1 Introduction

The task of building a robot is unlike any other in computer science. It's a strange amalgamation of computer, electrical, and mechanical engineering. Being able to program is great (and necessary), but if you can't get your program to interact with physical items like sensors and motors, then your robot will forever be a virtual one. If your motors can't move your rover without pulling more current than your circuits can source, the rover will be immobile until you find a solution —either different motors, a different circuit, or a lighter rover. And using a reed switch to determine when your rover runs into a wall is a great idea, until you discover that the switch you bought online can't stand up to the force of a 20-pound rover hitting a wall at 10 miles per hour. You need to learn to roll with the punches, fix what breaks, and—when possible—prevent it from breaking in the first place.

Building a robot also requires knowing your limits, related to both your knowledge and your materials. I *really* wanted to put a robotic gripper hand on this rover, and chances are I will eventually, but I'm aware that it probably won't happen without different tools and better materials than those I can find at the corner hardware store. Likewise, the ion-drive engine is going to have to wait a few years; in the meantime, electric car-seat motors will have to do. And be prepared to know and accept when one of your designs is just *wrong*, and to go back and redesign something. By following along in this book, hopefully you'll be taking advantage of my making the mistakes for you; rest assured that the rover design you see in this book is by no means the original design I had in my head, though I am pretty happy with the results.

The flip side to knowing your limits, of course, is being willing to stretch those limits when you think you can, and to be ready to think of unconventional ways to do things, especially when you're a backyard tinkerer—a Maker. PVC pipe, for example, is meant to be used for plumbing. However, it also makes excellent shock-absorbing drive axles (see Chapter 7). Yes, I'm using plumbing flex-hose to cover the guts of my robotic arm, and the rover's wheels are pulled straight off a Power Wheels vehicle. Sometimes you can experience great flashes of inspiration just by wandering the aisles of your local hardware (or toy) store. Sometimes you can solve a particularly knotty problem the same way.

I like to call this robot a *rover*, as I tried to pattern it after NASA's designs. Figure 1-1 shows the general outline of the finished rover.

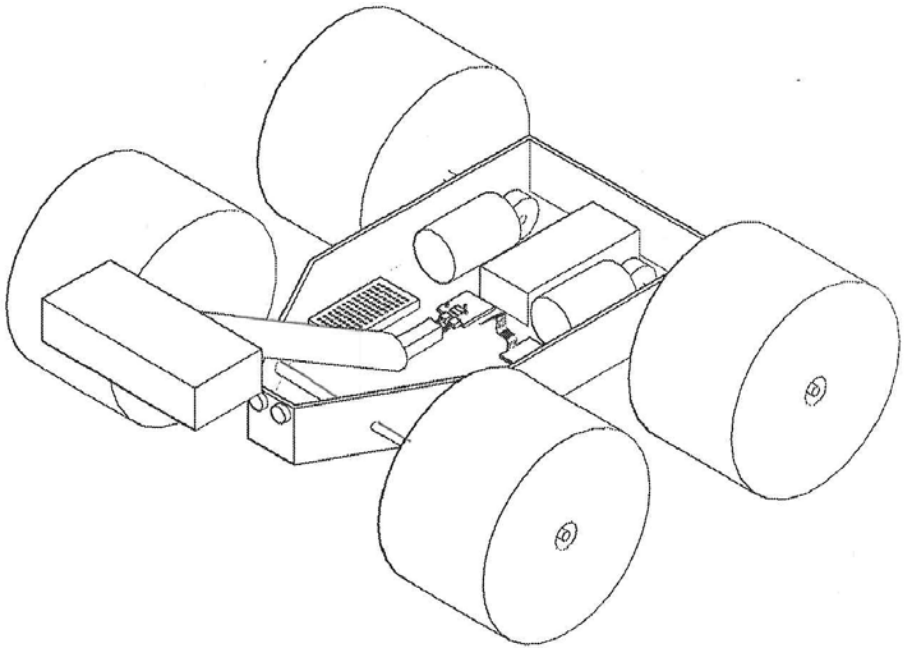

Figure 1-1. *The rover*

It's not nearly as robust as NASA's versions, of course, and you'll notice that its four (not six) wheels don't sit on their own independent shock absorbers, but the design is a proven one. And speaking of wheels: although I would very much like to program my own anthropomorphic android, such as C-3PO, it's a sad fact that the Raspberry Pi's computing power is most likely not up to the task of controlling a bipedal droid. You may think it's nothing special, but as it happens, getting a robot to not only balance on two legs, but also walk on them, is quite a challenge. The well-known ASIMO robot by Honda (Figure 1-2) required many years and many millions of dollars to finally be able to walk on its own.

To balance on two feet, a robot's internal sensors must constantly measure where the robot's center of gravity (COG) is, and then determine where the robot's feet are, and then check to see that the COG is over at least one of the robot's feet, preferably over a line between the robot's feet, or at most, very slightly offset from that line (but not too far). If the robot's COG is too far to one side, the robot's brain must send the command to flex the leg on that side to tilt the robot ever so slightly in the other direction, bringing the COG to a more stable location, without going too far in the other direction. And if the robot is carrying something, all those values need to be recomputed on the fly.

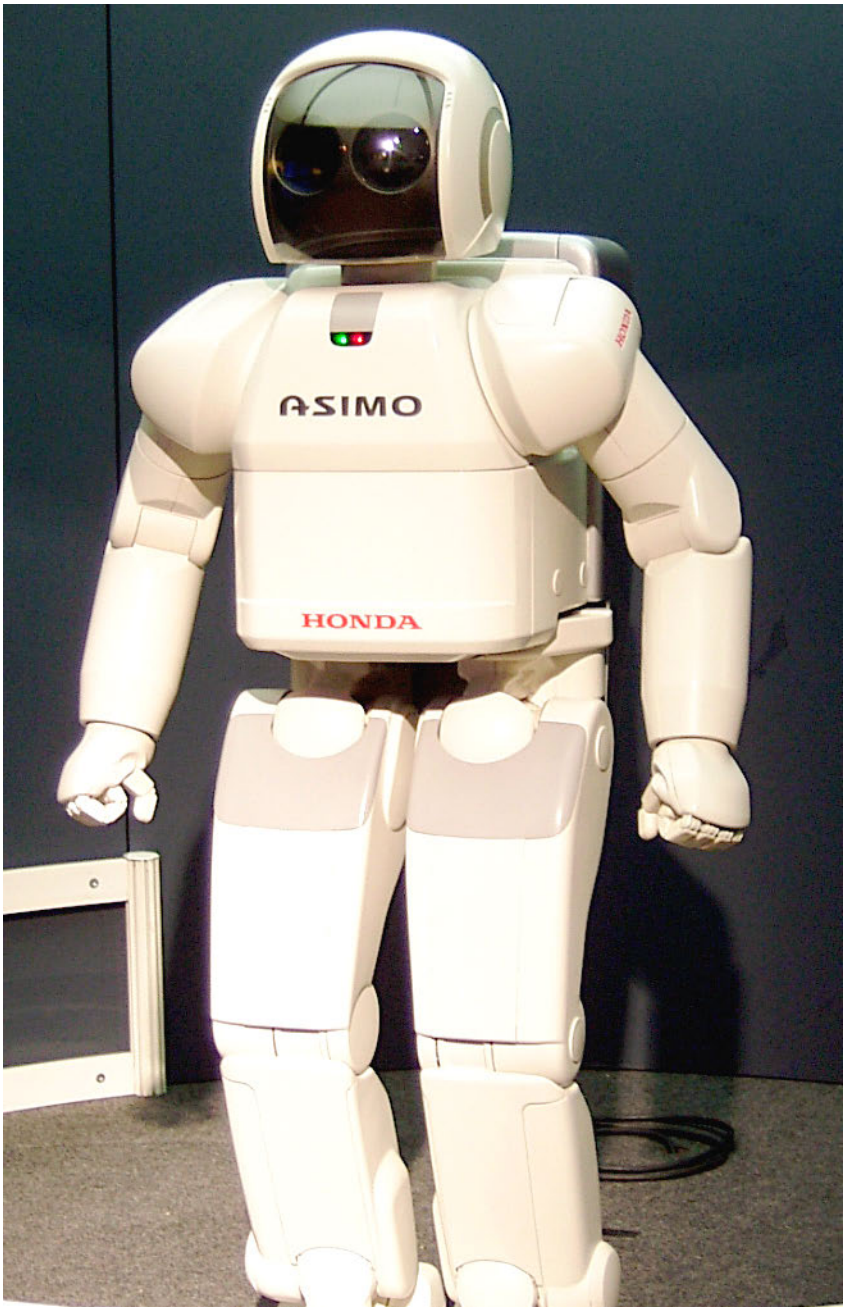

Figure 1-2. *More than the Pi can handle*

So there are several advantages to using wheels. First, not having to balance means that the Pi's computing power (and servo power) can be spared for other tasks, such as taking temper-

ature samples or moving the robot arm. Second, depending on the type of wheels you use, a wheeled vehicle can go all sorts of places that a bipedal robot can't. And third, wheels can also be cool—I refer you to R2-D2, the Mars Curiosity rover, and the Mars Exploration rovers (Spirit and Opportunity) for examples of pretty cool wheeled robots. Figure 1-3 shows the Mars rovers.

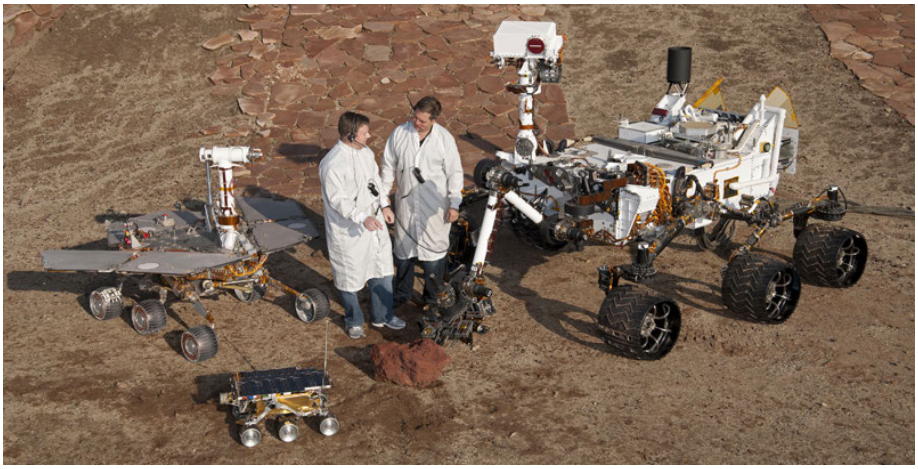

Figure 1-3. *Three bad-assed wheeled robots*

To increase the coolness factor to monster-truck levels, I decided to go with oversized wheels; it's common knowledge that almost any wheeled vehicle looks seven and a half times better with bigger tires. Figures 1-4 and 1-5 prove my point.

Figure 1-4. *Small tires: not so awesome*

Figure 1-5. *Big tires: AWESOME!*

This brings up more design challenges, however. Larger wheels tend to be heavier, and it's always—*always*—a good idea to keep your robot or rover as light as possible. A heavy robot is a power-hungry robot, and batteries and engines are heavy enough to begin with. Large wheels also have greater rolling resistance, though rolling resistance comes more into play at higher speeds and higher efficiencies than this rover is likely to experience. My solution: I used the wheels from a Power Wheels vehicle. They're large and impressive, but because they're made of plastic, they hardly weigh anything. Of course, that led to further challenges, such as mounting those wheels to a non–Power Wheels axle, but as you'll see in Chapter 7, those issues were solved as well, often with a combination of screws, nuts, bolts, and generous applications of epoxy and cold-weld.

The final design, assuming you follow these step-by-step instructions, can be seen in Figure 1-6.