Raspberry Pi Blueprints

The Raspberry Pi is one of the cheapest and most popular single-board computers, as such it has become a platform for users to create their own interesting hardware and software projects. It’s capable of doing everything you’d expect a desktop computer to do, from browsing the Internet and playing high-definition videos, to making spreadsheets, word processing, and playing games.

Filled with 10 interesting hardware projects, this book starts off by helping you get your Raspberry Pi set up with the popular Raspbian operating system, writing Python scripts, and interacting with some external hardware. As you progress through the next set of projects, you are introduced to additional pieces of software and will write scripts that bridge the gap between them and various sensors and hardware.

By the end of the book, you will be writing full Python applications that will both interface with a wide variety of hardware and be exposed to the Internet.

Who this book is written for

If you have already undertaken some simple projects with the Raspberry Pi and are looking to enter the exciting work of hardware interaction, then this book is ideal for you.

What you will learn from this book

- Install and configure an operating system on your Raspberry Pi
- Design electronics circuits using a range of standard components
- Interact with electronic circuits and hardware using software libraries
- Write efficient Python applications to control hardware and process data
- Build professional-quality enclosures using a range of construction methods
- Implement best practices to be able to power your projects while on the go
- Configure Linux to run your applications at specific times
- Make your applications interact with online data and communication services

Design and build your own hardware projects that interact with the real world using the Raspberry Pi
In this package, you will find:

- The author biography
- A preview chapter from the book, Chapter 1 'Raspberry Pi Pirate Radio'
- A synopsis of the book’s content
- More information on Raspberry Pi Blueprints

About the Author

Dan Nixon lives in England and is a 20-year old software engineering student who is currently studying at Newcastle University and is in his third year. He has long had an interest in electronics and embedded computing projects.

Previously, Dan has worked on a 360-degree camera system for the Raspberry Pi and this is where his interest in the platform started.

Currently, he is on a year-long work placement and is working on neutron data analysis software at the Rutherford Appleton Laboratory in Oxford, UK.
Raspberry Pi Blueprints

The Raspberry Pi is a small form factor, single board, ARM-based computer. It is capable of running on many desktop applications that can be run on a standard Linux computer. While the Pi is only slightly larger than a credit card, it uses very little power. As such, it has become very popular among the hacker and maker community, which uses the Pi to integrate more computing power in their projects.

The Pi is very easy to set up and in less than 20 minutes, you can run it on a modified version of popular, Linux-based operating systems. It will function just like you would expect a desktop PC to. Thanks to its small form factor, many people have used it as an alternative to a traditional PC for fast access to applications and the Internet. Since the Pi draws very little power, it can be left running.

Where the Pi really excels is that it brings more computing power to hardware and electronics projects. This is made possible by a wide range of interfaces on the Pi that are typically not found on conventional computers.

This has led to a rise in new projects that are made using the Pi, which otherwise would have been considerably more difficult or expensive to make.

What This Book Covers

Chapter 1, Raspberry Pi Pirate Radio, introduces the Pi and gives an overview of its setup and configuration procedure and some fundamental Linux concepts. This chapter then demonstrates the basic use of the GPIO header and Python scripting.

Chapter 2, Portable Speaker System, explores how to use the Pi as a portable, battery-powered speaker system with a self-contained media server that can be used without any ties to a power supply or home network.

Chapter 3, Mini Retro Arcade Cabinet, demonstrates how the Pi can be used to create a mini arcade cabinet complete with a traditional joystick and button controls and how it can be used to play a range of classic arcade and console games.

Chapter 4, GPS-enabled Time-lapse Recorder, covers how to use the Pi as a time lapse recorder that can also capture the location of each image and trigger the image capture based on the current position of and distance traveled by the camera.

Chapter 5, Home Theater PC, explores the way in which the Pi can be used as a home theater PC using the popular XBMC media center software and a custom, purpose-built enclosure.
Chapter 6, *Outdoor Weather Station*, delves into the topic of interfacing hardware to the Pi using intermediate devices, in this case, Arduino. We also take a look at Python web applications running on the Pi.

Chapter 7, *Home Security System*, explores how to use the Pi as a hub for a wireless network of sensors and how this data can be used and displayed on a web application.

Chapter 8, *Remote-operated Robotic Arm*, focuses on how to use the Pi to control and monitor devices remotely in the form of a robotic arm that can be controlled through a web application.

Chapter 9, *Magic Mirror*, expands your knowledge of woodworking and designing, which will prove to be important skills for any further projects that you do in the field of electronics and physical computing.

Chapter 10, *Bottle Xylophone*, covers how the Pi, several servos, and some empty bottles can be turned into a musical instrument driven by MIDI files.
Raspberry Pi Pirate Radio

In this chapter, we will take a quick look at the Raspberry Pi hardware and some of the software that will be used both in this project and the majority of others throughout this book.

As the Pi was based on hardware that was to be included in embedded or portable electronics (such as smartphones and tablets), it has a few extra hardware features that are not found on a typical desktop or laptop PC, one of which is the General Purpose Input and Output (GPIO) header. This is a set of pins (26 on the model A and B, and 40 on the model B+) that allows you to communicate with external hardware such as GPS sensors, accelerometers, and motors through programming languages such as Python, C, and C++. When we get further in this chapter, we will take a look at a little trick that can be done to turn one of these pins into an FM radio transmitter.

What you will need

You will need the following:

- The Raspberry Pi and power supply
- An SD card with at least 4 GB memory (16 GB is recommended as it holds a good amount of music)
- A thick wire or FM radio antenna
- Male to male 0.1 inch pin jumper wires
Setting up the Pi

In order to get the Pi up and running, the bare minimum you will need is a USB power supply, micro USB cable, an Ethernet cable to connect the Pi to your network, and an SD card that has at least 4 GB memory. Although, later on, some projects will need a larger capacity of the SD card.

While a monitor, mouse, and keyboard can also be used for a lot of the projects in this book, they are not actually required assuming that you have a network that runs a DHCP server, which you can connect with the Pi in order to set it up over SSH.

It is worth mentioning now that for a USB-powered device, the Pi is quite power-hungry (drawing around 600-700 mA), therefore, while the Pi can be powered from a USB port, which is usually rated for around 500 mA, it is recommended that you use a mains powered adapter. Without this, the Pi can become unstable when additional devices are connected that draw more power, for example, a USB Wi-Fi dongle or camera module.

Choosing a Linux distribution

There are a wide range of Linux distributions available for the Pi, some of which are very general purpose while others are built for specific purposes (a couple of which will be used in the later projects of this chapter). For now, we will use the most standard distribution, Raspbian, which as the name suggests is based on the Debian distribution.


Another relatively new distribution that is worth mentioning is Minibian (http://minibianpi.wordpress.com), a distribution based on Raspbian, which has the majority of its default software removed. While this may not seem that helpful, it means that the Pi boots in a useable OS in around 25 seconds and saves the SD card space. This type of OS is more suited to a project that has finished being developed and is going into a more natural usage environment. You usually would not expect a Wi-Fi router to take 3 minutes to boot, so why should your Pi-based Internet radio?
Writing an SD card

When you have all the relevant hardware, head over to http://www.raspberrypi.org/downloads and download the ZIP archive for Raspbian and extract it.

The next steps vary depending on your operating system.

Windows

Windows does not natively include a tool used to write disk images, therefore, the Win32 Disk Imager (http://sourceforge.net/projects/win32diskimager) application is used to write images to an SD card.

First, go to the download link mentioned in the preceding paragraph, download, and install Win32 Disk Imager. As writing images to drives requires administrator permissions, you will need to run Win32 Disk Imager by right-clicking on its entry in the Start menu, and selecting Run as administrator:
Next, select the .img file that was extracted from the Raspbian ZIP file using the folder icon under **Image File** and the drive letter of the SD card from the drop-down box:

Ensure that the correct drive letter is selected, and the SD card does not contain any data that has not been backed up, as this operation will destroy all of the data already present in the card.

Finally, click on **Write** to write the image to the SD card.

### Linux and Mac OS

Unix and Unix-like operating systems already have a tool to read and write images to external storage—`dd`.

First, you will need to find the path for your SD card. The easiest way to do this is to use the udev management tool, `udevadm`, to monitor the udev log, which logs activity from various devices on the system, including the SD card being inserted and will allow you to see the device path that was assigned to it and the partitions already existing on it. This can be done by running the following command:

```
udevadm monitor --udev
```

Then, insert the SD card, at which point, you should see a set of log messages similar to the ones shown in the following screenshot:
Here, the important information can be seen on the last three lines, which tells us that in this case, the path for the entire card is /dev/sdb, with two partitions at /dev/sdb1 and /dev/sdb2.

We must now ensure that none of these partitions on the SD card are currently mounted, as some desktop managers (such as GNOME, the default desktop manager on Ubuntu) will try to automatically mount partitions when they are detected.

To do this, we will run the following command for every partition (that is, in my case, /dev/sdb1 and /dev/sdb2), where PATH is the path to the partition:

```
umount PATH
```

If the partition was mounted, you will not see any output from the command; however, if the partition was not mounted, you will get the following message:

```
dan@lannixon-envy-ubuntu:~$ umount /dev/sdb1
umount: /dev/sdb1 is not mounted (according to mtab)
dan@lannixon-envy-ubuntu:~$
```

Once you know the path for your SD card and have ensured that no existing partitions are mounted, you can then write the Raspbian image to your SD card using the following command, where PATH is the path to the SD card (/dev/sdb in my case):

```
sudo dd if=path/to/raspbian_image.img of=PATH
```

Be certain that the path to the SD card is correct, and the card does not contain any data that has not been backed up as this operation will destroy all of the data already present on the card.
Where `path/to/raspbian_image.img` is the extracted image file and `sdX` is the path to your SD card. Note that this step can take up to 5-8 minutes, since no output is given on the screen, the SD card reader's busy/data LED is a good indication that the image is being written.

Since the GNU Coreutils (which include commands such as `cp`, `mv`, `dd`, and so on) do not provide much (or any) output to indicate the progress, you may want to take a look at the Coreutils Viewer tool (https://github.com/Xfennec/cv), which shows the progress of the Coreutils commands.

**Booting the Pi for the first time**

Once you have the image written to the card, it is time to boot the Pi and perform the initial setup. The most common way to do this is by using a keyboard, mouse, and monitor; however, if you have access to a network that provides DHCP (as most home networks do), then the setup can be done entirely over Secure Shell (SSH).

First, set up the hardware and boot the Pi by inserting the SD card, connecting the power, and the Ethernet cable. Within a few seconds, you will see that the ACT LED starts to flicker. If it flashes for very short pulses or does not light at all, then this indicates an issue with either the SD image, the connection between the card and the Pi (a common issue for the models A and B), or the card itself.

Once the Pi has booted (indicated by less frequent flashing of the ACT LED), you will need to determine its IP address. There are two main ways to do this: by accessing the DHCP allocations via your router or by scanning the local network.

**Network scanning**

The cross-platform tool, the Nmap utility (http://nmap.org) can be used to scan a network.

To do so, you will need to find the IP address of your PC (in order to find the subnet on your local network to search for the Pi in). On Windows, this can be done by opening the Command Prompt and executing this command:

`ipconfig`
This should give you the information about your network interfaces similar to the following:

On Unix, this can be done by using the following command:

```
ifconfig
```

This command gives output similar to the following:
The search IP range that is given to Nmap is obtained by replacing the last number of IPv4 or InetAddress with *. In our case, it will be 192.168.0.*.

Now that we know the address range in which we will be looking, we can open a terminal and run the following command:

```
nmap --open 192.168.0.*
```

Replacing the IP range with your search range will try to make contact with every host in the given IP address range and will return with a list of every host that is up, with a list of their open ports, and what services they correspond to. In our case, we are looking for any hosts that have an open SSH port:

```
Starting Nmap 5.21 ( http://nmap.org ) at 2014-09-21 11:58 BST
Nmap scan report for 192.168.0.8
Host is up (0.0064s latency).
Not shown: 999 closed ports
PORT   STATE SERVICE
22/tcp open  ssh
Nmap done: 256 IP addresses (5 hosts up) scanned in 5.84 seconds
```

Here, I have only shown the report for the Pi. Usually, after the first boot, SSH will be the only service that is started, so it is usually given away as the device that only provides SSH.

## Connecting to the Pi via SSH

On Unix, accessing the Pi via SSH is as simple as executing the following:

```
ssh pi@[Pi IP]
```

Here, Pi IP is the IP address of the Pi. You may get a warning similar to this the first time you run the command:

```
The authenticity of host '192.168.0.8 (192.168.0.8)' can't be established.
Are you sure you want to continue connecting (yes/no)?
```

This can safely be ignored by typing yes and pressing Enter. You will then be asked for a password, and as this is the first boot, the default is raspberry.

On Windows, PuTTY (http://www.putty.org) can be used to SSH into the Pi. Once downloaded, run PuTTY and enter the IP address of the Pi and the Host Name field, ensuring that Port is set to 22 and SSH is selected. Then, click on Open:
You will see a warning message similar to the following about the identity of the host:

![PuTTY Security Alert](image)

The server's host key is not cached in the registry. You have no guarantee that the server is the computer you think it is.

The server's rsa2 key fingerprint is:

```
```

If you trust this host, hit Yes to add the key to PuTTY's cache and carry on connecting.

If you want to carry on connecting just once, without adding the key to the cache, hit No.

If you do not trust this host, hit Cancel to abandon the connection.
Again, this can be ignored by clicking on Yes. You will then be asked for a username and password in the PuTTY terminal window. Since this is the first boot, the defaults are pi and raspberry.

**Common Linux commands**

At this point, it would be good to learn a few Linux shell commands, which you will no doubt come across while working with the Pi:

- **cd**: The change directory command sets the working directory to a given directory, for example, `cd /home/pi`
- **ls**: The list command lists the contents of the current working directory
- **mkdir**: The make directory command creates a new directory within the current working directory, for example, `mkdir code`
- **cat**: The concatenate command can perform operations on text files, and it can also display its contents on the console, for example, `cat /etc/passwd`
- **pwd**: The print working directory command tells you the directory you are currently in
- **chmod**: The change mode command changes the access permission to a file, for example, `chmod 744 file` (this gives read, write, and execute permissions to the owner, and read-only permission to everyone else)
- **sudo**: The super user do command executes the following commands as root, for example, `sudo apt-get upgrade`
- **mv**: The move command moves the file or directory in the first argument to the second, for example, `mv file.txt misc_files`
- **cp**: The copy command creates a copy of a file or directory in another directory, for example, `cp file.txt misc_files/file2.txt`
- **rm**: The remove command deletes a file, for example, `rm file.txt`

More information can be obtained about a given command via its man page, which can be accessed using the following command, where `command` is the command you wish to know more about:

`man command`
The initial setup

Once you are able to SSH in the Pi, run the configuration utility using the following command:

```
sudo raspi-config
```

This utility allows you to configure the Pi hardware and perform useful configuration tasks such as changing passwords and resizing the root partition on the SD card, both of which, we will do now.

First, select the **Expand Filesystem** option and press *Enter*. The utility will modify the partition table and report that it has completed, and that the changes made will only affect the filesystem after the next reboot. Press *Enter* again to return to the main menu.

Now, select **Change User Password** and hit *Enter* twice. You will be taken back to the shell and prompted to enter a new password, which has to be entered twice. Once done, a confirmation box will notify you that the password was changed successfully; press *Enter* to return to the menu.

As this is all that needs to be done in raspi-config for now, press the left arrow key twice to select **Finish** and *Enter* to confirm. You will be asked whether you want to reboot now; select **Yes** and press *Enter*. Once the Pi is rebooted, SSH back into it using your new password.

Once you have access again, check whether you now have access to the full storage space on the SD card using:

```
df -h
```

The command should report the size of rootfs much closer to the SD card capacity:

```
rootfs  16G  2.5G  13G  17%  /
```

All that is left to do now is to make sure that the software on the Pi is up to date. Since we are using an image that was just downloaded, it is not likely that there will be a large number of updates; however, it is a good practice to keep an installation up to date.

Running the following commands will first update the list of available packages then update any installed packages with newer versions than what was installed:

```
sudo apt-get update
sudo apt-get upgrade
```
Note that this process can take up to 10 minutes depending on the number of updates, and usually takes longer than a desktop PC or laptop as this process is writing to an SD card rather than a traditional hard drive.

Setting up the pirate radio

It's now time to download and set up the PiFM software, which will allow you to use the GPIO header as an FM transmitter. First, we'll need to download the software using the following commands:

```
wget http://omattos.com/pifm.tar.gz
```  
```
tar -xzvf pifm.tar.gz
```  

This will get the gzipped archive that contains the PiFM software, the Python library, and some test files and uncompress them.

For now, a single male to male 0.1 inch pin jumper will suffice as an antenna; this should be connected to GPIO 4 (pin number 7) on the GPIO header and made to stand as upright as possible to ensure the best range (refer to the following image):
You are now ready to test the setup with an FM radio; firstly, you will need to choose a frequency that will not overlap with any licensed broadcasts in your area and is within the FM radio transmission range in your area (usually between 88 Mhz and 108 Mhz). A lot of radios that do automatic tuning also tend to prefer frequencies that are a multiple of 0.1 MHz (for example, 99.9, 101.3, and so on).

Once you have chosen your frequency, tune your radio in to it, and run the following command on the Pi, where `freq` is the frequency you wish to broadcast on:

```
sudo ./pifm left_right.wav freq 22050 stereo
```

You should now hear a sample auto clip demonstrating the ability to broadcast stereo audio.

### Transferring MP3 files to the Pi

For our media player, we are going to need a selection of MP3 files on the Pi SD card. There are two main ways to do this, either by moving the card back to your PC and mounting it like a regular SD card, or by using the Secure File Transfer Protocol (SFTP).

In this case, since the Pi is already running, I have opted to use SFTP. This can be done using the FileZilla (https://filezilla-project.org) FTP client. Once installed and opened, enter the same details that you used to connect to the Pi over SSH in the fields at the top of the window (using 22 for Port), and click on Quickconnect.
You should now be able to browse the filesystems of both your computer and the Pi, as shown in the following screenshot. Files can be uploaded by right-clicking on them and selecting **Upload**.

![File browser screenshot](image)

In order to play the music you have just transferred to the Pi, you will need to install the **ffmpeg** utility, which is a tool that is commonly used to transcode media files and can be installed using the following command:

```
sudo apt-get install ffmpeg
```

Now that you have some of your own music on the Pi, you can try playing it by piping the output of **ffmpeg** to PiFM:

```
ffmpeg -i file.mp3 -f sl16e -ar 22.05k -ac 2 - | sudo ./pifm - freq 22050 stereo
```

Here, `file` is the MP3 file to be broadcasted and `freq` is the frequency to broadcast it on. Since we are telling **ffmpeg** to provide output for two audio channels (`-ac 2`) and giving the stereo option to PiFM, this should give a stereo audio broadcast.
Scripting a media player

Now that the FM transmitter is working, we can make it do something a little more useful; in this case, we will use it to broadcast a personalized radio station with a collection of your own music.

To do this, we will write a Python script that manages to search for MP3 files and calls PiFM to broadcast them.

Calling PiFM from Python

In the PiFM download and on the PiFM website, you may have noticed that there is a Python library that can be used to control PiFM. Although, looking at the source code for it, you can see that all the library can do is call the pifm executable with the minimum number of commands, and therefore will not allow us to play MP3 files.

However, it is a simple process to create our own function that will allow us to pass the filename of an MP3, a frequency, and that will allow Python to call ffmpeg and PiFM in order to broadcast the audio in the file for us:

```python
def play_file(filename, frequency):
    command = 'ffmpeg -i "%s" -f s16le -ar 22.05k -ac 2 - | sudo ./pifm
    -f 22050 stereo' % (filename, frequency)
    subprocess.call(command, shell=True)
```

This code is an extract from the `player.py` file; all we are doing here is taking the shell commands used to play an MP3 file and replacing the filename and broadcast frequency with values that are passed to the function as parameters. Then, we are using the Python subprocess module to execute the command as if it was typed into a shell.

Download the example code

You can download the example code files from your account at [http://www.packtpub.com](http://www.packtpub.com) for all the Packt Publishing books you have purchased. If you purchased this book elsewhere, you can visit [http://www.packtpub.com/support](http://www.packtpub.com/support) and register to have the files e-mailed directly to you.
Searching for MP3 files

For our script to play any MP3 files, it needs to be able to find them first. Finding an MP3 file involves taking a look at each file in a starting directory to check whether a file is an MP3 file, and then repeating this process for every directory within the start directory. Thankfully, Python makes this very easy:

```python
mp3_files = list()
for root, dirs, files in os.walk(directory):
    for filename in files:
        if filename.endswith('.mp3'):
            mp3_files.append(os.path.join(root, filename))
```

Here, `directory` is the directory we want to search for files media in. The `os.walk` function returns a tuple; the first element (`root`) is a string that contains the absolute path to the search directory, `dirs` is a list of directories within the search directory and its subdirectories, and `files` is a list of all the files within the search directory and its subdirectories.

The `if` statement is a simple way to check whether a file is an MP3 file based solely on its file extension. If it has the `.mp3` extension, it is added to the `mp3_files` list, which is then passed to the playlist code.

Getting input from a command line

In order to set various settings for our player script, for example, the directory in which you can search for files, we need a way to get input from the user via a command line, and in this case, from the arguments passed to the script when it is started. To do this, we will use the `argparse` Python module:

```python
parser = argparse.ArgumentParser(description='Broadcast a set of MP3s over FM')
parser.add_argument('--frequency', default=101.1, type=float, help='Frequency on which to broadcast')
params = parser.parse_args()
```

The `argparse` Python module allows you to define a set of arguments that can be passed to a Python program, it allows you to parse the arguments when the script is run, and it automatically allows you to generate a help page (accessed by passing `-h` to the script).
In this case, we will add an argument for the broadcast frequency, which is set using either -f or --frequency; type is used to validate input from the user, default is what is read by the program if the user does not set a value, and help is what is shown for this argument on the help page.

**Queuing the media files to be played**

Our media player will also need a way to manage which file should be played next. We will implement this in two ways: linear playback in the order the files were discovered and randomized playback:

```python
file_number = -1
while True:
    if params.random:
        file_number = random.randint(0, len(filenames) - 1)
    else:
        file_number += 1
    if file_number >= len(filenames):
        return
    play_file(filenames[file_number], params.frequency)
```

Here, if the --random parameter has been passed to our script, whenever we are about to play a file, the file to be played will be selected at random using the `randint` function in the `random` Python module. This number is then used to get a certain file from the list of filenames that were previously discovered.

**Using the media player script**

The `player.py` script can be invoked using the following command:

```bash
python player.py -d music -f 99.9 --random
```

This will search for all MP3 files under the music directory and broadcast them at 99.9 MHz in a random order. A full list of commands will be available to you if you run:

```bash
python player.py -h
```

Since the radio is not much of use when you have to SSH into it to start playing the music, we will add a cron job that will start the `player.py` script when the Pi boots up.
To do this, we will use a shell script to start the Python script (`start_player.sh`):

```bash
#!/bin/bash
sleep 20
cd /home/pi
python player.py -d music --random -f 99.9 &
```

This will ensure that the player script is executed in the correct folder, in this case, `/home/pi`, as this is where our PiFM executables and music directories are.

Next, we will add an entry in the crontab, which is where cron jobs are defined. This can be edited with the following command:

```
sudo crontab -e
```

This command will open the default command-line text edit on the Pi, `nano`, which will allow you to add entries to the crontab. To start our media player script, we will need to add the following line to the end of the crontab that is opened:

```
@reboot /home/pi/start_player.sh
```

In our case, we will use the `@reboot` cron rule instead of specifying a time for the command to be run, and as the name suggests, this will run the given command when the OS starts.

Press `Ctrl + X` followed by `Y` and `Enter` to save the changes to the crontab, then use the following command to reboot the Pi, and if all goes as planned, start your personalized radio station:

```
sudo reboot
```

## Summary

In this chapter, we covered the basic setup of the Pi hardware and Raspbian OS. You learned how to discover the IP address of the Pi in order to allow remote access to it over SSH; this process will be used in the majority of the projects later in this book.

We also took a quick look at the GPIO expansion header, and how it is typically used to interface with external hardware. We also used this as a crude FM transmitter through which we wrote a single Python script to broadcast music.

In the next chapter, we will go into further details regarding how to set up various software packages in order to run them on the Pi, as well as you will learn the various ways to configure networking when we make a portable speaker system that runs the Logitech Media Server.
Where to buy this book

You can buy Raspberry Pi Blueprints from the Packt Publishing website.

Alternatively, you can buy the book from Amazon, BN.com, Computer Manuals and most internet book retailers.

Click here for ordering and shipping details.