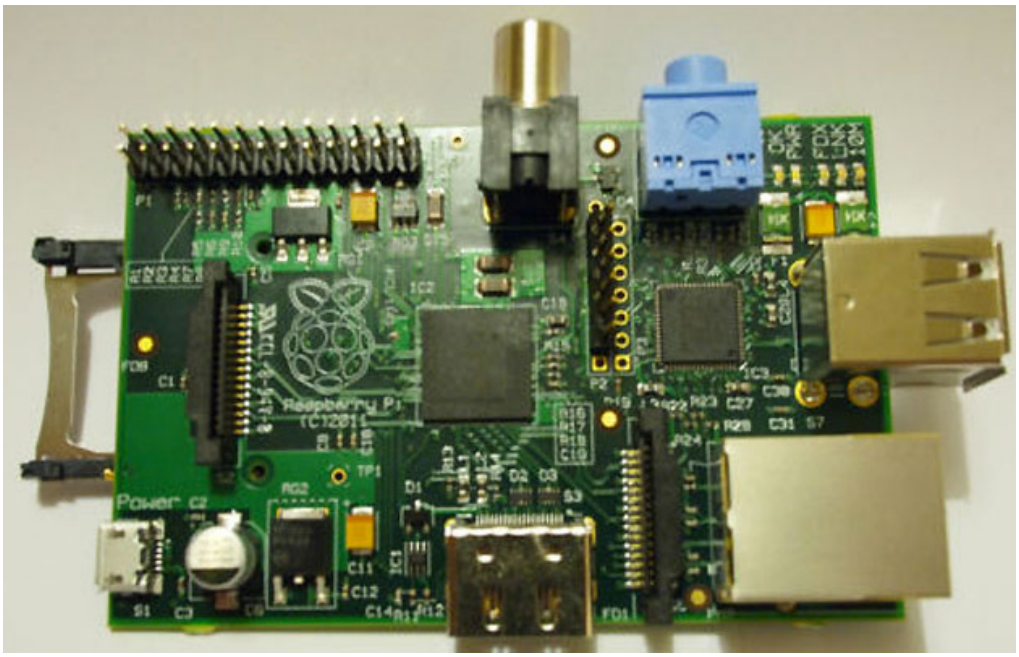
Quick Start Guide

The Raspberry Pi – Single Board Computer



Source: Raspberry Pi & Wiki

Chapter 1: RPi Hardware Basic Setup

Typical Hardware You Will Need

While the RPi can be used without any additional hardware (except perhaps a power supply of some kind), it won't be much use as a general computer. As with any normal PC, it is likely you will need some additional hardware.

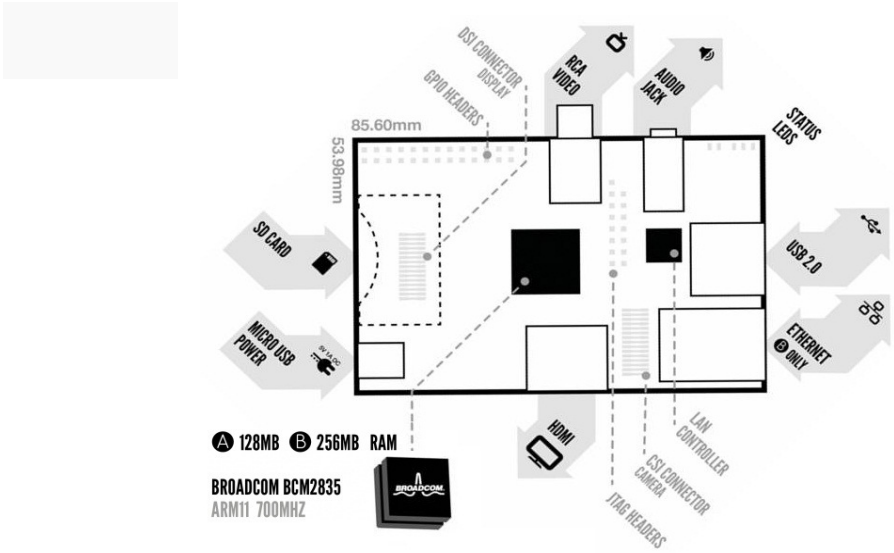
The following are more or less essential:

- Raspberry Pi board
- Prepared Operating System SD Card
- USB keyboard
- Display (with HDMI, DVI, Composite or SCART input)
- Power Supply
- Cables

Highly suggested extras include:

- USB mouse
- Internet connectivity - a USB WiFi adaptor (Model A/B) or a LAN cable (Model B)
- Powered USB Hub
- Case

Connecting Together



You can use the diagram to connect everything together, or use the following instructions:

1. Plug the preloaded SD Card into the Pi.
2. Plug the USB keyboard and mouse into the Pi, perhaps via a USB Hub. Connect the Hub to power, if necessary.

3. Plug the video cable into the screen (TV) and into the Pi.
4. Plug your extras into the Pi (USB WiFi, Ethernet cable, hard drive etc.). This is where you may really need a USB Hub.
5. Ensure that your USB Hub (if any) and screen are working.
6. Plug the power source into the main socket.
7. With your screen on, plug the other end of the power source into the Pi.
8. The Pi should boot up and display messages on the screen.

It is always recommended to connect the MicroUSB Power to the unit last (while most connections can be made live, it is best practice to connect items such as displays/h/w pin connections with the power turned off).

The RPi may take a long time to boot when powered-on for the first time, so be patient!

Prepared Operating System SD Card

As the RPi has no internal storage or built-in operating system it requires an SD-Card that is set up to boot the RPi.

- You can create your own preloaded card using any suitable SD card you have. Be sure to backup any existing data on the card.
- Preloaded SD cards will be available from the RPi Shop.

This guide will assume you have a preloaded SD card.

Keyboard & Mouse

Most standard USB keyboards and mice will work with the RPi. Wireless keyboard/mice should also function, and only require a single USB port for an RF dongle. In order to use a Bluetooth keyboard or mouse you would need to use a Bluetooth dongle, which again uses a single port.

Remember that the Model A has a single USB port and the Model B only has two (typically a keyboard and mouse will use a USB port each).

Display

There are two main connection options for the RPi display, *HDMI* (high definition) and *Composite* (low definition).

- HD TVs and most LCD Monitors can be connected using a full-size 'male' HDMI cable, and with an inexpensive adaptor if DVI is used. HDMI versions 1.3 and 1.4 are supported, and a version 1.4 cable is recommended. The RPi outputs audio and video via HMDI, but does not support HDMI input.
- Older TVs can be connected using Composite (a yellow-to-yellow cable) or via SCART (using a Composite to SCART adaptor). PAL and NTSC TVs are supported. When using composite video, audio is available from a 3.5mm (1/8 inch) socket, and can be sent to your TV, to headphones, or to an amplifier. To send audio your TV,

you will need a cable which adapts from 3.5mm to double (red and white) RCA connectors.

Note: There is no VGA output available, so older VGA monitors will require an expensive adaptor.

Using an HDMI to DVI-D (digital) adaptor plus a DVI to VGA adaptor will not work. HDMI does not supply the DVI-A (analogue) needed to convert to VGA - converting an HDMI or DVI-D source to VGA (or component) needs an active converter. (It can work out cheaper to buy a new monitor.) The lack of VGA has been acknowledged as a priority issue.

Power Supply

The unit uses a Micro USB connection to power itself (only the power pins are connected - so it will not transfer data over this connection). A standard modern phone charger with a micro-USB connector will do, but needs to produce at least 700mA at 5 volts. Check your power supply's ratings carefully. Suitable mains adaptors will be available from the RPi Shop and are recommended if you are unsure what to use.

You can use a range of other power sources (assuming they are able to provide enough current ~700mA):

- Computer USB Port or powered USB hub (will depend on power output)
- Special wall warts with USB ports
- Mobile Phone Backup Battery (will depend on power output) (in theory - needs confirmation)

To use the above, you'll need a USB A 'male' to USB micro 'male' cable - these are often shipped as data cables with MP3 players.

Cables

You will probably need a number of cables in order to connect your RPi up.

1. Micro-B USB Power Cable
2. HDMI-A or Composite cable, plus DVI adaptor or SCART adaptor if required, to connect your RPi to the Display/Monitor/TV of your choice.
3. Audio cable, this is not needed if you use a HDMI TV/monitor.
4. Ethernet/LAN Cable

Additional Peripherals

You may decide you want to use various other devices with your RPi, such as Flash Drives/Portable Hard Drives, Speakers etc.

Internet Connectivity

This may be an Ethernet/LAN cable (standard RJ45 connector) or a USB WiFi adaptor. The RPi ethernet port is auto-sensing which means that it may be connected to a router or directly to another computer (without the need for a crossover cable).

USB-Hub

In order to connect additional devices to the RPi, you may want to obtain a USB Hub, which will allow multiple devices to be used.

It is recommended that a **powered** hub is used - this will provide any additional power to the devices without affecting the RPi itself.

USB version 2.0 is recommended. USB version 1.1 is fine for keyboards and mice, but may not be fast enough for other accessories.

Case

Since the RPi is supplied without a case, it will be important to ensure that you do not use it in places where it will come into contact with conductive metal or liquids, unless suitably protected.

Expansion & Low Level Peripherals

If you plan on making use of the low level interfaces available on the RPi, then ensure you have suitable header pins for the GPIO (and if required JTAG) suitable for your needs.

Also if you have a particular low-level project in mind, then ensure you design in suitable protection circuits to keep your RPi safe.

Chapter 2: RPi Advanced Setup

Finding hardware and setting up

You'll need a preloaded SD card, USB keyboard, TV/Monitor (with HDMI/ DVI/ Composite/ SCART input), and power supply (USB charger or a USB port from a powered USB Hub or another computer).

You'll likely also want a USB mouse, a case, and a USB Hub (a necessity for Model A). A powered USB Hub will reduce the demand on the RPi. To connect to the Internet, you'll need either an Ethernet/LAN cable (Model B) or a USB WiFi adaptor (either model).

When setting up, it is advisable to connect the power after everything else is ready.

Serial connection

The Serial Port is a simple and uncomplicated method to connect to the Raspberry Pi. The communication depends on byte wise data transmission, is easy to setup and is generally available even before boot time.

First interaction with the board

Connect the serial cable to the COM port in the Raspberry Pi, and connect the other end to the COM port or USB Serial Adapter in the computer.

Serial Parameters

The following parameters are needed to connect to the Raspberry. All parameters except **Port_Name** and **Speed** are default values and may not need to be set.

- **Port_Name**: Linux automatically assigns different names for different types of serial connectors. Choose your option:
  - Standard Serial Port: ttyS0 ... ttySn
  - USB Serial Port Adapter: ttyUSB0 ... ttyUSBn
- **Speed**: 115200
- Bits: 8
- Parity: None
- Stop Bits: 1
- Flow Control: None

The Serial Port is generally usable by the users in the group **dialout**. To add oneself to the group **dialout** the the following command needs to be executed with **root** privileges:

```
$useradd -G {dialout} your_name
```

- **Super Easy Way Using GNU Screen**

Enter the command below into a terminal window

```
screen Port_Name 115200
```

- **Super Easy Way Using Minicom**

Run minicom with the following parameters:

```
minicom -b 115200 -o -D Port_Name
```

- **GUI method with GtkTerm**

Start *GtkTerm*, select Configuration->Port and enter the values above in the labelled fields.

- **Windows Users**

Windows Users above Windows XP must download putty or a comparable terminal program. Users of XP and below can choose between using *putty* and *Hyperterminal.*

First Dialog

If you get the prompt below, you are connected to the Raspberry Pi shell!

```
prompt> #
```

First command you might want try is "help":

```
prompt> # help
```

If you get some output, you are correctly connected to the Raspberry Pi! Congratulations!

SD card setup

Now we want to use an SD card to install some GNU/Linux distro in it and get more space for our stuff. You can use either an SD or SDHC card. In the latter case of course take care that your PC card reader also supports SDHC. Be aware that you are not dealing with an x86 processor, but instead a completely different architecture called ARM, so don't forget to install the ARM port for the distro you are planning to use.

Formatting the SD card via the mkcard.txt script

1. Download **mkcard.txt** .
2. $ chmod +x mkcard.txt

3. $ ./mkcard.txt /dev/sd*x*, where *x* is the letter of the card. You can find this by inserting your card and then running dmesg | tail. You should see the messages about the device being mounted in the log. Mine mounts as **sdc**.

Once run, your card should be formatted.

Formatting the SD card via fdisk "Expert mode"

First, lets clear the partition table:

```
================================================================================
$ sudo fdisk /dev/sdb

Command (m for help): o
Building a new DOS disklabel. Changes will remain in memory only,
until you decide to write them. After that, of course, the previous
content won't be recoverable.

Warning: invalid flag 0x0000 of partition table 4 will be corrected by
w(rite)
================================================================================
```

Print card info:

```
================================================================================
Command (m for help): p

Disk /dev/sdb: 128 MB, 128450560 bytes
....
================================================================================
```

Note card size in bytes. Needed later below.

Then go into "Expert mode":

```
================================================================================
Command (m for help): x
================================================================================
```

Now we want to set the geometry to 255 heads, 63 sectors and calculate the number of cylinders required for the particular SD/MMC card:

```
================================================================================
Expert command (m for help): h
Number of heads (1-256, default 4): 255

Expert command (m for help): s
Number of sectors (1-63, default 62): 63
Warning: setting sector offset for DOS compatiblity
================================================================================
```

NOTE: Be especially careful in the next step. First calculate the number of cylinders as follows:

- B = Card size in bytes (you got it before, in the second step when you printed the info out)
- C = Number of cylinders

```
C=B/255/63/512
```

When you get the number, you round it DOWN. Thus, if you got 108.8 you'll be using 108 cylinders.

```
================================================================================
Expert command (m for help): c
Number of cylinders (1-1048576, default 1011): 15
================================================================================
```

In this case 128MB card is used (reported as 128450560 bytes by fdisk above), thus 128450560 / 255 / 63 / 512 = 15.6 rounded down to 15 cylinders. Numbers there are 255 heads, 63 sectors, 512 bytes per sector.

So far so good, now we want to create two partitions. One for the boot image, one for our distro. Create the FAT32 partition for booting and transferring files from Windows. Mark it as bootable.

```
================================================================================
Expert command (m for help): r
Command (m for help): n
Command action
  e   extended
  p   primary partition (1-4)
p
Partition number (1-4): 1
```

```
First cylinder (1-245, default 1): (press Enter)
Using default value 1
Last cylinder or +size or +sizeM or +sizeK (1-245, default 245): +50

Command (m for help): t
Selected partition 1
Hex code (type L to list codes): c
Changed system type of partition 1 to c (W95 FAT32 (LBA))

Command (m for help): a
Partition number (1-4): 1
================================================================================
=====
```

Create the Linux partition for the root file system.

```
================================================================================
=====
Command (m for help): n
Command action
  e   extended
  p   primary partition (1-4)
p
Partition number (1-4): 2
First cylinder (52-245, default 52): (press Enter)
Using default value 52
Last cylinder or +size or +sizeM or +sizeK (52-245, default 245):(press
Enter)
Using default value 245
================================================================================
=====
```

Print and save the new partition records.

```
================================================================================
=====
Command (m for help): p

Disk /dev/sdc: 2021 MB, 2021654528 bytes
255 heads, 63 sectors/track, 245 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes

  Device Boot      Start         End      Blocks   Id  System
/dev/sdc1   *           1          51      409626    c  W95 FAT32 (LBA)
/dev/sdc2              52         245     1558305   83  Linux

Command (m for help): w
The partition table has been altered!

Calling ioctl() to re-read partition table.
```

```
WARNING: Re-reading the partition table failed with error 16: Device or
resource busy. The kernel still uses the old table. The new table will be
used at the next reboot.

WARNING: If you have created or modified any DOS 6.x partitions, please see
the fdisk manual page for additional information.
Syncing disks.
================================================================================
=====
```

Now we've got both partitions, next step is formatting them.

NOTE: If the partitions (/dev/sdc1 and /dev/sdc2) does not exist, you should unplug the card and plug it back in. Linux will now be able to detect the new partitions.

```
================================================================================
=====
$ sudo mkfs.msdos -F 32 /dev/sdc1 -n LABEL
mkfs.msdos 2.11 (12 Mar 2005)

$ sudo mkfs.ext3 /dev/sdc2
mke2fs 1.40-WIP (14-Nov-2006)
Filesystem label=
OS type: Linux
Block size=4096 (log=2)
Fragment size=4096 (log=2)
195072 inodes, 389576 blocks
19478 blocks (5.00%) reserved for the super user
First data block=0
Maximum filesystem blocks=402653184
12 block groups
32768 blocks per group, 32768 fragments per group
16256 inodes per group
Superblock backups stored on blocks:
        32768, 98304, 163840, 229376, 294912

Writing inode tables: done
Creating journal (8192 blocks): done
Writing superblocks and filesystem accounting information:
================================================================================
=====
```

All done!

NOTE: For convenience, you can add the -L option to the mkfs.ext3 command to assign a volume label to the new ext3 filesystem. If you do that, the new (automatic) mount point under /media when you insert that SD card into some Linux hosts will be based on that label. If there's no label, the new mount point will most likely be a long hex string, so assigning a label makes manual mounting on the host more convenient.

Setting up the boot partition

The boot partition must contain:

- bootcode.bin : 2nd stage bootloader, starts with SDRAM disabled
- loader.bin : 3rd stage bootloader, starts with SDRAM enabled
- start.elf: The GPU binary firmware image, provided by the foundation.
- kernel.img: The OS kernel to load on the ARM processor. Normally this is Linux - see instructions for compiling a kernel.
- cmdline.txt: Parameters passed to the kernel on boot.

Optional files:

- config.txt: A configuration file read by the GPU. Use this to override set the video mode, alter system clock speeds, voltages, etc.
- vlls directory: Additional GPU code, e.g. extra codec's. Not present in the initial release.

Additional files supplied by the foundation

These files are also present on the SD cards supplied by the foundation.

Additional kernels. Rename over kernel.img to use them (ensure you have a backup of the original kernel.img first!):

- kernel_emergency.img : kernel with busybox rootfs. You can use this to repair the main linux partition using e2fsck if the linux partition gets corrupted.

Additional GPU firmware images, rename over start.elf to use them:

- arm128_start.elf : 128M ARM, 128M GPU split (use this for heavy 3D work, possibly also required for some video decoding)
- arm192_start.elf : 192M ARM, 64M GPU split (this is the default)
- arm224_start.elf : 224M ARM, 32M GPU split (use this for Linux only with no 3D or video processing. It's enough for the 1080p frame buffer, but not much else)

Writing the image into the SDcard and finally booting GNU/Linux

The easiest way to do this is to use PiCard. It even saves you from some hassles explained above. You will need your SD card + reader and a Linux pc to use PiCard. After that, just plug the card into your Rpi.

Setting up the boot args

Wire up your Raspberry Pi and power it up

As explained in Chapter 1

SD Card Cloning/Backup

```
Note: Update these instructions if required once they've been tried
```

From windows you can copy the full SD-Card by using Win32DiskImager. Alternatively, you can use the following instructions;

```
Note:
Many built-in SD card readers do not work, so if you have problems
use an external SD-USB adapter for this.
```

Required Software Setup

- download a windows utility dd.exe from *http://www.chrysocome.net/dd*
- rename it windd.exe

(This executable can to write to your harddisk so exercise caution using it!)

- make a copy named dd-removable.exe

(That executable refuses to write to your hard disk as it is named dd-removable As long as you use dd-removable.exe you cannot lose your hard disk)

- Connect an SD card to the computer
- run "dd-removable –list"

Should give something like this:

```
rawwrite dd for windows version 0.6beta3.
Written by John Newbigin <jn@it.swin.edu.au>
This program is covered by terms of the GPL Version 2.

NT Block Device Objects
\\?\Device\Harddisk1\Partition0
link to \\?\Device\Harddisk1\DR8
Removable media other than floppy. Block size = 512
size is 4075290624 bytes
```

This "\\?\Device\Harddisk1\Partition0" is the part you need.

Reading an image from the SD Card

BEWARE: DO THIS WRONG AND YOU CAN LOSE YOUR HARDDISK!!!

Obviously, you can NOT use 'dd-removable' to read an image as that executable refuses to write to your hard disk (so extra care is required here as you use 'windd').

- To **read** an SD-card image from the SD-card use:

```
windd bs=1M if=\\?\Device\Harddisk1\Partition0 of=THE_IMAGE_READ -size
Your disk name ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
```

Copying an image to the SD Card

BEWARE: DO THIS WRONG AND YOU CAN LOSE YOUR HARDDISK!!!

- To **copy** an image named "THEIMAGE" to the SD-card do this:

```
dd-removable bs=1M if=THEIMAGE of=\\?\Device\Harddisk1\Partition0
                   Your disk name ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
```

Software Development/Proving

A supported platform for the Raspberry is Qt , which is already being worked on. C/C++ is supported through a gcc cross-compiling tool chain.

After compiling, using QEMU and a Linux VM would be one way of testing your apps. This also works on Windows. Search the forum for the readymade ARM images. The choice of programming languages, IDEs and other tools ON the R-Pi is only determined by:

- The operating system compatibility (at the moment the specific Linux distro used)
- The status of the respective ARM package repositories and their binary compatibility
- The possibility to build other software + its dependencies for the R-Pi from sources.

For more guides and projects involving the Raspberry Pi, see RPi Projects (http://elinux.org/RPi_Projects).