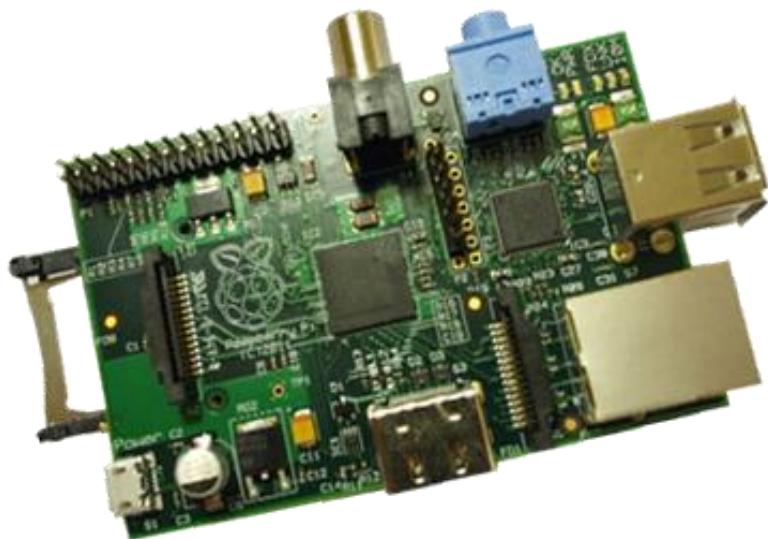
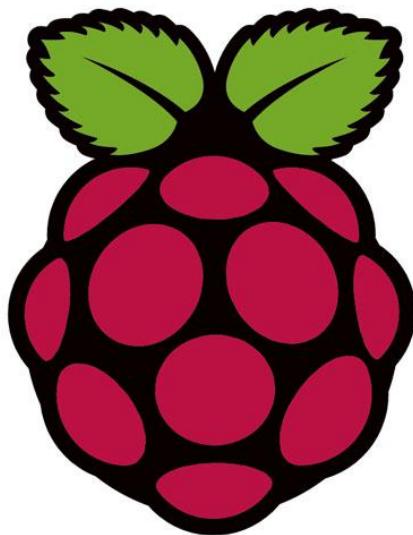


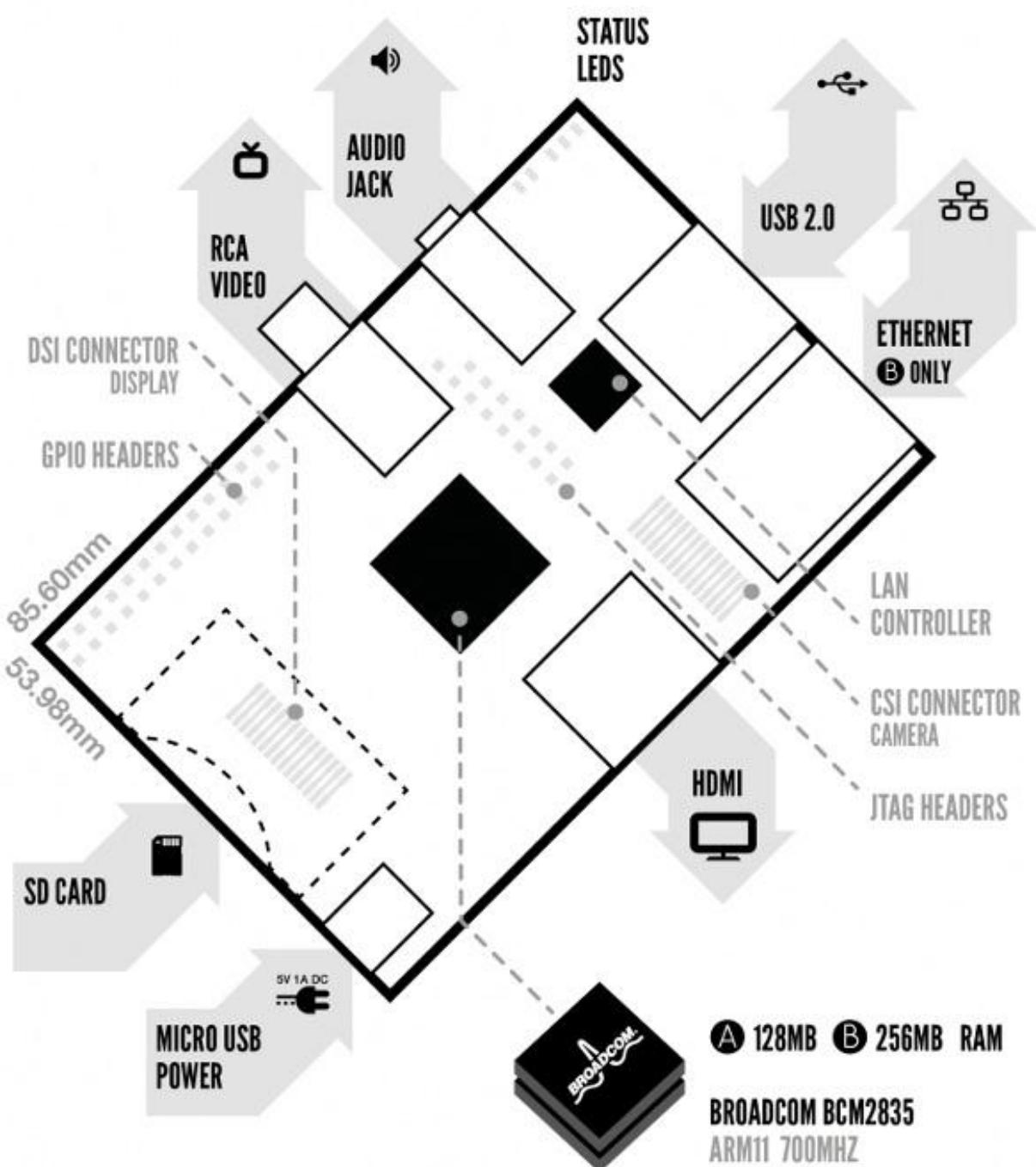
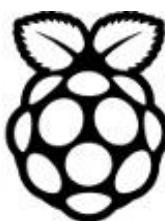
# Raspberry Pi

## Getting Started Guide



**RS Components Vsn 1.0 3/2012**

# Raspberry Pi Model A B



# 1. Raspberry Pi Basic Hardware Setup

## 1.1. Extra Hardware You Will Need

The Raspberry Pi board contains a processor and graphics chip, program memory (RAM) and various interfaces and connectors for external devices. Some of these devices are essential, others are optional. RPi operates in the same way as a standard PC, requiring a keyboard for command entry, a display unit and a power supply.

It also requires 'mass-storage', but a hard disk drive of the type found in a typical PC is not really in keeping with the miniature size of RPi. Instead we will use an SD Flash memory card normally used in digital cameras, configured in such a way to 'look like' a hard drive to RPi's processor. RPi will 'boot' (load the Operating System into RAM) from this card in the same way as a PC 'boots up' into Windows from its hard disk.

The following are essential to get started:

- SD card containing Linux Operating system
- USB keyboard
- TV or monitor (with HDMI, DVI, Composite or SCART input)
- Power supply (see Section 1.6 below)
- Video cable to suit the TV or monitor used

Recommended optional extras include:

- USB mouse
- Internet connection, Model A or B: USB WiFi adaptor
- Internet connection, Model B only: LAN (Ethernet) cable
- Powered USB hub
- Case

## 1.2. Connecting Everything Together

1. Plug the preloaded SD Card into the RPi.
2. Plug the USB keyboard and mouse into the RPi, perhaps via a USB hub. Connect the Hub to power, if necessary.
3. Plug a video cable into the screen (TV or monitor) and into the RPi.
4. Plug your extras into the RPi (USB WiFi, Ethernet cable, external 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 supply into the mains socket.
7. With your screen on, plug the power supply into the RPi microUSB socket.
8. The RPi 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 with the power turned off).

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

## 1.3. Operating System SD Card

As the RPi has no internal mass storage or built-in operating system it requires an SD card preloaded with a version of the Linux Operating System.

- You can create your own preloaded card using any suitable SD card (4GBytes or above) you have to hand. We suggest you use a new blank card to avoid arguments over lost pictures.
- Preloaded SD cards will be available from the RPi Shop.

## 1.4. 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 will need a Bluetooth USB dongle, which again uses a single port.

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

## 1.5. Display

There are two main connection options for the RPi display, *HDMI* (High Definition) and *Composite* (Standard Definition).

- HD TVs and many 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 HDMI, but does not support HDMI input.
- Older TVs can be connected using Composite video (a yellow-to-yellow RCA cable) or via SCART (using a Composite video to SCART adaptor). Both PAL and NTSC format TVs are supported.

When using a composite video connection, audio is available from the 3.5mm jack socket, and can be sent to your TV, headphones or an amplifier. To send audio to your TV, you will need a cable which adapts from 3.5mm to double (red and white) RCA connectors.

**Note: There is no analogue VGA output available. This is the connection required by many computer monitors, apart from the latest ones. If you have a monitor with only a D-shaped plug containing 15 pins, then it is unsuitable.**

## 1.6. Power Supply

The unit is powered via the microUSB connector (only the power pins are connected, so it will not transfer data over this connection). A standard modern phone charger with a microUSB connector will do, providing it can supply at least **700mA at +5Vdc**. 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.

**Note: The individual USB ports on a powered hub or a PC are usually rated to provide 500mA maximum. If you wish to use either of these as a power source then you will need a special cable which plugs into two ports providing a combined current capability of 1000mA.**

## 1.7. Cables

You will need one or more cables to connect up your RPi system.

- Video cable alternatives:
  - HDMI-A cable
  - HDMI-A cable + DVI adapter
  - Composite video cable
  - Composite video cable + SCART adaptor
- Audio cable (not needed if you use the HDMI video connection to a TV)
- Ethernet/LAN cable (Model B only)

## 1.8. Additional Peripherals

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

### 1.8.1. Internet Connectivity

This may be via an Ethernet/LAN cable (standard RJ45 connector) or a USB WiFi adaptor. The RPi Model B 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).

### 1.8.2. 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.

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

### 1.8.3. 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.

### 1.8.4. Expansion & Low-Level Peripherals

If you plan on making use of the low-level interfaces available on the RPi, then ensure you have a suitable plug for the GPIO header pins.

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

## 2. Raspberry Pi Advanced Setup (Geeks only)

### 2.1. 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.

### 2.2. Serial connection

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

#### 2.2.1. First interaction with the board

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

#### 2.2.2. Serial Parameters

The following parameters are needed to connect to the RPi. 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: ttyUSBO ... 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
```

#### 2.2.3. Super Easy Way using GNU Screen

Enter the command below into a terminal window

```
screen Port_Name 115200
```

#### **2.2.4. Super Easy Way using Minicom**

Run minicom with the following parameters:

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

#### **2.2.5. GUI method with GtkTerm**

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

#### **2.2.6. Windows Users**

Windows 7 or Vista users must download putty or a comparable terminal program. Users of XP and below can choose between using *putty* and *Hyperterminal*.

#### **2.2.7. 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!

### **2.3. SD card setup**

Now we want to install a GNU/Linux distro on an SD card and make 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.

#### **2.3.1. Formatting the SD card via the mkcard.txt script**

1. Download **mkcard.txt** .
2. `$ chmod +x mkcard.txt`
3. `$ ./mkcard.txt /dev/sdx`, 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.

#### **2.3.2. 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*

*...*

```
=====
```

Make a note of the card size in bytes. You will need it later.  
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 compatibility*

```
=====
```

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

$C = B / 255 / 63 / 512$  where

- $B$  = Card size in bytes (The number you wrote down earlier.)
- $C$  = Number of cylinders

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

```
=====
```

*Expert command (m for help): c*

*Number of cylinders (1-1048576, default 1011): 15*

```
=====
```

In this case a 128MB card is used (reported as 128450560 bytes by *fdisk* above), thus  
 $128450560 / 255 / 63 / 512 = 15.6$  rounded down to 15 cylinders. 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
```

---

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

*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*) do 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 hexstring, so assigning a label makes manual mounting on the host more convenient.

## **2.4. 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 CODECs. Not present in the initial release.

## **2.5. Additional files supplied by RPi 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 : 128MB ARM, 128MB GPU split (use this for heavy 3D work, possibly also required for some video decoding)
- arm192\_start.elf : 192MB ARM, 64MB GPU split (this is the default)
- arm224\_start.elf : 224MB ARM, 32MB 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)

## **2.6. Writing the image to the SDcard and 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.

## **2.7. Wire up your Raspberry Pi and power it up**

As explained in Section 1

## **2.8. 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.*

## 2.9. 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 hard disk 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.

## 2.10. Reading an image from the SD Card

**BEWARE: DO THIS WRONG AND YOU MIGHT CORRUPT YOUR HARD DISK!**

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 ^^^^^^^^^^^^^^^^^^
```

## 2.11. Copying an image to the SD Card

**BEWARE: DO THIS WRONG AND YOU MIGHT CORRUPT YOUR HARD DISK!**

- 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 ^^^^^^^^^^^^^^^^^^
```

## 2.12. 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 RPi 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 RPi from sources.

**For more guides and projects involving the Raspberry Pi, see RPi Projects**

[http://elinux.org/RPi\\_Projects](http://elinux.org/RPi_Projects)