Android 系統開發流程 - Beagle 開發板之移植

張翊帆 林盈達 張尚揚
國立交通大學資訊科學系
300 新竹市大學路 1001 號
Tel: 03-5712121 ext. 56667
E-MAIL：evanchang.pcs00g@g2.nctu.edu.tw, ydlin@cis.nctu.edu.tw,
csyang@cs.nctu.edu.tw
October 2012

摘要

Android 是以 Linux 為基礎的半開放式作業系統，主要用於手持式裝置[1]。BeagleBoard 是低功耗的單晶片計算機[2]，其具有 open source hardware 的特色因而被廣泛用於學術研究及教學方面。Android 需要透過系統移植的方式，使其能夠在 BeagleBoard 上運作，相對於應用程式開發文件的完整性，Android 並沒有提供完善的開發文件，相對增加了移植的困難度。國外有名為 rowboat[3]的專案，專門維護及開發 BeagleBoard 上可運作的 Android。本文中移植的平台除 BeagleBoard 外還有拓展板(expansion board)的整合，其搭配有 LCD、Touch Panel 及 Wi-Fi 等硬體設備。因此，在本文中將說明何以在眾多版本中選擇 Android 4.0 版進行移植，如何透過六個步驟將 Android 移植到 BeagleBoard 上，如何修改驅動程式以整合 expansion board 的硬體模組，包括編譯環境及目標平台、Anroid 系統架構以及硬體規格與驅動程式。完整的敘述移植的步驟、流程、及修改的程式碼，讓初接觸 Android 的系統開發者也能夠依照此流程來建立 Android 的執行環境，並讓系統能夠正常運作。

1 簡介

Android 系統介紹

為什麼 Android 發展如此快速？根據 Nielsen 調查[4]，Android 在智慧型手機市場的市佔率約為 51.8%，如圖一，較之封閉式作業系統 iOS 的 34.3%為高，且至今世界上已有超過 190 個國家、數億部手持式裝置搭載 Android 系統[5]。除應用程式具有可攜性外，半開放式作業系統的特色也受到軟硬體開發廠商的青睞。其除了具有一般應用在手持式裝置作業系統的特色，包括有限資源、節電之外，更增加了 Android HAL(Hardware Abstract Layer)、Dalvik 虛擬機，前者可將 Android framework 與 Linux kernel 隔開，使得 Android 能在移動設備上獲得更高的執行效率；後者可支援轉換為 Dalvik Executable 的 Java 應用程式的執行，讓

應用程式可以在不同的實體機器上執行，而不需經過重新編譯的動作。這些特色使得 Android 被搭載在許多不同的手持式裝置上，使用狀態也更多元。

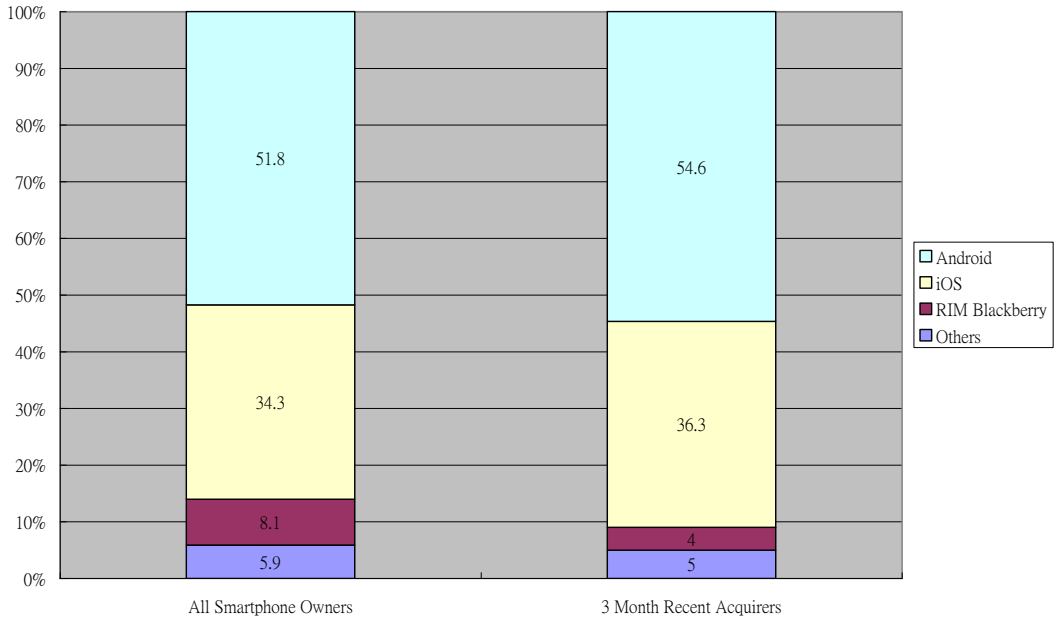

圖 一：智慧型手機系統分佈圖

Android 版本

自 2008 年 9 月 23 日 Android 官方版本 1.0 版 Astro 的發行，迄今共有 10 個正式版本，以約每半年一次的步伐進行升級，分別以 A、B、C、D、E、F、G、H、I、J 的英文字首順序來排列，參照表一。由於 Android 平台的高度開放和對軟體升級架構尚欠規範，導致 Android 產品往往因新版軟體的硬體要求過高，或致設備廠商無法保證對其產品提供最新版軟體更新，或致廠商拒絕支援而無法更新，令產品壽命週期大幅縮短。

表 一：Android 版本歷史

版本	代號	發布日期
1.0	Astro	2008/09/23
1.1	Bender	2009/02/02
1.5	Cupcake	2009/04/30
1.6	Donut	2009/09/15
2.0	Éclair	2009/10/26
2.2	Froyo	2010/05/20
2.3	Gingerbread	2010/12/06
3.0	Honeycomb	2011/02/22
4.0	Ice Cream Sandwich	2011/10/19

目前市佔率最高的版本為 2.3 版 Ginger Bread 的 57.5%及 4.0 版 Ice Cream Sandwich 的 20.9%，參照圖二。

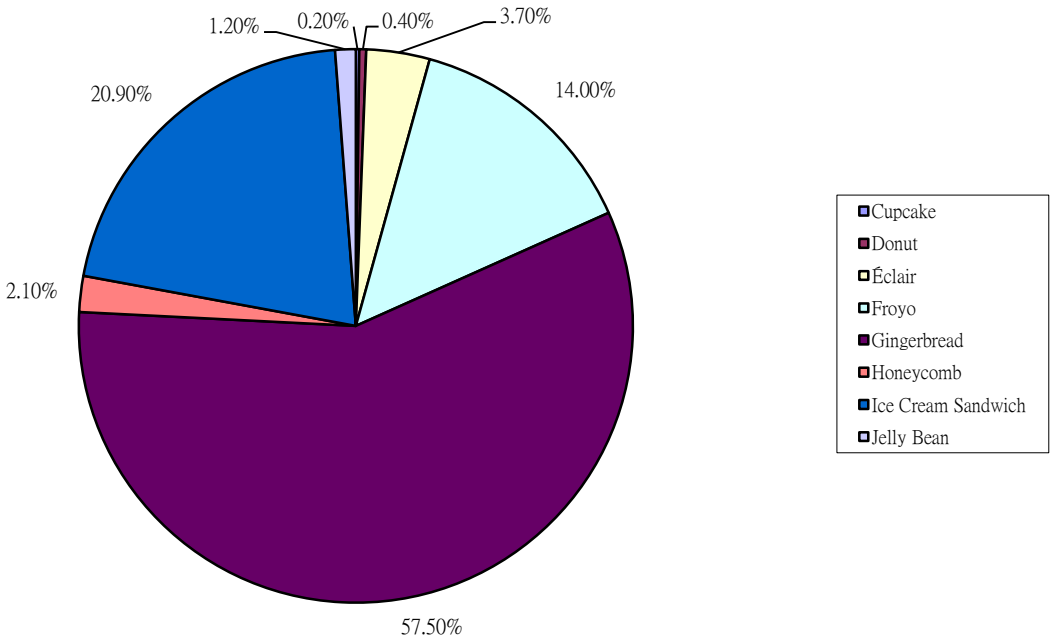

圖 二：不同版本用戶比例

實驗平台 BeagleBoard-xM Rev. C

本文所使用的實驗平台為 BeagleBoard-xM 是由 BeagleBoard 的更新，自 2010 年 8 月 27 日開始銷售，其規格如表三。使用的處理器為 ARM Cortex-A8，記憶體為 512MB DDR RAM，更搭載 SGX 圖像處理器，使得作業系統及應用程式能更有效率的運行。周邊裝置包含 HDMI Tx、RS-232、Ethernet、USB 等。方便開發者能夠進行通訊、查看資訊，更能方便除錯的進行。BeagleBoard 的外觀如表二。

表 二：BeagleBoard-xM 規格

Specifications	Descriptions
Package on Package POP CPU/memory chip.	● Processor TI DM3730 Processor - 1 GHz ARM Cortex-A8 core ● 'HD capable' TMS320C64x+ core (800 MHz up to 720p @30 fps) ● Imagination Technologies PowerVR SGX 2D/3D graphics processor supporting dual independent displays

	● 512 MB LPDDR RAM ● 4 GB microSD card is supplied with the BeagleBoard -xM loaded with Angstrom.
Peripheral connections	● DVI-D (HDMI connector chosen for size - maximum resolution is 1400x1050) ● S-Video ● USB OTG (mini AB) ● 4 USB ports ● Ethernet port ● MicroSD/MMC card slot ● Stereo in and out jacks ● RS-232 port ● JTAG connector ● Power socket (5 V barrel connector type) ● Camera port ● Expansion port

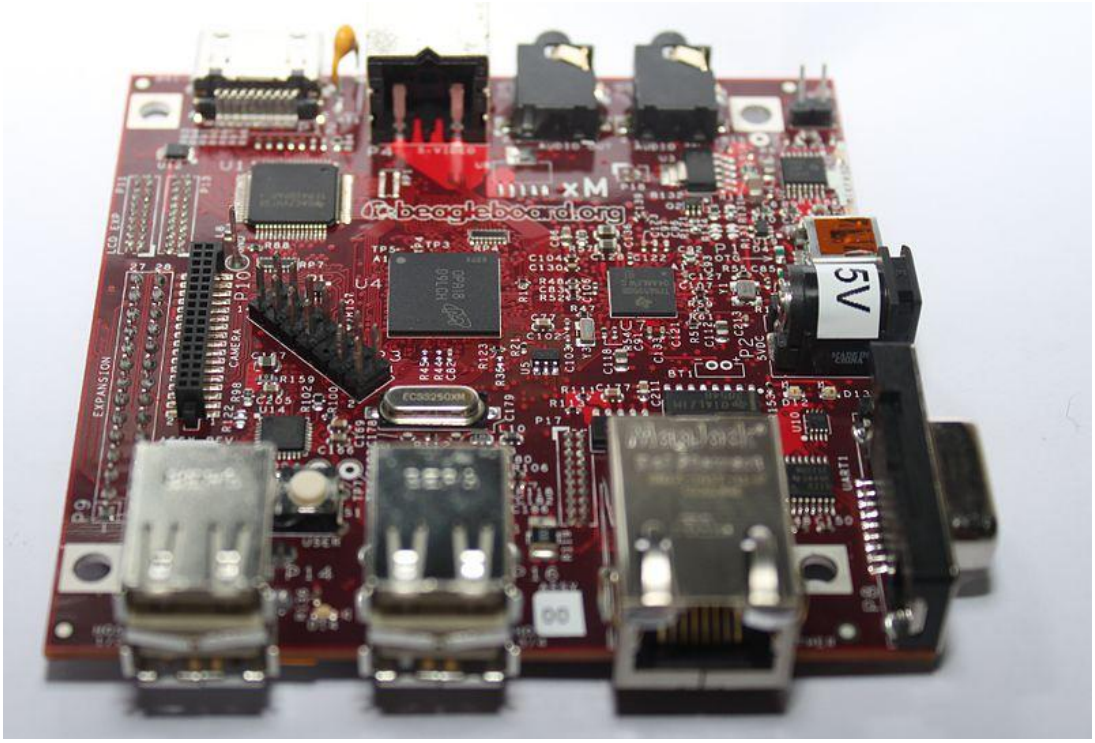

圖 三：BeagleBoard-xM

圖片來源: http://upload.wikimedia.org/wikipedia/commons/8/8e/BeagleBoard_xM.JPG

除 BeagleBoard 外，實驗平台尚有搭配拓展板 Beagleboard-xM LCD Touch Panel Expansion Module V2 (LTM V2)，其規格如表三[6]。外觀如圖四。

表 三：LTM V2 規格

Specifications	Descriptions

Hardware	● 7.0 inch LCD,800*480 pixel ● Four- wire resistance touch panel ● Wi-Fi ● 3-Axis Digital Accelerometer ● Five user buttons which Android system always need

圖 四：LTM V2

圖片來源:

http://www.catcan.com.tw/_/rsrc/1317608664158/products/beagle-series/beagleboard-xm-ltm-v2/IMG_4322%282%29.jpg?height=200&width=188

根據用戶使用比例及 rowboat 專案所提供 Android 版本的支援，以及拓展板有提供支援 Gingerbread 及 Froyo 的更新的前提下，本文選定以 Android 4.0 版 Ice Cream Sandwich 做為主要的移植版本。

2 移植要點

Android 移植至 BeagleBoard 之需求

移植 Android 作業系統需要更動哪些程式？基本上，根據不同硬體平台，首先需要考慮的就是不同處理器的機器語言都不相同，因此針對不同處理器，我們將會需要跨平台的編譯器。Android 提供的原始碼中，也有提供已編譯好的工具，方便實現不同處理器的編譯問題，可以在專案根目錄下的 prebuild 資料夾中看到，路徑如下。

/prebuilt/linux-x86/toolchain/arm-eabi-4.4.3/bin

其次，必須考慮硬體驅動程式的移植，以 BeagleBoard 而言，rowboat 專案已提供對應的驅動程式移植。那麼，本文的移植重點放在何處？rowboat 專案所

提供的原始碼只針對 BeagleBoard-xM，而文中所使用的 LTM expansion board 並不在專案支援的範圍內，諸如 7 吋的 LCD 顯示、Wi-Fi 的使用、Touch panel 等，由於沒有合適的硬體驅動程式，這些硬體模組在實驗過程中將會完全失效。

註：雖然作業系統為 64 位元，但 Android 所提供的 cross compile 仍只有 32 位元的版本，所以在此我們以 32 位元的 corss compile 作示範

BeagleBoard-xM expansion LTM

因此，本文將首先介紹 rowboat 專案的編譯及安裝過程，待安裝完畢之後將可看到初步的結果。由於硬體驅動的關係，實驗過程將會修正相關的程式碼，經過重新編譯及安裝，將可看到修正後的結果。本文將以 LCD 移植為例，介紹 Android 作業系統的移植要點。

3 移植方法

安裝平台套件及工具

文中的實驗平台選定 Ubuntu LTS (10.04) 64-bit 作業系統做為操作平台，目標平台則為 BeagleBoard-xM 及 LTM。根據 Android Developers[7]的說明，首先安裝編譯所需的套件。在此之前，必須先安裝 Oracle JDK 6[8]，需要下載的檔案為 jdk-6u35-linux-x64-rpm.bin。根據以下的說明安裝[9]到 Ubuntu 中。

```
$ chmod +x 6u <version>-linux-x64.bin
$ ./jdk-6u <version>-linux-x64.bin
$ mv jdk1.6.0_<version> java-6-oracle
$ sudo mkdir /usr/lib/jvm
$ sudo mv java-6-oracle /usr/lib/jvm
$ wget http://webupd8.googlecode.com/files/update-java-0.5b
$ chmod +x update-java-0.5b
$ sudo ./update-java-0.5b
```

接著，需要安裝編譯過程所需的套件。這些套件包括版本控制軟體、編譯過程中所需要的軟體庫(Library)、映像檔(Inage)製作軟體等編譯必備的套件。

```
$ 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
lib32readline5-dev lib32z-dev libgl1-mesa-dev g++-multilib mingw32 tofrodos
python-markdown libxml2-utils xsltproc libsdl-dev libesd0-dev libwxgtk2.6-dev
minicom tftpd uboot-mkimage expect
```

註: 雖然官方建議平臺為 Ubuntu，但經筆者實際操作，此操作流程確實可實現在 CentOS 此 Linux 發行版，也就是說，Redhat 或 Fedora 等相關 Linux 也可行此 SOP 流程。唯安裝指令為 yum，且相依套名稱件略有不同，在此不贅述。

使用 Git 下載 rowboat 提供的原始碼及工具，或是使用 rowboat 封裝好的原始碼[10]。

```
$ mkdir ~/bin
$ PATH=~/bin:$PATH
$ curl https://dl-ssl.google.com/dl/googlesource/git-repo/repo > ~/bin/repo
$ chmod a+x ~/bin/repo
$ mkdir ~/rowboat-android
$ cd ~/rowboat-android
$ repo init -u git://gitorious.org/rowboat/manifest.git -m
TI-Android-ICS-4.0.3-DevKit-3.0.0.xml
$ repo sync

OR

$ mkdir ~/rowboat-android
$ cd ~/rowboat-android
$ wget
http://software-dl.ti.com/dsps/dsps_public_sw/sdo_tii/TI_Android_DevKit/TI_Android_ICS_4_0_3_DevKit_3_0_0/exports/TI-Android-ICS-4.0.3_AM37x_3.0.0.bin
$ chmod a+x TI-Android-ICS-4.0.3_AM37x_3.0.0.bin
$ ./TI-Android-ICS-4.0.3_AM37x_3.0.0.bin
```

設定 Toolchain 的路徑，這裡需要注意 rowboat 專案的位置及子目錄路徑是否與下列指令一致，必要時請修改相關的路徑。

```
$ export
PATH=$HOME/rowboat-android/prebuilt/linux-x86/toolchain/arm-eabi-4.4.3/bin:$PATH
```

編譯及安裝六大步驟

步驟一：x-load

x-load 是簡化版的 u-boot，負責初始化外部的 RAM Controller，並將 u-boot 從 NAND/MMC 讀出至外部的 RAM，最後將控制權轉移至 u-boot。

```
$ cd ~/rowboat-android/x-loader
$ make CROSS_COMPILE=arm-eabi- distclean
$ make CROSS_COMPILE=arm-eabi- omap3beagle_config
$ make CROSS_COMPILE=arm-eabi-
```

編譯完成後會在~/rowboat-android/x-loader/產生 x-load.bin 檔，這裡需要額外的工具 signGP，可由以下指令下載及使用。

```
$ cd ~
$ git clone git://gitorious.org/rowboat/external-ti_android_utilities.git
external-ti_android_utilities
$ cd –
$ cp ~/external-ti_android_utilities/signGP/signGP ./
$ ./signGP ./x-load.bin
$ mv x-load.bin.ift MLO
```

步驟二：uboot

主要用於嵌入式系統的開機載入程式，負責載入 Android kernel 並將控制權轉移到作業系統。編譯完成後會在~/rowboat-android/u-boot/產生 u-boot.bin 檔。

```
$ cd ~/rowboat-android/u-boot
$ make CROSS_COMPILE=arm-eabi- distclean
$ make ARCH=arm CROSS_COMPILE=arm-eabi- omap3_beagle_config
$ make ARCH=arm CROSS_COMPILE=arm-eabi-
```

步驟三：kernel

Android 作業系統，在接管自 u-boot 移轉的控制權後，負責載入驅動程式、管理並排程應用程式，做為硬體及應用程式間的溝通管道。最後會在 /~/rowboat-android/arch/arm/boot/中產生 uImage 檔

```
$ cd ~/rowboat-android/kernel
$ make ARCH=arm CROSS_COMPILE=arm-eabi- distclean
$ make ARCH=arm CROSS_COMPILE=arm-eabi-
```

```
omap3_beagle_android_defconfig
$ make ARCH=arm CROSS_COMPILE=arm-eabi- uImage
```

步驟四：Android Filesystem

檔案系統是組織系統內檔案及資料的架構，讓檔案的存取變得容易，並維護檔案在裝置中的物理位置，或被當成存取資料的界面。Android 檔案系統的架構與 Linux 檔案系統架構相似。

```
$ cd ~/rowboat-android/
$ make TARGET_PRODUCT=beagleboard OMAPES=5.x -j<N>
```

- <N>是操作平台上處理器數目的兩倍。例如系統 CPU 數量為 2，或者單一 CPU 但核心數為 2，則鍵入 make TARGET_PRODUCT=beagleboard OMAPES=5.x –j4

```
$ cd out/target/product/beagleboard
$ mkdir android_rootfs
$ cp -r root/* android_rootfs
$ cp -r system android_rootfs
$ ../../../../build/tools/mktarball.sh ../../../host/linux-x86/bin/fs_get_stats
android_rootfs . rootfs rootfs.tar.bz2
```

步驟五：boot script

在作業系統的開機過程中，會需要某些裝置的設定資訊，通常為裝置的初始設定值，又某些裝置只允許在開機過程中設定，如 DSS。命令列參數請參閱表四到表六。

```
$ cd ~/external-ti_android_utilities/mk-bootscr
$ vi mkbootscr
```

修改相應內容為：

```
setenv bootargs 'console=ttyO2,115200n8 androidboot.console=ttyO2 mem=256M
root=/dev/mmcblk0p2 rw rootfstype=ext4 rootdelay=1 init=/init ip=off
omap_vout.vid1_static_vrfb_alloc=y vram=8M omapfb.vram=0:8M
omapdss.def_disp=dvi omapfb.mode=dvi:1024x768-16'
```

```
$ ./mkbootscr
```

表 四：V4L2 Driver Command Line Arguments

Argument	Description
video1_numbuffers	Number of buffers to be allocated at init time for Video1 device
video2_numbuffers	Number of buffers to be allocated at init time for Video2 device
video1_bufsize	Size of the buffer to be allocated for video1 device
video2_bufsize	Size of the buffer to be allocated for video2 device
vid1_static_vrfb_alloc	Static allocation of the VRFB buffer for video1 device
vid2_static_vrfb_alloc	Static allocation of the VRFB buffer for video2 device
debug	Enable debug messaging

表 五：FBDEV Driver Command Line Arguments

Argument	Description
mode	Default video mode for specified displays
vram	VRAM allocated memory for a framebuffer, user can individually configure VRAM buffers for each plane/device node
vrfb	Use VRFB rotation for framebuffer
rotate	Default rotation applied to framebuffer
test	Draw test pattern to framebuffer whenever framebuffer settings chang
debug	Enable debug messaging

表 六：DSS Library Command Line Arguments

Argument	Description
def_disp	Name of default display, to which all overlays will be connected
debug	Enable debug messaging

步驟六：寫入 SD 卡

```
$ mkdir ~/image_folder
$ cp ~/rowboat-android/kernel/arch/arm/boot/uImage ~/image_folder
$ cp ~/rowboat-android/u-boot/u-boot.bin ~/image_folder
$ cp ~/rowboat-android/x-loader/MLO ~/image_folder
$ cp ~/external-ti_android_utilities/mk-bootscr/boot.scr ~/image_folder
```

```
$ cp ~/rowboat-android/out/target/product/beagleboard/rootfs.tar.bz2
~/image_folder
$ cp ~/external-ti_android_utilities/mk-mmc/mkmmc-android.sh ~/image_folder
$ cd ~/image_folder
$ sudo ./mkmmc-android /dev/sd<device-name> MLO u-boot.bin uImage boot.scr
rootfs.tar.bz2
```

- SD 卡裝置的代碼，<device-name>可能為 b 或 c，視 mount point 決定。

  註 1: 若在編譯時，遇到錯誤訊息" /bin/bash^m bad interpreter no such file or directory", 請先鍵入" fromdos mkmmc-android.sh"指令。若在 CentOS/Fedora 系統，則鍵入" dos2unix mkmmc-android.sh"指令

  註 2：正確裝置代碼可在 Linux 下由"df"指令查得

透過以上編譯及安裝步驟將 Android 開機所需的檔案寫入 SD 卡中，由 SD 卡開機後可由 HDMI 輸出到外部螢幕作為顯示，同時也可由 LTM 的 LCD 觀察開機畫面，此即為實驗得到的第一個結果，參照圖五及圖六。由此結果可觀察到，輸出的畫面與 LCD 的解析度並不相容，此為 rowboat 專案所提供的驅動程式所致，其標準輸出為 HDMI 1024x768 的解析度，與 LTM 的 LCD 螢幕解析度 800x480 有所出入，因此，本文將接著介紹如何修改相關的原始碼以提供一個合適的實驗平台。

圖 五：開機畫面

圖 六：執行畫面

驅動程式修正

Android Display Subsystem 介紹

移植 Android 的顯示驅動程式之前，需要先對 Android 的 Display Subsystem, DSS 有所瞭解。Android 的顯示基本上與 Linux 相同，採用 Frame buffer 的方式輸出。在圖七[11]的 Linux kernel 層中可以看到，Frame buffer Driver 及 Video Drivers 都是經由 DSS 與 Panel driver 溝通，再經由實體 Hardware 層輸出到螢幕上，換句話說，無論是應用程式或多媒體的畫面，都是經由同樣的方式輸出。在圖八[12]的 Android 驅動程式架構中可以更顯著的觀察到。

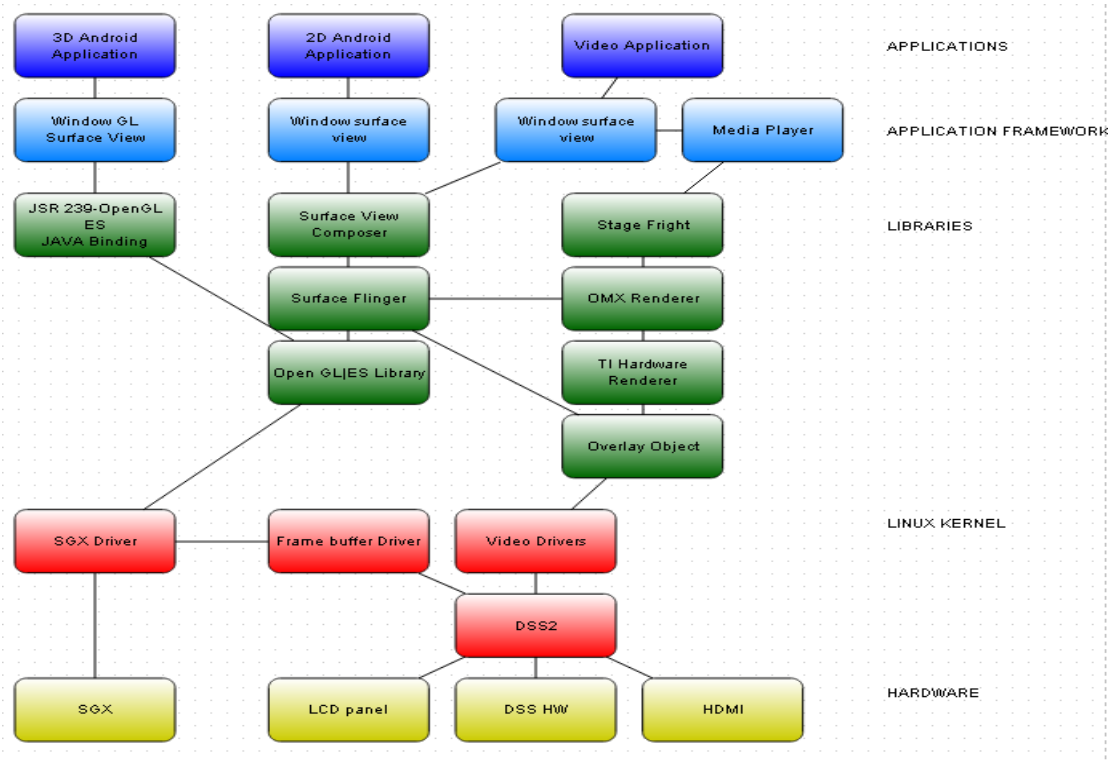

圖 七：Android Display Subsystem

驅動程式架構

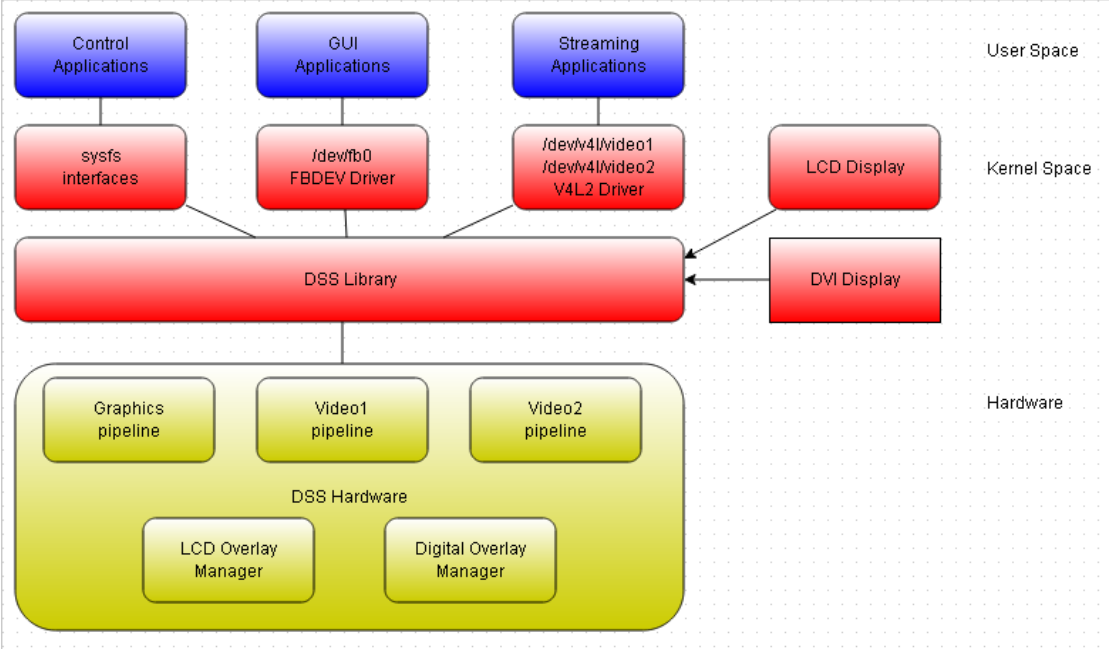

圖 八：OMAP3x Display Subsystem Architecture

接下來的步驟將是進行修正驅動程式原始碼，更動其 timing 設定時所需要的資訊，以達到合適解析度的結果，其路徑如下：

/kernel/drivers/video/omap2/displays/panel-generic.c

原始內容：

```
static struct omap_video_timings generic_panel_timings = {
        /* 640 x 480 @ 60 Hz    Reduced blanking VESA CVT 0.31M3-R */
        .x_res          = 640,
        .y_res          = 480,
        .pixel_clock    = 23500,
        .hfp            = 48,
        .hsw            = 32,
        .hbp            = 80,
        .vfp            = 3,
        .vsw            = 4,
        .vbp            = 7,
};
```

修正內容：

```
static struct omap_video_timings generic_panel_timings = {
        /* 7 inch LCD */
        .x_res          = 800,
```

```
        .y_res          = 480,
        .pixel_clock    = 36000,
        .hfp            = 1,
        .hsw            = 48,
        .hbp            = 46,
        .vfp            = 12,
        .vsw            = 3,
        .vbp            = 23,
};
```

修正 boot script

```
setenv bootargs 'console=ttyO2,115200n8 androidboot.console=ttyO2 mem=256M
root=/dev/mmcblk0p2 rw rootfstype=ext4 rootdelay=1 init=/init ip=off
omap_vout.vid1_static_vrfb_alloc=y vram=8M omapfb.vram=0:8M
omapdss.def_disp=dvi omapfb.mode=800x480-16@60 mpurate=1000'
```

修正上述檔案之後，只需要分別重新編譯 kernel 及重新產生 boot script，再把這兩個檔案複製到 image_folder 中，以同樣的方式寫入 SD 卡，接著將 SD 卡插入 BeagleBoard 再按下 reset 按鈕，即可觀察到實驗的最終結果。

4 實驗結果及結論

在 Android 的移植過程中，需要把握幾個重點：**編譯環境及目標平台**、**Anroid 系統架構**以及**硬體規格與驅動程式**。針對目標平台所使用的處理器，利用跨平台編譯器(Cross-Compiler)編譯出符合目標平台的可執行檔，本實驗中所使用的目標平台處理器為 ARM，因此本文也必須使用 Anroid 提供的跨平台編譯器進行匯編的動作。為了使目標平台上的硬體能夠正常運作，除了需要瞭解 Android 的系統架構外，也需要瞭解硬體規格，才有辦法修改對應的驅動程式。本實驗藉由敘述實際修改的過程，讓系統開發者有所參考，其運作畫面如圖九及圖十。不過此修正僅使螢幕顯示正常，操控方面尚無觸控功能，仍需仰賴其它輸入工具(如 USB 滑鼠)。觸控螢幕及 Wi-Fi 模組的修正過程與方式和前文所敘述的方法相似，在經過修正驅動程式之後重新編譯 Android 核心即可達成，蓋因本文敘述重點為過程之導引，因此不再贅述。

圖 九：修正後的開機畫面

圖 十：修正後的執行畫面

5 參考文獻

[1] Android (operating system):
http://en.wikipedia.org/wiki/Android_(operating_system)

[2] BeagleBoard:
http://en.wikipedia.org/wiki/BeagleBoard

[3] arowboat.org
http://code.google.com/p/rowboat/

[4] Two Thirds of New Mobile Buyers Now Opting For Smartphones:
http://blog.nielsen.com/nielsenwire/online_mobile/two-thirds-of-new-mobile-buyers-now-opting-for-smartphones/

[5] Android, the world's most popular mobile platform:
http://developer.android.com/about/index.html

[6] BeagleBoard-xM LTM V2:
http://www.catcan.com.tw/products/beagle-series/beagleboard-xm-ltm-v2

[7] Initializing a Build Environment:
http://source.android.com/source/index.html

[8] Java SE Downloads:
http://www.oracle.com/technetwork/java/javase/downloads/jdk6u35-downloads-1836443.html

[9] JDK Self-Extracting Installation for Linux (64-bit):
http://www.oracle.com/technetwork/java/javase/install-linux-64-self-extracting-142068.html

[10] TI-Android-ICS-4.0.3-DevKit-3.0.0 DevelopersGuide:
http://processors.wiki.ti.com/index.php/TI-Android-ICS-4.0.3-DevKit-3.0.0_DevelopersGuide

[11] Android Display Subsystem Overview:
http://processors.wiki.ti.com/index.php/TI-Android-GingerBread-2.3.4-DevKit-2.1_PortingGuides#Android_Display_Subsystem_Overview

[12] Driver Architecture:
http://processors.wiki.ti.com/index.php/UserGuideDisplayDrivers_PSP_04.02.00.07#Driver_Architecture

[13] Android 作業系統移植之研究與實現, 鍾文昌、梁文耀 2009