# Purpose

This document summarizes Raspberry Pi setup as development platform. It will cover kernel build, external module building and loading, file share setup on RPI. This document gives a generic guideline of RPI setup for specific information on how to build an SDK from trunk. Please refer to another document.

We are using Raspberry Pi 1 model B. Current kernel binary is **raspbian debian wheezy version 1.4.0 2015-02-18** from <http://www.raspberrypi.org/downloads/>.

## Getting kernel source code for RPI

If you have an svn access, then RPI kernel tar ball for **raspbian debian wheezy distribution package** version mentioned above is available at

*<http://aussvn01/svn/apps/vproc_sdk/trunk/platform/raspberry/kernel/linux-0be82f722c097340632a59b879fbfee9c6148f53.tar.gz>*

which can be untarred using command:

*tar –zxvf linux-0be82f722c097340632a59b879fbfee9c6148f53.tar.gz*

If not, then read on to get information how to get it from Raspberry git hub repository:

The RPI linux OS distribution package **raspbian debian wheezy version 1.4.0 2015-02-18**

is based on linux version 3.18.7 and downloaded via referring to link <https://github.com/notro/rpi-source/wiki>. Please note though rpi-source deprecated now but last updated worked well to fetch kernel 3.18.7+ kernel source and firmware binaries. The kernel source code as mentioned in link <http://www.raspberrypi.org/documentation/linux/kernel/building.md> fetches the latest kernel source code and different from the one Raspbian Debian Wheezy Version 1.4.0 2015-02-18 built. Other than rpi-source, user may refer to this link to fetch matching source code to binary image <http://www.raspberrypi.org/forums/viewtopic.php?f=66&t=82811&p=726802#p726802>.

## Building for Raspberry Pi

In order to build any source code for RPI on native or cross compiler, user needed to download linux kernel and firmware source code along with toolchain of matching version as distribution package or binary currently running on raspberry pi board. Once a matching kernel tarball downloaded, do the following steps

* + 1. **Cross Compiling**
       1. Copy 3.18.7 version kernel tar as downloaded by rpi-source or svn on build server (ex. Ubuntu 14.04 LTS machine), untar it and do following:
       2. Go to untarred folder i.e linux-*0be82f722c097340632a59b879fbfee9c6148f53*
          1. \*Export PATH to include Raspberry kernel tools path. Example, PATH=$PATH:/home/vermas/tools/arm-bcm2708/gcc-linaro-arm-linux-gnueabihf-raspbian. Ensure that toolchain path setup properly by giving “which arm-linux-gnueabihf-gcc’
          2. make ARCH=arm CROSS\_COMPILE=arm-linux-gnueabihf- bcmrpi\_defconfig
          3. make ARCH=arm CROSS\_COMPILE=arm-linux-gnueabihf-
          4. go to your module directory. It can be in parallel to linux directory or within kernel
          5. vi Makefile
          6. add obj-m = filename.o <example dumb.o if containing one file dumb.c>
          7. make ARCH=arm CROSS\_COMPILE=arm-linux-gnueabihf- -C <path to kernel source code> /home/vermas/raspberry/linux-0be82f722c097340632a59b879fbfee9c6148f53 M=$(pwd) modules. (Or add an all rule with this call)

Example, make ARCH=arm CROSS\_COMPILE=arm-linux-gnueabihf- -C /home/vermas/raspberry/linux-0be82f722c097340632a59b879fbfee9c6148f53 M=$(pwd) modules

* + - * 1. Above make should create dumb.ko
        2. Copy it to samba share on pi

login to pi. Do insmod dumb.ko . check console output by running dmesg

* + 1. **Native Compiling**

As mentioned above, user can use rpi-source and other shared link.

* + - 1. Copy module source code in your home directory on pi
      2. Untar linux source code on to pi and create symbolic link

sudo ln -s /home/pi/src/linux /lib/modules/`uname -r`/build  
sudo ln -s /home/pi/src/linux /lib/modules/`uname -r`/source

* + - 1. cd linux
      2. make clean
      3. make bcmrpi\_defconfig
      4. make modules\_prepare
      5. zcat /proc/config.gz > .config
      6. cp ../firmware/extra/modules.symvers ./modules.symvers
      7. cd to module/driver to compile
      8. do ‘make’

## Building SSL driver for Raspberry pi

SSL Driver currently is compiled as simple object files which are further supposed to be linked by its user whose end format may be kernel loadable module or an archive library or exe

* + 1. Go to trunk/
    2. Customize env.sh with your local path folder for kernel directory and source code
    3. Run command “source env.sh”
    4. Go to platform/raspberry and follow steps as mentioned in section above to build kernel
    5. Once kernel is built, go to trunk/platform/raspberry/drivers/ssl
    6. Give command : make BUILD\_TYPE=DEBUG (if building debug version)
    7. This will build ssl.o files

## Setting up samba share of home directory on RPI

Follow instructions here: http://elinux.org/R-Pi\_NAS

## Troubleshooting

* Having issues with gcc version using rpi-source?

Raspberry Pi kernel distribution package Debian Wheezy comes with gcc version 4.3.3 however most of the kernels are built with version 4.8.8 or higher. If you see issues while building on rpi of gcc-version you can follow instructions at https://github.com/notro/rpi-source/wiki. There are two links to update and install gcc to 4.8, please use jessie version. As non-jessie mode will install gcc-4.8.2 and that version shows issues compiling downloaded kernel. I used Jessie mode and it installed version 4.8.4 for me. When I executed rpi-source after installing this, it executed modules\_prepare command successfully. Much of the information on this link was helpful <http://www.embedded.com/electronics-blogs/open-mike/4437876/Getting-Started-with-Embedded-Linux---Part-Nine-Self-hosted-development>

* Booting RPI with another kernel version:

Booting raspberry with 3.18.9+ version works. If you follow cross compiler instruction here to compile source code <http://www.raspberrypi.org/documentation/linux/kernel/building.md> and copy kernel image on boot sector of sdcard , you can boot through that kernel. However if you build some module on that kernel version and simply copy it on Pi it fails to load.

Also, modules\_install command on cross compiler fails with message “arm-linux-gnueabi-gcc:Command not found”. Booting with another kernel needs some exercise and will be tried later

Download toolchain: git clone https://github.com/raspberrypi/tools

Firmware: git clone <https://github.com/raspberrypi/firmware>

However this will pull latest on branch. You will need to know commit-id of the one matching to your kernel image and check out that. Please refer to this link to get some help: <http://www.raspberrypi.org/forums/viewtopic.php?f=66&t=82811&p=726802#p726802>