# The RaspiRobot



## 11. The RaspiRobot

**In this** chapter, you will learn how to use the Raspberry Pi as the brain for a simple robot rover, shown in Figure 11-1. The Pi will take commands from a wireless USB keyboard and control the power to motors attached to a robot chassis kit. The robot will also (optionally) have an ultrasonic range finder that tells it how far away obstacles are as well as an LCD screen that displays information from the range finder.

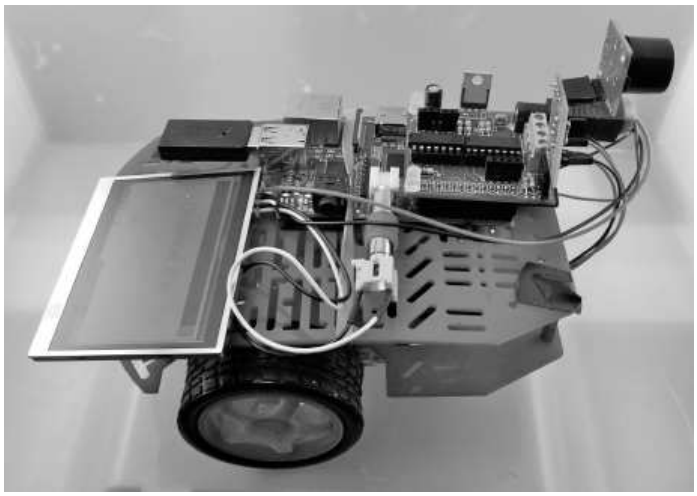



*Figure 11-1. The RaspiRobot*

Like the project in the previous chapter, this project is split into two phases. In the first phase, we create a basic rover that you can drive with a wireless keyboard; in the second phase, we add the screen and range finder.

***WARNING*** *If batteries are attached to the RaspiRobotBoard, they will supply power to the Raspberry Pi. Do not, under any circumstances, power your Raspberry Pi from its power adaptor* and *the RaspiRobotBoard at the same time. You can leave the RaspiRobotBoard attached to your Raspberry Pi, but do not attach the motors or batteries to it.*



### 11.1. What You Need

To build this project, you need the following parts. Suggested part suppliers are listed, but you can find other suppliers on the Internet.

| Part | Suppliers | Guide Price (in U.S. Dollars) |
|---|---|---|
| Raspberry Pi | Farnell, RS Components, CPC, Newark | $35 |
| RaspiRobotBoard | www.raspirobot.com | $TBA |
| Range finder serial adapter * | www.raspirobot.com | $5 |
| Maxbotix LV-EZ1 serial range finder * | SparkFun (SEN-00639), Adafruit (Product 172) | $25 |
| 3.5-inch LCD screen * | Adafruit (Product 913) | $45 |
| Male-to-male RCA adaptor for screen * | Adafruit (Product 951) | $2 |
| **** Phase 2 only.*** | | |
| ***+ This is a different battery box design than the one I used. It terminates in a 2.1 mm power plug, so the PP3 battery clip is not required if you use this battery box. If you only intend to build the first phase, and you are using the Adafruit battery box, you do not need either the 2.1 mm power-to-screw terminal adaptor (male) or the PP3-style battery clip.*** | | |

| Part | Suppliers | Guide Price (in U.S. Dollars) |
|---|---|---|
| Magician Chassis | SparkFun (ROB-10825) | $15 |
| 2.1mm power-to-screw terminal adaptor (male) + | Adafruit (Product 369), SparkFun (PRT-10288) | $2 |
| Six AA battery holder | Adafruit (Product 248), + Newark (63J6606), Maplins (HQ01B) | $5 |
| PP3-style battery clip + | RadioShack (270-324), Maplins (NE19V) | $2 |
| Six AA batteries (rechargeable or alkaline) | | |
| Wireless USB keyboard | Computer store or supermarket | $10 |
| **** Phase 2 only.*** | | |
| ***+ This is a different battery box design than the one I used. It terminates in a 2.1 mm power plug, so the PP3 battery clip is not required if you use this battery box. If you only intend to build the first phase, and you are using the Adafruit battery box, you do not need either the 2.1 mm power-to-screw terminal adaptor (male) or the PP3-style battery clip.*** | | |



## 11.2. Phase 1: A Basic Rover

Figure 11-2 shows the basic rover. The basis for this rover is the Magician Chassis kit. This useful kit is composed of a plastic chassis, gear motors, wheels, and all the nuts and bolts to assemble the chassis. It also includes a battery box for four AA batteries, but in this project, that box will be replaced by one that takes six AA batteries.

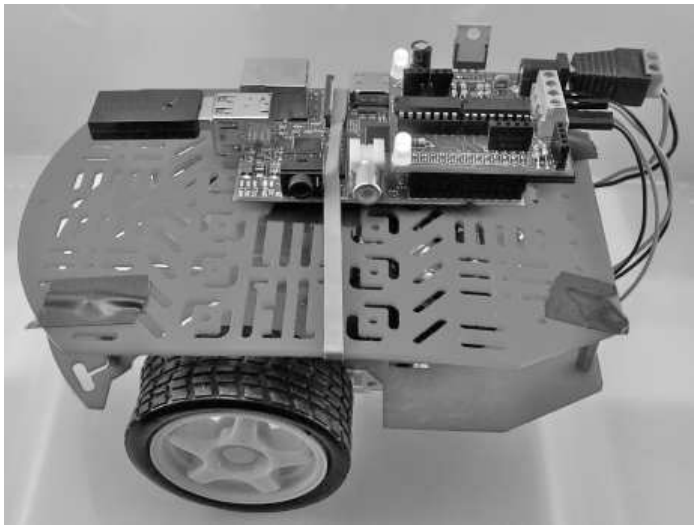



*Figure 11-2. The basic rover*



### 11.2.1. Hardware Assembly

This project is assembled from a number of different kits of parts. If you search around, you may find already-assembled options when buying the RaspiRobotBoard and the range finder serial adapter, which means the entire project can be built without any soldering (or, in fact, any tools more difficult to use than a screwdriver).



#### 11.2.1.1. Step 1: Assemble the Chassis

The Magician Chassis comes as a kit of parts that must be assembled. Included in the kit are detailed assembly instructions. When assembling the chassis, you need to replace the supplied battery box (four AA cells) with your six-AA-cell version (see Figure 11-3). If your battery box is the kind that holds two rows of three cells, you will find that the top plate of the Magician Chassis can hold it in place. In fact, it will be quite a tight fit and spring out a little; therefore, you will not need to fit the middle strut.

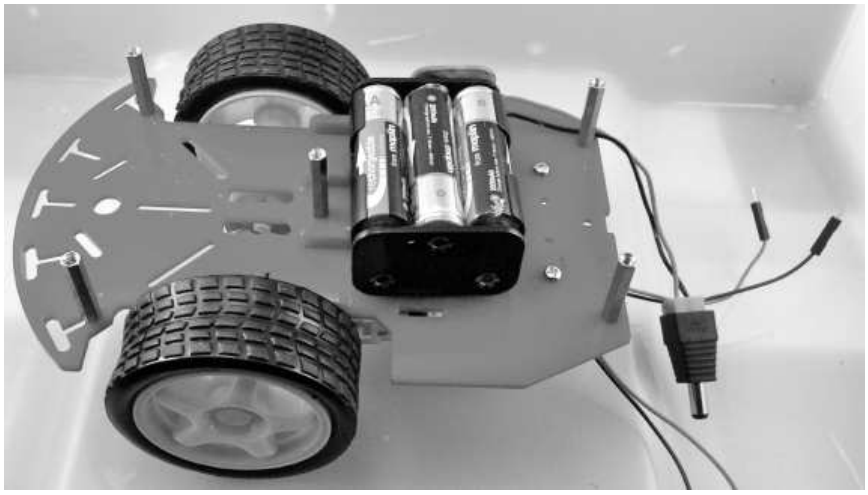



*Figure 11-3. Replacing the battery box*

If your battery box has the cells all in a single row, you will probably need to use the screws that came with the Magician Chassis for its battery box to fix your battery holder securely onto the chassis.

Attach the battery clip to the battery box and the trailing leads from the battery clip to the screw terminal in order to power plug adapter. Be very careful to get the polarity correct (red to plus)!

Before you attach the top surface of the Magician Chassis, slip a rubber band over the top surface. This will be used to hold the Raspberry Pi in place (refer to Figure 11-2).



#### 11.2.1.2. Step 2: Assemble the RaspiRobotBoard

At the time of writing, it was not clear whether the RaspiRobotBoard will be available already assembled or in kit form only. If it is available in kit form, you need to follow the instructions that accompany it to build the board. Once assembled, the board should look like the one shown in Figure 11-4.

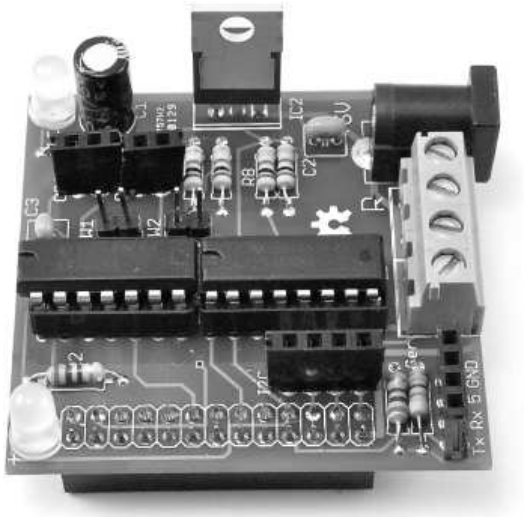




*Figure 11-4. The RaspiRobotBoard*

Note that these instructions are for Version 1 of the board. The position of the connectors may change in later versions. See the book's website (`www.raspberrypibook.com`) for more information. All the connections we are interested in are on the right side of Figure 11-4. At the top is the power socket, and beneath that are the screw terminals for the left and right motors.



#### 11.2.1.3. Step 3: Install the Software on the Raspberry Pi

To control the robot, we are going to write a program in Python that detects key presses and use them to control the power to the robot's motors on the robot. To do this, we will use pygame, which provides a useful way of finding out whether or not keys are pressed.

It will be much easier to set the program up before we attach the Raspberry Pi to the chassis. Therefore, attach the RaspiRobotBoard to the Raspberry Pi, but leave the motors and battery disconnected and power up the Pi from your normal USB power supply.

The RaspiRobotBoard has its own Python library, but also relies on some other libraries that must be installed. First of all, it requires the `RPi.GPIO` library that you first met in Chapter 5 and then again in Chapter 10. If you have not already done so, install the `RPi.GPIO` library. You will also need to install the `PySerial` library. See the instructions for this in the Arduino section toward the end of Chapter 9.

The RaspiRobotBoard library can be installed from the following website: `http://code.google.com/p/raspirobotboard/downloads/list`

Installation is the same as for any other Python package. Because we are using Python 2 in this project, the library should be installed using the command on the following page.

```
tar -xzf raspirobotboard-1.0.tar.gz

cd raspirobotboard-1.0

sudo python setup.py install
```

The actual Python program for this version of the robot is contained in the file 11_01_rover_basic.py, which must be run as super user. Therefore, just to try things out (still with the motors disconnected), run the program by changing to the "code" directory and entering the following in the terminal:

```
sudo python 11_01_rover_basic.py
```

A blank pygame window should appear and the two LEDs go out. We can test the program without the motors because the program sets the LEDs as well as controls the motors. Thus, if you press the UP ARROW key, both LEDs should light once more. Press the SPACEBAR to turn them off again. Then try the LEFT and RIGHT ARROW keys; an LED should light that corresponds to the key you pressed.

We are not going to have leads trailing from our robot to a monitor and mouse, so we need to arrange for this program to automatically start when our Raspberry Pi has finished booting up. To do this, we need to place the file raspirobot_basic.desktop (included in the "code" directory) into a directory named /home/pi/.config/autostart. You can do all this with the File Manager. Just type **/home/pi/.config** in the address bar at the top of the screen. Note that directories that start with a dot are hidden, so you cannot navigate to it in the File Manager simply by clicking.

If there is no directory inside .config called autostart, so create one and copy the file raspirobot_basic.desktop into it. We can make sure our autostart works by rebooting the Pi. If all goes well, the pygame window will appear automatically.

We will return later to look at the code for this project, but for now, let's just get everything working.



#### 11.2.1.4. STEP 4: CONNECT THE MOTORS

Shut down and disconnect the Raspberry Pi from its power supply. Be sure to put it away so that you do not accidentally apply both it and the battery connection. Put the batteries into the battery holder and fix the top plate of the chassis into place. Cover the metal screws with little patches of insulating tape or Scotch tape to prevent accidental shorts with the Raspberry Pi and then slip the Pi under the elastic band. Next, attach the motors to the terminal block, as shown in Figure 11-5.

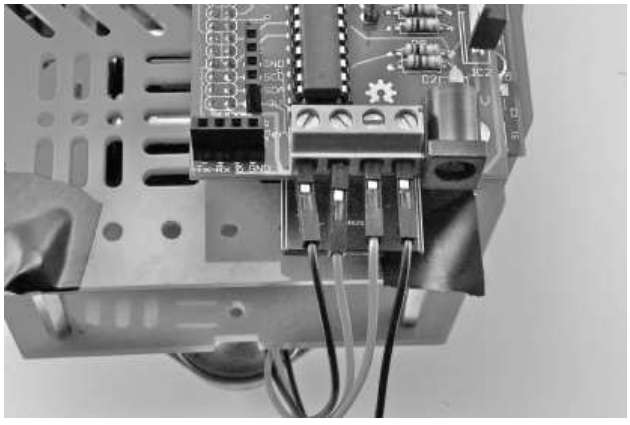



*Figure 11-5. Attaching the motors*

Each motor has a red and a black lead. Therefore, find the leads going to the left motor and attach the black lead to the leftmost terminal in Figure 11-5. Attach the red lead from the same motor to the second-from-left terminal block. For the other motor, put the red lead in the third-from-left terminal and the black lead in the remaining screw terminal.

#### 11.2.1.5. STEP 5: TRY IT OUT

That's it. We are ready to go! Attach the USB dongle from the wireless keyboard to the Pi and then attach the plug from the battery lead into the power socket on the RaspiRobotBoard. The LEDs on the Raspberry Pi should flicker as it starts to boot. If this does not happen, immediately disconnect the battery and check your work.

Initially the LEDs on the RaspiRobotBoard should both be lit, but when the Python program starts to run, they should both turn off. Wait another second or two to allow the program to start up properly and then try pressing the ARROW and SPACEBAR keys on your keyboard. Your RaspiRobot should start to move!



### 11.2.2. About the Software

The software for the first phase is listed here:

```
from raspirobotboard import *

import pygame

import sys

from pygame.locals import *

rr = RaspiRobot()

pygame.init()

screen = pygame.display.set_mode((640, 480))

pygame.display.set_caption('RaspiRobot')

pygame.mouse.set_visible(0)

while True:

       for event in pygame.event.get():

              if event.type == QUIT:

                     sys.exit()

              if event.type == KEYDOWN:

                     if event.key == K_UP:

                            rr.forward()

                            rr.set_led1(True)

                            rr.set_led2(True)

                     elif event.key == K_DOWN:

                            rr.set_led1(True)

                            rr.set_led2(True)

                            rr.reverse()

                     elif event.key == K_RIGHT:

                            rr.set_led1(False)

                            rr.set_led2(True)
```

```
                rr.right()
            elif event.key == K_LEFT:
                rr.set_led1(True)
                rr.set_led2(False)
                rr.left()
            elif event.key == K_SPACE:
                rr.stop()
                rr.set_led1(False)
                rr.set_led2(False)
```

***NOTE*** *If you skipped Chapter 8 on pygame, now might be a good time to read through it.*

The program starts by importing the library modules it needs. It then creates an instance of the class `RaspiRobot` and assigns it to the variable `rr`. The main loop first checks for a `QUIT` event, and if it find one it exists the program. The rest of the loop is concerned with checking all of the keys and issuing the appropriate commands if a key is pressed. For example, if the UP ARROW key (`K_UP`) is pressed, the RaspiRobot is sent the command `forward`, which sets the motors to both go forward as well as sets both LEDs on.



## 11.3. Phase 2: Adding a Range Finder and Screen

When you complete Phase 2, your RaspiRobot will look like the one shown earlier in Figure 11-1. Disconnect the battery from the RaspiRobotBoard so that we can start making the necessary changes to complete this phase.



### 11.3.1. Step 1: Assemble the Range Finder Serial Adapter

The serial range finder module, shown in Figure 11-6, outputs an inverted signal; therefore, a tiny board with a single transistor and resistor on it must be used to invert the signal back to normal. Full instructions for assembling this little adaptor board can be found on the book's website (`www.raspberrypibook.com`).

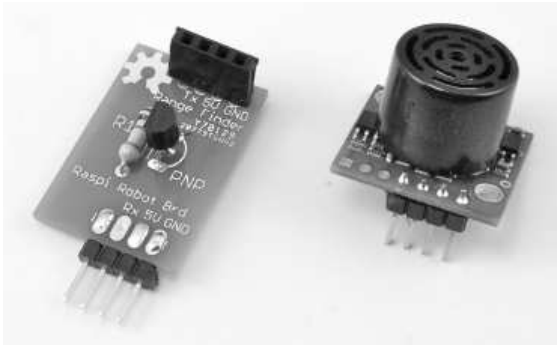




*Figure 11-6. The range finder serial adapter and range finder module*

The range finder module plugs into the top of the adapter, and the bottom of the adapter fits into the serial socket, as shown in Figure 11-7.

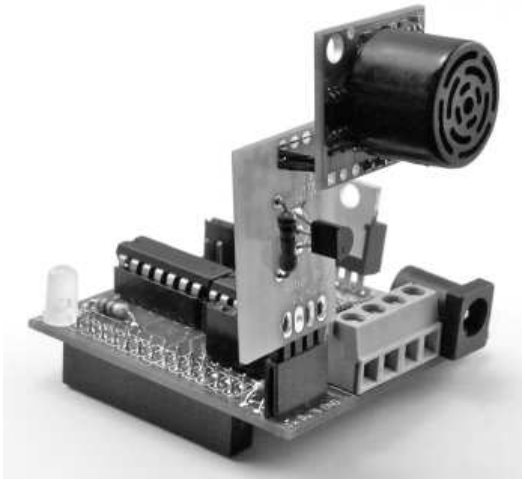



*Figure 11-7. Assembling the range finder and adapter*



**11.3.2. Step 2: Attach the Screen**

The LCD screen comes in two parts: the screen itself and the driver board. These are connected together with a rather delicate ribbon cable. I attached the two together with a cushioned self-adhesive pad. Be careful where you attach the screen, and treat the display delicately.

The screen comes with power leads (red and black) as well as two RCA plugs. To neaten things up, I cut off one of the RCA plugs (the one connected to the middle cables in the white connector plug). Refer to Figure 11-8. If this seems too drastic, an alternative is to fasten it somewhere with a cable tie so that it's out of the way.

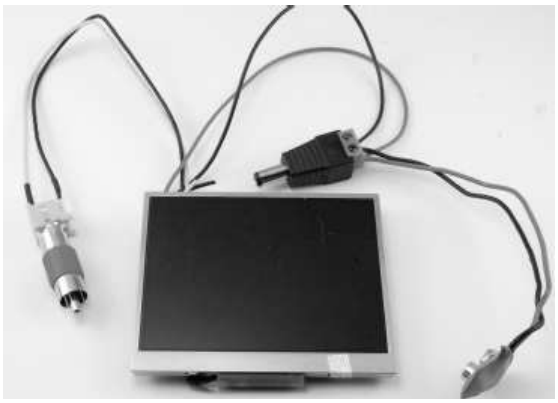



*Figure 11-8. Wiring the display*

To the remaining RCA plug I attached the male-to-male RCA adapter. The power leads are then twisted onto the power leads of the same color from the battery clip and inserted into the screw terminals of the plug adapter. If your battery box is terminated in a plug, you can snip off the plug and strip the insulation of the wires. These wires can then be used as if they were the leads from the battery clip. Either way, the project wiring is summarized in Figure 11-9.

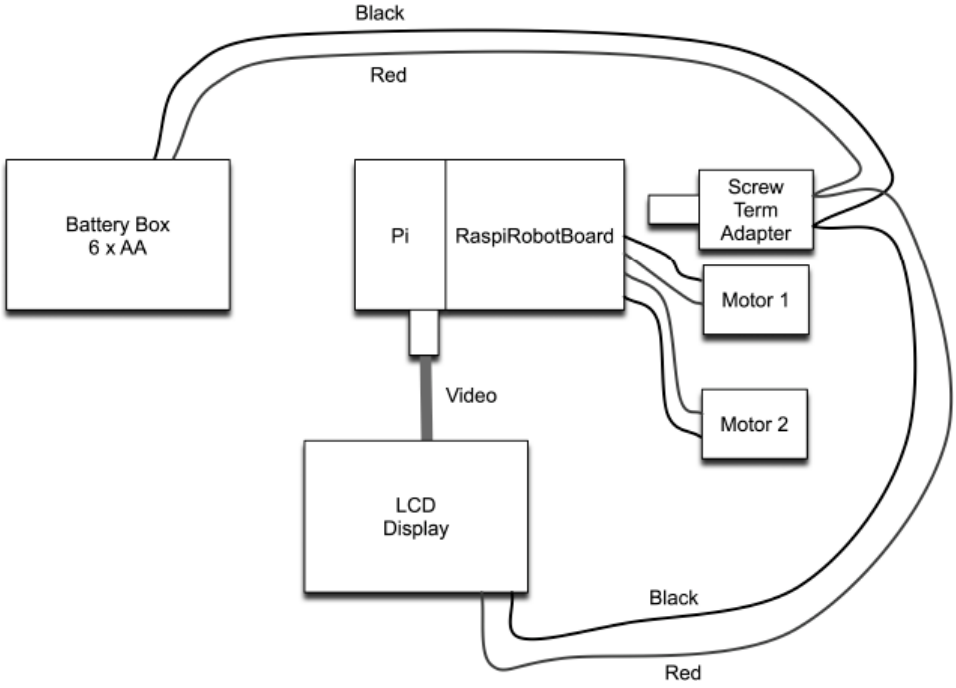




*Figure 11-9. Wiring diagram*

One consequence of this wiring arrangement is that the display will still be connected to the battery, even if you unplug the power on the RaspiRobotBoard. For this reason, use the battery clip on the battery box to turn the robot on and off.

***NOTE*** *More adventurous readers might like to add the luxury of an on/off switch.*

The screen is attached to the chassis by means of adhesive putty. This is not a good permanent solution; some kind of plastic bracket might be better.



#### 11.3.3. Step 3: Update the Software

The updated hardware needs an update to the software to accompany it. The program can be found in the file 11_02_rover_plus.py. You also need to arrange for this program to start rather than the simpler old program. To do this, copy the file raspirobot_plus.desktop into the directory /home/pi/.config/autostart and remove the raspirobot_basic.desktop file from that folder; otherwise, both programs would start.

Note that because in this phase of the project, the Raspberry Pi has a screen and a keyboard (albeit a very small one), it is possible to make the changes described here, but you'll be using a tiny screen. If this proves too difficult, then as before, disconnect the battery, detach the motors, and power the Raspberry Pi from its USB power supply with its regular monitor, keyboard, and mouse.



#### 11.3.4. Step 4: Run It

That's it! The project is ready to run. As always, if the LEDs on the Raspberry Pi don't come on straight away, disconnect the batteries and look for the problem. The Raspberry Pi is pretty power hungry for a battery-powered device. The screen also uses quite a lot of power. Therefore, to avoid too much recharging of batteries, you should disconnect them when not in use.



#### 11.3.5. Revised Software

The new program is bigger than the old one, so it is not listed here in full. You can open it up in IDLE to take a look. The main differences are, as you would expect, the distance sensing and the display. The function `get_range` is shown here:

```
def get_range():
        try:
                dist = rr.get_range_inch()
        except:
                dist = 0
```

```
    return dist
```

This function is a very thin wrapper around a call to `get_range_inch` in the RaspiRobot module. The exception handling is added because if the range finder does not work for any reason (say, it isn't plugged in), exceptions will be raised. This function just intercepts any such exceptions and returns a distance of `0` if that happens.

The `update_display` function first gets the distance and then displays it along with a graphical indication of the closeness of any obstacles, as shown in Figure 11-10.

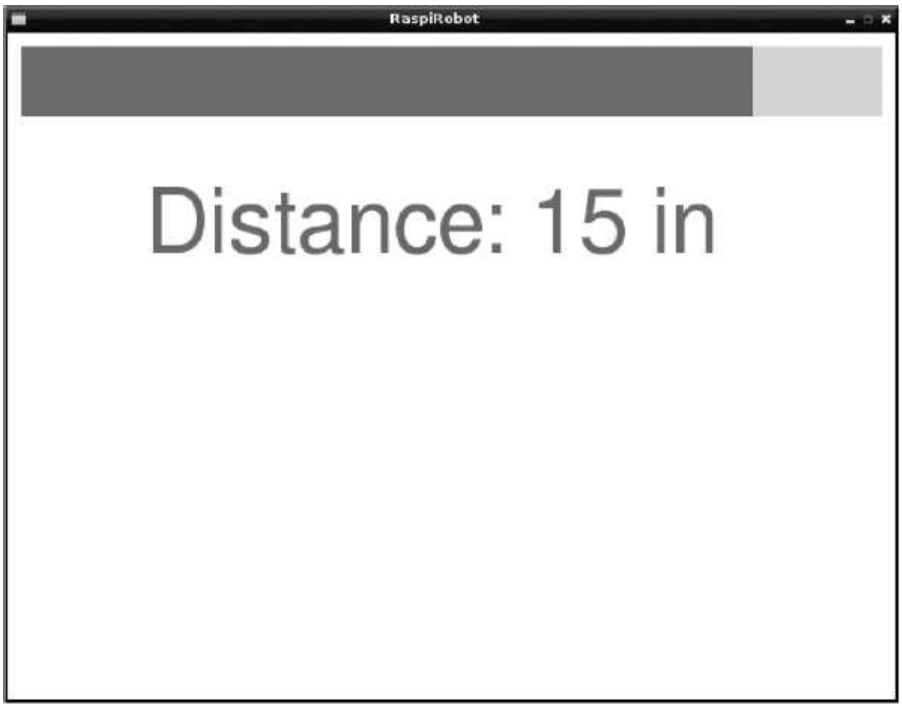




*Figure 11-10. The RaspiRobot display*

The code for this is shown here:

```
def update_distance():
    dist = get_range()
    if dist == 0:
        return
    message = 'Distance: ' + str(dist) + ' in'
    text_surface = font.render(message, True, (127, 127, 127))
    screen.fill((255, 255, 255))
    screen.blit(text_surface, (100, 100))
    w = screen.get_width() - 20
    proximity = ((100 - dist) / 100.0) * w
    if proximity < 0:
        proximity = 0
    pygame.draw.rect(screen, (0, 255, 0), Rect((10, 10),(w, 50)))
    pygame.draw.rect(screen, (255, 0, 0), Rect((10, 10),(proximity, 50)))
    pygame.display.update()
```

The distance is measured and a message is constructed into a surface that is then blitted onto the display. The graphical representation is created by drawing a fixed-size green rectangle and then drawing a red rectangle on top of it whose width depends on the distance sensed by the range finder.



### 11.4. Summary

This project can be treated as the basis for your own robot projects. The RaspiRobotBoard has two extra outputs that could be used to drive a buzzer or control other electronics. Another interesting way of extending the project would be to write software that allows the robot to spin on the spot, and use the range finder to create a sonar-style chart of the room containing the robot. With a Raspberry Pi camera module and a Wi-Fi dongle, all sorts of other possibilities for tele-presence devices arise!

The final chapter of this book looks at where to go next with your Raspberry Pi and provides some useful pointers to other Raspberry Pi resources.