1 Introduction

This chapter will get you started building and designing prototypes for embedded systems. You will learn basic principles that you'll follow in Chapters 3 and 4 as you build the Stalker Guard and Robot Insect. Prototypes in this book are just the beginning. Once you know the techniques, you'll be able to build prototypes for your own inventions.

IN THIS CHAPTER

Building Philosophy

When you break a programming problem down into smaller pieces, be sure to test and validate each piece as you go. If you don't do this, you could find yourself wildly off track by the time you've gotten through a few pieces.

Prototype

This book provides techniques for building *prototypes*, or test versions of a device. A prototype such as the one shown in Figure 1-1 provides a proof of concept—a concrete realization of a device's intended functions.

Try to finish a functional prototype as quickly as possible. Once you've documented a working prototype, you can build in improvements in later versions.

You can make a working end result by stripping out unnecessary functions and taking shortcuts. If it makes testing quicker, use rubber bands and duct tape when you have to. Don't try to optimize your code in the first version.

It's much easier to build an impressive version once the first prototype is finished. Usually, you'll find that many challenging problems you face in the prototype don't even need to be solved for the final version. In the same way, building a prototype can reveal new opportunities for development.

Figure 1-1. *Jari Suominen testing a prototype made of Legos*

Having a prototype can also help you secure funding for your project. Who would you believe more: someone who talks about a walking robot, or someone who has actually built one?

Start with Hello World

How does Arduino say "hello" to the world? By blinking an LED. You'll learn more in Chapter 2.

Starting a project with Hello World is usually a good idea, because it's the simplest possible program. Typically, Hello World will print a row of text to a computer screen or blink an LED. It is used for testing to make sure the development environment works.

If your next, more complicated iteration doesn't work, you can search for the cause of the problem within the added code. Hello World lets you know that the microcontroller, development environment, interpreter, and USB port all function correctly.

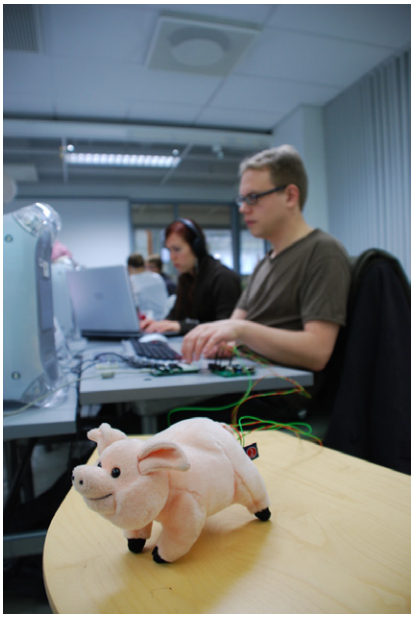

Figure 1-2. *Juho Jouhtimäki and Elise Liikala building a motion-sensitive soft toy*

Build in Small Steps

Complex problems (see Figure 1-2, Figure 1-3, and Figure 1-4) are hard to solve, but you can usually make them easier by breaking them down into smaller pieces. You can then solve the problem one manageable piece at a time.

A student of ours once built a burglar alarm after studying embedded systems for a week. The alarm buzzed whenever an infrared sensor detected movement. Users could log into the system wirelessly by presenting an ID in the form of a keychain. Once the system approved the login, the user could then move freely in the space without triggering an alarm.

Figure 1-3. *Welding a robot hand*

Figure 1-4. *Jari Suominen's strobo owl, which uses aesthetics from printed circuit boards and components*

A project like this can sound quite complex to a novice, but it really consists of three clearly separate components (motion detector, buzzer, RFID reader). First, the student programmed and tested the motion sensor. That section was finished when the program could detect movement and sound the alarm.

The three components of the system do not affect one another in any way, and the only unifying factor is the code. Program code can check with the motion detector to determine whether movement is present and, if so, it can switch on the buzzer.

Test in Steps

"I wrote the code for a singing and dancing robot that can walk up stairs. The code is 30,000 lines long. I just tried compiling it, but it doesn't work. Do you have any advice?"

Conduct testing as early as possible. If, for example, you build a walking robot, the first thing to test is whether you can make the servo motor move. The next test can make the servo move back and forth.

After you have tested the functionality of a specific version of code, save it separately from the version you are working on.

Revert to the Last Known Good Version

When you have developed your code into a confusing and nonfunctional state, the solution is easy. Go back to the last working version.

More specifically, go back to a working stage when the situation was already becoming confusing. This method removes the problem areas and lets you start over with a functional clean slate, helping you isolate what went wrong.

Read the Friendly Manual

RTFM is an old Internet acronym. (Actually, the *F* is not always *friendly*, so we usually stick with just RTM.) The point of the expression is that most answers are out there, written in a manual. When you're surrounded by parts (see Figure 1-5), you're going to need answers.

Friends and students sometimes wonder how we know so much. How do we know the Arduino operating voltage or the way to install SSL encryption to the Apache web server?

Figure 1-5. *Mikko Toivonen, surrounded by robots and microcontrollers*

The answer is easy. You can find instructions for almost anything if you know where to look.

Instructions don't always come with devices and parts, but you can often find them on manufacturer's web pages (such as *http://www.parallax.com*) or by searching in Google. Good search terms include device names (e.g., "ping ultrasonic sensor") or a sequence of numbers on a circuit board (e.g., "H48C").

Figure 1-6. *Jenna Sutela and David Szauder demoing functions of a wearable prototype*

You could also combine a search sequence with a technology—for example, "H48C arduino." Some web pages are devoted specifically to Arduino—for example, *http://arduino.cc* and our site, *http://BotBook.com*.

Document

Most things appear easy once you know them. The details of a project seem obvious on the day you complete them ("*of course* I remember when I programmed the 16-servo walker"). But a week after building, coding details begin to disappear from your memory. After a year or so, it can be hard for someone who builds many projects to remember anything about a specific one.

For this reason, it is worthwhile to document all projects. Typing notes avoids the potential problem of illegible handwriting, and shooting stages with a digital camera provides an accurate visual snapshot of each stage.

You might also consider publishing your results on the Web. Some projects that would otherwise be collecting dust in your drawer might actually be useful to others. You might even find your own instructions (long since forgotten) when looking to solve a new problem with similar logic. Two sites where you can publish projects are Make: Projects (*http://www.makeprojects.com*) and Instructables (*http://www.instructables.com*).

Reusing Parts

Not every device is safe to salvage: for example, a CRT (Cathode Ray Tube) TV retains a hazardous voltage for a long time after you unplug it from the wall.

Prototype mechanics (see Figure 1-6) need all kinds of parts, such as frames, limbs, and joints. Finding appropriate materials can seem daunting. Customizing more complicated parts using homebrew methods isn't always easy, and even basic materials—such as lightweight and sturdy metal plates—can be significantly expensive at hardware stores.

As a starting point, we recommend using recycled parts. Old devices are filled with usable materials, so remove all salvageable parts before you throw them away.

One additional perk that comes with using recycled parts is a unique aesthetic. Old parts often have interesting shapes, curves, and worn areas (Figure 1-7).

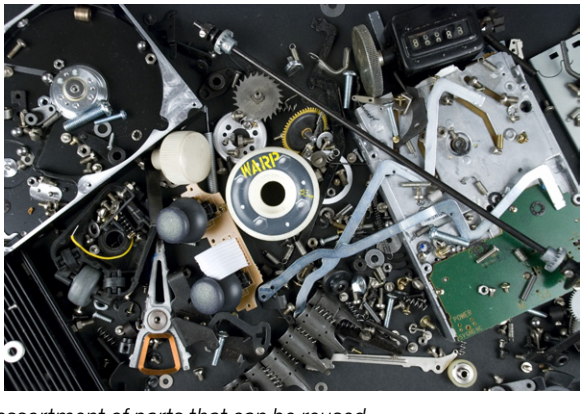

Figure 1-7. *An assortment of parts that can be reused*

Computer DVD drives and hard drives can make great frames for robots, because their covers are often made of lightweight, easily drillable, and sturdy material. You can also remove DC (direct current) motors and gears from DVD drives. Nowadays, there is more readily available computer junk than you can gather and store in your home. Educational institutions and corporations are particularly good sources, as they're continuously throwing out old devices.

Flea markets can also hold great finds. Mechanical typewriters deserve a special mention here. Though they are relatively hard to disassemble, they house an unbelievable amount of small springs, metal pieces of different shapes, and screws.

Disassemble devices as soon as you find them and then discard or recycle unnecessary parts. This way, you'll avoid turning your home into a graveyard of retired devices, and more importantly, the parts will be immediately usable when you really need them. When you are searching for a suitable attachment piece for a servo, you probably don't want to start a six-hour disassembly operation. Parts usually won't find a new purpose until you've removed them from the original device, at which point inspiration might strike. You might even wonder how a specific "whatchamacallit" fits a new purpose so perfectly.

When you begin working on some difficult new mechanism, think about where you might have seen something similar. You'll often find everyday solutions to many problems. For example, parts purchased from bicycle or automotive shops can sometimes work in other projects. Figure 1-8 shows a hand with fingers that are moved with servo motors; every joint in each finger bends. The fingers were made by attaching sections of a steel pipe to a bicycle chain. They bend when a brake cable is pulled down. Typewriter parts welded to the opposite side of the structure pull the fingers back into a straight position.

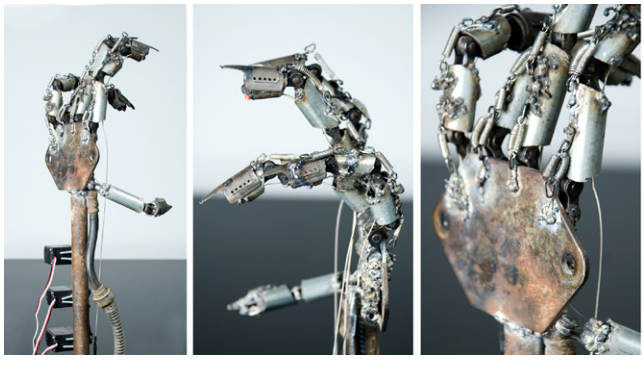

Figure 1-8. *Robot hand made of junk*

Also keep your eyes open in military surplus stores, where you can find inexpensive, sturdy, and personalized enclosures for prototypes. Various parts and accessories in these shops can also, with a bit of creativity on your part, give devices significantly more street cred. For example, Figure 1-9 shows a porcupine robot cover built from an MG/42 machine gun ammunition belt.