CHAPTER 6

Building Low-Cost MySQL Data Nodes

Data nodes are a key component in IOT solutions. Your solution could use one or more data aggregators to send data to a database server in the cloud or one or more database servers in the solution itself. If your IOT solution uses custom-designed hardware, you may even incorporate a database server on an embedded computing component. Whichever the choice, you need to know more about using a database server in your solution.

At a minimum, you need to know how to get those nodes or components in your solution to send data to the database server. This could be a sensor node with a microcontroller that sends data to a data aggregator, a sensor node that sends data to a data aggregator, or the data aggregator that sends data to the database server.

While there are several choices for a database server including a desktop or server computer, IOT solutions tend to use smaller computing devices like those you saw in Chapter 3 such as a single-board computer. For example, you could use a mini-PC like the pcDuino, a single board computer like the Raspberry Pi, Beaglebone Black, or Intel Galileo.

Since the Raspberry Pi is one of the more popular choices, I will focus on the Raspberry Pi in this chapter, but I include notes about other platforms in case you'd like to use those. Just keep in mind some of these platforms are still evolving and may require more work than the Raspberry Pi. Following the discussion about the Raspberry Pi should give you the background needed for other platforms.

This chapter presents information about how to use data nodes in an IOT solution from a sensor networking point of view, that is, a solution that uses a network of nodes to distribute the processing either for cost or for physical distribution (such as reading sensors around a large agricultural or industrial complex).

You will learn how to create a data node (database server) in this chapter, which features a short introduction to the Raspberry Pi followed by a walk-through of how to set up a MySQL server using a Raspberry Pi. You will also learn how to connect to your database server from your sensor or data aggregation nodes.

Let's begin with a look into the Raspberry Pi.

Introducing the Raspberry Pi

The Raspberry Pi is a small, inexpensive personal computer. Although it lacks the capacity for memory expansion and can't accommodate on-board devices such as CD, DVD, and hard drives,[1] it has everything a simple personal computer requires. There have been several iterations of the Raspberry Pi. The newest version, the Raspberry Pi 2B (`http://raspberrypi.org/products/raspberry-pi-2-model-b/`), has four USB ports, an Ethernet port, HDMI video, and even an audio connector for sound.

[1]But can accept USB-based memory sticks and hard drives.

The Raspberry Pi 2B has a micro SD drive[2] that you can use to boot the computer into any of several Linux operating systems. All you need is an HDMI monitor (or DVI with an HDMI-to-DVI adapter), a USB keyboard and mouse, and a 5V power supply, and you're off and running!

■ **Note** I use the term *micro SD* to refer to a specific media and *SD* to refer to the drive or card in abstract.

You can also power your Raspberry Pi using a USB port on your computer. In this case, you need a USB type A male to micro-USB type B male cable. Plug the type A side into a USB port on your computer and the micro-USB type B side into the Raspberry Pi power port.

There have been many improvements to the Raspberry Pi over the few years it has been around. But the largest improvement is in the area of support. The `Raspberrypi.org` organization has worked very hard to improve the initial experience for new users. There is an easy-to-use and navigate web site that combines all of the old, hard-to-find wikis, lists, charts, and blogs into a central place. You can find just about anything you need to get started on Raspberrypi.org. Figure 6-1 shows an excerpt of the main page.

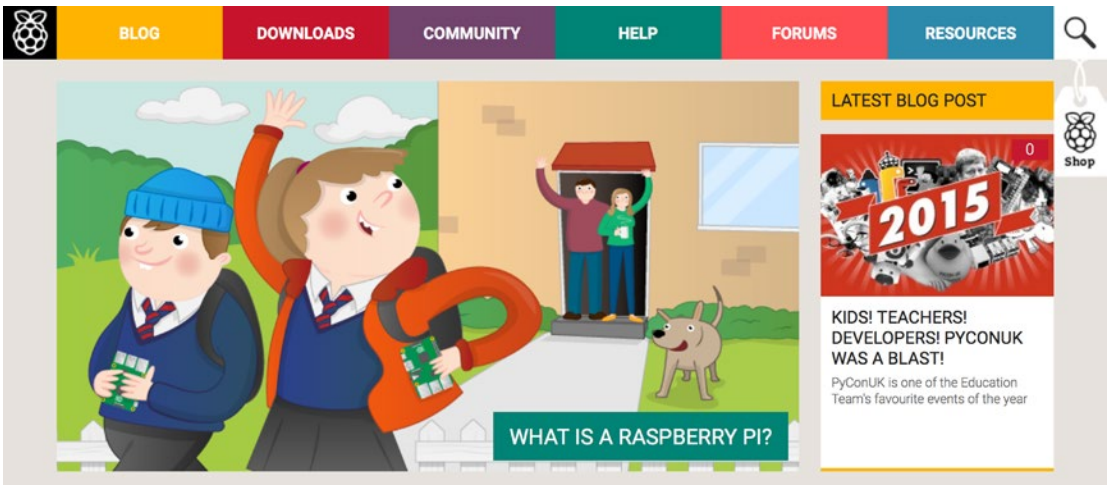


Figure 6-1. *Raspberrypi.org main page (courtesy of Raspberrypi.org)*

The menu across the top provides links to their extensive blogs with examples and how-to articles, a comprehensive downloads page for all the available operating systems and tools, community Raspberry Pi projects, documentation (help), discussion forums, and resources for teachers, students, and makers.

There is also a link to the Raspberry Pi online shop (see the tag to the right with the Raspberry Pi logo) where you can buy boards, accessories, swag, and more. The shop is based in the United Kingdom, but there are links to online retailers in case you want to find a dealer closer to you.

The Raspberry Pi board is available in several versions and comes as a bare board costing as little as $35 (or $5 for the new Raspberry Pi Zero). It can also be purchased online from electronics vendors such as Sparkfun and Adafruit. Most vendors have a host of accessories that have been tested and verified to work with the Raspberry Pi. These include small monitors, miniature keyboards, and even cases for mounting the board.

[2]Micro Secure Digital (micro SD): a small removable memory drive. See `http://en.wikipedia.org/wiki/Secure_Digital`.

In this section, you explore the origins of the Raspberry Pi 2B, take a tour of the hardware connections, and discover what accessories are needed to get starting using the Raspberry Pi.

Noble Origins

The Raspberry Pi was designed to be a platform to explore topics in computer science. The designers saw the need to provide inexpensive, accessible computers that could be programmed to interact with hardware such as servomotors, display devices, and sensors. They also wanted to break the mold of having to spend hundreds of dollars on a personal computer and thus make computers available to a much wider audience.

The designers observed a decline in the experience of students entering computer science curriculums. Instead of having some experience in programming or hardware, students are entering their academic years having little or no experience with working with computer systems, hardware, or programming. Rather, students are well versed in Internet technologies and applications. One of the contributing factors cited is the higher cost and greater sophistication of the personal computer, which means parents are reluctant to let their children experiment on the family PC.

This poses a challenge to academic institutions, which have to adjust their curriculums to make computer science palatable to students. They have had to abandon lower-level hardware and software topics because of students' lack of interest or ability. Students no longer want to study the fundamentals of computer science such as assembly language, operating systems, theory of computation, and concurrent programming. Rather, they want to learn higher-level languages to develop applications and web services. Thus, some academic institutions are no longer offering courses in fundamental computer science.[3] This could lead to a loss of knowledge and skillsets in future generations of computer professionals.

To combat this trend, the designers of the Raspberry Pi felt that, equipped with the right platform, youth could return to experimenting with personal computers as in the days when PCs required a much greater commitment to learning the system and programming it in order to meet your needs. For example, the venerable Commodore 64, Amiga, and early Apple and IBM PC computers had limited software offerings. Having owned a number of these machines, I was exposed to the wonder and discovery of programming at an early age.[4]

WHY IS IT CALLED RASPBERRY PI?

The name was partly derived from design committee contributions and partly chosen to continue a tradition of naming new computing platforms after fruit (think about it). The Pi portion comes from Python, because the designers intended Python to be the language of choice for programming the computer. However, other programming language choices are available.

The Raspberry Pi is an attempt to provide an inexpensive platform that encourages experimentation. The following sections explore more about the Raspberry Pi, including the models available, required accessories, and where to buy the boards.

[3]My alma mater has suffered a similar transition. I mourn for the loss of knowledge.
[4]My first real computer was an IBM PCjr. I followed it by building my own IBM PC AT computer, complete with a 10MB hard drive. Ah, those were the glory days of personal computers!

Models

The choices for Raspberry Pi boards have grown to include four models with several versions and iterations. I outline each of the choices here. Figures 6-2 through 6-4 show representations of each.

- *Raspberry Pi 2 Model B*: Second-generation board with a faster processor and 1Gb RAM but retains the layout of the Model B.
- *Raspberry Pi 1 Model B+*: New generation board with more GPIO pins, more USB ports (four), and a micro SD card slot. It also requires a bit less power than older boards.
- *Raspberry Pi Model A+*: Same new features of the Model B but cheaper with fewer USB ports, no Ethernet, and a slightly smaller footprint.
- *Compute Module Development Kit*: Incorporates a new, smaller edge-mounted Raspberry Pi module and expanded GPIO pins (120 instead of 40). It is designed for industrial applications.

■ **Note** At the time of this writing, the Raspberry Pi Zero was released. It has the same processor and memory as the original Raspberry Pi but lacks the Ethernet and auxiliary video ports. This was done to keep the cost to an amazingly low $5.00 and a form factor down to the size of a pack of chewing gum.

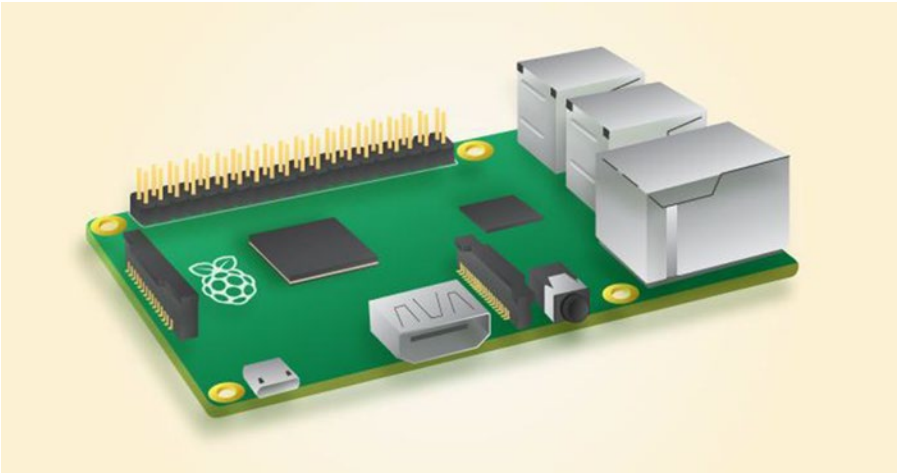


Figure 6-2. *Raspberry Pi 2 Model B and Pi 1 Model B+ (courtesy of Raspberrypi.org)*

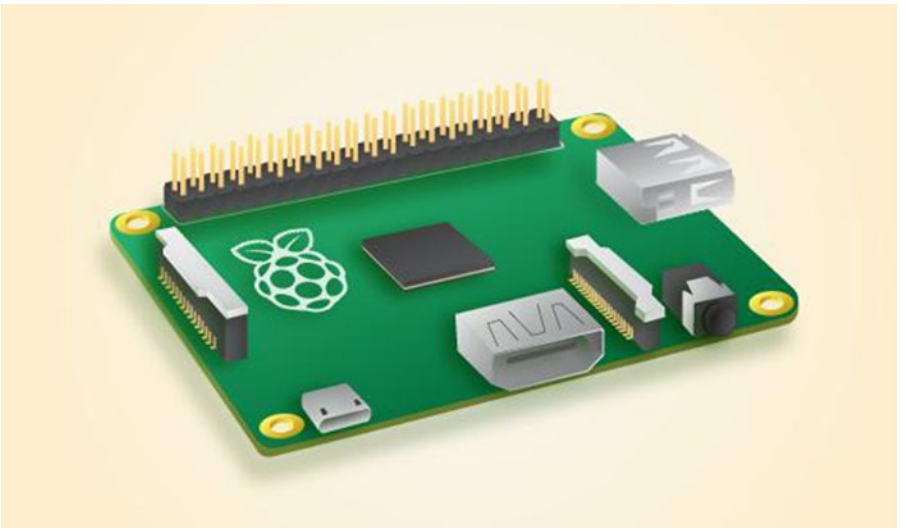


Figure 6-3. *Raspberry Pi Model A+ (courtesy of Raspberrypi.org)*

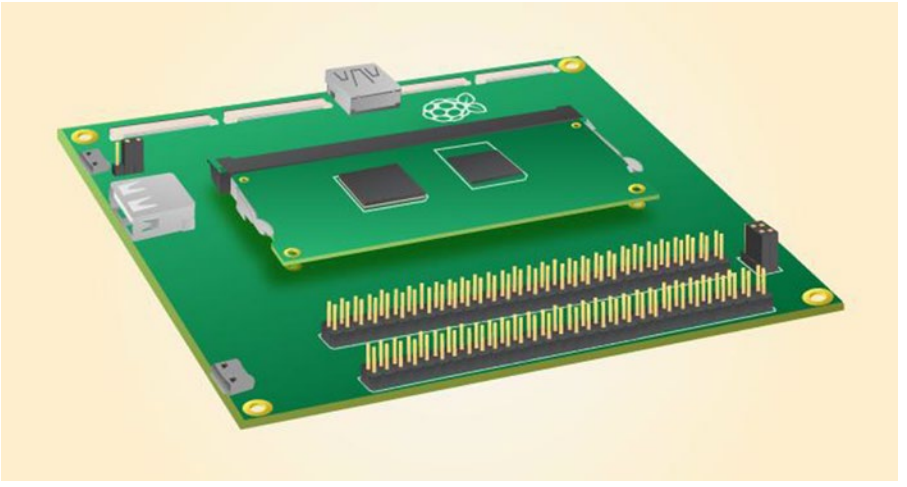


Figure 6-4. *Raspberry Pi Compute Module Development Kit (courtesy of Raspberrypi.org)*

WHAT HAPPENED TO THE $20 RASPBERRY PI?

If you have been following the release of the Raspberry Pi in various media, you have probably heard that the boards were priced at a mere $25. However, most retailers list the Raspberry Pi for $35 or more. Why is that?

The simple answer is the Model A+ is the one priced at $25, whereas the Model B's cost a bit more at $35. This is because the Model B has a few more features specifically Ethernet. If you do not need Ethernet or other B-specific options, you can save a bit by buying the A+.

However, because of supply and demand, you are likely to see average prices for either board (in the United States) at $40 or more. Shop wisely.

Figure 6-2 is a good representation of the Model B series. The Pi 2 and Pi 1 are difficult to distinguish. You must examine the printing on the board itself to determine the differences. There are subtle differences, but they are difficult to see. Fortunately, since they use the same layout, most Model B cases will fit both boards.

The examples in this chapter and the remaining chapters use the Model B variant.

A Tour of the Board

Not much larger than a deck of playing cards, the Raspberry Pi board contains a number of ports for connecting devices. This section presents a tour of the board. If you want to follow along with your board, hold it with the Raspberry Pi logo face up. I will work around the board clockwise.

In the center of the near side you see an HDMI connector. To the left is a microUSB connector used to supply power to the board, and on the right is an audio port. The power connector is known to be a bit fragile on some boards, so take care plugging and unplugging it. Be sure to avoid putting extra strain on this cable while using your Raspberry Pi.

The HDMI port is the primary way to connect a monitor. However, there is a small ribbon cable connector on the left called a DSI video connector. The 7" Raspberry Pi Touch Display (http://element14.com/community/docs/DOC-78156?ICID=hp-7inchpidisplay-ban) can be connected here to provide a really nice, small tablet-like experience. Figure 6-5 shows the Raspberry Pi Touch Display.

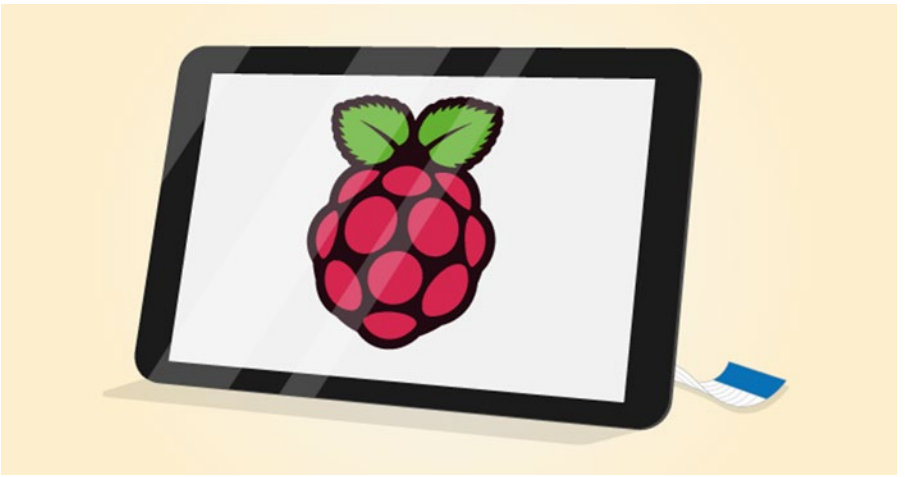


Figure 6-5. *Raspberry Pi 7" Touch Display (courtesy of Raspberrypi.org)*

There is also a camera connector located behind the Ethernet port for connecting a camera (useful for many applications).

■ **Note** Some enterprising makers have developed enclosures for the new monitor including a tablet form factor that can be 3D printed. See `http://thingiverse.com` for the latest prototypes and designs.

On the left side bottom of the board is the micro SD card slot on the bottom.

On the far side and on top of the board is the general-purpose input/output (GPIO) header (a double row pins), which can be used to attach to sensors and other devices.

On the right side of the board is where most of the connectors are placed. There are four USB connectors and the Ethernet connector. An external-powered USB hub connected to the USB ports on the Raspberry Pi can power some boards, but it is recommended that you use a dedicated power supply connected to the micro-USB connector.

Take a moment to examine the top and bottom faces of the board. As you can see, components are mounted on both sides. This is a departure from most boards that have components on only one side. The primary reason the Raspberry Pi has components on both sides is that it uses multiple layers for trace runs. This permits the board to be much smaller and enables the use of both surfaces. This is probably the most compelling reason to consider using a case—to protect the components on the bottom of the board and thus avoid shorts and board failure.

Required Accessories

The Raspberry Pi is sold as a bare system board with no case, power supply, or peripherals. Depending on how you plan to use the Raspberry Pi, you need a few commonly available accessories. If you have been accumulating spares like me, a quick rummage through your stores may locate most of what you need.

If you want to use the Raspberry Pi in console mode (no graphical user interface), you need a USB power supply, a keyboard, and an HDMI monitor (or the 7" Touch Display). The power supply should have a minimal rating of 700mA or greater operating at 5V. If you want to use the Raspberry Pi with a graphical user interface, you also need a pointing device (such as a mouse).

If you have to purchase these items, stick to the commonly available brands and models without extra features. For example, avoid the latest multifunction keyboard and mouse. Chances are, they require drivers that are not available for the various operating system choices for the Raspberry Pi.

You also must have a micro SD card. I recommend a 8GB or higher version. Recall that the micro SD is the only onboard storage medium available. You will need to put the operating system on the card, and any files you create will be stored on the card. I will demonstrate this in a later section.

If you want to use sound in your applications, you also need a set of powered speakers that accept a standard 3.5mm audio jack. Finally, if you want to connect your Raspberry Pi to the Internet, you need an Ethernet cable or a Raspberry Pi–compatible USB Wi-Fi dongle.

HOW CAN I TELL IF MY DEVICE WILL WORK?

If you want to make sure your device will work with the Raspberry Pi, the simplest thing to do is try it! If you prefer not to take chances, you can check the Raspberry Pi hardware compatibility list at http://elinux.org/RPi_VerifiedPeripherals. This list contains many devices and commentary from various users in the community who have tested the devices. If you are just starting out with the Raspberry Pi, look for devices that require little or no extra configuration or drivers.

Recommended Accessories

I highly recommend at least adding small rubber or silicone self-adhesive bumpers to keep the board off your desk. On the bottom of the board are many sharp prongs that can come into contact with conductive materials, which can lead to shorts or, worse, a blown Raspberry Pi. These bumpers are available at most home-improvement and hardware stores.

If you plan to move the board from room to room or you want to ensure that your Raspberry Pi is well protected against accidental damage, you should consider purchasing a case to house the board. Many cases are available, ranging from simple snap-together models to models made from laser-cut acrylic or even milled aluminum. The following list includes several excellent choices, ranging from inexpensive to top-of-the-line luxury cases:

- *Raspberry Pi 2B+ Clear case*: http://sparkfun.com/products/12996
- *Pi Tin*: http://sparkfun.com/products/13102
- *Bel-Aire*: http://makershed.com/products/the-bel-aire
- *Adafruit Pi Box*: http://adafruit.com/products/1985
- *Ninja Pibow*: http://adafruit.com/products/2081
- *Unibody Aluminum Case*: http://adafruit.com/products/2198

Another option is to print your own case with a 3D printer. If you do not have a 3D printer, you may find one in your local library or community college or perhaps a friend of a friend. If you ask nicely, chances are most 3D printer enthusiasts would be happy to print you a case. Indeed, there are many such case designs available for downloading and printing. Figure 6-6 shows one I printed on my 3D printer.

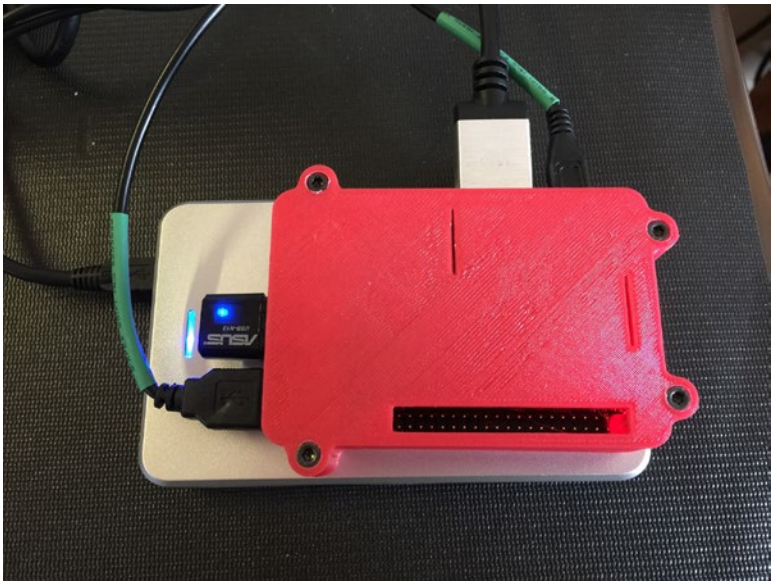


Figure 6-6. *3D-printed Raspberry Pi 2 Model B+ case*

■ **Tip** If you plan to experiment with the GPIO pins or require access to the power test pins or the other ports located on the interior of the board, you may want to consider either using the self-adhesive bumper option or ordering a case that has an open top to make access easier. Some cases are prone to breakage if opened and closed frequently.

Aside from a case, you should also consider purchasing (or pulling from your spares) a powered USB hub. The USB hub power module should be 700–1000mA. A powered hub is required if you plan to use USB devices that draw a lot of power, such as a USB hard drive or a USB soft missile launcher.

■ **Caution** Because the board is small, it is tempting to use it in precarious places like in a moving vehicle or on a messy desk. Ensure that your Raspberry Pi is in a secure location. The power, HDMI, and micro SD card slots seem to the most vulnerable connectors.

Where to Buy

The Raspberry Pi has been available in Europe for some time. It is getting easier to find, but few brick-and-mortar stores stock the Raspberry Pi. Fortunately, a number of online retailers stock it, as well as a host of accessories that are known to work with the Raspberry Pi. The following are some of the more popular online retailers with links to their Raspberry Pi catalog entry:

- *Sparkfun*: `http://sparkfun.com/categories/233`
- *Adafruit*: `http://adafruit.com/category/105`
- *Maker Shed*: `http://makershed.com/collections/raspberry-pi`

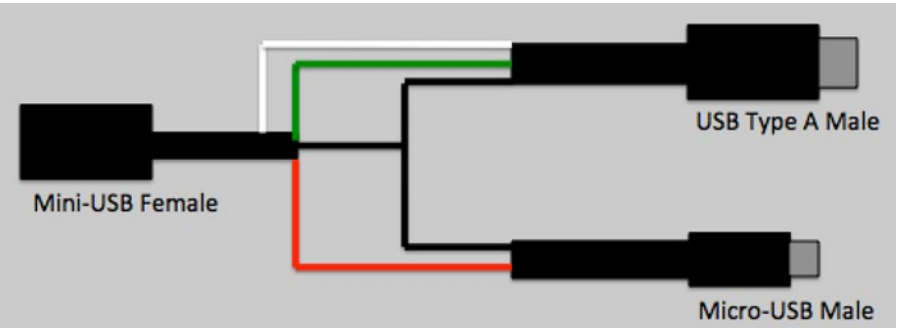
A RASPBERRY PI LAPTOP?

The Raspberry Pi has made a significant contribution to physical computing. Not only does it enable more sophisticated sensor nodes, but it also makes a decent lightweight general-purpose computer. With a proper monitor, mouse, and storage device, you can do most Internet and modest productivity tasks. In fact, some people have replaced their home desktop computer with a Raspberry Pi!

If you are like me and you need to be able to work from anywhere,[5] using a Raspberry Pi may not be very convenient, given that you must have a separate monitor and keyboard. Wouldn't it be great if you could take your Raspberry Pi with you? Well, now you can!

What you need is a surplus Atrix Lapdock from Motorola. Originally designed to allow the Aria phone to be used as a laptop, the Lapdock provides an 11.6" HDMI monitor, a USB keyboard, a mouse, a two-port USB hub, and speakers. More important, it is battery powered and can easily power the Raspberry Pi. The Lapdock has mini-HDMI and mini-USB ports that can be connected to the Raspberry Pi without modifying the Lapdock.

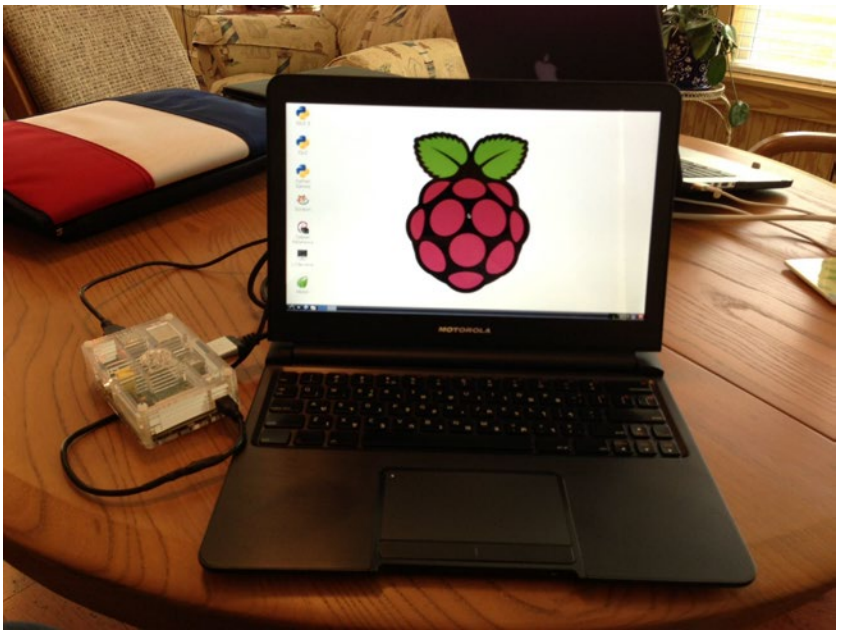
But there is a catch: you must purchase a mini-HDMI female-to-female adapter and a mini-HDMI male to HDMI male cable[6] and build your own Frankenstein USB cable from a micro-USB extension cable and a type A USB cable. The custom cable is needed to allow the Raspberry Pi to use the USB keyboard and mouse as well as power the board. The following figure shows how the cable is constructed. You can find a detailed tutorial video at www.adafruit.com/blog/2012/09/10/cables-adapters-for-the-atrix-raspberry-pi-laptop/.



To build the cable, cut a normal USB type A connector from a standard USB cable,[7] bisect a standard micro-USB extension cable, and splice the wires as shown here. You can abandon the wires in the ends that have no connections shown. The micro-USB male connector will be used to power the Raspberry Pi, and the USB type A male connector will provide connectivity to the Lapdock keyboard and mouse. Once you have this cable made, you're ready to go. For the best results, I recommend a sturdy case to mount your Raspberry Pi so as not to damage it during transport.

[5]Places like your couch, favorite recliner, patio, coffee bar, and so on.
[6]Be sure to select a cable that supports device sensing. If your Lapdock does not power on when the Raspberry Pi is connected, it is most likely the HDMI cable. Try another cable.
[7]Be sure to get permission before "borrowing" a cable from a friend or spouse—it won't be usable as a standard cable when you're finished modifying it.

To use the Raspberry Pi laptop, start by connecting the HDMI cable, then any peripherals (like a hard drive), and then the USB cable; open the lid of the Lapdock. Within a few seconds, your new laptop is ready to go! The next figure shows the ports on the rear of the Lapdock under the folding door, and the one after that shows the laptop in action.



Not only can you use this solution for your Raspberry Pi, but also several other boards work equally well. I have successfully used the Lapdock to power other boards. Indeed, the Lapdock makes an excellent pcDuino laptop! Even if your board doesn't support the keyboard or mouse, the HDMI screen is handy.

The best part of this project is the cost. You can find used and surplus Lapdock on auction sites and similar electronics clearance stores. For example, on eBay the Lapdock is priced at about $60. You can also find the cables online or at most electronics stores. However, the female-to-female mini-HDMI adapter is a bit harder to find. I was able to purchase one on eBay from a dealer in China. Shipping was surprisingly fast, and the cost was reasonable. My cost for a mobile Raspberry Pi (not including the Raspberry Pi) was less than $100.

The next section presents a short tutorial on getting started using the Raspberry Pi. If you have already learned how to use the Raspberry Pi, you can skim the section to see the latest improvements in getting your Raspberry Pi up and running.

Raspberry Pi Tutorial

The following sections present a short tutorial on getting started with your new Raspberry Pi, from a bare board to a fully operational platform. A number of excellent works cover this topic in much greater detail. If you find yourself stuck or wanting to know more about beginning to use the Raspberry Pi and more about the Raspbian operating system, see *Learn Raspberry Pi with Linux* by Peter Membrey and David Hows (Apress, 2012). If you want to know more about using the Raspberry Pi in hardware projects, an excellent resource is *Practical Raspberry Pi* by Brendan Horan (Apress, 2013).

As mentioned in the "Required Accessories" section, you need a micro SD card, a USB power supply rated at 700mA or better with a male micro-USB connector, a keyboard, a mouse (optional), and an HDMI monitor, an HDMI TV, or a DVI monitor with an HDMI adapter. However, before you can boot your Raspberry Pi and bask in its brilliance, you need to create a boot image for your micro SD card.

Choosing a Boot Image (Operating System)

The first thing you need to do is decide which operating system variant you want to use. There are several excellent choices, including the standard Raspbian "Jessie" variant. Each is available as a compressed file called an *image* or *card image*. You can find a list of recommended images along with links to download each on the Raspberry Pi foundation download page: `www.raspberrypi.org/downloads`. The following images are available at the site:

- *Raspbian (Jessie)*: Debian-based official operating system and contains a graphical user interface (Lightweight X11 Desktop Environment [LXDE]), development tools, and rudimentary multimedia features.
- *Ubuntu Mate*: Features the Ubuntu desktop and a scaled-down version of the Ubuntu operating system. If you are familiar with Ubuntu, you will feel at home with this version.
- *Snappy Ubuntu Core*: Developer's edition of core Ubuntu system; same as Mate but with addition of the developer core utilities.
- *Windows 10 IOT Core*: Windows 10 for the IOT. Microsoft's premier IOT operating system. Yes, it does look and feel like Windows, but without the heavy graphical user interface.
- *OSMC*: Open source media center. Build yourself a media center.
- *OpenElec*: Open embedded Linux entertainment center. Another media center option.

- *PiNet*: Classroom management system. A special edition for educators using the Raspberry Pi in the curriculum.
- *RISC OS*: Non-Linux, Unix-like operating system. If you know what IBM AIX is or you've used other Unix operating systems, you'll recognize this beastie.

■ **Tip** If you are just starting with the Raspberry Pi, you should use the Raspbian image. This image is also recommended for the examples in this book.

There are a few other image choices, including a special variant of the Raspbian image from Adafruit. Adafruit calls its image *occidentals* and includes a number of applications and utilities preinstalled, including Wi-Fi support and several utilities. Some Raspberry Pi examples—especially those from Adafruit—require the occidentals image. You can find out more about the image and download it at http://learn.adafruit.com/adafruit-raspberry-pi-educational-linux-distro/overview.

There are two methods for installing the boot image. First, you can use the automated, graphical user interface platform named New Out Of the Box Software (NOOBS[8]), or you can install your image from scratch onto a micro SD drive. Both require downloading and formatting the micro SD drive.

If you are just starting out, the NOOBS solution is by far the easiest. It will take a bit longer to get going (but not much) and simplifies the process. Aside from formatting the micro SD card, everything is automated. I present both options in the following sections.

Using NOOBS

NOOBS is by far the best way to get your Raspberry Pi up and running. With NOOBS, you download a base installer image that contains Raspbian Jessie. You can choose to install it or configure NOOBS to download one of the other images and install it. But first, you have to get the NOOBS boot image and copy it to your micro SD drive.

Begin by downloading the NOOBS installer from http://raspberrypi.org/downloads/noobs/. You will see two options, a network installer (sometimes referred to as the *offline installer*) that includes the Raspbian image or a base image that does not contain any operating systems. This base image is what you would use if you wanted to use the automated installer with an operating system not already included such as Adafruit's version.

■ **Tip** If your download bandwidth is limited, online retailers offer a preconfigured micro SD card that includes NOOB. In fact, you can often find micro SD cards with any of the popular Raspberry Pi operating systems. Just plug it in and go. They usually cost a few dollars more than a blank card of the same size.[9]

---

[8]Not to be confused with noob, which is a bit derogatory. See https://en.wiktionary.org/wiki/noob.
[9]The operating system is free. You're just paying for the convenience of someone having formatted and installed the image on the card for you.

Once you've downloaded the installer (to date about 1.4Gb), you will need to format a micro SD card of at least 8Gb. You can use a variety of ways to do this depending on your desktop platform. If you use Mac OS X, you can format the drive with Disk Utility. Or you can use the SD Formatter 4.0 utility available for Windows or Mac OS X (`http://sdcard.org/downloads/formatter_4/`). Simply download the application and install it. Then insert your micro SD card in your card reader and launch the application. Once you verify you've selected the correct media, enter a name for the card (I used NOOBS) and click format. Figure 6-7 shows the SDFormatter application.

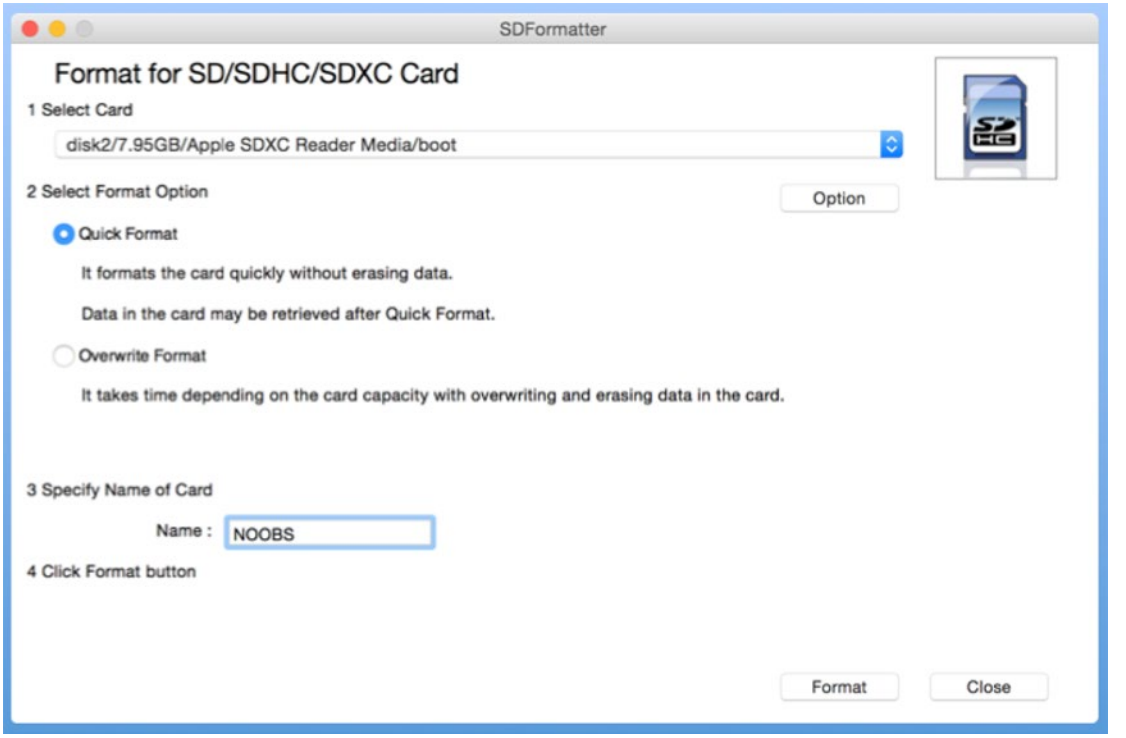


Figure 6-7. *SDFormatter 4.0*

BUT, I DON'T HAVE AN SD CARD READER!

You must have an SD card reader/writer connected to your computer. Some systems have SD card drives built in (Lenovo laptops, Apple laptops and desktops, and so on). If you do not have an SD card reader, you can find USB SD card readers anywhere electronics or photo equipment is sold. Most readers can accept various formats including micro SD or micro SD via a micro SD to SD adapter.

Once you've formatted the micro SD card, you now must copy the contents of the NOOBS image to the card. Right-click the file you downloaded and choose the option to unzip or unarchive the file. This will create a folder containing the NOOBS image. Copy all of those files (not the outside folder) to the SD card and eject it. You are now ready to boot into NOOBS and install your operating system. When this process has finished, safely remove the SD card and insert it into your Raspberry Pi.

You are now ready to hook up all of your peripherals. I like to keep things simple and connect only a monitor, keyboard, and (for NOOBS) a mouse. If you want to download an operating system other than Raspbian, you will also need to connect your Raspberry Pi to your network.

Once your Raspberry Pi powers on, you will see a scrolling display of various messages. This is normal and may scroll for some time before NOOBS starts. When NOOBS is loaded, you will see a screen similar to Figure 6-8.

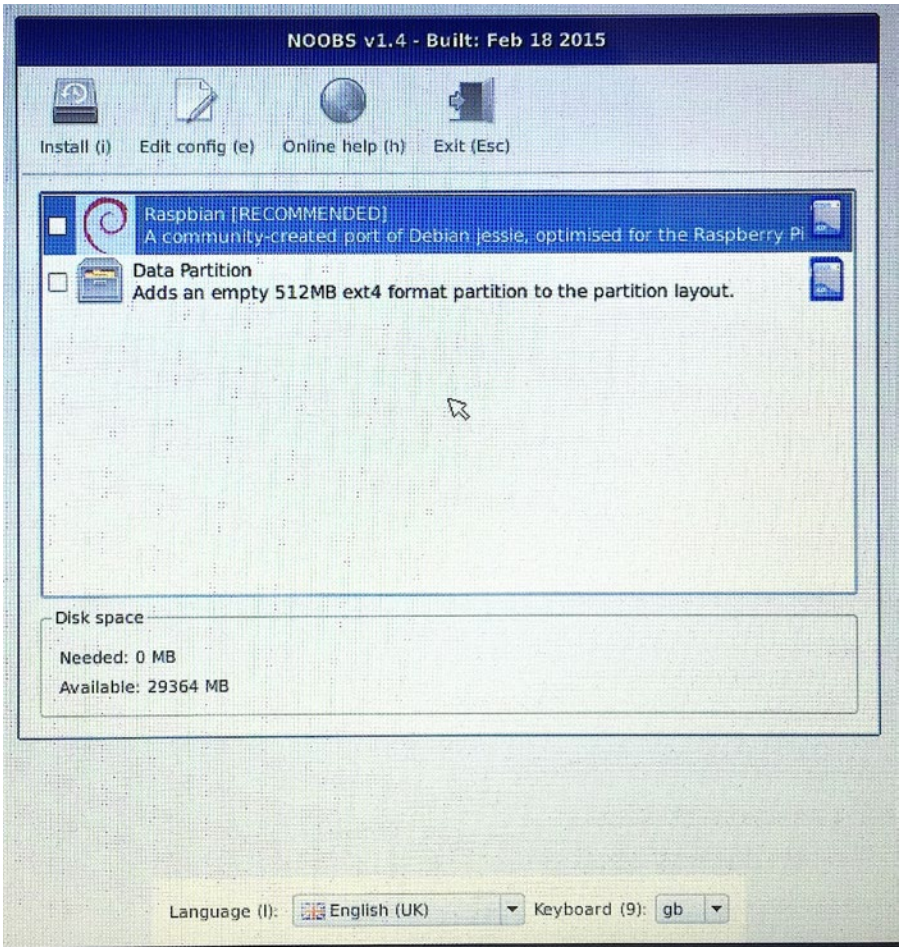


Figure 6-8. *NOOBS startup screen*

Notice you will see the Raspbian image in the list of operating systems. To install it, just tick the checkbox beside the thumbnail then click Install. However, note the two boxes at the bottom. This sets the language and keyboard for use in NOOBS. It does not affect the setup of Raspbian.

Once you initiate the install, you will see a series of dialogs as Raspbian begins its installation. This could take a while. The good news is the dialogs provide a lot of useful information to help you get started. You will learn about how to log in to Raspbian, tips for configuring and customizing, and suggestions for how to get the most out of your experience.

When installation finishes, click OK on the completed dialog and then wait for the Raspberry Pi to reboot into Raspbian. On the first boot, you may see the Raspberry Pi Configuration dialog. The configuration dialog is used to set the time and date for your region, enable hardware like a camera board, create users, change the password, and more. Figure 6-9 shows the configuration dialog.

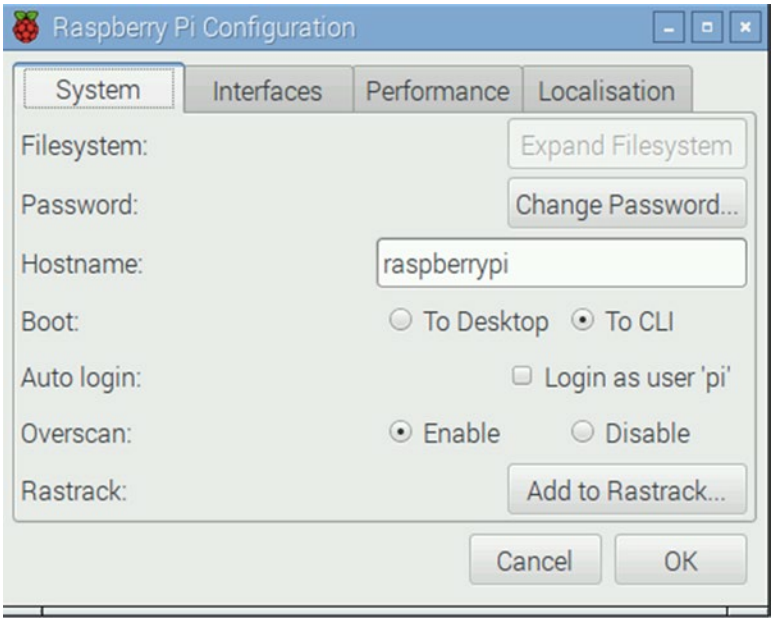


Figure 6-9. *Raspbian configuration dialog*

■ **Tip** You can also run the console-based configuration utility by opening a terminal and running the command `raspi-config`.

You will see four tabs that you can use to change settings for the system. I explain each briefly in the following list along with recommended settings for each. Once you have made your changes, click OK to close the dialog. Depending on which settings you choose, you may be asked to reboot.

- *System*: Board controls for the system. Use this panel to change the root password (highly recommended), hostname (optional), type of boot (use command-line interface [CLI] if you want to set up the Raspberry Pi to boot headless, and automatic login (not recommended).
- *Interfaces*: Used to enable system and hardware services such as the camera, SSH (recommended), and hardware interfaces for the GPIO header.
- *Performance*: Used to make changes to how the processor performs. You can choose to overclock (run the CPU faster), but I do not recommend this setting for a Raspberry Pi that will host a database or web server (or both).
- *Localisation*: Used to set the default language, keyboard, and date and time. If you change nothing else, be sure to set these to your local settings.

■ **Tip** The default login for Raspbian Jessie uses the username `pi` and password `raspberry`. I recommend changing this in the Raspberry Pi Configuration dialog.

To shut down or reboot Raspbian, click the menu, and then choose Shutdown. You will see a prompt for rebooting, shutting down, or returning to the command line. If you are at the command line, use the command `shutdown -h now` to shut down the system.

Installing Boot Image on a Micro SD Card

The process of installing a boot image involves choosing an image, downloading it, and then copying it to your micro SD card. The following sections detail the steps involved. This is a manual process that is a bit more complicated than the NOOBS option but not overly so.

Once you select an image and download it, you first unzip the file and then copy it to your SD card. There are a variety of ways to do this. The following sections describe some simplified methods for a variety of platforms.

Windows

To create the SD card image on Windows, you can use the Win32 Disk Imager software from Launchpad (`https://launchpad.net/win32-image-writer`). Download this file, and install it on your system. Unzip the image if you haven't already, and then insert your SD card into your SD card reader/writer. Launch the Win32 Disk Imager application, select the image in the top box, and then click WRITE to copy the image to the SD.

■ **Caution** The copy process overwrites anything already on the SD card, so be sure to copy those photos to your hard drive first!

Mac OS X

To create the SD card image on the Mac, download the image and unzip it. Insert your SD card into your SD card reader/writer. Be sure the card is formatted with FAT32. Next, open the System report (hint: use the Apple menu and then select About this Mac).

Click the card reader if you have a built-in card reader, or navigate through the USB menu and find the SD card. Take note of the disk number. For example, it could be disk4.

Next, open Disk Utility and unmount the SD card. You need to do this to allow Disk Utility to mount and connect to the card. Now things get a bit messy. Open a terminal, and run the following command, substituting the disk number for *n* and the path and name of the image file for `<image_file>`:

```
sudo dd if=<image_file> of=/dev/diskn bs=1m
```

At this point, you should see the disk-drive indicator flash (if there is one), and you need to be patient. This step can run for some time with no user feedback. You will know it is complete when the command prompt is displayed again.

Linux

To create the SD card image using Linux, you need to know the device name for the SD card reader. Execute the following command to see the devices currently mounted:

```
df -h
```

Next, insert the SD card or connect a card reader, and wait for the system to recognize it. Run the command again:

```
df -h
```

Take a moment to examine the list and compare it to the first execution. The "extra" device is your SD card reader. Take note of the device name (for example, `/dev/sdc1`). The number is the partition number. So, `/dev/sdc1` is partition 1, and the device is `/dev/sdc`. Next, unmount the device (I will use the previous example).

```
umount /dev/sdc1
```

Use the following command to write the image, substituting the device name for `<device>` and path and name of the image file for `<image_file>` (for example, `/dev/sdc` and `my_image.img`):

```
sudo dd bs=4M if=<image_file> of=<device>.
```

At this point, you should see the disk-drive indicator flash (if there is one), and you may need to be patient. This step can run for some time with no user feedback. You will know it is complete when the command prompt is displayed again.

Booting Up

To boot your Raspberry Pi, insert the SD card with the new image and plug in your peripherals. Wait to plug in the USB power last. Because the Raspberry Pi has no On/Off switch, it will start as soon as power is supplied. The system bootstraps and then starts loading the OS. You see a long list of statements that communicate the status of each subsystem as it is loaded. You don't have to try to read or even understand all the rows presented,[10] but you should pay attention to any errors or warnings. When the boot sequence is complete, you see a command prompt, as shown in Figure 6-10.

[10]They go by so fast; it is unlikely you can read them anyway. Basically, they're noise unless there is an error, and those usually appear in the last few lines displayed.

Figure 6-10. *Example boot sequence*[11]

You may be prompted to enter a username and password. The default user is simply *pi*, and the password is *raspberry* (no quotes, all lowercase). Enter that at the prompts, and the Raspberry Pi presents you with the configuration menu shown in Figure 6-11. If you do not see this, you can launch it by typing the command `raspi-config`.

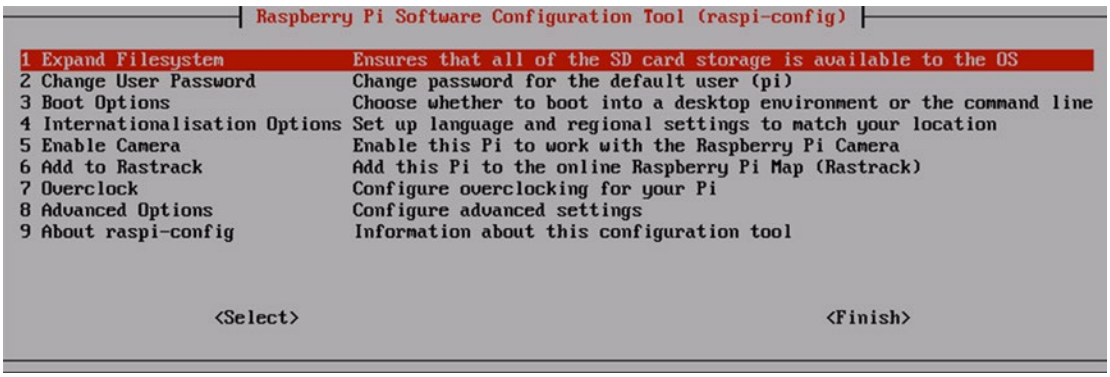


Figure 6-11. *Raspberry Pi configuration menu*

[11]Raspberry Pi images were generated with `fbgrab`. You can install it with `sudo apt-get install fbgrab`.

The configuration menu displays a list of common initial options you may want to set when using the Raspberry Pi. It is loaded on the first boot for convenience. You can navigate among the options using the up and down arrow keys and select the action buttons using the Tab key. The menu items are described briefly here:

- *Expand Filesystem*: Use the full space on the SD card.
- *Change User Password*: Change the password for the Pi user. Use this if your Raspberry Pi will be connected to a network and especially if it is accessible from the Internet.
- *Boot Options*: Enable/disable boot to the GUI windowing system.
- *Overscan*: Change how the video signal is sent to the monitor/TV. Use this if your output image from the Raspberry Pi does not fill the available display area.
- *Internationalisation Options*: Change keyboard mapping (country/language specific), time zone, and language, which sets country/language-specific display modes for how time, currency, dates, and so on, are displayed.
- *Enable Camera*: Turn on/enable the camera module.
- *Add to Rastrack*: Add your Raspberry Pi to the Pi Map, a global indicator of the Raspberry Pi's in the world.
- *Overclock*: Change the CPU timing settings (also called *speed*) for the system. Experts only. This is not needed for normal Raspberry Pi use.
- *Advanced Options*: Enable/disable various system services such as SSH.
- *About* `raspi-config`: Get information about how to use this tool.

■ **Note** Future releases of the configuration menu will include additional options. Once you have connected your Raspberry Pi to the Internet and executed the Update option, it is a good idea to check the menu for new options.

The first time you boot your system, you should use a few of these options. At a minimum, you should set the root file system to use the entire SD card space, change the keyboard setup, set your locale and time zone, and, if you want to be able to remotely log in, enable the SSH server.

When you first initialize an image on an SD card, the process does not use the entire space available. The Expand Filesystem option does this for you. In some cases, the system will be rebooted when the operation is complete, so make sure you don't have other things running before executing this option.

Setting the keyboard, locale, and time zone enables you to use the Raspberry Pi in a manner you are used to with a PC. In particular, your keyboard will have the special symbols where you expect them; dates, time, and similar values will be displayed correctly; and your clock will be set the correct local time. These operations may not require a reboot. You should set these prior to using the Raspberry Pi in earnest.

On future boots, the system will start, and, once you are logged in, it will be in terminal mode (unless you selected the option to start in the windowing environment). From here, you can explore the system using command-line utilities or start the graphical user interface with `startx`. Take some time and explore the system before proceeding. If you want to restart the configuration session, use the command `sudo raspi-config`.

Once your Raspberry Pi is running and you have spent time exploring and learning the basics for system administration, you are ready to start experimenting with hardware.

SD CARD CORRUPTION

Imagine this scenario. You're working away on creating files, downloading documents, and so on. Your productivity is high, and you're enjoying your new low-cost, super-cool Raspberry Pi. Now imagine the power cable accidentally gets kicked out of the wall, and your Raspberry Pi loses power. No big deal, yes? Well, most of the time.

The SD card is not as robust as your hard drive. You may already know that it is unwise to power off a Linux system abruptly, because doing so can cause file corruption. Well, on the Raspberry Pi it can cause a complete loss of your disk image. Symptoms range from minor read errors to an inability to boot or load the image on bootstrap. This can happen—and there have been reports from others that it has happened more than once.

That is not to say all SD cards are bad or that the Raspberry Pi has issues. The corruption on accidental power-off is a side effect of the type of media. Some have reported that certain SD cards are more prone to this than others. The best thing you can do to protect yourself is to use an SD card that is known to work with Raspberry Pi and be sure to power the system down with the `sudo shutdown -h now` command—and never, ever power off the system in any other manner.

You can also make a backup of your SD card. See `http://elinux.org/RPi_Beginners#Backup_your_SD_card` for more details.

Now that you know how to get your Raspberry Pi set up and running, let's now discover how to turn it into a database server for your IOT solution.

MySQL Installation and Setup

It is time to get your hands dirty and work some magic on your unsuspecting Raspberry Pi! Let's begin by adding a USB hard drive to it. Depending on the size of your data, you may want to seriously consider doing this.

That is, if your data will be small (never more than a few megabytes), you may be fine using MySQL from your boot image SD card. However, if you want to ensure that you do not run out of space and keep your data separate from your boot image (always a good idea), you should mount a USB drive that automatically connects on boot. This section explains how to do this in detail.

■ **Tip** Be sure you use a good-quality powered USB hub to host your external drive. This is especially important if you are using a traditional spindle drive because it consumes a lot more power. Connecting your external drive directly to the Raspberry Pi may rob it of power and cause untold frustration. Symptoms include random reboot (always a pleasant surprise), failed commands, data loss, and so on. Always be sure you have plenty of power for your peripherals as well as your Raspberry Pi.

The choice of what disk to use is up to you. You can use a USB flash drive, which should work fine if it has plenty of space and is of sufficient speed (most newer models are fast enough). You can also use a solid-state drive (SSD) if you have an extra one or want to keep power usage and heat to a minimum. On the other hand, you may have an extra hard drive lying around that can be pressed into service. This section's example uses a surplus 250GB laptop hard drive mounted in a typical USB hard drive enclosure.

■ **Tip** Using an external hard drive—either an SSD or traditional spindle drive—is much faster than accessing data on a flash drive. It is also typically cheaper per unit (gigabyte) or, as I mentioned, can be easily obtained from surplus.

Partitioning and Formatting the Drive

Before you can use a new or an existing drive with a file system incompatible with the Raspberry Pi, you must partition and format the drive. I find it easier to do this on my desktop computer and suggest you do the same. Thus, the following assumes the external drive has a single FAT (or FAT32) partition. That isn't so important because we will delete it and create a new partition with the ext4 file system for optimal performance.

Begin by connecting the drive to the Raspberry Pi. Then determine what drives are attached by using the `fdisk -l` command to see the available disks connected. You should see your hard drive listed as `/dev/sda` if you have a standard Raspbian image. If you use a different image or your device is labeled differently, use the address from your system in the following steps.

Once you identify the disk, launch `fdisk` again with the device as an option, as shown in Listing 6-1.

Listing 6-1. Partitioning a Hard Disk on Raspberry Pi

```
pi@raspberrypi ~ $ sudo fdisk /dev/sda

Welcome to fdisk (util-linux 2.25.2).
Changes will remain in memory only, until you decide to write them.
Be careful before using the write command.


Command (m for help): m

Help:

  Generic
   d   delete a partition
   l   list known partition types
   n   add a new partition
   p   print the partition table
   t   change a partition type
   v   verify the partition table

  Misc
   m   print this menu
   x   extra functionality (experts only)

  Save & Exit
   w   write table to disk and exit
   q   quit without saving changes
```

```
  Create a new label
   g   create a new empty GPT partition table
   G   create a new empty SGI (IRIX) partition table
   o   create a new empty DOS partition table
   s   create a new empty Sun partition table

Command (m for help): p
Disk /dev/sda: 111.8 GiB, 120034123776 bytes, 234441648 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: gpt
Disk identifier: 79OE7C68-FO89-45C7-A9E9-D7C2CA56BB31

Command (m for help): n
Partition number (1-128, default 1): 1
First sector (34-234441614, default 2048):
Last sector, +sectors or +size{K,M,G,T,P} (2048-234441614, default 234441614):

Created a new partition 1 of type 'Linux filesystem' and of size 111.8 GiB.

Command (m for help): w
The partition table has been altered.
Calling ioctl() to re-read partition table.
Syncing disks.
```

There are a number of things going on here. I've highlighted the steps in bold. Notice first I show the help menu for the utility with the `m` command per the prompt. Next, I use the `p` command to print the partition table verifying that there is no partition there. If you had partitions defined and wanted to delete them, you'd use the `d` command to do so.

■ **Caution** If you have a partition on your drive that has data you want to keep, abort now and copy the data to another drive first. The following steps erase all data on the drive!

You then create a new partition using the command `n` and accept the defaults to use all the free space. To check your work, you can use the `p` command to print the device partition table and metadata. It shows (and confirms) the new partition.

If you are worried that you may have made a mistake, do not panic! The great thing about `fdisk` is that it doesn't write or change the disk until you tell it to with the `w` or `write` command. In the example, you issue the `w` command to write the partition table. To see a full list of the commands available, you can use the `h` command or run `man fdisk`.

■ **Tip** For all Linux commands, you can view the manual file by using the command `man <application>`.

The next step is to format the drive with the ext4 file system. This is easy and requires only one command: `mkfs` (make file system). You pass it the device name. If you recall, this is `/dev/sda1`. Even though you created a new partition, it is still the first partition because there is only one on the drive. If you are attempting to use a different partition, be sure to use the correct number! The command may take a few minutes to run, depending on the size of your drive. The following example shows the command in action:

```
pi@raspberrypi ~ $ sudo mkfs.ext4 /dev/sda1
mke2fs 1.42.12 (29-Aug-2014)
Creating filesystem with 29304945 4k blocks and 7331840 inodes
Filesystem UUID: 0285ba01-3880-4ee5-8a19-7f47404f1500
Superblock backups stored on blocks:
    32768, 98304, 163840, 229376, 294912, 819200, 884736, 1605632, 2654208,
    4096000, 7962624, 11239424, 20480000, 23887872

Allocating group tables: done
Writing inode tables: done
Creating journal (32768 blocks): done
Writing superblocks and filesystem accounting information: done
```

Now you have a new partition, and it has been properly formatted. The next step is associating the drive with a mount point on the boot image and then connecting that drive on boot so you don't have to do anything to use the drive each time you start your Raspberry Pi.

Setting Up Automatic Drive Mounting

External drives in Linux are connected (mounted) with `mount` and disconnected (unmounted) with `umount`. Unlike with some operating systems, it is generally a bad idea to unplug your USB drive without unmounting it first. Likewise, you must mount the drive before you can use it. This section shows the steps needed to mount the drive and to make the drive mount automatically on each boot.

I begin with a discussion of the preliminary steps to get the drive mounted and ready for automatic mounting. These include creating a folder under the `/media` folder to mount the drive (called a *mount point*), changing permissions to the folder to allow access, and executing some optional steps to tune the drive.

```
pi@raspberrypi ~ $ sudo mkdir -p /media/HDD
pi@raspberrypi ~ $ sudo chmod 755 /media/HDD
pi@raspberrypi ~ $ sudo tune2fs -m 0 /dev/sda1
tune2fs 1.42.12 (29-Aug-2014)
Setting reserved blocks percentage to 0% (0 blocks)
pi@raspberrypi ~ $ sudo tune2fs -L MYSQL /dev/sda1
tune2fs 1.42.12 (29-Aug-2014)
pi@raspberrypi ~ $ sudo mount /dev/sda1 /media/HDD
```

These commands are easy to discern and are basic file and folder commands. However, the tuning steps using `tune2fs` (tune file system) are used to first reset the number of blocks used for privileged access (which saves a bit of space) and then label the drive as MYSQL. Again, these are optional, and you may skip them if you like.

■ **Tip** You can unmount the drive with `sudo umount /dev/sda1`.

At this point, the drive is accessible and ready to be used. You can change to the `/media/HDD` folder and create files or do whatever you'd like. Now let's complete the task of setting up the drive for automatic mounting.

The best way to do this is to refer to the drive by its universally unique identifier (UUID). This is assigned to this drive and only this drive. You can tell the operating system to mount the drive with a specific UUID to a specific mount point (`/media/HDD`).

Remember the `/dev/sda` device name from earlier? If you plugged your drive in to another hub port—or, better still, if there are other drives connected to your device and you unmount and then mount them—the device name may not be the same the next time you boot! The UUID helps you determine which drive is your data drive, frees you from having to keep the drive plugged in to a specific port, and allows you to use other drives without fear of breaking your MySQL installation if the drive is given a different device name.

To get the UUID, use the `blkid` (block ID) application.

```
pi@raspberrypi ~ $ sudo blkid
/dev/mmcblk0: PTUUID="000575b3" PTTYPE="dos"
/dev/mmcblk0p1: LABEL="RECOVERY" UUID="0761-F2EA" TYPE="vfat" PARTUUID="000575b3-01"
/dev/mmcblk0p3: LABEL="SETTINGS" UUID="13062706-1a48-47bc-9f3a-0ded961267e4" TYPE="ext4"
PARTUUID="000575b3-03"
/dev/mmcblk0p5: SEC_TYPE="msdos" LABEL="boot" UUID="07D7-3A9D" TYPE="vfat"
PARTUUID="000575b3-05"
/dev/mmcblk0p6: LABEL="root" UUID="c9d8e201-90e5-4d6b-9c8f-92d658fec13c" TYPE="ext4"
PARTUUID="000575b3-06"
/dev/sda1: LABEL="MYSQL" UUID="0285ba01-3880-4ee5-8a19-7f47404f1500" TYPE="ext4"
PARTUUID="ab7357a8-536a-4015-a36d-f80280c2efd1"
```

Notice the line in bold. Wow! That's a big string. A UUID is a 128-byte (character) string. Copy it for the next step.

To set up automatic drive mapping, you use a feature called *static information* about the file system (`fstab`). This consists of a file located in the `/etc` folder on your system. You can edit the file however you like. If you are from the old school of Linux or Unix, you may choose to use vi.[12] The resulting file is as follows:

```
pi@raspberrypi ~ $ sudo nano /etc/fstab
proc            /proc           proc    defaults          0       0
/dev/mmcblk0p5  /boot           vfat    defaults          0       2
/dev/mmcblk0p6  /               ext4    defaults,noatime  0       1
UUID=0285ba01-3880-4ee5-8a19-7f47404f1500 /media/HDD ext4 defaults,noatime 0 0
# a swapfile is not a swap partition, no line here
#   use  dphys-swapfile swap[on|off]  for that
```

The line you add is shown in bold. Here you simply add the UUID, mount point, file system, and options. That's it! You can reboot your Raspberry Pi using the following command and watch the screen as the messages scroll. Eventually, you see that the drive is mounted. If there is ever an error, you can see it in the bootup sequence.

```
$ sudo shutdown -r now
```

[12]What does *vi* mean? If you've ever had the pleasure of trying to learn it for the first time, you may think it means "virtually impossible," because the commands are terse (by design) and difficult to remember. But seriously, *vi* is short for *vim* or *Vi Improved* text editor. The name suggests that the original editor may very well have been *completely* impossible to use!

Now you are ready to build a MySQL database server! The following section details the steps needed to do this using your Raspberry Pi.

Installing MySQL Server

Turning a Raspberry Pi into a MySQL database server is easy. This section shows you how to install MySQL and then how to move its default data directory from your boot image to the new external drive you connected in the previous section.

The steps involved include updating your aptitude base (the package manager) and then installing MySQL. Although the process is rather lengthy, I felt it best to show you the entire thing in case your base image is different or you encounter errors.

Installing MySQL

To install MySQL or any software not already in your base image, you must be connected to the Internet. If you have not already done so, connect your Raspberry Pi to the Internet using the Ethernet port or a wireless networking device.

As you may recall, you are using the Raspbian Jessie distribution, which is Debian-based. If you use some other distribution, it may have a different package manager, and the commands in this section may not work. In that case, you should be able to find similar commands for your distribution.

Let's begin with updating the package manager package headers. This is always a good idea, especially if you are using a distribution that was released more than a few months ago. The command `apt-get update` tells the system to download the latest headers from known host distributions. This ensures that you get the latest version of whatever software you are installing.

After that, installing the software is as simple as telling aptitude to install it. The trick is knowing the correct name. In this case, you're looking for `mysql-server`. Listing 6-5 shows the steps for updating aptitude and installing MySQL. (I have omitted some lines for brevity.) In addition to entering the commands, you are asked to reply to the prompt asking if it is OK to download MySQL and its prerequisites and to enter a password for the root user for MySQL.

■ **Note** When you see the password *secret* in the examples, it is used as a placeholder for whatever password you have chosen—it is not explicitly the word *secret*.

Let's begin by updating the package manager with the `sudo apt-get update` command, as shown here:

```
pi@raspberrypi ~ $ sudo apt-get update
Get:1 http://archive.raspberrypi.org jessie InRelease [13.2 kB]
Get:2 http://mirrordirector.raspbian.org jessie InRelease [15.0 kB]
Get:3 http://archive.raspberrypi.org jessie/main Sources [22.4 kB]
Get:4 http://mirrordirector.raspbian.org jessie/main armhf Packages [8,961 kB]
Get:5 http://archive.raspberrypi.org jessie/ui Sources [5,197 B]
Get:6 http://archive.raspberrypi.org jessie/main armhf Packages [60.2 kB]
Get:7 http://archive.raspberrypi.org jessie/ui armhf Packages [7,639 B]
Get:8 http://mirrordirector.raspbian.org jessie/contrib armhf Packages [37.4 kB]
Get:9 http://mirrordirector.raspbian.org jessie/non-free armhf Packages [70.2 kB]
...
Fetched 9,194 kB in 1min 11s (129 kB/s)
Reading package lists... Done
```

Next, let's install MySQL with the `sudo apt-get install mysql-server` command as shown next. Once you begin the MySQL installation, you may be prompted for setting the root user password for MySQL. Be sure to choose a password you will remember.

```
pi@raspberrypi ~ $ sudo apt-get install mysql-server
Reading package lists... Done
Building dependency tree
Reading state information... Done
```

The following extra packages will be installed:

```
libaio1 libdbd-mysql-perl libdbi-perl libhtml-template-perl libmysqlclient18 libterm-
readkey-perl mysql-client-5.5 mysql-common
mysql-server-5.5 mysql-server-core-5.5
```

Suggested packages:

```
libclone-perl libmldbm-perl libnet-daemon-perl libsql-statement-perl libipc-sharedcache-perl
mailx tinyca
```

The following NEW packages will be installed:

```
libaio1 libdbd-mysql-perl libdbi-perl libhtml-template-perl libmysqlclient18 libterm-
readkey-perl mysql-client-5.5 mysql-common
mysql-server mysql-server-5.5 mysql-server-core-5.5
0 upgraded, 11 newly installed, 0 to remove and 3 not upgraded.
Need to get 8,121 kB of archives.
After this operation, 88.8 MB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://mirrordirector.raspbian.org/raspbian/ jessie/main libaio1 armhf 0.3.110-1 [9,228 B]
Get:2 http://mirrordirector.raspbian.org/raspbian/ jessie/main mysql-common all
      5.5.44-0+deb8u1 [74.3 kB]
Get:3 http://mirrordirector.raspbian.org/raspbian/ jessie/main libmysqlclient18 armhf
      5.5.44-0+deb8u1 [616 kB]
...
Setting up libaio1:armhf (0.3.110-1) ...
Setting up libmysqlclient18:armhf (5.5.44-0+deb8u1) ...
Setting up libdbi-perl (1.631-3+b1) ...
Setting up libdbd-mysql-perl (4.028-2+b1) ...
Setting up libterm-readkey-perl (2.32-1+b2) ...
Setting up mysql-client-5.5 (5.5.44-0+deb8u1) ...
Setting up mysql-server-core-5.5 (5.5.44-0+deb8u1) ...
Setting up mysql-server-5.5 (5.5.44-0+deb8u1) ...
151001 12:58:57 [Warning] Using unique option prefix key_buffer instead of key_buffer_size
is deprecated and will be removed in a future release. Please use the full name instead.
151001 12:58:57 [Note] /usr/sbin/mysqld (mysqld 5.5.44-0+deb8u1) starting as process 18455 ...
Setting up libhtml-template-perl (2.95-1) ...
Setting up mysql-server (5.5.44-0+deb8u1) ...
Processing triggers for libc-bin (2.19-18+deb8u1) ...
Processing triggers for systemd (215-17+deb8u2) ...
```

WHAT IF IT DOESN'T WORK?

Although highly unlikely, if it all goes completely wonky,[13] you can remove the MySQL installation bundle with the following commands. They uninstall every package and remove any files created by the installation.

```
sudo apt-get autoremove mysql-server mysql-server-5.5
sudo apt-get purge mysql-server mysql-server-5.5
```

Once you've done this, you can try the install steps again and correct your mistake.

Now that MySQL is installed, let's use the MySQL console and try to connect to the server. The command is `mysql -uroot -p<password>`, where `<password>` is the password you supplied when you installed MySQL. Listing 6-2 shows a successful connection to the new MySQL server. I executed some commands to test things and to gather information for the next step. Notice that the MySQL console displays the version of the MySQL server as well as a short name for the platform. In this case, I was connected to a MySQL 5.5.28-1 server on a Debian platform.

Listing 6-2. Connecting to MySQL

```
pi@raspberrypi ~ $ mysql -uroot -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 43
Server version: 5.5.44-0+deb8u1 (Raspbian)

Copyright (c) 2000, 2015, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> SHOW DATABASES;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
+--------------------+
3 rows in set (0.00 sec)
```

[13] A highly technical term for when computers don't do what you think they should.

```
mysql> SHOW VARIABLES LIKE '%dir%';
+----------------------------------------+----------------------------+
| Variable_name                          | Value                      |
+----------------------------------------+----------------------------+
| basedir                                | /usr                       |
| binlog_direct_non_transactional_updates | OFF                        |
| character_sets_dir                     | /usr/share/mysql/charsets/ |
| datadir                                | /var/lib/mysql/            |
| innodb_data_home_dir                   |                            |
| innodb_log_group_home_dir              | ./                         |
| innodb_max_dirty_pages_pct             | 75                         |
| lc_messages_dir                        | /usr/share/mysql/          |
| plugin_dir                             | /usr/lib/mysql/plugin/     |
| slave_load_tmpdir                      | /tmp                       |
| tmpdir                                 | /tmp                       |
+----------------------------------------+----------------------------+
11 rows in set (0.00 sec)

mysql>
```

In the example, I issued the `SHOW DATABASES` command to see the list of databases and the `SHOW VARIABLES` command to show all variables containing the name `dir`. Notice the `datadir` output from the last command: this is the location of your data.

In the next section, you tell MySQL to use the external drive instead for storing your databases and data.

Moving the Data Directory to the External Drive

Recall that you want to use MySQL to store your sensor data. As such, the sensor data may grow in volume and over time may consume a lot of space. Rather than risk filling up your boot image SD, which is normally only a few gigabytes, you can use an external drive to save the data. This section shows you how to tell MySQL to change its default location for saving data.

The steps involved require stopping the MySQL server, changing its configuration, and then restarting the server. Finally, you test the change to ensure that all new data is being saved in the new location. Begin by stopping the MySQL server.

```
$ sudo /etc/init.d/mysql stop
```

You must create a folder for the new data directory.

```
$ sudo mkdir /media/HDD/mysql
```

Now you copy the existing data directory and its contents to the new folder. Notice that you copy only the data and not the entire MySQL installation, which is unnecessary.

```
$ sudo cp -R /var/lib/mysql/*  /media/HDD/mysql
$ chown -R mysql mysql /media/HDD/mysql/
```

Next you edit the configuration file for MySQL. In this case, you change the `datadir` line to read `datadir = /media/HDD/mysql`. It is also a good idea to comment out the `bind-address` line to permit access to MySQL from other systems on the network.

```
$ sudo vi /etc/mysql/my.cnf
```

There is one last step. You must change the owner and group to the MySQL user who was created on installation. Here is the correct command:

```
$ sudo chown -R mysql:mysql /media/HDD/mysql
```

Now you restart MySQL.

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

You can determine whether the changes worked by connecting to MySQL, creating a new database, and then checking to see whether the new folder was created on the external drive, as shown in Listing 6-3.

Listing 6-3. Testing the New Data Directory

```
pi@raspberrypi ~ $ mysql -uroot -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 43
Server version: 5.5.44-0+deb8u1 (Raspbian)

Copyright (c) 2000, 2015, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> CREATE DATABASE TESTME;
Query OK, 1 row affected (0.00 sec)

mysql> SHOW DATABASES;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| test               |
| testme             |
+--------------------+
5 rows in set (0.00 sec)

mysql> quit
```

You can check to see that the database is created by displaying the file structure using the following command:

```
$ sudo ls -lsa /media/HDD/mysql
```

What you should see in the `mysql` folder is a separate folder for each database. Indeed, you should see the folder for the new database created represented as the folder `testme`. Well, there you have it—a new MySQL database server running on a Raspberry Pi!

WHAT ABOUT OVERHEATING?

Concerns about overheating a Raspberry Pi are mainly for those who attempt overclocking and other risky modifications; you should worry if your Raspberry Pi is run continuously. Typically you run a database server 24/7, shutting it down only for maintenance.

If you are concerned about overheating, you can add heat sinks to your Raspberry Pi's major components for a reasonable cost (about $15). However, I have not seen any issues with running a Raspberry Pi indefinitely if it is housed in an enclosure that permits heat dissipation and it is placed in a climate-controlled environment. A definitive answer to this question has been provided by one of the founders himself (see www.youtube.com/watch?v=Sz8NMp4MgGo).

Now that we have our Raspberry Pi configured and MySQL installed and working, let's now see how we can connect our devices to the database server for saving our data. But first, let me discuss some alternatives for the Raspberry Pi.

Other Platforms

While I focused on the Raspberry Pi for demonstrating how to build a low-cost MySQL database server, the Raspberry Pi isn't the only choice. Indeed, you can use any number of low-cost computers and embedded platforms for hosting a MySQL server. In this section, I present three alternatives: the BeagleBone Black, pcDuino, and Intel Galileo.

While the process is similar on all of these, there are some small differences and other things to consider. Thus, I present each in the following sections in a condensed overview. Having read the Raspberry Pi tutorial previously, you should be able to accomplish the following with ease.

BeagleBone Black

Recall from Chapter 3, the BeagleBone Black is a lower-cost version of the original BeagleBone. It comes with an onboard bootable Linux operating system preconfigured. Like the Raspberry Pi, it hosts a number of ports including USB ports for connecting memory devices.

Be sure to connect your BeagleBone Black to your network before powering on. I used the onboard Ethernet port and found it more than adequate for accessing the board remotely. The only issue you may have is discovering which IP address your board is using. I recommend using a port scanner. There are many such applications available for most platforms. Use that IP address and remote into your BeagleBone Black with `ssh root@<IP address>`.

■ **Note** The default root password on the BeagleBone Black is blank.

Installing MySQL

The MySQL installation on BeagleBone Black is similar to the Raspberry Pi example earlier except the commands are a little different but we execute them in the same order. The following shows the commands you execute. We must first update the local packages and package headers. Use the following command to do this:

```
$ opkg update
```

Next, we install MySQL with the following command:

```
$ opkg install mysql5
```

This process will take a while beginning with downloading a number of packages and supporting libraries. Once the installation is complete, issue the following command to launch MySQL:

```
$ /etc/init.d/mysqld start
```

Depending on the date of your BeagleBone Black's preinstalled operating system, you may see an error similar to the following:

```
/etc/init.d/mysqld: line 3: /etc/default/rcS: No such file or directory
BeagleBone - mysqld error.png
```

The problem is an error in the script that starts MySQL. We edit the file with the following command and comment out the third line in the file (more specifically, the line that reads `/etc/default/rcS`). Just put a # in the first column to comment it out.

```
$ vi /etc/init.d/mysqld
```

Save the file and then restart MySQL as follows:

```
$ /etc/init.d/mysqld start
```

■ **Note** There is no `sudo` on the BeagleBone Black default installation. To shut down, simply use the `shutdown -h now` command.

Configuring the Hard Drive

Configuring the hard drive on the BeagleBone Black uses the same commands as those on the Raspberry Pi. That is, you use `fdisk` to create a partition and `mkfs` to create the file system.

pcDuino

The pcDuino is a unique board. The newest versions support the A20 or later processors (multicore and a bit faster than other boards). The board also has many connectors including an onboard SATA port for supporting a SATA hard drive. Perhaps the most interesting and indeed the reason for the name are the board supports Arduino-compatible shields. As you saw in Chapter 3, this allows us to develop our Arduino solutions on a single device.

However, the really nice part is the pcDuino's onboard bootable operating system is a version of Ubuntu 12. This means it will behave similarly to a full installation of Ubuntu. Indeed, as a regular Ubuntu user (second choice to Mac OS X), I found myself right at home on the pcDuino, especially when I use my Lapdock connected to the pcDuino.

That is, with the pcDuino connected to a monitor, keyboard, and mouse, it operates very much the same way as a laptop. It's nearly as fast too! Plus, the onboard Wi-Fi capabilities of the pcDuino 3B make using the board convenient (no Ethernet cable strung across the room).

Since the pcDuino runs Ubuntu, installing MySQL on a pcDuino is the same process as installing MySQL on most any other Ubuntu machine. You can even use the same documentation and examples found elsewhere on the Internet. For pcDuino-specific documentation, see `http://linksprite.com/?page_id=874`.

■ **Tip** The default user for the pcDuino is `ubuntu` with password `ubuntu`.

Installing MySQL

The MySQL installation on the pcDuino requires the following commands. We begin with updating the packages and package headers as follows:

```
$ sudo apt-get update
```

Next, we install MySQL with the following command:

```
$ sudo apt-get install mysql-server
```

You will have to select a root user password twice during the install, but that's it! All you have left to do is configure MySQL to your needs. For example, you may want to edit the `my.cnf` file and set up your databases as described in the previous chapter.

Configuring the Hard Drive

Configuring the hard drive on the pcDuino is, once again, similar to the other platforms. Since the platform is a bit different, I will walk you through the process.

Begin by plugging the USB drive into the USB port. When the drive is recognized (the access LED may blink a few times and then go solid), execute `fdisk` much like we did on the Raspberry Pi to create a partition, `mkfs` to format, and then mount the drive. Finally, set up the drive using the same commands as we used on the Raspberry Pi to make the mount persistent and transfer the MySQL database `datadir` to the drive as shown previously.

Intel Galileo

The Intel Galileo, sister board to the Intel Einstein platform, is another board that includes headers that accept Arduino shields. Like the pcDuino and the newer Arduino boards that run a Linux operating system such as the Yun, the Intel Galileo provides access to the Arduino either from Linux or via a serial connection to your desktop computer. Thus, like the pcDuino, you can use it to write Arduino sketches as well as an embedded node in your IOT solution.

The Intel Galileo's onboard bootable Linux is minimal, but I recommend downloading the latest Intel SD-Card Linux Image (currently located at http://intel.com/support/galileo/sb/CS-035101.htm), uncompressing and installing it on a bootable micro SD drive (you can find instructions at https://software.intel.com/en-us/creating-bootable-micro-sd-card-for-intel-galileo-board), and then booting the Galileo from the micro SD card.

You can connect your Galileo directly to your network via an Ethernet cable and, like the BeagleBone Black example, use a port scanner to locate the IP address. Unfortunately, there isn't an HDMI display port on the Galileo, so using a monitor and keyboard isn't an option.

Also, the Galileo's Linux image is a bit on the lean side, and the packages built are not as complete as other platforms. In fact, we will have to use an alternative package repository to install MySQL.

Installing MySQL

The MySQL installation on the Intel Galileo is nearly the same as the BeagleBone Black, but as you shall see, it isn't as clean. One difference is there is no base MySQL package available for the default SD-Card Linux image. Fortunately, AlexT[14] has done the work for us. All you need to do is modify the package locations and perform the update. The following shows all the steps for completeness. I recommend starting with a clean boot of the SD-Card Linux image from Intel.

■ **Note** There is no root password for the SD-Card Linux image on the Galileo.

We begin by updating the package location file named SSS with the following command and data. Just paste this into the empty file.

```
$ vi /etc/opkg/base-feeds.conf
src/gz all     http://repo.opkg.net/galileo/repo/all
src/gz clanton http://repo.opkg.net/galileo/repo/clanton
src/gz i586    http://repo.opkg.net/galileo/repo/i586
```

We then update the local packages and package headers. Use the following command to do this:

```
$ opkg update
```

Next, we install MySQL with the following command:

```
$ opkg install mysql5
```

This process will take a while beginning with downloading a number of packages and supporting libraries. If you get an error, you may need to overwrite the uclibc files with the following command and then restart the installation:

```
$ opkg install --force-overwrite uclibc
```

[14] http://alextgalileo.altervista.org/package-repo-configuration-instructions.html

Once the installation is complete, you should be able to start the MySQL server as follows. If you see errors such as the following, you may need to do some extra work. I saw this on at least one Galileo board after an aborted installation. You may not need these steps, but I include them in case you get stuck by this issue. What happened was the installation failed before it created the special user `mysql`, which is common for most Linux installations. But again, it is easy to fix.

```
$ /etc/init.d/mysqld start
$ 010101 00:27:46 mysqld_safe Logging to '/var/log/mysqld.err'.
chown: unknown user mysql
010101 00:27:46 mysqld_safe Starting mysqld daemon with databases from /var/mysql
010101 00:27:47 mysqld_safe mysqld from pid file /var/lib/mysql/mysqld.pid ended
```

To fix this, simply create the `mysql` user as follows and start the server again. If you'd like to read more about this process, see the preconfiguration steps in the source code installation section in the online MySQL reference manual (http://dev.mysql.com/doc/refman/5.6/en/installing-source-distribution.html).

```
$ groupadd mysql
$ useradd -r -g mysql mysql
$ chown -R mysql /var/lib/mysql
$ chgrp -R mysql /var/lib/mysql
$ mysql_install_db --user=mysql
$ /etc/init.d/mysqld start
```

Configuring the Hard Drive

Configuring the hard drive on the Galileo is, once again, similar to the other platforms. Since the platform is a bit different, I will walk you through the process.

Begin by plugging the USB drive into the USB port. When the drive is recognized (the access LED may blink a few times and then go solid), execute `fdisk` much like we did on the Raspberry Pi to create a partition, `mkfs` to format, and then mount the drive. Finally, set up the drive using the same commands as we used on the Raspberry Pi to make the mount persistent and transfer the MySQL database `datadir` to the drive as shown previously.

DOES IT MATTER WHAT VERSION OF MYSQL I USE?

You may be wondering about the version of MySQL you should use. While the latest version of MySQL is 5.7, it is unlikely you will find this version in the package list or repositories for the Linux and similar operating systems for boards like those described here. Fortunately, most packages and repositories have either MySQL 5.1 or 5.5 available, which should be sufficient for most IOT solutions, and you are unlikely to need the latest features of MySQL.

However, if you do need the latest features for compatibility or conformity, you may need to download the source code directly from Oracle and build the installation locally. Or you could search for a precompile binary or even an installation package for your platform. While they're rare, I have encountered these packages from time to time. Don't be afraid to ask people on a support forum for your board for help. Chances are someone out there has already created what you need.

MySQL Clients: How to Connect and Save Data

You have already seen how to connect to the MySQL server with the MySQL client. That tool is an interactive tool where we can execute queries, but it isn't helpful for saving data from our sensors or data nodes. What we need is something called a *connector*. A connector is a programming module designed to permit our sketches (from an Arduino) or scripts or programs to send data to the database server. Connectors also allow us to query the database server to get data from the server.

I will cover two primary connectors you are likely to encounter when developing your own IOT solutions. I present each as a tutorial that you can use to follow along with your own hardware. I begin with a connector for use with the Arduino (Connector/Arduino) and then present a connector for use in writing Python scripts (Connector/Python).

DATABASE CONNECTORS FOR MYSQL

There are many database connectors for MySQL. Oracle supplies a number of database connectors for a variety of languages. The following are the current database connectors available for download from `http://dev.mysql.com/downloads/connector/`:

- *Connector/ODBC*: Standard ODBC compliant
- *Connector/Net*: Windows .Net platforms
- *Connector/J*: Java applications
- *Connector/Python*: Python applications
- *Connector/C++*: Standardized C++ applications
- *Connector/C* (`libmysql`): C applications
- *MySQL native driver for PHP* (`mysqlnd`): PHP 5.3 or newer connector
- *Connector/Arduino*: Arduino sketches

As you can see, there is a connector for just about any programming language you are likely to encounter—and now there is even one for the Arduino!

Introducing Connector/Arduino

With a new database connector made specifically for the Arduino, you can connect your Arduino project directly to a MySQL server without using an intermediate computer or a web-based service. Having direct access to a database server means you can store data acquired from your project in a database. You can also check values stored in tables on the server. The connector allows you to keep your IOT solution local to your facility—it can even be disconnected from the Internet or any other external network.

Saving your data in a database not only preserves the data for analysis at a later time but also means your project can feed data to more complex applications. Better still, if you have projects that use large data volumes for calculations or lookups, you can store the data on the server and retrieve only the data you need for the calculation or operation—all without taking up large blocks of memory on your Arduino. Clearly, this opens a whole new avenue of Arduino projects!

The database connector is named Connector/Arduino. It implements the MySQL client communication protocol (called a *database connector*) in a library built for the Arduino platform. Henceforth I refer to Connector/Arduino when discussing general concepts and features and refer to the actual source code as the Connector/Arduino library, the connector, or simply the library.

Sketches (programs) written to use the library permit you to encode SQL statements to insert data and run small queries to return data from the database (for example, using a lookup table).

You may be wondering how a microcontroller with limited memory and processing power can possibly support the code to insert data into a MySQL server. You can do this because the protocol for communicating with a MySQL server is not only well known and documented but also specifically designed to be lightweight. This is one of the small details that make MySQL attractive to embedded developers.

To communicate with MySQL, the Arduino must be connected to the MySQL server via a network. To do so, the Arduino must use an Ethernet or Wi-Fi shield and be connected to a network or subnet that can connect to the database server (you can even connect across the Internet). The library is compatible with most new Arduino Ethernet, Wi-Fi, and compatible clone shields that support the standard Ethernet library.

■ **Note** Compatibility isn't a hardware requirement so much as a software library limitation. That is, if you use a networking device that uses either the included Ethernet library or a library based on the standard `Ethernet.Client` class, your hardware should be compatible. Some of the newer, low-cost Ethernet modules may not be compatible.

There is a lot you can do with Connector/Arduino. What follows is a short primer on getting started with the connector. If you need more help or want a more in-depth look into the library as well as more examples, download the reference manual from https://github.com/ChuckBell/MySQL_Connector_Arduino/blob/master/extras/MySQL_Connector_Arduino_Reference_Manual.pdf.

WHAT ABOUT MEMORY?

Connector/Arduino is implemented as an Arduino library. Although the protocol is lightweight, the library does consume some memory. In fact, the library requires about 20KB of flash memory to load. Thus, it requires the ATmega328 or similar processor with 32KB of flash memory.

That may seem like there isn't a lot of space for programming your solution, but as it turns out, you really don't need that much for most sensors. If you do, you can always step up to a newer Arduino with more memory. For example, the latest Arduino, the Due, has 512KB of memory for program code. Based on that, a mere 20KB is an insignificant amount of overhead.

The library is open source, licensed as GPLv2, and owned by Oracle Corporation. Thus, any modifications to the library that you intend to share must meet the GPLv2 license. Although it is not an officially supported product of Oracle or MySQL, you can use the library under the GPLv2.

■ **Tip** There is a MySQL forum for discussing the connector. See http://forums.mysql.com/list.php?175. If you get stuck and need some help, check the forum for possible answers.

Installing Connector/Arduino

There are two ways to get and install Connector/Arduino. The first and the recommended method is to use the Library Manager to search for and install the library. From any sketch, click the Sketch ➤ Include Library ➤ Manage Libraries menu. This opens the Library Manager. In the filter search box, enter **MySQL**, then choose the connector, and finally click Install. In seconds, the new library is installed and ready for use. Cool, eh? Figure 6-12 shows the Library Manager dialog with the MySQL Connector/Arduino library selected for installation.

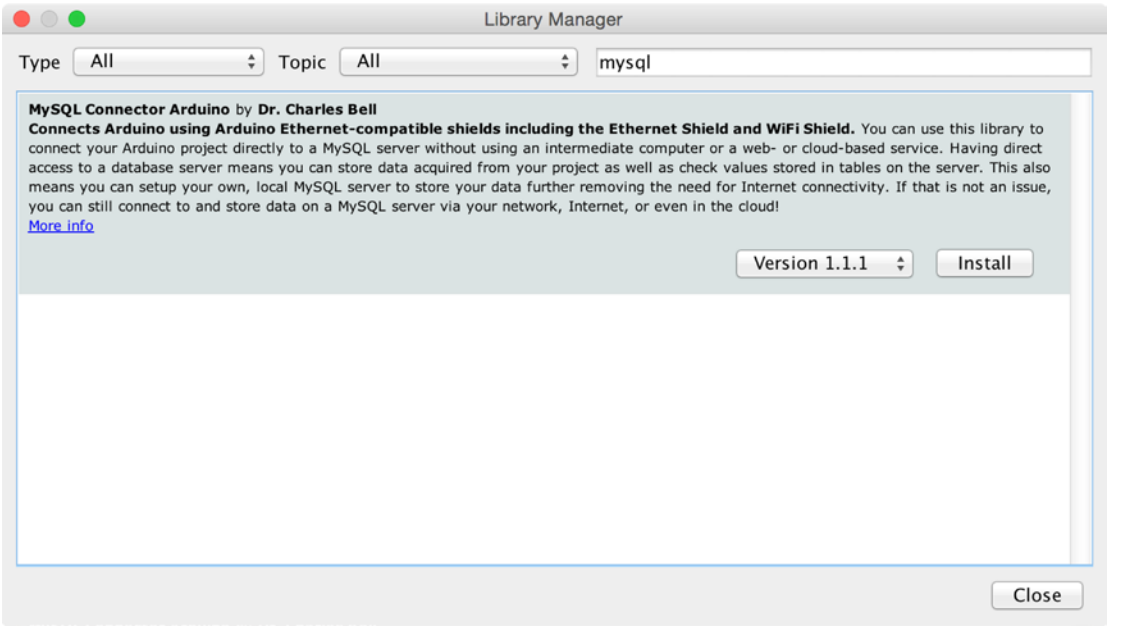


Figure 6-12. *Arduino Library Manager dialog*

If you do not want to use Library Manager or cannot use it because you're using a different IDE or editor, you can download it from the GitHub site (`https://github.com/ChuckBell/MySQL_Connector_Arduino`).

To manually install the connector, begin by navigating to the Connector/Arduino page on GitHub (`https://github.com/ChuckBell/MySQL_Connector_Arduino`). The latest version is always the one available for download. The file is named `MySQL_Connector_Arduino-master.zip`. Look on the right side of the page and click the button to download and save it to your computer. Once it is downloaded, uncompress the file. You will see a new folder in the location where you extracted the file.

You need to copy or move the folder to your `Arduino/Libraries` folder. Place the folder and its contents renamed to `MySQL_Connector_Arduino` in your Arduino library folder. You can find where this is by examining the preferences for the Arduino environment, as shown in Figure 6-13.

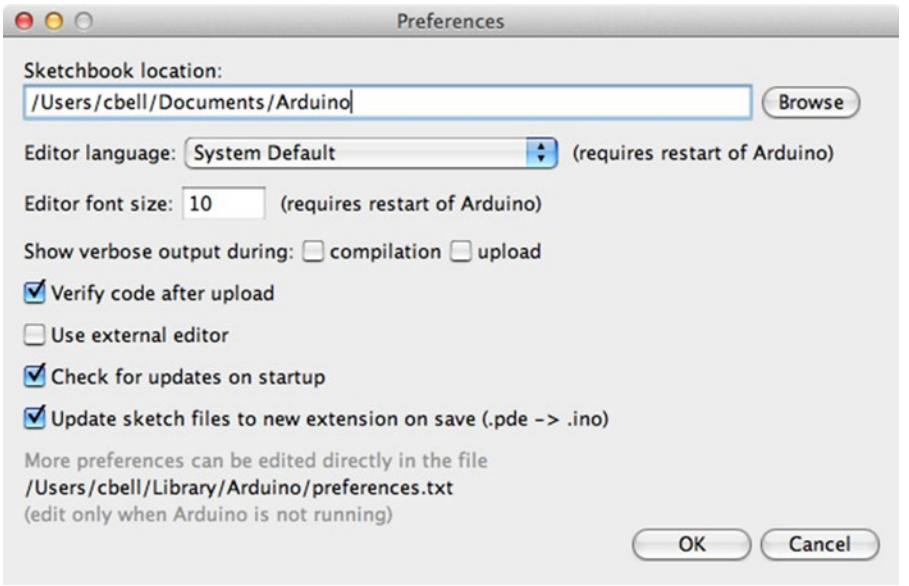


Figure 6-13. *Arduino Preferences dialog*

■ **Tip** If you copy a library to your `Libraries` folder while the Arduino application is running, you must restart it to detect the new library.

Now that you have the Connector/Arduino library installed, you are ready to start writing database-enabled sketches! Before you jump into to the library source code, let's first examine some of the limitations of using the library.

WAIT! I HAVE VERSION 1.0. CAN'T I USE THAT?

If you have already discovered the Connector/Arduino library and have been using version 1.0.4 or older, you will need to upgrade to the newer version to use the examples in this book. This is because a lot of changes were made in the 1.1 version, making it incompatible with the older versions.

But do not despair, because the old version remains in Launchpad and will be left there for some time. Best of all, the new version does not cause conflicts with any of your existing sketches. That is, your existing sketches will not be affected by installing the new library.

However, if you want to use the newest version in your existing sketches, you will have to change a few things. Please see the "Changes from Previous Versions" in the reference manual located in the extras folder of the library source code.

Using Connector/Arduino

Let's begin with a simple sketch designed to insert a single row into a table in MySQL. You are creating a "Hello, world!" sketch (but saved in a database table). All database-enabled sketches share the same common building blocks. These include setting up a database to use, creating a sketch with a specific set of include files, connecting to the database server, and executing queries. This section walks through the basic steps needed to create and execute a database-enabled sketch.

■ **Tip** The library includes a number of examples sketches to get you going quickly. Check out the examples in your quest to master the library. You will find examples of how to connect using WiFi and even how to build complex queries from variables in your sketch.

The first thing you need is a database server! Begin by creating a database and a table to use to store the data. For this experiment, you create a simple table with two columns: a text column (`char`) to store a message and a `TIMESTAMP` column to record the date and time the row was saved. I find the `TIMESTAMP` data type to be an excellent choice for storing sensor data. It is rare that you would not want to know when the sample was taken! Best of all, MySQL makes it easy to use. In fact, you need pass only a token `NULL` value to the server, and it generates and stores the current timestamp itself.

Listing 6-4 shows a MySQL client (named `mysql`) session that creates the database and the table and inserts a row into the table manually. The sketch will execute a similar `INSERT` statement from your Arduino. By issuing a `SELECT` command, you can see each time the table was updated.

Listing 6-4. Creating the Test Database

```
$ mysql -uroot -psecret
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 102
Server version: 5.6.14-log Source distribution

Copyright (c) 2000, 2011, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> CREATE DATABASE test_arduino;
Query OK, 1 row affected (0.00 sec)

mysql> USE test_arduino;
Database changed
mysql> CREATE TABLE hello (source char(20), event_date timestamp);
Query OK, 0 rows affected (0.01 sec)
```

```
mysql> GRANT ALL ON *.* to 'root'@'%' IDENTIFIED BY 'secret';

mysql> INSERT INTO hello VALUES ('From Laptop', NULL);
Query OK, 1 row affected (0.00 sec)

mysql> SELECT * FROM hello;
+-------------+---------------------+
| source      | event_date          |
+-------------+---------------------+
| From Laptop | 2013-02-16 20:40:12 |
+-------------+---------------------+
1 row in set (0.00 sec)

mysql>
```

Starting a New Sketch

It is time to start writing your sketch. Open your Arduino environment, and create a new sketch named `hello_mysql`. The following sections detail the parts of a typical MySQL database-enabled sketch. You begin with the required include files.

Include Files

To use the Connector/Arduino library, recall that it requires an Ethernet shield and therefore the Ethernet library. The Connector/Arduino library requires the `MySQL_Connection` library for connecting and the `MySQL_Cursor` library for running queries. Thus, you must include each of these in order. The following shows all the library header files you need to include at a bare minimum for a MySQL database-enabled sketch. Go ahead and enter these now.

```
#include <Ethernet.h>
#include <MySQL_Connection.h>
#include <MySQL_Cursor.h>
```

Preliminary Setup

With the include files set up, you next must take care of some preliminary declarations. These include declarations for the Ethernet library and Connector/Arduino.

The Ethernet library requires you to set up a MAC address and the IP address of the server. The MAC address is a string of hexadecimal digits and need not be anything special, but it should be unique among the machines on your network. It uses Dynamic Host Control Protocol (DHCP) to get an IP address, DNS, and gateway information. The IP address of the server is defined using the `IPAddress` class (which stores the value as an array of four integers, just as you would expect).

On the other hand, the Ethernet class also permits you to supply an IP address for the Arduino. If you assign an IP address for the Arduino, it must be unique for the network segment to which it is attached. Be sure to use an IP scanner to make sure your choice of IP address isn't already in use.

The following shows what these statements would look like for a node on a 10.0.1.X network:

```
/* Setup for Ethernet Library */
byte mac_addr[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED };
IPAddress server_addr(10, 0, 1, 23);
```

Next, you need to set up some variables for Connector/Arduino. You need to define a reference to the connection class. We dynamically allocate the cursor class later in order to manage memory better. The connection class requires a parameter that is an instance of the `Ethernet.Client` class. If you are using an Arduino Ethernet Shield, you can simply use the `EthernetClient` class. If you are using another library or a WiFi shield, you would use the appropriate client. For example, use WiFiClient for the Arduino WiFi shield.

You also need some strings to use for the data you use in the sketch. At a minimum, these include a string for the user ID, another for the password, and one for the query you use. This last string is optional because you can just use the literal string directly in the query call, but it is good practice to make strings for the query statements. It is also the best way to make queries parameterized for reuse.

The following is an example of the statements needed to complete the declarations for your sketch:

```
/* Setup for the Connector/Arduino */
EthernetClient client;
MySQL_Connection conn((Client *)&client);

char user[] = "root";
char password[] = "secret";
char INSERT_SQL[] = "INSERT INTO test_arduino.hello VALUES ('Hello from Arduino!', NULL)";
```

Notice the `INSERT` statement. You include a string to indicate that you are running the query from your Arduino. You also include the `NULL` value so that the server will create the timestamp for the row as shown in the manual execution previously.

Connecting to a MySQL Server

That concludes the preliminaries; let's get some code written! Next, you change the `setup()` method. This is where the code for connecting to the MySQL server should be placed. Recall that this method is called only once each time the Arduino is booted. The following shows the code needed:

```
void setup() {
  Ethernet.begin(mac_addr);
  Serial.begin(115200);
  while (!Serial);
  delay(1000);
  Serial.println("Connecting...");
  if (conn.connect(server_addr, 3306, user, password))
    delay(500);
  else
    Serial.println("Connection failed.");
}
```

The code begins with a call to the Ethernet library to initialize the network connection. Recall that when you use the `Ethernet.begin()` method, passing only the MAC address as shown in the example, it causes the Ethernet library to use DHCP to obtain an IP address. If you want to assign an IP address manually, see the `Ethernet.begin()` method documentation at http://arduino.cc/en/Reference/EthernetBegin.

Next is a call to serial monitor. Although not completely necessary, it is a good idea to include it so you can see the messages written by Connector/Arduino. If you have problems with connecting or running queries, be sure to use the serial monitor so you can see the messages sent by the library.

Now comes a call to the `delay()` method. You issue this wait of one second to ensure that you have time to start the serial monitor and not miss the debug statements. Feel free to experiment with changing this value if you need more time to start the serial monitor.

After the delay, you print a statement to the serial monitor to indicate that you are attempting to connect to the server. Connecting to the server is a single call to the Connector/Arduino library named `connect()`. You pass the IP address of the MySQL database server, the port the server is listening on, and the username and password. If this call passes, the code drops to the next `delay()` method call.

This delay is needed to slow execution before issuing additional MySQL commands. Like the previous delay, depending on your hardware and network latency, you may not need this delay. You should experiment if you have strong feelings against using delays to avoid latency issues. On the other hand, should the connection fail, the code falls through to the print statement to tell you the connection has failed.

Running a Query

Now it is time to run the query. We first instantiate an instance of the `MySQL_Cursor` class and pass in the connection instance. This will dtynamically allocate the class (think code). We then call the `execute()` method and pass in the query we want to run. Since there are no results returned (because we're running an `INSERT`), we can close the connection and delete the instance. The following shows all these steps in order.

Place this code in the branch that is executed after a successful connection. The following shows the previous conditional statement rewritten to include the method call to run the insert query:

```
if (conn.connect(server_addr, 3306, user, password))
  {
    delay(500);
    /* Write Hello to MySQL table test_arduino.hello */
    // Create an instance of the cursor passing in the connection
    MySQL_Cursor *cur = new MySQL_Cursor(&conn);
    cur->execute(INSERT_SQL);
    delete cur;
  }
  else
    Serial.println("Connection failed.");
}
```

Notice that you simply invoke a method named `execute()` and pass it the query you defined earlier. Yes, it is that easy!

Testing the Sketch

You now have all the code needed to complete the sketch except for the `loop()` method. In this case, you make it an empty method because you are not doing anything repetitive. Listing 6-5 shows the completed sketch.

■ **Tip** If you are having problems getting the connector working, see the "Troubleshooting Connector/Arduino" section in the MySQL Connector Reference Manual[15] and then return to this project.

---

[15] https://github.com/ChuckBell/MySQL_Connector_Arduino/blob/master/extras/MySQL_Connector_Arduino_Reference_Manual.pdf

Listing 6-5. "Hello, MySQL!" Sketch

```
/**
 * Example: Hello, MySQL!
 *
 * This code module demonstrates how to create a simple database-enabled
 * sketch.
 */
#include <Ethernet.h>
#include <MySQL_Connection.h>
#include <MySQL_Cursor.h>

/* Setup for Ethernet Library */
byte mac_addr[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED };
IPAddress server_addr(10, 0, 1, 23);   // The IP address of your database server

/* Setup for the Connector/Arduino */
EthernetClient client;
MySQL_Connection conn((Client *)&client);

char user[] = "root";
char password[] = "secret";
char INSERT_SQL[] = "INSERT INTO test_arduino.hello VALUES ('Hello from Arduino!', NULL)";

void setup() {
  Ethernet.begin(mac_addr);
  Serial.begin(115200);
  while (!Serial);
  Serial.println("Connecting...");
  if (conn.connect(server_addr, 3306, user, password))
  {
    delay(500);
    /* Write Hello, World to MySQL table test_arduino.hello */
    // Initiate the query class instance
    MySQL_Cursor *cur_mem = new MySQL_Cursor(&conn);
    // Execute the query
    cur_mem->execute(INSERT_SQL);
    delete cur_mem;
    Serial.println("Query Success!");
  }
  else
    Serial.println("Connection failed.");
}

void loop() {
}
```

Before you click the button to compile and upload the sketch, let's discuss a couple of errors that could occur. If you have the wrong IP address or the wrong username and password for the MySQL server, you could see a connection failure in the serial monitor like that shown in Figure 6-14.

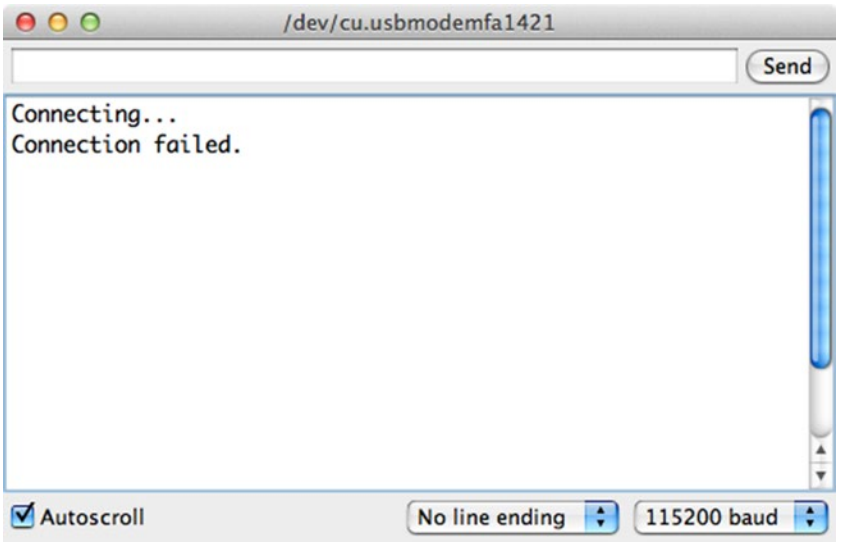


Figure 6-14. *Failed connection*

If your Arduino connects to the MySQL server but the query fails, you see an error in the serial monitor like the one shown in Figure 6-15.

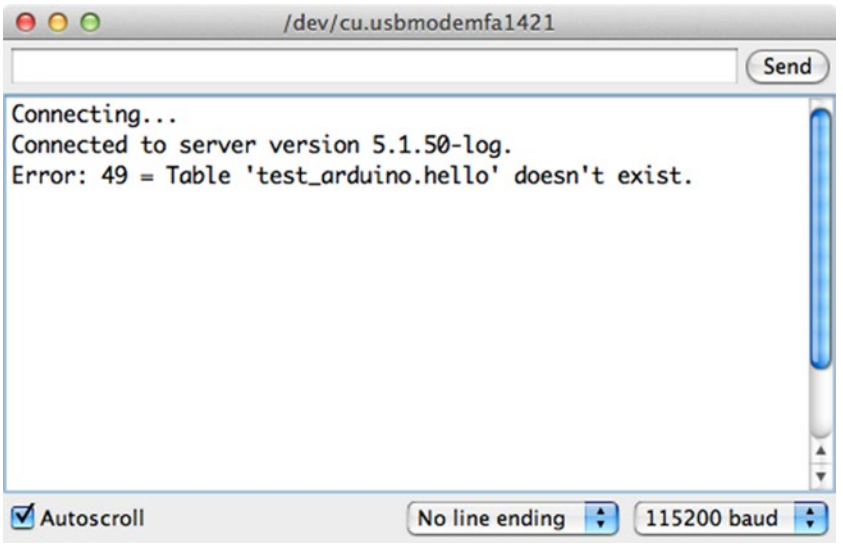


Figure 6-15. *Failed query*

Be sure to double-check the source code and the IP address of your MySQL server as well as the username and password chosen. If you are still encountering problems connecting, see the "Troubleshooting Connector/Arduino" section in the MySQL Connector Reference Manual[16] for a list of things to test to ensure that your MySQL server is configured correctly.

Once you have double-checked the server installation and the information in the sketch, compile and upload the sketch to your Arduino. Then start the serial monitor and observe the process of connecting to the MySQL server. Figure 6-16 shows a completed and successful execution of the code.

■ **Note** In the examples I am connecting to an older version of MySQL that was installed on my BeagleBone Black board. The sketch will connect to any version of MySQL from 5.0 and later.

Wow, is that it? Not very interesting, is it? If you see the statements in your serial monitor as shown in Figure 6-16, rest assured that the Arduino has connected to and issued a query to the MySQL server. To check, simply return to the `mysql` client and issue a select on the table. But first, run the sketch a number of times to issue several inserts in the table.

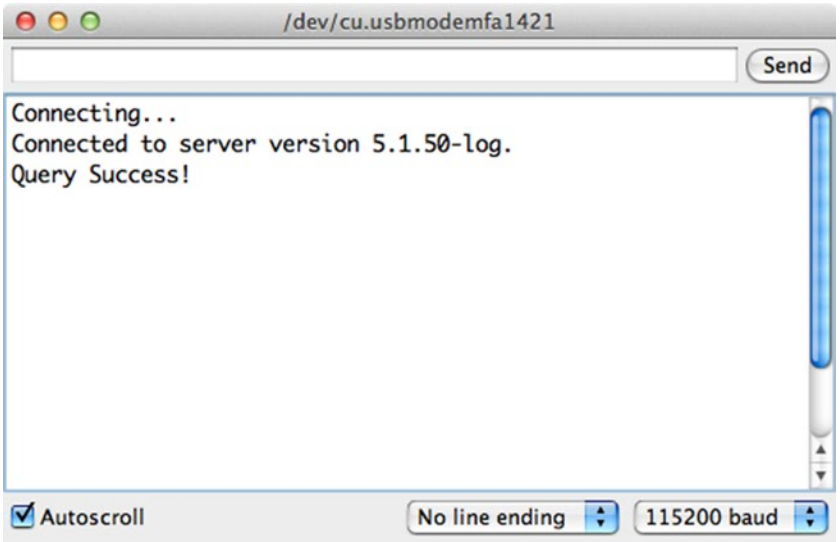


Figure 6-16. *Correct serial monitor output*

You can do this in two ways. First, you can press `RESET` on your Arduino. If you leave your serial monitor running, the Arduino presents the messages in order, as shown in Figure 6-17. Second, you can upload the sketch again. In this case, the serial monitor closes, and you have to reopen it. The advantage of this method is you can change the query statement each time, thereby inserting different rows into the database. Go ahead and try that now, and check your database for the changes.

[16] https://github.com/ChuckBell/MySQL_Connector_Arduino/blob/master/extras/MySQL_Connector_Arduino_Reference_Manual.pdf

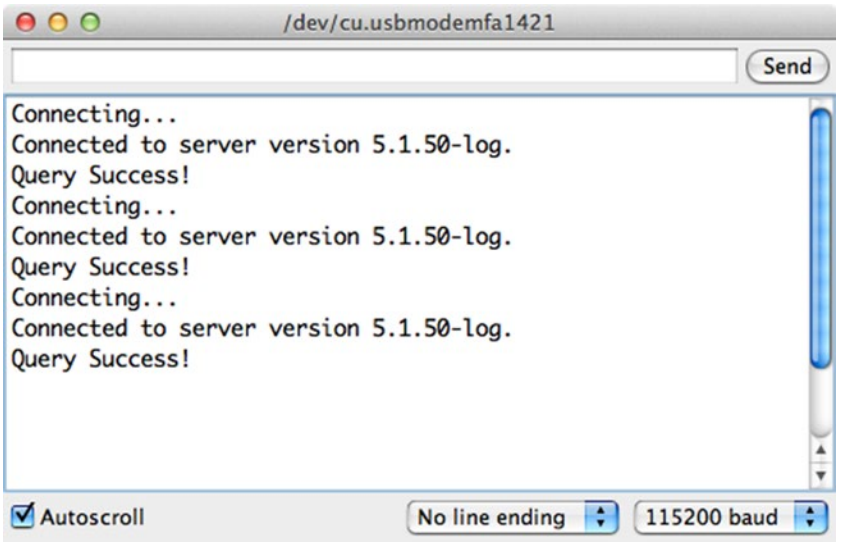


Figure 6-17. *Results of running the sketch several times*

Let's check the results of the test runs. To do so, you connect to the database server with the `mysql` client and issue a `SELECT` query. Listing 6-6 shows the results of the three runs from the example. Notice the different timestamp for each run. As you can see, I ran it once, then waited a few minutes and ran it again (I used the `RESET` button on my Arduino Ethernet shield), and then ran it again right away. Very cool, isn't it?

Listing 6-6. Verifying the Connection with the Serial Monitor

```
$ mysql -uroot -psecret
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 33
Server version: 5.6.14-log Source distribution

Copyright (c) 2000, 2010, Oracle and/or its affiliates. All rights reserved.
This software comes with ABSOLUTELY NO WARRANTY. This is free software,
and you are welcome to modify and redistribute it under the GPL v2 license

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> select * from test_arduino.hello;
+---------------------+---------------------+
| source              | event_date          |
+---------------------+---------------------+
| From laptop         | 2013-02-19 15:17:38 |
| Hello from Arduino! | 2013-02-19 15:18:12 |
| Hello from Arduino! | 2013-02-19 15:28:39 |
| Hello from Arduino! | 2013-02-19 15:29:16 |
+---------------------+---------------------+
4 rows in set (0.01 sec)

mysql>
```

You can do much more with the connector than shown here. In fact, you can query the database server for lookup data; discover values of variables; create databases, tables, views; and so on. You can do just about anything you want, within reason. Querying the database server for massive rows or large data rows is likely beyond the memory limitations of the Arduino. Fortunately, most of the sketches you will write will be saving data with simple `INSERT` statements.

■ **Tip** You can find many more examples of how to use the connector in the Connector/Arduino reference manual located in the `extras` folder. The file is named `MySQL_Connector_Arduino_Reference_Manual.pdf`.

Now that you've seen an Arduino connector, let's look at a connector more general purpose and one you can use on your laptop, desktop, low-cost computer, embedded system, and more, Anything that can run Python can write to a MySQL database!

Introducing Connector/Python

The connector for Python from Oracle is a full-featured connector that provides connectivity to the MySQL database server for Python applications and scripts. The latest version is release-2.1.3GA. Unlike the Connector/Arduino, Connector/Python is fully supported and actively maintained by Oracle.

Connector/Python features support for all current MySQL server releases from version 4.1 and newer. It is written to provide automatic data type conversion between Python and MySQL, making building queries and deciphering results easy. It also has support for compression, permits connections via SSL, and supports all MySQL SQL commands. The current version, 2.1.3, is augmented with C libraries to improve performance.

Using Connector/Python in your Python scripts consists of importing the base module, initiating a connection, and executing queries with a cursor, which is similar to Connector/Arduino. That is not surprising since I wrote Connector/Arduino using Connector/Python as a model.

However, unlike Connector/Arduino, Connector/Python has no such memory limitations, allowing you to accomplish a great deal of processing. Indeed, I would move most of my string, date, and mathematical processing to a Python script rather than attempting it on the Arduino.

Before we jump into how we can use Connector/Python to write some MySQL database–enabled applications, let's talk about how to get and install Connector/Python.

PYTHON? ISN'T THAT A SNAKE?

The Python programming language is a high-level language designed to be as close to like reading English as possible while being simple, easy to learn, and powerful. Pythonistas[17] will tell you the designers have indeed met these goals.

Python does not require a compilation step prior to being used. Rather, Python applications (whose file names end in `.py`) are interpreted on the fly. This is very powerful, but unless you use a Python integrated development environment (IDE) that contains an automatic syntax checker, some syntax errors will not be discovered until the application is executed. Fortunately, Python provides a robust exception-handling mechanism that will communicate what has gone wrong.

[17]Python experts often refer to themselves using this term. It is reserved for the most avid and experienced Python programmers.

If you have never used Python or you would like to know more about it, the following are few good books that introduce the language. A host of resources are also available on the Internet, including the Python documentation pages at www.python.org/doc/.

- *Programming the Raspberry Pi* by Simon Monk (McGraw-Hill, 2013)
- *Beginning Python from Novice to Professional, 2nd Edition*, by Magnus Lie Hetland (Apress, 2008)
- *Python Cookbook* by David Beazley and Brian K. Jones (O'Reilly Media, 2013)

Interestingly, Python was named after the British comedy troupe Monty Python and not the reptile. As you learn Python, you may encounter campy references to Monty Python episodes. Having a fondness for Monty Python, I find these references entertaining. Of course, your mileage may vary.

Installing Connector/Python

Downloading is the same process as you discovered for the server. You can download Connector/Python from Oracle's MySQL web site (http://dev.mysql.com/downloads/connector/python/). The page will automatically detect your platform and show the available downloads for your platform. You may see several choices. Be sure to choose the one that matches your configuration.

Installing on Desktop/Laptop Platforms

Since most platforms come with Python installed, you may not need to do anything to prepare your system; just download the installer and install it. You can find the latest Python installers on the product download page (http://dev.mysql.com/downloads/connector/python/). Note that Connector/Python requires either Python 2.7 (recommended) or Python 3.3.

Figure 6-18 shows a typical download page for the Ubuntu platform. Use the drop-down box to select a different platform if yours is not listed. However, if you use a platform such as Windows that does not include Python, you should install Python first.

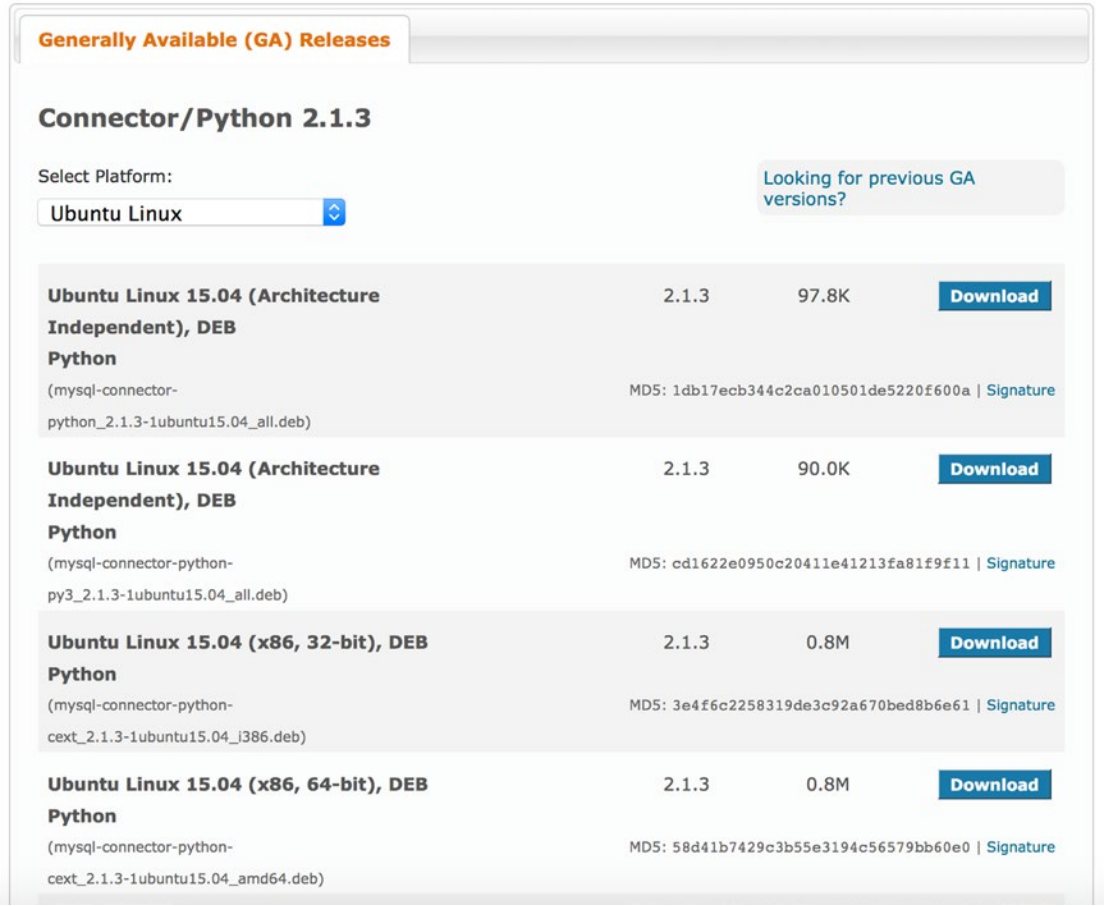


Figure 6-18. *Connector/Python download page*

As I mentioned, the current version of Connector/Python is 2.1.3, but most versions 2.0 or later should be fine for your solutions. The newest releases add high availability features for use with the enterprise-level high availability solution named MySQL Fabric. Any release of 2.0 or 2.1 will be fine for use with IOT solutions.

■ **Tip** See the online reference manual for specific notes about installing on some platforms (`http://dev.mysql.com/doc/connector-python/en/connector-python-installation.html`).

Installing on Low-Cost Platforms

Installing Connector/Python on the smaller platforms is a bit more involved but not overly so. In short, you install Connector/Python by using the packaging mechanism for the platform (`apt-get`, `opkg`, and so on). The following shows how to install Connector/Python on the Raspberry Pi using Raspbian Jessie. Other platforms are similar.

```
$ sudo pip3 install mysql-connector-python --allow-external mysql-connector-python
Downloading/unpacking mysql-connector-python
  mysql-connector-python an externally hosted file and may be unreliable
  Downloading mysql-connector-python-2.0.4.zip (277kB): 277kB downloaded
  Running setup.py (path:/tmp/pip-build-mm9szi9x/mysql-connector-python/setup.py) egg_info
  for package mysql-connector-python

Installing collected packages: mysql-connector-python
  Running setup.py install for mysql-connector-python

Successfully installed mysql-connector-python
Cleaning up...
```

Notice I used the Python package manager (from PyPi) to get and install the connector. This installs an older version of the connector, but it is fully functional and will meet your IOT solution needs.

Checking the Installation

Once Connector/Python is installed, you can verify it is working with the following short example. Begin by entering the command `python`. This will open an interactive prompt that permits you to enter one line of Python code at a time and execute it; it's a Python command-line interpreter and useful in testing small snippets of code. Just enter the following lines as shown in the example:

```
$ python
Python 2.7.6 (default, Mar  4 2014, 16:53:21)
[GCC 4.2.1 Compatible Apple LLVM 5.0 (clang-500.2.79)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> import mysql.connector
>>> print mysql.connector.__version__
2.1.3
>>> quit()
```

What you should see is the version of Connector/Python printed. If you see any errors about not finding the connector, be sure to check your installation to ensure it worked. Once you can successfully access Connector/Python, you're ready to move on to some examples.

■ **Tip** If you have multiple versions of Python installed and installed Connector/Python under a different Python version than the default, use the version-specific Python executable. For example, if you installed Connector/Python under Python3 but Python2.7 is the default, use the command `python3` to start the interpreter. Otherwise, you may see errors when doing the import.

Using Connector/Python

Let's start with a simple example where we connect to the MySQL server and get a list of databases. In this case, we start by importing the Connector/Python `connector` class and then call the `connect()` method to connect to the server. To keep things tidy, we use a dictionary to store the connection information. Be sure your MySQL server is running and you change the following example to match your setup. For example, provide the correct password and hostname for the server.

```
$ python
Python 2.7.6 (default, Mar  4 2014, 16:53:21)
[GCC 4.2.1 Compatible Apple LLVM 5.0 (clang-500.2.79)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> import mysql.connector
>>> server = {
...   'user': 'root',
...   'password': <secret>,
...   'host': '127.0.0.1',
...   'database': 'employees',
...   'raise_on_warnings': True,
... }
>>> cnx = mysql.connector.connect(**server)
```

If you get errors at this point, go back and check your connection parameters. It is most likely that your MySQL server is unreachable, it is down, or you have the wrong credentials specified. If you get no response from the interactive interpreter, that's OK—you've connected and you are ready to go!

Now, leave the interpreter running and add the following statements. Here we will open a cursor object for executing queries and retrieving rows. I wrote the loop in a simplistic way to show you how to loop through all available rows (there are several other, valid ways to write that bit).

```
>>> cur = cnx.cursor()
>>> cur.execute("SHOW DATABASES")
>>> rows = cur.fetchall()
>>> for row in rows:
...     print row
...
```

When you get the `...` (which is a prompt), press Enter and observe the results as shown here. Your list may be slightly different depending on what databases are on your server.

```
(u'arduino',)
(u'employees',)
(u'library',)
(u'mysql',)
(u'performance_schema',)
(u'plant_monitoring',)
(u'test',)
(u'test_arduino',)
(u'world',)
(u'world_innodb',)
```

But we're not done. There are two more steps needed. We need to close the cursor and connection and then exit the interpreter.

```
>>> cur.close()
True
>>> cnx.close()
>>> quit()
```

Congratulations! You've written your first MySQL-enabled Python script. Now that we know the basics, let's move on to more powerful scripts using a source file instead of the interactive interpreter.

Most Python scripts (applications) are built using a file named `<something>.py` and executed from the command line as follows. We will use this method to execute the following examples. Thus, for each example, you should open a file and enter the text as shown.

```
$ python my_script.py
```

Now let's see how we can insert some data in a table. In this case, we simply want to read data from a file and insert it into a table. I'll let you use your imagination for how you could change the file to reading sensors. In fact, I will show you how to do this in a later chapter.

■ **Note** Refer to Chapter 5 for the table layout. Be sure to empty the table if you performed the examples from Chapter 5 so you can avoid key violations when running this example.

Open your favorite text editor and enter the code shown in Listing 6-7. Save the file with the name `simple_insert.py`.

Listing 6-7. Inserting Data with Connector/Python

```
import mysql.connector
server = {
  'user': 'root',
  'password': <secret>,
  'host': '127.0.0.1',
  'database': 'employees',
  'raise_on_warnings': True,
}
cnx = mysql.connector.connect(**server)
cur = cnx.cursor()
# read rows from a file for inserting into plant_monitor table
f = open("plants_data.txt")
lines = f.readlines()
f.close()
# now insert the data
for line in lines:
    cols = line.strip('\n').split(",") # comma-separated row
    query = "INSERT INTO plant_monitoring.plants (name, location, climate)" \
            " VALUES ('{0}','{1}',{2});".format(cols[0], cols[1], cols[2])
    print query
    cur.execute(query)
cnx.commit()
cur.close()
cnx.close()
```

Here we see the same startup code as the previous example only this time we're reading values from a file and performing an `INSERT` SQL statement on each. Take a moment to study the code and how it works. Note that it uses string substitution.

Notice the line in bold. This command is necessary to ensure the rows are written to the table. More specifically, since I used a transactional storage engine and my server is set up for transactions, none of the data is written until I explicitly commit the changes. Your server may be set up differently, but it does not hurt to add this command here. You could add it inside the loop too, but it is best to push your commits out to the latest code block, and in this case it is outside the loop.

The file we are reading has only a few rows and is a mockup of the plant-monitoring system example from Chapter 5. Listing 6-8 shows the file contents. Note that I labeled it `plants_data.txt`. If you change the file name, be sure to change the code accordingly.

Listing 6-8. Sample Data

```
Jerusalem Cherry,deck,2
Moses in the Cradle,patio,2
Peace Lilly,porch,1
Thanksgiving Cactus,porch,1
African Violet,porch,1
```

To run the script, issue the following command from the folder where you stored the file. Be sure to put the data file in the same folder first. I show the results of running the script.

```
$ python ./simple_insert.py
INSERT INTO plant_monitoring.plants (name, location, climate) VALUES ('Jerusalem
Cherry','deck',2);
INSERT INTO plant_monitoring.plants (name, location, climate) VALUES ('Moses in the
Cradle','patio',2);
INSERT INTO plant_monitoring.plants (name, location, climate) VALUES ('Peace
Lilly','porch',1);
INSERT INTO plant_monitoring.plants (name, location, climate) VALUES ('Thanksgiving
Cactus','porch',1);
INSERT INTO plant_monitoring.plants (name, location, climate) VALUES ('African
Violet','porch',1);
```

Now let's check our table. If we started with an empty table, we should see the following:

```
mysql> SELECT * FROM plant_monitoring.plants;
+----+--------------------+----------+---------+
| id | name               | location | climate |
+----+--------------------+----------+---------+
| 30 | Jerusalem Cherry   | deck     | outside |
| 31 | Moses in the Cradle | patio    | outside |
| 32 | Peace Lilly        | porch    | inside  |
| 33 | Thanksgiving Cactus | porch    | inside  |
| 34 | African Violet     | porch    | inside  |
+----+--------------------+----------+---------+
5 rows in set (0.00 sec)
```

You can do much more with the connector than shown here. In fact, you can do just about anything you want. Typically, I use Python scripts for performing complex operations on my databases and database servers. For example, I may write a script to set up all of my databases, tables, functions, and so on, so that I can reload a test or start an experiment on any server I want; I just supply different connection parameters and run it.

MYSQL UTILITIES: PYTHON-BASED DATABASE ADMINISTRATION

If you are a Python developer and find yourself working more and more with MySQL, especially if you have found yourself in the role of administrator, you may want to look at MySQL Utilities from Oracle. MySQL Utilities is a set of Python scripts and a Python library for managing MySQL servers. You can do all manner of things from copying user permissions to cloning servers to discovering differences between two databases. See http://dev.mysql.com/downloads/utilities/ for more details.

You may also want to write complex Python scripts for manipulating your data from your sensors or IOT data collectors. That is, you could use a Raspberry Pi as a data aggregator for preparing data for storage. You can even use Python applications to read sensors directly from the Raspberry Pi as I demonstrated in my book *Beginning Sensor Networks with Arduino and Raspberry Pi* (Apress, 2014).

For more complex examples including executing transactions, creating tables, and running complex queries, see the coding examples section in the Connector/Python online reference manual (http://dev.mysql.com/doc/connector-python/en/connector-python-examples.html).

Summary

This chapter introduced MySQL and gave you a crash course on how to set up a Raspberry Pi, install MySQL, and use it. You also learned how to write data to your database server using an Arduino sketch and a Python program on another machine (Raspberry Pi, BeagleBone Black, pcDuino, and so on).

Although it does not have nearly the sophistication of a high availability, five-nines uptime (99.999 percent) database server, the low-cost Raspberry Pi with an attached USB hard drive makes for a very small-footprint database server that you can put just about anywhere. This is great because IOT solutions, by nature and often by necessity, need to be small and low cost. Having to build an expensive database server is not usually the level of investment desired.

In the next chapter, we will explore an advanced topic: high availability. More specifically, we will see how to make our database server more reliable by providing redundancy as well as an ability to separate reads (`SELECT`) and writes (`INSERT`, `UPDATE`, `DELETE`) across multiple database nodes.