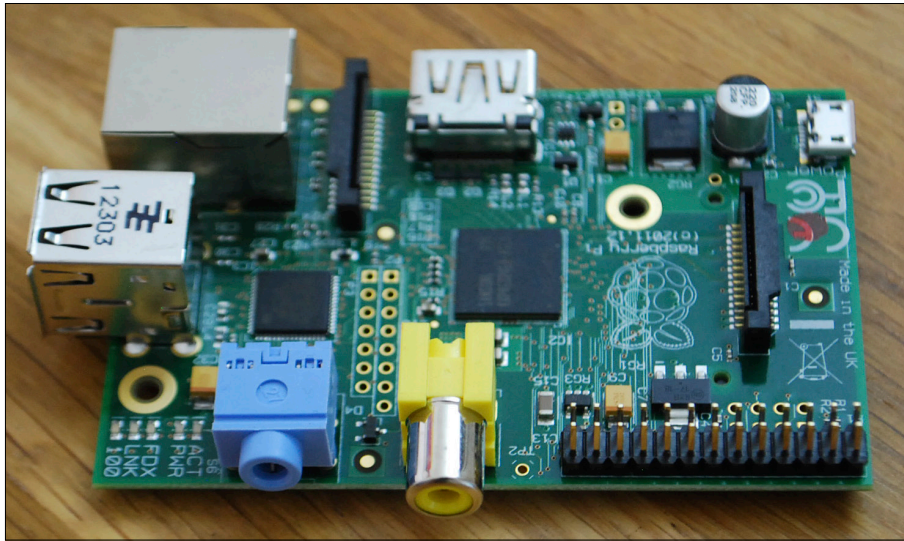


THE GENERAL PURPOSE INPUT OUTPUT (GPIO) pins on the Raspberry Pi are what make it really special. The behaviour of these pins can be controlled or programmed—by you! You can use the pins to sense and control physical objects in the real world, like lights and switches. The pins are located on the main board of the Raspberry Pi, shown in Figure 8-1.

In this adventure, you'll learn some basics of electronics and then discover how to **output** to a light-emitting diode (LED), making it light up using your Raspberry Pi. For the final project in the adventure, you'll hook up a marshmallow (yes, a real marshmallow) to **input** a signal to your Raspberry Pi to play a Scratch marshmallow game that senses the press of a button.



**FIGURE 8-1** The General Purpose Input Output (GPIO) pins on a Model B Raspberry Pi



**Input** refers to the raw data or information that can be entered into a computer system like a Raspberry Pi before it is processed. An example of an input device is a push button or a microphone. The Raspberry Pi has pins that can be connected to these and other devices.

**Output** refers to the data or information that is communicated to you as it exits a computer system like a Raspberry Pi after it has been processed. An example of an output device is a speaker or monitor screen.

## Using a Raspberry Leaf Diagram

Raspberry Pi projects using GPIO pins give you the opportunity to use electronics concepts and techniques to make something happen electronically, such as making an LED light up. Many of the pins have different purposes, and the instructions in this adventure tell you which pin to use for each connection.

There are two *revisions* of the GPIO pin layout. Later in this adventure you will learn how to find out whether your Raspberry Pi is a Revision (Rev) 1 or a Rev 2. It is important to know this before you begin any exercises in this adventure, as the instructions here are based on Rev 2. Figure 8-2 shows the layout of the pins on a Raspberry Pi Rev 2 board. You should refer to this diagram when connecting cables to the pins, and when writing your code to program them.

Raspberry Pi Revision 2			
3.3V	1	2	5V
I2C1 SDA	3	4	5V
I2C1 SCL	5	6	GROUND
GPIO4	7	8	UART TXD
GROUND		10	UART RXD
GPIO 17	11	12	GPIO 18
GPIO 27	13	14	GROUND
GPIO 22	15	16	GPIO 23
3.3V	17	18	GPIO 24
SP10 MOSI	19	20	GROUND
SP10 MISO	21	22	GPIO 25
SP10 SCLK	23	24	SP10 CE0 N
GROUND	25	26	SP10 CE1 N

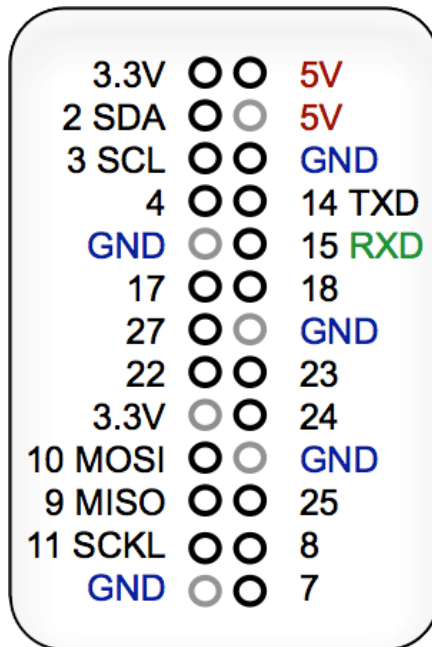
**FIGURE 8-2** Raspberry Pi GPIO layout, Revision 2 board

You can find this diagram, along with a diagram of the Rev 1 board, at [www.hobbytronics.co.uk/raspberry-pi-gpio-pinout](http://www.hobbytronics.co.uk/raspberry-pi-gpio-pinout).

If you have a Rev 1 Raspberry Pi, you can still follow the steps, but ensure that you refer to the layout diagram for your board and use the pin numbers from that diagram in your programs, and not those specified in the instructions.



To make it easier to tell which pin is which, Dr Simon Monk has created a *Raspberry Leaf* template with a label for each pin—you can cut these out and place them over the pins. It is a good idea to download, print and cut out a Raspberry Leaf to help you know which pins to use in the projects in this adventure. You can download the template from the Adventure in Raspberry Pi website or from Dr Monk’s electronics website ([www.doctormonk.com/2013/02/raspberry-pi-and-breadboard-raspberry.html](http://www.doctormonk.com/2013/02/raspberry-pi-and-breadboard-raspberry.html)). Dr Monk’s site has templates for both the Rev 1 and Rev 2 boards; be sure to download the correct version for your board. Figure 8-3 shows the Raspberry Leaf for the Rev 2 board.



**FIGURE 8-3** The Raspberry Leaf for the Rev 2 board



You should take great care when connecting cables to the GPIO pins on your Raspberry Pi. There are two reasons for this. First, you should be cautious to protect yourself from harm. Second, the Raspberry Pi is a 3.3V device, and if you plug in anything at a higher voltage than that it will damage the processor and possibly render your board useless.

It is also very important that you connect any external components to the correct pins on the Raspberry Pi, so it is essential that you refer to the correct GPIO revision layout diagram.

# Electronic Basics

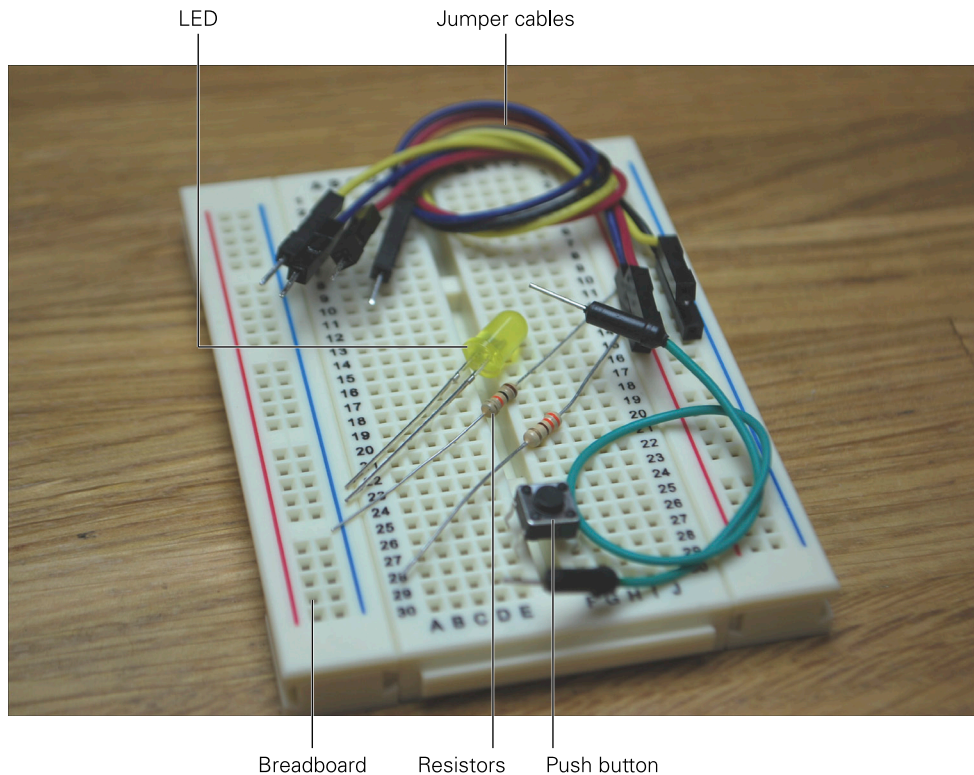
You are delving into a new world of electronics by using the Raspberry Pi's GPIO pins. If you have never created any electronic circuits before, following the tutorials in this adventure will be a good place to learn basic electronics. To get started, you should become familiar with the electronic concepts and components in the following list.

- **Current** is the rate at which electrical energy flows past a point in a circuit. It is the electrical equivalent of the flow rate of water in pipes. Current is measured in amperes (A). Smaller currents are measured in milliamperes (mA).
- **Voltage** is the difference in electrical energy between two points in a circuit. It is the electrical equivalent of water pressure in pipes, and it is this pressure that causes a current to flow through a circuit. Voltage is measured in volts (V).
- **Resistors** are electrical components that resist current in a circuit. For example, LEDs can be damaged by too much current, but if you add the correct value resistor in series with the LED in the circuit to limit the amount of current, the LED will be protected. Resistance is measured in *ohms*. You need to pick a resistor with the correct value to limit the current through a circuit; the value of a resistor is shown by coloured bands that are read from left to right. The exercises in this adventure will use some resistors and explain how to read the value.
- A **diode** is a device that lets current flow in only one direction. A diode has two terminals, called *anode* and *cathode*. Current will flow through the diode only when positive voltage is applied to the anode, and negative voltage to the cathode.
- A **light-emitting diode** or **LED** is a diode that lights up when electricity passes through it. An LED is an example of an output device. LEDs allow current to pass in only one direction. They come in a variety of colours, and have one short leg and one long leg, which helps you to determine which way round they need to be placed in a circuit for current to flow through them. The exercises in this adventure will use some LEDs.
- A **capacitor** is used to store an electric charge. The capacity that this component has is measured in farads (F). A farad is a very large quantity, so most of the capacitors you see will be measured in microfarads.
- A **breadboard** is a reusable device that allows you to create circuits without needing to solder all the components. Breadboards have a number of holes into which you can push wires or jumper cables and components to create circuits. The two columns of holes on either side of the breadboard, between red and blue lines, are for power. The column next to the red line is for positive connections and the column next to the blue line is for negative connections (see Figure 8-4 for an example of a breadboard). The exercises in this adventure use a breadboard.
- **Jumper cables** can be used to connect the GPIO pins on the Raspberry Pi to a breadboard or other components. They are reusable and do not require

soldering. They come in different formats: female-to-male; female-to-female; and male-to-male.

- A **circuit diagram** shows you which electronic components, represented by symbols, are connected to complete a circuit and in what order they should be placed. The exercises in this adventure include circuit diagrams to help you understand how the circuit works and to show you the order in which the components need to be placed for current to flow through.

Figure 8-4 shows a half size breadboard, a variety of jumper cables, an LED, a push button and some resistors.



**FIGURE 8-4** Electronic components

## Using the Python Library to Control GPIO

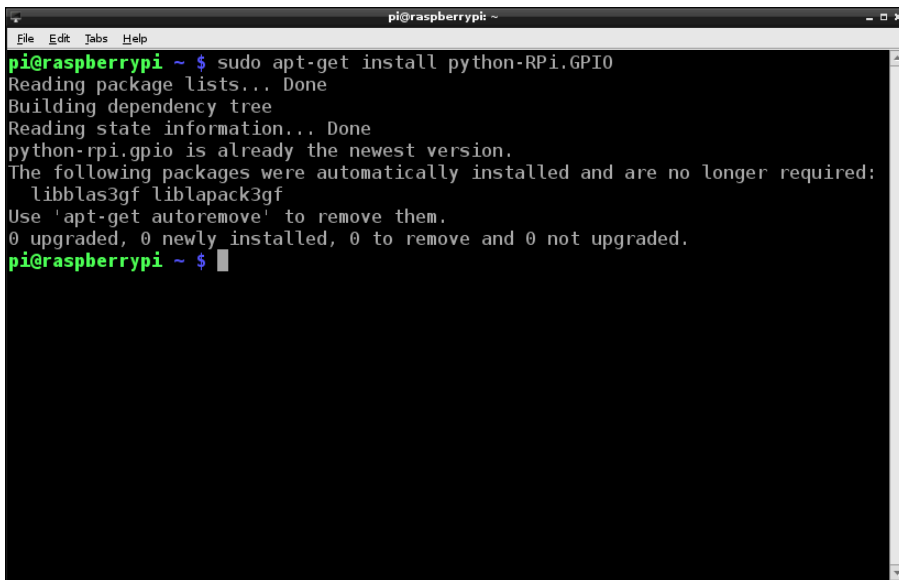
To use Python to control the GPIO pins, you first need the Python GPIO library installed onto your Raspberry Pi. A library is a collection of already written code that



you can use. For example, libraries contain modules that you can use in your code, like the `sleep` function from the `time` module that you used in previous adventures. In this adventure, you use the GPIO library to sense and control the pins. To check whether you have it, open an LXTerminal window from the desktop and type the following command:

```
sudo apt-get install python-RPi.GPIO
```

The Python library to control GPIO should be preinstalled on your Raspberry Pi. If you are using an early distribution of Raspbian or another Pi operating system, however, you may need to download and install it. To do so, follow the onscreen instructions, as shown in Figure 8-5.

A screenshot of a terminal window titled 'pi@raspberrypi ~'. The terminal shows the command 'sudo apt-get install python-RPi.GPIO' being executed. The output indicates that the package is already installed and is the newest version. It also lists some automatically installed packages that are no longer required: 'libblas3gf' and 'liblapack3gf'. The terminal ends with the prompt 'pi@raspberrypi ~ \$' and a cursor.

```
pi@raspberrypi ~ $ sudo apt-get install python-RPi.GPIO
Reading package lists... Done
Building dependency tree
Reading state information... Done
python-rpi.gpio is already the newest version.
The following packages were automatically installed and are no longer required:
  libblas3gf liblapack3gf
Use 'apt-get autoremove' to remove them.
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
pi@raspberrypi ~ $
```

**FIGURE 8-5** Downloading and installing the Python GPIO Library on the Raspberry Pi

## Do You Have a Rev 1 or a Rev 2 Board?

As I have mentioned, there are two versions of the Raspberry Pi GPIO pin layout: the original Rev 1 and the newer, improved Rev 2. Before you start the exercises in this adventure, you need to find out which version your Raspberry Pi has. There is a simple way to do this.

In the LXTerminal window opened from the Raspberry Pi Desktop, type this code:

```
sudo python
import RPi.GPIO as GPIO
GPIO.RPI_REVISION
```



Using `sudo` in the command `sudo python` runs Python as the root or super user so that it can gain access to the GPIO hardware, which is not accessible to a normal user.

The first line runs the Python interpreter in interactive mode, while the second line uses `import` to use the `RPi.GPIO` library. The last line will determine what revision of Raspberry Pi you are using. Figure 8-6 shows this code and the output. The Raspberry Pi in this figure has returned the value of `2`, meaning that it is a Raspberry Pi Rev 2 board.

A screenshot of an LXTerminal window titled "pi@raspberrypi ~". The terminal shows the following text:

```
pi@raspberrypi ~ $ sudo python
Python 2.7.3 (default, Jan 13 2013, 11:20:46)
[GCC 4.6.3] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> import RPi.GPIO as GPIO
>>> GPIO.RPI_REVISION
2
>>>
```

**FIGURE 8-6** Using `GPIO.RPI_REVISION` in LXTerminal to find out which Raspberry Pi revision you have

To close the Python shell, type:

```
quit()
```

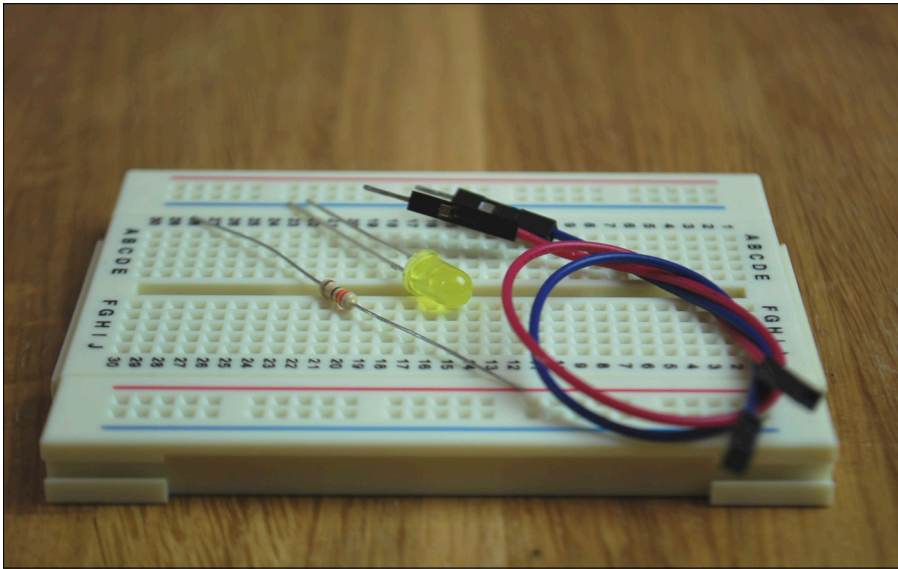


# Making an LED Blink

Now that you have the Python library installed, you can use the GPIO pins to make something physical happen. In this project you will control an LED and make it blink.

Along with your Raspberry Pi you will need the following items, shown in Figure 8-7:

- A breadboard
- Two jumper cables
- An LED
- A 330 ohm resistor



**FIGURE 8-7** Components for exercise to make an LED blink

You can purchase the components that you need from the following shops:

Adafruit—[www.adafruit.com/](http://www.adafruit.com/)

CPC Farnell—<http://cpc.farnell.com>

RS Components—<http://uk.rs-online.com/web>

SKPang—[www.skpang.co.uk](http://www.skpang.co.uk)



## Creating the LEDblink Python Code

The first part of the project is to write the code that will make your LED blink.



For a video that walks you through the LEDblink project, visit the companion website at [www.wiley.com/go/adventuresinrp](http://www.wiley.com/go/adventuresinrp). Click the Videos tab and select the LEDblink file.

1. Open Python IDLE 3 to program the GPIO pins, and select File→New Window to create a blank text editor window to write your Python code to control the GPIO pins.

Type the following code into your text editor window:

```
import RPi.GPIO as GPIO
import time
```

These two lines import the modules and their functions that you will need to control the GPIO pins on the Raspberry Pi, and to create timed delays between the LED turning on and off. (You used the `time` module in Adventure 5 to wait for user input in the inventory program and the text-based adventure game.)

2. Underneath those lines, set the mode of pin numbering you are going to use, either BCM or BOARD (see the “Digging into the Code” sidebar for more details). You are then able to set up the individual pins on the Raspberry Pi.

```
GPIO.setmode(GPIO.BCM)
GPIO.setup(24, GPIO.OUT)
```

In this project, you are outputting to an LED. Therefore you need to set up the pin that the LED connects to on the Raspberry Pi as an output. To do this, use the command `GPIO.setup(the GPIO number, GPIO.OUT)`.

3. Next, use a `while True` loop to set the output of GPIO 24 to `True`, which will turn on the LED followed by a pause for one second. Then set the output of GPIO 24 to `False`, followed by another one-second pause. When this is looped over and over, the LED will repeatedly turn on (true) and off (false) with a one-second delay between each.

```
while True:
    GPIO.output(24, True)
    time.sleep(1)
    GPIO.output(24, False)
    time.sleep(1)
```

4. Save the file as `LEDblink.py` in `Documents`.

## DIGGING INTO THE CODE

You might be wondering about the `setmode` and `BCM` in the first line of the Step 2. There are two ways—*modes*—of numbering the GPIOs within RPi. GPIO—`BCM` and `BOARD`. It is important that you set the mode (`setmode`) you wish to use in your Python code. `BOARD` means that the numbers passed into the GPIO functions refer to the pin numbers of the header on the Raspberry Pi. `BCM` means that numbers passed into the GPIO functions refer to what the BroadCoM chip uses for its GPIOs (which are different from the actual pin numbers on the RPi header).

The exercises in this adventure use the `BCM` numbers, which is why you needed to include `GPIO.setmode(GPIO.BCM)` at the start of your GPIO program. To use the `BOARD` mode, you would write `GPIO.setmode(GPIO.BOARD)`.

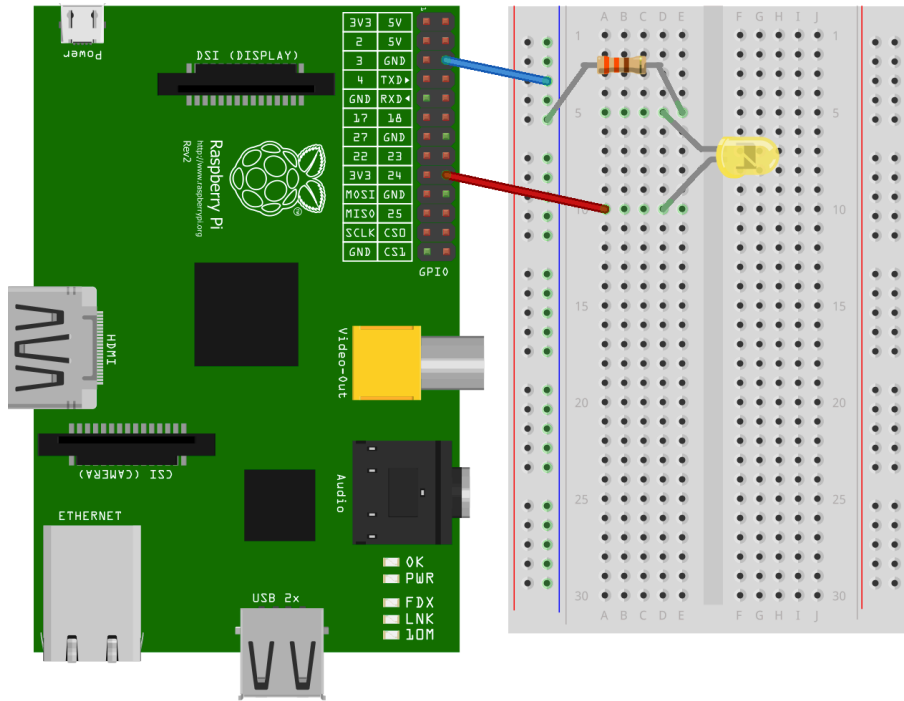
## Connecting the LEDblink Components

Before running your program to make the LED blink, you need to assemble the electronic components and connect them to the Raspberry Pi GPIO pins. Figure 8-8 shows the Raspberry Pi on the left and a breadboard on the right. Use this diagram and the following steps to help you connect the right cables and components in the circuit.

As noted earlier, these instructions are written for a Rev 2 board. If you have a Rev 1, you must adjust the instructions based on the Rev 1 layout diagram. You could cause permanent damage your Raspberry Pi if you connect circuits to the GPIO pins incorrectly.



1. Start by plugging a female-to-male jumper cable from GPIO pin 24 of your Raspberry Pi to the A10 hole on your breadboard (the red cable in Figure 8-8). It helps to use different coloured wires. The jumper cable will easily fit onto the pins of your Raspberry Pi on one end, and into the holes on the breadboard. Make sure that you gently push them down as far as they will go to make a secure connection.
2. Next, plug another female-to-male jumper cable from a ground pin, sometimes represented as GND on Raspberry Pi GPIO diagrams (the blue cable). On a Rev 2 Raspberry Pi board, the third pin from the top on the outside strip is a ground pin (see Figure 8-8). Remember to use Dr Monk's Raspberry Leaf so that you know which pins are which!



Made with  Fritzing.org

**FIGURE 8-8** Circuit diagram to connect components to Raspberry Pi for a blinking LED

3. Plug the other end of the jumper cable into a hole in the second column, between the red and blue lines on the breadboard. Remember that these two columns between the red and blue lines are for power—one for positive (red) and one for negative (blue). You want to plug your jumper cable into the blue negative column, three rows down.
4. Now add a 330 ohm resistor by pushing one of its legs into E5 and the other leg into a hole in row five of the blue negative power column on the breadboard. It does not matter which way round the resistor is placed.

Remember that LEDs can only pass current in one direction. For the LED to work, you must make sure that you place the longer leg into the same row (D10) as the jumper cable connecting to GPIO 24. The short leg must be placed into a hole on the same row as the resistor (D5). Refer to Figure 8-8 for guidance.

## Running LEDblink.py as the Super User root

The program will not run unless you run it as the super user `root` on the Raspberry Pi. In previous adventures, when you created Python programs you selected File→Run

Module in the Python shell screen. If you do this for the `LEDblink` program, you will get an error message explaining that you do not have the correct permissions to run this program. The hardware of the Raspberry Pi is only accessible to the super user (root) and not the user account you logged in as. Using `sudo` temporarily gives you super user privileges so that you do not need to log out and back in to run a program. Instead, you need to open an LXTerminal window and type the following command to move into the folder or directory where you have saved your Python program:

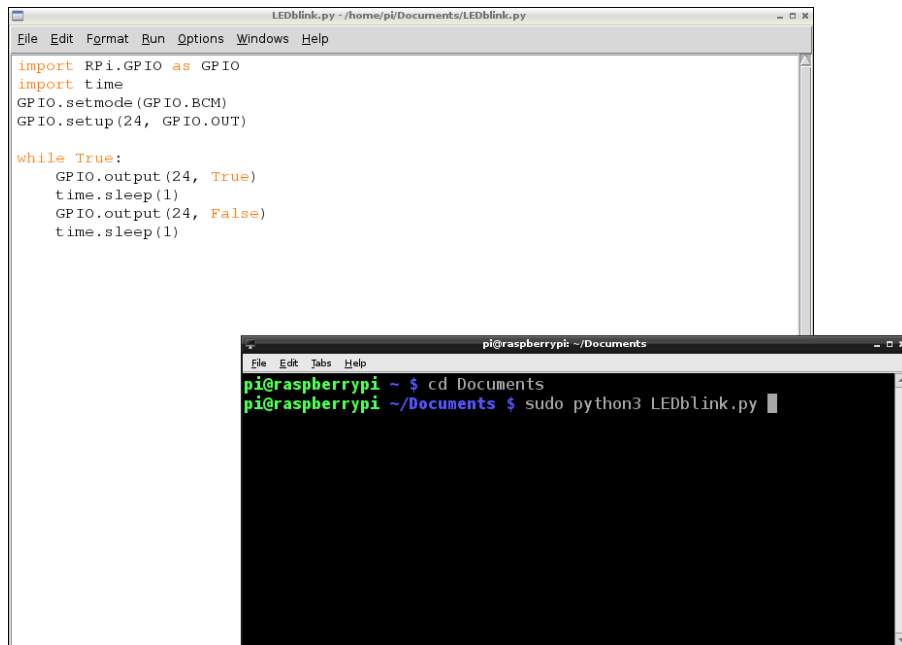
```
cd Documents
```

Then type the following line to run the program:

```
sudo python3 LEDblink.py
```

Is that cool or what!

To interrupt the program and return to the command line in LXTerminal, press CTRL + C on the keyboard. Figure 8-9 shows the code.



**FIGURE 8-9** Programming in Python 3 on Raspberry Pi to make an LED blink

# Using a Button to Turn on an LED

So far, you have controlled an LED, which is an output device. In this exercise you will add an input device in the form of a button that will start the light sequence when pushed.

For this project, you need your Raspberry Pi plus the following items:

- A breadboard
- Six jumper cables
- A simple push button
- An LED
- A 330 ohm resistor to protect the LED
- A 10k ohm resistor for use with the push button



For a video that walks you through the LEDbutton project, select the buttonLED video from the companion website at [www.wiley.com/go/adventuresinrp](http://www.wiley.com/go/adventuresinrp).

## Creating the buttonLED Python Code

In Python IDLE 3, amend your `LEDBlink` program to include the following lines (highlighted in bold):

```
import RPi.GPIO as GPIO
GPIO.setmode(GPIO.BCM)
import time

GPIO.setup(23, GPIO.OUT)
GPIO.setup(24, GPIO.IN)
```

### CHALLENGE

Why did you add `time.sleep(0.1)` to the `while` loop in the `buttonLED.py` program? What might happen if you remove it? Have a go to find out!

As before, you will set a GPIO for output (GPIO 23). You also need to set a GPIO pin to detect *input*. Use GPIO 24 for this purpose.

```
while True:
    if GPIO.input(24):
        GPIO.output(23, True)
    else:
        GPIO.output(23, False)
    time.sleep(0.1)
```

In the previous project, you used a `while True` loop to repeatedly turn an LED on and off with a one-second interval. In this project, you only want the LED to turn on when the button is pushed; therefore, you need to introduce a condition. You use `if` to set the condition: *if the button (GPIO.input(24)) is pressed, then turn the LED (GPIO.output(23)) on (or true)*. But this is only one part of the condition. You also need to set the conditions for the button not being pushed; what should the LED do then? For this part of the condition, you use `else`: *else the LED should be off (or false)*.

Save the file as `buttonLED.py` in `Documents`.

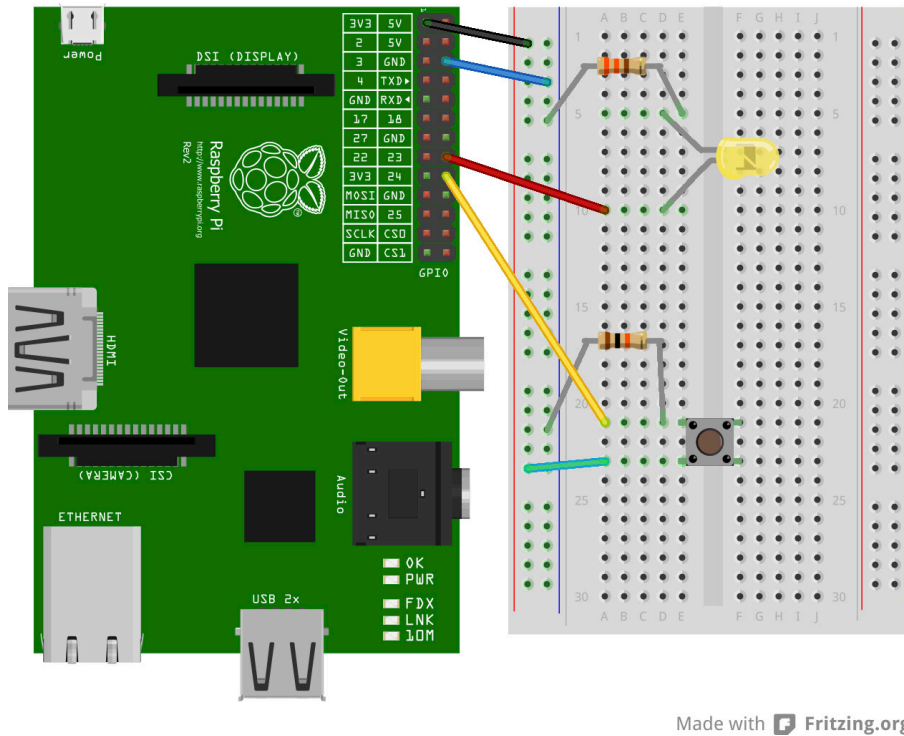
## Connecting the buttonLED Components

As with the program for making an LED blink, before you run your button LED program you need to assemble the electronic components and connect them to the Raspberry Pi GPIO pins. If you use the component configuration from the previous exercise, you will only need to add a few extra parts. Figure 8-10 shows the Raspberry Pi on the left and a breadboard on the right. Follow this diagram to help you connect the right cables and components in the circuit.

1. Leaving the original circuit complete, take a small button switch and place it across the ride in the centre of the breadboard, with two legs in holes on column E and two legs in holes in column F on rows 21 and 23, as shown in Figure 8-10.
2. When the button is in place, take a 10k resistor and push one end into D21 next to one of the button legs. Place the other end of the resistor into a hole in the blue (negative) power column of the breadboard.
3. Next take a male-to-male jumper cable (green cable) and place one end into A23 on the same row as the other leg of the button on column E. Place the other end into the red (positive) power column of the breadboard.
4. Connect a male-to-female jumper cable (yellow cable) from A21 on the breadboard to GPIO pin 24 on the Raspberry Pi.
5. Finally, add a male-to-female jumper cable (black cable) from the red column near the top of the breadboard to the top GPIO pin 3V3, which will power the circuit.



If you have a Rev 2 Raspberry Pi, you can compare your configuration to the one shown in Figure 8-10.



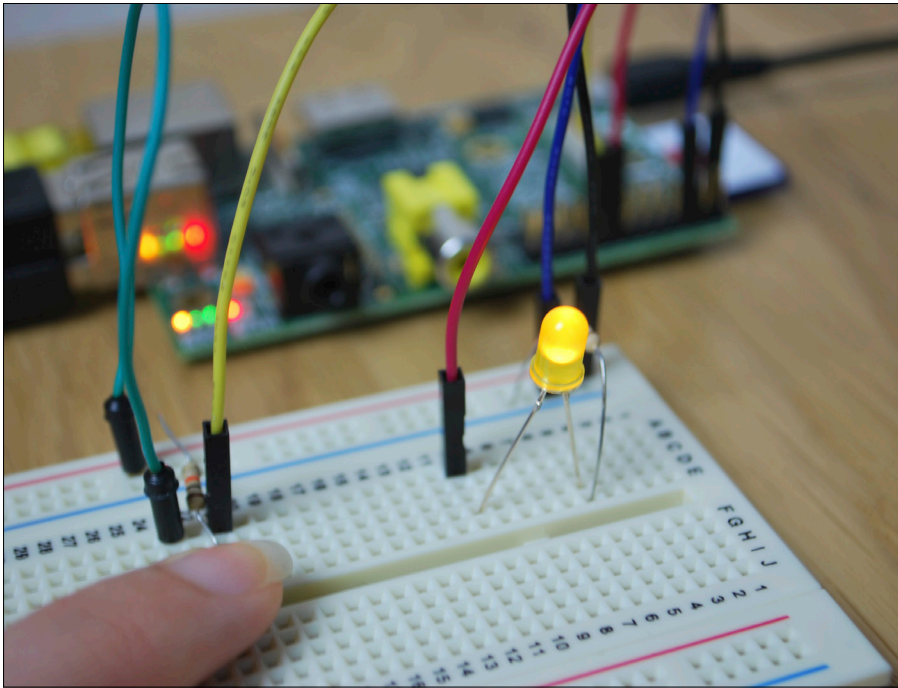
**FIGURE 8-10** Circuit diagram to connect components to Raspberry Pi for button LED

## Running `buttonLED.py` as the Super User `root`

As before, your program will not run unless you are logged in as the super user `root` on the Raspberry Pi. Using the LXTerminal window, run your program by typing this command:

```
sudo python3 buttonLED.py
```

When you press the button, the LED should light up, as shown in Figure 8-11. The LED should not light up until you press the button on your circuit. If the light comes on before you press the button or does not light when you press the button, go back and check your wiring using the diagrams.



**FIGURE 8-11** Pressing the button makes the LED light up!

## CHALLENGE

Why not improve your code so that when you press the button once it turns on the LED, and when you press the button again it turns off the LED?

## The Marshmallow Challenge

Using the Raspberry Pi GPIO pins for electronic projects can be a great way to learn. However, you can have some fun too. In this project, you will use your new GPIO pin Python programming knowledge to use a marshmallow as an input button, map it to a keyboard letter and use it to control a game that you have created in Scratch! Figure 8-12 shows the game in action.

You will need these items:

- Two female-to-female jumper cables
- Some marshmallows (yes, real ones!)
- Two metal pins or metal paper clips
- Your Raspberry Leaf to help you identify the GPIO pins that you need to use