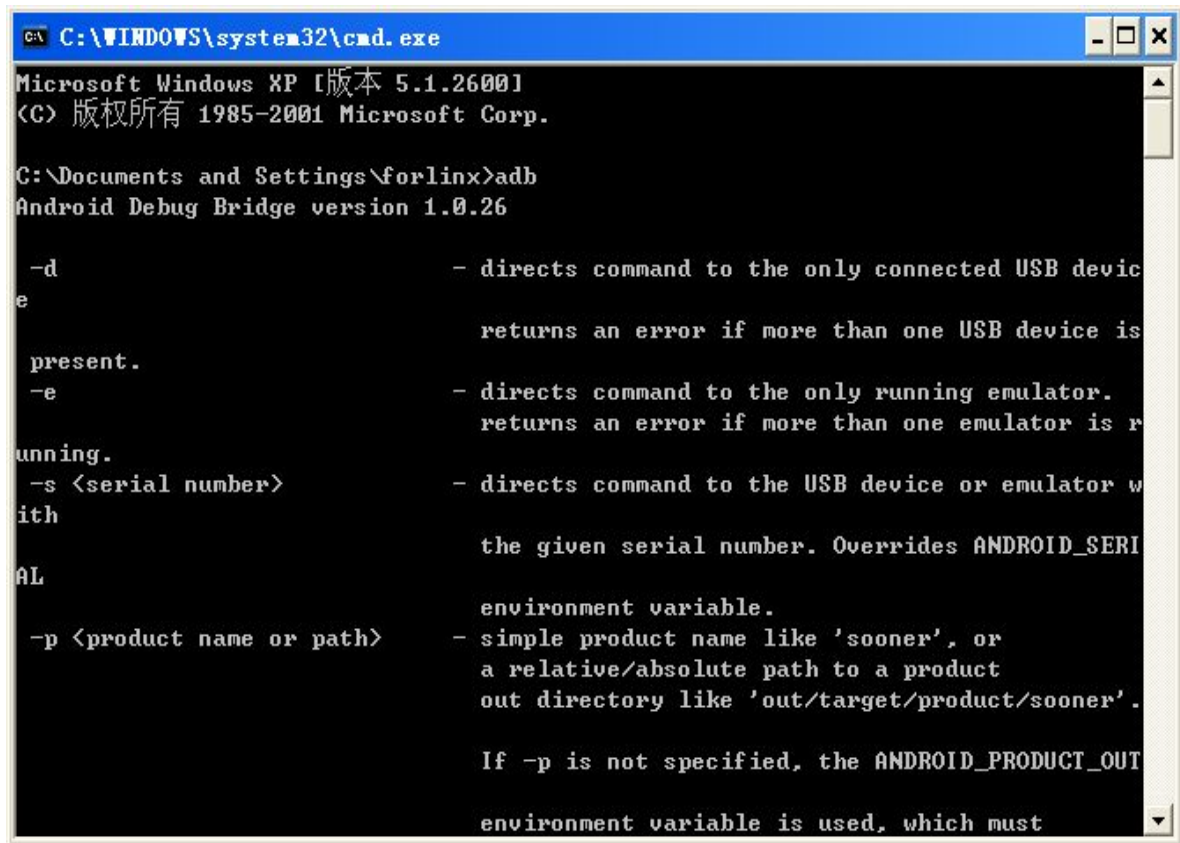


## 5.1.4 使用 adb

您也可以使用 adb 安装并调试应用程序(您的 adb 已经在导入 android\_sdk 时已经加入到环境变量内)。

### 1. 测试 adb 功能

通过点击开始菜单，在开始菜单下方的搜索框中输入 cmd，在 cmd.exe 上按回车来启动 DOS 窗口，在 DOS 窗口中，输入 adb 按回车，如果显示以下信息表示环境变量设置 OK：



```

C:\WINDOWS\system32\cmd.exe
Microsoft Windows XP [版本 5.1.2600]
(C) 版权所有 1985-2001 Microsoft Corp.

C:\Documents and Settings\forlinx>adb
Android Debug Bridge version 1.0.26

-d                - directs command to the only connected USB device
                  returns an error if more than one USB device is
                  present.
-e                - directs command to the only running emulator.
                  returns an error if more than one emulator is r
                  unning.
-s <serial number> - directs command to the USB device or emulator w
                  ith
                  the given serial number. Overrides ANDROID_SERI
                  AL
                  environment variable.
-p <product name or path> - simple product name like 'sooner', or
                  a relative/absolute path to a product
                  out directory like 'out/target/product/sooner'.

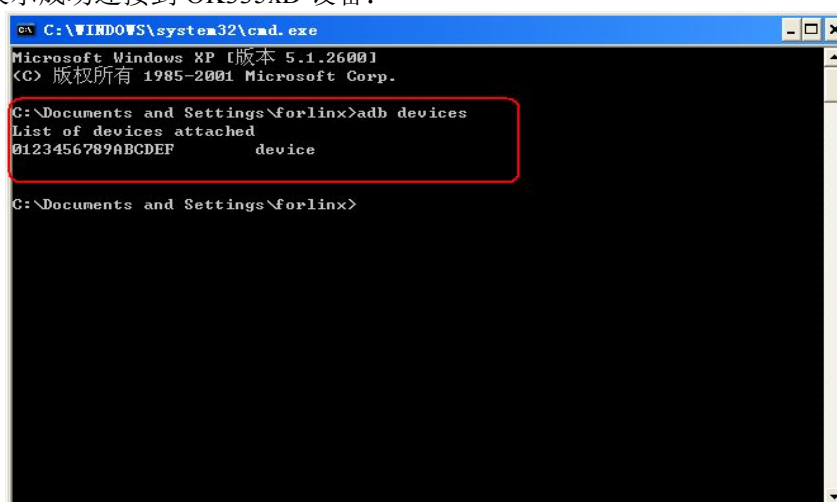
                  If -p is not specified, the ANDROID_PRODUCT_OUT
                  environment variable is used, which must
    
```

### 2. 测试 adb 的功能

首先查看设备连接状态。在 OK335xD 上启动 Android，然后用 mini USB 线将 OK335xD 与 PC 相连，在 DOS 窗口上输入以下命令验证开发板是否已连接：

```
#adb devices
```

显示以下内容表示成功连接到 OK335xD 设备：



```

C:\WINDOWS\system32\cmd.exe
Microsoft Windows XP [版本 5.1.2600]
(C) 版权所有 1985-2001 Microsoft Corp.

C:\Documents and Settings\forlinx>adb devices
List of devices attached
0123456789ABCDEF      device

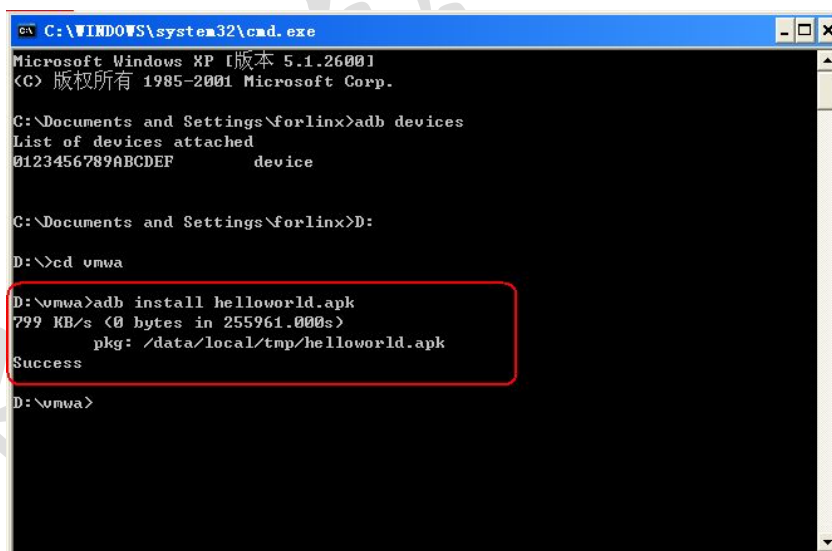
C:\Documents and Settings\forlinx>
    
```

接下来进入 ADB shell，使用以下命令可以进入 OK335xD 终端：

```
#adb shell
```

注：在 OK335xD 终端上输入 **exit** 退回到 DOS 提示符。

用 ADB 安装软件，以安装 D:\helloworld.apk 的程序为例，在 DOS 窗口中输入 `adb install D:\helloworld.apk` 命令进行安装。



```

C:\WINDOWS\system32\cmd.exe
Microsoft Windows XP [版本 5.1.2600]
(C) 版权所有 1985-2001 Microsoft Corp.

C:\Documents and Settings\forlinx>adb devices
List of devices attached
0123456789ABCDEF      device

C:\Documents and Settings\forlinx>D:
D:\>cd vmwa
D:\vmwa>adb install helloworld.apk
799 KB/s (0 bytes in 255961.000s)
pkg: /data/local/tmp/helloworld.apk
Success

D:\vmwa>
    
```

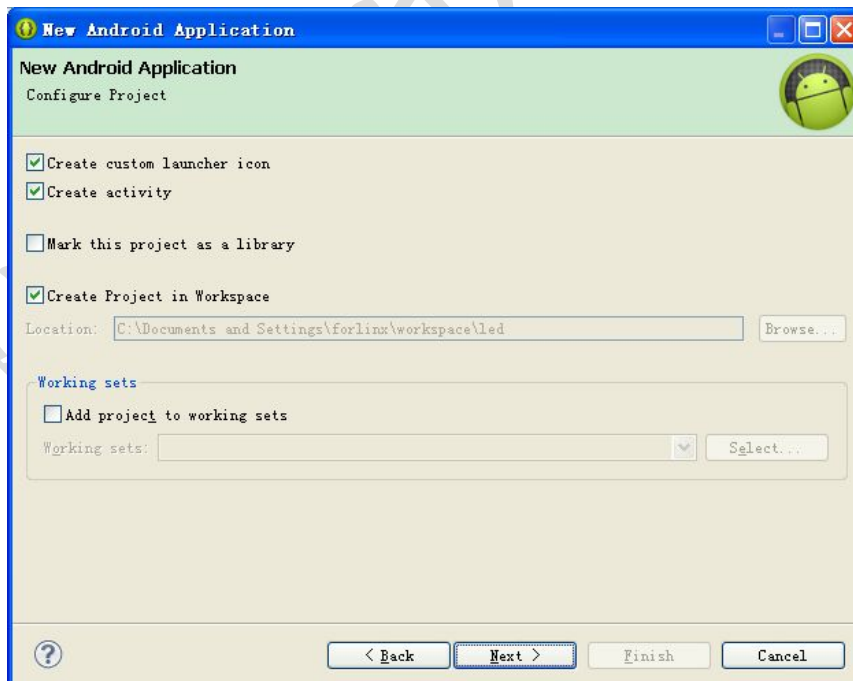
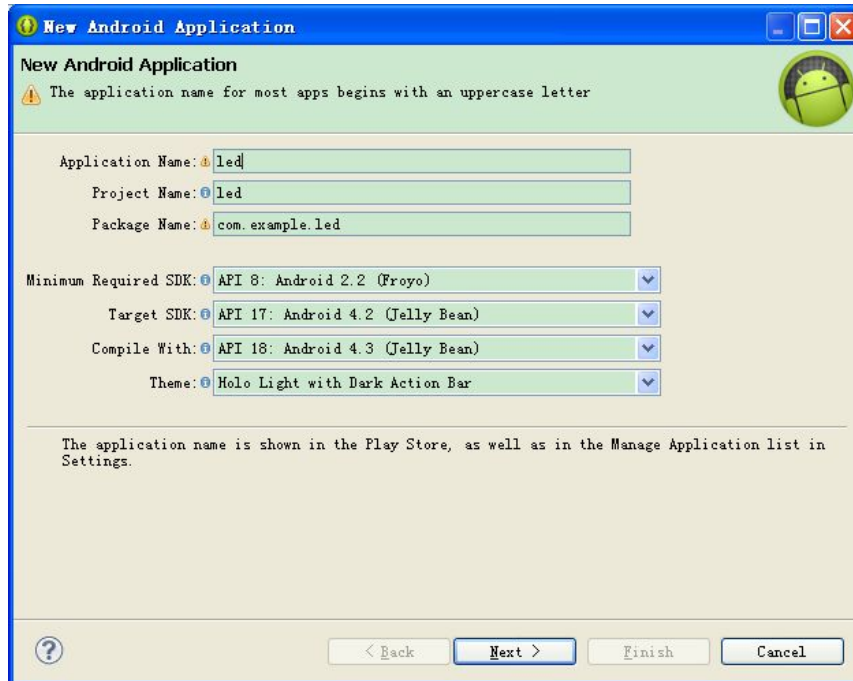
其它功能

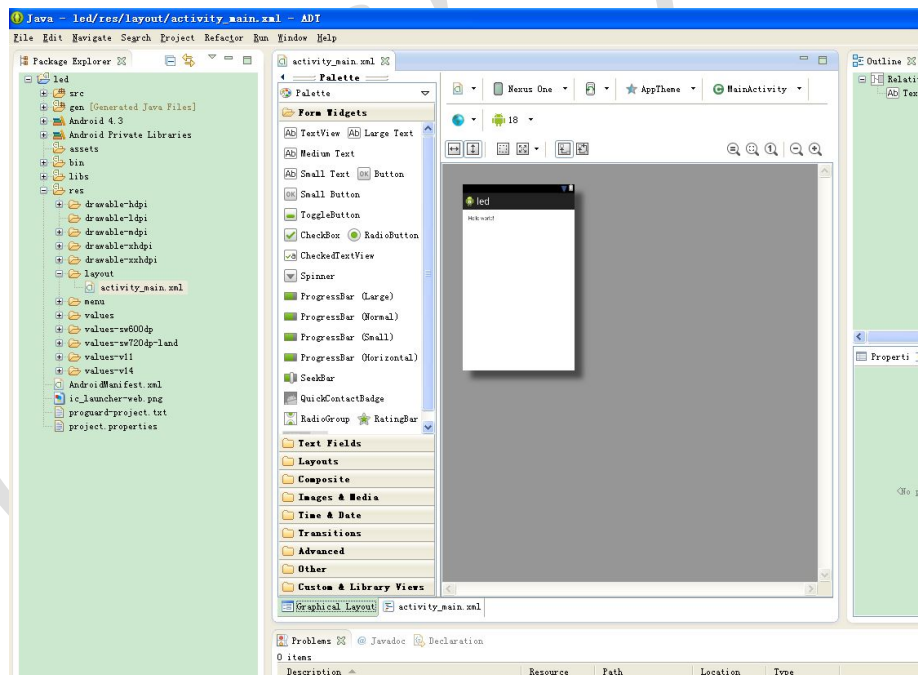
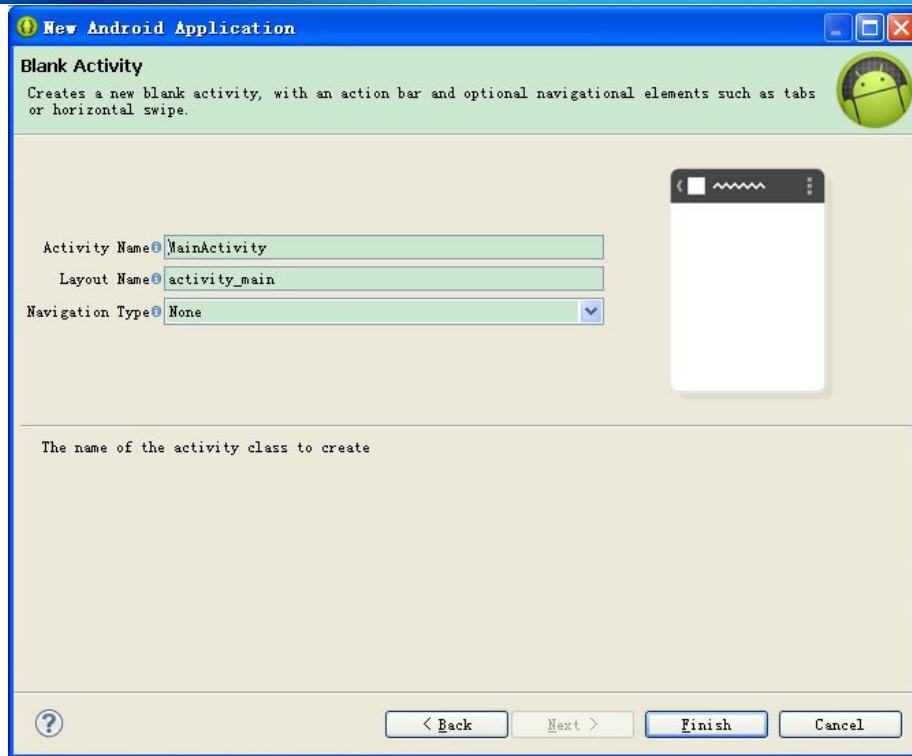
ADB 功能非常强大，除了安装软件、调试、Shell 功能外，还可以往开发板上传送文件等，读者可以自己挖掘。

## 5.2 在 Android 程序中访问硬件

现在我们以 jni 方式测试 LED 为例

1. 首先打开 eclipse，建立一个 android 工程





2. 在 MainActivity.java 中加入 public static native int led(int i, int j);  
MainActivity.java:

```
package com.example.led;
```

```
import android.os.Bundle;
import android.app.Activity;
import android.view.Menu;
```

```
public class MainActivity extends Activity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.main, menu);
        return true;
    }

}

public static native int led(int i, int j);
static
{
    System.loadLibrary("MainActivity");
}

}
```

3. 编译项目文件,bin 目录下会生成 led.apk。

4. 把 C:\Documents and Settings\forlinx\workspace 下的 led 文件夹复制到 ubuntu 下。进入 led 目录下建立 jni 目录

```
root@panda-System-Product-Name:/share/led# ls
AndroidManifest.xml  bin  ic_launcher-web.png  libs  proguard-project.txt  res
assets               gen  jni                  obj   project.properties    src
```

5. 利用 javah 命令生成头文件,该头文件中包含了符合 jni 格式的函数名,

```
javah -classpath bin/classes -d jni com.example.led.MainActivity
```

```
root@panda-System-Product-Name:/share/led# javah -classpath bin/classes -d jni
com.example.led.MainActivity
root@panda-System-Product-Name:/share/led# ls jni/
com_example_led_MainActivity.h
root@panda-System-Product-Name:/share/led#
```

## 6. jni 目录下新建 led.c

led.c:

```
#include <jni.h>
#include <stdio.h>
#include <stdlib.h>
#include <fcntl.h>
#include <unistd.h>
#include <sys/ioctl.h>
#include <android/log.h>

#define LOG_TAG "LED" //android logcat
#define LOGI(...) __android_log_print(ANDROID_LOG_INFO,LOG_TAG,__VA_ARGS__)
#define LOGE(...) __android_log_print(ANDROID_LOG_ERROR,LOG_TAG,__VA_ARGS__)
jint JNICALL Java_com_example_led_MainActivity_led(JNIEnv *env, jclass thiz, jint led_nu, jint on)
{
    int fd;

    fd = open("/dev/led", O_RDWR);
    if(fd < 0)
        printf("Can't open /dev/led!\n");

    ioctl(fd, on, led_nu);
    LOGI("led_nu=%d,state=%d\n", led_nu, on);
    close(fd);

    return 0;
}
```

## 7. 在 jni 目录下新建 Android.mk 和 Application.mk

Android.mk:

LOCAL\_PATH := \$(call my-dir)

include \$(CLEAR\_VARS)

LOCAL\_MODULE := MainActivity

LOCAL\_SRC\_FILES := led.c

LOCAL\_LDLIBS := -llog

LOCAL\_C\_INCLUDES :=

/work/forlinux/TI\_Android\_JB\_4.2.2\_DevKit\_4.1.1/frameworks/base/core/jni/android/graphics \

include \$(BUILD\_SHARED\_LIBRARY)

Application.mk :

APP\_ABI := armeabi armeabi-v7a x86

用 ndk-build 编译生成 so 库

下载 linux 下的 ndk 安装包 android-ndk-r9-linux-x86\_64.tar.bz2

解压出来

命令: tar -zxvf android-ndk-r9-linux-x86\_64.tar.bz2

解压之后会有一个 android-ndk-r9 目录, 进入 android-ndk-r9 目录下, 用 pwd 看一下您的目录这是我的

```
root@panda-System-Product-Name:/share/android-ndk-r9# pwd
/share/android-ndk-r9
root@panda-System-Product-Name:/share/android-ndk-r9#
```

#cd led

#cd jni

# /share/android-ndk-r9/ndk-build

```
root@panda-System-Product-Name:/share/android-ndk-r9# pwd
/share/android-ndk-r9
root@panda-System-Product-Name:/share/android-ndk-r9# cd /share/led/
root@panda-System-Product-Name:/share/led# cd jni
root@panda-System-Product-Name:/share/led/jni# /share/android-ndk-r9/ndk-build
Android NDK: WARNING: APP PLATFORM android-18 is larger than android:minSdkVersion 8 in /share/led/AndroidManifest.xml
Install      : libMainActivity.so => libs/armeabi/libMainActivity.so
Install      : libMainActivity.so => libs/armeabi-v7a/libMainActivity.so
Install      : libMainActivity.so => libs/x86/libMainActivity.so
root@panda-System-Product-Name:/share/led/jni#
```

8. 把 jni 目录拷贝到 C:\Documents and Settings\forlinx\workspace\led 下, 把 ubuntu 上 led 下的 libs 中的文件拷贝到 C:\Documents and Settings\forlinx\workspace\led\libs 下。

9. 修改布局, 在 activity\_main.xml 中为界面添加按钮

res->layout->activity\_main.xml

activity\_main.xml:

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical" >
```

```
<ToggleButton
    android:id="@+id/btn1"
    android:layout_width="140dip"
    android:layout_height="wrap_content"
    android:textOff="led1 off"
    android:textOn="led1 on" />
```

```
<ToggleButton
    android:id="@+id/btn2"
    android:layout_width="140dip"
    android:layout_height="wrap_content"
    android:textOff="led2 off"
    android:textOn="led2 on" />
```

```
<ToggleButton
    android:id="@+id/btn3"
```

```
android:layout_width="140dip"
android:layout_height="wrap_content"
android:textOff="led3 off"
android:textOn="led3 on" />
```

```
<ToggleButton
    android:id="@+id/btn4"
    android:layout_width="140dip"
    android:layout_height="wrap_content"
    android:textOff="led4 off"
    android:textOn="led4 on" />
```

```
</LinearLayout>
```

## 10. 在 MainActivity.java 中添加监听按键代码

MainActivity.java:

```
package com.example.led;
```

```
import android.app.Activity;
import android.content.Intent;
import android.net.Uri;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.ToggleButton;
import android.util.Log;
import android.widget.CompoundButton.OnCheckedChangeListener;
```

```
public class MainActivity extends Activity {
    private static final String TAG = "LED";
    private ToggleButton button1;
    private ToggleButton button2;
    private ToggleButton button3;
    private ToggleButton button4;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        Log.w(TAG, "layout");

        button1 = (ToggleButton)findViewById(R.id.btn1);
        button2 = (ToggleButton)findViewById(R.id.btn2);
        button3 = (ToggleButton)findViewById(R.id.btn3);
        button4 = (ToggleButton)findViewById(R.id.btn4);
        Log.w(TAG, "button");
        button1.setOnClickListener(new Button.OnClickListener()
        {
            public void onClick(View v)
            {
                if (button1.isChecked())
                {
```

```
        Log.w(TAG,"----led1 on");
        led(0, 1);
    }
    else
    {
        Log.w(TAG,"----led1 off");
        led(0, 0);
    }
}
});

button2.setOnClickListener(new Button.OnClickListener()
{
    public void onClick(View v)
    {
        if (button2.isChecked())
        {
            Log.w(TAG,"----led2 on");
            led(1, 1);
        }
        else
        {
            Log.w(TAG,"----led2 off");
            led(1, 0);
        }
    }
});

button3.setOnClickListener(new Button.OnClickListener()
{
    public void onClick(View v)
    {
        if (button3.isChecked())
        {
            Log.w(TAG,"----led3 on");
            led(2, 1);
        }
        else
        {
            Log.w(TAG,"----led3 off");
            led(2, 0);
        }
    }
});

button4.setOnClickListener(new Button.OnClickListener()
{
    public void onClick(View v)
    {
        if (button4.isChecked())
        {
            Log.w(TAG,"----led4 on");
            led(3, 1);
        }
        else
        {
            Log.w(TAG,"----led4 off");
```

```
        led(3, 0);
    }
}

};

}

public static native int led(int i, int j);

static
{
    System.loadLibrary("MainActivity");
}

}
```

11. 编译整个项目,在 bin 目录下生成 led.apk, 您可以使用 adb install 安装到开发板中。

注：飞凌提供测试串口、led、看门狗的源码，源码位置在/src/test 中。

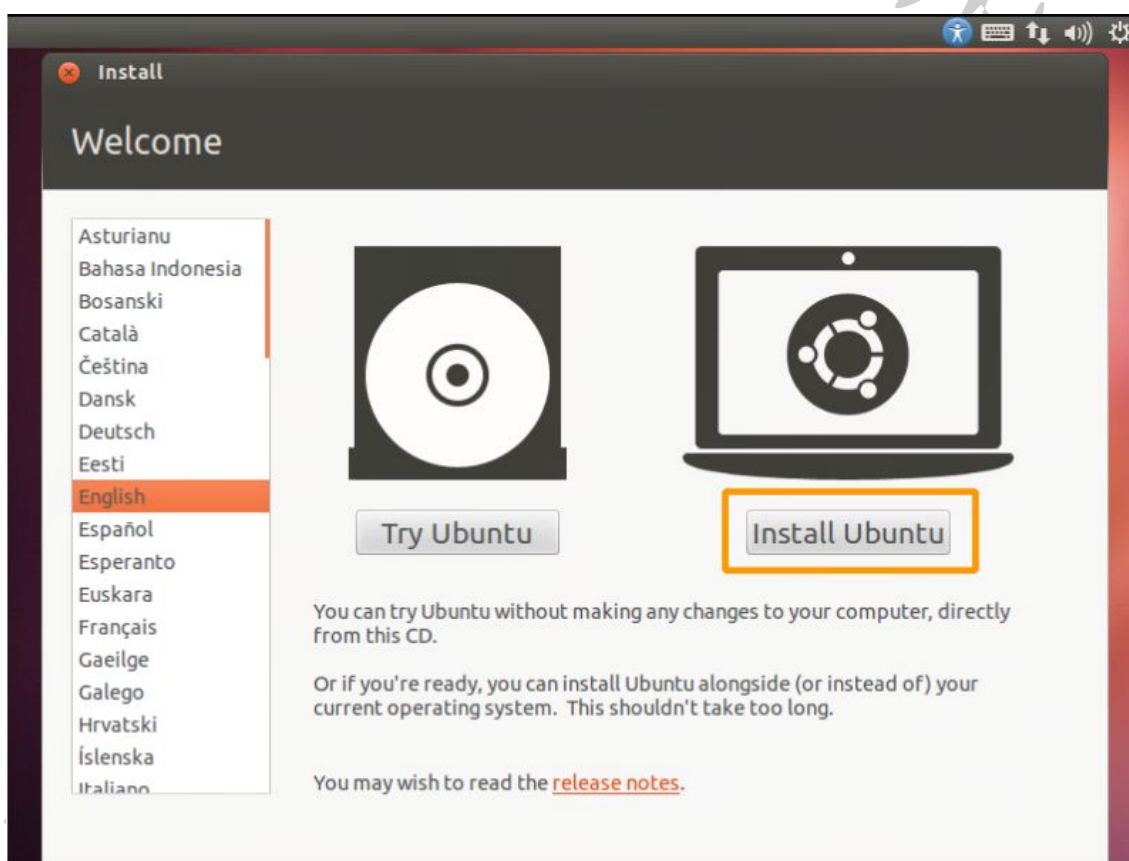
## 附录 1: Ubuntu 的安装与设置

### F1.1 Ubuntu 的安装

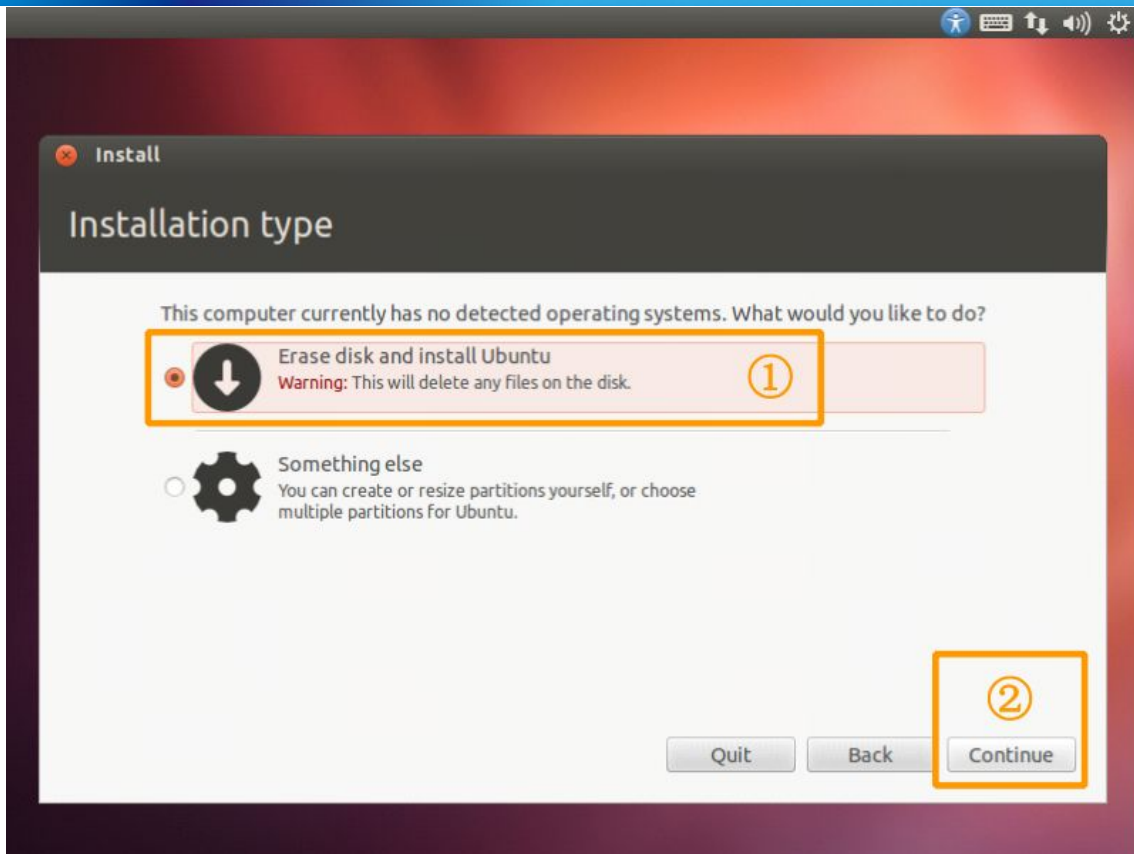
**步骤 1:** 首先准备一张 Ubuntu12.04 .02 的安装光盘。将光盘插入光驱，在 PC 的 bios 中把 PC 启动方式设置为光驱启动，启动 PC。

用户光盘 A: \Tools\ubuntu-12.04.2-desktop-amd64.iso

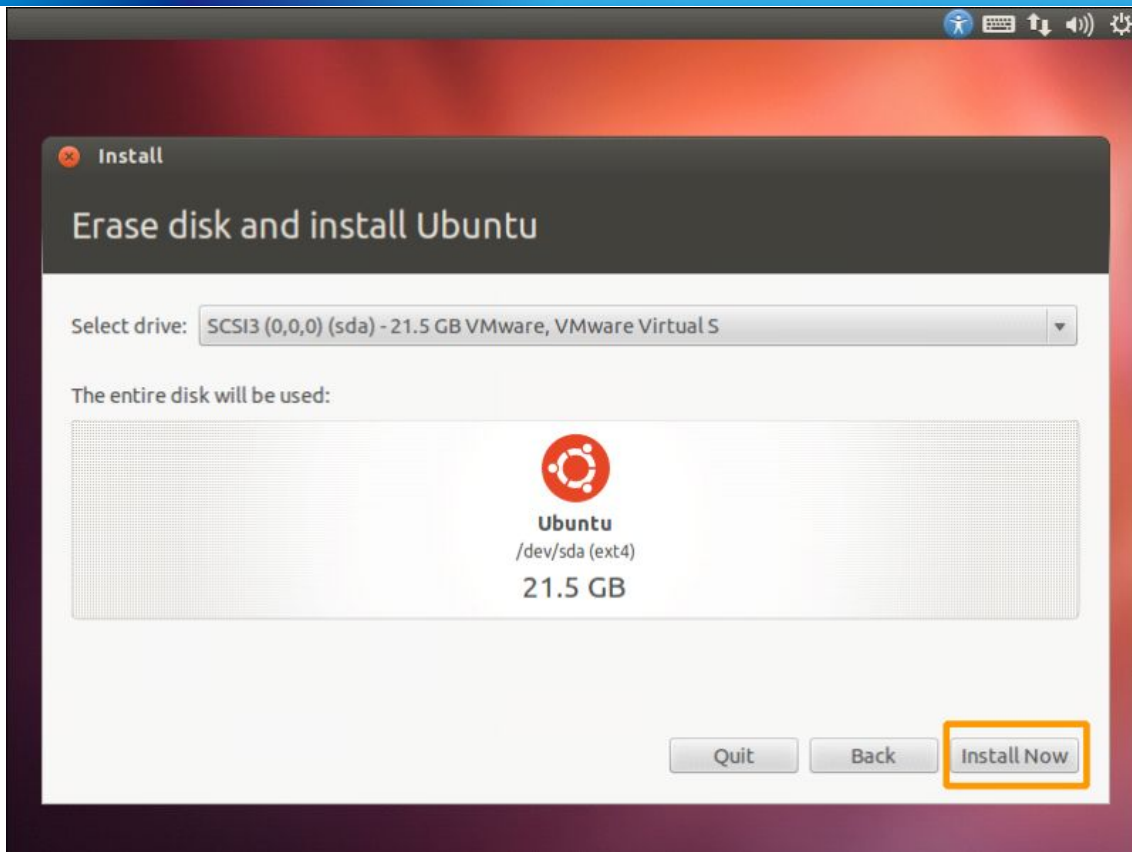
**步骤 2:** 启动 PC 后，安装盘会提示选择安装语言种类。使用 PC 键盘的方向键选择在安装过程中显示的语言，在这里我们选择 English>>点击“Install Ubuntu”按钮。



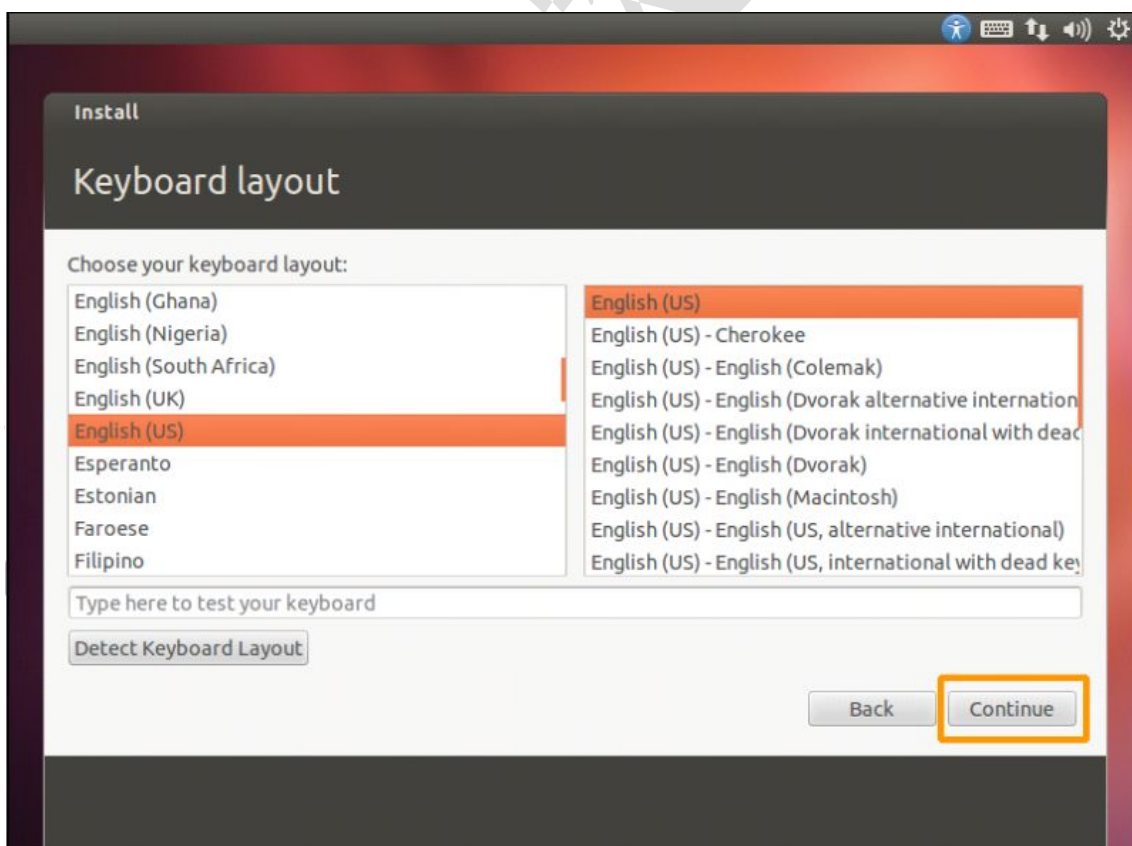
**步骤 3:** 选择“Erase disk and install Ubuntu”>>点击“Continue”按钮  
(注意：非虚拟系统会格式化您的硬盘！请谨慎操作。)



**步骤 4:** 硬盘空间以及挂载点的分配。这里选择默认。也可根据个人需要来进行设置,单击 “Install Now”



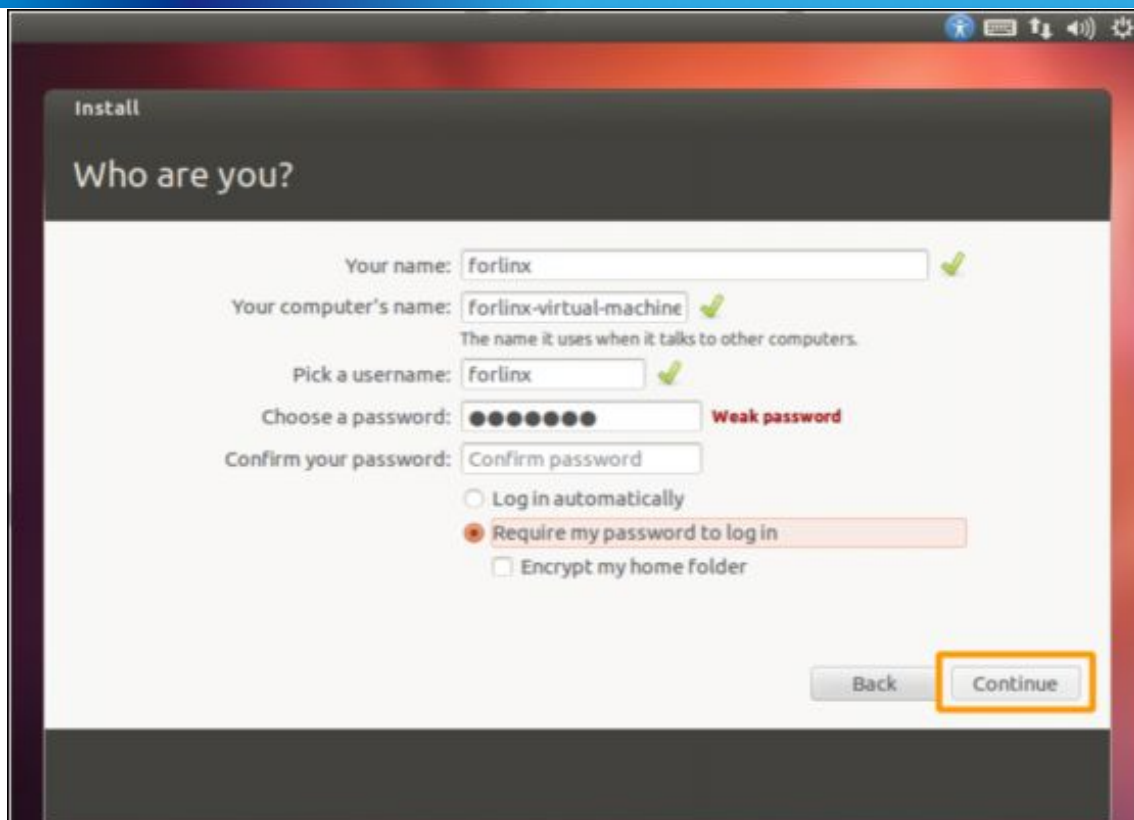
步骤 5：选择键盘布局，默认即可，单击“Continue”



**步骤 6:** 选择所在地,上海, Continue



**步骤 7:** 输入系统用户名和密码，这里输入的用户名:forlinx, 密码:123456 仅作为事例。点击“Continue”按钮,系统会完成安装。



注意：这里的用户名属于普通用户，不具备 root 用户权限。

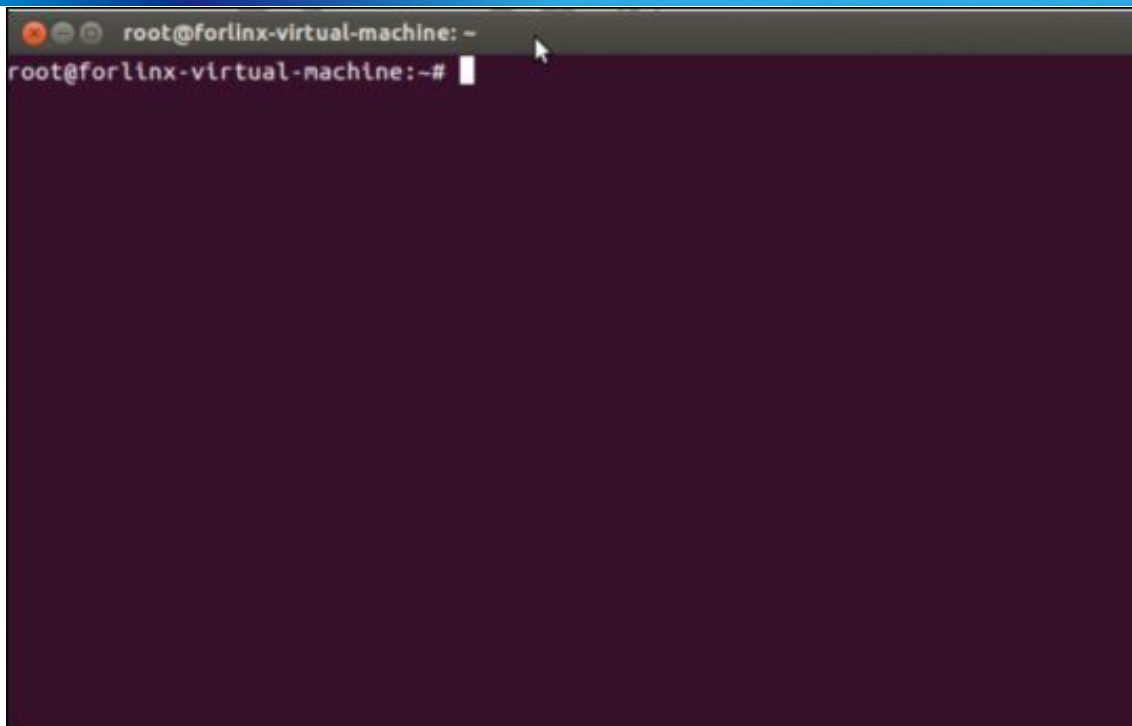
## F1.2 Linux 终端

在 Linux 系统中终端是一个很实用的与操作系统交互的窗口，您可以使用终端来编译应用程序，启动各种系统服务，在 Linux 系统中占据着非常重要的位置。

如图进行操作，即可将终端设置为快捷键。



以后每次单击启动面板上的图标即可运行终端窗口：



## F1.3 Ubuntu12.04.02 root 用户登录设置

Ubuntu12.04.02 默认是不允许 root 登录的，在登录窗口只能看到普通用户和访客登录。以普通身份登陆 Ubuntu 后我们需要做一些修改。

**步骤 1:** 普通用户登录后，修改系统配置文件需要切换到超级用户模式，在终端窗口里面输入命令后回车：

```
$sudo -s
```

**步骤 2:** 然后输入安装 Ubuntu12.04.02 时设置的系统密码，即可进入 root 用户权限模式。

**步骤 3:** 在终端窗口中执行

```
$gedit /etc/lightdm/lightdm.conf
```

**步骤 4:** 在文本全部内容删除并粘贴以下七行内容：

```
[SeatDefaults]
allow-guest=false
autologin-user=root
autologin-user-timeout=0
autologin-session=lightdm-autologin
user-session=ubuntu
greeter-session=unity-greeter
```

**步骤 5:** 然后我们启动 root 帐号：

```
$ sudo passwd root
```

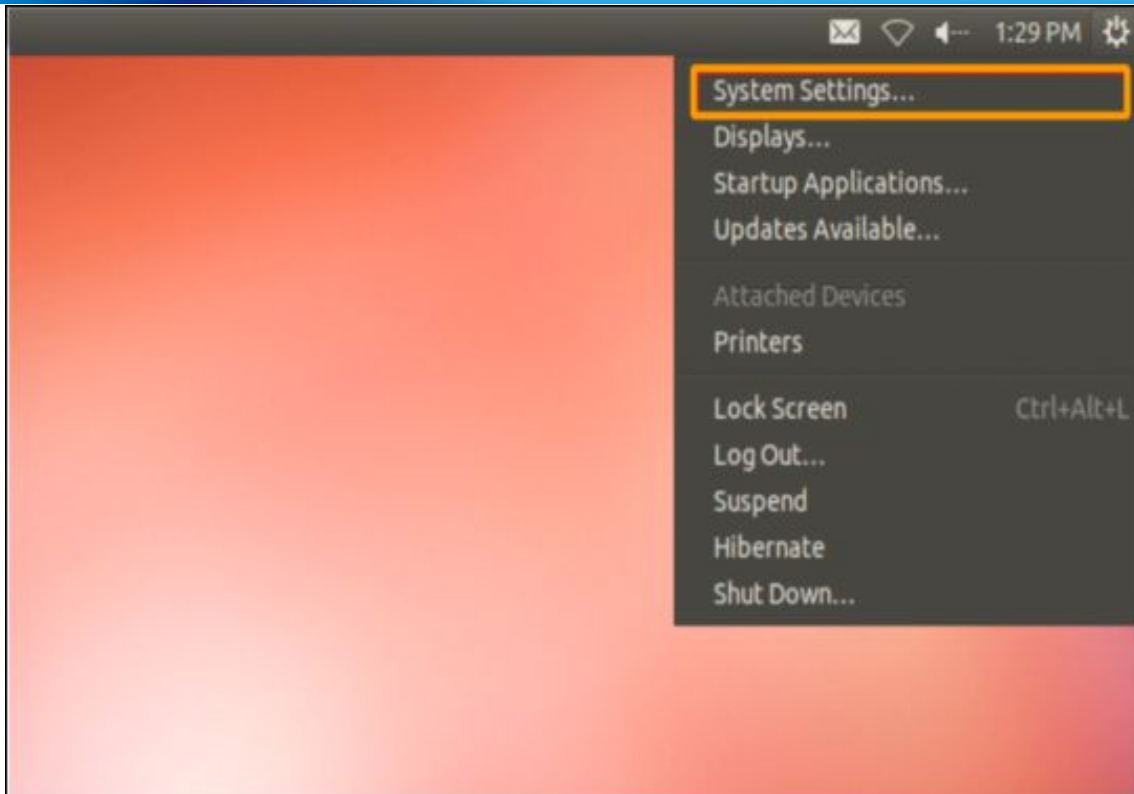
**步骤 6:** 根据提示输入 root 帐号密码。（注意：Linux 中输入密码无符号显示）

**步骤 7:** 重启 ubuntu，不需要手动输入 root 用户名密码，系统会自动进入 root 用户。

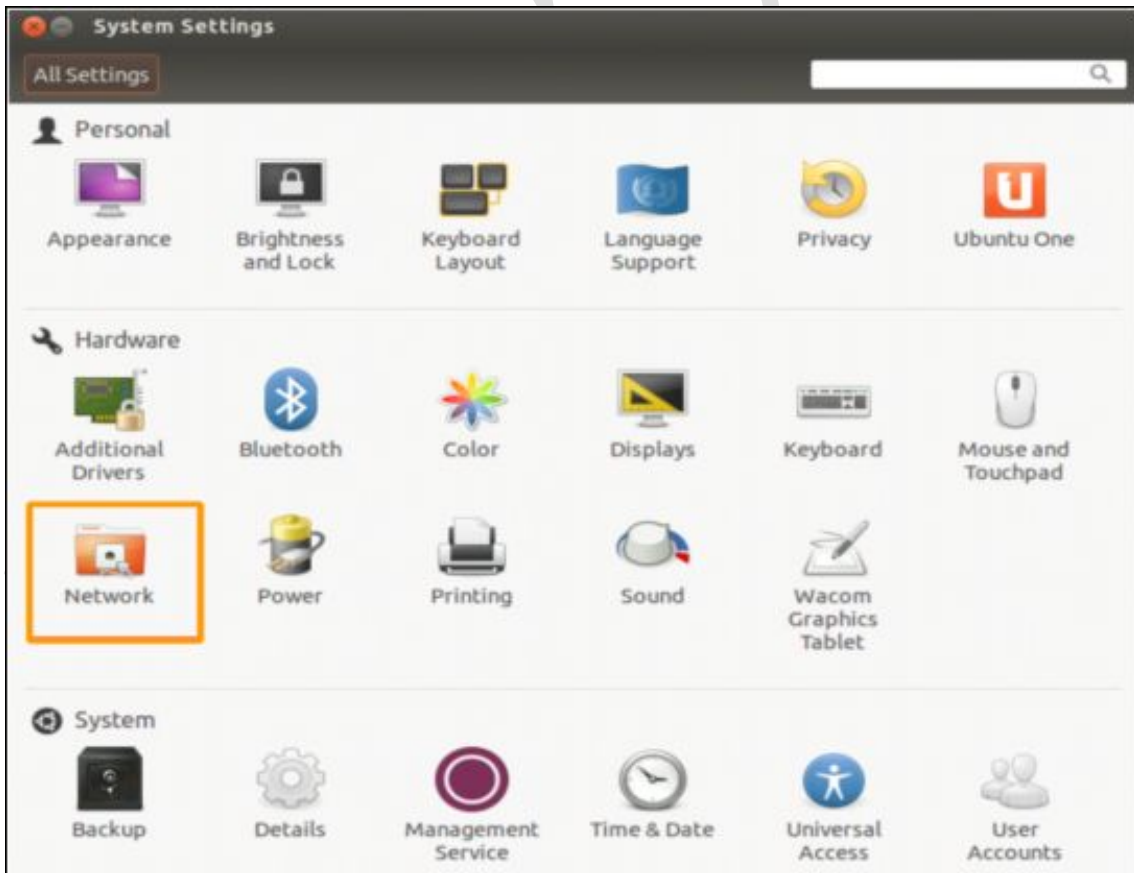
## F1.4 设置 Ubuntu 网络参数

因为每个 PC 的网络环境不一定一样，所以需要您根据自己的实际情况来设置 Ubuntu 的网络，如果设置不成功，可以去 Ubuntu 的官方论坛上咨询。本手册 PC Linux 网络的设置方法，仅供参考。

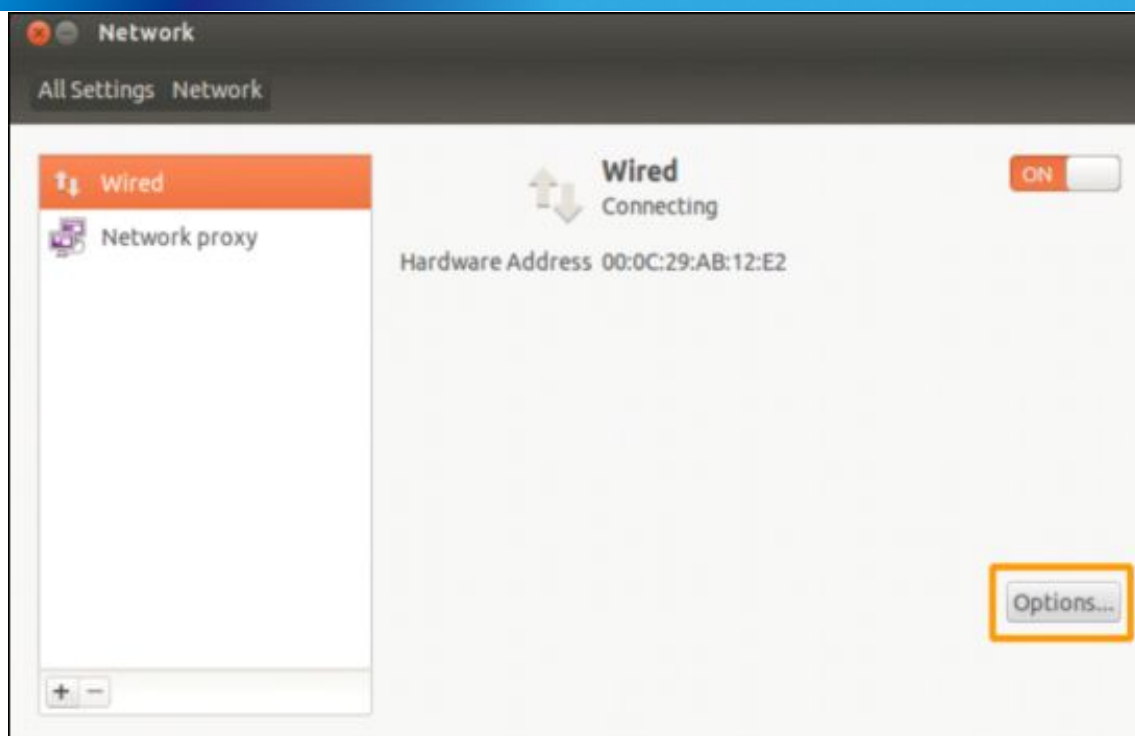
**步骤 1:** 启动 Ubuntu，root 用户登陆系统，单击桌面最右上端的按钮 ，弹出如下选项：



步骤 2: 选择 System Settings，双击 Network 进入网络设置项：



步骤 3: 单击 Options 按钮



**步骤 4:** 选择 IPV4 设置，输入您的 IP 地址、子网掩码、网关、DNS，点击 Save...，网络设置成功。



**步骤 5:** 测试一下，笔者的宿主机 IP 为 192.168.0.30，用虚拟机来 ping 宿主机，如下图证明网络设置成功。

```

root@forlinux-virtual-machine: ~
root@forlinux-virtual-machine:~# ping 192.168.0.30
PING 192.168.0.30 (192.168.0.30) 56(84) bytes of data.
64 bytes from 192.168.0.30: icmp_req=1 ttl=64 time=0.785 ms
64 bytes from 192.168.0.30: icmp_req=2 ttl=64 time=0.239 ms
64 bytes from 192.168.0.30: icmp_req=3 ttl=64 time=0.174 ms
64 bytes from 192.168.0.30: icmp_req=4 ttl=64 time=0.179 ms
64 bytes from 192.168.0.30: icmp_req=5 ttl=64 time=0.251 ms
64 bytes from 192.168.0.30: icmp_req=6 ttl=64 time=0.115 ms
64 bytes from 192.168.0.30: icmp_req=7 ttl=64 time=0.249 ms

```