Creating Cool MINDSTORMS® NXT Robots

■ ■ ■

Daniele Benedettelli

Apress®

Creating Cool MINDSTORMS® NXT Robots

Copyright © 2008 by Daniele Benedettelli

ISBN-13 (pbk): 978-1-59059-966-2

ISBN-10 (pbk): 1-59059-966-7

ISBN-13 (electronic): 978-1-4302-0599-9

ISBN-10 (electronic): 1-4302-0599-7

Printed and bound in the United States of America 9 8 7 6 5 4 3 2 1

Lead Editor: Jeffrey Pepper
Technical Reviewer: Claude Baumann
Developmental Editor: John R. Vacca
Editorial Board: Clay Andres, Steve Anglin, Ewan Buckingham, Tony Campbell, Gary Cornell, Jonathan Gennick, Matthew Moodie, Joseph Ottinger, Jeffrey Pepper, Frank Pohlmann, Ben Renow-Clarke, Dominic Shakeshaft, Matt Wade, Tom Welsh
Project Manager: Beth Christmas
Copy Editor: Susannah Davidson Pfalzer
Associate Production Director: Kari Brooks-Copony
Production Editor: Kelly Winquist
Compositor: Kinetic Publishing Services, LLC
Proofreader: April Eddy
Indexer: Becky Hornyak
Artists: April Milne, Dina Quan
Cover Designer: Kurt Krames
Manufacturing Director: Tom Debolski
Distributed to the book trade worldwide by Springer-Verlag New York, Inc., 233 Spring Street, 6th Floor, New York, NY 10013. Phone 1-800-SPRINGER, fax 201-348-4505, e-mail orders-ny@springer-sbm.com, or visit http://www.springeronline.com.

For information on translations, please contact Apress directly at 2855 Telegraph Avenue, Suite 600, Berkeley, CA 94705. Phone 510-549-5930, fax 510-549-5939, e-mail info@apress.com, or visit http://www.apress.com.

The source code for this book is available to readers at http://www.apress.com. You will need to answer questions pertaining to this book in order to successfully download the code.

To my brother Alessandro

Contents at a Glance

Contents

About the Author

DANIELE BENEDETTELLI appeared in this world on December 2, AD 1984 in Grosseto, the capital city of the beautiful Maremma Toscana. While attending high school, apart from his compulsory studies, his main passion was writing music and playing the piano, a passion to which he devoted his childhood. When he was not playing the piano, you could find him playing with LEGOs.

This last passion took a backseat during his "dark age of LEGO," when real-life interests got the better of building plastic creations. In 2000, Daniele scraped enough savings together to get the LEGO MINDSTORMS Robotics Invention System, and from that moment on, a new way of relating to LEGO began: his adult career in the LEGO community started! In 2006, he got a Bachelor of Science degree cum laude in Computer Engineering (Automation concentration) from the University of Siena with a thesis whose approximately translated title is "LEGO MINDSTORMS–based mobile robots team." A toy—a destiny, we could say. Now he's studying for a Master of Science degree in Robotics and Automation at the University of Siena.

In 2006, he was selected by The LEGO Group as member of the MINDSTORMS Developer Program (MDP), and in 2007 as one of MINDSTORMS Community Partners (MCP).

2007 was a turning point for Daniele. He gave birth to a LEGO NXT robot that can solve automatically any 3×3 Rubik's Cube in less than a minute. This robot is the mechanical part of the project called the LEGO Rubik Utopy. The world has gone crazy over this wonderful contraption. His activity with LEGO on the NXT line is continuing now with the group called the MINDSTORMS Community Partners 2.0.

About the Technical Reviewer

Since 1999, **CLAUDE BAUMANN** has taught advanced LEGO MINDSTORMS robotics in after-school classes and maintains the related widely known web site `http://www.convict.lu/Jeunes/RoboticsIntro.htm`. He participated in beta testing of the ROBOLAB software that originated at Tufts University. He also has been in charge—in collaboration with Professor Chris Rogers— of the creation of ULTIMATE ROBOLAB, a cross-compiler environment that allows graphical programming of RCX firmware, and of a unique RCX self-replicating program (also called a “virus”). Claude has been the assessor of various high-school robot projects (among which is the famous LEGO humanoid robot GASTON). He is the author and coauthor of several related articles and conference presentations, and he was the technical reviewer of *Extreme NXT: Extending the LEGO MINDSTORMS NXT to the Next Level* by Michael Gasperi et al. (Apress, 2007). In 2004 and 2005, he was guest speaker at the annual ROBOLAB Conference in Austin, Texas. He’s married and has three children, is the director of a boarding institution in Luxembourg, and is the radio amateur LX1BW.

Acknowledgments

I always find reading a book's Acknowledgments section interesting: it is a sort of back stage, where you can get an idea of the work hidden behind these pages.

As is customary, let me first thank all my family. In particular, a thanks goes to my father for the support and inspiration through the development process of the robotic creations in this book. Also, the one-way chats with my mother led me to think aloud and solve many tricky building and programming issues: thanks to her for listening to my incomprehensible ponderings! Thanks to my brother Alessandro, talented guitarist, who spurred me on with suggestions such as "Go and do something serious, instead of playing!" However, this English translation is just a pale rendering of the Italian "*Ma vai a lavoro!*"

Now, I have to thank and acknowledge a lot of people who helped me, more or less directly, during the creation of this book. Forgive me if one name happens to appear before another. Keeping the list in chronological order is as good a rule as another.

I wish to thank Mario Ferrari, major author of well-known LEGO books, who guided my first steps in the LEGO community, since the LEGO Fest where I met him and the members of ITLUG, the Italian LEGO Users Group.

I wish to acknowledge the great work of John Hansen, programmer and MDP/MCP fellow. He is the creator of Not eXactly C (NXC), the powerful textual programming language for the NXT that has been used for the robots of this book. Also, he wrote a number of utilities to interface the LEGO bricks to the computer. One above all, Bricx Command Center (BricxCC), is the environment you'll use to program your robots in NXC. Thanks, John. You gave me the words to instruct my robots!

In this book, you'll enjoy hundreds of detailed building instruction step images, the result of many, many hours of hard work that's not only mine. In fact, the LDRAW system, which I used to draw the 3D CAD models of my robots, is powered by many people who made up the LEGO elements' virtual counterparts. A huge thanks go to all the authors of the parts and of the software: talented people who built up the LDRAW system as we know it today. In particular, I wish to thank the ones who designed the NXT parts you'll see in the following pages: Steve Bliss, Matthias Paul Scholz, and Marc Klein. I myself contributed a little bit, making an early version of the Ultrasonic Sensor front shell.

Philippe Hurbain and Kevin Clague are among those parts' authors, and deserve special thanks. Philippe Hurbain (Philo) is another MDP member and Apress author, who took priceless time to design great-looking CAD versions of most parts of the NXT system. Above all, his masterpieces are the NXT brick, the Sound Sensor, the servomotors, and the BIONICLE claw weapon.

Kevin Clague—MDP member, book author, and creator of many inspiring LEGO bipeds—wrote some really useful programs that I used to assemble the layouts for the building instructions: LEGO Publisher (LPUB) and LEGO Synth, a LEGO bendable parts synthesizer, used to draw the flexible cables. He helped me in learning LPUB, during a period of testing and debugging of the software. Kevin, thank you for your great patience!

Speaking of patience, another person I really want to thank is Claude Baumann, the technical reviewer of this book. My gratitude goes to him for many reasons. First, he believed in my Rubik Solver robot, and invited me, in Luxembourg, to The Science Circus. There I had occasion to see his skill for instruction and passion for teaching. I am glad he accepted the position of my technical reviewer. Apart from the mere technical reviewing of the programs and robots in this book, I wish to thank him for having tried to teach me how to teach. I say "tried" because I was not the ideal pupil! Anyway, your words of advice are precious. Thank you.

Thanks also to the Apress staff, whose names were reported a few pages ago. In particular, thanks to editor Gary Cornell: my brief e-mail was enough for him and his staff to believe in this book that did not exist at that time, except in my head. So, Apress, thanks for having believed in the robots shown on my web page and the material for a book! Thanks to Jeffrey Pepper, lead editor. Thanks to Beth Christmas, my kind project manager, who kept order in my (sometimes) messy material submissions, and John Vacca for his work in getting the "English" into English. On the writing side, thanks to Susannah Davidson Pfalzer, copy editor, who made my style fluent and grammar-examination–proof!

The photos of JohnNXT and the NXT AT-ST robots on the book cover are by Benjamin Maier, professional photographer and friend of mine. I hope he enjoyed spending his time finding my robots' best side and lighting direction. Anyway, I won't steal his time again, so he can direct another feature film—then I'll have the occasion to compose another soundtrack. Thanks, Ben!

I conclude with a little note of regret, while I thank Narendra Sakharam Gaonkar, my native-English-speaking friend and linguistic consultant. Narendra, if only one of the epic and far-reaching alternative titles we conceived for this book had been accepted!

Introduction

You are a LEGO MINDSTORMS NXT owner, aren't you?

If you have this book in your hands, maybe you've already tried (and maybe exhausted) the possibilities offered by the NXT retail set, and the building and programming guides. If not, I recommend that you use those official LEGO guides to start. So, you should have at least a basic idea of what a robot is—otherwise I suspect that you would not even have thought about reading this book!

I began to think of this book as a way to introduce LEGO users to some advanced topics of robotic programming, always keeping it simple, without scaring anyone. In the few theoretical discussions you'll find, you won't have time to get bored: all the theory is explained in order to understand the practice better.

This book is divided into two parts. With the first, I want to break away from the same boring wheeled robots—there are too many of them around. We're used to vehicles; we want to move on legs! So, this part is devoted to walking robots—bipeds in particular. In Chapter 1, I tried to summarize the state of the art for LEGO bipedal walkers. Subsequent Chapters 2, 4, and 5 present three biped robots in order of complexity. Chapter 3 is the only real theoretical chapter, where you learn the finite state machine software technique to give your robot personality and autonomous behaviors. In Chapter 6, the NXT Turtle is described. This is a quadruped robot, featuring a funny autonomous behavior.

The second part is about wheeled robots. I could not write a book without them. That's also because, apart from the Mine Sweeper (an object-collecting vehicle), the other wheeled robot is the great JohnNXT: a replica of Johnny 5, robot star from the *Short Circuit* 1980s movies. I haven't counted the number of people who directly contacted me to ask for JohnNXT instructions, but they are in the hundreds. So, JohnNXT could not be missing from this book.

Except for Chapters 1 and 3, the other chapters containing a robot are organized as follows. At the beginning, the robot is introduced and its capabilities are described. Then, the Not eXactly C (NXC) programs to implement those capabilities are reported and described in detail. Various arguments are deepened, taking advantage of the occasion to discuss programming techniques that arise over and over. The building section is at the end of the chapter. This placement avoids chopping the reading flow in two. The building instructions are introduced first with a detailed bill of materials; then, each step is commented to help with the building. At the end of some chapters, you might find a few exercises, meant to be inspirational cues.

Who Is This Book For?

Mainly, this is a book that should entertain everyone. If the reading will add something to your knowledge, so much the better! So, this book is for the following:

- Those from 6 to 106 years old, wishing to build cool LEGO robots to have fun, without being expert programmers.
- Those who want to build a Johnny 5 replica (more than you might think!).
- Those who need inspiration for their own new creations.
- Those who are tired of exploring the equivalent area of hundreds of computer screens occupied by the graphical NXT-G block programs, who want to change radically the way to program the NXT.
- Those wishing to learn a textual C-like programming language without getting frustrated by complicated useless programs for novices, or bored by abstract exercises. Every program in this book produces visible results.
- Those wishing to learn new programming techniques.

Children, remind your parents that LEGO MINDSTORMS is not their exclusive toy. Ask them for help if you want—you'll have a great time! Parents and grandparents, you can use this book as an excuse to start playing seriously with LEGO robots, while spending time with your kids and grandkids. But let go of that NXT brick—let 'em play too!

What You Need to Use This Book

You can build all the robots using the parts from a single LEGO MINDSTORMS NXT retail set (code number 8527), except the last big one, JohnNXT, and the remote control. So you can enjoy the building and relax—you won't find out that you're missing a needed part when you're a step from the end! If you plan to build and control JohnNXT remotely, I suggest you find all the parts first: you need three NXT sets, and many other extra parts, all listed in the appropriate bills of materials in Chapters 8 and 9.

Then you need a computer to write and send the programs to your NXT robots. The software I used runs on Windows. Mac and Linux releases of the NXC compiler exist, but you'll have to find an alternative Integrated Development Environment (IDE) for the handy Bricx Command Center (BricxCC).

To enjoy this book, you do not have to be a programmer, although it can help you learn the basics to become a programmer. *You can also follow the building instructions and then download the programs provided to your robots, without having to write a single line of code.*

About the software: you should already have the NXT-G program provided by LEGO in your retail set. The other software is the BricxCC IDE and the NXC compiler, both downloadable from `http://bricxcc.sourceforge.net/` and `http://bricxcc.sourceforge.net/nbc/`.

When facing a new programming topic, I recommed that you keep an eye on the complete *Not eXactly C (NXC) Programmer's Guide* by John Hansen, which you can download from `http://bricxcc.sourceforge.net/nbc/nxcdoc/NXC_Guide.pdf`. To get an idea of what the NXC language looks like, you can also read the tutorial I wrote, *Programming LEGO NXT Robots using NXC*. This paper is available for free at `http://bricxcc.sourceforge.net/nbc/nxcdoc/NXC_tutorial.pdf`.

Source Code and Extras for This Book

You can download the complete source code for the programs from the Source Code/Download area on the Apress web site, at `http://www.apress.com`. Also, you can visit the web site `http://robotics.benedettelli.com`.

PART 1

Look, Mom! No Wheels!

After building your first wheeled robots, you can feel bored. Okay, they're built for precision, you can make them go exactly where you want, at the speed you want . . . but they still use wheels! LEGO itself, planning a new MINDSTORMS line, never thought about a wheeled robot becoming its logo and NXT mascot—months before the product release, the figure of Alpha Rex filled every advertising space. In this first section of the book, you'll discover how to leave the wheels behind and get moving on legs.

CHAPTER 1

Building Biped Robots

I can imagine your impatience—the urge to skip this introduction chapter altogether, and go directly to building the robots that are shown in the next chapters of this part, which are entirely devoted to walker robots. However, you would entirely miss the essentials necessary to understand why the walkers presented in this book actually work; you would miss finding out how to let your robots leave the wheels behind them and get on their own two feet.

This chapter will present the state of the art in LEGO walking biped robots. I'll introduce some basic notions of that branch of physics called statics (balancing forces) to help you develop steady biped robots that do *not* need to use any advanced sensor to balance, such as accelerometers, tilt sensors, or gyroscopes. The stability of those robots is guaranteed only by the hardware configuration.

LEGO Bipedal Walking: The State of the Art

It has been almost ten years since LEGO MINDSTORMS users like you developed various bipedal walking techniques. The numerous biped robots, created fairly successfully during those years, can be categorized as follows:

- Interlacing legs bipeds
- Jerky center of gravity (COG from now on) shifting bipeds
- Smooth COG shifting bipeds

I'll describe these categories in detail, focusing on their level of complexity and the mechanical solutions used, with the help of some visual examples. In the next chapters you'll find the practical examples of this categorization: Quasimodo (Chapter 2) is an interlacing legs biped, *Star Wars* AT-ST chicken walker (Chapter 4) is a jerky COG shifting biped, while the Omni-Biped (Chapter 5) is a smooth COG shifting biped.

Interlacing Legs Bipeds

Robots that fall in the category of interlacing legs bipeds generally use the simplest walking technique. In other words, you must figure out what is the best way to put one foot in front of another. The solution is usually a cam shaft (see Figure 1-4*c*, *d*, and *f*). With the parts provided in the NXT retail set, you can easily build a cam shaft using the holes in the 24-tooth or the 40-tooth gear (see Omni-Biped's legs in Chapter 5). You must attach the legs to an off-center hole to achieve what's called *eccentric motion*. As an alternative, you can use the black 3-long liftarms (as in Quasimodo, Chapter 2).

The particular shape of the feet stabilize a robot of this category (see Figure 1-1*a*)—feet interlace each other (hence the name of this category). Usually, in this kind of robot, the center of gravity is not shifted from side to side. So, you must pay close attention to designing the structure in such a way that the COG projection on the ground always falls inside the area that supports the feet, during each phase of walking. This condition must hold when the feet are on the ground together, but also when one of them is lifted from the ground.

Figure 1-1 shows the various walking phases. In *a*, both feet are on the ground; in *b* and *c*, the right foot begins to step forward and is lifted from the ground, leaving all the weight loaded on the left foot and reducing the support area to only the left foot; in *d* both feet are on the ground again. Next, the process starts again with the left foot leaving the ground and stepping forward.

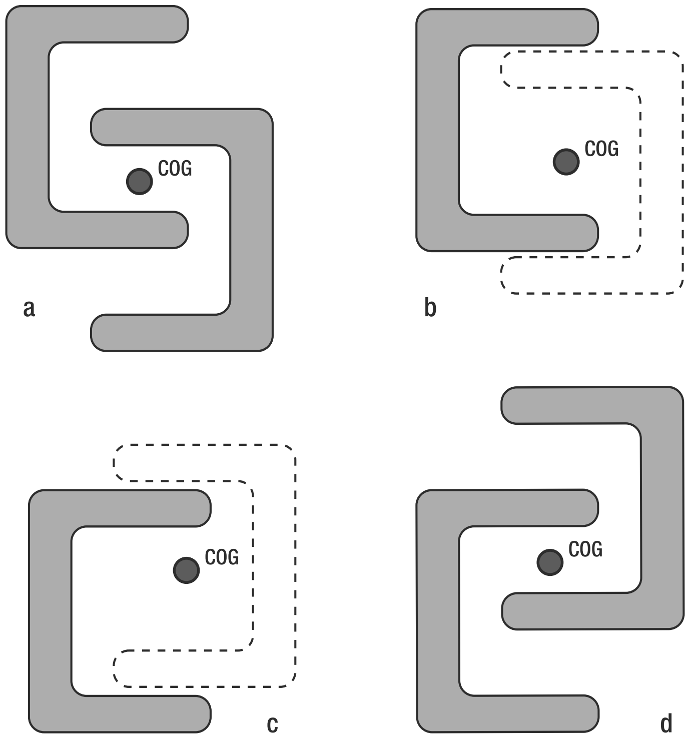

Figure 1-1. *Interlacing legs biped footprints in the various phases of walking*

The preceding approach suffers from the slackness of the LEGO joints: the legs tend to bend inside and the feet do not lift completely from the ground. To solve this problem, you can place a wedge in the inner side of the feet as shown in Figure 1-2*c*, or you can provide your biped with a sort of "hip tendon" made with rubber bands or LEGO parts, as shown in Figure 1-2*d*. All bipeds in nature have similar muscles to keep their equilibrium, so that they can walk steadily.

LEGO veteran users tend to be purists and quite conservative. They might consider it a sacrilege to use non-LEGO parts in your robot building. If you are willing to use rubber bands, you should use original LEGO rubber bands, although it is not the most elegant solution.

However, the best solution is to keep the whole leg frame short and rigid by using cross-bracing. The weaker parts of the leg are usually the moving joints, at the ankle and hip level.

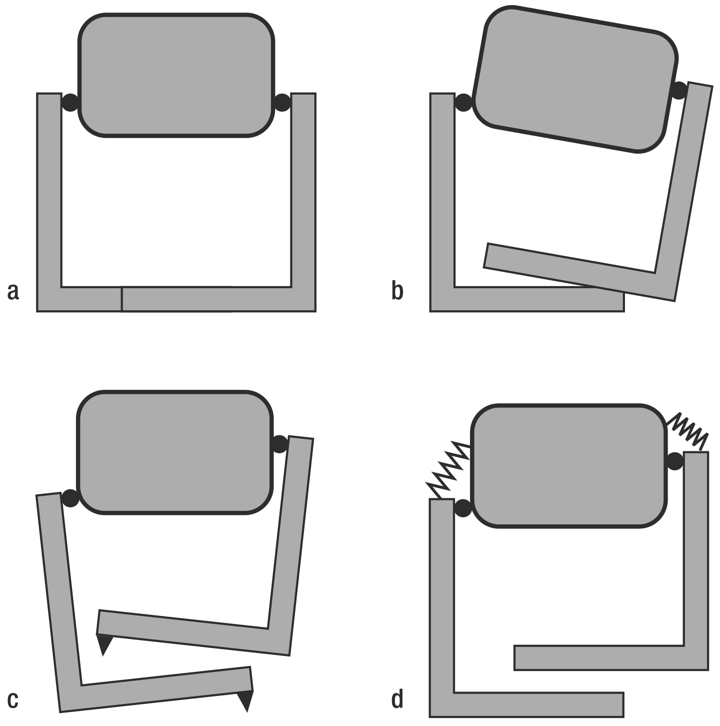

Figure 1-2. *Solving the problems related to LEGO joints' flexible nature*

All this might seem a bit abstract to you, but it will all come clear when you build your own Quasimodo—a biped, and the subject of the next chapter. In Chapter 2, I'll emphasize the defects of this particular walking mechanism and will cover the remedies step by step.

Jerky COG Shifting Bipeds

If you want to get really serious, or if your robot begins to get heavier, you should start thinking about COG shifting. As the term implies, the robot shifts its weight on the foot that remains on the ground, and unloads the other foot that is in the flight phase.

The adjective "jerky" implies that shifting the COG and stepping forward occurs in distinct stages. It also implies that it is done with different motors.

You can accomplish weight shifting while moving the whole mass (RCX or NXT, where most weight resides) from side to side, or by bending the legs at the ankle, knee, or hip level (see Figure 1-3).

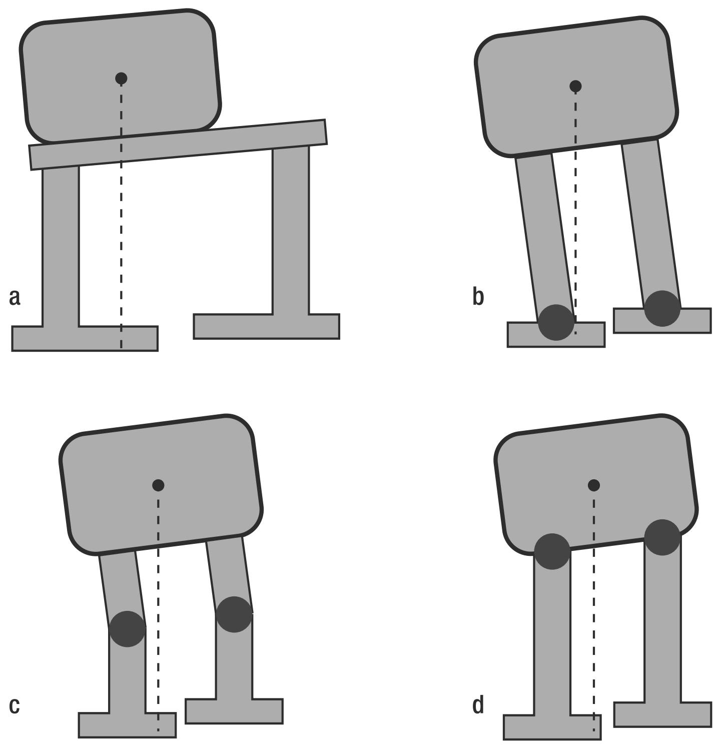

Figure 1-3. *Weight shifting methods*

You can achieve stepping by rotating the legs or *translating* them, keeping them parallel to each other (see Figure 1-4*a*, *b*, and *e*). In these cases, if the feet are moved while they are both touching the ground, the resulting effect is that the robot turns slightly in place. You'll use this feature in the AT-ST biped to make it turn (see Chapter 4).

Figure 1-4 from *c* through *f* shows various stepping solutions. In these pictures, just one leg is shown for clarity: you must attach the hidden one (grayed in *c* and *d*) on the other side of the robot 180 degrees out of phase. For example, in Figure 1-4*c* and *d,* you should attach the legs, on the opposite sides of the robot, at the leftmost position on the cam (the white circle), so that one leg is ahead of the other one. You must apply the same concept when attaching the other leg in elements *e* and *f*.

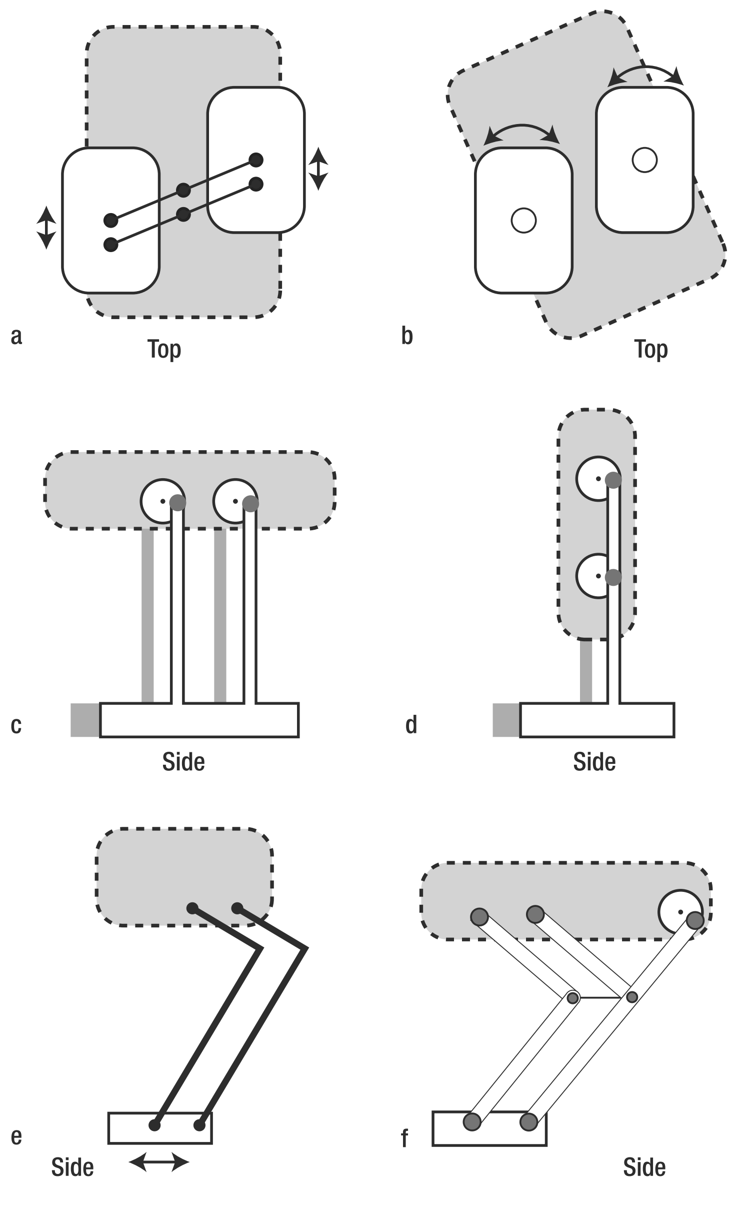

Figure 1-4. *Various stepping methods: a) this top view shows that the legs are translated, keeping them parallel each other; b) another top view where the legs are rotated together; c) side view showing a cammed mechanism keeping the body horizontal; d) a similar side view showing a cammed mechanism with the body kept vertical; e) this side view shows another solution for stepping; f) this cammed mechanism works like the one in c but is better looking.*

As shown in Figure 1-5, many of the bipeds I've built follow this line, such as the Matrix APU (February 2004), the OWL (May 2004), and the RCX-based AT-ST (December 2004). The new NXT AT-ST that you will find in Chapter 4 falls in this category. The dates in parentheses are the construction dates, showing the month in which these creations were published on the Web.

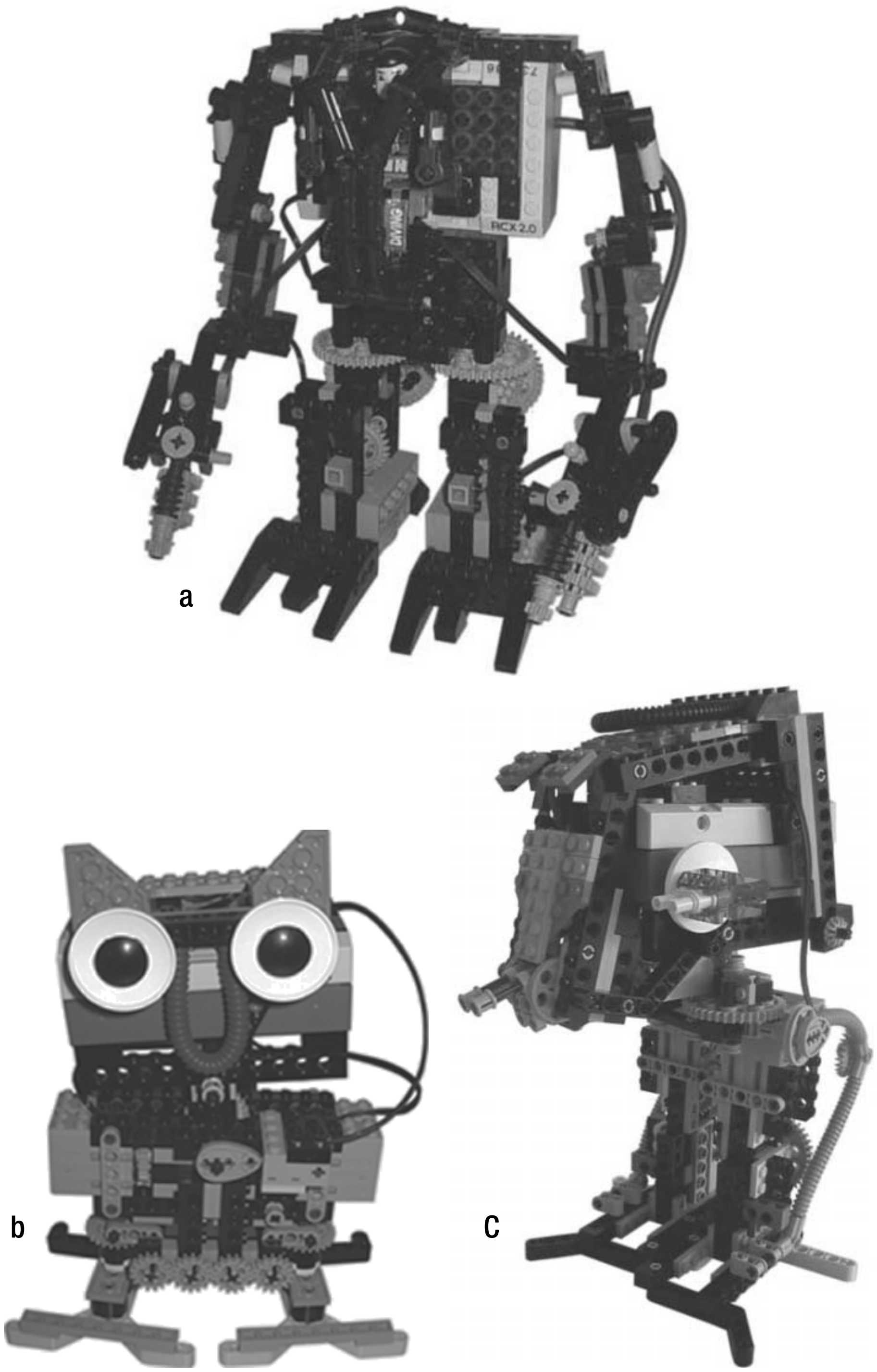

Figure 1-5. *RCX-based jerky COG shifting bipeds: a) Matrix APU, b) OWL, c) AT-ST*

Smooth COG Shifting Bipeds

I prefer smooth COG shifting bipeds due to their speed and smoothness—you probably will, too. They use a single motor to achieve both weight shifting and stepping; hence you need a refined mechanism.

In Figure 1-6, you can see the biped robot that I built in August 2004. This particular robot could only walk straight using COG shifting, and was meant to be a prototype for future bipeds that could also turn.

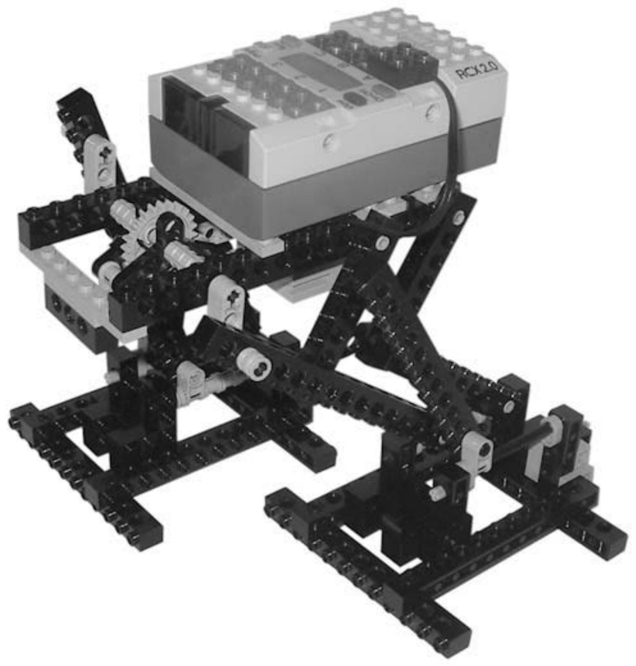

Figure 1-6. *COG-shifting Biped I*

My Advanced COG Biped (dated October 2004), shown in Figure 1-7, features a smooth walking gait and can turn thanks to a motorized rotating ankle. The left foot includes a motor to allow it to turn, while the other foot is built of normal parts (in the Robotics Invention System kit, there were only two motors), trying to balance the other foot's weight. On each foot, a touch sensor lets the robot know which foot is touching the ground.

The robot uses this sequence for turning: when the left foot is lifted from the ground, it is swung outside at a little angle; then the robot steps to load the weight on the foot, to bring it onto the ground. The ankle is then rotated back in its place and the whole robot turns left.

You can repeat this routine as many times as needed to turn in place, and you can modify it slightly to turn in the other direction.

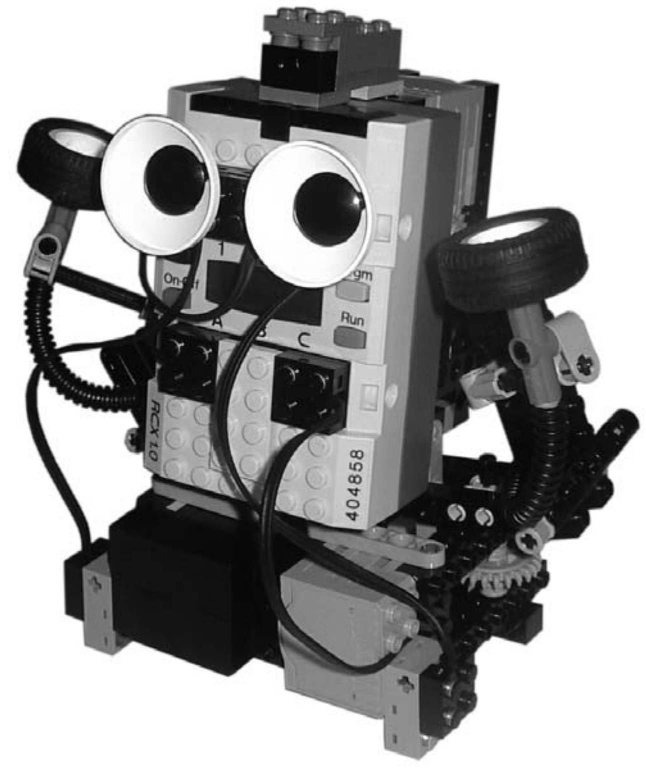

Figure 1-7. *Advanced COG Biped*

Summary

In this introductory chapter, you saw various approaches to bipedal walking. You were shown some startup ideas that might shed light on the mystery of how to develop a biped that not only walks, but can also turn.

To make a biped that walks without resembling a drunken sailor (and that is not at risk of falling down with every step it takes), the clever designer should try to bind the COG projection inside the area supporting the feet. You should do this during every phase of the walking cycle, when both feet are touching the ground and when one of them is in flight phase, leaving the other to support the whole robot's weight. This condition is essential for creating a biped without advanced sensors such as accelerometers or gyroscopes. Such a robot is a static one, because dynamic effects are not taken into account. Stability is guaranteed only by hardware structure, with no need of sensor feedback.

In the next chapters, you'll create three bipeds using all these techniques.

CHAPTER 2

Quasimodo

The hunchbacked biped described in this chapter fits our educational purposes perfectly. By building it, you'll get to see in practice all the theoretical tips, tricks, and rules explained in Chapter 1 about the stability of biped robots.

I decided to call this biped Quasimodo (after the hunchback of Victor Hugo's famous book *Notre-Dame de Paris*), because of its particular shape and the funny way it walks. When you build it, you'll see what I mean. You can get an idea of how Quasimodo looks in Figure 2-1.

I recommend that you read the following sections with a bookmark in Chapter 1, as you'll see that the practical examples of this chapter and the theory that lies behind them (in Chapter 1) correspond.

Throughout this book, in chapters where a model is being built, the chapters will be organized the same way. The first part of the chapter will discuss key features of the model, and its overall function; the second part will discuss the program that has been created to make the model run; and the third part shows the building instructions, part by part, so that you can create the given model yourself.

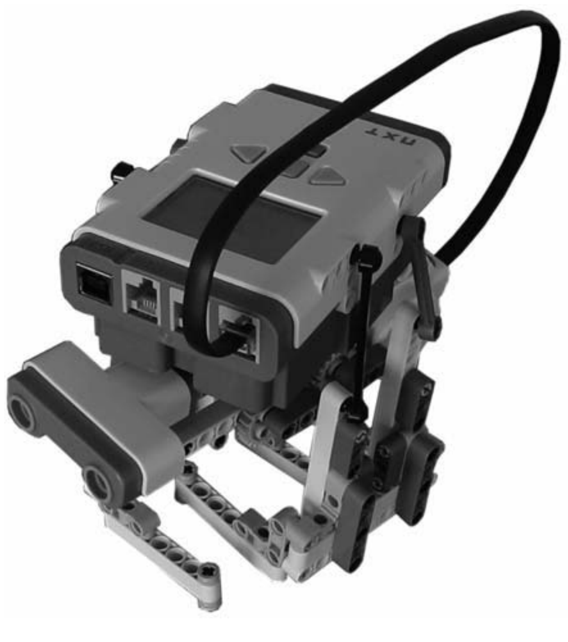

Figure 2-1. *The NXT biped of this chapter, Quasimodo*

Applying What You Learned

Quasi (to his friends) is an interlacing legs biped, the simplest approach you can adopt for a biped. When designing such a biped, you might start thinking about the leg shape and the cammed mechanism needed to step forward. You can choose two paths: should the cams be aligned in a horizontal or vertical position (as shown in Figure 1-4*c* and *d*, respectively)? Placing the geartrain on a vertical beam yields a much higher and more unstable biped, because the motor and the NXT would be placed in a vertical position. It would be more difficult to design the structure to have the NXT center of gravity (COG) inside the feet area. The COG itself would be placed higher, so a slight oscillation due to leg joint flexibility would result in a large oscillation at the top of the robot, causing the COG to go outside the feet area and the robot to fall. For these reasons, I chose to settle the motor, the geartrain beam, and the NXT horizontally.

After I had designed the legs and the motor placement, I didn't know where to put the NXT brick itself. I ended up putting it on the robot's back, just like a hump, and attaching it to the legs so that it swings left and right as the legs move. This way I also achieved COG shifting (as explained in Chapter 1), with surprising ease. Often you have to think hard about a satisfactory solution, but in some rare cases, as here, things seem to work out by themselves.

Let me guide you through the design process that led to the final look of Quasi. This biped is meant to be an educational model, to help me to emphasize the defects and problems you could encounter while designing your first bipeds. The leg attachment to the cams is the weak part of this interlacing legs biped, because the cam pins fit a bit loosely in the leg beam holes. As you'll see, you can solve these problems to result in a biped that walks well.

■ **Caution** When I talk about LEGO parts' flexibility, I do not mean that LEGO beams actually bend. *Do not try* to force LEGO parts, or you might risk breaking something! Here, I mean the looseness of the parts shown in certain assemblies, due to their construction tolerance and material. Examples of this slackness are the pins, which don't fit tightly inside the beam holes, making the leg structure not perfectly rigid.

The feet are shaped as shown in Figure 1-1, to let the COG projection always fall inside the supporting area of the feet. You attach the legs to the motor geartrain by a cammed shaft that keeps them 180 degrees out of phase; this simply means that when one leg is up, the other is down; when one is forward, the other is back.

Unfortunately, as you might expect, such a structure is so loose that it risks falling apart. Figure 2-2 shows what I mean. As already explained in Chapter 1, the weaker points of such a biped are the attachments at the hip and ankle levels.

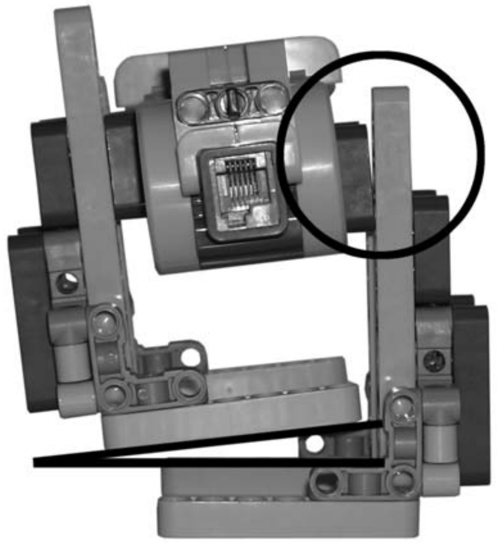

Figure 2-2. *The biped from behind (with the NXT on top removed) shows the looseness of the structure at hip level. The ankle is quite rigid instead.*

It's easy to run into similar problems when working with LEGO parts. Having such a loose structure is a problem that can arise, but don't worry. You can solve it as indicated in the schematic shown in Figure 1-2*c*, by adding the wedges in the inner side of the feet beams. The result of our biped is shown in Figure 2-3.

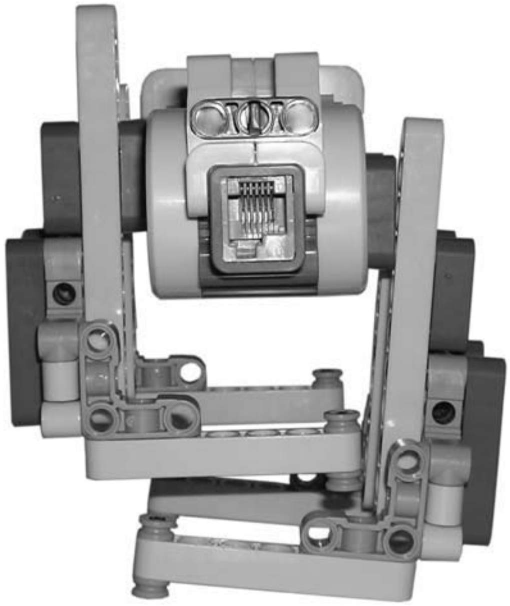

Figure 2-3. *Putting wedges in the inner side of both feet compensates for the leg joints' slackness.*

Even after the wedge additions, the hip joint still tends to be quite loose (it's made with long gray pins, which connect the cam to the leg). To solve this tricky issue, I adopted the hip tendons shown in Figure 1-2*d*, a matter that could have been obscure to you just after reading Chapter 1. Don't worry though, it will become clear now.

Compare Figure 2-4 (before the treatment) with Figure 2-5 (after the treatment). Notice how this elegant solution with the tendons made from LEGO steering links with ball joints, prevents the legs from bending. This last idea of creating tendons is particularly good because it does two things for the price of one: it solves the looseness problem and allows us to connect to the NXT brick in an original way.

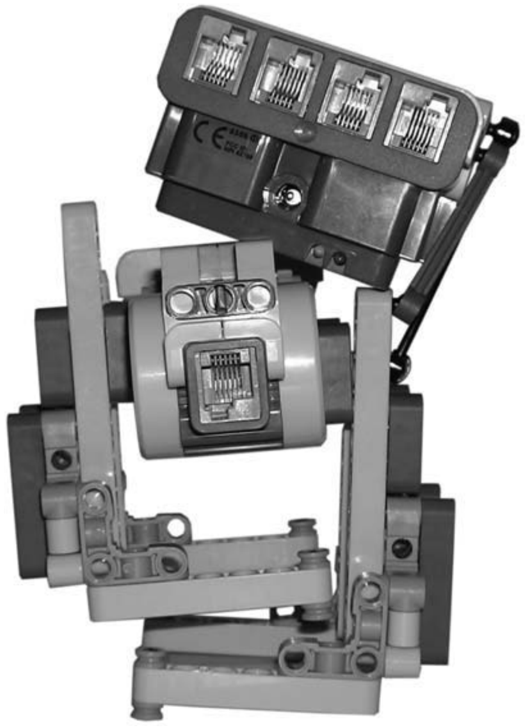

Figure 2-4. *Tendons are not attached yet.*

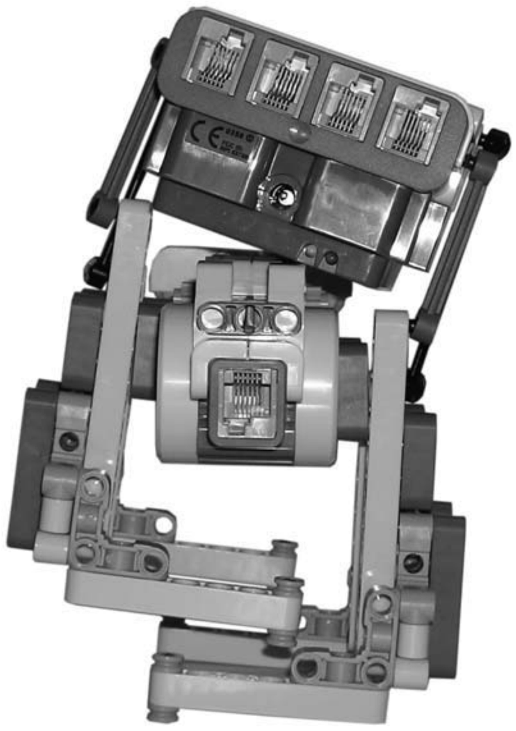

Figure 2-5. *Tendons are now attached.*

Because the tendons are connected to the legs, they swing the NXT in harmony with Quasimodo's gait, and the NXT seems as light as a butterfly. Don't forget that our beloved programmable brick acts as a hump here! I came up with this COG shifting mechanism almost without noticing it, and I must admit this combination of technique, inspiration, and luck is rare. Such a mix makes this robot special. In its simple shape, it summarizes a lot of theory about an unusual way of walking. Ah, I almost forgot: Quasimodo can only walk straight. To create a biped that turns, read the following chapters.

Introducing NXT Technology

Before going on, it's worth introducing the LEGO MINDSTORMS NXT technology briefly. In your NXT retail set, you have LEGO parts, of course, but also some electronic devices that make the NXT system special: three interactive servomotors, a Touch Sensor, a Light Sensor, an Ultrasonic Sensor, and the NXT programmable brick itself. In addition, you have a user guide, and the LEGO software CD-ROM, which allows you to program the NXT using the NXT-G graphical programming language.

The LEGO elements are well assorted, so that you can start creating every kind of robot at once, without having to look for additional spare parts. The set includes LEGO TECHNIC studless elements, except for a few parts. Unlike the common LEGO studded bricks, you do not have to place one brick on another, like building a wall, but you have to start thinking more three-dimensionally, attaching beams and liftarms using pins.

The *NXT servomotors* are different from the common LEGO motors. They are interactive, meaning that they include a Rotation Sensor (optical encoder) that allows you to control interactively the shaft position with 1 degree of resolution, and to set the rotation speed from –100 to 100. A whole shaft rotation is equal to 360 degrees.

The *Touch Sensor* gives your robots the sense of touch: it detects when it is pressed or released, returning a Boolean reading that can be 1 or 0. The *Light Sensor* can distinguish between light and dark colors, measuring the amount of light reflected by the surface illuminated by its LED; it can also measure the light intensity in the environment with the LED off. The *Sound Sensor* makes your robot hear, measuring the sound intensity in decibels. Its readings go from 4 in a silent room to 100, corresponding to people shouting or loud music.

The *Ultrasonic Sensor* enables your robot to see obstacles, measure distances, and detect movement. This digital sensor measures the distance from an object like a bat does, calculating the time needed by an ultrasonic sound wave to hit the object and return. It can measure distances from 0 to 255 centimeters, with an error of **5** 3cm.

Finally, the brain of your robot is the *NXT brick*. It is a microcomputer, programmable with a PC, that lets your LEGO robots come alive, just like JohnNXT (see Chapter 8). You can connect the NXT brick to your PC using a USB cable or Bluetooth. Bluetooth wireless communication is useful if you want to control your robots remotely, or just program it without annoying cables around. You can also connect more NXTs using Bluetooth, to make big complex robots. The NXT has three output ports for attaching motors and four input ports to connect sensors; it has a large dot-matrix screen to display text, numbers, and images. Also, your robots can produce sounds, because the NXT features a loudspeaker to play tones and WAV-like sound files. Two microprocessors are at the base of the NXT brick. The main processor is an Atmel ARM7 (like the one you might have in your mobile phone), and works at 48 MHz, on 32 bits. This allows your robots to deal with large numbers, making calculations at a high speed. The NXT has 256KB of nonvolatile memory; you can store files into it and they won't be erased, even if you remove the batteries.

Oh, I forgot! The NXT needs six AA batteries to work, but can also be powered by the LEGO Li-Ion rechargeable battery. For other details, you can always consult the NXT User Guide included in your retail set.