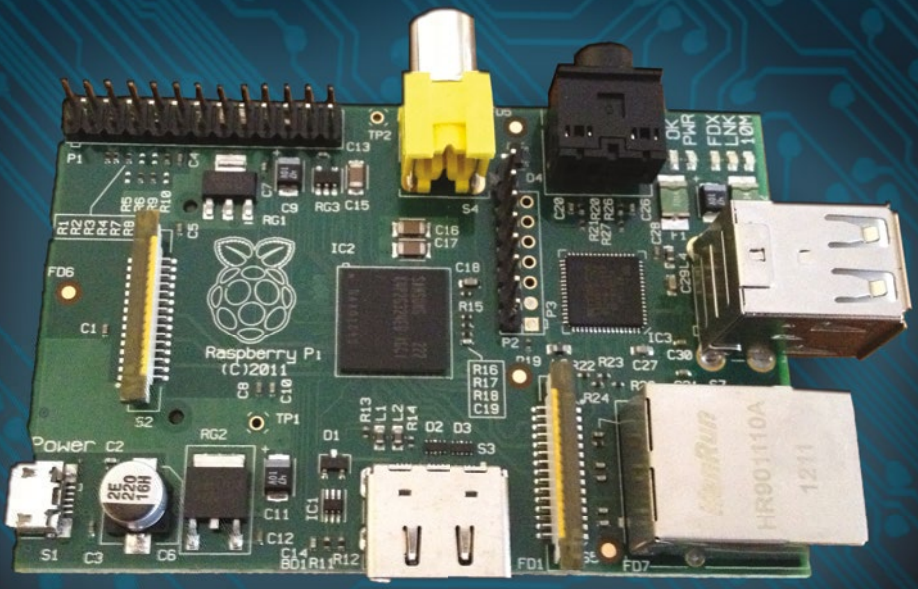
TECHNOLOGY IN ACTION™

Raspberry Pi System Software Reference

AN INDISPENSABLE GUIDE TO LINUX ON THE RPI, FROM MASTERING THE RASPBERRY PI



Warren Gay

For your convenience Apress has placed some of the front matter material after the index. Please use the Bookmarks and Contents at a Glance links to access them.

Apress®

Contents at a Glance

Introduction

The Raspberry Pi, implemented as a System on a Chip (SoC), really does embody the very idea of a "system". There are several large hardware components that make up this complex whole, that we call a system. When we then examine the software side that drives this hardware, we see another body of components that make up the operating system software.

As different as hardware is from software, they form a symbiotic relationship. The hardware provides for external interactions with the world while the software internalizes its inputs and determines the external actions that should result.

Content of this Book

This particular book is focused on system software aspects of the Raspberry Pi. The content was extracted from the complete volume "Mastering the Raspberry Pi" for those that want to focus on this area alone.

While the Pi is not restricted to running one operating system, this title assumes that the student will be working with Raspbian Linux. This is the platform supported by the Raspberry Pi Foundation and as a result, will represent the easiest path for learning.

Chapter 1 covers some very basic things that the beginning student may want to hit the ground running with. Things like setting up a static IP address, using ssh and VNC. Chapter two then jumps right into what happens when your Raspberry Pi boots. Files connected with booting and boot configuration is covered as a reference. But what happens after booting? Chapter 3 looks at how Raspbian Linux pulls itself up from its bootstraps. It examines how services get started and terminated.

Chapter 4 documents the vcgencmd command, which is unique to the Raspberry Pi. It reports and configures special aspects of the hardware. The Linux Console chapter covers the configuration of Linux consoles, including serial port consoles.

Chapters 6 and 7 are important to software developers of the Pi. The first is dedicated to the building and installing a cross-compiler environment for faster, more convenient development on fast hardware like your desktop system. For those that want to modify their Linux kernel, the last chapter is for you. This chapter will guide you through the steps needed to customize and build the Raspbian Linux kernel.

Assumptions about the Reader

Linux tends to be used as a catchall name to include more than just the kernel. In fact some people suggest that it should be referred to as GNU/Linux instead. This is because so much of the involved software is actually provided by the Free Software Foundation

(FSF), aside from the Linux kernel. However you wish to frame it, the reader is assumed to either have some experience with Linux, or is developing some along the way.

For the section about Raspbian Linux initialization, knowledge of shell programming is an asset when creating or modifying the system startup procedures. Otherwise, a basic concept of Linux processes is sufficient for understanding.

Users of cross-compilers for application software development are expected to have some familiarity with the Linux developer tools. This includes the make command and the compiler/linker. Building the kernel can be done by the less experienced, but developer experience is an asset.

Working with the Raspberry Pi often results in bumping into a number of terms and acronyms like GPU or TCP. This book has assumed an intermediate to advanced level audience and consequently these terms are generally not explained. For readers encountering these terms for the first time, the Glossary in Appendix A is there to help.

Finally, as the end user installs or configures his Raspberry Pi, some of the example administration commands found in Appendix C may be useful as a cheat sheet. In most cases, the commands necessary will have already been presented in the text where needed. Mac OS X users will also find Mac specific tips in Appendix E.

CHAPTER 1

Preparation

While it is assumed that you've already started with the Raspberry Pi, there may be a few things that you want to do before working through the rest of this book. For example, if you normally use a laptop or desktop computer, you may prefer to access your Pi from there. Consequently, some of the preparation in this chapter pertains to network access.

If you plan to do most or all of the projects in this book, I highly recommend using something like the Adafruit Pi Cobbler (covered later in this chapter). This hardware breaks out the GPIO lines in a way that you can access them on a breadboard. If you're industrious, you could build a prototyping station out of a block of wood. I took this approach but would buy the Adafruit Pi Cobbler if I were to do it again (this was tedious work).

Static IP Address

The standard Raspbian SD card image provides a capable Linux system, which when plugged into a network, uses DHCP to automatically assign an IP address to it. If you'd like to connect to it remotely from a desktop or laptop, then the dynamic IP address that DHCP assigns is problematic.

There are downloadable Windows programs for scanning the network. If you are using a Linux or Mac host, you can use Nmap to scan for it. The following is an example session from a MacBook Pro, using the MacPorts collection `nmap` command. Here a range of IP addresses are scanned from 1–254:

```
$ sudo nmap -sP 192.168.0.1-254
Starting Nmap 6.25 (http://nmap.org) at 2013-04-14 19:12 EDT
. . .
Nmap scan report for mac (192.168.0.129)
Host is up.
Nmap scan report for rasp (192.168.0.132)
Host is up (0.00071s latency).
MAC Address : B8:27:EB:2B:69:E8 (Raspberry Pi Foundation)
Nmap done : 254 IP addresses (6 hosts up) scanned in 6.01 seconds
$
```

In this example, the Raspberry Pi is clearly identified on 192.168.0.132, complete with its MAC address. While this discovery approach works, it takes time and is inconvenient.

If you'd prefer to change your Raspberry Pi to use a static IP address, you can find instructions in the "Wired Ethernet" section in Chapter 7 of *Raspberry Pi Hardware Reference* (Apress, 2014).

Using SSH

If you know the IP address of your Raspberry Pi or have the name registered in your hosts file, you can log into it by using SSH. In this example, we log in as user `pi` on a host named `rasp` (in this example, from a Mac):

```
$ ssh pi@rasp
pi@rasp's password:
Linux raspberrypi 3.2.27+ #250 PREEMPT ... armv6l
...
Last login : Fri Jan 18 22:19:50 2013 from 192.168.0.179
$
```

Files can also be copied to and from the Raspberry Pi, using the `scp` command. Do a `man scp` on the Raspberry Pi to find out more.

It is possible to display X Window System (X-Window) graphics on your laptop/desktop, if there is an X-Window server running on it. (Windows users can use Cygwin for this, available from `www.cygwin.com`.) Using Apple's OS X as an example, first configure the security of your X-Window server to allow requests. Here I'll take the lazy approach of allowing all hosts (performed on the Mac) by using the `xhost` command:

```
$ xhost +
access control disabled, clients can connect from any host
$
```

From the Raspberry Pi, connected through the SSH session, we can launch Xpdf, so that it opens a window on the Mac:

```
$ export DISPLAY=192.168.0.179:0
$ xpdf &
```

Here, I've specified the Mac's IP address (alternatively, an `/etc/hosts` name could be used) and pointed the Raspberry Pi to use the Mac's display number `:0`. Then we run the `xpdf` command in the background, so that we can continue to issue commands in the current SSH session. In the meantime, the Xpdf window will open on the Mac, while the Xpdf program runs on the Raspberry Pi.

This doesn't give you graphical access to the Pi's desktop, but for developers, SSH is often adequate. If you want remote graphical access to the Raspberry's desktop, see the next section, where VNC is introduced.

VNC

If you're already using a laptop or your favorite desktop computer, you can conveniently access your Raspberry Pi's graphical desktop over the network. Once the Raspberry Pi's VNC server is installed, all you need is a VNC client on your accessing computer. Once this is available, you no longer need a keyboard, mouse, or HDMI display device connected to the Raspberry Pi. Simply power up the Pi on your workbench, with a network cable plugged into it.

You can easily install the VNC server software on the Pi at the cost of about 10.4 MB in the root file system. The command to initiate the download and installation is as follows:

```
$ sudo apt-get install tightvncserver
```

After the software is installed, the only remaining step is to configure your access to the desktop. The `vncserver` command starts up a server, after which you can connect remotely to it.

Using SSH to log in on the Raspberry Pi, type the following command:

```
$ vncserver :1 -geometry 1024x740 -depth 16 -pixelformat rgb565
```

You will require a password to access your desktop.

```
Password:
Verify:
Would you like to enter a view-only password (y/n ) ? n
New 'X' desktop is rasp:1

Creating default startup script/home/pi/.vnc/xstartup Starting applications
specified in/home/pi/.vnc/xstartup
Log file is/home/pi/.vnc/rasp:1.log
$
```

The password prompts are presented only the first time that you start the VNC server.

Display Number

In the `vncserver` command just shown, the first argument identifies the display number. Your normal Raspberry Pi X-Window desktop is on display `:0`. So when you start up a VNC server, choose a new unique display number like `:1`. It doesn't have to be the number 1. To a limited degree, you can run multiple VNC servers if you find that useful. For example, you might choose to start another VNC server on `:2` with a different display resolution.

Geometry

The `-geometry 1024x740` argument configures the VNC server's resolution in pixels. This example's resolution is unusual in that normally 1024×768 would be used for a display resolution, a common geometry choice for monitors. But this need not be tied to a *physical* monitor resolution. I chose the unusual height of ×740 to prevent the VNC client program from using scrollbars (on a Mac). Some experimentation may be required to find the best geometry to use.

Depth

The `-depth 16` argument is the pixel-depth specification. Higher depths are possible, but the resulting additional network trafficc might curb your enthusiasm.

Pixel Format

The last command-line argument given is `-pixelformat rgb565`. This particular example specifies that each pixel is 5 bits, 6 bits, 5 bits—for red, green and blue, respectively.

Password Setup

To keep unauthorized people from accessing your VNC server, a password is accepted from you when you start the server for the first time. The password chosen can be changed later with the `vncpasswd` command.

Server Startup

If you often use VNC, you may want to define a personal script or alias to start it on demand. Alternatively, have it started automatically by the Raspberry Pi as part of the Linux initialization. See Chapter 3 for more information about initialization scripts.

VNC Viewers

To access your VNC server on the Raspberry Pi, you need a corresponding VNC viewer on the client side. On the Mac, you can use the MacPorts collection to install a viewer:

```
$ sudo port install vnc
```

Once the viewer is installed, you can access your VNC server on the Raspberry Pi at 192.168.0.170, display `:1`, with this:

```
$ vncviewer 192.168.0.170:1
```

If you have your Raspberry Pi in the hosts file under `rasp`, you can use the name instead:

```
$ vncviewer rasp:1
```

When the VNC viewer connects to the server, you will be prompted for a password. This obviously keeps others out of your VNC server.

For Ubuntu Linux, you can install the xvnc4viewer package. For Windows, several choices are available, such as RealVNC and TightVNC.

If you find that the screen resolution doesn't work well with your client computer, experiment with different VNC server resolutions (`-geometry`). I prefer to use a resolution that doesn't result in scrollbars in the viewer. Scrolling around your Raspberry Pi desktop is a nuisance. You can eliminate the need for scrolling by reducing the geometry dimensions.

Stopping VNC Server

Normally, you don't need to stop the VNC server if you are just going to reboot or shut down your Raspberry Pi. But if you want to stop the VNC server without rebooting, this can be accomplished. Supply the display number that you used in the VNC server startup (`:1` in this example) using the `-kill` option:

```
$ vncserver -kill :1
```

This can be useful as a security measure, or to save CPU resources when the server isn't being used. This can also be useful if you suspect a VNC software problem and need to restart it.

Prototype Station

The danger of working with the tiny Raspberry Pi's PCB is that it moves all over the surface as wires tug at it. Given its low mass, it moves easily and can fall on the floor and short wires out in the process (especially around curious cats).

For this reason, I mounted my Raspberry Pi on a nice block of wood. A small plank can be purchased from the lumberyard for a modest amount. I chose to use teak since it looks nice and doesn't crack or warp. Even if you choose to use something like the Adafruit Pi Cobbler, you may find it useful to anchor the Raspberry Pi PCB. Mount the PCB on the wood with spacers. Figure 1-1 shows my prototype station.

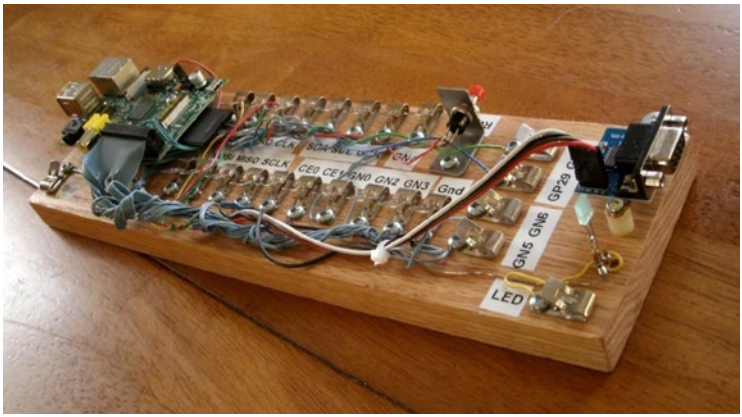


Figure 1-1. *A simple prototype station*

Retro Fahnestock clips were installed and carefully wired to a connector on header strip P1 (the wiring was the most labor-intensive part of this project).

■ **Tip** Fahnestock clips can be economically purchased at places like www.tubesandmore.com (part # S-H11-4043-6).

A small PCB for the RS-232 interface was acquired from eBay ($2.32 total) and mounted at the end of the station. Wires from the RS-232 PCB were routed back to RX/TX and +3.3 V clips and simply clipped into place (this allows you to disconnect them, if you wish to use those GPIO pins for some other purpose). The RS-232 PCB is permanently grounded for convenience.

The RS-232 PCB is necessary only for those who wish to use a serial console or to interface with some other serial device. The PCB acquired was advertised on eBay as "MAX232CSE Transfer Chip RS-232 To TTL Converter Module COM Serial Board." The converter (based on the MAX232CSE chip) will work with TTL or 3.3 V interfaces. Connecting the RS-232 converter's VCC connection to the Raspberry Pi +3.3 V supply makes it compatible with the Pi.

■ **Caution** Do not connect the RS-232 converter to +5 V, or you will damage the Pi. For additional information about this, see Chapter 9 of *Raspberry Pi Hardware Reference* (Apress, 2014).

In Figure 1-1 you can see a simple bracket holding a small push button (top right). This has been wired up to P6 for a reset button. This is not strictly required if your power supply is working correctly (power-on reset works rather well). Unlike an AVR setup, *you are not likely to use reset very often*. Chapter 3 of *Raspberry Pi Hardware Reference* (Apress, 2014) has more details about this.

The LED was added to the station last. It was soldered to a pair of half-inch finishing nails, nailed into the wood. The LED's cathode has a 220 Ω resister soldered in series with it to limit the current and wired to ground. The anode is connected to the Fahnestock clip labeled LED. The LED can be tested by connecting an alligator lead from the LED clip to the +3.3 V supply clip (this LED also tolerates +5 V). Be sure to choose a low- to medium-current LED that requires about 10 mA or less (16 mA is the maximum source current from a GPIO pin).

To test your prototyping station, you may want to use the script listed in the "GPIO Tester" section in Chapter 10 of *Raspberry Pi Hardware Reference* (Apress, 2014). That script can be used to blink a given GPIO pin on and off in 1-second intervals.

Adafruit Pi Cobbler

A much easier approach to prototype connections for GPIO is to simply purchase the Adafruit Pi Cobbler kit, which is available from the following site:

`learn.adafruit.com/adafruit-pi-cobbler-kit/overview`

This kit provides you with these features:

- Header connector for the Pi's P1
- Ribbon cable
- Small breakout PCB
- Breakout header pins

After assembly, you plug the ribbon cable onto the header P1. At the other end of the ribbon cable is a small PCB that provides 26 pins that plug into your prototype breadboard. A small amount of assembly is required.

Gertboard

Students might consider using a Gertboard, which is available from this site:

`uk.farnell.com`

The main reason behind this recommendation is that the Raspberry Pi's connections to the outside world are sensitive, 3.3 V, and vulnerable to static electricity. Students will want to connect all manner of buttons, switches, motors, and relays. Many of these interfaces require additional buffers and drivers, which is what the Gertboard is there for.

In addition to providing the usual access to the Pi's GPIO pins, the Gertboard also provides these features:

- Twelve *buffered* I/O pins
- Three push buttons
- Six open collector drivers (up to 50 V, 500 mA)
- A motor controller (18 V, 2 A)
- A two-channel 8/10/12 bit digital-to-analog converter
- A two-channel 10-bit analog-to-digital converter
- A 28-pin DIP ATmega microcontroller

This provides a ready-made learning environment for the student, who is anxious to wire up something and just "make it work." Many of the 3-volt logic and buffering concerns are eliminated, allowing the student to focus on projects.

Bare Metal

Despite the availability of nice adapters like the Gertboard, the focus of this text is on interfacing directly to the Pi's 3 V GPIO pins. Here are some of the reasons:

- No specific adapter has to be purchased for the projects in this book.
- Any specified adapter can go out of production.
- You'll not likely use an expensive adapter on each *deployed* Pi.
- Bare metal interfacing will exercise your design skills.

If we were to do projects with only wiring involved, there wouldn't be much learning involved. Facing the design issues that arise from working with weak 3 V GPIOs driving the outside world will be much more educational.

The third bullet speaks to finished projects. If you're building a robot, for example, you're not going to buy Gertboards everywhere you need to control a motor or read sensor data. You're going to want to economize and build that yourself. This book is designed to help you *face* those kinds of challenges.