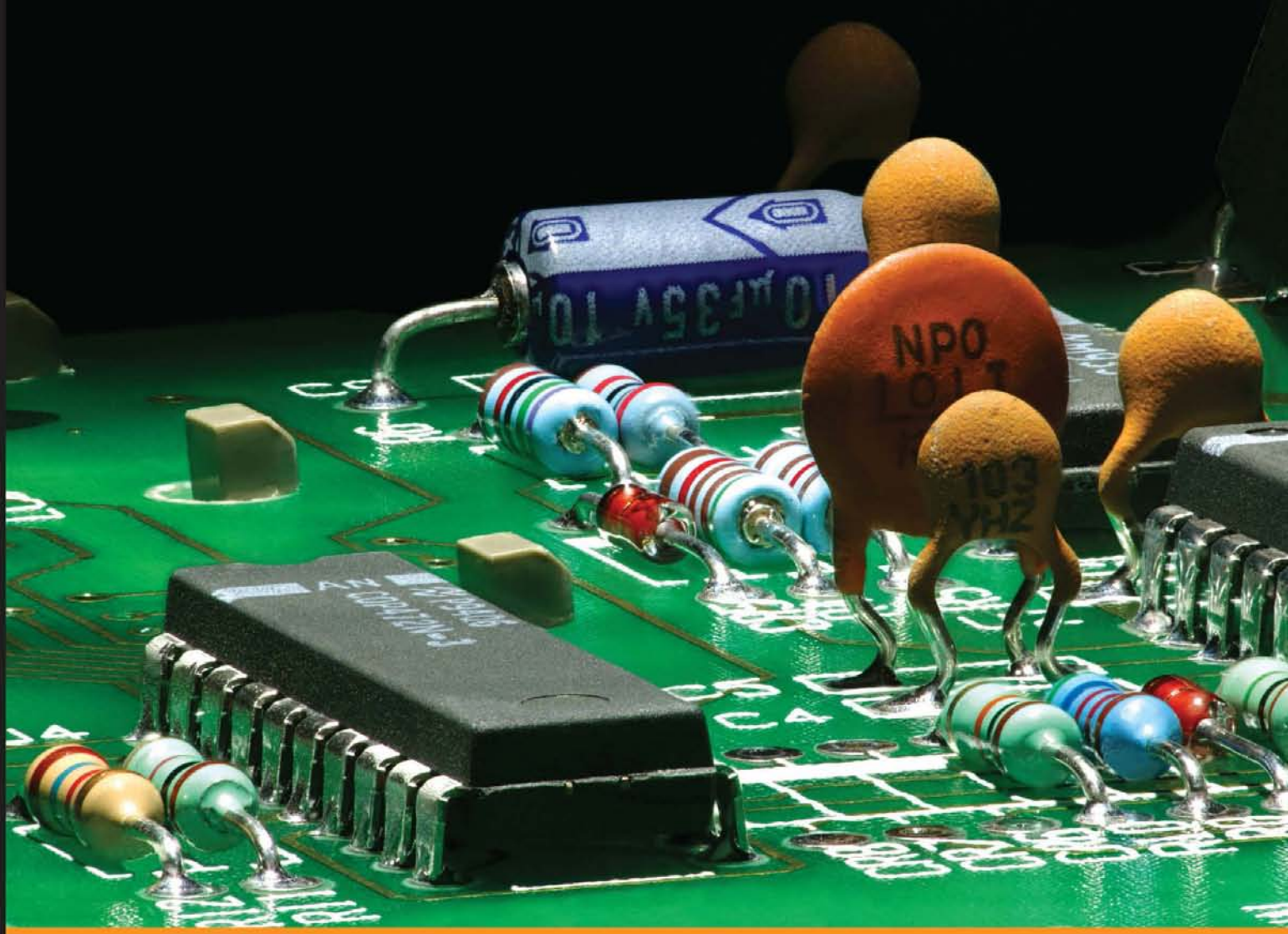


Community Experience Distilled

Raspberry Pi Robotics Essentials

Harness the power of Raspberry Pi with Six Degrees of Freedom (6DoF) to create an amazing walking robot

Richard Grimmett

[PACKT] PUBLISHING

Raspberry Pi Robotics Essentials

Harness the power of Raspberry Pi with Six Degrees of Freedom (6DoF) to create an amazing walking robot

Richard Grimmett

[PACKT]
PUBLISHING
BIRMINGHAM - MUMBAI

Raspberry Pi Robotics Essentials

First published: June 2015

Production reference: 1150615

Published by Packt Publishing Ltd.
Livery Place
35 Livery Street
Birmingham B3 2PB, UK.

ISBN 978-1-78528-484-7

www.packtpub.com

Credits

Author
Richard Grimmett

Reviewers
Ashwin Pajankar
Werner Ziegelwanger

Commissioning Editor
Neil Alexander

Acquisition Editor
Tushar Gupta

Content Development Editor
Kirti Patil

Technical Editor
Mahesh Rao

Copy Editors
Aditya Nair
Sameen Siddiqui
Stuti Srivastava

Project Coordinator
Kranti Berde

Proofreader
Safis Editing

Indexer
Priya Sane

Graphics
Sheetal Aute

Production Coordinator
Shantanu N. Zagade

Cover Work
Shantanu N. Zagade

About the Author

Richard Grimmett has more fun working on robotic projects than should be allowed. He also enjoys teaching computer science and electrical engineering at Brigham Young University, Idaho. He has a bachelor's and master's degree in electrical engineering and a PhD in leadership studies. He has written books on how to use Raspberry Pi, Arduino, and BeagleBone Black for robotics projects.

About the Reviewers

Ashwin Pajankar is a Bangalore-based engineer who wears many different hats depending on the occasion. He graduated from IIIT Hyderabad in 2012 with a master of technology degree in computer science and engineering. He has a total of 5 years of experience in the software industry, where he has worked in different domains, such as testing, data warehousing, replication, and automation. He is very well versed in DB concepts, SQL, and scripting with Bash and Python. He has earned professional certifications in products from Oracle, IBM, Informatica, and Teradata. He's also an ISTQB-certified tester.

In his free time, he volunteers for different technical hackathons or social-service activities. He was introduced to the Raspberry Pi in one of the hackathons, and he's been hooked on it ever since. He writes a lot of code in Python, C, C++, and Shell on his Raspberry Pi B+ cluster. He's currently working on creating his own Beowulf cluster of 64 Raspberry Pi 2 models.

Werner Ziegelwanger, MSc, has studied game engineering and simulation, and he got his master's degree in 2011. His master's thesis was published with the title *Terrain Rendering with Geometry Clipmaps for Games*, by Diplomica Verlag. His hobbies include programming and games and working with all kinds of technical gadgets.

Werner was a self-employed programmer for some years and mainly worked on Web projects. During this time, he started his own blog (`http://developer-blog.net`), which is about the Raspberry Pi, Linux, and open source.

Since 2013, Werner has been working as a Magento developer and the head of programming at mStage GmbH, an eCommerce company focused on Magento.

www.PacktPub.com

Support files, eBooks, discount offers, and more

For support files and downloads related to your book, please visit `www.PacktPub.com`.

Did you know that Packt offers eBook versions of every book published, with PDF and ePub files available? You can upgrade to the eBook version at `www.PacktPub.com` and as a print book customer, you are entitled to a discount on the eBook copy. Get in touch with us at `service@packtpub.com` for more details.

At `www.PacktPub.com`, you can also read a collection of free technical articles, sign up for a range of free newsletters and receive exclusive discounts and offers on Packt books and eBooks.

`https://www2.packtpub.com/books/subscription/packtlib`

Do you need instant solutions to your IT questions? PacktLib is Packt's online digital book library. Here, you can search, access, and read Packt's entire library of books.

Why subscribe?

- Fully searchable across every book published by Packt
- Copy and paste, print, and bookmark content
- On demand and accessible via a web browser

Free access for Packt account holders

If you have an account with Packt at `www.PacktPub.com`, you can use this to access PacktLib today and view 9 entirely free books. Simply use your login credentials for immediate access.

Table of Contents

Preface

There have been many recent technological advances that have really changed the way we live, work, and play. The television, the computer, and the cell phone all have dramatically affected our lives. Each of these generally started out with a few early adopters, for the most part, individuals with lots of resources that were able to afford the new technology. However, soon after, there was a movement to make the technology more affordable for a wider range of people.

The latest technological movement is robotics. The number, kind, and use of robots is growing dramatically. The first of these robots were developed in university labs or in military research centers. However, just as with the adaption of the computer, there is already a growing grassroots movement of do-it-yourself developers that has sprung up to make robots a part of our everyday life.

This movement has been fueled by inexpensive hardware and free, open source software. However, it has also been enabled by a community of developers who are willing to help others get started or overcome challenges that they have experienced.

This book is offered in the spirit of this do-it-yourself movement. Inside the book, you'll find details about how to take Raspberry Pi B 2, an inexpensive, small, but versatile computer, and marry it with inexpensive hardware and open source software to build a bipedal robot that can walk, sense barriers, and even see its surroundings.

However, be careful—this sort of information can be dangerous. Before long, you may be creating the next generation of thinking, walking, sensing machines that will be at the heart of the robotic revolution.

What this book covers

Chapter 1, *Configuring and Programming Raspberry Pi*, begins with a discussion on how to connect power, and it continues through setting up a full system that's configured and ready to begin connecting any of the amazing devices and Software capabilities to develop advanced robotics applications.

Chapter 2, *Building the Biped*, shows how to construct the mechanics of the biped platform whether you want to use 3D print, purchase, or construct your own legs and body.

Chapter 3, *Motion for the Biped*, talks about how once you have the platform built, you'll need to program it to walk, wave, play dead, or perform any number of neat motion segments so that you can coordinate the movement of your platform.

Chapter 4, *Avoiding Obstacles Using Sensors*, shows you how to add IR sensors so that you can avoid running into barriers.

Chapter 5, *Path Planning and Your Biped*, covers how to plan the movement of your biped. As you move around, you'll want to be able to move from point A to point B.

Chapter 6, *Adding Vision to Your Biped*, provides the details of how to connect a webcam, the hardware, and the software so that we can use it to input visual data into our system.

Chapter 7, *Accessing Your Biped Remotely*, covers the basics of how to configure the Raspberry Pi as a wireless access point so that you can control your biped remotely.

What you need for this book

Here is the list of what you need:

- Raspbian
- putty
- Image Writer for Windows
- libusb-1.0-0-dev
- VncServer

Who this book is for

This book is for anyone who has some background in using the Raspberry Pi to create robotics projects. Some programming background is assumed as you create a biped robot that can walk, sense its environment, plan its movements, and follow movement and color—all autonomously.

Conventions

In this book, you will find a number of text styles that distinguish between different kinds of information. Here are some examples of these styles and an explanation of their meaning.

Code words in text, database table names, folder names, filenames, file extensions, pathnames, dummy URLs, user input, and Twitter handles are shown as follows: "However, you do need to find the `/dev` device label for your card"

Any command-line input or output is written as follows:

```
sudo dd if=2015-01-31-raspbian.img  of=/dev/sdX
```

New terms and **important words** are shown in bold. Words that you see on the screen, for example, in menus or dialog boxes, appear in the text like this: "Clicking the **Next** button moves you to the next screen."

Warnings or important notes appear in a box like this.

Tips and tricks appear like this.

Reader feedback

Feedback from our readers is always welcome. Let us know what you think about this book—what you liked or disliked. Reader feedback is important for us as it helps us develop titles that you will really get the most out of.

To send us general feedback, simply e-mail `feedback@packtpub.com`, and mention the book's title in the subject of your message.

If there is a topic that you have expertise in and you are interested in either writing or contributing to a book, see our author guide at `www.packtpub.com/authors`.

Customer support

Now that you are the proud owner of a Packt book, we have a number of things to help you to get the most from your purchase.

Downloading the color images of this book

We also provide you with a PDF file that has color images of the screenshots/diagrams used in this book. The color images will help you better understand the changes in the output. You can download this file from `https://www.packtpub.com/sites/default/files/downloads/Raspberry_Pi_Robotics_Essentials_Graphics.pdf`.

Errata

Although we have taken every care to ensure the accuracy of our content, mistakes do happen. If you find a mistake in one of our books—maybe a mistake in the text or the code—we would be grateful if you could report this to us. By doing so, you can save other readers from frustration and help us improve subsequent versions of this book. If you find any errata, please report them by visiting `http://www.packtpub.com/submit-errata`, selecting your book, clicking on the **Errata Submission Form** link, and entering the details of your errata. Once your errata are verified, your submission will be accepted and the errata will be uploaded to our website or added to any list of existing errata under the Errata section of that title.

To view the previously submitted errata, go to `https://www.packtpub.com/books/content/support` and enter the name of the book in the search field. The required information will appear under the **Errata** section.

Piracy

Piracy of copyrighted material on the Internet is an ongoing problem across all media. At Packt, we take the protection of our copyright and licenses very seriously. If you come across any illegal copies of our works in any form on the Internet, please provide us with the location address or website name immediately so that we can pursue a remedy.

Please contact us at `copyright@packtpub.com` with a link to the suspected pirated material.

We appreciate your help in protecting our authors and our ability to bring you valuable content.

Questions

If you have a problem with any aspect of this book, you can contact us at `questions@packtpub.com`, and we will do our best to address the problem.

1
Configuring and Programming Raspberry Pi

Robots are beginning to infiltrate our world. They come in all shapes and sizes, with a wide range of capabilities. And, just like the evolution of the personal computer before them, much of what is happening in the robot development world is coming from hobbyists and do-it-yourselfers that are using a new generation of inexpensive hardware and free, open source software to build machines with all kinds of amazing capabilities. In this book, you will learn how to build robots by building a robot, a four-legged quadruped with sensor and vision capabilities. The skills you will learn, however, can also be used on a wide variety of walking, rolling, swimming, or flying robots.

In this chapter, you'll learn:

- How to configure your Raspberry Pi, the control center of your robot, with the Raspbian operating system
- How to set up a remote development environment so you can program your robot
- Basic programming skills in both Python and C so you can both create and edit the programs your robot will need to do all those amazing things

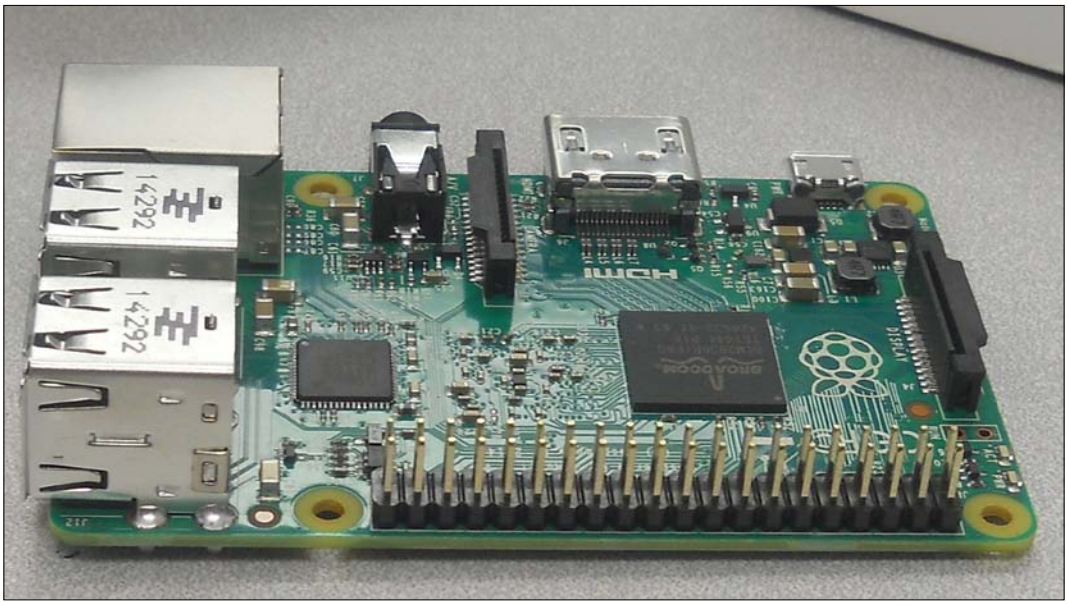
Configuring Raspberry Pi – the brain of your robot

One of the most important parts of your robot is the processor system you use to control all the different hardware. In this book, you'll learn how to use Raspberry Pi, a small, inexpensive, easy-to-use processor system. Raspberry Pi comes in several flavors – the original A and B model, and the new and improved A+ and B+ model. The B+ flavor is the most popular and comes with additional input/output capability, four USB connections, more memory, and will be the flavor we'll focus on in this book.

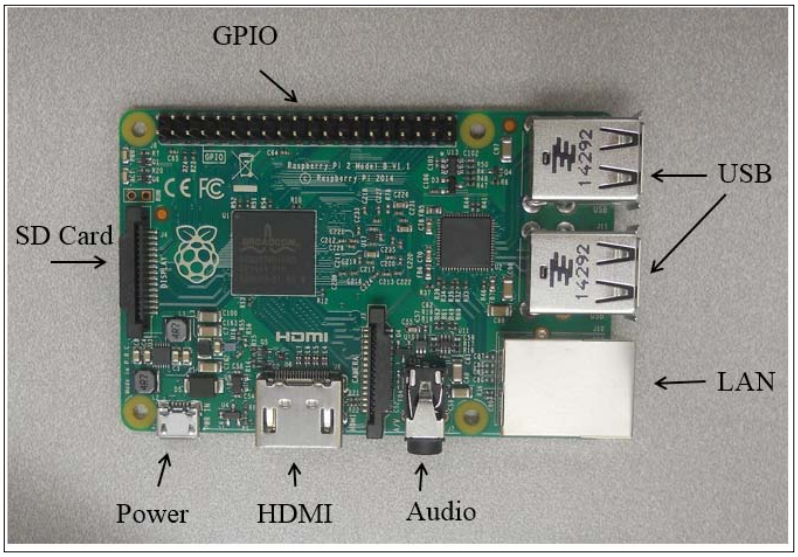
Here are the items you'll need to set up an initial Raspberry Pi development environment:

- A Raspberry Pi, Model B 2. There are three other Raspberry Pi models, the B+, the B, and the A. These are models with less processing power and different hardware configurations. In this book, we'll focus on the Raspberry Pi Model B 2; it has the best processing power and the most useful input/output access. However, many of the items in this book will also work with the Raspberry Pi B+ and A versions, perhaps with some additional hardware.
- The USB cable to provide power to the board.
- A microSD card — at least 4 GB.
- A microSD card writer.
- Another computer that is connected to the Internet.
- An Internet connection for the board — for the initial configuration steps, you'll need a LAN cable and wired LAN connection.
- A wireless LAN device.

Here is what the Raspberry Pi B 2 board looks like:



You should also acquaint yourself with the different connections on the board. Here they are on the B 2, labelled for your information:



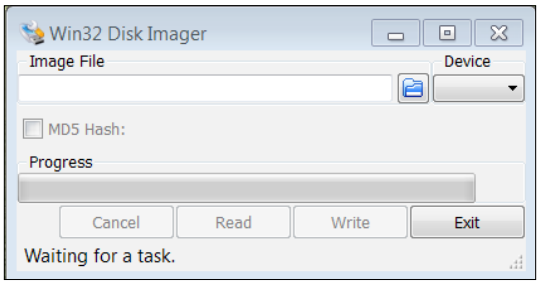
Installing the operating system

Before you get started, you'll need to download and create a card with the Raspbian operating system. You are going to install Raspbian, an open source version of the Debian version of Linux, on your Raspberry Pi.

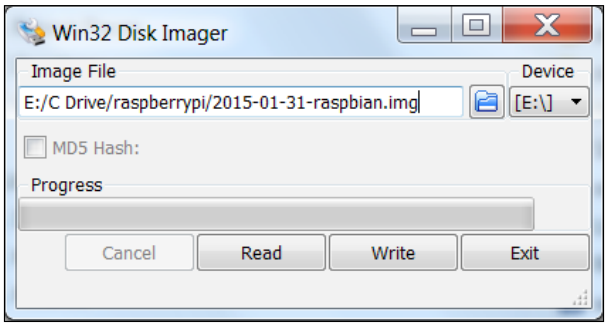
There are two approaches to getting Raspbian on your board. The board is getting popular enough that you can now buy an SD card that already has Rasbpian installed, or you can download it onto your personal computer and then install it on the card. If you are going to download a distribution, you need to decide if you are going to use a Windows computer to download and create an SD card, or a Linux machine.

No matter which machine you are going to use, you'll need to download an image. Open a browser window. Go to the Raspberry Pi site, `www.raspberrypi.org`, and select **Downloads** from the top of the page. This will give you a variety of download choices. Go to the **Raspbian** section and select the `.zip` file just to the right of the image identifier. This will download an archived file that has the image for your Raspbian operating system. Note the default username and password; you'll need those later.

If you're using Windows, you'll need to unzip the file using an archiving program like 7-Zip. This will leave you with a file that has the `.img` extension, a file that can be imaged on your card. Next, you'll need a program that can write the image to the card. Use Image Writer if you are going to create your card using a Windows machine. You can find a link to this program at the top of the download section on the `www.raspberrypi.org` website. Plug your card into the PC, run this program, and you should see this:

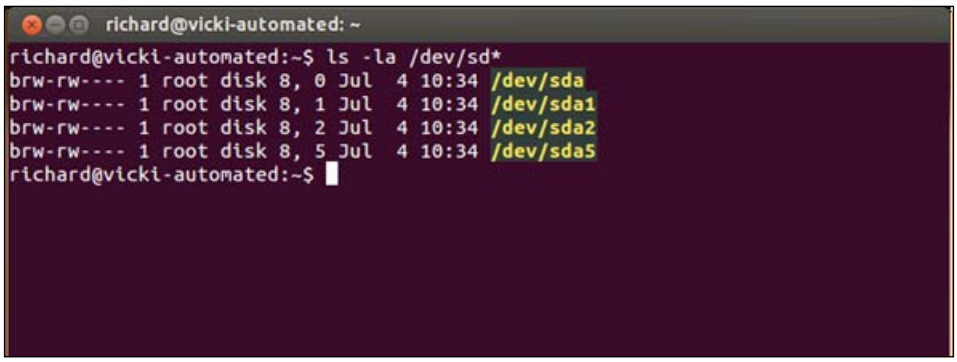


Select the correct card and image; it should look something like this:

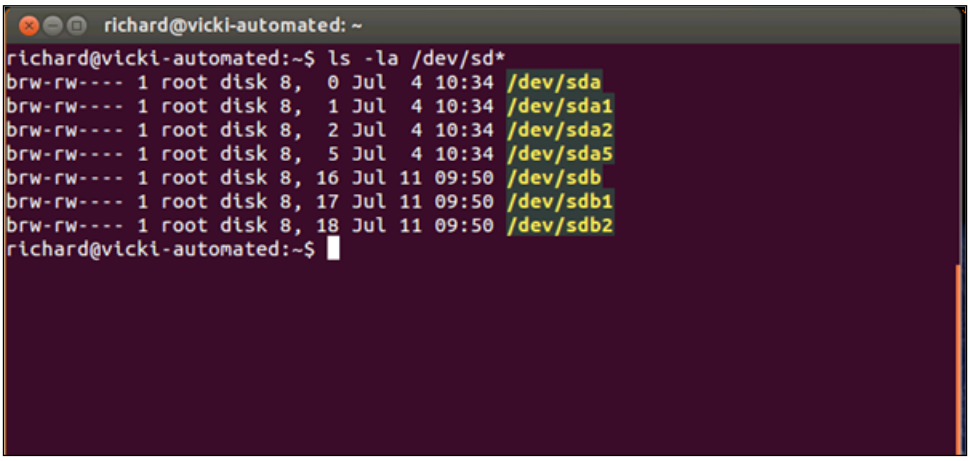


Then select **Write**. This will take some time, but when it is complete, eject the card from the PC.

If you are using Linux, you'll need to unarchive the file and then write it to the card. You can do all of this with one command. However, you do need to find the `/dev` device label for your card. You can do this with the `ls -la /dev/sd*` command. If you run this before you plug in your card, you might see something like the following:



After plugging in your card, you might see something like the following:



Note that your card is `sdb`. Now, go to the directory where you downloaded the archived image file and issue the following command:

```
sudo dd if=2015-01-31-raspbian.img  of=/dev/sdX
```

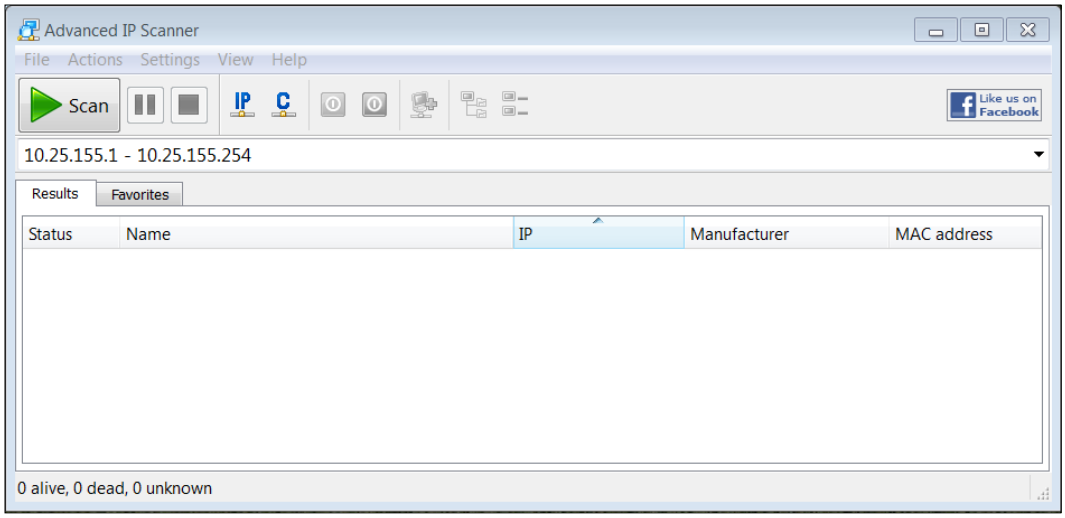
The `2015-01-31-raspbian.img` command will be replaced with the image file that you downloaded, and `/dev/sdX` will be replaced with your card ID; in this example, `/dev/sdb`.

Once your card image has been created, install it on the Raspberry Pi. You'll also need to plug your Raspberry Pi into the LAN cable, and the LAN cable into your wired LAN network.

If you don't have a wired connection, you can complete the following steps by connecting your Raspberry Pi directly to a monitor, keyboard, and mouse.

Power the device. The **POWER LED** should light and your device should boot from the card. To configure the card, you'll need to access it remotely. To do this, you'll now need to connect to the device via SSH, a secure protocol that allows you to control one computer remotely from another computer.

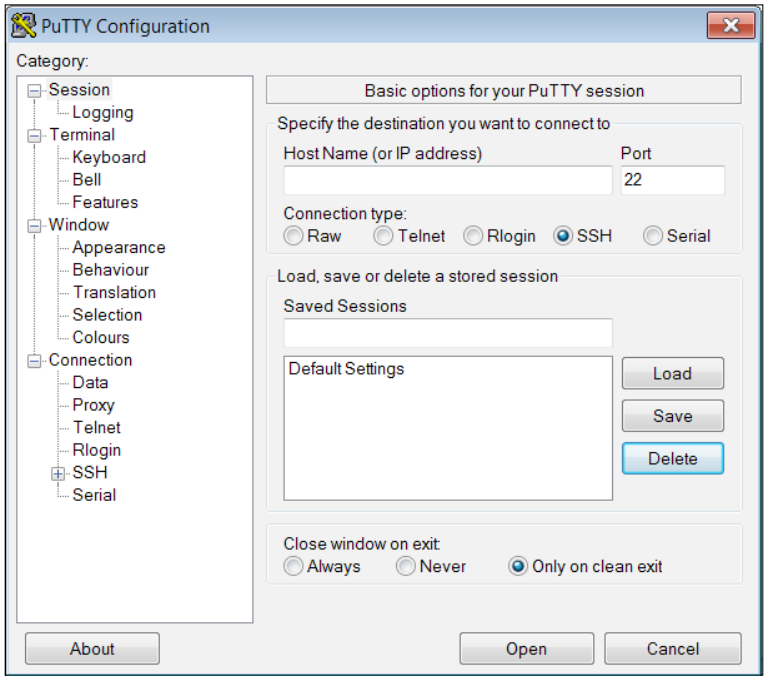
One of the challenges of accessing the system remotely is that you need to know the IP address of your board. There is a way to discover this by using an IP scanner application. There are several scanners available for free; on Windows, a possible choice is Advanced IP Scanner, which is available from `http://www.advanced-ip-scanner.com/`. Here is what the program looks like when it is run:



Clicking on the **Scan** selector scans for all the devices connected to the network. You can also do this in Linux; one application for IP scanning in Linux is called Nmap. To install Nmap, type in `sudo apt-get install nmap`. To run Nmap, type in `sudo nmap -sP 10.25.155.1/154` and the scanner will scan the addresses from 10.25.155.1 to 10.25.155.154.

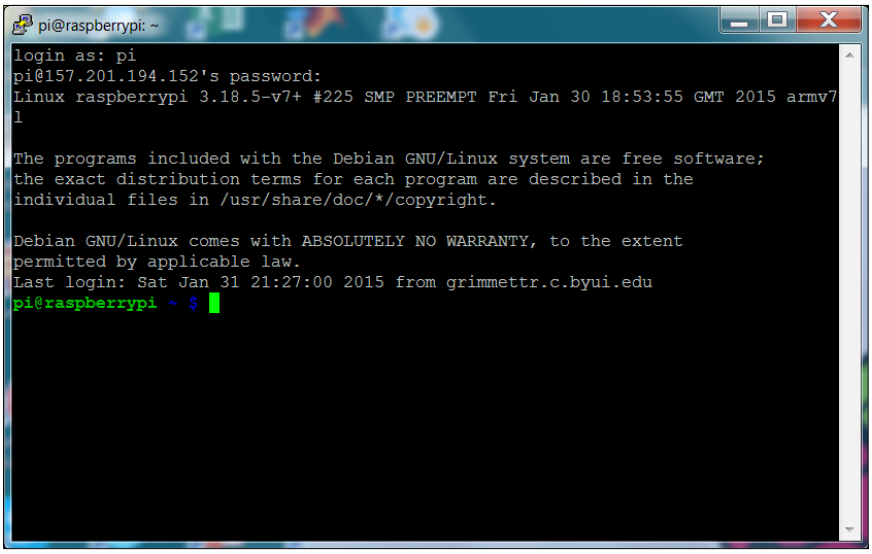
These scanners can let you know which addresses are being used, and this should then let you find your Raspberry Pi IP address. Since you are going to access your device via SSH, you'll also need an SSH terminal program running on your remote computer. If you are running Microsoft Windows, you can download such an application. One simple and easy choice is Putty. It is free and does a very good job of allowing you to save your configuration so you don't have to type it in each time. This program is available at `www.putty.org`.

Download Putty on your Microsoft Windows machine. Then run `putty.exe`. You should see a configuration window. It will look something like the following screenshot:



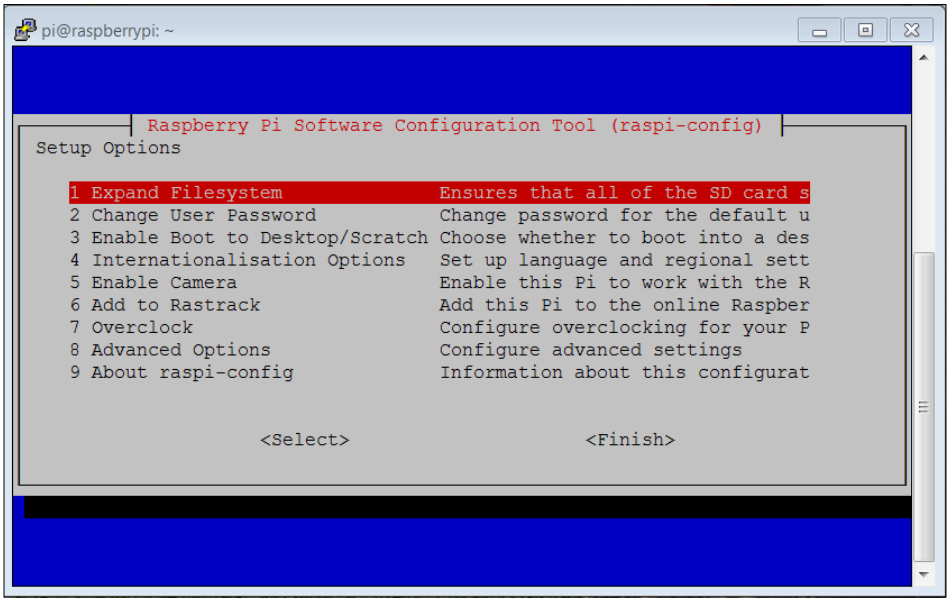
Type in the `inet addr` from the IP Scanner in the **Host Name** space and make sure that the SSH is selected. You may want to save this configuration under Raspberry Pi so you can reload it each time.

When you click on **Open**, the system will try to open a terminal window onto your Raspberry Pi via the LAN connection. The first time you do this, you will get a warning about an RSA key, as the two computers don't know about each other; so Windows is complaining that a computer it doesn't know is about to be connected in a fairly intimate way. Simply click on **OK**, and you should get a terminal with a login prompt, like the following screenshot:

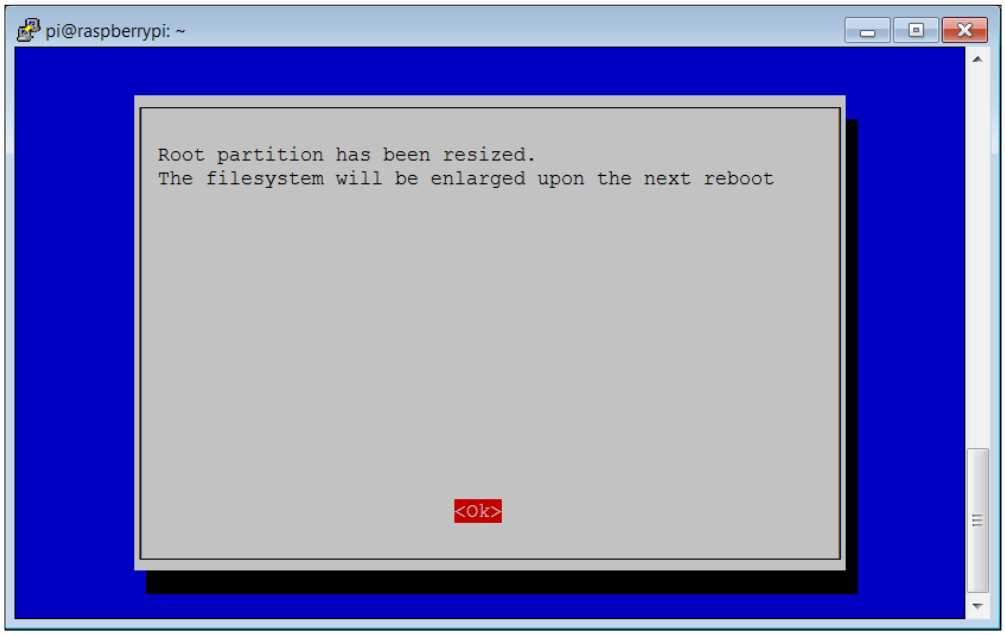


Now you can log in and issue commands to your Raspberry Pi. If you'd like to do this from a Linux machine, the process is even simpler. Bring up a terminal window and then type in `ssh pi@xxx.xxx.xxx.xxx -p 22`, where `xxx.xxx.xxx.xxx` is the `inet addr` of your device. This will then bring you to the login screen of your Raspberry Pi, which should look similar to the previous screenshot.

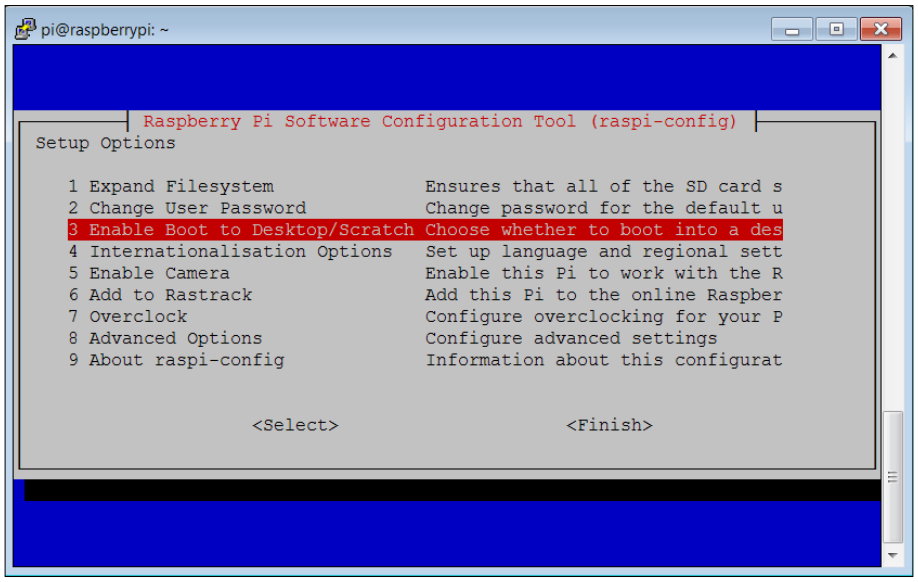
After your log in, you should get a screen that looks like the following:



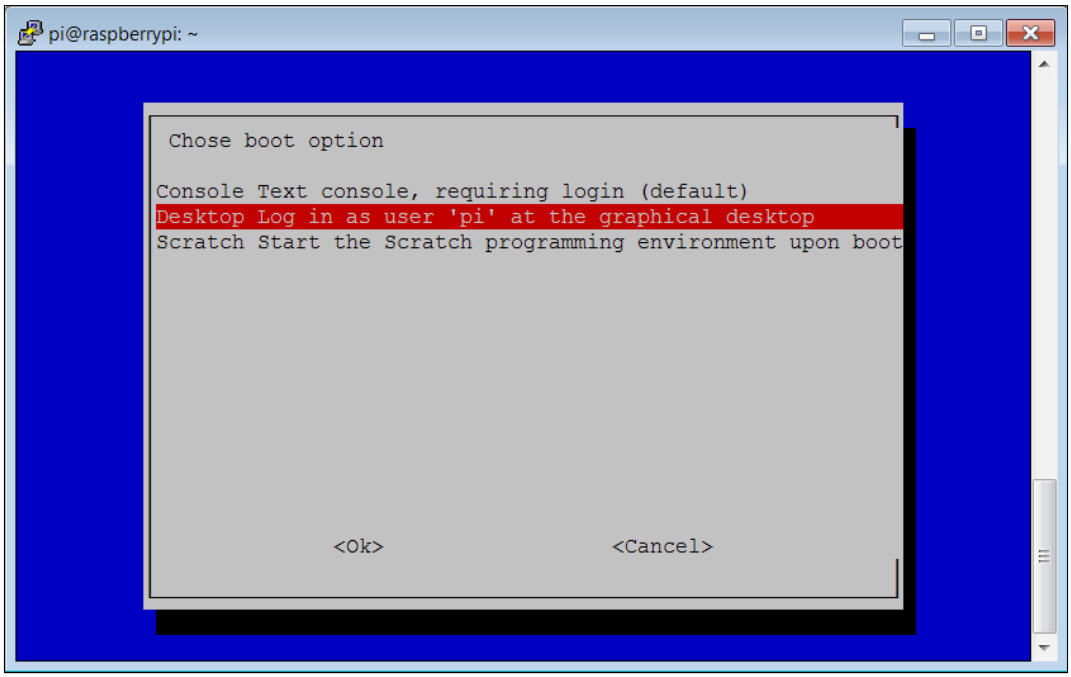
First, you'll want to expand the file system to take up the entire card. So, hit the *Enter* key, and you'll see the following screen:



Hit *Enter* once again and you'll go back to the main configuration screen. Now, select the **Enable Boot to Desktop/Scratch** option.



When you hit *Enter*, you'll see the following screen:



You can also choose to overclock your device. This is a way for you to get higher performance from your system. However, there is a risk that you can end up with a system that has reliability problems.

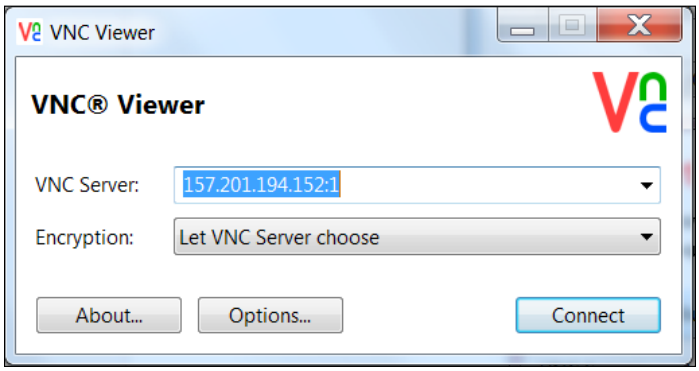
Once you are done and are back at the main configuration menu, hit the *Tab* key until you are positioned over the **<Finish>** selection, then hit *Enter*. Then, hit *Enter* again so that you can reboot your Raspberry Pi. This time, when you log in, you will not see any configuration selections. However, if you ever want to change your configuration choices, you can run the configuration tool by typing in `raspi-config` at the command prompt.

Adding a remote graphical user interface

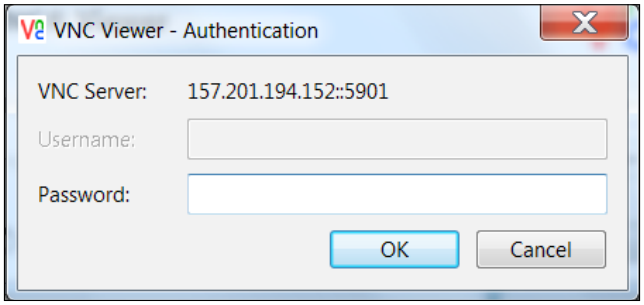
For some steps in your robot build, you will need a graphical look at your system. You can get this on your Raspberry Pi using an application called vncserver. You'll need to install a version of this on your Raspberry Pi by typing in `sudo apt-get install tightvncserver` in a terminal window on your Raspberry Pi.

Tightvncserver is an application that will allow you to remotely view your complete graphical desktop. Once you have it installed, you can do the following:

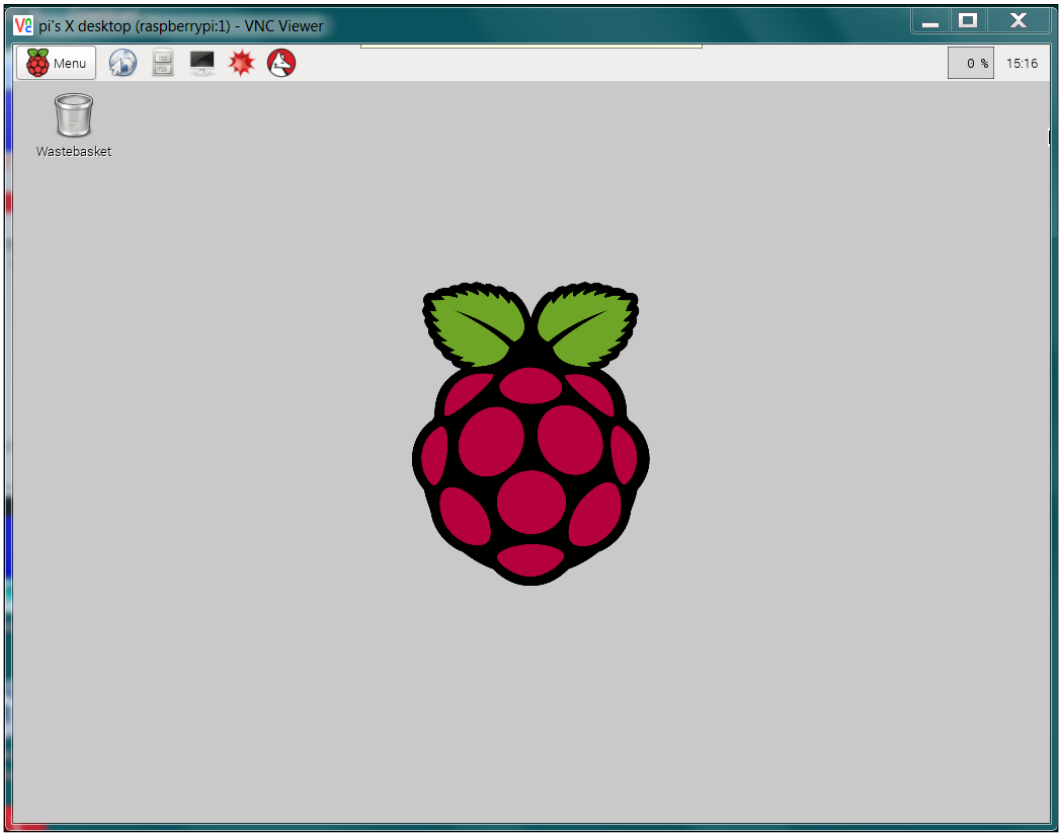
1. You'll need to start the server by typing in `vncserver` in a terminal window on the Raspberry Pi.
2. You will then be prompted for a password, prompted to verify the password, and then asked if you'd like to have a view only password. Remember the password you entered; you'll need it to remotely log in via a VNC viewer.
3. You'll need a VNC viewer application for your remote computer; a good choice is Real VNC, available from `http://www.realvnc.com/download/viewer/`. When you run it, you should see this:



4. Enter the VNC server address, which is the IP address of your Raspberry Pi, and click on **Connect**. You will get a warning about an unencrypted connection; select **Continue** and you will get this pop-up window:



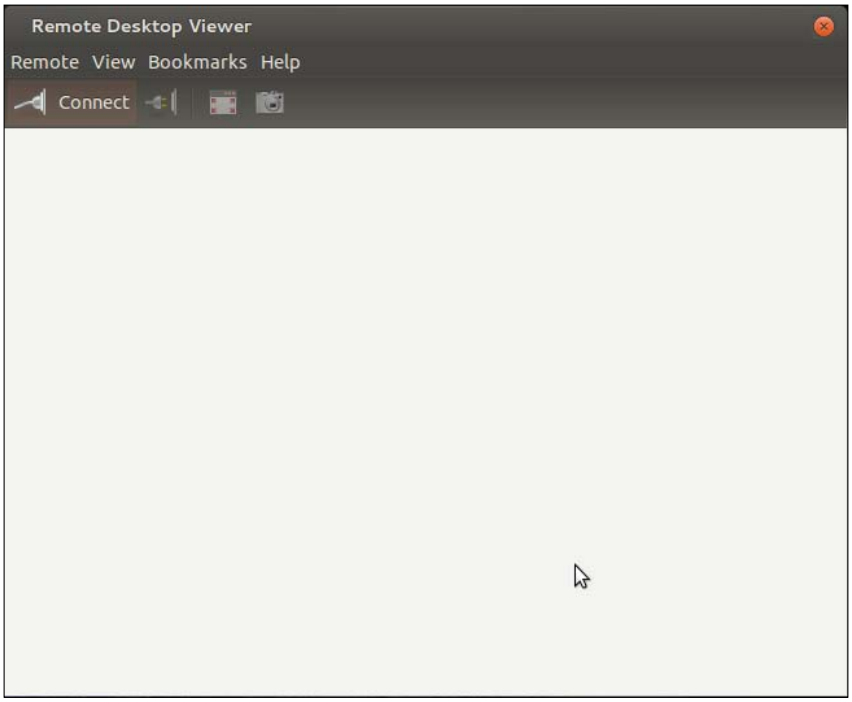
5. Type in the password you just entered while starting the vncserver, and you should then get a graphical view of your Raspberry Pi, which looks like the following screenshot:



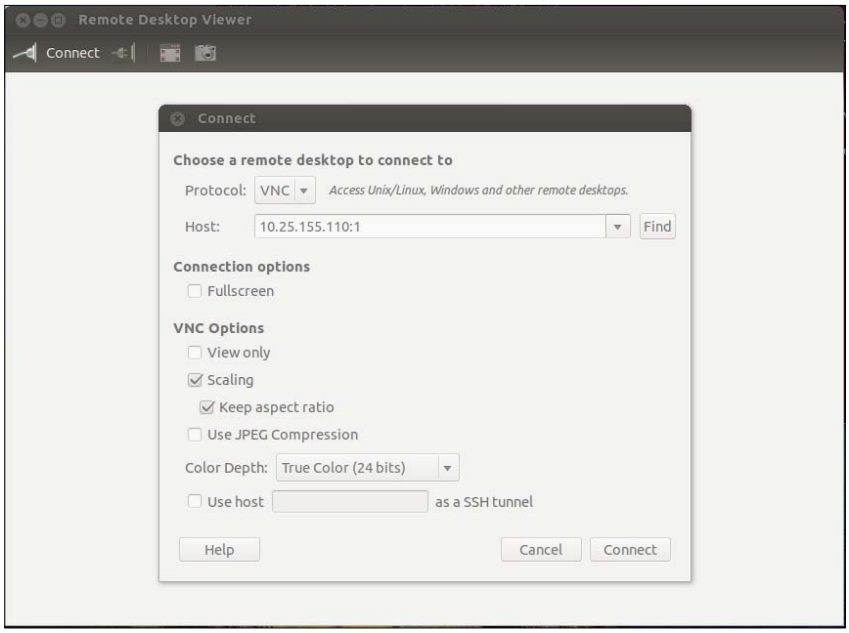
You can now access all the capabilities of your system, albeit they may be slower if you are doing a graphics-intense data transfer. To avoid having to type in vncserver each time you boot your Raspberry Pi, use the instructions at `http://www.havetheknowhow.com/Configure-the-server/Run-VNC-on-boot.html`.

Vncserver is also available via Linux. You can use an application called Remote Desktop Viewer to view the remote Raspberry Pi Windows system. If you have not installed this application, install it using the updated software application based on the type of Linux system you have. Once you have the software, do the following:

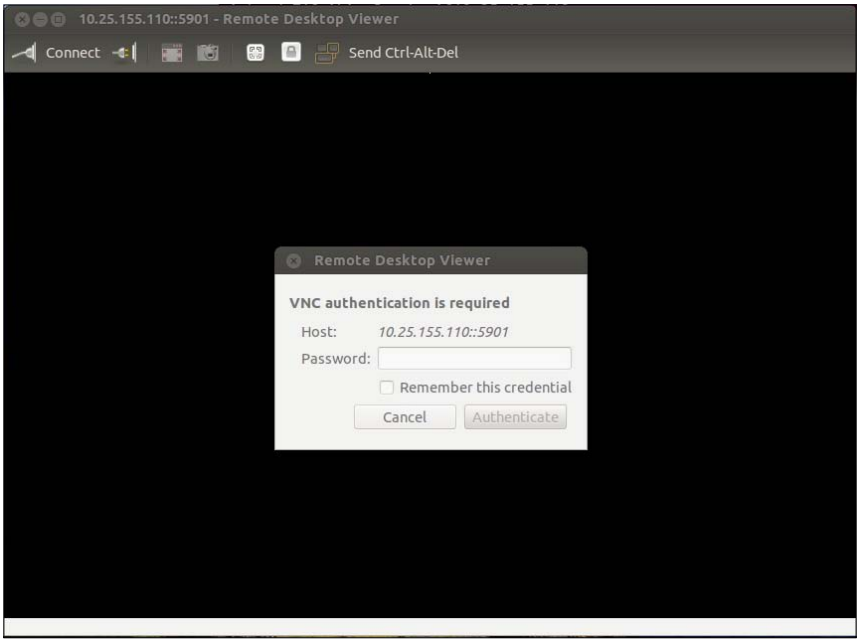
1. Run the application, and you should see the following screenshot:



2. Make sure that vncserver is running on the Raspberry Pi; the easiest way to do this is to log in using SSH and run vncserver at the prompt. Now, click on **Connect** on the **Remote Desktop Viewer**. Fill in the screen as follows. Under the **Protocol** selection, choose **VNC**, and you should see the following screenshot:



3. Now, enter the Host inet address—make sure to include `:1` at the end, and then click on **Connect**. You'll need to enter the vncserver password you set up, like the following screenshot:

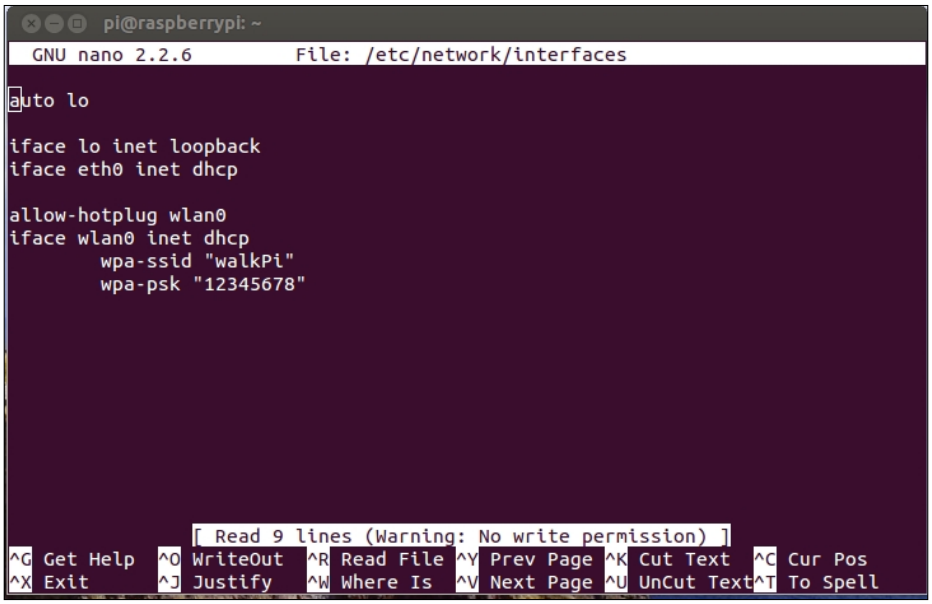


You should now see the graphical screen of the Raspberry Pi. You are ready to start interacting with the system!

Establishing wireless access

Now that your system is configured, the next step is to connect your Raspberry Pi to your remote computer using wireless. To do this, you'll add a wireless USB device and configure it. See `http://elinux.org/RPi_USB_Wi-Fi_Adapters` to identify wireless devices that have been verified to work with Raspberry Pi. Here is one available at many online electronics outlets:

To connect to your wireless LAN, boot the system and edit the network file by typing in `sudo nano /etc/network/interfaces`. Then, edit the file to look like this:



Reboot your device and it should now be connected to your wireless network.

If you are using a US keyboard, you may need to edit the keyboard file for your keyboard to use nano effectively. To do this, type in `sudo nano /etc/default/keyboard` and change `XKBLAYOUT="gb"` to `XKBLAYOUT="us"`.

Your system has lots of capabilities. Feel free to play with the system, which will give you an understanding of what is already there and what you'll want to add from a software perspective.

Programming on Raspberry Pi

One last bit of introduction. You'll need some basic programming skills to be successful on your project. This section will touch a little on Python and C programming on the Raspberry Pi.

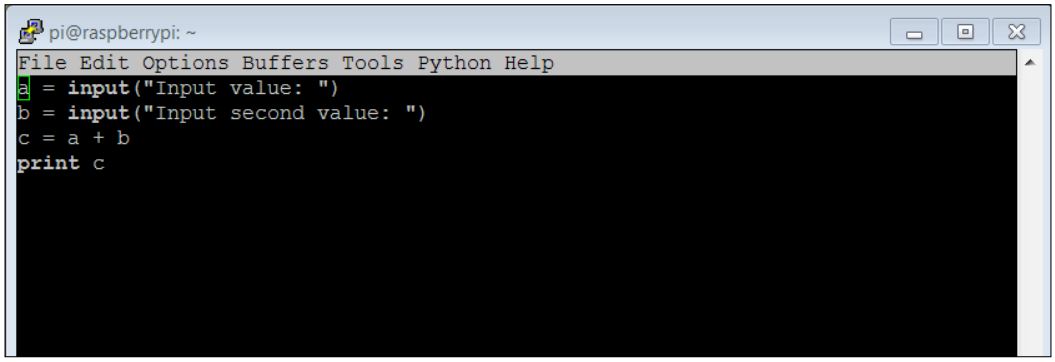
Creating and running Python programs on the Raspberry Pi

You'll be using Python for two reasons. First, it is a simple language that is intuitive and very easy to use. Second, a lot of open source functionality in the robotics world is available in Python. To work the examples in this section, you'll need a version of Python installed. Fortunately, the basic Raspbian system has a version already installed, so you are ready to begin.

If you are new to programming, there are a number of different websites that provide interactive tutorials. If you'd like to practice some of the basic programming concepts in Python using these tools, go to `www.codeacademy.com` or `http://www.learnpython.org/` and give it a try.

But, to get you started, let's first cover how to create and run a Python file. It turns out that Python is an interactive language, so you could run Python and then type in commands one at a time. However, you want to use Python to create programs, so you are going to create Python programs and then run these programs from the command line by invoking Python.

Open an example Python file by typing in `emacs example.py`. Now, put some code in the file. Start with the lines shown in the following screenshot:

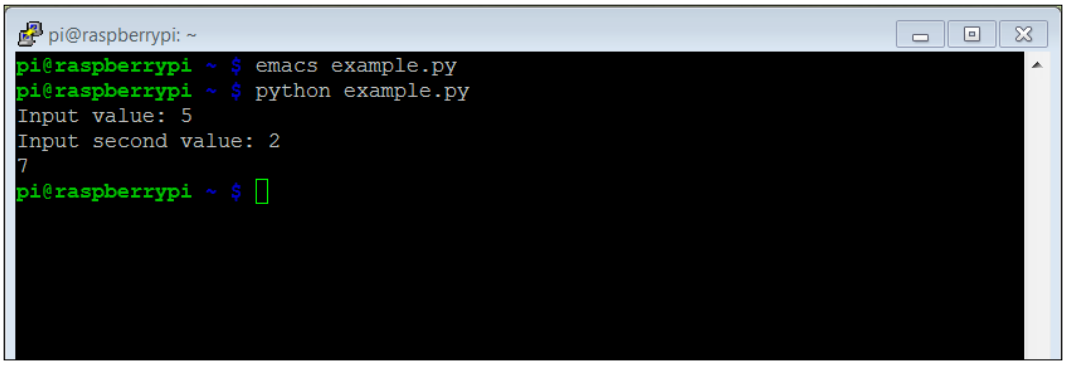


Your code may be color coded. I have removed the color coding here so that it is easier to read.

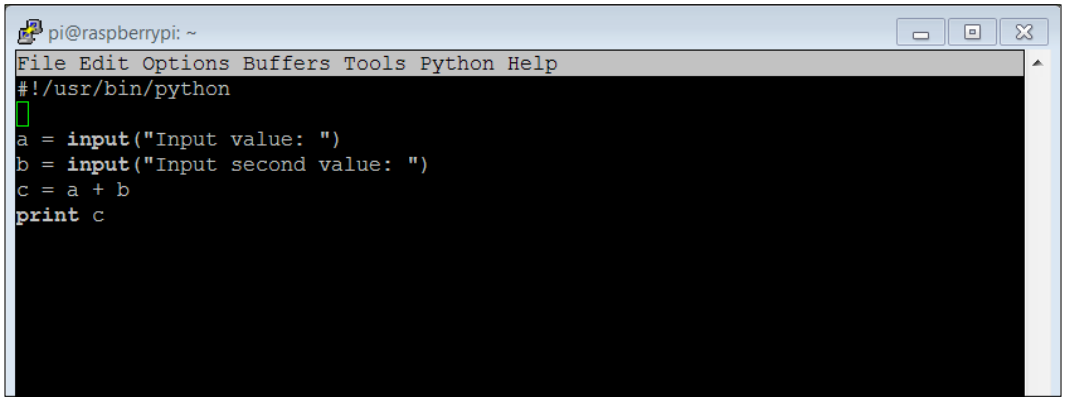
Let's go through the code to see what is happening:

1. `a = input("Input value: ")`: One of the basic needs of a program is to get input from the user. The `raw_input` part allows us to do that. The data will be input by the user and stored in `a`. The prompt `"Input value:"` will be shown to the user.
2. `b = input("Input second value: ")`: This data will also be input by the user and stored in `b`. The prompt `"Input second value:"` will be shown to the user.
3. `c = a + b`: This is an example of something you can do with the data; in this example, you can add `a` and `b`.
4. `print c`: Another basic need of our program is to print out results. The print command prints out the value of `c`.

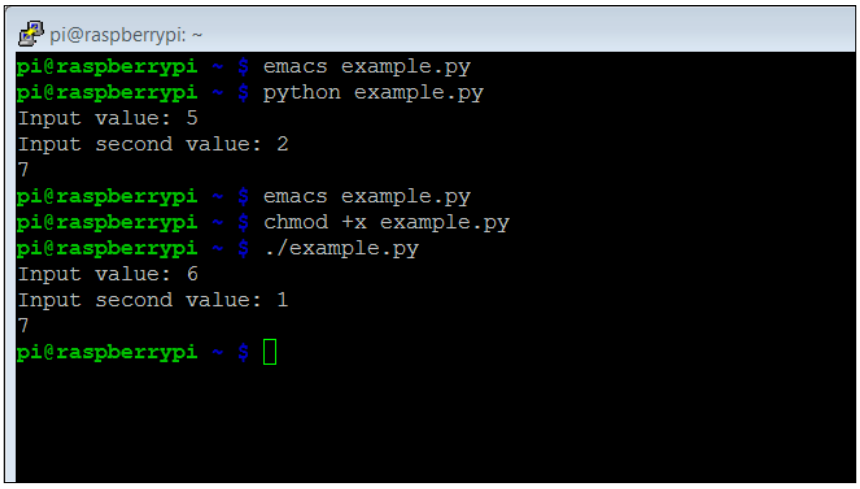
Once you have created your program, save it (using `ctrl-x ctrl-s`) and quit emacs (using `ctrl-x ctrl-c`). Now, from the command line, run your program by typing in `python example.py`. You should see something similar to the following screenshot:



You can also run the program right from the command line without typing in `python example.py` by adding one line to the program. Now, the program should look like the following screenshot:



Adding `#!/usr/bin/python` as the first line simply makes this file available for us to execute from the command line. Once you have saved the file and exited emacs, type in `chmod +x example.py`. This will change the file's execution permissions, so the computer will now believe it and execute it. You should be able to simply type in `./example.py` and the program should run, as shown in the following screenshot:



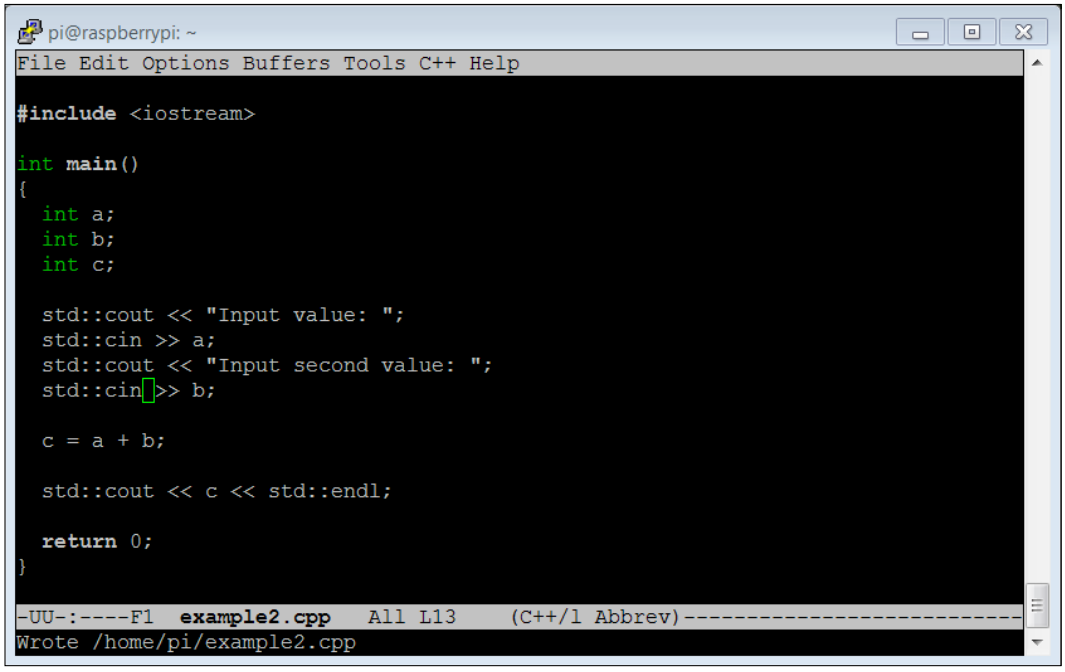
Note that if you simply type in `example.py`, the system will not find the executable file. In this case, the file has not been registered with the system, so you have to give it a path to the file. In this case, `./` is the current directory.

An introduction to the C/C++ programming language

Now that you've been introduced to a simple programming language in Python, you'll also need a bit of exposure to a more complex, but powerful, language called C. C is the original language of Linux and has been around for many decades, but is still widely used by open source developers. It is similar to Python, but is also a bit different, and since you may need to understand and make changes to C code, you should be familiar with it and know how it is used.

As with Python, you will need to have access to the language capabilities. These come in the form of a compiler and build system, which turns your text files into ones that contain programs to machine code that the processor can actually execute. To do this, type in `sudo apt-get install build-essential`. This will install the programs you need to turn your code into executables for the system.

Now that the tools are installed, let's walk through some simple examples. Here is the first C/C++ code example:



The following is an explanation of the code:

- `#include <iostream>`: This is a library that is included so that your program can input data from the keyboard and output information to the screen.
- `int main()`: As with Python, we can put functions and classes in the file, but you will always want to start execution at a known point; C defines this as the `main` function.
- `int a;`: This defines a variable named `a`, of type `int`. C is what we call a strongly typed language, which means that we need to declare the type of the variable we are defining. The normal types are `int`, a number that has no decimal points; `float`, a number that requires decimal points; `char`, a character of text, and `bool`, a `true` or `false` value. Also note that every line in C ends with the `;` character.
- `int b;`: This defines a variable named `b`, of type `int`.
- `int c;`: This defines a variable named `c`, of type `int`.
- `std::cout << "Input value: ";`: This will display the string `"Input value: "` on the screen.
- `std::cin >> a;`: The input that the user types will go into the variable `a`.
- `std::cout << "Input second value: ";`: This will display the string `"Input second value: "` on the screen.
- `std::cin >> b;`: The input that the user types will go into the variable `b`.
- `c = a + b`: The statement is a simple addition of two values.
- `std::cout << c << std::endl;`: The `cout` command prints out the value of `c`. The `endl` command at the end prints out a carriage return so that the next character appears on the next line.
- `return 0;`: The main function ends and returns `0`.

To run this program, you'll need to run a compile process to turn it into an executable program that you can run. To do this, after you have created the program, type in `g++ example2.cpp -o example2`. This will then process your program, turning it into a file that the computer can execute. The name of the executable program will be `example2` (as specified by the name after the `-o` option).