Chapter 1

Robots and Programs

Introduction

Without a program, a robot is just an assembly of electronic and mechanical components. This book shows you how to give it a program.

The LEGO® Mindstorms® kit contains the parts and directions for building the following four different types of robots:

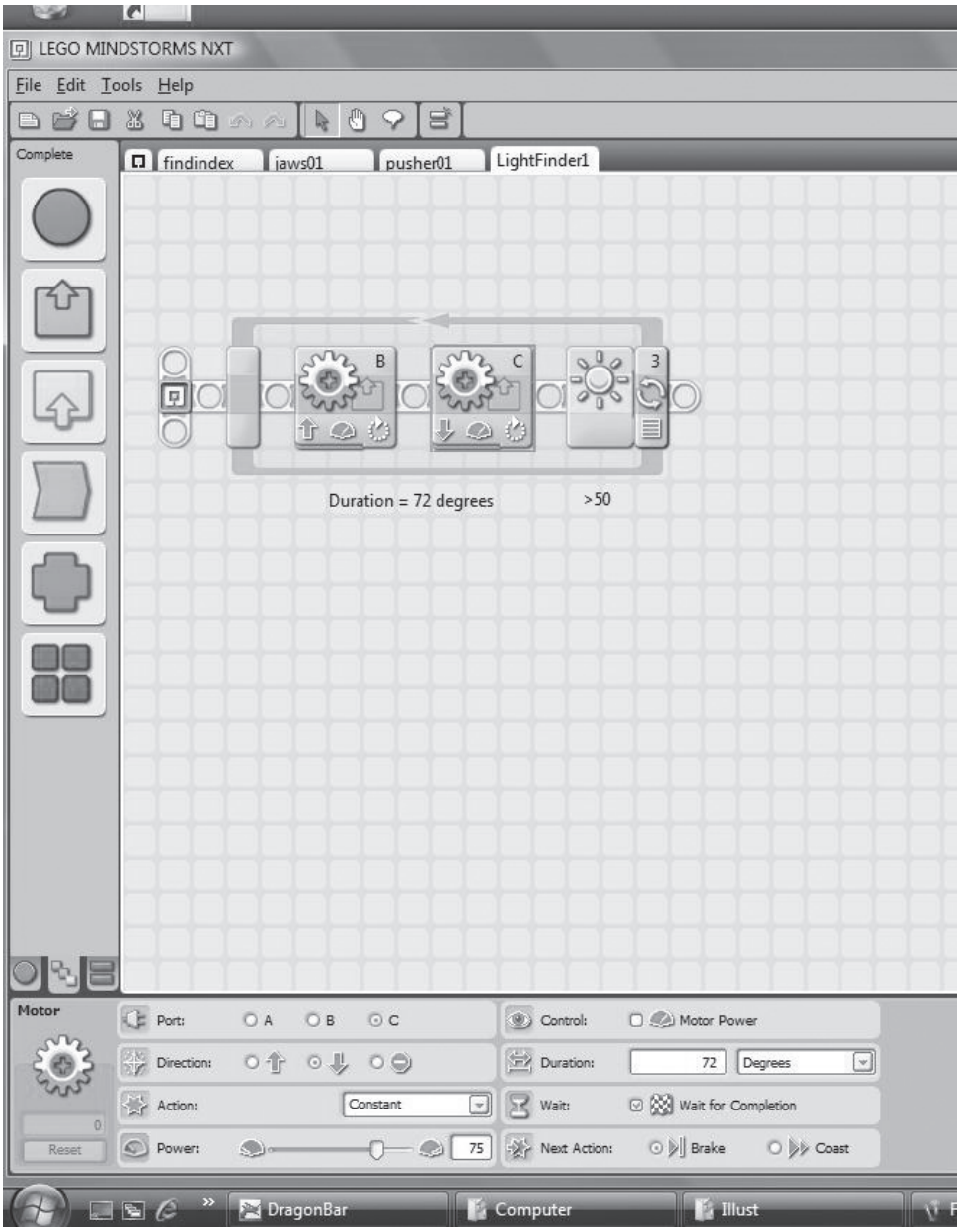
- A vehicle called a *Tribot*
- A machine called a *RoboArm*
- An animal called *Spike* the scorpion
- A humanoid *Alpha Rex*

Of these, *Alpha Rex* is closest to most people's idea of what a robot is. But *Tribot* and *RoboArm* are typical of robots commonly seen in industrial situations. It is a lot of fun building them, testing them with the few simple programs that come with the kit, and then getting them to do new and interesting things by writing your own programs.

As well as these four, there are dozens of other robots waiting to be built and programmed. For example, the monkey and the mouse in the book by Dave Astolfo, Mario Ferrari, and Guilio Ferrari, "Building Robots with LEGO Mindstorms NXT" (Syngress, 2007). There are also dozens of robots that you can invent and program using the same kit of parts!

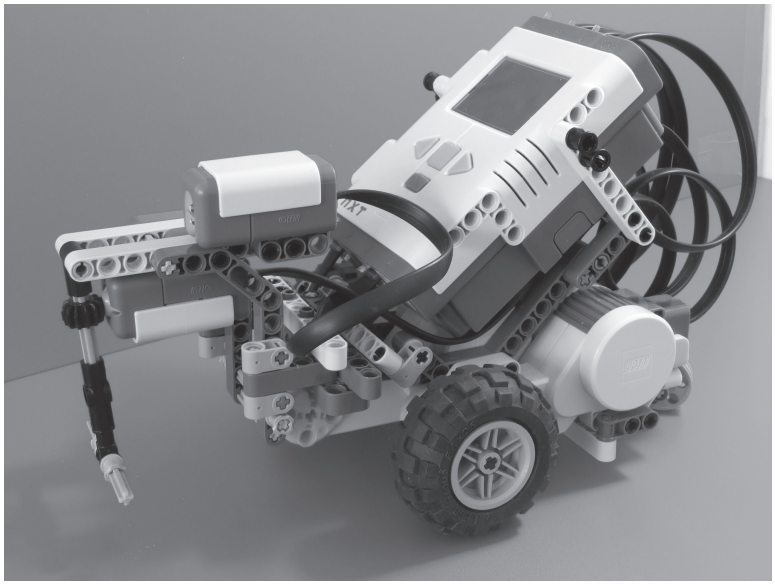
In this book, the programs and program snippets are written using the Mindstorms NXT-G language that comes with the kit. NXT-G is described as a visual language. Instead of being written in text, it is created as a diagram (see Figure 1.1). NXT-G is described in more detail in Chapter 2.

Figure 1.1 This is the computer screen during a NXT-G programming session. On the left-hand side is the palette from which you drag the icons that represent the programming blocks. The icons are assembled into a program on the grid of the working area. Below the working area is the configuration pane, which displays the detailed settings of the currently selected block.



Most of the NXT-G programs in this book are based on the *Tribot* (Figure 1.2) and the *RoboArm*, because these are the most versatile robots. Also, there are several programs specially written for *Alpha Rex*.

Figure 1.2 This version of the *Tribot* is equipped with bumper sensor, to detect objects that it runs into, and also has a light sensor pointing forward with which it can seek out light sources in the room.



What is a Program?

A program is essentially a list of instructions that tells the robot what to do and when and how to do it.

A NXT-G program does not look anything like a list (see Figure 1.3). The program is built up from icons, each icon representing one or more instructions. The icons are arranged in order on a framework. The design of each icon tells us what the icon does (see Figure 1.4). When the program is run, the icons are taken in the order in which they are on the framework, or sequence beam, and executed in that order. This is the same as having a written list of instructions and following them in the order in which they are listed.

Figure 1.3 A NXT-G program consists of a number of programming blocks, which are executed in the order in which they are placed on the sequence beam. Sometimes the blocks are linked by data wires, which transmit data from block to block. This program switches on the drive motors for 1 second. Four seconds later, it starts to switch motor A on and off every 2 seconds.

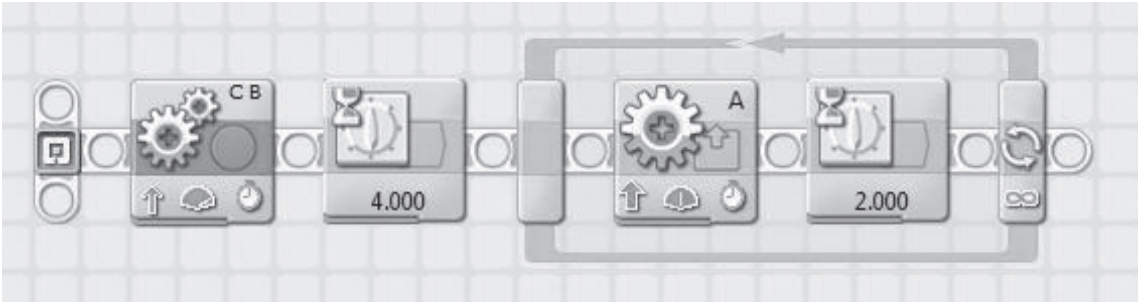
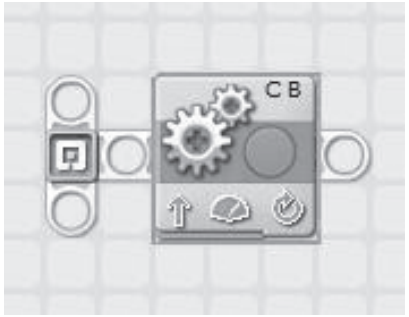
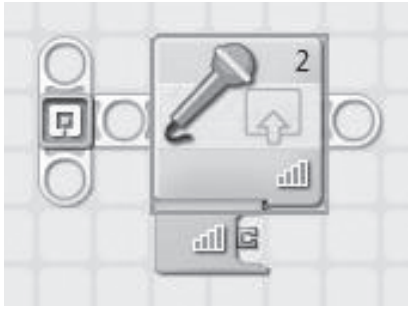


Figure 1.4 The gear wheels on the upper icon indicate that it is a move block. The lower icon is obviously a sound sensor block.





Now comes the problem. Humans can understand the icons and sequences of icons, but robots can't. Robots only understand an arrangement of electrical charges in the electronic circuits in their memories. The program in icon form has be turned into electronic form. This is where a compiler program comes in. The compiler program is run by clicking on one of the buttons at the bottom right-hand corner of the screen. When the program has been compiled, it is read and executed by the microcontrollers in the NXT brick.

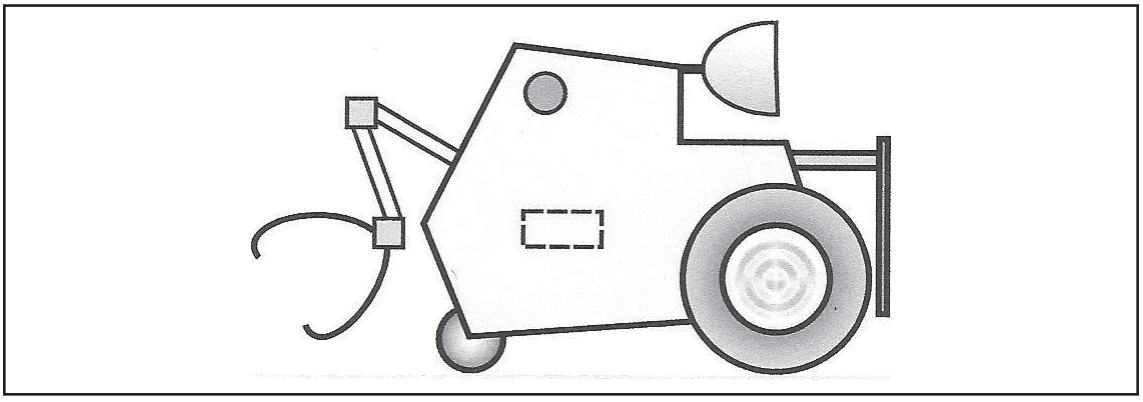
MICROCONTROLLERS

The microcontrollers are the two vital electronic devices inside the *NXT*. They read and then execute the program instructions stored in the *NXT*'s memory.

Designing a Robot

When you have finished building and programming the robots that come with the kit, you will almost certainly want to design and build robots of your own. This is when things get really exciting. When you are designing your own robots, the first question that must be asked is "What do I want it to do?". Do I want it to pick up objects? If so, it will need one or more arms with grippers (see Figure 1.5). Do I want it to move around? If so, it needs to have wheels, or legs, if its environment includes steps or rough surfaces. Does it need an ultrasonic sensor for navigation? Does it need an audio sensor for receiving spoken commands?

Figure 1.5 Equipping a robot. This robot is going to be mobile, so it needs drive wheels with motors. The rear wheel is a caster. The robot has an ultrasonic sensor for navigation, and a bumper for detecting obstacles. It has an arm with a gripper for handling objects and a microphone for detecting spoken commands. Like all robots, it must have at least one microcontroller or microprocessor. To make the whole system operational it must have a program.



By considering these types of questions you build up a specification for the robot that is related to what it has to do and where it has to operate.

Specify the Robot's Environment

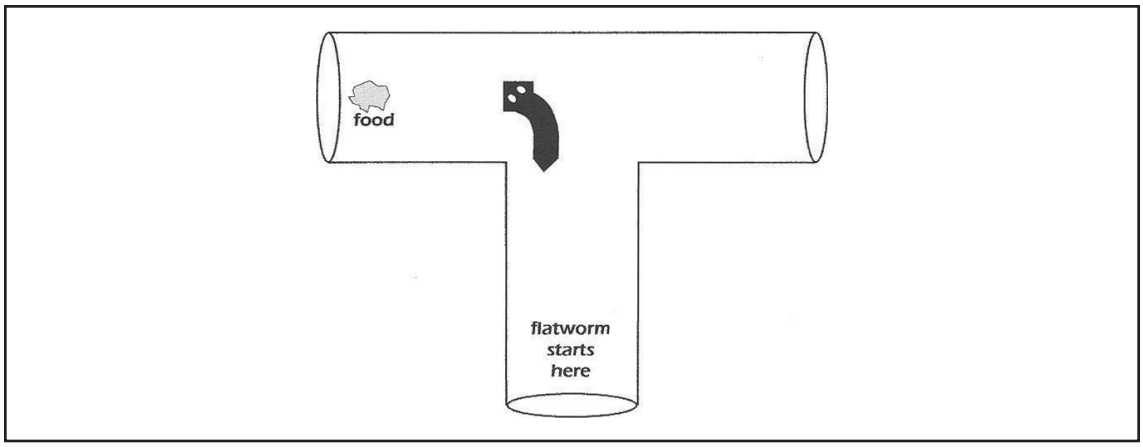
Animals (including humans) adapt to the environment in which they live. It is different with robots. Sometimes it makes things simpler if we adapt the environment to the robot instead of adapting the robot to the environment. One example is if the robot is designed to find its way to a source of light, it's a good idea to make sure there is only one source of light in the room. Later you can try to get it to distinguish between one bright source of light and other sources of light. Give it a simple environment to start with, and when the program is working properly you can try improving it to work in a more complex environment. Start with something easy and progress to something more difficult.

In the case of a robot that handles objects, it may be a good idea for the objects to be all the same size and shape. Later you can try programming the robot to handle objects of different sizes and shapes. If the robot has to move from one place to another, paint a line on the floor for it to follow. It is much easier for a robot to follow a line, rather than to navigate between the walls and furniture. There are many ways of adapting the environment to simplify the task of the robot.

Intelligent or Smart?

Most of the programs in this book give the robot a pattern of smart behavior. A robot that heads toward a bright source of light while avoiding obstacles may be smart, but can hardly be called intelligent. It is interesting to compare the *NXT* brick with the brain of a very simple animal such as the flatworm. This animal can solve a simple T-maze (see Figure 1.6) by trial and error. The experimenter put the worm at the base of the T. In the early trials the worm found its way to the food by chance, sometimes going left and sometimes going right. After the trial was repeated several times, it gradually learned that going left would take it to the food. From then on it always went to the left. This is one type of intelligent behavior. The animal learned to find where its food was placed. Robots can do the same thing. Later in this book we describe how to program a *Mindstorms* robot to solve a maze with three or more junctions. At first it operates by trial and error, but when it finds a successful pattern of behavior it remembers it and repeats it as long as it continues to be successful.

Figure 1.6 The maze for training the flatworm is made of glass and filled with water. It has only one junction. If food is always placed in the left arm of the maze, the flatworm soon learns to turn left at the junction.



Summary

If it is possible to program a robot to solve a maze better than a flatworm, it should be easy to get it to mimic the behavior of other lowly animals. Why not program the *Tribot* to behave like a bat, using an ultrasonic sensor to steer its way across an obstacle-littered floor in total darkness. One of the programs later in this book shows how a *Tribot* can be programmed to behave like a housefly, ignoring slow changes in the light intensity, but accelerating rapidly away when the shadow of a swatting hand suddenly A bit further up the intelligence scale, we see how *RoboArm* can be programmed to perform a magical illusion in the same way that a human magician does. But is this intelligent behavior or just smart? This is for the reader to decide.