Configure these variables to match your system

```
export DISKNAME=RasPI                                          # Name of disk or
folder where we'll store the files
export DISKMOUNTLOCATION=/Volumes/${DISKNAME}                  # Path to folder
where we'll store the files
export BUILDDISK=${DISKMOUNTLOCATION}/gateway_rpi_kernel       # Root dir for
source tree and binaries
export BUILDSOURCES=${BUILDDISK}/raspberrypi-linux             # Location of
source tree
export SOURCECOMMITID=9301f9aff699bc90c194e41c3fa6117a36c4954a # Commit ID for
kernel source tree
export BUILDOUTPUT=${BUILDDISK}/buildFiles                     # Output of build
process
export RASPBIANIMAGEDOWNLOADLOCATION=~/Documents/RasPI-Images  # Location of
full Raspbian images
export PIIPADDRESS=192.168.16.122                              # IP address of
Raspberry PI
```

Set up build toolchain

MAC Version

Assumes the MAC toolchain based on MacPorts has been installed.

See ../MAC_toolchain/MAC_toolchain.pdf

MAC binutils are located in /opt/local

Set the Cross-compiler prefix:

```
export CCPREFIX=/opt/local/bin/arm-none-eabi-
```

Create a case sensitive disk

This is needed for the MAC, not for Ubuntu

```
cd ~/Documents/RasPI   # Go to a place where you will create the case-sensitive
disk image
hdiutil create -fs HFSX -size 16G -type SPARSEBUNDLE -volname ${DISKNAME}
${DISKNAME}
hdiutil attach ${DISKNAME}.sparsebundle/ -owners on
```

Now doubleclick on the RasPI.sparsebundle file from inside the MAC Finder to mount it

Ubuntu Version

Install the ARM cross-compile toolchain using the following

```
sudo apt-get install gcc-arm-none-eabi
```

Set the Cross-compiler prefix:

```
export CCPREFIX=/usr/bin/arm-none-eabi-
```

Get and update sources

Get the RasPI soures

```
cd ${BUILDDISK}

git clone https://github.com/raspberrypi/linux ${BUILDSOURCES}/
cd ${BUILDSOURCES}/
```

The following is optional, but will reset your clone to the version that was used for this Howto

```
git checkout ${SOURCECOMMITID}
```

Verify the kernel version

```
export KERNELVERSION=`make kernelversion`
echo $KERNELVERSION
```

You should get this response:

```
3.12.36
```

Get the TFT framebuffer files

Download the source code from adafruit.com

```
cd ${BUILDSOURCES}/drivers/video/
git clone https://github.com/adafruit/adafruit-rpi-fbtft.git
cd ${BUILDSOURCES}/
```

Patch source files

Fix some bugs

Make menuconfig failed with

```
HOSTLD  scripts/kconfig/mconf
Undefined symbols for architecture x86_64:
```

Fix this as following:

```
nano scripts/kconfig/Makefile
```

Find the line with HOSTLOADLIBES_mconf and add the tinfo library dependency

```
HOSTLOADLIBES_mconf   = $(shell $(CONFIG_SHELL) $(check-lxdialog) -ldflags
$(HOSTCC))
HOSTLOADLIBES_mconf += -ltinfo
```

Patch a bug in the CP210X driver

```
nano drivers/usb/serial/cp210x.c
```

Look for

```
{ USB_DEVICE(0x10C4, 0x8875) }, /* CEL MeshConnect USB Stick */
```

Change it to:

```
{ USB_DEVICE(0x10C4, 0x8856) }, /* CEL EM357 ZigBee USB Stick - LR */
{ USB_DEVICE(0x10C4, 0x8857) }, /* CEL EM357 ZigBee USB Stick */
{ USB_DEVICE(0x10C4, 0x8977) }, /* CEL MeshWorks DevKit Device */
```

Add Support for the TFT framebuffer drivers

Add fbtft support in the Kconfig file

```
  nano ${BUILDSOURCES}/drivers/video/Kconfig
```

Add the following line below the first few "source drivers/XXXXX" lines

```
  source "drivers/video/adafruit-rpi-fbtft/Kconfig"
```

Modify the Makefile

```
nano ${BUILDSOURCES}/drivers/video/Makefile
```

Add the following line: (anywhere should be fine, but add it below the other "obj-y" lines)

```
obj-y                           += adafruit-rpi-fbtft/
```

Compiling the fbtft module will crash complaining about a missing <linux/input/ft6x06_ts.h> header. All we need for this to compile is the header file, so let's just get it and add it to the tree:

```
cd ${BUILDSOURCES}/include/linux/input
wget https://github.com/adafruit/adafruit-raspberrypi-linux/raw/rpi-
3.10.y/include/linux/input/ft6x06_ts.h
```

Customize the drivers for the display

```
cd ${BUILDSOURCES}
```

Create a clean config file for a bcm rpi device (the Raspberry PI)

```
make ARCH=arm CROSS_COMPILE=$CCPREFIX bcmrpi_defconfig
```

Run the menu config

```
make ARCH=arm CROSS_COMPILE=$CCPREFIX menuconfig
```

Enable LCD module support in the Kernel

```
Go to "Device Drivers"
    ==> "Graphics support"
    Select "Support for small TFT LCD display modules (NEW)"
    Hit space once so the symbol changes to "M" (supported as modules)
    Hit Enter
    Enable modules for the following (Hit Space)
        FB driver for the HX8357D LCD Controller (NEW)
        Module to for adding FBTFT devices
```

Disable RTC support

Some of the RTC drivers crash, and we don't need them anyway, so we'll just disable them

```
Go back to "Device Drivers"
    Make sure "Real Time Clock" is not selected
```

Add support for the STMPE touchscreen

It is important to do the following in this exact order:

```
Go to "Device Drivers"
    ==> "Multifunction device drivers"
    ==> Select "STMicroelectronics STMPE Interface Drivers"
        Hit enter
        ==> Select option to "STMicroelectronics STMPE SPI Interface (NEW)"
        ==> Select option to "STMicroelectronics STMPE I2C Interface (NEW)"

Go to "Device Drivers"
    ==> "Input device support"
    ==> Select "Touchscreens"
    Hit Space, then Enter
        ==> Select "STMicroelectronics STMPE touchscreens"
```

Enable support for the Edimax EW-7811Un WLAN driver

```
Go to "Device Drivers"
    ==> "Network device support"
    ==> "Wireless LAN"
        Hit Enter
        ==> ""Realtek 8192C USB WiFi""
            Hit Space to select as a module

Exit and Save
```

Compile the kernel and modules

Set the following environment variables to make things easier

```
export CPUP1=$(($(sysctl hw.ncpu | awk '{print $2}')+1))
```

Compile (and cross fingers). This will take a while, so now is a good time to write a clean Raspbian image to your SD card (next step), and after that, read this xkcd comic on alternative uses of the "sudo" command http://xkcd.com/149/

```
make ARCH=arm CROSS_COMPILE=$CCPREFIX -j${CPUP1}
```

Make the modules

```
make ARCH=arm CROSS_COMPILE=$CCPREFIX modules -j$CPUP1
```

Install a clean Raspbian image to the SD card

Download the Adafruit image for the 3.5" PiTFT

```
cd ${RASPBIANIMAGEDOWNLOADLOCATION}
wget http://adafruit-
download.s3.amazonaws.com/PiTFT35R_raspbian140909_2014_09_18.zip
unzip PiTFT35R_raspbian140909_2014_09_18.zip
```

See SDCardAccess on more details how to write the image to an SD card.

Set the location of the SD card. Double-triple check this or you'll mess up your machine

```
export SDCARDLOCATION=/dev/rdisk3
```

Insert the SD card to your computer Unmount the inserted disk and write the image

```
sudo diskutil unmountDisk ${SDCARDLOCATION}
sudo dd if=${RASPBIANIMAGEDOWNLOADLOCATION}/PiTFT35R-140909-140918.img
of=/${SDCARDLOCATION} bs=1M
```

Copy the kernel and modules

you could mount the filesystem and cp the files over, but mounting an ext4fs file system on a MAC is not straightforward, so we're going to remotely upload the right files.

Boot your Pi and connect the ethernet port

```
rm -rf ${BUILDOUTPUT}/*
mkdir -p ${BUILDOUTPUT}/boot
```

Install the modules to the temporry directory

```
cd ${BUILDSOURCES}
make ARCH=arm CROSS_COMPILE=${CCPREFIX} INSTALL_MOD_PATH=${BUILDOUTPUT}
modules_install
```

Install the kernel image to the boot subdirectory

```
make ARCH=arm CROSS_COMPILE=${CCPREFIX} INSTALL_PATH=${BUILDOUTPUT}/boot/
INSTALLKERNEL=none install
```

Transfer the new kernel and modules

```
cd ${BUILDOUTPUT}
mv boot/vmlinux-${KERNELVERSION} boot/kernel.img
tar cvzf kernelFiles.tgz ./*
```

Find out the IP address of your PI (from your router, or the serial console) and transfer the files

```
scp kernelFiles.tgz pi@${PIIPADDRESS}:~/
```

Log into the PI

```
ssh pi@${PIIPADDRESS}
```

Check kernel version

```
uname -a
```

This showed the old kernel:

```
Linux raspberrypi 3.12.28+ #737 PREEMPT Wed Jan 14 19:40:07 GMT 2015 armv6l
GNU/Linux
```

Update some settings

```
sudo raspi-config

Go to "Internationalisation ==> change Timezone"
    Select US ==> Central (or whatever you need)
Go back and select "Advanced Options"
    Select "Hostname"
    Change it to "solutions-pi"
Go back and select "Expand Filesystem"
Go back and agree to reboot.
```

Log back in

```
ssh pi@${PIIPADDRESS}
```

Check the date

```
date
```

Sometimes this is wrong. If the date is back in the past, extracting the tar will give a lot of ugly warnings. In that case, do the following:

```
sudo date -s "22 Jan 2015 20:00:00" #Something close to now
sudo /etc/init.d/ntp restart
```

Verify the integrity of your tarball. Check the output of the following command for errors.

```
tar tvzf ~/kernelFiles.tgz
```

Extract and install

```
rm -rf new-kernel-files
mkdir new-kernel-files
cd new-kernel-files
tar xvzf ~/kernelFiles.tgz
```

Remove the old modules directory to make room (Especially if your new kernel version is the same as the currently installed one (after the update))

```
sudo rm -rf /lib/modules/3.12.*
```

Copy the kernel and modules

```
sudo mv /boot/kernel.img /boot/kernel_ORIG.img
sudo cp -r boot/* /boot/
sudo cp -r lib/* /lib/
```

Do some sanity checks

```
ls boot/
```

Make sure this shows something like this

```
System.map-3.12.36  kernel.img
```

Check for the kernel that will be booted

```
grep kernel= /boot/config.txt
```

Make sure this is either empty or contains

```
kernel=/boot/kernel.img
```

Reboot

```
sudo reboot
```

Log back in

After a while, log back in and check the kernel version

```
ssh pi@${PIIPADDRESS}
```

Check the kernel version

```
uname -a
```

I got this pointing to my new kernel version and its build time

```
Linux raspberrypi 3.12.36 #2 PREEMPT Fri Jan 16 16:02:43 CST 2015 armv6l
GNU/Linux
```

Build the module dependencies

```
sudo depmod -a
```

Now cross your fingers again

```
sudo reboot
```

Log in again

```
ssh pi@${PIIPADDRESS}
```

Set the PI to boot to desktop, and change the timezone (optional)

```
sudo raspi-config
```

Let it reboot and log in again

```
ssh pi@${PIIPADDRESS}
```

Force some modules to be loaded on startup

```
sudo nano /etc/modules
```

Add the following:

```
# /etc/modules: kernel modules to load at boot time.
#
# This file contains the names of kernel modules that should be loaded
# at boot time, one per line. Lines beginning with "#" are ignored.
# Parameters can be specified after the module name.

usbserial
cp210x
snd-bcm2835
spi-bcm2708
i2c-bcm2708
fbtft_device
rpi_power_switch
```

Reboot again

```
sudo reboot
```

==========

|| Congratulations, you're done! ||

==========

Potential problems and their fixes

ssh fails with a warning about changed ssh keys

Do

```
nano ~/.ssh/known_hosts
```

Find the line with the PI's IP address and delete it (CTRL-K)

or

```
rm -rf ~/.ssh/known_hosts
```

Green light on the side is blinking, but that's about it

Count the number of flashes:

```
3 flashes: start.elf not found
4 flashes: start.elf not launched
7 flashes: kernel.img not found
```

No response from PI (Ethernet, keyboard,...)

Hookup a serial terminal to the UART and see what's going on

```
Connect GND, TX and RX to the GND, RX, TX pins indicated here
http://elinux.org/RPi_Low-level_peripherals
```

Update the PI Firmware and bootloader

Option 1

The Raspberry PI B+ will need the new firmware in order to support the new ethernet chip
Download the firmware

```
cd ${BUILDFIRMWARE}
git clone git://github.com/raspberrypi/firmware.git
```

Firstly, update the required boot files in the Raspberry Pi boot directory with those you've downloaded. These are:

```
bootcode.bin
fixup.dat
start.elf
```

Check your compiler options

```
${CCPREFIX}gcc -v 2>&1 | grep hard
```

If something prints out, and you can see --with-float=hard, you need the hard float ones. If you don't see it, you should select the hard float option in make menuconfig and restart.

Remove the /opt/vc directory from the Raspberry Pi root, then:

For hard float, copy vc from the hardfp/opt directory into /opt in the Raspberry Pi root directory

Reboot

```
sudo reboot
```

Option 2

boot the Raspberry with the Wheezy 09-09-2104 image and make sure you have an ethernet connection Login

```
ssh pi@1${PIIPADDRESS}

sudo apt-get update
sudo apt-get upgrade
```

It identified and upgraded these packages:

```
apt apt-utils base-files bash ca-certificates cpio curl dbus dbus-x11 dosfstools
file firmware-brcm80211 gnupg gpgv libapt-inst1.5 libapt-pkg-dev libapt-pkg4.12
libavcodec53 libavutil51 libc-bin libc-dev-bin libc6 libc6-dev libcurl3
libcurl3-gnutls libdbus-1-3 libevent-2.0-5 libflac8 libgcrypt11 libjasper1
libkeyutils1 libmagic1 libnss3 libraspberrypi-bin libraspberrypi-dev
libraspberrypi-doc libraspberrypi0 libssl1.0.0 libtasn1-3 libxml2 libyaml-0-2
locales mime-support multiarch-support ntp openssl perl perl-base perl-modules
python-picamera python-rpi.gpio python3-picamera python3-rpi.gpio raspberrypi-
artwork raspberrypi-bootloader rsyslog sonic-pi tzdata unzip wget wpagui
wpasupplicant xdg-utils xserver-common xserver-xorg-core
```

Sit back and have a coffee

Reboot

```
sudo reboot
```

Option 3

```
sudo rpi-update
```

Reboot

```
sudo reboot
```

Illegal option for xargs on a MAC build

The make command failed with

```
xargs: illegal option -- r
usage: xargs [-0opt] [-E eofstr] [-I replstr [-R replacements]] [-J replstr]
         [-L number] [-n number [-x]] [-P maxprocs] [-s size]
         [utility [argument ...]]
```

This is because BSD xargs doesn't have the same options as GNU xargs. Install GNU xargs from macports (It should show up earlier in your search path)

```
which args
sudo port install findutils
which xargs
```

Make sure that the "which" command points to a new xargs

Ncurses error

Making menuconfig complained about missing ncurses libraries (on MAC OS X 10.10 Yosemite), so do this (the install might not be needed)

```
sudo port selfupdate
sudo port install ncurses
```

Missing tinfo library during make menuconfig

It somehow expects library tinfo, which seems to basically be ncurses, so do:

```
sudo ln /usr/include/tinfo.h /usr/include/tinfo.h
sudo ln /usr/lib/libncurses.dylib /usr/lib/libtinfo.dylib
```