第一步当然是下载android源码啦~ TI官网有提供。

地址如下：

<http://software-dl.ti.com/dsps/dsps_public_sw/sdo_tii/TI_Android_DevKit/TI_Android_ICS_4_0_3_DevKit_3_0_1/index_FDS.html>

下载：[TI Android ICS 4.0.3 DevKitV3.0.1 AM335x EMV-SK Sources](http://software-dl.ti.com/dsps/dsps_public_sw/sdo_tii/TI_Android_DevKit/TI_Android_ICS_4_0_3_DevKit_3_0_1/exports/TI-Android-ICS-4.0.3-DevKit-EVM-SK-3.0.1.bin)

下载的环境：UBUNTU （我用WIN7下载卡死在了99.99% ）PS：虚拟机的空间要足够大（25G以上）

下载完成解压前，得先安装需要的软件。

sudo gedit /etc/apt/sources.list

在最后一行加入：

deb <http://us.archive.ubuntu.com/ubuntu/hardy> multiverse 保存退出

接下来：

$ sudo apt-get update

$ sudo apt-get install git-core gnupg sun-java6-jdk flex bison gperf libsdl-dev libesd0-dev libwxgtk2.6-dev build-essential zip curl libncurses5-dev zlib1g-dev minicom tftpd uboot-mkimage expect

安装软件时，安步骤走即可。

之后就可以安装下载好的 android源码啦。

步骤：

$ mkdir $HOME/rowboat-android

$ cd $HOME/rowboat-android

$ chmod a+x TI-Android-ICS-4.0.3\_AM335x\_3.0.1.bin

$ ./TI-Android-ICS-4.0.3\_AM335x\_3.0.1.bin

大概15分钟后，就可以安装完成。安装完成后，得到如图资源：

**To Build Bootloader**

**cd u-boot**

$ make CROSS\_COMPILE=arm-eabi- distclean

$ make CROSS\_COMPILE=arm-eabi- am335x\_evm\_config

$ make CROSS\_COMPILE=arm-eabi-

**大概10分钟后编译完成。在当前目录下得到生成的MLO以及u-boot.Img**

### To Build Android Linux Kernel

### 我的开发板环境是beaglebone，自己设计了扩展板，为了使得源码适合我的扩展板，所以在编译内核之前，修改一下源码。

cd kernel

gedit arch/arm/mach-omp2/borad-am335xevm.c

**举个例子：**

**1 增加用户指示灯：**

**找到：**

static struct pinmux\_config gpio\_led\_mux[] = {

{"gpmc\_ad4.gpio1\_4", OMAP\_MUX\_MODE7 | AM33XX\_PIN\_INPUT},

{"gpmc\_ad5.gpio1\_5", OMAP\_MUX\_MODE7 | AM33XX\_PIN\_INPUT},

{"gpmc\_ad6.gpio1\_6", OMAP\_MUX\_MODE7 | AM33XX\_PIN\_INPUT},

{"gpmc\_ad7.gpio1\_7", OMAP\_MUX\_MODE7 | AM33XX\_PIN\_INPUT},

{NULL, 0},

};

**改成：**

static struct pinmux\_config gpio\_led\_mux[] = {

{"gpmc\_a5.gpio1\_21", OMAP\_MUX\_MODE7 | AM33XX\_PIN\_INPUT},

{"gpmc\_a6.gpio1\_22", OMAP\_MUX\_MODE7 | AM33XX\_PIN\_INPUT},

{"gpmc\_a7.gpio1\_23", OMAP\_MUX\_MODE7 | AM33XX\_PIN\_INPUT},

{"gpmc\_a8.gpio1\_24", OMAP\_MUX\_MODE7 | AM33XX\_PIN\_INPUT},

{NULL, 0},

};

并在static struct evm\_dev\_cfg beaglebone\_dev\_cfg[] 中添加

**2增加7寸液晶驱动以及触摸屏驱动**

**增加液晶驱动需要做的：**

1 添加对应的struct da8xx\_lcdc\_platform\_data

2修改文件中 disp\_panel与lcd\_cfg的值

3 在da8xx-fb.c中修改液晶时序参数

4 在static struct evm\_dev\_cfg beaglebone\_dev\_cfg[] 中添加对应初始化函数

5 修改 bone\_setup\_daughter\_board函数的内容

需要玩转android，还得加入按键。按键的驱动可以修改lcd\_cape\_gpio\_buttons的对应管脚来实现。

以上内容代码太多，就不在此粘贴复制了 。

改好之后就可以编译内核了。

$ make ARCH=arm CROSS\_COMPILE=arm-eabi- distclean

$ make ARCH=arm CROSS\_COMPILE=arm-eabi- am335x\_evm\_android\_defconfig

$ make ARCH=arm CROSS\_COMPILE=arm-eabi- uImage

编译的时间大概要20分钟左右，看你的机器了。

后面就制作SD卡了。

建议从<http://software-dl.ti.com/dsps/dsps_public_sw/sdo_tii/TI_Android_DevKit/TI_Android_ICS_4_0_3_DevKit_3_0_1/index_FDS.html>

下载Pre-built Image栏下的[Beaglebone](http://software-dl.ti.com/dsps/dsps_public_sw/sdo_tii/TI_Android_DevKit/TI_Android_ICS_4_0_3_DevKit_3_0_1/exports/beaglebone.tar.gz)

然后 tar -xzvf Beaglebone.tar.gz

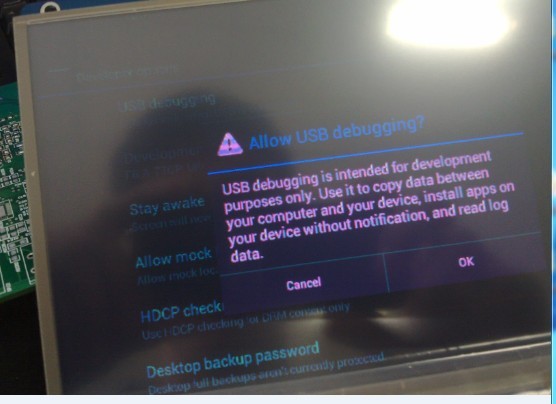
然后把之前编译好的U-boo/.下的MLO u-boot.img 以及 kernel/arch/arm/boot下的uImage复制到刚解压出来的Boot\_Images/目录中

然后 cd beaglebone

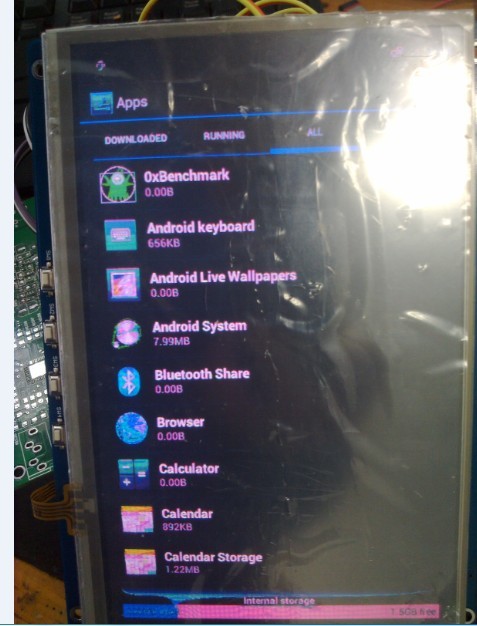
./mkmmc-android.sh <你的SD卡盘符>

确认后，大概5分钟就制作好了。

看看启动后的效果吧：

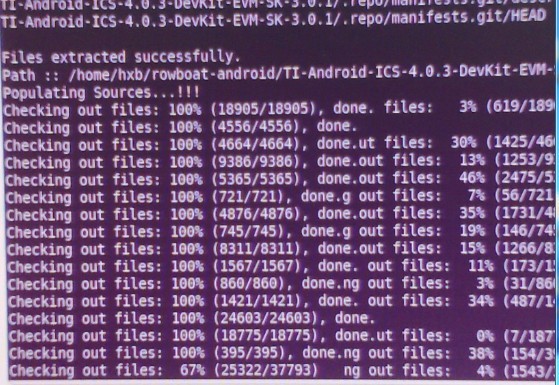






**ges**

补一张编译安装源码包的截图吧.



PS：我正在做一个基于beaglebone的遥控终端，很高兴能在截至前参加这次活动，由于第一个打样的板子用的不是这个型号的液晶，因此这个液晶是用杜邦线连的。附上一份我硬件电路图的设计经验分享：