**APPENDIX I**

**Building Raspberry Pi Kernel for Debian Wheezy / Raspbian with eGalax Touchscreen Support on Ubuntu 12.10**

To run this download Ubuntu 12.10.  
  
Install the package dependencies, git and the cross-compilation toolchain:  
  
*sudo apt-get install git-core gcc-4.6-arm-linux-gnueabi  
  
sudo ln -s /usr/bin/arm-linux-gnueabi-gcc-4.6 /usr/bin/arm-linux-gnueabi-gcc*  
Get the source code for the Pi kernel, and some tools. There are two ways to get this, either via git or by downloading an archive  
  
*mkdir raspberrypi  
cd raspberrypi*  
by git:  
*git clone https://github.com/raspberrypi/tools.git  
git clone https://github.com/raspberrypi/linux.git  
cd linux*  
by archive download:  
*wget https://github.com/raspberrypi/linux/archive/rpi-3.10.y.tar.gz  
wget https://github.com/raspberrypi/tools/archive/master.tar.gz*  
extract the archives (*tar -zxvf filename*) and rename the *linux-\** source directory so it is called ***linux***, and the master-tools to be called tools.  
  
Next generate a .config file. (*cd into the linux dir*)  
*make ARCH=arm CROSS\_COMPILE=/usr/bin/arm-linux-gnueabi- bcmrpi\_defconfig*(you can substitute bcmrpi\_cutdown\_defconfig for bcmrpi\_defconfig to create a smaller kernel, if you care.)  
  
To make changes to the configuration, run make menuconfig  
*make ARCH=arm CROSS\_COMPILE=/usr/bin/arm-linux-gnueabi- menuconfig*  
  
(Install the curses lib to get the menu working *sudo* *apt-get install libncurses5-dev*)  
From this menu select ***Device Drivers->Input Device Support->TouchScreens***.

Select (press space bar) so that Touchscreens has a \* beside it, press enter for a further submenu. In this menu select **USB touchscreen Driver** which contains the driver for the eGalax touchscreen as one of its sub-components. *NOTE: there may be other eGalax entries in the list above this* ***do not*** *select 'eGalax multitouch'.*  
  
Once these selections are made press *ESC-ESC* to get out or select *exit* from the bottom. At the *yes/no* prompt, select ***yes*** to save the config.  
  
The build is now ready to start. Speed up the compilation process by enabling parallel make with the *-j flag*. The recommended use is ‘processor cores + 1', e.g. 3 if a dual core processor is being used:  
*make ARCH=arm CROSS\_COMPILE=/usr/bin/arm-linux-gnueabi- -k -j3*  
  
This process can range from 20 minutes to several hours.

Assuming the compilation was successful, create a directory for the modules:  
  
*mkdir ../modules*  
  
Now compile and ‘*install*’ the loadable modules to the temp directory:  
  
*make modules\_install ARCH=arm CROSS\_COMPILE=/usr/bin/arm-linux-gnueabi- INSTALL\_MOD\_PATH=../modules/*  
  
*imagetool-uncompressed.py* from the tools repo must now be used to get the kernel ready for the Pi.  
  
*cd ../tools/mkimage/  
./imagetool-uncompressed.py ../../linux/arch/arm/boot/Image*  
  
This creates a kernel.img in the current directory. Plug in the SD card of an existing Debian image to install the new kernel on. Delete the existing kernel.img and replace it with the new one, substituting “*boot-partition-uuid*” with the identifier of the partition as it is mounted in Ubuntu.  
  
*sudo rm /media/boot-partition-uuid/kernel.img  
sudo mv kernel.img /media/boot-partition-uuid/*  
Remove the existing */lib/modules* and *lib/firmware* directories, substituting “*rootfs-partition-uuid*” with the identifier of the root filesystem partion mounted in Ubuntu.  
  
*sudo rm -rf /media/rootfs-partition-uuid/lib/modules/  
sudo rm -rf /media/rootfs-partition-uuid/lib/firmware/*  
Go to the destination directory of the previous *make modules\_install*, and copy the new modules and firmware in their place:  
  
*cd ../../modules/  
sudo cp -a lib/modules/ /media/rootfs-partition-uuid/lib/  
sudo cp -a lib/firmware/ /media/rootfs-partition-uuid/lib/  
sync*  
Eject the SD card, and boot the new kernel on the Raspberry Pi.