



Free DVD!

NOOBS 1.9.0

Choose your favorite Pi distro...

Raspbian • OpenELEC • OSMC • RISC OS • Windows 10 IoT Core



12

**COOL
RASPBERRY PI
PROJECTS!**



Raspberry Pi

LINUX Special
MAGAZINE

HANDBOOK

4th Edition!

Beginners Welcome!

Get started ordering, assembling, and hacking your Raspberry Pi!

**GAMES AND
PROGRAMMING**
Tools for kids!

EXPLORE THE AMAZING
\$35 Computer

From novice to expert with this
handy single-volume reference!

SUPPORTS
ALL PI MODELS
INCLUDING
THE NEW
RASP PI 3!



Expert Techniques

- Program the GPIO
- Build the Pi camera into your own projects
- Integrate your Rasp Pi with Lego Mindstorms
- Create a home multimedia center

£7.99

Issue #25



Linux Magazine Special

25

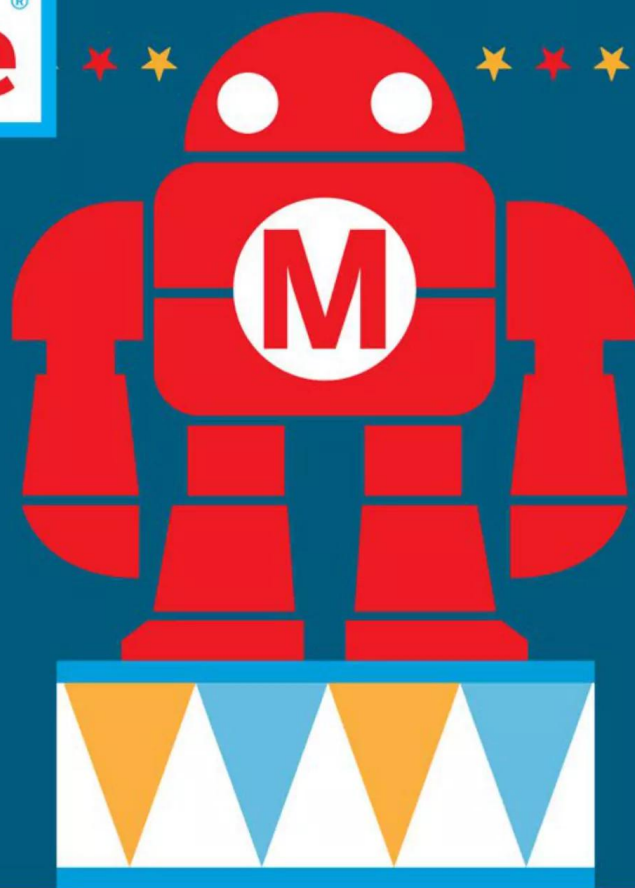
WWW.LINUX-MAGAZINE.COM

2nd

ANNUAL NATIONAL

Maker Faire®

GREATEST
Show
& TELL
ON
Earth



“THE MESSAGE OF MAKER FAIRE IS: ‘You can do this project and the people here can help!’

These young minds will change or save the world.”

— EEJournal, Oct 2015

JUN 18+19 2ND ANNUAL NATIONAL MAKER FAIRE
UNIVERSITY OF THE DISTRICT OF COLUMBIA

OCT 1+2 7TH ANNUAL WORLD MAKER FAIRE
NEW YORK HALL OF SCIENCE

GET YOUR
TICKETS
TODAY!



makerfaire.com

BROUGHT TO YOU BY **Make:**



Joe Casad, Editor in Chief

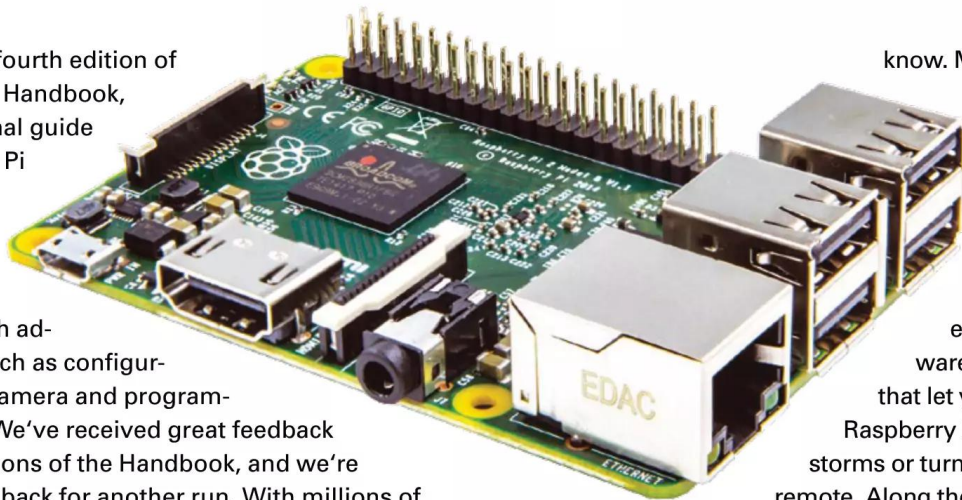
Tasting Raspberry Pi

Welcome to the fourth edition of the Raspberry Pi Handbook, your own personal guide to the Raspberry Pi experience. This single-volume bookazine will lead you from first boot through advanced topics such as configuring the Rasp Pi camera and programming the GPIO. We've received great feedback on previous editions of the Handbook, and we're proud to bring it back for another run. With millions of Raspberry Pi computers out in the world, we know this new edition will find enthusiastic readers who are new to the Pi and looking for a way to get started.

What is the Raspberry Pi phenomenon? What does it mean? What does it say about the world? No marketing guru envisioned it. No tech "expert" predicted it. We all just looked at the news one day, and Raspberry Pi had arrived.

No matter who you are or what your background is, the Raspberry Pi calls you to be an electrical engineer and think like a programmer – the perfect adventure for curious souls who have spent a lifetime pointing and clicking in vacuous GUIs that strive to make the computer itself disappear completely from the experience.

The Raspberry Pi Handbook will show you some basic skills you'll need to work with the Raspberry Pi and lead you through an exciting collection of projects that bring you closer to the system and show you what the experts



know. Make your Pi into a tiny web server or media center. Connect to your Raspberry Pi from a remote location. We even explore some hardware hacking projects that let you connect your Raspberry Pi to Lego Mindstorms or turn your Pi into an IR remote. Along the way, you'll learn techniques for extending, adapting, and interfacing the Pi that will help you create your own Raspberry Pi projects.

Also enclosed in this issue is a free DVD with NOOBS 1.9.0. NOOBS provides a handy installation manager that lets you choose from a number of popular Raspberry Pi distros, including Raspbian, OpenELEC, OSMC, RISC OS, and Windows 10 IoT Core. See page 6 for more on our featured DVD.

What you won't find attached to this magazine is the Raspberry Pi itself. Raspberry Pi computers are available from many online electronics vendors. See the Raspberry Pi website (<http://www.raspberrypi.org/>) for more on how to order from an approved vendor, or just search the site of your favorite electronics shop. See the "What You'll Need" box in the "First Steps" article for a list of other supplies you'll need along with your Raspberry Pi.

Read on, and good luck with your Raspberry Pi adventure!

CODE FOR THIS ISSUE

Code for some of the listings referenced in this issue is available in electronic form at our anonymous FTP site: ftp://ftp.linux-magazine.com/pub/listings/magazine/RaspPi_Handbook/

Raspberry Pi HANDBOOK

Welcome to the Raspberry Pi Handbook, a single-volume bookazine with all you need to install, configure, and discover the amazing Raspberry Pi computer. Read on for a gallery of exciting projects that highlight the best features of the Raspberry Pi environment ...

84 **Camera Module:** Build a camera into your Rasp Pi projects.



Get Started Skills

8 First Steps

We describe how to assemble your Pi, install an operating system, and configure your computer.

18 Understanding Linux

Raspbian is a fully functional Linux operating system. If you're new to the free Linux OS, this crash course will help you get your bearings.

Service

3 Welcome

6 DVD

98 Masthead

24 Raspbian LXDE

Explore the graphical user interface available for your Raspberry Pi system.

27 NOOBS 1.4

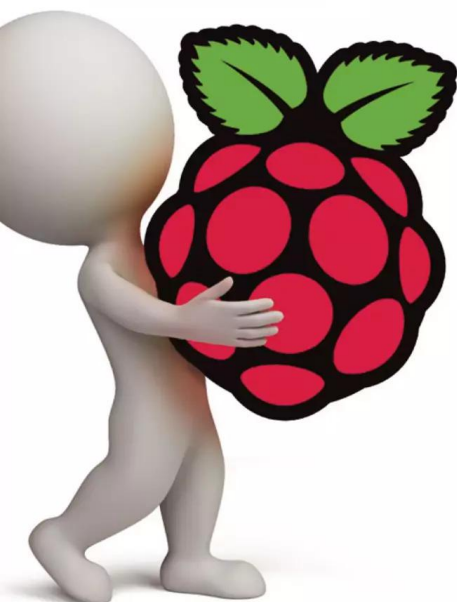
The NOOBS boot manager helps beginners try out various Rasp Pi operating systems.

30 Pi Software

Raspbian comes with an exciting assortment of desktop apps, games, and programming tools.

34 Scratch Programming

Write cool programs with Scratch – a programming language for beginners.





46 Kodi Media Center: Use your Raspberry Pi as a do-it-yourself home theater system.



88 Programming the GPIO: Expand and extend your projects by integrating advanced programming techniques into your Pi creations.

Server Tricks

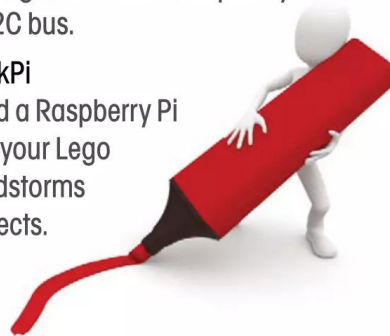
- 38 Pi File and Print Servers**
File service, automated backup, and print service for the home network.
- 42 A Pi Web Server**
Test your web creations with a tiny Pi web server.

- 46 Kodi Media Center**
With the free Kodi software, turn your Raspberry Pi into a media center with a fancy interface – and a whole lot more.
- 52 Remote Access**
Use SSH for remote connections to your Raspberry Pi.

Hardware Hacks

- 56 Interfacing for Beginners**
Get started connecting your Pi to external devices.
- 60 Steady Hands**
A game to test your fine motor skills.
- 64 IR Remote**
Turn a Raspberry Pi into an IR remote control.

- 68 I2C Bus: Basics**
Getting to know the Raspberry Pi I2C bus.
- 73 BrickPi**
Build a Raspberry Pi into your Lego Mindstorms projects.

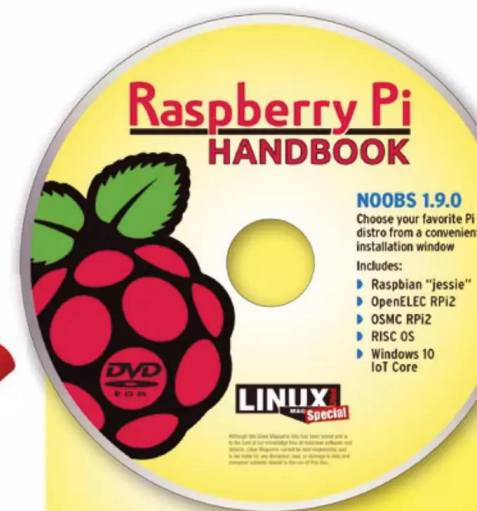


More Fun

- 76 The Rasp Pi Display**
The new Rasp Pi display provides a compact option for viewing screen output – at a Pi-like low price of \$60.
- 80 Retro Gaming on the Pi**
Transform your Pi into a platform for classic gaming.



- 84 Rasp Pi Camera Module**
The Raspberry Pi camera module opens up a whole new world of useful projects.
- 88 Programming the GPIO**
Export ports, handle interrupts, and program the general-purpose I/O.
- 92 Mathematica on the Pi**
Wolfram Mathematica and Wolfram language are pre-installed on the Pi.



NOOBS 1.9.0

Choose your favorite Pi distro from a convenient installation window

- ▶ Raspbian "jessie"
- ▶ OpenELEC RPi2
- ▶ OSMC RPi2
- ▶ RISC OS
- ▶ Windows 10 IoT Core

On the DVD

NOOBS 1.9.0

NOOBS (New Out Of the Box Software) provides an easy way to install operating systems on your Rasp Pi. The NOOBS installation manager offers the following operating systems:

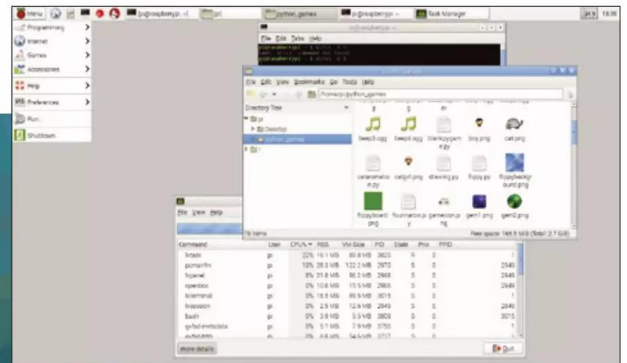
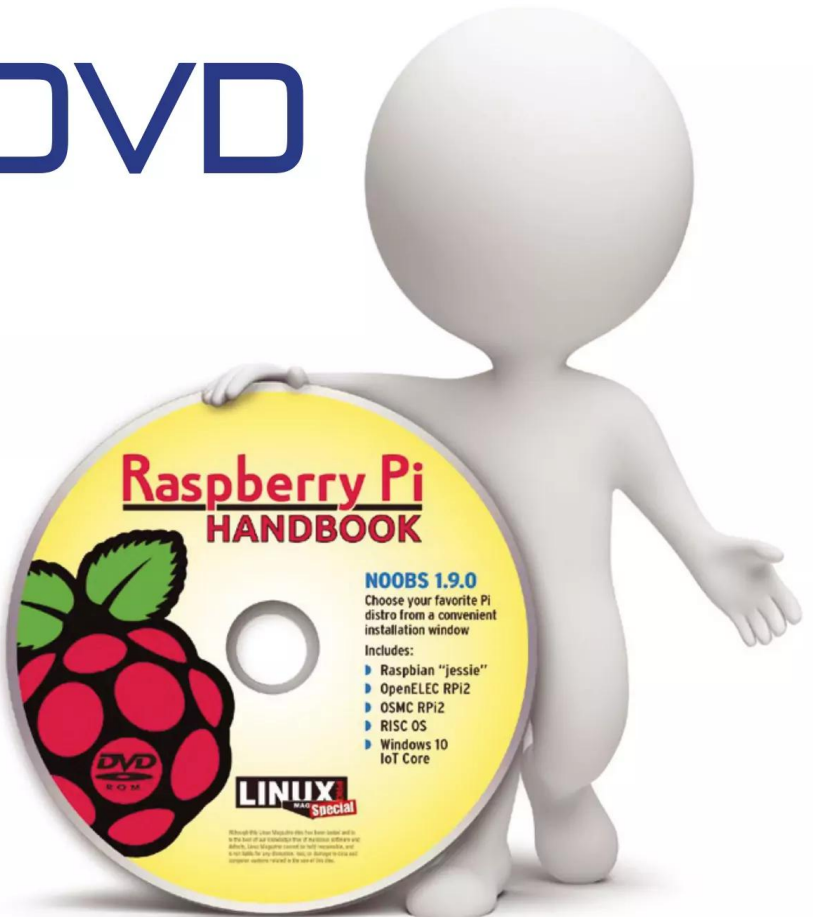
Install from the DVD

- Raspbian “jessie” – Debian-based distribution recommended for beginning users. Network Installation (must be connected to the Internet)

Install from the Internet

- OpenELEC (RPi2 and RPi3) – Open Embedded Linux Entertainment Center is a Linux distribution that turns your Raspberry Pi into a Kodi media center.
- OSMC (RPi2 and RPi3) – Formerly called Raspbmc, OSMC is another Kodi media center distro.
- RISC OS – Not Linux, but a separate operating system that has been around since 1987 and traces its roots to some of the original developers of the ARM chip.
- Windows 10 IoT Core (RPi2 and RPi3) – Run Universal Windows Platform apps with this Pi-ready version of Windows.

The Raspbian image is located on the SD card and can be installed without access to a network. Other options require a network connection for network-based installation.



RESOURCES

- [1] Getting Started with NOOBS: <http://www.raspberrypi.org/help/noobs-setup/>
- [2] Raspbian: <http://www.raspbian.org/>
- [3] OpenELEC: <http://openelec.tv/>
- [4] OSMC: <https://osmc.tv>
- [5] RISC OS: <https://www.riscosopen.org/content/>
- [6] Windows 10 IoT Core: <https://dev.windows.com/iot>
- [7] SD Formatter (Windows and Mac): https://www.sdcard.org/downloads/formatter_4/
- [8] Installing operating system images (Linux/Mac/Windows): <https://www.raspberrypi.org/documentation/installation/installing-images/>

Outside the Box

The DVD enclosed with this issue comes with a collection of some of the best and most popular Raspberry Pi distributions. To install a distribution from the NOOBS installation manager, format an SD card of at least 8MB with an SD card formatter (Windows and Mac) [7] and drag the files inside the uncompressed NOOBS folder to the SD card. For more information, and for instructions on how to prepare an SD card on Linux, check out the instructions on the Raspberry Pi website [8] or in this magazine. Finally, unmount the SD card, insert it into the Raspberry Pi, and reboot.

In the NOOBS boot menu, select the operating systems you want to install. To try different systems later, hold down the Shift key at the Raspberry splash screen to bring back the NOOBS boot menu.

LOST YOUR BOOKSTORE?

LET US BE YOUR BOOKSTORE

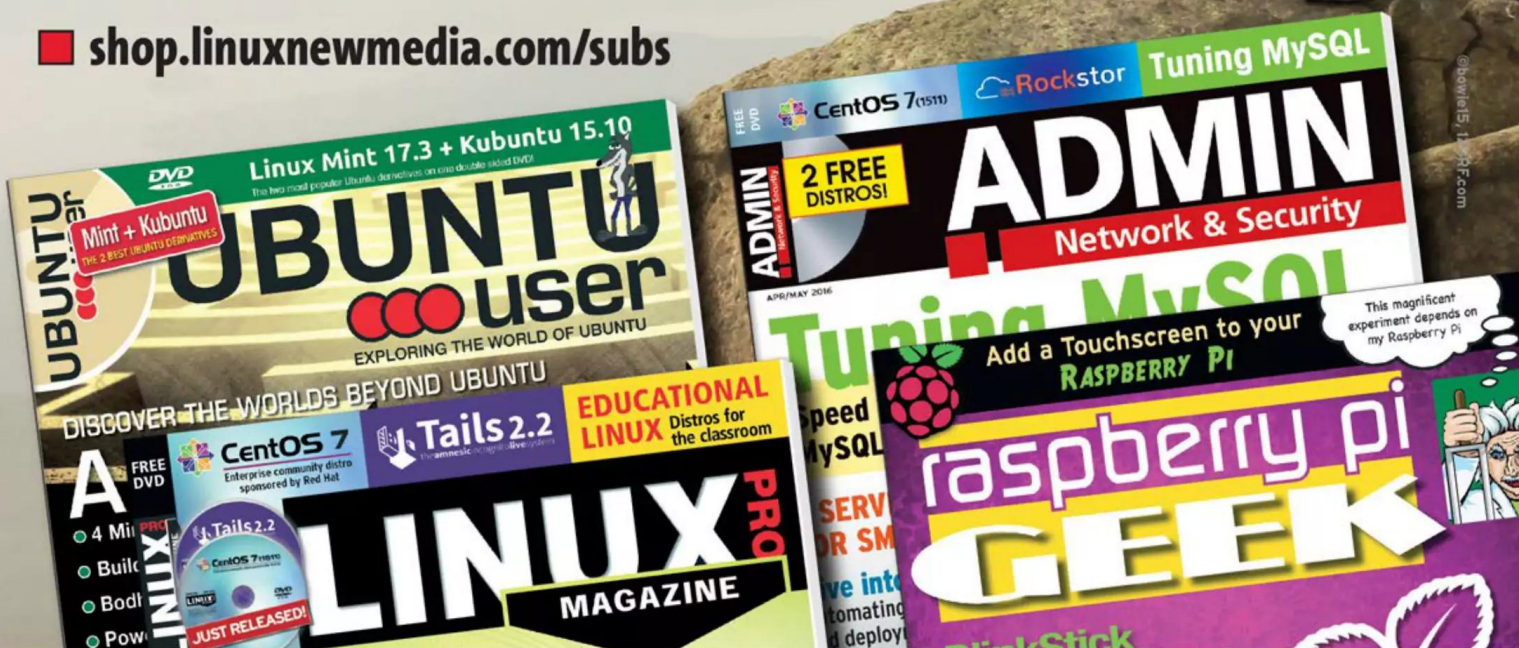


Browse our shop for single issues of *ADMIN*, *Linux Pro*, *Linux Magazine*, *Ubuntu User*, and *Raspberry Pi Geek* – delivered right to your door.

■ shop.linuxnewmedia.com/single

Better yet, subscribe, and you won't need a bookstore.

■ shop.linuxnewmedia.com/subs



shop.linuxnewmedia.com

DIGITAL AND PRINT EDITIONS AVAILABLE!



Assembling and starting your Raspberry Pi

First Bite of Pi

Your shiny Raspberry Pi computer just arrived in the mail – now what? We'll show you how to assemble your Pi and boot it with the easy and popular Raspbian operating system.

By Paul C. Brown, Joseph Guarino, and Joe Casad



The arrival of the Raspberry Pi small-board computer (SBC) introduced a whole new world of hardware and software hacking fun. The Raspberry Pi was developed with the goal of inspiring us all to explore the world of computer science. The Raspberry Pi Foundation, which develops and releases the Raspberry Pi, is particularly focused on promoting computer sciences to students in primary and secondary schools. The Raspberry Pi (RPi) allows anyone with between \$5 and \$35 to tinker with any number of juicy and exciting comp-sci tasks. Almost as soon as it was released, the Raspberry Pi was put to work at everything from robotics to weather controller, and even home brewing. The Pi is

sweet hacking bliss no matter what your project aim.

The Raspberry Pi comes in several models (see Table 1). Models A and A+ are designed for low-resource, low-cost scenarios and are lighter on memory and power usage. Models B and B+, with 256 or 512MB of RAM (RPi1) and a 700MHz single-core ARMv6 processor, were the most popular versions for schools and hobbyists until February 2015, when the RPi2 Model B (RPi2) arrived with an upgraded quad-core processor and 1GB of RAM (Figure 1). Then, in February 2016, the Raspberry Pi Foundation released the third-generation Model B (RPi3) with a new, even more powerful 64-bit quad-core processor.

TABLE 1: Raspberry Pi Consumer Models*

Model	Price	CPU	SDRAM (shared with GPU)	USB Ports	Storage	GPIO (pins)	Power Rating
RPi1A	\$25	700MHz ARMv6	256MB	1	SD MMC	26	300mA/1.5W
RPi1A+	\$20	700MHz ARMv6	256MB	1	MicroSD	40	200mA/1.0W
RPi1B Rev 2	\$35	700MHz ARMv6	256 or 512MB	2	SD MMC	26	700mA/3.5W
RPi1B+	\$25	700MHz ARMv6	256 or 512MB	4	MicroSD	40	600mA/2.0W
RPi2B	\$35	900MHz quad-core ARMv7	1GB	4	MicroSD	40	800mA/4.0W
RPi3B	\$35	1.2GHz quad-core ARMv8	1GB	4	MicroSD	40	800mA/4.0W
Pi Zero	\$5	1GHz ARMv6	512MB	1 USB OTG	MicroSD	40	160mA/0.8W

* The Raspberry Pi Compute Module, intended for embedded systems prototyping, is not shown in this table.

Lead Image © Corina Rosu, 123RF.com

In a separate development stream, the Pi Zero, which measures about 2-1/2 by 1-1/4 inches, is meant to fill the super-low-cost niche at \$5. You'll have to get an adapter for the mini-HDMI port, a microUSB to USB OTG adapter to hook up a USB hub, and a microUSB power adapter. The GPIO and composite video headers are "unpopulated," which means you have to solder the pins in yourself. Otherwise, the Zero is a fully functional Raspberry Pi with the processor used in the first generation of Pis, but running at 1GHz.

The Raspberry Pi Foundation revised the design of the RPi1 in the fall of 2012. The new boards were known as Rev 2 systems. The previous Rev 1 systems had a slightly different layout, including some differences in the configuration of the GPIO pins, which is described in the "Hardware Hacks" section of this issue.

If you already own first- and second-generation Pis, you don't need to cast them away unless you are specifically looking for better performance. This special edition was originally created for the RPi1 Rev 2, but all the exercises will work with the entire line of Raspberry Pis unless otherwise noted.

Be aware that the RPi2 uses the ARMv7 and the RPi3 the ARMv8, rather than the ARMv6 processor of the first-generation Raspberry Pi. The processors have a high degree of compatibility; however, if you're using the RPi2/3, be sure you have an operating system that supports the newer ARM processors. Recent editions of the Raspbian

operating system (OS), including the version on the NOOBS DVD attached to this issue, support the RPi2 and RPi3. The OpenELEC and OSMC media centers on the DVD are suitable for the RPi2/3. (See the article on NOOBS elsewhere in this issue.)

The RPiB+/2/3 boards lack the RCA jack found in earlier versions, but they provide two additional USB ports (four instead of two) and 14 additional GPIO pins (40 instead of 26).

Another important difference between the first and later generation boards is that the more recent Models A+ and B+ and the RPi2/3 use a microSD card, whereas earlier boards use the larger SD MMC. If you only have old-style SD cards around, you won't be able to use them with your newer boards. No worries, though – just burn a new OS to a smaller microSD card.

Buy Some Pi

You can buy your official Pi from a number of resellers. The Raspberry Pi website links

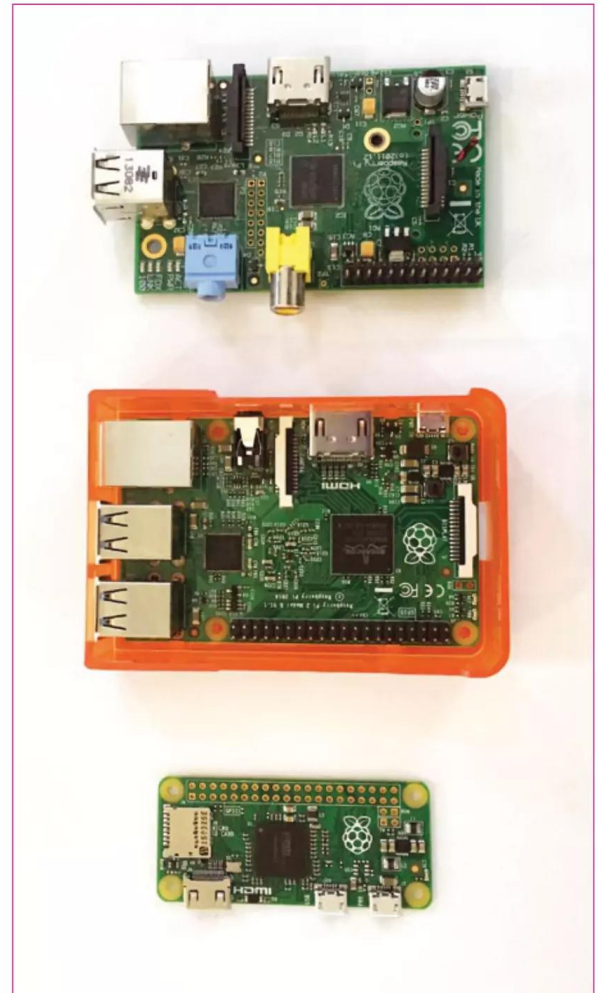


Figure 1: Some of the Raspberry Pi models. The RPi1B (top), the RPi2 (middle), and the Pi Zero (bottom). The RPiB+ and RPi3 form factors are similar to the RPi2.

WHAT YOU'LL NEED

The small Raspberry Pi SBC won't do much for you all alone. Be sure you have the following on hand before you start:

- 1 Power supply
- 1 MMC SD or microSD card – depending on the model of your Rasp Pi
- 1 USB on-the-go (OTG) cable for the Pi Zero
- 1 Powered USB hub
- 1 USB mouse (optional if you will use the command line only)
- 1 USB keyboard
- 1 DVI- or VGA-to-HDMI converter (if needed) to connect your monitor via HDMI, and a mini-HDMI adapter for the Pi Zero
- 1 case (optional). A free PDF template is also available for you to create your own cardboard case [2] [3]
- 1 SD card writer (usually on most computers)
- An Ethernet cable or WiFi dongle, if you have a model earlier than the RPi3, for Internet access
- 40-pin male header for the Pi Zero GPIO and soldering materials

Notes and caveats on the purchase list:

- **Power supply** – Except for the RPiA and Pi Zero, the Raspberry Pi needs a 5V micro-USB power supply that can cope with currents of 600-800mA. This is important for system stability. The other Pis in the series can get by with USB cables often found in conventional phone chargers. Websites offer 5V cables for the Raspberry Pi that range from 1A to 2.4A. Failing to pay attention to the details of these power requirements will leave you with lockups and other system problems. See the Raspberry Pi compatible device list [2] for more information.
- **Powered USB hub** – Using a powered USB hub is a good idea for system stability and expandability. The RPi1B has two USB ports, but you might want to connect more than two devices. The RPiB+, RPi2, and RPi3 have four ports, but a powered hub still helps to minimize the risk of drawing too much power and causing stability issues.

HARDWARE COMPATIBILITY

A number of considerations factor into whether a peripheral will work with your Raspberry Pi, including the operating system you are using, the CPU (i.e., which Pi?), and the availability of drivers that work with both.

For example, many people have reported problems using USB 3.0 hubs, especially with the Pi Zero. Before you spend money on WiFi dongles, hubs, and other add-ons to your Raspberry Pi, check out the eLinux.org web site of RPi Verified Peripherals [4].

to a page that lets you browse official vendors around the world [1]. Keep in mind that when you buy a Pi, a few more purchases are required in addition to the unit itself. See the boxes titled “What You’ll Need” and “Hardware Compatibility.”

Getting Raspbian

You might have acquired a Raspberry Pi in kit form, with a protective case, some cables, and an SD card preloaded with Raspbian. If this is the case, and if you’re impatient to get your Pi up and running, you might want to skip this section for the time being and go directly to the “First Boot” section. Then again, you might need to return to this information if you want to create a new card or install a different operating system.

In this article, we describe how to install the Raspbian operating system on your Raspberry Pi. Raspbian is officially sponsored by the Raspberry Pi Foundation and, therefore, can be downloaded from their site [5]. Raspbian is free software, so there is no charge, and once downloaded, you can copy and distribute to your heart’s content. Several other ARM architecture Linux operating systems are also designed to work with the Raspberry Pi (see the article on NOOBS). However, Raspbian is the most common and most popular Rasp Pi system in use today. The projects in this issue mostly use Raspbian for examples and configuration scenarios, but other systems are similar. If you want to try one of these alternative systems, consult the project documentation for how to get the system up and running.

If you download an OS file from the Internet, you might want to check its integrity to make sure the file is complete, has not been tampered with, and does not include any malware. The creators of the file often provide a SHA-1 hash, which is the long string of letters and numbers you see under the link on the download page. Running a program on your downloaded file will produce a similar string. If both the string on the site and the string you produced locally are the same, the file is okay.

On Windows, you can download a program [6] to use on the command line. Linux and Mac OS X have preinstalled tools. On Linux, simply type the following in a terminal window:

```
shasum 2016-03-18-raspbian-jessie.zip
```

On Mac OS X, open a terminal and type:

```
openssl sha1 2016-03-18-raspbian-jessie.zip
```

Note that your Raspbian file might have a different date.

Once you’re happy that the file you have downloaded is okay, you can unzip it and install it on an SD card.

Installing Raspbian on Your SD Card

The unzipped Raspbian image file is not a standard file but contains the Raspbian OS already installed and half configured. This type of file is called an *image file* because it represents a disk image. You can’t just copy the file over to the SD card and expect it to work.

The first step is to format the SD card. Some cards come pre-formatted; if your card is not pre-formatted, you’ll need to format the card with the FAT32 filesystem. Windows and Mac OS systems have built-in tools for formatting partitions. The SD Association recommends that Windows and Mac OS users use their free formatting tool, which is designed specifically for SD cards. You can find the SD formatter at the SD Association website [7].

Linux

Linux users have several options for how to format an SD card. Running the following command with superuser privileges

```
lsblk
```

lists block devices mounted on the system (Figure 2). (This procedure assumes your Linux system is configured to automount disks when inserted.) The SD card appears in the list as Type *disk*. Partitions on the disk appear below the device in a tree structure. Look for a disk with a size that is approximately equal to the size of your SD card, and try to verify that this is the correct device. (Be sure you don’t choose a hard disk!)

If the SD card device has the name `mmcblk0`, as shown in Figure 2, enter the following command to format the card FAT32:

```
mkdosfs -F 32 -v mmcblk0
```

To create a working SD card with Raspbian, you have to do a byte-by-byte copy from the file to the card. Byte-by-byte copies are risky if you are not sure what you’re doing because you completely erase the data on the destination drive. If you get confused and choose the wrong destination (e.g., you choose your hard disk on your computer instead of your SD

card), you could lose all your data. That said, making byte-by-byte copies is not difficult, and you just have to pay attention at key moments in the process.

Both Linux and Mac OS X can use their terminal windows and the `dd` tool to do the copy. Another Linux command you can use to find the disk name is `dmesg`. To find the SD card this way, plug the card into the computer and immediately run the `dmesg` command from a terminal window:

```
$ dmesg
...
[... ] mmc0: new high speed SDHC card at 7
      address 59b4
[... ] mmcblk0: mmc:59b4 7.4GiB
[... ] mmcblk0: p1 p2 < p5 p6 > p3
...
```

The last lines show you where the operating system has found your card. Here, the card appears, as before, at `/dev/mmcblk0`. Be sure you have the right drive! The formatting process destroys all data on the drive.

To use `dd`, the card must be unmounted, that is, not assigned to a directory in the file system. To do so, run

```
sudo umount /dev/mmcblk0
```

(note that the command is spelled “umount”). If the card was not mounted to start with, `umount` will just display an error and exit.

Now navigate to the directory where you downloaded Raspbian:

```
cd </Raspbian/download/directory>
```

where `</Raspbian/download/directory>` is the path to your Raspbian image on Linux (often `~/Downloads`). Then, you can carry out the byte-by-byte copy to the card with:

```
$ sudo dd if=2016-03-18-raspbian-jessie.img
of=/dev/mmcblk0
```

where `if` is the name of the Raspbian distribution, and `of` is the SD device (from the `dmesg` command). This copy will take a while, and `dd` will not show any sort of progress bar.

Mac OS X

On the Mac, format the SD card with the freely downloadable `SDFormatter.app` [7] and run the command

```
diskutil list
```

to find the name of the SD card. Look for the device formatted FAT32 (Figure 3) and note its device name (here, `/dev/disk1`). Now, unmount the disk,

```
diskutil unmountDisk /dev/disk1
```

and navigate to the directory where you downloaded Raspbian:

```
cd ~/Downloads
```

Then, you can carry out the byte-by-byte copy to the card with

```
sudo dd bs=1m
if=2016-03-18-raspbian-jessie.img
of=/dev/rdisk1
```

where `if` is the name of the Raspbian distribution, and `of` is the SD device (from `diskutil list`) prefaced with an ‘r’.

If `bs=1m` bombs (that’s a one, not an ‘el’), try `bs=1M`. If you don’t specify the block size, it will take a very long time to burn the image.

```
jcasad@ThinkPad:~$ lsblk
NAME        MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
sda          8:0    0 232.9G  0 disk
├─sda1       8:1    0 170.9G  0 part /media/jcasad/5042a3cb-a465-448e-829f-cf7eba8874d71
├─sda2       8:2    0    1K  0 part
├─sda3       8:3    0   39.1G  0 part /
└─sda5       8:5    0    5.4G  0 part [SWAP]
sr0         11:0    1 1024M  0 rom
mmcblk0     179:0    0    7.5G  0 disk
├─mmcblk0p1 179:1    0    56M  0 part /media/jcasad/boot
└─mmcblk0p2 179:2    0    7.2G  0 part
jcasad@ThinkPad:~$
```

Figure 2: The output of `lsblk` on a Linux computer. The SD card is `mmcblk0` and has two partitions (which will be lost when formatted to prepare it for the Raspbian OS).

```
Americanos-Mac-mini:~ rsooby$ diskutil list
/dev/disk0
#:          TYPE NAME                    SIZE      IDENTIFIER
0:      GUID_partition_scheme             *500.1 GB   disk0
1:          EFI EFI                      209.7 MB    disk0s1
2:      Apple_HFS Americanso              325.6 GB    disk0s2
3:      Apple_Boot Recovery HD             650.0 MB    disk0s3
4:      Microsoft Basic Data LINUX         173.6 GB    disk0s4
/dev/disk1
#:          TYPE NAME                    SIZE      IDENTIFIER
0:      FDisk_partition_scheme            *4.0 GB     disk1
1:          DOS_FAT_32 RASPI              4.0 GB     disk1s1
Americanos-Mac-mini:~ rsooby$
```

Figure 3: The SD card (`/dev/disk1`) was formatted FAT32 and named RASPI with the Mac SD Formatter tool.

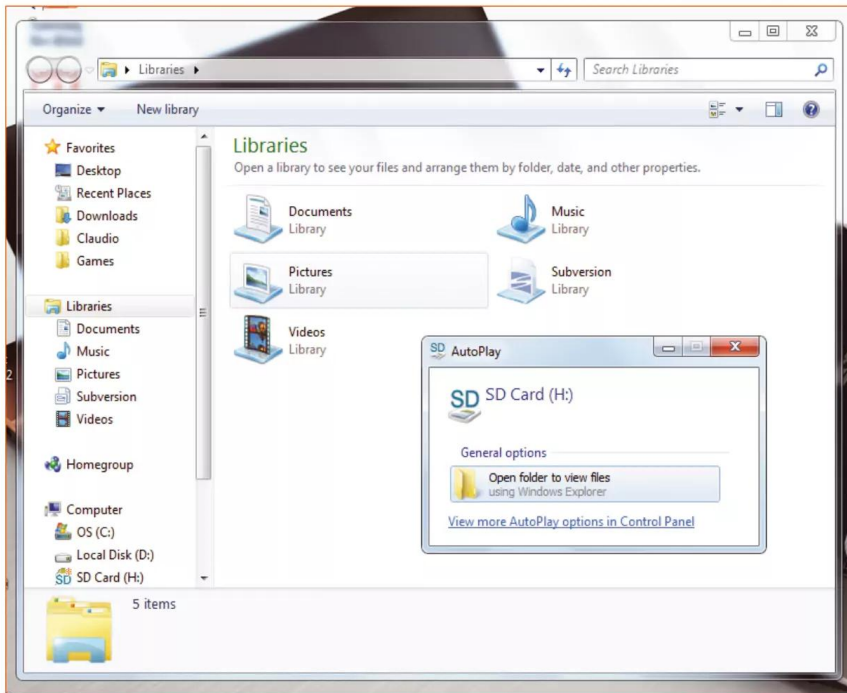


Figure 4: In this case, Windows has assigned the SD Card to drive H:.

Windows

If you are using Windows, format your SD card with the SD Formatter app [7], then you can use a program called Win32 Disk Imager [8] to install Raspbian on the card. Download the program and install it, then plug an SD card into your computer. (If you don't have an SD card reader slot, you can purchase a cheap external adapter that plugs in to a USB port.) Note where Windows mounts the card. If you look at Figure 4, you can see that Windows assigned the card to the H: drive.

Now you can use the Win32 Disk Imager program to do a byte-by-byte copy from the Raspbian file to the card. In the Image File text box, navigate to the image file (e.g., 2016-03-18-raspbian-jessie.img), and in the Device drop-down box, choose the letter assigned to your card (in the case shown in Figure 4, you would choose H:) and start copying. Please note that all data on the destination drive will be lost, so use an empty card or one that contains data you don't mind deleting.

Assembling Your Pi

Once you have loaded the operating system onto your SD card, assemble your Pi into the case of your choice. You can purchase an inexpensive case, build your own, print a case from card stock, use Legos, or explore any number of alternative options. Cases are, of course, optional but they do help you keep your Pi safe from accidental damage or electrostatic discharge.

Next, insert the SD card into the bottom of the Rasp Pi. For the older Pis, you just push in and pull out the SD MMC; the microSD slots (except on the Pi Zero) are spring-loaded, so you push in both to engage and remove the card. You can only insert the card one way for it to function. If you require a hammer, rest assured you are inserting it incorrectly.

Connect your monitor to the Raspberry Pi via the HDMI port. If you have an older monitor that has only VGA or DVI, you will need a converter to connect it.

If you're running a first-generation Pi system, you might want to attach a powered USB hub to increase the available USB ports. Even if you are using an RPiB+/2/3 (with four available ports), you might prefer to use a powered USB hub to avoid potential instability associated with drawing too much power. Next, plug in an Ethernet cable for your local LAN and, finally, plug in your power supply.

First Boot

Earlier versions of Raspbian launched a complicated configuration dialog at first boot, allowing the user to configure the locale, camera, login, and other settings. (See the next section on booting Debian Wheezy.) With the latest Jessie release, the Raspbian developers have simplified the first login by eliminating the choices and booting the system to a default configuration.

Just slip the SD card into the slot and turn on the power. The system will boot directly to the Raspbian desktop. You can then make the necessary configuration changes through the available configuration tools. In particular, the Raspberry Pi Configuration Tool was developed to encapsulate many of the settings a user might want to address when starting a new system.

To open the Raspberry Pi Configuration Tool, click on the *Menu* button in the upper-

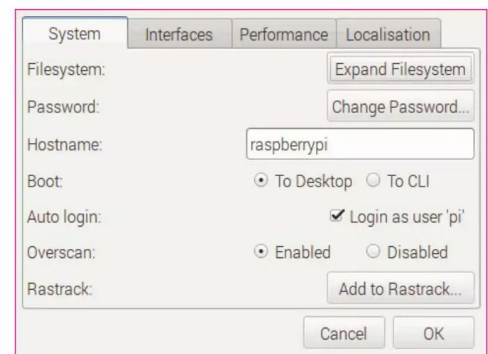


Figure 5: The Raspberry Pi Configuration Tool System tab lets you configure login settings and define system parameters.

right corner of the Raspbian desktop and choose *Preferences | Raspberry Pi Configuration Tool*. The tool opens to the *System* tab (Figure 5), which lets you manage the following options:

- **Expand Filesystem** – in some cases, Raspbian is only able to use part of the space on the SD card. This option expands into all the available space. Your configuration might already be using all the space, in which case a message will tell you this option isn't necessary.
- **Change Password** – the default password for the default account is `raspberry`, which isn't much of a password and, even if it were, everyone knows it. If you are concerned about the security of your system, change the default password. Be sure you write down your password or have a clear means for remembering it.
- **Boot to desktop or CLI** – The articles in this issue assume you are using a graphic desktop system. If you are an experienced user and would prefer to boot to the command line (CLI means "Command Line Interface"), select the CLI option. A checkbox below lets you choose whether you want the system to prompt the user for login credentials at startup or log in automatically.

See the Wheezy discussion in the next section for more on the *Overscan* and *Rastrack* options.

The *Interfaces* tab (Figure 6) lets you choose whether to enable the camera and SSH, which you will learn about in later articles. The *Performance* tab lets you choose whether to overclock the Raspberry Pi system. Overclocking increases the speed but can pose issues for stability, cooling, and durability of your system. You won't need overclocking for the exercises in this issue. Another option in the *Performance* tab lets you set the amount of memory assigned to the Graphics User Interface (GPU). You probably

won't need to change this setting either, but if you're using your Raspberry Pi for graphic-intensive applications, such as gaming or video playback, you might want to experiment with increasing the GPU memory.

The *Localisation* tab includes settings that vary based on your location, such as the language, timezone, and keyboard layout.

The next article in this issue discusses some other options for customizing your user environment, such as adding new applications and changing the appearance of your desktop.

Alternative Configuration Tool: raspi-config

Another way to configure your Raspbian system is using the `raspi-config` utility. `Raspi-config` (Figure 7) used to be the primary configuration tool for Raspbian systems prior to the "Jessie" release. If you're using Raspbian "Wheezy" or an earlier version, `raspi-config` comes up automatically when your system boots for the first time. Even if you're using the latest "Jessie" release, you can still access the `raspi-config` tool with the command:

```
sudo raspi-config
```

Some users prefer `raspi-config` to the GUI-based Raspberry Pi Configuration tool accessible from the *Preferences* menu (described the preceding section). The tools might look different at a glance, but you'll soon notice that many of the configuration options are the same.

To navigate the `raspi-config` menus, use the arrow keys to move up and down; press the Tab key jump to the *Select* option at the bottom screen, then press Enter to select the item.

The first option, *Expand Filesystem*, lets you decide whether you want to expand the root filesystem so that it takes up all the space on the SD card. You can learn more

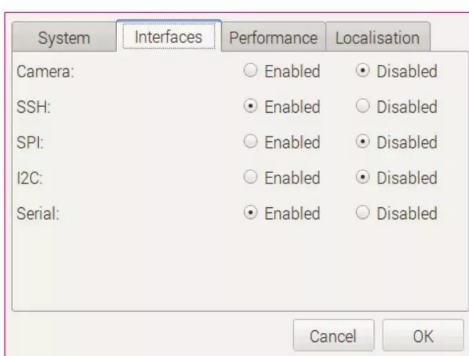


Figure 6: The Configuration Tool's Interfaces tab lets you enable or disable the Rasp Pi camera and SSH.

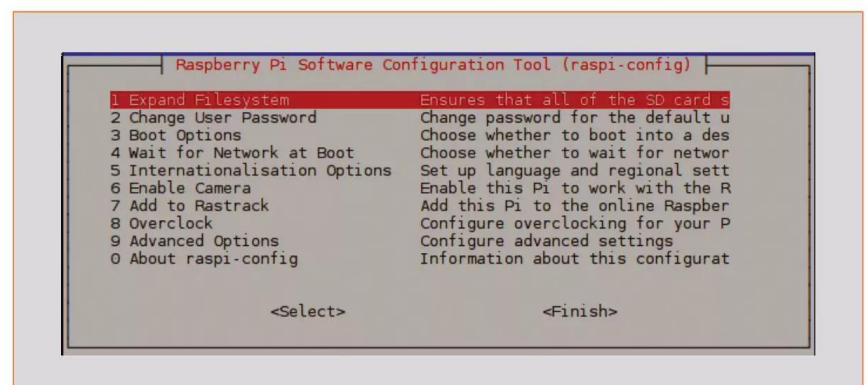


Figure 7: The Raspbian configuration tool allows you to establish the default settings of your Pi.

GET STARTED

First Steps

COMMAND LINE BASICS

If you elect to operate your Raspberry Pi from the command line, you should learn the following common commands:

- `startx` – Start the graphical desktop.
- `sudo halt` – Power down the Pi.
- `sudo reboot` – Restart the Pi.
- `exit` – Log out.

about the Linux filesystem in a later article; for now, go ahead and choose this option. The Raspbian image takes up just less than 4GB. Most SD cards are 8GB or more nowadays. If you don't expand the root filesystem, you'll have 4GB or more that you won't be able to use, and you will find yourself working with very little free space on your SD card, which will be eaten up in minutes; you won't be able to install new apps or save any files. If you are using NOOBs, you'll get a message saying your filesystem is already expanded.

The *Change User Password* option allows you to change the administrator's default password. The administrator (known as *superuser* or *root* in Linux parlance) has complete control over the computer, and anyone who has access to superuser privileges can cause serious damage. Therefore, if your Raspberry Pi is going to be used by more than one user or is going to be open to a local network or the Internet (e.g., as a server), changing the default password is a good idea. The default administrator's username, by the way, is *pi* and the default password is *raspberry*.

The third option, *Boot Options*, lets you boot into the graphical desktop, the console, or automatically into either environment as user *pi*. If you intend to use your Pi as a server (e.g., as a media center on your local network), a graphical desktop only gobbles up resources. In that case, you are advised to boot to the console. When you next reboot, Raspbian will present you with a text login,

where you will land at a command line after typing your username and password. (See the box titled "Command Line Basics.")

Internationalisation Options allow you to change localization settings. The first option in this category, *Change Locale*, sets the language, country, character set, sort order, and so on. When you arrow or page down into the correct box, press the Spacebar to choose. More than one locale can be marked. If you want to deselect the UK (*en-GB*) English default, find its box marked with an asterisk and press the spacebar to deselect. On the next screen, you can set the default language from among the choices you made, then Tab to *Ok*.

The *Change Timezone* option is self-explanatory: Choose the time zone you want to use with your Pi. If your Raspberry Pi is connected to the Internet, Raspbian will automatically try to contact an NNTP server to get the correct GMT time and then add or subtract the number of hours to calculate your local time.

The third localization option, *Change Keyboard Layout*, allows you to establish the model, layout, and language of the keyboard you are using. (If the console kicks out errors and shows that *en-GB.UTF-8* is still the preferred language, refer to the "Bug Box" sidebar.) If you don't see your keyboard in the first screen, arrow down to *Other* and Tab to *Ok*. After a few seconds, you will see a long list of keyboard models. *Generic 105-key (Intl) PC* is the default option and is the most common PC keyboard layout. Once you have chosen your keyboard layout, use the Tab key to highlight *Ok* and hit Enter.

In the following screen, you can choose your keyboard's language, and, in the next screen, the variant of your language (Figure 8). If you select *English (US)*, for example, you will then be able to choose from among a wide range of variants and keyboard models. The next screens let you define your keyboard's modifier key and Com-

TABLE 2: Recommended Clock Rates

	arm_freq (MHz)	core_freq (MHz)	sdrain_freq (MHz)	over_voltage
Raspberry Pi 1 Model B/B+				
None	700	250	400	0
Modest	800	250	400	0
Medium	900	250	450	2
High	950	250	450	6
Turbo	1000	500	600	6
Raspberry Pi 2				
None	900	250	450	0
High	1000	500	500	

pose key (Multi-key) and whether you want to use Ctrl + Alt + Backspace to terminate the X server (graphical environment).

The final internationalization option, *Change Wi-fi Country*, lets you select the country in which you are using your Pi.

The *Enable Camera* option is simple and straightforward, allowing you to enable or disable the Raspberry Pi camera add-on [9], and *Add to Rastrack* allows you to register your Pi on the Rastrack site [10], which maps Raspberry Pi users all over the world. Of course, your Pi has to be connected to the Internet for this to work.

Most computers ship with the CPU set to a certain speed, but the microprocessor is capable of running much faster. Overclocking is the technique by which you make a microprocessor run at a higher speed than was originally configured. Although overclocking has its risks and could lead to instabilities within the system, with electronics heating up beyond their design specs, the Foundation does not automatically void your warranty if you decide to overclock according to its recommendations. Table 2 shows these overclock values for the various models. At print, overclocking had not been implemented for the RPi3, and if you try to overclock the Pi Zero in raspi-config, you get the message *This Pi cannot be overclocked*.

Overclocking might shorten the processor's life, so your Raspberry Pi won't last as long. My advice is to leave the *Overclock* option alone until you are sure you need the extra speed and understand all the risks it poses to your Pi.

Advanced Options includes *Overscan*, which allows you to establish a black border around the screen. This option was useful when mon-

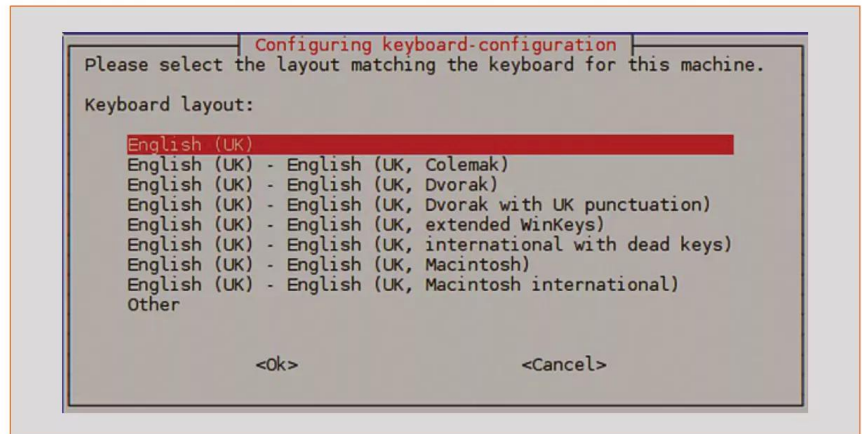


Figure 8: You can configure the layout and language of your keyboard under *Internationalisation Options* | *Change Keyboard Layout*.

itors and TVs had a physical plastic or wooden border that overlapped onto the viewing area. Overscan made the viewing area smaller, thus avoiding information being cut off by the physical border. Most modern monitors don't have this problem or have their own configuration menus to shrink or move the viewing area. The default is *Disable*, which you should probably leave as is.

The Raspberry Pi has two processors on-board. One is the CPU, which does all the general calculations and executes most of

NETWORKING

In the default configuration, Raspberry Pi will join the LAN by requesting an IP address from a DHCP server. (In most home environments, the local router/firewall device acts as a DHCP server, assigning IP addresses to the hosts on the LAN.)

This setup is fine for simple configurations, but if you want to put your Pi to work as a server system (as described in other articles in this issue), you might want to set it up with a permanent static IP address as follows. To begin, open the following file:

```
cd /etc/network
sudo nano interfaces
```

Replace the line `iface eth0 inet dhcp` with `iface eth0 inet static` (for Ethernet; if you have a working WiFi connection, use `wlan0` instead of `eth0`) and add the IP address, netmask, and gateway address you want to use for the Raspberry Pi. The address, netmask, and gateway will depend on the address configuration for your network. Talk to your network admin, or consult an online tutorial on TCP/IP addressing. The following example shows a sample entry for an address in the `192.168.77.0` address space

```
iface eth0 inet static
address 192.168.77.50      # Static IP you want
netmask 255.255.255.0
gateway 192.168.77.1      * IP of your router
```

If you configure a static IP address, you'll need to tell your Rasp Pi system where to find a DNS server. To set up name resolution, edit `resolv.conf` in `/etc` to point to the DNS server for the network:

```
sudo nano resolv.conf
nameserver 192.168.77.1
```

Several useful tutorials on Linux networking are available online. Raspbian is based on the Debian Linux distribution, so the Debian wiki is a good source for networking information [11].

BUG BOX

Sometimes, `raspi-config` does not change from `en-GB.UTF-8` to your keyboard language of choice (e.g., `en-US.UTF-8`) and throws error messages. If this happens to you, choose *Cancel* to get out of the keyboard configuration window, then Tab to *Finish*, and say *No* to a reboot. You need to edit the `.bashrc` file. To do so, enter:

```
sudo nano /home/pi/.bashrc
```

Arrow down to a blank line and enter:

```
export LC_ALL=C
```

Now press Ctrl+X; hit `y[es]` to save and Enter to save to the same file name. Next, enter

```
sudo raspi-config
```

to return to the configuration program. You should now be able to change your keyboard settings under *Internationalisation Options*.

the commands, and the other is the graphics processor, or GPU, which is used to render graphics in games, play videos, and so forth. In bigger machines, each processor has its own, separate memory, but not so with the Pi. The RAM has to be shared between the CPU and the GPU. The advanced *Memory Split* option therefore allows you to decide how much memory to allocate to the GPU, with valid selections suggested.

For everyday use, the defaults should be okay. But, if you are going to run GPU-intensive programs, such as 3D games or a Kodi media center application, you might want to give the graphics processor a bit more RAM.

The SSH (Secure SHell) protocol allows you to log in to a computer from a remote location over a secure and encrypted “SSH tunnel.” If you enable SSH on your Pi, you can use the `ssh` command to administer your Pi from another computer, even over the Internet (see the “Networking” box). You will be able to install software, start and stop processes, and even power off or reboot your Pi remotely. You can use your computer’s keyboard, screen, and mouse as peripherals for the Pi; moreover, you’ll be able to do anything you can do from the command line (which in Linux is *a lot*), and you’ll be able to transfer files to your Pi’s SD card from your computer or download whole directories from your Pi to your laptop. See the article on Remote Access later in this issue for more information.

If you have not changed your default password, or if your username/password combination is easy to guess, it doesn’t matter how secure the SSH channel is. Opening up the secure shell port for connections is a security risk. Make sure you change the default administrator’s password before you do so.

Other advanced options let you enable the *SPI* and *I2C* interfaces and download the appropriate kernel modules. The *Audio* option lets you choose whether to force audio through HDMI or the headphone jack. The autodetection feature should choose correctly for you. *GL Driver* should be enabled only on an RPi2/3. This option uses OpenGL drivers for GPU hardware acceleration. Note that if you enable the use of OpenGL on an RPi2/3 then move the SD card to an RPi1, the Pi will not boot. This option is off by default.

Another advanced option on the Raspberry configuration tool is *Update*. This option looks online for updates for the `raspi-config`

program and, if they exist, downloads and installs them. This is quite a good idea because new entries are being added to the `raspi-config` application all the time.

When you’re done configuring, use the Tab key to highlight the *Finish* button and hit Enter to exit the tool. Raspbian will ask whether you want to reboot. Answering Yes reboots immediately. Answering No applies your changes on the next bootup. The next time you boot into Raspbian, you will see the text login screen or GUI, depending on the choices you just made in the configuration program. Either way, if there is something you would like to change, you can run the Raspbian Configuration Tool from the desktop or start up `raspi-config` at any time from the command line or from a terminal window on the desktop.

Conclusion

Setting up Raspbian is not difficult. Thanks to the `raspi-config` tool, most things are easy to set up and do not require users to get their hands very dirty. This short introduction should be enough to get you started. If you decide to use another Raspberry Pi distribution instead of Raspbian, consult the project documentation on how to boot your Pi and get the system configured. ●●●

INFO

- [1] RS vendors by country: <http://www.raspberrypi.org/help/faqs/#buyingWhere>
- [2] Template for cardboard case (2 USB ports): http://squareitround.co.uk/Resources/Pun-net_net_Mk1.pdf
- [3] Template for cardboard case (4 USB ports): <http://sixes.net/rdcHQ/mosh/raspberry.pi.b.plus.pdf>
- [4] RPi Verified Peripherals: http://elinux.org/RPi_VerifiedPeripherals
- [5] Download Raspbian from the Raspberry Pi’s official site: <http://www.raspberrypi.org/downloads>
- [6] SHA1Sum tool for Windows: <http://www.softpedia.com/progDownload/SHA1Sum-Download-143137.html>
- [7] SD Formatter for Windows and Mac OS: https://www.sdcard.org/downloads/formatter_4/
- [8] Win32 Disk Imager: <http://sourceforge.net/projects/win32diskimager/>
- [9] The Raspberry Pi camera: <http://www.raspberrypi.org/product/camera-module/>
- [10] Rastrack site: <http://rastrack.co.uk/>
- [11] Network configuration in the Debian wiki: <http://wiki.debian.org/NetworkConfiguration>

An entire year of
**RASPBERRY PI PROJECTS
AND OPERATING TIPS**
for one low price!
Two bundles to choose from:



2014



2015

Hurry while supply lasts!

ORDER NOW!



<http://shop.linuxnewmedia.com/us/magazines/raspberry-pi-geek/catchup.html>

A brief introduction to the Linux operating system

Crash Course

Raspbian is a fully functional, high-powered Linux operating system, but if you come from a non-Linux background, it can be difficult to understand. In this article, we give you a quick introduction to the basics.

By Paul C. Brown

The Raspbian operating system, which is the default operating system of the Raspberry Pi and is distributed on the SD cards that come with most kits, is a Linux operating system based on a distribution called Debian.

The projects throughout this handbook assume you have some basic knowledge of the Linux filesystem and of how to use the shell, also known as the Linux command line. But if you are new to Linux, don't despair! The following pointers will help you get up to speed with the essentials.

Method to the Madness: The Filesystem

For newcomers to Linux, one of the most bewildering things is how the filesystem – the set of directories and subdirectories and the files they contain – is organized. Although you might have a hard time finding things in the beginning, the filesystem structure is highly logical, and you will find the same basic layout on all Linux distributions (distros).

The Linux filesystem is organized like a tree. The bottommost directory, from which all others sprout, is called the root directory and is designated with a slash (/). The root directory contains a series of subdirectories, such as /bin, /lib, /home, which in turn contain more subdirectories, and so on. Figure 1 shows part of the Linux directory tree.

You can see the contents of / by opening the desktop file manager (the file cabinet icon on the left in the top panel) and clicking twice on the arrow pointing up in the toolbar. If you are using the command line, you can type

```
$ ls /
```

The names of the directories are not arbitrary. All /bin and /sbin directories, for example, contain executable files, which are programs you can run (although, despite what the names of these directories imply, not all of the executable files are binary files). You will find bin and/sbin directories hanging off the root directory, and then again within the /usr directory; the root bin and/sbin directories contain the bare essential programs that the operating system needs to work, whereas the bin and/sbin directories in /usr contain extras. Thus, you will find an essential program like mount in /bin, but, if you install a game, you would probably find it in /usr/bin.

Some Linux distributions have joined both sets of directories so that you have only bin and/sbin directories hanging off /usr, but this is not the case with Raspbian, yet. If you're wondering about the differences between bin directories and/sbin directories, /sbin directories usually contain programs

THE AUTHOR

Paul Brown has been writing about technology professionally since 1996, when he got his first break writing a monthly column on Usenet trends, trolls, and topics for the Spanish tech underground magazine ARROBA. Since then, he has written extensively about silly Internet fads, bad web design, creative programming and fancy gadgets, as well as free software and free hardware. He is currently the Editor in Chief of *Ubuntu User* and *Linux Magazine Spain*.

that are reserved for the administrative user, or superuser.

If you want to know where a certain program lives, you can use the program `which` plus the name of the app from the command line, as follows:

```
$ which ls
/bin/ls
```

I will not delve too deeply into what each directory is for because that's beyond the scope of this article, but you must know that users' directories hang off `/home`. Thus, when you start using your Raspberry Pi, after you log in and before you start creating new users (see later), you will find yourself in `/home/pi`. In this directory, you can create, modify, and delete files and subdirectories.

If you want to attempt the same activities in other directories (e.g., in `/`, `/etc`, or `/usr`), you will need special privileges – usually superuser privileges. Please note that tampering with files outside your `/home` directory is dangerous and can lead to malfunctioning programs or even a trashed system. You can learn more about the Linux filesystem structure online [1].

Navigating the Shell

The shell has often been described as “unfriendly,” but a more correct term would be “misunderstood.” The Linux shell (a.k.a. the command line) is a very powerful tool, and on a machine like the Raspberry Pi, with comparatively little power and memory to match, the shell gives the user a complete set of apps with a wide range of functionalities that could never be stuffed into a graphical program and expected to work so well on the Pi's modest hardware.

If your system is configured to boot to a command prompt, you can start entering commands as soon as you log in. If your Pi boots to a graphic desktop, find and start a terminal emulator application. In Raspbian, you'll see an icon for LxTerminal in the top-left corner of the main window. You can also choose *Terminal* in the *Accessories* menu.

The filesystem is highly structured in Linux and, although you can navigate it using the desktop's file browser, doing so from the command line is also very convenient – often even more so. When you open a command line, most likely the terminal will open in your home directory. Type `ls` to list the contents of your directory.

You can use the `cd` (change directory) command to move to another directory. You'll

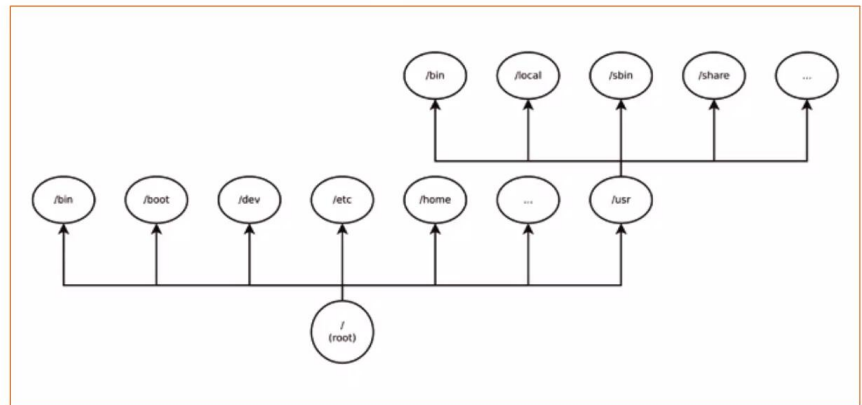


Figure 1: Part of the Linux directory tree.

also need to mention the path to the target directory:

```
$ cd /home/pi/Documents
```

The shell lets you use a dot (`.`) in the path to represent the current directory. In other words, you could move from your home directory to the `Music` subdirectory by typing:

```
$ cd ./Music
```

A double dot means “go back one level in the directory path,” so if you want to go from the `/home/pi/Music` directory back to your home directory (`/home/pi`), you could type:

```
$ cd ..
```

Many systems also use the tilde character (`~`) to represent the home directory, so no matter where you are, you can always return to your home directory with:

```
$ cd ~
```

If you get lost when you are navigating around in the directory structure, you can always enter the `pwd` (print working directory) command to display the name of the current directory.

To create a new directory, enter the `mkdir` command with the name you want to give to the directory:

```
$ mkdir /home/pi/Music/Beatles
```

Or, if you were already in the `Music` directory, you could just type:

```
$ mkdir ./Beatles
```

Pro tip: You can make subdirectories recursively with the `-p` option. For example,

```
$ mkdir -p Music/Beatles/Help
```

will create the `Music`, `Beatles`, and `Help` subdirectories all in one go.

The `cp` command lets you copy files (the angle brackets, `<>`, indicate a parameter that you supply):

```
cp <source_filename> <destination_filename>
```

The default is to look in the current directory; however, you can include a path with the source or destination to copy to or from a different directory. Of course, you must have the necessary permissions to access the directory. The `mv` instruction moves files or whole directories from one place to another. If the instruction is used on files or directories that are not moving, it renames them. For example,

```
mv file1 dir/
```

will move `file1` into directory `dir/` hanging off the current directory. But

```
mv file1 file2
```

will change `file1`'s name, renaming it `file2`.

To delete a file, use the `rm` (remove) command, and to delete a directory, use `rm -r` or `rmdir`. Needless to say, you must be careful how you use these commands. A summary

of these basic commands is shown in Table 1. Each of these commands includes additional options that you can enter at the command line, which you can see by typing `man <command>`.

Users

Linux and other Unix-based systems use the concept of a user account, which allows the system to manage identities and restrict access to a collection of resources associated with a specific person or the groups to which that person belong. One aspect of a user account is the familiar login prompt

that often greets users who want to access a computer. The user's identity is also a means for assigning access permissions to files, directories, and other resources.

Linux also lets the system administrator manage access to resources through group membership. A group is a collection of users, typically with a common purpose and, therefore, a common need for access to a collective set of resources. For example, a group called *Accounting* might contain users who are part of the accounting team with the need for a common level of access to spreadsheets and other financial data. Rather than manually giving each user permissions for every file, the administrator can assign access permissions to the *Accounting* group and then place users in the group to give them access to the files.

To see which groups your user belongs to, you can use the command `groups` from the command line. This is what you'll see if you use `groups` while logged in as the default Raspbian user, `pi`:

```
$ groups
pi adm dialout cdrom sudo audio
video plugdev games users netdev
input indiecity
```

You could have more than one user who accesses your Raspberry Pi, and to keep things clean and safe, it is a good idea to create an account for each person. The most common command-line utilities used to create new users and groups are `useradd` and `groupadd`. These utilities live in `/usr/sbin` by default, and to use them, you need to be the superuser. On Raspbian, you'll also find `adduser`, which is a script that is meant to be a friendlier front end to `useradd`, `groupadd`, and `usermod`.

Just run `adduser` from the command line using the name of the user you want to create as a parameter

```
sudo adduser <username>
```

The program will guide you through the process (Figure 2), asking for a password and personal details, which are optional. At the end of the process, the new user will have a personal directory within the `/home` directory.

To check that everything works, try logging in from the command line by typing:

```
$ su <username>
```

The shell will then ask you for the user's password. If everything went well, you should be

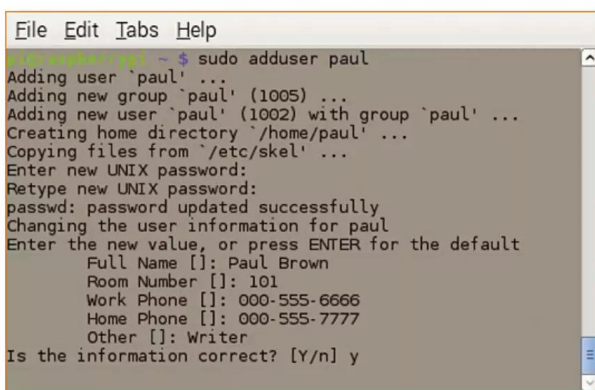


Figure 2: Using the `adduser` utility.

TABLE 1: Some Basic Shell Commands

Command	Action
<code>ls</code>	List contents of the current directory
<code>cd</code>	Change directory
<code>pwd</code>	Show current working directory
<code>mkdir</code>	Make directory
<code>cp</code>	Copy file(s)
<code>mv</code>	Move or rename a file or directory
<code>rm</code>	Remove file(s)
<code>rmdir</code>	Remove directory


```

File Edit Tabs Help
pi@raspberrypi ~ $ ls -l
total 20
drwxr-xr-x 2 pi pi 4096 Mar  2 18:26 Desktop
drwxr-xr-x 2 pi pi 4096 Mar 31 20:58 Documents
drwxr-xr-x 2 pi pi 4096 Mar 31 20:59 Music
drwxr-xr-x 2 pi pi 4096 Mar 31 20:59 Pictures
drwxr-xr-x 2 pi pi 4096 Jan  1 1970 python_games
pi@raspberrypi ~ $

```

Figure 3: The `ls -l` command shows files' permissions.

able access the new user's account. To return to your default user, type `exit`. To add a new user to a group, you can use the `usermod` command:

```
sudo usermod -a -G sudo newuser
```

This will make *newuser* a member of the `sudo` group with the ability to run commands with superuser privileges using the `sudo` command modifier (see the "Superuser with `sudo`" box).

Privileges

For every file (and directory, device file, etc.), Linux defines which users may read, write, and execute that file. Also, every file belongs to an *owner* (an individual user) and to a *group*. Other users might also be able to use files and directories, to some extent, that belong to third parties or be barred completely. To see the owners, group owners, and permissions for each file and directory in your current directory, type

```
$ ls -l
```

into your terminal window (Figure 3).

The first column on the left shows the permissions, which I will explain shortly; the second column indicates the number of hard links (additional names for a file) to the file or directory; the third column shows how much space (in bytes) the entry takes up on the SD card; and the fourth column lists the owner, followed by the group. Finally, you have the date and time at which the file or directory was created and its name.

If you look at the first column, the following three permissions are assigned separately for owners, groups, and other users:

- Read permission (r flag): Users can display the contents of a file or folder on screen, copy the file, and do a few other things. Directories should additionally have the x flag (see later) to allow users to change to that folder; otherwise, only a list of files can be displayed.

- Write permission (w flag): Users can change files and directories and store their changes. Write permission also includes the ability to delete the file.
- Execute permission (x flag): For programs, this means the user is permitted to run the program. Execute permission for a directory means the user is permitted to change to the directory (the user additionally needs read permission to be able to view the folder content).

As you can see, permissions are indicated by the letters r (for read), w (for write), and x (for execute). In the `ls` output, note the three sets of r, w, and x (or -) at the beginning of the file name entry. The first block shows the permissions for the *owner*, the second block refers to the *group*, and the third block refers to *all other users*. Folders are indicated by a d (for "directory") and regular files by a single dash (-) at the start of the list. A number of other types of entries exist, such as symlinks, block devices, and character devices, but I will not go into them here.

The Raspbian graphic file manager also can show a file's permissions if you right-click it and choose *Properties* from the pop-up menu (Figure 4).

The `chmod` program lets you modify file and directory permissions, assuming you are the owner or the system administrator, with the use of either letters or numbers. Here, I'll concentrate on using letters, because they're easier to remember.

Using letter notation, u stands for user (owner), g for group, and o for others (all other users). I described previously the meanings for r, w, and x. A combination of these letters (without spaces!) with plus, minus, and equals signs tells `chmod` to add, remove, or assign these permissions (Table 1). To give a group read and write permissions for a file, type `chmod g+rw <file>`.

Removing permissions follows the same pattern: The `chmod o-rwx <file>` command removes all permissions for all users who are neither the owner nor members in the owner group. You could combine these two commands:

SUPERUSER WITH SUDO

Traditionally, many Linux distros had a completely separate account for the superuser (also known as root), and the system administrator only accessed the superuser to accomplish system administrator stuff. To do this, the sys admin could do one of three things:

1. Log in as root.
2. Use the `su` command to switch user to superuser.
3. Include his/her account in the `/etc/sudo` file or join the `sudo` group and use the `sudo` command to run programs as root.

This last approach is the one used in Raspbian. To all practical effects, pi is a superuser that can access superuser privileges just by using the `sudo` command.

To keep things safe, change the pi's default password (as explained in the previous article) and create new users with limited privileges that cannot trash your system. If you do need to create a user that has superuser privileges, include him in the `sudo` group as explained elsewhere in this article.

Each time the user needs to do something as root, he will have to write `sudo <command>` and enter his password.

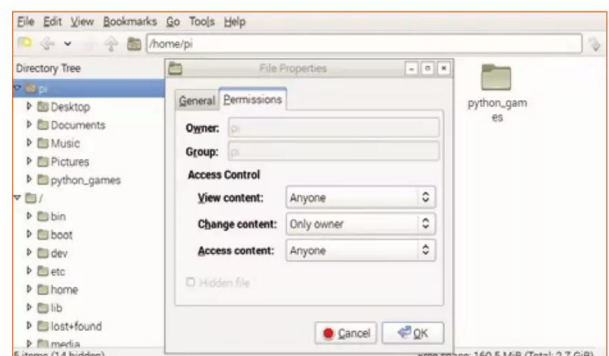


Figure 4: You can see the permissions of a file by right-clicking on it and choosing *Properties* from the file browser.

GET STARTED

Understanding Linux

RIGHTS OF OWNERSHIP

Although I said before that `chown` is only available for the root user, this is not entirely true – a “normal” user may use the tool in some situations. For example, `chown pi:audio <file>` changes the group membership of the file. The user `pi` is allowed to use this command if they are a member of the group `audio` and own the named file.

```
chmod g+rw,o-rwx <file>
```

As I mentioned before, an equals sign lets you assign precisely the permissions specified at the command line. For example, the command

```
chmod ugo=rwx <directory>
```

gives the owner, group members, and all other users read, write, and execute permissions for the directory in question. Instead of `ugo`, you could alternatively use a (for “all”) to assign user, group, and other permissions.

To change group membership for files and directories, you can use the `chgrp` tool. Keep in mind that Linux takes extra precautions with this command: As a “normal” user, you are allowed to assign your own files to specific groups. As long as you are a member of the group in question. The root user, as always, has no restrictions.

As mentioned before, to see the groups to which your current user belongs, you can type groups into a terminal window:

```
$ groups
pi adm dialout cdrom sudo audio 2
video plugdev games users 2
netdev input indiecity
```

In this case, the user called `pi` (the default Raspbian user) may change access to their own files for members of the groups `pi`, `adm`, `dialout`, `cdrom`, `sudo`, `audio`, and so on. The `chgrp` command first expects information about the new group and then the name of the file or directory. To assign a file to the `audio` group, just type:

```
chgrp audio <file>
```

On a Linux system, the system administrator is allowed to assign new owners and new groups to files and directories. To give a file to user `pi`, simply use the `chown` command:

```
chown pi <file>
```

Also, you can define a new group in the same command. To do so, add the name of the group after a colon:

```
chown pi:audio <file>
```

The file now belongs to user `pi` and group `audio` (see the “Rights of Ownership” box).

All three tools – `chmod`, `chgrp`, and `chown` – support the `-R` option for recursive actions. If you want members of the `video` group to access a directory and the files it contains, just type:

```
chgrp -R video <directory>
```

The `-R` option can also save you some typing in combination with the `chmod` command. To remove read, write, and execute permissions from this folder for all users who are not the owner or members of the `video` group, type:

```
chmod -R o-rwx <directory>
```

Be careful when you run recursive commands that remove the execute flag. If you mistakenly type `a-x` instead of `o-x`, you will lock yourself out: `chmod` will remove execute permissions from the parent directory and your ability to make changes to the directory and modify the files (Listing 1).

Using the `find` command can help you avoid this kind of dilemma (Listing 2). The `find` command first discovers files (`-type f`) in the test directory (and possible subfolders) and then runs `chmod` against them, ignoring the directory itself.

Apt Package Manager

Another thing you’ll be using a lot from the command line is the package manager. Although software for the Pi is covered in more depth in other articles, it is worth going over the basics of the Raspbian’s Apt (Advanced Package Tool) system.

First things first: *Do not* go hunting for software on the Internet without checking the Raspbian repositories first. Linux users download and install the majority of their software from approved and reliable repositories and very, *very* rarely stray onto the web to download apps. Furthermore, Raspbian comes with a set of official online repositories already configured, so, if you need to install new programs, you can get started right away.

The basic command for adding a software package is simply

```
sudo apt-get install <package-name>
```

When you enter the command, you usually get a complete summary of what will happen

LISTING 1: Oops ... Locked Out!

```
$ ls -l test
total 0
-rwxr-xr-x 1 pi pi 0 Nov 4 12:12 bar
-rwxr-xr-x 1 pi pi 0 Nov 4 12:12 foo
$ chmod -R a-x test
chmod: cannot access 'test/bar': Permission denied
chmod: cannot access 'test/foo': Permission denied
```

LISTING 2: Using the `find` Command

```
$ find test -type f -exec chmod a-x {} +
$ ls -l test
total 0
-rw-r--r-- 1 pi pi 0 Nov 4 12:12 bar
-rw-r--r-- 1 pi pi 0 Nov 4 12:12 foo
```


if you go through with the installation, including the dependencies that will be installed, the packages that will be upgraded and removed, and the amount of disk space that will be required (Figure 5). Unless the action can proceed automatically without affecting anything else, you then have the choice of continuing the process or not. Just to be sure what you typed doesn't include any unpleasant surprises, you should read the summary carefully before continuing.

As `apt-get` works, it shows which package it is downloading and its progress, as well as the download speed and the amount of time required to finish the operation. The times are only estimates and will change as the Internet connection speed changes. Once the downloads are complete, `apt-get` installs the software, sometimes pausing to ask questions about how you want it installed.

After everything is done, `apt-get` exits with a summary of any problems that it encountered, if necessary. As a final touch, if the software you just installed is a graphical application, it is added to your desktop menus.

You can remove packages by using

```
sudo apt-get remove <package-name>
```

but this process could leave configuration files behind. If you want to get rid of all traces of a package, use

```
sudo apt-get purge <package-name>
```

The instruction `sudo apt-get autoremove` will remove all old and unused packages.

If you want to check for updates for all the software installed on your Pi, you can use

```
$ apt-get update
```

and Apt will track down all the newer versions in the on-line repositories and install them for you.

The Apt system includes several other tools, but by far, the most useful package utility is `apt-cache`, which offers a treasury of information about packages and your system. For example,

```
apt-cache showpkg <packagename>
```

shows which version is installed, the latest version available in the repositories you are using, and the reverse dependencies of the packages (i.e., which packages depend on it).

Similarly, `apt-cache dump` lists all the packages you have installed, and `apt-cache stats` offers information such as the number of installed packages and the total number of dependencies. An especially useful option is `apt-cache search <string>`, which tracks down the exact name of a package or packages that you might want to install by searching for the `<string>` you provide as an argument. If you wanted to install, say, a Minesweeper-type game, you could type

```
$ apt-cache search minesweep
```

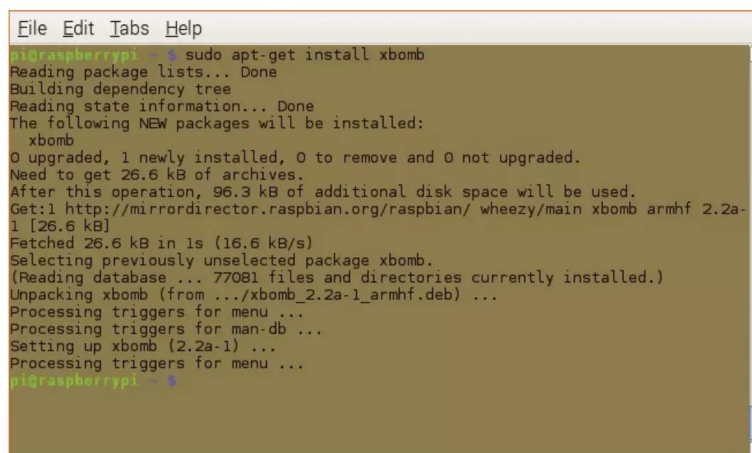
and Apt will return all programs that contain the word “minesweep” in its name or description.

Further Reading

I've provided a very fast and superficial introduction to the very basic tools used by Linux users. The number of programs, commands, and options included by default with any Linux distribution, including Raspbian, is vast. Also, don't forget `man`, which brings up a manual for a given command, and check out our sister magazines, *Ubuntu User* [2] and *Linux Pro Magazine* [3], which run regular columns on command-line instructions, applications, and options. ●●●

ACKNOWLEDGMENTS

The author expresses his gratitude to Bruce Byfield, Nathan Willis, Joe “Zonker” Brockmeier, Heike Jurzik, and Hans-Georg Esser for their help and input for this article.



```
pi@raspberrypi ~$ sudo apt-get install xbomb
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following NEW packages will be installed:
  xbomb
0 upgraded, 1 newly installed, 0 to remove and 0 not upgraded.
Need to get 26.6 kB of archives.
After this operation, 96.3 kB of additional disk space will be used.
Get:1 http://mirrordirector.raspbian.org/raspbian/ wheezy/main xbomb armhf 2.2a-1 [26.6 kB]
Fetched 26.6 kB in 1s (16.6 kB/s)
Selecting previously unselected package xbomb.
(Reading database ... 77081 files and directories currently installed.)
Unpacking xbomb (from .../xbomb_2.2a-1_armhf.deb) ...
Processing triggers for menu ...
Processing triggers for man-db ...
Setting up xbomb (2.2a-1) ...
Processing triggers for menu ...
pi@raspberrypi ~$
```

Figure 5: The `apt-get` tool allows you to install software from the Raspbian repositories.

INFO

- [1] The Linux/Unix filesystem tree: http://en.wikipedia.org/wiki/Filesystem_Hierarchy_Standard
- [2] *Ubuntu User* magazine: <http://www.ubuntu-user.com/>
- [3] *Linux Pro Magazine*: <http://www.linux-magazine.com/>

Exploring the Raspbian desktop

Desktop Delight

The Lightweight X11 Desktop Environment, or LXDE, brings all the advantages of a graphical interface to the Raspberry Pi.

By Paul C. Brown

The Linux community has a saying that Linux has *at least* two of each thing. For instance, you can get the OpenOffice, LibreOffice, or Calligra office suite, and Linux also provides multiple tools for accounting, drawing, and photo processing. The same holds true for desktop environments: For Mac OS X and Windows you have, well, OS X and Windows. But on Linux, you can choose between Gnome, KDE, Cinnamon, Unity, Mate, Enlightenment, Xfce, and many more.

This embarrassment of choices might be confusing for a non-Linux user, but each of these projects was started to satisfy a specific need.

For example, the creators of Gnome are very conscious of usability and want to make everything uber-simple for the end user. KDE is jam-packed with tweakable features, and Enlightenment seeks to bring spectacular 3D desktop effects to even the most underpowered machine.

The environment chosen for Raspbian, the official distribution for the Raspberry Pi, is LXDE, and it is a very good fit. LXDE [1] is extremely lightweight, and although it might not be as pretty

as Gnome or have as many settings and modes as KDE, it still manages to include most of the essential features you would expect to find in a modern-day desktop.

Getting Started with LXDE

As you learned in a previous article, you can configure Raspbian to boot to the command line or directly to the graphical desktop. Even if you boot to the command line, you can reach the desktop by entering `startx` and then hitting Return.

The default layout is similar to the Gnome/Mac OS X format (Figure 1), with a bar across the top (we'll get to that in a minute) and a default light gray desktop (Figure 1.1) with the Raspberry Pi logo in the center. Right-clicking on an empty space on the desktop pops up a menu with options that allow you to modify some of its aspects. By choosing *Desktop Preferences*, you can change the wallpaper, default fonts, and so on.

A brand-new Raspbian installation comes with a single trashcan icon in the upper left hand corner (Figure 1.2). Although it is quite easy to put another icon up, it is not as simple as dragging an icon to the desktop. If you want to add new icons to your desktop, see the "Personalizing" section later in this article.

Along the top is a *panel* (Figure 1.3) with a menu launcher in the top left corner (Figure 1.4). Starting from the left, you will see an application bar (Figure 1.5) that contains icons to launch the a web browser, a file browser, a terminal, the Wolfram Mathematica app, and a link to the Wolfram scripting language.



The middle section of the panel is reserved for the *taskbar* (Figure 1.6), which shows the windows you have open. On the far right is an app that shows the CPU usage (Figure 1.7) and an icon that opens the clock.

By right-clicking on the panel and choosing *Panel Settings* from the pop-up menu, you can change what the panel looks like, add more items, move them around, and delete the ones you don't want. In the *Geometry* tab, you get to choose the size and position of the panel and its icons; in *Appearance*, you can select the color and theme; in *Panel Applets*, you can add more items and sort them using the *Up* and *Down* buttons (Figure 2). (In the default panel, *Up* means "move one position to the left" and down means "move one position to the right.")

Many of the items within the panel are also configurable. For example, right-clicking *Application Launch Bar* brings up a menu that allows you to include more apps in the bar, which is very useful for creating shortcuts to apps you use often.

Double Desktops

LXDE, like many Linux desktop environments, gives you the ability to work with several desktop spaces at the same time. Although you can have up to 16, more than four is probably overkill, and it could overload your Pi and make it run slowly.

To add a new desktop, use the mouse wheel to click in an empty space on the desktop and choose *Add new desktop*. Also, you can spin the mouse wheel to navigate between desktops, or you can add the desktop pager to your panel and then just click on the desktop you want to visit in the pager. Right-clicking on a title bar and choosing *Send to desktop* moves a window to a different desktop (Figure 3), and dragging windows to the side of the current desktop lets you move windows from desktop to desktop.

Personalizing

Adding new apps to the desktop or menu for easy access is complicated with LXDE, but the process has been simplified in recent versions. Open your file browser (the second icon from the left in the default application bar – it looks like a filing cabinet) and navigate to the `/usr/share/applications` folder. Within that you will see all the icons of your installed apps. Right-click on the one you want to put on your desktop and select *Copy* from them pop-up menu, then right-click on an empty space on your desktop and select *Paste*. That's it, or at least that's the easy way.

In fact what you're doing is duplicating files, which is kind of messy. If you want to be a real pro, open the terminal (third icon from the left in the applications panel – it looks like a black computer monitor), `cd` into the `Desktop/` directory with:

```
$ cd Desktop
```

and create a soft link to the application you want to put on the desktop with

```
$ ln -s /usr/share/applications/<application name>
```

If you don't remember the exact name of the application, type the command through `/applications/`:

```
$ ln -s /usr/share/applications/
```

and then hit the `Tab` key twice. The Linux shell will show you all the valid options. If you see *More* at the bottom of the window, hit the spacebar to see the rest of the list. You can start typing the name of the application (say you want to put *galculator* onto the desktop, type `galc`) and hit `Tab` again. The Linux shell will autocomplete the name for you. Your line should end up looking like this:

```
$ ln -s /usr/share/applications/galculator.desktop
```

To end, hit `Return`.

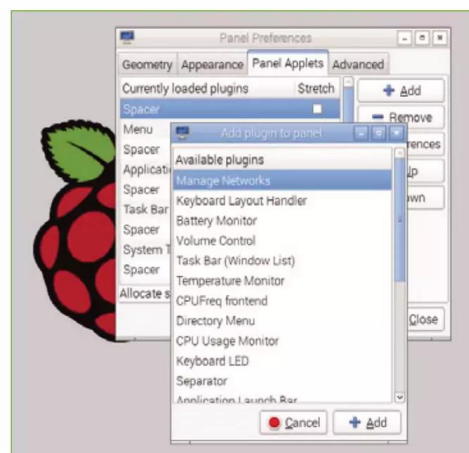


Figure 2: Adding new items to the panel is easy.

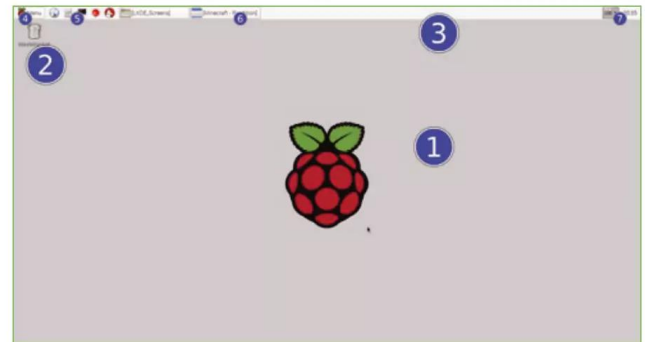


Figure 1: The parts of the default Raspbian desktop.

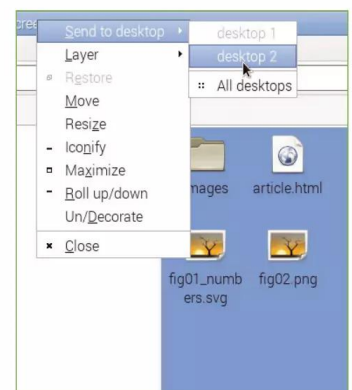


Figure 3: You can send windows to different desktops by right-clicking on their title bars.



Figure 4: You can add icons to the desktop using Copy and Paste. Notice the little arrow at the top left of the Monitor Settings icon, indicating it is a link.

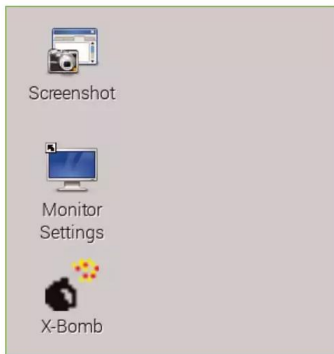


Figure 5: .desktop files allow you to create shortcuts to your apps on the desktop.



Figure 6: LXDE will put your new apps into the correct category if you have tagged them correctly.

INFO

- [1] The Lightweight X11 Desktop Environment: <http://lxde.org/>
- [2] Categories allowed in a .desktop file: <http://standards.freedesktop.org/menu-spec/menu-spec-1.0.html#category-registry>

The icons you add like this will show up with a little arrow in the upper left hand corner, indicating they are a link to the file that runs the app, not the file itself (Figure 4).

If you right-click on one of your new desktop icons and choose *LibreOffice Writer* or select *Openwith...* and choose *Accessories | Text Editor* (or *Leafpad*, the default text editor for LXDE), you'll see that the icon is associated with a text file that looks something like Listing 1. This little file, which has a .desktop extension, contains the information necessary to show and run an app from the desktop:

- The **Name** field contains the label shown below the icon.
- **Comment** contains a brief description of the app, which is useful when you want to include a new app in the menu. Mousing over the app shows this text as a hint bubble.
- **Exec** contains the path to the program. See the *Understanding Linux* article elsewhere in the issue for more on executable files.
- **Icon** points to the location of the icon for the app.
- **Terminal** tells the desktop whether this app will run from the command line. If set to **True**, LXDE opens a terminal window and executes the app.
- **Type** is generally **Application**, but it can also be **Link** (pointing to a web page) or **Directory** (pointing to a directory on the SD card).
- **Categories** contains the name of the sub-menu under which the application is classified when it is placed in the menu. Although you can't put just anything in this field, the accepted list of categories [2] is quite comprehensive.

Many more lines are valid in a .desktop file (e.g., look at the *exo-web-browser.desktop* file in */usr/share/applications*), but those listed above are the essential fields.

Most preinstalled apps can be found in the */usr/share/applications* folder, but if what you're looking for isn't there, you will have to build your own .desktop file. As an example,

LISTING 1: Typical .desktop File

```
[Desktop Entry]
Name=App Name
Comment=Brief description
Exec=</path/to/program>
Icon=</path/to/icon>
Terminal=false
Type=Application
Categories=Application;Development;
```

download and install the X-Bomb game (a basic minesweep clone) and create a shortcut on the desktop. In a terminal window, type:

```
$ sudo apt-get install xbomb
```

Once the game is installed, make sure it's not in the */usr/share/applications* folder (it won't be). You can use your file browser to navigate to */home/pi/Desktop*, right-click on an empty space within the folder, and choose *Create New | Empty File* from the pop-up menu.

Name the file *xbomb.desktop* and a text icon will appear in your folder and on your desktop. Right-click on the file and choose *Text Editor* to open the file for editing. Next, just copy the lines shown in Listing 2 into your new .desktop file.

If you're wondering where this information came from, check out the */usr/share/menu/xbomb* file. The data is not in the same format, and it won't work if you just copy and paste it into the Desktop folder, but all the pieces you need are there.

When you save the file, the text page icon will turn into a cartoon bomb (Figure 5). And, if you double-click the bomb icon, X-Bomb will launch.

Still one more thing to do: If you look in the *Games* entry in your start menu, you will see that X-Bomb is not there. Copy and paste the .desktop file you just created into the */local/share/applications* directory in your home folder, and X-Bomb will show up on your menu in the *Games* category. If the category didn't exist before, LXDE will create it (Figure 6).

Conclusion

LXDE is a very capable desktop environment that is lightweight and highly configurable. At first glance, LXDE might not seem as straightforward as other more popular Linux desktops (i.e., KDE and Gnome), but with a little practice, it is every bit as user-friendly and as feature-rich as other comparable desktops and a perfect fit for the Raspberry Pi. ...

LISTING 2: xbomb.desktop

```
[Desktop Entry]
Name=X-Bomb
Comment=Find all the mines!
Exec=/usr/games/xbomb
Icon=/usr/share/pixmaps/xbomb.xpm
Terminal=false
Type=Application
Categories=Game;
```


NOOBS multi-installer for the Raspberry Pi

Piece of Pie

The NOOBS boot manager helps beginners try out Raspberry Pi operating systems and lets advanced users dig into the structure of the systems to adapt them as they like.

By Ferdinand Thommes

The Raspberry Pi was created to acquaint young people with the subject of programming by exposing them to various aspects of its hardware.

Many of the millions of units sold are in the hands of users who have little previous experience with hardware or with Linux. As a result, many users have felt overwhelmed by first contact with the Rasp Pi because they don't exactly know how to equip it with software.

This is where NOOBS [1] comes into play. The “New Out Of Box Software” is designed to help novices over the first hurdles of installing one or more operating systems. NOOBS lets you easily install and administer several operating systems on an SD card from a well-arranged user interface.

NOOBS allows each system to administer its own kernel, which means you can install two operating systems simultaneously. The basis of NOOBS is a minimal Linux system with the BusyBox [2] reduced versions of Unix utilities, along with an adapted version of the Enlightenment desktop and the Qt5 graphical framework.

DOWNLOAD OPTIONS

When downloading, you can choose between two versions of NOOBS. The *NOOBS offline and network install* includes Raspbian, plus a selection of operating systems that are downloaded from the Internet before being installed. The *NOOBS Lite network install only* version has only the operating system installer, and all systems are downloaded before installation.

To install anything other than Raspbian with NOOBS, you need access to the Internet via Ethernet or WiFi. Both NOOBS versions are available for direct download or as Torrent files on the Downloads page of the Raspberry Pi Foundation website [3]. Additionally, commercial distributors (and the Foundation [4]) offer SD cards with NOOBS preinstalled, which may

be purchased alone or in combination with a Rasp Pi.

THE RIGHT SD CARD

The project recommends an SD card with at least 4GB of storage when working with NOOBS, although 8GB or more is better if you want enough space for more than one operating system, as well as data.

Even if you have a brand new SD card, it should always be formatted before you use it. You will need your password to format the card to the FAT filesystem. On Windows [5] or Mac OS X [6] systems, the Raspberry Pi Foundation recommends the use of software developed by the SD Association. Instructions on the websites guide you through the formatting process. Make sure under Windows to set the *FORMAT SIZE ADJUSTMENT* option to *ON* and under Mac OS to choose *Overwrite Format*.

On Linux, first find the name of your SD card with:

```
df -H
```

Look for the name on the far left side of the output (the *Size* column will help you choose the correct device). It will probably be something like `/dev/mmcblk0p1`. Ignore the two other characters hanging off the end and format the card by entering

```
mkdosfs -F 32 -v <devname>
```

where `<devname>` is the name of the device you found with `df -H`. In the example here, that would be `mmcblk0`.



THE AUTHOR

Ferdinand Thommes lives in Berlin and works there as a Linux developer, independent writer, and city guide.

MISSING OPERATING SYSTEMS

NOOBS used to offer a wider variety of OSs, including Pi-dora and Arch Linux. However, these distros stopped providing ready-made images with the change from ARMv6 to ARMv7/8 (Pi2/3). You can find instructions for creating an Arch image for ARMv7 online [7]. A new community is working on a Fedora-based distro, FedBerry [8], for the new chip architectures, as well.

See the “First Steps” article for detailed information on preparing the SD card on Linux, Mac OS X, or Windows.

INSTALLATION

After you unzip the NOOBS archive, downloaded from either the Raspberry Pi website or the DVD from this issue, switch to the newly created directory and move all the files in the NOOBS_v1_9_0 folder (in this case) to the SD card. Next, you can eject the card from the PC and insert it into the appropriate slot of the Rasp Pi.

To boot the Pi, connect the power supply with the board and the HDMI port with the display, such as a monitor or a TV. If your display does not have an HDMI input, you need to use a DVI-to-HDMI adapter. After a brief interval, the graphical user interface of NOOBS appears and presents a list of the various distributions, that are available (Figure 1).

If you see a rainbow square at the top right of the display or the red LED on the Pi flickers, this means you are not getting enough power (most often seen on a Pi3). Another symptom of low power is continuous rebooting. If you see any of these problems, you will need a new power supply unit (PSU; 5V, 2.5A) and good-quality lead from the PSU to the Pi. Power fluctuations can corrupt the SD card, so make sure it is still good.

If the display remains blank, the system probably has not recognized the video mode correctly. In this case, hold down the 1 key on the keyboard for HDMI, 2 for HDMI safe mode, 3 for Composite PAL mode, or 4 for Composite NTSC mode. Safe mode should always work. NOOBS writes the

chosen option in the config.txt file of the distribution you are running so that the display device is automatically recognized in the future.

In most cases, however, the system chooses the correct mode automatically, and the selection screen shows the operating systems to be installed. All the operating systems shown in Figure 1 are for the ARMv7/8 architecture. Raspbian [9] is the most popular Linux distribution for the Pi. OpenELEC and OSMC are the Rasp Pi implementations of the Kodi Media Center software. See the “Missing Operating Systems” box for information about other Linux OSs.

An alternative is RISC OS [10], which deviates from the other systems in that it is not based on Linux. Instead, it is a native ARM-based operating system originally developed for the Archimedes computer at the end of the 1980s. Today, RISC OS continues to be developed as free source code.

A recent addition to the NOOBS lineup is Windows 10 IoT Core, a Microsoft Windows system designed to run on small embedded systems and single-board computers. IoT Core lets you run Universal Windows Platform (UWP) apps on your Raspberry Pi.

When selecting the desired distributions in the dialog box, the needed and available storage space figures are shown at the bottom of the window. The number of systems you may install is limited only by the size of your SD card. To navigate the list, you need to use the arrow keys on the keyboard. At this point, you should select the language and keyboard pref-

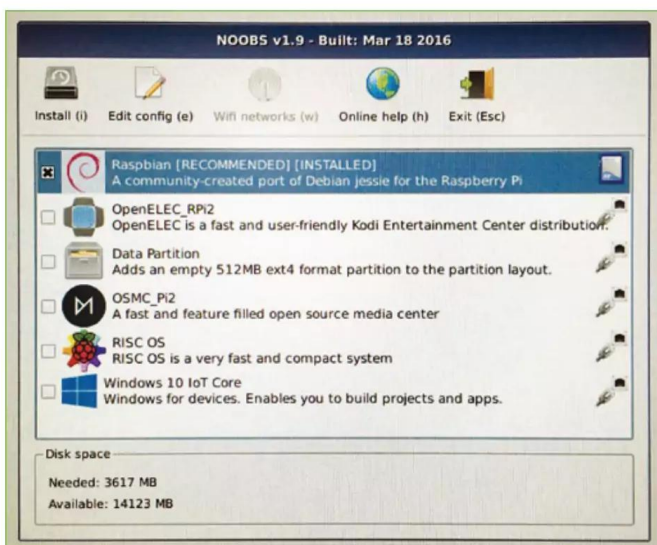


Figure 1: After bootup, you should select the systems you want installed from the list of operating systems.

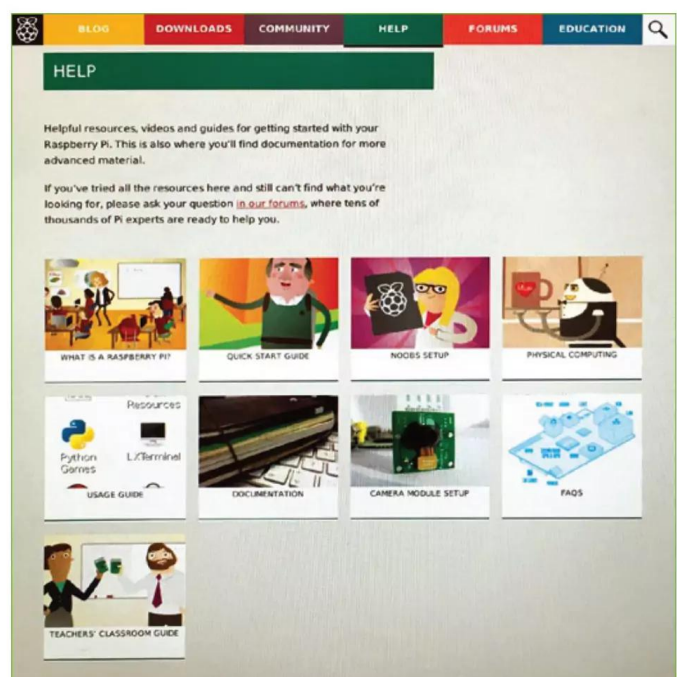


Figure 2: The online help page opens from the system selection dialog.

erence at the bottom edge of the window. Finally, click on the *Install* icon at the top left to start the installation process.

If you have an Internet connection, the *Online help* icon at the top of the window lets you call up help on the Internet (Figure 2). Clicking the *Exit* icon at the far right or the Esc key closes the input dialog. The option *Edit config* icon lets you edit the configuration files of each of the installed systems. The icons to the right of the selected entries show whether NOOBS installs the indicated system from the SD card or the Internet.

FRESHLY PARTITIONED

As a first step, the installer repartitions the SD card by changing the single partition, which was created during formatting, into three partitions. At the beginning of the storage area on the card, the installer creates a boot partition. NOOBS itself boots from this partition and then presents a list of the installed operating systems. The boot partition is followed by the second partition, which contains all the installed operating system images. The third partition is an initially small recovery partition.

If more than one operating system has been installed, NOOBS displays a menu after installation has completed, from which you select the system to be launched (Figure 3). Your selection is remembered, and the most recently booted system will be started again unless you alter the selection within 10 seconds.

The selection menu does not appear if you install only one distribution; that system will launch automatically. If you press the Shift key when you see the screen instructing you to do so during bootup, you will again see the dialog shown in Figure 1 that lists the operating systems available for installation. In this way, you can install additional distributions as needed.

Note that the new selection(s) will delete all previously installed systems. Working in recovery mode allows you to avoid accidental deletion and save one or more operating systems that have already been installed into the persistent recovery partition mentioned earlier. Note that this will make the recovery partition grow.

ADVANCED OPTIONS

Aside from the user-friendly installation of multiple operating systems, NOOBS lets you adapt configuration files according to your own needs. For example, you can extend or completely replace the list of operating systems to be installed. Additionally, you can install different and individualized versions of the same operating system. In a school setting, for example, this approach would allow

a teacher to install a system for producing multimedia content for one half of a classroom and a lean Python programming environment for the other half, where both system versions could be incarnations of Raspbian.

Adding additional operating system images of your own creation is accomplished by editing some configuration files. Just as easily, you can configure NOOBS in such a way that an image stored on the SD card is installed automatically when the Rasp Pi boots up.

To get NOOBS to perform this automatic installation, you should first copy the image to be installed to the `os` folder on the SD card. For a silent installation, simply edit the file `recovery.cmdline` by adding the option `silentinstall` to the list of arguments. If several versions of the same operating system exist on the card, you will need to indicate the desired flavor in the `flavors.json` file.

Additional switches in the `recovery.cmdline` file allow you to specify that the language and keyboard layout be in English via `lang=en` or `keyboard=us`. The display mode described above can also be changed at this point with the `display=1` switch setting, which would correspond to HDMI mode.

If you want NOOBS to launch a system automatically after 10 seconds without user input, set the `partition=<number>` switch. The `forcetrigger` argument makes sure the recovery mode starts each time the Rasp Pi is booted up.

All of this can be accomplished via the GPIO connector [11] by unlocking it with the `gpio-triggerenable` argument. For example, this would allow you to always connect pins number 3 and 25 during the start-up phase to activate recovery mode. Achieving the same outcome by modifying one of the preinstalled images is more complicated. For more information on how to configure operating system images, go to the lower portion of the `README.md` file, which you can find on the NOOBS GitHub page [12].

CONCLUSION

Although NOOBS is primarily intended for Raspberry Pi beginners, experienced users can also profit from the uncomplicated installation of several parallel systems. Thus, it is very easy to try out different operating systems without having to dig into the depths of the system.

INFO

- [1] NOOBS: <http://www.raspberrypi.org/archives/4863>
- [2] BusyBox: <http://www.busybox.net/about.html>
- [3] NOOBS download: <http://www.raspberrypi.org/downloads>
- [4] NOOBS preinstalled: <http://swag.raspberrypi.org/products/noobs-8gb-sd-card>
- [5] Formatting tool for Windows: https://www.sdcard.org/downloads/formatter_4/eula_windows/
- [6] Formatting tool for Mac OS: https://www.sdcard.org/downloads/formatter_4/eula_mac/
- [7] Arch Linux: <https://archlinux-arm.org/platforms/armv7/broadcom/raspberry-pi-2>
- [8] FedBerry: <http://fedberry.org>
- [9] Raspbian: <http://www.raspbian.org>
- [10] RISC OS: <https://www.riscosopen.org/content/downloads/raspberry-pi>
- [11] GPIO: <http://learn.adafruit.com/adafruit-raspberry-pi-lesson-4-gpio-setup/overview>
- [12] NOOBS on GitHub: <https://github.com/raspberrypi/noobs/blob/master/README.md>

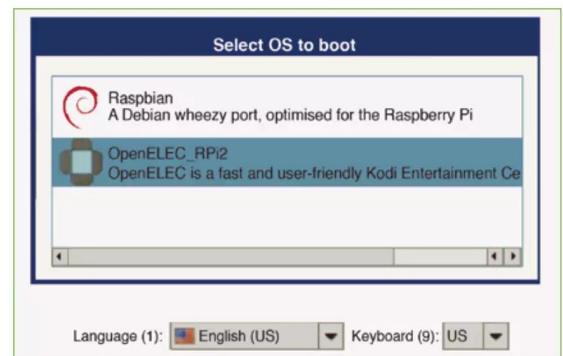
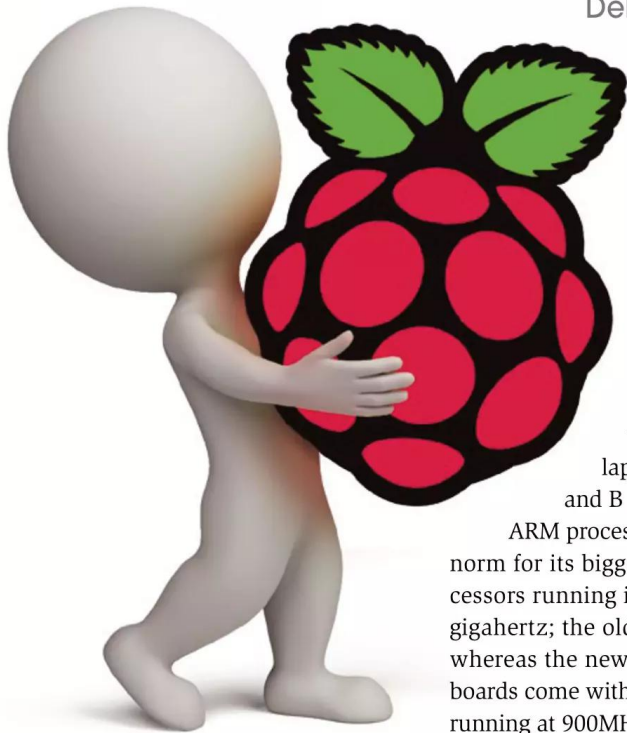


Figure 3: After installation is complete, NOOBS displays a boot menu that allows you to launch one of the installed operating systems.

The software of Raspberry Pi

Softberries

Raspbian, the official operating system for the Raspberry Pi, comes with some preinstalled software, but you'll find much more in the Debian repositories. We walk you through what's immediately available, what you can install, and what you should avoid. *By Paul C. Brown*



Despite its unquestionable merits, the Raspberry Pi is not as powerful as even the most modest modern laptop. The original models A and B (Pi1A and Pi1B) run a single ARM processor at 700MHz, whereas the norm for its bigger brothers is multiple processors running in the region of several gigahertz; the old Pi1B has 512MB of RAM, whereas the new Rasp Pi 2 Model B (Pi2B) boards come with a Quadcore ARM Cortex-A7 running at 900MHz with 1GB of RAM.

However, even low-end laptops nowadays run Core i3 processors at over 1GHz with several gigabytes of RAM. Furthermore, although you can easily find machines with terabytes of storage space, the Pi is limited to what fits on an SD card.

Given these resource limitations, it is still amazing what the Pi can handle in the way of large and complex applications. The Pi can also comfortably fulfill its role as an educational tool and even as a lightweight workstation, as long as you know what you're doing.

On the Menu

The Raspberry Pi was originally conceived as a cheap and portable device to aid teaching primary and secondary students about computers and how to program. The software that comes with the Raspbian reflects this history. Apart from essentials, such as a web browser, accessories, and apps to personalize your desktop, you'll find a well populated *Programming* submenu with various links,

including a link to the Scratch educational programming environment [1]. Scratch uses virtual Lego-like blocks of commands and language structures that snap together to build complex programs. (See the “Scratch Programming” article for more information.)

In the panel along the top of the desktop, you'll find the high-level Wolfram Language “with built-in support for solving a very wide variety of computational problems” [2] and Mathematica, for computer-based mathematical computations. (See the “Mathematica on Pi” article.)

Back in the *Programming* submenus, you'll find a relative newcomer to the Raspberry Pi menagerie: *Sonic Pi* (Figure 1), a framework and language that allows students to learn music through programming or, take your pick, programming through music. As in Scratch, the panel is divided into several panes, including an editor pane (top left), where you input your code, and a log viewer (right), which shows messages about what is happening. The bottom panes have a tabbed tutorial with examples, descriptions of synths (artifacts that generate synthetic sounds), pre-recorded samples, effects, and a glossary of the language's keywords (left) with a pane for explanations (right) for whatever you choose on the left.

Sonic Pi is a really versatile and powerful language, and it comes with enough samples and effects to make it fun for kids and adults.

The *Python 2* and *Python 3* entries open IDLE and IDLE3 (Figure 2), two nearly identical IDEs – integrated development environments – that allow you to develop and test Python programs. Two IDEs are present because Python developers are currently transi-

tioning from version 2 of the language to version 3, and they are not completely compatible with each other [3].

When you run IDLE, the program opens a Python shell in which you can try out commands, set variables, and test structures such as loops, functions, and classes. To start writing programs, click *File | New Window*. To see examples of programs, you can click *File | Open* and navigate to the *python_games* directory to get a list of preinstalled and relatively simple Python applications.

Although a full tutorial for the Python programming language is well beyond the scope of this article, you can find many resources online and in print [4] [5].

The *Office* menu contains office productivity software from the LibreOffice suite. Click on *Office* and you will see applications in the LibreOffice collection. If you're accustomed to Windows systems, you'll notice that LibreOffice is similar to the Microsoft Office suite, with a word processor (LibreOffice writer), a spreadsheet (LibreOffice Calc), a database (called Base), and a presentation tool (called Impress).

In the *Internet* menu, apart from the Epiphany web browser, you'll find the Claws email program and *Raspberry Pi Resources*. *Raspberry Pi Resources* consists of links to educational projects and other HowTo information for students. Also in the *Internet* menu is a link to the Pi Foundation's *MagPi* magazine.

The *Games* menu has entries to run the Python games mentioned earlier and Minecraft for Raspberry Pi (Figure 3). Note that the Pi version of Minecraft is a bit limited in that you can only use the "creative" playing



Figure 1: Sonic Pi allows students to learn music through programming and programming through music.

mode. This is similar to early versions of the game for mobile devices. This mode has no mobs (animals, monsters, and non-player humans – "villagers"); you can dig, but you can't pick up blocks (mine) or transform combinations of blocks into new objects (craft). Instead, you have an inventory of all the blocks and tools you need, of which you have an infinite amount, and you use those to do your building. This may make the Pi version less fun to start with, but it is offset by the fact that the creators have included a simple API that lets you program new features into the game using Python.

In *Accessories*, you can find things like a calculator, a text editor, an image viewer, a PDF viewer, and so on. You will also find a very useful terminal emulator, and some versions of Raspbian include a utility for configuring a WiFi connection. The Rasp Pi does not come with WiFi onboard, but many USB WiFi dongles will work fine [6].

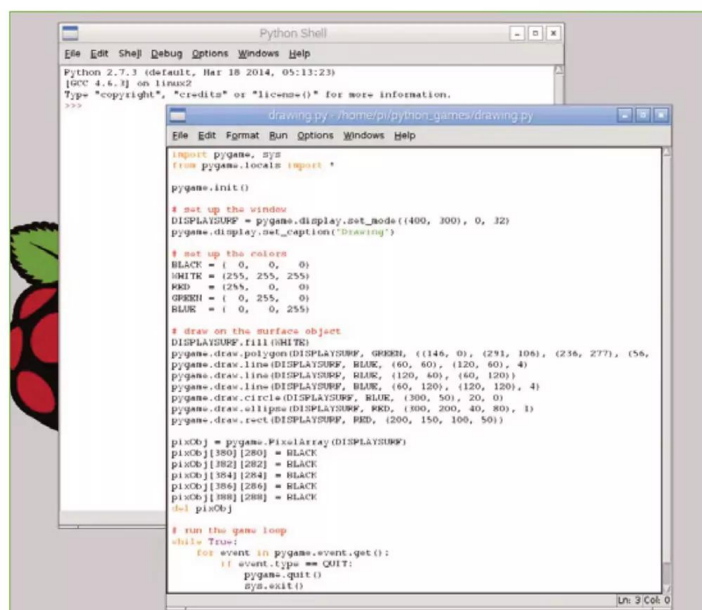


Figure 2: IDLE is the integrated development environment of choice for the Python programming language.



Figure 3: Minecraft for Raspberry Pi allows you to expand the game with your own Python programs.

Getting More Stuff

Once you start exploring the menus, you'll notice some glaring absences. For instance, you might be missing an image manipulation program or one of your favorite games.

Raspbian is pretty bare bones, but this is deliberate: It is up to you to furnish your system with the applications you need. As explained in the "Understanding Linux" article elsewhere in this issue, the terminal and Apt package manager are the way to download and install software.

Suppose, for example, that you want to install Gimp, the free image manipulation program. You can open a terminal and type:

```
apt-cache search gimp
```

to get the exact name of the package to install.

Before you install (`sudo apt-get install gimp`), you should always:

1. Update your repositories and system:

```
sudo apt-get update
```

2. Check the application dependencies. Remember, resources on the Rasp Pi are more limited than on your average computer. If the application you want to install is going to drag in a ton of dynamically linked libraries, which are loaded into memory

when the program runs, chances are it will run slowly or not at all.

Before installing, you can simulate an install with the `-s` modifier to see what is going to be copied to the Rasp Pi. For Gimp, you would type

```
apt-get -s install gimp
```

The number of dependencies Gimp pulls in is considerable but not terribly excessive. If you compare Gimp's list of dependencies with that for Krita, another open source image manipulation tool, you'll see that Krita's list is way beyond reasonable. Your best bet in this case is to go with Gimp.

As described earlier in the article, Raspbian comes with the LibreOffice office suite, which is very versatile but also very resource hungry. You might wish to use a lighter word processor, such as AbiWord (Figure 4), or you could try Gnumeric (Figure 5), a great and nimble spreadsheet program.

You can enter:

```
sudo apt-get install abiword gnumeric
```

to install both programs in one step.

Memory Splitting

Given the information so far, you might think that the older Pi models wouldn't be very good for games. You'd be wrong. The fact is that one of the most famous videos on YouTube of the early models was of users playing Quake 3 on the device.

Id Software's famous first-person shooter can be installed with Apt; however, to run it, you might have to assign more memory to the graphics processor. You can work out how much memory is taken up and by what component with the `free` command. Listing 1 shows a typical split for a Pi1B.

The `-m` modifier tells `free` to show the results in megabytes, instead of bytes. The `total` shown in the first column is the memory assigned to the CPU. Subtract that number from the total memory on your Raspberry Pi (256MB on an Pi1A, 512MB on a Pi1B, and 1,024MB on the Pi2B and Pi 3) and take away approximately 10MB more for Linux's internal use, and you'll get the amount of RAM assigned to the GPU. This normally will be an exact power of 2 (i.e., 64, 128, or 256). For the case in Listing 1,

$$512 - (438 + 10) = 64,$$

which is the default GPU memory allocation.

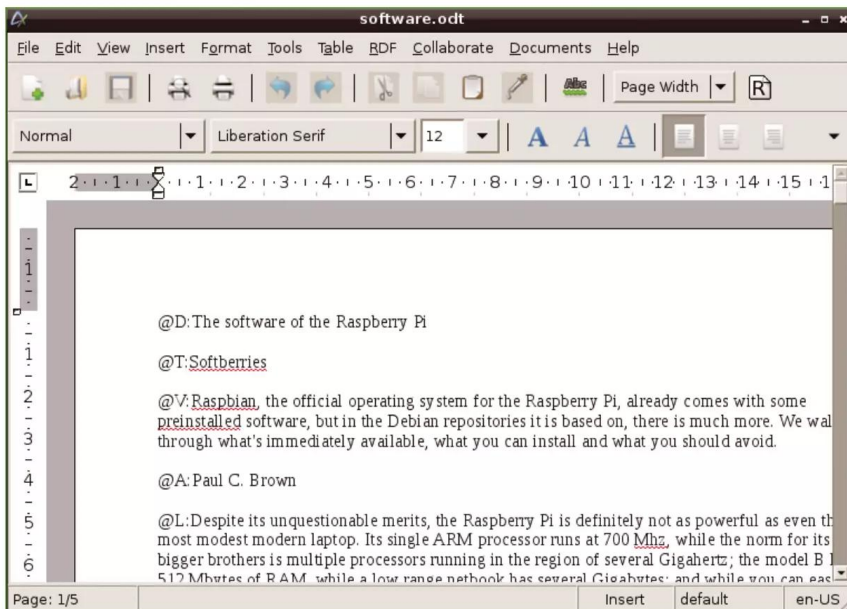


Figure 4: AbiWord is a full-featured but lightweight word processor.

LISTING 1: Calculating Assigned Memory

```
$ free -m
```

	total	used	free	shared	buffers	cached
Mem:	438	359	79	0	14	223
-/+ buffers/cache:		121	317			
Swap:	99	0	99			

To run graphics-intensive applications you might need to assign more RAM to the GPU, especially if you have an older system. To assign more RAM to the GPU, run the `raspi-config` tool

```
sudo raspi-config
```

and choose *Advanced Options* | *Memory Split* from the menu.

This option brings up a dialog in which you can enter the amount of memory you want to assign to the GPU (Figure 6). How much you need depends on the application: Assign too little, and the program could run slowly or crash. Assign too much, and the rest of the system could become slow or unresponsive. In the Raspberry Pi forums [7], you can see how other users have set up their Rasp Pis, or you could just experiment. After all, you are not going to damage your Pi just by changing the memory split.

Once you have configured the split, you will have to reboot the Pi for the changes to take effect. Use `free` again to make sure the memory is shared to your liking.

Media Players

Media players and media centers are also graphics-intensive applications that often require more than the default RAM assigned to the GPU. If you want to watch videos locally, your best bet is probably VLC, which can be installed with:

```
sudo apt-get install vlc
```

Although VLC can stream video over a network, if you're looking for a feature-rich multimedia center that can show photos and play music and videos over a network, you probably want to take a look at Kodi (Figure 7). (Kodi used to be called XBMC and is still referred to as XBMC in some repositories.)

If you do install XBMC/Kodi on Raspbian (see the "Kodi Media Center" article in this issue), it will probably be sluggish no matter how much RAM you assign the GPU. The correct approach is to install the new Kodi as a standalone media center (e.g., via the OpenELEC or OSMC distributions [8]).

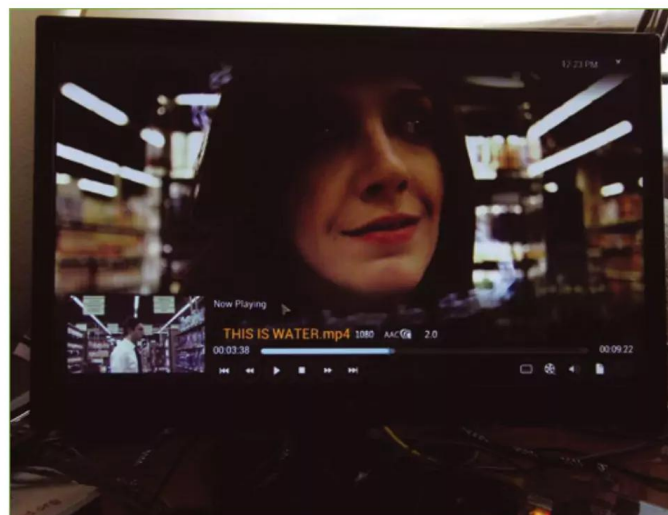


Figure 7: XBMC/Kodi is a free, full-fledged media center that also looks great.

Figure 5: Gnumeric is a good alternative to more bloated spreadsheet programs.

This solution works because you take Raspbian completely out of the equation by booting a minimal Linux that takes up very few resources, leaving most of the RAM and other resources to Kodi.

Conclusion

Although you can do a lot with the Raspberry Pi, you can hit a wall quite soon if you're not careful. To avoid this, you need to understand the limitations and look around for lightweight applications. It also pays to understand whether the software you want to run is more processor or graphics intensive and adjust your RAM accordingly.

That said, the Raspberry Pi is amazingly capable for its small size, and you'll be surprised at what it can handle. ●●●

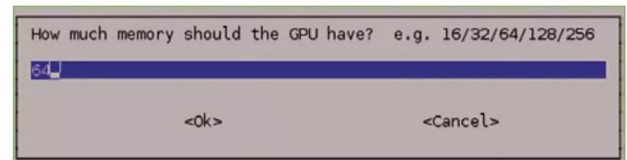


Figure 6: Use `raspi-config` to assign more memory to the GPU for graphics-intensive apps.

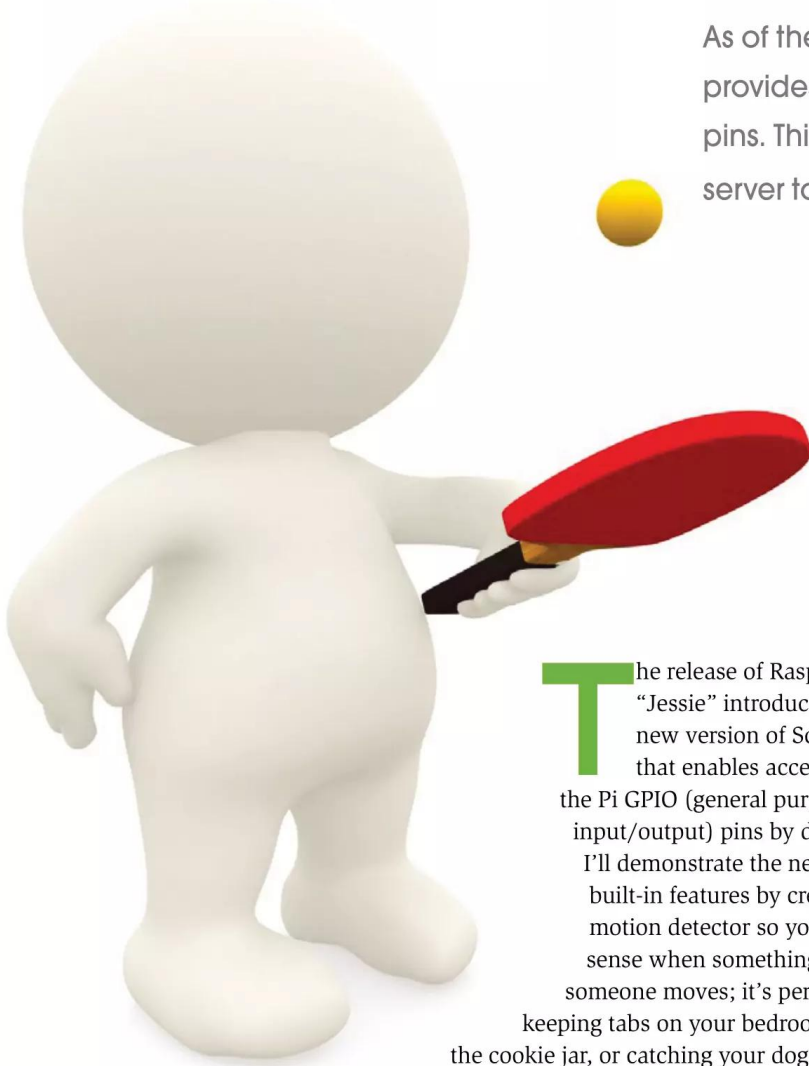
INFO

- [1] Scratch: <http://scratch.mit.edu/>
- [2] Mathematica and the Wolfram language: <http://www.raspberrypi.org/mathematica-on-raspberrypi-a-guest-post-from-wolfram-research/>
- [3] Python: <http://www.python.org/>
- [4] "Python Workshop: Part 1" by Mike Mueller, *Ubuntu User*, issue 15, pg. 50
- [5] "Python Workshop: Part 2" by Mike Mueller, *Ubuntu User*, issue 16, pg. 52
- [6] Compatible WiFi dongles: http://elinux.org/RPi_USB-Wi-Fi_Adapters
- [7] Raspberry Pi forums: <http://www.raspberrypi.org/forums/>
- [8] Raspberry Pi operating systems: <http://www.raspberrypi.org/downloads/>

Building a Motion Detector using Scratch GPIO server

In Motion

As of the Raspbian Jessie release, Scratch provides easy access to the Raspberry Pi's GPIO pins. This project will incorporate the new GPIO server to build a motion detector. *By Michael Badger*



The release of Raspbian “Jessie” introduces a new version of Scratch that enables access to the Pi GPIO (general purpose input/output) pins by default.

I’ll demonstrate the new built-in features by creating a motion detector so you can sense when something or someone moves; it’s perfect for

keeping tabs on your bedroom or the cookie jar, or catching your dog eating off the counter when you’re not home. In addition to an updated version of Scratch, you will need two LEDs, resistors, a passive infrared (PIR) sensor, and your breadboard to build a motion detector.

Accessing the Pi’s GPIO pins from within Scratch has been possible for several years, but it previously required a separate installation of Scratch called ScratchGPIO, which I’ve used in previous Raspberry Pi articles, from Simon Walters (aka *cymplecy*). The Scratch GPIO server now enables access from the default installation of Scratch. In addition to the GPIO pins, you gain access to several other commands to get the time, IP address, and photos from the camera module.

The Scratch GPIO server – like the Scratch-GPIO and Mesh features I’ve covered in previous *Raspberry Pi Geek* articles – rely on broadcast messages to communicate outside of the Scratch interface. This puts a special emphasis on using carefully constructed commands (syntax) to make Scratch do what you expect. In a Scratch programming environment, the focus on syntax creates a more advanced deviation from the drag-and-drop interface that makes Scratch so easy to understand.

UPGRADING SCRATCH AND RASPBIAN

Before you can build this project, you’ll need to install or upgrade to Raspbian Jessie [1]. To complete the project, it’s important that you are using the September 2015 version or later of Jessie that introduced the new Scratch.

TURNING ON THE GPIO SERVER

When you create a new project, the GPIO server is not turned on, but it’s the GPIO server feature that will provide the communication between Scratch, the Pi, and the pins. The easiest way to activate the server is to click on Scratch’s *Edit* menu and chose *Enable GPIO server*.

THE AUTHOR

Michael Badger wrote the Scratch 1.4 and 2.0 Beginner’s Guide series from Packt Publishing. Learn more at scratchguide.com

However, you can also control the state of the server by sending the message `gpioserveron` in a `broadcast()` block, as seen in Figure 1. This ensures that the GPIO server is always enabled when the project runs, assuming the person running the project has the appropriate version of Scratch.

Scratch will remember the state of the GPIO server. If you enable it in a project, the next time you open the project, the GPIO server will be on. However, it's always a good idea to turn on the server as part of the project initialization if the functionality is required for the project. If you want to turn off the server, broadcast a message named `gpioserveroff`.

CONFIGURING PINS AND LIGHTING LEDs

As I work toward the motion detector, I'll first show how to control an LED through the GPIO server so that you can see the steps necessary to configure and control a pin. For the simple circuit, connect an LED to pin 15 through your breadboard. Make sure to use a resistor to protect the LED.

The Scratch GPIO server uses the BCM numbering system to address pins, rather than the physical pin location. Refer to Figure 2 to identify the pin numbers. BCM refers to the chip maker, which is Broadcom. The BCM numbering does not follow a human-friendly pattern, which is why you should always reference a pin diagram.

Before you can turn the LED on or off, you need to configure it at the start of the project with a broadcast message. The `broadcast(config15out)` block accomplishes the goal. The command to configure the pin consists of three parts (`config` + pin number + configuration option).

For this example, the only configuration option I'm using is `in` or `out`, but there are others. Using `in` or `out` sets the specified pin number as either an input or an output. Inputs take sensory information from the physical world and send it to Scratch. An example is the PIR sensor that I'll use later in this project. Outputs send information from Scratch to the physical world, such as lighting an LED.

Using the GPIO server, a pin can be configured as either input or output, making it critically important to your project that you specify the configuration before you attempt to use it.

You can experiment with the commands by turning an LED on and off. Figure 1 shows a script that turns the LED on and then off, which is happening with the `gpio15on` and

`gpio15off` broadcast messages. When you control a pin, the message consists of three parts (`gpio` + pin number + setting). My example uses `on` and `off` as the settings, but you can also substitute `low` and `high`, respectively.

ADDING A PIR SENSOR

The PIR sensor detects movement. The sensor has three labeled pins. Connect the sensor pin labeled GND to the ground on the Pi. Connect to pin labeled OUT to pin 21.

The sensor pin labeled VCC can be connected to a 5V or 3.3V pin on the Raspberry Pi, depending on the power requirements of your sensor (Figure 3). My sensor happens to work with either. Refer to the pin diagram in Figure 2 to find the power pins. The 5V and 3.3V power pins supply power to the sensor continuously.

To detect movement, you need to monitor the OUT pin of the sensor, which should be connected to pin 21 on the Raspberry Pi. As part of the project initialization, set pin 21 as an input. Figure 4 shows the initial configuration for all the pins in the project.

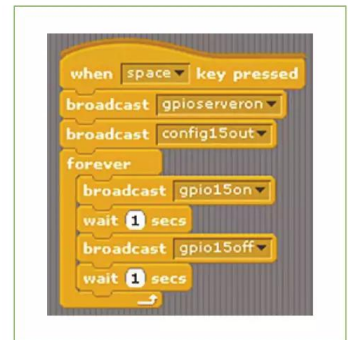


Figure 1: A script to turn an LED on and off using the Scratch GPIO server.

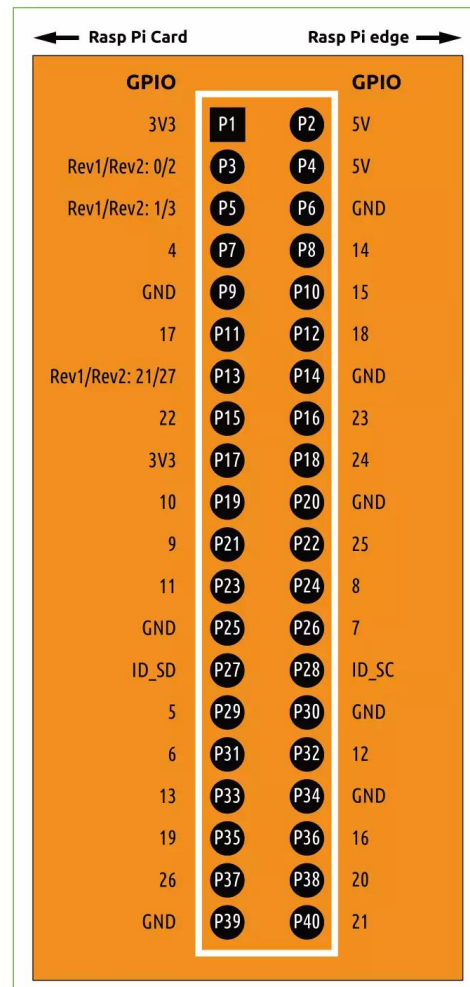


Figure 2: BCM pin numbers are labeled "GPIO" here.

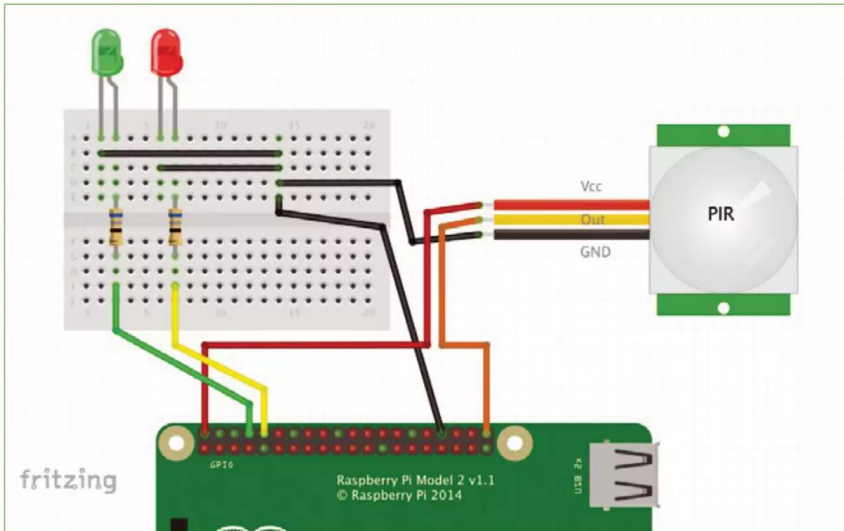


Figure 3: Diagram of the wiring between the PIR sensor, the Rasp Pi, and the LEDs. (Fritzing)

Pins 14 and 15 are configured as outputs. Connect pin 14 to a green LED via a resistor and connect pin 15 to a red LED through a resistor. The LEDs provide a visual indicator to help you see if motion is detected; red will signal motion, and green will signal no motion.

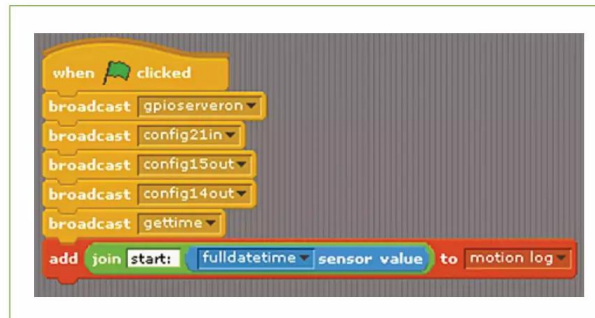


Figure 4: Script to configure the initial values for the motion detector project

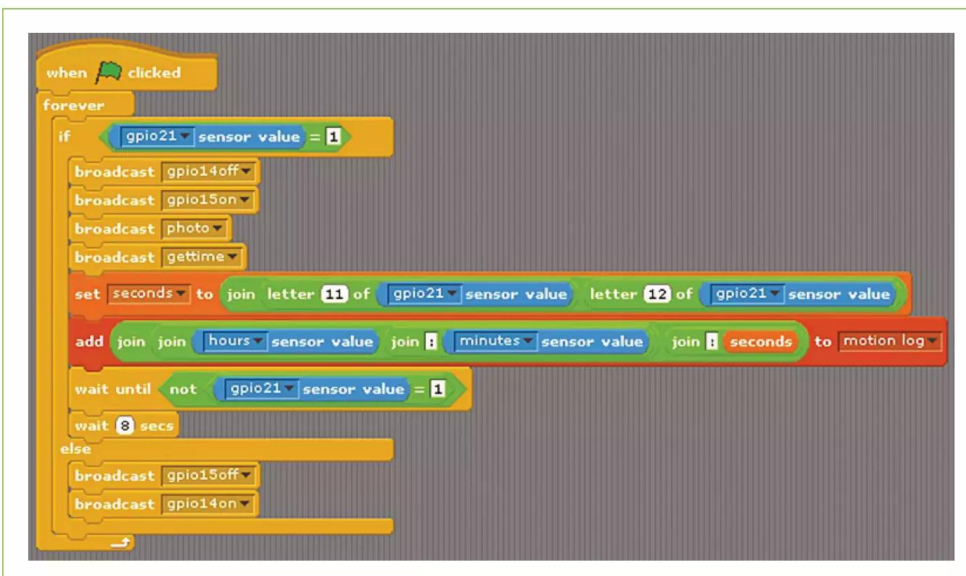


Figure 5: Script to monitor the PIR sensor value and detect movement.

DETECTING MOTION

The PIR sensor reports a value of 0 (no motion is detected) or 1 (motion detected). The script checks for movement by evaluating the sensor value of GPIO21, and when it's equal to 1, it turns on the red LED and records an entry in the motion log list (Figure 5).

If the sensor value is not 1, then turn on the green LED to signify there is no motion. You'll also notice that when the green LED is lit – that is, `broadcast(gpio14on)` – the script turns off the red LED – `broadcast(gpio15off)` – and vice versa. This ensures that both LEDs are never lit simultaneously.

GPIO input pins will display as variables in the () sensor value block located in the Sensing palette. Before the pin will show as a sensor value, you first need to run the broadcast block that configures the pin as an input. For example, create the `broadcast(config21in)` block and then run it. After you run the block (e.g., by double-clicking the block), the pin will show in the list of sensor values.

Before I move on to the new non-GPIO commands you see in the script, I want to draw your attention to the wait until()

block that runs when the sensor detects motion. This block provides some additional control by waiting until there is no more movement (the GPIO21 value is not 1). That keeps the script from continually evaluating to true and writing consecutive entries in the motion log for the same motion event.

Non-GPIO COMMANDS IN SCRATCH SERVER

The Scratch GPIO server introduces three new commands that interact with other parts of the Pi but are not connected to the GPIO. Those new broadcast commands are `getip`, `gettime`, and `photo`. In the script, I use `photo` and `gettime`; `getip` returns the IP address of the Pi. The `photo` command integrates with the Raspberry Pi camera. When you use the `photo` command, the camera takes a picture and adds the image as a costume to the sprite.

The `gettime` command creates the following sensor variables: `fulldatetime`, `hours`, and `minutes`. The script in Figure 4 ini-

tializes the motion log list with a beginning timestamp using the `fulldatetime` value. Then, each time the script (Figure 5) detects motion, it records the event by adding the hours, minutes, and seconds to the motion log. Whereas hours and minutes variables are default sensor values, seconds is a variable created by me from the `fulldatetime` value in the `set()` line.

When you examine the `fulldatetime` value, you will note a 12-digit number in the format year + month + day + hour + minute + second, which can also be represented as `YYMMDDHHMMSS`. As you see in the script, you can extract the 11th and 12th values of `fulldatetime` and join them together to create a standalone seconds value.

If you are fortunate enough to have a camera board attached to your Pi, then this script will correlate the entry in the motion log to the costume number. For example, the first detected motion will be in list position two and the first photo would be costume two (assuming you start with a one-costume sprite).

The `broadcast(gettime)` block retrieves the date information and makes it available

via the sensor values, but the time values are not continuously updated after you send the `gettime` broadcast one time. Each time you want to use the date and time, you need to run `broadcast(gettime)` to ensure you get the current values.

If your time values are not correct, you can set your time zone by using the `raspi-config` utility. Run it from a terminal window with the command `sudo raspi-config`. The Raspberry Pi defaults to the UTC (Coordinated Universal Time).

GETTING MORE INFORMATION

For additional Scratch GPIO server options and configurations, check out the Scratch GPIO documentation [2]. The project files for this article can be found at my website [3]. Happy Scratchin'.

...

INFO

- [1] Raspian Jessie: <https://www.raspberrypi.org/documentation/installation/installing-images/README.md>
- [2] Documentation: <https://www.raspberrypi.org/documentation/usage/scratch/gpio/README.md>
- [3] Scratch Guide: <http://www.scratchguide.com>

Shop the Shop

shop.linuxnewmedia.com

Discover the past and invest in a new year of IT solutions at Linux New Media's online store.

Want to subscribe?

Need training?

Searching for that back issue you really wish you'd picked up at the newsstand?

➤ shop.linuxnewmedia.com

DIGITAL & PRINT SUBSCRIPTIONS



SPECIAL EDITIONS



TRAINING



- LPIC-1 LPI 101 - CompTIA Linux+ LX0-101
- LPIC-1 LPI 102 - CompTIA Linux+ LX0-102
- LPIC-1 - CompTIA Linux+ 101 + 102

The Raspberry Pi as a file or print server

Store and Print

Thanks to its tiny size and low power consumption, the Pi is well suited for the role of a miniature 24/7

home file server. *By Marko Dragicevic*

Most households today are networked, and exchanging data on the local wireless network is not just the domain of desktop PCs; tablets and smartphones also can join in. A file server lets you keep personal files in a central location. Many home networks also use a local file server to schedule backups and support network printing.

When I think of a solution that gives me all of this functionality quickly, inexpensively, and flexibly, I think of the Raspberry Pi. Its cost is negligible, the power consumption is low, and you can configure all the features I've mentioned with a little time and a handful of console commands. The only downside is that, although the Raspberry Pi is perfect as a server solution in the home, a busier network will quickly push it to its limits. Thus, a company with 100 employees doesn't need to consider the Raspberry Pi, but for private purposes, the tiny Pi is all you need to set up a home file server system.

In this article, I will look at how you can set up your Rasp Pi as a file server. The possibilities are endless. If you have additional questions, consult the many excellent online tutorials for working with Samba,

rsync, CUPS, and the other tools described here.

The Basic Setup

To begin, download the latest version of the Raspbian distribution [1] and flash it onto an SD card. For the very first configuration steps, I will assume that your Pi is connected to a monitor. After booting from the newly installed SD card, you end up in a dialog that offers a few basic settings (you can access this any time later with the `sudo raspi-config` command). As with any other Raspbian reconfiguration, it makes sense to change the root password with the *Change User Password* menu item and fully exploit the entire space on the SD card using the *Expand Filesystem* item. Because I am setting up a server, the *Enable Boot to Desktop/Scratch* option is also important: If you set *Console Text console* here, you can avoid unnecessarily wasting resources. If you *Enable* the SSH selection under *Advanced Options*, you can use the Pi in headless mode without a monitor from now on.

Lead Image © kancho Kostov, 123RF.com

To discover the IP address of the Pi on the LAN, you will also want to run `ifconfig` and record the information that follows *inet addr*:. Now shut down the device and put it in its future place – for example, next to the router you want to connect it to via a patch cable. Otherwise, the only additional hardware you need is a power lead for the power supply.

After turning on the Raspberry Pi, sit down in front of another device that has a monitor and log in to the Pi – for example, by using the shell command:

```
ssh pi @192.168.2.129
```

You will need to replace the IP address with the one you noted in the `ifconfig` output. If you have changed the pi user's default password (it is *raspberrypi* for Raspbian), you need to use the new password. Also remember to set up your router so that the DHCP server always assigns the same IP address to the Pi. In the course of this article, the examples will assume that the IP is 192.168.2.129. Replace this with your own address.

External Storage

Theoretically, you could just use the space on the SD card as storage for the file server; however, in most cases, you'll need more space – especially if you want to store movies, music, or backup files on the file server. If you want room for your file server to grow, it is a good idea to outfit it with a permanently connected USB stick or external hard drive.

Format your external storage medium with a stable, Linux-supported filesystem (e.g., ext4). If you're not familiar with how to format a storage drive in Linux, see the ext4 how-to [2]. After plugging the drive into the USB port on the Raspberry Pi, run `dmesg` on the console. This command returns the current kernel message buffer. Toward the end of the output, you will see a string something like *sda: sda1* (Figure 1). This shows the shortcut the operating system uses to access the partition on the USB stick.

Next, run the commands

```
sudo mkdir /mnt/ExternalStorage
sudo nano /etc/fstab
```

and add the following line to the `fstab` file (replacing *sda1* with the name you saw in the `dmesg` output):

```
/dev/sda1 /mnt/ExternalStorage 2
ext4 defaults 0 0
```

This command automatically mounts the external device on every boot in the file tree below `/mnt/ExternalStorage/`. The following configuration steps assume that this directory exists. So that all services and users can later write to and read from this storage space, you now need to run another recursive command:

```
sudo chmod -R ugoa+rwX /mnt/ExternalStorage
```

If you are using an external hard drive instead a USB stick, make sure it does not consume too much power (the Pi itself requires very little). `hdparm` can help here. Simply enter

```
sudo hdparm -S 12 /dev/sda
```

in the `/etc/hdparm.conf` file (replacing `/dev/sda` with the directory for your external device). Then, after a 60-second idle period, the hard drive automatically performs a power-saving spin-down. The number after the `-S` parameter multiplied by five is the number of seconds to wait (in this case, $12 \times 5 = 60$).

Samba

The next step is to install the appropriate services for all the communication protocols used by the people and devices on your LAN.

```
[ 9.527580] bcn2835 ALSA chip created!
[ 9.537025] bcn2835 ALSA chip created!
[ 9.545745] bcn2835 ALSA chip created!
[ 9.553111] bcn2835 ALSA chip created!
[ 17.682520] smc95xx 1-1.1:1.0: eth0: link up, 100Mbps, full-duplex, lpa 0x40E1
[ 19.852580] Adding 102396k swap on /var/swap. Priority:-1 extents:1 across:102396k SS
[ 1343.912036] usb 1-1.2: new high-speed USB device number 5 using dwc_otg
[ 1344.364437] usb 1-1.2: New USB device found, idVendor=090c, idProduct=1000
[ 1344.364468] usb 1-1.2: New USB device strings: Mfr=1, Product=2, SerialNumber=0
[ 1344.364486] usb 1-1.2: Product: Spaceloop 4GB
[ 1344.381957] scsi0 : usb-storage 1-1.2:1.0
[ 1345.609317] scsi 0:0:0:0: Direct-Access Spaceloop 4GB 1100 PQ: 0 ANSI: 0 CCS
[ 1345.613610] sd 0:0:0:0: [sda] 7884800 512-byte logical blocks: (4.03 GB/3.75 GiB)
[ 1345.614602] sd 0:0:0:0: [sda] Write Protect is off
[ 1345.614638] sd 0:0:0:0: [sda] Mode Sense: 43 00 00 00
[ 1345.615617] sd 0:0:0:0: [sda] No Caching mode page present
[ 1345.615648] sd 0:0:0:0: [sda] Assuming drive cache: write through
[ 1345.624592] sd 0:0:0:0: [sda] No Caching mode page present
[ 1345.624627] sd 0:0:0:0: [sda] Assuming drive cache: write through
[ 1345.626025] sda: sda1
[ 1345.629975] sd 0:0:0:0: [sda] No Caching mode page present
[ 1345.630009] sd 0:0:0:0: [sda] Assuming drive cache: write through
[ 1345.630028] sd 0:0:0:0: [sda] Attached SCSI removable disk
pi@raspberrypi ~ $
```

Figure 1: Finding the USB stick with `dmesg`.

If you have a mixed network, you will want to use Samba as a file service. Samba uses the SMB file service protocol used on Microsoft networks, and it is compatible with Windows, Mac, and Linux systems.

After installing via

```
sudo apt-get install samba samba-common-bin
```

you can edit the Samba configuration file by typing:

```
sudo nano /etc/samba/smb.conf
```

If the other computers that need to access the Pi, use a specific workgroup name, edit the line `workgroup = WORKGROUP` to match. If you do not want to give everyone on the LAN access to the file server, you can easily enable authentication. To do so, specify `security = user` and delete the hash used to comment it out. Now scroll down to the bottom of the file and add the lines in Listing 1, save the file, and start the service with

```
sudo /etc/init.d/samba restart
```

for the changes to take effect. First, create a user account for the shell. If you want the other computers to authenticate with a username of *fileserver*, enter

LISTING 1: Additions to `smb.conf`

```
[public]
comment = Public
path = /mnt/ExternalStorage/
valid users = @users
force group = users
create mask = 0660
directory mask = 0771
read only = no
```

LISTING 2: Addition to `rsyncd.conf`

```
use chroot = true
hosts allow = 192.168.2.0/24

transfer logging = true
log file = /var/log/rsyncd.log
log format = %h %o %f %l %b

[public]
comment = Public
path = /mnt/ExternalStorage/
read only = no
list = yes
uid = nobody
gid = nogroup
```

```
sudo useradd fileserver -m -G users
```

and change the shell password via

```
sudo passwd fileserver
```

The command

```
sudo smbpasswd -a fileserver
```

now gives this user a Samba password required for access to the service itself. Other computers can now use their file managers to search the network for drives and mount the Raspberry Pi's storage.

If you don't want to go to the trouble of setting up a complete Samba configuration and you only have to transfer files occasionally, you might want to use the SSH File Transfer Protocol (SFTP) instead. If you have an SSH server running, no additional configuration steps are needed. (See the "Remote Access" article elsewhere in this issue.) In the latest version of Raspbian, even the basic configuration supports SFTP transfers.

Backups

You might also want to use the Pi's storage for backups (e.g., to back up your daily work across the network). Rsync is a popular network backup tool. It synchronizes data between two network locations so that only files with a newer timestamp (i.e., files that have been modified since the last sync operation) are transferred.

Because `rsync` is a shell command, you can add it as a cronjob to execute on a regular schedule [3]. Install `rsync` on the Pi and then open the configuration file for the `rsync` daemon with a text editor:

```
sudo apt-get install rsync
sudo nano /etc/rsyncd.conf
```

The `rsyncd.conf` file will probably not contain any text. Paste the information from Listing 2 into that file. This allows any device in the IP range of your local LAN to synchronize with the USB stick connected to the Pi. (If you use an IP address range different from 192.168.2.0 on your LAN router, you need to adjust accordingly.) Now, when you run the console command

```
rsync -avz ~/workspace/* 2
rsync://192.168.2.129/public
```

on your desktop PC, all the files and subdirectories from `~/workspace/*` that have under-

gone changes since the last backup operation will be uploaded to your Pi.

If only selected users are allowed to perform backups, then add `auth users = backup` to the `rsyncd.conf` file (this means that only the user `backup` can run `rsync`); adding `secrets file = /etc/rsyncd.scr` means that the password for this user is stored in the file `/etc/rsyncd.scr`. You only need to create the file and add a line such as `backup:password`.

Print Server

Your Raspberry Pi also can act as a network print server. A tool known as CUPS (Common Unix Printing System) [4] lets you configure your Pi as a print server and use the second USB port to connect a printer accessible by all devices on the LAN. After installing CUPS (`sudo apt-get install cups`), you need to release the printer to the network:

```
sudo cupsetl --share-printers \
--remote-printers --remote-admin
```

Adding `--remote-admin` means you also can manage the print server in the browser on your PC. However, you still need to give your standard user `pi` access to the printer:

```
sudo usermod -a -G lpadmin pi
```

After typing `https://192.168.2.129:631/admin` into the browser address bar, you should see the configuration interface (Figure 2). Select **Add Printer**, then choose your printer model. When prompted to authenticate, type the credentials for the `pi` user. Now the printer attached to the USB port on your Raspberry Pi will appear under Local Printers (Figure 3). You can select your printer and press *Continue*.

In the next configuration step, you need to specify the printer URI, which depends on the manufacturer and model. The help link for Network Printers on this page helps you discover the correct URI for your device. Press *Continue* and assign a name for the printer, as well as a description of its location to ensure that other users on the network can later identify which device it is (just in case you have a large LAN with many printers). If needed, check the *Share This Printer* box and then click *Continue*. Again select the specific printer type to find the correct driver internally. Finally, press *Add Printer* to complete the configuration.

If you want Windows PC users to use your printer, you need to change the Samba

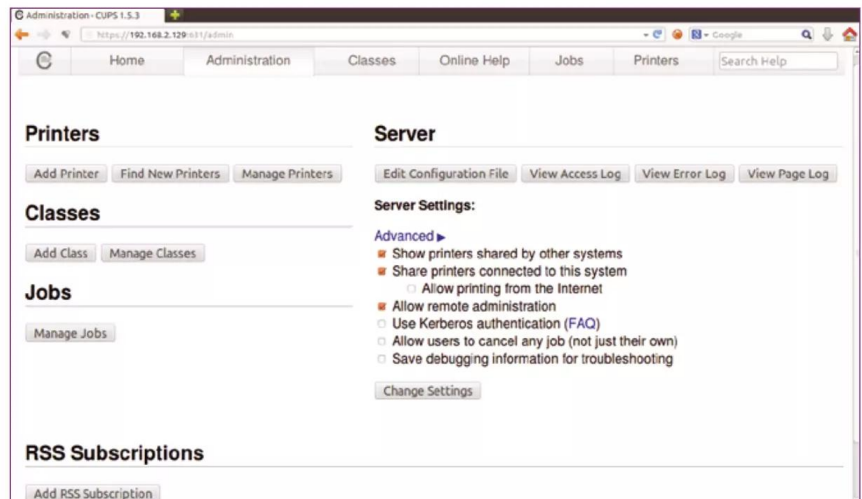


Figure 2: CUPS can also be administered remotely via the browser.

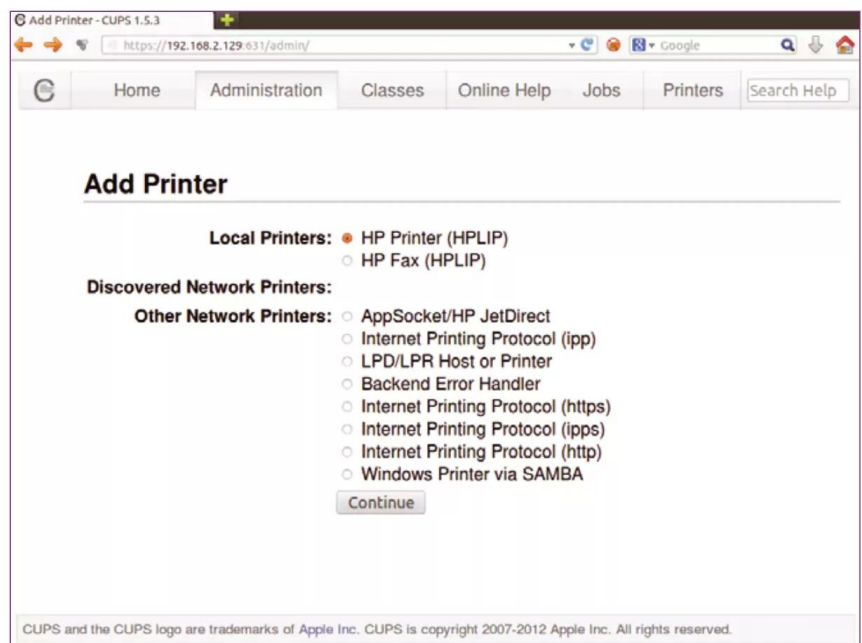


Figure 3: The printer connected to your Pi can be found under Local Printers.

configuration again: Open the `/etc/samba/smb.conf` file, delete the semicolon that comments out the lines `printing = cups` and `printcap name = cups`, and restart Samba (`sudo /etc/init.d/samba restart`). On other systems, the printer can now be added as a normal network printer.

The Future

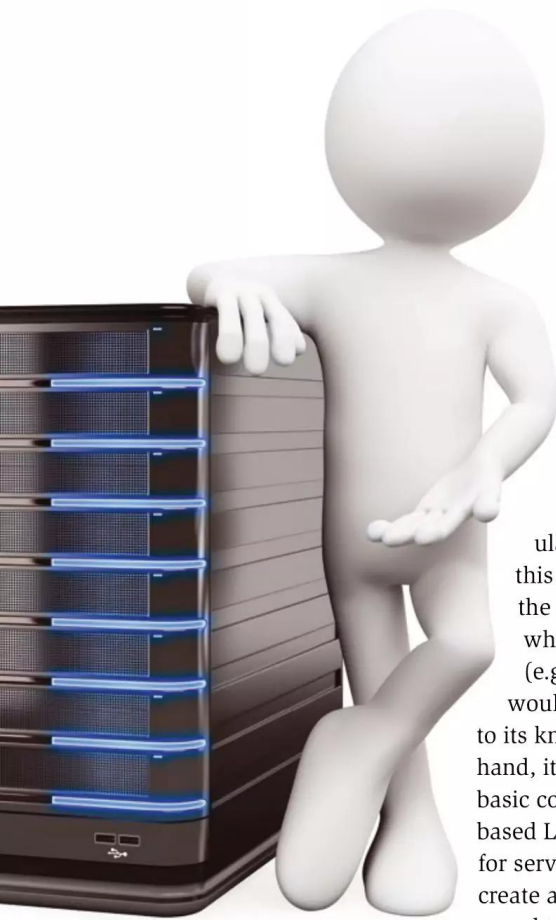
You now have your own file and print server. Remember that your Raspberry Pi has fewer resources than most systems used as file servers, so if you don't need a specific service, you should disable it to reduce the system load. Then, if you need other services later, you can add them. Because Raspbian is a full-fledged Debian derivative, many additional tools and services are available for your Rasp Pi home server. ●●●

INFO

- [1] Raspbian image: <http://www.raspberrypi.org/downloads>
- [2] Ext4 how-to: https://ext4.wiki.kernel.org/index.php/Ext4_Howto
- [3] cron: <http://en.wikipedia.org/wiki/Cron>
- [4] Linux Foundation CUPS printing tutorial: <http://www.linuxfoundation.org/collaborate/workgroups/openprinting/databasecupsprintingtutorial>

The Raspberry Pi as a web server

Web Server @home



Leasing a dedicated server from a data center to create dynamic websites could expose you to the danger of being hacked if you don't have experience administering web servers. Pi on the home LAN is by far the better alternative for beginners.

By Marko Dragicevic

The Raspberry Pi Foundation designed its miniature computer for training purposes, and particularly for the applications considered in this article, it works great. On one hand, the limited hardware lets you see clearly which processes generate too much load (e.g., poorly written PHP scripts) and would thus bring a much stronger server to its knees given more users. On the other hand, it is still powerful enough to practice basic configuration steps within a Debian-based Linux distribution that is often used for server applications. In this way, you can create a website and test it without being exposed to the dangers of hacking because the NAT (network address translation) firewall on the local router protects the Raspberry Pi. In the meantime, if you want to use the mini-computer for some other purpose, you can just boot it from another SD card.

For a Raspberry Pi to act as a server, the first obvious step is to make it “headless” – that is, run without a monitor and managed purely via an SSH connection set up on another computer on the same LAN. The first few sections of the “Pi File and Print Servers” article in this issue contains the details on how to do this; be sure to remember the IP address discovered with the `ifconfig` command.

After switching off the desktop of a freshly installed Raspbian (as described in the file server article) to enable the SSH server and discovering the IP address of the Raspberry Pi, you can log in to it via an SSH connection and carry on with the steps described in this article. However, if you do not have a PC available and instead want to use the Raspberry Pi as a web server, as well as a desktop browser to view the pages you create, then leave out the steps described in the file and print server article. In that case, `127.0.0.1` is always the IP address that points to the local machine (i.e., your Pi).

Apache

Apache is the most popular web server in the world for a good reason. At the beginning of 2013, it was used on 53 percent of all Internet sites, whereas competing software products were all less than 20 percent [1]. On the basis of this long-standing popularity alone, Apache is a good choice because it has many possibilities and any problems you might have are easy to discover on Internet forums if the detailed documentation does not have the right answers. Later, I discuss cases in which a different web server might be a better choice.

If you have the latest version of Raspbian on the SD card, the first step is to run

```
sudo apt-get install apache2
```


In older versions, this command results in an installation error because the user group *www-data* that is necessary for installation does not exist. If, for some reason, you are using an older Raspbian installation, you must run the shell command

```
sudo groupadd www-data
```

before installing Apache. In general, however, it is always a good idea to use the latest stable updates.

Next, enter the IP address of the Raspberry Pi in your browser; you should see the Apache default *index.html* page with the text “It works!” (Figure 1). This index page is stored on your Raspberry Pi in the */var/www/* directory. As a test, you can edit the index page by typing:

```
sudo nano /var/www/index.html
```

After changing and saving the text and reloading the page in the browser on your PC, you should see the changes immediately.

PHP and MySQL

A static website like this seems like a relic from the Internet stone age, so you’ll need to pimp your installation with the use of PHP and MySQL. You can install the requisite packages with:

```
sudo apt-get install php5 2
libapache2-mod-php5 mysql-server 2
mysql-client php5-mysql
```

Sit back and enjoy a cup of coffee. Your nanocomputer has a lot of work to do because it needs to install a number of packages on which these programs depend. Eventually, you will be asked in a text dialog for a password for the MySQL root user. For training purposes, type one that you can remember easily (e.g., *raspberrypi*). Now, using Nano, you can create a script named *firsttest.php* in */var/www*

```
sudo nano /var/www/firsttest.php
```

and enter the following three lines of content:

```
<?php
phpinfo();
?>
```

Now when you call the script via the IP address of your Raspberry Pi (e.g., *http://<IP*

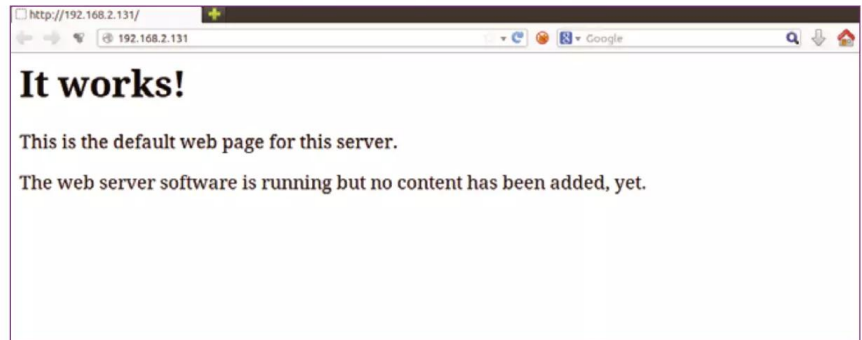


Figure 1: After installing Apache, this terse index page displays.

Address>/firsttest.php), you should see output like that shown in Figure 2. This also indicates which options PHP is configured to use. If you later install a script that does not work for configuration reasons, you can use the *firsttest.php* page to investigate which parameter is causing the problem and change it in the */etc/php5/apache2/php.ini* file.

To manage the content of the MySQL database in the browser, I also recommend installing phpMyAdmin. If you write your own PHP scripts, you can use phpMyAdmin to verify that they are correct and that they store the data as intended. Installation is possible via the standard package manager:

```
sudo apt-get install phpmyadmin
```

Then, you can log in on a PC via the LAN at *http://<IP Address>/phpmyadmin* (Figure 3), replacing the IP address with the one you discovered with *ifconfig* (i.e., per the instructions in the file and print server article). The username is *root*, and the password is the one you specified in the MySQL installation.

Everything should be set up now to allow you to create HTML and PHP pages and use your own Pi as a web server. Two approaches are possible for creating the pages: You can

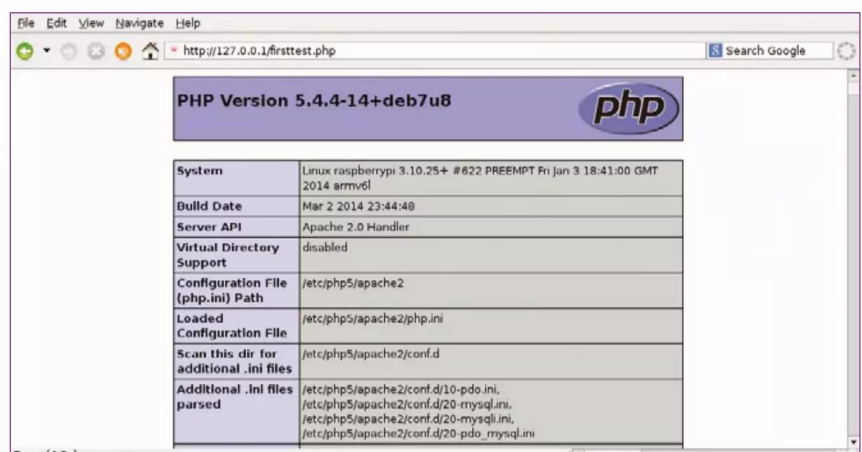


Figure 2: The *firsttest.php* script outputs information about the PHP installation.

create files for the website either below `/var/www` and edit them directly on the Pi in Nano or you can use any desktop PC with a software or development environment of your choice and transfer them to the Pi via SFTP.

In this case, the SFTP user needs appropriate access privileges for the website directory. The least complicated approach is to add user *pi* to the `www-data` user group with

```
sudo usermod -a -G www-data pi
```

and grant all members of the group write permissions for the website directory:

```
sudo chmod 775 /var/www
```

If you want to transfer the files as a user other than *pi*, you need to use

```
sudo adduser <Username> www-data
```

to create the user.

A Leaner Approach

In some cases, your Pi might be too low powered for the feature-rich Apache server (e.g., if you are not just experimenting with small scripts at home but giving several people access to the web server on your LAN); you might notice that your Pi is serving your website very slowly to users' browsers.

In this case, you should use more streamlined web server software that offers fewer,

yet important, options and that leverages the resources of the nano-computer more efficiently. Lighttpd is a good candidate for this scenario.

If you have already used Apache before installing Lighttpd, you will have to remove the installed Apache packages:

```
sudo apt-get remove \n  apache2 php5 \n  libapache2-mod-php5 \n  php5-mysql
```

As you have probably already guessed, you will first need to download the Lighttpd package via the package manager:

```
sudo apt-get install lighttpd
```

Next, install MySQL in the same way as in an Apache environment; however, you need to include PHP as a CGI module in this case. To do so, enter the commands:

```
sudo apt-get install php5-common php5-cgi php5\nsudo apt-get install php5-mysql
```

The order here is important. If you install the `php5-mysql` package first, the package manager will automatically assume you want to use it with Apache and download and set up Apache as an unfulfilled dependency.

Finally, you need to allow the web server to parse PHP scripts explicitly:

```
sudo lighty-enable-mod fastcgi-php
```

After restarting Lighttpd with

```
sudo service lighttpd force-reload
```

the setting is active: The `firsttest.php` page should now return output similar to that of the Apache installation (Figure 2).

Content Management Systems

In the early years of the Internet, it was common to upload data via FTP after each change to a website. Meanwhile, content management systems (CMSs) have become a more efficient vehicle. Not only can you now use a browser on any device to add new content to your website quickly and easily (even with a smartphone), but CMSs also facilitate the entire workflow. For example, if you work with several acquaintances on a club website, each contributor can post new content independently via an editor login, and only the technology manager has advanced access privileges.

The market for CMSs is nearly endless, so you are spoiled for choice. Some beginners rely on WordPress (which was actually not designed as a CMS, but as blogging software). With its clear-cut user interface, WordPress is not a bad choice. Nevertheless, it is better in most cases to rely on other packages, such as Joomla, which has a well-considered and intuitive user interface. True CMSs offer far more options and thereby impose fewer restrictions, should your needs eventually go beyond the capabilities of WordPress.

For a first installation of Joomla, you can download a ZIP archive from the official



Figure 3: phpMyAdmin simplifies the administration of MySQL databases.

website [2] and transfer it to the `/var/www` directory on your Pi using SFTP. Then, you can unzip the archive and delete the now unneeded ZIP file:

```
unzip joomla.zip -d /var/www
rm /var/www/joomla.zip
```

When you type the IP address of your Raspberry Pi in the browser on your PC, Joomla guides you through several dialogs for the initial configuration of the CMS.

If the web configuration fails, the permissions on the directories and files might need to be adjusted for your Joomla version. This can be done on the Pi by changing directory to `/var/www`, running the following console commands,

```
find . -type f -exec chmod 644 {} \;
find . -type d -exec chmod 755 {} \;
```

and restarting the browser configuration.

Future Plans

Now you have a home web server with a generous set of software. If at some point

your website grows beyond the test and experimental stage, you might consider uploading it to the web for production operations. If the site is of medium size, affordable web space should be sufficient. The configurations offered by web hosting companies typically support the most popular content management systems; their FAQ should provide more information.

To take this project one step further, you could lease a VServer or an entire root server. However, you should think twice before signing the agreement, because you need to administer the server completely, and an open web server on the web needs far more hardening against external attacks than your installation on a Pi on your home LAN. If someone hacks your leased server and uses it to send spam, or worse, you could experience some unpleasant consequences.

Moving up to the web requires experience in many areas of server administration and security – for example, configuring a firewall with iptables or “locking” directories with chroot. However, where better to learn and practice all of these skills than on your Raspberry Pi?

INFO

- [1] Web services statistics:
<http://news.netcraft.com/archives/category/web-server-survey/>
- [2] Joomla download:
<http://www.joomla.org/download.html>

RISK-FREE TRIAL!



**Practical.
Technical.
Elegant.**

SUBSCRIBE NOW

**3 ISSUES
+ 3 DVDs
\$3**

shop.linuxnewmedia.com

£3 / €3 / \$3 / \$9 rest of world
Depending on the region you live in.
Terms and conditions: <http://goo.gl/SSSQer>

Constructing a small media center using Kodi

Raspberry Multimedia

With the free Kodi software, you can turn your Raspberry Pi into a media center with a fancy interface – and a whole lot more.

By Tim Schürmann

Modern televisions now have access to the Internet and can play media from a connected hard drive. However, operating these smart TVs is often tricky; they are usually fussy about supported formats, and many models also send information about the user to the manufacturer.

A Raspberry Pi along with the media center software Kodi [1] is a convenient and secure alternative. The duo even brings older televisions into the age of the Internet. In contrast to other set-top boxes or living room PCs, the Raspberry Pi both takes up less space and also is very modest in power consumption.

Couch Potato

Essentially, Kodi provides a simplified interface that can be used to select and play films and music. You can expand the basic structure with numerous other functions using add-ons. Provided you have access to the network, you can tap into Internet radio stations, and access YouTube videos (localized

for a target audience) – assuming you also have an appropriate subscription (see the “I Watch (No) TV” box). You also can customize the appearance of the interface to your own tastes using skins.

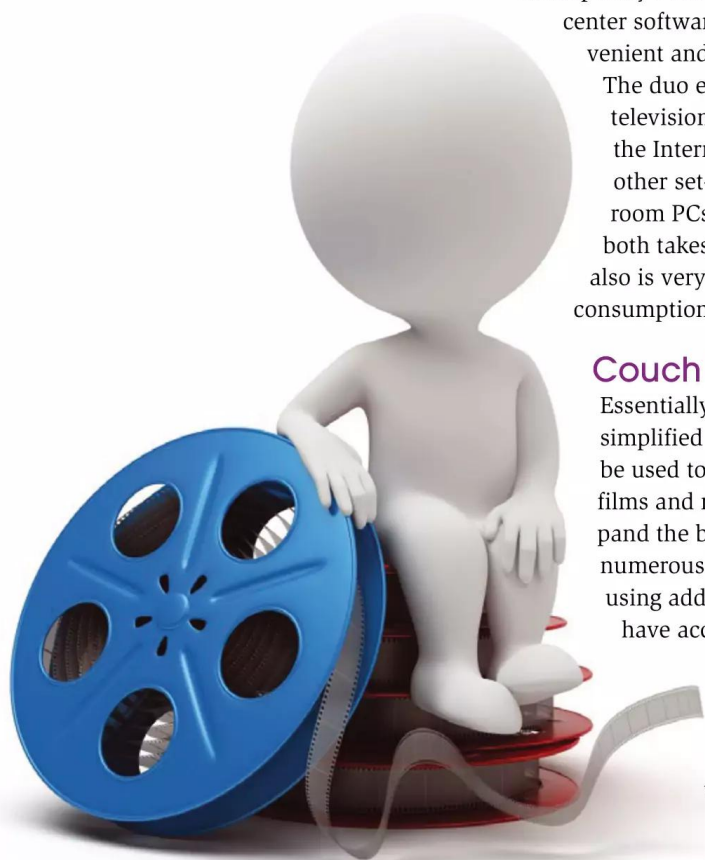
The B version of the media center is running on all Raspberry Pis. HD videos even

I WATCH (NO) TV

The information in this box applies only to those living in Europe, parts of Africa and the Middle East, much of Asia, Australia, and New Zealand. North America uses the ATSC, not the DVB, television standard. You can find out online [2] whether your country uses DVB.

Connecting the television via the Internet to receive terrestrial, satellite, or cable television calls for a few changes. First, you need a DVB receiver connected via USB. The Raspberry Pi operating system must recognize and integrate it, which does not work smoothly for all receivers. You might need to locate a suitable firmware file and store it in the `/lib/firmware` directory.

Once the receiver has been connected, the `dmesg | grep -i dvb` command will reveal both whether this is necessary in your case and the name of your firmware file. Because the DVB sticks consume quite a lot of electricity, you should always connect them to the Raspberry Pi via an active USB hub. Kodi also needs the help of the Tvheadend software to get the television channels from the airwaves. The setup for this would require an article of its own, but you will find the documentation at the Tvheadend wiki [3].



run smoothly on the first-generation (Rev 1) model thanks to the support of the installed graphics chip. However, if available, you should still favor a Rev 2, B+, Rasp Pi 2 Model B, or Rasp Pi 3. Menus establish more quickly with the more powerful hardware, more sophisticated issues run more smoothly, and the software overall reacts much more quickly to inputs. Note that you will need an external drive for playing DVD and Blu-ray discs; however, Kodi will only play copy protection-free Blu-ray media.

Until Autumn 2014, Kodi was still called XBMC (for Xbox Media Center). To avoid a legal dispute with Microsoft, the developers renamed the program with version 14.

All in One

You will initially need a keyboard to set up Kodi, and one is advisable for later use as well: It will make typing music titles much easier and searching the Internet significantly faster.

For living room use, it is worth getting a device with Bluetooth or a compact wireless keyboard with an integrated touchpad. However, for navigating the menus, a remote control or a corresponding Android app are sufficient [4]. You can access the Internet by either dragging a network cable through the living room or getting a WiFi adapter for the Raspberry Pi.

You can equip a Raspberry Pi with Kodi in several ways: The easiest way is to start the installation using the OpenELEC distribution. This is a prepared system that starts Kodi directly. For the installation, you simply need to transfer a file to an SD card and then push it into the Raspberry Pi.

However, the system version offered on the official Raspberry Pi homepage at the time of publication was out of date. You will find a version with the current version of Kodi online [5] in the RaspberryPi Builds. If you are using the first model of the Raspberry Pi, download the file under *ARM11 builds*. Owners of the Raspberry Pi 2 should access *ARMv7 builds*. Get the images that include *Diskimage* in the name

You will receive an archive that you need to unpack. To do this in Windows, for example, you can use the free 7-Zip packer pro-

gram [6]; in Mac OS X and Linux, open a terminal and type the following command in the directory with the file:

```
gunzip -d OpenELEC-RPi.arm-5.0.5.img.gz
```

Then, write the unzipped file on an SD card. The Win32 Disk Imager program is available for Windows users; use the `dd` command [7] on Mac OS X and Linux in the terminal.

The first start takes a while. A wizard appears, and you can navigate between the points using the arrow keys; press Enter to select. In this way, you can choose a language and go to *Next*. Confirm the computer name by clicking *Next*. All detected networks appear in the list. Click *Next* three times, and you will end up in the interface.

Some alternatives to OpenELEC are available, but they are either still under development, such as OSMC [8] and XBian [9], or are orphaned. The once-popular Raspbmc, which is supposed to be merged into OSMC [10] is in the latter group [11].

Play-Ins

The subsequent installation from a system such as Raspbian provides an alternative to an installation from scratch: Simply enter the commands from Listing 1 one at a time into a terminal. Submit each command with the Enter key and wait until the Raspberry Pi has done its job.

The commands integrate the repository provided by Michael Gorven with all the Kodi packages tailored for the Raspberry Pi, collect the corresponding certificate, and finally install the software. Answer the question in the last command by pressing Enter.

Furthermore, the Raspberry Pi graphic card requires at least 96MB of memory from RAM; you should expand to 128MB (Rasp Pi 1) or 256MB (Rasp Pi 2). To do this, call up `sudo raspi-config`, change to *Advanced Options | Memory Split*, enter the appropriate value, and confirm by pressing Enter. Exit the configuration via *Finish* and reboot the computer.

You will find Kodi in the *Start* menu in the *Entertainment* group in the interface. It is possible to start the media center directly

LISTING 1: Installation

```
$ echo "deb http://archive.mene.za.net/raspbian wheezy contrib" | sudo tee -a /etc/apt/sources.list.d/mene.list
$ sudo apt-key adv --keyserver keyserver.ubuntu.com --recv-key 5243CDED
$ sudo apt-get update
$ sudo apt-get install kodi
```

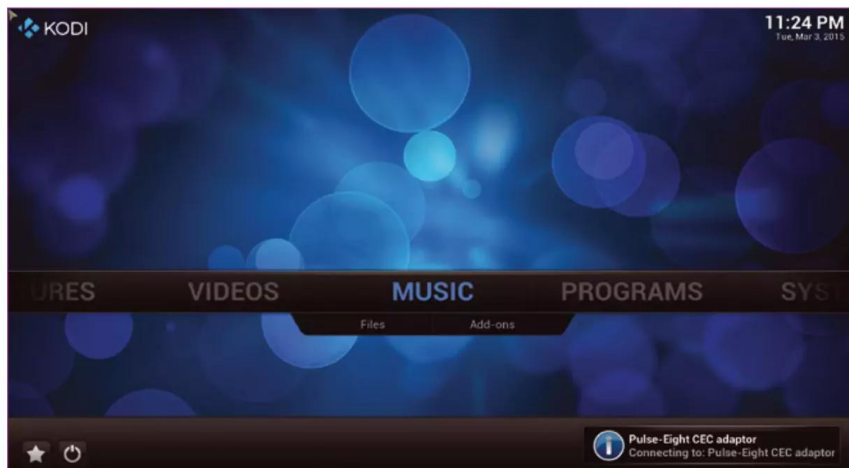


Figure 1: Kodi with the default Confluence skin consists of a type of conveyor belt main menu with the individual menu items embedded in the belt.

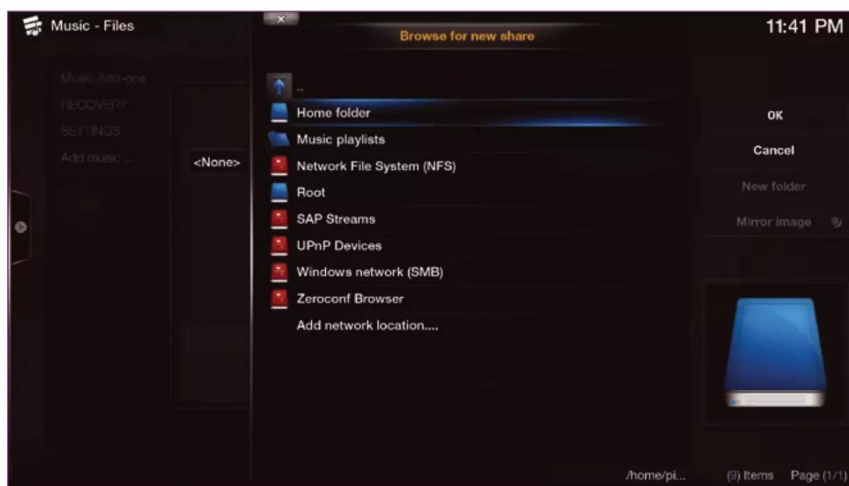


Figure 2: The media center also integrates media located on a network share.

from the command line using the command `Kodi-standalone`. The blue user interface from Figure 1 appears in each instance.

Conveyor Belt

After starting, Kodi displays an application menu in a band across the background image. A series of status messages appear in the bottom right corner. You can select a menu item using your keyboard's arrow keys and pressing Enter to select. Pressing Esc will always return you to the previous menu.

Under certain circumstances, the Kodi user interface will protrude beyond the edge of the screen. Announcements and texts placed here will appear cut off. In this case, you should search for a function called "Over-scan" in the television settings and temporarily disable them.

You can also calibrate or move the image in the Kodi settings. To this end, move a level higher in the menu by pressing Esc and call up *Video calibration* under *System* | *Set-*

tings | *System* | *Video output*. However, in tests, Kodi ignored the overscan settings in the file `/boot/config.txt`.

Matter of Opinion

Return to the main menu by repeatedly pressing Esc. Select a menu item and press Enter to view or play images, videos, or music. When you call up one of the menu items for the first time, Kodi will point out that another menu is hidden on the left edge of the screen. Close the message by clicking OK.

Kodi now wants to know which source it should use to collect the media. If the files are stored on the SD card or an attached hard drive, choose *Add Pictures*, *Add Videos*, or *Add Music* under the appropriate *Files* sub-menu. Select *Browse* in the new window, press Enter, and select the directory with the files (Figure 2). Kodi does not yet show the files contained in the directory, but instead requires that you know in which directory the videos or music are stored.

Once you have found the appropriate folder, confirm by clicking OK and add it as a new source. Register other folders in the same way. Once you are finished, close the window by clicking OK. The sources established this way will now appear in the list. If you choose one there, you will receive a list of all pictures, music, or video files.

If you press the left arrow key with music files, a menu appears with buttons for playback. Alternatively, you can go back to the main menu by pressing Esc, where Kodi also offers buttons. Videos appear in full screen by default. Pressing the Enter button will show control buttons, Esc jumps to the selection list, and pressing Esc again displays the main menu in front of the current video. The buttons are also available here.

Subject to Charge

The Kodi wiki cites the video formats officially supported on the Raspberry Pi as H.264, MPEG-4, Xvid, DivX (but not DivX in the old version 3), VC-1, and MPEG-2 used on DVDs. The MJPEG, VP6, VP8, and Ogg Theora formats remain limited to DVD resolutions. In the tests, however, only the sound could be heard for an Ogg video (ending .ogg), and the same applied to WebM films. However, the media center did not have any problems with an AVI file or the exotic TSCC1 codec. So, some testing can be useful here.

You need a suitable license online [12] to play MPEG-2 and VC-1 videos. However, this only affected the Raspberry Pi 1 in the

tests; Kodi played MPEG-2 videos on a Raspberry Pi 2 without a license. Thus, you should first test whether playing MPEG-2 and VC-1 videos works for you. If not, an MPEG-2 license costs about £2.40 (\$3.60), the license for VC-1 costs around £1.20 (\$1.80) (Figure 3).

However, you can only purchase licenses if your Raspberry Pi was manufactured after August 8, 2012. If this is the case, first call up the following command:

```
$ cat /proc/cpuinfo | grep Serial
```

You receive the device's serial number. You can then use this number to purchase the appropriate licenses online [12]. After making the purchase, you will receive email with a key that is personalized to your Rasp Pi. Store this in the file `/boot/config.txt` using the commands from Listing 2. To do so, replace `0x12345678` with the key you received.

All the functions should work smoothly if you are using the Raspberry Pi 2. You may experience interruptions when accessing the SD card. This happens when tapping into streams from the Internet and Kodi is buffering data or even downloading a whole video. Above all, streams in HD resolution show blocking artifacts shortly after starting.

Upgraded

You can expand the functionality of the software using add-ons. To this end, call up the *System | Settings* menu item and switch to *Add-ons*. Access *Get Add-ons* and select *Kodi Add-on repository*. You will see several categories. In *Weather*, you will find add-ons that display the current weather at a particular location.

The add-ons in the video category are likely to be of particular interest. Here, you will find extensions that allow access to YouTube to be localized for a target audience (Figure 4). To install an add-on, press the Enter key for the corresponding entry in the list and then press *Install*. Kodi will activate the add-on directly.

Operating the retrofitted functions varies from case to case. You can access the enabled video and audio add-ons in the main menu via *Videos* or *Music* and then the menu item *Add-ons*. To watch a YouTube video, jump to *Videos | Add-ons | YouTube*.

If you do not like Kodi's appearance, go to the *System | Settings | Appearance | Skin* menu item. You can choose a different design via *Skin* and download more skins via *Get More*.

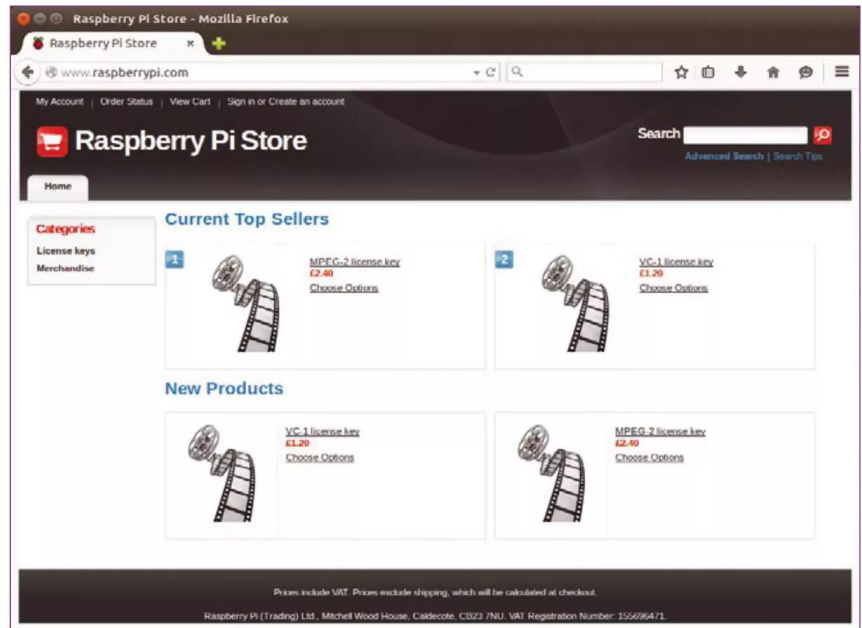


Figure 3: You can purchase licenses for the MPEG-2 and VC-1 codecs in the online shop



Figure 4: You can comfortably watch videos on YouTube from your sofa using the corresponding add-on.

Select a skin from the list by pressing Enter and then install it by selecting the appropriate button. Once Kodi has downloaded the skin from the Internet, you may enable it directly. The Nebula [13] (Figure 5), Back Row [14] (Figure 6), or Eminence [15] (Figure 7) skins show how different Kodi can look.

Kick Start

Exit Kodi by selecting the switch-off icon at the bottom left in the main menu using the arrow keys and then pressing Enter. Kodi starts on bootup on a Rasp Pi with the Kodi distributions OpenELEC and Xbian installed.

LISTING 2: Storing the License Key

```
$ echo "decode_MPG2=0x12345678" | sudo tee -a /boot/config.txt
$ echo "decode_WVC1=0x12345678" | sudo tee -a /boot/config.txt
```

SERVER TRICKS

Kodi Media Center

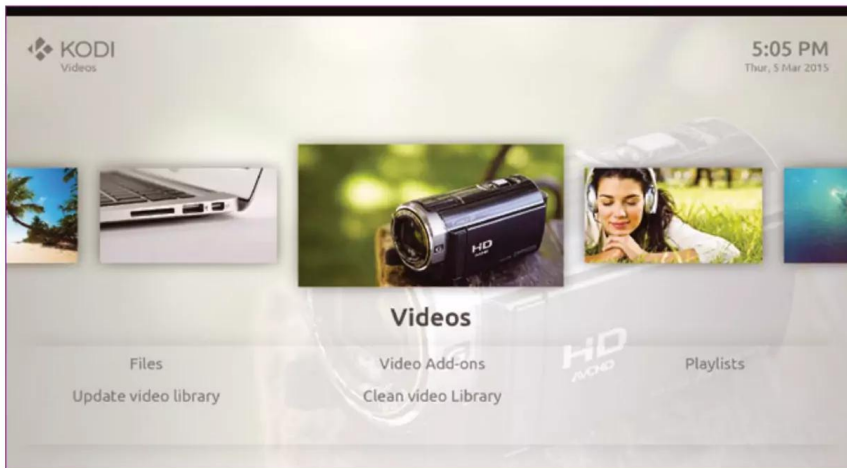


Figure 5: Nebula skin.

If you have installed the software on Raspberry and everything is working as desired, it will offer to set up start on bootup, too.

To do this, type the command `groups kodi` on the command line or in a terminal. You will now receive a list of all user groups where the user is `kodi`. The groups include: *audio, dialout, input, plugdev, tty, and video*.



Figure 6: Back Row skin.

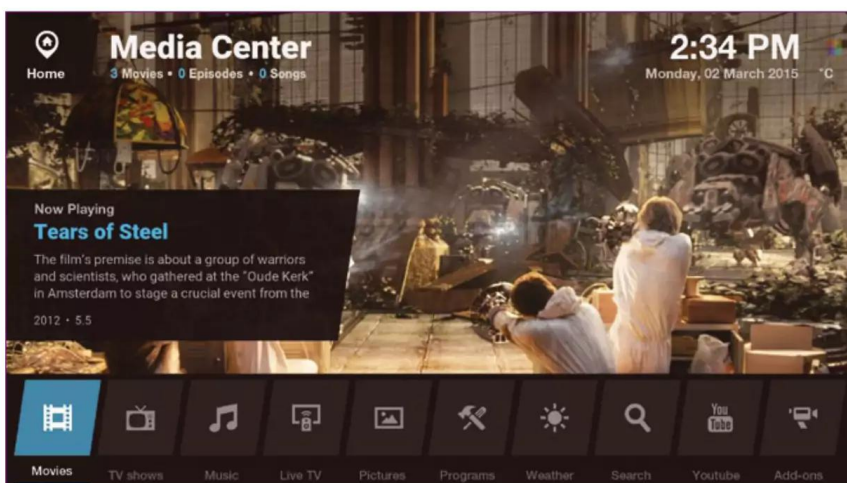


Figure 7: Eminence skin.

If a group is missing, you can add it using the command `sudo addgroup kodi <input>`. Replace `<input>` with the name of the missing group. If `kodi` is in all user groups, invoke the command

```
$ sudo nano /etc/default/kodi
```

to open the `/etc/default/kodi` file in the Nano text editor. Change the line `ENABLED=0` to `ENABLED=1`. Also make sure the line `USER=kodi` exists – this should be the case by default. Save your changes using the key combination `Ctrl + O` followed by `Enter` and then exit the editor by pressing `Ctrl + X`. Restart the system using the command `sudo re-start`.

Conclusions

Kodi on a Raspberry Pi lets you set up a small living room PC and then tap into its archive of movies and music. Navigating the sometimes deeply nested submenus takes a bit of getting used to. Try a different skin if you find the default Confluence skin too confusing. The Kodi wiki [16] can help if you have further questions, but make sure the respective instructions are valid for the Raspberry Pi. ●●●

INFO

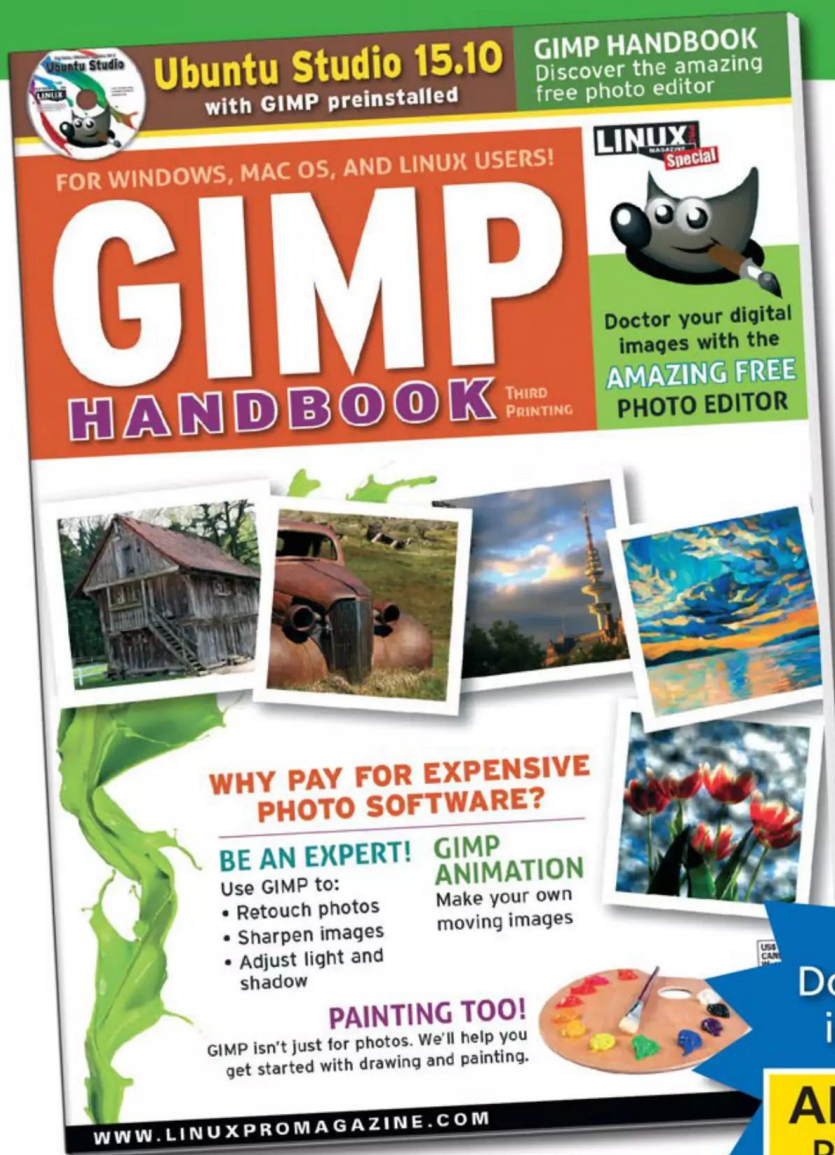
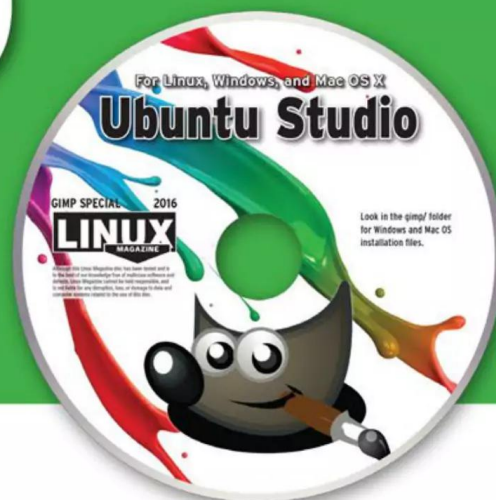
- [1] Kodi: <http://kodi.tv>
- [2] Countries using DVB: http://en.wikipedia.org/wiki/DVB-T#Countries_and_territories_using_DVB-T_and_2For_DVB-T2
- [3] Tvheadend wiki: <https://tvheadend.org/projects/tvheadend/wiki>
- [4] XBMC Android remote: <https://play.google.com/store/apps/details?id=fr.beungoud.xbmcremote&hl=en>
- [5] OpenELEC: <http://openelec.tv/get-openelec>
- [6] 7-Zip: <http://www.7-zip.org>
- [7] Installing operating system images: <http://www.raspberrypi.org/documentation/installation/installing-images>
- [8] OSMC: <https://osmc.tv>
- [9] XBian: <http://www.xbian.org/getxbian>
- [10] Download OSMC for Raspberry Pi: <https://osmc.tv/download/images>
- [11] Raspbmc: <http://www.raspbmc.com>
- [12] Raspberry Pi Store: <http://www.raspberrypi.com>
- [13] Nebula: <http://xbmc-skins.com/skin/nebula>
- [14] Back Row: <http://xbmc-skins.com/skin/back-row>
- [15] Eminence: <http://xbmc-skins.com/skin/eminence>
- [16] Kodi wiki: <http://kodi.wiki>

Shop the Shop

shop.linuxnewmedia.com

GIMP

HANDBOOK



**SURE YOU
KNOW LINUX...**
but do you know **GIMP?**

- Fix your digital photos
- Create animations
- Build posters, signs, and logos

Order now and become an expert in one of the most important and practical open source tools!

GIMP
Doctor your digital images with the
AMAZING FREE PHOTO EDITOR!

Order online:

shop.linuxnewmedia.com/specials



FOR WINDOWS, MAC OS, AND LINUX USERS!

Use SSH to connect remotely to your Raspberry Pi

Secure Connection

We show how to access your Raspberry Pi securely without adding the cost of a keyboard, monitor, or mouse. *By Ruth Suehle*



Many Raspberry Pi projects are “headless,” which means they don’t have an attached monitor, keyboard, or mouse. When it comes time to read the screen. Going without a monitor will keep the cost of your Rasp Pi low.

You can always access the system with Secure Shell, better known as SSH [1].

SSH is simply a secure, encrypted way of remotely logging in to a computer. OpenSSH, a free set of SSH tools developed by the OpenBSD project, is included in Linux distributions. If you’re using a recent version of a Linux distribution meant for the Raspberry Pi, the SSH service is almost certainly already available. The easiest way to find out without attaching a monitor and keyboard is to attempt to SSH to it.

If the instructions don’t work and you suspect SSH is not installed, plug in a monitor and keyboard, then at a shell prompt, enter:

```
$ service ssh status
```

If SSH is working, it returns something like Listing 1 or just *sshd is running*. If SSH is not

available, you can install it easily. On an RPM-based distribution like Fedora (the Fedora Raspberry Pi remix), use:

```
$ sudo yum install \
openssh-server openssh-clients
```

On a Debian-based distribution, such as Raspbian, use:

```
$ sudo apt-get install ssh
```

Once you’ve installed it, you can enter:

```
$ sudo /etc/init.d/ssh start
```

to start the *sshd* daemon (that’s the “d” at the end).

Find your Raspberry Pi’s IP Address

Before you can connect via SSH, you’ll need your Raspberry Pi’s IP address. Of course, you could start with a monitor and keyboard setup before going headless, and in that case, you would just start it up and find the IP address using *ifconfig*. However, I’ll assume

LISTING 1: Checking for SSH

```
'Redirecting to /bin/systemctl status  sshd.service
sshd.service - OpenSSH server daemon
   Loaded: loaded (/usr/lib/systemd/system/sshd.service; enabled)
   Active: active (running) since Wed 2013-02-13 13:06:40 EST; 28min ago
   Process: 273 ExecStartPre=/usr/sbin/sshd-keygen (code=exited, status=0/SUCCESS)
   Main PID: 280 (sshd)
   CGroup: name=systemd:/system/sshd.service
           280 /usr/sbin/sshd -D
```


the dog ate your HDMI cable and you're going headless from the start.

Pidora takes advantage of two tools called `ip-read` and `ip-flash` to make running in headless mode easier. With some simple configuration additions on your SD card, your Raspberry Pi will announce its assigned IP address (using `eSpeak`) through attached speakers two minutes after it is powered on. Thirty seconds later, it will also flash the IP address on the Pi's OK/ACT LED [2].

To begin, plug in the SD card loaded with the Raspberry Pi operating system of your choice. Next connect the Pi via an Ethernet cable to the network (see also the "Laptop SSH" box); then, plug in the power to start Pi. Remember, just because you're not watching the Raspberry Pi boot up on a screen doesn't mean it doesn't take the same amount of time. Give it a few minutes to boot up.

Next, you can use your router's internal IP address to see where the Pi is connected. Your router's default internal IP address is listed in its manual and quite possibly on a sticker somewhere on the hardware. If you haven't intentionally changed it, many commonly used routers default to 192.168.0.1, 192.168.1.1, or 192.168.2.1. If none of those sound familiar, check for yourself with

```
$ route -n
```

which will output something like Table 1.

Although you'll see several lines, you want the one with the UG flags, which indicate the status of the route (up, U, or active) and the gateway for the route (G). The numbers under "Gateway" are what you're looking for.

Most routers will send you to a control panel if you enter that address in a web browser. Log in if required and find the list of devices connected to your network. Look for one named *raspi* or *raspberrypi* and note the IP address.

SSH to the Raspberry Pi

Once you know your Pi's IP address, you can connect to it by typing

```
$ ssh <username>@<host>
```

where *host* is the IP address you found for your Raspberry Pi. The *username* is either an account that you've set up on the machine, or, if this is a fresh installation, the default account for the operating system you've chosen. Table 2 shows a few of the default logins for commonly used Raspberry Pi operating systems.

If you're using another distribution, check its documentation for the default login and password combination. You can also consult the table online [4].

For the rest of these examples, I'll use 192.168.0.115 as the Raspberry Pi's IP address and *fedora* as the username. Change these as appropriate for your own IP and Raspberry Pi user.

When you log in to a machine for the first time, it checks your `.rhosts` file and creates a random key; then, it checks this key to be sure you've connected to the host you intended. The first time you connect, you'll see something like:

```
$ ssh fedora@192.168.0.115

The authenticity of host
'192.168.0.115 (192.168.0.115)'
can't be established.
RSA key fingerprint is
83:d7:be:fe:5e:91:98:90:ff:eb:87:0b:
88:d2:e9:e9.
Are you sure you want to continue
connecting (yes/no)?
```

When you enter *yes*, you'll see:

```
Warning: Permanently added
'192.168.0.115' (RSA) to the list
of known hosts.
```

If you've logged in on a system that starts you as root, the first thing you'll want to do is create a user account to use rather than doing everything as root.

The second thing to do is disable SSH by root. To do this, type

```
$ vi /etc/ssh/sshd_config
```

and uncomment the line that says `PermitRootLogin yes`, then change the *yes* to *no*.

If you would prefer to let only specific users SSH to your Raspberry Pi, you can list

TABLE 1: Kernel IP Routing Table

Destination	Gateway	Genmask	Flags	Metric	Ref	Use Iface
0.0.0.0	192.168.0.1	0.0.0.0	UG	0	0	0wlan0

TABLE 2: Default Logins

Distribution	Default login::password
Arch Linux ARM	root::root
OpenELEC	root::openelec
Pidora	root::raspberrypi
Raspbian wheezy (and derivatives like Raspbmc)	pi::raspberry

LAPTOP SSH

If you're only interested in SSH-ing directly between your laptop or desktop and the Rasp Pi, instead of through an established network, you'll find an in-depth guide with instructions for Windows, Mac, and Linux machines online [3].

LISTING 2: Generating a Key Pair

```
Generating public/private rsa
key pair.

Enter file in which to save
the key (/home/fedora/.ssh/
id_rsa): <press Enter>

Created directory '/home/
fedora/.ssh'.

Enter passphrase (empty for
no passphrase): <your
chosen passphrase>

Enter same passphrase again:
<your chosen passphrase>

Your identification has been
saved in /home/fedora/.ssh/
id_rsa.

Your public key has been
saved in /home/fedora/.ssh/
id_rsa.pub.

The key fingerprint is:
3f:9e:8d:65:1b:0f:45:f7:22:a7
:69:b3:c4:4d:8d:de
fedora@raspi.local

The key's randomart image is:
+--[ RSA 2048 ]-----+
|           |
|           |
|           . .|
|           . +.|
|      S   . * o|
|           . . X o|
|           o % o E|
|           . X B |
|           + + . |
+-----+
```

them in the `sshd_config` file as well. At the end of the file, add `AllowUsers` with the list of usernames that should be allowed to connect via SSH:

```
AllowUsers hannah ian
```

To make the change take effect, restart the service:

```
$ service sshd restart
```

To leave the SSH session and return to your local computer's shell prompt, type the `exit` command.

Setting Up your Distribution

If you're using Pidora, it would usually run a shell script the first time you boot up that has you set up the root password and a user; however, this script runs only if input devices are detected. If Pidora is booted headless, the system will configure the Ethernet interface via IPv4 DHCP, allowing you to access the system through SSH. You can set up any other preferences once you've connected.

When you first boot a headless Raspberry Pi on Raspbian wheezy, you'll see:

```
NOTICE: this Raspberry Pi has not
been fully configured.
Please run 'sudo raspi-config'
```

If you had logged in from the Pi itself, the `raspi-config` tool would have launched automatically. You can now run it from the SSH session to set up the Raspbian installation as if you were directly on the machine.

Generate a Public/Private Key Pair

In public key cryptography, you create a public key to encrypt data and a private key to decrypt it. It might make more sense to think of the public key not as a key but as a lock that the private key opens. Thus, you can freely send your public key to other systems or over email. For better security, you should protect your private key with a passphrase when you create it. All of this is easy to do, and you can create separate keys if you have accounts for multiple systems. To get started, create a key pair:

```
$ ssh-keygen
```

You will be prompted for a directory to which you save the key and a passphrase.

The passphrase can be just that – a set of words with spaces and so on – so make it a good one and make it something you can remember. The information you enter is shown in angle brackets in Listing 2.

Next, copy the public key that you created onto the Raspberry Pi:

```
$ ssh-copy-id fedora@192.168.0.115
fedora@192.168.0.115's password: ?
<enter your password>
```

Try logging in to the machine, with `ssh 'fedora@192.168.0.115'` (use your IP address) and check in the

```
~/.ssh/authorized_keys
```

file to make sure you haven't added extra keys that you weren't expecting.

This procedure will make connecting to your Pi one step faster because you won't have to enter a password each time.

Create Users

To add new users who can SSH to the Raspberry Pi, you simply add users to the system,

```
$ sudo useradd newuser
$ sudo passwd newuser
```

and then create keypairs for these new users.

If you're using Pidora and are more comfortable creating users through the GUI, follow the instructions in the section "Use Graphical Programs over SSH"; then, start the `system-config-users` tool.

Transferring Files to and from the Raspberry Pi

The two ways you're mostly likely to transfer files to your Raspberry Pi through the command line are via SCP (secure copy) and SFTP (secure file transfer protocol). The first tool, `scp`, is most useful when you just want to copy a small number of files and then exit. If you want an open connection from which you can run a series of interactions, you should use `sftp` instead. You don't have to SSH into the Raspberry Pi first to use `scp`. The command format is:

```
$ scp <filename> <username>@<hostname>:<b>?
</path/to/destination/>
```

To copy the `opensource.odt` file from your current directory (enter a full path if you are

not in the file's directory) to the web directory of your Raspberry Pi with IP address 192.168.0.115 and user *fedora*, type the command:

```
$ scp opensource.odt 2
fedora@192.168.0.115:/var/www/html/
```

You can copy an entire directory (e.g., *foss*) by using the `-r` flag, which tells `scp` to descend through the directory recursively, copying its contents.

```
$ scp -r foss fedora@192.168.0.115:.
```

Note in the previous example, that rather than specifying a directory at the end of the line, you enter only a period. This step will copy the file or directory into the *fedora* user's home directory.

For a secure version of FTP, you can use SFTP, which uses the SSH protocol for file transfer. With this tool, you connect much as you would if you were using `ssh`, except you use `sftp`:

```
$ sftp fedora@192.168.0.115
```

You now have an SFTP session open through which you can use a variety of commands. Table 3 shows a sample SFTP session demonstrating the most common commands you'll

need, which you'll see is similar to working in the shell.

Using Graphical Programs

When you use the graphical interface of your Linux machine, the interaction happens through the X Window System, more commonly referred to simply as X. If you want to start a GUI application over SSH, you need to add the `-X` flag when you connect to enable X forwarding, which "forwards" graphical programs through your remote connection:

```
$ ssh -X fedora@192.168.0.115
```

Note that you use a capital X. A few applications just won't run over X forwarding. If, however, you think it should be working (e.g., you've run this application through X forwarding on another machine, but it isn't working on the Raspberry Pi), check that `X11Forwarding` is set to `yes` in `/etc/ssh/sshd_config`. You should then be able to launch graphical programs on the Pi from your SSH session to use on your remote screen. To test this, type `midori` at the prompt to launch the lightweight browser included with most Raspberry Pi Linux distributions.

With these tools, you're ready to tackle just about any task without adding the size or cost of a monitor and keyboard to your Raspberry Pi. ●●●

THE AUTHOR

Ruth Suehle works in Red Hat's Open Source and Standards group, which aims to help upstream open source software communities. She also leads the Fedora Project's marketing team and is co-author of the upcoming *Raspberry Pi Hacks* book from O'Reilly. Previously an editor for Red Hat Magazine, she now leads discussions about open source principles as a moderator at opensource.com. Ruth is also a core contributor to GeekMom.com, where she covers the adventures of motherhood alongside technology and sci-fi.

INFO

- [1] SSH: http://en.wikipedia.org/wiki/Secure_Shell
- [2] Pidora wiki: <http://zenit.senecac.on.ca/wiki/index.php/Pidora-Headless-Mode>
- [3] SSH from laptop to Pi: <https://pihw.wordpress.com/guides/direct-network-connection/>
- [4] Rasp Pi distributions: http://elinux.org/RPI_Distributions

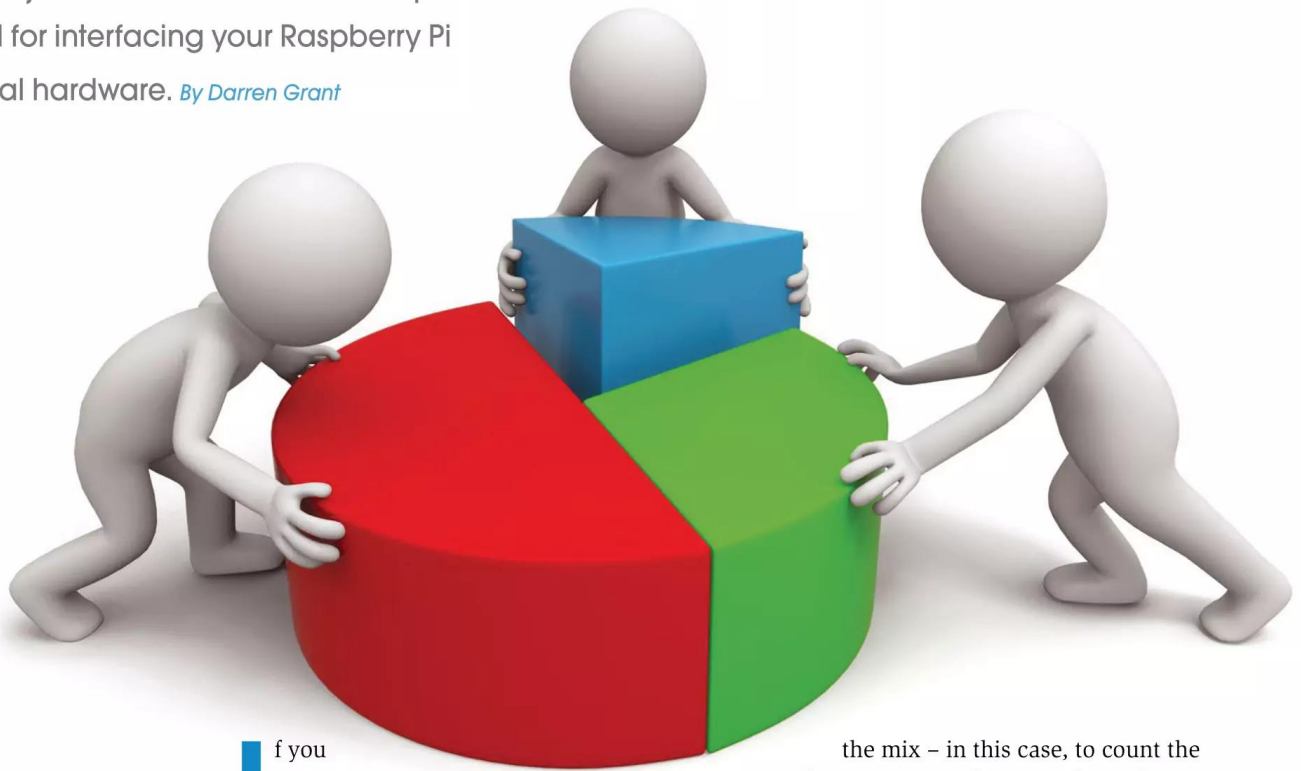
TABLE 3: Example SFTP Session

Command	Action
[rsuehle@localhost ~]\$ sftp fedora@192.168.0.115 fedora@192.168.0.115's password: Connected to 192.168.0.115	Connect to the Raspberry Pi with SFTP
sftp> pwd Remote working directory: /home/fedora	Show the directory you're currently in on the Raspberry Pi
sftp> lpwd Local working directory: /home/rsuehle	Show the directory you're currently in on your local machine
sftp> ls Desktop Documents Downloads Music Pictures Public	List the contents of the directory you're currently in on the Raspberry Pi
sftp> cd Documents	Change to a new directory (in this example, Documents)
sftp> ll fedora.png opensource.odt	List the contents of the directory you're currently in on your local machine
sftp> put fedora.png Uploading fedora.png to /home/fedora/Documents/fedora.png fedora.png 100% 556KB 556.1KB/s 00:01	Transfer the specified file (in this example, <i>fedora.png</i>) to the currently used directory on the Raspberry Pi
sftp> get opensource.odt Fetching /home/fedora/Documents/opensource.odt to opensource.odt /home/fedora/Documents/opensource.odt 100% 556KB 556.1KB/s 00:00	Transfer the specified file (in this example, <i>opensource.odt</i>) from the currently used directory on the Raspberry Pi to your local machine
sftp> exit	Exit SFTP

An interfacing project for beginners

First Slice

This easy project introduces some concepts you'll need for interfacing your Raspberry Pi with external hardware. *By Darren Grant*



If you want a change from playing games or browsing the Internet, what else can you do with a computer? In this article, I will share some of the fascinating possibilities of using a computer for control. You can get the computer to switch things on and off, respond to events, and take a variety of measurements. This process is known as *interfacing*. One of the most powerful features of the Raspberry Pi is its ability to interface with other kinds of devices. Whether you are using your Rasp Pi to control a robot, respond to a temperature sensor, or water your plants through a remote connection, you will need to understand the principles of interfacing to put your Pi in control.

This article describes a very simple project to help you get started with interfacing your Raspberry Pi with other devices. I'll show you how to set up a button as an input device for your Raspberry Pi system and how to make the Rasp Pi respond to a button press. Then, I'll work a simple Python script into

the mix – in this case, to count the button presses. Once you learn these concepts, you can extend them to your own scripts to address more complex scenarios. The Raspberry Pi is the perfect tool to let you start experimenting with interfacing. The low cost of the Pi means you don't need to worry about damage to an expensive computer, and the Rasp Pi comes already equipped with the necessary ports.

The Ins and Outs

Computers are all about ins and outs. Input something on the keyboard or click with a mouse, and you get an image out on the screen or sound out of the speakers. Interfacing is simply finding other ways of getting information in and out of the computer.

The first thing to understand is that the basic building blocks of a computer are made up of millions of switches called transistors. Computers see everything as either on or off, so when you click the mouse button, the switch is on and when you let go, it is off. The computer has no idea how hard the but-

This article is sponsored by Tandy.

This article originally appeared in The MagPi magazine: www.themagpi.com



Lead Image © Liu Ming, 123RF.com

ton has been pressed. This is known as a *digital binary system*: When a switch is on, it is represented by a 1 (one), and when it is off, it is represented by a 0 (zero). Signals that are variable, such as those measuring temperature, are called analog. Because analog signals are more complicated than just a simple on or off state, extra equipment is needed so the computer can read them.

For now I will stick with a simple, digital interface. The Raspberry Pi has a set of connections called the GPIO (General Purpose Input Output) ports. *General Purpose* means these connections have not been assigned a specific purpose. GPIO connections can be used for whatever purpose you like. Each connection can be used as an *input*, where the computer is waiting for an event to happen, such as a switch being pressed, or an *output*, where the computer is sending a signal out, for example, to switch a light or buzzer on and off.

The GPIO ports on the Raspberry Pi are low powered, so an expansion board is recommended to provide higher-powered signals that can be used for controlling things like lamps, buzzers, and motors. A GPIO expansion board also provides some protection to the Raspberry Pi, just in case something is connected incorrectly.

Don't worry if you don't have a GPIO expansion board at this stage and you are raring to get going with your first experiment. Providing you take care, you can start with a simple circuit using a switch and an LED (Light Emitting Diode).

IMPORTANT: *Before connecting anything to the Raspberry Pi, be aware that incorrect connections could cause damage. Please take care.*

Switch Types

An electrical switch is used to make or break an electrical circuit. Switches are binary devices: They are either completely on (closed) or completely off (open), making an ideal starting point for interfacing experiments. Many different types of switches exist, with the simplest type being one in which two electrical conductors are brought in contact with each other by manually moving a mechanism.

Switches come in two basic types: latching switches and momentary types. A latching switch stays open or closed when activated, for example, a light switch; when you switch the light on or off, the switch stays in the same position until you switch it again. A momentary switch, on the other hand, will

only operate while pressure is being applied and will return to its previous state when released.

Momentary switches can be found on a game controller, for example, where the control only responds while the button is being pressed. A momentary switch wouldn't make a very good light switch, as you wouldn't want to have to keep the button pressed to keep the light on. The electrical symbol for a simple momentary push switch is shown in Figure 1.

For this experiment, I will use a small momentary push button switch, known as a normally open tactile switch (Figure 2).

LEDs

LEDs (see Figure 3) can be found everywhere; they are commonly used as indicator lights on most electronic equipment, such as the standby light on your TV. Because of the small amount of power needed to make an LED light up, I can safely use them in this experiment without damaging the Raspberry Pi. The symbol for an LED is shown in Figure 4.

To avoid soldering wires together, a breadboard and a selection of jumper wires are highly recommended. A breadboard is a board that serves as a base for electronic connections (see Figure 5); they make it very easy to connect things together. The grid reference numbers are laid out on a small 270-point Tandy breadboard. A bigger board will work just as well, but the reference numbers might not match, so you will need to modify this description if your board has a slightly different grid layout.

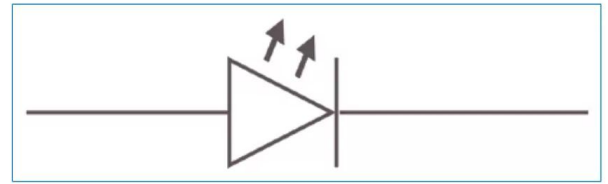


Figure 1: Electrical symbol for a simple momentary push switch.



Figure 2: A momentary push button or "open tactile" switch.



Figure 3: Light Emitting Diodes (LEDs) are a common component for electronic projects and products.

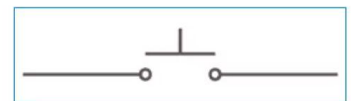


Figure 4: Electrical symbol for an LED.

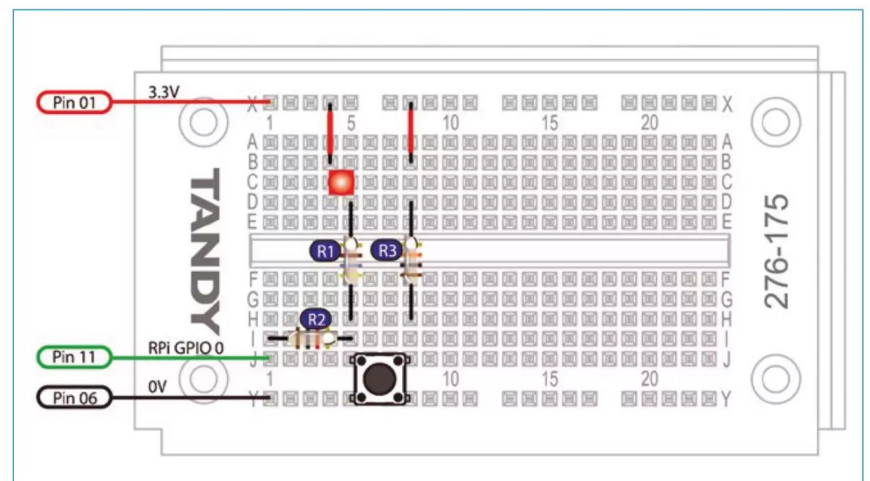


Figure 5: Place the circuit components on a breadboard for easy assembly.

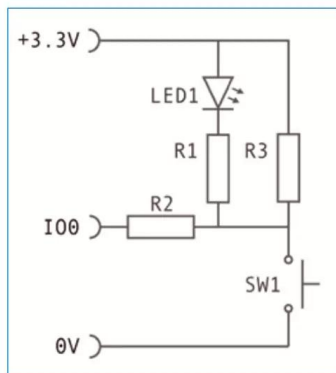


Figure 6: Schematic diagram of the circuit shown in Figure 5.

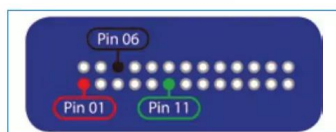


Figure 7: Pin positions on the 26-pin Raspberry Pi corresponding to the pins shown in Figure 5.

Place the components on the breadboard (using Figure 5 as a guide). The configuration shown in Figure 5 matches the schematic in Figure 6. See Table 1 for a summary of breadboard positions for the layout.

Figure 7 shows the corresponding pin positions on the Raspberry Pi. If your tactile switch has inward-pointing legs, like the one shown in Figure 2, use a small pair of pliers to straighten them. The switch needs to be inserted into the breadboard so that the legs are on the left and right edges. It is also important that the LED is connected in the correct way; the positive terminal called the anode is the longer leg and should be connected to point C4.

Once all the components are in place, connect the circuit to the Raspberry Pi using three female-to-male jumper wires, taking extra care to connect the correct terminals (Figure 7). The Raspberry Pi GPIO pins in Figure 7 are shown when looking at the board with the GPIO header in the top left corner.

When you have connected everything together, you can power up the Raspberry Pi, and try pressing the button – the LED should light up. If it doesn't work, check that all the components and wires are connected correctly, paying particular attention to the LED. If the LED stays on without pressing the button, then the switch is probably connected incorrectly and will need to be turned 90 degrees.

Circuit Description

Because the Raspberry Pi GPIO port is not protected with a buffer, I have included a 1k-ohm resistor (R2) between the switch and the GPIO pin as a precaution to prevent damage if the GPIO port is accidentally set as an output. The tactile switch is connected to the 0V or ground line so that when it is pressed, the circuit is completed between the GPIO pin and 0V.

The LED, along with a 470-ohm current-limiting resistor (R1), is connected to the 3.3V positive rail. Pressing the switch com-

pletes the circuit between the LED and 0V, causing it to light up; this provides a visual indication when the switch is pressed. Finally, I need a 10k-ohm pull-up resistor (R3).

Digital Logic States

As mentioned before, computers see the world as a series of zeros and ones. The Raspberry Pi uses 3.3V CMOS logic, meaning that a binary 1 is created by applying a voltage of 3.3V to an input. When the voltage is present, it is referred to as a logic HIGH state. Alternatively when the GPIO pin is connected to 0V, a binary 0 or logic LOW state is created. In the switch example, pressing the switch creates a LOW state by connecting the I/O port to 0V.

Digital logic, in fact, has three possible states, known as tri-state logic. I have already discussed the HIGH (+3.3V) and LOW (0V) states, but the system also has a third state called *floating*. Basically, floating means the state is not clearly defined, so attempting to determine the status of a connection that is floating might result in unpredictable results. You might think that if the pin is not connected to a voltage, the voltage must be low: the problem is you cannot guarantee that you will get a reliable LOW signal by simply relying on the absence of a voltage on the pin.

To illustrate the problem, imagine that you hold a piece of ribbon between two gate posts; the two gate posts would be connected by the ribbon. If you let one end of the ribbon go and allow it to fall to the ground, the two posts are no longer connected. But what happens when the wind blows? The ribbon will flap around where sometimes it will be in contact with the ground and other times it will touch or even become entangled with the other post. To prevent the ribbon from moving in the wind, you tie it either to the ground or the post so it won't move unless you want it to. You need to do the same thing with a logic circuit.

To avoid unpredictable behavior, you can tie the connection to either 3.3V or 0V using what is known as a pull-up or pull-down resistor, which effectively creates a default state. This experiment uses a pull-up resistor so the I/O port will always be connected to +3.3V, making it the default state. The state of the I/O pin will always appear as HIGH until you press the switch that will change it to LOW.

Software

Once I have completed the circuit construction, I now need to get the Raspberry Pi to

TABLE 1: Breadboard Positions for Circuit in Figure 5.

Components (see Fig. 6)	Description	Breadboard Positions
SW1	Tactile switch	J5, J8, Y5, Y8
R3	10k-ohm resistor	D8, H8
R2	1k-ohm resistor	I1, I5
R1	470-ohm resistor	D5, H5
LED1	Red LED	C4, C5
Wire link		X4, B4
Wire link		X8, B8

SHOPPING LIST

COMPONENTS: 1x 3mm red LED (standard brightness), 1x 470-ohm resistor, 1x 1k-ohm resistor, 1x 10k-ohm resistor, 1x miniature PCB tactile switch

ACCESSORIES: 1x breadboard, 3x male-female jumper wires, 1x selection of short jumper wires

TOOLS: If you don't have them already, then a set of small long-nosed pliers for bending component leads and wire cutters will make construction easier.

recognize the switch. To do this, I need to create a software program that will instruct the Pi to monitor the switch status.

I will be using Python to write the program. This discussion assumes you are using the standard Debian “squeeze” operating system from the Raspberry Pi website – and that you have some familiarity with Python. If you are not familiar with Python, consult one of the many informative tutorials online or in print.

The first step is to install a Python package that adds the capability to control the GPIO pins on the Raspberry Pi. Download the RPi.GPIO Python package [1]. At the time of this writing, the latest version of RPi.GPIO is RPi.GPIO-0.5.11. If you find a later version at a later time, you might need to modify these instructions accordingly.

The next step is to open a terminal window to install the package at the command prompt. The file is compressed and archived; to install it, you can decompress and unarchive the file by typing the following commands:

```
gunzip RPi.GPIO-0.5.11.tar.gz
tar -xvf RPi.GPIO-0.5.11.tar
```

Go to the new RPi.GPIO-0.5.11 directory and run the setup script; type the following commands to complete the installation.

```
cd RPi.GPIO-0.5.11
sudo python setup.py install
```

You might be asked for your password when running the install script. If you have not changed the password, the default is *raspberrypi*.

Once the package installation is complete, you can write a program to monitor the switch and write something to the screen each time the switch is pressed.

Before you start programming, don't forget to type the following

```
cd ..
```

to return to the home directory.

Python Program

Create a new text file named `mybutton.py` and enter the program shown in Listing 1. Listing1 starts by including the `time` and `RPi.GPIO` packages in the program, so I can make use of their functions by using `import`. Next, the program configures GPIO pin 11 on the Raspberry Pi as an input, so it can detect the switch. By using `while True`, I create a never-ending loop; everything below this will be repeated until I choose to stop it.

The program keeps checking the status of pin 11. The value will always be `True` (High) while the button is not being pressed. As soon as the button is pressed, the GPIO pin goes low and the result will be `False`, at which point the program prints out the word *giggle* on the screen.

Because the program runs much more quickly than I can press and release the button, I add a small 0.2-second delay before checking the button status again so that the screen does not fill up with giggles.

To start the program, type

```
sudo python mybutton.py
```

into the terminal window. Pressing the Ctrl + C keys stops the program.

Counting

Try the Python program in Listing 2 to count the number of times the button is pressed.

Conclusion

This easy hardware project should help you get started interfacing the Raspberry Pi. I covered how to make the computer respond to a switch circuit and described a simple program for counting button presses. Once you have mastered the basics, you can dream up your own ways to improve the

program. For example, you could make a timer that will start and stop counting when the button is pressed. You also could use the button to trigger some other process managed by the Raspberry Pi system. Send your programs and questions to in.control@the-magpi.com or post your ideas to the official Raspberry Pi forum.

INFO

[1] RPi.GPIO: <http://pypi.python.org/pypi/RPi.GPIO>

LISTING 1: mybutton.py

```
01 #!/usr/bin/python
02 import time
03 import RPi.GPIO as GPIO
04 GPIO.setup(11, GPIO.IN)
05 while True:
06     mybutton = GPIO.input(11)
07     if mybutton == False:
08         print "giggle"
09     time.sleep(.2)
```


LISTING 2: Counting the Button Presses

```
01 #!/usr/bin/python
02 import time
03 import RPi.GPIO as GPIO
04 GPIO.setup(11, GPIO.IN)
05 count = 0
06 while True:
07     mybutton = GPIO.input(11)
08     if mybutton == False:
09         count = count+1
10     print "count", count
11     time.sleep(.2)
```

Learn about your Pi – and test your fine motor skills – with this easy game project

Steady Hands

This simple game will bring hours of fun, and it gives you an up-close look at some important hardware hacking concepts. *By Mike Cook*



You don't have to get too complicated to get a good deal of fun from a project that interfaces with the GPIO. Electrically, this project is just about as simple as you can get; however, it has a very good fun-to-technology ratio.

Steady hands is a very old game, but the Raspberry Pi can give it a new twist. The idea is that you have to guide a wire loop along a bent wire without letting the loop and the bent wire touch. You can make this game as difficult or as easy as you like by putting more or fewer bends in the wire or by making the loop smaller. You can make the apparatus from a metal coat hanger or a length of tinned copper wire and a block of wood.

In this project, I show you how to use the Raspberry Pi computer to time and score the Steady Hands game. The bent wire and wire loop are connected to the Pi through the GPIO board (see Figure 1). The Pi also monitors resting points (stops) for the loop at the beginning and end of the bent wire

(points A and B) to mark the beginning and end of the game.

A simple Python program running on the Raspberry Pi monitors when the wire loop is lifted from the starting rest point (point A) and measures the time it takes until the loop touches down at the ending rest point (point B), meanwhile keeping a count of the number of times the wire loop touches the bent wire.

The program calculates a score for the player on the basis of the total time for the run, so the player with the lowest score is the winner (see the box titled “Scoring”). However, just as in horse jumping, penalty points matter, and the best score is the lowest time with no penalty points.

Construction

1. Drill holes for the bent coat hanger in the wooden block that are just smaller than the diameter of the wire so that when the wire is inserted, it stays upright. Make sure to space the two holes far enough apart to accommodate your design.



This article originally appeared in The MagPi magazine: www.themagpi.com

Lead image © dny3d, 123RF.com

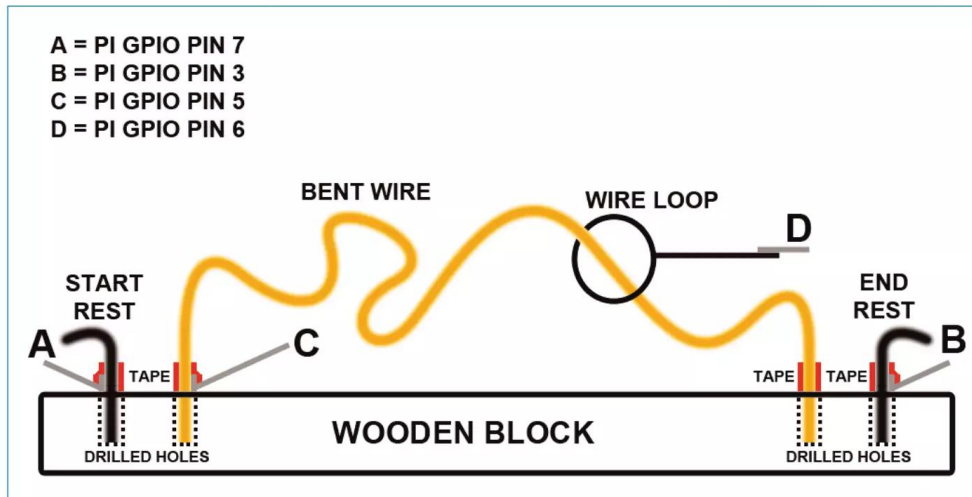


Figure 1: Assembling the Steady Hands game.

TABLE 1: Pinout

Steady Hands	Pin	GPIO (R1)	GPIO (R2)
Wire Loop	6	Ground	Ground
Bent Wire	5	1	3
Start Rest	7	4	4
End Rest	3	0	2

2. Make the wire loop and solder a length of electrical wire to it. You might want to cover the end you hold with insulation tape or, better, self-bonding tape.
3. Put the bent wire through the wire loop and then into the wooden block.
4. Solder a length of stranded wire (insulated) to one end of the bent wire.
5. Drill two holes on either side of the bent wire, as shown in Figure 1, for the rests.
6. Put two short lengths of coat hanger wire into these holes to act as the rests. These rests will detect when the game starts and when the wire loop reaches the end. Bend

them so the loop will rest against them without touching the bent wire and shorting out.

7. Solder a length of normal wire (insulated) to each end stop.
8. On each end of the bent

wire and on both rests, tape the wire where it connects with the block of wood using electrical tape 4cm high.

Table 1 shows how to connect each part of the Steady Hands apparatus with the Raspberry Pi GPIO. Raspberry Pi pin positions are shown in Figure 2. Note that this article is based on the Rev 1 board. (See the box titled “Other Rasp Pi Boards” for more information on how to adapt this procedure to a newer board.)

Basically, the configuration calls for three signal wires and a ground. Using GPIO 0 and 1 means that a pull-up resistor is already

SCORING

When the loop is first lifted off the start rest, the program stores the current state of the time clock in a variable. Each time the loop is detected touching the bent wire, a penalty total is incremented and there is a 70ms delay. So every 70ms time period that contact is made with the bent wire, the penalty variable is incremented.

When the loop makes contact with the end rest, the total time for the run is calculated by subtracting the time the loop left the start rest from the current time. To this time is added the penalty total multiplied by 0.07; this represents the amount of time the loop spent touching the wire. This time is not accounted for in the time for the run because the program is in sleep mode. Remember, like any race, the lowest score wins.

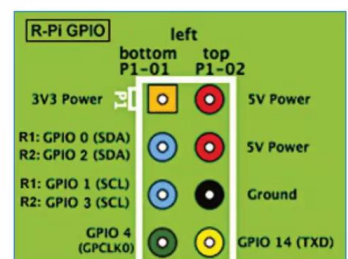


Figure 3: Pin changes from Rev1 to Rev2. (eLinux.com, CC BY-SA 3.0)

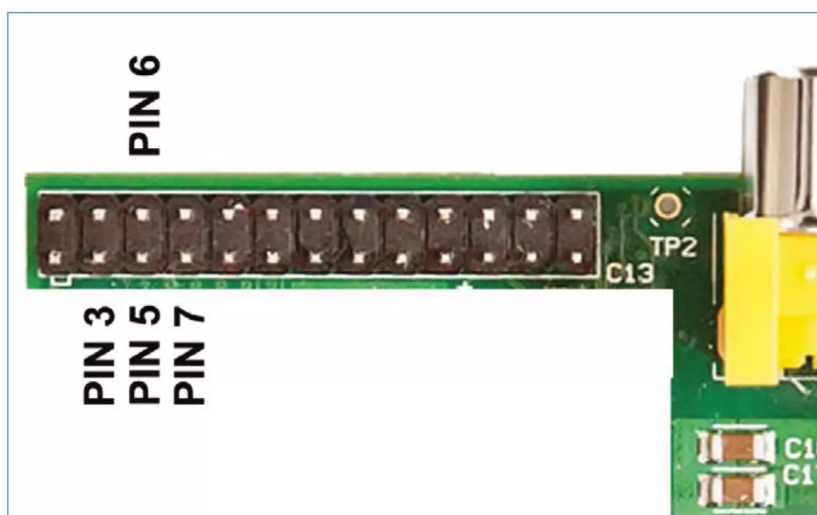


Figure 2: Pin positions on a Raspberry Pi 1 Model B Rev 1 board. For the Rev 2 board, the B+, or the Raspberry Pi 2, see the box “Other Rasp Pi Boards.”

OTHER RASP PI BOARDS

This article was written for the original (Rev 1) Raspberry Pi, now referred to as the Rasp Pi 1. Since then, new boards – the Model B Rev 2, Pi 1 Model B+, and Pi 2 Model B – have been issued. The Rev 1 to Rev 2 change involved three GPIO pins on the connector (Figure 3). Unfortunately, two of these were pins used by this project. If you have a Rev 2 or more recent board, you need to change the values assigned to the end_rest and wire variables (Listing 1). Also, uncomment (remove the #) the two GPIO.setup lines (lines 18 and 19). Comments in the code tell you to do this.

A Rev 2 board can be identified by the two rows of four empty holes at the left end of the GPIO connector; the Model B+, Pi 2, and Pi 3 boards have 40-pin GPIOs.

connected on the Pi, just leaving GPIO 4 to have either an external pull-up attached or activating the internal pull-up resistor. I opted for the latter option (line 13).

The Software

The software was my first venture into writing in the Python language. Listing 1 shows the Steady Hands Python code. The program is quite straightforward: To begin, the three lines must be set up as inputs. Although they boot up as inputs anyway, it is always good practice to initialize the lines you want to use. I used the GPIO numbers and not the physical pin numbers in the code because the rest of the world uses this convention when referring to pin numbers on connectors.

The game progresses in three phases.

1. Wait until the loop is placed on the start rest.
2. Wait until the loop is removed from the start rest.
3. Time the interval from lifting it off the start rest until it reaches the end rest. While it is

in this phase, monitor the bent wire for touches.

See the comments in Listing 1 for more information on these phases and how they are expressed in the code. The program repeats forever, until a Ctrl + C is pressed to stop the program.

Conclusion

In this article, I just present the bare bones of what is possible. A good way to learn anything is to extend and modify from a base. This is your base. One extension would be to add a sound whenever the bent wire is touched. You could also keep track of the best scorer, or even have a table of best scores along with the names; make the score chart permanent by writing it out to a file and reading the file when the program first starts up; or add penalty points into the time (e.g., three seconds per point) to give a single score. On a more practical level, see if you can abort a timed run when the loop is placed back on the start loop.

Have fun.



LISTING 1: Steady Hands in Python

```
01 # python3
02 # Steady hands game
03 # run with - sudo python3 Steady.py
04
05 import RPi.GPIO as GPIO # get the library to access the
    GPIO pins
06 import time
07
08 # use BCM GPIO numbering -
    use anything else and you are an idiot!
09 GPIO.setmode(GPIO.BCM)
10
11 # set up GPIO input pins
12 # (pull_up_down be PUD_OFF, PUD_UP or PUD_DOWN,
    default PUD_OFF)
13 GPIO.setup(4, GPIO.IN, pull_up_down=GPIO.PUD_UP)
14 # GPIO 0 & 1 have hardware pull ups fitted in the Pi so
    don't enable them
15 GPIO.setup(0, GPIO.IN, pull_up_down=GPIO.PUD_OFF)
16 GPIO.setup(1, GPIO.IN, pull_up_down=GPIO.PUD_OFF)
17 # uncomment these next two lines if you have an issue 2
    board
18 #GPIO.setup(2, GPIO.IN, pull_up_down=GPIO.PUD_OFF)
19 #GPIO.setup(3, GPIO.IN, pull_up_down=GPIO.PUD_OFF)
20
21
22 print("Hi from Python :- Steady Hands game")
23 start_rest = 4
24 end_rest = 0 # change to 2 if you have an issue 2 board
25 wire = 1 # change to 3 if you have an issue 2 board
26
27 while True:
28 #1) Wait until the loop is placed on the start rest.
29 print("Move the loop to the start rest")
30 while GPIO.input(start_rest) != 0:
    # returns 0 when loop is on the start rest
31 time.sleep(0.8)
32
33 #now we are at the start of the bent wire
34 print("Start when you are ready")
35 #2) Wait until the loop is removed from the start rest.
36 while GPIO.input(start_rest) == 0:
    # returns 0 when loop is on the start rest
37 time.sleep(0.1)
38 print("Your off")
39 #3) Time the interval from lifting it off the start rest
    until it reaches the end
40 penalty = 0
41 run_time = time.clock()
42
43 while GPIO.input(end_rest) != 0:
    #returns 0 when loop is on the end rest
44 if GPIO.input(wire) == 0:
    #returns 0 when the loop is touching the wire
45 penalty = penalty + 1
46 print("Penalties total", penalty, " points")
47 time.sleep(0.07)
48 score = time.clock() - run_time + (penalty * 0.07)
49 print("The run time was", score, "seconds with",
    penalty, "Penalty points")
50 #finished a run so start again
```


Shop the Shop

shop.linuxnewmedia.com

Want to subscribe?

Need training?

Searching for that back issue you really wish you'd picked up at the newsstand?

Discover the past and invest in a new year of IT solutions at Linux New Media's online store.

➤ shop.linuxnewmedia.com

DIGITAL & PRINT
SUBSCRIPTIONS



SPECIAL EDITIONS



TRAINING



- LPIC-1 LPI 101 - CompTIA Linux+ LX0-101
- LPIC-1 LPI 102 - CompTIA Linux+ LX0-102
- LPIC-1 - CompTIA Linux+ 101 + 102

Raspberry Pi IR remote

In Control

Turn a Raspberry Pi into an IR remote control for your DSLR, TV, or any other device with an IR port.

By Dmitri Popov



Connect an LED with a limiting resistor to Raspberry Pi's GPIO pins, and you can control the diode using code written in your preferred scripting language. However, if you can make the LED blink using the Raspberry Pi, couldn't you turn the little machine into an infrared (IR) remote control using an IR LED instead?

It's possible, but it does require a handful of additional components and some work.

Although the idea of turning a Raspberry Pi into a glorified IR transmitter might sound like an interesting academic exercise, the final result can be used as a versatile replacement for a humble remote control. The most obvious advantage of using a Raspberry Pi-based IR remote control is that you can program it via scripts. You can also add some clever functionality on top. For example, you can write a simple server that makes it possible to access and control the Raspberry Pi-based IR transmitter from any machine or device.

BUILDING AND TESTING AN IR LED CIRCUIT

Although you can connect an IR LED directly to GPIO pins on the Raspberry Pi, the LED's output signal will

be too weak, and the IR transmitter will have a very limited range. A simple transistor circuit solves the problem by amplifying the current output from a pin and thus increasing the IR LED's signal strength.

To build a transistor-powered IR transmitter, you need two resistors (220ohm and 10K), a transistor (2N2222, BC547, or practically any other transistor will do), and a 940nm IR LED. Additionally, you'll need a breadboard and jump wires to assemble an IR transmitter prototype (Figure 1). Wire the components as shown in Figure 2 to assemble the IR transmitter. The next step is to check to see whether the IR transmitter actually works. To do this, you can use a simple LED Python blinking script (Listing 1) that turns the LED connected to pin 22 on and off.

Because the IR LED is not a regular light-emitting diode, how do you actually find out whether it blinks or not? You can use a camera with an LCD screen or smartphone camera. Point the camera at the circuit and look at the screen. If the circuit works, you should see the IR LED blinking.

INSTALLING AND CONFIGURING THE LIRC PACKAGE

To control a device with an IR receiver, the IR LED transmitter must send a specific signal

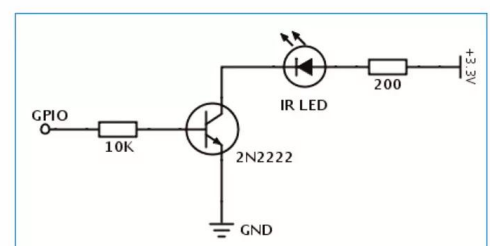


Figure 1: Transistor-powered IR transmitter schematics.

Lead Image © roger costa moreira, 123RF.com

LISTING 1: Python Blinking LED Script

```

01 #!/usr/bin/python
02 import RPi.GPIO as GPIO
03 import time
04 GPIO.setwarnings(False)
05 GPIO.setmode(GPIO.BCM)
06 GPIO.setup(22, GPIO.OUT)
07 while True:
08     GPIO.output(22, True)
09     time.sleep(1)
10     GPIO.output(22, False)
11     time.sleep(1)

```

sequence, and the LIRC package [1], which emulates the infrared signals of many remote controls, is the perfect tool for the job. LIRC is available in the Raspbian software repositories, so installing it on Raspberry Pi is just a matter of running:

```
sudo apt-get install lirc
```

Once you've done that, you need to enable and configure the `lirc_rpi` kernel module. To do so, open modules in the Nano editor

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

and add the lines below to the file:

```
lirc_dev
lirc_rpi gpio_out_pin=22
```

Make sure that the `gpio_out_pin` parameter points to the pin controlling the IR LED (in this case, it's pin 22). Next, open the file `/etc/lirc/hardware.conf` in Nano as before with `sudo` and add the following configuration to the file:

```

LIRCD_ARGS="--uinput"
LOAD_MODULES=true
DRIVER="default"
DEVICE="/dev/lirc0"
MODULES="lirc_rpi"
LIRCD_CONF=""
LIRCMD_CONF=""

```

Now, reboot the Raspberry Pi using the

```
sudo reboot
```

command to activate the configuration. Finally, you need to specify a profile that emulates a specific remote control. The project's website [2] offers a long list of profiles that emulate practically any remote control in existence, including remote controls for DSLR cameras. So, if you want to use Raspberry Pi to control a Nikon D90 DSLR camera, point the browser to lirc.sourceforge.net/remotes/nikon/ML-L3 and copy the profile. Next, open the `/etc/lirc/lircd.conf` file in Nano, paste the copied profile into it, save the changes, and restart LIRC with:

```
sudo /etc/init.d/lirc restart
```

Turn on the DSLR camera and enable the IR triggering mode. On your Raspberry Pi, issue:

```
irsend SEND_ONCE Nikon2 shutter
```

If everything works properly, your camera should fire.

CONTROLLING THE IR REMOTE WITH ANDROID

To use the Raspberry Pi-based IR remote control, you need direct access to the com-

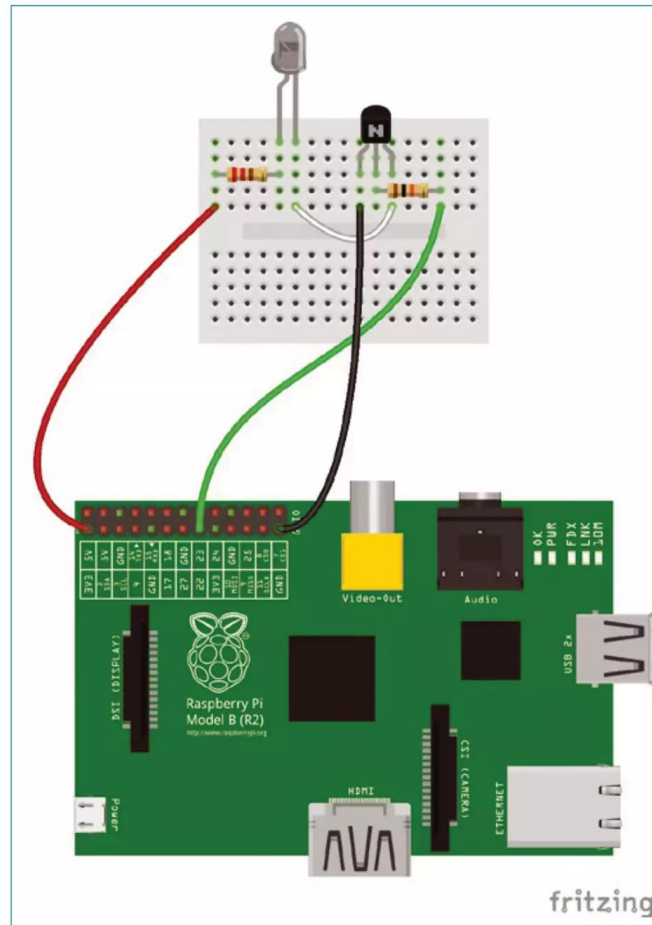


Figure 2: Wiring diagram for IR transmitter.

puter, or you need to establish an SSH connection to it from a remote machine. Needless to say, this approach is hardly practical in real-world situations. However, if you happen to have an Android device, you can use it to connect to Raspberry Pi and control the IR trigger.

For this trick to work, your Android device must support USB tethering and have the VX ConnectBot app [3] installed on it. On Raspberry Pi, open `/etc/network/interfaces` in Nano, and add the following configuration:

```
iface usb0 inet static
address 192.168.42.42
netmask 255.255.255.0
network 192.168.42.0
broadcast 192.168.42.255
```

This step effectively turns the first USB port into a network interface. Reboot Raspberry Pi, connect your Android device to it via USB, and enable USB tethering. Then, launch VX ConnectBot and use the `pi@192.168.42.42` address to establish an SSH connection to the Raspberry Pi (see the “Wireless Network Option” box for more information).

WRITING A SIMPLE IR REMOTE SERVER

Controlling the IR remote control via an SSH connection is not particularly user friendly. To make the Raspberry Pi-based IR remote easier to use, you might want to add a web interface to it. This approach also allows you to use a browser instead of an SSH utility or app to control the IR remote from any device. For this task, you can use the Bottle micro framework [4] for building web applications with Python. Installing Bottle on the Raspberry Pi requires the first two of the following commands:

```
sudo apt-get install python-pip
sudo pip install bottle

nano ir_remote_server.py
```

WIRELESS NETWORK OPTION

If you plan to use the IR remote indoors, you can simply configure the Raspberry Pi to connect to the local wireless network (this requires a USB wireless adaptor). You can then use any machine or mobile device on the same network to connect to the Raspberry Pi and control the IR remote.

The third command then creates the `ir_remote_server.py` Python script and opens it for editing. Here, you enter the code in Listing 2, then save the script and make it executable with:

```
chmod +x ir_remote_server.py
```

The script generates a simple web app containing a single `submit` button. When the button is pressed, the script uses the `os.system` function to run the `irsend SEND_ONCE Nikon2 shutter` command. Run the `sudo ./ir_remote_server.py` command to launch the script, and you can access the web application by pointing the browser to `http://<127.0.0.1>:8080` (replace `<127.0.0.1>` with the actual IP address of the Raspberry Pi).

The created script is rather bare-bones, but you can make it prettier and include additional functionality (Figure 3). Your options are only limited by your coding skills, but you can use the improved version of the script in Listing 3 as a starting point. This version adds styling and implements the feature that makes it possible to take a specified number of photos at the predefined time intervals.

When you run the script using the `sudo ./ir_remote_server.py` command, you have to keep the terminal window (or session if you connect to Raspberry Pi remotely) open. To stop the script, you can either use the `Ctrl + C` keyboard shortcut or simply

LISTING 2: Simple IR Remote Server

```
01 #!/usr/bin/python
02 from bottle import post, route, request, run
03 import os
04 @route('/')
05 @route('/', method='POST')
06 def release_control():
07     if request.method == 'POST':
08         os.system("irsend SEND_ONCE Nikon2 shutter")
09     return ""
10 <meta name="viewport"
    content="width=device-width,
    initial-scale=1">
11 <form method="POST" action="/">
12 <input id="submit" name="submit"
    type="submit" value="Shutter
    Release">
13 </form>
14 ""
15 run(host="0.0.0.0", port=8080,
    debug=True)
```


close the terminal window. To make the script run as a background process, use the

```
nohup sudo ./ir_remote_server.py &
```

command. This way, you don't have to worry about accidentally stopping the script.

Instead of launching the script manually, you can create a cron job that automatically runs the script when the Raspberry Pi boots. To do this, run the

```
crontab -e
```

command and specify the following cron job:

```
@reboot nohup sudo \
/home/pi/ir_remote_server.py.py &
```

Save the changes, and the script will run automatically when the Raspberry Pi boots.

GOING FURTHER

This project provides a starting point for experimenting with an IR LED and the Linux Infrared Remote Control (LIRC) package on the Raspberry Pi and for extending the basic setup with new features and functionality, such as

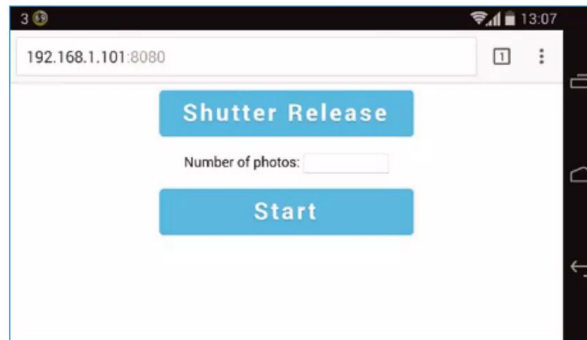


Figure 3: Bare-bones web app for the Raspberry Pi-based IR remote.

adding an IR receiver to generate a LIRC remote profile using an existing remote control. You can also replace the basic Bottle-based server with a more advanced version; clues on how to do this can be found online [5]. ●●●

INFO

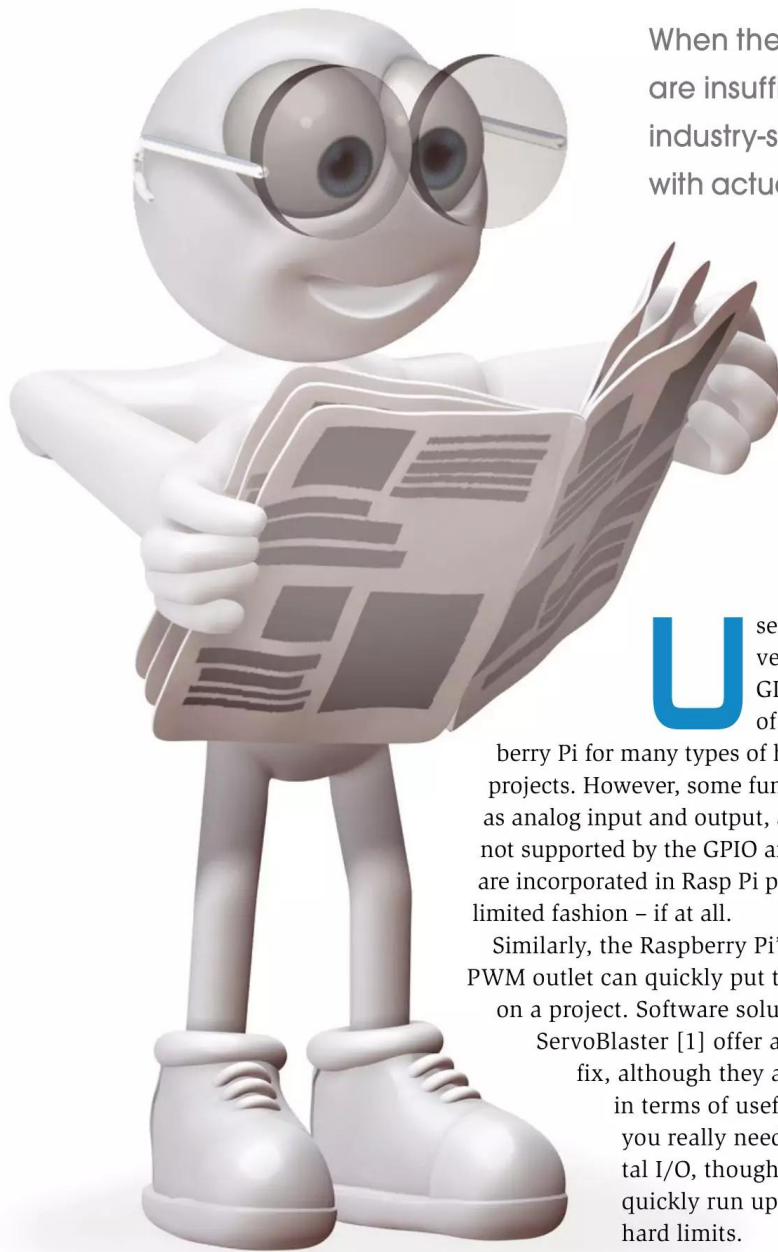
- [1] LIRC: www.lirc.org
- [2] LIRC remote control profiles: lirc.sourceforge.net/remotes
- [3] VX ConnectBot Android app: play.google.com/store/apps/details?id=sk.vx.connectbot
- [4] Bottle Python web framework: bottlepy.org
- [5] Web-based IR remote on the Raspberry Pi: randomtutor.blogspot.co.uk/2013/01/web-based-ir-remote-on-raspberry-pi.html

LISTING 3: Improved Version of the IR Remote Server

```
01 #!/usr/bin/python
02 from bottle import post, route, request, run
03 import os, time
04 @route('/')
05 @route('/', method='POST')
06 def release_control():
07     if (request.POST.get("shutter_release")):
08         os.system("irsend SEND_ONCE Nikon2 shutter")
09     if (request.POST.get("number")):
10         i = 1
11         number = int(request.forms.get('number'))
12         while (i <= number):
13             os.system("irsend SEND_ONCE Nikon2 shutter")
14             time.sleep(3)
15             i = i + 1
16     return ""
17 <meta name="viewport" content="width=device-width,
18   initial-scale=1">
19 <form method="POST" action="/">
20 <div id="content"><p><input id="submit" name="shutter_
21   release" type="submit"
22   value="Shutter Release"></p>
23   <p>Number of photos: <input name="number" type="text"
24   size="9"/></p>
25   <input id="submit" value="Start" type="submit" />
26 </div>
27 <style>
28   body {
29     font: 15px/25px 'Fira Sans', sans-serif;
30   }
31   #content {
32     margin: 0px auto;
33     text-align: center;
34   }
35   #submit {
36     width: 11em; height: 2em;
37     background: rgb(66, 184, 221);
38     border-radius: 5px;
39     color: #fff;
40     font-family: 'Fira Sans', sans-serif;
41     font-size: 25px;
42     font-weight: 900;
43     text-shadow: 0 1px 1px rgba(0, 0, 0, 0.2);
44     letter-spacing: 3px;
45     border:none;
46   }
47 </style>
48 ""
49 run(host="0.0.0.0",port=8080, debug=True)
```

Getting to know the Raspberry Pi I2C bus

I-Square-C



When the Rasp Pi GPIO connection capabilities are insufficient for your project, you can turn to the industry-standard I2C data bus to communicate with actuators and sensors. *By Martin Mohr*

Users rely on the very efficient GPIO interface of the Raspberry Pi for many types of hardware projects. However, some functions, such as analog input and output, are generally not supported by the GPIO and therefore are incorporated in Rasp Pi projects in a limited fashion – if at all.

Similarly, the Raspberry Pi's one lone PWM outlet can quickly put the brakes on a project. Software solutions like ServoBlaster [1] offer a convenient fix, although they are restricted in terms of usefulness. If you really need more digital I/O, though, you will quickly run up against hard limits.

The I2C bus is a simple and professional solution for projects of large scope, and the Raspberry Pi has two corresponding interfaces on-board. In this article, I discuss the PCF8574 semiconductor, which has an interface for the I2C bus.

GENERAL INFORMATION

The I2C bus is a serial master-slave bus suitable for communication over short dis-

tances – within a circuit board or within a device. The early 1980s technology came out of Philips Semiconductors (now part of NXP Semiconductors) for building control electronics in entertainment systems.

Data transmission occurs synchronously over two bidirectional lines: the serial data line, *SDA*, and the serial clock line, *SCL*. Resistors pull both lines up to a positive potential. The master node specifies the speed and operating mode and initiates communication byte for byte. The transmission speed of the bus varies from 100Kbps bidirectional in standard mode up to 5Mbps unidirectional in ultrafast mode (Table 1).

The I2C bus works with an address range of 7 bits, for up to 128 addresses. However, 16 of these addresses are reserved for special tasks, leaving 112 free. The eighth bit tells the slave whether it should receive data from the master or transmit data to the master.

Usually, you can only choose lower bits of the addresses on the slaves because the upper bits are predefined. The best way to figure out which addresses are available is to check the information sheet that comes with the product. Table 2 shows some examples of various address spaces.

If you are working on a large project and need more than the 112 available addresses,

TABLE 1: I2C Clock Rates

Mode	Maximum Transmission Rate	Direction
Standard mode	100Kbps	Bidirectional
Fast mode	400Kbps	Bidirectional
Fast mode Plus	1Mbps	Bidirectional
High-speed mode	3.4Mbps	Bidirectional
Ultrafast mode	5.0Mbps	Unidirectional

TABLE 2: I2C Address Space

Type	Function	Address
PCF8574	Port expander	0x20-0x27
PCF8574A*	Port expander	0x38-0x3F
PCF8591	ADC/DAC	0x90-0x9F
PCF8583	Clock/Calendar	0xA0-0xA2

*Only the address space differs between PCF8574 and PCF8574A.

you can attach a bus multiplexer to the primary I2C bus.

The I2C bus is susceptible to disturbances because it was originally designed to bridge just a few centimeters. Various possibilities exist to help improve the electrical characteristics of the bus, ranging from adaptation of the pull-up resistors to tunneling through CAN bus drivers.

TESTING

You will find an I2C interface directly on the Raspberry Pi GPIO. Pins P1-03 (P1 header, pin 3), *SDA_1*, and P1-05, *SCL_1*, already have the necessary 1.8kohm pull-up resistors built-in that pull the Raspberry Pi to 3.3V on idle.

You can access a second I2C interface via the P5 connector, but you will first have to solder this onto the back of the board. You can communicate with the second interface via the P5-3 (*SDA_0*) and P5-4 (*SCL_0*) pins, although they have no premounted pull-up resistors.

To run a test, you can attach a PCF8574 microcontroller [2] to the I2C bus (Figure 1) and then observe the most important of its basic functions. The circuit diagram available online [3] for this simple test setup includes four LEDs and four switches.

GPIO AND THE I2C DRIVER

To use the I2C bus, you need to load the I2C driver, which requires the WiringPi library [4]

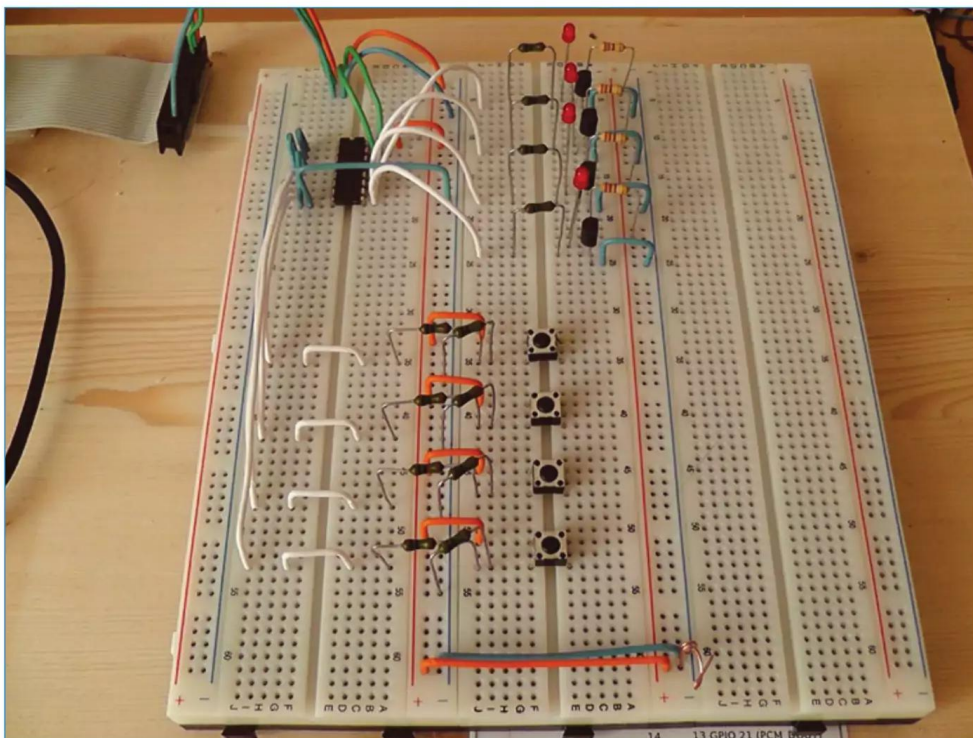


Figure 1: A PCF8574 microcontroller and four LEDs, each with its own switch.

```
pi@i2c ~ $ gpio readall
```

BCM	wPi	Name	Mode	V	Physical	V	Mode	Name	wPi	BCM
2	8	3.3v	IN	1	1	2		5v		
3	9	SDA.1	IN	1	3	4		5V		
4	7	SCL.1	IN	1	5	6		0v		
		GPIO. 7	IN	1	7	8	1	ALT0	TxD	15
		0v			9	10	1	ALT0	RxD	16
17	0	GPIO. 0	IN	0	11	12	0	IN	GPIO. 1	1
27	2	GPIO. 2	IN	0	13	14		0v		18
22	3	GPIO. 3	IN	0	15	16	0	IN	GPIO. 4	16
		3.3v			17	18	0	IN	GPIO. 5	1
10	12	MOSI	IN	0	19	20		0v		23
9	13	MISO	IN	0	21	22	0	IN	GPIO. 6	4
11	14	SCLK	IN	0	23	24	1	IN	GPIO. 5	5
		0v			25	26	1	IN	CE0	10
									CE1	11
										7
28	17	GPIO.17	IN	0	51	52	0	IN	GPIO.18	18
30	19	GPIO.19	IN	0	53	54	0	IN	GPIO.20	20
										29
										31

```
pi@i2c ~ $
```

Figure 2: Checking the WiringPi library installation with `gpio readall`.

(Listing 1). You might already be familiar with the library from other projects. Before downloading, you update the system and install the Git client (line 3, `git clone`). The output from `gpio -v` or `gpio readall` (Figure 2) will let you know whether the library has been installed successfully.

Now you have access to the GPIO, although the system and the I2C bus are not

yet able to understand one other. For this you need the *i2c-tools* from the packet sources; then, you should incorporate the standard user, *pi*, for the system in the user group *i2c*:

LISTING 1: Installing WiringPi

```
$ sudo apt-get update && sudo apt-get upgrade
$ sudo apt-get install git-core
$ git clone git://git.drogon.net/wiringPi
$ cd wiringPi
```

LISTING 2: `i2cdetect`

```
$ i2cdetect -y 1

 0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
10:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
20: 20  --  --  --  --  --  --  --  --  --  --  --  --  --  --
30:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
40:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
50:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
60:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
70:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
```

```
$ sudo apt-get install i2c-tools
$ sudo adduser pi i2c
$ gpio load i2c
```

A single logout and login will activate the group membership. The final line loads the I2C driver using the `gpio` command:

The `i2cdetect` command will now display which devices are hanging on the bus and the address for each of them.

The test setup communicates with the controller via the address `0x20` (Listing 2).

The upper 4 bits of the I/O component are in contact with the LEDs, whereas the low nibble (4 bits or one-half byte) is connected to the switches.

All of the PCF8574 pins are set HIGH by default, so you should not be surprised when all of the LEDs light up. You can turn all of the LEDs off at once with the first command in Listing 3.

The following lines then turn the LEDs on one at a time. Initially, you should set the corresponding pins HIGH to read out the switches. The PCF8574 is largely responsible for having to do things this way.

To get started at this stage, you should therefore set the pin with `i2cset` and read out the condition of the switches with `i2cget`, using `watch` to echo the output to the console:

```
$ i2cset -y 1 0x20 0x0f
$ watch 'i2cget -y 1 0x20'
```

Now you are familiar with one possible way to access the I2C bus. The commands used at the command line can be called from all programming languages. As a result, even more complex I2C applications do not present obstacles. However, many developers start getting enthusiastic only when they are able to use an API for their favorite language.

I2C WITH C

To avoid going beyond the scope of this article, I will stick to addressing just the C and Java APIs. Microcontroller enthusiasts often use C because of its thin layer of abstraction. Java is exactly the opposite. The Java programmer doesn't want to know anything at all about the hardware that executes a program. Therefore, I will begin with C.

LISTING 3: Turning LEDs Off and On

```
$ i2cset -y 1 0x20 0x00
$ i2cset -y 1 0x20 0x10
$ i2cset -y 1 0x20 0x20
$ i2cset -y 1 0x20 0x40
$ i2cset -y 1 0x20 0x80
```


TABLE 3: Basic Functions in C

Function	Purpose
<code>int wiringPiI2CRead(int handle)</code>	Simple read
<code>int wiringPiI2CWrite(int handle, int data)</code>	Simple write
<code>int wiringPiI2CReadReg8(int fd, int reg)</code>	Read 8-bit values from a register
<code>int wiringPiI2CWriteReg8(int fd, int reg, int data)</code>	Write 8-bit values to a register
<code>int wiringPiI2CReadReg16(int fd, int reg)</code>	Read 16-bit values from a register
<code>int wiringPiI2CWriteReg16(int fd, int reg, int data)</code>	Write 16-bit values to a register

In C, access to an I2C device is related to accessing a file. First you should get a file handle with:

```
wiringPiI2CSetup(<address>)
```

Here, <address> is the value output from `i2cdetect`. Table 3 lists the most important C function calls.

Other libraries, which I will look at more closely in future articles, build on these functions for more complex program compo-

nents. By way of concluding this introduction to C, Listing 4 makes the LEDs in the test setup light up one after the other. When compiling, you need to make sure you incorporate the WiringPi library:

```
$ cc i2c_test.c -lwiringPi
$ ./a.out
```

PROGRAMMING IN JAVA

To control the GPIO with Java, you need to use the corresponding library of the Pi4J project [5], but you should undertake some preparatory work first. Once again, you need the WiringPi library, this time for Java. Subsequently, you should install the current Java development kit, JDK, from Oracle [6].

The easiest approach to this installation is to transfer the JDK via SFTP onto the Raspberry Pi (Listing 5, line 1). (The best way to do this in Windows is with WinSCP [7].) Next, log on to the Rasp Pi via SSH (line 2) and decompress the tarball into `/opt/java` (lines 3-5).

After that, you should let the system know via `update-alternatives` which version of Java it should use. Finally, you should make Java display information about the version (line 8).

The Oracle JDK is now ready to go. To be on the safe side, you should set the environment variable `$JAVA_HOME` immediately, be-

LISTING 4: Light LEDs in C

```
01 #include <wiringPiI2C.h>
02
03 int main(void) {
04     int handle = wiringPiI2CSetup(0x20) ;
05     wiringPiI2CWrite(handle, 0x10);
06     delay(5000);
07     wiringPiI2CWrite(handle, 0x20);
08     delay(5000);
09     wiringPiI2CWrite(handle, 0x40);
10     delay(5000);
11     wiringPiI2CWrite(handle, 0x80);
12     delay(5000);
13     wiringPiI2CWrite(handle, 0x00);
14     return 0;
15 }
```

LISTING 5: Setting Up Java

```
$ scp jdk*.tar.gz pi@<RaspPi-IP>:~
$ ssh pi@<RaspPi-IP>
$ sudo -s
$ mkdir -p -v /opt/java
$ tar -xzf jdk*.tar.gz -C /opt/java/
$ update-alternatives --install "/usr/bin/java" "java" "/opt/java/jdk<Version>/bin/java" 1
$ update-alternatives --set java /opt/java/jdk<Version>/bin/java
$ java -version
```

LISTING 6: Java Environment Variable

```
$ echo 'export JAVA_HOME="/opt/java/jdk<Version>/bin"' >> ~/.bashrc
$ export JAVA_HOME="/opt/java/jdk<Version>/bin"
```

cause some programs need this. If you want to specify the variable for the entire system, you have to enter it with root rights in the `/etc/` environment file.

During initial trials, it is fine simply to establish the variable in the user-specific `~/.bashrc` (Listing 6) so that it becomes available immediately after the next login. If you want to continue working without logging out, an exception allows you to set the variable for the current session at hand.

Now all preconditions for the system have been fulfilled, and the pathway to the Pi4J library should be open. The quickest way to install is to use a setup script, which you can load from the Internet and execute in one step:

```
$ curl -s get.pi4j.com | sudo bash
```

Installation will take a bit of time, but the wait is worthwhile.

LISTING 7: Java – Light LEDs

```
01 import com.pi4j.io.i2c.I2CBus;
02 import com.pi4j.io.i2c.I2CDevice;
03 import com.pi4j.io.i2c.I2CFactory;
04 public class TestI2C {
05     private static final int i2cBus = 1;
06     private static final int address = 0x20;
07     public static void main(String[] args) {
08         try {
09             I2CBus bus=I2CFactory.getInstance(I2CBus.BUS_1);
10             I2CDevice dev = bus.getDevice(address);
11             dev.write((byte)0x10);
12             Thread.sleep(5000);
13             dev.write((byte)0x20);
14             Thread.sleep(5000);
15             dev.write((byte)0x40);
16             Thread.sleep(5000);
17             dev.write((byte)0x80);
18             Thread.sleep(5000);
19             dev.write((byte)0x00);
20         }
21         catch(Exception e) {
22             System.out.println(e);
23         }
24     }
25 }
```

TABLE 4: I2C Bus Microcontroller

Component	Description
PCF8591	ADC/DAC
PCF8583	Timer/clock
TLC5940NT	PWM component
LM75	Temperature sensor
PD515A	Driver
82B715	Driver
PCA9554	8-bit-I/O
PCF8574	8-bit-I/O
ADCD818	8-channel ADC

This addition makes it somewhat easier to try out small Java programs. The `pi4j --help` command illustrates additional program functions.

To compile the example, enter:

```
$ pi4j -c TestI2C.java
```

The `-c` switch builds the program. If Pi4J doesn't report an error, you should find a file named `TestI2C.class` in the updated directory. This is the Java class you will execute with Pi4J, this time using the `-r` switch:

```
$ pi4j -r TestI2C
```

As in the first example, the Java program from Listing 7 makes the four LEDs light up one after the other. Likewise in Java, you should first create a connection to the I2C device and then write or read byte by byte. The example program `TestI2C.java` available online [3] can be the basis for developing your own ideas.

To make your Java life a little easier, the developers have included a small script with their `pi4j` library that can be used to set the correct class path automatically when you compile.

The LEDs on the test circuit should turn on if this process has been successful.

CONCLUSION

In this article, I laid out the basics for examining the extensive possibilities afforded by the I2C bus when it is controlled using terminal commands or the C or Java API. Various other microcontrollers that work with the I2C bus are listed in Table 4 and are suitable for creating many new projects. ●●●

INFO

- [1] ServoBlaster: <https://github.com/richardghirst/PiBits/tree/master/ServoBlaster>
- [2] PCF8574 data sheet: http://www.nxp.com/documents/data_sheet/PCF8574.pdf
- [3] Schematics and code: ftp://ftp.linux-magazine.com/pub/listings/linux-magazine.com/RaspPi_Handbook/
- [4] WiringPi: <http://wiringpi.com>
- [5] Pi4J: <http://pi4j.com>
- [6] JDK 8 for ARM: <http://www.oracle.com/technetwork/java/javase/downloads/jdk8-arm-downloads-2187472.html>
- [7] WinSCP: <http://winscp.net/>

Building Rasp Pi into Mindstorms projects with BrickPi

PiBot

BrickPi lets you build a Raspberry Pi into your Lego Mindstorms projects – you can even connect your Pi to Mindstorms sensors and motors. *By Marko Dragicevic*

It would be a mistake to dismiss Mindstorms as a child's toy just because it is made by Lego. When he was in college, Google co-founder Larry Page built a functioning ink jet printer out of Mindstorms parts and related accessories. All over the world, tinkerers are expanding the capabilities of Lego robots by hooking several Lego NXT control modules together or connecting them to a PC via Bluetooth to access PC storage space. A doctoral student from London has even combined Mindstorms elements with other modules and other components produced by a 3D printer to make a functioning atomic microscope.

Lego published the first version of Mindstorms in 1998. At that time, the only goal envisioned for the product was to better acquaint young people with the subject of robotics. However, the system quickly became so popular that the manufacturers began to include it in their general product line. These build-it-yourself robots have now become a part of the curriculum at many universities.

The benefit of Mindstorms is in the hardware. Even in the absence of extensive background knowledge, a user can experience success simply by plugging the parts together. Mindstorms parts from the Lego Technic and Lego System series aren't the large blocks you remember from preschool. In addition to the blocks and moving parts, servo motors and numerous sensors deal with sound, color, ultrasound, touch, and temperature. The goal is to dock all of these components onto other building blocks and connect them via cable to the socket of a large, central control module that makes CPU, memory, and interfaces available.

From 2006 to 2013, the NXT module was Lego's control module for Mindstorms robots.

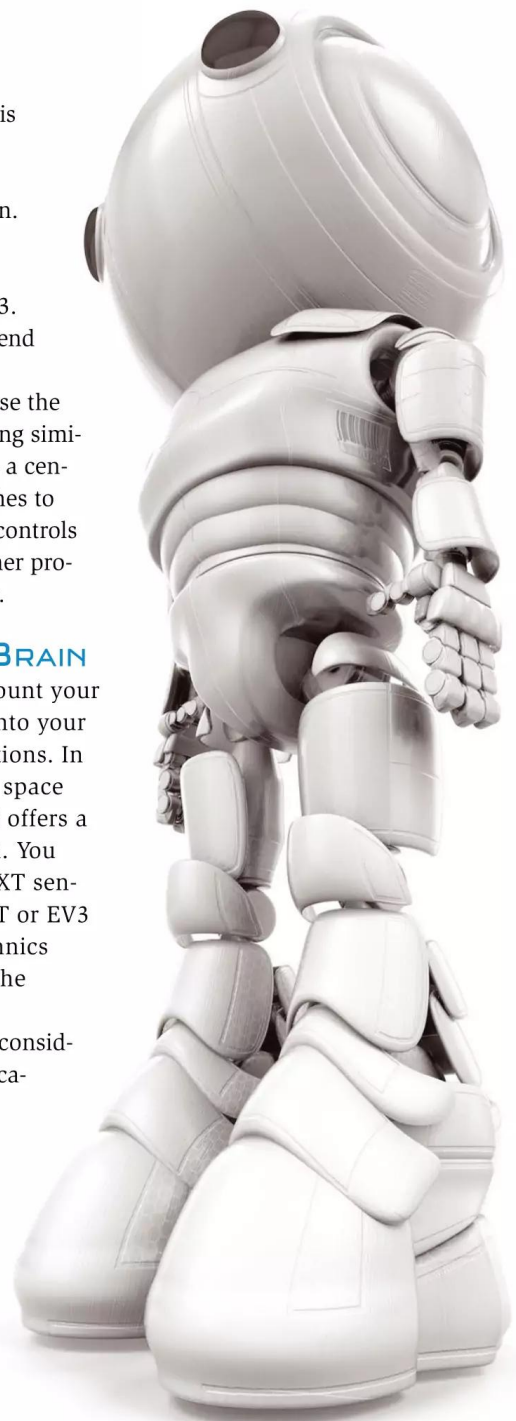
The technology for NXT is getting a bit long in the tooth, but it nonetheless enjoys a wide distribution. The successor to NXT is called EV3, and it appeared in September 2013. I will discuss EV3 at the end of this article.

The BrickPi lets you use the Raspberry Pi as something similar to the NXT module – a central processor that attaches to Mindstorms bricks and controls motors, sensors, and other programmable components.

RASP PI AS A BRAIN

The BrickPi lets you mount your Raspberry Pi directly onto your Mindstorms robot creations. In addition to providing a space for the Pi itself, BrickPi offers a power supply for the Pi. You can attach up to five NXT sensors and up to four NXT or EV3 motors. And, Lego Technics bricks attach easily to the BrickPi case.

The Raspberry Pi has considerably more computing capacity, RAM, and disk space than Lego's own NXT module. Additionally, you can log in to the Pi via WiFi and integrate arbitrary software packages and libraries offered by Raspbian or another Pi distro.



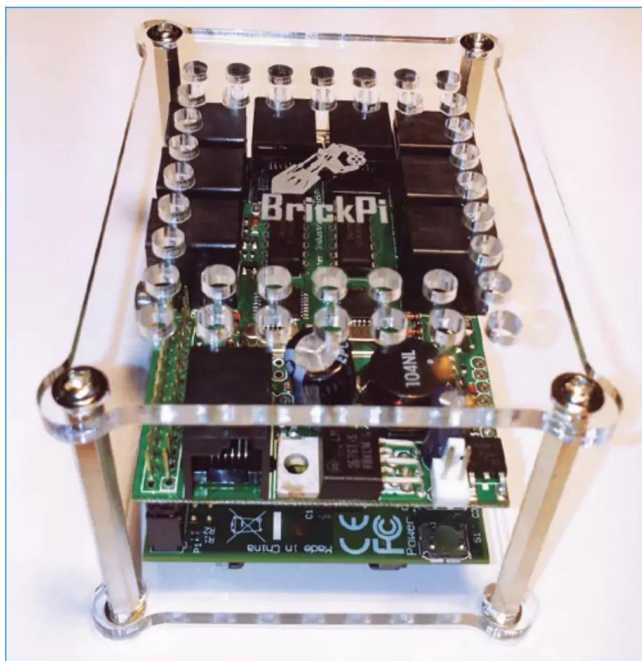


Figure 1: The BrickPi comes disassembled. The assembled block is shown above. At the bottom is a Pi and at the top is an Arduino.

According to the American manufacturer of BrickPi, Dexter Industries [1], European users can order the BrickPi via the French online shop Génération Robots [2].

The delivery set contains two acrylic disks, one circuit board that is based on Arduino with connectors for sensors and motors, and various screws and battery cables. The Raspberry Pi, which you will

also need, is not part of the delivery package. You'll need a Revision 2 Raspberry Pi board, because Rev 2 is the only one that has two drill holes, which you will need to screw the mini computer onto the acrylic disks that belong underneath.

Mounting the Arduino circuit board is not complicated. All you need to do is plug it into the GPIO port on top of the Rasp Pi like a shield. Although the fit is a bit crooked, tests have shown that there are no stability problems. The finishing step is to screw in the second acrylic disk to function as the top part of the housing. As you can see from the

finished block (Figure 1), the disks have holes that are Mindstorms compatible.

SOFTWARE SETUP

The BrickPi needs a specially adapted version of Raspbian as an operating system. You can make the necessary adaptations [3] yourself; however, the Dexter Industries website provides a link to an appropriately prepared image file.

A word of caution is in order. The image specifies 192.168.2.0/24 as the network, and, for the Rasp Pi, it assigns a static IP address of 192.168.2.144. To modify this address, you need to briefly plug the monitor and keyboard into the mini computer. Then, you can change the IP addresses with the text editor in the `/etc/network/interfaces` and `/etc/resolv.conf` files. At the same time, you can enter the SSID and password for your WiFi network into `/etc/network/interfaces`.

After you power down, remove all of the cables. You can then attach a battery to the top circuit board of the BrickPi building block to serve as the power supply. To create access to the wireless network for this block, just plug a mini WiFi stick into the USB port of the Rasp Pi. The BrickPi has now become a central brain you can build into any robot. (Make sure you leave room in your additional Mindstorms projects for a case that can carry the battery.)

The BrickPi homepage offers instructions for several example projects, including a gripper arm with a joint, as well as a small car. I tested the instructions for building portions of the car. Two servo motors control the front wheels, as shown in Figure 2. In the back is a smaller wheel, as shown in Figure 3, that pivots as the car is steered.

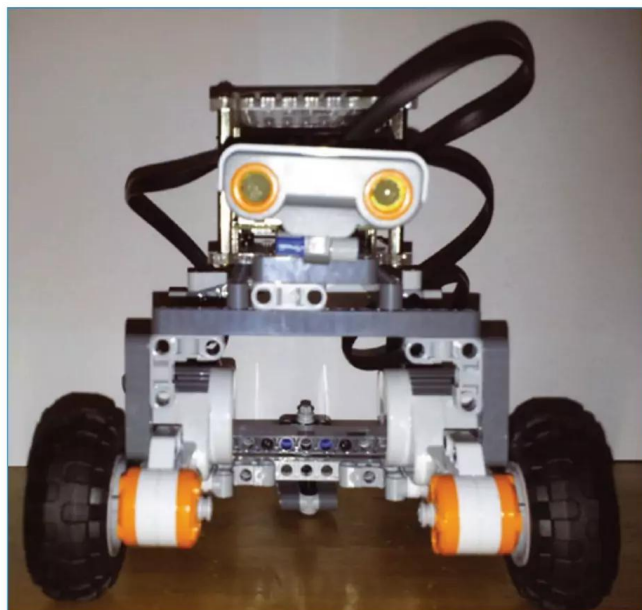


Figure 2: The BrickPi can autonomously control a car ...

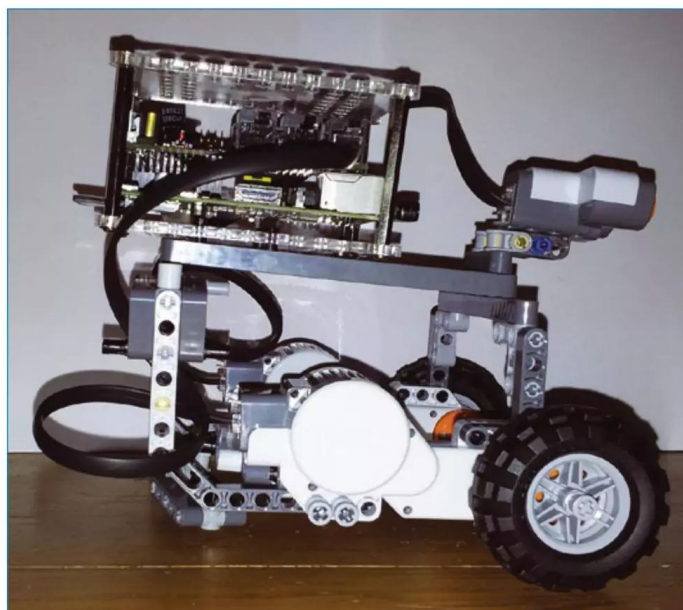


Figure 3: ... that can identify obstacles with an ultrasound sensor on the front.

Additionally, the robot carries an ultrasound sensor on the front so that it can recognize obstacles and introduce evasive maneuvers by means of programmed actions.

PROGRAMMED STUFF

To get access to the robot and thus start scripts that you have programmed yourself, you should use the headless capabilities of the Rasp Pi and log onto the BrickPi via SSH or VNC.

If have built your own robot, it will still have to receive instructions by means of software. The whole point of an on-board processor is to write programs that use the Rasp Pi to control the robot's motors and sensors. You will, therefore, need to set up the Raspberry Pi to support your favorite programming language. Listing 1 shows how to download and install the necessary tools for using Python.

If you write your own Python script for the robot, you have to import the library for controlling the BrickPi functions with the command `from BrickPi import *` and also initialize the serial port with the command `BrickPiSetup()`. Because the robot almost always uses servo motors, you will need to call the following command for each of the motors:

```
BrickPi.MotorEnable[Port] = 1
```

Depending on which of the four motor ports of the BrickPi the cable of each servo motor has been plugged into, you will have to replace the `Port` placeholder with one of the `PORT_A` to `PORT_D` identifiers.

In the last initialization step, you need to inform the BrickPi library of the types of sensors used and define any ports that are connected to sensors. For an ultrasound sensor plugged into port 1, the corresponding line will read:

```
BrickPi.SensorType[PORT_1] = 2
TYPE_SENSOR_ULTRASONIC_CONT
```

All in all, five ports are reserved for sensors. In contrast to the motors, sensors do not receive letters as identifiers. Instead, they are numbered in ascending order. Each sensor type is associated with a predefined constant. For example, the constant indicating a color sensor is `TYPE_SENSOR_COLOR_FULL`. To finish the sensor initialization process, invoke the `BrickPiSetupSensors()` command.

Communicating with the individual robot parts is easy while the program is running. A servo motor is controlled via

```
BrickPi.MotorSpeed[Port] = 2
rate_of_speed
```

For the rate of speed, you can enter a whole number between -255 and +255, with 0 bringing the

motor to a stop. Positive and negative values designate the direction for turning. To make the robot turn, you can slow down the motor of one wheel to zero while the other motor continues to turn the other wheel.

For interrogating sensors, you can call `BrickPiUpdateValues()` to obtain the current values. By subsequently invoking `BrickPi.Sensor[Port]`, you obtain a numeric value for a particular sensor. For example, an ultrasound sensor will return the distance to the nearest object.

Using the C language, you can program the BrickPi according to the same scheme. At the beginning of the program, import the required header files (Listing 2). The C names for constants and methods in the BrickPi library correspond exactly to those for the Python version, except that you call the methods according to the rules of C syntax. Fans of Scratch will find an implementation that makes graphical building blocks available for building the control functions.

CONCLUSION

If your Mindstorms set was purchased before September 2013, you will easily see many advantages of the BrickPi over the NXT. The BrickPi has considerably more computing power, more RAM, and more mass storage, without any dependency on an external PC. You can easily integrate every type of software and hardware available for the Rasp Pi into the robot. Moreover, the BrickPi opens up the possibility of installing not only Mindstorms sensors but also the less expensive Arduino sensors.

The BrickPi also offers advantages for those who possess the new EV3 system. Granted, the EV itself already has a clock speed of over 300MHz, 64MB of RAM, a USB host mode, and an operating system based on Linux. However, the Rasp Pi offers even more options.

If you still do not have a Mindstorms set, you won't have to go to the expense of buying an expensive new starter package. You can build out the BrickPi with separately purchased sensors, motors, and Lego Technic building blocks. Despite its benefits, the BrickPi is not a particularly easy system for beginners; you will fare better if you have some basic knowledge of Mindstorms and Linux. ●●●

LISTING 1: Setting up BrickPi for Python

```
$ git clone https://github.com/DexterInd/BrickPi_Python.git
$ cd BrickPi_Python
$ sudo apt-get install python-setuptools
$ sudo python setup.py install
```

LISTING 2: Including Header Files in C

```
#include <wiringPi.h>
#include "BrickPi.h"
```

INFO

- [1] Dexter Industries:
<http://www.dexterindustries.com/BrickPi/>
- [2] French BrickPi Distributor:
<http://www.generationrobots.com/en/>

Testing the new Raspberry Pi touchscreen display

Touch Power

The new Rasp Pi display provides a compact option for viewing screen output – and it comes at a Pi-like low price of only \$60.

By Christoph Langner

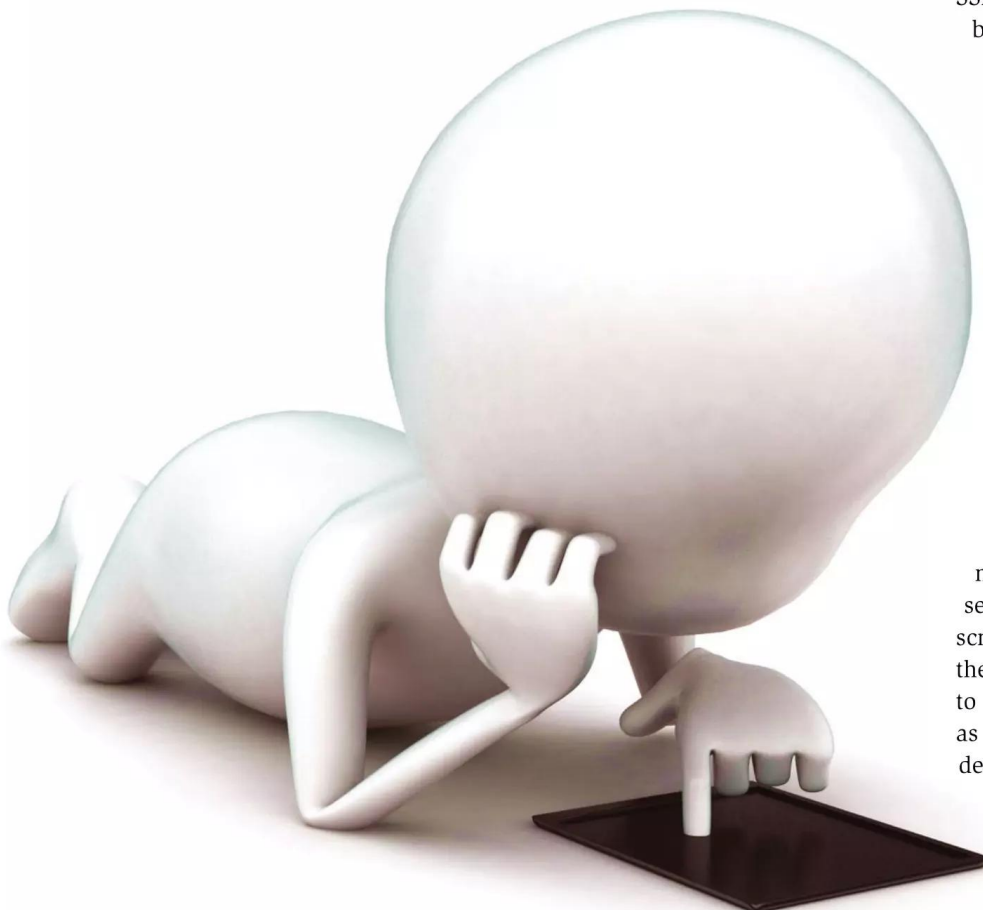
A working Raspberry Pi system requires lots of other parts in addition to the tiny Pi board itself. You'll need a power supply, a mouse, various cables, and, unless you already own one, a monitor to display screen output for your Raspberry Pi system. Raspberry Pi users are accustomed to hunting up old monitors from long-discarded computers, or temporarily requisitioning a monitor from a currently running system to use as a display for the Rasp Pi.

The clutter of a full-size monitor, however, often invades the elegant simplicity of the Raspberry Pi. Monitors tend to be big and bulky, and they typically require their own power cable, which adds more congestion and complication to the tabletop configuration. A Rasp Pi by itself can inhabit an unobtrusive corner of the room, but add a monitor to it, and it really doesn't look much different from a conventional PC.

Some users solve the monitor problem by running their Rasp Pi headless through an SSH connection, but SSH requires you to boot another computer on the network (which, undoubtedly, has its own monitor anyway).

Since the first days of the Rasp Pi, users have dreamed of a mini-monitor that runs on the Rasp Pi power and is tailored for the needs of the Rasp Pi environment. Some third-party Rasp Pi displays have appeared in the past, and we have reviewed some in this magazine, but those who follow the Pi have known for some time that the Raspberry Pi Foundation would eventually announce an official Pi display.

The Raspberry Pi display [1] was announced on September 8. In addition to serving as a general device for viewing screen output, according to the foundation, the new display also “gives users the ability to create all-in-one, integrated projects such as tablets, infotainment systems, and embedded projects.” The new 7-inch display has touchscreen capabilities. With a price tag of \$60, the Rasp Pi display came to market nine months past



its originally planned release date [2]. The foundation has explained that the delays resulted from efforts to integrate the Raspberry Pi 2 and difficulties encountered in locating a supplier for the screen.

The display screen measures 7 inches across the diagonal or 155x86mm. The entire screen, including the frame, measures 194x110mm, and the overall depth, minus an attached Rasp Pi, is about 20mm, with a weight of 277 grams. The screen has 10-finger capacitive touch, although standard Linux desktop environments currently do not handle touch gestures well.

The Rasp Pi Display Screen

The Rasp Pi supports a number of display interfaces, such as HDMI. For some screens that are already available for the Raspberry Pi, the little computer also relies on the GPIO interface. The Raspberry Pi Foundation looked to the underused Display Serial Interface (DSI) in order to connect the Rasp Pi with the screen. You will find the DSI port on the side that is opposite from the network and USB ports.

The DSI connector transmits more than just images. It also sports an integrated I2C-Bus, which makes it possible to transmit gesture entries and control signals for background lighting. As a result, the screen will only work with second-generation Raspberry Pis or the plus models, which are the first series Rasp Pi A+ and the B+. The display board does have two extra pins for the Rasp Pi B. However, according to Pi developer Gordon Hollingworth, the foundation still needs to work on support for the pins.

The first purchasers receive the display as a kit that requires assembly. Even though no assembly instructions are included, the components are easily put together. You should first set the display board upside down next to the screen and connect the board to the ribbon cable that comes out of the screen. Then, turn the board over and adjust it so you can properly mount it.

Next, very carefully plug the second ribbon cable that comes out of the screen into the small jack on the front side of the display board. This cable carries signals from the touchscreen. With the braid side facing upwards, plug in the ribbon cable, which will be connected to the Rasp Pi, and then, using a spacer, screw the board to the screen (Figure 1).

Now you need to decide how you want to supply power to the Rasp Pi and to the screen. Like the Rasp Pi, the screen has a

micro USB connector, which means you can either run the power from the display to the Rasp Pi via the USB jack, or you can use the jumper cables that are included to wire the Rasp Pi for 5V and GND with the pins that are on the pin strip. We chose the jumper cable method for our test. The first step is to connect the red cable to the 5V pin and the black cable to the GND pin, then set the Rasp Pi on the spacer. Then, connect the two connectors to the 5V and GND on the Rasp Pi.

Figure 2 shows the pin assignment of the GPIO port for a Raspberry Pi 2. To reduce the size of the depth of this arrangement, you can also mount the Rasp Pi the opposite way around to the display board. The final step to this process is to connect the DSI cable with the USB power supply (Figure 3).

Starting the Screen

With respect to the software, all you need to do is update the system (Listing 1). If updates are not properly installed, you will be able to see images on the screen, but the screen will not accept any entries. After a restart, the Rasp Pi will promptly display the image that is familiar from the HDMI port on the screen without any modifications to the system. Touch entries will also immediately function properly within the X server GUI. The 800x480 resolution should suffice for everyday use, even though it is not a substitute for a desktop monitor (Figure 4). The touchscreen has the same level of precision as tablets or smartphones that are currently available on the market.

You can install a virtual keyboard like the Matchbox keyboard in order to enter

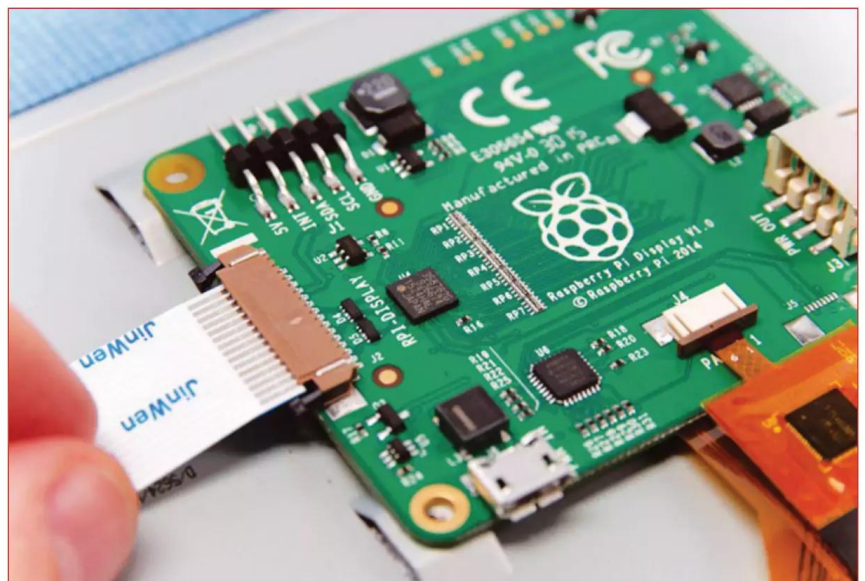


Figure 1: Purchasers have to assemble the first Rasp Pi screens on their own. Future versions of the display will come preassembled.

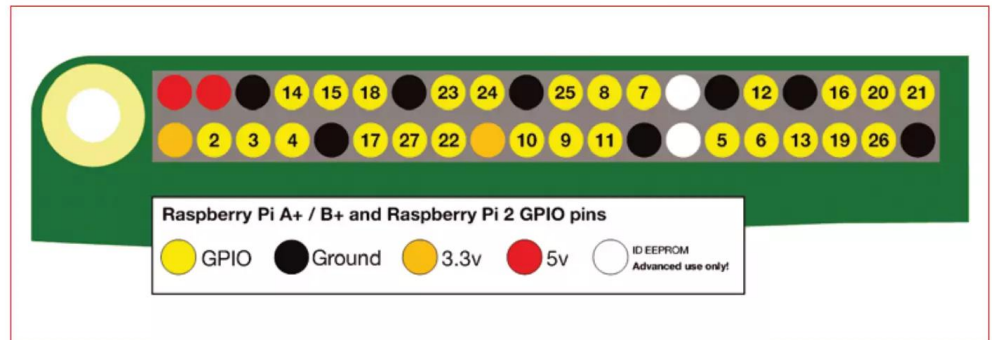


Figure 2: The GPIO pin assignment for a Raspberry Pi 2. The user will find the 5V and GND pins across from pins 2 and 3 for connecting to a power supply.

text without a USB keyboard (Listing 2). Typical desktop environments like LXDE from Raspbian generally do not function properly with touchscreen entries. The problems are especially noticeable in the web browser, where you can't just type and scroll from just anywhere on a page. Instead, you need to use the scroll bar on the edge of the window, which is hard to do with a finger. It is likewise difficult to use touch gestures for adjusting the size of program windows.

Connecting the Rasp Pi display screen through the DSI port frees the HDMI port, which is then available for attaching a second monitor. However, you do not automatically get an image on a second monitor. Instead, you first need to indicate to the program, such as a video player, that you would like to send the images to an external screen.

The example from Listing 3 starts the video on the second screen while the GUI on the Raspberry Pi screen remains free for other activities. At this point, there is still no genuine dual-display function that turns the

Rasp Pi display and a monitor connected via HDMI into a large display. The graphical tools used for setting up the resolution via *Menu | settings | screen settings* still only recognize the Rasp Pi screen. Instructions for manually configuring a monitor set up via the `/etc/X11/xorg.conf` file are not yet available.

Mediacenter and Car PC

The Raspberry Pi functions in many homes as a media center. Distributions such as OpenELEC and OSMC come with an optimized Kodi media center for the Rasp Pi. Makers are also using the Rasp Pi to construct Internet radios from old radio models. One of the most impressive of these types of projects is the portable VCR Raspberry Pi media center. You'll find detailed instructions for building your own media center online [3].

Up to now, one of the challenges for such a project has been to find a suitable display screen. Most screens did not have a large enough diagonal to play movies. Nor did they have a touchscreen, energy efficiency, and simple connection capabilities. Now you won't have to look much further than the Raspberry Pi screen. It is possible to use batteries intended for smartphone and tablets to

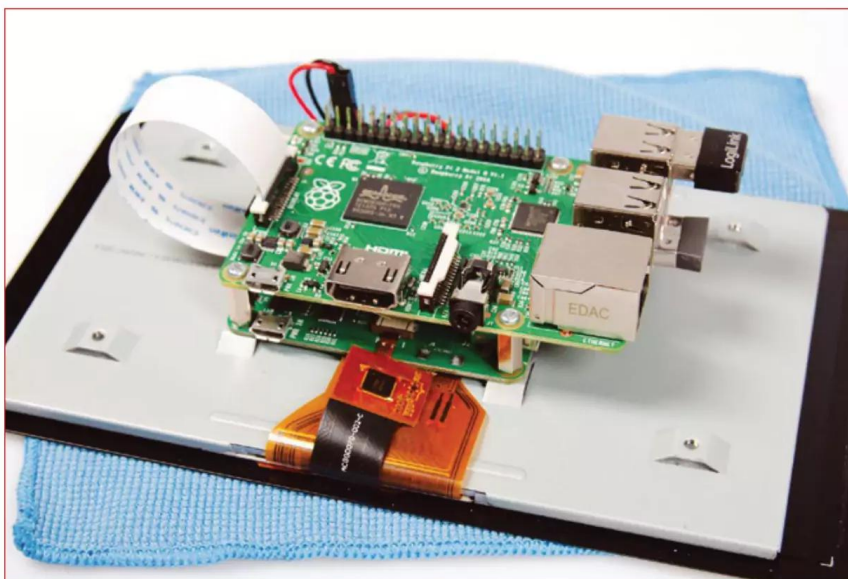


Figure 3: You can attach the Rasp Pi directly onto the screen board.

LISTING 1: Update the System

```
$ sudo apt-get update
$ sudo apt-get dist-upgrade
$ sudo reboot
```

LISTING 2: Virtual Keyboard

```
$ sudo apt-get install matchbox-keyboard
$ matchbox-keyboard &
```

LISTING 3: Second Screen

```
$ omxplayer --display=5 beispiel.avi
```


supply the additional 2.25 Watts needed to operate the screen.

As it turns out, OpenELEC and OSMC are still overtaxed by the screen. Both manage to put their images onto the screen, but with some distortion. Moreover, the current version of the Kodi media center does not yet react to touch entries. The latest changes to the OSMC kernel [4] indicate that these problems might soon be a thing of the past. In the future, it should be possible to build a portable media center with a touch-optimized skin like Rapier [5] or Re-Touched [6].

If you want to use the Rasp Pi together with a touchscreen display as a PC in your car, you will need to overcome a number of challenges. The bright display is easy to read even in broad daylight. With its 70-degree angle of vision, the screen is also clearly readable from the side. However, the display is too big due to a large edge.

Altogether it measures 194x110mm, and a double DIN slot only has enough space for a device measuring 180x100mm. It seems appealing to think about cutting off the part of the glass edge that extends past the screen with a glass cutter. However, Clive Beale from the Raspberry Pi Foundation assures us that this procedure only works in rare cases [7], and of course, the user would lose any product guarantee.

Conclusion

The touchscreen display from the Raspberry Pi Foundation is suitable for those hobbyists who don't want to connect a big monitor to the mini computer. The screen is an especially good match for all projects that need a display and simple data entry capabilities. For instance, you could use the Rasp Pi display for a weather station, an FHEM front end, or an SIP door phone system. However, you might need additional software. Genuine

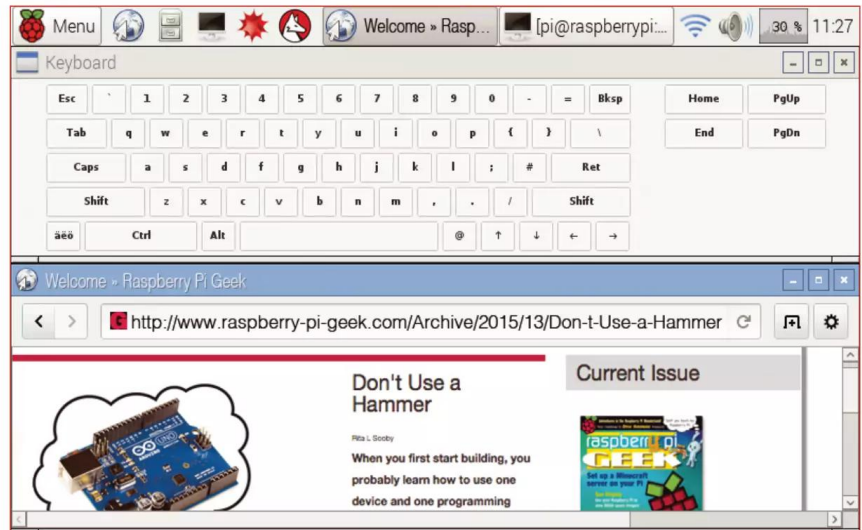


Figure 4: The 800x480 resolution of the RasPi touchscreen is sufficient for most projects. The Matchbox keyboard from the package sources even gives the user a virtual keyboard.

dual-screen operation is not possible yet under Raspbian, and the typical Linux systems still struggle with touch gestures. Toolkits like Kivy [8], which you can also install on the Raspberry Pi, help developers create touch-friendly applications quickly.

The Raspberry Pi Foundation is also thinking about additional display screens. Raspberry Pi founder Eben Upton has mentioned a possible future 10-inch screen with full HD resolution. If a 10-inch screen ever appears, the Rasp Pi will work as a full-fledged mobile tablet.

The foundation has no official plans for a case that will hold both the Raspberry Pi and the display screen, but some members of the community have stepped up with ideas. For example, MCM Electronics, the US branch of Farnell element14, has released CAD data for a suitable 3D-printed cover. Additionally, Pi3g, where we got our test touchscreen, is already developing a special case for the new Raspberry Pi display. ●●●

INFO

- [1] Raspberry Pi 7-Inch Touchscreen Display: <http://swag.raspberrypi.org/products/raspberry-pi-7-inch-touchscreen-display>
- [2] "The eagerly awaited Raspberry Pi Display": <https://www.raspberrypi.org/blog/the-eagerly-awaited-raspberry-pi-display/>
- [3] Portable VCR Raspberry Pi Media Centre: <http://www.instructables.com/id/1981-Portable-VCR-Raspberry-Pi-Media-Centre/>
- [4] OSMC Commits Support: https://twitter.com/OSMC_Commits/status/641276058289971201
- [5] Kodi-Skin Rapier: <http://kodi.wiki/view/Add-on:Rapier>
- [6] Kodi Skin Re-Touched: <http://kodi.wiki/view/Add-on:Re-Touched>
- [7] "How not to hack the Raspberry Pi Display": <https://www.raspberrypi.org/blog/dont-try-this-at-home-how-not-to-hack-the-raspberry-pi-display>
- [8] Kivy: <http://kivy.org>

Emulating game console classics on the Raspberry Pi

Old Heroes

The Raspberry Pi is particularly well suited for playing classic games, and a pair of promising Rasp Pi distros come preconfigured with several built-in game emulators. *By Marko Dragicevic*



Twenty-five years ago, playing a video game meant attaching a small box to the family television in the living room. This box was either a game console or a small home computer. For control, you used a joystick or a gamepad. Now you can recreate those classic gaming moments using the Raspberry Pi and a specialized retro gaming Linux distribution.

PIPLAY

A favorite SD image for Raspberry Pi retro gamers is PiPlay (formerly known as PiMAME) [1]. PiPlay is a Raspbian-based Raspberry Pi distro preconfigured with emulators for a number of the most popular retro gaming platforms. (See the box titled “PiPlay Emulators” for a list of the emulators supported by PiPlay.) Although you could theoretically set up these emulators on Raspbian – or almost any other Linux distribution – PiPlay saves you several configuration steps and even displays the emulators in a menu for easy access.

After bootup, the emulators appear in a small text menu, which you can navigate using the cursor keys (Figure 1). If you are

playing on a TV with an HDMI connection, you should have no problems with the display of the text menu. However, many retro fans prefer to play '80s and '90s era console games on old CRT televisions. Thanks to the RCA video and audio analog outputs present on Pi 1 Rev 2 B systems, it is possible to connect to an old system [2].

PIPLAY EMULATORS

According to the project website, PiPlay emulates the following legacy platforms.

- * MAME – AdvanceMAME and MAME4ALL
- * CPS I/CPS II – FinalBurn Alpha
- * Neo Geo – GNGeo
- * Playstation – PCSX-ReARMed
- * Genesis – DGen
- * SNES – SNES9x
- * NES – AdvMESS
- * Game Boy – Gearboy
- * Game Boy Advance – GPSP
- * ScummVM
- * Atari 2600 – Stella
- * Cave Story – NXEngine
- * Commodore 64 – VICE

On old TVs, because of interlacing and softer transitions, the picture appears more true to the original than the needle-sharp, digital representation. On digital TVs, the picture might look unappealing when you display pixelated graphics. If you have decided to use an old CRT television for display, you will need to use a set that's not too small. Otherwise, the text menu of PiPlay might be too difficult to read.

PiPlay's emulator menu includes PlayStation 1, SNES, Game Boy, Game Boy Advance, Atari 2600, and Neo Geo. Additionally, the ScummVM emulator is part of the set (see the box titled "ScummVM Is In"). By using MAME4All and AdvanceMAME on the Pi, you can bring many video games from 1980s arcades back to life.

Most of the emulators are for game console systems. The only emulator from the home PC era is the C64. (PC games were recent enough, and hardware-intensive enough, that you might not achieve optimum performance emulating a PC on the Raspberry Pi.) By contrast, the console emulators supported by PiPlay showed no speed problems. The classics I tested all ran fluidly.

WHERE ARE THE GAMES?

PiPlay specializes in emulators – not in games. Many of the games played on these legacy systems are closed source, and the vendors still maintain control of the licensing. With ScummVM, you will only need the files that were on the game media.

With the C64 and the emulator consoles, you will need to have the image files of the



Figure 1: PiPlay is limited to a simple, but nonetheless well-structured, text menu.

game media. These image files are often simply referred to as ROM files, even though, technically, this term only applies to consoles with cartridges.

You should not use the ROM files available over the Internet when you don't own the original cartridge. Strictly speaking, even if you own the cartridge, you will be operating in a legal gray area. However, you will probably not need to worry about any problems in practice, because you have legally acquired the user rights for the game.

You have several options for copying game files onto the Pi. For example, because PiPlay is based on a modified Raspbian image, you can start `raspi-config`, activate the SSH server, and use `sftp` to upload the files over your home LAN to the mini computer. Another method is to use `startx` to call up the desktop inside the PiPlay console, then start a browser so you can download the games from a cloud server, where you might have deposited them earlier.

The games for some of the older consoles are often only a few kilobytes in size. Therefore, even with a slow Internet connection, you can transfer many of them onto the Raspberry Pi very quickly. However, for the

SCUMMVM IS IN

Fans of graphical adventure games will be pleased to know that ScummVM is part of the PiPlay package (Figure 2). ScummVM exploits the fact that, back in the day, game developers like LucasArts or Sierra did not program their games in a machine-oriented language, but instead used internally developed interpreted languages.

Open source programmers have used reverse engineering to uncover the structure of the coded interpreter files and developed ScummVM as their own interpreter for executing the files. For example, if you want to use an MS-DOS graphical adventure from the olden days, you will not need to emulate the complete DOS-PC, which would be too slow for complex games. Instead, you can simply copy the game files to the ScummVM subdirectory, and ScummVM will provide you with native and therefore pleasantly smooth playback.



Figure 2: Experience the everyday life of the police with ScummVM and the adventure game Police Quest.

PlayStation and graphical adventure games for ScummVM, the situation might be different. A single title can easily take up 500MB of space, and even more when continuous voice output is involved. When copying such large amounts of data to the Raspberry Pi, it is a good idea to use an external hard drive or a USB stick.

You only need to copy the ROM files once onto the Pi. Thereafter, the process is uncomplicated and quite smooth. Simply boot the SD card, select the desired console (or C64 or ScummVM) in the menu, and start play. Inside the individual emulators, you can normally access the internal menu with one of the F keys. Via the internal menu, you can end the program and return to the game console selection menu.

RETROPIE

The RetroPie project [3] takes an approach similar to PiPlay. RetroPie is an SD card

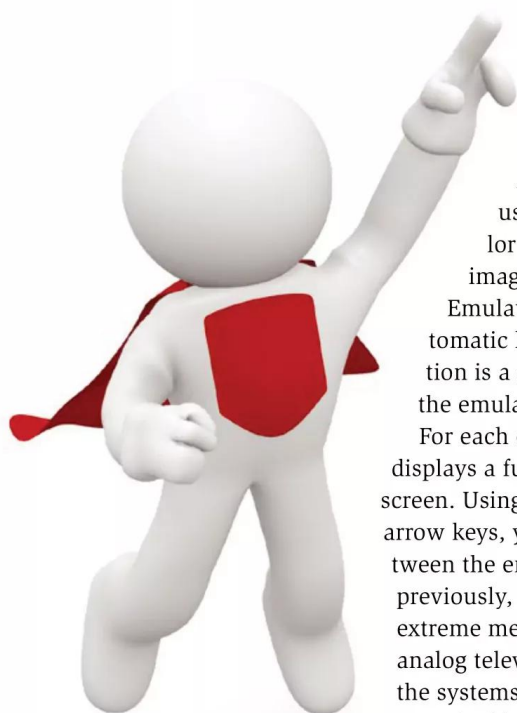


image you can use to boot your Pi, then you can start pre-configured emulators with a mere push of the button.

Alternatively, you can use a setup script to tailor the environment. The image starts the program

EmulationStation after an automatic login. EmulationStation is a graphical launcher for the emulators.

For each emulator, the program displays a full-screen splash screen. Using the left and right arrow keys, you can switch between the emulators. As indicated previously, if you have taken the extreme measure of using a small analog television, the names of the systems will be more easily recognizable. In all other cases,

this type of presentation is actually annoying.

RetroPie shows you a list that allows you to see all the available systems at a glance. EmulationStation, however, displays only those systems for which games can be found in the ROM directories. If you want to emulate not more than three consoles, you will be able to easily deal with the layout. The launcher lets you select the desired game even before the emulator is started.

RetroPie contains a few emulators not found on PiPlay (Figure 3), such as a Sinclair



Figure 3: The Pi can even emulate the famous Breadbox.

ZX Spectrum emulator as well as the MS-DOS emulator DOSBox. The biggest appeal of the RetroPie project is the description on the homepage showing how to connect the original gamepads of old game consoles to the Rasp Pi [4].

Theoretically, you could plug every joystick and every joystick into the Raspberry Pi with a USB connector; however, each controller would need its own USB port. The additional power draw might cause problems with the power supply. You could avoid possible power issues by connecting an active USB hub; however, the proliferation of USB connections might detract from the retro gaming experience.

Consequently, the approach taken by the RetroPie project for connecting original gamepads of old consoles to the GPIO pins of the Pi is particularly appealing. On eBay, an original joystick for the Super Nintendo costs on average only five Euros. For the complete SNES console, you often pay ten times as much; and, depending on the condition and equipment, the cost could be even higher. To connect the gamepad, you can wire the gamepad directly to the appropriate pins of the Raspberry Pi according to the directions on the homepage.

RetroPie also offers an adapter board for purchase that simplifies the configuration process and utilizes protective wiring. In terms of support software, a small C program runs in the background on the Pi, polls the GPIO pins, and passes the control actions on to the emulators.

LEVEL UP

If all these options do not meet your burning retro-gaming needs, you can find dozens of projects online devoted to the additional task of designing a retro housing for the Pi. These designs range from repurposed console cartridges to replicas of arcade machines, all with the Pi on the inside. You can build these designs by hand or with the help of a 3D printer. ●●●

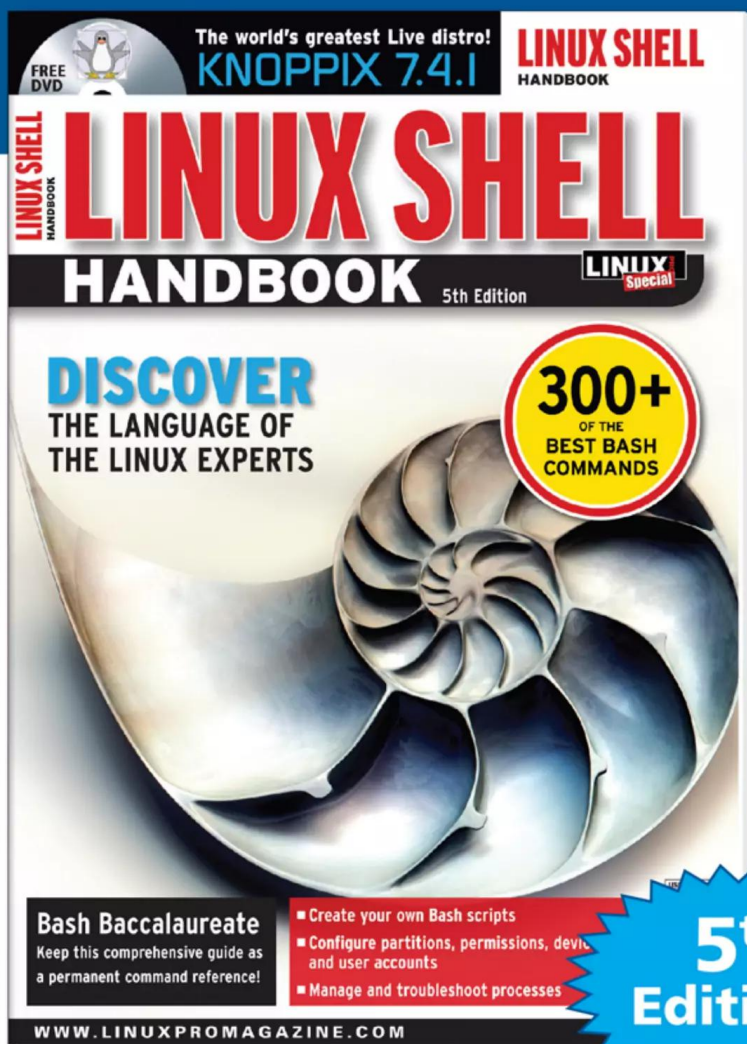
INFO

- [1] PiPlay: <http://pimame.org>
- [2] RetroPie: <http://www.raspberrypi.org/phpBB3/viewtopic.php?f=2&t=4991>
- [3] RetroPie: <http://blog.petrockblock.com/retropie/>
- [4] Attaching a SNES-controller to the Pi: <http://blog.petrockblock.com/2012/07/03/>

Shop the Shop

shop.linuxnewmedia.com

EXPERT TOUCH



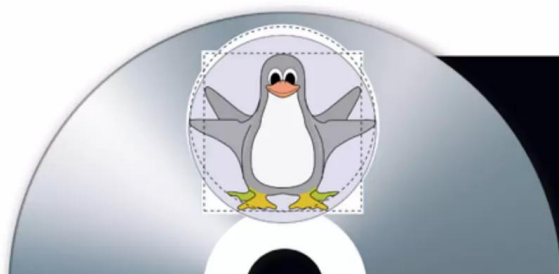
Linux professionals stay productive at the Bash command line – and you can too!

The Linux Shell special edition provides hands-on, how-to discussions of more than 300 command-line utilities for networking, troubleshooting, configuring, and managing Linux systems. Let this comprehensive reference be your guide for building a deeper understanding of the Linux shell environment.

You'll learn how to:

- Filter and isolate text
- Install software from the command line
- Monitor and manage processes
- Configure devices, disks, filesystems, and user accounts
- Troubleshoot network connections
- Schedule recurring tasks
- Create simple Bash scripts to save time and extend your environment

The best way to stay in touch with your system is through the fast, versatile, and powerful Bash shell. Keep this handy command reference close to your desk, and learn to work like the experts.



FREE DVD INSIDE!

The world's greatest Live distro!

KNOPPIX 7.4.1

ORDER ONLINE:

shop.linuxnewmedia.com/specials

Working with the Raspberry Pi camera modules

Snapshot

The amazing Raspberry Pi camera and Pi NoIR camera modules open a whole new world of useful projects. We'll show you how to take your first photos with the Pi cameras. *By Aaron Shaw and Bernhard Bablok*



The camera module, designed specifically for the Rasp Pi boards, brought with it a fair number of features that make it superior to many sophisticated webcams on the market (see the box titled “Camera Specs”).

It's easy to see why this camera is now the preferred choice for almost every application that requires a dedicated camera for still photography or video capture with the Rasp Pi.

This article will guide you through the process of setting up and using the Rasp Pi camera module before looking at the Pi NoIR.

SETTING UP THE CAMERA

To attach the camera module to the Raspberry Pi, first make sure your Rasp Pi is completely disconnected from any power source. Before removing the camera module from its grey anti-static bag, make sure you touch a radiator, a tap, or other grounded metal object to get rid of any static that you might have built up. (The camera module is fairly sensitive, and static electricity can actually damage it.)

The camera board CSI connector is located between the Ethernet and HDMI ports on the Rasp Pi Model B. The Ethernet port does not exist on a Model A board, but the camera connector is in the same place. On the Model B+ and Rasp Pi 2, the connector is by the HDMI port and clearly marked “CAMERA.”

Open the connector by pulling the tabs on either side upward and away from the HDMI port. You should then be able to insert the ribbon cable on the camera module (with silver connectors facing toward the HDMI port) into the connector. Hold the cable firmly in place and close the CSI connector by pushing the top part toward the HDMI port and down. It should click into place nicely and hold the camera ribbon cable securely. Take care not to bend the ribbon cable at too

The Raspberry Pi has seen several fantastic improvements since its original release in April 2012. In May 2013, the Raspberry Pi Foundation officially released its first add-on board: the Rasp Pi camera module (Figure 1). In October 2013, the Foundation announced a camera module without an infrared filter, the Pi NoIR (NoIR = No InfraRed.)

Before the camera module, you could, of course, access a camera feed on the Rasp Pi using a suitable webcam. Camera functionality had already worked its way into a variety of projects, from live weather monitoring to robotics. However, the clever folks at the Raspberry Pi Foundation were also aware of the fact that, if the highly enthusiastic Rasp Pi community would repurpose their existing webcams, an add-on camera module stood a strong chance of success.

This article was written in partnership with The MagPi magazine: www.themagpi.com



Figure 1: Raspberry Pi's powerful camera module.

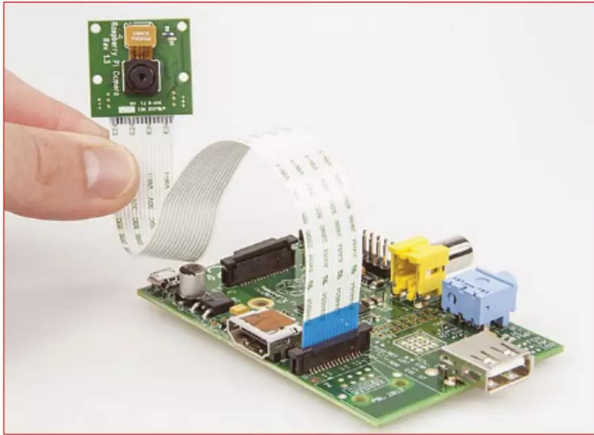


Figure 2: The camera module connected to a Model A Raspberry Pi.

much of an angle. Figure 2 shows a picture of the camera module connected to the Rasp Pi Model A.

Next, you need to power up the Rasp Pi and enable the camera module. After logging in to Raspbian, start the `raspi-config` program with:

```
$ sudo raspi-config
```

You will see a screen similar to Figure 3.

Use the Down arrow to move to the *Enable camera* option, then hit Enter. The following screens ask you to confirm and reboot; then, the camera module will be enabled.

BASIC OPERATION

Now that you have the camera module enabled, taking pictures and videos is quite simple and can be done either from the terminal or within a program.

The basic commands are `raspivid` and `raspistill`, which (as the names suggest) are for capturing video and still images, respectively. Each of these commands can accept a number of parameters after them. Several parameters are available, but the most basic are:

- `-o` or `-output` sets the output file name;
- `-t` or `-timeout` sets the time in milliseconds and displays a preview. (The default is five seconds, and when using `raspistill`, it captures the last frame of the preview and saves it to the specified file name. When using `raspivid`, the `-t` parameter defines the capture time.)

To display a five-second preview and save a JPEG picture called `test.jpeg`, enter:

```
$ raspistill -o test.jpeg
```

To take a 30-second video and save it in h264 format, use the command:

```
$ raspivid -o test.h264 -t 30000
```

For a list of other parameters, type either of the following commands:

```
$ raspistill | less
$ raspivid | less
```

Play with all of these settings and see how they change the picture (Figure 4).

You will find that the little camera is quite powerful. If you want to take a video that cycles through all the available image modes, you can append `-d` or `-demo` to the end of the `raspivid` command. Additionally, check out the Raspberry Pi Foundation's camera documentation for an in-depth look at all of the functionality available from the camera module [1].

THE CAMERA AND PYTHON

You can often save time and energy by automating commands within a program. The camera board is not supported by an official Python module from the Raspberry Pi Foundation, however, an unofficial Python library has been created for camera support.

Possibly the simplest way to automate the camera (using a Python library that will already be present on your Rasp Pi) is to use the `call` function within the `subprocess` library. You can add that function to a Python program as follows (from `subprocess` import `call`):

```
call(["raspistill -o test.jpeg"], shell=True)
call(["raspivid -o test.h264 -t 30000"],
     shell=True)
```

PURE PYTHON LIBRARY

A pure Python library for the camera module has been developed by Dave Jones, an enthusiastic member of the Rasp Pi community,

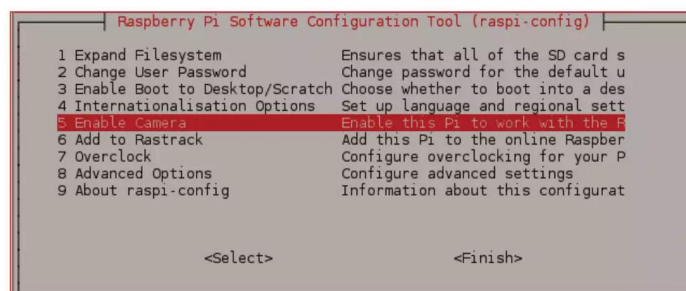


Figure 3: Enabling the camera module.

CAMERA SPECS

- Omnivision 5647 CMOS Sensor in a fixed-focus package.
- Capable of 5MP (2592x1944 pixels) images and video recording in three modes – 1080p at 30fps, 720p at 60fps, and 640x480 at 60 or 90 fps.
- Connection using a 15-pin ribbon cable via the CSI connector. CSI stands for Camera Serial Interface, which is especially for use with cameras.
- Very small form factor and weight – at just 25mm x 20mm x 9mm and 3g.
- Native support in Raspbian, the Raspberry Pi Foundation's preferred OS.
- Lots of adjustments available – exposure, white balance, image effects, metering modes, and many more.
- All for a price of \$30.00 or just over £16!



Figure 4: From top left to right: cartoon, sketch, solarize, negative, emboss, and colorswap.

to make it easier and neater to use the Rasp Pi camera module within Python programs. The new library was just released on September 24, 2013, and as such, it is in early stages of development. This means that the Python library has not been released in a .deb package yet, so you can't install it using the standard apt-get method. However, it is still fairly easy to install; just type:

```
$ sudo apt-get install python-setuptools
$ easy_install --user picamera
```

These commands will install the Picamera library for the current user only, not for the entire system. Installing the library for the current user not only makes it easier to uninstall later, it also helps you avoid complications later if a full system install occurs using the apt-get method (when a suitable package becomes available).

Once installed, the interface is very Python like, and therefore intuitive and easy to use. For example, to start a preview for 10 seconds with the default settings:

```
import time
import picamera
camera = picamera.PiCamera()
try:
    camera.start_preview()
    time.sleep(10)
    camera.stop_preview()
finally:
    camera.close()
```

You'll find more information on how to use the Python library online [2].

TIME-LAPSE PHOTOGRAPHY

Time-lapse photography – taking pictures at regular intervals and stitching them together

into a video – is a fantastic way to make use of the Rasp Pi camera is. It is really easy to do with the -t1 parameter. For example, to take a picture every 10 seconds for an hour, type:

```
$ raspistill -o test_%03d.jpeg
-t1 10000 -t 3600000
```

Executing this command from within a directory (e.g., /home/pi/timelapse or similar) that contains only the time-lapse photos will make the next step much easier.

The preceding command will generate 360 images. The %03d allocates three decimal places for the number in the file name. In this case, the file names start with test_001.jpeg and end with test_360.jpeg. If you are working over a longer period, you would probably need to increase this to %04d or %05d to accommodate a four- or five-digit number in the file name.

STITCHING

To stitch the time-lapse images into a video, you need mencoder:

```
$ sudo apt-get install mencoder
```

Start in the /home/pi/timelapse directory (or wherever you saved all the pictures) and type

```
$ ls *.jpeg > timelapse.txt
```

to make a list of files to pass to mencoder. Next, create an 1080p HD video of the images:

```
$mencoder -nosound -ovc lavc -lavcopts
vcodec=mpeg4:aspect=16/9:vbitrate=8000000
-vf scale=1920:1080 -o timelapse.avi
-mftype=jpeg:fps=24 mf://@timelapse.txt
```

To change the name of the output file, change timelapse.avi to another name. Once you get to know the time-lapse photography option, think of all the creative videos you can make! You also could set up a battery-powered, time-lapse camera rig using a model A Rasp Pi, a camera module, and a large battery. Add a solar panel and place it in a remote location, and you could be snapping for days on end.

PI NoIR

The unusual tonal values found in infrared photography offer a new perspective on the world. So-called IR cameras are costly, and converting a normal camera to function as an IR camera is expensive. The NoIR camera

THE AUTHORS

Aaron Shaw is a volunteer at *The MagPi* magazine (www.themagpi.com), a magazine dedicated to Raspberry Pi users. He also founded Pi Supply (www.pi-supply.com) with a Kickstarter-funded add-on board for the Raspberry Pi (<http://kck.st/UVBXTE>). For help relating to this article, you can contact him through sales@pi-supply.com or via Twitter ([@shawaj2](https://twitter.com/shawaj2)).

Bernhard Bablok works in Managed & Operations Services at Allianz SE in Munich, Germany, as an SAP HR developer. He enjoys music, cycling, and hiking and also working with Linux, programming, and nanocomputers. mail@bablokb.de

module for the Rasp Pi [3] lets you build your own infrared camera.

THEORY

Infrared light sits on the electromagnetic spectrum [4] right next to what humans call visible light. Depending on the source, visible light lies between 380 and 700 nanometers (nm). Infrared light occupies the region above and up to 1,400nm. This form of light remains invisible to the human eye, but photo diodes on the sensors of a camera are sensitive to it. As a result, electronics can make infrared light visible to humans.

Every wavelength has a different refraction index. Lenses made from specialty glasses correct these indexes so that images can be captured in focus. This correction process is easier when the image includes only visible light. Moreover, infrared light that has been made visible on an image leads to unusual shifts in color.

Camera manufacturers therefore put a band elimination filter in front of the sensor to prevent infrared light from coming through. The typical camera can only be used in a limited fashion for infrared photography. An additional filter layer, the Bayer filter, is built in alongside the IR band elimination filters. Consisting of red, blue, and green fields, this additional layer turns the light sensors that would otherwise be color blind into color-sensitive sensors. The camera or the RAW converter mixes the color values of the pixels by interpolating the values.

If the pixels of a Bayer filter were porous only with respect to colors, there would be no need for an IR band elimination filter, and there would be no IR photography. However, a Bayer filter lets light pass and overlap in different places along the spectrum depending on the color. As a result, infrared images appear more or less colorful in accordance with the porousness of the color pixels.

A DIFFERENT IR FILTER

Although the NoIR module for the Rasp Pi does have a Bayer filter, it does not have one for infrared. The wavelength range that reaches the sensor is therefore significantly larger. The fact that the range is larger means that photographs captured by the sensor turn out with too many red tones (Figure 5, left). Apparently, the red sensor pixels are the most sensitive in the IR range. Only the infrared range of the electromagnetic spectrum plays a role in IR photography; thus, the camera needs an additional band elimination filter capable of filtering out normal light.

Specialty stores carry these filters, often labeling them wrongly as *infrared filters*. In fact, it is actually the exact opposite. The wrongly labeled filter blocks everything *outside* of the IR range. Filters come in several versions (e.g., 715, 720, 830nm); the number equals the maximum wavelength at which the filter will block light. A lower number means that a greater amount of visible light from the upper wavelength range comes through. Theoretically the 7xx filters can be used for fake color infrared photography. The 830nm filter on the other hand only lets information regarding light intensity pass through. Here, the color pixels of the Bayer filter let almost the same amount of light pass through.

USING THE PI NOIR

The Picamera library you installed for the Rasp Pi camera module also powers the Pi NoIR, and its commands are used in the same way. The Picamera website provides a number of useful recipes [5] to help get you started using your camera modules.

CONCLUSION

We hope this introduction will inspire you to create some awesome projects using your Rasp Pi and its camera modules. Whether you have ambitions to create a time-lapse camera set up, take aerial pictures of your house, or indulge in infrared photography, the capability of the Rasp Pi camera module is truly astounding.

What project wouldn't be cooler with a camera?

...

INFO

- [1] RaspiCamDocs: https://github.com/raspberrypi/userland/blob/master/host_applications/linux/apps/raspicam/RaspiCamDocs.odt
- [2] Picamera documentation: <http://picamera.readthedocs.org>
- [3] NoIR camera module: <http://www.raspberrypi.org/products/pi-noir-camera/>
- [4] Electromagnetic spectrum: http://en.wikipedia.org/wiki/Electromagnetic_spectrum#Range_of_the_spectrum
- [5] Picamera recipes: <http://picamera.readthedocs.org/en/release-1.10/recipes1.html>



Figure 5: The photograph has far too many red tones (left) because the NoIR module lacks a band elimination filter. A 720nm filter blocks the normal light (right), so the camera captures only the infrared range.

Advanced tricks for working with the GPIO interface

Out and In

By Richard Ryniker



After a few initial experiments operating LEDs and reading switches on a Raspberry Pi, the “It works!” euphoria fades, and astute users understand they will face problems when they extend those simple programs to more complex environments.

In this article, I’ll explore some techniques you’ll need when you start to deploy the general-purpose input/output (GPIO) interface in more advanced scenarios. You’ll learn about the general Linux facility for accessing the GPIO programmatically, as well as how to export a GPIO so it is available to a program running without root privileges. Also, you’ll get a look at how to use interrupts to replace wasteful status check loops.

General Purpose GPIO

Linux has a facility for managing GPIO resources. This GPIO system creates a convenient interface for user programs, protects GPIO resources used by device drivers such as I2C and SPI, and delivers pin-specific access so one application does not need to worry about what other programs do with other GPIO pins. This individual pin inter-

face is important because, without it, every GPIO application would have to worry about race conditions with other applications that share a bank of GPIO pins (which means that locks, interrupt management, and other complexities would be needed).

The Linux GPIO facility uses files in the directory `/sys/class/gpio/`, which like many system configuration or control files, are owned by root. I shall ignore this for now, but later I present a tool that encapsulates the privileged operation needed to use this file in a responsible way.

Shell programs often use the `echo` command to send messages to standard output or, with redirection, to a file. The simple command

```
echo Hello there.
```

writes the output *Hello there*. With output redirection,

```
echo Hello there. >file_01
```

creates the file `file_01` that contains the same message.

Applications running on your Raspberry Pi system can communicate with the GPIO by writing to special files that the system uses to pass information to and from the hardware. The `echo` command is sometimes useful for writing this information.

For instance, consider pin 23, which is convenient and easily accessible at terminal 16 of the 26-terminal Raspberry Pi header (Figure 1). (See also the “Other Raspberry Pis” box.)

To create a user interface for pin 23, use:

```
sudo echo 23 >/sys/class/gpio/export
```

This article originally appeared in The MagPi magazine: www.themagpi.com



Lead image © yanhongbin, 123RF.com

This command causes the kernel to create a directory named `/sys/class/gpio/gpio23` that contains four files relevant to this discussion: `active_low`, `direction`, `edge`, and `value`. The initial values in these files (if no external connection to this pin) are:

```
active_low      0
direction      in
edge           none
value          0
```

These files hold information that passes to pin 23. For instance, to make pin 23 an output pin (continue to use `sudo`), enter:

```
echo out >/sys/class/gpio/gpio23/direction
```

If the output value should be initialized first, before the output driver is enabled, one of the following commands may be used to set the pin direction with an initial value:

```
echo low >/sys/class/gpio/gpio23/direction
echo high >/sys/class/gpio/gpio23/direction
```

To set the pin output on or off, use:

```
echo 1 >/sys/class/gpio/gpio23/value
echo 0 >/sys/class/gpio/gpio23/value
```

To invert the pin logic, use:

```
echo 1 >/sys/class/gpio/gpio23/active_low
```

Do this before reading an input or setting an output value. When `active_low` is 1 (or anything other than 0) and `value` is set to 1, the pin is driven low, and so on.

How fast can this mechanism change GPIO pin values? The simple Python program shown in Listing 1 will generate pulses at 19kHz. If this is written in C [2] the frequency increases to roughly 120kHz. The actual frequency varies because the Raspberry Pi does other things that temporarily suspend the loop – clock maintenance, network activity, and other user and system processes.

GPIO Without Root

A frightful number of “Run this program as root” instructions has been published for the Raspberry Pi user. To an experienced user, this advice sounds like “Here,

OTHER RASPBERRY PIS

The pinout illustration in Figure 1 displays the header information for the Raspberry Pi revision (Rev) 2. The Rev 1 models differ in pins 3 and 5 [1]. All boards produced since Fall 2012 are Rev 2 boards, so unless you have a very early model, the chances are your Pi is Rev 2.

As Figure 1 illustrates, Rasp Pis from the Model B+ forward use a 40-pin GPIO, but the first 26 pins are the same as those for earlier Rev 2 boards. On the expanded GPIO, two of the new pins provide an ID EEPROM interface, and the rest are general I/Os with a scattering of ground pins.

LISTING 1: gpio23-max.py

```
01 #!/usr/bin/python
02
03 # Toggle GPIO pin 23.
04 # Output pulse frequency is about 18 kHz (varies
    due to other activity).
05
06
07 pin_path = '/sys/class/gpio/gpio23'
08
09 def write_once(path, value):
10     f = open(path, 'w')
11     f.write(value)
12     f.close()
13     return
14
15
16 # Set pin for output, with initial value low.
17 write_once(pin_path + '/direction', 'out\n')
18
19 f = open(pin_path + '/value', 'w')
20
21 # Blink as fast as possible...
22
23 while 1:
24     f.write('1')
25     f.flush()
26
27     f.write('0')
28     f.flush()
29
30 f.close()
```

Figure 1: Pin GPIO23 is easily accessible at terminal 16 on the Raspberry Pi.

children, these are razor blades. Take them outside, and see what you can cut with them.”

Root privilege should be viewed as a last resort. The proper use of root privilege is system creation and configuration – the establishment of a protected environment, where faults in one program will not affect other applications and cannot cause failure of the operating system.

Root privilege makes it easy to interfere with system activities. If one is lucky, the result is immediate panic, and the system crashes. In less fortunate circumstances, malicious software could be installed in a system. Communication with the GPIO requires root privilege, so it is a good idea to be very careful.

The C program `gpio_control.c`, which you will find online [3], executes operations that require root privilege to export a GPIO pin for use by ordinary users. Comments at the beginning of the program describe how to compile and install it.

Once the program is installed (by root), because of its setuid characteristic, when a user invokes it, the program runs with an effective userid of root, and therefore it has the privilege needed to export or unexport a GPIO pin and set appropriate permissions for the files used to control that pin.

Programs that execute with root privilege should be written by really paranoid programmers. Most of the code in `gpio_control.c` simply checks that the arguments are reasonable and tries to be informative if anything unusual happens.

To use `gpio_control` to export pin 23 so all of the pin manipulations discussed earlier do not require root privilege, simply execute:

```
gpio_control 23 export
```

`gpio_control.c` can be configured easily, before it is compiled to allow GPIO access to all users or only users in the caller’s group. Each of the GPIO pins can be individually configured to permit or forbid export.

The Raspberry Pi uses GPIO pin 16 to control the *Status OK* green LED. If one tries to export GPIO pin 16, the operation fails because the kernel is using this resource:

```
$ gpio_control 16 export
export failed: Device or resource busy
```

Other kernel programs may acquire GPIO pins, which can make them unavailable to users. This is good. Little harm could come

from a user turning the status LED on and off, but what about the kernel I2C driver? It could easily suffer erratic failures if the pins it uses are changed in ways it cannot understand. (See also the “More Pins” box.)

The kernel remembers the state of GPIO pins. For example, suppose a pin is exported, set by the user as an output pin, then unexported. The userspace files disappear, but the pin remains an output pin with the last value set. If this pin is again exported, the userspace files are recreated to manifest the saved state.

Handling Interrupts

Some of the simple programs described in this issue operate in an infinite loop, wherein a program repeatedly reads the value of an input signal and performs some operation when it changes.

The basic elements of GPIO control can be used to replace this rudimentary “infinite loop” operation with a vastly more efficient program that only runs when the input signal changes. With only one input, and absolutely nothing else to do, a loop might not be a problem; however, such a loop tries to consume 100 percent of the CPU resources and therefore competes aggressively with everything else that might want some piece of the Raspberry Pi.

One could introduce a delay in the poll loop, say a `sleep 0.5` command to delay one-half second before starting the next loop iteration. This delay allows other activity to run during the sleep period but means there is an average delay of one-quarter second before any change in the input is observed. Shorter delay, faster response, more wasted CPU, ... an ugly choice.

As the number of inputs grows and the number of responses to those inputs becomes larger and more varied, it often is necessary to manage tasks with different priorities. Interrupts are a fast way to connect an input (e.g., “There is a chasm directly in front of the vehicle”) to a response (“Stop!”).

Another Python program in Listing 2 illustrates GPIO interrupt handling. This program configures pin 23 as an input, sets the pin’s edge file to both so interrupts will occur for “falling” and “rising” transitions, then opens the pin’s value file.

Invocation of `select.poll()` creates a polling object `po`, then `po.register()` adds the GPIO pin’s value file as one of the sources that can satisfy a subsequent `po.poll()` request. This program uses only the one inter-

MORE PINS

Only a few GPIO pins are accessible on the Raspberry Pi, but several people have shown how inexpensive ICs such as MCP23017 can use the I2C interface to expand this number. A design such as the Slice of Pi/O [4] can be used up to eight times to add 128 digital I/O pins to a Raspberry Pi. The MCP23017 open drain interrupt configuration connects interrupt signals from multiple devices to a single GPIO pin. A pull-up resistor to 3V3 keeps the input high, until a connected device drives it low. When an interrupt occurs, the interrupt handler has to read values from all the interrupt-generating devices to learn which have active interrupt signals (possibly several), launch the appropriate response programs, then clear all the interrupt requests (so the GPIO input returns to the high state) to allow the next interrupt to occur.

rupt source, but other GPIO pins, and many other sources, can be registered with the poll object. For instance, a pipe that connects to another process or a socket that receives data over a network could be an interrupt source.

The second operand to `po.register` specifies which of three conditions will be recognized as interrupts. The `select.POLLPRI` value specifies that only “priority data to read” will cause an interrupt. The other possible conditions – “data available” and “ready for output” – are always true for a GPIO pin; therefore, a poll operation when either of these is enabled would always complete immediately. If other interrupt sources are registered with `po`, they might use these conditions.

Sometimes, the absence of an expected signal could be important. The `po.poll(60000)`

call will wait for an interrupt, but only for 60 seconds (60,000ms) before it returns an empty list of interrupt signals to indicate it timed out. The kernel maintains the value file for a GPIO pin with 2 bytes of content: A 0 or 1 character to represent the pin’s current value and a newline character. `f.seek(0)` resets the current location in the file to the beginning, so the value of the first character may be read again.

Conclusion

As you envision more sophisticated tasks for your Raspberry Pi, you’ll need an expanded collection of programming tricks. The techniques described in this article, such as accessing GPIO pins, exporting pins, and handling interrupts, will serve you well as you imagine complex, real-world uses for your Raspberry Pi.

...

LISTING 2: interrupt_test23.py

```
01 #!/usr/bin/python3
02
03 # Test interrupts.
04
05 import select, time, sys
06
07 pin_base = '/sys/class/gpio/gpio23/'
08
09 def write_once(path, value):
10     f = open(path, 'w')
11     f.write(value)
12     f.close()
13     return
14
15
16 f = open(pin_base + 'value', 'r')
17
18 write_once(pin_base + 'direction', 'in')
19 write_once(pin_base + 'edge', 'both')
20
21 po = select.poll()
22 po.register(f, select.POLLPRI)
23
24 state_last = f.read(1)
25 t1 = time.time()
26 sys.stdout.write('Initial pin value = {}\n'.format(repr(state_last)))
27
28 while 1:
29     events = po.poll(60000)
30     t2 = time.time()
31     f.seek(0)
32     state_last = f.read(1)
33     if len(events) == 0:
34         sys.stdout.write(' timeout delta = {:.4f} seconds\n'.format(t2 - t1))
35     else:
36         sys.stdout.write('value = {} delta={:.4f}\n'.format(state_last, t2 - t1))
37     t1 = t2
```

INFO

- [1] Raspberry Pi pinout: <http://www.raspberry-pi-geek.com/howto/GPIO-Pin-out-Rasp-Pi-1-Rev1-and-Rev2>
- [2] Generating pulses in C: <http://ryniker.ods.org/raspberrypi/MagPi/23-maxa.c>
- [3] Code for this article: ftp://ftp.linux-magazine.com/pub/listings/magazine/RaspPi_Handbook
- [4] Slice of Pi/O: <http://store.acmeun.com/products/slice-of-pi-o-kit.html>

Mathematica and the Wolfram language on Raspberry Pi

Mathematical Mind

Wolfram Mathematica and the powerful Wolfram language are now available for the Raspberry Pi. We'll help you get started with this fascinating tool for calculation, presentation, visualization, and general programming. *By Aaron Shaw*

Wolfram Mathematica is a computational software program created originally by the British scientist Stephen Wolfram and now developed by Wolfram Research (with Wolfram at the helm). Mathematica is mostly used in the fields of science, engineering, and mathematics, but it has a diverse range of capabilities that make it suitable for many other applications in other fields.

At the heart of Wolfram Mathematica is the Wolfram language, which has been around for more than 25 years in various iterations but only recently took the Wolfram name. Wolfram Research calls the Wolfram language a “revolutionary knowledge-based programming language” and claims it is “the world’s most productive programming language” – a very bold claim indeed! The language, which focuses heavily on symbolic computing, is very large because of the huge amount of built-in specialization that provides functions for everything – from differential equation solvers to 3D graphical analysis tools and much, much more.

The company also has a product they call Wolfram Alpha, which they describe as a “Computational Knowledge Engine.” You might not have heard of Wolfram Alpha, but if you spend any time with Internet technologies, you have probably made use of it: Wolfram Alpha is one of the answer engines that form part of Apple’s Siri and Microsoft’s Bing. If you feel like experimenting with

Wolfram Alpha, I recommend visiting the website [1] and having a play, because it is really useful: I have used Alpha many times at university to validate my work. (Unfortunately, they make you pay for certain advanced features, but the free version is still serviceable.)

WHY IS WOLFRAM SO SPECIAL?

Why use Mathematica and the Wolfram language when popular, mainstream languages like Python are already available for the Raspberry Pi? One of the main benefits is that Mathematica uses a multiparadigm input method, which accepts code in various forms – obviously Mathematica can run native Wolfram language code, but you can also build up code using a more graphical interface called “palettes,” and you can even type in plain English and let Mathematica translate the English statements into code using its own algorithms. These capabilities make Mathematica a flexible and intuitive experience for beginners and professionals alike. Another interesting feature of Mathematica is that it is laid out in what they call “notebooks,” which enables you to add notes and word-processed text to the same document in which you store your code. This integration of documents and images means you can use the Notebook feature to create impressive presentations in Mathematica.

Until recently, the Mathematica software package (and hence the Wolfram language) has been completely proprietary – costing upward of \$149 a year. Last November, Wolfram, who is a big supporter of Raspberry Pi and its goal of integrating computers with education, announced that he was developing a free version of Mathematica and the Wolfram language that would run

This article was written in partnership with The MagPi magazine: www.themagpi.com



on the Pi. The release of the Raspberry Pi Mathematica edition opens up the brilliant software package to a huge number of students, hobbyists, and professionals who probably would not run across the commercial version of Mathematica in their everyday life.

IF YOU NEED TO INSTALL

Mathematica now comes bundled with the standard Raspbian OS image available at the Raspberry Pi website [2]. If you downloaded Raspbian after November 21, 2013, or if you installed Raspbian through a NOOBS version [2] released after November 21, you should already have Wolfram Mathematica installed and available to you.

If you set up your SD card before this date, you'll need to download and install Mathematica. Be sure you have at least 600MB of free space on your SD card to install the software. To install the Wolfram engine in Raspbian, be sure you are connected to the Internet, then open an LXTerminal (command-line) session and type:

```
sudo apt-get update
sudo apt-get install wolfram-engine
```

Following the update, the Wolfram installation begins, and you will likely be asked whether you want to continue with the download and installation. Hit Y and then Enter and the package manager will install the Wolfram engine, as well as any other prerequisite software, on your Raspberry Pi. The installation can sometimes take a few minutes, especially if you have not updated your Raspberry Pi in a while. A blue screen pops up asking you to accept the Wolfram license agreement (Figure 1).

Use the Down arrow to scroll through the text. Press the Right arrow key to select OK and then hit Enter. Another blue screen asks you if you accept the license agreement. If you are happy to proceed, press the Left arrow key to select Yes and then hit Enter.

Once Wolfram and Mathematica are finished installing, you will find them in the app launcher under the *Programming* menu (Figure 2).

GETTING STARTED

With the Wolfram Engine on the Raspberry Pi, you effectively have two ways of programming with the Wolfram language. The command-line version of the Wolfram language lets you execute commands from an LXTerminal command-line session or from a

previously saved executable .m script. The alternative method is to use the Mathematica notebook interface, which is a word-processor-like environment with the added capability of really intuitive programming and interaction with the Wolfram language. This approach is similar to programming in other languages using an Integrated Development Environment (IDE). However, the word processor interface of Mathematica adds a whole new level of capability that is not present in a conventional IDE: You can actually calculate in real time within a notebook, effectively adding programming capability to a word processing environment.

If you want to run the command-line program (which is not necessarily the best place to start for a beginner), click on the Wolfram icon (the one that looks like a wolf) in the top panel. Alternatively, you can open an LXTerminal session, type

```
sudo wolfram
```

and hit Enter; you should see an interface in the LXTerminal window that looks like Figure 3.

If you would prefer to work from the Mathematica user interface (which I recommend for the exercises in this article), simply launch Mathematica from the top panel. You should then see a splash screen with the Mathematica “Spikey” logo (technically speaking, the shape is a hyperbolic dodecahedron with what Wolfram calls “embellishments”) while the program loads (Figure 4). Once the program fully loads, two screens appear – a blank notebook file and a help window (Figure 5).

If at any point you need or want to open a new notebook, click *File | New | Notebook*, or alternatively, you can just use the keyboard shortcut Ctrl + N.

You can also save your work in a notebook by using the keyboard shortcut Ctrl + S or by clicking on *File | Save*.

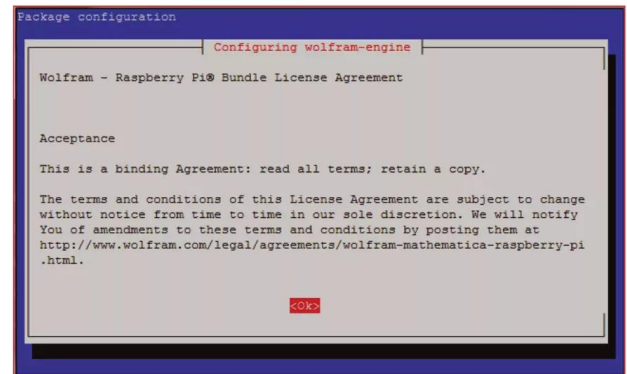


Figure 1: Viewing the Wolfram license.

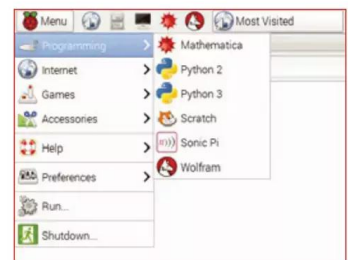


Figure 2: Look for Mathematica and the Wolfram language in the Menu.

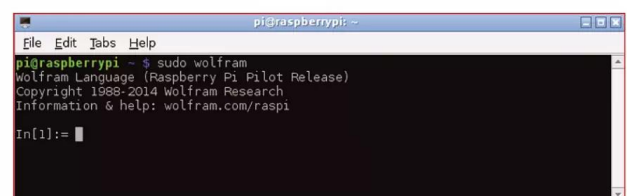


Figure 3: Starting up Wolfram in a terminal window.

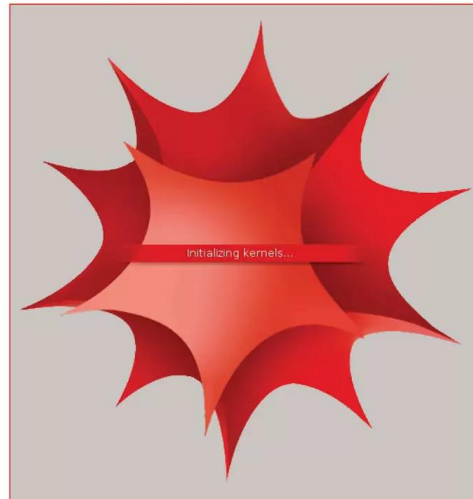


Figure 4: The Mathematica logo “Spikey” covers over the desktop while the program loads.

In this article, I focus on the Mathematica user interface; however, all the commands should work from the command-line Wolfram interface as well. Additionally, it is important to note that I focus on the actual Wolfram language programming aspect of the Mathematica interface, rather than the word processing and layout features also available in the Mathematica notebook interface.

Customarily every programming lesson begins by outputting the string *Hello world!* In Mathematica, you just have to type:

```
Print ["Hello world!"]
```

In the Mathematica GUI, you then press Shift + Enter to execute the command, whereas if you are on the command line, you just press Enter. The result should be:

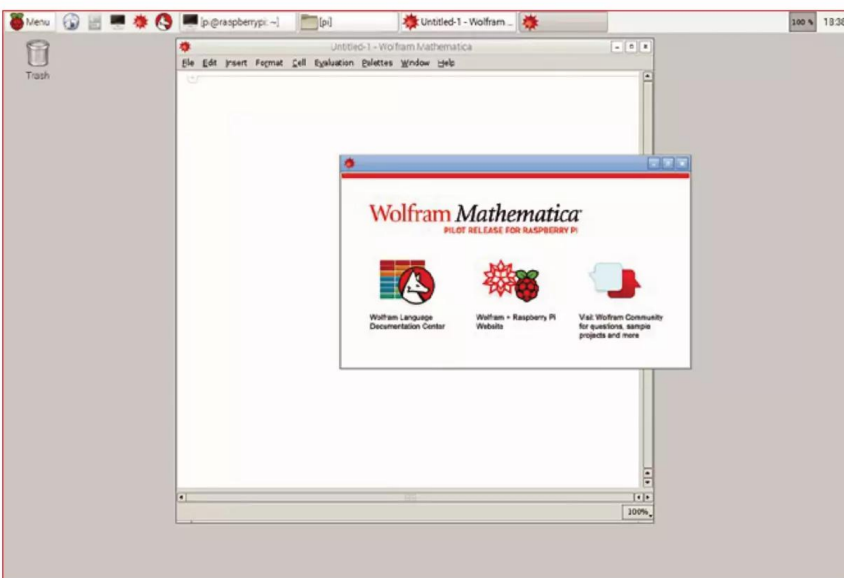


Figure 5: The Mathematica user interface.

```
Hello world!
```

Although this simple program is somewhat gimmicky, it does introduce three important aspects of the Wolfram language:

- Functions
- Arguments
- Strings

I'll explain these concepts in the following sections, along with other important Wolfram components, such as lists and variables.

FUNCTIONS

In the most basic terms, a function takes in an input, performs some kind of processing operation on it, and gives an output. For example, squaring or dividing a number is a function. The Wolfram language has more than 5,000 built-in functions, and it is even possible to create your own. In the Wolfram language, functions always start with a capital letter. In the preceding example, the function is `Print`.

Another example is the `N` function. Mathematica likes to keep numbers in the most exact form. For instance, when dividing in Mathematica, the software attempts either to divide the number perfectly into an integer value (i.e., $200/4 = 50$) or, if the result is not an integer, to keep the answer in fraction notation, converting it to the simplest possible form (i.e., $200/6 = 100/3$). However, sometimes you will want to know what the numerical version of a fraction is, and this is where the `N` function comes in. If you type

```
1/3
```

and press Shift + Enter, you get the output:

```
In[4]:= 1 / 3
```

```
Out[4]= 1 / 3
```

However, if you use the `N` function, type the code

```
N[1/3]
```

and press Shift + Enter, you get the output:

```
In[5]:= N[1/3]
```

```
Out[5]= 0.333333
```

ARGUMENTS

An argument is a parameter passed into a function. These parameters always directly follow the function in square brackets. For the `Print` example, the argument is:


```
["Hello world!"]
```

And for the `N` function example, the argument is:

```
[1/3]
```

Some functions do not require arguments, but the large majority do, so it is important to read up about the particular function you are working with and find out exactly how to use it. For information on how to use a function and what arguments it will accept, you can type the function into the notebook window, preceded by a question mark. For instance, if you wanted more information on the `Print` function, you type

```
?Print
```

and press `Shift + Enter` to see information about the function (Example A).

If you are looking for a function but can't remember the full name, you can use this question mark notation to find it. Say, for example, you know that the function you are looking for begins with `Print`, but you can't remember the rest. You can type

```
?Print*
```

and press `Shift + Enter`. You will be presented with a list of functions that start with the word `Print` (Example B).

You can then click on any of these functions, and Mathematica will bring up a description of what it does, so you can determine whether it is the one you are thinking about before you use it in your program (Example C).

You can also look for functions ending in a particular word by putting the asterisk in front of the function name `*Print`. Alternatively, you can search for information about functions from the online documentation

to access the documentation center within Mathematica is to click on *Help | Online Documentation*.

STRINGS

The string data type (integers and floating-point numbers are other data types) is present in practically every programming language. A string is usually a sequence of characters typically used to display or use text within a program; however, it is possible to define strings that contain only numbers as well.

In Mathematica, as in most programming languages, a string is defined by placing text in quotation marks. If you do not put a string in quotation marks, Mathematica will probably try to evaluate it as either a variable or a function, and this will likely cause errors within your code.

In the preceding example, `"Hello world!"` is a string. In Mathematica, because of its ability to work as a multiparadigm language with natural text input, you don't need the quotation marks; however, I highly recommend getting into the habit early on of enclosing your strings in quotation marks because this prudent practice will likely save you some grief in the future.

LISTS

Lists are a fundamental part of the Wolfram language. Other programming languages use lists as well (e.g., Python). Lists also are sometimes referred to as arrays or matrices; however, in the Wolfram language, lists, arrays, and matrices are three distinct things. That said, the three are pretty similar: An array is effectively a list of functions, and a matrix is a list of lists.

In the Wolfram language, a list is contained in curly brackets:

```
{1, 2, 3, 4, 5}
```

```
{"this", "is", "a", "list", "of", "strings"}
```

The curly brackets are shorthand notation for creating a list; the full function for creating a list is:

```
In[1]:= ? Print
```

`Print[expr]` prints *expr* as output.

Example A

```
In[1]:= ? Print*
```

▼ System

<code>Print</code>	<code>PrintingOptions</code>	<code>PrintPrecision</code>
<code>PrintAction</code>	<code>PrintingPageRange</code>	<code>PrintTemporary</code>
<code>PrintForm</code>	<code>PrintingStartingPageNumber</code>	
<code>PrintingCopies</code>	<code>PrintingStyleEnvironment</code>	

Example B

```
In[2]:= ? Print*
```

▼ System

<code>Print</code>	<code>PrintingOptions</code>	<code>PrintPrecision</code>
<code>PrintAction</code>	<code>PrintingPageRange</code>	<code>PrintTemporary</code>
<code>PrintForm</code>	<code>PrintingStartingPageNumber</code>	
<code>PrintingCopies</code>	<code>PrintingStyleEnvironment</code>	

`PrintingStartingPageNumber` is an option for notebooks that specifies what number to assign to the first page of a notebook when printed.

Example C

```
In[11]:= Range[3, 10, 1/2]
Out[11]= {3, 7/2, 4, 9/2, 5, 11/2, 6, 13/2, 7, 15/2, 8, 17/2, 9, 19/2, 10}
```

Example D

```
In[13]:= Table[i+1, {i, 3, 10, 1/2}]
Out[13]= {4, 9/2, 5, 11/2, 6, 13/2, 7, 15/2, 8, 17/2, 9, 19/2, 10, 21/2, 11}
```

Example E

```
List[1, 2, 3, 4, 5]
```

However, when you run this code, it is automatically changed to the curly bracket format, so it is usually easier just to use the shorthand notation.

Building lists by manually entering data is a very long-winded process, and for some applications, it is much better to do this in an automated fashion. Luckily, the Wolfram language has you covered, with a number of ways in which to build lists automatically. The first of these methods is the `Range` function, which autofills a list of numbers:

```
Range[3, 10, 1/2]
```

This code will output a list of values starting at 3 and finishing at 10, with a step size of 1/2 (Example D).

The `Table` function is essentially a more complex version of the `Range` function that allows you to determine start, end, and step size values, and it also introduces an operator that acts on the number before building the list. For example,

```
Table[i+1, {i, 3, 10, 1/2}]
```

takes the values from the preceding `Range` function and adds 1 to each value, giving the output shown in Example E.

You can change the calculation within the `Table` function from `i+1` to anything you like (e.g., `i^3` or `i*3`).

To create a matrix, just create a list of lists:

```
matrix = {{5, 1, 3}, {7, 3, 0}, {9, 2, 4}}
```

```
In[16]:= Sqrt[{5, 1, 3, 8, 9, 4}]
Out[16]= {√5, 1, √3, 2√2, 3, 2}
```

Example G

```
In[14]:= matrix = {{5, 1, 3}, {7, 3, 0}, {9, 2, 4}}
Out[14]= {{5, 1, 3}, {7, 3, 0}, {9, 2, 4}}
```

```
In[15]:= MatrixForm[matrix]
```

```
Out[15]//MatrixForm=
⎛ 5 1 3 ⎞
⎜ 7 3 0 ⎟
⎝ 9 2 4 ⎠
```

Example F

You can then display this in a 2D format using the `MatrixForm` function,

```
MatrixForm[matrix]
```

Which gives the output in Example F.

You can even apply functions to an entire list because the Wolfram language treats them as a single entity. For instance,

```
Sqrt[{5, 1, 3, 8, 9, 4}]
```

gives the output in Example G.

Last, but not least, you can pull out individual values from within a list very easily using:

```
list = {1, 2, 3, 4, 5, 6}
2 * list[[1]]
```

You can also perform functions on individual elements within the list and then write the result back to the list itself:

```
list[[4]] = Sqrt[list[[4]]]
```

The output of those last few commands is shown in Example H.

As you can see, the final `Sqrt` command only performed the square root function on the fourth element in the list, changing it from 4 to 2, unlike the previous square root example, in which the entire list was changed.

```
In[19]:= list = {1, 2, 3, 4, 5, 6}
2 * list[[1]]

Out[19]= {1, 2, 3, 4, 5, 6}
Out[20]= 2

In[21]:= list[[4]] = Sqrt[list[[4]]]
Out[21]= 2

In[22]:= list
Out[22]= {1, 2, 3, 2, 5, 6}
```

Example H

Bear in mind when using lists in the Wolfram language that the index of a list starts at 1 – in other words, if you want to address the first item in a list, you use the code `list[[1]]`. In many programming languages (including Python), a list index starts at 0.

VARIABLES

In programming, a variable is a symbol or string that represents a quantity in a mathematical or computational expression. It is convenient to use variables when you have a number or expression that repeats. Additionally, if you ever need to make a change to a number in a calculation, you can use variables instead of actual numbers. In that way, if you want to change a value, you won't have to change every instance of that expression – all you have to do is change the definition of the variable.

For example, say you know the radius of a circle is 5, and you are asked to make a number of calculations to establish the magnitude of the diameter, circumference, and area of the circle. If you did not have variables, you would have to type:

```
diameter = 2 * 5
circumference = 2 * Pi * 5
area = Pi * 5^2
```

Now if you were told the radius is actually 7, you would have to change three instances of the old radius, 5, to the new radius, 7. Imagine instead that you used variables to achieve this same result:

```
radius = 5
diameter = 2 * radius
circumference = 2 * Pi * radius
area = Pi * radius^2
```

Now, if you were told to change the radius to 7, you would only have to change one parameter – the definition of the variable `radius` at the top. This is only a simple example to demonstrate a point, but as I am sure you can imagine, in a long and complicated series of calculations, variables are extremely useful for this reason.

If you have typed in and executed any of the above code, you will have realized that for every input line you type in, an output line is generated in Mathematica. If you would rather not see this output, you can add a semicolon to the end of the line. Not only does this neaten up the output by removing unnecessary feedback, it also significantly speeds up the calculation of the results, because instead

of using up CPU time to output results to the user interface, Mathematica can use all of the CPU available to it for the calculation process. For small programs like those I have used here, this is not such a problem, but it is good practice to get into the habit early of putting a semicolon at the end of every line for which you do not require output.

In the above example for the circle parameters, you are defining the radius, so you do not need or want Mathematica to generate output for this line. Therefore, you can adapt the code to

```
radius = 5;
diameter = 2 * radius
circumference = 2 * Pi * radius
area = Pi * radius^2
```

and press Shift + Enter to run. Notice the difference in output from the previous example that did not include the semicolon.

CONCLUSION

I hope this introduction to Mathematica and the Wolfram language has been useful and inspires you at least to have a go with the software and programming in this impressively intuitive language. It is really difficult to do justice to a programming language like Wolfram and a software package like Mathematica in a single magazine article, so I would highly recommend visiting the Wolfram Community Raspberry Pi forum [4] or the documentation center for the Wolfram language [5] and Mathematica [6]. Almost everyone reading this article will find something useful that could potentially make a job they are trying to achieve a whole lot easier.

Before the free release on Raspberry Pi, this software would have cost a fair bit, so it is definitely worth having a go with a piece of software that a lot of people would pay good money for – if for no other reason than to see what you have been missing out on.

Don't forget to keep us updated on your progress on the Raspberry Pi Geek Facebook page [6].

THE AUTHOR

Aaron Shaw is a volunteer at The MagPi magazine (www.themagpi.com), a magazine dedicated to Raspberry Pi users. He also founded Pi Supply (www.pi-supply.com) with a Kickstarter-funded add-on board for the Raspberry Pi (<http://kck.st/UVBXTE>). For help relating to this article, you can contact him through sales@pi-supply.com or via Twitter (@shawaj2).

INFO

- [1] Wolfram Alpha: <https://www.wolframalpha.com>
- [2] Raspbian and NOOBS: <http://www.raspberrypi.org/downloads/>
- [3] Mathematica documentation: <http://reference.wolfram.com/mathematica/guide/Mathematica.html>
- [4] Wolfram Community Raspberry Pi Forum: <http://community.wolfram.com/content?curTag=raspberry%20pi>
- [5] Wolfram language documentation: <http://reference.wolfram.com/language/>
- [6] Raspberry Pi Geek Facebook page: <https://www.facebook.com/RasPiGeek>

Write for Us!

We are always looking for good articles on Linux and the tools of the Linux environment. Although we will consider any topic, the following themes are of special interest:

- System administration
- Useful tips and tools
- Security, both news and techniques
- Product reviews, especially from real-world experience
- Community news and projects

If you have an idea, send a proposal with an outline, an estimate of the length, a description of your background, and contact information to edit@linux-magazine.com.

The technical level of the article should be consistent with what you normally read in *Linux Magazine*. Remember that *Linux Magazine* is read in many countries,

and your article may be translated into one of our sister publications. Therefore, it is best to avoid using slang and idioms that might not be understood by all readers.

Be careful when referring to dates or events in the future. Many weeks could pass between your manuscript submission and the final copy reaching the reader's hands. When submitting proposals or manuscripts, please use a subject line in your email message that helps us identify your message as an article proposal. Screenshots and other supporting materials are always welcome.

Additional information is available at:

http://www.linuxpromagazine.com/contact/write_for_us.

AUTHORS

Michael Badger	34
Paul Brown	8, 18, 24, 30
Joe Casad	3, 8
Mike Cook	60
Marko Dragicevic	38, 42, 73, 80
Darren Grant	56
Joseph Guarino	8
Christoph Langner	76
Martin Mohr	68
Dmitri Popov	64
Richard Ryniker	88
Tim Schürmann	46
Aaron Shaw	84, 92
Ruth Suehle	52
Ferdinand Thommes	27

DISCLAIMER

All brand or product names are trademarks of their respective owners. Contact us if we haven't credited your copyright; we will always correct any oversight.

CONTACT INFO

Editor in Chief
Joe Casad, jcasad@linuxnewmedia.com

Managing Editor
Rita L Sooby, rsooby@linuxnewmedia.com

Contributing Editors
Paul C. Brown, Aaron Shaw

Localization & Translation
Ian Travis

Copy Editor
Amber Ankerholz

Layout
Dena Friesen, Lori White

Cover Design
Lori White, Illustration based on graphics by Les Cunliffe, 123RF.com and Raspberry Pi

Advertising – North America
Ann Jesse, ajesse@linuxnewmedia.com
phone +1 785 841 8834

Advertising – Europe
Penny Wilby, pwilby@sparkhausmedia.com
phone +44 1787 211100

Publisher
Brian Osborn, bosborn@linuxnewmedia.com

Marketing Communications
Darrah Buren, dburen@linuxnewmedia.com

Customer Service / Subscription
For USA and Canada:
Email: cs@linuxnewmedia.com
Phone: 1-866-247-2802
(toll-free from the US and Canada)
Fax: 1-785-856-3084

For all other countries:
Email: subs@linuxnewmedia.com
Phone: +49 89 9934 1167
Fax: +49 89 9934 1199
Linux New Media USA
616 Kentucky St. Lawrence, KS 66044
www.linuxpromagazine.com

While every care has been taken in the content of the magazine, the publishers cannot be held responsible for the accuracy of the information contained within it or any consequences arising from the use of it. The use of the DVD provided with the magazine or any material provided on it is at your own risk.

Copyright and Trademarks © 2016 Linux New Media USA, LLC

No material may be reproduced in any form whatsoever in whole or in part without the written permission of the publishers. It is assumed that all correspondence sent, for example, letters, email, faxes, photographs, articles, drawings, are supplied for publication or license to third parties on a non-exclusive worldwide basis by Linux New Media unless otherwise stated in writing.

Linux Magazine Special (ISSN 1757-6369) is published by Linux New Media USA, LLC, 616 Kentucky St, Lawrence, KS, 66044, USA.

Printed in Germany

Distributed by COMAG Specialist, Tavistock Road, West Drayton, Middlesex, UB7 7QE, United Kingdom

Published in Europe by: Sparkhaus Media GmbH, Putzbrunner Str. 71, 81739 Munich, Germany

Your Roadmap to the Open Hardware Revolution ...

An exciting world of projects, tips, and skill-building tutorials awaits you in every issue of Raspberry Pi Geek.

Order your subscription today and tune in to the revolution!

shop.linuxnewmedia.com

Print Sub

Carry our easy-to-read print edition in your briefcase or backpack - or keep it around the lab as a permanent reference!

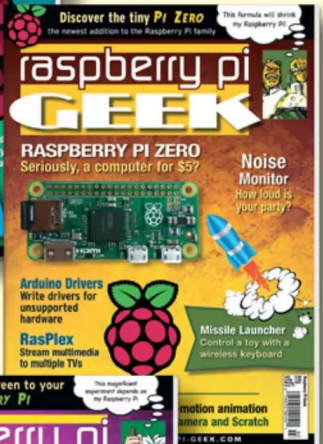
Digital Sub

Our PDF edition is a convenient option for mobile readers.

6 print issues with 6 DVDs or
6 digital issues for only

\$59.95 £37.50 €44.90

We're also in
Google Play
and the
iTunes Store!



Download on the
App Store



GET IT ON
Google play



iTunes Store



Google US

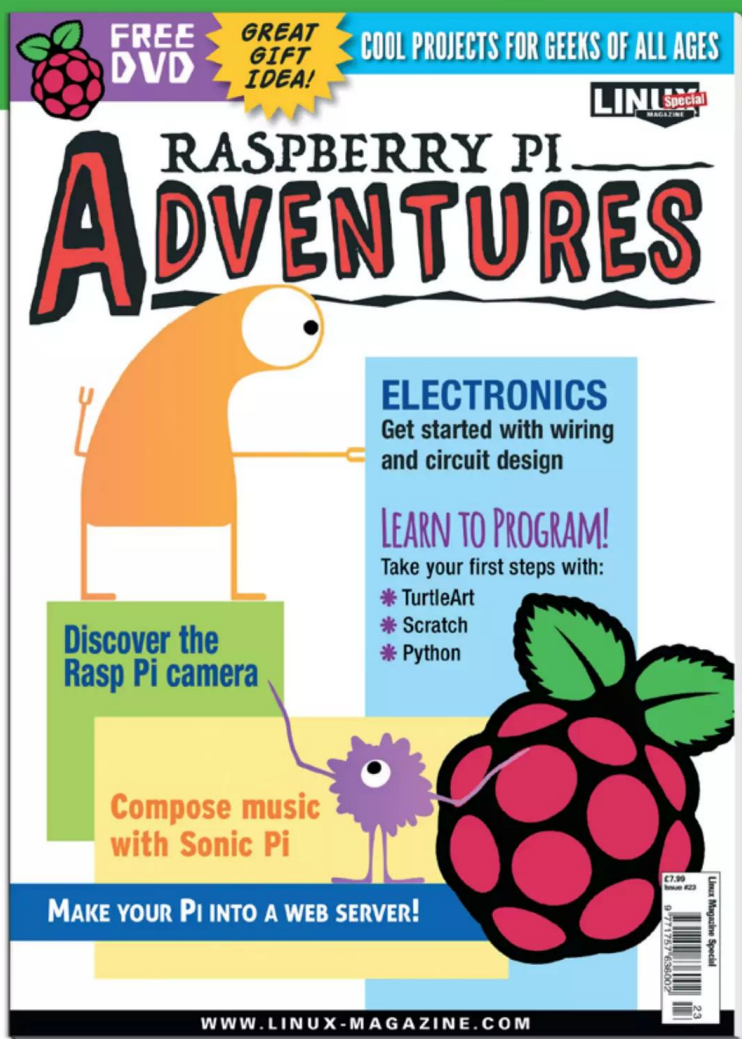


Google UK

Shop the Shop

shop.linuxnewmedia.com

RASPBERRY PI ADVENTURES



**COOL
PROJECTS
FOR GEEKS
OF ALL
AGES**



RASPBERRY PI ADVENTURES

is a one-volume special edition magazine for curious Raspberry Pi beginners. This easy, hands-on guide starts with an introduction to computers and offers a series of special hands-on projects illustrating many of the most popular uses for the Raspberry Pi.

Great Gift Idea!

ORDER YOUR VERY OWN ISSUE!



ORDER ONLINE:

shop.linuxnewmedia.com/se23