TECHNOLOGY IN ACTION™

LEGO® MINDSTORMS® EV3

The Mayan Adventure

Second Edition

Mark Bell
James Floyd Kelly

Apress®

LEGO® MINDSTORMS® EV3

The Mayan Adventure

Second Edition

Mark Bell
James Floyd Kelly

Apress®

LEGO® MINDSTORMS® EV3: The Mayan Adventure

Mark Bell
Northridge, California,
USA

James Floyd Kelly
Smyrna, Georgia,
USA

ISBN-13 (pbk): 978-1-4842-2261-4
ISBN-13 (electronic): 978-1-4842-2262-1
DOI 10.1007/978-1-4842-2262-1

Library of Congress Control Number: 2017936711

Managing Director: Welmoed Spahr
Editorial Director: Todd Green
Acquisitions Editor: Jonathan Gennick
Development Editor: Laura Berendson
Technical Reviewer: Andrew Milluzzi
Coordinating Editor: Jill Balzano
Copy Editor: Kezia Endsley
Compositor: SPi Global
Indexer: SPi Global
Artist: SPi Global

Distributed to the book trade worldwide by Springer Science+Business Media New York, 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 www.springeronline.com. Apress Media, LLC is a California LLC and the sole member (owner) is Springer Science + Business Media Finance Inc (SSBM Finance Inc). SSBM Finance Inc is a Delaware corporation.

For information on translations, please e-mail rights@apress.com, or visit http://www.apress.com/rights-permission.

Apress titles may be purchased in bulk for academic, corporate, or promotional use. eBook versions and licenses are also available for most titles. For more information, reference our Print and eBook Bulk Sales web page at http://www.apress.com/bulk-sales.

Any source code or other supplementary material referenced by the author in this book is available to readers on GitHub via the book's product page, located at www.apress.com/9781484222614. For more detailed information, please visit http://www.apress.com/source-code.

Printed on acid-free paper.

"For me, there is only the traveling on paths that have heart, on any path that may have heart,
and the only worthwhile challenge is to traverse its full length—
and there I travel looking, looking breathlessly."
(from Carlos Castaneda, The Teachings of Don Juan: A Yaqui Way of Knowledge)
I dedicate this book to my wife Shelley, traveling on that path of heart.

—Mark Bell

I'd like to dedicate this book to Ashley. My wife truly understands that
"the difference between men and boys is the price of their toys."
Thank you for your support and encouragement.
Up next—our own special, little project . . .

—James Floyd Kelly

Contents at a Glance

About the Authors xv

About the Technical Reviewer xvii

Acknowledgments xix

Introduction xxi

■ **Chapter 1: Tomb, Trap, and Trigger** 1

■ **Chapter 2: ExploroBot: Planning and Design** 7

■ **Chapter 3: ExploroBot: Build It** 17

■ **Chapter 4: ExploroBot: Program It** 41

■ **Chapter 5: String, Pebbles, and Gravity** 57

■ **Chapter 6: StringBot: Planning** 61

■ **Chapter 7: StringBot: Build It** 71

■ **Chapter 8: StringBot: Program It** 93

■ **Chapter 9: Scroll, Key, and Camera** 109

■ **Chapter 10: SnapShotBot: Planning and Design** 117

■ **Chapter 11: SnapShotBot: Build It** 129

■ **Chapter 12: SnapShotBot: Program It** 155

■ **Chapter 13: Get In, Grab It, Get Out** 169

■ **Chapter 14: GrabberBot: Planning and Design** 175

■ **Chapter 15: GrabberBot: Build It** 183

■ **Chapter 16: GrabberBot: Program It** 217

Chapter 17: Bravery, Wisdom, and Honor 231

Chapter 18: PushBot: Planning and Design 237

Chapter 19: PushBot: Build It 243

Chapter 20: PushBot: Program It 291

Chapter 21: Discovery, Secret, and Home 313

Appendix A: The MINDSTORMS Community and EV3 Web Sites 315

Appendix B: Robot Commander Remote Control App 319

Appendix C: Kit Organization: Where Do All Those Parts Go? 323

Appendix D: Building Instructions for Bots 327

Index 331

Contents

About the Authors xv
About the Technical Reviewer xvii
Acknowledgments xix
Introduction xxi

Chapter 1: Tomb, Trap, and Trigger 1
Day 2: King Ixtua Tomb Excavation, 4:42 PM 1
Tunnel Challenge 3
Evan's Solution 4

Chapter 2: ExploroBot: Planning and Design 7
The ExploroBot 7
The Robot Description 9
The Task List 10
Limitations and Constraints 11
Mindstorm 13
Sketches 15

Chapter 3: ExploroBot: Build It 17
Never Be Afraid to Experiment 17
Step by Step 18
First Section: Infrared Sensor and Neck 18
Second Section: Bot Body and Motors 21
Third Section: Rear-Wheel Assembly and Reinforcement Strut 26
Engineering: Axles as Reinforcements: Strong and Adjustable 31
Fourth Section: Put It All Together 33

Chapter 4: ExploroBot: Program It **41**
Some Experience Required 41
Into the Tunnel 45
Engineering: *Encoders:* How Does a Motor "Know" Where It Is? 49
Out of the Tunnel 51
What the Degree, Kenneth? (With Apologies to REM) 53
Engineering: Measuring Actual Motor Rotations from the Brick Screen 53
Opening the Tomb Door 55
Chapter 5: String, Pebbles, and Gravity **57**
Day 3: Inside King Ixtua's Tomb, 8:13 AM 57
More Monkey Business 57
Vine Challenge 59
Evan's Solution 60
Chapter 6: StringBot: Planning **61**
Design and Planning 61
The StringBot 61
The Robot Description 62
The Task List 63
Limitations and Constraints 64
Mindstorm 66
Sketches 69
Chapter 7: StringBot: Build It **71**
Where to Start? 71
Step by Step 72
First Section: Right-Side Motor Assembly with IR Sensor 73
Second Section: Left-Side Motor Assembly with Guides and Rubber Wheel 77
Third Section: Carrier and Pebble Release Mechanism 85
Fourth Section: The Intelligent Brick and Putting It All Together 89

Chapter 8: StringBot: Program It 93
Get Familiar with the Blocks 93
The STOP Block 96
The SWITCH Block 96
Getting to the Vase 97
Above the Vase 101
Back for More 104
Testing: Filling the Vase 106
Chapter 9: Scroll, Key, and Camera 109
Day 3: Tomb Reception Area, 6:08 PM 109
The King's Library 110
Key Retrieval Challenge 112
Grace's Solution 114
Chapter 10: SnapShotBot: Planning and Design 117
SnapShotBot Planning and Design 117
The Robot Description 118
The Task List 119
Task 1 120
Task 2 121
Task 3 121
Task 4 122
Task 5 124
Task 6 125
Task 7 125
Task 8 125
Task 9 125
Task 10 125
What About the Twine? 125

Limitations and Constraints 126
Mindstorm 126
Sketches 128
Chapter 11: SnapShotBot: Build It 129
Engineering: "Good Enough" with a Peg-Leg? 129
Jump In 130
Step by Step 131
First Section: Basic Body with Hardpoints 131
Engineering: The Concept of Stiffness 134
Engineering: Hardpoints 135
Second Section: Color Sensor, IR Sensor, and the Camera Trigger 135
Engineering: Gears Trade Speed for Power 143
Third Section: Adjustable Camera Frame 146
Engineering: Early and Rapid Mechanical Testing 149
Chapter 12: SnapShotBot: Program It 155
One Block at a Time 155
Finding the Basket 159
Getting Around the Basket 160
Getting the Bot Home 163
Chapter 13: Get In, Grab It, Get Out 169
Day 4: Outside King Ixtua's Library, 8:43 AM 169
The Throne Room 169
Locate the Burial Chamber 170
Scroll Challenge 172
Max's Solution 174
Chapter 14: GrabberBot: Planning and Design 175
GrabberBot Planning and Design 175
The Robot Description 175

The Task List 176
Limitations and Constraints 178
Mindstorm 178
Sketches 181

Chapter 15: GrabberBot: Build It 183

First Section: Main Body Tank-Treads 184
Second Section: Grabber Assembly (Lifting Arm Mechanism) 194
Third Section: Infrared Sensor, Touch Sensor, and Wiring 210
Engineering: How to Hold the Jaw Shut? An Elastic Grabber Latch 213

Chapter 16: GrabberBot: Program It 217

Down the Tunnel . . . Again 217
Engineering: A Well-Regulated START 218
Engineering: Precise Calculation of Distance: Rotations and Degrees 219
Approaching the Scroll 221
Acquiring the Scroll 224
Engineering: Parallel Processes 225
Engineering: Testing and Making Changes 228
Engineering: Timing the Parallel Paths 228

Chapter 17: Bravery, Wisdom, and Honor 231

Day 5: Inside King Ixtua's Throne Room, 10:12 AM 231
The Burial Chamber 232
Famous Figures 233
The Final Challenge 234
Evan's Solution 235

Chapter 18: PushBot: Planning and Design 237

PushBot Planning and Design 237
The Robot Description 237
The Task List 239

Limitations and Constraints 240
Mindstorm 241
Sketches 242

Chapter 19: PushBot: Build It 243

Step by Step 244
Engineering: The Difference Between Designing and Explaining 244
First Section: Figurine Jaw Cage/Medium Motor Mechanism 244
Engineering and the Concept of Stiffness II: Angles, Brackets, and Connectors 252
Second Section: Neck/Infrared Sensor Frame Assembly with Caster Wheel 260
Engineering: Rear-Wheel Designs and Their Constraints 260
Third Section: Main Body and Motors 275
Engineering: Prototypes and Parts You Don't "Need" 280

Chapter 20: PushBot: Program It 291

Getting the PushBot Into Position 291
Positioning Three Figurines 297
The Final Figurine 306

Chapter 21: Discovery, Secret, and Home 313

Day 7: Base Camp, King Ixtua's Tomb, 11:05 AM 313

Appendix A: The MINDSTORMS Community and EV3 Web Sites 315

LEGO Club User Account 315
Community Gallery 316
Online Reference and Support 317
Web Sites 317
Blogs and Forums 318

Appendix B: Robot Commander Remote Control App 319

Downloading and Running the Robot Commander App 319
The Perfect Hardware-Testing App 320
Engineering: Isolating Design Issues in Testing 321

Appendix C: Kit Organization: Where Do All Those Parts Go?..... 323
The Top Tray for the Small Parts..... 323
Bottom Tray for the Large Parts..... 324
A Secure Lid Is Your Friend 325
Appendix D: Building Instructions for Bots 327
What Will the Background Be?..... 327
Step by Step Picture-Taking Strategy 328
Lighting, Camera, and Lens Choices 329
Index..... 331

About the Authors

Mark Bell has for years been teaching *Introduction to Robotics* with CTY, the Johns Hopkins Center for Talented Youth summer program. From the beginning, he used the First Edition of *Mayan Adventure* for every one of those classes, because of its archeologist storyline and strong support for engineering process and requirements analysis. Prior to working with CTY, he was a Principal Application Engineer and sometime Director of Marketing for software design and requirements engineering tools companies. His customers included NASA JPL, TRW, Boeing, Hughes Aircraft, and the NRO (among others). He is also engaged as a Middle School Science Teacher in Valley Village, CA. Mark's *EV3 Mayan Adventure* web site is `http://ninevolt.ninja` - stop on by!

James Floyd Kelly is a full-time writer from Atlanta, Georgia. James has written over 30 books on a variety of subjects that include LEGO robotics, Computer-Aided Design for 3D printing, and Minecraft. He also teaches a number of technology summer camps where the kids learn game programming, basic making skills, and even how to build their own computer. He and his wife have two boys who never tire of projects.

About the Technical Reviewer

Andrew Milluzzi is a life-long LEGO fan. A LEGO MINDSTORMS Community Partner and seasoned *FIRST* volunteer, Andy loves to share his passion for robotics with others. Besides creating with LEGO MINDSTORMS, he is an avid Maker and amateur radio operator. With degrees in engineering from Rose-Hulman Institute of Technology and the University of Florida, Andy is always experimenting and solving problems. To connect with Andy or learn more about his projects, check out his web site at http://O8milluz.com.

Acknowledgments

Second Edition Acknowledgements: Mark Bell

As Jim says, writing a book is a lot of work! And that's certainly true for a revision, since everything in a technical book needs to be as accurate and up-to-date as possible.

Richly deserved thanks go to Jonathan Gennick, Assistant Editorial Director with Apress. Jonathan encouraged me to produce the revision proposal for this book and was closely involved with the original. He supplied me with a *long* book proposal form and lots of support. That Apress proposal form was a handsomely designed author-education in its own right, and the work to fill it out showed both of us that this revised book was a Go.

Jill Balzano was our Coordinating Editor at Apress, keeping all the moving parts lined up, making sure those pieces worked and the deadlines were met. She was businesslike and a pleasure to work with.

Technology books, just as Jim says, are indebted to the technical editor. I was grateful for Andy Milluzzi as he built the robots directly from the text and ran the software to make sure it does what it is supposed to do. He identified errors and, beyond that, supplied vital knowledge of how to "speak to the LEGO Culture." The book is better because of his contributions. Thank you, Andy, for catching errors and finding clearer ways to explain some of the models.

Finally, acknowledgement goes to my Robotics teaching assistants at Johns Hopkins CTY—Center for Talented Youth. This remarkable program staffs up every summer and recruits first-class people. Several of mine have gone on to earn PhDs and establish academic and publishing careers of their own. Thanks to Shana Graff, Steve Earth, Brian Tsui, Michael Pepper, and Justin Ning Hui Li, who started it all. And thanks to my CTY mentors—Deborah Rosenquist got me in, Huseyin Sencan got me started, and good ol' Rick Nestoff made a heck of a team in CTY Hong Kong.

And to a couple of recent students here in California—Dallas and Dashiel—I say "Boosh!"

First Edition Acknowledgements: Jim Kelly

Writing a book is a *lot* of work! And I'm not talking about my work. There are so many people who have contributed some excellent work to the book you're holding, and I'm glad to have this opportunity to thank them for their hard work.

First, thanks go to Jim Sumser, Lead Editor with Apress. Jim read my original proposal for the book and must have seen something promising because I've had nothing but complete support during the entire writing and editing process. When he e-mailed to let me know that Apress wanted to do the book, I think my heart actually skipped a beat when I realized he was serious and that I would actually have to write this thing!

I figured out quickly that I was going to need some serious organizational help with getting this book written. I have to thank Tracy Brown Collins at Apress, my Project Manager, for her work and apologize for any stress I may have caused her. For a book with this many drawings, figures, photos, and screenshots, I am amazed at her ability to keep it all organized and keep me moving forward toward a finished project. Tracy, you helped this writer stay on track, and you did it with kindness and support. Thank you so much!

Evun an Englush major can make spelling and grammur mustakes (just kidding). For helping me clean up the text and fix quite a few errors, I have Susannah Davidson, Bill McManus, and Ami Knox to thank. Going over their fixes and suggested changes was a good review for me—thank you all for the great work!

You might have noticed that this book has a lot of figures. Well, someone had to help me clean them up and redraw my horrid hand sketches. I was fortunate to have a team that really deserves the credit, and this includes April Milne. Thank you all for taking what looked perfect in my head but terrible on paper and turning it into something to be proud of! And a special thank you goes to Kurt Krames for the great cover mixture of Mayan pyramid and NXT robot. Who could imagine these two images sharing a book cover?

And finally, with any technology book, a huge amount of thanks has to go to the technical editor. I was fortunate to have Brian Davis help me with testing the robots and double-checking my programs. He caught quite a few errors and offered up some suggestions for better ways to explain some complicated subjects. The book is much improved thanks to his efforts. Thank you, Brian, for your attention to detail and for your feedback.

Another thank you goes to Jeff Gennick who provided questions and feedback during the writing of my chapters. Jeff (and his dad, Jonathan Gennick, another Apress staff member) purchased an NXT kit early on and helped me with early testing of the initial robot designs. Thanks go to both of them for their help.

I'm certain there are others who were working hard behind the scenes, and I'd like to thank everyone who had a hand in getting this book completed.

Introduction

Second Edition Introduction: Mark Bell

Welcome to *LEGO MINDSTORMS EV3: The Mayan Adventure*. Echoing Jim Kelly's original introduction, since you're holding this book, I'd surmise that you are either an owner of the new LEGO MINDSTORMS EV3 kit, someone who knows of the earlier book, or a student using *The Mayan Adventure* in a class. The First Edition became justly popular for the reasons Jim explains. I always use it in my summer *Introduction to Robotics* classes with Johns Hopkins CTY (Center for Talented Youth). I enjoy using the book since it offers clear instructions and a quality engineering process/rationale for building. As we put the book to work, I sit on the floor along with the teaching assistant and have the 14 or so boys and girls "circle up." I then read the archeologist narrative. Out loud. Shortly after that, students race up to us with their completed Design Journal pages as engineering proposals. We play the role of Engineering Management, signing off so they can begin their projects.

You can see many of these student designs and share with this book's user community at `http://ninevolt.ninja`.

This edition contains all five of the bots from the original edition, fully updated to EV3. Importantly, Jim Kelly's story line is preserved. Now, let's make a distinction here: the difference between *what* a bot does and *how* it does it. Since my five bots support Jim's story, they perform the same "*what* they do." But *how* shall they do it? These are new designs, leveraging Jim's, and taking advantage of new EV3 features. And, added to this edition are a dozen *Engineering* sidebars with explanations of real-world engineering principles and processes.

These examples are built with the Retail Kit, as they were in the First Edition. Users of the Educational Kit will nevertheless find this book a worthwhile candidate for classwork, particularly because of the emphasis on how engineers work. (Students with the Education Kit will sometimes need to substitute parts, particularly sensors.) And now, on to Jim's description of what this book is about!

First Edition Introduction: Jim Kelly

Welcome to *LEGO MINDSTORMS EV3: The Mayan Adventure*. I don't like to make assumptions, but since you're holding this book, I'm guessing that you are either an owner of the new LEGO MINDSTORMS EV3 robotics kit or are interested in the robotics kit and what can be done with it. (Or maybe you just thought the cover looked interesting and were wondering what robots have to do with the ancient Mayan civilization.)

This book is fairly unique, and I'll tell you why. For the earlier version of MINDSTORMS (called MINDSTORMS Robotics Invention System, or RIS for short), numerous books were written, most of which focus on building rather extravagant robots, hacking the MINDSTORMS processor (called the Brick), and doing other wild things with the product. And the books are great! Many of them show you, step by step, how to build and program very unique creations. But after reading them, I felt that a few things were missing.

The first thing I noticed was a minimal amount of "where to start" type information. The first time you open a MINDSTORMS robotics kit, you might feel a little overwhelmed at the sheer number of pieces (almost *all* of them small) in the box. You get an instruction manual and some sample robots to build, but there is very little information for those new designers who are asking, "How do I start designing a robot?"

The second item I found lacking was incentive. There are lots of robots you can build, but many MINDSTORMS owners get stuck trying to come up with a problem to solve. "What should I build?" is a frequently asked question. There are robotics competitions, with fixed tasks to complete and well-defined conditions for winning, but what if you're not into competitions or lack access to them? Where can a person find challenges to take on and accomplish?

The last gap involves training. Many of the books on the market for the RIS are great at telling you how to build and program your robots, but many times the explanations aren't really explanations—they're instructions: "Put this piece here" and "Drop that there." But the reasons for doing something (or, at least, the authors' reasons) are often missing.

With *The Mayan Adventure*, I've tried to fill in these gaps as follows:

- To answer the question, "How do I start designing a robot?", I've provided something called a Design Journal page. This is a worksheet that I use (and encourage you to use) to demonstrate the development of the book's robots, using a step-by-step method. It's not the only method out there, but it's my hope that you will find it useful as a way to keep your thoughts organized and to help you move forward in a constructive way.
- As for lack of incentive, I've divided the book into five sections, each of which involves a challenge. Each section is part of a fictional storyline that sets up a *reason* for building a robot. The story is fictional, but the challenges give you plenty of encouragement to experiment and develop your own robots.
- And when it comes to training, I provide solutions to the five challenges by walking you through the development of my robots, their construction, and their programming. I give you some "do this" and "do that," but always with explanations.

I don't use a lot of fancy, technical terms. There are some in there (it's unavoidable when dealing with programming), but I think you'll find that the book is written in an easy-to-follow way and, hopefully, you'll also find the process fun.

If you're completely unfamiliar with EV3, you should install the EV3-G software (available as a download from LEGO) and go through the included tutorials. These tutorials will teach you the basics of how to use the software as well as give you some basic construction skills. To get the most out of this book, you do need to at least be comfortable with using the EV3 components, opening the EV3-G software, creating and saving programs, and uploading programs to your robot. If you're comfortable with this short list, then you're almost ready to start.

How This Book Is Organized

As I mentioned earlier, I've divided the book into five sections. Each section is further divided into four chapters. The fictional storyline starts in Chapter 1, continues in Chapters 5, 9, 13, and 17, and concludes in Chapter 21. The storyline is where you find the details of a particular challenge (for that section); these details are important because they help you determine the robot's objectives.

Chapters 2, 6, 10, 14, and 18 are what I call the "theory" chapters. Don't let that word scare you, though. When I say theory chapters, I simply mean that these chapters give you plenty to think about—what does the robot need to do, what can it *not* do, what parts should be used, and what parts should *not* be used. I use the Design Journal page in these theory chapters, and I've provided five blank copies in the back of the book for you to follow along with me (or you can use them to develop your own robots).

Chapters 3, 7, 11, 15, and 19 are the building instructions for the robots. In each chapter, you'll find a set of photos that walks you through building my version of the robot. You can follow my steps and build the exact same robots I include in the book, or you can come up with your own creations. (If you find you're missing a part or something just doesn't snap together properly, the best part about LEGO robots is that

there's always a workaround—another way to connect something or a combination of parts that can be used as a substitute. Don't stress about it—use your creativity and find an alternative solution!)

Chapters 4, 8, 12, 16, and 20 provide the programming instructions. I use *plenty* of screenshots to show you how to configure each block that is used in the NXT-G programming language. If you have used my building instructions, you can also use the programming instructions. These chapters also include instructions for how to set up a test environment for testing your robots and see if they can complete the challenges.

Finally, I've included some appendixes for you; references, instructions for documenting your own robots, and other stuff. Check them out.

Who Is This Book For?

It doesn't matter if you are 10 years old or 50, building robots is fun. This book is for everyone who wants to build some new MINDSTORMS EV3 robots and have fun. I don't expect you to be a programming guru—I'm certainly not. I also don't expect you to have advanced degrees in robotics, engineering, or computer science. Let's all remember that LEGO MINDSTORMS EV3 is, ultimately, a LEGO product. It's a TOY! It's supposed to be fun, not stressful.

If you're a kid, this book can be a great way to get your mom or dad interested in your hobby. And if you're a parent, this book can be a great way to have some fun with your kids. I think you'll see that it's fun to create challenges for yourself (or someone else) and then try to build some great robots to overcome those challenges.

What You Need to Use This Book

The only things you need besides this book are a LEGO MINDSTORMS EV3 robotics kit and a computer to run the software and upload programs to your robots. There are currently two versions of the MINDSTORMS EV3 kit—the retail version that you can buy online or in stores, and the education version that LEGO sells to teachers, schools, and individuals. There are differences in the types of parts that come in the two kit versions, so please be aware that all the robots in this book have been built with the retail version. If you own the education version, that's okay. It just means that if you find I'm using a part that you don't have, you'll have to improvise. Don't let that bother you—just look at it as another challenge to overcome and something new to learn.

Extras for This Book

Extra Design Journal pages can be downloaded from the Source Code page on the Apress web site at `http://www.apress.com`.

CHAPTER 1

Tomb, Trap, and Trigger

Location: Southwest Guatemala

85 miles SW of Guatemala City

Coordinates: 14° 02' N/90° 42' W

Weather Conditions: 94° Fahrenheit, Humidity 40%

Day 2: King Ixtua Tomb Excavation, 4:42 PM

Evan leaned against a large stone at the base of the pyramid and sprayed more insect repellent on his left arm. The smell was horrible, but unlike the sunscreen, at least it worked. Evan wasn't sure which was more burned, his nose or his ears. He couldn't wait to get inside the pyramid and the shade it offered. The pyramid and the base camp were completely covered by the jungle and not visible from above, but the sunlight still managed to find its way through the leaves and branches and heat the air.

"A slight problem here," said Uncle Phillip as he walked away from the large stone entry door to the Mayan tomb. He continued walking across the camp, with his two assistants, Max and Grace, running to keep up.

Evan turned and ran to catch up with his uncle. "What's wrong?" he asked, almost running into two Guatemalan guides carrying a box of excavation equipment.

"Follow me, Evan, and I'll show you," Uncle Phillip replied as they continued walking toward the communications tent.

Dr. Phillip Hicks was the lead excavator for a newly discovered Mayan tomb, deep in the Guatemalan jungle. Evan's uncle was a professor of archaeology and taught at Florida State University, but he jumped at any chance he could find to leave the classroom and do some hands-on research. Two weeks ago Evan's parents had received a call from Uncle Phillip, asking if Evan would like to tag along; his parents had agreed to let him travel with his uncle for a few weeks to finish off his summer vacation. It would also be a nice break from Evan's younger twin brothers, Les and Wes.

As they entered the communications tent, Uncle Phillip threw his FSU cap on a nearby chair. Sitting next to the chair was a large, opened chest with numerous books and specialized equipment. Uncle Phillip was an expert in Mayan history, and earlier in the day he had shown Evan a picture of a Mayan glyph from one of the books. Uncle Phillip told Evan that the strangely drawn symbol represented King Ixtua. That same symbol was carved in stone above the tomb's entry door, confirming that the Mayan pyramid was the tomb of the ancient Mayan king.

Uncle Phillip began flipping maps on a large table, looking for something. "Where's the enlargement of the Tupaxu manuscript? That drawing makes sense now," he said.

One of the assistants, Max, was looking on a side table. Evan stood quietly, not wanting to interfere. The other assistant, Grace, began to dig through the chest of books.

M. Bell and J. F. Kelly, *LEGO® MINDSTORMS® EV3*, DOI 10.1007/978-1-4842-2262-1_1

"Have you ever heard the story of King Ixtua, Evan?" asked Uncle Phillip. He continued to shuffle maps and papers on the table.

Evan shook his head. "No, sir. My history grades aren't so hot. Sorry."

"That's okay," replied Uncle Phillip. "I didn't really become interested in history until college. But I think you'll like this story."

Evan's last history grade had not been impressive; science and math were more to his liking. But when his uncle had told him that this pyramid was unopened and had been hidden for more than 700 years, Evan couldn't resist. He had packed his clothes, smartphone, laptop, and the new robotics kit his parents had given him for his birthday last month, and met his uncle at the airport. If the pyramid turned out to be one big boring rock, he'd have his music and could at least spend some time designing some robots to show his friends when he got home.

"This King Ixtua liked monkeys, you see," said Uncle Phillip as he continued to search through a smaller pile of maps and papers. "He had numerous spider monkeys that he trained to do tricks. The story tells us that King Ixtua had a pyramid built as his final resting place. To keep out tomb robbers and other unwelcome guests, he had the builders design the pyramid so only someone friendly to his monkeys could gain access to the tomb. A nice legend, huh?"

"Weird," Evan said, and then laughed. His uncle smiled at him and laughed, too.

"Here it is!" yelled Max, as he pulled a large sheet off the small table and walked over to Uncle Phillip. Evan watched as his uncle carefully placed the sheet in front of his team.

"Two years ago, Evan, one of my old professors on a dig found a Mayan manuscript in a sealed jar and gave it to me. I translated the writing and found it was written by Tupaxu, the king's pyramid builder. It gave a general description of the location of the pyramid, among other things," said Uncle Phillip. "Look at this," said Uncle Phillip, pointing his finger at a curious drawing on the sheet. (See Figure 1-1.)

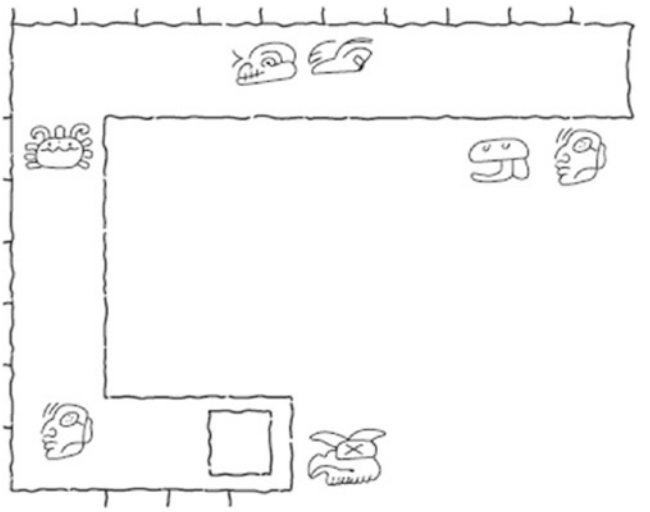


Figure 1-1. *Tunnel drawing*

Evan looked at the small drawing. It was surrounded by the strange Mayan writing he had seen on the various ruins in the camp.

Evan shook his head. "What is it?" he asked.

Grace pointed at the Mayan writing. "It's a drawing of a small tunnel. Your uncle was right about the monkeys, it seems," she said. "That drawing shows us how to unlock the stone entry doorway to the tomb."

Evan still didn't understand, and he frowned. "What are these symbols?" he asked, pointing at the small shapes.

"Measurements," answered his uncle. "These measurements translate to a tunnel entrance roughly 18 inches high by 18 inches wide. Too small for a person, but just the right size for a small spider monkey."

"But if you've found the door, why can't you just drill through it or knock it down?" asked Evan.

Uncle Phillip shook his head. "First, we don't destroy or damage any ruins. And second, the door has a trap that is disabled by a pressure switch. If the switch isn't pressed, the trap, whatever it is, will go off if we open or tamper with the door. Tupaxu was a very smart designer."

"So you just need to find this pressure switch and press it, right?" asked Evan.

"The first part is easy, Evan," said Uncle Phillip. "We found the pressure switch, but it's in a very bad location. Come on, I'll show you."

Tunnel Challenge

Evan pointed his flashlight down the tunnel. The bright beam ended about ten feet ahead, where the tunnel turned to the left and continued. (See Figure 1-2.)

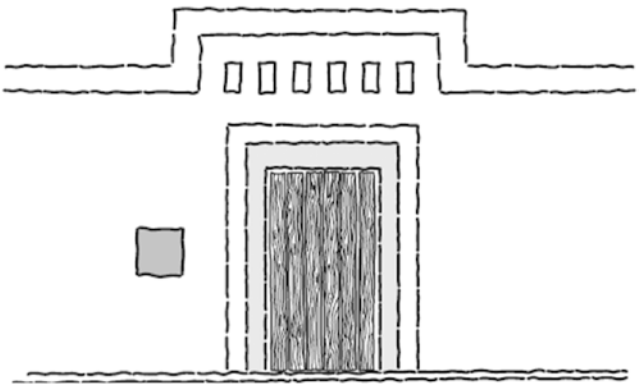


Figure 1-2. *Tunnel at tomb entry*

"According to the manuscript, the tunnel is about ten feet deep. It then goes left about six feet and then turns left again for another three feet. At the end of the tunnel is a small pressure plate," said Uncle Phillip. "If the pressure plate isn't triggered, we can't get in."

Grace was busy taking measurements of the tunnel with a long steel tape. She nodded and wrote in her notebook. "It definitely matches the dimensions of the drawing. It looks like you were right; a trained monkey would follow the path and step onto the pressure plate, triggering the release for the doorway," she said. "The legend of King Ixtua is true."

Uncle Philip shook his head. "Unfortunately, the story of the monkeys is also true. I don't think we have any trained monkeys in our tents. And Evan may be too big to send down the tunnel," he replied.

"What!" yelled Evan. "Are you serious?"

Uncle Phillip laughed. "Just kidding, Evan," he said. "We'll find another way."

Evan watched his uncle scratch his head and turn to walk back to camp. It appeared that the exploration of the tomb was at a standstill. Evan felt sorry for his uncle and the assistants, knowing they had spent so much time planning this expedition. It was hard to believe the solution to the problem was something as simple as a little monkey stepping on a pressure plate.

Uncle Phillip, Grace, and Max talked quietly as they walked back to camp. Evan looked down the small tunnel and shook his head. *If only we had a small trained monkey,* he thought.

And then the idea came to him.

"Wait!" Evan yelled and then spun to face the others. "I've got it!"

Evan's Solution

Back in the communications tent, Evan set a plastic yellow toolbox on the table. Next to it was his dad's old laptop, currently booting up. Evan opened the toolbox and reached in, pulling out a small rectangular object.

"This is the LEGO MINDSTORMS EV3 Intelligent Brick," he said. "This is the brains of any robot I build with this kit." Evan handed it to his uncle and continued pulling out various other objects. He watched as his uncle turned the Brick over in his hands and examined it closely.

"And these are sensors, motors, and other parts that are used to build a robot." He set a few of the items on the table in front of the team and then logged into the computer. (See Figure 1-3.)

Figure 1-3. *LEGO MINDSTORMS EV3 Intelligent Brick, motor, sensors, and other LEGO Technic components*

Max and Grace each picked up some of the components and examined them, and Uncle Phillip handed the Brick back to Evan. Evan set the Brick on the table and pointed at the computer screen.

"This software allows me to program the robot to do various tasks. It's a graphical language called EV3-G. I can tweak the software until I get the robot to do exactly what I want it to do. Pretty cool, isn't it?" he asked.

Uncle Phillip smiled and nodded. "Are you telling me that you can build a small robot with this stuff that can go down that tunnel and trigger the pressure plate?"

Evan smiled. "Yep. And I don't think it will take me that long, either," he replied. He pulled out a brown notebook from the laptop case and opened it. He had been playing with the LEGO MINDSTORMS EV3 kit for about a month and had plenty of notes and comments written in his notebook.

Max handed his component to Evan and pointed at it. "What does that do?" he asked.

"That's a servo motor. It does a lot of different things, but I use it mainly to give a robot wheels to move. Grace is holding the Touch Sensor," Evan said, pointing at a smaller block that Grace was examining.

Uncle Phillip pulled a chair over to the table and sat down with Evan. He looked over all the parts Evan was placing on the table and nodded.

"This might work," Uncle Phillip said. "How much time do you think you'll need?"

"Well, I'll need to do some planning first, mainly to figure out the best parts to use. The actual building and programming will take some time, too. I'm guessing three or four hours," Evan replied.

Max and Grace looked at Professor Hicks, waiting for his decision.

Evan's uncle looked at his watch. "It's almost dinner time, and the sun will be down in a few hours. There's really no point in trying to open the tomb tonight. If you can get the robot working, we'll let you send it down the tunnel tomorrow," he said. "Is there anything else you need, Evan?"

Evan thought for a moment, looking at the robotics kit in front of him and all the components.

"I just need some time to work through my design notebook. I'll start building and programming after dinner," he said. "This'll be fun."

Uncle Phillip smiled at Evan. "All right, this sounds like a good plan," he said, and stood up. "Dinner is in 40 minutes."

Evan watched as Uncle Phillip, Grace, and Max left the tent, and then he took a deep breath. "Time to get started," he said.

Story continues in Chapter 5 . . .

CHAPTER 2

ExploroBot: Planning and Design

In this chapter you're going to learn (drum-roll, please—cue the announcer) . . .

A PLANNING AND DESIGN PROCESS!

Please don't let the words scare you. Yes, "planning and design process" could sound boring, but I promise that you'll have fun with this chapter. I know you're ready to start putting pieces together to build a bot, but if you take some time and go through these P&D chapters, you'll be building and programming your own robots in less time, with fewer mistakes.

So, let's get started. That tomb door is still locked, and you're going to need the ExploroBot to open it.

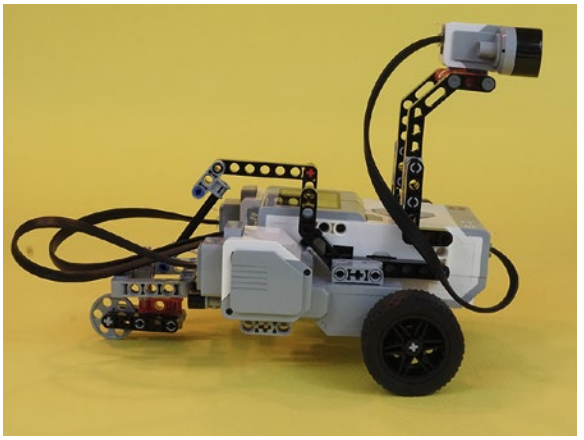
The ExploroBot

Do you have a picture of the ExploroBot in your mind already? If so, I'll bet that it doesn't look exactly like the one pictured here (see Figure 2-1, a-c). (If it does, you are an amazing mind reader. Call me—we can make a fortune on the stock market.)

Note There are five blank Design Journal pages in the back of this book that you can cut out. You're going to use them to design robots, both from this book and robots of your own design! If you need more pages, you can find a file titled `DesignJournal.pdf` in the Source Code package available for download on GitHub through the Apress web site (`www.apress.com`).

At the top of the Design Journal page you'll see the words `Robot Name`. Go ahead and write **ExploroBot** in the box and pat yourself on the back. The PLANNING AND DESIGN PROCESS has begun. (You could write something else, such as **RobotThatOpensTombDoors**, but you might run out of space.)

M. Bell and J. F. Kelly, *LEGO® MINDSTORMS® EV3*, DOI 10.1007/978-1-4842-2262-1_2



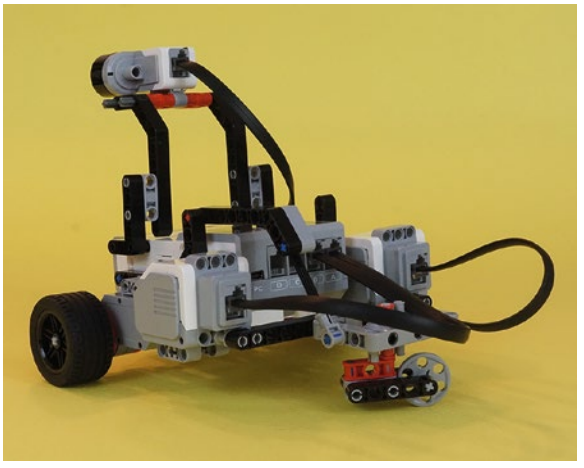


Figure 2-1a-c. *The ExploroBot from three different angles*

Are you wondering how that little robot is going to open the tomb door? Good question. And you're going to answer that question by following along using a page out of Evan's Design Journal.

The Robot Description

Okay, now that you've named your robot, it's time to describe it. No, I'm not talking about "Short, grey and white, with wheels." What I mean is, what is this robot supposed to do? At this point, I hope you've read Chapter 1. If not, I'll wait . . . Go back and read it. Okay, have you finished it? Good. Now, what is this robot supposed to do? Don't say it, write it.

Look on your Design Journal page, and you'll see `Robot Description` and a large blank box. Don't be shy here. This is where you're going to try your hardest to describe accurately what this robot will do for you. Look back to Figure 1-2 in Chapter 1 if you need a reminder about the path the robot needs to follow. Let me show you what I wrote down, and you can compare it to your description, okay? Here goes (see Figure 2-2).

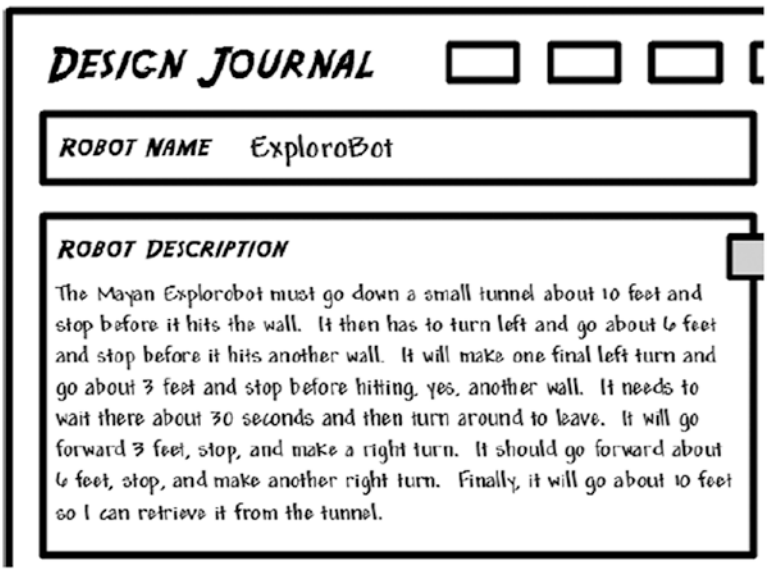


Figure 2-2. *Robot description*

If your description isn't exactly like mine, that's okay. What *is* important is that you got the major points: Robot moves forward about ten feet, stops, turns. Robot moves six feet, stops, turns. And on and on. Trust me—without an accurate description of the robot, it will be more difficult to build (Chapter 3) and program (Chapter 4). Don't worry if your description missed something; you'll get better at this, I promise. You're going to have more opportunities to write robot descriptions later in the book. By the time you're finished, you'll be an expert.

So, what's next, you ask? Okay, I'll tell you—you're going to take the description you wrote and break it down into small, single-item tasks.

The Task List

On your Design Journal page, locate the `Task List` section. This section is where you're going to list each individual task that the robot must perform. The *good news* is that if you wrote down a detailed description (see the previous section), then this section is almost already done.

What do I mean by "individual task?" An individual task is something like "Walk forward five feet" or "Turn doorknob." Something like "Press the button and turn the wheel" is not an individual task. Your goal is to list the actions your robot will perform, one at a time. Take a look at my task list (see Figure 2-3).

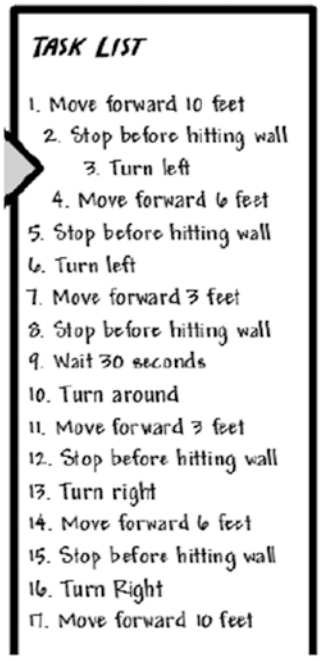


Figure 2-3. *The ExploroBot task list*

Compare your task list to mine. Were you able to break down the robot description into individual tasks? These individual tasks will help you in many ways, including assembling the correct form for your bot, picking the appropriate sensors to be used, and later when programming the bot.

I'll give you a small preview of how we'll use the task list later. Look at Steps 2, 5, 8, 12, and 15—"Stop before hitting wall." Are you already thinking about how to do this? You've got options, of course. There's the Touch Sensor that can be programmed to stop the robot when it's triggered. And what about the Infrared Sensor? That sensor sends out a beam of infrared light that's detected when it bounces back off an object in front of it, such as our wall. It can sense distance from that reflected beam. So you can see that this task list will help you to start thinking about the EV3 components you'll use. For now, let's leave the `Task List` and move on to the next section of the Design Journal.

Limitations and Constraints

You're going to encounter one obstacle quickly when you begin to design your robots using the LEGO MINDSTORMS EV3 kit. What is it? It's the number of parts in your kit.

It would be nice if you had access to an unlimited number of sensors, motors, connectors, beams, and other components. But for this book, and the robot designs included in it, I'm assuming that you have the LEGO MINDSTORMS EV3 retail kit. All of the software will work if you have the LEGO MINDSTORMS EV3 education version of the kit, but there are different parts. For example, the education kit uses an Ultrasonic sensor to sense distance. And the education kit comes with a rechargeable battery pack, which is thicker than the flat-bottom battery compartment in the retail kit. That will affect the ExploroBot and PushBot, among others.

When you begin to design your robots, you need to be aware of limitations (or constraints) such as this one. Limitations and/or constraints can come from many different places. Besides the number of parts in your robotics kit, you need to keep in mind issues such as the following:

- Robot size and weight (tall, short, heavy, light, wide, thin, square, circular)
- Weather and lighting conditions (outdoors, indoors, artificial light, no light)
- Floor or surface conditions (soft, hard, wet, slippery, and so on)
- Movement requirements (up, down, left, right, forward, backward, diagonal)

There may be some constraints that you won't encounter until you begin building and testing your robots. Don't worry if this happens. Your goal at this point should be to write down any limitations that come immediately to mind. Just look back at your `Robot Description` and `Task List` and the environment or objects where the robot will interact. Do any constraints come to mind? Write them down on your Design Journal page in the `Limitations/Constraints` area.

Take a look at Figure 2-4. I've written down a couple of sentences that describe what I think are some major constraints for the ExploroBot. Remember, there may be other constraints that won't show up until we begin testing our design. The important part is to try to identify any obvious constraints before you begin to design and build your bot.

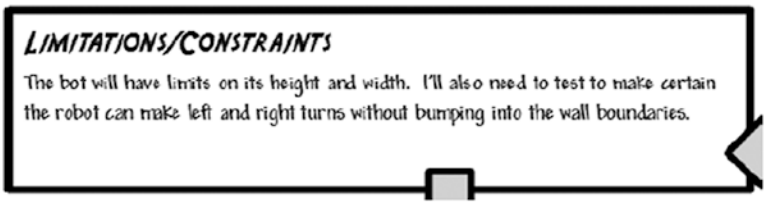


Figure 2-4. *The ExploroBot has a few constraints to consider*

The constraints for the ExploroBot aren't too difficult to work around. Let's take a look at the challenge and see how these constraints will affect the robot design.

First, the robot will enter a tunnel that has a fixed height and width. So the robot you build cannot be too wide or too tall or it simply won't fit into the tunnel. We know the measurement of the tunnel is 18 inches tall and 18 inches wide. We'll keep that in mind when we begin to design.

The second constraint is a little trickier. Take a look at Figure 2-5. This is an overhead view of the tunnel and its dimensions. Pay attention to the two corners where the robot will turn.

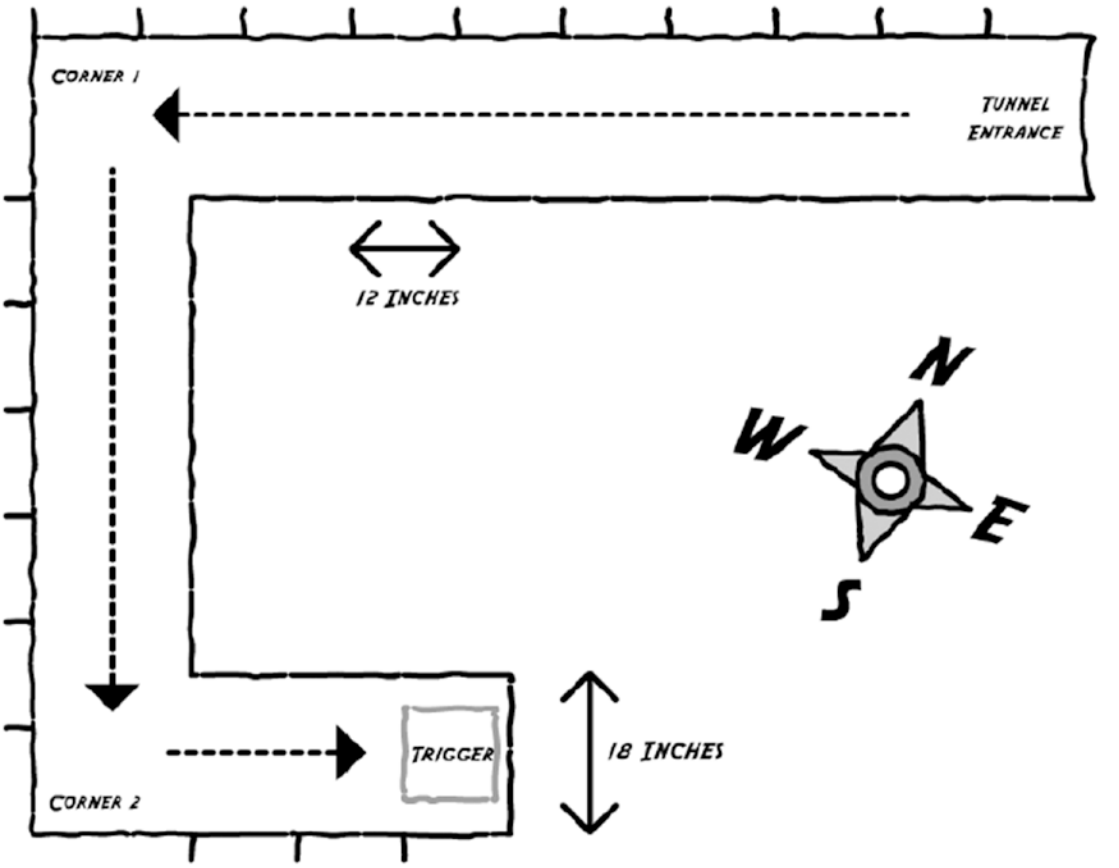


Figure 2-5. *The ExploroBot has to turn a few corners to land on the trigger*

The robot must stop before it hits the wall at Corner 1. When it stops, it will turn left and proceed to Corner 2. When it's turning, we'll have to be careful to give the robot sufficient space to turn and not bump into the wall. There are numerous methods for building and programming a robot to make a right-angle turn, and I encourage you to experiment with other methods.

So, how can you do this right-angle (or 90-degree) turn and give the robot plenty of room to avoid bumping the walls? Glad you asked.

If you look at Figure 2-6, you'll notice I've zoomed in on the first corner and included some measurements, including the length and width of the ExploroBot. Ideally, we would like the robot to stop a certain distance from the wall and turn left, and the best place for the robot to stop would be directly in the middle of the corner. I don't want this to get too complicated, so just keep in mind that when the robot turns, it cannot be too close to the wall or, when it turns, it will bump the wall with its front right wheel. So be aware that during the building and programming of the ExploroBot, we'll be "tinkering" and "tweaking" to get the bot to perform well in a corner.

■ **Note** During the building and programming of the bot, you'll perform many tests. During this phase, you'll test many of the bot's functions—forward speed, stopping speed, detecting the wall, stopping at a proper distance, and more. I'll cover this in more detail in Chapters 3 and 4.

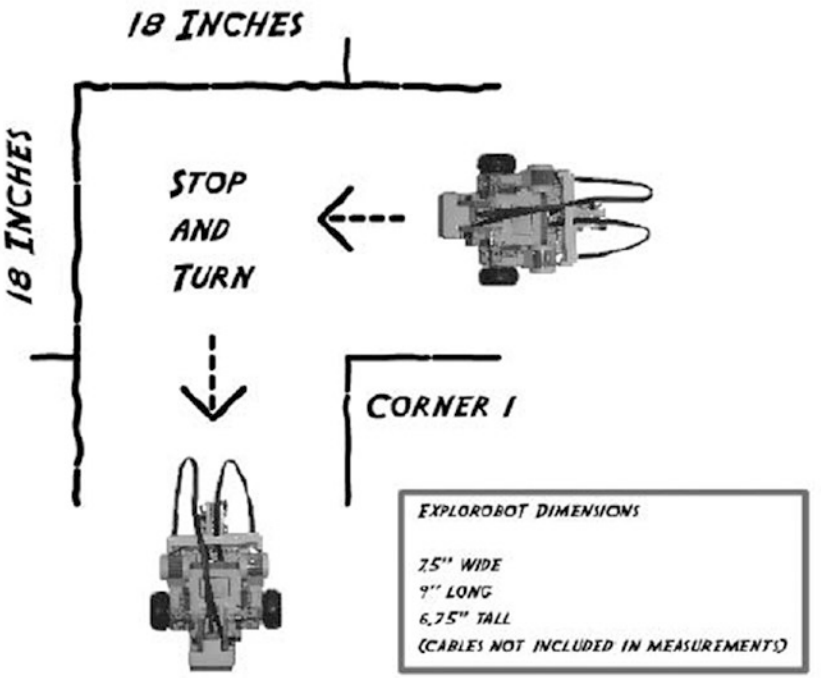


Figure 2-6. *The ExploroBot should turn while centered in the corner*

The final constraint isn't so much a constraint as it is a condition that might affect the bot. The surface of the tunnel is stone. It's a flat rough surface, not made of sand. I'm including this constraint only to demonstrate that you must always be aware of the external conditions the robot will face. Because the surface is flat and rough, we should be able to use the large rubber wheels to move the robot, because they'll have a good grip on the surface of the tunnel. But this might not always be possible. A wet surface can sometimes cause plastic or thin wheels to simply spin without getting traction, keeping the robot from moving. And what if the robot doesn't have a surface to roll or walk across? I'll answer that question in Chapter 6.

Just try to keep an open mind when you're thinking about the obstacles your robot will face. Examine the robot's environment, its tasks, and its overall goal as you start to brainstorm about how you'll solve the problem. And that's what you're going to do next. You're going to brainstorm about this bot's design, components, and overall appearance in this next Design Journal section—`Mindstorm`.

Mindstorm

Convenient name for this section, huh? The LEGO MINDSTORMS EV3 robotics kit uses that unique word, *Mindstorm*. For us, to mindstorm (or brainstorm) is our chance to use our creativity and start developing ideas for how this bot will be designed and built. This is an easy section to complete. What I want you to do is simply write down your questions, observations, and ideas that have been popping into your head since you became aware of the challenge. There are no incorrect items to place in this section except for sketches—those come last. So, to get you started, take a look at Figure 2-7. You'll see some of my initial thoughts on this challenge, this bot, and the direction I want to take for my initial design.

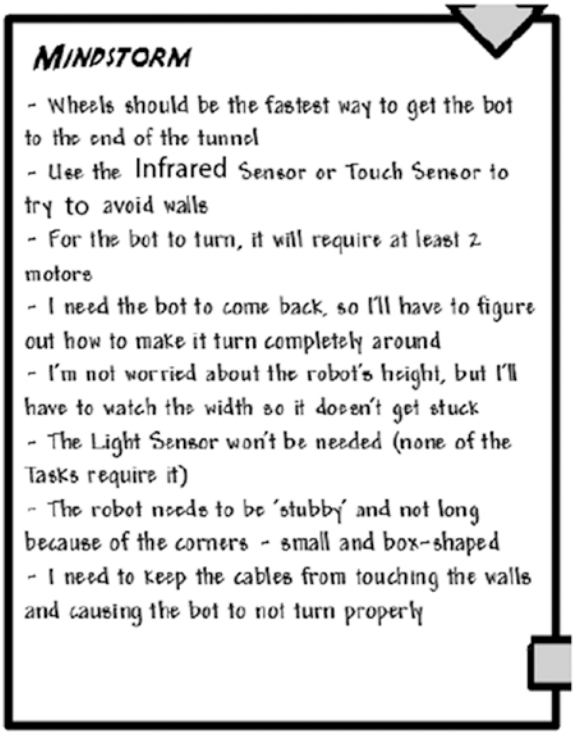


Figure 2-7. *The Mindstorm section contains my initial thoughts*

I'm not going to cover all my `Mindstorm` items here, but I would like to mention a couple and explain how and why I wrote them.

One of my observations was "For the bot to turn, it will require at least 2 motors." This might seem like common sense, but then again, maybe not. In order for a bot to turn, it has to have a force that makes it turn. Two wheels, connected to a single motor, only give forward and backward motion. For the bot to turn left and right, it requires another motor. By spinning one motor (and its wheel) in one direction and spinning the other motor in the reverse direction, you can cause the bot to turn. This can also be accomplished simply by locking the second motor in place and keeping it from spinning. One wheel will spin, the other wheel will not spin, and the bot will turn.

Another observation was "Sound-control would require Sound sensor and add complexity—should avoid it." One of my initial ideas was to control the bot by using different sounds and tones. One tone would stop the bot, another tone would make it turn right, and yet another would make it turn left. Then it occurred to me that this would be just too much trouble. My goal is to make the bot as independent as possible and allow it to find its way down the tunnel and back. So the Sound sensor was eliminated.

Your objective here is simply to have some fun and write down some of your initial thoughts on what you'd like to do with your bot design. You might have to take a completely different direction after some testing. You might find you exhaust your supply of a particular component. What you write down isn't going to lock you in to a particular design. You can change the design anytime, and even start over completely. Print another Design Journal page and try a different design. It's supposed to be fun, so make it fun. Go crazy with your ideas—the crazier, the better!

Now you're done with the `Robot Description` and the `Task List` is full. You've identified some `Limitations/Constraints` and your `Mindstorm` items section is overflowing with your thoughts and observations. It's now time to finish up with the Design Journal's final section—`Sketches`.

Sketches

When I draw stick figure people, they tend to have very short legs and very long arms. I still color outside the lines with crayons. I guess what I'm trying to say is that if you're a professional artist, you don't have to worry about any competition from me.

I can, however, draw shapes that are fairly close to squares, circles, rectangles, and triangles. And that's good enough for what I'm going to ask you to do in this section. I want to give you some suggestions before starting on the building of your bot, and I'm going to take my own advice and show you my actual sketches for the ExploroBot.

I'm going to reference some of the ideas I wrote down in the `Mindstorm` section and show you how I came up with the size and shape of the ExploroBot. First, I'll start with the shape. Take a look at Figure 2-8 and notice that I started with a basic shape to help determine the placement of sensors, motors, and other parts.

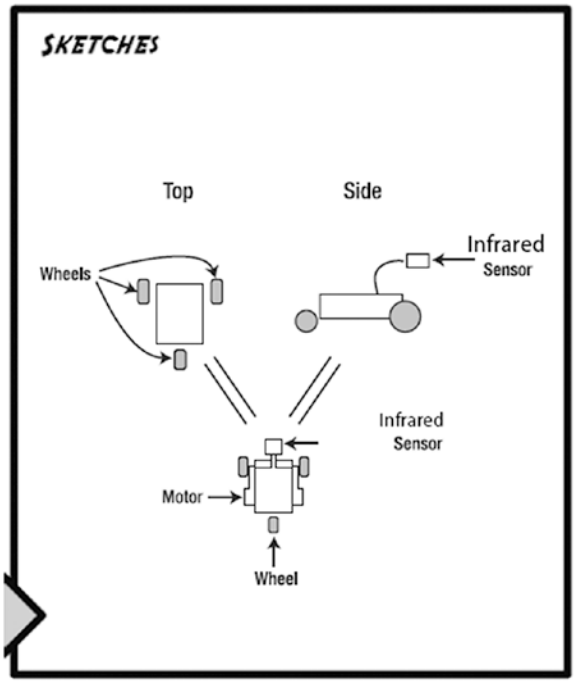


Figure 2-8. *In the Sketches section, try to start with placing basic shapes*

First, I need to decide between the Touch Sensor and the Infrared Sensor for detecting an approaching wall or obstacle. If I use the Touch Sensor, it will need to be placed far in front of the bot, possibly on a long neck or pole, to allow it time to stop the bot and give it room to turn. But if I use the Infrared Sensor I can place it closer to the bot's body because it can detect a wall or an obstacle from a distance and it doesn't require an impact with the wall or obstacle. Because my goal is to keep the ExploroBot as short in length as possible, I'm going to use the Infrared Sensor.

My ExploroBot will require two motors (for turning), the Infrared Sensor, and the Intelligent Brick. And unless I want the Brick to scrape the ground while it moves, I'll need to give it one or two extra wheels. I think I'll try to save some weight and keep the size down by using only three wheels.

What I'm envisioning is using the Intelligent Brick as the main body. Two large motors, one on each side of the Brick, will spin the two wheels used for forward and backward motion and for turning. I'll configure a small third wheel that will pivot to make the bot's turning a little smoother and give less resistance.

What do your sketches look like? Did you take a different approach to the design of the ExploroBot? Remember, there is no *right* or *wrong* solution. If your ExploroBot reaches the end of the tunnel, lands on the pressure plate, and then returns to the tunnel entrance, you've succeeded in opening the tomb door.

In Chapter 3, I'm going to walk you through the assembly of my version of the ExploroBot. Feel free to change it up! Move the Infrared Sensor to the back or simply change it to the Touch Sensor. Try giving it four wheels instead of three. Chapter 4 will show you how to program the ExploroBot; at the end of the chapter, I'll also give you some ideas on how to set up the challenge and test your bot.

Now, let's build the ExploroBot!

CHAPTER 3

ExploroBot: Build It

Before you begin building the ExploroBot, take a look at Figure 3-1. Remember, this is just one possible version of the ExploroBot. Some of you might choose to design and build your own version without going through this chapter, but for those who want to build the one pictured, I have a few suggestions.

Figure 3-1. *Evan's version of the ExploroBot*

Never Be Afraid to Experiment

I've included actual pictures of the construction of the ExploroBot. I've tried to provide enough detail in each figure for you to understand which parts are used and where those parts are placed. If you find that what you're holding in your hands doesn't quite look like the picture, do the following:

1. Take a deep breath.
2. Remember this is supposed to be fun.
3. Go back to the previous step and confirm you've made it that far.

M. Bell and J. F. Kelly, *LEGO® MINDSTORMS® EV3*, DOI 10.1007/978-1-4842-2262-1_3

4. Look at the current step and examine the figure's details for clues to where components should be placed. Counting holes to determine where two or more parts connect is useful. So is skipping ahead a few figures to try and see the parts from another angle.

5. When in doubt, take your best guess and move forward.

If you do hit a roadblock and just can't figure out how to proceed, try taking a look at the next few steps. Often, another figure will show the ExploroBot from a different angle and you'll see the solution. Enjoy the building process and realize that if your final bot doesn't look exactly like the one in this chapter, that's okay. Remember: getting the bot to work and solving the challenge is your main goal.

■ **Note** If you modify or create your own version of the ExploroBot (or any other bot in this book), please take a picture and e-mail it to me. I would enjoy seeing your final bot in action. I've included my web site in the Introduction.

And now, on to the construction of the ExploroBot!

Step by Step

I'm going to break the ExploroBot into four sections. The first section is the Infrared Sensor and the "neck" used to support it. The second section explains the body, including the two large motors and two wheels. The third section explains the rear-wheel framework and reinforcing strut. You'll completely assemble the ExploroBot in the fourth section using the head/neck, body/motors, frame/rear-wheel, and strut subcomponents. For many sections, I simply jump from figure to figure. I make comments where I feel an area might be tricky or where I feel you need to be made aware of a special assembly instruction. Pay particular attention to those figures showing the individual parts lying unassembled. These will help you determine what parts to locate for the assembly figures that follow. Here we go …

First Section: Infrared Sensor and Neck

Starting with Figure 3-2 and continuing through Figure 3-6, you'll be assembling the ExploroBot's head and neck. Take a look at Figures 3-2 and 3-3. They show the parts you'll be using in this section.

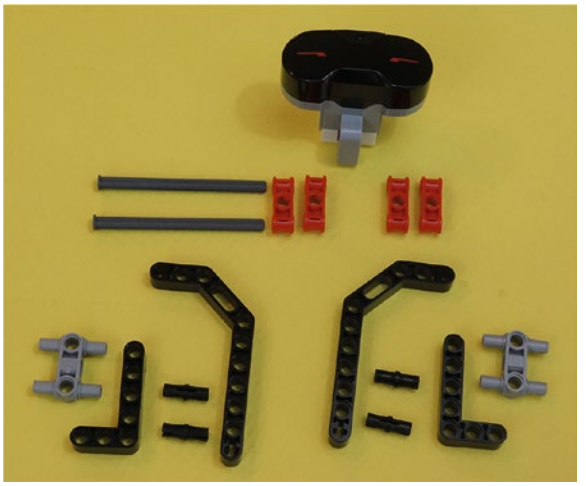


Figure 3-2. *These are all the parts you'll need for the ExploroBot's head and neck*

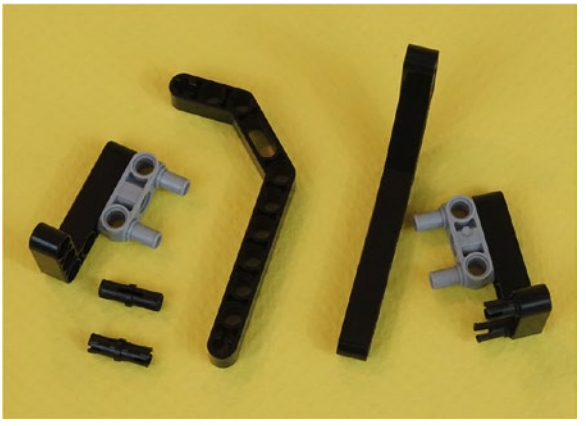


Figure 3-3. *Beginning assembly of the parts for the left and right sides of the neck that will hold the ExploroBot's head*

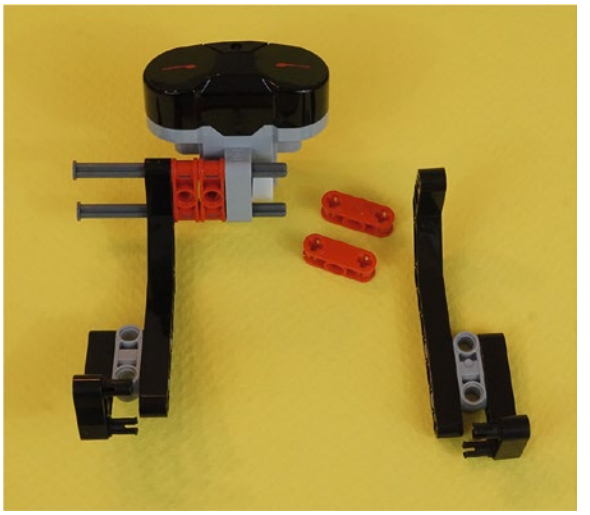


Figure 3-4. *Partial assembly of the head and neck, with long axles pushed in part way*

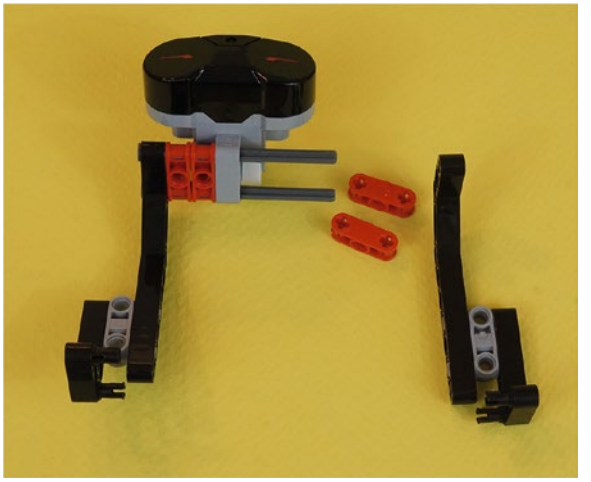


Figure 3-5. *Further assembly of the sides of the neck and head, with long axles driven all the way through the left side*

Figure 3-6. *The fully assembled ExploroBot head and neck—set this aside for now*

It's time for a quick check here. Take a look at your ExploroBot head and neck. Does it look like the one shown in Figure 3-6? If not, try to identify where the difference is and go back to the figures showing the assembly. If the difference is obvious, spend a few minutes looking at the parts and you'll find the mistake. When working with such small parts, it's not uncommon to connect a part in reverse or upside down. I do it all the time.

When you're satisfied with the ExploroBot head and neck assembly, set it aside for now. We'll come back to it after you've completed the main body, rear-wheel, and support strut sections.

Second Section: Bot Body and Motors

In this section, you're going to build the ExploroBot's main body and its two motor/wheel combinations. Take a quick look at Figures 3-7, 3-8, and Figure 3-9. These three figures show all of the parts you'll be using in this section.

Figure 3-7. *The body will consist of the Intelligent Brick, with motors and assemblies to be fastened to it*

Figure 3-8. *Side view of the Intelligent Brick*

In Figure 3-9, you can determine the length of the gray axle rod by comparing it to the length of the beam, up to the bend. In the figure, there are two full-thickness round axle bushings (red) and two half-thickness round axle bushings (yellow). These two half-thickness bushings are shown standing up so you can see the thin groove. They are used as spacers to keep the rubber wheels from touching the orange motor hub face. Figure 3-10 shows the first steps of assembly.

What I call a "round axle bushing" is also called by its official LEGO name: `Part# 3713 Technic Bush`. Why do I tell you this? Because I want you to be aware that LEGO gives a unique part number to every component it sells. (Also, because this is where I tell you that I usually avoid using the official part numbers.) If you're unable to determine the identity of a component from a figure, take a look at the next figure or two. You'll most likely be able to figure out what the component is by seeing it from a different angle or by seeing it where it's placed.

Let's build the base.

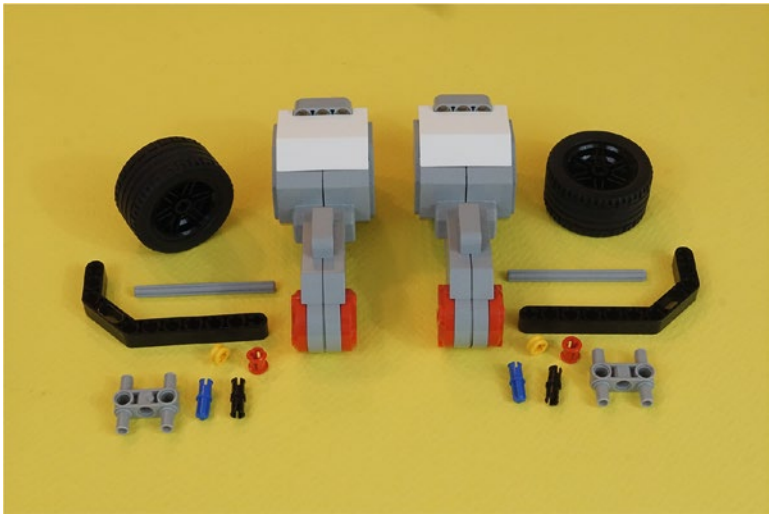


Figure 3-9. *All the components for the left and right side motor/wheel combinations*

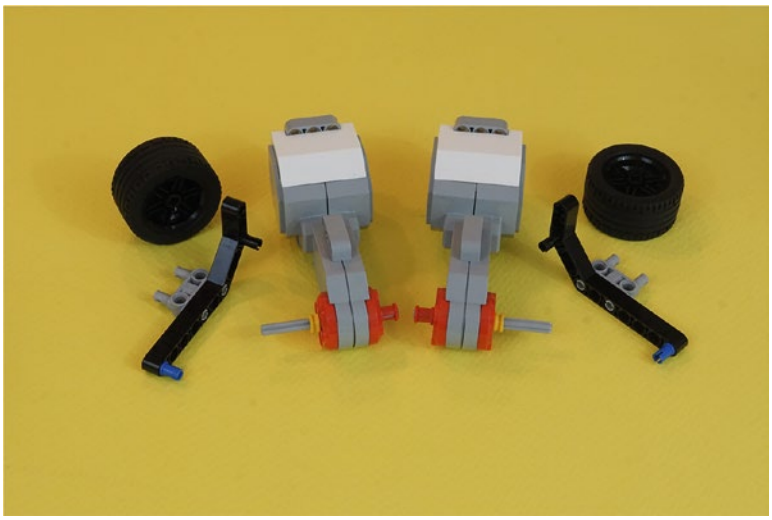


Figure 3-10. *Partial assembly of the left and right motor/wheel combinations*

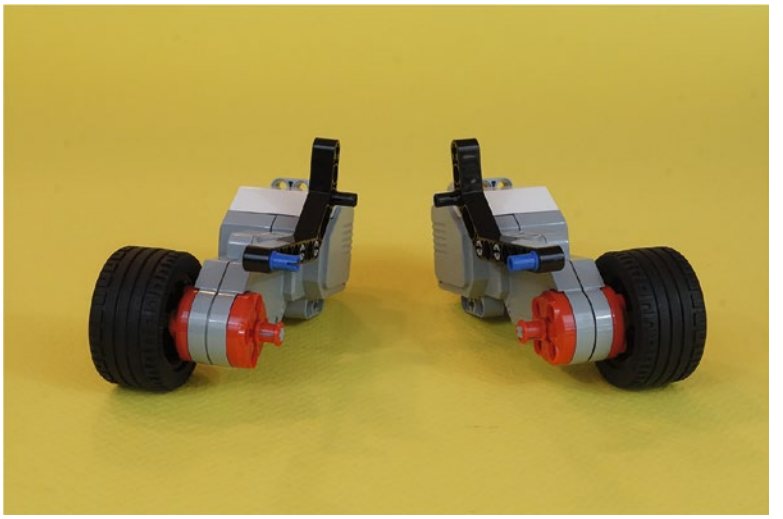


Figure 3-11. *The completed motor/wheel combinations. The long angle brackets have been positioned on the inside edge of the large motors.*

Figure 3-12. *The Intelligent Brick and the motor/wheel combinations, before putting them together*

Okay, it's time again to check your work. Compare your ExploroBot body with the one shown in Figure 3-13. You can use Figure 3-13 to compare things such as the location of the black connectors in the angle beams. This helps you confirm that you have the motor/wheel combinations located properly.

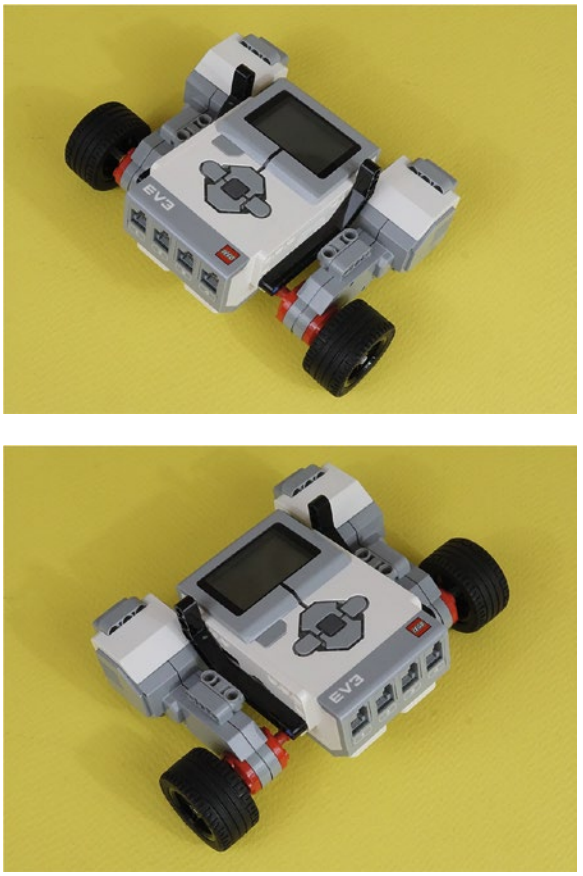


Figure 3-13. *So far, this robot is the same on both sides—it's symmetrical*

Again, the key here is to look for any major differences between your ExploroBot's body and the one shown in Figure 3-13. If you find any differences, just go back and examine the figures carefully to determine where the mistake appears to have been made. If necessary, take the assembly apart and start over. You have plenty of time and you're having fun, right?

Are you happy with the ExploroBot's body? Congratulations! You're half done with the ExploroBot. We're going to move on to the third section, which gives us a rear-wheel assembly. Just take your time on this section and enjoy the building process. Remember my advice—if you get stuck, just take a deep breath and remind yourself that this is fun. You can't make a mistake because you can always start over. No harm done!

Now let's finish the ExploroBot's rear-wheel section and then get this bot assembled…

Third Section: Rear-Wheel Assembly and Reinforcement Strut

There are two separate structures to build in this section. First you're going to build a small but quite useful rear-wheel assembly. You will probably use this design on other projects since it is a fine way to make a two-wheel robot that maneuvers well. You'll also make a base to hold the rear-wheel. This base has relatively few parts because it uses a rectangular frame piece, making it fairly strong by itself. Then you'll add a reinforcement strut, which further strengthens the bot and supports the rear-wheel. Your bot will work without the reinforcement strut, but it does make it more solid. You can see this in the Figure 3-37 side view. Notice that the rear would sag without the strut.

Take a quick look at Figures 3-14, 3-19, and 3-23. These figures show most of the parts you'll be using in this section, but in some later figures you'll notice I've added some new parts. My best advice is to pull out the parts you need in these three figures. You'll know when I've added some new parts because you'll have used all the previous ones you set aside! And again … when in doubt, just pause and examine the figures more closely. I've made sure to include enough detail for you to figure out what parts are being added to the mix.

Next, Figure 3-14 shows the parts that go in to the rear-wheel. The following figures show how to assemble the wheel.

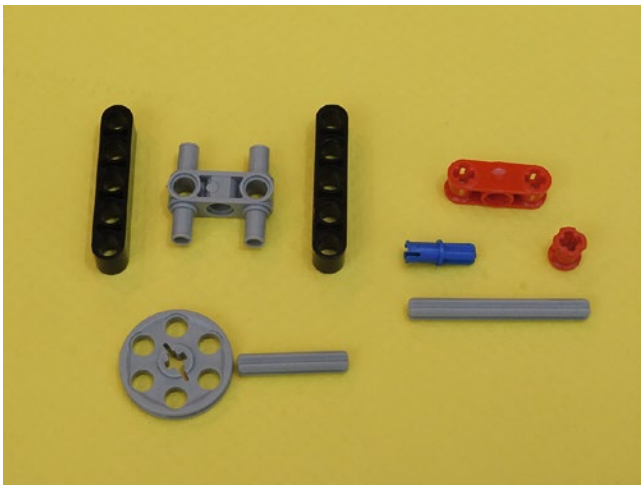


Figure 3-14. *These are the parts for beginning the rear-wheel*

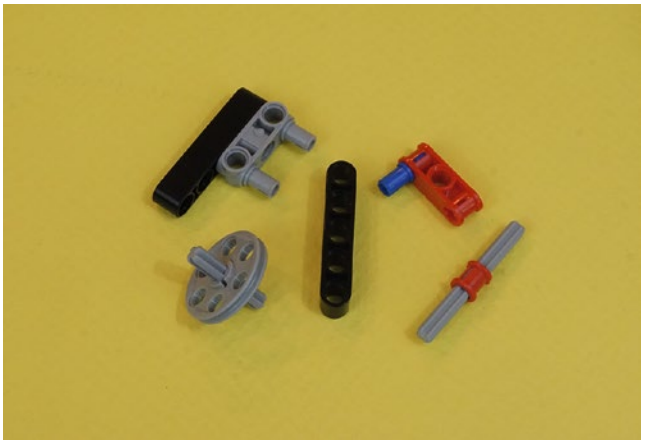


Figure 3-15. *It's not quite a rear-wheel yet, but keep going*

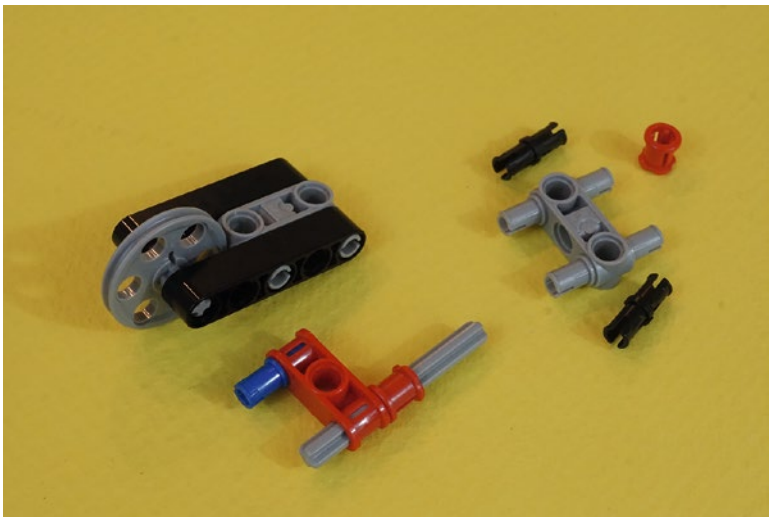


Figure 3-16. *The rear-wheel is starting to come together*

■ **Note** In Figure 3-16, you'll notice that I've added two new parts that weren't seen in Figure 3-14. This will happen again in later figures, so be on the lookout for it. As I mentioned, if you set out the parts that you need for each section, when I add parts like this to a figure, it will be obvious to you. "Hey, that's not in my parts pile!"

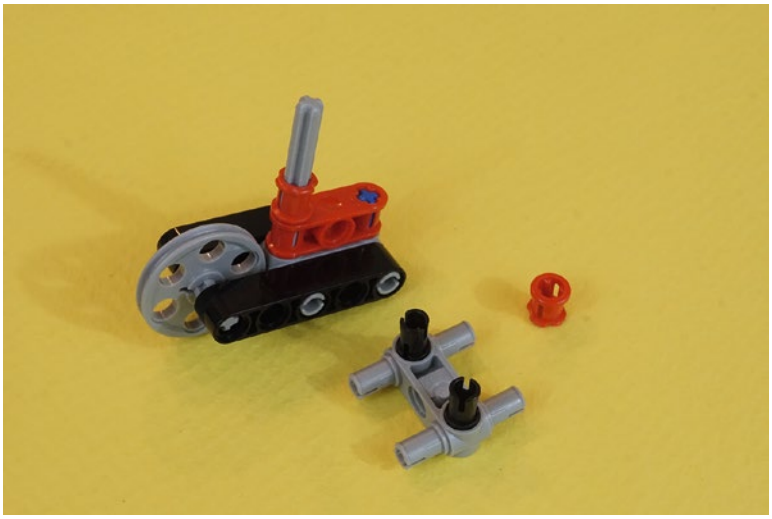


Figure 3-17. *The rear-wheel is almost done…*

Figure 3-18. *The complete rear-wheel assembly—set this aside for now*

This is a useful little assembly—low friction and quick to make a turn without changing the robot's path on its own. (You'll see this fellow again in Chapter 19.)

Finally, Figure 3-19 shows the parts needed to make the rear-wheel frame. Three black connectors have already been inserted into the rectangular component.

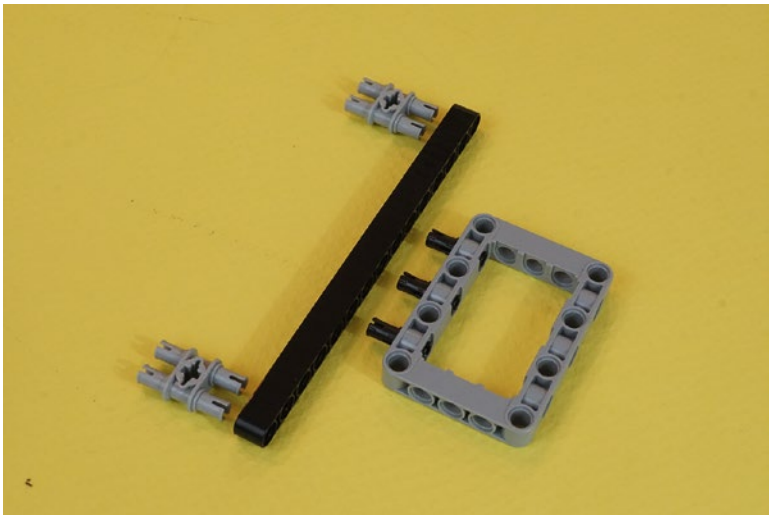


Figure 3-19. *Gather these parts to build the base frame that will hold the rear-wheel*

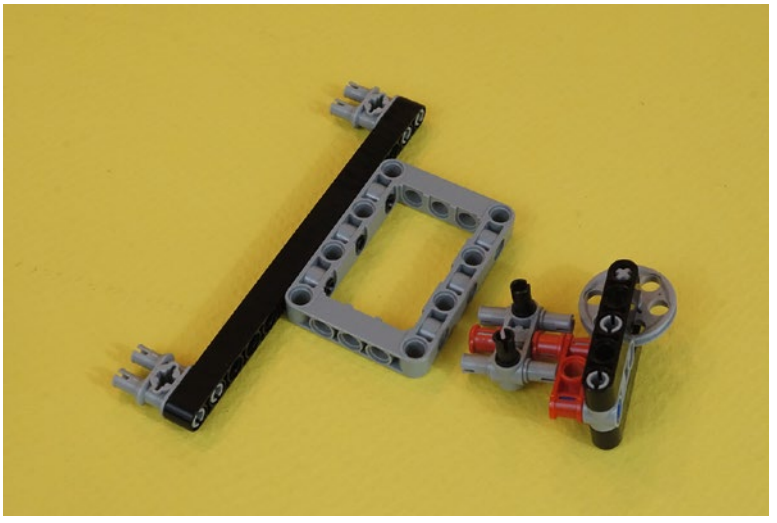


Figure 3-20. *Connect the parts as shown. The gray rectangular frame is actually off-center by one hole. This will position the wheel so it ends up centered.*

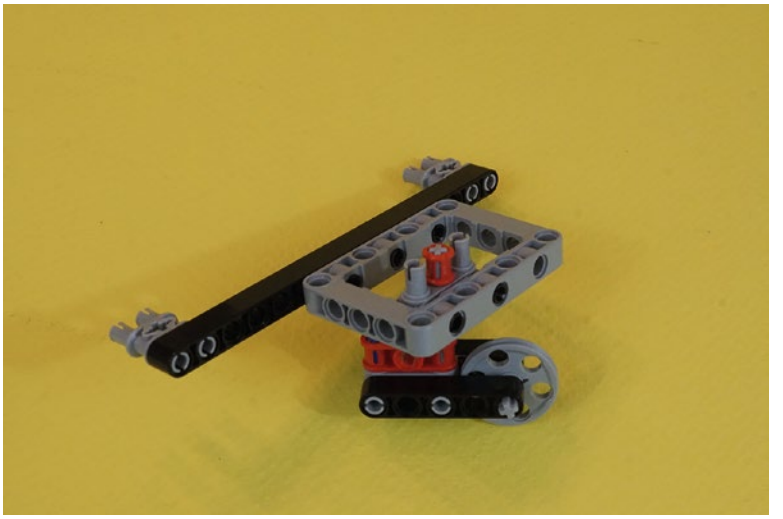


Figure 3-21. *Place the wheel assembly inside the rectangular frame piece*

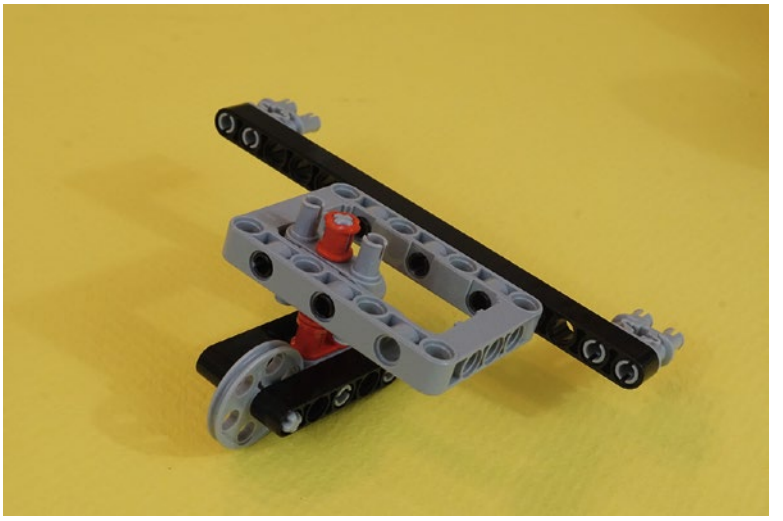


Figure 3-22. *Another view of the completed rear-wheel base assembly. Counting the beam holes on each side shows how the gray frame is off center on the black beam.*

Congratulations! You've finished the rear-wheel base. Do a quick check and compare the base you assembled to the one shown in Figures 3-21 and 3-22. The rectangular frame piece is offset by one hole from being centered, which allows the rear-wheel to end up right in the middle.

Engineering: Axles as Reinforcements: Strong and Adjustable

Now it's time for the reinforcement strut. Here are a couple of engineering ideas that will prove useful later. The first involves the strength of axles. In Figure 3-23, there are three short axle-like parts and one longer axle. Many times you will have a choice between short black connectors and the equally short blue half-axle/half-connectors. If you look on the end of the black angle piece, you'll see it has an axle-shaped hole. These connections are stronger than regular connectors.

■ **Tip** **Using Axles** When designing ways to hold things together, always remember to ask, "Will some kind of axle make this stronger? Or more adjustable? Can I use some of those odd axle connectors?"

The other engineering idea is that axle connections are adjustable. Beam connections are not. This means that you can use an axle where a beam would not quite fit. This pays off if your angle is not an even 45 or 90 degrees. You can also make an axle as tight as you like by sliding it along its connectors or adding more collars.

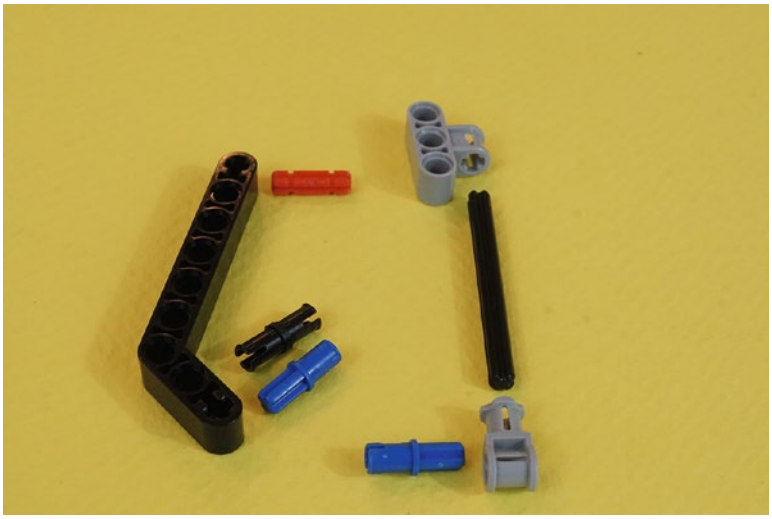


Figure 3-23. *These are all of the rear-wheel reinforcement strut parts*

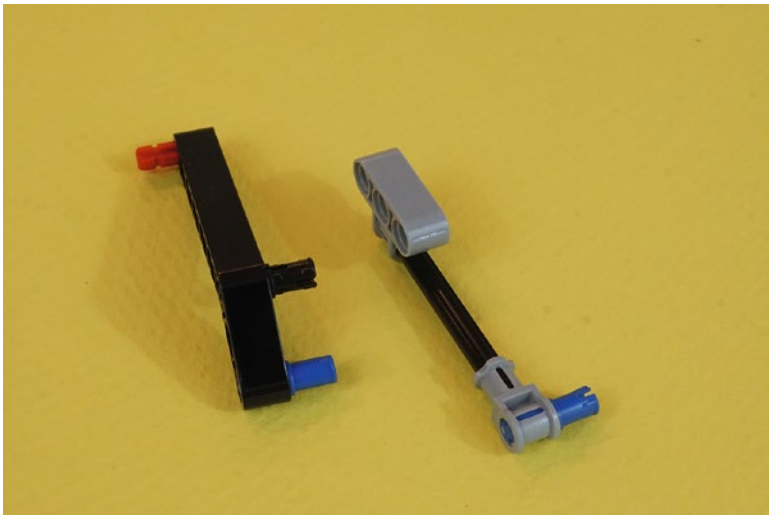


Figure 3-24. *The partially completed rear-wheel reinforcement strut*

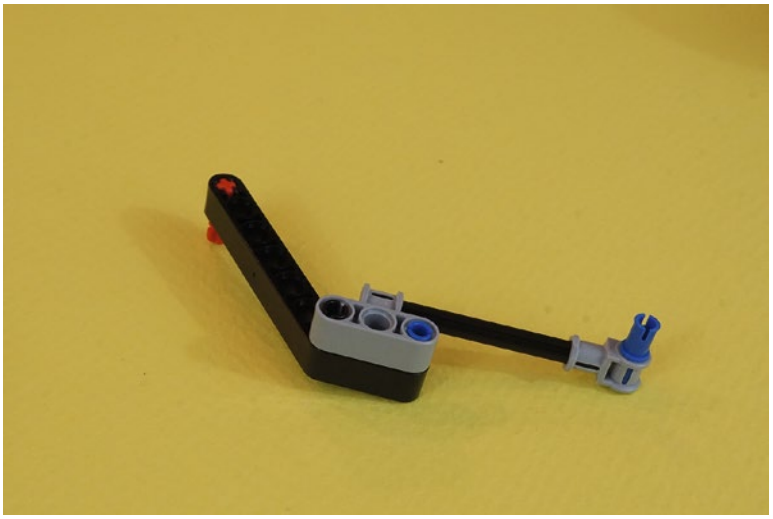


Figure 3-25. *The completed rear-wheel reinforcement strut with adjustable axle length. This controls the firmness of the strut when it is on the ExploroBot.*

Congratulations! With this reinforcement strut you've finished all of the sub-assemblies. Now it's time to pull all of them together.

You should have the following ready to go:

- Head and neck assembly
- Main body with motor/wheel combinations
- Rear-wheel base
- Rear-wheel reinforcement strut

Place them all within reach. It's time to finish building the ExploroBot.

Fourth Section: Put It All Together

Take the main body assembly and the rear-wheel/base and turn them upside down. You'll see this in Figure 3-26. Connecting the rear-wheel/base and the main body should be straightforward. Before attempting it, take a thoughtful look at Figures 3-26 and 3-27. This is also a good place to mention that I'll sometimes use "callouts" in the figures. These "callouts" are simply text in the figure itself, sometimes with lines or arrows. In this case, I use arrows to show the destination and direction of the quad-connectors hooking everything together.

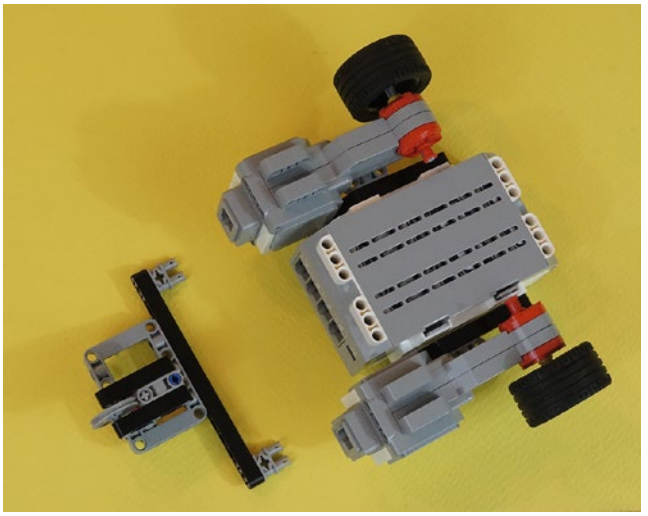


Figure 3-26. *Turn the main body and the rear-wheel/base upside down to connect them. The rear-wheel is pointing up in this view.*

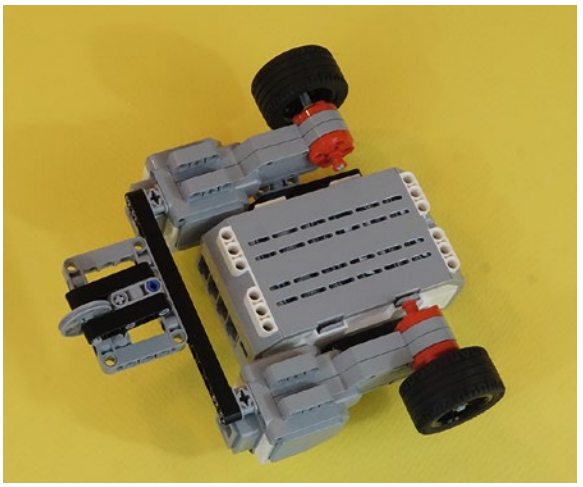


Figure 3-27. *The main body and the rear-wheel/base are connected*

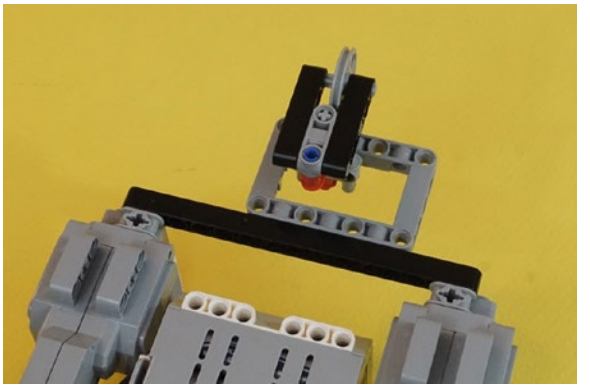


Figure 3-28. *The assembled main body and rear-wheel/base*

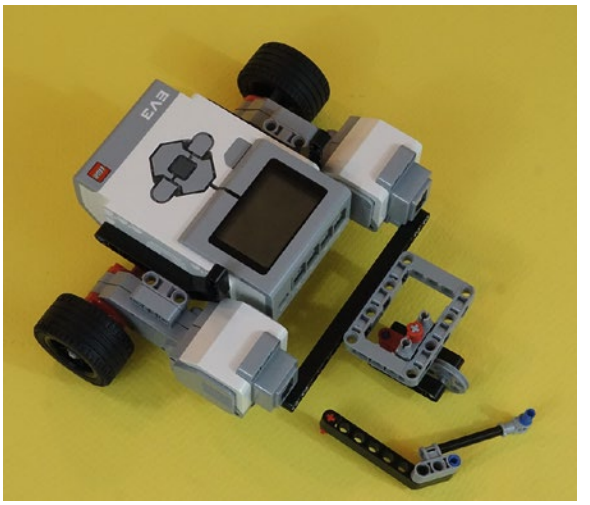


Figure 3-29. *Rear-wheel reinforcement strut ready for installation between body and rear-wheel/base*

Remember the tip about axles? Now you are putting that idea to work. The axle in the reinforcement strut is adjustable! This means you can slide it up or down to get the amount of stiff reinforcement you desire. That is, if you notice the rear-wheel is drooping, tighten this axle connection. Does it seem to come loose? Add more collars to the axle to make it stronger.

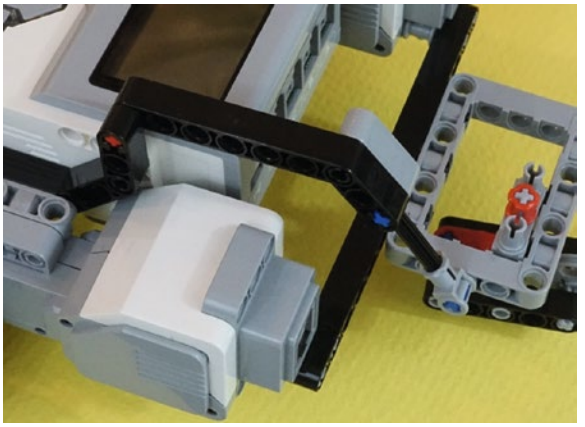


Figure 3-30. *Rear-wheel reinforcement strut, main body, and rear-wheel/base assembly*

Next, gather the head/neck assembly, as shown in Figure 3-31.

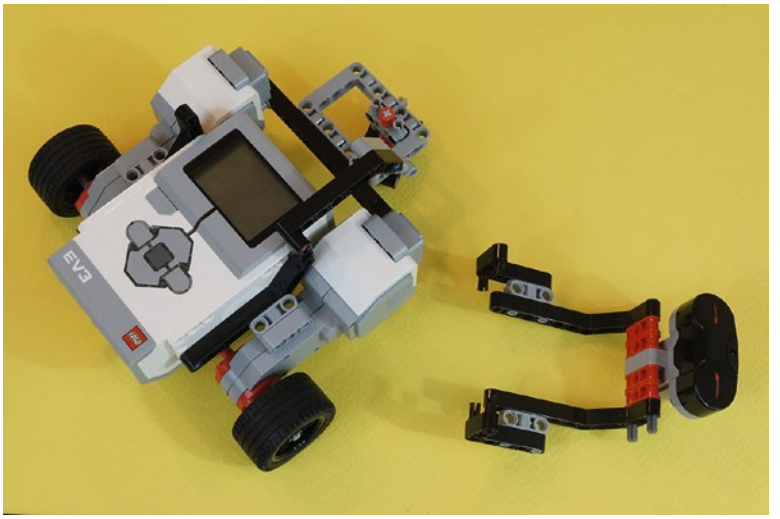


Figure 3-31. *Head/neck assembly ready for mounting on main body*

Finally, on the front of the main body, you're going to connect the head/neck assembly, as shown in Figure 3-32. You'll stretch the sides apart to get the connectors lined up with the body.

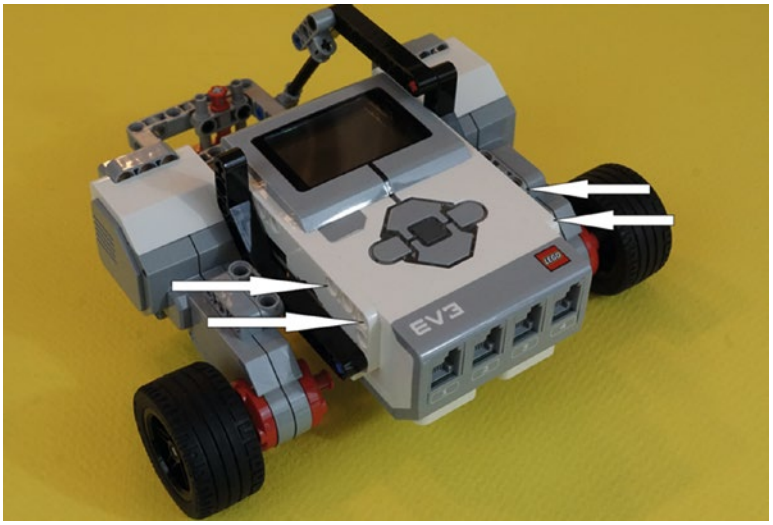


Figure 3-32. *The head/neck assembly will connect to the main body here*

You see the head/neck assembly connected to the main body in Figure 3-33. Take a close look and you'll see exactly where the left and right neck components connect to the Intelligent Brick.

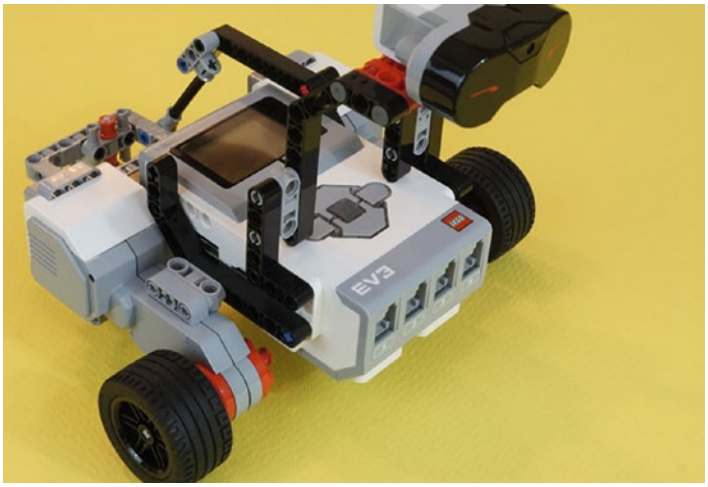


Figure 3-33. *The head/neck assembly connected to the main body (angle view)*

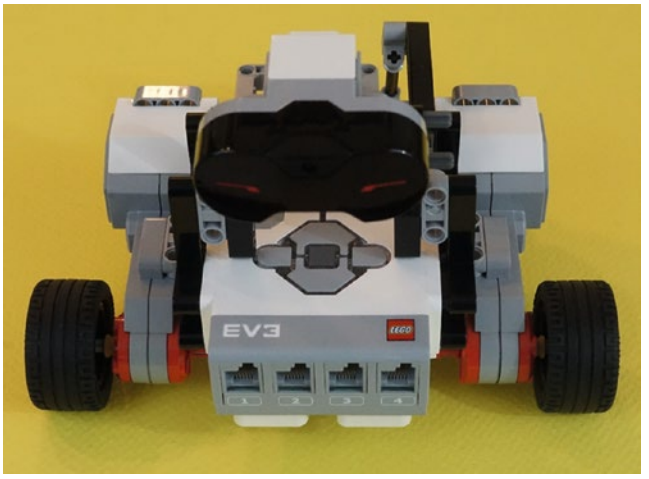


Figure 3-34. *The head/neck assembly connected to the main body (front view)*

Guess what? You're done with building the ExploroBot. To complete the wiring, do the following:

1. Connect a short cable from the Infrared Sensor to port 1, as shown in Figure 3-35.
2. Looking at the back of the ExploroBot as shown in Figure 3-36, connect a cable from the left motor to port B.
3. Connect a cable from the right motor to port A, also shown in Figure 3-36.

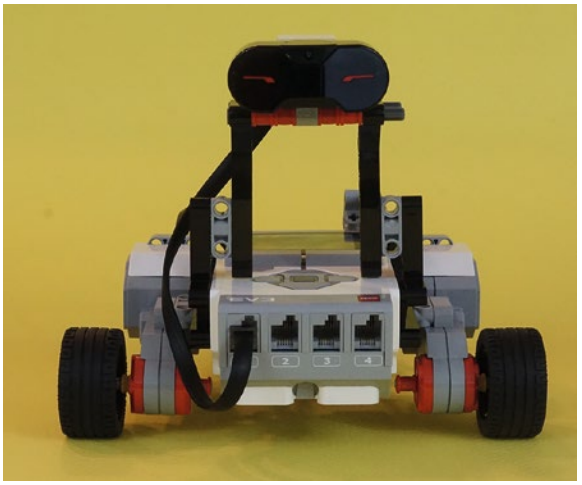


Figure 3-35. *The Infrared Sensor connects to port 1*

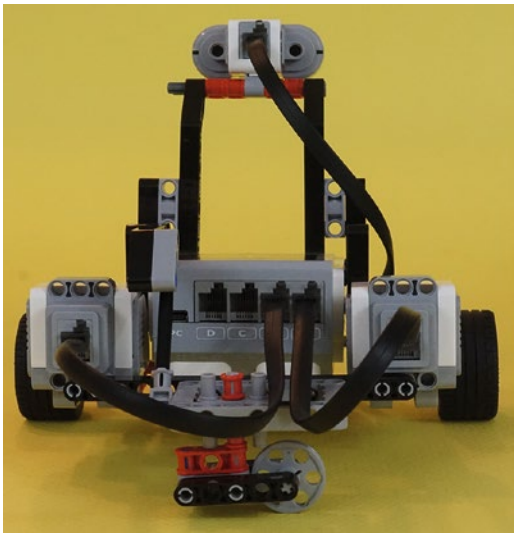


Figure 3-36. *The large motors are connected to ports A and B*

And that's it. You've built the ExploroBot. Figure 3-37 shows the completed project from various perspectives.

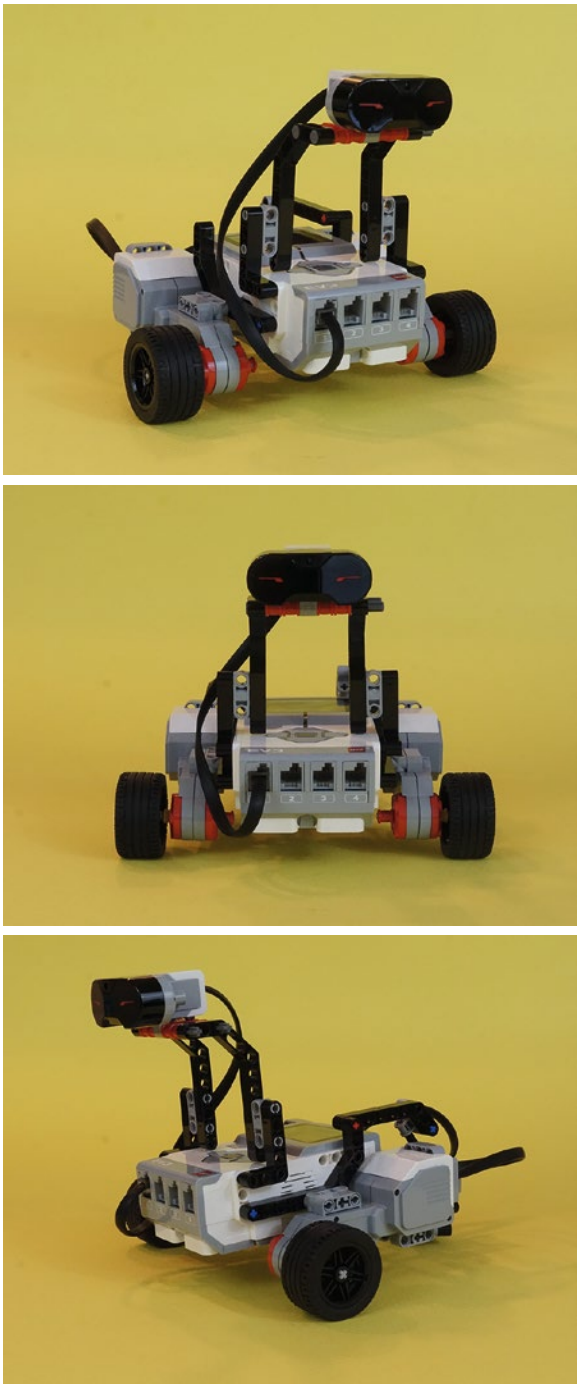
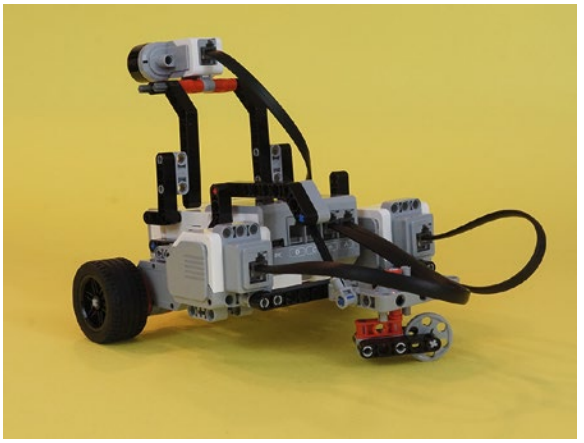
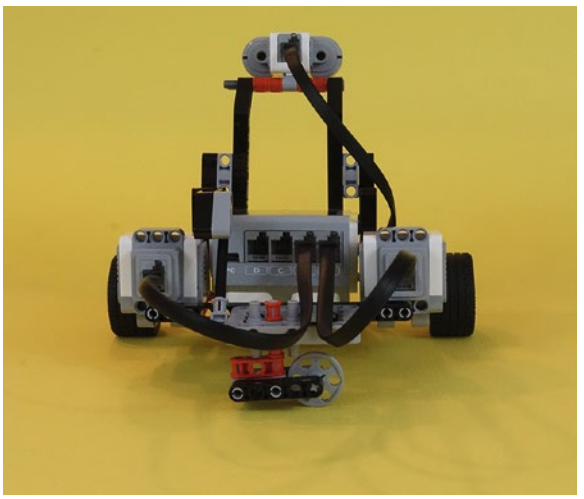


Figure 3-37. *The completed ExploroBot from various angles*





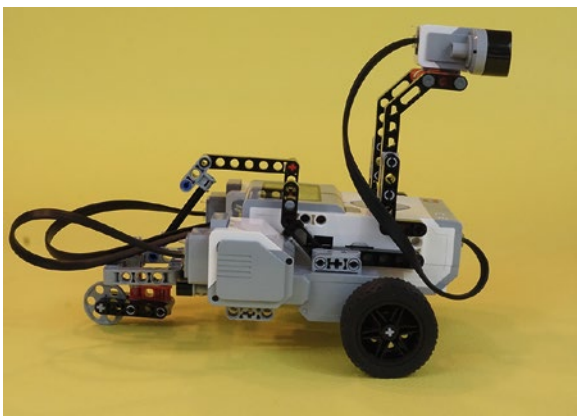


Figure 3-37. (*continued*)

But you're not done. Sorry.

Chapter 4 covers programming your ExploroBot. Without the programming to tell the bot how to behave, you have a nice paperweight and a locked tomb door.

So, on to the next chapter!

CHAPTER 4

ExploroBot: Program It

Your ExploroBot looks nice, but it really doesn't do much yet, does it? That's about to change. In this chapter you're going to create the program that sends the bot down the tunnel (and back) to trigger the locked tomb door. So, let's get started.

Some Experience Required

This chapter isn't about teaching you the basics of the software. Included with the LEGO MINDSTORMS EV3 software is a collection of software tutorials. At this point, I'm assuming that you've built the bots included with the MINDSTORMS EV3 kit and you've gone through the tutorials for programming the bots. During these tutorials, you picked up some basic skills in selecting programming blocks, dropping them into the workspace, and configuring the blocks.

In this chapter, I'm going to show you how to use your completed Design Journal sheet for the ExploroBot to help you construct the program, block by block. So go ahead and open the LEGO MINDSTORMS EV3 software (see Figure 4-1).

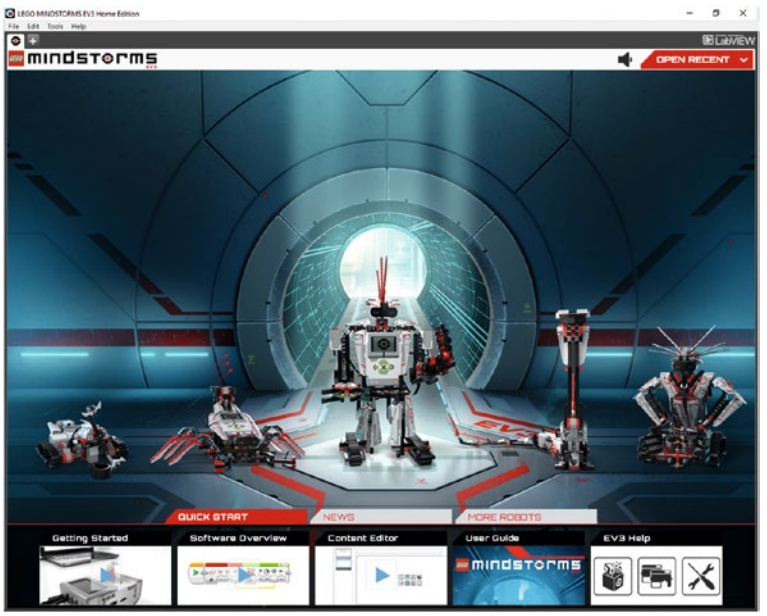


Figure 4-1. *The LEGO MINDSTORMS EV3 software*

M. Bell and J. F. Kelly, *LEGO® MINDSTORMS® EV3*, DOI 10.1007/978-1-4842-2262-1_4

You're going to create a new project and program, so touch the Plus sign tab at the upper-left corner. Click on the default title Project and type **ExploroBot**, replacing the default title (see Figure 4-2).

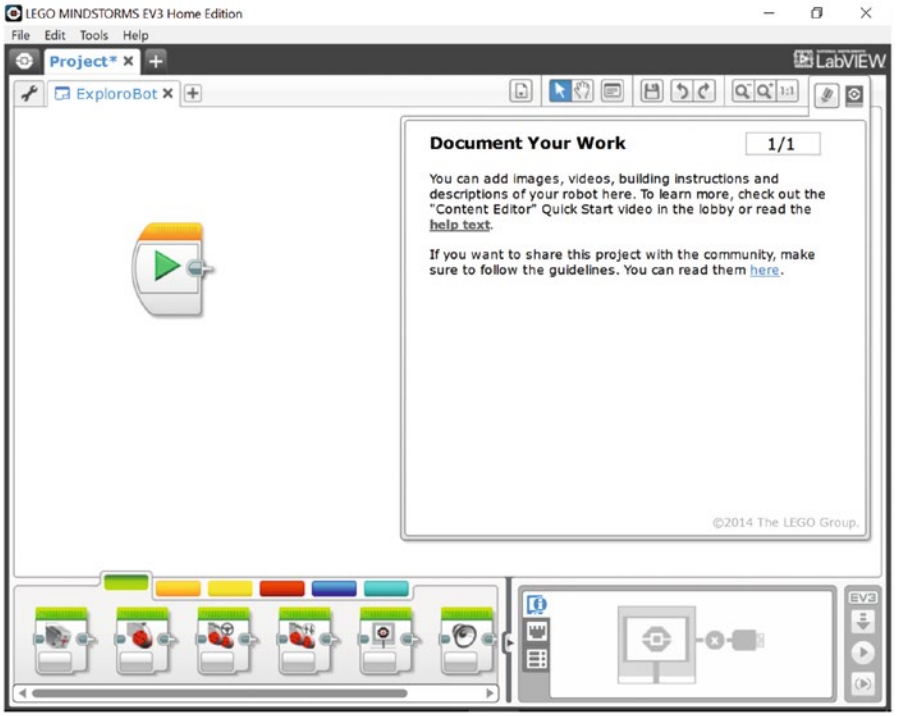


Figure 4-2. *Enter a name for the new program underneath the Project tab*

■ **Note** To have more workspace visible on your screen, minimize the Document Your Work area on the right by clicking the EV3 Button symbol at the far right of the top tab bar. (This symbol is shaped like a little stop sign—that is, an octagon. It resembles the buttons on your EV3 Intelligent Brick.)

Now, before we start dropping blocks all over the place, we need to think about what this program is supposed to do. Remember the `Task List` from the Design Journal? This is where that list is going to come in handy (see Figure 4-3).

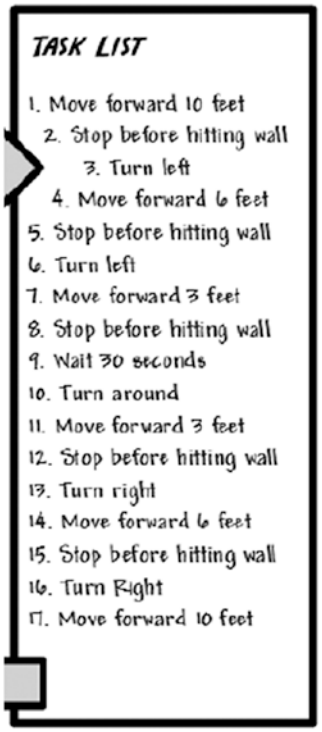


Figure 4-3. *The Task List will help us program the ExploroBot*

We're going to use each of the numbered items from the `Task List` to determine what types of programming blocks will be placed on the workspace. Like the construction of the actual ExploroBot, there are also numerous ways to program the bot. As you experiment with the LEGO MINDSTORMS EV3 software (often called EV3-G for short), you'll probably discover new (and better) methods for programming. You might find a way to shorten the program so it takes less memory space in the Intelligent Brick. Or you might choose to switch out the Infrared Sensor with the Touch Sensor, which requires slightly different programming blocks. My point is this: there's no *perfect* method for programming the ExploroBot. With that in mind, let's do a little planning before dropping some blocks.

Take a look at the `Task List`'s first item: "Move forward 10 feet." What do you think—maybe throw in a Light Sensor block? Just kidding. The bot first has to move forward down the tunnel, so that means a `Move Steering` block (see Figure 4-4).

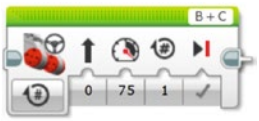


Figure 4-4. *The basic Move Steering block. This one can control two motors at once*

■ **Note** You drag and drop a `Move Steering` block from the green tab, as highlighted in Figure 4-2. The colored tabs each represent a different category of blocks, such as Action, Flow Control, Sensor, etc. You can hover over each tab to see which category it is. It is worth investigating all the tabs, which will help to familiarize you with the other available programming blocks. If you find one you're not familiar with, drop it on the workspace and play around with it for a few minutes. There's no rush.

We'll configure the `Move Steering` block a little later. For now, let's move to the next `Task List` item: "Stop before hitting wall." Obviously, this would use a Touch Sensor or Infrared Sensor. The ExploroBot in Chapter 3 uses the Infrared Sensor, so we know we'll be configuring it later.

One interesting thing to note is that the bot will make four left turns during its travel down the tunnel. How did I come up with this number? Take a look at Figure 4-5 and you'll see all four turns.

After the robot has reached the end of the tunnel and turned completely around, it will make two right turns on its way out of the tunnel (see Figure 4-5).

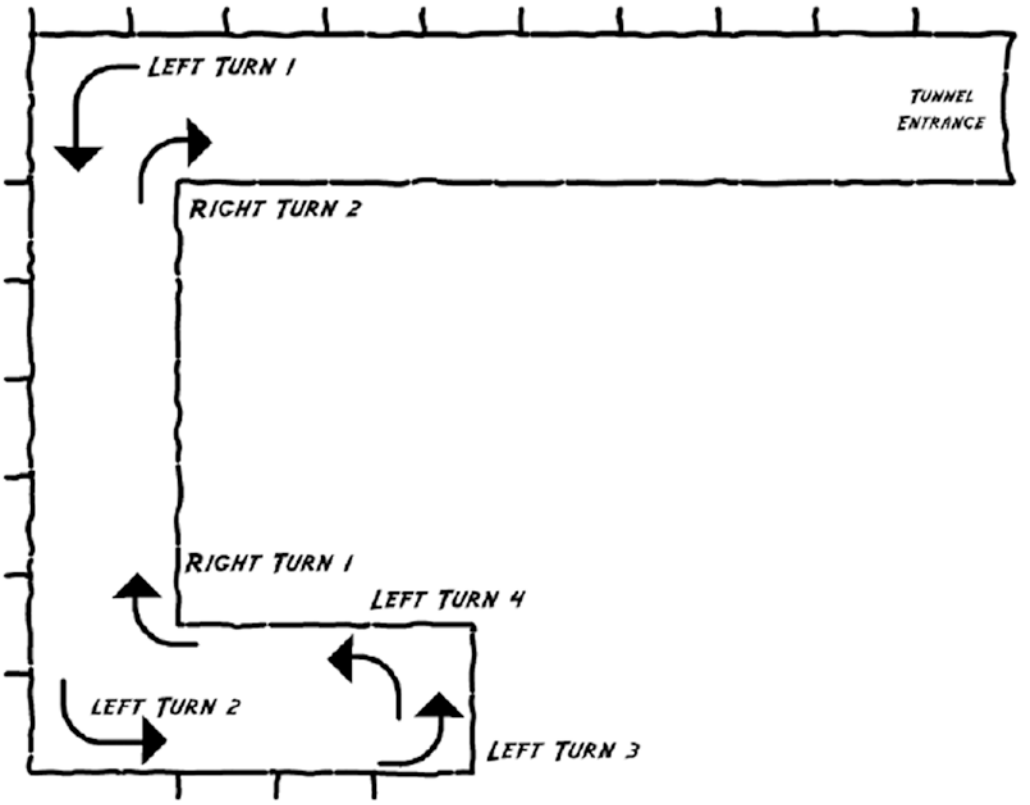


Figure 4-5. *The ExploroBot will make four left turns and two right turns during its travels*

Now, why is this important? Because when programming, sometimes it pays to try to reduce the amount of work you need to do. And the EV3-G software comes with a useful block that will save you some time. We're going to use the LOOP block here (see Figure 4-6).

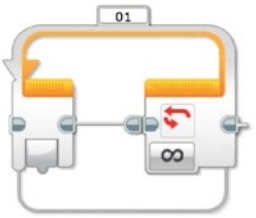


Figure 4-6. *The basic LOOP block*

The final block I want to discuss is the WAIT block (see Figure 4-7). When the ExploroBot reaches the trigger, I want it to stay there for a short period of time, just to make sure the pressure plate is triggered.

Figure 4-7. *The basic WAIT block*

With these four blocks, you have all the tools you need to program the ExploroBot to perform its duties.

Into the Tunnel

To save some time, I want you to look at the `Task List` again. I'm going to group some of the tasks together like this:

(Group 1) Forward – Detect Wall – Stop – Turn Left (first corner)

(Group 2) Forward – Detect Wall – Stop – Turn Left (second corner)

(Group 3) Forward – Detect Wall – Stop – Turn Left (end of tunnel)

Turn Left (this final turn is so the bot is pointed in the direction to leave)

Notice the first three groups are *identical*. When programming the ExploroBot, you could choose to place a `Move Steering` block, another block for the Infrared Sensor to detect the wall, another block to stop the motors, and another block to turn the bot left. And this will only get you past the first corner! You've still got group 2 and group 3 to go!

This could take a while. But not really. You're going to use the LOOP block in such a way that you can have the ExploroBot do all three groups with minimum programming. First, place a LOOP block after the START block (see Figure 4-8). It's not much to look at, but hang on.

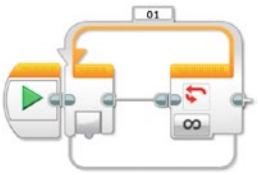


Figure 4-8. *You'll start with the LOOP block to save some time*

The LOOP block repeatedly executes blocks within it (inside its orange borders) an unlimited amount of times *or* a specified number of times. The LOOP block can also repeatedly execute a sequence of blocks inside it until a condition is met (such as a sensor being triggered). You can even put a LOOP block inside a LOOP block!

■ **Note** Putting a LOOP block inside another LOOP block is called "nesting." We're going to use this concept with the ExploroBot. Keep reading.

Because you have three groups of similar tasks, you're going to set this first LOOP block to run three times. After you've dropped the LOOP block onto the workspace, take a look at how the LOOP block is configured (see Figure 4-9).

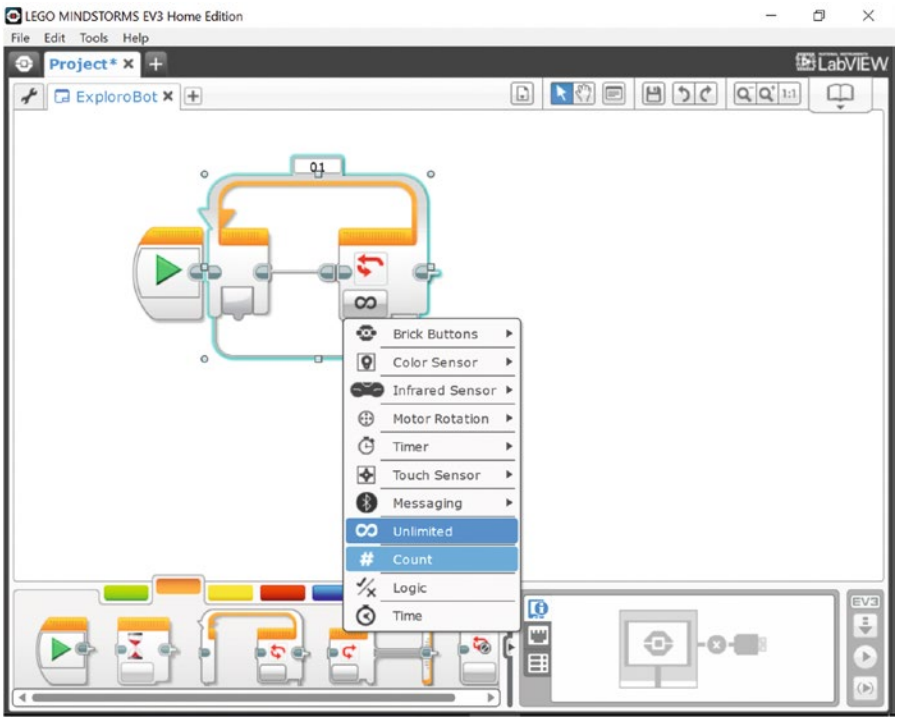


Figure 4-9. *Configuring the LOOP block*

From the Control drop-down menu, select Count. In Figure 4-9, Unlimited is highlighted in darker blue, since it is the control type presently selected (that is, the *infinity* sign, representing Unlimited). When you hover over Count, it is highlighted in lighter blue, confirming that it is the one you want to select. Press Count. Then, your block will look like the one in Figure 4-10.

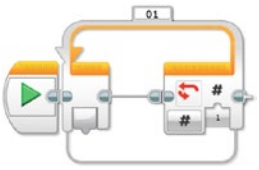


Figure 4-10. *The LOOP block set to Count, but only 1 time*

You now have a Count text box. Click on the numeral 1 and enter the numeral **3**. That means that any blocks or actions that you place inside this loop will be performed three times before this LOOP block is finished and the program continues. Now, what goes in the loop? Would you believe another LOOP block?

You want the ExploroBot to begin rolling down the tunnel and continue until the Infrared Sensor detects the first corner, right? Well, to program this you need a `Move Steering` block to continue turning Motors B and C until the Infrared Sensor detects the wall. To do this, you'll place a `Move Steering` block inside a LOOP block that's configured to use the Infrared Sensor to break the loop. Your first step is to just drop another LOOP block inside the first LOOP block (see Figure 4-11); it will expand to allow the new LOOP block.

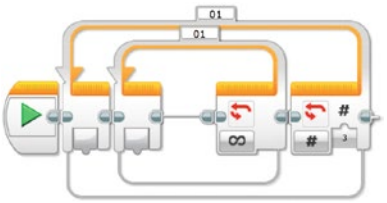


Figure 4-11. *Nest another LOOP block inside the first LOOP block*

Now you need to configure the inner LOOP block to use the Infrared Sensor (see Figure 4-12). Using the configuration panel for the inner LOOP block, click on its infinity sign to change the Control section to Sensor. In the Sensor section of the drop-down menu, select Infrared Sensor and set it to Proximity. You'll see a 4 next to it, which is the default setting giving you < (Less Than). That we can leave unchanged. The Distance setting defaults to 50, which you'll change to 5, representing five inches. Figure 4-12 shows the two LOOP blocks after you've made these changes.

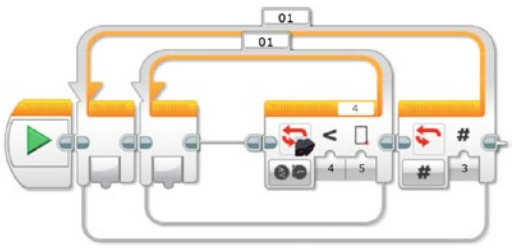


Figure 4-12. *The nested Loops configured to run the inner LOOP block three times*

■ **Note** At the end of this chapter, I'll cover testing of the ExploroBot. The setting of five inches might not be correct for your ExploroBot. You'll have to test different sensitivities of the Infrared Sensor to determine the proper distance for the sensor to break the Loop and stop the bot's forward movement.

Next you'll drop in the Move Steering block (see Figure 4-13) and click on the symbols to configure it. You want the motors to turn continually and only stop when the Infrared Sensor breaks the LOOP block. You're also using large motors B and C and motion set to forward. Click the circular symbol in the lower left of the MOVE block (indicated with black arrow). A pull-down will appear. The default is On for Rotations. Move your cursor to On and click.

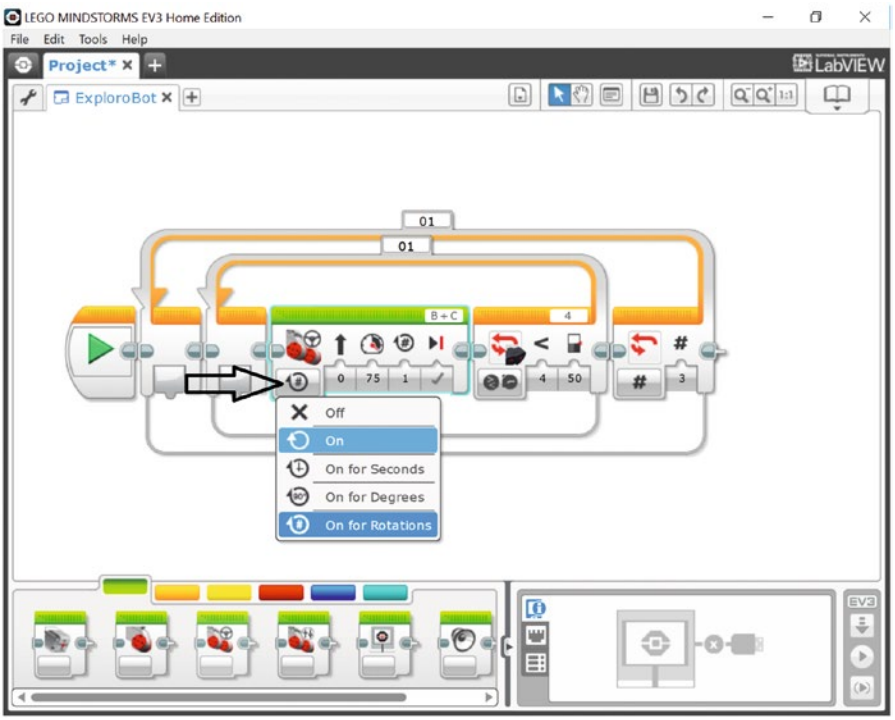


Figure 4-13. *Place a Move Steering block inside the inner LOOP block*

Once the LOOP block is broken (by the Infrared Sensor), the bot needs to stop the forward motion and turn left. This also is done three times (see the earlier three groups). Now you need to drop in a block to stop the motors, and another block to turn the bot.

First, let's drop in a Move Steering block to stop the motors (see Figure 4-14). It's easy to configure. Just select the Stop action (as shown) instead of forward movement. This is represented by an **X** in the lower-left side of the block.

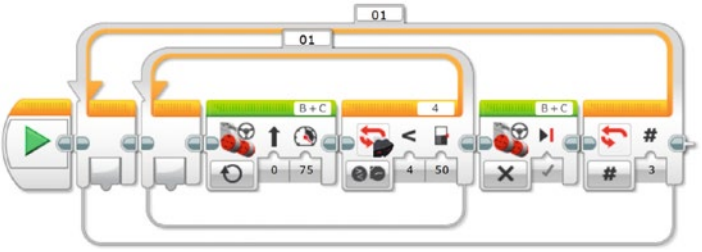


Figure 4-14. *Another Move Steering block stops the motors from turning*

The final action for the outer `LOOP` block is to turn the bot to the left. We'll drop in a block called `Large Motor` and configure it to run at 25% power for 360 degrees of rotation (see Figure 4-15). We have not used this block before. It controls the large motor as the `Move Steering` block does, but only operates a single selected large motor instead of two motors.

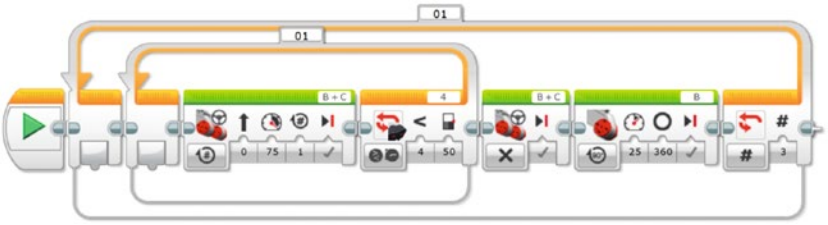


Figure 4-15. *This Move Steering block turns the bot to the left*

Engineering: *Encoders:* How Does a Motor "Know" Where It Is?

We're going to tell the motors to turn a certain number of degrees. But how do they "know" where they are, or how far they have to go? These motors contain a circuit called a *motor encoder*. This allows the motor to "know" where it is in terms of rotations. So if you command a motor to turn, say, 180 degrees, it will turn from its present location until it fulfills that many degrees. It will not simply "give up."

As an example, if you have a jaw that is supposed to close on an object, you might tell it "turn 90 degrees" and find that it does close that far. However, what happens if something is blocking its motion? At that moment, the encoder circuit has read that it turned, say, 75 degrees. But the command was to go for 90. The encoder is operating from a command to get to 90, so it tells the motor to keep turning. That is why it won't "give up." And that means your program *stays on that block* until the full move has happened! It will not proceed until it has done what it was told to do.

Returning to our ExploroBot, only one motor (B) is configured to turn in this `Large Motor` block. When motor B turns and motor C is motionless, the bot will turn left. At the end of this chapter, I'll show you how to determine the proper number of degrees needed to make a good left turn. For now, let's set it to 360. We'll also cut the power down to 25 for this block to make the turn slow and steady.

Let's pause here and take a look at what we've done. I'm also going to ask you to add some comments to your program. Comments are a built-in feature of the software and will come in handy if you ever want to make changes later or give the program to a friend. With comments, someone else looking at your program should be able to follow along and better understand what your program does and why.

At this point, you've got an outer LOOP block that will run three times. Everything inside it will occur three times and three times only. This means the bot will move forward until the Infrared Sensor detects an obstruction in its path (the tunnel wall), stops moving, and turns left—three times. If you go back to Figure 4-5, you'll find that when the outer LOOP block completes, the bot should be sitting on the trigger and facing north.

I suggest that you use the Comment tool (see Figure 4-16) and type in a description of the blocks that you've just created. In Figure 4-16, I've included my comments for you to see.

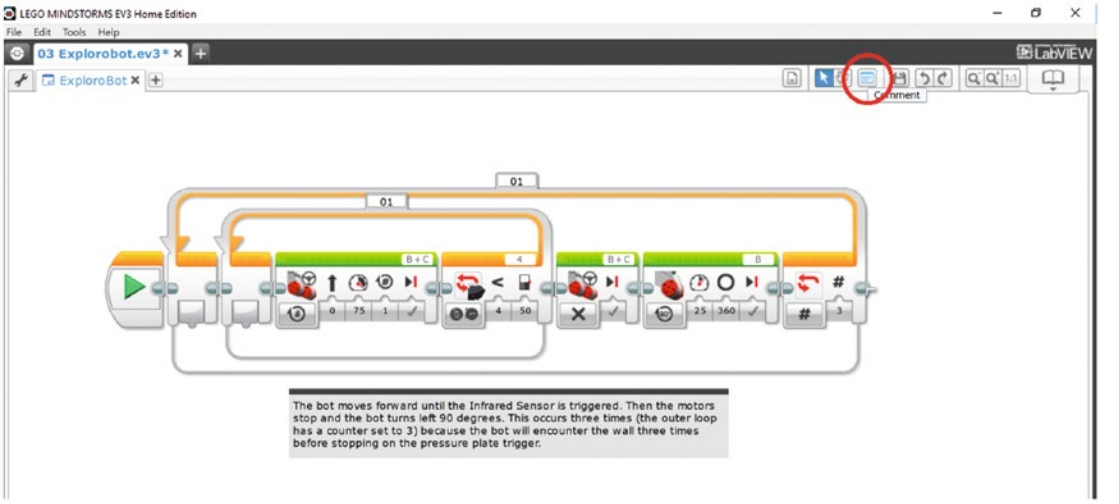


Figure 4-16. *Place comments for your blocks using the Comment tool (circled in the upper right)*

Now that the robot is on the trigger, let's go back to the `Task List`. Step 9 is for the bot to pause for 30 seconds. This is simple—you just drop a `WAIT` block at the end of the program and set it for 30 seconds (see Figure 4-17).

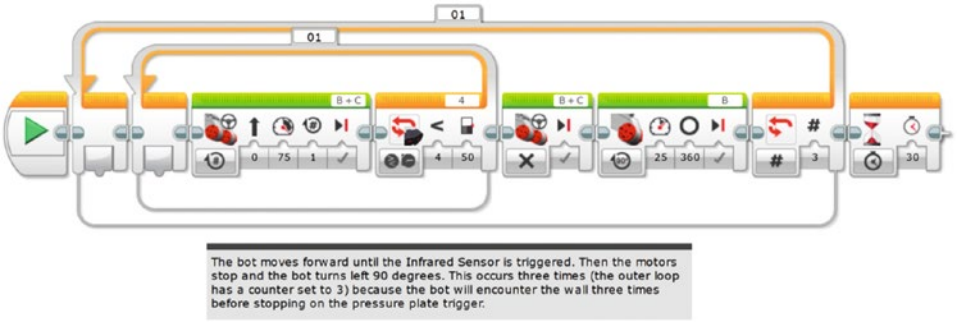


Figure 4-17. *A WAIT block is added for the bot to pause on the trigger*

Because the bot is facing north, you need it to make one more left turn (see Figure 4-5) so it can go back the way it came and leave the tunnel. Easy enough. Just drop a `Large Motor` block in and configure it like the other `Large Motor` block that turns the bot (see Figure 4-18). Like the earlier `Large Motor` block, let's set this one with a power setting of 25 and duration of 360 degrees until we determine the actual turn required.

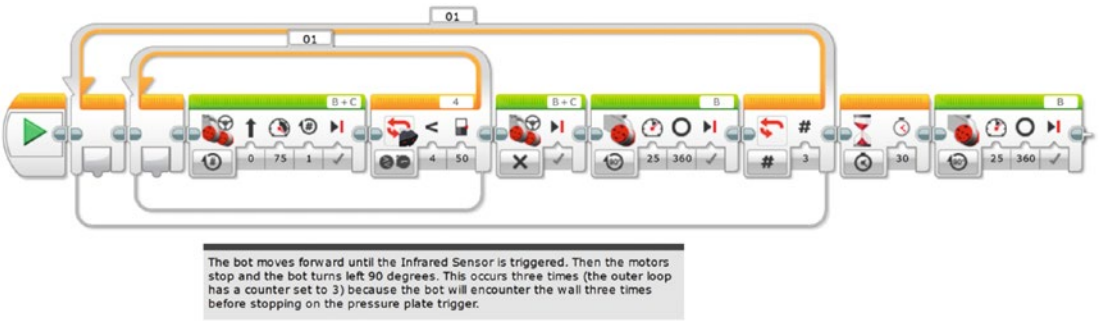


Figure 4-18. *A Large Motor block is added to make the final left turn*

Figure 4-19 shows a comment I've added at this point in the program. Again, I highly recommend comments. You never know when you might come back to this program and having comments will quickly help you to refresh your memory of how things work and are configured.

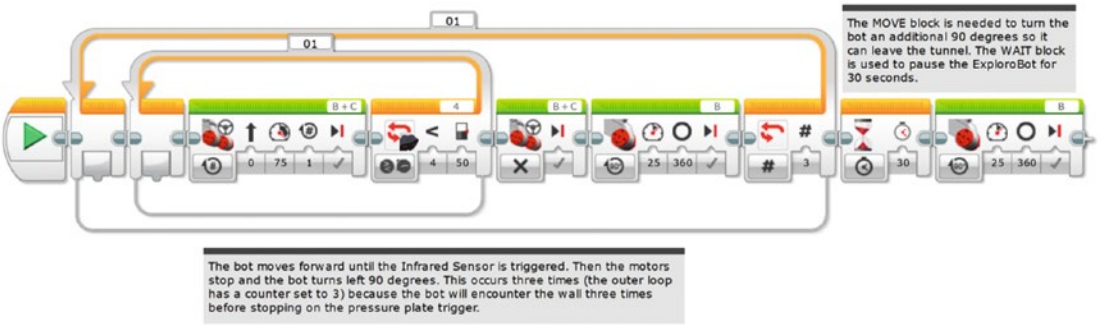


Figure 4-19. *I've updated the comments in the program*

At this point, it's time to program the exit of the ExploroBot.

Out of the Tunnel

Right now, the ExploroBot is facing west, ready to begin its exit from the tunnel. Just like the entrance, the bot will make some moves that are duplicates:

(Group 1) Forward – Detect Wall – Stop – Turn Right (second corner)

(Group 2) Forward – Detect Wall – Stop – Turn Right (first corner)

Turn Right (this final turn allows the robot to leave the tunnel)

So, once again, there's an opportunity to use two LOOP blocks, one nested inside the other. There are just a couple differences with these two LOOP blocks:

- The bot will be making right turns instead of left turns.
- The bot will make two right turns, so the outer LOOP block needs its Count set to 2 instead of 3.

Knowing this information, let's place the two LOOP blocks and configure the outer LOOP for three repetitions and the inner LOOP for the Infrared Sensor (see Figure 4-20). But didn't I say that it only needed to make two right turns? After that second right turn, you want the bot to keep rolling. It won't encounter another wall, but it should encounter your hands, waiting for it to come out of the tunnel. For that reason, you can configure this LOOP with a count of 3, even though it won't make an actual third turn.

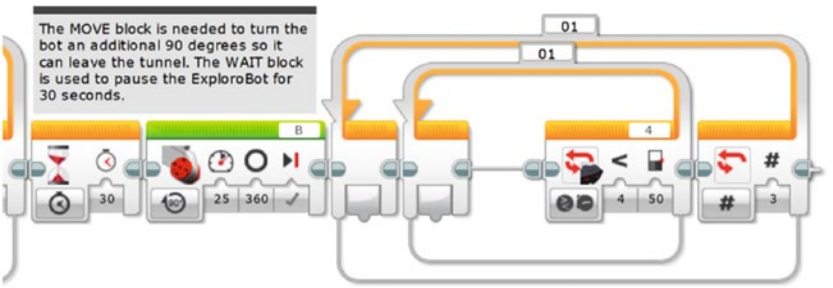


Figure 4-20. *Two LOOP blocks placed and configured*

Next, place three `Move Steering` blocks: one for moving the bot, one for stopping the motors, and one for turning the bot. The first `Move Steering` block is configured with Unlimited duration for Motors B and C. The second `Move Steering` block is configured to stop motors B and C, and the third `Large Motor` block is configured to turn *only* large motor C at a power level of 25 and a duration of 360 degrees (see Figure 4-21).

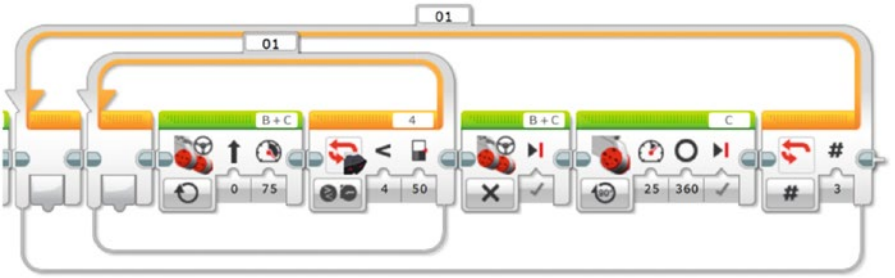


Figure 4-21. *Place two Move Steering blocks here, followed by the third Large Motor block*

Comments are placed describing these new LOOP and motor control blocks (see Figure 4-22).

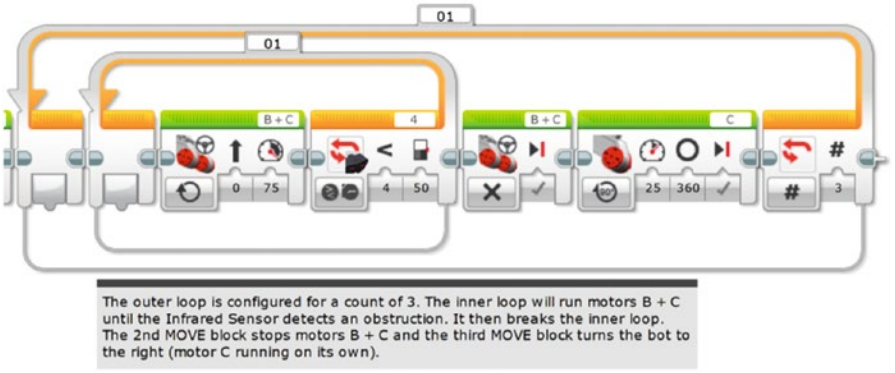


Figure 4-22. *Comments are added to describe these new blocks*

That's it. Well, not quite. But we're close. Remember all those `Large Motor` blocks that turn the bot? We set them to 360 degrees. That was just a guess. When I downloaded this program to my ExploroBot and ran it, those 360 degrees of rotation on the motors actually turn the bot only about 20 degrees to the left. To make those right and left turns, we're going to need the motors to turn quite a bit more. Luckily, the Intelligent Brick comes with a nice little tool to help you determine the correct number of degrees needed to turn the bot. Read on . . .

What the Degree, Kenneth? (With Apologies to REM)

Engineering: Measuring Actual Motor Rotations from the Brick Screen

First, I need you to turn on the EV3 Intelligent Brick.

Using the Brick's right or left buttons, scroll through the list of options until you find Port View. This will be the third tab from the left. Press the center dark gray Select button on the Brick, which will give you the third screen in Figure 4-23. Now you're going need to select the port to monitor. The first turn will be done by motor B, so use the up button to get to the top row (showing motor status) and then the right button to get to port B. As you get there, you'll see that the LCD screen will show **B:L-MOTOR-DEG**, with the number **0**$^{\textbf{deg}}$ below. (These EV3 Brick screens are shown in Figure 4-23.)

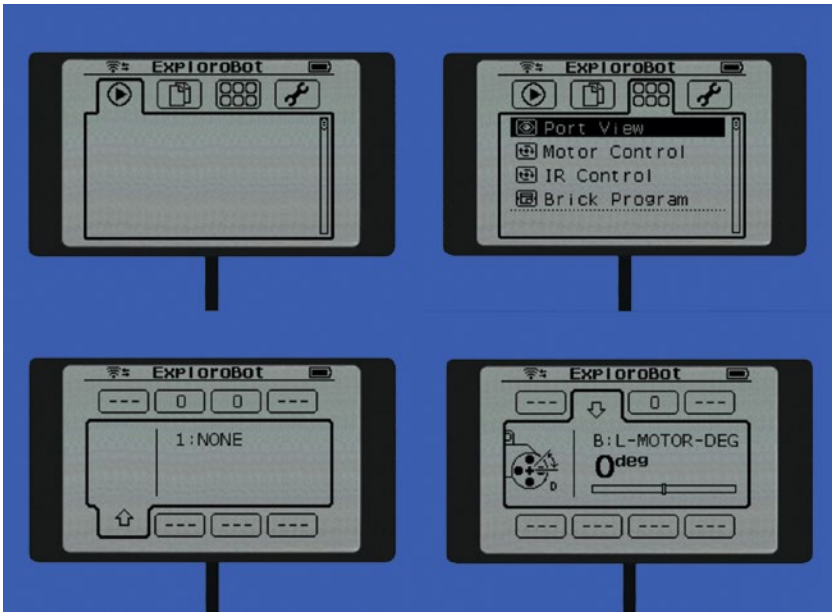


Figure 4-23. *Choosing the motor Port B to see 0 degrees as its starting point*

Just for grins, turn the wheel on motor B and watch what happens on the screen. That number tells you how many degrees the motor turns. If you turn the motor forward, you'll get a positive number (1 and climbing). If you turn the motor backward, you'll get a negative number (-1 and dropping). Press the center Select button to reset the degree counter.

Now, place the ExploroBot on a flat surface and press the center Select button again to reset the counter. Next, manually turn the ExploroBot 90 degrees to the left. For best results, try to keep the wheel for motor C from rotating. Just steadily turn the ExploroBot left so that the wheel on motor B turns. When you're done, take a look at the LCD screen. Your results may be different, but for my ExploroBot I got a reading of 535 (see Figure 4-24). Now do the same thing for port C. A true right turn, done the same way, should give you the same result for motor C (or very close).

Figure 4-24. *Observing how many degrees the motor on port B rotates, with the ExploroBot turned 90 degrees by hand*

The degrees you get for large motor B and large motor C is the number you'll enter for the Duration setting for the `Large Motor` blocks used to turn the ExploroBot. (See Figures 4-15, 4-18, and 4-21 for coverage of the three `Large Motor` blocks used for turning.) Go back and enter the number for all three `Large Motor` blocks (two for motor B and one for motor C). Save your changes and upload the updated program to your ExploroBot.

Keep in mind that your ExploroBot will be different from every other ExploroBot—some motors are little more stiff, some less so. My rubber wheels might behave a little differently on a wood floor versus a concrete floor. Battery levels will be different. There are so many factors that can affect how your ExploroBot operates. Don't get frustrated . . . just tweak and tweak and then tweak some more (see my own tweaks at the end of this chapter).

At this point, you'll have to do some testing to determine that the right and left turns are as close to 90-degree turns as possible. Don't take risks—if the bot is off by a few degrees, you'll take the chance that your bot might not go straight and get stuck somewhere where you can't reach it. You want the bot to turn as close to 90 degrees as possible and then move forward on a straight path. Tweak the individual Duration settings until you're happy that the bot is turning left and right correctly (see Figure 4-25).

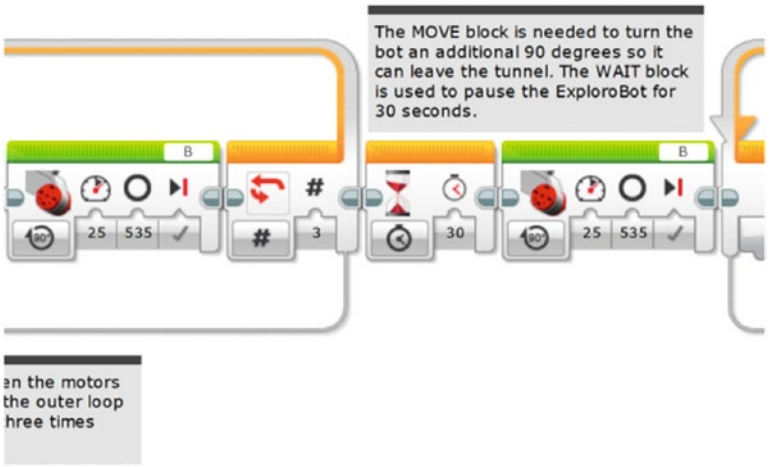


Figure 4-25. *I've entered 535 for the Duration settings on my three MOVE blocks for turning*

Opening the Tomb Door

Now it's time to simulate (that is, test) the tunnel challenge. If your ExploroBot navigates a test tunnel and returns successfully, great job. You can rest assured that the real tunnel would be no challenge to your well-designed bot.

Here are my suggestions for setting up your test tunnel.

■ **Note** You don't have to construct a real test tunnel to get results. Remember, your bot simply needs to move in fairly straight lines, react properly to obstacles (by turning left or right), stop on the "trigger" and wait, and come back to the starting point.

For my test "tunnel," I used the wooden floors of my home. I found a large open area and measured a distance of ten feet. At the end of the ten feet, I stood up a hardback book to simulate the wall of the first corner.

Next I measured six feet to the left and placed another hardback book to simulate the wall of the second corner.

For the end of the tunnel, I measured three feet left from the second corner and placed three hardback books to simulate the dead end (see Figure 4-26). If I run this as a challenge for several people, I'll make walls along the whole route, using boxes. I'll also tape a six-inch square of paper for the pressure plate in the center of the three books.

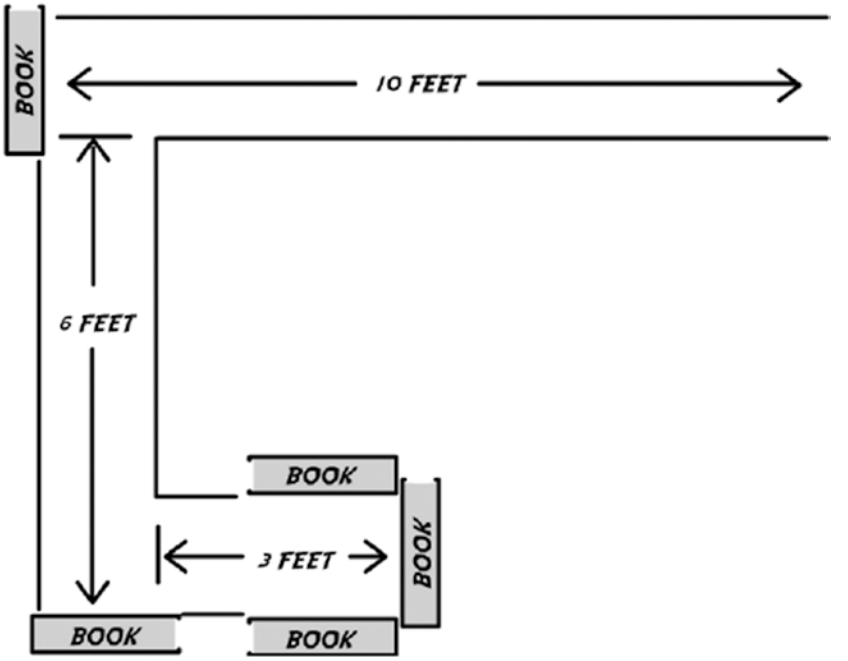


Figure 4-26. *Bird's-eye view of my test "tunnel"*

And my results? It took four runs before I was successful in having my ExploroBot return to me. Here's a summary of the runs:

- *First run*: My ExploroBot got too close to the wall before stopping. The simple fix for this was to change the Infrared Sensor setting from five inches to seven inches. The Infrared Sensor detected the wall earlier and was able to stop the bot so it had room to turn properly.
- *Second run*: When the ExploroBot successfully stopped at the wall and turned left, it turned just a little less than I needed. My original motor B setting was 535 and I changed it to 550.
- *Third run*: I had the same problem as the second run, but this time it turned a little too much. I had to change the setting again from 550 to 548. This time it worked.
- *Fourth run*: On the exit trip, I had motor C set to 535. I changed it to 548 and it worked great. The ExploroBot "exited" the tunnel and was back in my hands.

I could hear some odd noises behind the tomb door as the pressure plate was triggered and the tomb door lock was released . . .

CHAPTER 5

String, Pebbles, and Gravity

Location: Southwest Guatemala

Weather Conditions: 87° Fahrenheit, Humidity 88%, Rain 20%

Day 3: Inside King Ixtua's Tomb, 8:13 AM

Evan looked into the tunnel. The ExploroBot was about four feet away but still moving toward the tunnel exit. A few minutes earlier the ExploroBot had reached the trigger; some unusual sounds were heard behind the tomb door and then a loud BANG! While Evan listened for his bot to turn around and return, his uncle and a few other team members began pushing on the large stone door.

"It's opening!" yelled Uncle Phillip. "Keep pushing."

As the ExploroBot reached the end of the tunnel, Evan picked it up, looked it over for any damage, and then turned it off. He then turned to watch as Uncle Phillip, Max, and Grace finished pushing against the large stone door. Beyond the door, all Evan could see was darkness.

Uncle Phillip picked up a large flashlight, turned it on, and shone the beam into the tomb. Evan stepped closer for a look. As Uncle Phillip moved the light around the small inner room, the group could see a stone floor, but that was about it; few details were visible.

"It doesn't appear that any damage was done when we opened the door," said Uncle Phillip.

"Good, I was worried my robot might not actually be able to trigger the pressure plate," replied Evan.

Uncle Phillip turned to Evan and smiled. "Evan, if I were your history teacher, I'd give you straight As for the rest of the school year. That was amazing," he said.

Max and Grace nodded and smiled.

"Thanks," Evan replied. "Glad I could help. So . . . what's next?"

Uncle Phillip took off his cap and scratched his head. "Well, I think we need to go and review the manuscript again."

Evan raised his eyebrows. "More traps?" he asked with a smile.

Evan watched as Uncle Phillip instructed one of the Guatemalan guides to guard the door and let no one inside. Uncle Phillip then waved his hand. "Come on. Let's take a look at that manuscript."

More Monkey Business

Back in the tent, Uncle Phillip had pulled out the enlargements of the Tupaxu manuscript. Each enlargement was actually a photograph of a page of the manuscript. From what Evan could determine, the original pages were slightly larger than notebook paper sheets, but the enlargements were poster-sized.

Max took one of the enlargements, clipped it to a clothesline running across the tent, and then turned to Uncle Phillip.

M. Bell and J. F. Kelly, *LEGO® MINDSTORMS® EV3*, DOI 10.1007/978-1-4842-2262-1_5

Uncle Phillip was seated next to Evan and Grace. “Okay, this is the page that corresponds to the tomb’s reception room. Grace, you’re a better translator than I. What do you think?”

Grace stood up and walked over to the hanging picture. She then pointed at some of the Mayan writing. “One of the written legends about King Ixtua mentioned the word ‘akh.’ The translation is ‘vine.’ This drawing here contains the Mayan glyphs for vine and monkey. Take a closer look at this sketch right here,” she said, pointing to a small picture (see Figure 5-1).

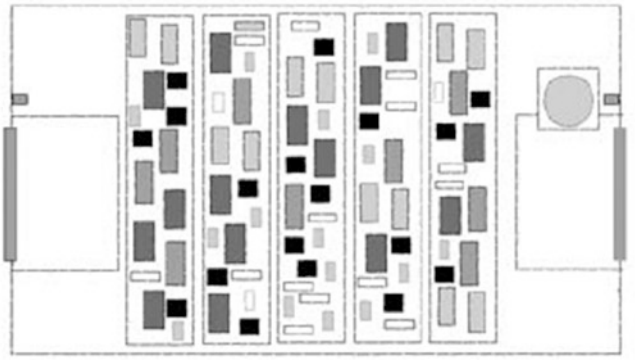


Figure 5-1. *Sketch of the reception room*

“Keep in mind,” Grace continued, “that Tupaxu was well known for designing other tombs with elaborate triggering mechanisms. Some of these triggers were to open doors or passageways . . . and some were to trigger traps. What I think you’re looking at here is a combination of the two.”

Uncle Phillip, Evan, and Max examined the small picture.

“That looks like a jar or cup,” Evan said, pointing to the image. “And that’s got to be another door, right?”

Uncle Phillip nodded. “Yes. The manuscript does mention a series of obstacles before reaching King Ixtua’s burial chamber. If this is one of those obstacles, my guess is that there is a trigger in this room to open that door,” he said.

Grace pointed at a small block of Mayan symbols below the picture. “This roughly translates to ‘Chuen Ra Rock Drop.’”

“What’s a ‘Chuen Ra’?” asked Evan.

“Chuen means monkey,” said Max. “The monkey’s name must be Ra.”

Grace nodded. “Yes, and this little bit of writing here is very important.” She pointed to some more Mayan glyphs. “This translates to ‘shaky ground.’ I think this part of the reception room floor is a trap,” she said.

Uncle Phillip pointed to a tiny round circle in the picture. Above the circle was a line drawing of a monkey and below it was the jar Evan had pointed out. Several of the tiny circles were drawn inside the jar as well. “Could that be a pebble?” he asked.

“We’ll need to examine the actual space, but I have a guess about this room,” said Grace. “I think the floor is a trap and cannot be crossed until this jar is filled with pebbles. The jar is probably sitting on a small trigger that will open the next door when the weight of the pebbles in the jar becomes heavy enough. It’ll probably also allow us to cross the floor without triggering a trap.”

“I think you’re right,” said Uncle Phillip. “And it does fit with the legend about using monkeys to enter the tomb. I don’t think this vine would be strong enough for a person to cross, do you? Okay, it’s time to go take a look.”

Vine Challenge

At Uncle Phillip's request, a few of the expedition's guides installed two large tripod lamps just inside the tomb entryway. Thick power cables ran out of the tomb and were plugged into a small, running generator. Bright light from the lamps flooded the reception room.

Max and Grace had followed Uncle Phillip into the room. Evan had been asked to wait until it was determined the reception room was safe. After a few minutes of quiet discussion, Evan heard his uncle yell out, "Come on in, Evan!"

Evan walked slowly into the tomb. To his right were the two tripod lamps. Uncle Phillip, Max, and Grace were standing on a stone platform, approximately eight feet wide and six feet deep. Beyond the stone platform, the floor was a pattern of small rust-brown bricks that extended for another 15 feet, and on the other side of the room were another small platform and a door.

"Stay on the platform, Evan," said Uncle Phillip.

"Yes, sir," Evan replied.

Grace pointed at the floor. "The bricks are probably covering pressure plates. If we step on them, the far door will likely be blocked for good," she said.

"Or worse," replied Max. "The Rupa tomb was designed by Tupaxu and it had poisonous spears that dropped from holes in the ceiling."

The group all looked up at the tomb's ceiling and took a few steps back.

Evan looked across the room. "There's the jar, right where the picture indicated. And what's that above it?" he asked.

To the left of the door was a raised platform with a modest, carved wooden jar sitting on top. Three feet above the jar was a small wooden peg embedded in the wall.

Grace turned and looked to her left. "It's a peg. Identical to this one," she replied and pointed. On the left wall near the tomb entrance was a similar wood peg, embedded in the wall at the same height as the other peg (see Figure 5-2).

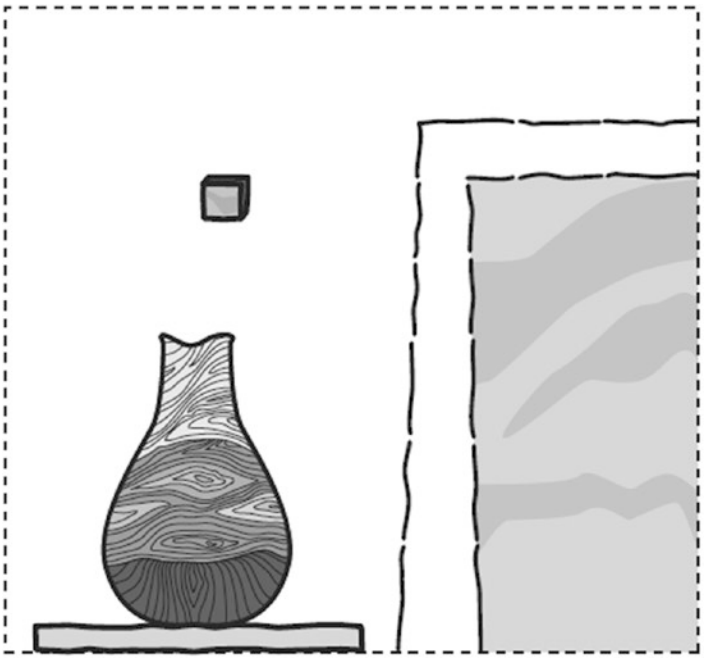


Figure 5-2. *Near the stone door is a peg above a wooden jar*

"Those pegs are where the vine was tied," said Max. "Of course, 700 years have passed and the vine has decayed." Max scratched his head. "You know, if we could manage to get a rope tied to both pegs, someone could climb over to the other side."

Uncle Phillip shook his head. "It won't work. Look at those pegs. They're not very large, and I doubt they'd hold the weight of any one of us. Tupaxu designed them correctly. Only a monkey could get across."

"Or a little robot," replied Evan.

Uncle Phillip, Max, and Grace all turned and looked at him.

Evan's Solution

"I think the hardest part is going to be getting some string around that far peg," said Evan.

Uncle Phillip had gathered his team in the equipment tent. Everyone was busy digging in boxes, looking for a ball of string or twine. "That's if we can even find some," Max replied.

"Keep looking. I know we brought some strong twine," said Uncle Phillip. "Okay, Evan. Tell us again what you have in mind."

Evan had pitched his idea quickly in the tomb about using a robot, but hadn't given any details. He took a deep breath and spoke slowly. "From what I could see, the monkey would cross to the other side of the room on the vine, holding one or two small pebbles with his feet, and drop the pebbles into the jar. So what I need to do is create something that holds a little pebble, crosses over on a string or some twine, drops the pebble, and comes back for another one."

"Found it!" yelled Grace. She held up a ball of tan-colored twine.

Uncle Phillip pulled a camp chair over to where Evan was seated and sat down. "It would have to be fairly lightweight, Evan. I don't think those pegs will hold much weight. And you'll have to figure out a way to drop a pebble accurately into the jar."

Evan nodded. "I'm pretty certain I can do this," he said. "What I need someone else to do is find a way to get that string tied to the far peg."

Max walked over to Evan and sat down. "How long do you think you'll need to create the bot?" he asked.

"I'm not sure," replied Evan. "At least four or five hours. Sorry."

Max smiled. "Don't apologize. It might take us that long to get the string looped around the peg."

"I have an idea," said Uncle Phillip. "We might be able to take some of the lightweight fiberglass rods that we use to reinforce our tents and tape them together. Make a long pole. It won't bend or break."

Grace nodded. "If we tie the twine to a little ring, we could slide the ring off the pole and onto the peg," she said. "I think we can do this."

Uncle Phillip nodded. "Okay, then. Evan is going to get started on the robot. Grace and Max are going to get the string attached to both pegs. And since it's almost lunch time, I'm going to get all of you some sandwiches so you don't have to stop working."

While Max and Grace began to talk about locating the fiberglass poles, Evan opened up the yellow toolbox and pulled out the small brown notebook. He flipped it open to a blank journal page and began to write. "Let's see—I'll list some of these robot functions, and hang on—let's get those constraints down…"

Story continues in Chapter 9 . . .

CHAPTER 6

StringBot: Planning

What's needed to solve this latest challenge is a bot that can move along a string, carry a small object, drop that object at a specific location, and return for another object to do it all over again. In this chapter, we're going to figure out how to make that happen.

Design and Planning

I'm sure I didn't fool you by reversing the words in the section title. Since you are no longer scared by the phrase "planning and design," I'm just a little worried you might become a little bored with it and try to skip it. So I'm going to shake it up a bit in this chapter . . . challenge you a little bit to come up with something truly unique. The jar isn't going to magically fill with pebbles, so you've got some work to do.

The StringBot

In Chapter 2, I gave you an advanced look at the ExploroBot before we began working on the Design Journal page. While I've got my version of the StringBot ready to go, try to keep from skipping ahead to Chapter 7 to look at it. I want you to start seeing your own StringBot in your head without my design influencing you.

We're going to use the Design Journal page again in this chapter. If you followed along in Chapter 2 carefully, this chapter will follow the same steps. Get a blank Design Journal page and a pen and let's start designing.

Note There are four blank Design Journal pages left in the back of this book (if you used one for Chapter 2). If you need more pages, feel free to make photocopies of the Design Journal page or visit the Apress web site to download the page in PDF format.

At the top of the Design Journal page, in the `Robot Name` box, go ahead and write **StringBot**. Feel free to create your own name for the bot; names such as TwineBot, MonkeyBot, or Gort are available. (You get Big Bonus Points if you recognize the "Gort" reference.) Once you've selected your bot's name, it's time to work on the `Robot Description`.

M. Bell and J. F. Kelly, *LEGO® MINDSTORMS® EV3*, DOI 10.1007/978-1-4842-2262-1_6

The Robot Description

Remember, the `Robot Description` doesn't need to be pages and pages of written text. Your goal is to keep it simple and uncluttered. As I said in Chapter 2, this isn't where you describe the bot as "lightweight, a mixture of parts, some motors, and maybe a sensor or two." This section is where you try to accurately describe the overall process the robot will follow.

Ask yourself the question, "What is this robot supposed to do?" and start writing inside the `Robot Description` box. Write "visually." What do I mean by this? Picture a box (a shoebox, for example) in your hands—the box doesn't have any sensors or motors or any other items yet, it's just a box. Imagine this box doing what you believe needs to be done to solve this challenge (refer to Chapter 5 if you need a reminder). Now compare what you wrote down to my `Robot Description` in Figure 6-1.

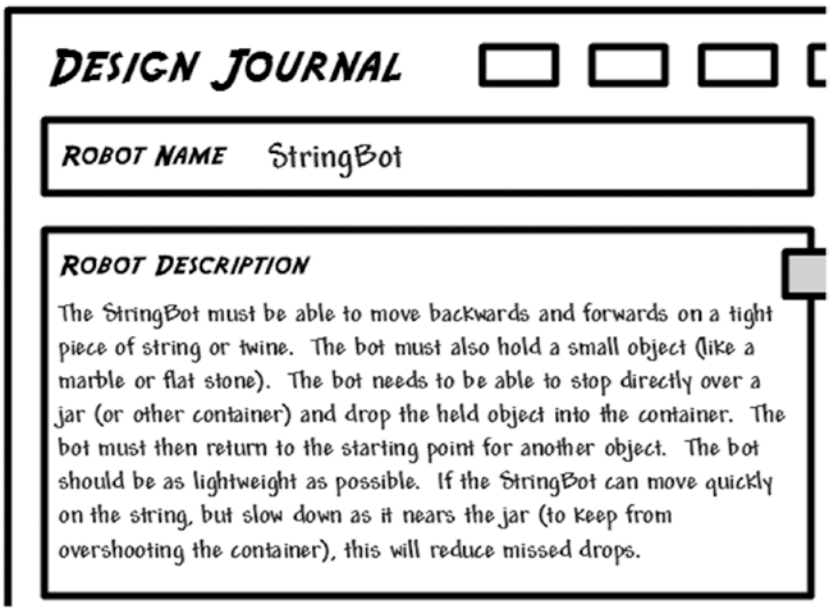


Figure 6-1. *The StringBot's Robot Description should be short and simple*

Did you cover the major requirements? Again, don't worry if your `Robot Description` doesn't match mine exactly. Although it wasn't mentioned in the challenge (Chapter 5), I had a thought that if the bot were moving too quickly on the string, it might overshoot the jar. So I added "If the StringBot can move quickly on the string, but slow down as it nears the jar (to keep from overshooting the container), this will reduce missed drops." You might not have added that to your `Robot Description`—maybe you have an idea to prevent that from happening. Great! Put it into your `Robot Description`.

The major items you need in your `Robot Description` are the bot's need to traverse the room on a string, backward and forward, and to be able to hold a small object and drop it into a container. Make sure that the bot returns for another object (it will take many visits to fill the jar) and that you have a good `Robot Description`. If so, you are ready to move on to the next step—the `Task List`.

The Task List

If you'll remember back to Chapter 2, the `Task List` takes your `Robot Description` and breaks it down into the bot's individual functions—move forward, stop, drop rock, etc. Look on your Design Journal page. As I stated before, if you have a good `Robot Description` written down, the `Task List` almost writes itself. Go back and review your `Robot Description` and start listing the individual tasks the StringBot will perform. I've made my list for the StringBot, and you can view it in Figure 6-2.

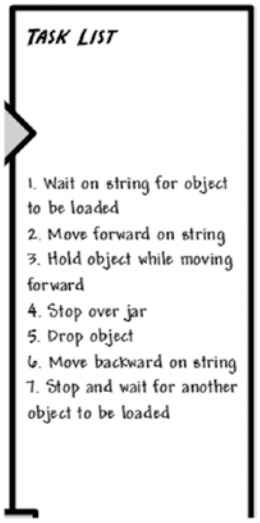


Figure 6-2. *The StringBot's task list is almost identical to the robot description*

The `Task List` is one of the most important sections of the Design Journal; each of the tasks you list will affect the construction of your bot. Since each item is an action item, each action must be paired up with a physical assembly that will either perform the action or assist with the performance of that action. Have you ever heard the phrase "form follows function?" Basically what it means is that the shape of an item (its form) is usually determined by what it will do (its function). Your bot is no different. In order for your StringBot to perform its duties, you must keep its main job in mind while you are designing it.

Now, back to your `Task List`. Read over it again and look at your `Robot Description`. Did you catch everything? Did anything new come to mind?

In my `Task List`, you might have noticed that the first item, "Wait on string for object to be loaded," isn't mentioned in my `Robot Description`. That's okay! This task was an afterthought. While the idea is fairly common sense (what good is sending the bot down the string if it's not holding an object, right?), I decided to add it to my `Task List` anyway. I could have gone back and added this at the end of my `Robot Description`, but as long as I have it *somewhere* on my Design Journal page, I should be okay.

I want you to keep in mind that your Design Journal page is for you to keep track of *all* of your thoughts on this project. It doesn't have to be neat and clean—as a matter of fact, the best Design Journal page will have scribbles and notes and things crossed out—a real mess! Later, if you want to record your work for history, feel free to rewrite your Design Journal page all nice and clean.

The final point I want to make on the `Task List` is an item that is *not* listed. Look back at the last sentence of my `Robot Description`—"If the StringBot can move quickly on the string, but slow down as it nears the jar (to keep from overshooting the container), this will reduce missed drops." Do you see it on my `Task List`? Why isn't it there?

Well, it turns out that this really isn't a task. It's an observation that I want to remember later, when I start designing. It would fit perfectly in the `Mindstorm` box, but I didn't want to take the chances that I'd forget my thought later, so I put it down in the spot I was currently working—`Robot Description`. I mention this only as another example of how important it is to write down everything that comes to mind! That great thought you have now might slip away in 30 minutes when you're sketching or skateboarding or eating lunch. Remember: when the thought pops into your head, write it down! (I'll revisit that last sentence in my `Robot Description` later in the "Mindstorm" section.)

This `Task List` was shorter than the one for the ExploroBot (mainly because this bot goes back and forth with one task in mind—the ExploroBot moved forward different distances, which were each given their own task), but each item is just as important.

Limitations and Constraints

I want to remind you here that you should write down in the `Limitations/Constraints` box *only* limitations that come quickly to mind. Don't spend too long thinking over this section because, truthfully, until you start building, you really can't imagine all the limitations that you are going to run into. Just visualize the StringBot (or the shoebox) moving down that string and write down where you think you might run into trouble. Spend five to ten minutes (max) thinking about a bot that moves along a string to drop a pebble in a jar. What could go wrong? Well, take a look at Figure 6-3 to see what I came up with for the `Limitations/Constraints` box.

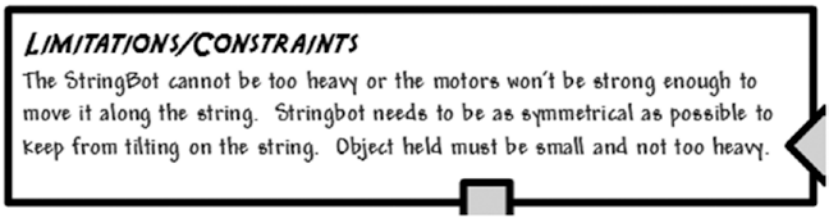


Figure 6-3. *The StringBot does have some limitations to overcome*

One *huge* constraint that we really need to focus on for the StringBot is how it will move.

When it comes to the StringBot's size and weight, we need to keep it at a minimum. Think about a simple piece of string or twine tied between two poles (or pegs). If you put something extremely heavy on that string, the string will dip down. If it dips down too much, it might slide to the center of the string and stop moving completely. And even if the string doesn't dip down, another problem can occur. If a bot's motor spins too fast *or* if the motor is simply too powerful, the bot might just spin in place and not move at all! The EV3 parts are made of plastic. Like a rubber tire on a wet road, if you try to move plastic parts on a string too quickly, it can be difficult for the plastic parts to "grab" on and move. Figure 6-4 demonstrates these two problems.

Figure 6-4. *Problems with a heavy bot on a string (left image) or a motor that spins so fast that the bot doesn't move on the string (right image)*

The next item on my `Limitations/Constraints` list is "StringBot needs to be as symmetrical as possible to keep from tilting on the string." If you didn't think of this one here, don't sweat it. It would have become *very* obvious once you started building. When it comes to your StringBot, what is on the left side of the string needs to be fairly symmetrical to what's on the right side. If one side has more components or motors, it will tend to pull the bot in that direction and can cause the bot to slip off the string (see Figure 6-5).

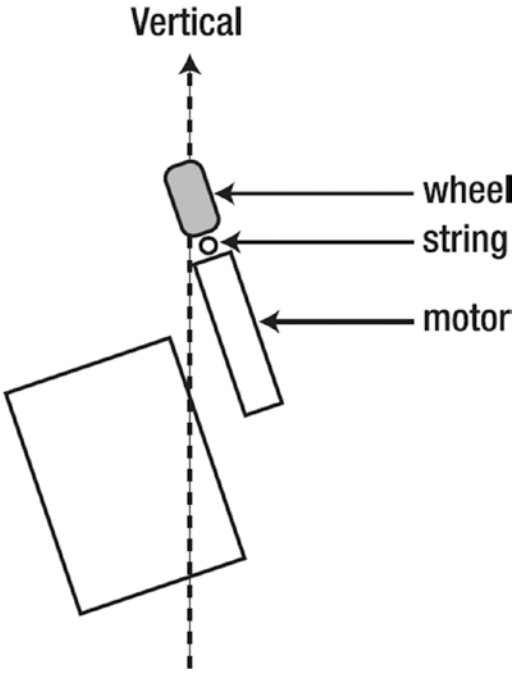


Figure 6-5. *A nonsymmetrical bot can lean*

The final constraint, "Object held must be small and not too heavy," is also a fairly important consideration. You can design the most lightweight bot on the planet, but if all you've got available for it to carry is a steel padlock or other heavy item, the bot isn't going anywhere.

The design of the bot should include some sort of "carrier" that has been built for a small item. Sure, it will take a lot of trips to fill a jar with smaller, lighter items, but the bot will also travel faster. A larger, heavier bot designed to carry a heavier item will take longer to make the round trip. Ultimately, it's your call. A heavier load will use up battery power faster because the motors will draw more electricity. But it could be argued that with a lighter object, it will take many more trips to the jar, which in turn will use up the batteries. (If you're really curious, try it both ways—design a HeavyStringBot and a LightStringBot and see which one solves the challenge faster.)

You might encounter more constraints as the project moves forward. Or maybe you've come up with a constraint or two not mentioned here. Perfect. Keep them in mind when designing your own version of the StringBot.

Mindstorm

I like this section. It's my favorite part of the process before I actually start grabbing and putting pieces together. When you are mindstorming, there are no right or wrong ideas. Just find a comfortable chair and start thinking about your bot. You've probably already got a ton of ideas floating around in your head; this is the time to put the best of them down on paper. And if you think you're not that creative, let me make another suggestion.

Sometimes I have difficulty "getting creative." I might be tired or just not in the mood to do some heavy thinking. If this happens to you, especially now that you're trying to get your StringBot design started, it can be frustrating. What I do when I find that my creative energies are not at full strength is to play a game called "Won't Work."

With "Won't Work," instead of focusing on solutions to the challenge, you're going to think about solutions that (ta-da) won't work. The idea is that if you're not feeling creative (typically considered a positive emotion), you can be anti-creative. (Okay, I made up that word, but it does work. Trust me.) By thinking about things that won't work, I typically begin to start finding things that *will* work . . . and then I'm thinking creatively. Here are a couple of examples of my "Won't Work" session:

- I'll never get the bot to cross the string by walking across like a human because that requires too much balance.
- I can't lower the bot into the jar (its weight might set off the trigger). If the weight isn't enough, I won't get the bot back for further attempts.

Your vision of the StringBot is probably going to be very different from mine. Whether you use the "Won't Work" trick or immediately start putting your thoughts down on the Design Journal page, you're beginning to finalize the design of your StringBot and how it will work.

Take a look at Figure 6-6 and you'll see the ideas I've written down for the `Mindstorm` section of the Design Journal page.

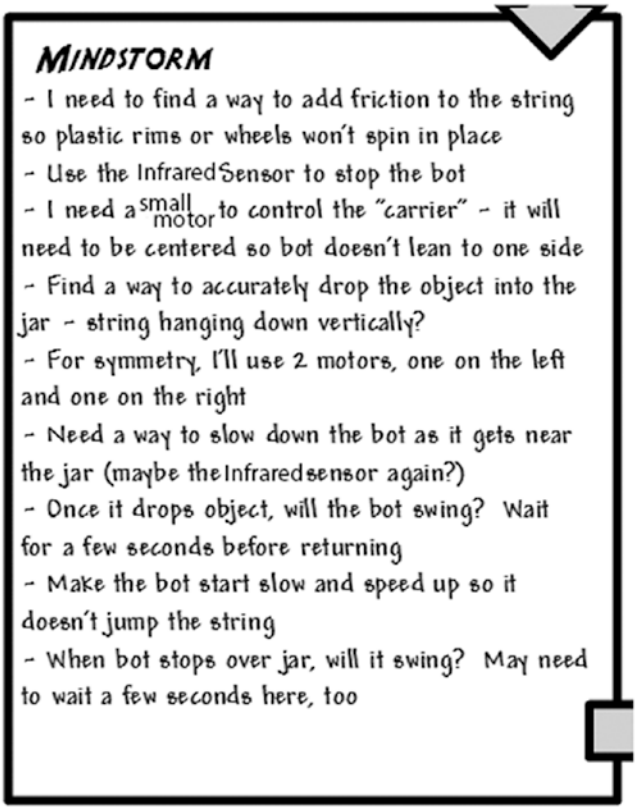


Figure 6-6. *The Mindstorm section will help you complete your StringBot design*

The first `Mindstorm` entry came to me fairly quickly—"I need to find a way to add friction to the string so plastic rims or wheels won't spin in place." I know from playing around with my MINDSTORMS EV3 kit that most of the pieces are made of very smooth plastic. My thoughts for the StringBot don't include hands and arms like a monkey. I plan on building a StringBot that moves along a string like a cable car (see Figure 6-7). I'm concerned that if I use one or two pulley wheels, the small wheels will slip on the string. When I begin to build, I'm going to try to incorporate one of the rubber wheels, because I think the rubber probably won't slip on the string.

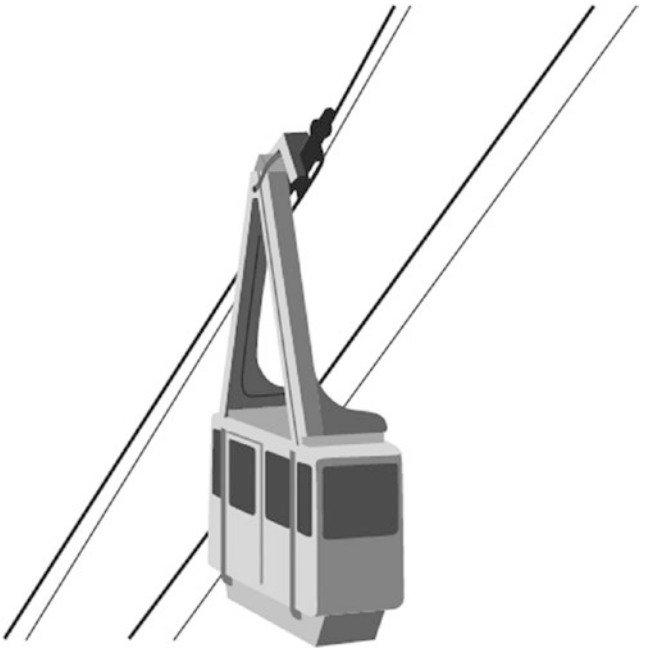


Figure 6-7. *My StringBot will work like a cable car. But will the small pulley wheels slip?*

I won't go through every item on my `Mindstorm` list, but there are a couple more items I feel are important for me to explain. The first is, "Find a way to accurately drop the object into the jar—string hanging down vertically?"

Most jars that I own have small openings at the top. I don't want my StringBot wasting time moving down the string and missing the drop. Maybe for your challenge setup, you'll use a large jar or box, but where's the fun in that? So it occurred to me that I better come up with a way for my bot to accurately drop its held object right into the jar. I'll obviously test my bot many times by observing where the object falls when the carrier releases it. What I think I'll do is tie a piece of loose string around my bot. Gravity will keep the string hanging straight down; when the string moves over the jar and is roughly in the center of the jar, I'll know to stop the StringBot and release the object.

Two other items in my `Mindstorm` box mention making the bot pause after certain actions. I believe (because I haven't tested it) that the bot will swing a little when it comes to a stop *and* after it drops the object. I want the bot to stabilize before it drops the object, and then again before it begins its return trip. If the bot is swinging, the dropped object might miss the jar. Also, a swinging bot might jump the string, and then it won't be able to return.

Your main objective here is to simply have some fun and write down some of your initial thoughts on what you'd like to do with your bot design. You may have to take a completely different direction after some testing. You may find that you exhaust your supply of a particular component. What you write down is not going to lock you in to a particular design. You can change the design anytime—even start completely over. Print out another Design Journal page and try a different design. It's supposed to be fun, so make it fun. Go crazy with your ideas—the crazier, the better!

The final `Mindstorm` item I want to quickly discuss is, "Make the bot start slow and speed up so it doesn't jump the string." I added this item because, in my earlier EV3 experiments, I've seen that when the motors start spinning quickly, sometimes the power of the motors can make a bot twist or "jump" and can slightly alter its direction. I offer this only as a suggestion—sometimes it's best to slowly increase the power of your motors while testing your bot to determine the best speed. Keep that in mind when you begin testing your StringBot.

Well, we're almost done with the Design Journal page. It's time to take everything you've collected—`Robot Description`, `Task List`, `Limitations/Constraints`, and `Mindstorm` information—and start with some rough sketches to help you when you begin construction.

Sketches

Even though we're not in the same room, I'll know if you're snickering. I told you before that I'm not an artist, so keep that in mind when you begin to look at my sketches of the StringBot.

Luckily, I know a little bit about what I'm looking for in terms of the shape and size of the StringBot. One of my constraints mentioned keeping the StringBot symmetrical. I know that if the bot isn't symmetric, it will lean to one side . . . and if it comes off the string, that will be bad, right?

In my `Mindstorm` box, I indicated that I want to use two motors, one on the left and one on the right. If I do this, I'll have to place that third motor near the middle of the bot to keep it from leaning. Since I'm going to use the idea of a cable car, I want the pulley wheels to be on top. And since I know my StringBot won't work without the Intelligent Brick, I've got to decide whether to have the Brick mounted horizontally, with the buttons parallel to the ground, or mounted vertically, so I can view the LCD screen as the bot moves away from me.

Okay, so it's time to draw. I tend to draw using basic shapes like rectangles, squares, and circles, so my sketches will consist of components drawn using their most basic shapes (the motor, for example, is two circles connected by a small rectangle). Figure 6-8 shows my initial sketches for the StringBot—no laughing!

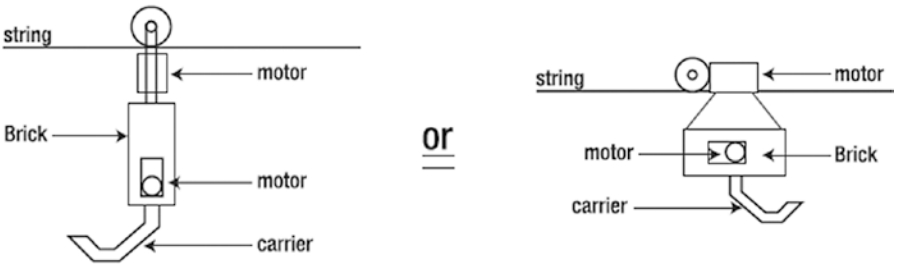


Figure 6-8. *I use basic shapes to define the StringBot in the Sketches area*

I've got quite a few concepts to test out when I start building. I might find that the vertical mounting of the Brick doesn't balance as well on the string as mounting the Brick horizontally. Or I might find that the weight of the third motor (for the carrier) isn't properly balanced and causes the bot to lean. Ultimately, if the StringBot moves to the end of the string, drops the object, and returns, I don't care what it looks like. (Okay, maybe I do a little—everyone loves a nice-looking bot.)

Chapter 7 shows you how to assemble my version of the StringBot. Compare it to my sketches and maybe you'll see the final version hidden in those rough drawings. Feel free to build my version. Or, if you're really proud of your design, go ahead and build YOUR StringBot. Remember, if your bot can fill a jar with dropped objects, you shouldn't have any trouble completing this challenge. Time to build the StringBot!

CHAPTER 7

StringBot: Build It

Figure 7-1 shows my version of the StringBot. It's a strange-looking device, but it does work. If you ever want to have some fun, try showing your bot design to others who are unfamiliar with it. Ask them, "What do you think it does?" and be ready for some unexpected answers! I showed this to a couple of people and the most surprising response was this: "How does it roll along the floor with only one wheel? The smaller wheels don't look like they'll help hold it up." Just goes to show that you cannot judge a book (or a bot, in this case) by its cover.

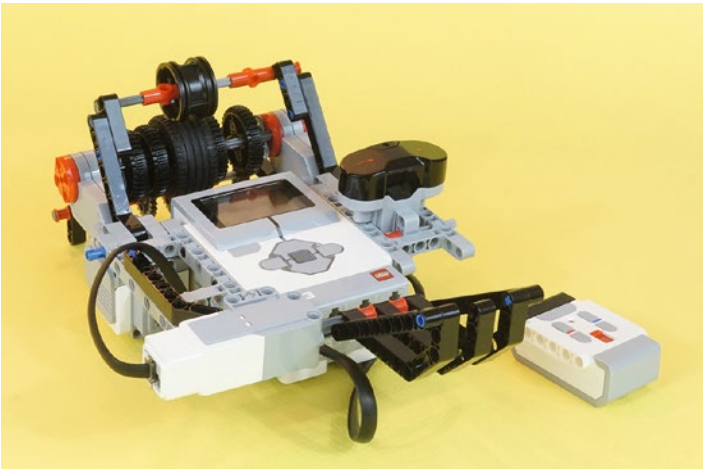


Figure 7-1. *Evan's version of the StringBot*

Where to Start?

Some people have no trouble just clicking pieces together. Others have more difficulty and find themselves sitting and staring at all those wonderful plastic parts with no idea where to start. I often find myself in both situations. Some days I'm ready to go, with ideas flowing so fast I have difficulty deciding which one to build first. Other days, I pick up a few pieces, snap them together, shake my head, and take them apart. It can be frustrating.

Hopefully by now you're starting to see the importance of the Design Journal page. It can help you to at least start building something—*anything*—with a basic shape and concept. Just sit down with your Design Journal page and look over the sketches. Read over the `Mindstorm` section and see if anything sticks out as a good place to start. I usually start with the Intelligent Brick. Since it's a required part for all bots, it just seems to me to be the best place to begin.

M. Bell and J. F. Kelly, *LEGO® MINDSTORMS® EV3*, DOI 10.1007/978-1-4842-2262-1_7

Before we move into the actual building instructions for the StringBot, I want to mention a few things that happened during my initial construction:

- I mounted the motors on the sides of the Brick, with the Brick parallel to the floor. It worked, but I couldn't see the LCD screen when the bot moved away from me. I chose to mount the Brick hanging vertically.
- I tried to run my StringBot on the string without the rubber wheel. As predicted, at high speeds the pulley wheels would just spin and the bot wouldn't move. I had to set the motors to a very low speed for it to move. It took too long to reach the jar and come back.
- I then used the rubber wheel, with a normal hub, to ride the string. But no matter what angle I ran at, the StringBot wouldn't really run successfully. It turned out that the weight of the robot made a large dent in the rubber tire itself. That gave a lot of traction but it behaved like an underinflated tire—too difficult for the motors to turn.
- I wanted to be able to open the motors apart to allow a string to be threaded onto the drive wheel, even with both ends of the string tied up. That is, thread it without having a free end to work with. But the first few designs of the axle assembly tended to fall apart mid-ride and drop the robot to the floor. The axle had to be strengthened, no matter what it took.

One thing I would suggest is to keep a builder's notebook (along with your Design Journal pages). In this notebook, you should write down observations while you are playing—I mean building robots. Write down things like you see in the previous list. The next time you decide to build a robot that possibly rides a string, a quick review of your notebook will remind you that the small pulley wheels aren't the best method. You can also use it to draw sketches of bots, special assemblies, and other things that you might use in the future.

This is the last time I will mention it (because most people don't like to be told things twice), but as you're following along with the building instructions, if you get stuck or can't figure out the next step, just take a short break. Look ahead at the next few figures in the chapter and see if you can use the different camera angles to determine the proper placement of a confusing piece. Trust me, I've built these bots over and over again to test the instructions. I've done my best to give you clear views, from the best angles, for where parts should be placed. Count holes on beams to determine where I've placed those small connectors (for example, 5 holes from the top on the 15-hole beam) or skip ahead and look at the final bot image. If your StringBot looks similar to this one with a couple of minor differences, that's okay! You might have found a better solution anyway!

You'll figure it out, so don't let the StringBot stress you out. You're still having fun, right?

■ **Note** I would enjoy seeing your modifications to the StringBot—or maybe you've created something completely different. Take a picture and e-mail it to me. My web site address is in the Introduction.

Okay, let's build the StringBot.

Step by Step

I've broken the StringBot building instructions into three sections, plus the final assembly. The first section consists of the right-side motor assembly with the IR Sensor. The second section adds the left-side motor assembly, with the rubber tire and the string guides. The third section covers the carrier arm and motor assembly that drops the pebbles. These three parts will be connected to the Intelligent Brick, wired up, and there's your StringBot.

I continue to add comments to the sections where I think a little help is needed. And when you see an image with text and arrows, pay special attention because there might be a tricky placement that you need to focus on.

With these instructions, I show you all the parts for each sub-assembly in the first figure for that sub-assembly.

First Section: Right-Side Motor Assembly with IR Sensor

Figure 7-2 shows you the components used to build the right-side motor assembly with the IR Sensor. Work up through Figure 7-11 to complete this assembly.

Figure 7-2. *All the pieces for the StringBot right-side motor assembly with IR sensor. The axles are five holes in length.*

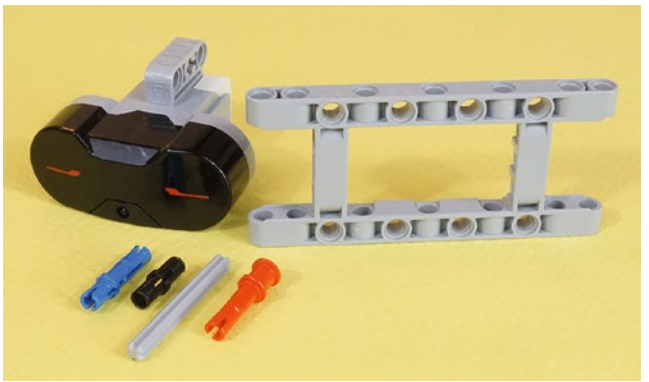


Figure 7-3. *To begin, start with this group of parts*

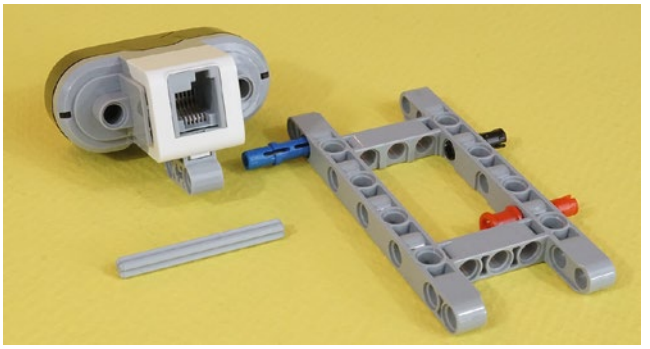


Figure 7-4. *In go the connectors. The blue connector is in only half way. The small black connector is in all the way.*

■ **Note** What I call the "small black connectors" have an official LEGO part number. I don't like using numbers to reference parts unless absolutely necessary—but if you must have this information, the part number is 2780. The LEGO community calls these "black friction pins."

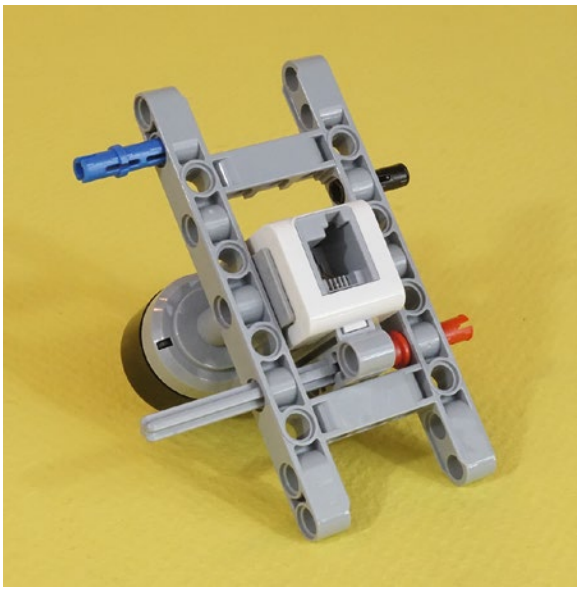


Figure 7-5. *The axle engages the sensor in its middle axle-hole, then is seated in the red connector*

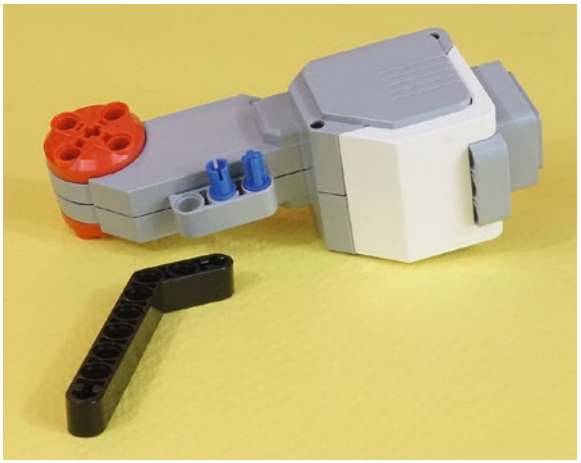


Figure 7-6. *Notice that these short axle connectors provide a very firm connection*

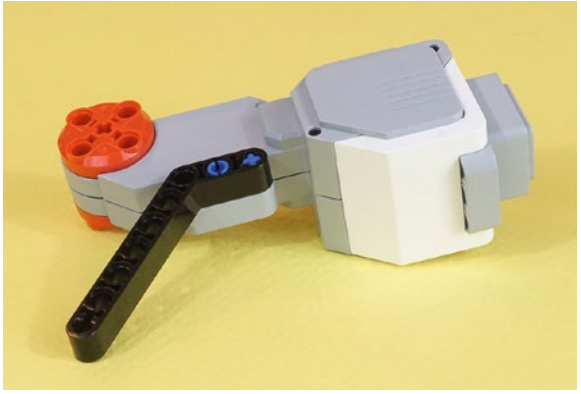


Figure 7-7. *On goes the angle piece*

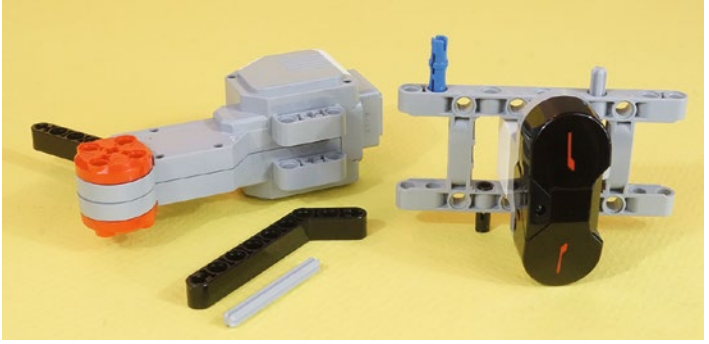


Figure 7-8. *These two sub-assemblies are about to be joined*

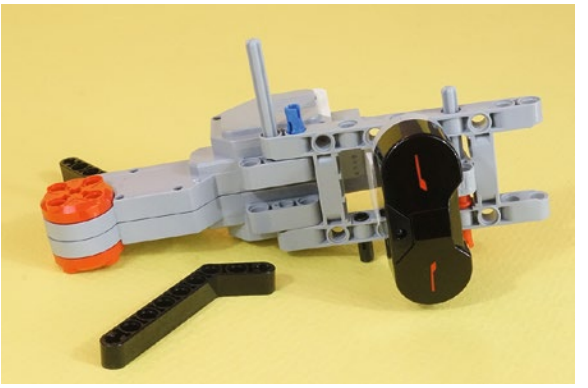


Figure 7-9. *Using this axle to join the parts gives added strength. Note that the axle does go through the axle-hole on the motor mount.*

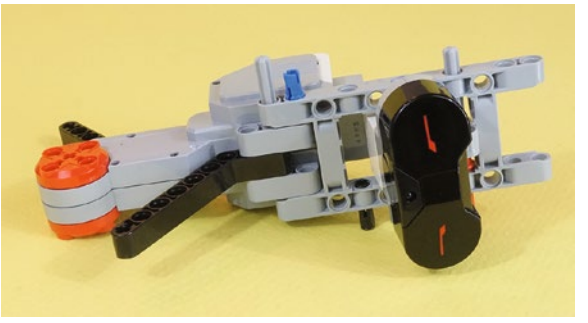


Figure 7-10. *Now the axle passes through the axle-hole on the angle beam. Axles are an effective, simple way to make strong connections with parts that have axle-holes.*

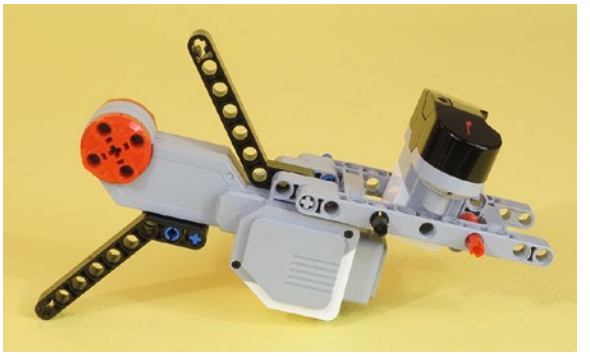


Figure 7-11. *The finished right-side motor assembly*

At this point, take a short break and compare your right-side motor assembly to Figure 7-11. Do they match? If not, try to correct the problem by going back through the figures to find the difference. You'll be connecting the motors to the Brick in very specific locations, so it's important that your framework match the one shown in Figure 7-11 as closely as possible. Okay, let's continue. . .

Second Section: Left-Side Motor Assembly with Guides and Rubber Wheel

Left-Side Motor Sub-Assembly

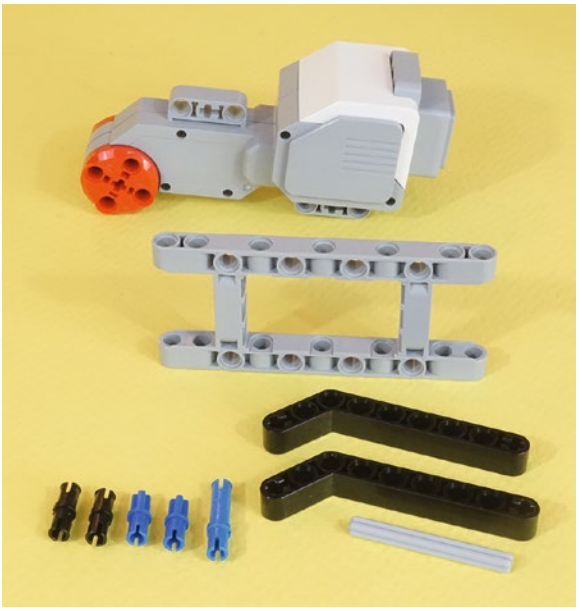


Figure 7-12. *These are all of the parts for this sub-assembly. These axles are the kind with five-hole length.*

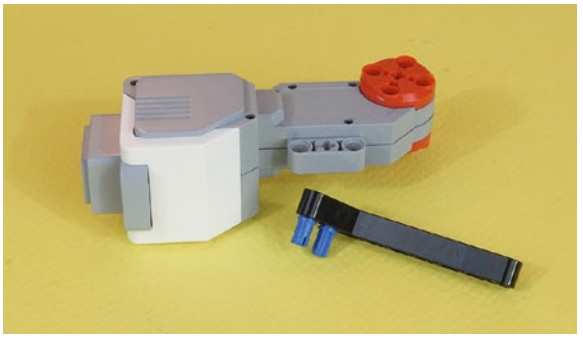


Figure 7-13. *Two blue half-axle connectors. One faces in to the axle-hole on the angle beam. The other faces out. This is a mirror image of the motor shown in Figure 7-7.*

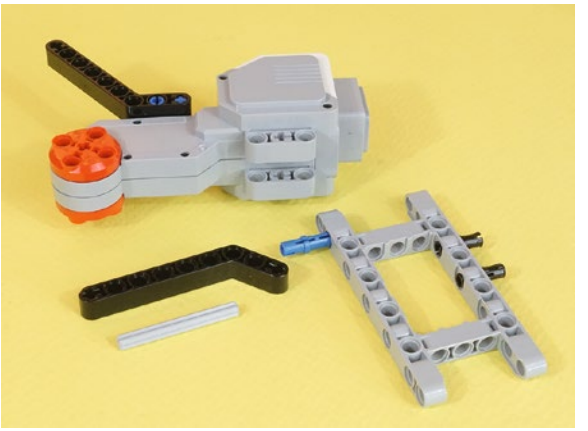


Figure 7-14. *Two black connectors go into the frame, and a long blue connector is inserted only half way*

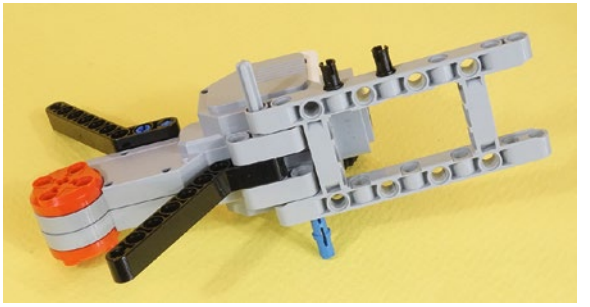


Figure 7-15. *This axle does the same job as seen in the right-side assembly in Figure 7-10*

Guides, Axles, and Rubber Wheel

Now we build the two guide axles. Four small rubber wheels would make sense, but the kit only has two. Gears can perform the same duty. Figure 7-16 shows an exploded view. The assembled view is shown in Figure 7-17.

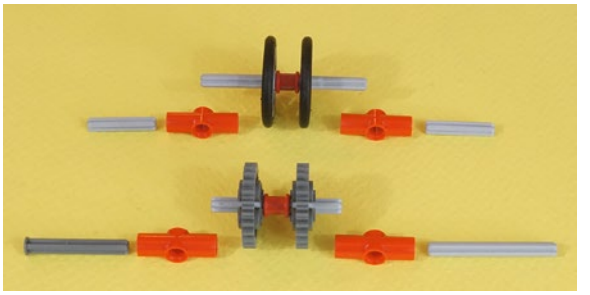


Figure 7-16. *These are all of the parts used for these guides. There are two three-hole axles, two five-hole axles, and one seven-hole axle. The nailhead axle has a length of four holes.*

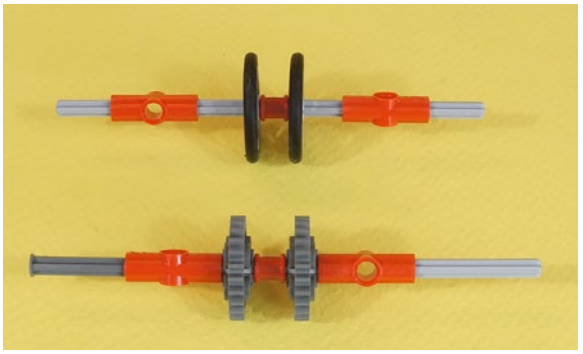


Figure 7-17. *Here are the assembled guide axles*

Figure 7-18 shows the parts used to make the main drive axle with its rubber tire. Now, this is a case where you might think, "Why do we need all of those parts for a simple axle with a tire?" Glance ahead to Figure 7-28 to see the completed drive axle. This abundance of parts ensures that the bot can hold the weight of the Brick and the three motors. If we simply used two long axles pressed into a center wheel, they would most definitely separate after cruising five or ten feet on the rope. I described trying this in the "Where to Start?" section at the beginning of this chapter. The bot wants to come apart at the center of the axle because all the force of gravity is pulling on it and trying to separate the two halves of the bot.

So let's follow all of these reinforcing parts to where they go and see how strong the center axle becomes!

Figure 7-18. *All of the axle parts, including lots of reinforcements. Five gears. A nine-hole axle. Five six-hole axles. But where is the hub for the tire?*

Figure 7-19. *For the hub, use these two gears instead. They make a stronger connection to your axles and reinforcement parts.*

Figure 7-20 shows the new, reinforced tire with two gears stuffed inside. The gears are lined up with each other. The other major benefit from this reinforced tire is that it does not compress from the weight of the StringBot on the string. If the tire did compress (that is, get dented from the string), it would create much more friction. That's just like running a car on underinflated tires—it's more work for the motors. When I tried it, it was *too much* work for them and they could not propel the bot along the string.

Figure 7-20. *The rubber tire with an "artificial hub" made of two gears. Line up the holes in each one so they match.*

Figure 7-21 shows another sub-assembly that will mate with the tire hub for a very strong joint. There are three red collars, more clearly visible in Figure 7-23. These gray connectors are among the strongest pieces in your kit. There are also two additional axles, visible in the next several figures.

Figure 7-21. *The double gray connector about to be fully inserted in the third gear. See Figure 7-23 for a side view of this part.*

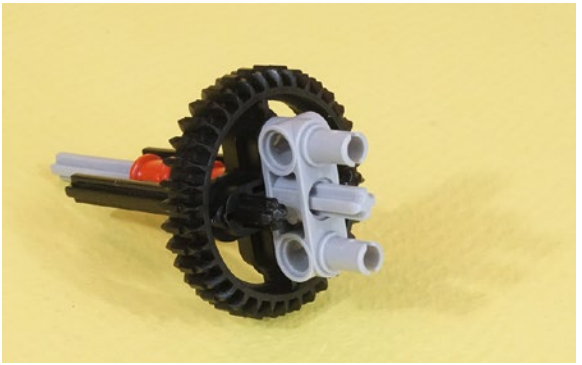


Figure 7-22. *When the gray connector is seated, the axle should be level with the gray pins*

Figure 7-23. *Another view of the three-axle gear assembly, about to be inserted into the three axle-holes of the tire*

In Figure 7-24, notice that the three axles go into the hub axle-holes, not the round holes.

Figure 7-24. *The three-axle assembly partially inserted. Push it in until the collars stop it.*

Figure 7-25. *Now make this similar three-axle assembly for the other side, using the fourth gear. No red collars on this one.*

As Figures 7-26 and 7-27 show, the two side axles will go into the tire hub axle-holes and the dual gray connector will go into the plain holes. You'll find there's only one way it will actually fit, so you're okay no matter what. The other thing you'll find is that it makes an exceptionally strong connection. It will not fall apart under stress. In Figure 7-26, the gear on the left has also received a dual gray connector. The three-axle assembly from Figure 7-25 is now in the center of Figure 7-26.

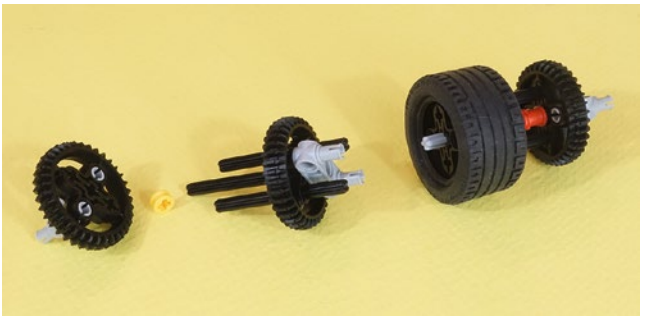


Figure 7-26. *The second three-axle assembly ready for insertion into the tire hub. On the left is the fifth gear with the last double gray connector in place.*

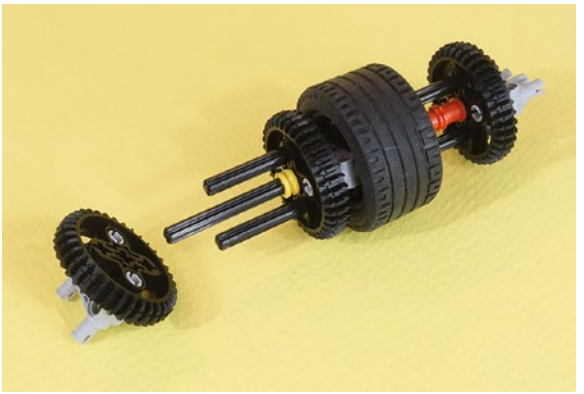


Figure 7-27. *Insert this final three-axle connection and the main drive hub is done*

The force of the string, with the weight pulling down, makes axles want to break apart at the center. But this reinforced main drive axle will resist that force. It also resists coming out of the motors themselves, since each end has *three* connections to the motor hub, not just one skinny axle.

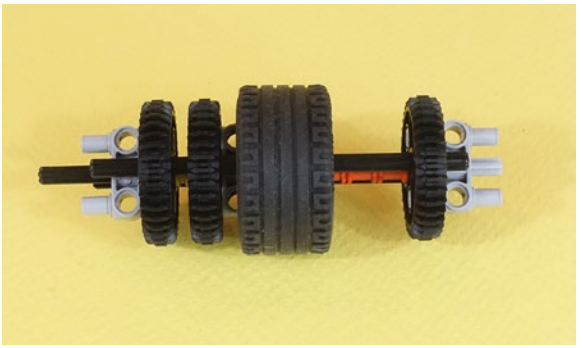


Figure 7-28. *This is one seriously strong axle, with all five gears in place*

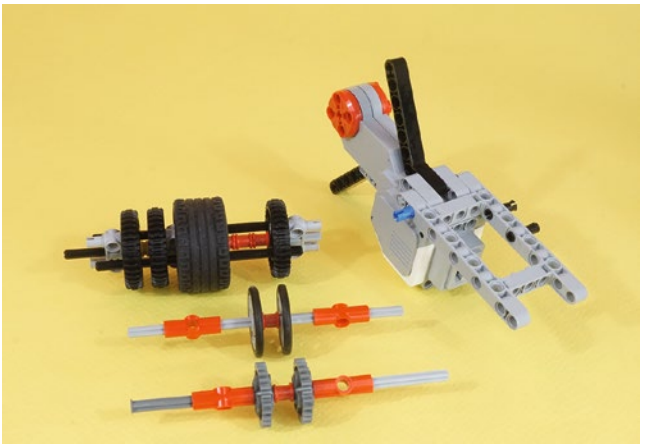


Figure 7-29. *Now to build the whole left-side motor assembly*

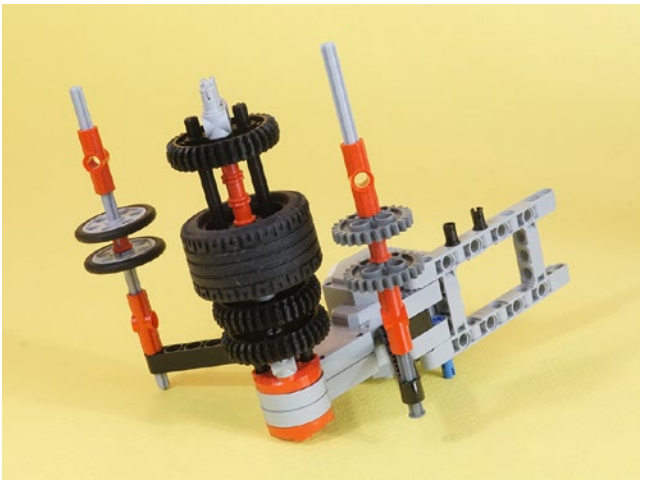


Figure 7-30. *Here it is. You'll need to undo the dark gray nail-head axle to put it through the angle beam.*

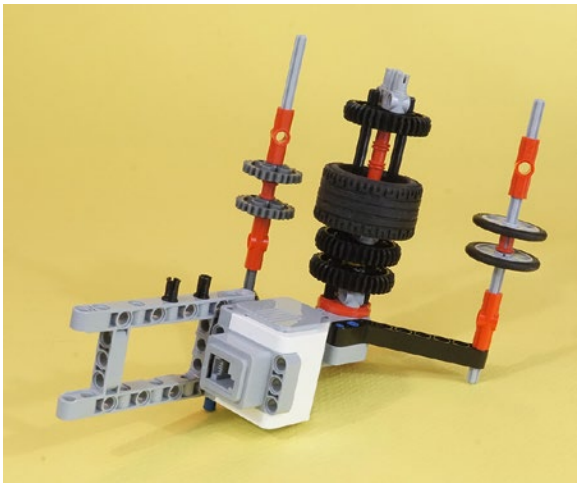


Figure 7-31. *Another view of the left-side motor assembly with axles and the rubber wheel*

Okay, the StringBot is almost finished. Are you beginning to see how it will work? You'll feed a string over the front string guide. The string will then go under the rubber wheel, and then on top of the rear guide. The guides will keep the string centered on the rubber tire (which has excellent traction), and your StringBot will reliably move along the string.

That covers back and forth movement on the string, but you still need to be able to drop those small objects in the jar. That's accomplished with the carrier motor and basket. It lets your StringBot carry a pebble or coin to be dropped in the jar. The medium motor will be used to control the carrier, and we'll fasten it on the side of the left motor assembly.

Figure 7-32 shows the pieces you'll need for the carrier basket itself, and Figure 7-35 shows the pieces needed for the motor mount. Go ahead and grab them and let's start building.

Third Section: Carrier and Pebble Release Mechanism

Two nine-hole beams and three large angle beams are shown in Figure 7-32. You'll use these to make the carrier basket that holds the pebble until it is dropped.

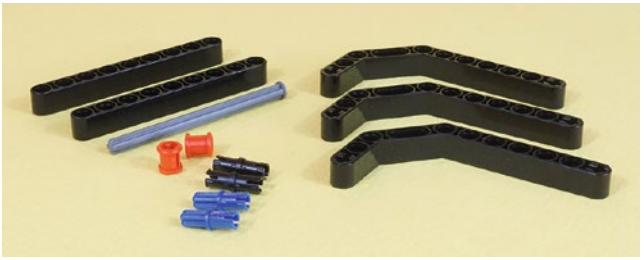


Figure 7-32. *All the parts needed for the carrier basket*

Figure 7-33. *These three beams form an open basket. This is lighter than using five beams.*

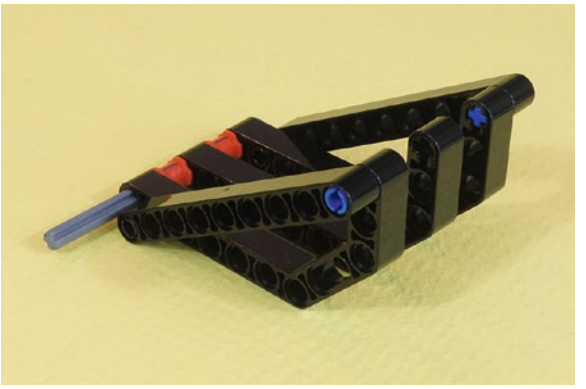


Figure 7-34. *The completed basket*

Figure 7-35 shows the medium motor and its frame pieces. This will result in a simple, reliable assembly, with the carrier basket centered under the StringBot. This is the first time you've seen the medium motor. It serves us well here because of its compact size, even though it lacks the power of the large motors. It also has a rotational encoder, just as the large motors do. (That is, the motor keeps track of how many degrees it moves and reports that information to the Brick.)

Figure 7-35. *All the parts for the motor frame. The axle is a three-hole length type.*

In Figure 7-36, notice that this is another case of using an axle for superior strength with a minimum of parts. Again, you'll gain more strength from axle connectors than an equivalent number of simple round connectors. Axles are an under-appreciated asset in building strong, simple bots. The connections do not twist, making them stronger than round connectors.

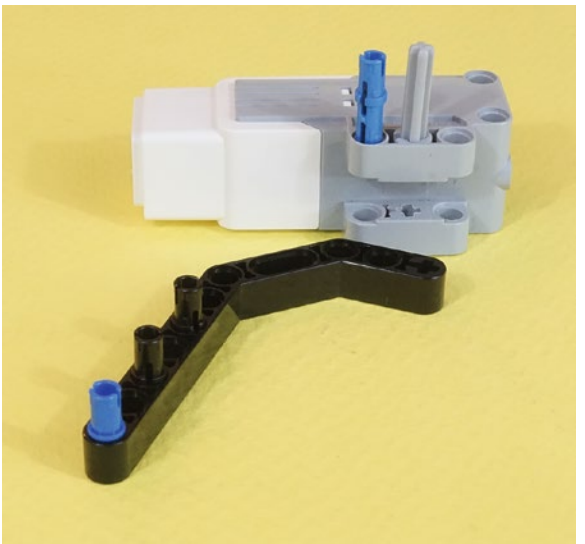


Figure 7-36. *Place the connectors as shown*

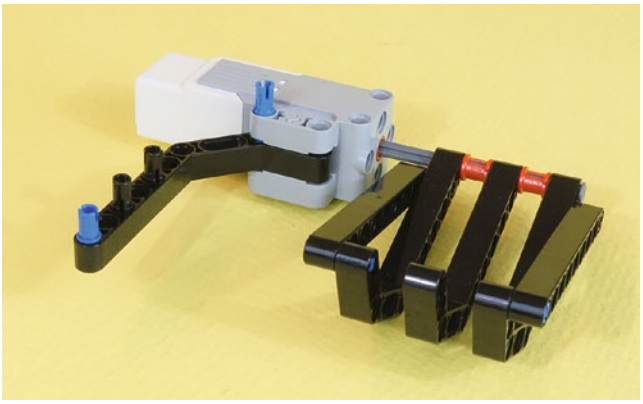


Figure 7-37. *The completed carrier assembly*

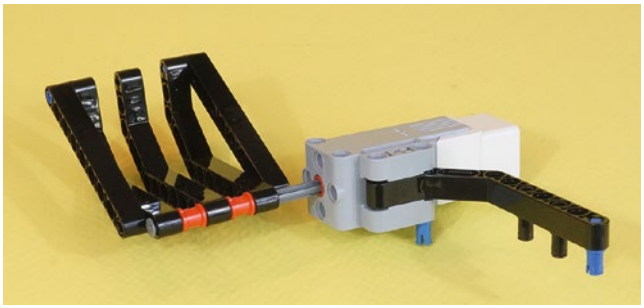


Figure 7-38. *The carrier assembly with the basket in drop position*

Another reinforcing piece for the front of the robot—an 11-beam and two connectors. They are not strictly necessary, but protect against the two motor frames trying to pull apart.

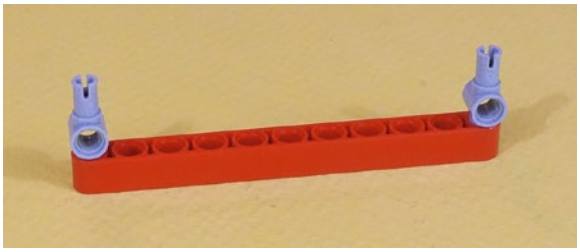


Figure 7-39. *Front reinforcement beam*

Fourth Section: The Intelligent Brick and Putting It All Together

In Figure 7-40, the Brick picks up four connectors as shown. One goes to each outer front corner.

Figure 7-40. *The Intelligent Brick makes its appearance! Connectors go into the outer four corners*

Now let's gather all the assemblies and four short wires, as shown in Figure 7-41.

Figure 7-41. *All the assemblies, wires, and the IR remote control*

For final assembly, we'll make it look like Figure 7-42. Attach the left-side motor-axle assembly to the Brick. Each side has two connectors so there should be only one way it can go together. Then connect the medium motor carrier assembly to that left-side rectangular frame, so the basket is directly below the Brick. We'll attach the wires next.

Figure 7-42. *It's starting to look like a StringBot. The front reinforcement 11-beam goes on last; it's shown for completeness.*

Next, select the short wires and attach them to the motors, the IR Sensor, and the ports, as shown in Figure 7-43. Table 7-1 describes where each wire goes.

Figure 7-43. *Wiring your StringBot*

Table 7-1. *Wire Assignments to Ports*

Port	Destination: Motor or Sensor
A	Medium motor—carrier
C	Large motor—left side
D	Large motor—right (IR) side
1	Infrared (IR) Sensor

Figure 7-44 is a close-up of how the right-side motor is connected to the Brick and the axles. It's a bit tricky to line up the connectors and axles all at once. What may be easiest is to align the connectors with the Brick, begin to insert them, and then concentrate on the main drive axle. Since it has two connectors and its own axle that attach to the orange motor hub, it needs to be aligned to go in at all. If you get it partially connected, you can bend the thin guide axles and contort them in. Then you snap the whole side shut. Or, get all three axles in place and then snap the motor body to the Brick.

Figure 7-44. *The right-side motor clips on to your StringBot*

The advantage of this design is that you can remove the motor assembly to thread your string onto the drive wheel, even if the string is attached at both ends. And since the main drive axle is strong, it should not fall apart in the center.

And that's it! You are now the proud owner of your very own StringBot. You still have the programming to complete, but at this point if you have some string, you can place your StringBot on the string and test to see how it will roll along.

What I suggest is tying one end of the string to a doorknob (on a closed door, of course). You'll feed the string through the StringBot and then tie the other end of the string to something heavy or possibly another doorknob. Cut yourself a long piece of string so you'll have plenty of options. And my last bit of advice: tie the string so that it is as tight as you can make it with the StringBot sitting on the string. If there's much slack in the string, the StringBot will slide toward the middle. It also might lack the power to move along the string.

Figure 7-45 shows how I've fed the string through the front string guide, under the rubber tire, and then out the rear string guide. This bot has the IR sensor facing forward. When you write your EV3-G program to control it in Chapter 8, you should be able to face the IR sensor toward your vase and away from your IR Remote. It actually works by seeing the *reflection* of your IR beam. But what if your far wall, with the vase, does not reflect very well? You would need to change the design, placing the IR sensor facing the back. Or, change the program so the front becomes the back!

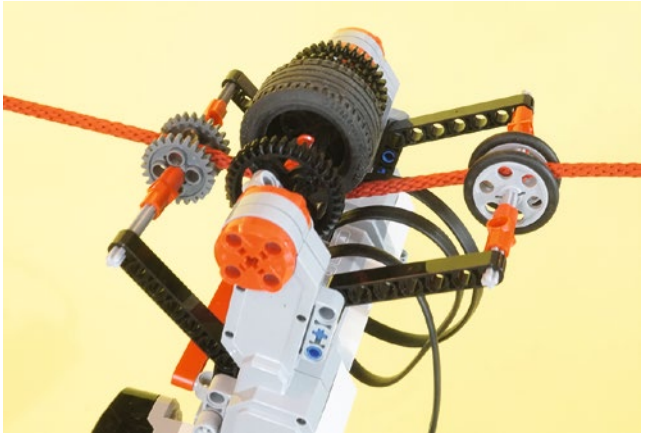


Figure 7-45. *String up your StringBot*

Once you have the StringBot programmed, this little fellow will move forward on the string, stop when signaled to stop, and then move slowly toward the jar until signaled to stop for the final time. At this point, the carrier arm will drop whatever object it is holding, pause for a few seconds, close the arm, and then return to you. Simple!

And how do we signal it to stop? The IR sensor is mounted in front, so it could measure the distance to the wall. That worked great for ExploroBot. But what if we don't know that distance? Is there another way we could signal it to stop?

Chapter 8 shows you how to program the StringBot to do all this and more. I'm going to give you some ideas on increasing the functionality of the StringBot with some additional programming. Once you have your StringBot working, you'll fill that jar with some coins or small rocks to trigger the pressure plate. The traps on the floor won't be dangerous anymore and the expedition crew will be able to cross the room and continue investigating the tomb.

CHAPTER 8

StringBot: Program It

The StringBot is a unique little bot. When you ask someone to describe a robot, my guess is that he or she will probably have a mental picture of a wheeled robot or a bot with legs. Our StringBot has neither—it's going to move along a string. But whether your bot has wheels, legs, or some other method for moving, it has to be programmed properly to accomplish its desired tasks. The StringBot needs to move along a string, successfully drop a small object into a jar, then return to you. This chapter will show you how to make that happen.

Get Familiar with the Blocks

In Chapter 4, you used the large motors with the `MOVE STEERING`, `WAIT`, and `LOOP` programming blocks to get the ExploroBot to complete its task. For the StringBot, you're going to use these same blocks. With the ExploroBot, you used the Infrared Sensor to measure distance, but with the StringBot, you're going to use the Infrared Sensor in a new way—with the remote control.

Just like you started doing in Chapter 4, I'm going to ask you to refer back to your completed Design Journal page for the StringBot to help determine the proper blocks to use. Go ahead and get the StringBot Design Journal page out and open the LEGO MINDSTORMS EV3 software (see Figure 8-1).

M. Bell and J. F. Kelly, *LEGO® MINDSTORMS® EV3*, DOI 10.1007/978-1-4842-2262-1_8

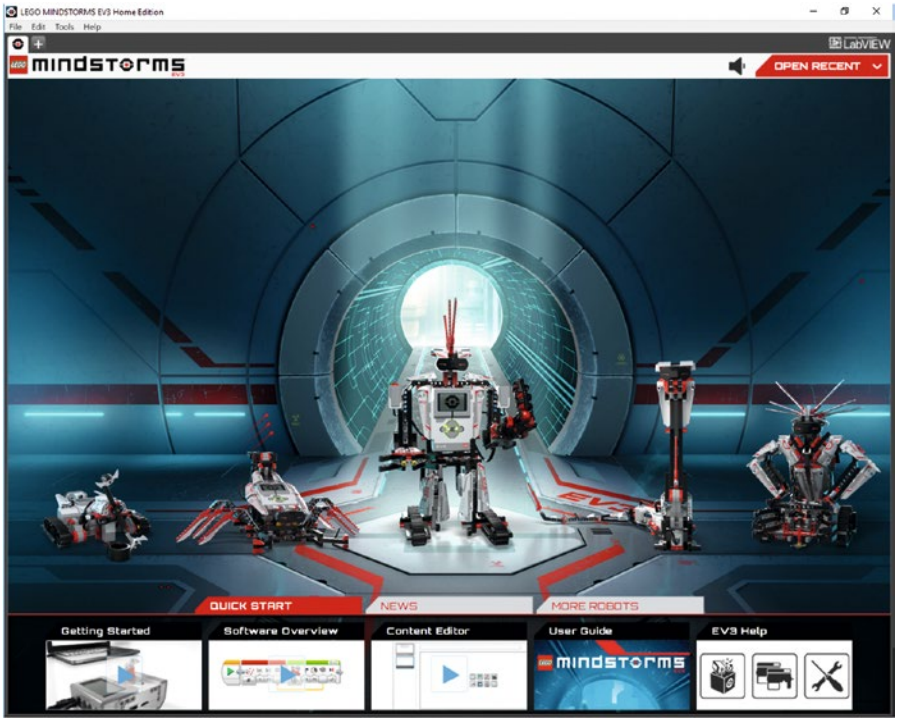


Figure 8-1. *The LEGO MINDSTORMS EV3 software*

Because you're creating a new program, go ahead and type **StringBot** into the blank text field labeled Start New Program and then click the Go button (Figure 8-2 shows the result).

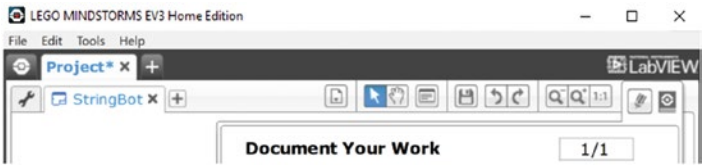


Figure 8-2. *Enter StringBot for the new program name*

■ **Note** To have more workspace visible on your screen, minimize the Document Your Work area on the right by clicking the EV3 Button symbol at the far right of the top tab bar. (This symbol is shaped like a little stop sign—that is, an octagon. It resembles the buttons on your EV3 Intelligent Brick.)

Take a look now at the `Task List` on your Design Journal page (see Figure 8-3). Our `Task List` wasn't as long as the one for the ExploroBot, but we still need to go back through it to help us with the bot's program.

Figure 8-3. *Use the Task List to help with programming the StringBot*

Let's go down the `Task List` and place potential programming blocks next to each step. This might help you understand what the final program's structure will look like. At this point, you're just guessing about the blocks you might use, but at least this will get you thinking about what could be needed.

1. Wait, sitting on the string, for object to be loaded: `WAIT`, `LOOP`
2. Move forward on string: large motor `MOVE STEERING`, `LOOP-UNTIL-INFRARED-SENSOR`
3. Hold object while moving forward: `MOVE`
4. Stop over jar: `STOP`, `WAIT`
5. Drop object: medium motor `MOVE`
6. Move backward on string: `MOVE`
7. Stop and wait for another object to be loaded: `MOVE`, `LOOP-UNTIL-INFRARED-SENSOR`

I also reviewed the `Mindstorm` section of the Design Journal page, and there are a couple items in there that I'd like to implement using programming:

- We need a way to slow down the bot as it gets near the jar
- If the bot might swing, we need to add some wait time before beginning a new task

We'll keep these in mind as we begin to assemble our StringBot program. We're going to use the same blocks covered in Chapter 4: `MOVE`, `LOOP`, and `WAIT`, plus a new block, called `STOP`.

The STOP Block

The STOP block is straightforward (see Figure 8-4). When you place the STOP block in your program, the bot will cease all function when it reaches that block. No further programming blocks will be executed by the Intelligent Brick.

Figure 8-4. *The basic STOP block*

The STOP block doesn't need to be configured. There are no special settings for it. Second, if any motors are running when the STOP block is reached in the program, they'll coast, not brake. The final item I want to mention about the STOP block requires a short diversion in our discussion. I want to introduce to you another programming block: the SWITCH block. (We'll also take a brief glance at the SOUND and DISPLAY blocks.)

The SWITCH Block

The SWITCH block will allow you to program your bot to perform actions based on input conditions. For example, the most basic SWITCH block will test for two conditions. If one condition is met (TRUE), then one path of the SWITCH block will be taken. If the second condition is met (meaning the first condition is FALSE), then the second path of the block will be taken. We'll use the SWITCH block for a later bot design, but for now I want you to look at Figure 8-5. In this example, the SWITCH is inside a LOOP to run forever. The SWITCH has been programmed to test the Infrared Sensor. It will run a MOVE STEERING block if the Infrared Sensor is closer than 50 centimeters, and a big smile display if it is farther than that.

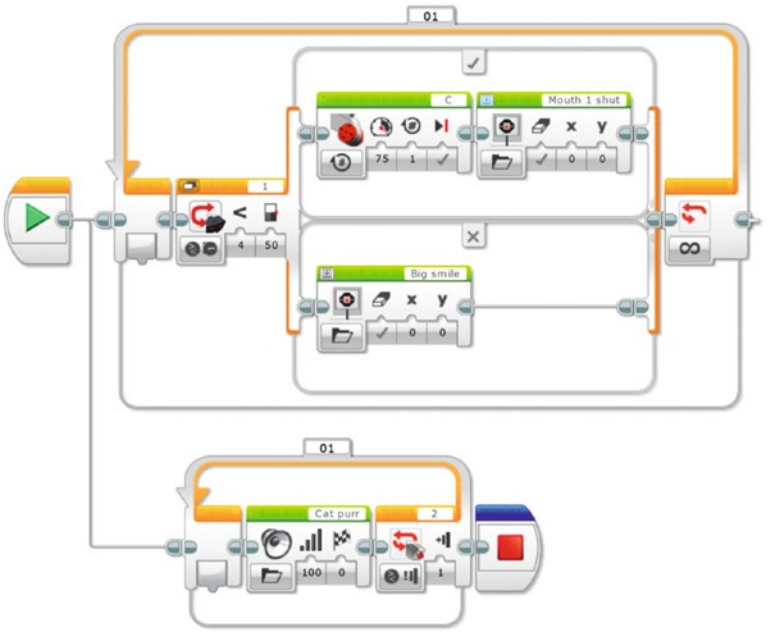


Figure 8-5. *The basic SWITCH block and a STOP block*

Below that is a LOOP followed by a STOP block. This LOOP will play the Cat Purr sound until the Touch Sensor on port 2 has been pressed.

When the Touch Sensor is pressed and the STOP block is executed, the Infrared Sensor LOOP and its motor will (you guessed it) stop on that STOP block!

■ **Note** The SWITCH block is located under the Flow Control tab. As you're exploring the blocks, you can hover the mouse over a block for a moment and a small "hint" message will appear telling you the name of that block. This mouse hover also works for the colored tabs, telling you the green tab is Action, the orange tab is Flow Control, etc.

Okay, let's get back to programming the StringBot.

Getting to the Vase

As you begin to put the program together, I want to suggest that you start adding comment text as you move forward. The comments you add will help you remember not only *why* you're using a certain block, but they can also be used to tell you how a block is configured. In our remaining programming figures, I'll include comments so you'll see what I mean.

Now, what was that first `Task List` item? Fairly simple: wait, while hanging on the string, and load an object into the carrier. You're going to use a LOOP block, but this one is a little different from the one in Chapter 4. With this LOOP block, instead of testing for a sensor to be triggered (such as the Infrared Sensor or Touch Sensor), you're going to have the block wait for a button on the Brick to be pressed. (You could use a simple WAIT block configured to wait for a button to be pressed, but you're using the LOOP block because you can later add more blocks inside the LOOP block if you want, and add more options for your bot later on.) Take a look at Figure 8-6 to see how I did this.

In Figure 8-6, I've configured the LOOP block to check to see if the Center button (middle dark gray button) on the Brick has been pressed. Once the button is pressed, the rest of the program will begin to execute. I could just as easily have included a Touch Sensor on my StringBot and had the LOOP block test to see if the Touch Sensor were pressed. But because I wanted to keep my StringBot lightweight, reducing the number of sensors was important. The Brick has to be included, so I might as well use its buttons in place of the Touch Sensor, right?

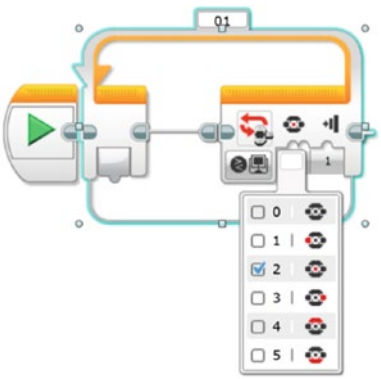


Figure 8-6. *Using a LOOP block to hold the StringBot until an object is placed in the carrier. The blue highlighting and pull-down menu appear when you click on the tab below the EV3 button symbol.*

So, while the StringBot is sitting on the string waiting, we'll place an object in the carrier. Once the carrier is loaded, the next step is to start the StringBot moving along the string. If you remember, one of our conditions was that we wanted the StringBot to slow down as it neared the jar. We then want the motors to spin again, but slowly, so that we have more control over the accuracy of putting the bot over the jar. To do this, you're going to use LOOP blocks again. The first LOOP block will run the motors at high speed and the second LOOP block will run the motors at a reduced speed.

This bot has the IR Sensor facing forward—that should let us use the same distance sensor idea that worked for our ExploroBot. All we should have to do is enter the distance between the vase and the far wall. Go fast until a few feet away, then slow down until the exact distance is reached, directly above the vase.

But there's a problem. We don't know what that distance is. Could we find a way to monitor the bot by eye and remotely command it to drop the object? The Remote Infrared beacon lets us do exactly that! The only question then is, if the IR Sensor on the StringBot is facing forward, how can it "see" the signals from the remote? Infrared light, like any light, will bounce off of walls. The reflections should be able to tell our Infrared Sensor when to slow down and stop.

The first step, then, is to use your IR remote (with the Infrared Sensor) to control when the bot switches from high speed to low speed. You'll place the first LOOP block, as shown in Figure 8-7.

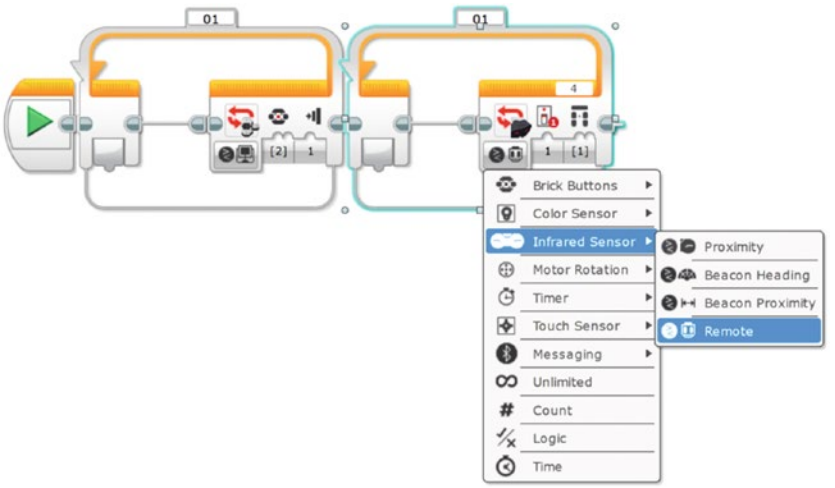


Figure 8-7. *This LOOP block will allow you to stop the high-speed bot with your IR remote. The LOOP pull-down menu is shown, programming the Infrared Sensor to look for remote commands.*

In Figure 8-7, I've configured the LOOP block to wait for the Remote Infrared beacon. I set the pull-down to 1 so the LOOP will end when the Infrared Sensor receives a press from the remote's upper-left button. Now you need to place the MOVE STEERING block so the motors will run at the high speed of 80 (see Figure 8-8).

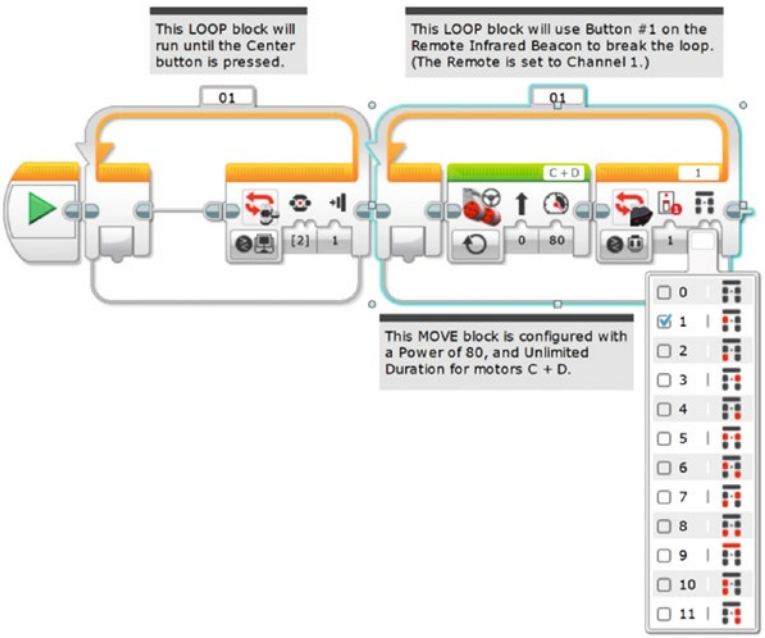


Figure 8-8. *This MOVE STEERING block runs the motors at high speed. Clicking 1 in the pull-down menu gives us button 1 at the upper left on the remote.*

After the Infrared Sensor is triggered, you want the motors to stop. Figure 8-9 shows the program so far.

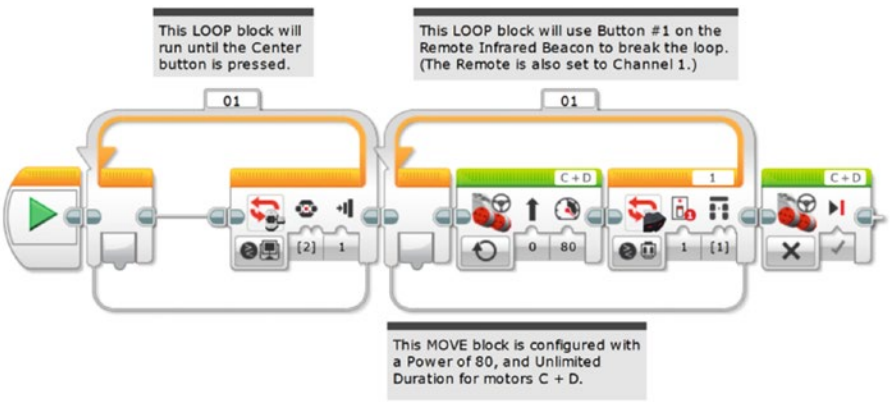


Figure 8-9. *Now the motors need to stop*

Next, you want the bot to pause for a short period just in case it's swinging on the string. Drop in a `WAIT` block for this and configure it for three seconds (see Figure 8-10).

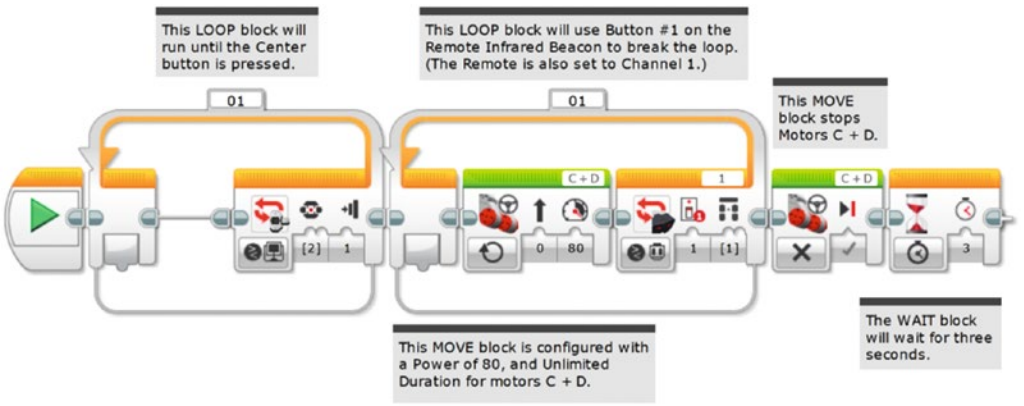


Figure 8-10. *The WAIT block gives the bot time to stop swinging*

Now that the bot is stopped, you need to configure the program to start the bot moving again, but this time at a slower speed. This is done just as before, with a `LOOP` block configured to break when the Infrared Sensor again receives a signal from the remote's upper-left button.

Configure the `MOVE STEERING` block inside it at a slower power setting of 40. Figure 8-11 shows the final `LOOP` block. When you test it, you may decide to make those motors go even slower. That's what I did in the "Testing" section at the end of this chapter—I set them to a speed of 20.

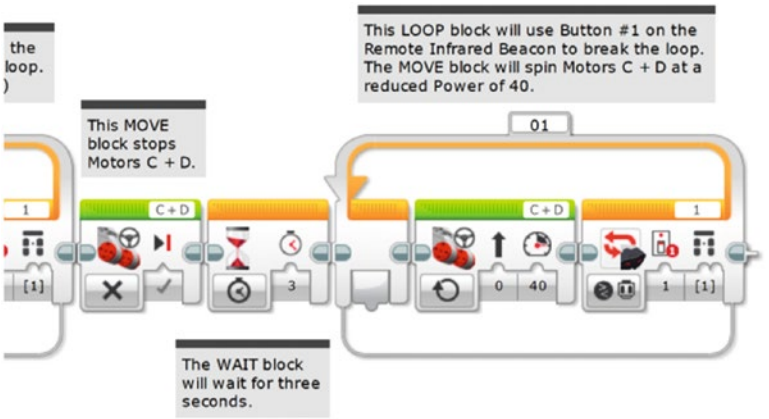


Figure 8-11. *The motors run at a slower speed until the Infrared Sensor is again triggered*

When you exit this `LOOP` block, you need to stop the StringBot's movement and give it time to stop swinging. What you'll do first is throw in a `MOVE STEERING` block to stop the motors (see Figure 8-12).

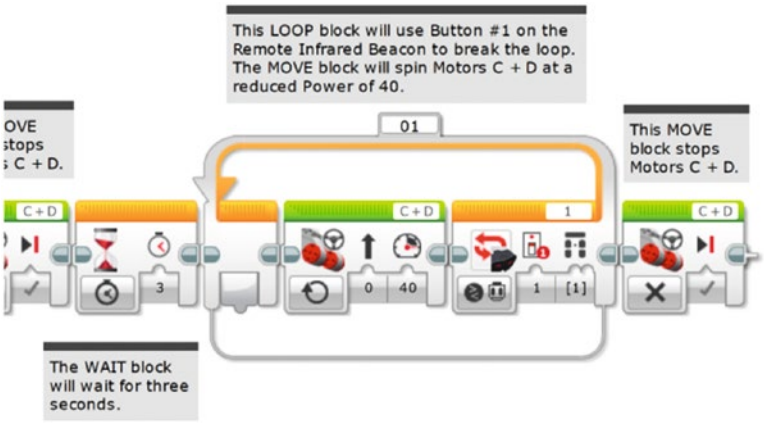


Figure 8-12. *Now stop the motors—the StringBot should be directly over the vase*

Your StringBot should be directly over the vase at this point, but it might be swinging. Add a `WAIT` block (see Figure 8-13) before releasing the object from the carrier—a five-second wait ought to do it.

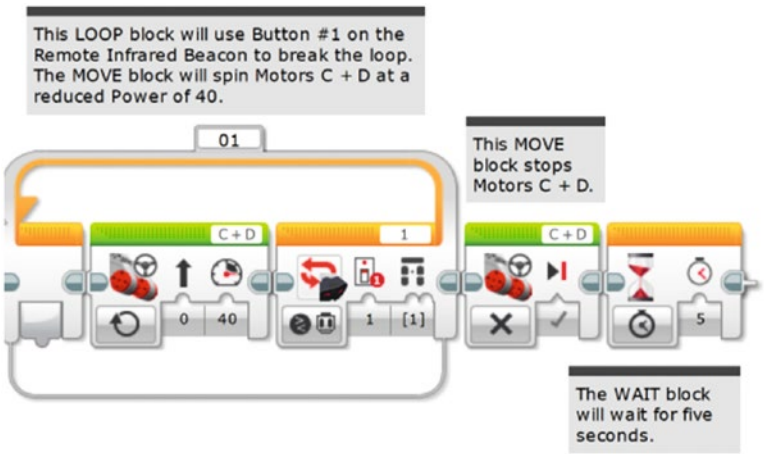


Figure 8-13. *Add a WAIT block to allow the StringBot to stop swinging above the vase*

Above the Vase

Now you have the StringBot placed directly over the vase. That wasn't too bad, was it? Well, the next part is even easier. All you need to do is have the carrier release the held object and return to its starting point, ready for another object to be loaded.

During my testing, I found that if I had the carrier move too quickly, I didn't have very accurate control when the object dropped. I began to tweak the speed of the motor and found that a slower release resulted in a more accurate drop. For some of my tests, I was dropping quarters and nickels. I experimented with dropping multiple coins at the same time; sometimes both coins dropped in the vase, sometimes neither did. You'll have to experiment with your own StringBot to determine what you want to drop and how many you want to attempt to drop at the same time. I have also tried dropping Ping Pong balls and golf balls.

The other factor is how far back you want the carrier to spin—experimentation with my StringBot revealed that a minus 90-degree rotation was sufficient for the object to fall out of the carrier and into the vase. You might also need to experiment to determine the proper rotational setting.

■ **Note** These motors will get your bot stuck if they cannot complete the rotations in your program. Let's say you program a motor to rotate 90 degrees, but it cannot rotate all the way because something's blocking it. The motor *does not* just "give up." It will keep straining to complete the command. If it cannot finish the 90-degree turn, the program remains on that instruction. If this happened to the carrier motor on your StringBot, it would be trapped at the end of the vine and never return. You always need to be sure the carrier is set in the correct position before it leaves.

Take a look at Figure 8-14. I've added the `MEDIUM MOTOR` (Move) block, which will tell the medium motor to slowly lower the carrier. For now, I've set the speed of the medium motor to 20.

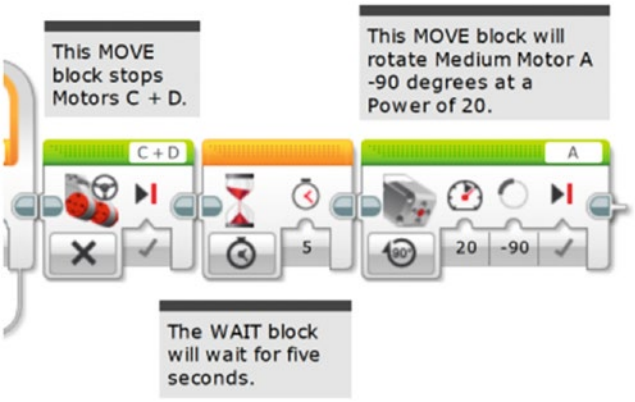


Figure 8-14. *Use a MEDIUM MOTOR (Move) block to control the carrier's release speed*

Okay, so now that the carrier has released its object, let's wait a few seconds and then close the carrier. Technically, we don't have to close the carrier, but I just don't like the look of the StringBot with the carrier hanging down.

First, add a `WAIT` block (see Figure 8-15).

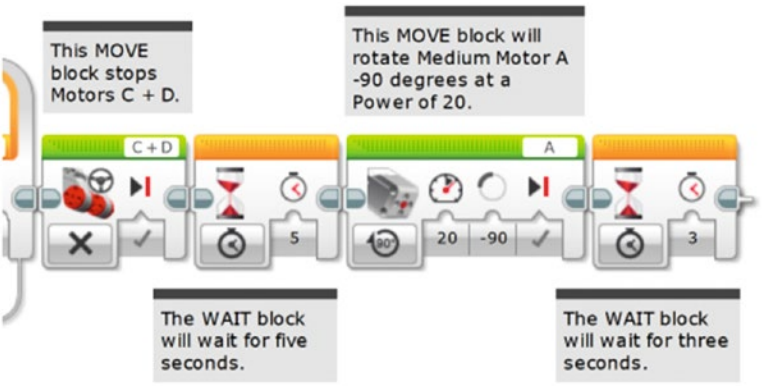


Figure 8-15. *After the object is dropped, we'll wait for a few seconds before starting back*

Then, add a reverse `MEDIUM MOTOR` block to close the carrier (see Figure 8-16).

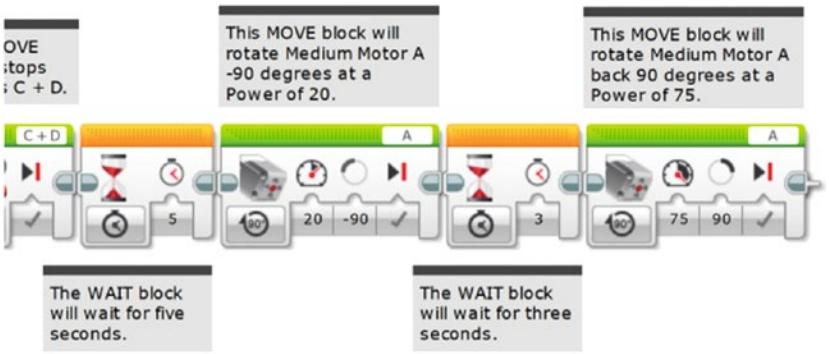


Figure 8-16. *Another MEDIUM MOTOR block closes the carrier*

Before you program the StringBot to return to its starting position, throw in one more `WAIT` block, just in case the StringBot is swinging on the string (see Figure 8-17).

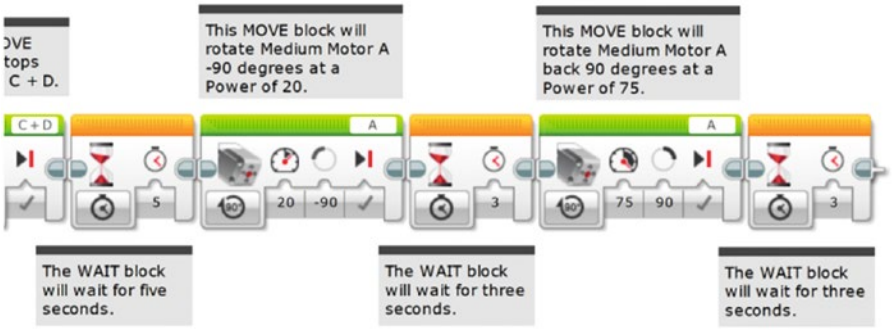


Figure 8-17. *This WAIT block lets the StringBot stop swinging before returning*

Back for More

Our final task for the StringBot is to return to its starting position and get another object to drop. But before finishing up the programming, I do want to talk about the importance of testing your bot before putting it into actual use.

During my first test of the StringBot, I made a *huge* mistake. Take a look at Figure 8-18 and you'll notice that the final bit of programming I added was a LOOP block to wait for the IR signal to start the trip back. Since the IR Sensor seemed to be a good idea so far, why not use the IR remote to start the reverse journey?

The LOOP in the middle of Figure 8-18 would receive the return signal, wait three seconds, and jump in to the final LOOP. This would start the StringBot back with a MOVE STEERING block inside, set at the speed of 80. The MOVE STEERING block was configured with a Duration of Unlimited, and the LOOP block was again configured to respond to the Infrared Sensor. And that middle loop was my mistake.

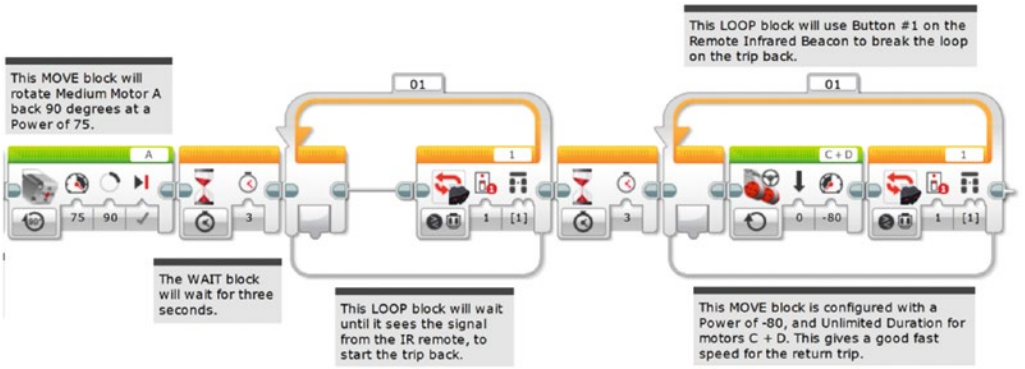


Figure 8-18. *The StringBot program had a big mistake that could have spelled disaster*

On the fourth or fifth practice return trip, the StringBot ended up closer to the wall than before. The Infrared Sensor sat there waiting for the signal to come back. But since the IR Sensor faces forward, it relies on the infrared signals bouncing off the far wall and landing on the sensor. Since it was too close to the wall, the infrared beam could not get past the bot's own body to reflect on to the sensor. The StringBot stopped. If this were the real situation, my StringBot would be stuck at the far end of the room without any way to retrieve it! I might have been able to rig up something to reach out and grab it, but maybe not.

Fortunately, I was testing the StringBot in my kitchen. I just walked over and grabbed the StringBot and brought it back to its starting position. If I had only performed the test once or twice, I would probably not have caught my mistake. So, let me repeat it again and again—test your bot, and test it often.

Once I caught this mistake, I realized this program is more complicated than it needs to be. I had "fallen in love" with the IR loop idea and used it even though I didn't need to. What's a better way? Simply have the bot return *on its own* without waiting for a command! And have the return motors run continuously until I have the robot in my hands. Turn it off manually. So I deleted those final IR LOOP commands. In their place, I used a simpler LOOP block. This one I set to loop forever (see Figure 8-19).

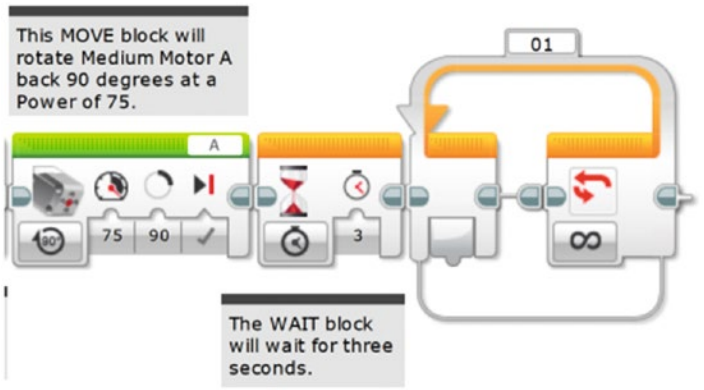


Figure 8-19. *Set the last LOOP block to loop forever*

Now place another `MOVE STEERING` block inside this `LOOP` block and configure it with a Duration set to Unlimited and a Power of 80 (see Figure 8-20).

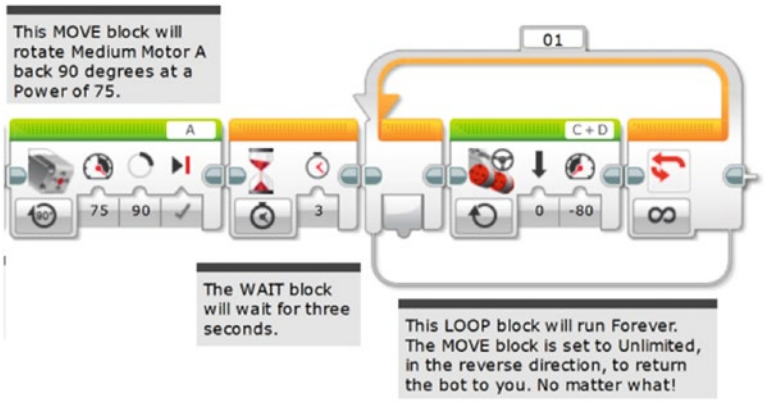


Figure 8-20. *This MEDIUM MOTOR block returns the StringBot to you*

There's no Infrared Sensor this time to stop the StringBot, so when it returns to the starting position, you'll have to grab it and stop the program manually by pressing the Cancel button on the Brick.

One final programming item I'd like to mention is the addition of the `STOP` block (see Figure 8-21). Although the `STOP` block isn't really needed in this program, I find it's a good habit to use the `STOP` block anyway at the end of a program. It's an old habit, but many programmers will tell you that it's still a good habit. In some programs you'll create, the `STOP` block is good to have because it will tell the Brick to exit the program and you won't have to press the Cancel button on the Brick.

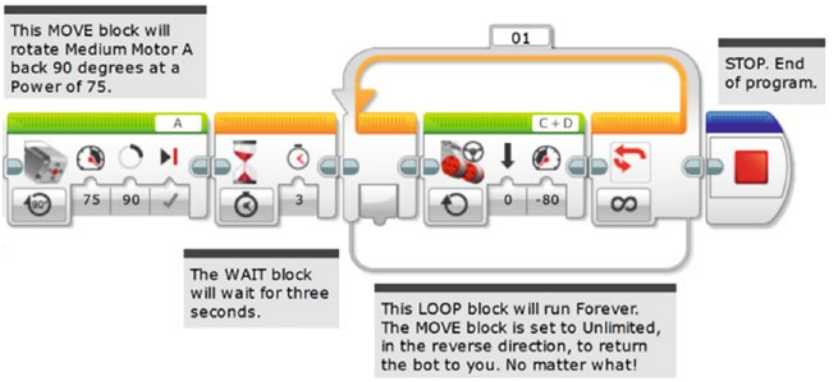


Figure 8-21. *Put a STOP block at the end of your programs*

Guess what? You're done with programming the StringBot! All that's left is to upload your program to the Brick and test it.

Testing: Filling the Vase

Okay, you have your StringBot built and you've uploaded your program to the Brick. It's time to set up the challenge so you can put your bot to the test.

When I tested my StringBot, I didn't need to spend a lot of time creating a test environment. To start, I simply tied one end of a piece of string to the doorknob of a closed door. I then cut the string to a length of around 20 feet. Take a look at Figure 8-22 and you'll see that I fed the string through the rear string guide (the two small wheels on the back side of the Brick). The string then goes under the rubber tire and back between the pair of gears acting as the front string guide. I tied the other end of the string to a heavy chair and then moved the chair away from the door to tighten the string. The final setup was placing a small jar a few feet from the door and directly beneath the string.

Figure 8-22. *The string passes through the rear string guide, around the rubber tire, and out the front guide*

This design will also work with light rope or clothesline. In addition, it will even work if you cannot untie your end of the string. If both ends of your string are already fastened, the IR Sensor motor assembly can be pulled apart with its motor and guides, shown in Figure 8-23. Remove the red 11-beam and pull the right motor assembly off the body and axles. You can then thread the string under the tire. Press the halves back together and replace the red 11-beam on the front, and you're good to go.

Figure 8-23. *The StringBot can be "opened" to place the bot on a string, rather than threading the string through. (Red 11-beam removed to allow the assembly to separate.)*

One thing I played with with was changing the angle of the string. In one test, the string was parallel to the floor, with the StringBot hanging about four feet above the ground. In another test, I tied the string to the bottom of the chair. The StringBot was forced to climb at a fairly steep angle, but it did work. You'll have to experiment with your own StringBot to determine the maximum angle it can climb. The wide rubber tire produces quite a bit of traction and the guides keep the string centered.

One key to testing your bot is to have some fun! Play around with your StringBot in different ways. Try tying the string higher than the doorknob so the StringBot has to drive at a downward angle. Experiment with a larger carrier. Maybe even try to attach a heavier object to your StringBot that can be released and dropped into the vase. How heavy an object can your StringBot carry?

I ran a series of tests using my StringBot and I want to share them with you. Some were surprising, some were disappointing, and a few of the tests were perfect:

- *First test—string parallel to floor*: My StringBot carried a single quarter. I was able to get the bot to drop the quarter directly in the jar on the first attempt. Because of this, I got a little cocky in the next few tests.
- *Second test—string parallel to floor*: This time, I put a few quarters and nickels in the carrier. About halfway across the string, one of the coins dropped out of the carrier. When the bot reached the vase and dropped the coins, only one nickel fell accurately into the jar. And it still seemed a little too fast to easily control, so I lowered the motor speed from 40 to 20. You can see the original speed of 40 as programmed in Figure 8-13. Try lowering it to 20.
- *Third test—string tied to bottom of chair*: I wanted my StringBot to climb the string at a large angle. About halfway up, the StringBot didn't have enough power to finish the climb and it stopped moving.
- *Fourth test—string tied to middle of chair*: I changed the angle of the string, reducing the angle. This time, the StringBot successfully carried two quarters and dropped them into the jar.
- *Fifth test—string tied higher than doorknob*: I changed the angle again, this time with the StringBot moving "down" the string. This was a *bad* idea. The StringBot slid at one point and the swinging motion caused most of the coins to fall out of the carrier.
- *Sixth test—string tied higher than doorknob*: I reduced the angle, making it less steep. The StringBot successfully made it to the vase, but when it dropped the coins, a slight swing in the bot caused the coins to miss the jar.

As you can see, the StringBot had some successes as well as failures. In the most basic test (string parallel to floor), the bot did what it was supposed to do. However, I ran that first test 12 times (16 minutes), and only one run failed. With these results, I imagine that I could fill that jar with about 50 to 60 trips in a little more than an hour.

When the vase was about half full, a loud cracking sound was heard underneath. I think it might be safe to cross the reception room floor . . .

CHAPTER 9

Scroll, Key, and Camera

Location: Southwest Guatemala

Weather Conditions: 94° Fahrenheit, Humidity 46%, Rain 0%

Day 3: Tomb Reception Area, 6:08 PM

It had taken Max and Grace more than an hour to get the string properly looped over the reception area room's far peg. They had taped four fiberglass tent rods end to end, making one long rod that looked like an odd fishing pole. They were careful not to let the long pole dip too far down—it could accidentally tip over the jar and then the expedition would be in real trouble.

Before tying the other end of the string to the peg nearest the entry door, Evan had fed the string through the StringBot's guidance rims and rubber wheel. When the string had finally been tied to the other peg, the StringBot was sitting on the string, about five feet above the floor. Over the past hour, the StringBot had dropped about 30 pebbles in the vase with only four pebbles missing the target.

Uncle Phillip, Max, and Grace had brought in some chairs and everyone was taking turns preparing the StringBot.

"Want me to take over, Evan?" asked Uncle Phillip.

Evan nodded. "Let me just load the carrier," he replied, placing a small pebble in the carrier. Uncle Phillip had asked Evan to use pebbles, explaining that if it were good enough for King Ixtua, it should be good enough for them to use. Evan had agreed.

"Okay, here goes," said Evan. He pressed the dark gray Select button on the front of the Brick and the StringBot began to move.

The team watched as the small bot began to move forward on the string. The spinning of the motors was the only sound in the quiet room. The bot moved quickly to the other side of the room, getting closer to the jar.

"Stop!" yelled Evan, as he pressed the upper-left button on his IR remote. He took a deep breath, hoping the bot worked just like in his earlier tests. The Infrared Sensor triggered and the StringBot quickly stopped, swinging slightly left and right on the twine. It again began to move toward the jar, but this time at a slower speed.

Evan watched as the bot's carrier approached the opposite wall. He had practiced dozens of times so he could determine when to stop the bot properly so the carrier could drop the pebble into the jar. The bot moved a little closer . . .

"Stop!" Evan yelled again, louder this time, even though the StringBot was actually responding to the infrared remote.

Uncle Phillip, Max, and Grace all held their breath as the StringBot stopped directly above the jar.

M. Bell and J. F. Kelly, *LEGO® MINDSTORMS® EV3*, DOI 10.1007/978-1-4842-2262-1_9

A few seconds passed, and then the carrier slowly swung down. The pebble slid from the carrier and dropped straight into the mouth of the jar.

CRACK!

A few loud popping sounds were heard coming from the opposite side of the room, followed by another loud CRACK underneath the stone floor.

"Yes!" yelled Uncle Phillip, jumping to his feet. He smiled at Evan. "I think it worked!"

"Way to go, Evan," said Grace. "Your little robot is amazing."

Evan's face turned red; he wasn't used to this kind of attention. He watched the StringBot's carrier arm close and then the motors began to spin again. The StringBot began its return on the string, ready for another pebble to be loaded.

When the StringBot reached Evan, Evan pressed the Cancel button on the upper-left front of the Brick. "What now?" Evan asked, looking at the team.

Max and Grace quickly stepped outside and then returned a few seconds later, holding opposite ends of a large metal chest. "We test the floor," Max said, as he and Grace placed the chest on the floor and then walked back outside the tomb.

"Evan, let's go outside for a minute," Uncle Phillip said.

Evan followed Uncle Phillip out of the tomb. As he stepped outside into the heat and bright sunlight, he saw Max holding a thick wooden rod about ten feet in length. He placed the wood on the floor and pushed it into the tomb until it connected with the metal chest. "Ready?" Max asked.

Uncle Phillip nodded.

Max pushed on the end of the wooden rod and the chest slid off the platform and onto the floor of the reception area. He pushed some more and the chest slid farther to the middle of the chamber.

"Well, it looks like the trap in the room has been disabled," said Uncle Phillip with a smile. "Are we ready to see what's next?"

Max, Grace, and Evan all nodded.

"Alright, let me cross to the other side. Everyone can follow, one at a time, when I'm across." Uncle Phillip walked slowly across the floor to the darkened corridor.

The King's Library

Max had moved one of the large tripod lamps across the room. The dark corridor was now well lit, and the team could see that the tunnel extended about 20 feet forward but with a downward slope. Halfway down the corridor, Evan saw a small square opening at the bottom-left side wall. The corridor was only four feet wide, so the team had followed Uncle Phillip down the hallway, one at a time. The corridor is sketched in Figure 9-1.

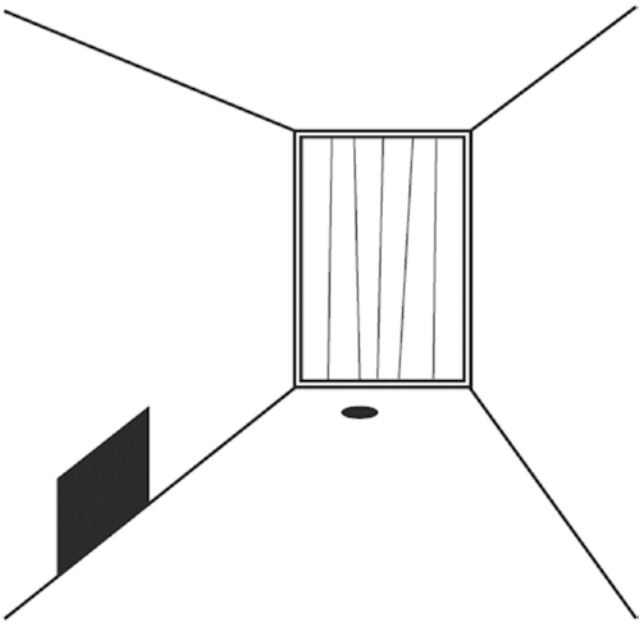


Figure 9-1. *Sketch of the corridor with the square opening and locked doorway*

"Just like the manuscript describes," said Uncle Phillip.

Grace, walking behind Uncle Phillip, pointed farther down the hallway. "And there's the hole in the floor for the pa'aachi," she said, pointing at a wooden doorway at the end of the corridor.

"Pa'aachi?" asked Evan, looking around Grace and Uncle Phillip to try to get a glimpse.

"It means key," said Uncle Phillip. "The manuscript indicates that a basket, called a hu'un, was placed in the king's library. Inside the basket is a carved bone that needs to be inserted into that hole before the door will open. And by the look of that hole, it's going to be a very large key."

"Where is the key?" asked Evan, already suspecting the answer.

Uncle Phillip stopped and pointed at the square opening in front of him. "In there."

Evan watched as his uncle dropped to the floor and turned on his flashlight. He pointed the flashlight into the opening and squinted. "It's hard to see, but the room's dimensions look about right."

"Can you see the basket?" asked Max.

Uncle Phillip tried to angle his head so he could see deeper into the opening. "Nope. If it's in there, it won't be visible to someone looking in. Once again, Tupaxu's design appears to require one of the king's trained monkeys."

"Another robot?" asked Evan.

"I don't know, Evan," said Uncle Phillip as he stood up. "This room's a little different and I don't know if your bot will work here."

While they were talking, Grace had moved down the hallway and was examining the door. She took some measurements of the door, but didn't push against it. "The door may or may not be locked, but we cannot open it without the pa'aachi. The manuscript says if we open the door without the key being inserted, it will trigger another trap," she said.

Uncle Phillip scratched his head. "Well, it's getting late and I'm hungry. Let's go get some food and talk this over. Evan, I'll show you a picture of the room so you'll understand why I'm worried about using a robot. Everyone back to the tent."

Evan nodded and turned, walking with his uncle out of the tomb.

Key Retrieval Challenge

While Evan finished his macaroni and cheese dinner, his uncle walked over with one of the manuscript enlargements and placed it on the table. Evan set his plate aside and stood up to take a look.

"King Ixtua had one of the largest collections of scrolls and carved tablets in existence at the time of his death. Tupaxu had the scrolls and tablets copied and placed in the tomb. This could be one of the biggest discoveries of Mayan history ever found," said Uncle Phillip.

Max pulled a chair closer to the table. "The manuscript doesn't indicate that the library has any traps. Maybe I could crawl through the opening and get the key," he said. "The opening is large enough for a person to fit through. And I might be able to open the throne room door from the inside. I believe there's a door in there that accesses the throne room directly."

Grace smiled at Max. "Or I could crawl through," she replied. "I am smaller than you."

"Rock, Paper, Scissors?" asked Max, holding up his hands, ready to compete for the chance to see the library first.

Uncle Phillip raised his hands. "No one is going into the library until we determine it is completely safe," he said. "And I'm certain that the door in the library is probably locked as well. There will be plenty of time to examine the library and its contents later, but we need to get that key. Now, Grace, what can you tell us about this room?"

Grace stepped closer to the table and looked down at the large document on the table. She then pointed to the left side of the page. "Against this wall are jars and tables that supposedly hold the scrolls and tablets. There's no information on how many of these artifacts were placed in the library, unfortunately. On the opposite side of the room are some large tables, probably used by visiting priests or the king's family to sit and read."

"The king's family would come in to read?" asked Evan. "Why?"

Uncle Phillip smiled. "The king's family was always welcome to visit his tomb. Priests would keep the tomb clean, bring fresh flowers for all the rooms, and look after other tasks. King Ixtua was considered a great king and was very good to his people. The library was the king's way to make his family more comfortable when they came to visit him."

"Weird," said Evan. "Sorry to interrupt, Grace."

"No problem," she replied. "Now, here's the important part. To get into the library, visitors would have to place the pa'aachi to open the door in the corridor. The next room will be the king's throne room, and I'm certain there's a door in there that will allow us to access the library. That key is found in the basket right here," she said, pointing to the drawing (see Figure 9-2).

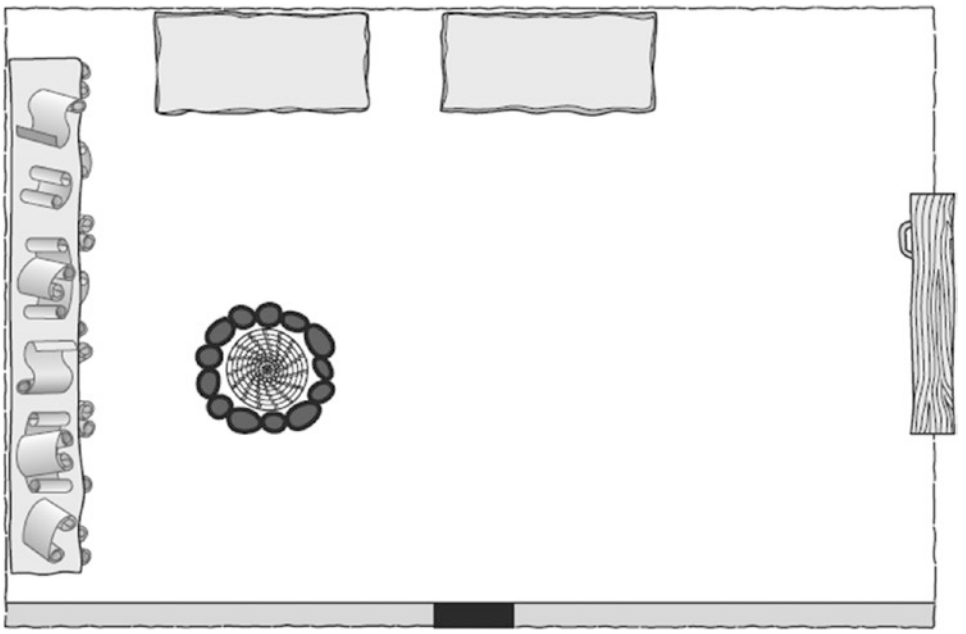


Figure 9-2. *View of King Ixtua's library and the basket holding the pa'aachi*

Evan looked at the drawing. The hand-drawn basket was surrounded by a black ring. "What is this black ring?" asked Evan.

"Probably obsidian," replied Max. "It's a type of glass that comes from lava that has cooled. The Mayan people used it to make weapons, but they also used it for other purposes because of its shiny black color. It's probably set into the stone floor for decoration."

Grace nodded. "King Ixtua probably had a monkey trained to find the black ring and then bring out the basket with the key inside. The real question is whether the rest of the floor is safe to walk across," she said.

Uncle Phillip finally sat down in his chair. "It's too dark inside the room to tell, and we can't just throw a flashlight in there to check. My guess is that the room isn't trapped, but I'm certain that if the basket is there with the key inside, it's probably too heavy for one of Evan's robots to bring back."

Evan nodded. "The motors are pretty powerful, but there's no guarantee it could grab the key or even push the basket back to us."

"And it's dark, too," said Max. "Won't your robot at least need some light?"

"Well, I have a Color Sensor that can create its own light with a small LED, but it's only useful for determining changes in color," replied Evan. "It's not bright enough to light up the room."

"Wait a minute," said Grace. "I have an idea."

Grace's Solution

"You said one of the sensors could detect a change in color," said Grace. "Can it be programmed to detect the change between the rock floor color and the black obsidian ring around the basket?"

Evan scratched his head for a moment, then nodded. "Maybe. It might be possible to have a bot search out the ring, but I still agree with Uncle Phillip that the basket and key are probably too heavy for my little bots to push. We could try it, but if the bot isn't strong enough, we won't be able to get it back," replied Evan.

Grace walked over to one of the boxes holding equipment and began to rummage.

"Do you have something in mind for the weight problem, Grace?" asked Uncle Phillip.

"Yes," she said. "If I can only find the . . . Here it is!" she yelled. She turned and held up a large ball of heavy twine.

"Twine?" asked Max.

Grace nodded. "It's just an idea, but what if we tie the twine to the bot, send it out and around the basket, and then back to us. We untie one end of the twine from the bot and cut the other end off the ball. If we pull on the two ends of twine, we might be able to pull the basket toward us." (See Figure 9-3.)

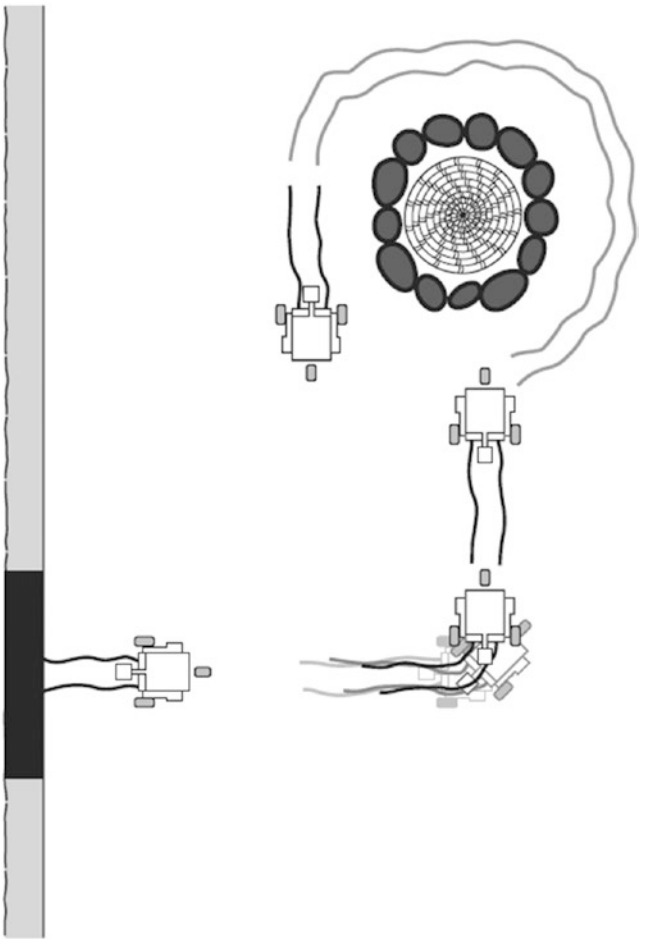


Figure 9-3. *The bot will wrap the twine around the basket and the team can pull it out*

Uncle Phillip smiled. “Grace, I think you’ve got the right idea. How about we make it stronger by sending out two separate lines of twine? We could have one line go around the bottom of the basket and maybe one around the middle,” he added. “What do you think, Evan?”

Evan had been listening to the discussion, trying to imagine what kind of bot he’d have to build to accomplish this task. An idea began to form and he smiled.

“I think I can build a little bot to do that,” Evan replied. “I’ll have to do some testing with the Color Sensor, though. But I’m fairly sure I can do this.”

“Excellent. It looks like that little robot kit of yours is really saving us time and money. Max, when we get back home, remind me to put in a request for the archaeology department to buy a couple of them, okay?”

“Sure. I think it’ll be a great investment,” Max replied, then frowned.

“What’s wrong, Max?” asked Uncle Phillip.

Max sighed. “Well, I just wish we could see the library as it is now—something to record this little bit of history.”

Evan raised his eyebrows and thought about the request. “Well, I could attach a smartphone to the bot and have it take video before it moves around the basket,” he said.

“You could do that?” asked Uncle Phillip.

“Sure. I’ll just need a way to attach the phone at the right angle,” Evan replied.

Grace smiled. “Let’s think about this video idea. With our smartphones, the video is not nearly as sharp as the photos they can take. Could your bot push the button to take an actual picture with the phone? Would that work?”

Evan nodded. “Give me some time to do some testing, and I’ll have it ready by morning.”

Uncle Phillip stood up. “Well, we’ve got about three hours before the camp has lights out. If you can’t finish tonight, take what time you need tomorrow and we’ll test Grace’s idea,” he said. “We need to take care of some paperwork, Evan. So if you need us, just yell.”

“Okay. I’ll get started right now.”

Max, Grace, and Uncle Phillip walked out of the tent, and Evan pulled out his robotics kit and Design Journal. As he was looking for a pen in his backpack, Uncle Phillip poked his head through the tent flaps.

“Evan, thanks for all your help. We’re really glad you came along,” Uncle Phillip said. “You’re doing great work.”

“You’re welcome, Uncle Phillip,” he said. “Thanks for inviting me on the trip.”

Uncle Phillip disappeared, leaving Evan to his work.

Story continues in Chapter 13 . . .

CHAPTER 10

SnapShotBot: Planning and Design

There are many elements to this challenge's bot. First, we have to figure out how to properly place the bot in the room. Then, the bot needs to take a picture of the library. After that, the bot needs to circle around the basket (still holding the twine) and return to the team so that they can pull the two ends of the twine to retrieve the basket and the key. Like I said, we have a lot to accomplish with this little bot. So let's get to work.

SnapShotBot Planning and Design

In Chapter 6, I didn't show you an initial picture of my final design for the StringBot. This was intentional, because I wanted you to start developing your own ideas for how the bot should look and function. I'm going to do the same in this chapter—I don't want to show you the final design of my SnapShotBot because I don't want to influence your planning and design ideas. That's the great thing about building bots with the LEGO MINDSTORMS EV3 kit: no two bots are likely to ever look the same because you're going to have your own ideas about how you want to solve challenges. So, no skipping ahead to Chapter 11 just yet. This is a challenging little bot to build and I want you to begin to focus on your own version of the SnapShotBot.

Once again, get a blank Design Journal page and pen and let's move forward.

Note There are three blank Design Journal pages left in the back of this book (if you used only one for Chapter 1 and one for Chapter 2). If you need more pages, feel free to make photocopies of the Design Journal page or visit the Apress web site to download the page in PDF format.

In the Robot Name box, go ahead and write **SnapShotBot** or, again, feel free to create your own name for the bot. Possible alternative names are LibRover, CamBot, or maybe StringBot2. After you've selected a name for your bot, it's time to move on to its description.

M. Bell and J. F. Kelly, *LEGO® MINDSTORMS® EV3*, DOI 10.1007/978-1-4842-2262-1_10

The Robot Description

The secret to success with a robot that has this many jobs to do is to try to keep the `Robot Description` as simple as possible. Visualize the tasks the robot needs to perform and write down your `Robot Description` once you can see it in your mind.

Ask yourself this question: "If I were walking behind this bot taking notes of its actions, what would I see?" Write the answers inside the `Robot Description` box. Just as with the earlier bots, simply picture a small box on the ground and visualize this box doing what you believe needs to be done to complete the challenge (the box is a placeholder for your SnapShotBot, because you don't know what it really looks like yet). Feel free to compare your `Robot Description` to the one I came up with, which is shown in Figure 10-1.

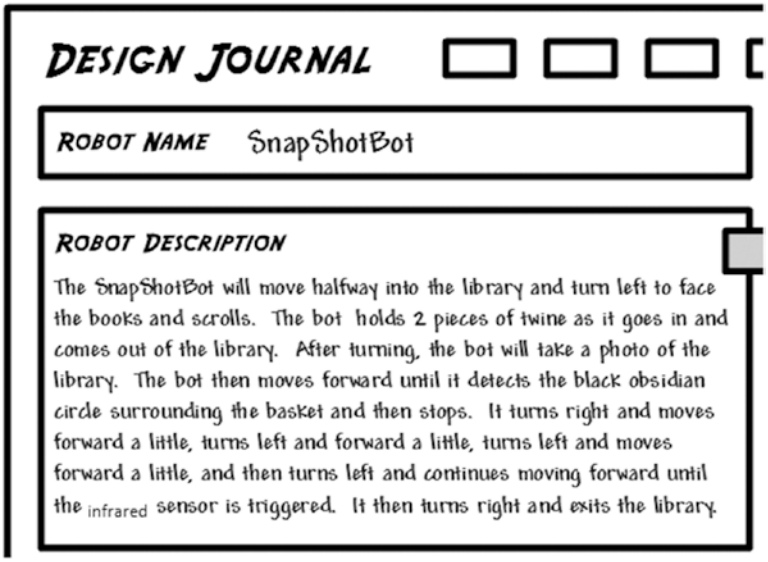


Figure 10-1. *The SnapShotBot Robot Description is the best place to start planning its design*

I'm guessing that your `Robot Description` doesn't match mine word for word. And that's okay. The main objectives you should have covered in your `Robot Description` include holding the twine, taking the picture, discovering the obsidian ring around the basket, navigating around the basket, and somehow returning to the team. Did you get all of it? If not, that's perfectly okay. Why? Because sooner or later during your actual building of your SnapShotBot you would have realized that your bot was missing a task to perform and you would simply go back and write in the missing task into the Robot Description box. Simple!

Here's another test you can do to make certain you've caught everything: pretend *you* are the bot! Look at Figure 10-2 for the layout of the library. Place a basket (or some other item) in the center of a room. Now walk through the tasks you need your bot to perform and compare them to your `Robot Description`. If you've missed something, just add it:

1. Move to the center of the room (holding two pieces of string) and turn left. (Easy . . .)
2. Take the picture of the library. (Piece of cake . . .)
3. Move toward the basket and stop. (Simple . . .)
4. Walk around the basket. (Check.)
5. Move back to your starting position. (Done.)

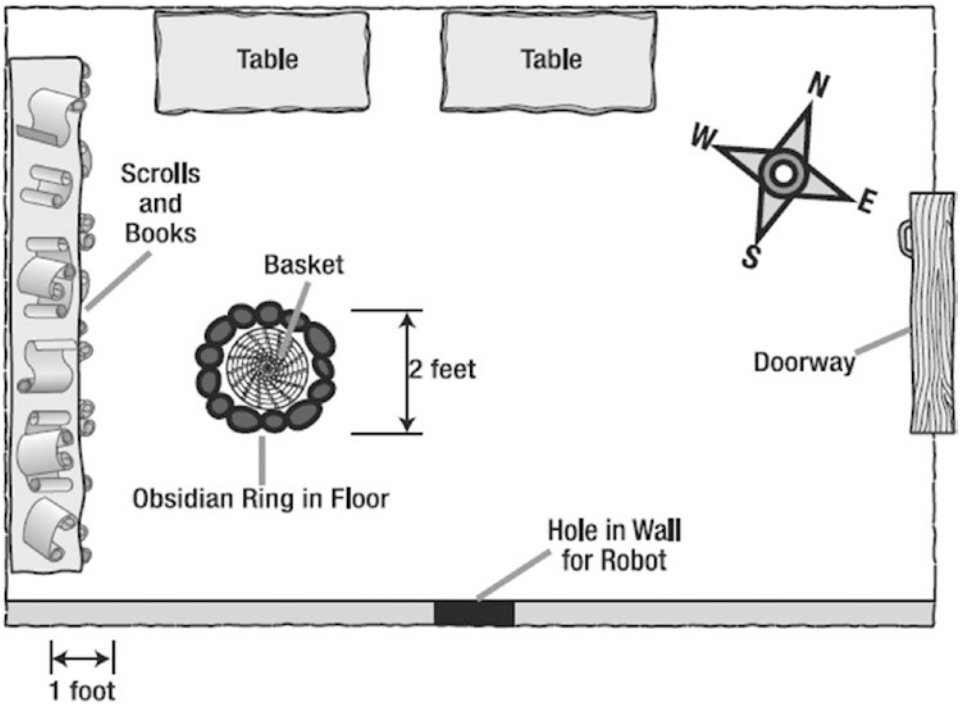


Figure 10-2. *The King's library*

You'll find that pretending to actually be the robot can reduce your design time because you'll have a better understanding of the jobs your little bot must soon perform.

Okay, now it's time for the `Task List`.

The Task List

Remember the purpose of the `Task List`? From your Robot Description, the `Task List` will help you break down the bot's individual functions so that you can properly build a solution for each of its tasks—move forward, take a picture, stop when sensor is triggered, etc. Review your Robot Description and begin writing down each separate task the SnapShotBot will perform. My `Task List` for the SnapShotBot is shown in Figure 10-3.

Figure 10-3. *The SnapShotBot Task List is built from the Robot Description*

The `Task List` is probably *the* most important part for this Design Journal page; there are so many things going on with the SnapShotBot, and a well-documented `Task List` will help you keep track of all the actions for this challenge. Because this is such an important `Task List`, I'm going to cover each item in the list in a little more detail.

Task 1

The first task is "Move forward to the center of room and stop." That seems pretty simple, doesn't it? If you look back to Figure 10-2, you'll notice that the room is square-shaped and that the distance from the bot's starting position to the rear wall is 12 feet. Well, the halfway point will be 6 feet, but how do we tell the SnapShotBot to stop at 6 feet?

When you program the motors, you can only specify the number of rotations, the number of degrees of rotation, or the time in seconds to spin. With the ExploroBot, we looked at how we could figure out how many degrees we need to complete a turn. This six-foot-stopping problem benefits from the same kind of thinking. There are various ways you can get you your bot to stop at six feet. I'm including three of them here, but there are many more methods:

- *Time in seconds*: Using a stopwatch, record how many seconds it takes your bot to move six feet. That's how many seconds you should run the motors to place the bot close to the halfway point.
- *Number of rotations*: Determine the circumference of the bot's wheels (the distance around the tire) in inches. Divide 72 inches (6 feet = 72 inches) by the wheel circumference and that will tell you how many rotations to program the bot's motors to spin to place the bot halfway across the room.
- *Number of degrees of rotation*: Gently push the robot across the floor six feet and, using the Brick's View option of the feedback panels in the configuration pane, record how many degrees of motor rotation the move takes. (This is just like what we did with the ExploroBot to measure our turns!)

For my SnapShotBot, I'm going to use the second method. I've measured a wheel and determined that the wheel's circumference is 5.5 inches. When I divide 72 inches by 5.5 inches, the number I get is 13.0. Will this be exact enough, since I only measured the wheel's circumference with a cloth tape measure? Don't worry; just getting it close to the middle of the room will be enough, so you don't have to be exact. So, when I program my bot to move to the center of the room, I'm going to set the motors for a duration of 13 rotations.

Task 2

Okay, now that the bot is in the center of the room, let's take a look at Task 2, "Turn Left to face library." You already know how to do this task from Chapter 4. You can refer to Chapter 4, after you build your bot, to find the instructions for manually turning your bot 90 degrees and getting the correct angle to turn one of the motors, but here's a quick summary:

1. Turn on the Intelligent Brick.
2. Use the Brick's right or left button to scroll to the View option and then press the dark gray Select button.
3. Use the right or left button to find Motor Degrees and then press the dark gray Select button.
4. Select a port to monitor (I'm choosing Port B for the wheel on the right side of the Brick) and press the dark gray Select button.
5. Place the SnapShotBot on a flat surface and press the dark gray Select button to reset the counter.
6. Twist and turn the SnapShotBot left 90 degrees so that *only* the right-side wheel turns.
7. Write down the result displayed on the LCD screen. (For the record, I got -678 degrees.)
8. Press the dark gray Select button to reset the counter.
9. Twist and turn the SnapShotBot right 90 degrees so that *only* the left-side wheel turns.
10. Write down the result from the LCD screen. (For this turn, I got -640 degrees.)

The first turn your bot will make is a left turn. But later your bot will also make a right turn, so be sure to record both results (left and right turn). Record these values somewhere on your Design Journal page. (Of course, you'll perform these steps *after* you've built the robot.)

Task 3

Task 3 is for your bot to take a picture, so let's give this a little thought. We've already allocated two large motors to control the right and left wheels. So, we'll use the third, medium motor to build a device that can press the button on a smartphone or a disposable camera. Can you picture the robot in the middle of the room, facing the library? The robot is low to the ground, so I'll probably want to angle the camera up a little bit so it can take a better picture—I'll have to remember to put that in my `Mindstorm` section. I'll also need to make some sort of holder for the camera. At this point, Task 3 is simple enough—I just need to plan on something to hold the camera and something that will allow the motor to press the button.

Task 4

Task 4 is "Move forward until black obsidian ring is detected and stop." From the `Robot Description` of the obsidian stone ring, we know that it's a different color than the stone used in the rest of the floor. This is a perfect test of our Color Sensor. The Color Sensor can detect changes in color, so we can add the Color Sensor to our bot and program it to detect the color of the regular floor. We can also program the Color Sensor to detect a change in the sensor reading as the bot begins to move forward toward the ring. When the sensor reading drops (black will give a lower reading than the brown stone color), we'll instruct the bot to halt.

This is a good place to stop and do a few experiments with the Color Sensor. After you build your SnapShotBot, you're going to set up a test environment. So, right now you need to figure out where you're going to perform your test. Possible locations are a living room floor (wood floor or carpet), a garage floor, or even a tile floor; as long as the floor isn't black or a very dark color, you shouldn't have any trouble. Once you determine your test floor, find a piece of black paper (or a piece of paper that is much darker than your floor color). Now, here's what I want you to do:

1. Turn on the Intelligent Brick. The screen might look like Figure 10-4 if there is no program loaded.

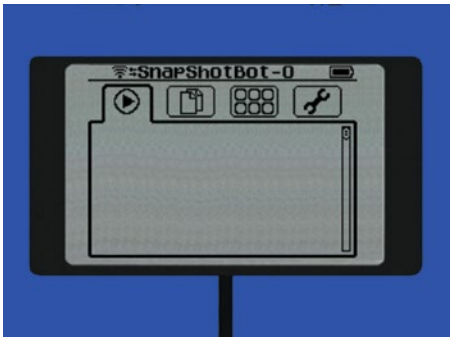


Figure 10-4. *The starting screen*

2. Use the Brick's right or left button to scroll to the View option (third tab). This will place you on the Port View selection, as shown in Figure 10.5.

Figure 10-5. *The View screen with the Port View selection highlighted*

3. Then press the dark gray Select button. If your Color Sensor is attached to Port 1, the screen will look like Figure 10-6.

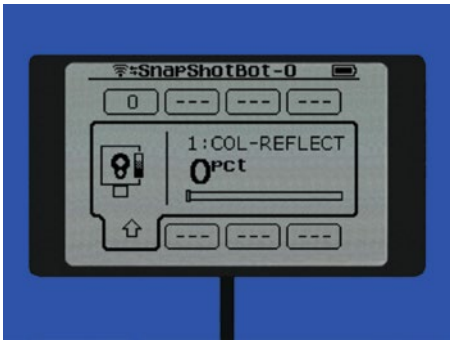


Figure 10-6. *The Color Sensor screen in the COL-REFLECT mode*

4. If needed, use the right or left button to select the port used by your Color Sensor and then press the dark gray Select button—the LED light on the Color Sensor should turn on and shine red light, if it is not already doing so. Your screen would look like Figure 10-7.

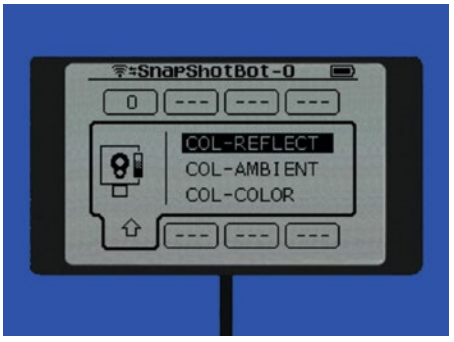


Figure 10-7. *Selecting the COL-REFLECT mode*

5. It should be on the COL-REFLECT option. If necessary, use the up or down button to select COL-REFLECT. Then press the dark gray Select button and your screen should resemble the one in Figure 10-6.
6. Point the Color Sensor near your test floor's surface. A good result can be achieved by holding it no more than two to three centimeters above the surface.
7. Write down the result displayed on the LCD screen for the "normal surface."
8. Place the black (or dark) paper under the Color Sensor at the same distance and notice the value changes.
9. Write down the new result displayed on the LCD screen for the "obsidian surface."

Record the results of your test somewhere on your Design Journal page. Write **Normal Floor** and the number that was displayed for it, and then write **Dark Paper** and the number for it, too. You'll need these numbers when you begin to program your bot.

Okay, so now your bot is near the basket. The next four tasks are simply steps to get the bot around the basket and heading back in the direction from which it came.

Task 5

Task 5 starts the journey around the basket by having the bot turn right, move forward a little bit, and then stop. But what do we mean by "a little bit"?

Well, we know that the black obsidian ring surrounding the basket is about two feet wide and two feet long (we see this in Figure 10-2). And earlier in the chapter, you found out that the circumference of the wheels is about 5.5 inches. We also know that the bot should be near the center of the ring, so let's figure out how many rotations we need it to make to get safely around the basket.

As shown in Figure 10-4, after the bot turns right, it needs to move only about one foot forward (to Point A). One foot is 12 inches, so we divide that by the circumference of our wheels (that is, 12/5.5 inches = 2.2 inches). That tells us we should program the bot's motors to spin 2.2 rotations, which should place it close to Point A. On the back of your Design Journal page, copy Figure 10-8 and write down the rotation values you calculate—between the bot and Point A, write **Rotations:** and the number that you calculated.

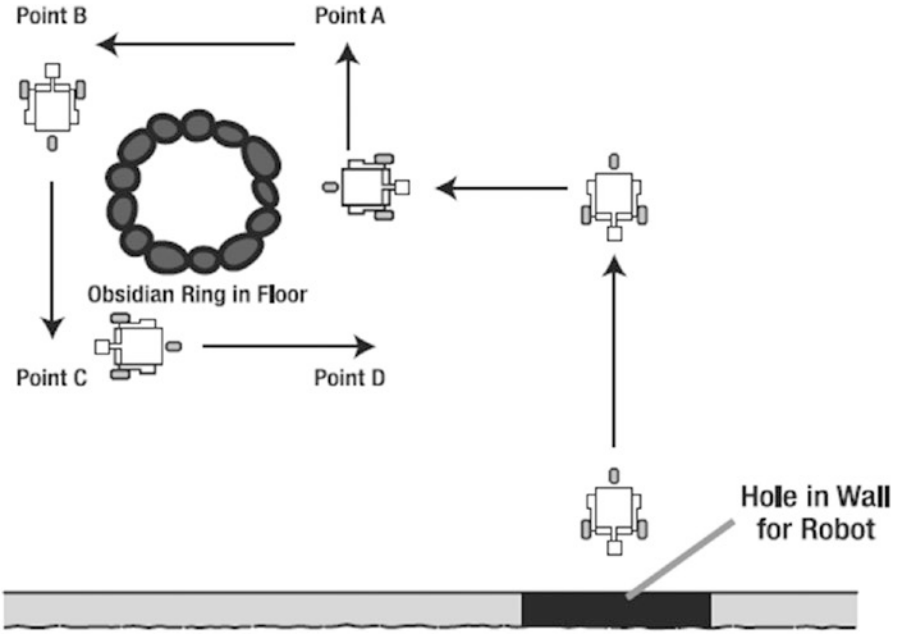


Figure 10-8. *The SnapShotBot needs to go around the basket*

Task 6

Task 6 says "Turn left, move forward short distance and stop." Again, what is a "short distance?" Well, Figure 10-2 tells us the ring is two feet in diameter. So, in Figure 10-8 we can see that the distance from Point A to Point B would be about 4 feet, or 48 inches. Once again, divide 48 inches by the wheel's 5.5-inch circumference and we get about 8.8 rotations. On your map, write down **Rotations:** and the number calculated for the motors to spin between Point A and Point B.

Task 7

Task 7 is identical to Task 6. This will get you from Point B to Point C. Write **Rotations:** on your map and then write the number calculated for the motors to spin between point B and point C (that is, 8.8). Just to be safe, if you like you can add in another few rotations to make certain the bot gets completely around the basket. Now you have the proper number of rotations to get your bot to point A, then from point A to point B, and finally from point B to point C (or slightly beyond point C).

Task 8

Starting at point C, your bot simply needs to make a left turn and move back to the center of the room. If everything has been done properly, your SnapShotBot will have encircled the basket with the two pieces of twine that will be used to pull the basket.

Task 9

Okay, so let's pretend that the SnapShotBot is moving back to the center of the room. You, as an archeology team member, will be looking through the small opening in the wall. As soon as you see the SnapShotBot, you need a way to tell it to stop, turn right, and come back to you. Won't the Remote Infrared Beacon work perfectly here? Task 9 is "Stop when Infrared Sensor is triggered," and this task will be easy to handle. You'll need to remember to include the Infrared Sensor and its remote in the `Mindstorm` section later in the chapter.

Task 10

Task 10 is our final task: "Turn right and move forward to library exit." Once the bot has stopped (triggered by the remote's signal to the Infrared Sensor), it will turn 90 degrees to the right. You recorded the number of degrees needed to make a right turn earlier, remember? After the bot has turned, it should be facing the opening in the wall. Start the bot moving forward and just wait for it to come back to you—that's it!

What About the Twine?

Now, you may have noticed there's no mention of the twine in the `Task List`. I intentionally left it out because it's not really a bot task (well, other than to pull the string). You'll need to include some sort of beam or component where you can tie two pieces of twine to the SnapShotBot. When the bot returns to you after completing Task 10, you'll use the two ends of twine that you held on to (while the bot was moving in the library) and the other two ends of twine tied to the bot and you'll PULL! As you pull on the twine, the twine will catch on the basket and allow you to pull the basket toward you!

This was a *long* `Task List`, but this bot has a lot of things to do. We're not done just yet, though. Hopefully, as you were thinking about the tasks involved, you came up with some limitations or constraints that you will need to keep in mind as you begin building.

Limitations and Constraints

Your bot has one primary job—to go around the basket while holding the twine and return to the team. There are no obstacles in the way (other than the basket itself), and all you have to worry about are the walls and tables. If you can just navigate your bot around the basket, you're home free. The few limitations I came up with are listed in Figure 10-9.

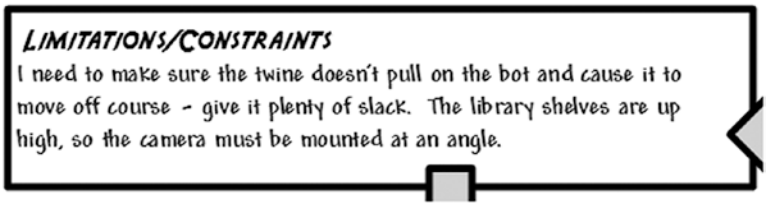


Figure 10-9. *The SnapShotBot does have a few limitations to think about*

As for taking a photograph, however, there is a small limitation to consider. Your bot will most likely be fairly low to the ground. (A tall bot is more likely to tip over or possibly be wobbly, but I don't want to put any limits on what your SnapShotBot will look like. If you choose to build a taller bot, you may find that you have an additional constraint or two to work around, but that's all part of the fun, right?)

For my little bot, if I simply set the camera on top of a flat spot, the final picture will be of mostly the lower part of the library wall. I want to get as large a picture as possible, showing as much of the books and scrolls as I can manage. Because of this limitation, I'm fairly certain I'll have to angle the camera back a little bit so that it's pointing slightly upward. Just imagine you are the bot, down low on the floor. Look up to the middle of the wall, and that's the angle that you'll want to use to mount the camera.

One minor constraint that I also added involves the twine that will be tied to the bot. Imagine a dog on a leash. If you hold the leash tight, the dog will not be able to continue moving away from you. The same goes with your robot. You'll need to keep feeding out a lot of additional twine so the bot never gets stopped. There's also a risk of the bot moving a little off course if the twine ever gets tight. So, my best recommendation to you is to provide a *lot* of slack to your bot by paying attention to the tightness of the twine and never letting it get too tight.

Did you come up with any additional constraints? If so, write them down in your `Limitations/Constraints` section and keep them in mind during the building and programming of your bot.

Mindstorm

Most likely, you've already got an image in your mind of what your SnapShotBot will look like. Feel free to look at Chapter 11 if you'd like to see my final version of the SnapShotBot. My guess is that my version doesn't look like what you're imagining. Am I right? Perfect!

For this bot, you're welcome to use my steps to build the SnapShotBot, but maybe you're feeling good about your design skills at this point and want to build your own version first. Give it a try! If you get stuck, take some of my ideas. The goal is for you to try to build your own version of this bot. And the `Mindstorm` section of your Design Journal page is the ideal place to begin putting down your ideas before you begin to build.

In Figure 10-10, you can view my ideas for my bot in the `Mindstorm` section of my Design Journal page.

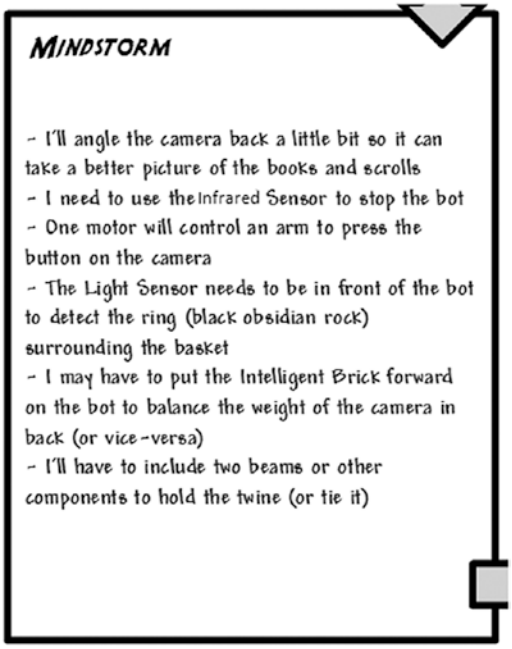


Figure 10-10. *The Mindstorm section for my SnapShotBot*

The first `Mindstorm` item was mentioned in the previous section: "I'll angle the camera back a little bit so it can take a better picture of the books and scrolls." Because the books and scrolls are stacked along the left wall (see Figure 10-2) and the bot is so low to the ground, I'll angle the camera back and it will be able to take a picture that doesn't include most of the floor. As stated earlier, if the camera is just facing directly forward, the final picture will contain a large portion of the floor and lower wall. This might be okay for you, so feel free to place your camera any way you like.

Again, I'm not going to cover every item on my `Mindstorm` list, but I would like to cover two more that I feel are important. First, "The Color Sensor needs to be in front of the bot to detect the ring (black obsidian rock) surrounding the basket." This is easy to understand. If you place the Color Sensor on the back of your bot, there is a chance that your bot might hit the basket before stopping at the black ring. Placing the Color Sensor at the front of the bot allows the bot to detect the black ring early and stop before most of the robot body crosses inside the ring (and possibly collides with the basket).

Second, "I may have to put the Intelligent Brick forward on the bot to balance the weight of the camera in back (or vice versa)." This is based on my experiences with building small bots with the MINDSTORMS EV3 kit. Too often, I've found that some of my bots tend to have too much weight in the front or back or on a side. Because of this, some bots are unbalanced, increasing the risk of tipping over. As I build my bot, I'll try to remember to place the Brick and the camera (with its motor) on opposite sides of my bot. I might choose to put the camera in front and the Brick in back, or vice versa. I haven't decided yet on this point and will probably wait until I begin to build.

Now, take your `Mindstorm` ideas, your `Task List`, and the `Robot Description` and pull them all together by creating some sketches—these will help you when you start to build. When you have an idea in your mind, putting it down on paper will help you start snapping pieces together in a logical manner to achieve that mental image. So let's finish up with the `Sketches` section of your Design Journal page.

Sketches

Alright, I'm warning you again to keep the snickering to a minimum! My sketches aren't the best, but I think you'll get the idea. Take a look at Figure 10-11 to see my rough sketches of the (soon-to-be-built) SnapShotBot.

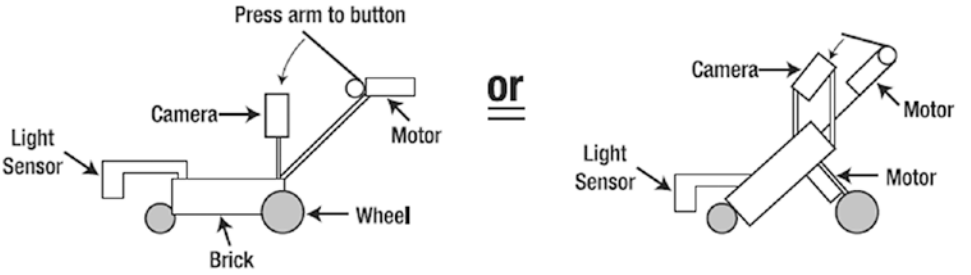


Figure 10-11. *I'm using basic shapes to represent parts of my SnapShotBot*

Notice that I'm not drawing with extreme detail. Yes, this is partially because I'm not the best artist. But more importantly, it's because I don't need to draw every beam and connector and sensor to develop a working shape and concept of where everything will go. Wheels are circles, the Brick is a rectangle, sensors are smaller rectangles, and motors . . . well, the motors just look weird. Sorry.

After you complete your sketches, there's no point in waiting, so start building! Take everything you've written and drawn on your Design Journal page and build *your* version of the SnapShotBot . . . or whatever name you have given it.

In Chapter 11, you'll find the step-by-step instructions for building my version of the SnapShotBot. Steal ideas from it—or just ignore it all—and make your version bigger (or smaller), better, faster (or slower), and as unique as you can imagine.

CHAPTER 11

SnapShotBot: Build It

Go ahead and take a look at Evan's version of the SnapShotBot in Figure 11-1. It's a weird-looking little bot, but, then again, I *like* strange-looking robots . . . and this one definitely qualifies. As you can see, the Color Sensor is located at the front. I've placed the camera toward the center of the bot at an upward angle. The weight of the Brick is sufficient to keep the bot from tipping over, thankfully.

You'll also notice during the building process that I use the longer beams (15 holes) and large angle beams to provide reinforcement and strength to the camera frame and motor attachments. Since the camera frame is angled back and needs to be sturdy for the shutter release to work, I rely on these types of support beams a lot—it's a personal building style that works for me. I also use the double gray connectors whenever possible. They are simple, strong, and seem to fit a lot of situations.

Engineering: "Good Enough" with a Peg-Leg?

Let's think about this "good enough" idea. I've used a "peg-leg"—that is, a long axle from the LEGO MINDSTORMS EV3 kit as the rear-wheel. You'll see this trick used from time to time with EV3 bots because the peg-leg drags easily and allows bots to turn more smoothly than does an ordinary rubber wheel. I could have used something similar to the pivot wheel found on the ExploroBot, but I wanted to try out this option. Of course, using a peg-leg is "good enough" only if the floor is smooth. On a carpet, it would not work at all!

As a designer and engineer, you'll often face the question "Do I just throw this thing together, or do I make it elegant?" In this SnapShotBot, the peg-leg rear-wheel is hardly elegant. But it is simple, reliable, and it works—on hard, smooth floors. You could take the time to make an elegant rear-wheel as found on your ExploroBot. Later in this SnapShotBot, there **is** an elegant solution to the hard problem of "How do I make a shutter trigger using the relatively weak Medium Motor?" It can take a fair amount of force to operate shutter buttons. If you look at Figure 11-15, you'll see how many small parts it takes to make the worm drive you see in Figure 11-1. But it is undeniably elegant, very reliable, definitely strong enough to do its job, and with clocklike complexity.

Judging when to pursue an elegant solution over a "good enough" solution is true engineering thinking. This SnapShotBot offers excellent examples of both kinds of solutions.

M. Bell and J. F. Kelly, *LEGO® MINDSTORMS® EV3*, DOI 10.1007/978-1-4842-2262-1_11

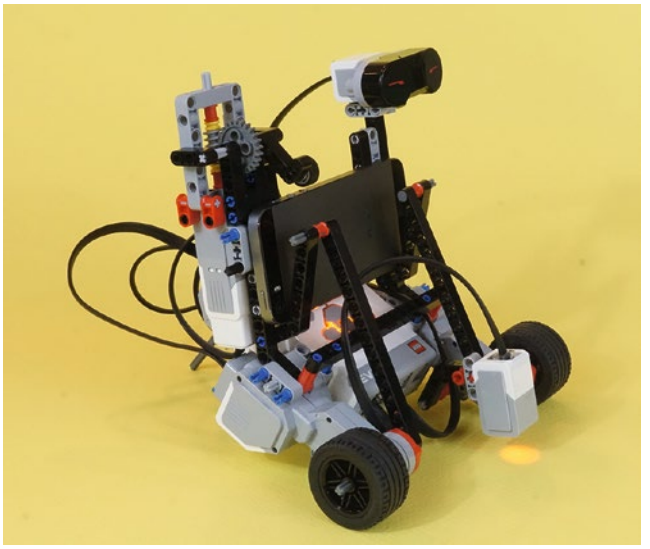


Figure 11-1. *One possible version of a SnapShotBot with a smartphone to take the photo of the king's library. The Color Sensor is lit up, ready to look for that obsidian ring in the floor!*

Jump In

I had no trouble getting my basic design for the SnapShotBot. Using my sketches from Chapter 10, I thought that my design would have two rear-wheels to drive this heavier bot. I placed the camera over the center of the bot because I wanted it to be as stable as possible (even with the angle) to take a picture. But as I worked with the parts, the front motor design seemed to work better and I ended up with the peg-leg rear-wheel.

■ **Note** I used a smartphone to take the pictures, but I could have found a disposable camera for this bot. (See Figures 11-38 and 11-39 for examples.) The frame is widened to keep the disposable camera from falling out. You may have to play around with the camera frame a bit to get it to fit the smartphone or camera you decide to use. Keep your smartphone or camera nearby as you build so you don't have to guess about the dimensions of the frame. A few extra pieces here and there will allow the bot to hold your device firmly.

If I later decide to use a disposable or digital camera that advances automatically, I can reprogram the bot to take numerous photos. If I were to send my bot into an unknown area with the risk that it might not come back, I'd rather use a disposable camera so I wouldn't lose my expensive smartphone. But for now, all I need my bot to do is go in and take one photo.

Follow along with my instructions to build my version *or* build your own version and feel free to compare your design to mine. You may find an idea or two that you can use with your own bot. This is certainly true of the shutter mechanism, using a worm gear to increase the strength of the medium motor.

■ **Note** I really would like to see your version of the SnapShotBot, so take a picture and e-mail it to me. My web address is in the Introduction.

Okay, let's go build the SnapShotBot!

Step by Step

I've divided the SnapShotBot building instructions into three sections. The first section contains the basic body—the Intelligent Brick and two wheel motors, with reinforcement beams. The second section adds the Color Sensor, the IR sensor, and the Camera Trigger assembly. The third section completes the SnapShotBot by adding the adjustable camera frame. Build these three sections, wire them up, and your SnapShotBot will be complete and ready for programming.

As with previous chapters, I add comments to sections where a little help might be needed. And remember, if you see an image with text and arrows, pay attention because the text and arrows are probably pointing to a placement of a part (or parts) that otherwise might not be immediately noticeable in the picture.

And, finally, I'll add new parts to an image to let you know what you'll need in upcoming steps.

First Section: Basic Body with Hardpoints

Figures 11-2 through 11-8 show the steps and components used to build the basic body and wheels, as well as a few pieces needed before you begin the second section to add the Color Sensor, IR sensor, and shutter assembly.

Start with the pieces you see in Figure 11-2. The figure shows the pieces for one set. But you'll need two sets of these pieces, since the two motors are mirror images.

Figure 11-2. *Starting pieces for one of the SnapShotBot's motors. The two motor assemblies are mirror images, starting with identical parts. So you'll need twice the number of parts shown in this figure to build both motors.*

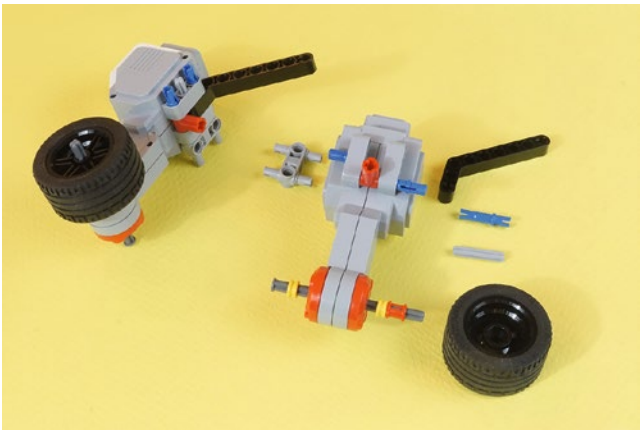


Figure 11-3. *Using the duplicate sets of parts from Figure 11-2, assemble the right and left sides. Notice that the long axle is a nailhead. The heads will be facing the center.*

Figure 11-4. *Both sides assembled, ready to be fastened to the Brick*

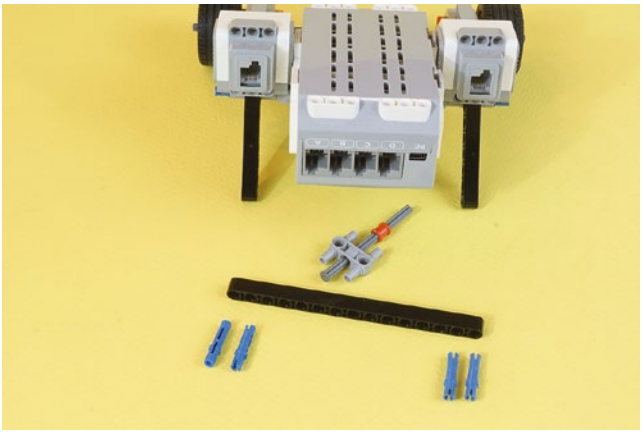


Figure 11-5. *The parts for the rear peg-leg and the 15-hole beam to reinforce the two motors. Another long nailhead axle, shown partially assembled in the peg-leg connector.*

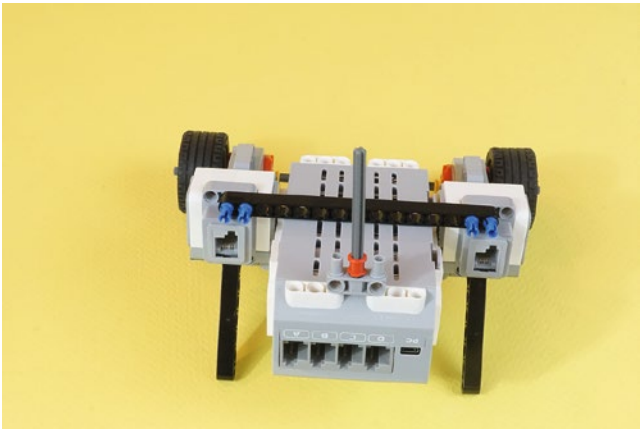


Figure 11-6. *The 15-hole beam in place. The long blue connectors are used for easier disassembly. They will remain with the beam if it is removed for new batteries, etc.*

Engineering: The Concept of Stiffness

When building robots like this, it is pretty common to temporarily remove parts. For example, you might need to change the batteries. Here, the long blue connectors stay attached to the 15-hole beam and make it easy to put the beam back in place. With the smaller black connectors, they would not stay captive to the beam.

A robot like this can be built without the long beams for reinforcement. You could say they're "optional." But if you don't put them in, the bot ends up being wobbly. The motors sometimes fall off at the least convenient times! These reinforcement beams prevent that. Try wiggling the robot in your hands with and without the beams—you'll see what I mean.

This puts us into the concept of *stiffness*. It's a frequent requirement in the engineering world that parts have a minimum level of rigidity, or stiffness. These reinforcing beams give that by their strength and the fact they oppose the forces that can cause the robot to fall apart. That complicated load-bearing axle in the StringBot was engineered to give us adequate stiffness.

Figures 11-7 and 11-8 show the red angle connectors that will receive the blue half-axles attached to the long beam. These add further stiffness since they are snugly fitted between the motor hardpoints.

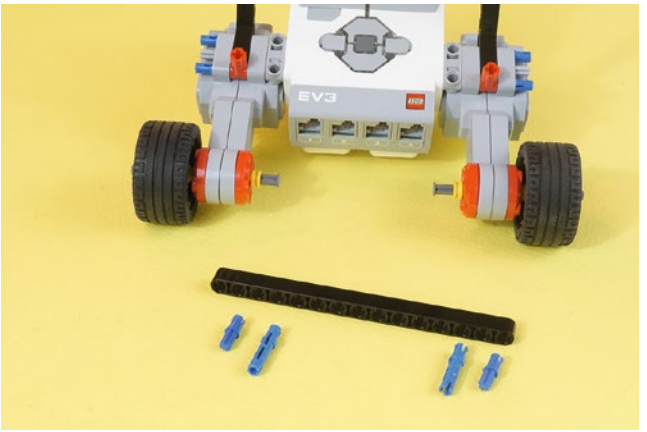


Figure 11-7. *Another 15-hole beam further reinforces the motors on the top side*

Figure 11-8. *The basic body, with hardpoints*

Engineering: Hardpoints

A *hardpoint* is a location on a frame, designed to carry a load. The angle brackets coming out of the motors are very strong, held in with the long blue connectors as well as short axles. In the LEGO kits, the angle brackets have an axle hole at each end. Axles hold on tighter than regular connectors, so I use them whenever I can. If you need to build a bot with low, solidly-mounted motors, this is an excellent starting point. The angle brackets—that is, the hardpoints—are an excellent place to mount other hardware, even if you haven't designed it yet. The brackets are there for when you need them.

You now have a fully functional rolling bot body that could be customized with other parts and sensors to perform other duties. If the peg-leg does not work on your floor, you could add a rear-wheel assembly like the one used on the ExploroBot.

Now it's time for the Color and IR Sensors, and the Camera Trigger Worm Gear.

Second Section: Color Sensor, IR Sensor, and the Camera Trigger

Figures 11-9 through 11-33 show you how to complete everything except the camera frame. Let's move forward.

Figure 11-9. *Here are the parts needed to mount the Color Sensor*

Figure 11-10. *The Color Sensor assembly*

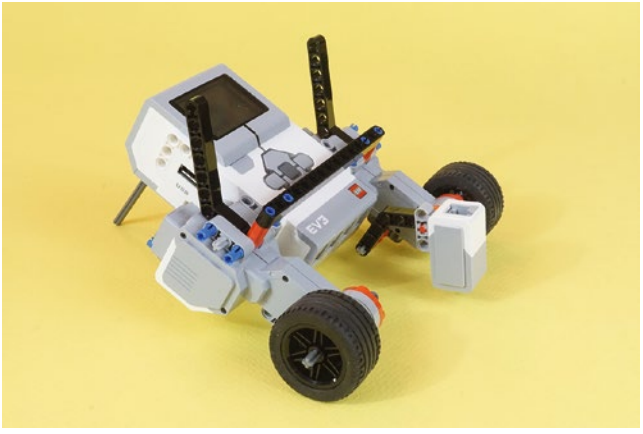


Figure 11-11. *The Color Sensor assembly mounts to the center axle hole on the right motor. It is quite close to the floor, without touching.*

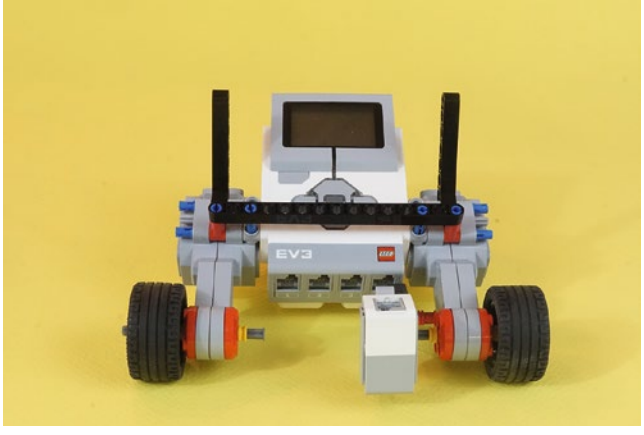


Figure 11-12. *Another view of the Color Sensor assembly, mounted in place*

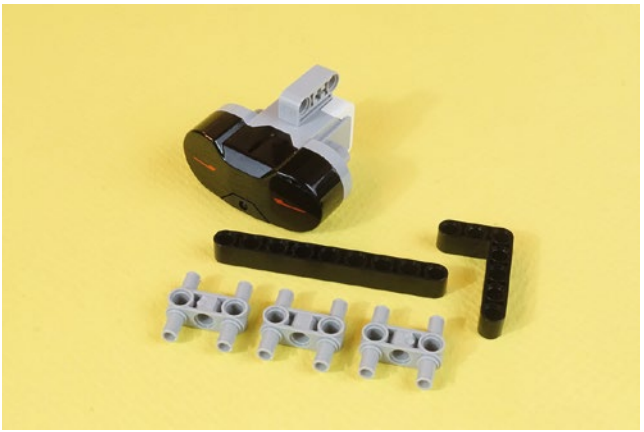


Figure 11-13. *The parts needed for the IR Sensor assembly*

Figure 11-14. *The IR Sensor assembly, ready to be mounted. Set it aside for now or mount it as shown in Figure 11-33.*

Figure 11-15. *All of the parts for the medium motor Camera Trigger assembly. These parts will take you up through Figure 11-28.*

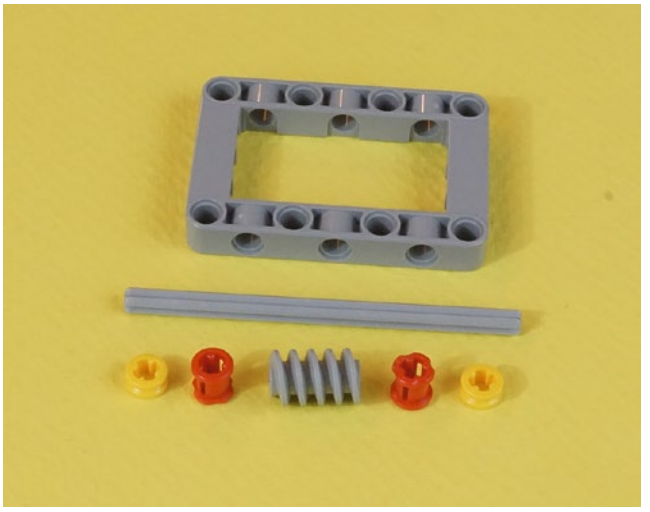


Figure 11-16. *These parts, from Figure 11-15, let you build the Camera Trigger Worm Gear sub-assembly*

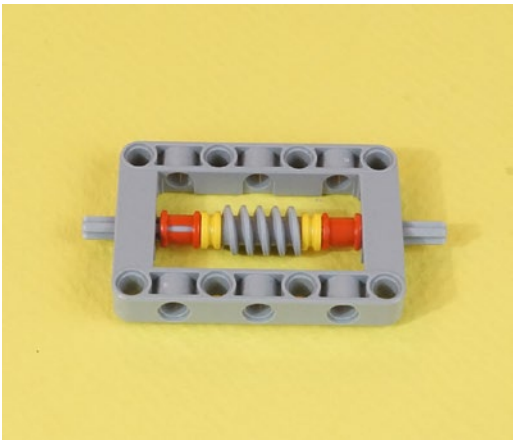


Figure 11-17. *The Camera Trigger Worm Gear sub-assembly. Each end of that axle will be useful later.*

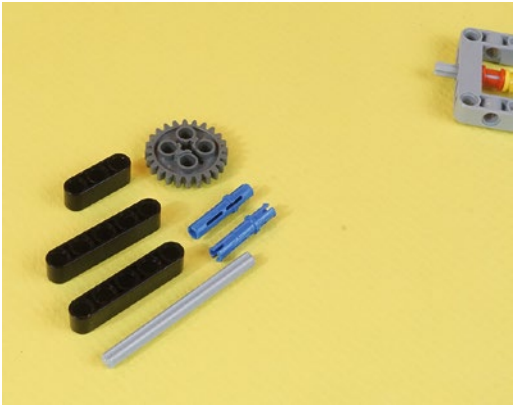


Figure 11-18. *More parts from Figure 11-15. They will become the Gear Yoke.*

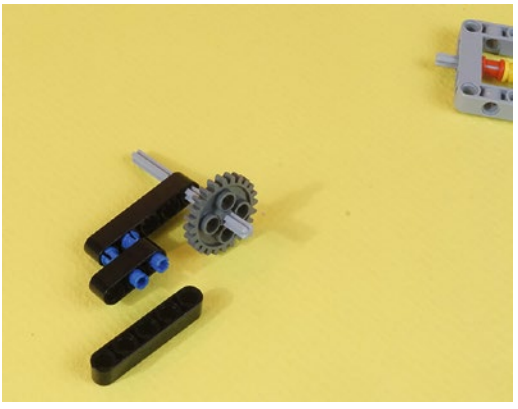


Figure 11-19. *The Gear Yoke comes together. The long blue connectors are our friends here.*

Figure 11-20. *Still more parts from Figure 11-15. Now for the actual trigger mechanism and 3-hole beam side fasteners that will secure it to the Worm Gear sub-assembly.*

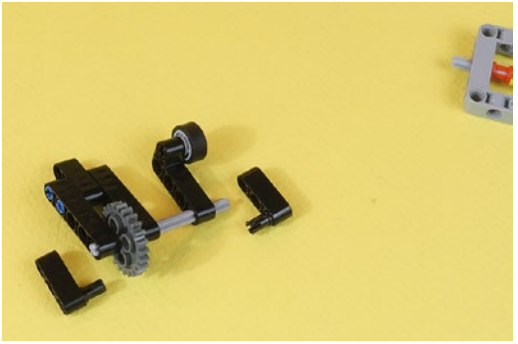


Figure 11-21. *The trigger mechanism uses a little rubber wheel to contact the camera shutter button. This wheel gives a larger contact area and more effective shutter operation.*

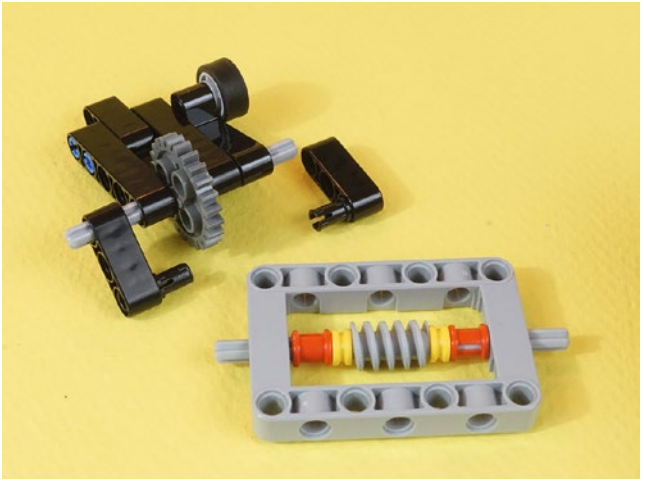


Figure 11-22. *Side fasteners preparing to attach the Gear Yoke to the Worm Gear sub-assembly*

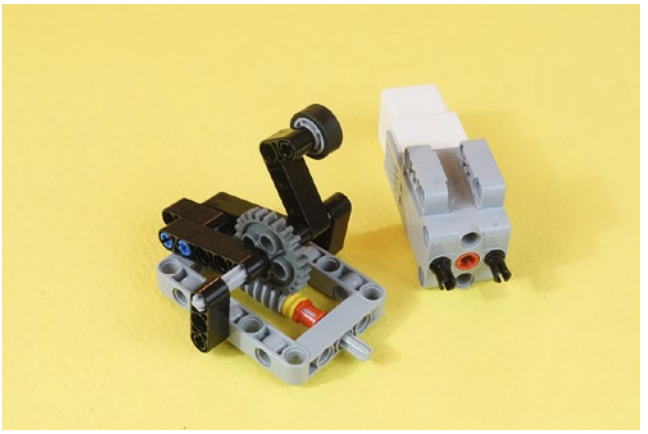


Figure 11-23. *Put two short connectors into the front medium motor holes as shown. The side fasteners will be wobbly at this stage. Flipping the Gear Yoke over gets you to the next picture.*

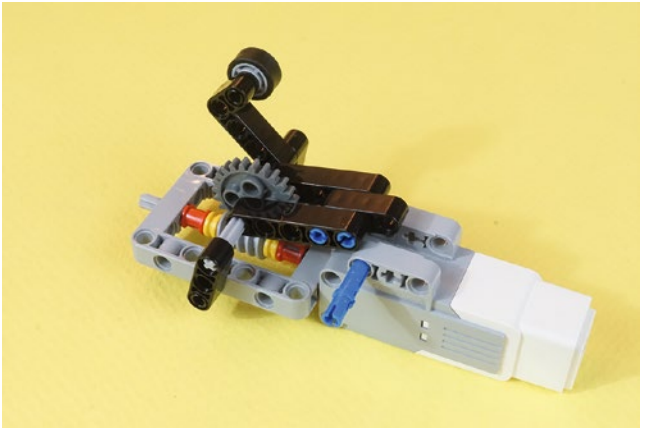


Figure 11-24. *Plug the Worm Gear sub-assembly into the end of the medium motor, matching up the short connectors and the axle as it goes into the end of the motor. Press the Gear Yoke into the mounting point of the motor and push the long blue connector in to secure it.*

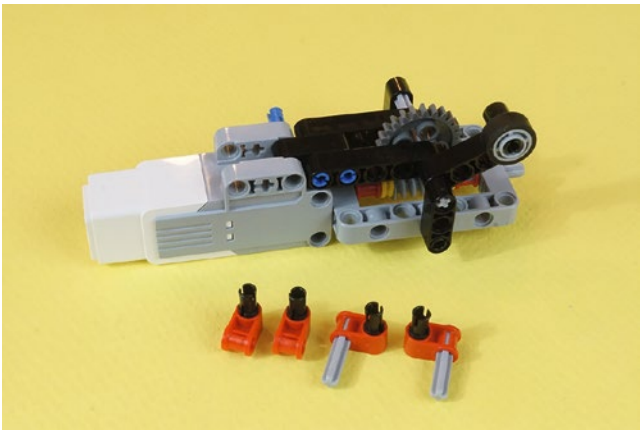


Figure 11-25. *A view from the other side. Assemble the four corner reinforcements, which are parts from Figure 11-15.*

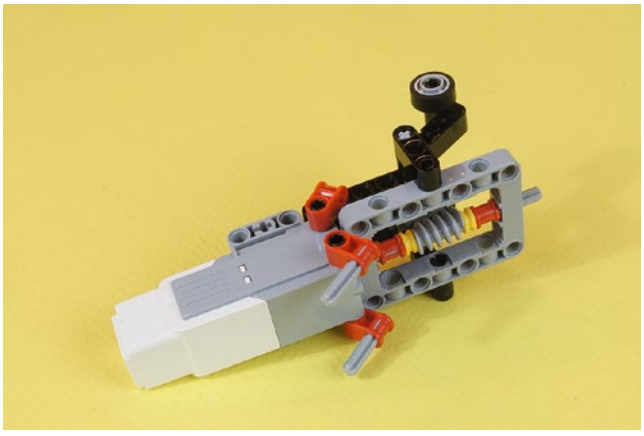


Figure 11-26. *The corner reinforcements on the medium motor. Twist the reinforcements to make them flat against the Worm Gear sub-assembly. Then press the short axles through to the other side.*

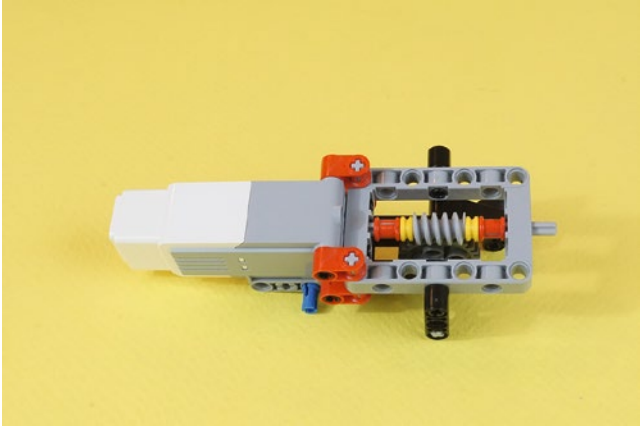


Figure 11-27. *The short axles are in all the way. This is now a very sturdy medium motor Camera Trigger assembly.*

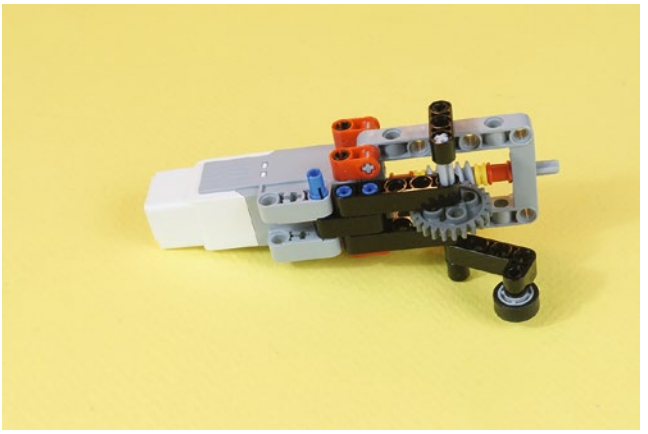


Figure 11-28. *Another view of the completed medium motor Camera Trigger assembly. The shutter lever wheel is at a good angle to operate a camera shutter.*

Engineering: Gears Trade Speed for Power

At this point, you have a medium motor Camera Trigger assembly. This assembly can be very useful anytime you want the medium motor to do something requiring strength. The worm gear gives you a 24-1 increase in power! It's nearly irresistible. Of course, the arm moves at one-twenty-fourth the speed of the medium motor. You can test this yourself—turn the worm gear six times and the arm will move 90 degrees—that is, ¼ of a turn. Twelve turns would move the arm 180 degrees—that is, ½ of a turn. With these gears, the arm moves at 1/24th the speed of the motor, but with 24 times more power.

Gears can be your friend.

But we need to finish the SnapShotBot, so let's fasten that medium motor Camera Trigger assembly to a hardpoint and move on to the camera frame.

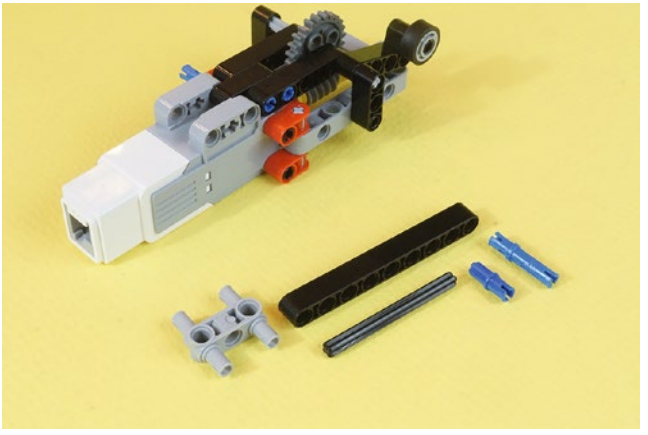


Figure 11-29. *These are new parts to mount your medium motor Camera Trigger assembly to a hardpoint, with a 9-hole beam*

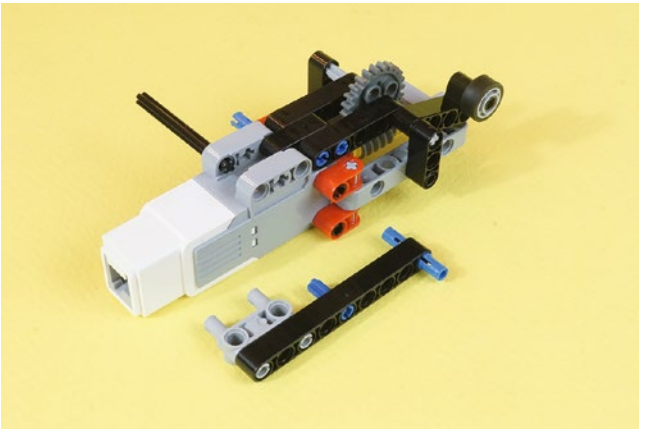


Figure 11-30. *This side beam and connectors will attach the medium motor Camera trigger assembly to the main body. The axle on the left goes through the bottom hole on the medium motor.*

Note the short axle in the middle of that 9-hole beam. It will connect to the middle axle hole on the medium motor. When you have a choice, using axles as connectors gives you more strength—that is, increased stiffness. You may need to twist the short axle a little to get it to align with the axle-hole mount on the medium motor. That mounting is shown in Figure 11-32.

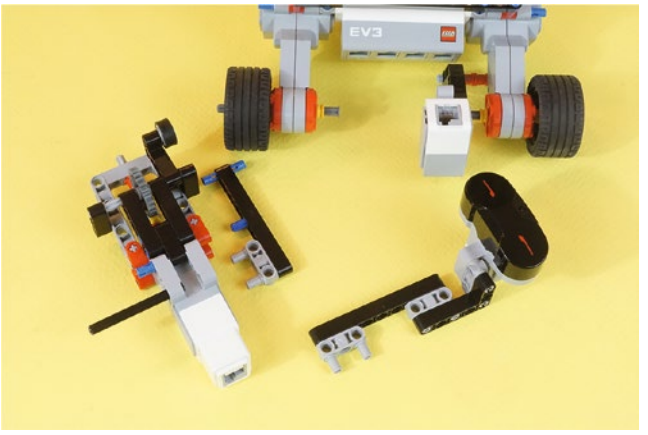


Figure 11-31. *The two assemblies are ready to connect to the hardpoints on the body*

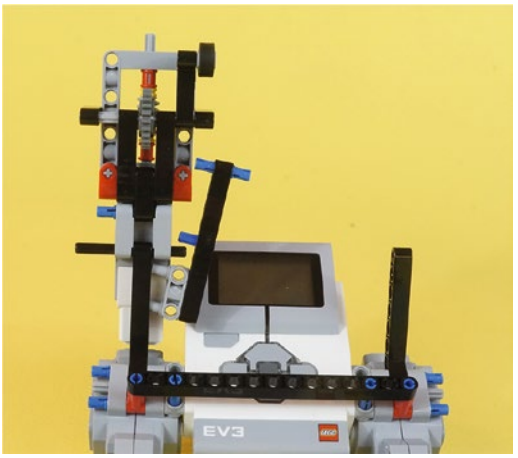


Figure 11-32. *Preparing to attach the medium motor Camera Trigger assembly. Position the assembly on the hardpoint and slide the axle through first. Then fasten the 9-hole beam with connectors. It may be necessary to twist the short half-axle to make it fit into the motor.*

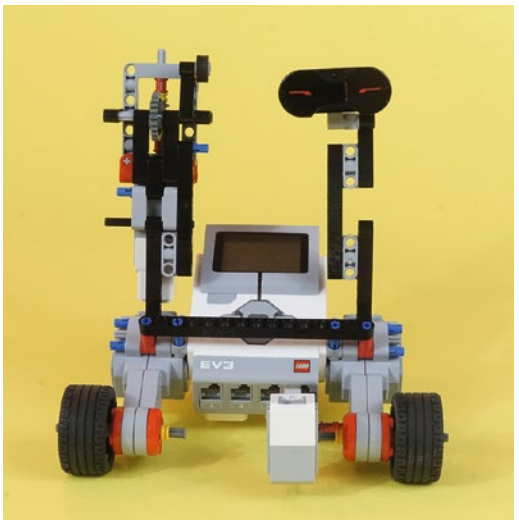


Figure 11-33. *The medium motor Camera Trigger assembly is now attached to the body. Snap the IR Sensor assembly in as shown on the right.*

■ **Note** These two tall components are where I will tie the two pieces of twine.

Third Section: Adjustable Camera Frame

We're almost finished. We need to assemble the camera frame and align the medium motor Camera Trigger assembly to press the camera's button. Figures 11-34 through 11-36 cover the completion of this bot prior to wiring.

Figure 11-34 shows the pieces needed to build the camera frame.

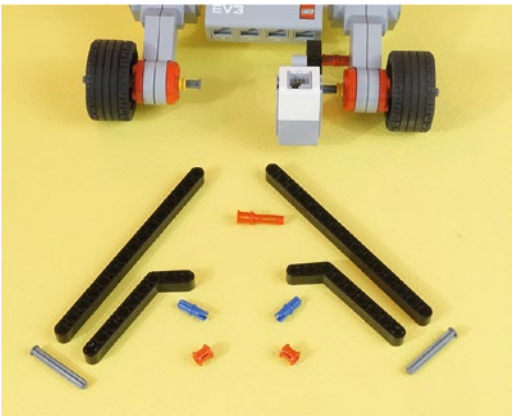


Figure 11-34. *All the parts you need to make the camera frame*

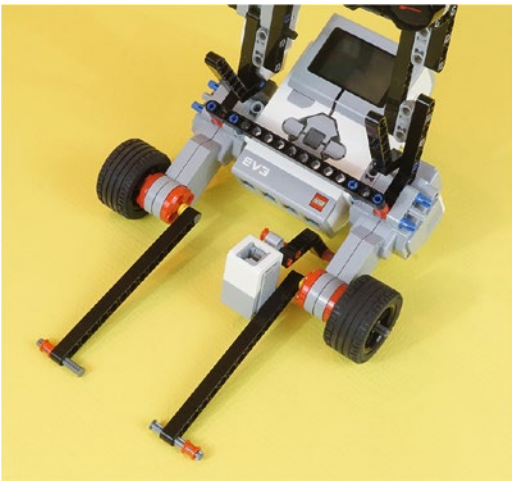


Figure 11-35. *Connect the two 15-hole beams to the axle ends. Fasten the short nailhead axles to the ends. Put the blue half-axles in the short end of the angle pieces and fasten them to the angle pieces mounted on the motors.*

The nailhead-style axles hold the long beams in place. It is inconvenient to remove the entire axle, wheel, and collars from the motor to put the beams on the nailhead end. But the reward is a simple and sturdy connection that does not fall off.

Figure 11-36. *Swing each 15-hole beam up to meet the end of the angle beam. Keeping the red collar in place, fasten this together with the short nailhead axle.*

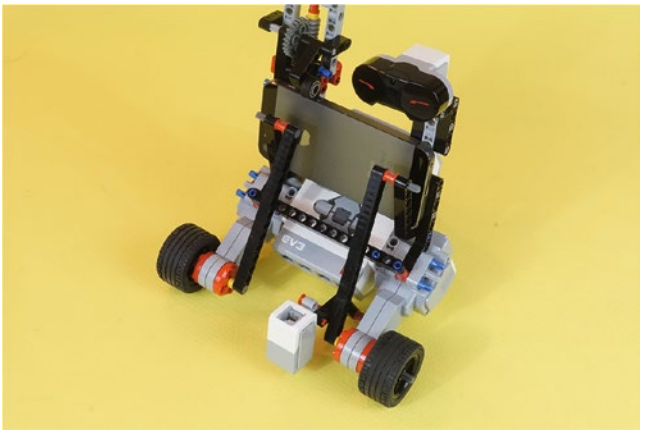


Figure 11-37. *With the original design, this SnapShotBot supports a smartphone camera. This type of smartphone will take a picture when the shutter lever wheel presses its side button. The lens is at the lower left of the phone and can see past the beams.*

Figure 11-38. *Move the 15-hole beams to a different position to accommodate this larger disposable camera. The angle pieces bend down, allowing a different point of connection. In this case, a longer lever would be needed for the shutter lever wheel to reach this camera's shutter.*

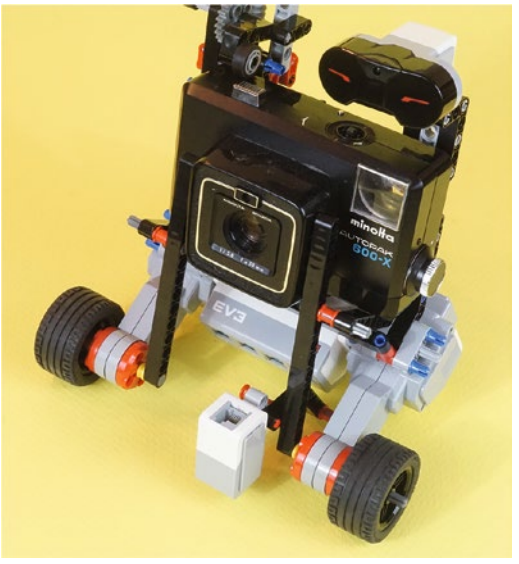


Figure 11-39. *Another style of camera. The lever on the shutter lever wheel looks just about right for this camera.*

Engineering: Early and Rapid Mechanical Testing

You'll often want to test your designs before you have a real program. In this case, you can use an angle piece as a crank to turn the worm gear. No program needed. Figure 11-40 shows how the crank can operate the worm gear, which moves the shutter lever wheel. When "auditioning" a particular camera or smartphone, being able to do this verifies the rather tricky task of effectively pushing the shutter button. Alignment is essential, and it can take a surprising amount of force to work a given shutter button. This crank idea lets you rapidly test and make adjustments.

In the engineering world, the ability to quickly test—*early in development*—can save a lot of time later. Sometimes a wonderful idea simply doesn't work, and you have to take many backward steps to come up with something that does work. In the case of this crank, you don't have to get all involved with software to figure out if there is some fundamental problem.

Catch problems early. Testing is your friend.

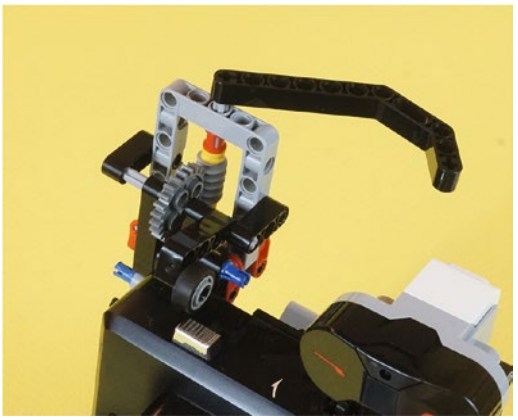


Figure 11-40. *Do you want to move the shutter lever wheel for testing? Use an angle piece as a temporary crank for the worm gear!*

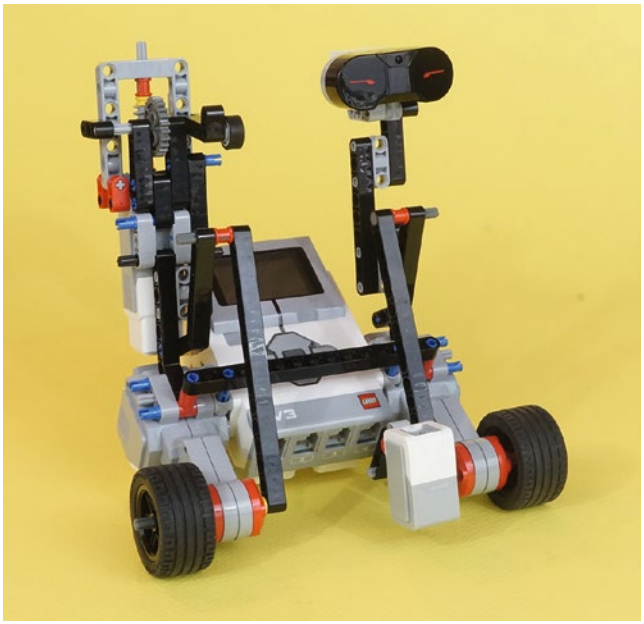


Figure 11-41. *The complete, unwired SnapShotBot*

Figure 11-42. *The completed, wired SnapShotBot, Camera Trigger assembly, front-side view*

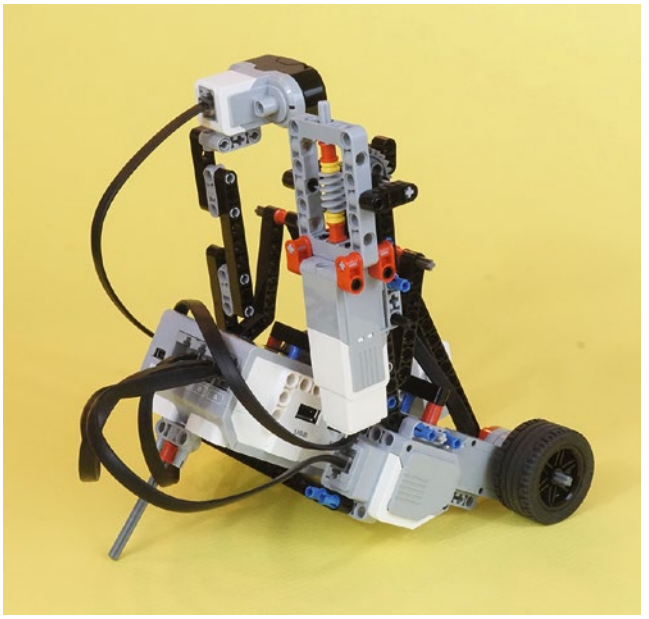


Figure 11-43. *The completed, wired SnapShotBot, Camera Trigger assembly, rear-side view*

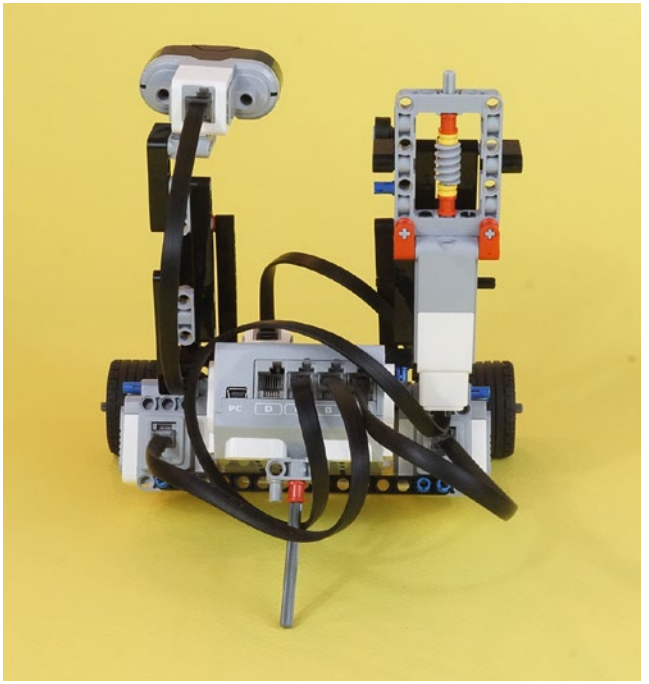


Figure 11-44. *The completed, wired SnapShotBot, rear view. This is a good view of the motor wires. The medium motor is in port C. Drive motors are in ports A and B.*

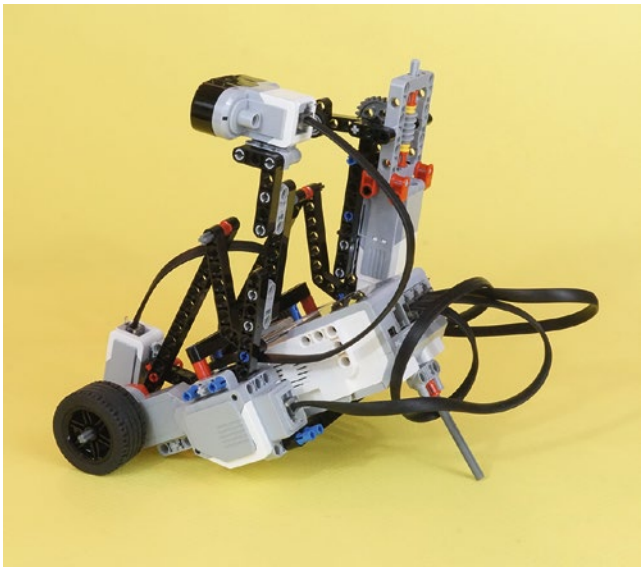


Figure 11-45. *The completed, wired SnapShotBot, IR Sensor rear-side view*

Figure 11-46. *The completed, wired SnapShotBot, front view. This is a good view of the sensor wires. The Color Sensor is in port 1. The IR Sensor is in port 2.*

Now, put your camera in the camera frame and position the arm with the shutter lever wheel so that it is directly over the button, but not pressing down on it. In Figure 11-47, you can see that I've got the small arm with the rubber wheel almost touching the camera button. When the medium motor drives the worm gear, the driven gray gear moves the shutter arm down slowly, but with the force multiplied. The worm gear turns 24 times for every full rotation of the gray gear.

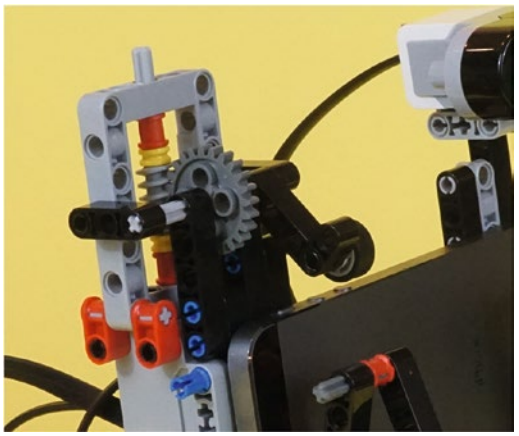


Figure 11-47. *The SnapShotBot shutter wheel, positioned close to the camera*

All that's left to do is wire it up and program it. Take another look at Figures 11-42 through 11-46 to see how I've wired up the bot.

Now that you've completed your SnapShotBot, it's time to program it. Chapter 12 walks you through the steps to program your little SnapShotBot so that it can enter the library, take a picture, circle around the basket, and let your team retrieve the key!

CHAPTER 12

SnapShotBot: Program It

You might think that programming the SnapShotBot is a little more involved than the previous bots. Although this program might be a little larger in size, the truth is that you're already experienced with all the programming blocks you'll need.

In this chapter, I'm going to walk you through the programming using a slightly different method. When I program my bots, many times I create the program in full and then download it to my bot for testing. From there, I add and remove blocks as needed. Although this is perfectly fine for many bots, for a large program this might not be the best way to test your bot. If you find a mistake early in the program, it can cause you to have to delete other portions of your program. And if you find a *really huge* mistake, you might find yourself deleting the program completely and having to start over.

So, let me show you another method for programming that involves building your program in small steps, downloading the program, then testing it. When you're done with this chapter, you should be able to decide for yourself when it might be beneficial to program in small steps or simply program the entire thing and then test and debug.

One Block at a Time

Get your SnapShotBot Design Journal page and open the LEGO MINDSTORMS EV3 software. Click on the + tab at the upper left and then double-click on the blue `Program` name. Type **SnapShotBot**; your screen should look like Figure 12-1.

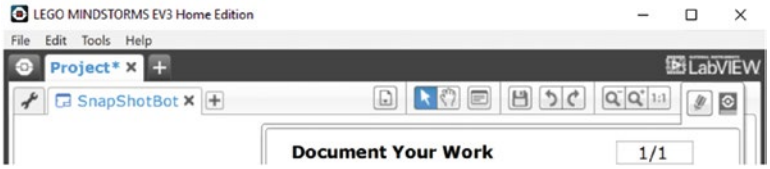


Figure 12-1. *Enter SnapShotBot for the new program name and click anywhere else on the screen*

Note To have more workspace visible on your screen, minimize the Document Your Work area on the right by clicking the EV3 Button symbol at the far right of the top tab bar. (This symbol is shaped like a little stop sign—that is, an octagon. It resembles the buttons on your EV3 Intelligent Brick.)

M. Bell and J. F. Kelly, *LEGO® MINDSTORMS® EV3*, DOI 10.1007/978-1-4842-2262-1_12

In the SnapShotBot building instructions in Chapter 11, you might have noticed I placed the Infrared Sensor facing the front of the bot. I plan on using the Infrared Sensor as a sort-of Start button for my bot to get rolling. So, the first item you'll place in your program will be a simple LOOP block (see Figure 12-2) that waits for the Infrared Sensor to be triggered (remote button pressed and then released). Once the sensor is triggered, the remaining program will begin. (You could use a WAIT block that breaks when the Infrared Sensor is pressed, but I like to use the LOOP block because I can later add blocks inside the LOOP if I want the bot to perform some other actions while it's waiting.)

"But the sensor is facing forward, not toward the rear," you say. That's true. But it turns out that the IR Remote has a fairly powerful signal that reflects nicely off of other surfaces, so the beam will end up on the IR Sensor. When you get the bot built, try "shooting" the sensor from different angles, including from behind. You'll find that reflected IR light works surprisingly well. If your wall does not reflect well enough, you could mount the IR Sensor facing the back. Or, you could change this program and run it backward!

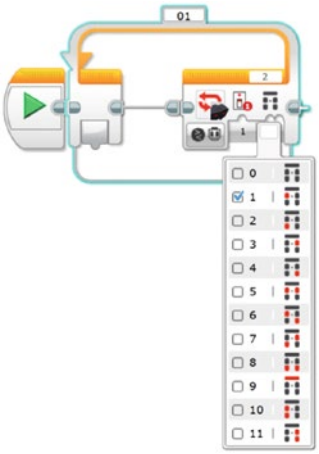


Figure 12-2. *Use a LOOP block and Infrared Sensor - Remote trigger to start the bot. The blue highlighting and pull-down menu appear when you click on the tab below the EV3 button symbol. The pull-down shows which Remote button will exit the LOOP.*

Now you can start on the actual programming blocks needed to perform the actions in the Task List. Look at the Task List on your Design Journal page (turn back to Figure 10-3). The first task is "Move forward to the center of room and stop." If you'll think back to Chapter 10, we determined that to get halfway across the room, we needed to program our motors to spin for 13 rotations. Each rotation moves the bot approximately 5.5 inches, so when the motors spin for 13 rotations, the bot will move forward approximately 71 inches, or almost six feet. That'll be good enough to get the bot near the center of the room.

Place a Move Steering block and configure it with a **Duration** of 13 rotations for motors B and C. Be aware that I've selected the direction for motors B and C as minus (see Figure 12-3). This is because the motors are reversed on my design—facing the front of the bot. Because of this, forward motion for the bot means having these motors spin in the "reverse" direction. If your bot differs, configure your motor directions based on your own design.

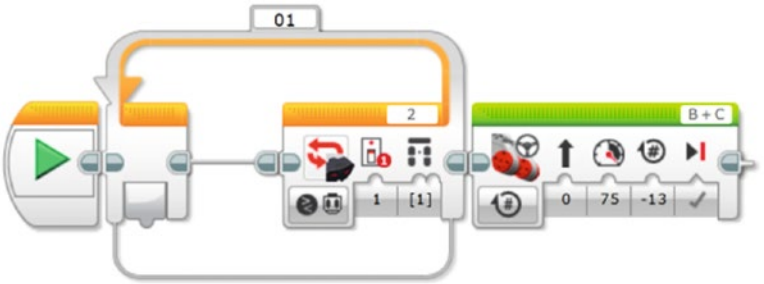


Figure 12-3. *Configure the Move Steering block to move the bot forward, halfway into the room*

Now it's time to test. I'm going to save the program (so I don't lose any of my work or the comments I've added) and download it to my bot. When I run the program, the bot should wait until I press the upper-left Infrared Sensor Remote button. After I trigger the sensor, the bot should move forward about six feet and stop. Here goes the test . . . and it worked, exactly as designed. (If your bot did anything differently, check your programming blocks and verify that the number of rotations is correct.)

Next on our Task List is "Turn left to face library." Back in Chapter 10, I reminded you how to test your bot and obtain the number of degrees to turn for making a right and left turn. When I tested my bot, I obtained a value of -678 degrees when *only* the right-side wheel was turned (motor B). Drop in your `Large Motor` block and configure it as shown in Figure 12-4.

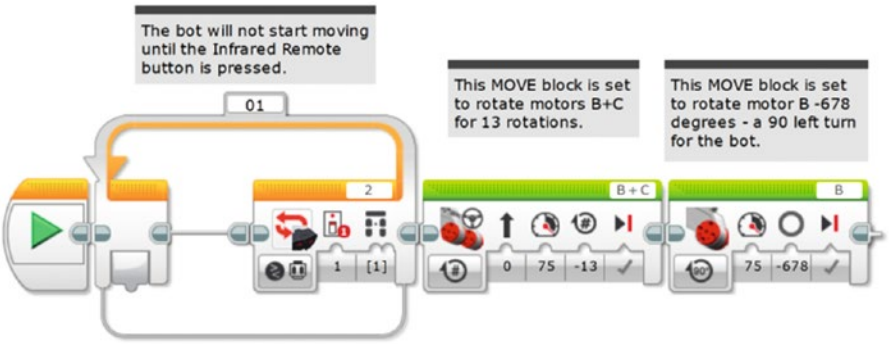


Figure 12-4. *This Large Motor block will make the bot turn left*

Now we test again. When I trigger the Infrared Sensor, the bot moves forward about six feet. It then makes a left turn when motor B turns -678 degrees. So far, so good.

The Task List shows our next step is "Take picture." Before I configure motor A, however, I do want to add in a `WAIT` block. I'm worried that motor A might push the button on the camera just as the bot is coming to a stop. This would result in a blurred picture. I'll configure the `WAIT` block for two seconds (see Figure 12-5).

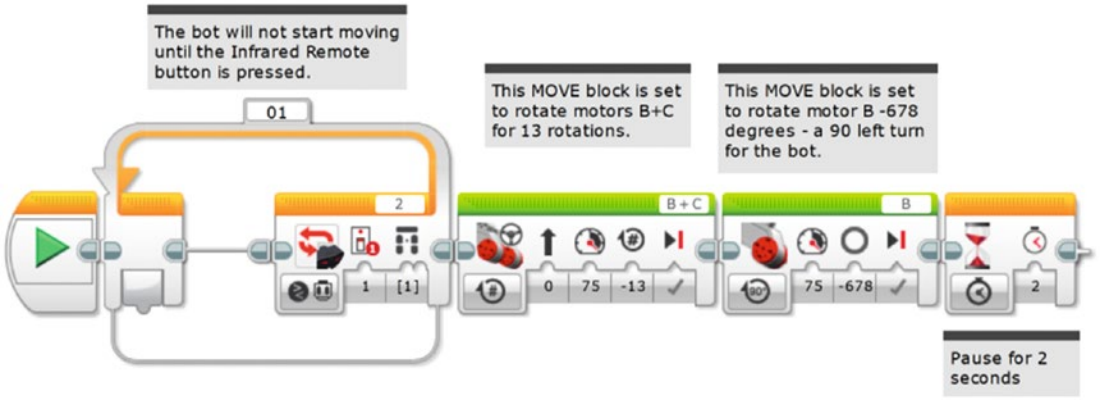


Figure 12-5. *A WAIT block allows the bot to stop moving before the picture is taken*

Next we need to configure Medium Motor A to rotate and press the camera button. Because I don't know how far to turn it, this is where testing comes in. I dropped in a `Medium Motor` block and configured the motor A to turn three rotations. Since the Medium Motor Camera Trigger Assembly divides motor rotations by a ratio of 24 to 1, three rotations results in exactly 45 degrees of shutter arm movement. During testing, I found this sometimes missed the button entirely. I reduced it to two rotations/30 degrees and tested again; I got the same result. For my third test, I placed the arm directly on the button of the camera and configured the `Medium Motor` block for one rotation/15 degrees; this time it worked! The button pressed and a picture was taken (see Figure 12-6). And, adding the reverse `Medium Motor` block for the shutter resets it, which is useful when you're doing one test after another.

■ **Note** Depending on the type of camera you're using, you'll probably have to perform a few different tests to determine the best way to press the camera button. When trying different motor settings, such as number of rotations or degrees, it's usually best to start big. Start with a large number of rotations or degrees—if it works, reduce it a little and try again. Keep reducing until you find the lowest setting that works. Your batteries will last much longer!

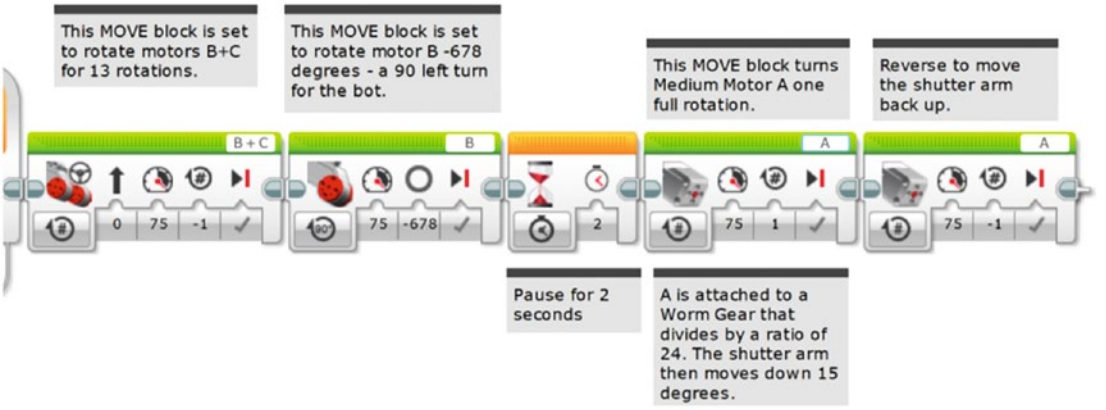


Figure 12-6. *Using the Medium Motor block to make Medium Motor A take a photograph*

Finding the Basket

Okay, at this point, our little bot has made it to the center of the room, turned left, paused, and then taken the picture. Looking at the Task List, I see that my next step is "Move forward until black obsidian ring is detected and stop." I want my bot to start moving forward until its Light Sensor detects the black obsidian rock surrounding the basket. I also want the bot to move forward slowly so it doesn't get too close to the basket. When the Light Sensor detects a change in the reflected light, I want the bot to stop.

Back in Chapter 10, I told you how to test your Light Sensor to obtain a reading for a "normal surface" and an "obsidian surface." When I tested my "normal floor" (a light-colored linoleum floor), the Light Sensor returned a value of 80. When I placed the dark paper (black) under the Light Sensor, I got a value of 10. Your values will probably differ a bit. I'm going to program my Light Sensor to check for a value of 20 or less to be safe. When the Light Sensor is triggered (that is, it has a value of 20 or less), the bot is to stop moving forward because it has detected the "obsidian surface."

Now, to keep the bot moving forward until the Light Sensor is triggered, you're going to need to use a LOOP block configured to test the Light Sensor. You can see the settings for the LOOP block in Figure 12-7.

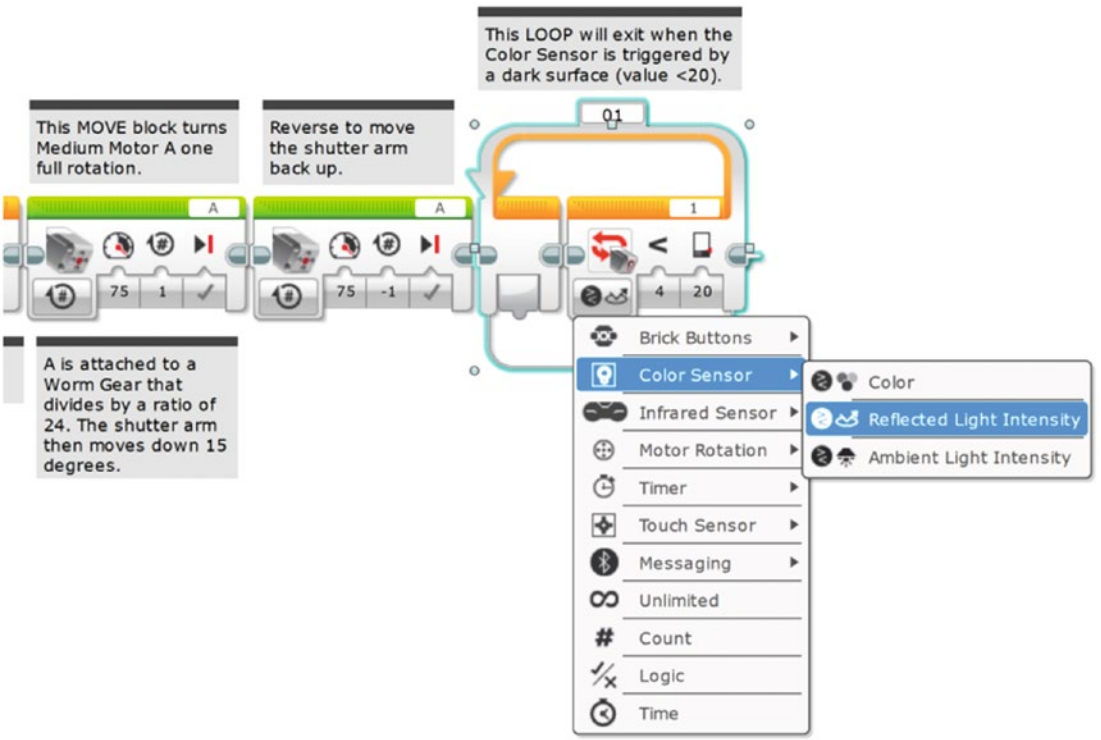


Figure 12-7. *Using a LOOP block to test whether the Light Sensor is over the obsidian surface. The pull-down menu shows how we select Color Sensor – Reflected Light Intensity.*

To make the bot actually approach the basket, throw in a `Motor Steering` block. With this `Motor Steering` block, configure motors B and C with a **Duration** of Unlimited. The Light Sensor will check to see if it's over the "obsidian surface." As soon as it is, the bot will exit the LOOP. Then it will coast another two or three inches. To prevent coasting, place a `Motor Steering` block configured to `STOP`. By doing it this way, you can make sure the bot doesn't rush forward quickly and cross too far over the obsidian ring (see Figure 12-8).

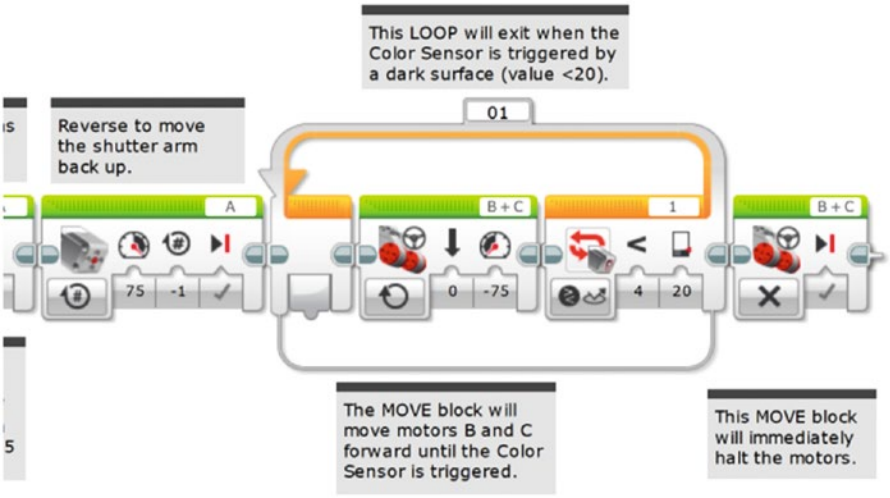


Figure 12-8. *Using a LOOP and Motor Steering block with the Light Sensor, followed by a Motor Steering set to OFF*

After the Light Sensor is triggered and the bot stops, the bot will need to go around the basket with the twine and prepare for the trip back.

Getting Around the Basket

Now the bot is in front of the basket. What's next? Look at the Task List: "Turn right, move forward short distance, and stop."

Our first task will be to get the bot to turn right. Back in Chapter 10, I recorded the value as -640 degrees for a right turn. Remember, the negative sign (–) indicates that motor C is turning in reverse. So you'll insert a `Motor Steering` block that's configured to turn motor C in reverse for -640 degrees (see Figure 12-9).

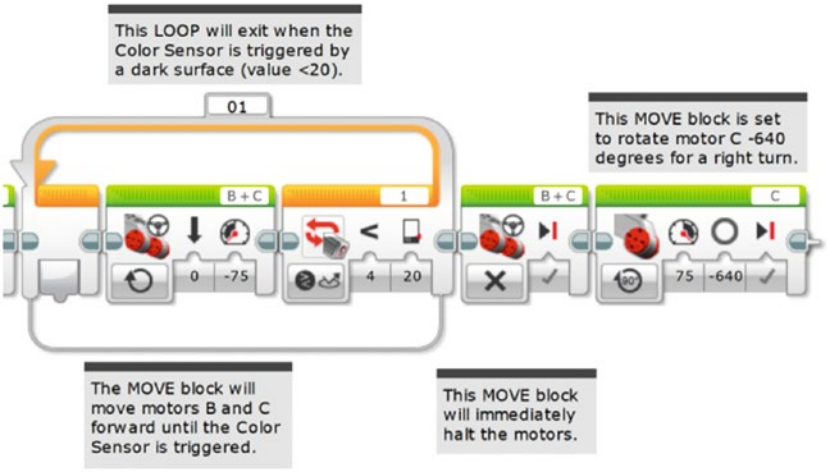


Figure 12-9. *A Large Motor block will allow our bot to make a right turn*

We determined in Chapter 10 that the "short distance" was about one foot, and that we would need motors B and C to perform 2.2 rotations. We'll insert a `Motor Steering` block (see Figure 12-10) that will get us to point A (turn back to Figure 10-8 for the mini-map).

If you examine the mini-map in Figure 10-8, you'll notice that once the bot reaches point A, it will make three left turns before heading back to its starting point. It will turn left (1) at point A, move forward, and stop at point B; turn left (2), move forward, and stop at point C; turn left (3), then finally move forward, and stop at point D. This is a perfect location to use a `LOOP` again. It allows us to reduce the number of programming blocks by performing the same actions three times. Those actions are "Turn Left and Move Forward a Short Distance." So let's place the `LOOP` first (see Figure 12-11).

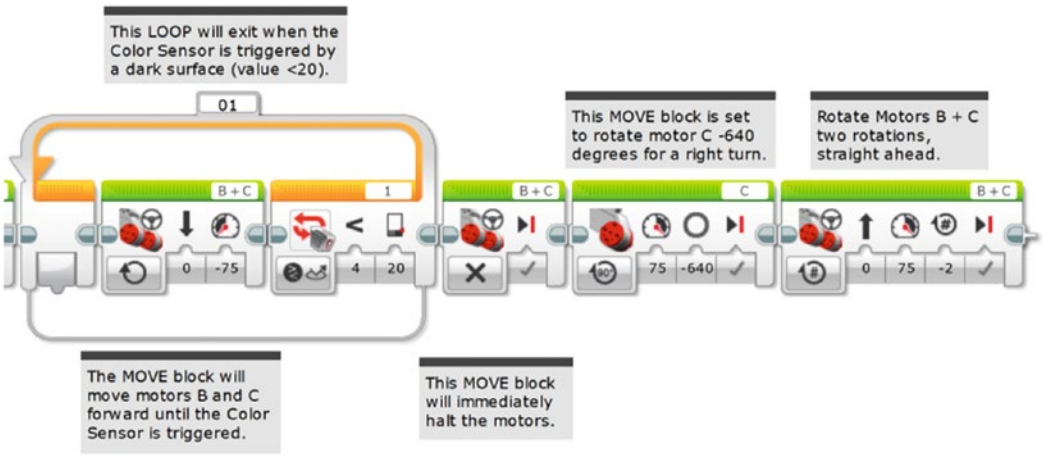


Figure 12-10. *Complete the move, with the motors coming to a stop*

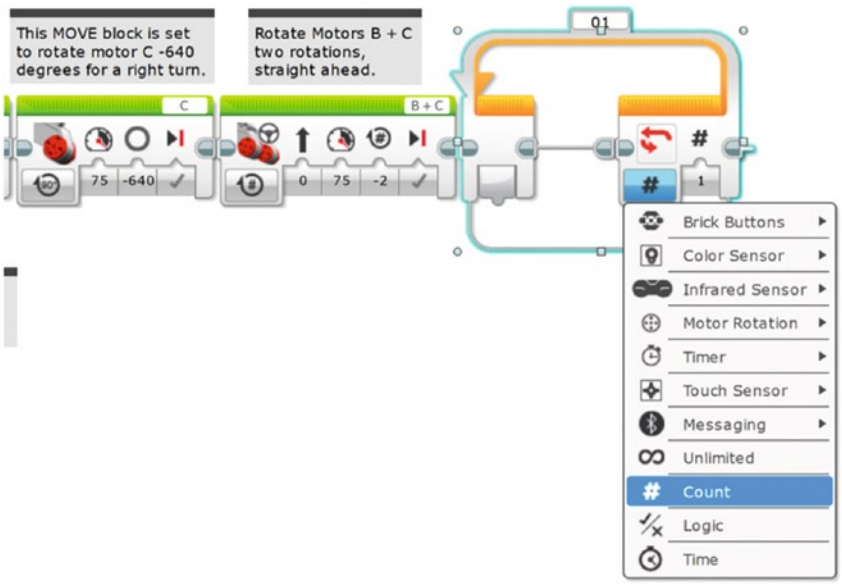


Figure 12-11. *The LOOP block will get the bot around the basket. The pull-down menu shows how to select Count.*

The LOOP block in Figure 12-11 will run three times. Any blocks placed in it will also run three times before the LOOP block breaks and the program continues. What will happen three times? First, a left turn: a `Motor Steering` block will turn the bot left. Second, the bot will move forward approximately two or three feet. Back in Chapter 10, we determined that 3.5 rotations would be needed to move our bot two feet. Let's round that number up to four rotations (28 inches), which should give us plenty of room for our bot to navigate around the basket. So, you'll first place the `Motor Steering` block for a left turn (see Figure 12-12).

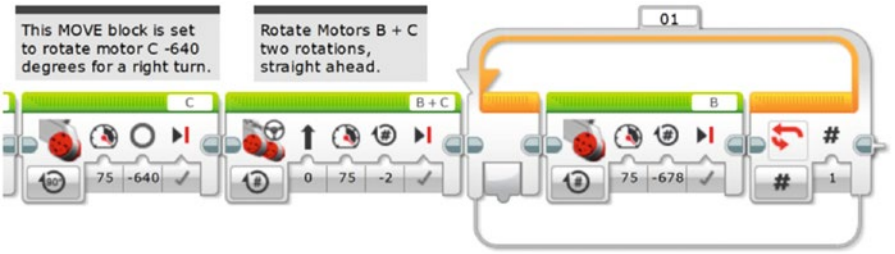


Figure 12-12. *This Motor Steering block turns the bot left*

Next, place the `Motor Steering` block to move the bot forward 28 inches (see Figure 12-13). But take a look at that `Large Motor` block. It is set for -678 degrees, right? Well, when I ran that program in Figure 12-12, motor B turned… and turned… and turned! It seemed to run forever. I interrupted the robot and looked more closely at the program. It was set for -678 *turns!* So I changed the Turns symbol to Degrees. That's what you see in Figure 12-13. I put the next `Motor Steering` block in, to advance motors B and C four rotations. Again, I downloaded the program, simply running it while holding the SnapShotBot in my hand to test it. All without getting up.

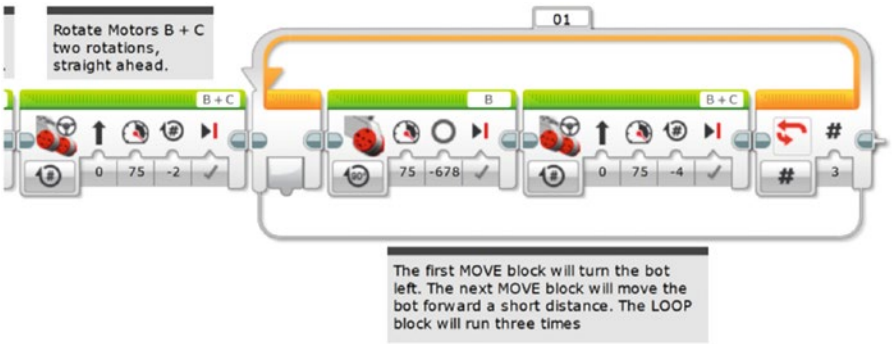


Figure 12-13. *This Motor Steering block moves the bot forward a short distance*

Now you need to test your bot. At this point, let me describe my test environment and show you how to set up your own.

First, I'm running this test in my living room. I've cleared away a chair so I'll have plenty of room for my bot to run. In Figure 12-14, you can see how I've set up my test run. I've placed the "basket" (actually a plastic container with a remote control inside for weight) about six feet forward and three feet left from my bot's starting position. I've also placed my piece of dark paper to the right of the basket for the bot's Color Sensor to detect. If all goes well with my practice run, my bot should end up at point D.

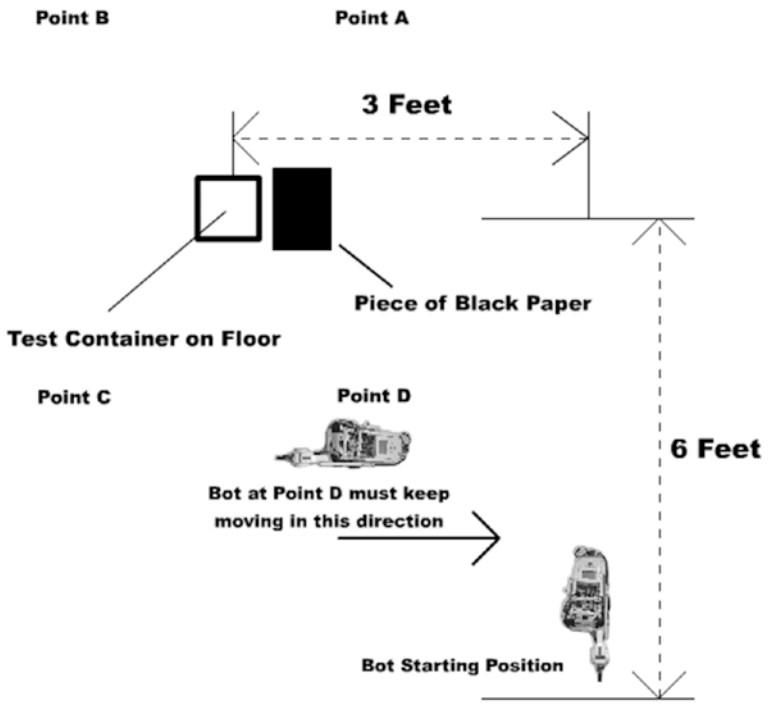


Figure 12-14. *Test environment for the SnapShotBot*

Because I've been testing my bot frequently, I wasn't surprised that my test run was successful. I knew from previous tests that my bot would enter the library, turn left, take the picture, and move forward until the Color Sensor was triggered. Getting the bot around the basket using a series of three left turns and forward movements worked perfectly. My bot ended up roughly in the area I've labeled point D, pointed right and ready to finish its return.

Getting the Bot Home

The hard part is done. All that's left at this point is getting the bot back to you. There are numerous ways to do this, but the one I've selected uses the Infrared Sensor. If you'll look again at Figure 12-14, my bot is stopped at point D, pointed to the right (back toward the dotted line where the bot first started its motion into the library). What I plan on doing is programming my bot to continue moving forward (moving to the right in Figure 12-14) until I "see" it. What I mean by "see" is that in the real library, the team will be looking through the hole in the wall. When the bot first enters the library and turns left toward the basket, the team will lose sight of the bot. Only when the bot returns and continues to move away from the basket will the team see the bot again. What I plan on doing is simply pressing a Stop button on the Infrared Remote. And as the programmer, I get to decide which button that will be! The Infrared Sensor will detect the IR Remote button press, and I'll program the bot to stop, turn right, and come back to the hole in the wall. Simple!

First, you add a `LOOP` that will break when the Infrared Sensor is triggered (see Figure 12-15).

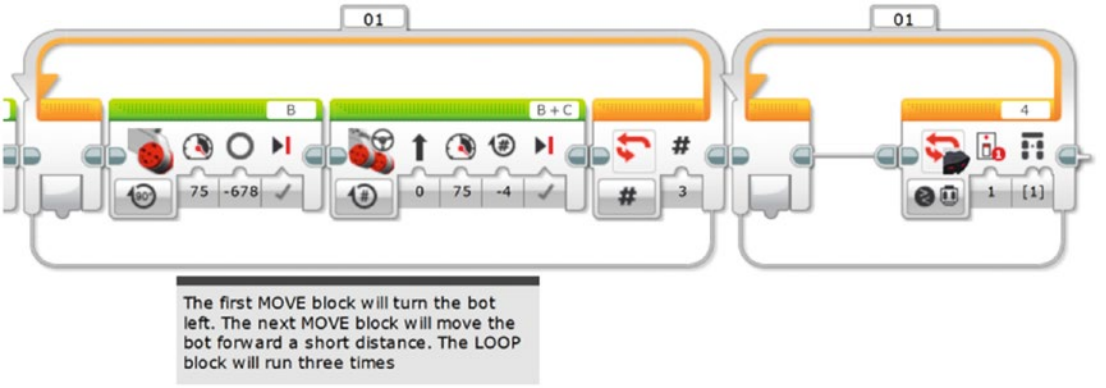


Figure 12-15. *Add a LOOP block with the Infrared Sensor configured. This will also use Button 1 of the Infrared Remote.*

Next, you add a `Motor Steering` block that will keep motors B and C moving until the Infrared Sensor is triggered (see Figure 12-16).

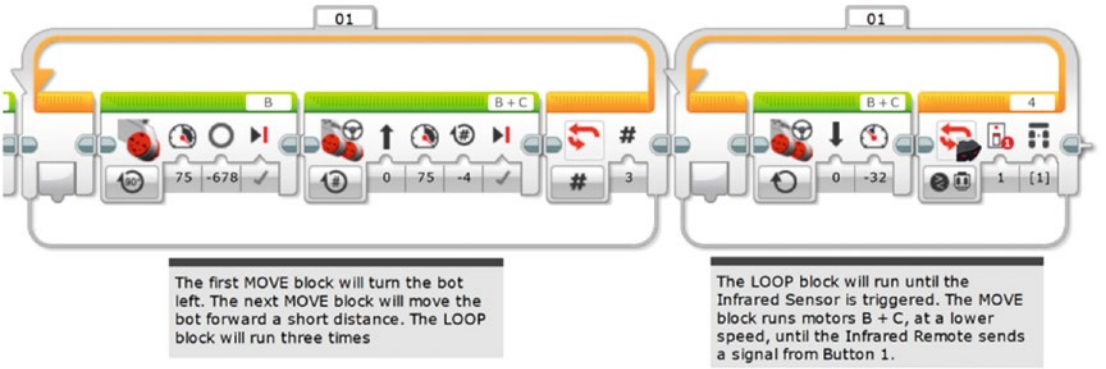


Figure 12-16. *The Motor Steering block will run the motors until the LOOP is broken by the Infrared Sensor*

Once again, we test. I ran the IR Remote with the same button I used before, and the Infrared Sensor didn't work! What's going on? Wrong button? No, it's the same button we used before. Nuts. Oh, the Infrared Sensor port is set to 4. But the sensor is still *wired to* port 2, where we used it before. Okay, we need to change port 4 to port 2 in the program.

Test. Again. With the robot in the hand. Now, the Infrared Sensor picked it up fine. Once the bot started rolling, it immediately stopped at my command. Note that it is a *lot* easier to do these little tests like this with the robot in your hand rather than stepping away from your computer, setting it on the ground, pressing the gray Select button to start the program, hitting the IR Remote, etc. Handling the light floor/dark tape scenario is easily done with white paper and something dark, all held in your hand or on your table.

The next task is "Turn right and move forward to library exit." That should be easy enough. Drop in a `Large Motor` block and configure it for a right turn (see Figure 12-17). And let's double check: yes, it is motor C, and yes, it is set for -640 degrees, just like the one in Figure 12-9. It pays to check and it saves us having to re-test, if we get it right the first time.

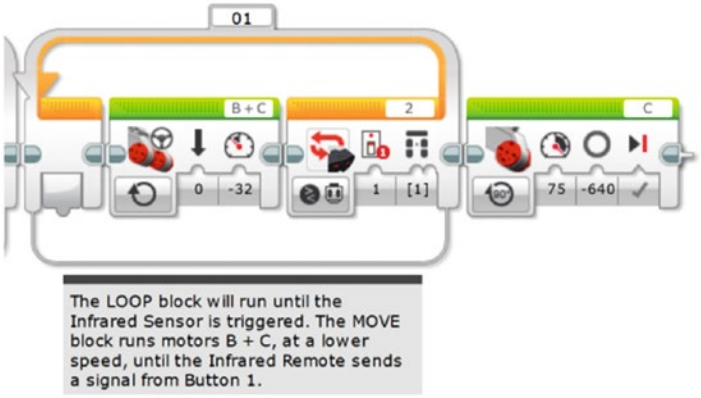


Figure 12-17. *This Large Motor block turns the bot right*

Use one final `LOOP` block to run motors B and C continuously. Drop in the `LOOP` block first (see Figure 12-18). Now, you might think "Why do I need that `LOOP` block? Can't I just drop in an Unlimited `Large Motor` block by itself as my last program instruction?" It turns out that does not work. The motor does have to be in the `LOOP` block or the last motor command will simply not execute. This can cause a few moments of head scratching since it certainly seems like it ought to work.

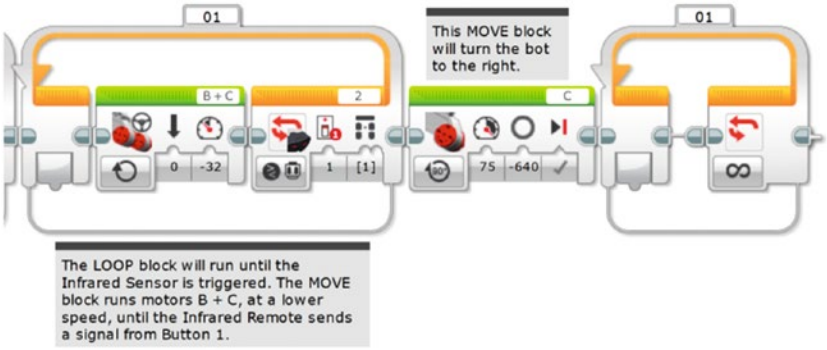


Figure 12-18. *This LOOP block will run the motors continuously*

Then drop in the final `Motor Steering` block, configured to run motors B and C for an Unlimited duration (see Figure 12-19). In this case, the slower speed of -32 seemed like a good choice since the bot would be dragging two pieces of twine.

When this is all working, you'll simply reach in, grab the bot, and turn it off . . . Oh, and be sure to share that smartphone photograph with the team!

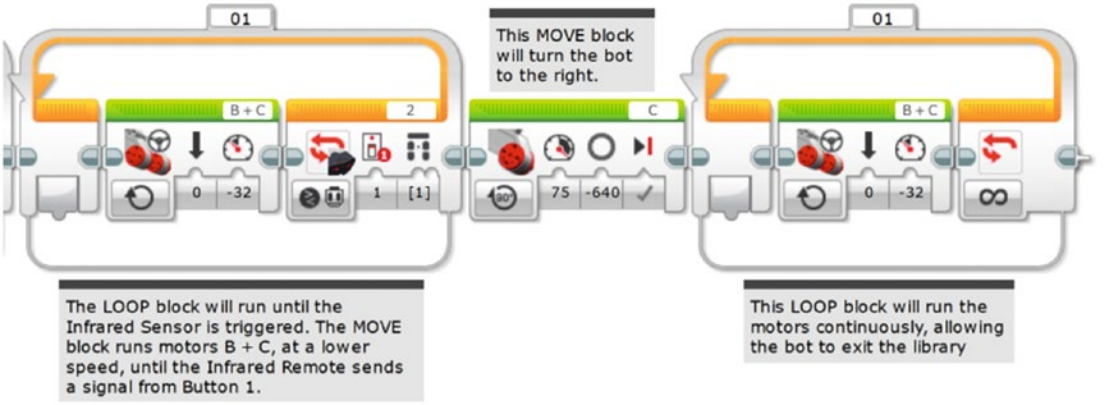


Figure 12-19. *This Motor Steering block runs motors B and C*

That's it! Download the program and test it. And test it again. Once you're confident that your SnapShotBot is running perfectly, set up your test environment one more time for the final test. Place the basket, the dark paper, and the bot in their starting positions. If you're using a regular camera or disposable camera, be sure you've wound the camera before sending the bot off. As shown in Figure 12-20, use the Angle Crank on the top of the Worm Gear SubAssembly to position the camera shutter arm up or down, directly above the button.

Get two rolls of string (or twine) and tie one end of each roll to the SnapShotBot. I am using the two tall assemblies on the top of my SnapShotBot to tie my strings. Remember to give plenty of slack on the string. Even better—have someone help you hold one of the rolls so you can both watch the strings and keep the tension off. When you're ready, press and release the Infrared Sensor and watch your SnapShotBot begin its job. As it rolls in, be sure to feed out plenty of string. You don't want the string to get tight and pull the bot off course. When the bot turns to take the photograph, feed some more string as it moves toward the basket.

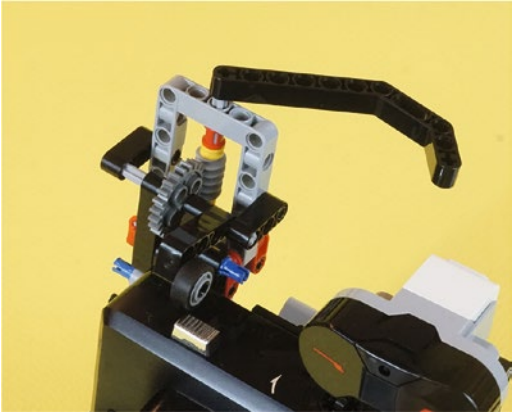


Figure 12-20. *This Angle Crank lets you move up or down, to precisely position the camera shutter arm with its small rubber wheel*

As the bot moves around the basket, you'll continue to feed it more string. Watch for it to return, and when you see it, press that IR Remote Stop button. The bot should turn (don't forget to keep feeding more string) and head in your direction. If you've given the bot plenty of string, it should come right back to you. Grab the bot and turn it off.

Now, cut the two string ends from your bot and cut the string from the two rolls. Take all four string ends and place them together in your hand. Start pulling very slowly. Keep pulling slowly until you feel some resistance. When you feel the string tighten a little, pull even more slowly. At this point, the string should be pulling the basket toward you. Keep pulling until you've got the basket (and the key!) in your hands.

Congratulations! You've got the key, which means the team can continue exploring King Ixtua's tomb.

CHAPTER 13

Get In, Grab It, Get Out

Location: Southwest Guatemala

Weather Conditions: 84° Fahrenheit, Humidity 48%, Rain 0%

Day 4: Outside King Ixtua's Library, 8:43 AM

Evan laughed as he watched his uncle slowly pull the twine. His uncle kept trying to grab the basket, but it was still four or five feet away. Evan could tell Dr. Hicks was anxious to retrieve the key.

"And I've got it!" Uncle Phillip yelled, his voice echoing down the hallway.

Uncle Phillip stood, turned, and faced the rest of the team, holding the basket above his head.

"Is the pa'aachi inside?" asked Grace, nervousness in her voice. Without the key, the team would be unable to continue its exploration of the tomb.

Evan watched as a smile slowly appeared on his uncle's face.

Uncle Phillip reached into the basket and pulled out an unusually shaped object. He turned it over in his hands, letting Max, Grace, and Evan get a good look.

Made of animal bone, the key was over a foot in length. The key was shaped like a walrus tusk, with one end almost two inches in diameter and the other end a small, dull point. Its surface was covered with carved Mayan glyphs, and the key had a dozen notches cut into it.

Max began taking photographs of the key. "Can you turn it over, please?" He took another photo and then lowered the camera. "I hate to rush things, but can we maybe try it out?" he asked.

Uncle Phillip laughed. "I was thinking the exact same thing. Let's do it," he replied.

The Throne Room

Uncle Phillip inserted the key into the hole in the floor. When nothing happened, he twisted the key clockwise. From behind the wooden door, the team heard a loud SNAP!

"I think that did it," said Evan.

"I think you're right," replied Uncle Phillip with a smile. "And if I'm right about what is behind that door, we're almost to the king's burial chamber. Max, take a photo of this, please."

Evan watched as Max photographed Uncle Phillip pushing against the large wooden door. He expected a loud creaking sound, but the door opened smoothly.

After the door was open, Uncle Phillip peered into the darkness, shining his weak flashlight around. "Grace, would you bring me those portable lights?"

M. Bell and J. F. Kelly, *LEGO® MINDSTORMS® EV3*, DOI 10.1007/978-1-4842-2262-1_13

Grace picked up two of the small battery-powered lights and handed them to Uncle Phillip. "The manuscript states the throne room is safe. No traps," she said.

"Let's light up the room first," Uncle Phillip said. "Just in case."

Evan watched as his uncle placed the two portable lights on the floor, just inside the room, and turned them on. The lights flickered for a few seconds and then flooded the room with a bright white light.

"Okay, Max. Let me have a few of those bags of sand," said Uncle Phillip.

Evan had wondered about the six bags that Max had just carried in. He had seen Max filling them with sand earlier. He watched as his uncle tossed a bag into the room, followed by another and then another. After Uncle Phillip had tossed all six bags, he took a step into the room. "Give me just a minute," he said. "Wait until I give the all-clear."

"Be careful, Uncle Phillip," said Evan, stepping closer to the door to watch his uncle.

"He'll be fine, Evan," said Grace. "The manuscript says that when Tupaxu built this tomb, King Ixtua requested that no traps be built in his throne room. He didn't want anyone hurt or trapped inside."

Evan continued to watch as his uncle walked slowly around the room, examining the corners, floor, and walls. The boy asked, "So is there anything special about the throne room?"

Grace shrugged her shoulders. "The manuscript doesn't give us any detail about the room other than it was designed to look just like the throne room the king used when he was alive."

"But many of the Mayan throne rooms that have been found also contain a secret passage to the burial chamber," added Max. "This throne room is the last room sketched in the manuscript. The burial chamber wasn't included. I'm betting that we'll find the burial chamber connected to this room."

Evan smiled. "That would be awesome to find," he said. "My friends will never believe me when I tell them what I've been doing this summer."

Max raised the camera to his eye. "Smile, Evan," he said as the camera flashed. "We'll take some more before you go home."

Before Max could reply, Uncle Phillip appeared in the doorway. "Okay, everyone come on in. Don't touch anything just yet," he said. "Max, I need you to take plenty of pictures and examine the king's throne. Grace, I need you to check out the door leading to the library and help me with a pedestal in the room. And Evan, I may have another special project for you."

Locate the Burial Chamber

One hour after the team had entered and examined the throne room, Uncle Phillip announced "Okay, team meeting in the tent. Let's go."

As the team assembled and pulled up chairs, Grace placed a large piece of posterboard on the table.

"All right, looking at Grace's drawing here, I'll give you my initial thoughts," said Uncle Phillip. (See Figure 13-1.)

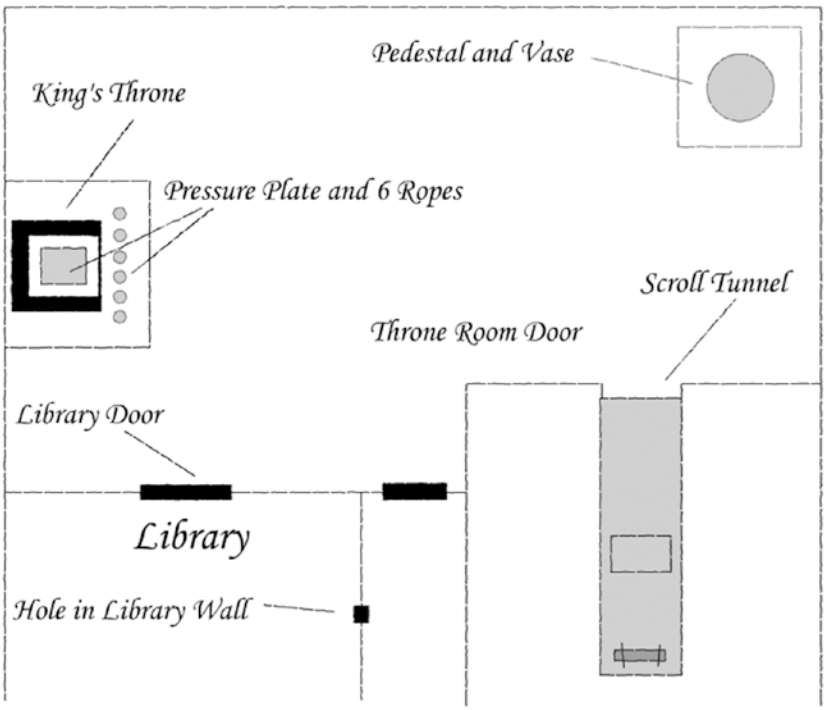


Figure 13-1. *Grace's sketch of the throne room*

"Since the manuscript doesn't give us any information on this room, we need to look at what we do have," said Uncle Phillip. "There do not appear to be any doors other than the library door and the door we used to enter. Max examined the throne and there does appear to be a pressure plate that triggers if someone sits down. There are also six small, thick ropes coming up from holes in the floor in front of the throne. There is a very large vase on a pedestal in the opposite corner of the room. Grace and I did not find any traps or similar pressure plates on this pedestal or under the vase. And, finally, there is a tunnel in one wall, about 12 feet deep, with a scroll at the end. Any thoughts? Grace?"

"I think the burial chamber is under the pedestal. If we don't figure out how to open it correctly, though, my guess is that a trap will trigger and close off the chamber permanently," Grace replied.

Max nodded. "The scroll in the tunnel is probably important in some way, too."

Uncle Phillip turned to Evan. "Evan, do you have any ideas?"

Evan looked at Grace's sketch of the room and smiled. "Those six ropes have to be important, too. When I was looking at them, my instinct was to pull them," he replied.

Uncle Phillip smiled at Evan. "I felt the same way. You almost can't resist pulling on them." He leaned back, crossed his arms, and looked at each of the team members. "Well, would anyone like to hear my guess?"

Evan, Grace, and Max all nodded and smiled together.

Uncle Phillip pointed first to the scroll. "We need to get the scroll. I believe I see three pressure plates in the tunnel that will trigger if anything heavier than a monkey, or a robot, crosses over the plates," he said with a smile and a nod to Evan. "The tunnel is large enough for a person to crawl down it, but I think that's a trick. If the pressure plates are triggered, the burial chamber will probably be lost to us for good."

"What about using a long pole with a hook on the end to grab the scroll?" asked Max.

"I thought of that, too," said Uncle Phillip. "My only concern is that Tupaxu might have thought of that as well. He was very smart. He probably designed the tunnel so that last pressure plate *must* be triggered when the scroll is lifted. It's probably more sensitive to weight and would guarantee that a small monkey was in the tunnel and not a human."

Max looked at Evan. "Have we told you how glad we are that you brought that robotics kit with you?" he asked.

Evan smiled. "I can probably get something built to get the scroll," he replied.

"Good," said Uncle Phillip. "Because I believe that scroll will have instructions on how to locate and enter the burial chamber. I think that someone will need to sit on the throne, triggering the pressure plate. Once the plate is triggered, my guess is that one of those ropes will need to be pulled. All the other ropes will probably trigger a trap that will also make the burial chamber inaccessible."

"But what if sitting on the throne triggers the trap?" asked Grace. "Maybe Tupaxu designed the throne only for King Ixtua to sit on?"

Uncle Phillip nodded. "You might be right. That's why the scroll is so important. I still think it will tell us exactly what we need to do."

Max and Grace nodded in agreement.

"Evan, why don't you and Max go and take a closer look at the tunnel. Take any measurements you need, okay? Grace and I will be in the library if you need us. This is important, so take whatever time you need. There is no rush on this one," Uncle Phillip said.

Max stood up and stretched. "Ready?" he asked Evan.

Evan took a deep breath and let it out slowly. "Okay," he said. "Let's go."

Scroll Challenge

"Two feet tall, two feet wide," said Max, measuring the height and width of the tunnel entrance.

Evan wrote down the information in his Design Journal. "That gives me plenty of room for a robot."

Max pointed his flashlight down the tunnel. "The measurements for the scroll will have to be estimates. What do you think? Does the scroll look about a foot in length?" he asked. "Looks about three or four inches from the back wall, too."

Evan peered down the tunnel. "Yeah, that's about right. And maybe four or five inches above the tunnel floor?"

Max nodded. "I wish we could be exact, but let's make it four inches to be safe," he replied. "I think your uncle is right. That does look like a pressure plate in front of the scroll." (See Figure 13-2.)

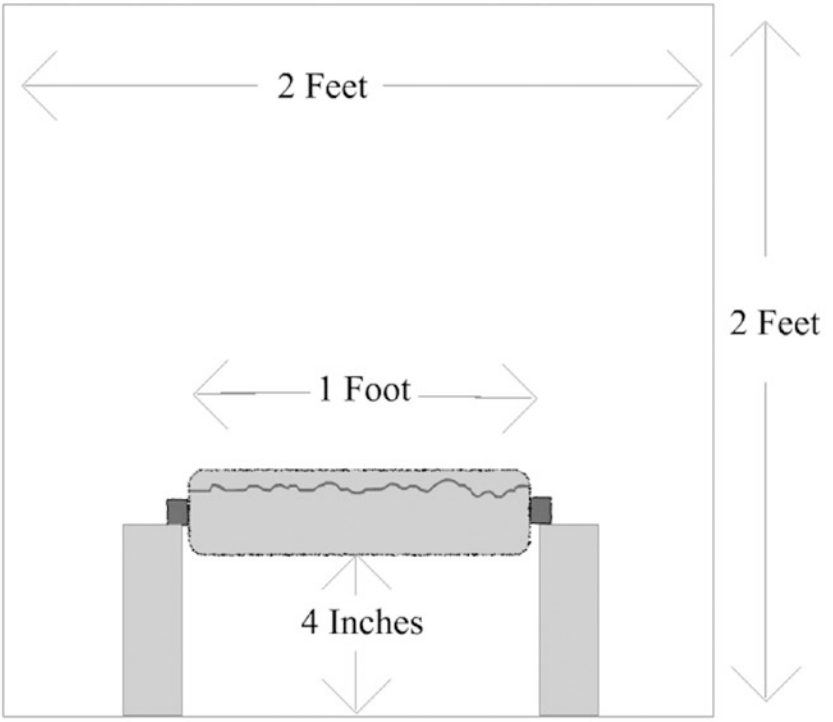


Figure 13-2. *Evan's drawing of the scroll at the end of the tunnel*

"Any idea how much a spider monkey would weigh?" asked Evan.

"No idea, but we'll find out. I guess you can't make your robot too heavy, huh?"

Evan shook his head. "I'll definitely have to keep the weight in mind."

Max smiled and turned off his flashlight. "If I were a monkey, I'd just get in, grab it, and get out," he said. "Fast and simple."

"Well, the motors are fast. But I don't think moving quickly is the right idea. It'd be too easy to make a mistake," said Evan. "As for grabbing the scroll, that's the tricky part, I think."

"Can I make a suggestion?" asked Max.

Evan nodded. "Sure."

"Whatever you build, it doesn't need to be fancy. From what I've seen of your other bots, this one should be fairly basic. Maybe just a simple lifting mechanism to get the scroll off those legs and then bring it back," Max said.

"You're right," said Evan. "But it can't move too fast or it might knock the scroll off the legs. Well, I'm done here."

"Okay. Let's go see your uncle," said Max.

Max's Solution

"I like Max's idea," said Evan, eating his lunch back at the tent. "I'll have to figure out how to build a lifting mechanism to support the scroll, but I definitely think it'll work."

Uncle Phillip turned to Max. "Maybe two arms?" he asked. "One on the left and one on the right?"

Max shook his head. "I was thinking of something even simpler, like a single lifting crane. Just something to get under the scroll, lift it up, and then haul it back," he replied. "But it's really Evan's decision. He's the one who has to build it."

Evan listened to the ideas being discussed. He knew that the scroll was important. The bot had to function properly or the expedition would probably be over.

"Can you give me three or four hours? That should be enough time to develop something," Evan said.

"Evan, you take all the time you need," said Uncle Phillip. "I don't like putting this kind of pressure on you, so you let me know when you're ready. So far, your little bots have worked perfectly. I have no doubt that whatever you design for the tunnel, it will get that scroll. We'll all be in the library taking inventory of all the artifacts, so take your time." Uncle Phillip grabbed Evan's shoulder and squeezed it, then turned and left.

Evan pulled out his Design Journal and began to write.

Story continues in Chapter 17 . . .

CHAPTER 14

GrabberBot: Planning and Design

Taking the suggestion from Chapter 13, this bot doesn't need to be quick but it does need to be simple. All we need is for the bot to move to the end of the tunnel, avoid knocking the scroll off its support legs, grab the scroll securely, and return to the tunnel entrance.

With those requirements in mind, let's move forward and develop a solution to this challenge.

GrabberBot Planning and Design

If you want to skip ahead to Chapter 15 and take a look at my final solution for the GrabberBot, feel free. Right now, I can imagine numerous bots that could be built to retrieve that scroll, and I'm sure you can, too. Get out a blank Design Journal page and a pen—your goal for this challenge is to develop an alternative to my GrabberBot.

Note There are two blank Design Journal pages left in the back of this book (if you used one each for Chapters 2, 6, and 10). If you need more pages, feel free to make photocopies of the Design Journal page or visit the Source Code area of the Apress web site to download the page in PDF format.

In the `Robot Name` box, write **GrabberBot** or create your own name; some other names I considered included SnatchBot and BringItBackBot. After you have your bot name picked out, it's time to think about the bot's description.

The Robot Description

A robot like this one—one that has just two or three jobs to perform—might seem fairly simple to describe. With many bots, the description is so simple that you might wonder if it's even worth spending the time writing it out. I've built plenty of bots without any written description; I just knew what I wanted it to do and I started building.

But don't let this bot fool you. There are some things about it that are a little tricky and deserve some attention. Take a look at my `Robot Description` in Figure 14-1. I might have run into some trouble if I had just started building without considering all the things involved in retrieving the scroll.

M. Bell and J. F. Kelly, *LEGO® MINDSTORMS® EV3*, DOI 10.1007/978-1-4842-2262-1_14

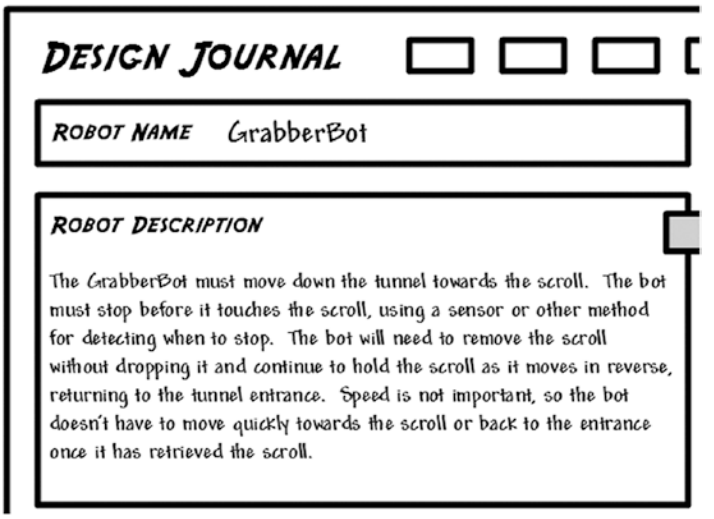


Figure 14-1. *The GrabberBot Robot Description revealed a couple of tricky items*

Where would I have encountered trouble? Well, when I just start building randomly, I sometimes find that I have to tear it apart and start over because I didn't take an external factor into consideration. With this bot, it is likely that I *would have* thought about a method for the robot to detect the end of the tunnel. But because I would have skipped the other parts of the Design Journal page, I *might not have* considered the limitations on the bot and where to place a sensor so that it didn't interfere with its primary task of grabbing the scroll. That's one of the risks of just snapping pieces together without a plan.

By using the Design Journal page and completing all the sections, you reduce the risk of starting to build and then having to start over when you run into a design that doesn't quite work the way it should.

Let's focus on one of my `Robot Description` sentences: "The bot will need to remove the scroll without dropping it and continue to hold the scroll as it moves in reverse, returning to the tunnel entrance." This should get you thinking about what's involved for the bot to "hold" the scroll. Will it hold it like a hand would hold an object, with fingers wrapped around the scroll? Could it somehow simply pinch the scroll and pull it away? If so, how could you make certain the scroll didn't come loose during the return trip? Will it come at the scroll from below or from above?

Your `Robot Description` should force you to start asking your own questions. And it's the answers to these questions that will help you start developing a picture in your mind of what the final bot will look like.

After I completed my `Robot Description`, I realized that my `Task List` was going to be very short, too. But a short `Robot Description` or `Task List` does not mean the design of your bot will be easy. Sometimes the difficulty is in the building of your bot, sometimes in the programming of it, and sometimes in both. I'll finish my Design Journal page before I get too excited about the simplicity of the bot's description.

The Task List

My `Task List` is shown in Figure 14-2. How does it compare to your own?

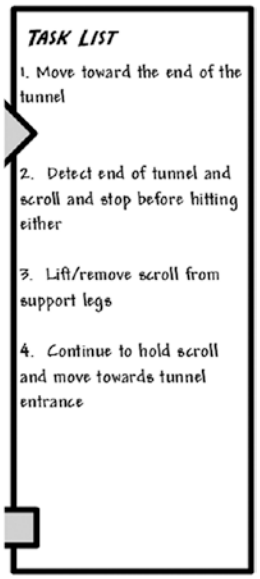


Figure 14-2. *The GrabberBot Task List may be short, but each item is important*

Let's go through each of the four tasks in detail and develop some ideas. These ideas can be used in the `Mindstorm` section, as long as they don't violate any of the limitations or constraints in the next section of your Design Journal page.

The first task is "Move toward the end of the tunnel." Simple enough. If we ignore the shape of our bot for now and just concentrate on this task, what attributes of the robot can we consider? Well, one item is the bot's speed. As it moves down the tunnel, do we want it moving fast or slow or somewhere in between? Personally, I don't want to wait forever, so I might plan on sending the bot down the tunnel at a high rate of speed until it gets close to the end of the tunnel. When it nears the scroll, I'd prefer that it slowly approach it, to avoid knocking it off the support legs.

The best part of moving down the tunnel is that the robot needs to move in a straight line only. No turns and no special zigzagging is needed—this bot will go straight to the scroll and then straight back to me.

The next task, "Detect end of tunnel and scroll and stop before hitting either," is a little more complicated. I could use the Infrared Sensor with Remote and program the bot to stop when I click Stop, but this could be less accurate than using one of the other sensors. I'll probably get more accurate results if I use the Touch Sensor or Infrared Sensor; it really depends on whether I want to actually touch the scroll or wall (Touch Sensor) or detect the proximity of the scroll or wall (Infrared Sensor).

The third task is "Lift/remove scroll from support legs." The scroll is sitting about four inches above the tunnel floor on two small support legs. If I accidentally knock the scroll off, I suppose I could build another bot to go down the tunnel and pick it up. But failure isn't an option for this bot (okay, sometimes it is, but let's think positive). I believe it will be safer to lift the scroll up, coming at it from underneath. If I create something to reach forward (like a hand) I might accidentally push the scroll off the pedestal. I could also create a mechanism that reaches down from above the scroll, but again, there is a slight chance I might cause the scroll to roll forward or backward and fall off the supports.

My final task, "Continue to hold scroll and move toward tunnel entrance," will only work if I've successfully grabbed the scroll and can hold it securely. If my bot has the scroll held securely, I can reverse the direction of the bot and have it move back toward the tunnel entrance. For speed, I'll probably have it move at a medium speed or slower; a fast-moving bot might get to me quicker, but I can afford to let it move slowly if it means not losing the scroll.

By examining my `Task List` in a little more detail, I've been able to start brainstorming (excuse me, *mindstorming*) about the size and speed of my bot and the components to use in it. But before I continue, I need to examine any constraints my bot might encounter.

Limitations and Constraints

Considering that the GrabberBot is heading down another tunnel, I know of at least two limitations for my bot: size and weight. All of the limitations for my bot are shown in Figure 14-3, in my `Limitations/Constraints` box.

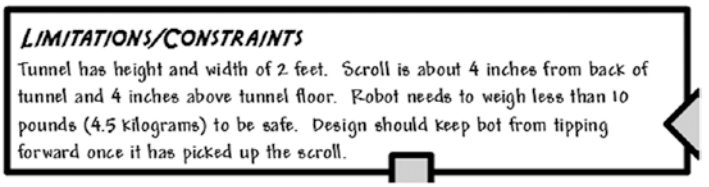


Figure 14-3. *The limitations for the GrabberBot must be taken into consideration*

Obviously, my bot must be no taller than two feet and its width should not exceed two feet, either. The bot's weight, on the other hand, could be an issue.

A quick Internet search revealed that a full grown spider monkey averages about 6 to 10 pounds (2.7 to 4.5 kilograms). If I can keep my bot's weight below that number, I should be okay. The pressure plates shouldn't trigger like they would if a grown person were to crawl down the tunnel.

My final constraint is related to the position of the scroll. The scroll is four inches from the rear of the tunnel and about four inches above the tunnel floor. Whatever method I use to pick up the scroll must take those measurements into consideration.

If I can design my bot so that it doesn't violate any of those limitations, everything should be good.

And now, taking into consideration the `Robot Description`, the short `Task List`, and the `Limitations/Constraints`, are you ready to begin mindstorming?

Mindstorm

I know my `Task List` was short, but the `Mindstorm` section of my Design Journal page is much longer. Take a look at Figure 14-4.

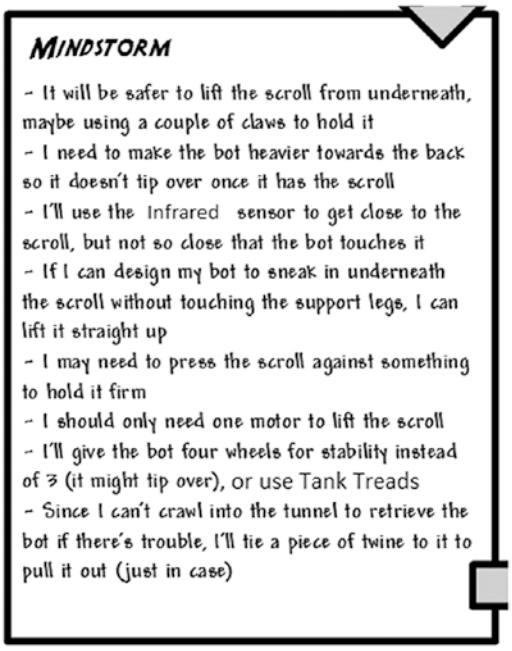


Figure 14-4. *The Mindstorm section for my GrabberBot isn't short at all*

The first `Mindstorm` item, "It will be safer to lift the scroll from underneath, maybe using a couple of claws to hold it," relates to how my bot will approach the scroll. Earlier in the chapter I mentioned that my bot could reach down and grab the scroll, reach forward and grab it, or reach from underneath. I still believe that the safest approach is to somehow get my bot under the scroll and find a way to lift an arm or trap or other mechanism to surround the scroll and lift it directly up off the support legs.

Try this little test: take a couple of tin cans (vegetable soup, for example) and set them about six or seven inches apart. On top of the cans, place something scroll-shaped—maybe a tube of toothpaste or cookie dough or paper towel roll. Now close your eyes. Have someone direct your hand to the "scroll" using *only* these verbal commands: forward, backward, stop, up, down, left, right, and grab—don't cheat! Don't use your fingers unless you plan on designing your bot with fingers (and I doubt you'll have enough motors to do this). Try different approaches to lifting the "scroll" off the tin cans. Which worked better? For me, an open palm coming in under the "scroll" is the easiest. If I avoid touching the tin cans, all I have to do is raise my hand (UP!) and the scroll will sit on my palm.

Another `Mindstorm` item that I think will be very important is, "I'll use the Infrared Sensor to get close to the scroll, but not so close that the bot touches it." Take a look at Figure 14-5. This is how I'm visualizing approaching the scroll.

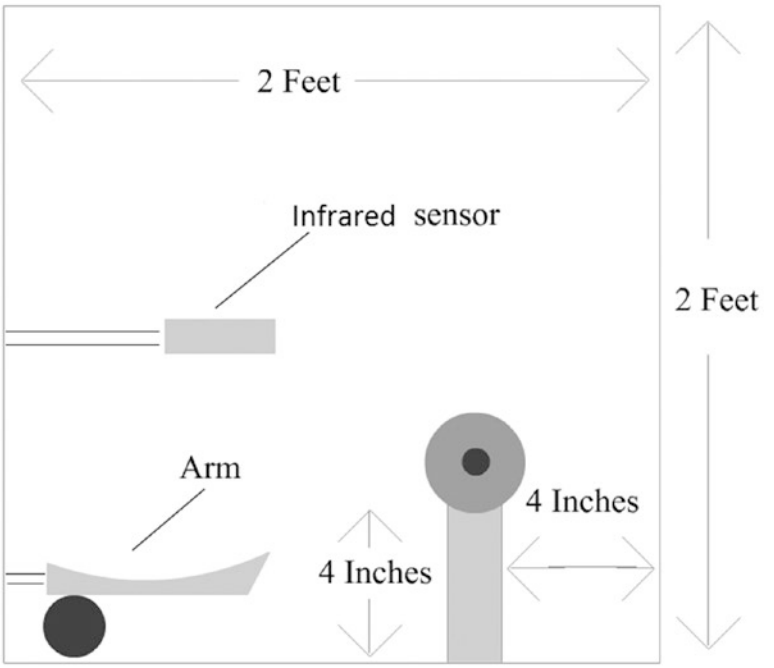


Figure 14-5. *Side view of the bot approaching the scroll*

What I'm considering doing is having the Infrared Sensor above the scroll and some sort of arm mechanism below the scroll. As the bot moves toward the scroll, I'll configure the Infrared Sensor to stop the bot at a certain distance from the wall. That distance will be determined through testing, but the idea is that it will stop the bot when the arm is directly underneath the scroll. The bot stops, the arm lifts up, and the scroll is captured.

Remember, we're just mindstorming right now, so I'm not getting too detailed in my drawing or design just yet. Testing might prove later that this method doesn't work—that's a risk, but that's why I'm spending time thinking about this bot before I start building.

The last `Mindstorm` item I want to cover is "I may need to press the scroll against something to hold it firm." If I can get the bot to successfully lift the scroll, there is a chance that when the bot begins to move in reverse, the scroll might roll off the arm. What I'd like to do (and I'll test this) is have the arm lift the scroll up against some beams or other EV3 components—this should hold the scroll firmly while the bot moves. Again, I'll have to test this and it might not work.

The rest of my `Mindstorm` items are fairly self-explanatory. A four-wheel bot will be more stable than a three-wheel bot; maybe a tank-style bot is more stable still; a bot that's heavy in the rear won't tip forward when it picks up the scroll; one motor should provide sufficient lifting power; and, something that occurred to me after I designed the SnapShotBot, tying a string to my bot will allow me to pull it out if the bot gets into trouble. If the scroll falls off the support legs, I don't know what will happen, but the best-case scenario for that happening would be to simply build another bot that goes down the tunnel and retrieves the fallen scroll. But let's not let that happen, okay?

Well, we're done with the `Mindstorm` section of our Design Journal, and there's only one section left before we begin to build.

Sketches

As you saw in Chapter 10 in Figure 10-11, I don't use detailed images of things like sensors and wheels. I don't like to spend time drawing components in detail when I can use a rough sketch to get my point across. I do the same thing when sketching my ideas for the shape of my bots. While completing my Design Journal page, I began to build some rough pictures in my mind of what I want my bot to look like. So take a look at Figure 14-6 and remember not to laugh too loud or you'll disturb your neighbors.

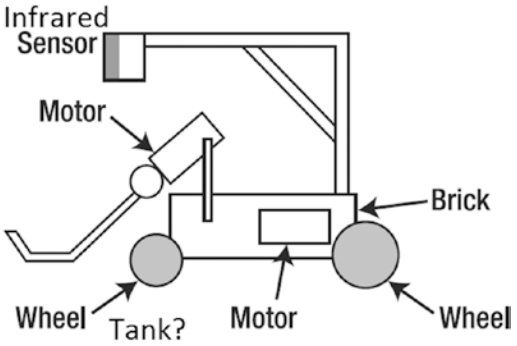


Figure 14-6. *Once again, basic shapes are used to represent parts of my GrabberBot*

Again, I'm not drawing every sensor, wheel, beam, and connector piece. I just want to use basic shapes to represent those items. The Brick is easy—it's a rectangle. Wheels are circles. Motors are still weird looking, but I label them so I remember what they are.

When I'm done, I've got a good idea of where to start building. And that's exactly what I'm ready to do. Let's move on to Chapter 15 to build the GrabberBot so it can retrieve that scroll.

CHAPTER 15

GrabberBot: Build It

The final design of my GrabberBot has a lot of potential modifications. I realized after I completed the design that if I removed the sensors and the Grabber assembly, I would be left with a nice base unit that could be used in future designs. But that's not what this chapter is about. This chapter gives you the building instructions for constructing a bot that can move down the tunnel and successfully retrieve the scroll.

If you've built your own version of the GrabberBot, congratulations. How does it compare to my design (see Figure 15-1)? I hope you're beginning to see that there are an unlimited number of designs for successful completion of these challenges. The only limits are your imagination and the parts in your kit.

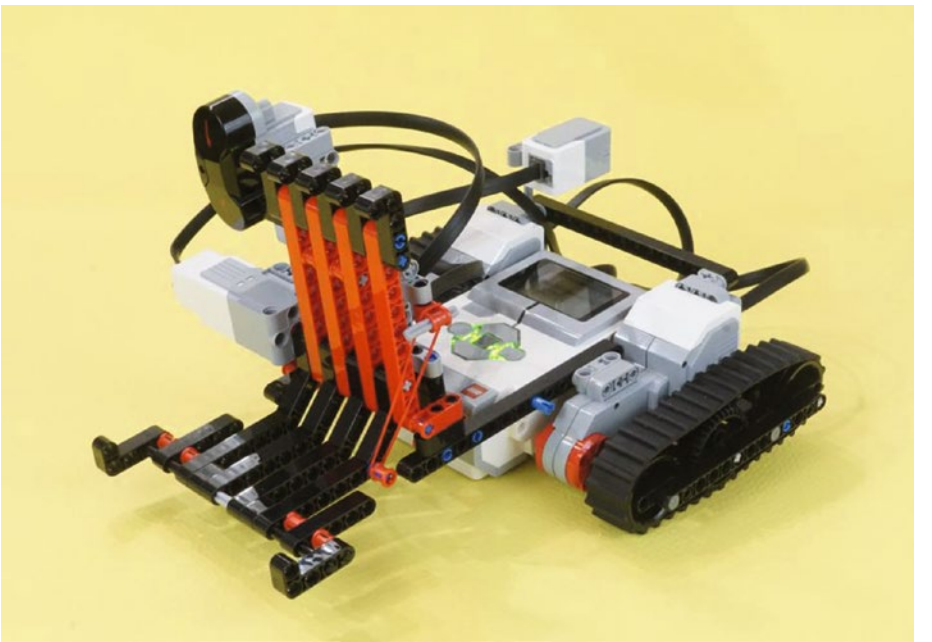


Figure 15-1. *One version of the GrabberBot*

Note E-mail me a picture of your version of the GrabberBot—I'd love to see it. You can find my web address in the Introduction at the start of the book.

M. Bell and J. F. Kelly, *LEGO® MINDSTORMS® EV3*, DOI 10.1007/978-1-4842-2262-1_15

Now, on to building the GrabberBot.

The GrabberBot's building instructions are divided into three sections. The first section consists of the main body (Brick, two large motors, and tank-treads). The second section adds the Grabber assembly with Medium Motor, and the third section adds the Touch Sensor, Infrared Sensor, and wiring. After you've built the GrabberBot, Chapter 16 provides you with the programming instructions.

Just like previous building instruction chapters, comments are provided for sections that might be a little tricky.

First Section: Main Body Tank-Treads

Figures 15-2 through 15-19 walk you through the steps for constructing the main body. Start with the large motor, tank-tread, and components you see in Figure 15-2. This picture gives you the parts for one entire motor and tread. You will build two of these, which will be identical mirror images.

Figure 15-2. *All of the components for one of the main body's tank-tread motor assemblies. You'll build two of these. Nail head axles are 8-hole. Gray axles are 5-hole.*

Follow these construction steps through Figure 15-9. You will have one completed tank-tread assembly. Then return to Figure 15-2 and gather the second set of parts just as you did for the first tank-tread assembly. Work up through Figure 15-9 again and you'll have both tank-tread assemblies. Then you'll assemble them to the motors, ending up with two mirror image tank-tread motor assemblies.

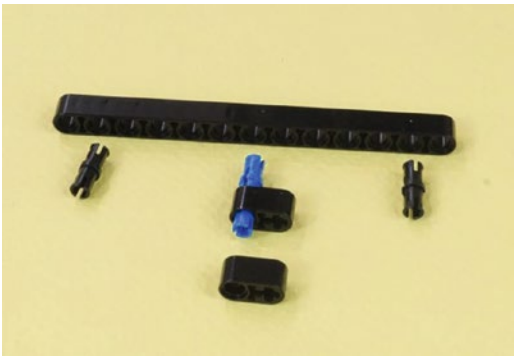


Figure 15-3. *Two short pins and one long pin are connected, as shown in Figure 15-4*

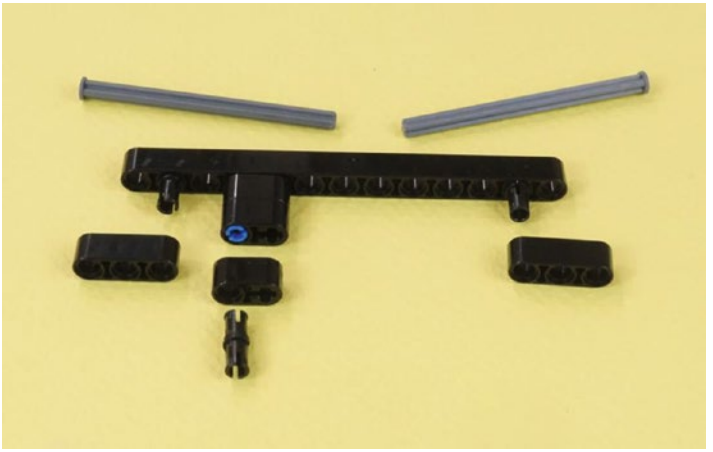


Figure 15-4. *Insert the two nailhead axles as shown in Figure 15-5. The left axle goes through the axle-holes on the short black connectors in the middle, making a strong assembly.*

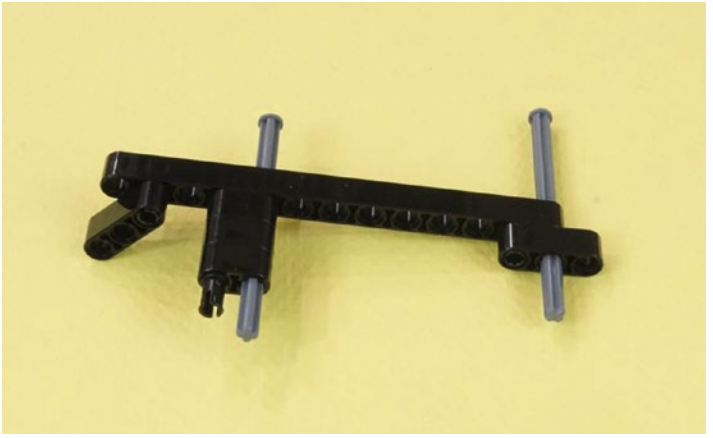


Figure 15-5. *The loose parts from Figure 15-4 all have a home now*

Figure 15-6. *Push the long axles all the way in and set the unit on its side. In this picture, the unit was turned around the other way from Figure 15-5. The hub, short axle, and yellow collar join the scene.*

Figure 15-7. *The hub and its axle are in position. Now the gear-wheel assembly and second hub with yellow collar arrive. There is a red collar between the two gears.*

Figure 15-8. *The second hub and the gear-wheel assembly are in place. The remaining 13-hole beam is ready to be mounted. The single black pin in the center will keep the beam from falling off. Mount the beam so all three axles go through. Then stretch the tank-tread over the end hubs.*

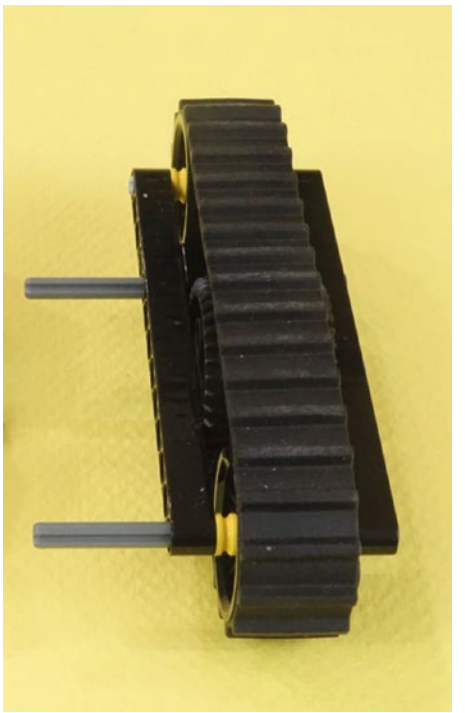


Figure 15-9. *A completed tank-tread assembly ready to be attached to its motor. Build two of these.*

When you get here with your first tank-tread motor assembly, set it aside and return to Figure 15-2. Gather the second set of components—the motor, tank-tread, wheels, and all of the others. Then build the second tank-tread motor assembly. When you have both assemblies built, fasten them to the motors using the parts shown in Figures 15-10 and 15-11.

You'll make two of these motor assemblies. Build the first one as shown, then make the second one the same way but in mirror image. You can see what both tank-tread motor assemblies look like in Figure 15-14.

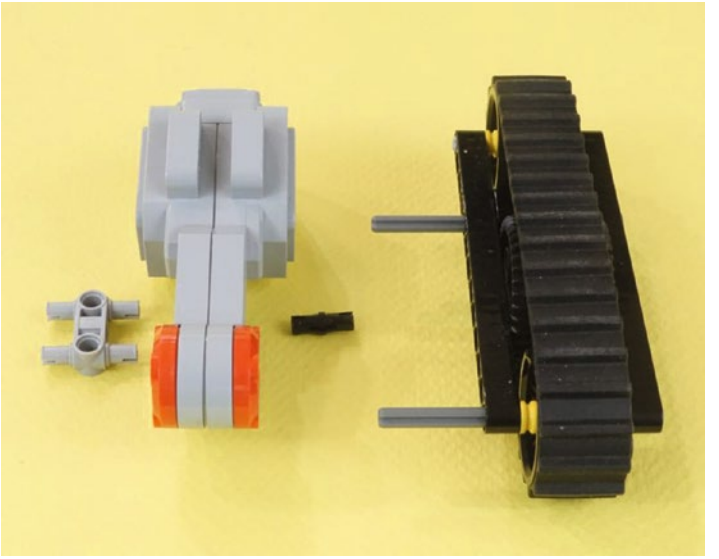


Figure 15-10. *These parts will give you a complete tank-tread motor assembly*

Figure 15-11. *Insert the lower axle into the motor and the upper axle into the axle socket in the motor mount. Connect the double gray component to the other side. The second motor will be a mirror image of this.*

Now it's time for some new parts. Gather the Brick and the parts shown in Figure 15-12. The two long 15-hole beams receive four pins each. In Figure 15-12, one of the beams has the short and long pins already inserted in the correct places. Fasten the pins into both long beams. Then connect the beams to the Brick, as shown in Figure 15-13.

Figure 15-12. *New parts. Two 15-hole beams with four pins apiece.*

In Figure 15-13, we see the right beam about to be attached to the Brick. It is shown partially inserted to give a clear picture of the connectors and how they attach to the Brick. The left beam is attached to the Brick in the same manner. This symmetry is a useful design concept, especially for driving bases. Also, note that there are four pins on each side of the Brick. When fastening to the Brick, you want to use as many pins as you can since it is the heaviest component and has the greatest need for reinforcement.

Figure 15-13. *The two 15-hole beams being attached to the Brick*

Now that you've completed the left and right wheel assemblies, it's time to connect them to the Brick (see Figures 15-14 and 15-15).

Figure 15-14. *The Brick and tank-tread motor assemblies give us the main body. One is connected, the other is ready to go.*

Figure 15-15. *Both tank-tread motor assemblies in position*

Now turn the Brick around. Next, you'll connect a pair of 15-hole reinforcing beams to the backs of the motors (see Figures 15-16 and 15-17). You can see that the motors are not very securely attached right now—in Figure 15-16 they're wobbly and spread apart a little to the sides! These two long rear beams will fix that. Later on, they will come in handy as *hardpoints* to fasten other parts to.

Figure 15-16. *New parts—four dual-connectors and two 15-hole beams. Assemble them as shown in Figure 15-17.*

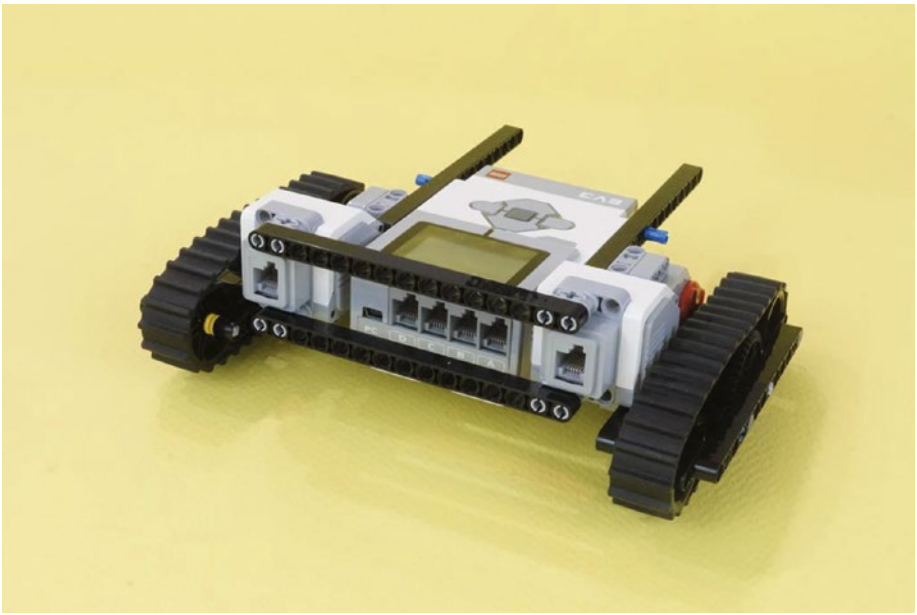


Figure 15-17. *The reinforcing beams are in place, and the wobble problem is solved. None of the long beams block any wire holes, and the USB connection is also accessible.*

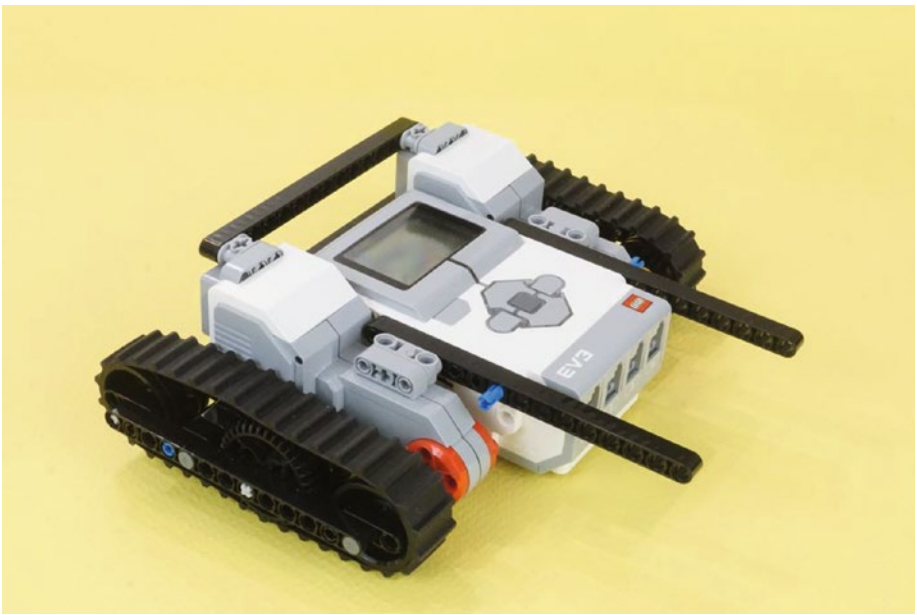


Figure 15-18. *Another view of the tank driving base. This is an excellent driving base for other projects. It rides rather low, but is very stable.*

Figure 15-19. *And one more view of the tank driving base. The two beams extending forward are excellent hardpoints for any project. The protruding blue pins are there just in case they will be useful as attachments.*

That's all for the main body. Up next is the scroll grabbing mechanism.

Second Section: Grabber Assembly (Lifting Arm Mechanism)

Figures 15-20 through 15-45 show you how to build the Grabber assembly and connect it to the main body. You will build a Grabber Back Wall and a Grabber Jaw, with the medium motor to actuate the jaw. Start with the components shown in Figure 15-20.

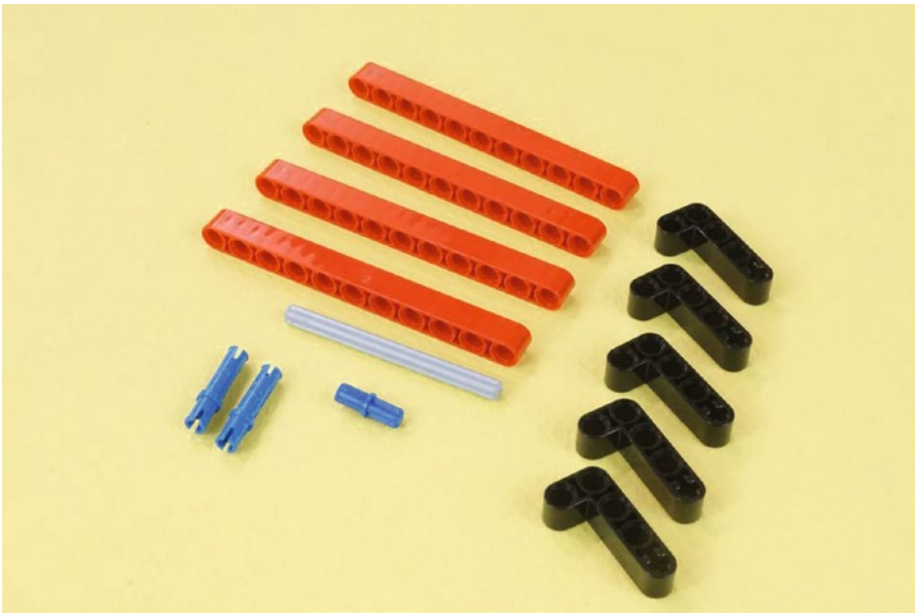


Figure 15-20. *Now to build the Grabber assembly, which will go in front of the Tank Driving Base. We'll start with these 13 parts to make the Grabber Back Wall.*

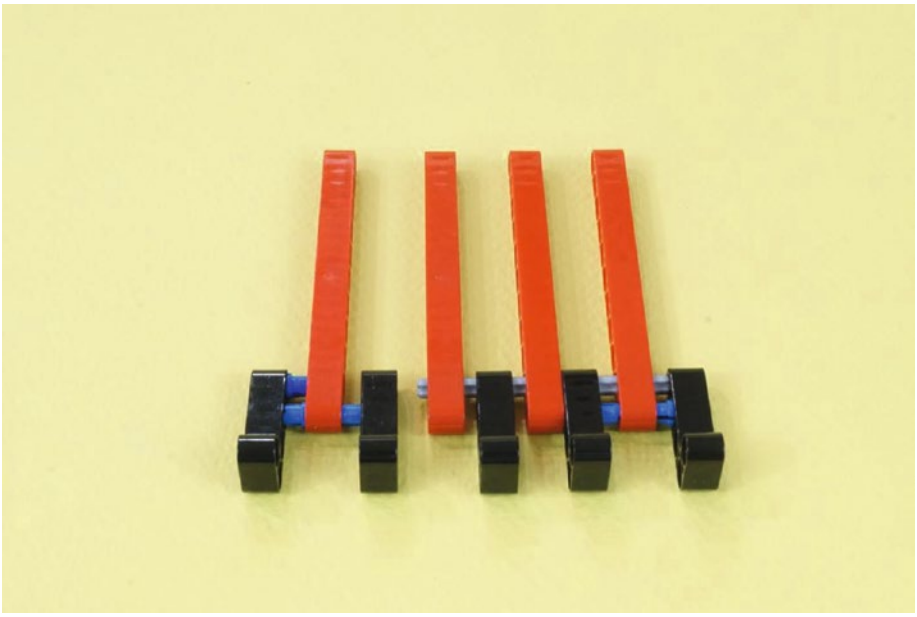


Figure 15-21. *A partially assembled view of the Back Wall. Notice that the short blue axle goes in the L-shaped piece on the left, along with the blue pin. The long axle holds the three right-side red beams together. Press the assembly together to finish, as shown in Figure 15-22.*

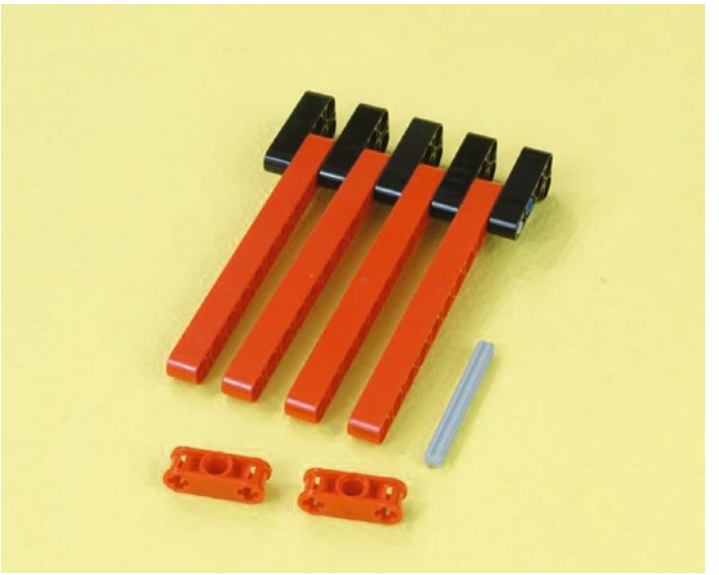


Figure 15-22. *The Back Wall is turned over. Add these three new parts. The 5-hole axle will hold the two red connectors as shown in Figure 15-23. Press the axle in sideways, in the fourth hole from the top.*

Figure 15-23. *Two new parts, a 7-hole axle and a pin connector with a center hole. Assemble as shown in Figure 15-24.*

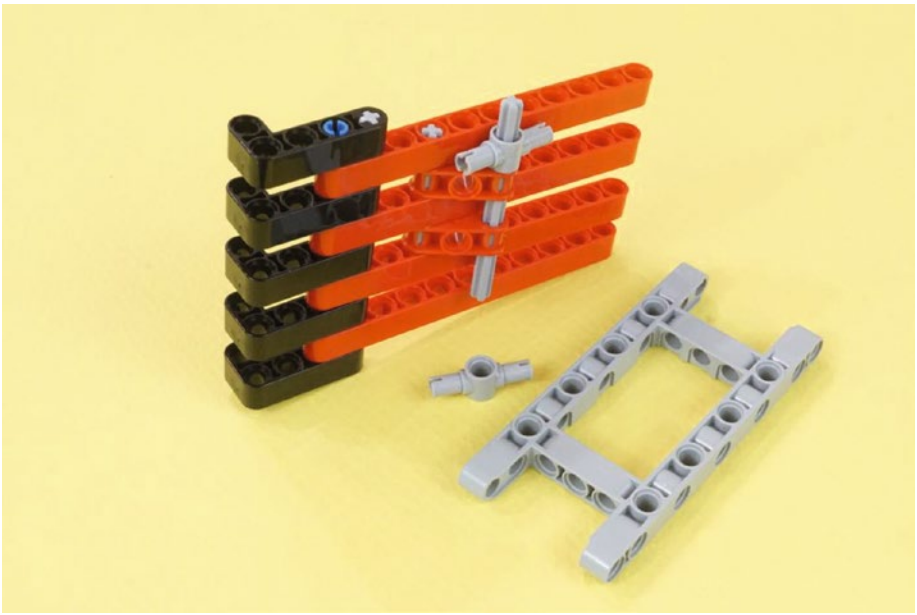


Figure 15-24. *New parts: A second pin connector with center hole to be inserted into the frame piece, as seen in Figure 15-25*

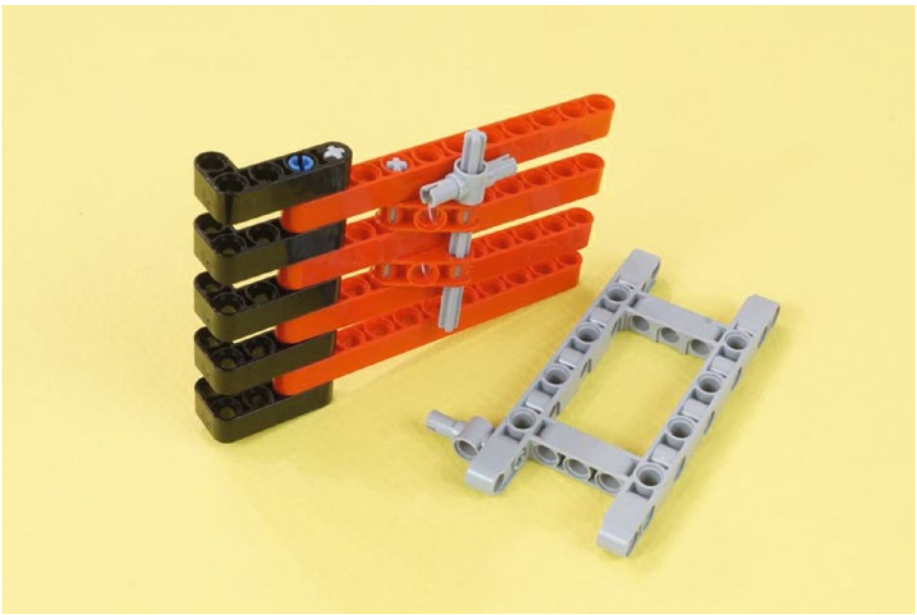


Figure 15-25. *Frame piece with connector, ready to attach, as shown in Figure 15-26*

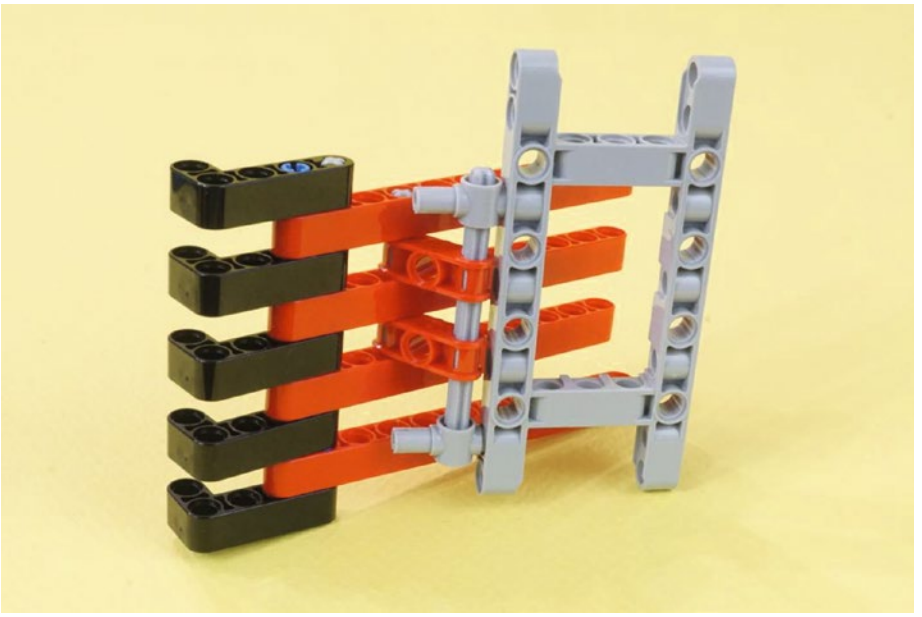


Figure 15-26. *This frame piece makes a strong foundation for the Grabber Back Wall. Fasten to red beam assembly as shown. We'll return to this assembly in Figure 15-29.*

Figure 15-27. *These twelve new pieces will make a lower frame piece for the Grabber Back Wall*

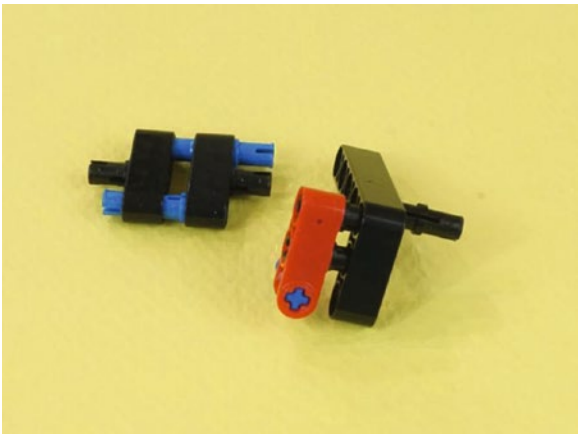


Figure 15-28. *This exploded view shows how the parts go together. The short blue axle connector goes in to the other side of the red connector. They'll end up as shown in Figure 15-29.*

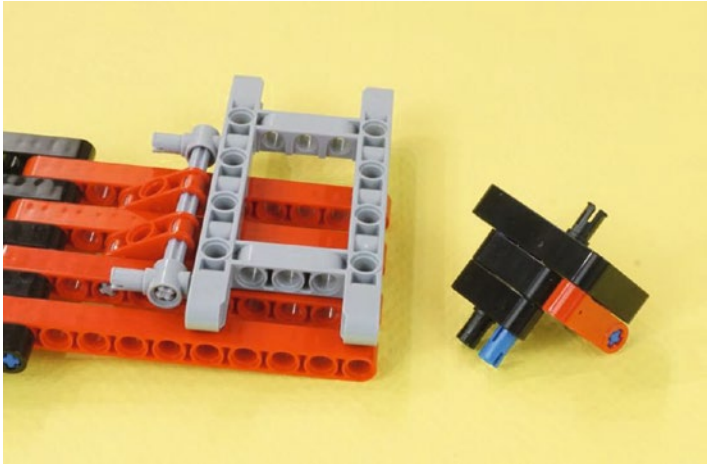


Figure 15-29. *The lower frame piece is done. Connect it to the gray rectangular piece, as shown in Figure 15-30.*

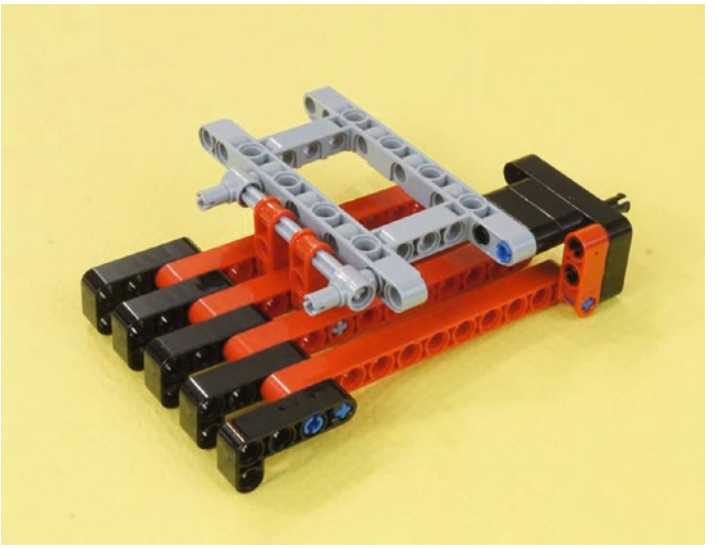


Figure 15-30. *That short blue axle mentioned in Figure 15-28 connects to the bottom of the outermost red 11-hole beam. Now the Grabber Back Wall is a good solid unit.*

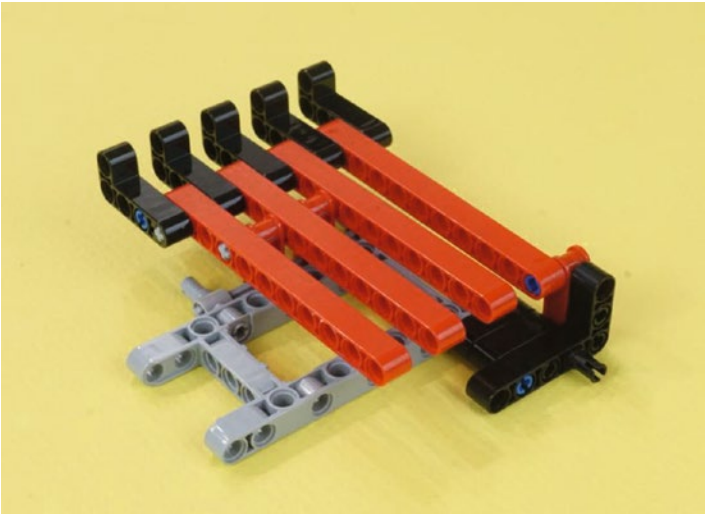


Figure 15-31. *Another view of the Grabber Back Wall, turned over*

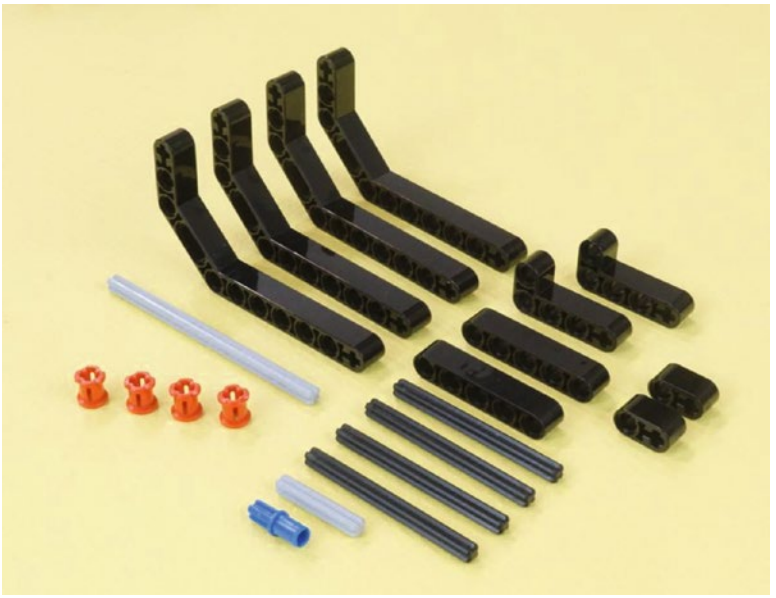


Figure 15-32. *New parts. These will become the Grabber Jaw, which will fasten on to the Grabber Back Wall.*

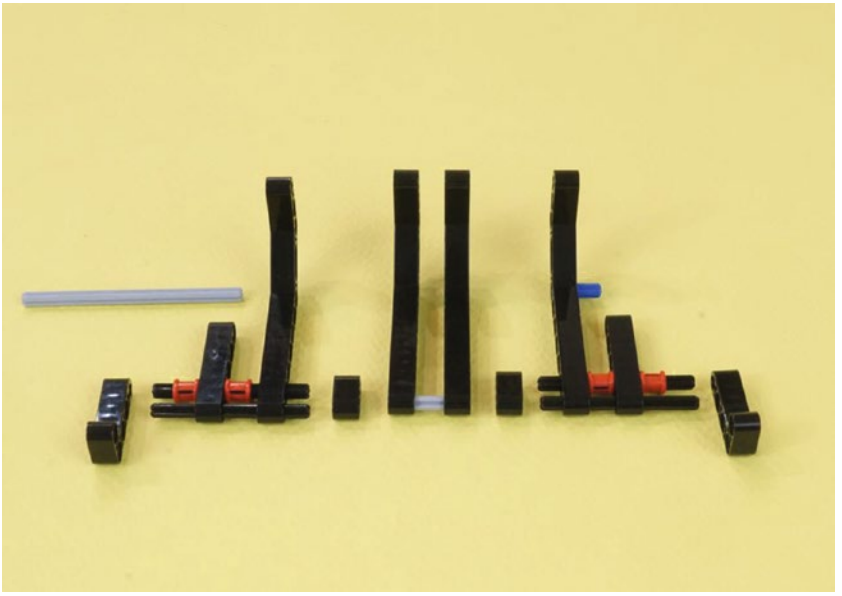


Figure 15-33. *This exploded view shows all of the parts from Figure 15-32. The 9-hole gray axle will be mounted in the next figure. 6-hole black axles.*

Press these components together as shown in Figure 15-34. Insert the 9-hole axle into the leftmost Grabber Jaw piece to prepare for assembly of the jaw onto the Grabber Back Wall.

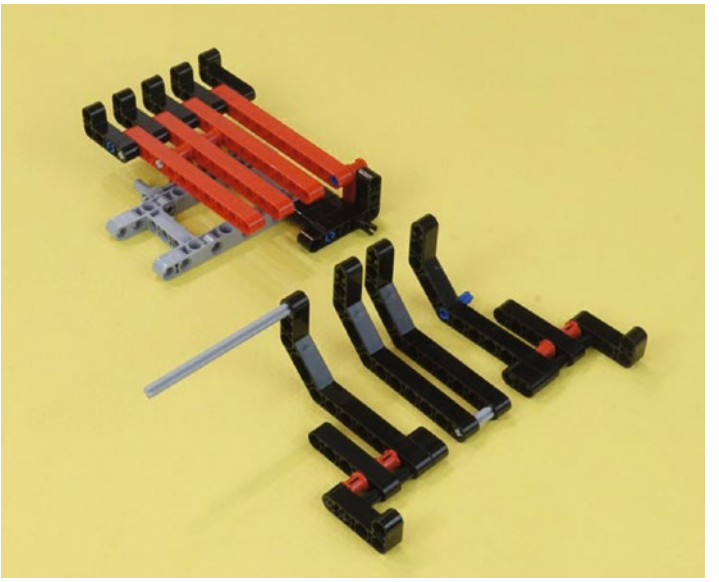


Figure 15-34. *Final components for the GrabberBot. The three parts of the Grabber Arm will be held by that long gray axle. Those three parts will be pressed together in Figure 15-35.*

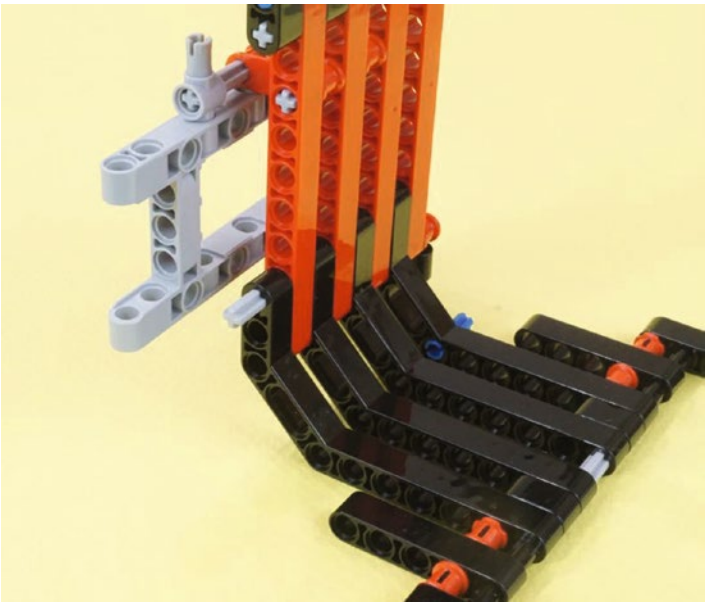


Figure 15-35. *Hinging the Grabber Jaw to the Grabber Back Wall. The long axle becomes a sturdy hinge. It will connect to the medium motor, so you need the end of the axle to protrude as shown.*

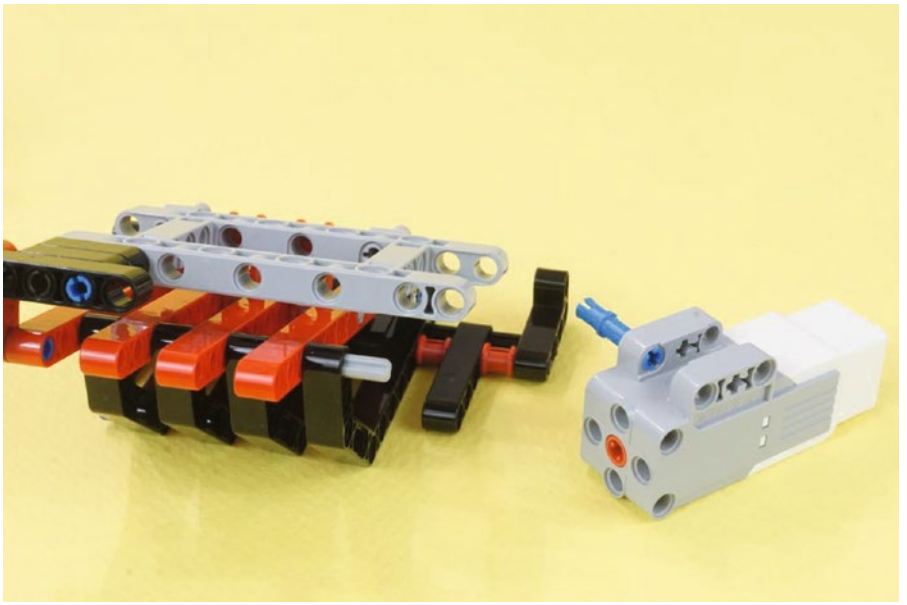


Figure 15-36. *It's time for the Medium Motor. Partially insert a blue connecting pin as shown. You'll push it in all the way in Figure 15-38.*

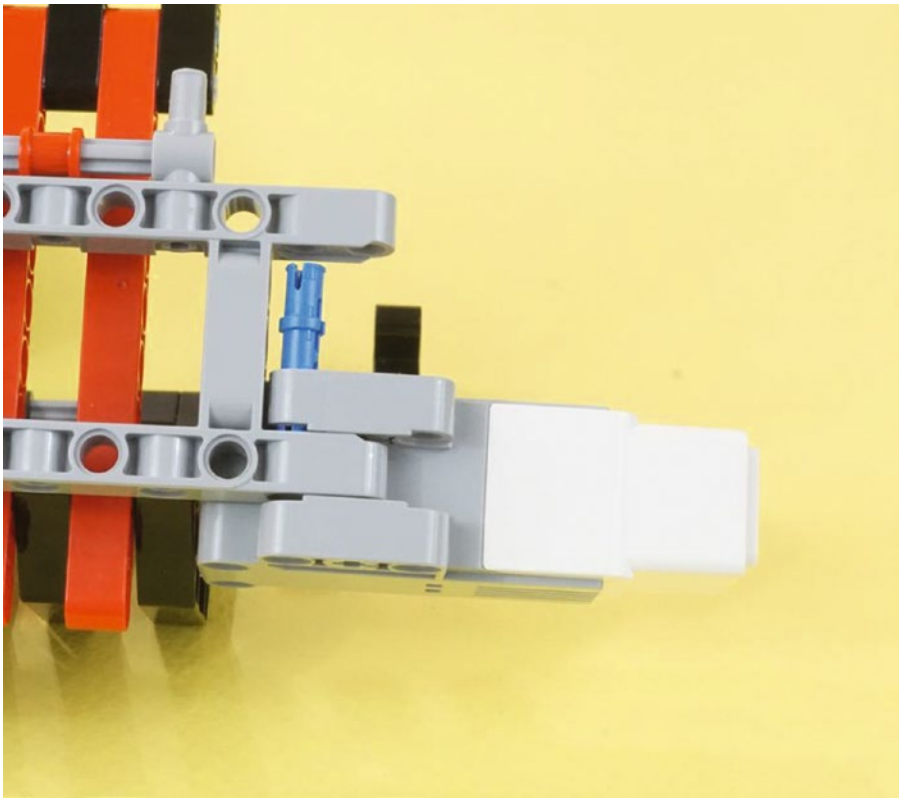


Figure 15-37. *Press the Medium Motor on to the protruding axle. Then push the blue connecting pin down, as shown in Figure 15-38.*

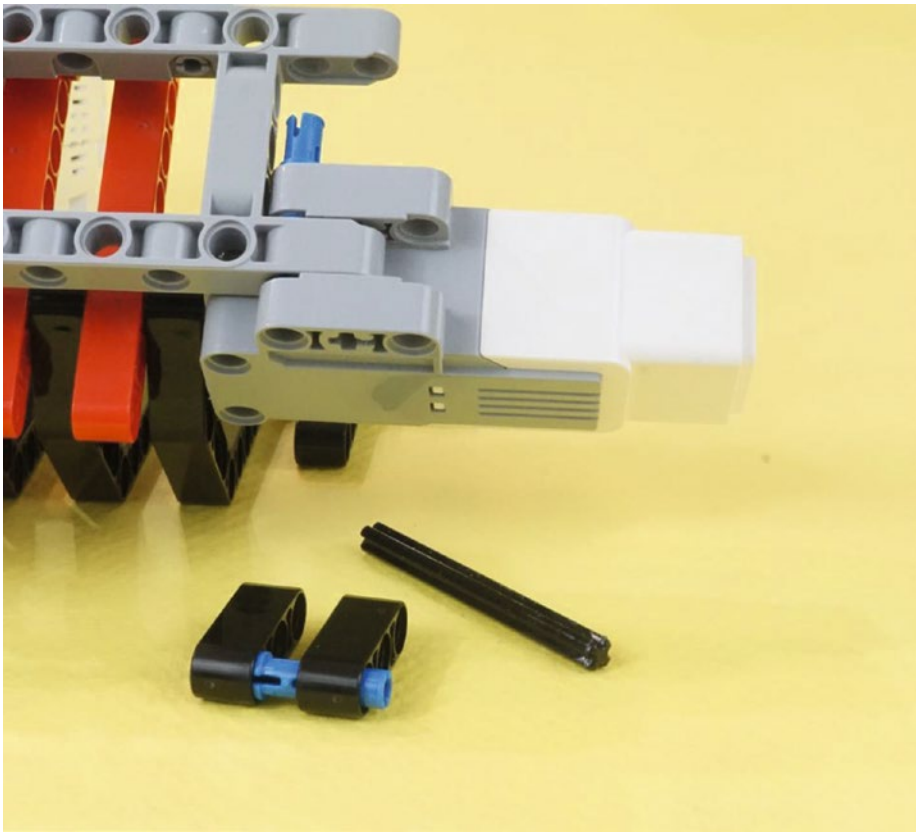


Figure 15-38. *Four new parts—an axle, long blue pin, and a pair of three-hole beams. The beams will be pressed together, and the unit is attached to the Medium Motor.*

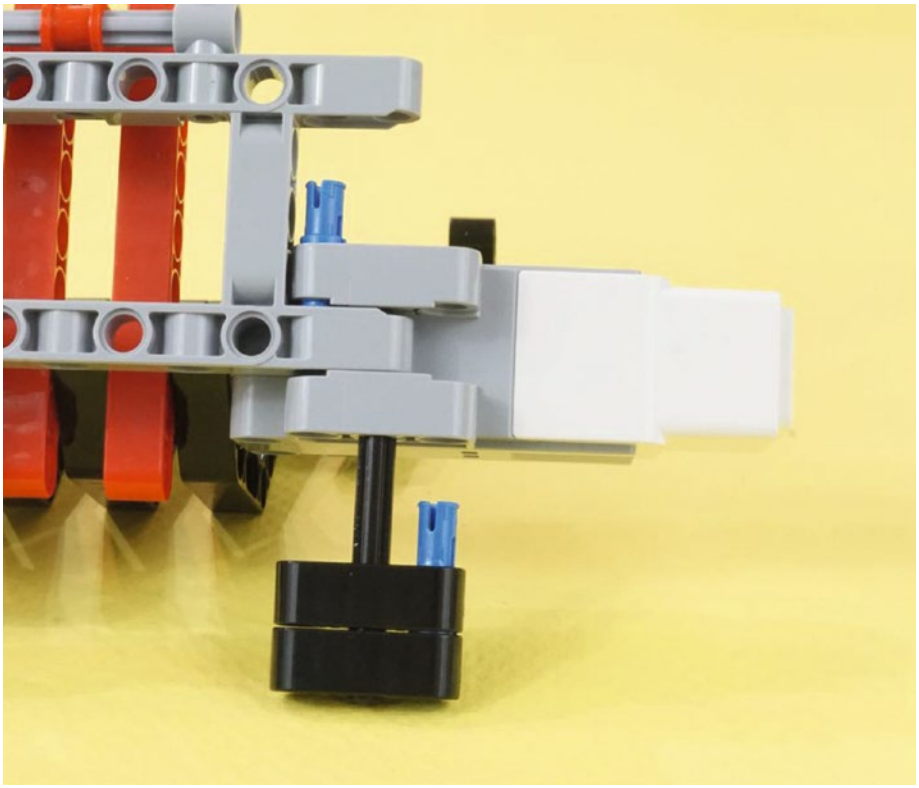


Figure 15-39. *The two beams and connectors being pressed into the Medium Motor mounting point*

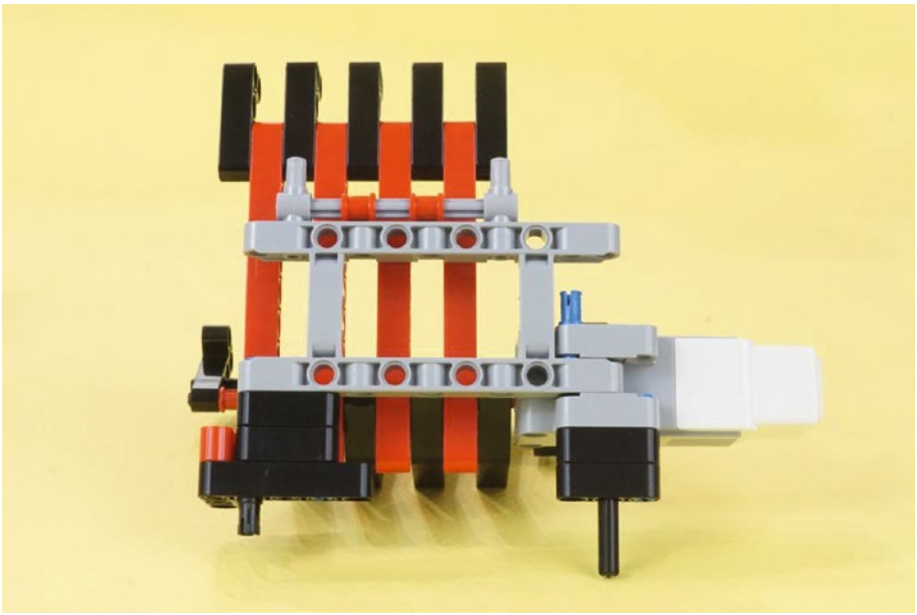


Figure 15-40. *The completed Grabber assembly. Back Wall, Grabber Jaw, and Medium Motor ready to mount to the Tank Driving Base.*

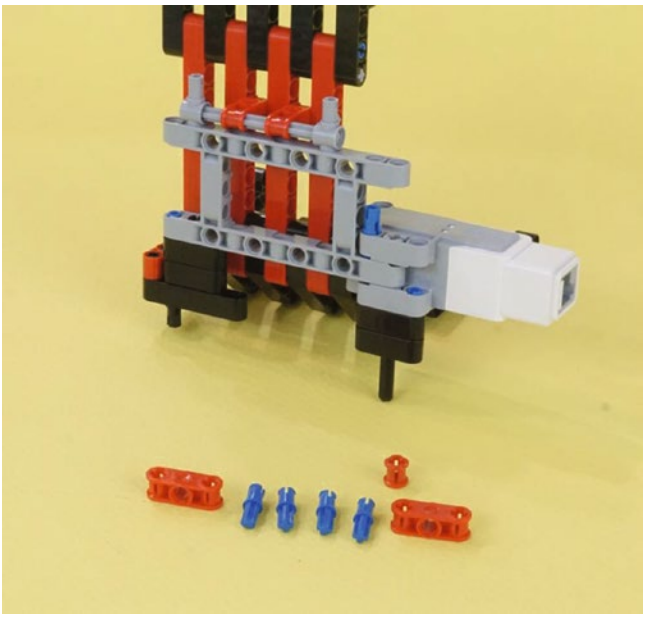


Figure 15-41. *These seven new parts will mount the Grabber assembly to the Tank Driving Base*

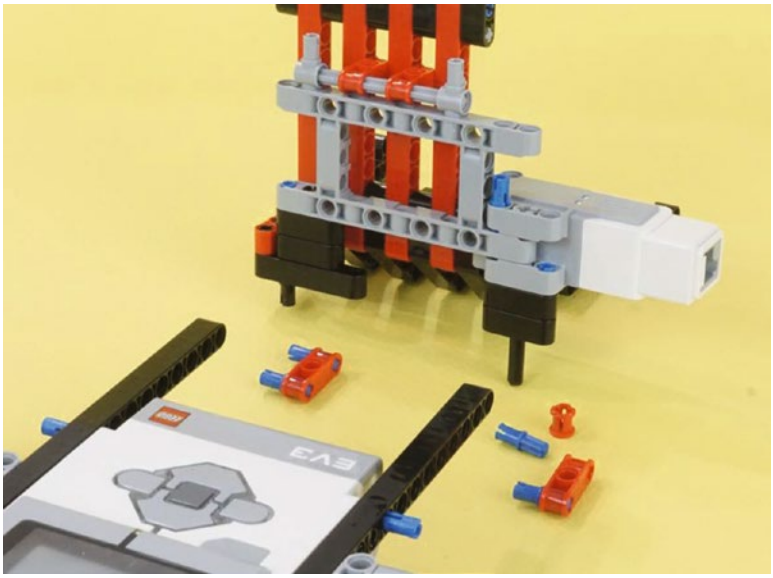


Figure 15-42. *Insert the short blue axles into the red pieces. Then attach them to the driving base beams as shown in Figure 15-43.*

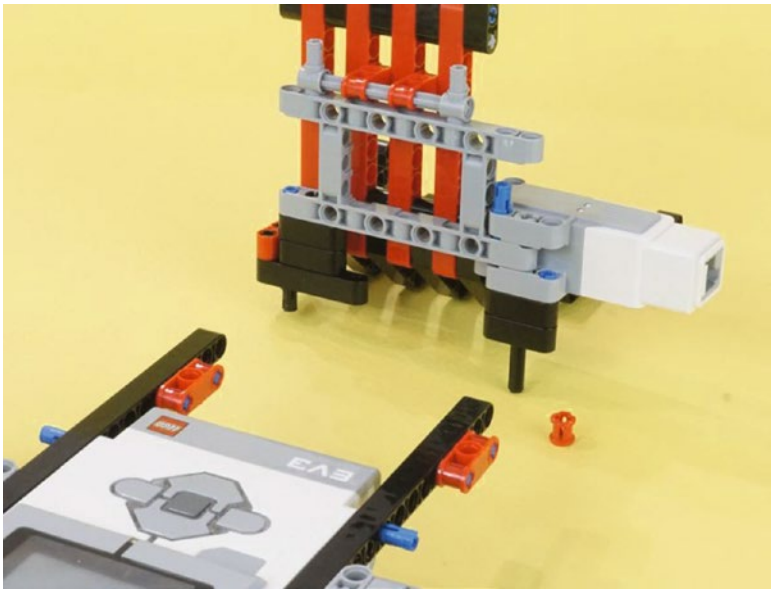


Figure 15-43. *The red pieces attached to the driving base beams. The red collar will go on the bottom of the black axle, as shown in Figure 15-44.*

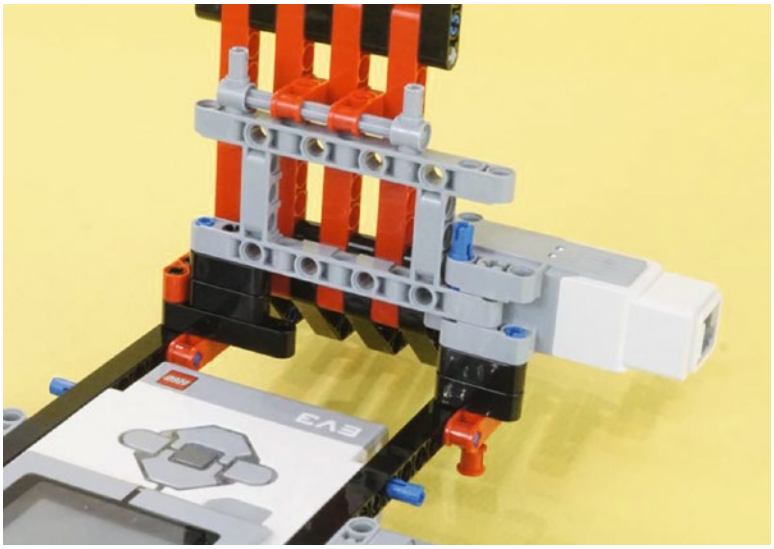


Figure 15-44. *Set the Grabber Jaw assembly on to the two red connectors. Then fasten the red collar to the bottom of the axle on the right-side beam.*

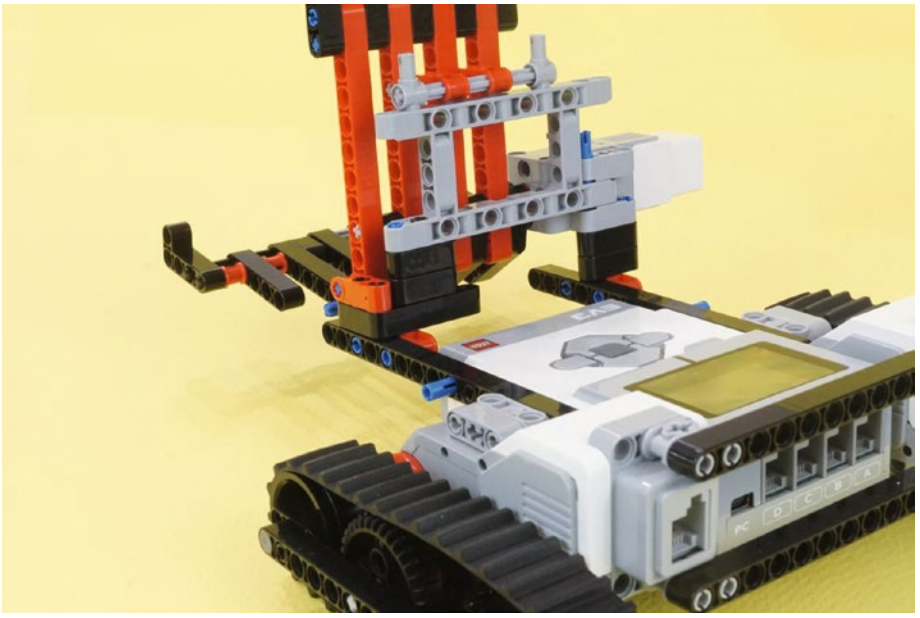


Figure 15-45. *The Grabber Jaw assembly viewed from the other side*

Now you have the basis of the GrabberBot. All that's left is to place the Infrared Sensor, Touch Sensor, and a few more parts.

Third Section: Infrared Sensor, Touch Sensor, and Wiring

You're almost done. The GrabberBot needs the Infrared Sensor to detect the scroll. I decided to use the Touch Sensor again as a Start button. And, there's the question of how to hold the jaws shut when they close on the scroll. Figures 15-46 through 15-56 show you how to complete the bot.

Figure 15-46. *All of the remaining parts (axles are 3 and 5-hole) needed to mount the Touch and Infrared Sensors. There is also a Grabber Latch assembly, discussed next. These parts will take you all the way through Figure 15-52.*

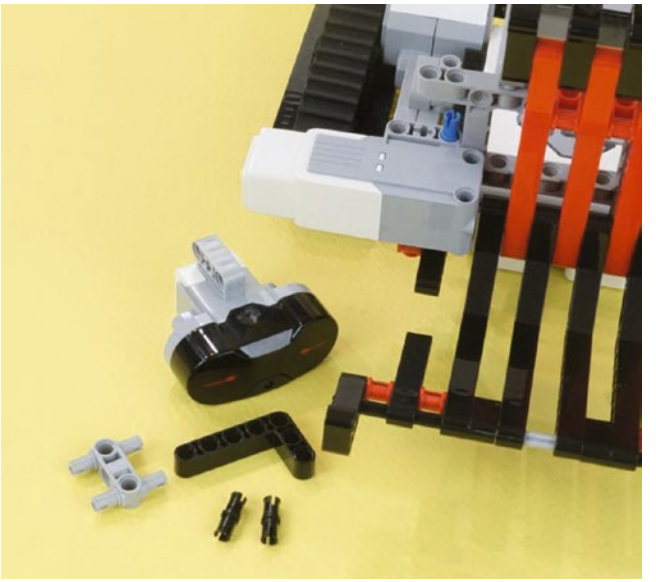


Figure 15-47. *These parts become the Infrared Sensor mount*

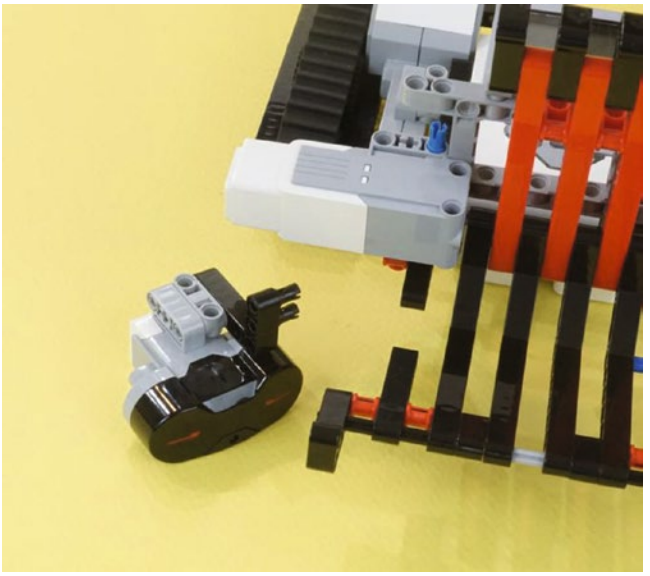


Figure 15-48. *The Infrared Sensor with its L-shaped mounting bracket attached*

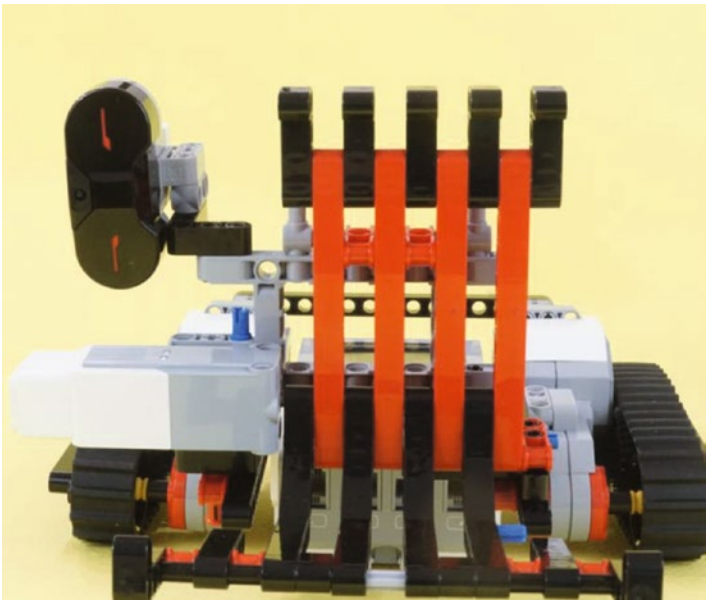


Figure 15-49. *The sensor mounts sideways on the Grabber Back Wall. It will measure the distance to the scroll just fine, even though it is sideways.*

Figure 15-50. *The Touch Sensor axle goes in the center Touch Sensor axle-hole. It mounts to the Tank Drive Base, as shown in Figure 15-51.*

Figure 15-51. *A convenient place to mount the Touch Sensor is in the double connector holding the 15-hole beam in place. This single point of attachment is sturdy enough to let the Touch Sensor be used as a Start switch.*

Engineering: How to Hold the Jaw Shut? An Elastic Grabber Latch

These last five parts in Figure 15-52 have an interesting purpose. When I tested this bot with scrolls, the Grabber Jaw would get a grip just fine, but then it would relax a bit after the medium motor was switched off. The scroll could come loose on the trip back. It would be nice to have a way to latch the Grabber Jaw so it would not let the scroll wiggle loose on a bumpy ride back.

These parts—the elastic band in particular—give us a Grabber Latch assembly. Even after the Medium Motor is turned off, our Grabber Latch will hold the Grabber Jaw on the scroll. So let's get the latch installed. Figure 15-52 shows the new Grabber Latch parts. Three of them are already assembled. That is a black pin attached to the top of the red side connector.

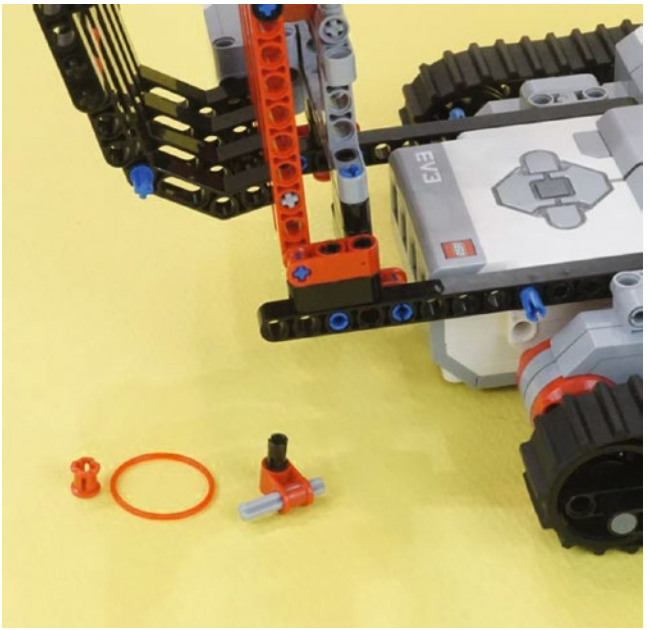


Figure 15-52. *The last five parts from Figure 15-46. They make a Grabber Latch with the red rubber band.*

Figure 15-53 shows the jaw in the closed position. When the jaw closes on the actual scroll, this rubber band will help hold it in place. That is, the band is functioning as a latch. Now look at how the rubber band is stretched in Figure 15-54. When the jaw is open, the force of the rubber band pulls against the axle itself, rather than trying to close the jaw. But as the medium motor begins to close the jaw, the force of the rubber band begins to add itself to the motor's closing force. The scroll will be securely held.

In engineering terms, there is energy stored in the elastic band when the jaw is opened. But it will not close the jaw by itself as long as it is fully opened. When the motor begins to move the jaw, this stored energy pulls the jaw shut and keeps it that way even though gravity might pull the jaw open on its own.

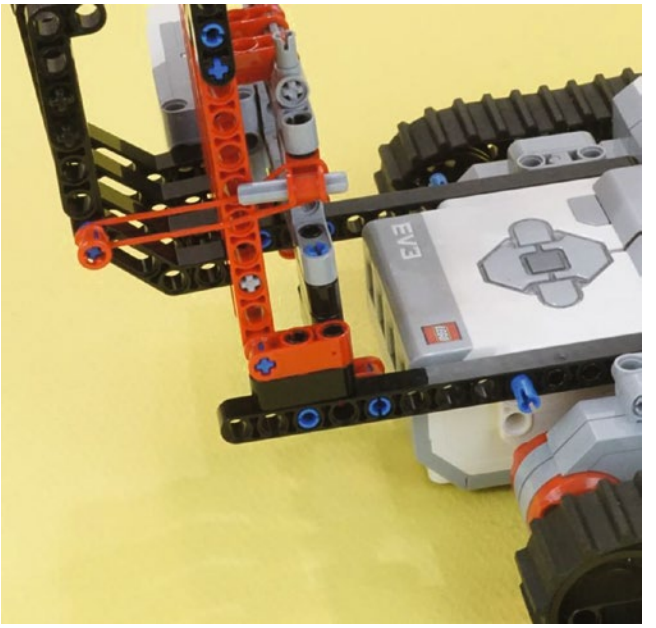


Figure 15-53. *Mount the rubber band holders as shown. Stretch the band between the connectors. This view shows the red rubber band holding the Grabber Jaw closed.*

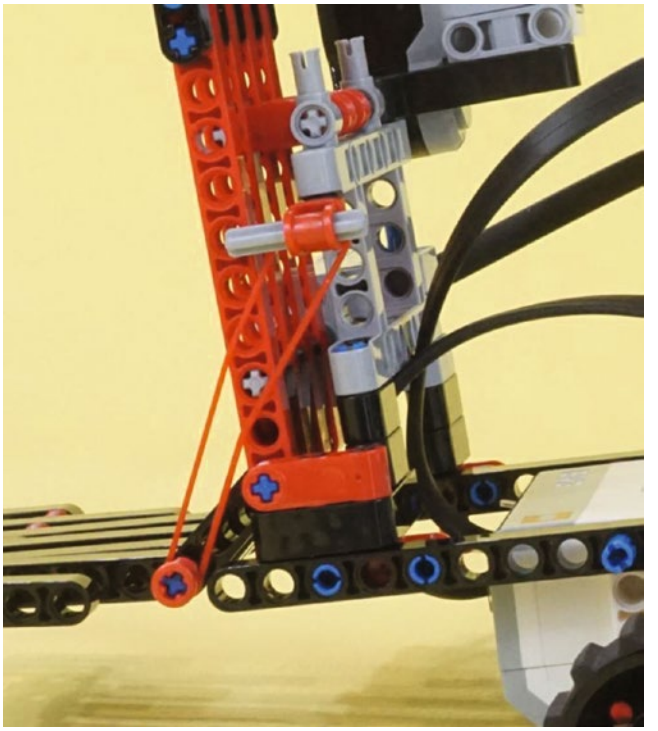


Figure 15-54. *How the band looks when the Grabber Jaw is opened. This also gives a close-up of how the upper rubber band holder is installed into the corner of the large gray frame.*

Now your GrabberBot needs to be wired up and programmed. Take a look at Figures 15-55 and 15-56 to see how I've connected the wires.

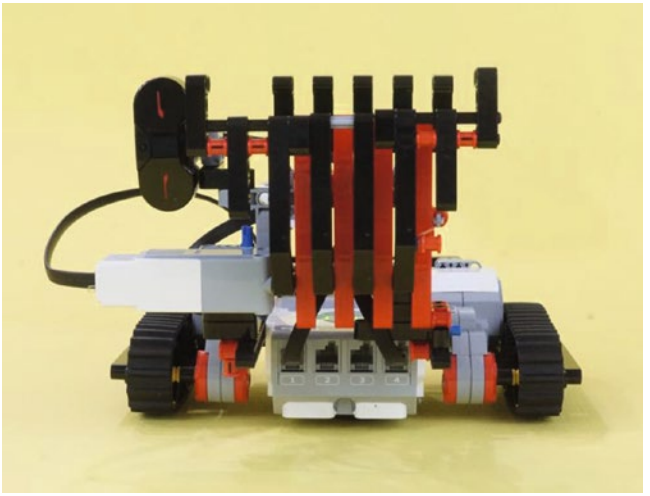


Figure 15-55. *The wired-up and ready-to-be-programmed GrabberBot. Sensor Port 1 goes to the IR Sensor. Sensor Port 4 goes to the Touch Switch.*

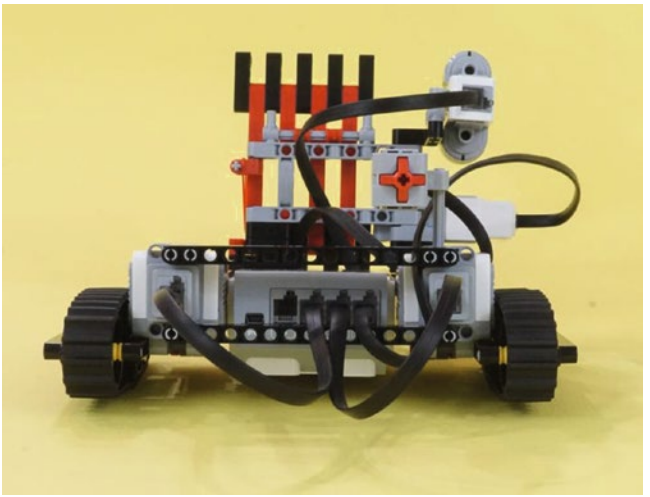


Figure 15-56. *The motor ports serving this GrabberBot. Motor ports B and C go to the large motors. Motor port A goes to the medium motor.*

Chapter 16 shows you how to program the GrabberBot so it can head down the tunnel with the Grabber Jaw open, as shown in Figure 15-57. Figure 15-58 shows it returning after grabbing the scroll.

■ **Note** This robot is front heavy. It will work well with reasonably light scrolls. But if you need it to lift heavier objects, you should add some ballast in back. Anything substantial from your LEGO kit, fastened to the rear, will make the bot less front heavy.

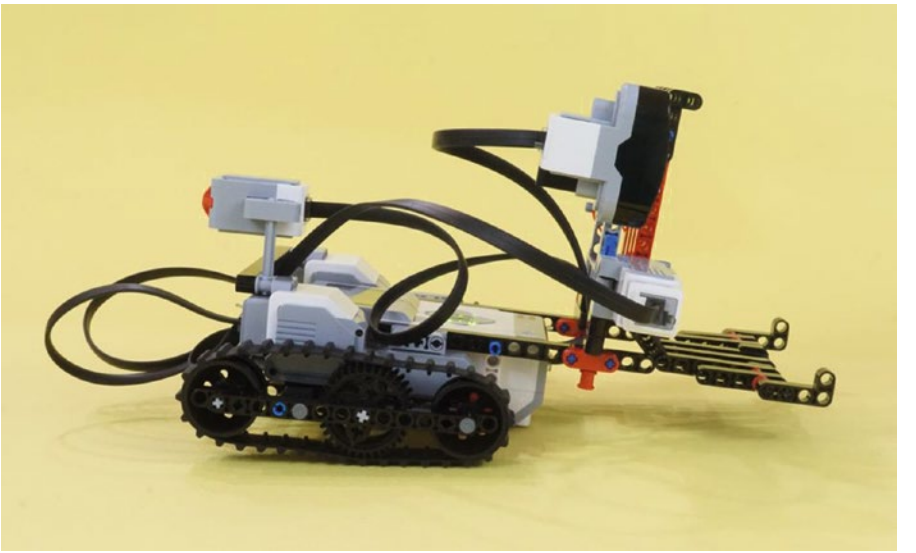


Figure 15-57. *The GrabberBot moving down the tunnel, Grabber Jaw wide open*

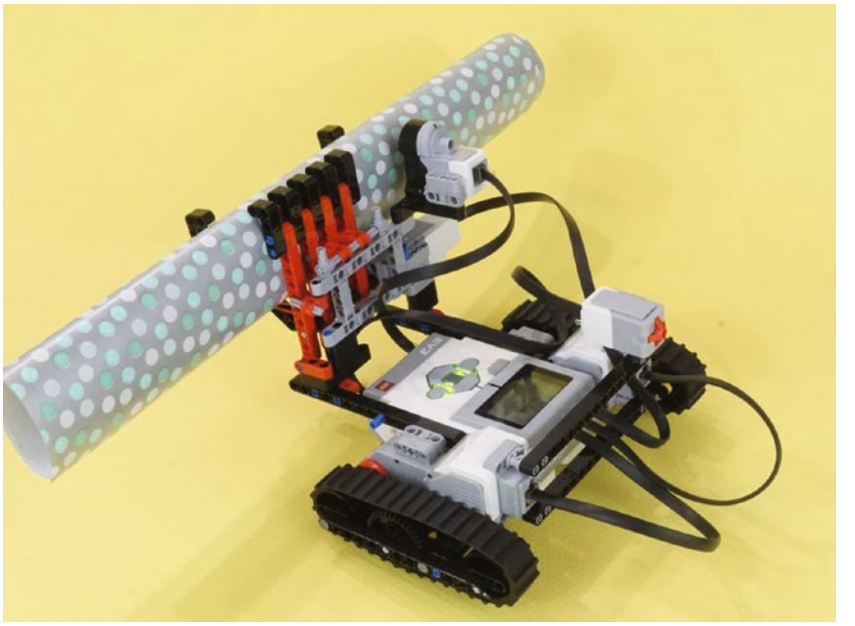


Figure 15-58. *The GrabberBot does its job! Some programming required…*

CHAPTER 16

GrabberBot: Program It

During the programming of the GrabberBot, I'm not going to tell you to use any particular method for programming; I'm leaving that up to you. You might choose to open up the EV3 software and start dropping blocks until you have the program completed; then you'll download the program, test it, and debug it if you find any problems. Or you might use the method described in Chapter 12, moving forward with your program only after you've successfully placed a block and then tested that the bot does what it's supposed to do.

Or you can follow along with the method I present in this chapter, which is a slight variation on the second method. I'll place a block, configure it, download it to my bot, and then test it. If the bot doesn't perform as anticipated, I'll reexamine the block and its configuration settings and see whether I've made a mistake. In some instances, such as the placement of a `WAIT` block, I might place two blocks and then test. The `WAIT` blocks are sometimes so simple I choose to test them at the same time as a more advanced block.

Whichever method you choose, take notes of your successes, but *especially* your failures. Try to determine where you made your mistakes—were you in a hurry and missed something important? Did you choose the wrong programming block for the job? If you pay attention to your mistakes and learn why you made them and how you solved them, your bot programming skills will definitely improve.

Down the Tunnel . . . Again . . .

If you're following along with the method I've chosen, open the LEGO MINDSTORMS EV3 software, click on the + tab at the upper left, and then double-click on the blue Program name. Then type **GrabberBot**. Your screen should look like Figure 16-1.

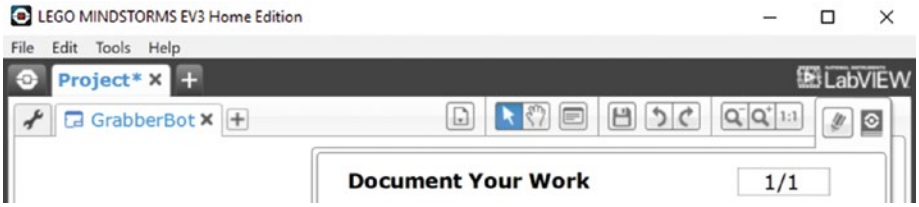


Figure 16-1. *Enter GrabberBot for the new program name and click anywhere else on the screen*

Note To have more workspace visible on your screen, minimize the Document Your Work area on the right by clicking the EV3 Button symbol at the far right of the top tab bar. (This symbol is shaped like a little stop sign—that is, an octagon. It resembles the buttons on your EV3 Intelligent Brick.)

M. Bell and J. F. Kelly, *LEGO® MINDSTORMS® EV3*, DOI 10.1007/978-1-4842-2262-1_16

Engineering: A Well-Regulated START

I placed the Touch Sensor on the back of the bot to use as a Start button for the GrabberBot. I've found that if I don't need the Touch Sensor for a bot design, I can include it as the Start button, and it keeps me from having to press any buttons on the Brick, and sometimes the location or orientation of the Brick can make pressing its buttons difficult. By placing the Touch Sensor in an easy-to-reach location, I don't have to worry about accidentally bumping my bot when I pull my hand away quickly after pressing the dark gray Run button. (Some programmers put a 5- or 10-second `WAIT` block at the beginning of their programs so they can press the dark gray Run button on the Brick and still have time to place their bot correctly before it begins to run. That works, too.)

The first block I'll place in my program is a `LOOP` block (see Figure 16-2) configured to detect the press and release of the Touch Sensor. When the button is pressed and released, the loop will break and the remaining programming blocks will begin to run.

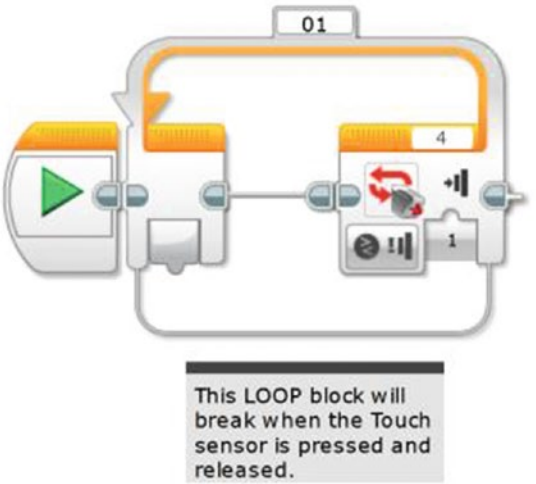


Figure 16-2. *A LOOP block and Touch Sensor trigger are used to start the GrabberBot*

Look back at my `Task List` (turn back to Figure 14-2). My first task is "Move toward the end of the tunnel." There are actually two movements here, though. The first is the bot moving toward the scroll and stopping a safe distance away to begin detecting its target. The second movement is toward the scroll (at a slower pace) and letting the Infrared Sensor detect the proper place for the GrabberBot to stop to grab the scroll. Should the Infrared Sensor look for the wall or for the scroll itself? The GrabberBot design in Chapter 15 has the sensor aimed precisely at the level of the scroll. If you use this design, you'll want to stop just short of the actual scroll. If your sensor were mounted significantly higher, then you'd want to use the distance to the back wall.

Right now, I need to perform the first movement, but I still haven't told you how I plan on programming the bot to keep from moving too far (and accidentally bumping the scroll). Look at Figure 16-3.

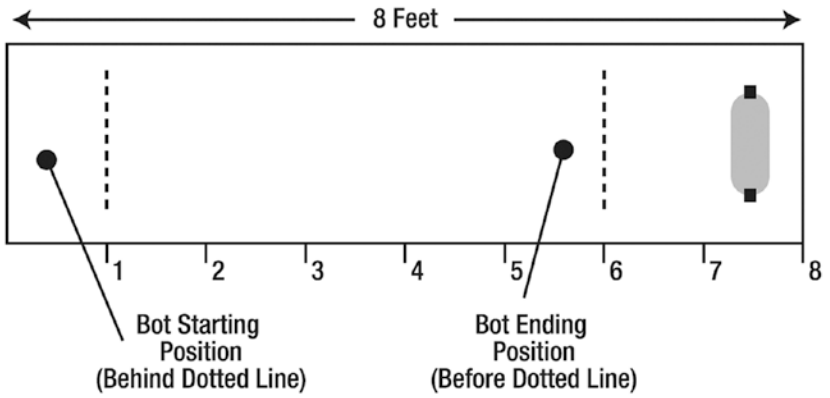


Figure 16-3. *Determining the correct distance for the GrabberBot*

In this figure, I've labeled the GrabberBot's starting position as being behind an imaginary line at the one-foot mark inside the tunnel. My bot is slightly longer than one foot, but my calculations will be based on the tank-treads being behind this line. The tunnel is eight feet deep, and the scroll is four inches from the back wall of the tunnel. I placed another imaginary line at the six-foot mark. If I can program my bot to stop before it reaches the six-foot point shown in Figure 16-3, it should be a safe enough distance away not to accidentally bump the scroll.

Engineering: Precise Calculation of Distance: Rotations and Degrees

The method I'm going to use for accomplishing this is more exact than the method you saw for the SnapShotBot in Chapter 12. In Chapter 10, I divided distances to travel by the circumference of the wheel (5.5 inches) and rounded the numbers up or down. The distances the SnapShotBot traveled weren't as exact as I need for my GrabberBot to travel. So I need a different method for fine-tuning the distances my GrabberBot will move. There's a little math involved (sorry), but I don't think you'll have trouble following along—but do grab a calculator. (Unlike your school teacher, I allow calculators!)

But what is the circumference of the "wheel" on a tank-style robot? It is probably less than 5.5 inches, since it is really the circumference of the hub plus the thickness of the tank-tread. But that's getting complicated. How about this: let's test it. Program the tank to move one rotation and measure it? We could, but a more accurate method would be to program it for ten rotations, measure that distance, and divide by ten.

Let's start with a simple observation. When the wheel makes ten complete rotations on the ground, it will have driven on the tank tracks for a distance of 39 inches. I measured this several times and found that, on a hard floor, it always travelled 39 inches. So one rotation is a tenth of that, 3.9 inches. Are you with me so far? Now it just so happens that when a wheel spins completely around for one rotation, it has also spun 360 degrees. When you travel around a circle and return to your starting point, you have made a trip of 360 degrees—got it? Okay, so if the wheel spins for half a rotation, how many degrees has it traveled? Did you answer 180 degrees? What about a quarter of a rotation? The answer is 90 degrees.

Now, let's move our bot around a little more. If your bot moves forward for 8 rotations, how many degrees has it traveled? To find the answer, you simply multiple 360 (degrees) by 8 (number of rotations) and the answer is 2880. If you wanted to configure a `MOVE` block to spin your robot for eight rotations, you could also program it to spin for 2880 degrees—it will move the same distance. Test it!

And the best part about converting rotations to degrees is that you don't have to round your number of rotations—if you want your bot to move 2.25 rotations, you simply multiply 360 by 2.25 and the answer is 810 degrees!

The math works both ways. If you have a bot you want to move forward 3432 degrees, how can you determine how many rotations to program it? Just do the reverse—divide the number of degrees of movement by 360, and the answer will be the number of rotations you need to program your bot to move. So, 3432 degrees divided by 360 equals 9.5 rotations (rounding your numbers to one decimal place, but don't worry—for most of your bots, this should be accurate enough).

So, before moving forward, let's make sure you understand two simple rules:

- To convert rotations to degrees, simply *multiply* the number of rotations by 360, and the answer is the equivalent number of degrees.
- To convert degrees to rotations, simply *divide* the number of degrees by 360, and the answer is the equivalent number of rotations.

How will I use this information to guide my GrabberBot to the proper location? Well, if the GrabberBot wants to move from the one-foot mark to the six-foot mark, it will need to move *no more than* five feet, right?

I know that five feet is 60 inches (there are 12 inches in 1 foot, so for 5 feet, I multiply 5 by 12 to get 60). Since one wheel rotation is 3.9 inches, if I divide 60 inches by 3.9 inches, this will give me the number of rotations a wheel will turn if it travels five feet. The answer is 15.4 rotations. Now comes the math. Rule #1 tells me all I need to do to convert 15.4 rotations to degrees is multiply it by 360. When I do this, I get an answer of 5538 degrees. When I program my first `MOVE` block, I will tell my GrabberBot to move forward 5538 degrees, and I know that it should be stopped on or behind that second imaginary line in Figure 16-3.

Before I continue, are you now aware that robotics can involve some mathematics? Would you believe that more advanced robots will require even more advanced math skills? Since I let you use a calculator, please allow me to encourage you to not ignore your math and science studies, okay? If you like designing robots and hope to work your way up to more advanced designs, you'll need to concentrate in school on *all* your subjects (including History)! Okay, enough lecturing—back to the programming.

So now I can place that first `Move Steering` block (see Figure 16-4). I've configured it to spin 5538 degrees, which will give us five feet.

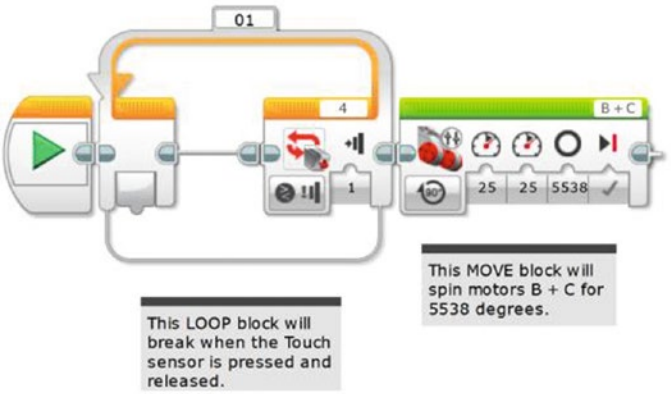


Figure 16-4. *This MOVE STEERING block controls the GrabberBot's movement down the tunnel*

What if my calculations were a little off? Well, I did round the number of rotations from 15.384 (when I divided 60 inches by 3.9 inches earlier), so could this make my GrabberBot possibly bump the scroll? If you look again at Figure 16-3, you'll notice that I have another foot-and-a-half of distance before the scroll, so I should be safe. That's why I intentionally chose to stop the bot at the six-foot mark and not the seven-foot position.

Now I'm going to download my program and test it. I have a measuring tape, and I want to make sure that my bot moves forward about five feet and no more. I test it and . . . it works. I press the Start button (Touch Sensor), and the bot moves forward almost exactly five feet from its starting point. Excellent.

Because I need to take a deep breath at this point (and maybe you do, too), I'm going to have my bot pause for 10 seconds. I'm doing this because as I mentioned in Chapter 14, if I find that my bot isn't lined up exactly where I want it, I could use a string tied to the rear of my bot and pull it back. It's not the best solution, but I really want my bot to make one attempt on the scroll, and I want it to be successful. So I'll add a `WAIT` block, which will let me look down the tunnel and make certain the bot is pointed directly forward and that the Grabber Jaw mechanism will fit smoothly underneath the scroll's supporting legs (see Figure 16-5).

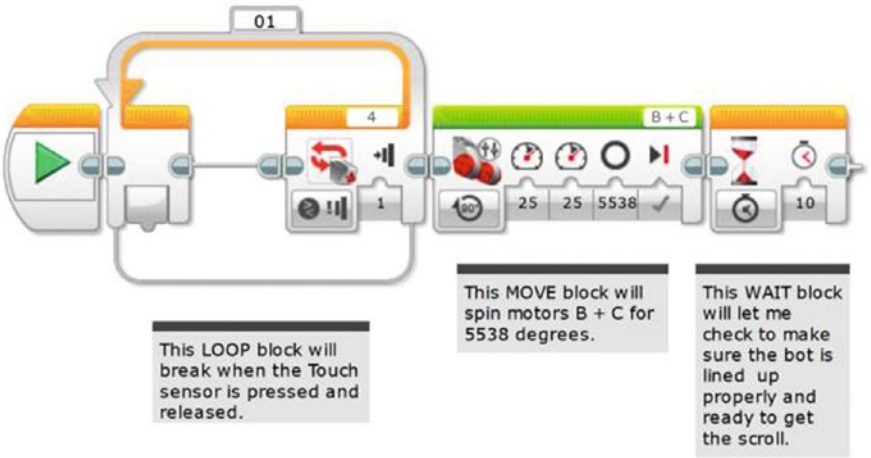


Figure 16-5. *A WAIT block will let me check to make sure the GrabberBot is ready to get the scroll*

I will tie a string to the rear of my bot, but hopefully I won't have to use it. The `WAIT` block is just a safety precaution that I've decided to add—feel free to leave it out if you feel your bot is ready to move forward and grab the scroll.

Approaching the Scroll

Now, the next movement for my GrabberBot is a little trickier. The bot must crawl forward, very slowly, and try to detect the surface of the scroll. Why the scroll itself and not the back wall, you may ask? Well, my GrabberBot is designed so that its arm mechanism—the Grabber Jaw—will come in *under* the scroll and lift up. It's small and low, and I've designed it to fit between the two support legs. If you take a look at my final GrabberBot design (refer back to Figures 15-1 and 15-57), you'll notice that the Infrared Sensor is level with the center of the scroll. What I'll do is test this over and over until I am certain the Infrared Sensor triggers at the right time—this position will be when the lifting arm is directly under the scroll, the best position for it to get a good lift-off. It could be one inch from the wall or less than that and I won't know until I test it.

Since I need to test it, this is a good place for me to explain my test environment, shown in Figure 16-6.

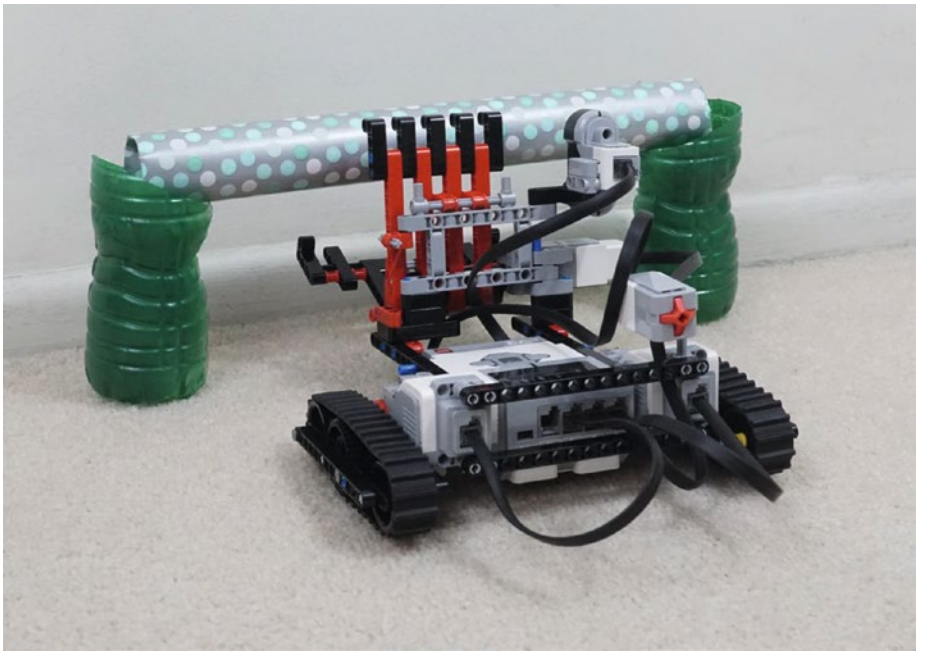


Figure 16-6. *Photograph of the "scroll" and support legs*

What I've done for my test environment is measure out 8 feet from a wall in my living room. I place some obstacles to represent the tunnel walls (okay, they are couch cushions—use what you have). I measure 4 inches from the wall and place two curved-cut plastic water bottles about 15 inches apart—these are the support legs for my scroll. These bottles are nice because they're available everywhere and you can make them exactly the height you want. The inside edge—the lower part of the curve—is a precise 4 inches from the ground. The outside, taller edge is perhaps an inch higher, to create a nice appearance. (And in this case I spray painted them green, to make them easier to see in this photograph. That makes them more Mayan, too.) The whole arrangement requires your robot move with precision—if it is a little off it will tip the bottles over. You could use tin cans if you want the supports to be sturdy on their own.

And finally, for the scroll, I set on top of the support legs a piece of gift wrap with rolled-up newspaper inside. Simple, but it all works great.

Now, before I begin testing, I need to program my bot to slowly approach the scroll. To do this, I'm going to use a `LOOP` block that will break when the Infrared Sensor is triggered. So I first drop in that `LOOP` block (see Figure 16-7).

For now, I've configured the Infrared Sensor to trigger when it gets one centimeter from the actual scroll.

Next, I'll drop in a `Move Tank` block (see Figure 16-8).

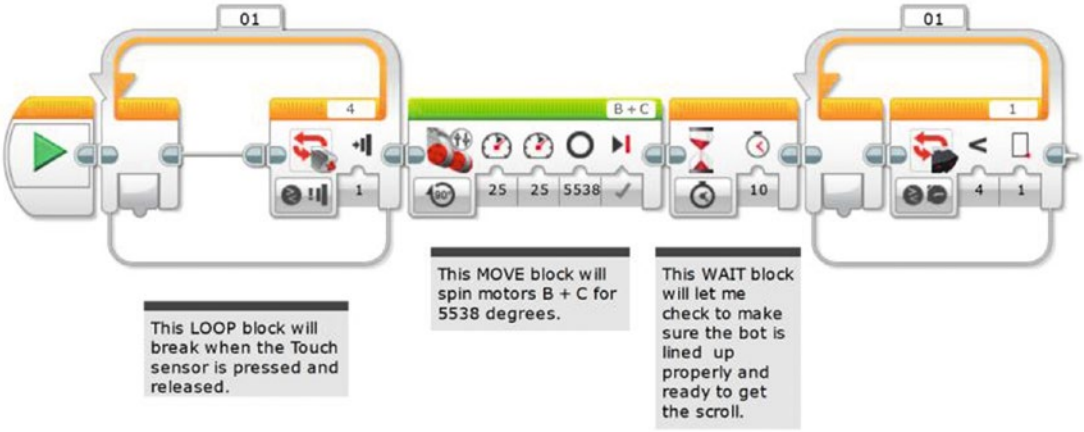


Figure 16-7. *The LOOP block will break when the Infrared Sensor on port 1 is triggered*

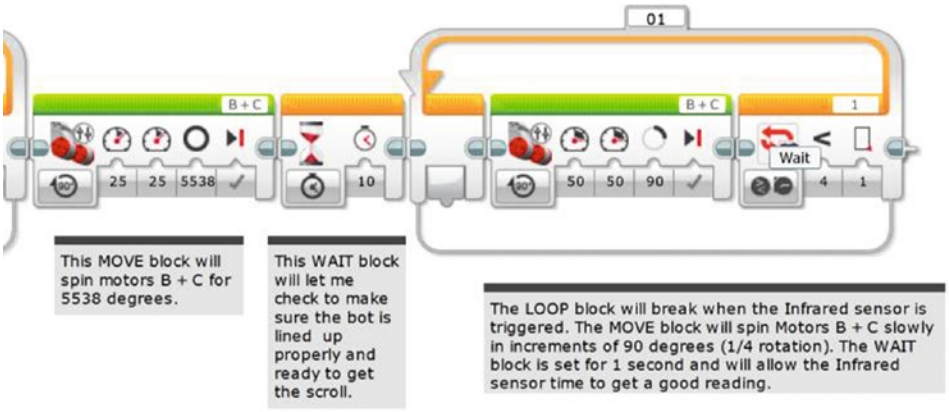


Figure 16-8. *The Move Tank block will move the bot forward slowly for a short distance*

I'm going to have the tank-tread motors B and C spin slowly (Power is set to 50), and a just a little bit at a time. The bot will move forward a short distance (90 degrees, which is .25 rotation), and the Infrared Sensor has a one-second delay to see whether the scroll is within one centimeter (see Figure 16-9). If it's not, the bot will move forward again. It will keep doing this until the Infrared Sensor breaks the loop.

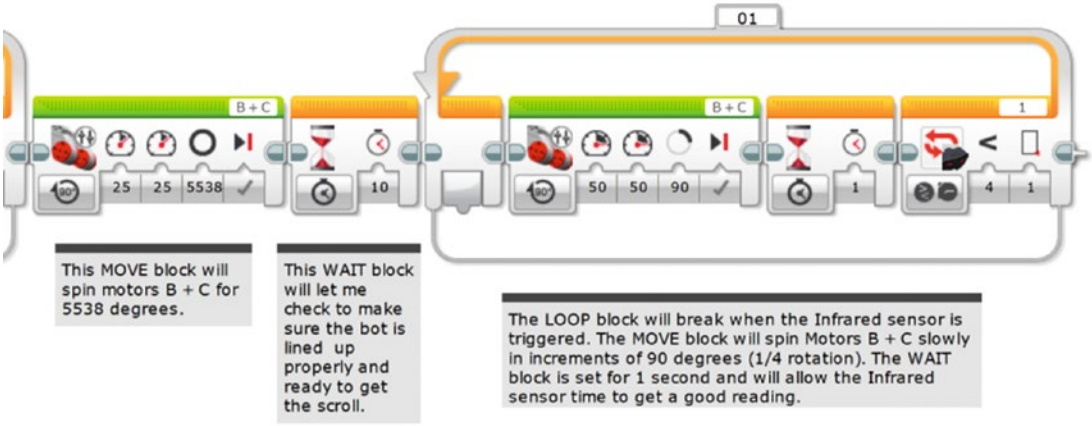


Figure 16-9. *The one-second WAIT block in the loop allows the Infrared Sensor to get a good reading*

Now I download the program and test. I tried four tests and found the Infrared Sensor stopped very close to the scroll every time. The Grabber Jaw was in the perfect position to do its work.

If I had mounted the Infrared Sensor high up to sense the back wall and not the scroll, I would have set it to look for the four-inch distance from the wall. That would be about ten centimeters. Then I would run a series of tests to make that distance exact.

■ **Note** Every Infrared Sensor is different, and your Infrared Sensor will have a slightly different sensitivity than my Infrared Sensor. Because of this, you need to test your bot and record your own results. The final value for my Infrared Sensor (one centimeter) might not work with your GrabberBot. Although the Infrared Sensor is fairly accurate, always test with your own values, not the ones I am using—just to be safe!

At this point, my bot will now move down the tunnel and stop about two feet in front of the scroll. It then begins to move slowly toward the scroll (stopping and pausing every .975 inches—that is, 3.9 divided by 4 since it is set to 90 degrees) until the Infrared Sensor detects the wall and triggers the bot to stop. That little bot's Grabber Jaw is directly under the scroll, ready to lift it up and return it to me.

Acquiring the Scroll

When my GrabberBot lifts the scroll off its support legs, I intend for it to pin the scroll against the Grabber Back Wall, held by the Grabber Jaw (flip back to Figure 15-58). By holding the scroll against those beams of the Grabber Back Wall, it is less likely that the scroll will roll or fall out of the arm mechanism on its return trip.

Now for this to work, I need Medium Motor A to apply continuous upward pressure against the scroll. To do this, I could simply put a `MEDIUM MOTOR` block into a `LOOP` set to loop Forever. But if I have the `LOOP` block configured this way, it will never break. How will the bot be given instructions to return with the scroll?

Engineering: Parallel Processes

The solution to this involves *parallel processes*. What are parallel processes? They are simply programming commands (that is, blocks) your bot will execute simultaneously. For example, you could program a bot with a `Move Steering` block that has motors B and C moving the bot randomly around the room, never stopping. At the same time, you could program the bot to also have motor A rotating the Sound Sensor back and forth, listening for sound input. So far, all of the bots in this book have only one block running at a time (like a `WAIT` block or a `LOOP` block). When one block finishes, the next block starts.

In order to program my bot to do two things at once, I will create another path where I'll place programming blocks that will run simultaneously. But before I do this, I'll finish this branch of the program as if motor A already has the scroll pinned. And when I have that done, I will go back and add that second sequence path containing the blocks for motor A.

Okay, so if I assume that my GrabberBot already has the scroll pinned, all that's left is for the bot to move backward and return to the end of the tunnel. I'll do this by adding in a `MOVE TANK` block for motors B and C (see Figure 16-10). I don't want the bot to return too quickly, so I'm also going to program it for a slower speed (Power is set to 40).

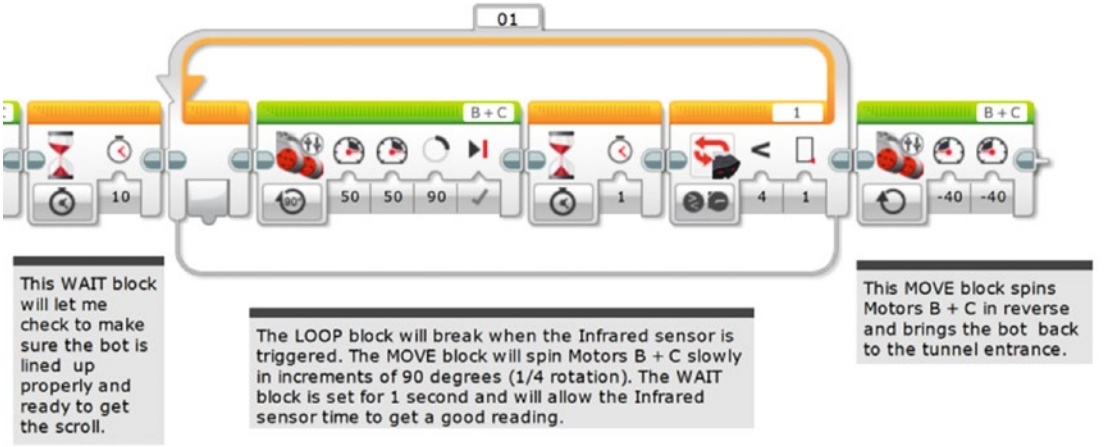


Figure 16-10. *A Move Tank block is used to bring the bot back to the tunnel entrance*

That program with the reverse `MOVE TANK` should work, so now I'm going to add the necessary blocks for the bot to grab the scroll. To do this, I'll be adding a parallel process as mentioned earlier.

The EV3 programming language allows us to do this by simply creating another execution path that breaks away from the main path. To do this, you need to first position the destination `Medium Motor` block that you want to hook up. As shown in Figure 16-11, that new `Medium Motor` block will be partially visible (or "dimmed") to show it is not yet properly connected to any programming path.

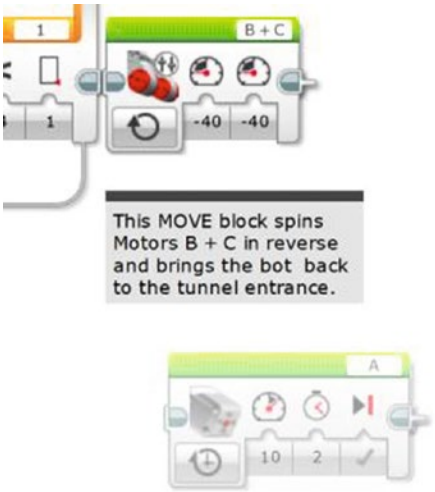


Figure 16-11. *Place the destination Medium Motor A block. It will be dimmed as shown.*

Then click on the point-of-origin as shown in the center of the enlarged Figure 16-12. The point-of-origin will turn blue when highlighted. You can now drag down from there and see a new path starting.

Figure 16-12. *Click on the blue point-of-origin for your new parallel path*

Now, while holding the mouse click, drag to the destination point. In our case, the left-side edge of the `Medium Motor` A block. When the path is established, it will turn gray and the destination block will change from dimmed to normal. You now have two parallel execution paths, as shown in Figure 16-13. That is, motors B and C will run at the same time as Medium Motor A. The EV3 software helpfully moved the `MOTOR STEERING` block a little to the side when the new path was added. We'll move its comment block by hand, as shown in Figure 16-14.

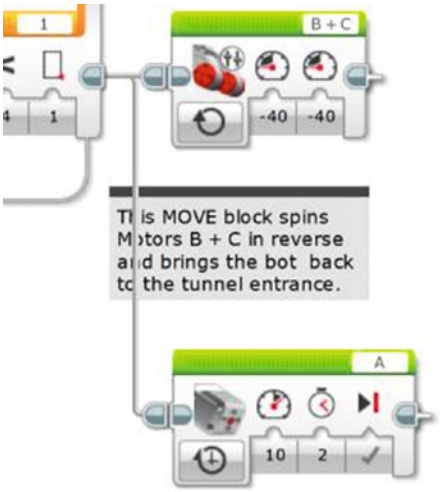


Figure 16-13. *Parallel execution paths, whereby the motors will run at the same time. We'll reposition the comment by hand.*

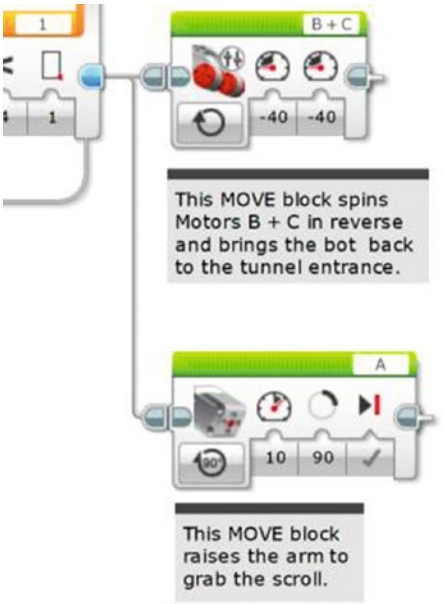


Figure 16-14. *The Medium Motor block will spin motor A to grab the scroll. Give it a Power of 10 and 90 degrees of movement. The top comment is moved to the right and another comment added for motor A.*

I can still place programming blocks on the original path, as you'll see in Figure 16-15.

Engineering: Testing and Making Changes

At this point in the program, my Grabber Jaw is under the scroll, and I want it to lift up just enough to pin the scroll against the Grabber Back Wall of my GrabberBot. I'll need to test motor A many times to determine the proper number of rotations or degrees to do this. But for now, I'll place a `MEDIUM MOTOR` block to perform the lifting, as shown in Figure 16-14. I configure it to lift slowly (the Power is set to 10) and for 90 degrees.

Now I download and test the program. And I encounter four problems. The first problem is that the 90 degrees I've configured motor A to spin is not enough (the scroll's weight may be the issue). I test it again and decide that instead of degrees, I'll use time. Two seconds is a good setting to pin the scroll against the Grabber Back Wall. With it pinned, the Grabber Latch from Chapter 15 will hold it for the return trip. Or, you could not build the rubber-band latch, and simply keep the motor on for enough time to make it all the way back. That uses significantly more battery since the motor would be in a "stall" mode that takes lots of electricity to maintain. The rubber-band latch is an appealing solution if you're going to run a large number of tests.

The second problem requires me to increase the Power setting from 10 to 40. I need this additional power to lift the weight of the scroll. Remember, *test often*!

The third problem is that setting the return B and C motors to ON should have worked. But it doesn't. We've seen this before. You either need to put that `MOTOR STEERING` block in a `LOOP` or set it to enough rotations to get it all the way back. If the total distance is eight feet, that's 96 inches. Each rotation gives us 3.9 inches of movement, so dividing 96 by 3.9 is about 25 rotations. That should fix it. The change is seen in Figure 16-15.

Engineering: Timing the Parallel Paths

The fourth problem requires a short explanation. When the program runs and the extra path splits off (see Figure 16-15), the program will execute the first block in the upper beam and the first block in the original beam at the same time. What this means is that motor A will lift (as we see the lower program execution path) at the same time as my `MOVE STEERING` block (in the upper, original path) instructs motors B and C to spin in reverse to return the bot. *Big problem*! The bot needs to first grab the scroll and *then* move in reverse.

To fix this, I'll simply add a `WAIT` block that will run at the same time as motor A is lifting. I do this because I want to give the bot time to grab the scroll and pin it to the Grabber Back Wall before moving back down the tunnel. I'll put in a `WAIT` block configured for five seconds, as shown in Figure 16-15. This five-second `WAIT` block allows the arm to close before the "reverse" `MOVE` block is executed and the robot returns with the scroll.

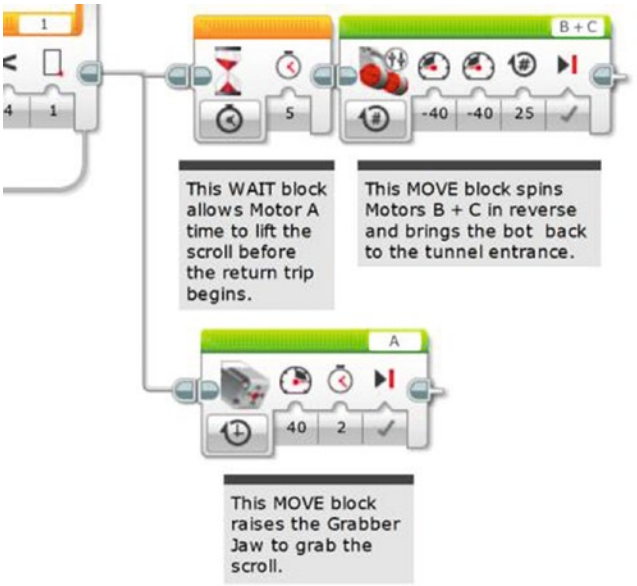


Figure 16-15. *The WAIT block will give medium motor A time to lift the scroll*

Let's handle one last detail. Earlier we said there'd be a string attached to the back of the robot to pull it back if it did not travel in a straight line. But we should allocate some time to decide whether to pull it back. Add another `WAIT` block configured for ten seconds, as shown in Figure 16-16. That ten-second `WAIT` block lets the robot pause before the grabbing operation and gives us time to decide if this trip down the tunnel was a "keeper."

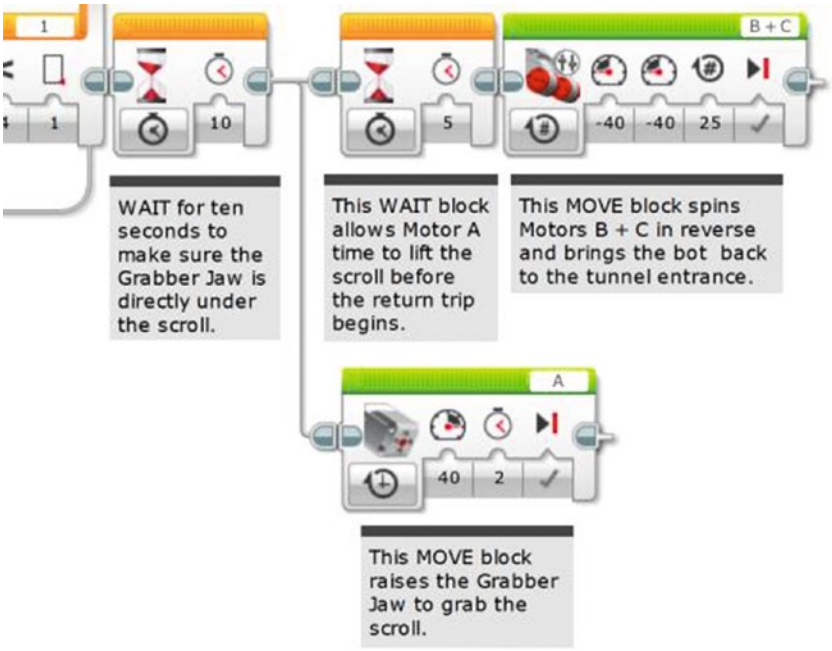


Figure 16-16. *An additional WAIT block gives the operator (Evan) time to decide whether to pull it back with the string*

And once again, I test my bot. Because I've been testing my bot often, usually after each major programming block is added, my bot doesn't require any additional changes. On its *very first test*, this is what happens:

1. I push the Start button (Touch Sensor).
2. The bot moves forward toward the scroll about five feet and stops. It doesn't touch the tunnel sides at all (okay, cushions from my couch).
3. The bot moves .975 inches forward, the Infrared Sensor tries to detect the scroll and fails, and the bot moves forward another .975 inches.
4. The bot repeats Step 3 about six times before the Infrared Sensor is triggered.
5. Motor A spins, and the Grabber Jaw raises the scroll, pinning it against the Grabber Back Wall of the GrabberBot.
6. A few seconds later, the bot begins moving backward and brings the scroll back to me.

Perfect test!

Just to be sure, I perform two more tests with the GrabberBot, and they're successful. With three successful runs of the GrabberBot, I am fairly confident in my bot's ability to run down the tunnel, find the scroll, lift it off the support legs, and then return the scroll to me.

Take your GrabberBot and test it until you are able to successfully retrieve the test scroll a few times. When you're ready, set up the scroll one final time and run the bot. If it works, congratulations!

You have the scroll, and the team is one step closer to finding King Ixtua's burial chamber . . .

CHAPTER 17

Bravery, Wisdom, and Honor

Location: Southwest Guatemala

Weather Conditions: 89° Fahrenheit, Humidity 52%, Rain 25%

Day 5: Inside King Ixtua's Throne Room, 10:12 AM

Evan's uncle had unrolled the small scroll with gloved hands, careful not to tear the thin parchment. After Max had finished photographing the Mayan writing on the scroll, Uncle Phillip had given the scroll to Grace for translation. That had been two hours earlier, and now Evan was sitting with his uncle and Max in the tent, waiting for Grace to complete her initial review of the scroll's contents.

Max used the time to inventory all the film he had used photographing the excavation and exploration of the tomb. He told Evan that he had over 1,000 photographs and would probably take an additional 300 or more before the team left for Florida.

Uncle Phillip had been working on his laptop, creating a proposal to present to the government of Guatemala for further excavation and study. Evan was surprised at the amount of paperwork that his uncle had to file in order to request more time at the site.

And Evan had used the time to dismantle his GrabberBot.

But all that stopped when Grace entered the tent carrying the scroll, a handful of paper sheets, and a large piece of posterboard. She walked over and placed it all on the table. "Well, I think we're ready," she said.

"What did you find," asked Uncle Phillip. "Did the scroll have the instructions we need?"

Grace nodded. "Even better," she said with a smile. "The scroll does tell us how to access the burial chamber. But it also tells us quite a bit about the burial chamber itself."

Uncle Phillip leaned back in his chair and stared at the tent's roof. "I was hoping for that," he replied. "It would be a shame to come this far and not get a glimpse of the burial chamber."

Max shook his head. "I was worried, too," he added. "The Tupaxu manuscript had no information on the burial chamber."

Evan looked at his uncle and then to Grace. "So, what's next?" he asked.

Uncle Phillip laughed and stood up. "Exactly right, Evan. Grace, what do we do?"

Grace pointed at a sheet of paper. "The scroll clearly states that to open the burial chamber, someone must be sitting on the throne. That pressure plate must be triggered. Once it is triggered, this rope must be pulled," she said, pointing at a drawing she had made. (See Figure 17-1.)

M. Bell and J. F. Kelly, *LEGO® MINDSTORMS® EV3*, DOI 10.1007/978-1-4842-2262-1_17

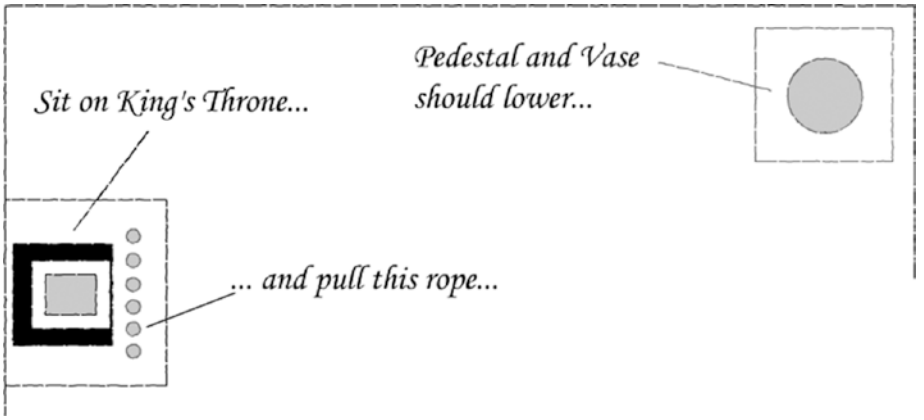


Figure 17-1. *Grace's sketch of the throne room*

"After the rope is pulled, this pedestal will lower and the burial chamber can be accessed," Grace added. "But, we'll need to be very careful with the burial chamber. I suggest that we go open the burial chamber before we do anything else. I need to verify that the room matches the scroll's description first."

Uncle Phillip nodded. "I agree. Let's go open the chamber and see what we're facing. We've come this far with no mistakes, so there's no point in rushing forward too fast."

Evan stood with the others and followed his uncle back to the tomb.

The Burial Chamber

Uncle Phillip sat on the King's throne and waited. "Anyone hear or see anything unusual?" he asked.

Evan tried to listen for any strange noises that might indicate a trap had been triggered, but the chamber was silent.

Max shook his head. "I don't hear anything."

"I don't either," said Grace.

Uncle Phillip pointed to the throne room's entry door. "I'd like all of you to stand outside while I pull the rope. Just in case something goes wrong," he said.

Max, Grace, and Evan walked back to the corridor and waited.

"Okay, I'm pulling the rope now," Uncle Phillip said.

A few seconds passed. And then a grinding sound was heard by the team.

Grace pointed at the corner. "The pedestal is dropping," she said.

Evan watched as the pedestal began to lower. The large vase on the pedestal began to disappear down a large square hole. Another minute passed and then the grinding sound stopped.

"Congratulations, team!" said Uncle Phillip as he stood, a huge smile on his face. "Come on in and let's take a look at King Ixtua's burial chamber."

Evan followed Max and Grace as they crossed the room. Max had turned on his flashlight and handed it to Evan's uncle.

Uncle Phillip took the flashlight and got down on his stomach. "Come on, all of you get in here and take a look, too."

Evan got down on the floor next to his uncle. Grace and Max were next to them on the adjacent side of the square hole.

"Do you hear that?" asked Evan.

"Running water," replied Uncle Phillip, shining the flashlight into the hole.

"That was part of the major trap of the tomb," said Grace. "If we had triggered any of the earlier traps, the tomb was designed to flood. Look over there," she said and pointed.

Evan angled his head to look where Grace was pointing. The beam of light from the flashlight was reflected in a large stream of water running across the room. But what caught his eye was the substantial stone sarcophagus behind the stream of water. There wasn't much light from the flashlight, but what he could see amazed him.

Uncle Phillip handed the flashlight to Grace. "Okay, Grace, take a look around and verify what you can about the room. Max, take some photographs, but no one goes in until we discuss our plan, okay?"

Grace and Max nodded.

"Evan and I are going to go back to the tent and give you some room to move around. As soon as you're ready, come back and let's figure out our next step."

Evan tried to make sense of the things he saw in the burial chamber as he stood up. He had seen what he thought were some small figurines, and some sort of ramp leading to the sarcophagus.

Famous Figures

Max was finishing one of his uncle's famous grilled burgers when Max and Grace returned. They sat down at the table and began to eat the lunch Uncle Phillip had prepared.

A few minutes later, Uncle Phillip took a seat at the table with the team.

"Max, did you get enough photographs?" Uncle Phillip asked.

"Plenty," replied Max. "I'll take more once we get some better lighting down there, though."

"Most definitely," said Uncle Phillip. "Grace, were you able to verify the accuracy of the scroll?"

Grace nodded and finished chewing. "Yes, and I'm happy to report that it appears that the burial chamber has not been disturbed. It matches the scroll's description exactly, including measurements."

"That sounds like good news," said Evan.

Grace stood and pulled the posterboard over where the team could easily view it. "This is a drawing of the burial chamber with a few measurements I was able to translate." (See Figure 17-2.)

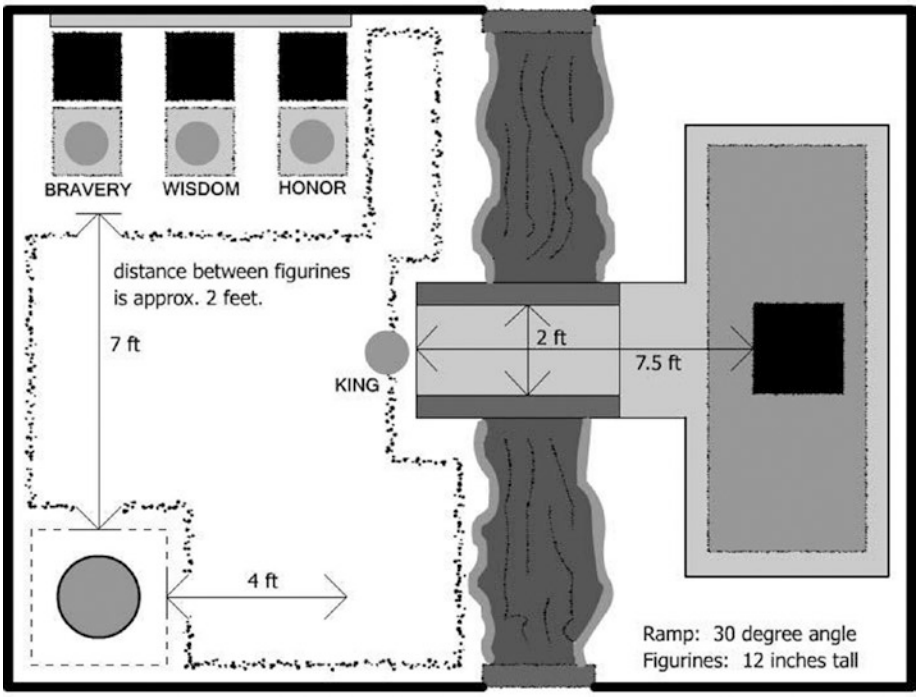


Figure 17-2. *Grace's sketch of the burial chamber*

Evan looked over the drawing. The sarcophagus was the largest item in the room, but it was the small objects that caught Evan's eye. He was certain there had to be at least one more challenge. "What are these circles?" he asked.

"Inside the chamber are four small statuettes. Each one is a carved wooden figure," said Grace. "This first one is U'laka. He was King Ixtua's bravest warrior. The next one is Raxu, considered to be the king's wisest friend. And this one is Ba'rii, Ixtua's most loyal military leader. Each figurine has a different Mayan glyph carved on it—Bravery, Wisdom, and Honor."

"What about this one," asked Max, pointing to a fourth circle on the drawing.

"That's a figurine of King Ixtua," replied Grace.

"And the water flows under this ramp here?" asked Uncle Phillip.

"Yes. And the scroll says that if the trap in this room is triggered, the room will flood. And that's the bad part," Grace added. "The floor around the pedestal is one large pressure plate."

"So, there is one more challenge," said Uncle Phillip.

"Oh, yeah," said Grace. "And it's an interesting one."

The Final Challenge

Uncle Phillip patted Evan on the back. "It looks like Tupaxu wasn't kidding about using monkeys," he said. "It's a good thing he didn't know about robots, Evan."

Evan smiled. "Okay, can we go back through this one more time? I want to make sure I understand this completely," he said.

Grace took a seat next to him and pulled the posterboard closer. "Absolutely, Evan," she replied. "Let's take this in stages."

She pointed first at the pedestal. "It's safe to stand on the pedestal, but that's as far as you can go. This line I've drawn around the front part of the room is a large pressure plate. Anything heavier than eight or nine pounds will trigger the trap. The water exits here, and if the trap is triggered, the exit will seal up, making the room flood. So far, so good?" she asked.

"Got it," Evan replied.

"Okay, the next task. Your bot must move across the room to these three figurines. Each figurine must be placed on the black obsidian pressure plate behind it. But you have to be careful because there's a vertical pressure plate on the wall behind the pressure plates. If a figurine tips over and touches the plate, this will also trigger the trap."

Evan pointed at the drawing. "Do you have accurate measurements of these statues?" he asked.

Grace nodded. "Accurate from the pedestal to the figures, but I don't have exact measurements from the black pressure plates to the back wall."

Evan frowned. "Okay, I'll have to remember that."

Uncle Phillip smiled. "The good news, Evan, is that the figurines are in front of their respective triggers. At least you won't have to shuffle them around," he said.

"Okay," said Evan. "And after that?"

"Your bot will need to push this final figurine of King Ixtua up the ramp onto this black pressure plate on top of the King's sarcophagus. That will disable the floor's large pressure plate, and then we should be able to enter the burial chamber safely. And that's it."

Evan laughed. "This Tupaxu really didn't want just anyone getting to the King's burial chamber, did he?"

"Tupaxu honored his king's wishes and designed the tomb to require a trained monkey's assistance," said Max. "I'd say he did a great job, too."

Uncle Phillip took a seat across from Evan. "Well, Evan, do you think you could design us a little robot that can handle this challenge?"

Evan's Solution

"I already have an idea," said Evan. "But I also was wondering…"

"Shoot," said Uncle Phillip.

"I need to know the approximate weight of those figurines," said Evan.

"They're carved from the wood of the Irichu tree. It's not extremely heavy, but it's not a lightweight wood either. I'll get one of our guides to cut a piece of wood the approximate size and shape of a figurine and we'll weigh it for you," said Max.

Uncle Phillip turned to Evan. "Anything else?"

Evan shook his head. "Maybe later, but for now, I need to give this one some thought."

"You take all the time you need, Evan," said Uncle Philip. "Once again, we're not in any rush and we don't want to make any mistakes."

An hour later, Uncle Phillip brought in a freshly carved object and set it on the table. "It's ugly, but it's about the same height and diameter as the statuettes in the burial chamber."

Evan looked at the piece of wood. Someone had carved the rough shape of a bird in flight into a foot tall piece of wood.

"It's about four inches in diameter," said Uncle Phillip. "And it weighs about two pounds."

"Thanks, Uncle Phillip," said Evan. "That will help."

"Good luck, Evan. We've all got some other work to keep us busy, so just let us know when you're done," Uncle Phillip replied, walking out of the tent.

Evan picked up the piece of wood and stared at it. An idea began to form.

Story continues in Chapter 21 . . .

CHAPTER 18

PushBot: Planning and Design

The GrabberBot we built in Chapter 15 performed its tasks by lifting and bringing the scroll back. The bot for this next challenge will perform its tasks by doing the opposite—pushing the figurines into their proper locations.

Why pushing? Well, lifting the figurines is definitely a possibility. And you might choose to attack the problem using a lifting motion again. But for the purposes of designing my bot, I've decided that I want to keep the figurines in contact with the floor to avoid dropping them as well as reduce the risk of tipping my bot over if the statues are heavy. With that in mind, let me walk you through the planning and design of my PushBot.

PushBot Planning and Design

In Chapter 19, you can view my final solution for the PushBot. I have a dozen different ideas for how I plan on completing this challenge, and they all involve exerting a pushing force on the figurines. But I'm not going to lock myself into any particular design just yet. So get out a blank Design Journal page and a pen and follow along with me as I begin to design this new bot.

Note There should be one blank Design Journal page left in the back of this book (if you used one each for Chapters 2, 6, 10, and 14). If you need more pages, feel free to make photocopies of the Design Journal page or visit the Source/Download section of the Apress web site (`http://www.apress.com`) to download the page in PDF format.

In the `Robot Name` box, write **PushBot** or another name for your new bot; once you've decided on your bot name, move on to the description.

The Robot Description

This little bot has some repeatable actions to perform. There are a variety of ways for it to complete the challenge, but until we better understand some of the problems the bot will encounter, it will be difficult to be too specific on which sensors it will use. Because of this, I'm going to keep my description as generic as possible and avoid specifying which components might be used.

Take a look at my `Robot Description` in Figure 18-1. Keep in mind that I have plenty of time to provide more specifics later on the Design Journal page.

M. Bell and J. F. Kelly, *LEGO® MINDSTORMS® EV3*, DOI 10.1007/978-1-4842-2262-1_18

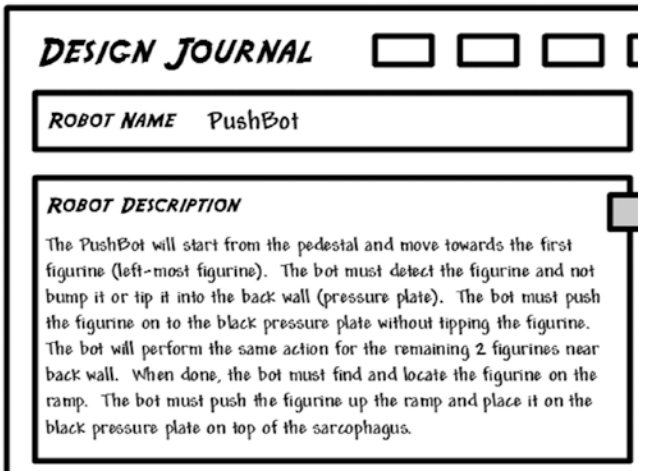


Figure 18-1. *The PushBot description isn't specific yet*

It's very generic, isn't it? I haven't specified how the bot will detect the figurines or how it will push the figurine. I haven't even decided how it will know when the figurine is on the black pressure plate. These are things I will definitely need to decide, but I don't have a complete picture of my bot in action just yet.

As with the GrabberBot design process in Chapter 14, I want to complete the Design Journal page before I actually begin to do any kind of building or testing. Why try to begin designing when you haven't even begun to consider the limitations the bot will encounter? To be truthful, my mind is already considering various options, but again, I'm not going to start building until I'm done with the planning and design process.

Now, that doesn't prevent me from giving my robot's description a little more thought. Let's walk through a couple of my bot's description sentences in more detail.

Consider the very first sentence: "The PushBot will start from the pedestal and move toward the first figurine (leftmost figurine)." I might as well take advantage of the fact that I can point my bot in the direction of the first figurine, right? If I were unable to see the first figurine, I'd probably need to add a sentence such as "The PushBot must start from the pedestal and FIND the first figurine," but luckily I won't have to do this. I can see the figurines the bot will move, and I'll use this to my advantage to get the bot off to a good start.

Take a look at the second sentence: "The bot must detect the figurine and not bump it or tip it into the back wall (pressure plate)." Although I didn't specify it, for the bot to detect the figurine without touching it will force me to use the Infrared Sensor. That's fine. I didn't specify the Infrared Sensor in the description, but just considering the statement has already helped me define how one sensor will work with my PushBot.

My next sentence is "The bot must push the figurine onto the black pressure plate without tipping the figurine." Well, since I know I'm going to push the figurine, I know my bot will need to move forward and actually touch the figurine. Now, imagine a figurine that's about one foot tall and has a small round base. If you want to push it along the floor, what do you think is the best location on the figurine to push so it won't tip over? The answer is obvious—the base. Pushing from the bottom of the figurine will reduce the risk of it tipping . . . but not completely. It is still possible that pushing at the bottom will make the figurine tip toward the bot. Don't believe me? Try it.

Place an object like a bottled drink or something similar in size to the figurine (about one foot tall and three to four inches in diameter). Push quickly at the bottom of the object and see whether it doesn't tip back toward your hand.

What does this tell me about using my bot to push an object? It tells me that I'll need my bot to push slowly. It also tells me that I might want to consider some sort of cage or other construction that will surround the object and prevent it from tipping forward or backward. But again, I don't need to consider the details just yet—I'll wait until the `Mindstorm` section before I start considering any details for accomplishing these tasks.

The remaining items for the bot's description are simply variations—find a figurine and push it. I'll need to do this for the two additional figurines near the wall and then for the figurine at the foot of the ramp.

What I need to do next is put the steps my bot will perform into the ordered `Task List`.

The Task List

I've broken down the PushBot's tasks in Figure 18-2.

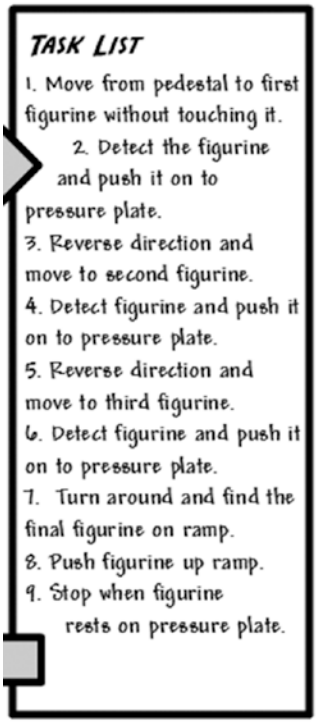


Figure 18-2. *The PushBot Task List has a lot of repetitive actions*

I'm not going to cover every task in the list; as you can see, there's a lot of duplication of steps. This is a good thing, though. It should make programming the bot fairly straightforward when you get to Chapter 20. When you've got a bot that performs a lot of similar actions, you can look for ways to simplify the programming.

For now, though, let's look at a few of the tasks in more detail.

The first task is "Move from pedestal to first figurine without touching it." I know the distance from the edge of the pedestal to the first figurine. Just as I demonstrated in Chapter 16, I can program my bot later to move a specific distance forward before it begins trying to detect the figurine.

The next task, "Detect the figurine and push it onto pressure plate," is still a little vague. I mentioned earlier that the Infrared Sensor would be useful to keep the bot from having to touch the figurine. But once the figurine is detected, I'll also need to somehow stop the bot from continuing to push the figurine beyond the pressure plate. I'm left with the Touch Sensor, the Color Sensor, and the Infrared Remote—any of these could possibly be configured to assist the bot with stopping properly. Right now, though, I don't need to decide which one. I'm going to put off making that decision until I begin to build.

The last task I want to cover is "Reverse direction and move to second figurine." This will involve some careful programming, but given that I know the measurements from one figurine to another, I should be able to program my bot to make the proper turns to put it in front of the two remaining figurines.

The other tasks in the `Task List` are, again, just variations of the same movements. After the bot has pushed the three figurines near the back wall, I'll have to get it placed properly so it can move up the ramp and push the final figurine.

With all the movements the bot will make, are there any obstacles to overcome? Let's consider those in the next section.

Limitations and Constraints

When I was considering the constraints my bot would be facing, I took another look at the burial chamber. But the bird's-eye view you saw in Figure 17-2 doesn't tell the entire story. Take a look at Figure 18-3, and you'll see the constraints and limitations I believe my bot will encounter.

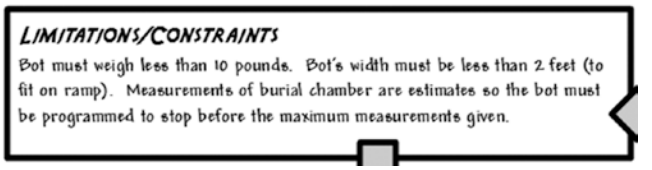


Figure 18-3. *The constraints on the PushBot*

Because my bot will need to fit on the ramp, it must be less than 2 feet in width. While there doesn't appear to be a limitation on how tall the bot can be, conservative design tells us to minimize the robot's height. Keeping the bot's components lower to the ground will reduce the risk of it tipping over when it moves up the ramp.

A very important observation from Figure 17-2 is that the measurements are not 100% exact. Take a good look at it, and you can see that many of the notes are estimates. It's difficult to measure the exact distance between the figurines, but by observation two feet appears to be a safe guess. I'll make sure my bot has plenty of room to turn and move so it doesn't accidentally bump one figurine while turning and/or pushing another figurine.

We know that the maximum weight of an adult spider monkey is approximately 10 pounds. None of my robots have exceeded that weight so far, but I'll need to be careful to keep this bot under that weight as well or the pressure plate on the floor will trigger the trap.

It might not seem like a lot of constraints, but every constraint puts more limitations on my bot's final design.

These will come into play when I begin the `Mindstorm` section of my Design Journal page in the next section.

Mindstorm

Now I can start putting down into words some of the images that are forming in my mind when I begin to think about the design of my PushBot. I've put my collection down on my Design Journal page, and you can see it in Figure 18-4.

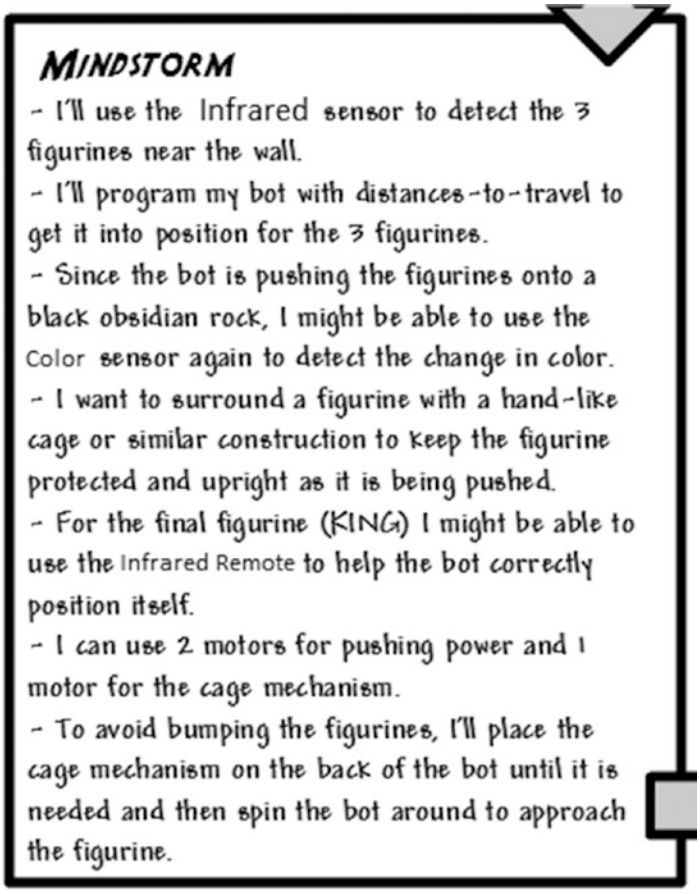


Figure 18-4. *The Mindstorm entries for the PushBot help us start developing the final design*

Let's go through some of the `Mindstorm` items and explain the reasoning behind them. The second one, "I'll program my bot with distances-to-travel to get it into position for the three figurines," will allow me to position the bot in front of each figurine without worrying about touching or tipping them over. As long as I program my bot to stop before the maximum distances shown in Figure 17-2, I should be safe. I'll have the bot back away from each figurine a good distance so it has plenty of room to safely turn and navigate to the next figurine.

The next `Mindstorm` item, "Since the bot is pushing the figurines onto a black obsidian rock, I might be able to use the Color Sensor again to detect the change in color," is a tried-and-true method I used with the SnapShotBot. The Color Sensor will be programmed to detect the change in color when the bot pushes a figurine onto the black pressure plate (black obsidian rock). This will require placing the Color Sensor on or very near the cage mechanism I intend to use so it detects the black pressure plate quickly.

Another Mindstorm item I find important is this one: "To avoid bumping the figurines, I'll place the cage mechanism on the back of the bot until it is needed and then spin the bot around to approach the figurine." What I'm envisioning is a grasping-type mechanism that will surround the figurine like a shell to protect it as it is being moved. This shell will need the ability to open and close; because of this, I think the shell will be fairly wide when it is fully open (but less than two feet—remember the constraints). I'm concerned that the cage mechanism might bump another figurine if the bot should turn left or right, so I'm going to try to place the mechanism on the back of the bot. When the Infrared Sensor detects a figurine, the bot should stop (and possibly back up a little bit), spin around, and then approach the figurine slowly. And how will it know when to stop and close the cage on the figurine?

For the answer, consider the next Mindstorm item, "I might be able to use the Infrared Remote to help the bot correctly position itself near figurines." I used the Infrared Remote to tell the StringBot when to stop, so why can't I use it here? What I'll do is watch my bot approach the figurine (slowly). When it is in position, I'll yell "STOP," push the button on the Remote, and the bot will then close the cage and proceed with pushing the figurine forward until the Color Sensor is triggered by the black pressure plate.

I do this four times for four figurines, and the burial chamber is ready to be explored by the team. All that's left before I begin to build my PushBot is for me to consider the best placement for the sensors, motors, Brick, and cage mechanism. This is where the final section of my Design Journal page comes into play.

Sketches

This is going to be a strange-shaped little robot. But, then again, all the bots have been a little unusual. (And admit it, that's the best thing about building robots—they're all unique!)

With this bot, it appears I'll be using every sensor—Color, Infrared, and Touch (for the Start button again—my favorite). I'll also be using all three motors and constructing some sort of cage mechanism.

I have some ideas on the placement of all these items, so take a look at Figure 18-5 and you'll see some of my initial thoughts on the PushBot's shape.

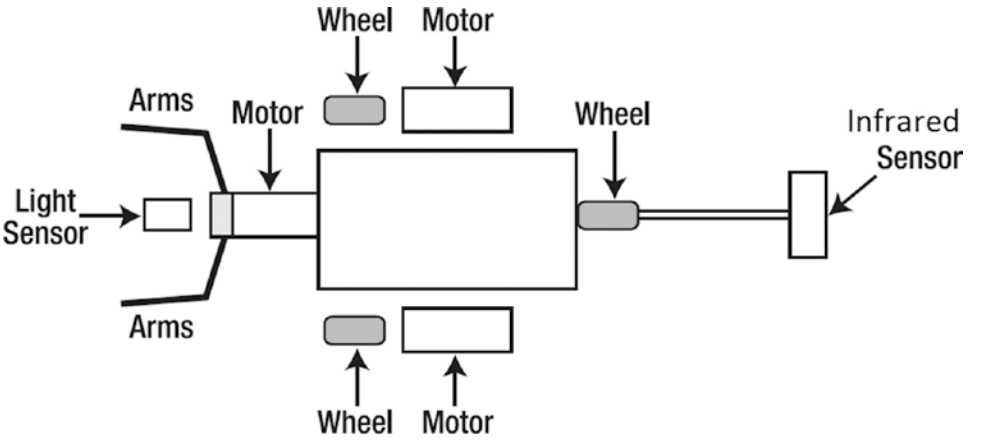


Figure 18-5. *Side view of the bot approaching the figurine*

I'll use basic shapes again to represent the motors, Brick, and sensors. I just want to start developing an idea so when I actually start building, I'll have a "target shape" to shoot for. My final design might not be exactly what is drawn, but the overall shape should be close.

Now that the sketches are done, it's time to start building and testing. In Chapter 19, I give you the steps to build my version of the PushBot.

CHAPTER 19

PushBot: Build It

It's time to build your final bot for the exploration team. If you've chosen to build your own version of the PushBot, I'm confident it won't look like the one in Figure 19-1. My version of the PushBot has some unique features that I'll cover in this chapter, so if you'd like to build the version shown here, let's get started.

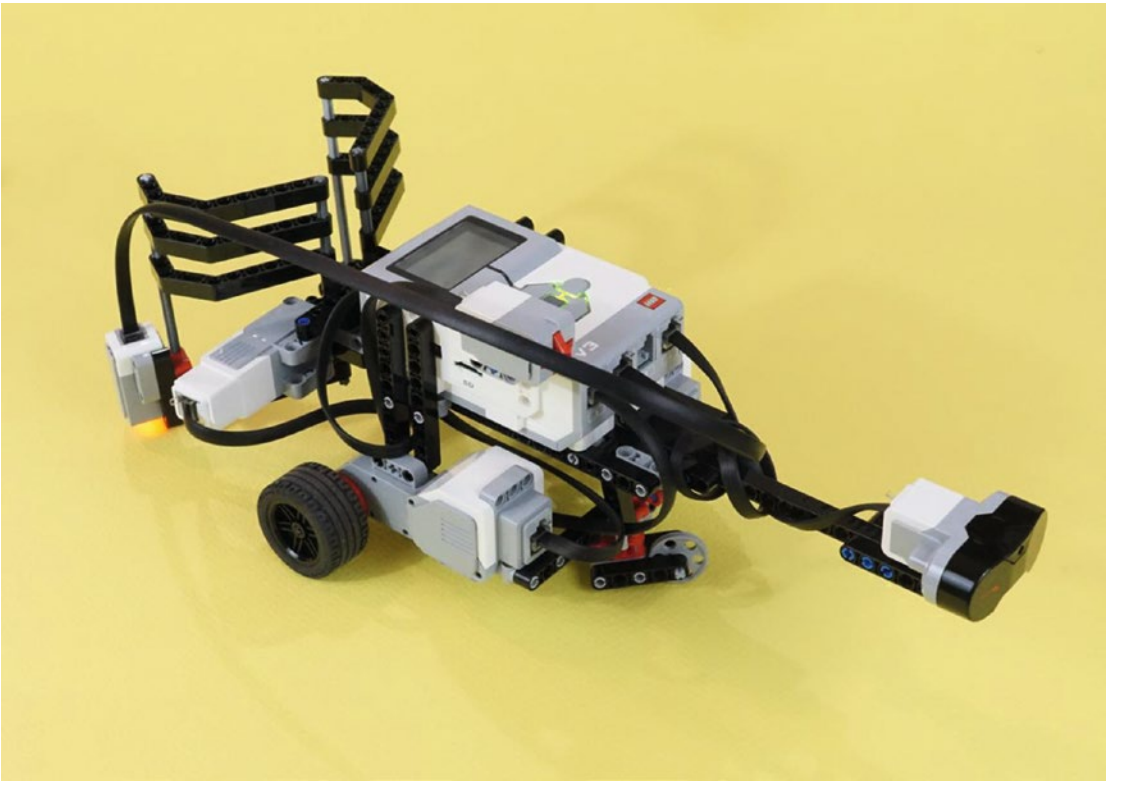


Figure 19-1. *The PushBot is a fun bot to experiment on. Your completed bot will have six wires in total ➤ three motors and three sensors.*

M. Bell and J. F. Kelly, *LEGO® MINDSTORMS® EV3,* DOI 10.1007/978-1-4842-2262-1_19

■ **Note** If you've built a unique version of the PushBot, please send me a picture. My web site address can be found in the Introduction near the beginning of the book.

Step by Step

I've divided the PushBot's building instructions into three sections. The first section covers the Jaw Cage mechanism with the medium motor. The second section shows you how to assemble the neck/Infrared Sensor assembly with its rear caster wheel, and the third section gives you the steps to build the main body of the PushBot, including the two main motors and drive wheels.

I'll continue to add comments for figures that might be a little tricky. And this next "Engineering" section is a sidebar that explains the thinking that goes in to creating instructions like these. If you want to read it later, that's okay! You can start building right now.

Engineering: The Difference Between Designing and Explaining

This chapter's description of how to build a PushBot needs to be as simple and straightforward as possible, so you can easily make a copy of this robot. But that is a different problem from *designing* a bot. In this case, I started the design by fiddling with the medium motor and a cage assembly, using the rounded gears at a 90-degree angle. I based it on an earlier design that used a large motor. But since I am using the retail EV3 kit, I have two large motors and one medium motor. So I started working on that problem first. I knew that designing a driving base would be fairly easy, but I had never designed a jaw assembly like this one.

"Designing with your hands" is a different mental process than sitting down with a piece of paper or design software on a computer screen. It's a different kind of thinking. So I picked up the medium motor, some axles, all four rounded gears, lots of pins, and a large frame piece. The main problem was how to convert the medium motor's "sideways-ness" into the "back-and-forth-ness" needed to work a pair of jaws. That is the design you see here. I made 15 or 20 changes as I worked through it. When you're doing this, sometimes a design seems to say, "This is elegant, makes good use of the materials, and solves the problem." The experience is as much esthetic as it is engineering.

But to create an explanation in this chapter, I took the completed robot and studied it on its own. I asked, "What's the easiest way to take this apart so someone else can understand it? And that meant dividing it into three or four sub-assemblies, each of which had its own story. (This kind of thinking is definitely accessible to a young person. I wrote my first instruction manual in second grade, when we got a new train set. We had an old train set that we were donating to charity, so I wrote a set of instructions how to hook it up.)

First Section: Figurine Jaw Cage/Medium Motor Mechanism

Figures 19-2 through 19-27 provide the steps for constructing the Figurine Jaw Cage/medium motor mechanism. Start with the motor and components you see in Figure 19-2. The completed mechanism is shown in Figures 19-25 through 19-27. It can be helpful to skip ahead to see what you're about to build.

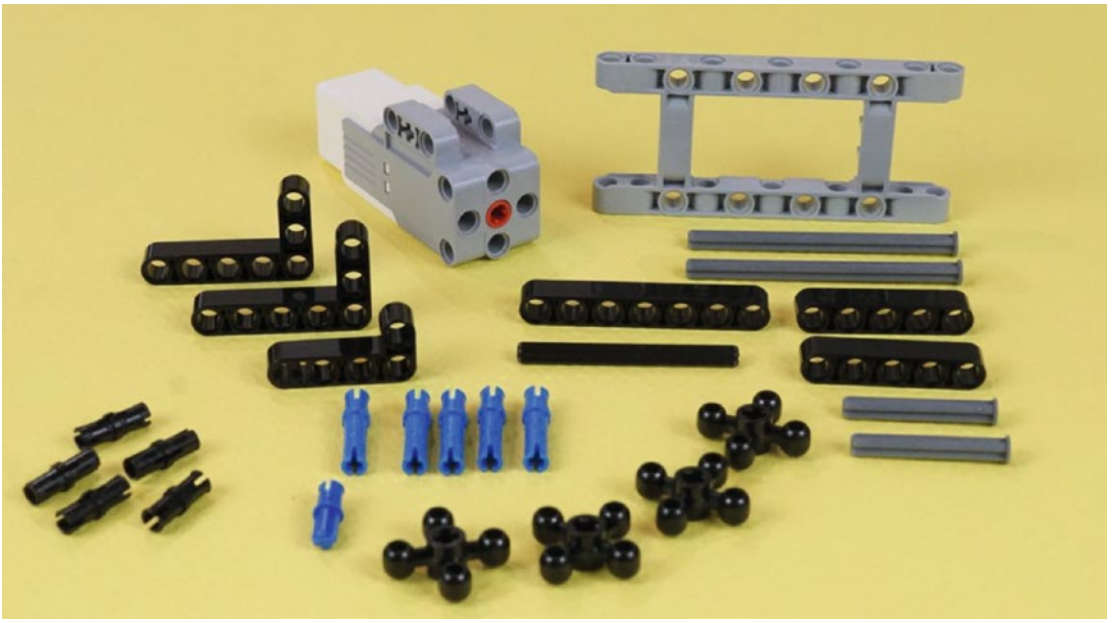


Figure 19-2. *These are all of the parts of the cage mechanism, except the jaws themselves. These parts will take you all the way through Figure 19-15. The long nailheads are eight holes long. The short nailheads are four holes long. The black axle is six holes long.*

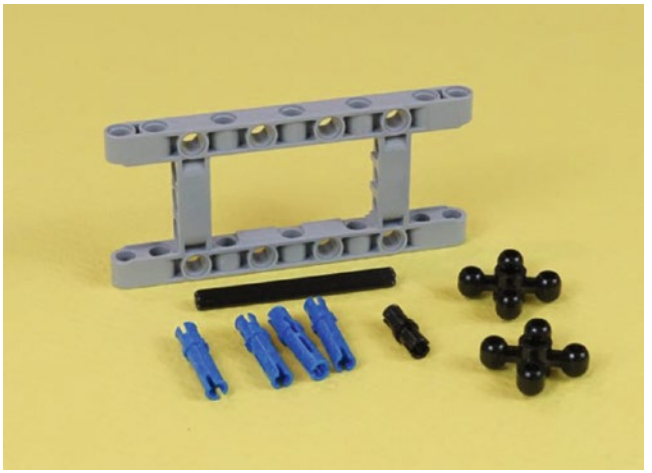


Figure 19-3. *Separate out this smaller group of parts. Connect them as shown in Figure 19-4. In that figure, the pins and axles are partially inserted to make them easy to see. This is the six-hole axle.*

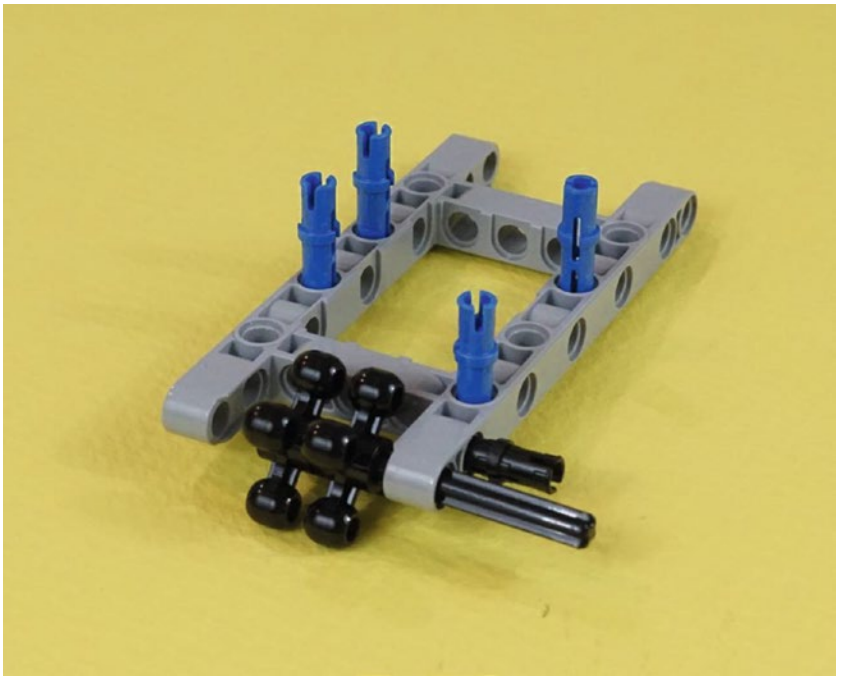


Figure 19-4. *In goes the axle, through the two rounded gears. Press it the rest of the way in, as shown in Figure 19-5. Also press the four long blue pins and one short black pin all the way in.*

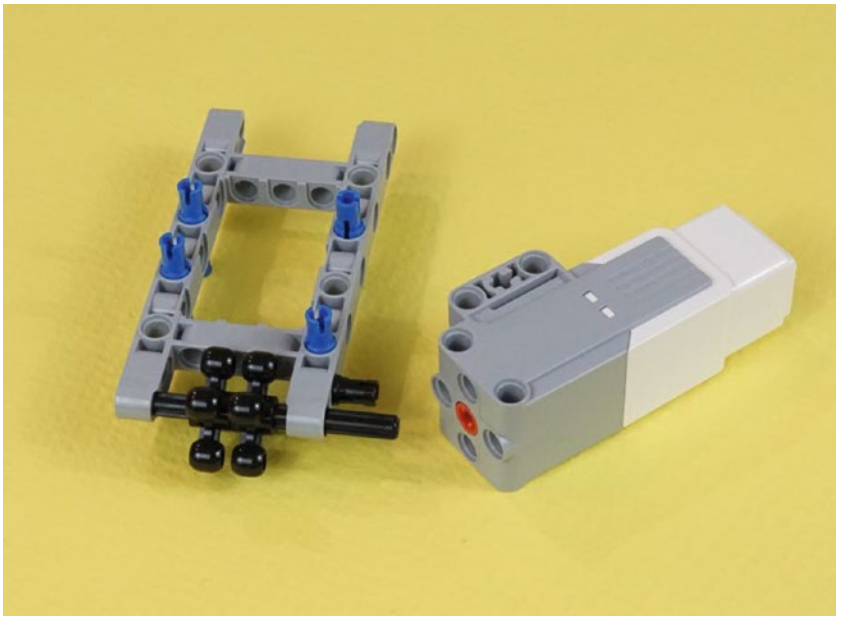


Figure 19-5. *Now let's get that Medium Motor mounted. The axle goes into the red motor hole, and that black pin engages the front motor mount.*

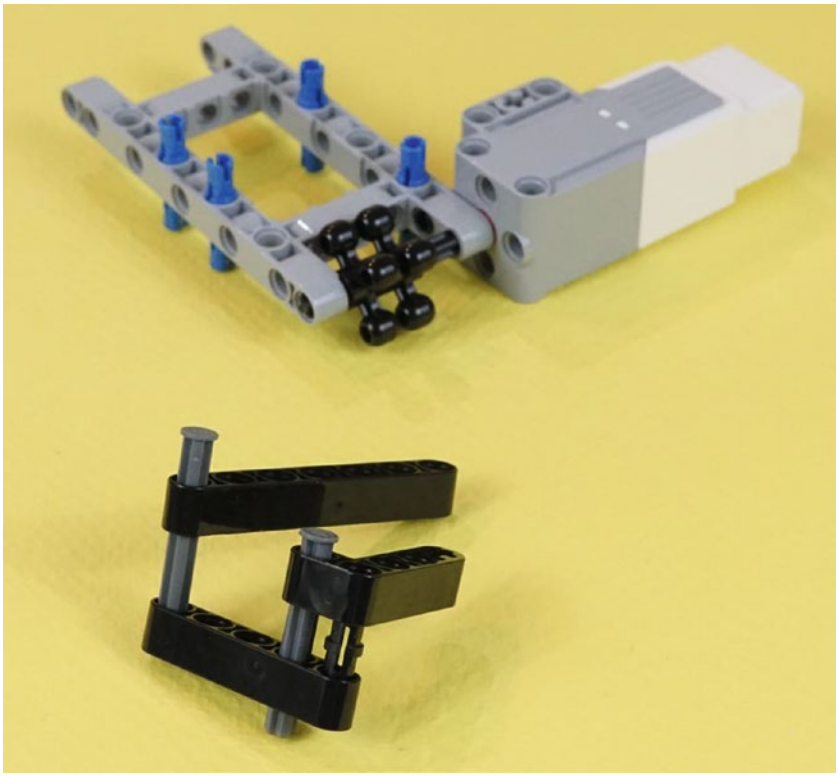


Figure 19-6. *More parts from Figure 19-2. A seven-hole beam, a five-hole beam, and a short L piece. Two four-hole nailhead axles and a black pin complete the bottom-axle sub-assembly.*

Press the nailhead axles all the way in and then put the sub-assembly underneath the motor-frame assembly. In Figure 19-7, we see them partially pressed into the long blue connectors. They are pressed all the way in when we get to Figure 19-8.

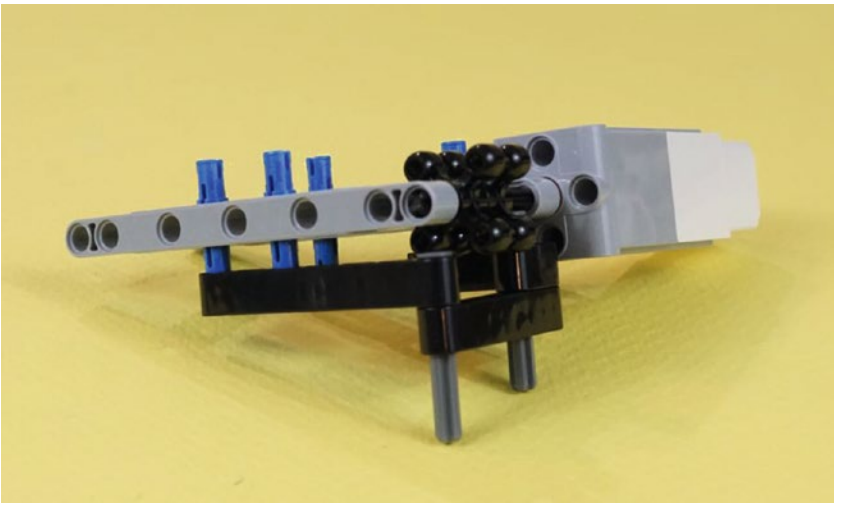


Figure 19-7. *The bottom-axle sub-assembly engaging with the motor assembly. Press the nailhead axles all the way in. In this view, the bottom-axle sub-assembly is pressed part way onto the long blue pins.*

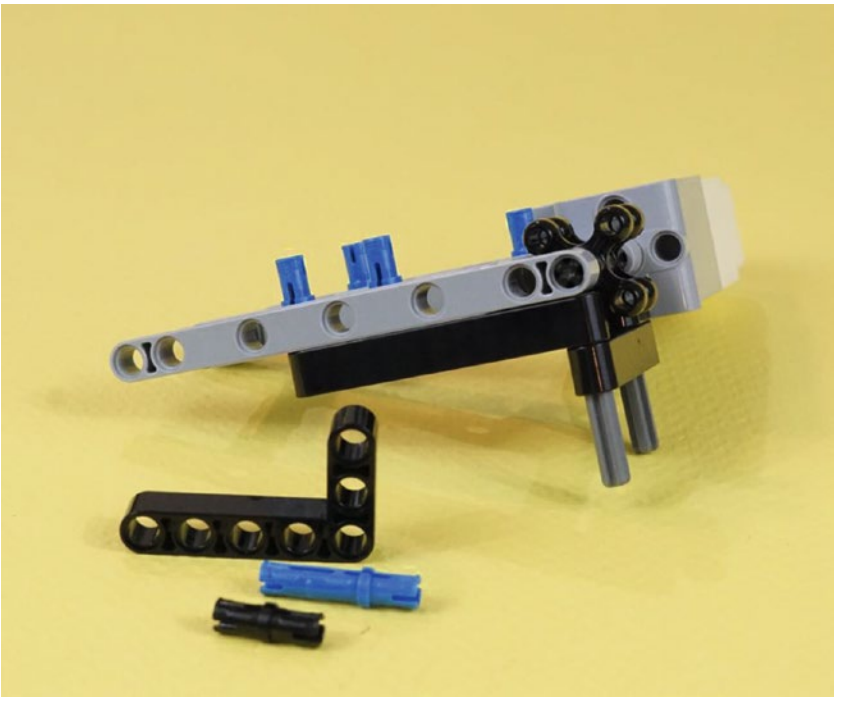


Figure 19-8. *The bottom-axle sub-assembly in Figure 19-7 is pressed all the way onto the blue pins. Three more parts from Figure 19-2 ➤ a black short pin, a long blue pin, and a large L-beam.*

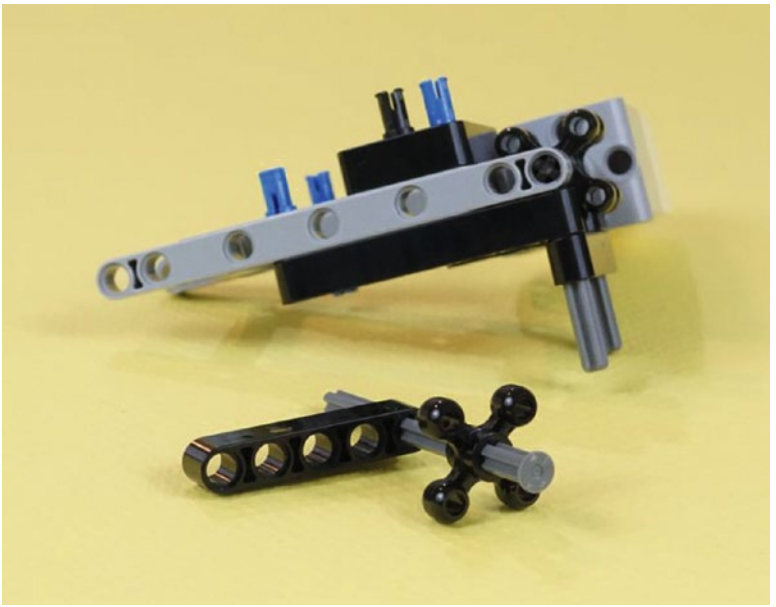


Figure 19-9. *The corner of the large L-beam engages the blue pin as shown. The black pin goes in just to the left of the blue pin. The long side of the L-beam points away from you. The three-hole short side is directly facing you. Pressing this sub-assembly down gets you to Figure 19-10.*

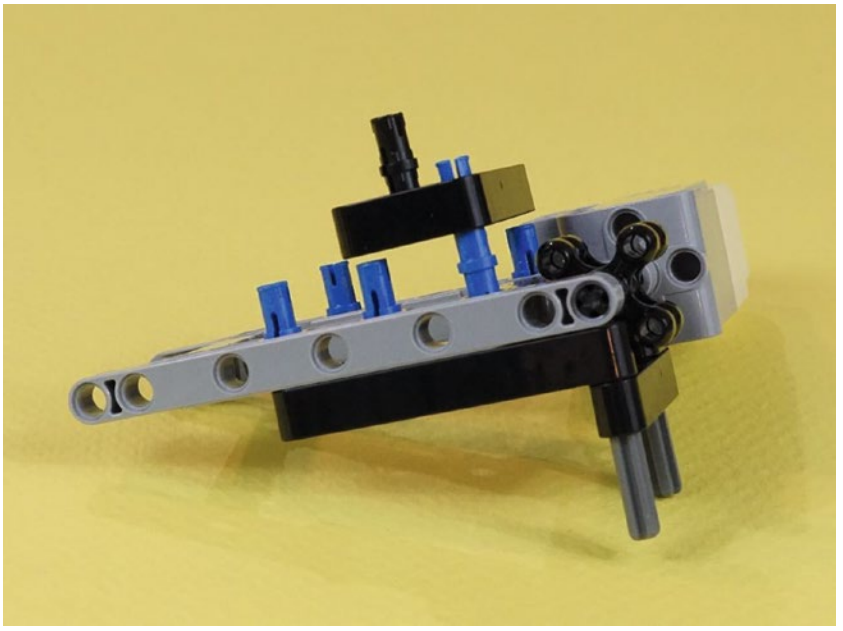


Figure 19-10. *Three more pieces from Figure 19-2's parts collection. A long nailhead axle, five-hole beam, and a rounded gear. Press the axle all the way into the gear and beam.*

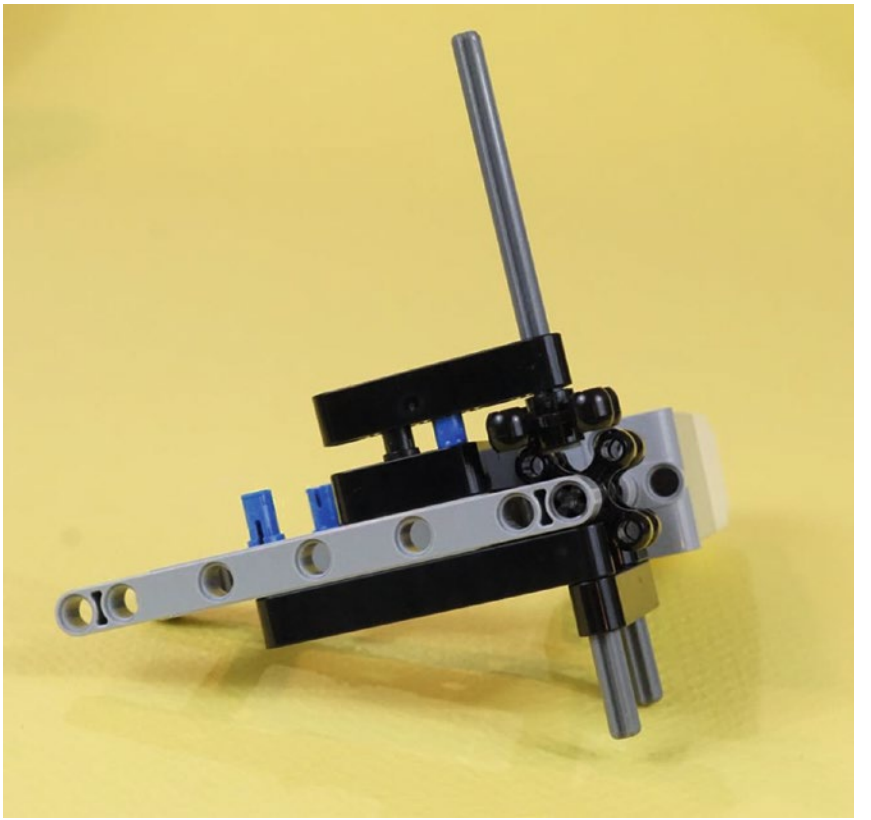


Figure 19-11. *Now set the five-hole beam with its axle assembly on those two available pins, as shown. You'll press them down in Figure 19-15.*

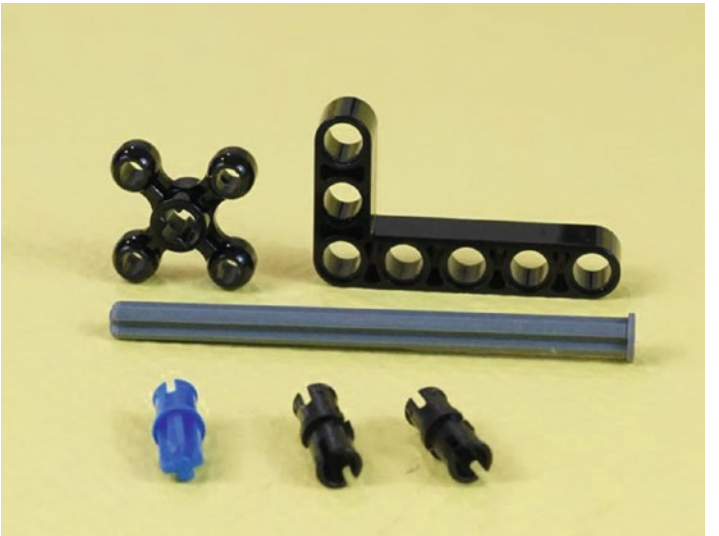


Figure 19-12. *Gather these last six parts from Figure 19-2. They go together as shown in Figure 19-13.*

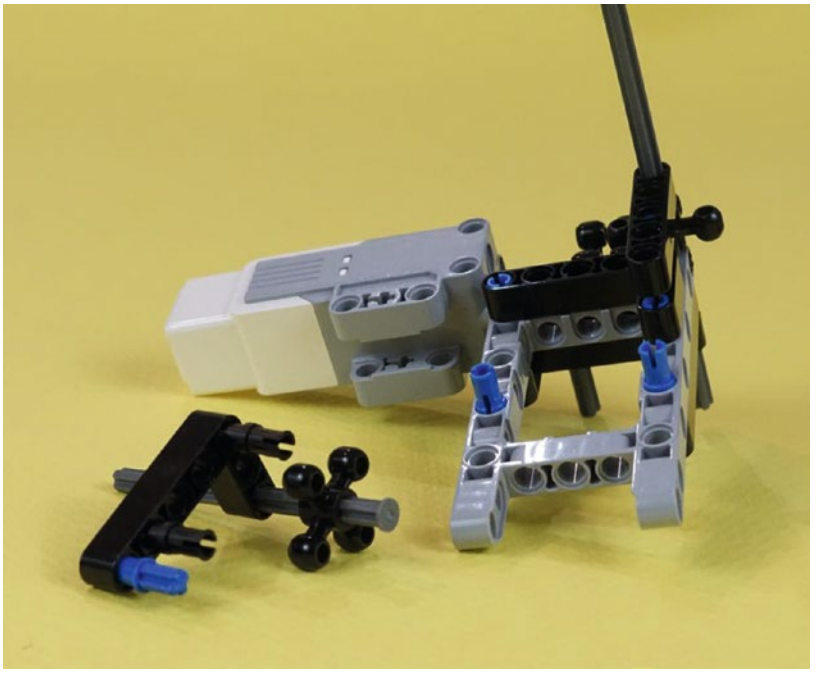


Figure 19-13. *The final six parts shown partly inserted. In Figure 19-15, you'll press them all the way in, including the nailhead axle.*

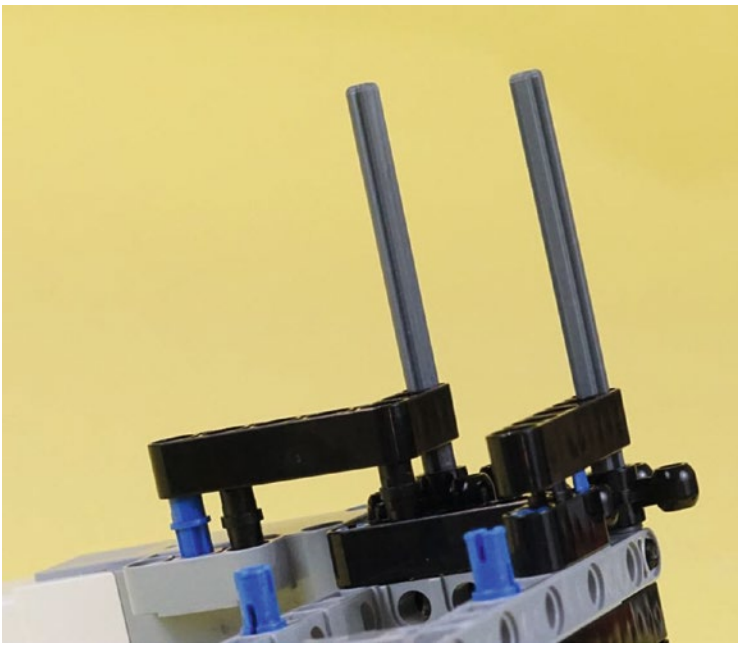


Figure 19-14. *Close-up of how the large L-beam with axle and gear is placed on the medium motor and the other frame pieces. This L-beam and the five-hole axle assembly from Figure 19-11 are now ready to be pressed all the way down.*

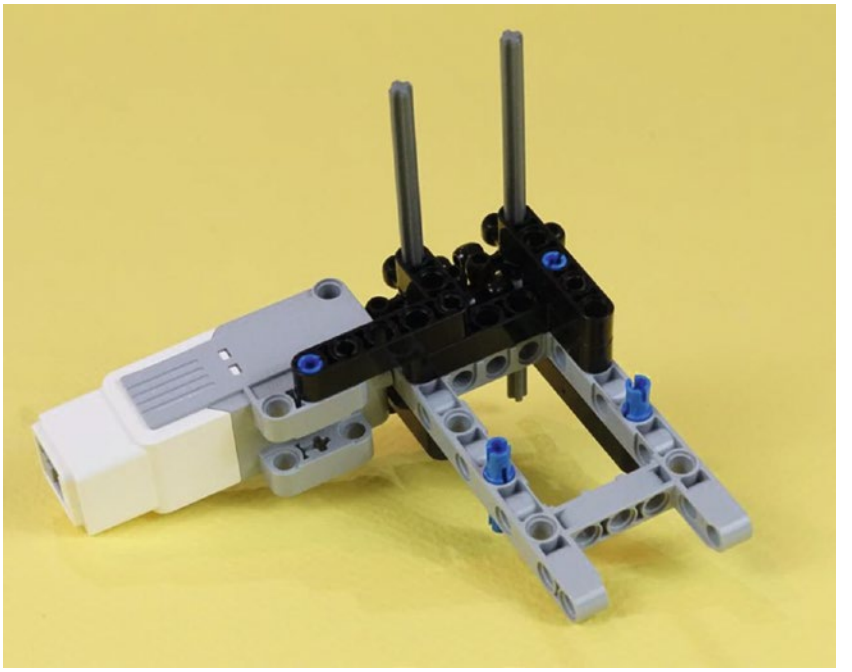


Figure 19-15. *Press the beams all the way down. This medium motor assembly is ready for some Figurine Gripper Jaws.*

Engineering and the Concept of Stiffness II: Angles, Brackets, and Connectors

You are now at an interesting point. That last L-beam, pressed firmly down on the medium motor assembly, did a lot. It finished the positioning of the gears, made the second long nailhead axle secure, and very usefully added two solid connectors to the medium motor. Before, that motor was held in only by the axle attached to the gears, along with one short pin. You now have a sturdy assembly that can support the jaws and the force of the motor itself. Figure 19-16 shows another view of this assembly.

It's worth spending a moment on how this piece is engineered. The angle pieces are the key to the strength of the assembly. Try moving the axles and gears back and forth. You'll see how sturdily they are mounted to the medium motor. The L-shaped pieces give you strength in two dimensions, and *every* hole that can be connected to a pin has one. Long blue pins are used as often as possible, further increasing the stiffness and extending the design to three dimensions. If you take a close look at all of the straight beams and L-beams, you'll see that any set of holes that can have a long blue pin has one. All remaining open holes have a black pin.

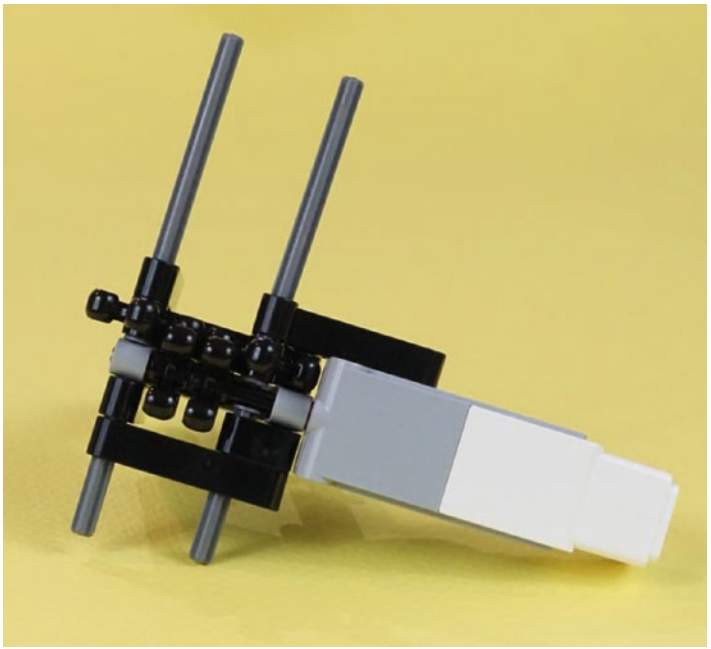


Figure 19-16. *The medium motor assembly seen from another side. Bring on those Figurine Gripper Jaws!*

We're going to collect two more batches of parts to finish the last two Figurine Jaws sub-assemblies. The first batch, shown in Figure 19-17, gives us the jaws themselves. The second batch will attach the Color Sensor to the edge of the left jaw. So, let's make those jaws. As LEGO would have it, we are well provided with some jaw-like pieces of just the right size. Those, along with some axles, will give us a handsome "jaw bite."

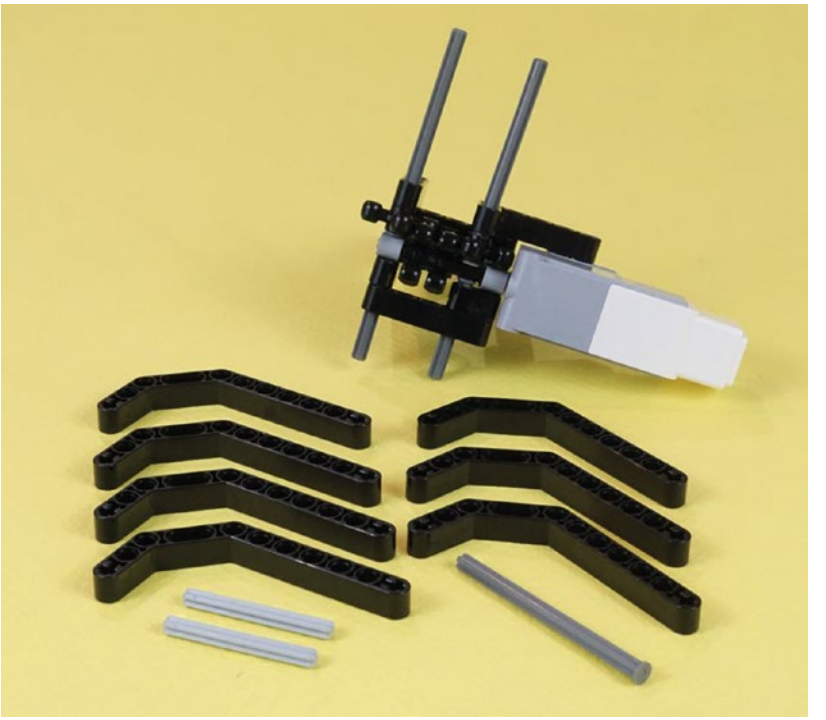


Figure 19-17. *These ten new parts will start the Figurine Jaw Cage sub-assembly. Notice the long axle is an eight-hole nailhead. The two short axles have five holes.*

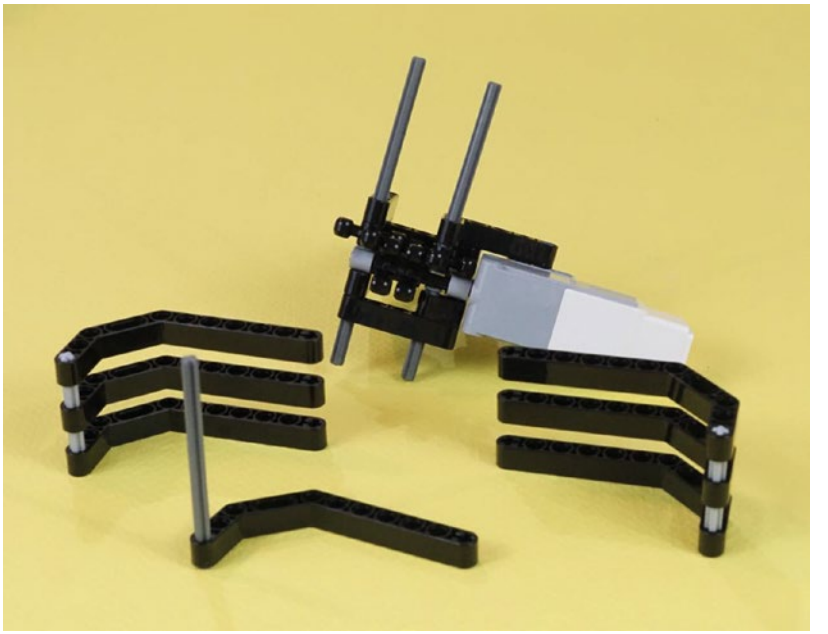


Figure 19-18. *The ten parts are partially assembled. The head of the nailhead axle is below the long angled beam.*

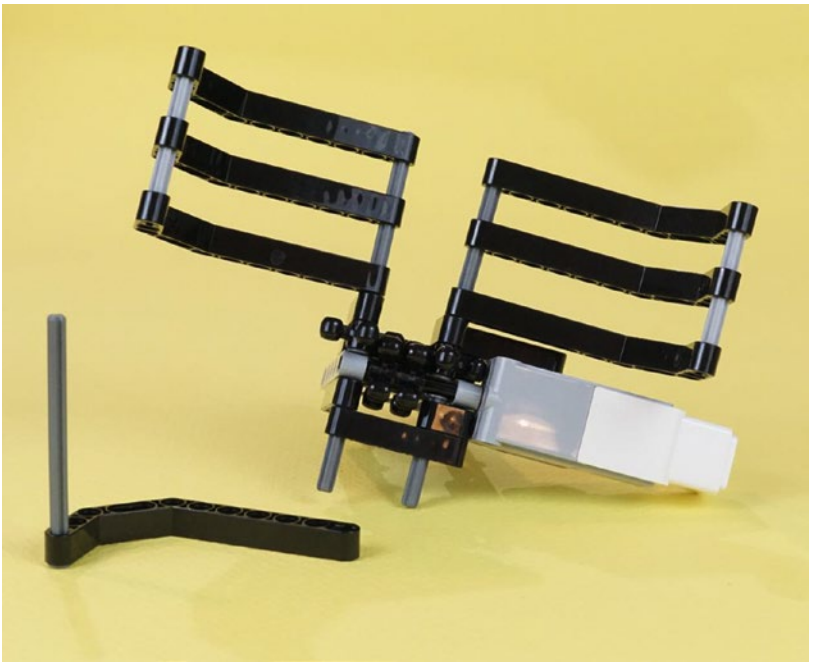


Figure 19-19. *Put the two jaw assemblies on the long nailheads at the top. This is starting to look like a Figurine Jaw Cage. More than halfway there…*

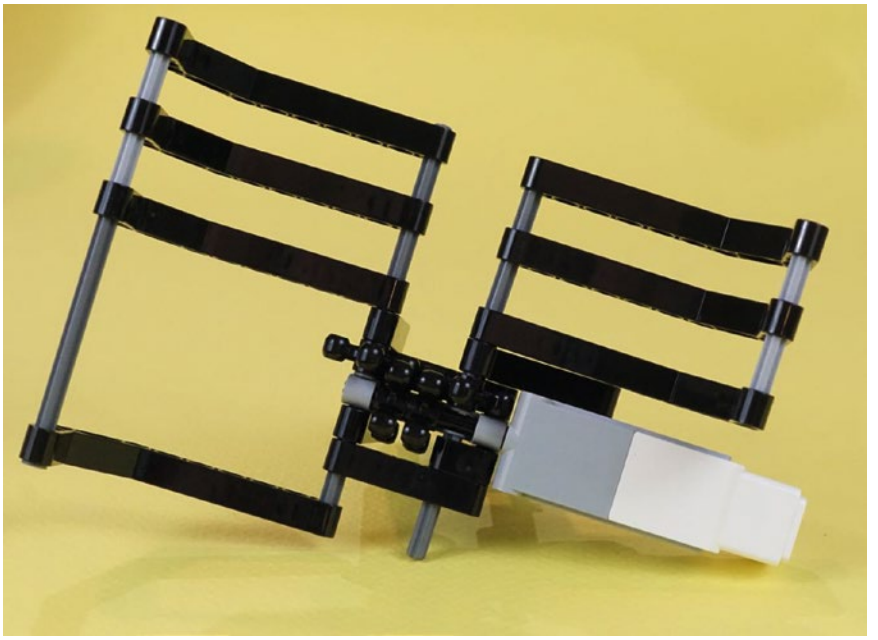


Figure 19-20. *Attach that lower jaw piece. Push a little of its nailhead axle into the upper jaw piece. Now the Figurine Jaw Cage is three quarters done. Let's gather a few more parts to mount the Color Sensor.*

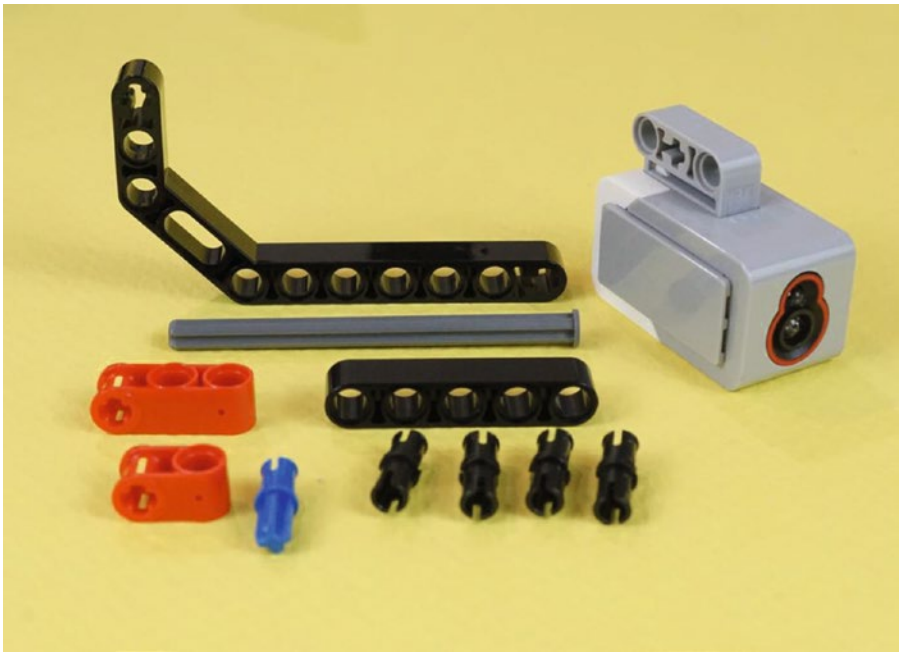


Figure 19-21. *These 11 new parts will let you complete the Figurine Jaw Cage/medium motor section, with the Color Sensor attached. That is an eight-hole nailhead axle.*

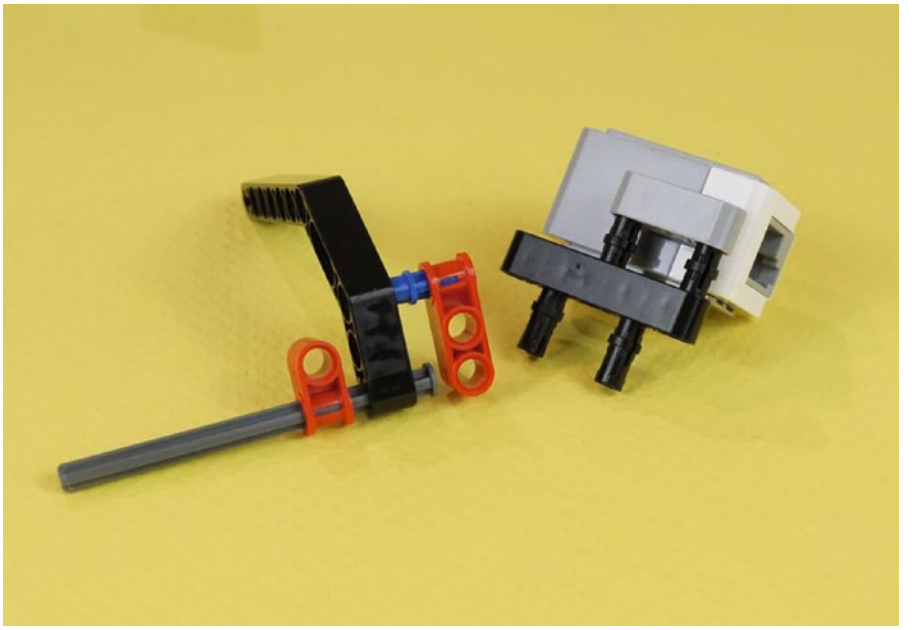


Figure 19-22. *Make these two sub-assemblies as shown. Then, squeeze the red parts together so they are centered on the nailhead axle.*

As you squeeze the red connectors together, their side holes will wind up at the correct distance to connect with the five-hole beam attached to the Color Sensor. You can see this in the close-up of Figure 19-23.

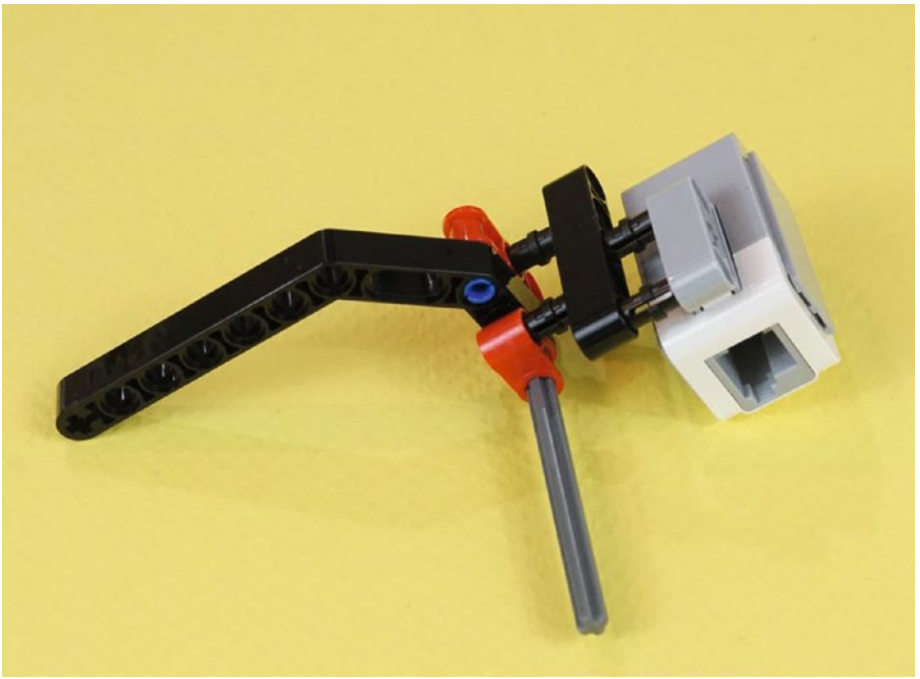


Figure 19-23. *This last sub-assembly is ready to have the pins pressed into their holes. The completed Color Sensor mount is shown in Figure 19-24.*

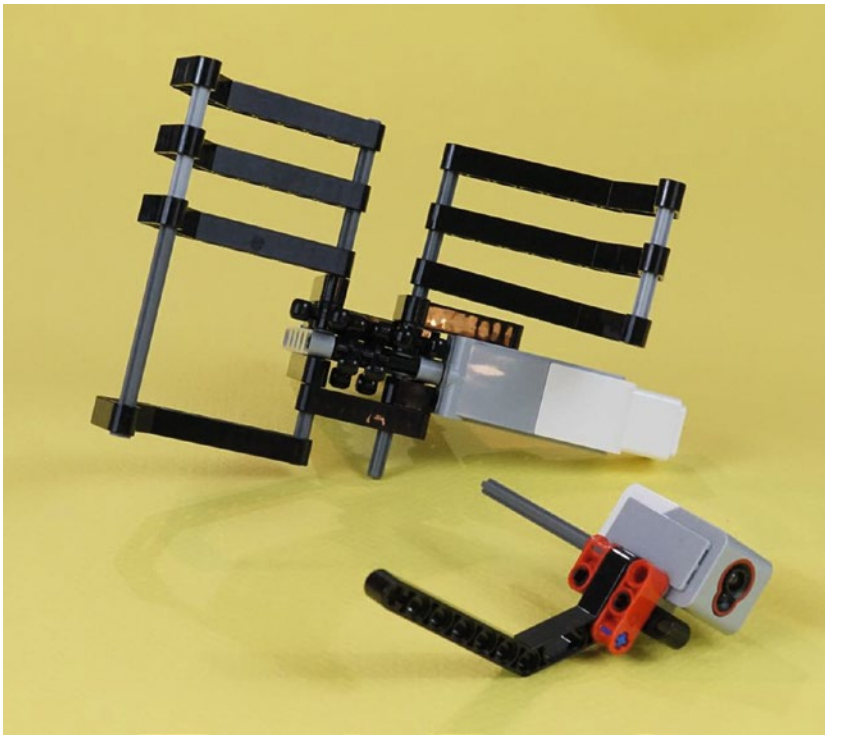


Figure 19-24. *The Color Sensor mount, ready to join up with the Figurine Jaw Cage*

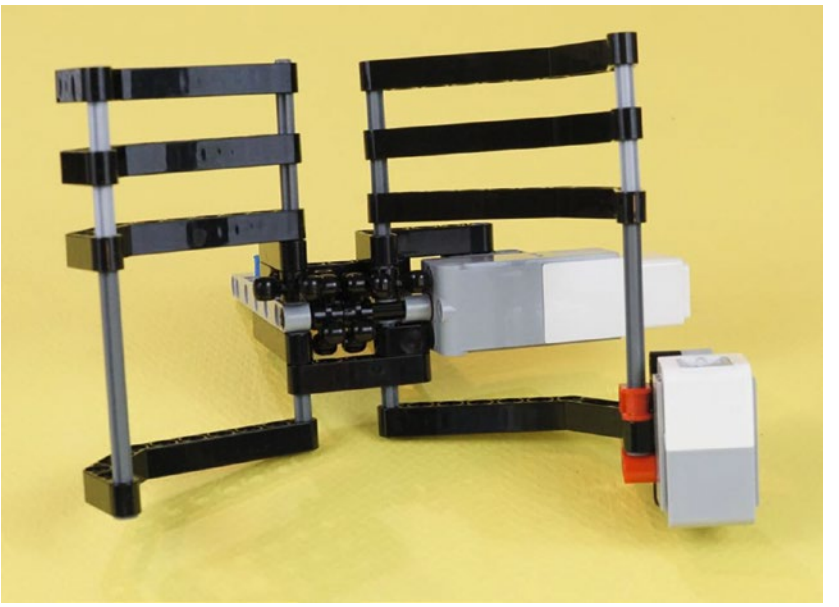


Figure 19-25. *The completed Figurine Jaw Cage, shown from the front. Press the long nailhead a little way into the upper angle piece and wiggle the jaw pieces to make them stable and symmetrical.*

This is a relatively complex assembly. It can be helpful to have other views, as you can see in Figures 19-26 and 19-27. When you've played with the motor and gears, and generally basked in the satisfaction of having made a cool part, set it aside and commence working in the second section. You will build the Neck/Infrared Sensor assembly with a small caster wheel for the rear.

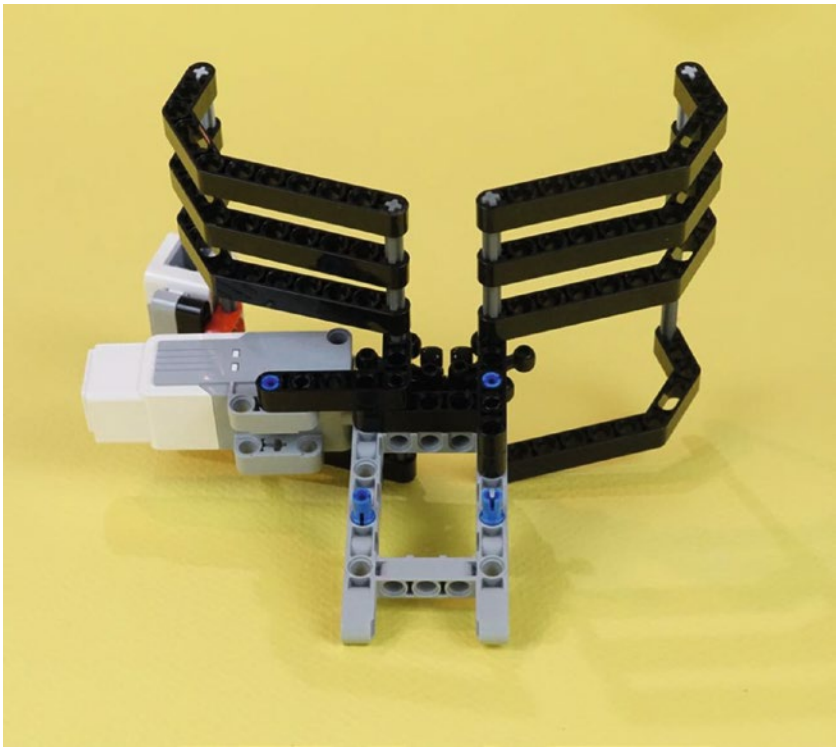


Figure 19-26. *The completed Figurine Jaw Cage, seen from the rear*

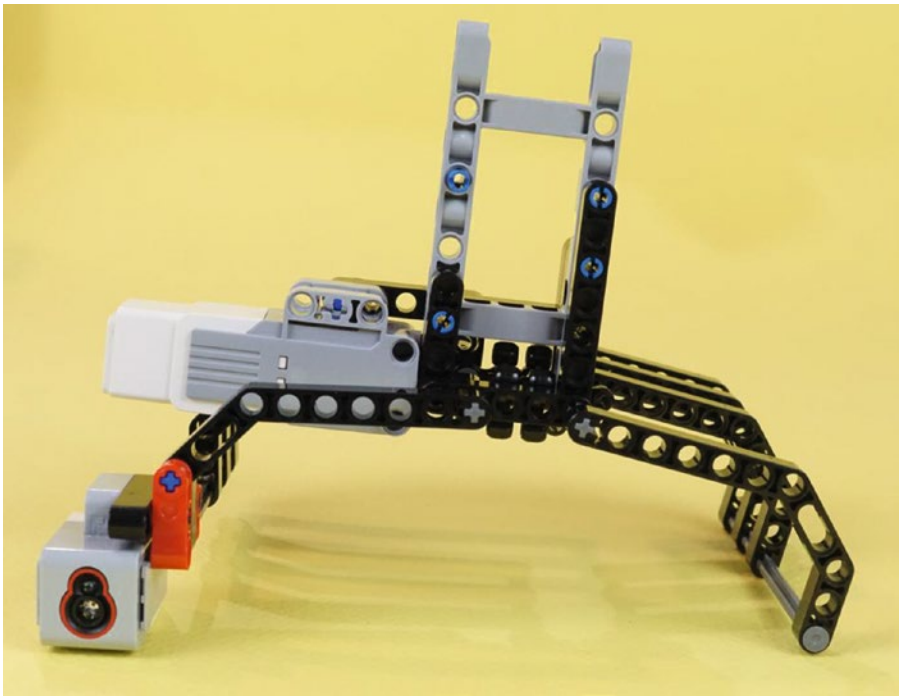


Figure 19-27. *The completed Figurine Jaw Cage, seen from the bottom*

Second Section: Neck/Infrared Sensor Frame Assembly with Caster Wheel

Figures 19-28 through 19-50 demonstrate the construction of the rear caster wheel and the Neck/Infrared Sensor Frame assembly. This caster wheel is similar to the one used by our ExploroBot in Chapter 3. The idea of a caster is that it can swivel easily and will not alter the robot's course as determined by the two main drive wheels.

Engineering: Rear-Wheel Designs and Their Constraints

There are other ways to make a rear-wheel. You can get a hard plastic ball, say, a little larger than a ping pong ball, and simply set it underneath a few frame pieces so it won't roll out. That works surprisingly well, since a ball will turn any number of degrees you want, right away. It effectively has a 360-degree "swivel." Another approach is to use the "peg-leg" as the SnapShotBot uses in Chapter 11. Peg-legs are particularly easy to design but they do require a rather smooth floor.

A caster will work on smooth floors and will be okay on a slightly irregular surface as long as the bumps aren't too big. If you're running on carpet, you'll need tank treads (see the GrabberBot in Chapter 15) or a ball of two or three inches in diameter.

Figure 19-28. *These are all of the parts you need for this section. You'll have the caster wheel and an important part of the frame. The axles are five-hole and three-hole lengths.*

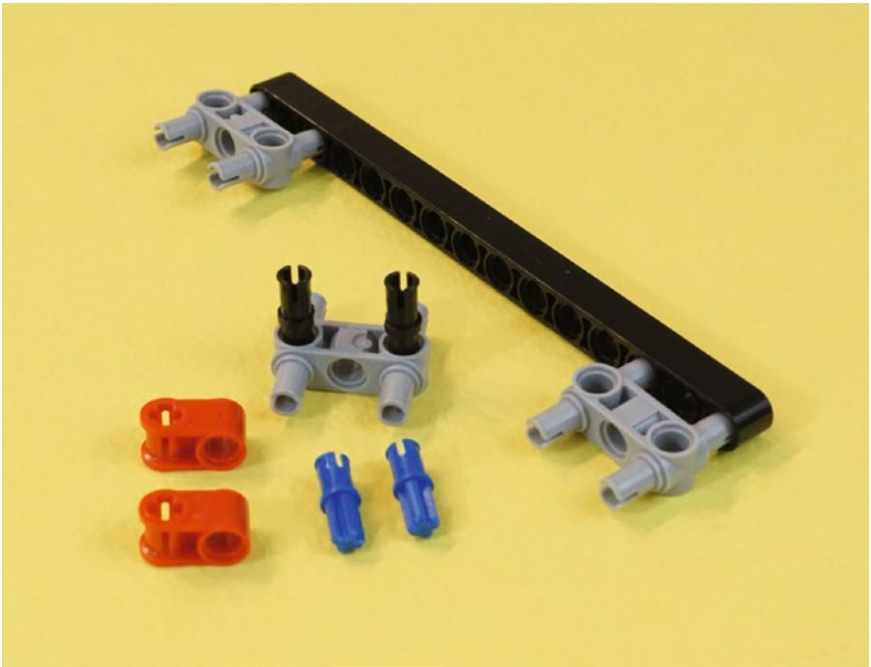


Figure 19-29. *Start with these parts. Note the partial assembly of these three double connectors.*

Press all of the parts all the way in. Flip the 15-hole beam over, to the orientation seen in Figure 19-30. Press the center gray double connector in place, in holes 7 and 9. That will center it on the 15-hole beam, attached by the two black pins. Then insert the blue half-axles into the red pieces. Place those on top of the center gray connector as shown. Press everything together, then set this piece aside for a moment as you build the caster wheel.

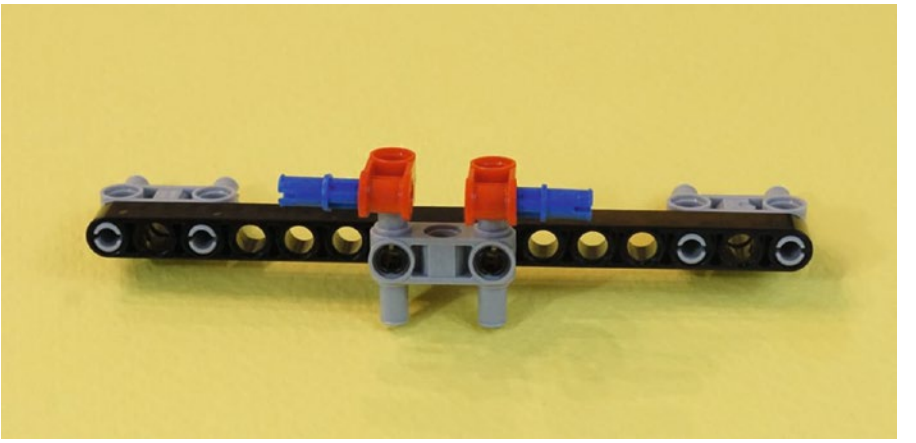


Figure 19-30. *Insert the blue half-axles into the red pieces, place them on top of the gray double connector, and press them all together. A completed view is shown in Figure 19-34.*

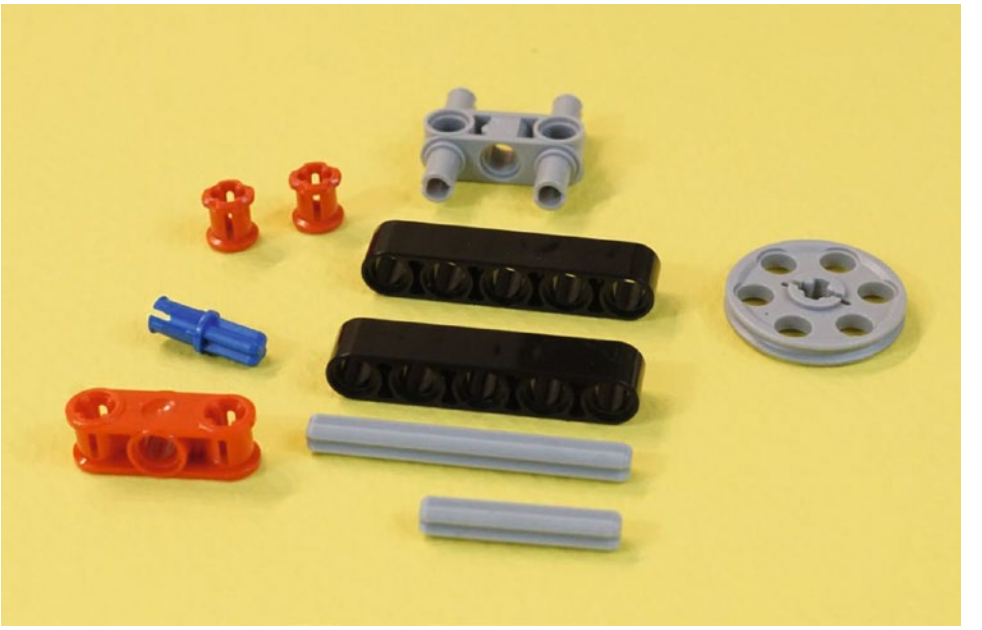


Figure 19-31. *These are the pieces remaining from Figure 19-28. Now to make the caster wheel.*

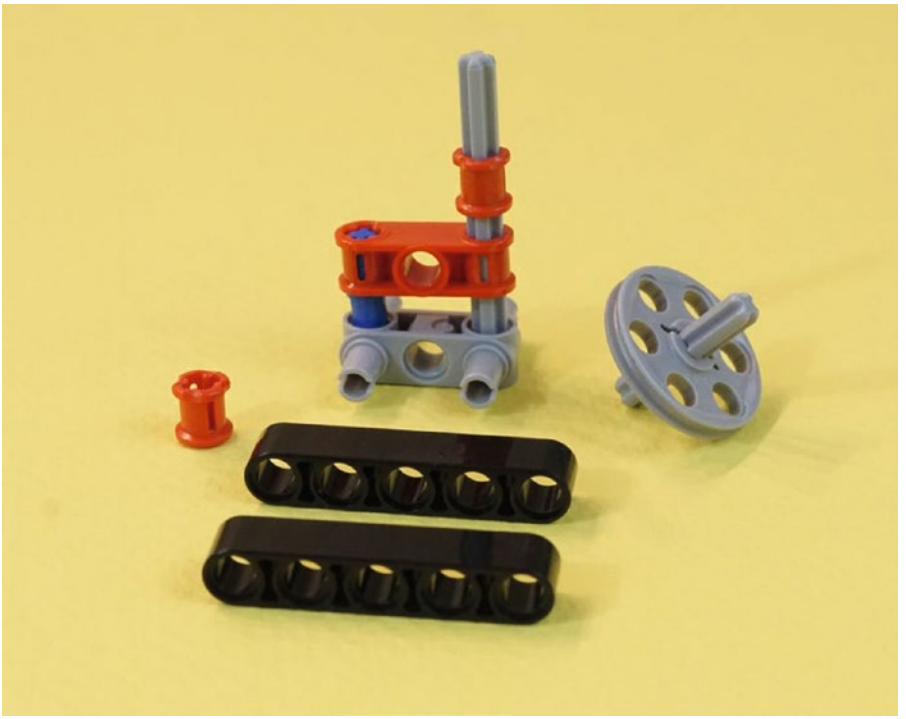


Figure 19-32. *The five-hole axle stacks into the gray double connector, with the red part, blue half-axle, and red collar. The short three-hole axle goes into the wheel. Press the red parts all of the way down onto the five-hole axle.*

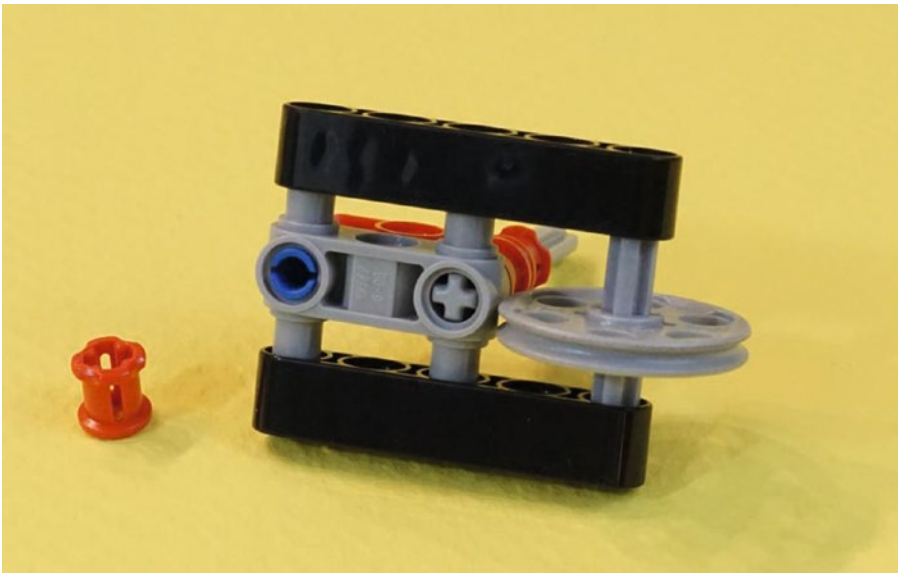


Figure 19-33. *The two five-hole beams make a little wheel frame. Press them together so the caster wheel assembly looks like the foreground of Figure 19-34.*

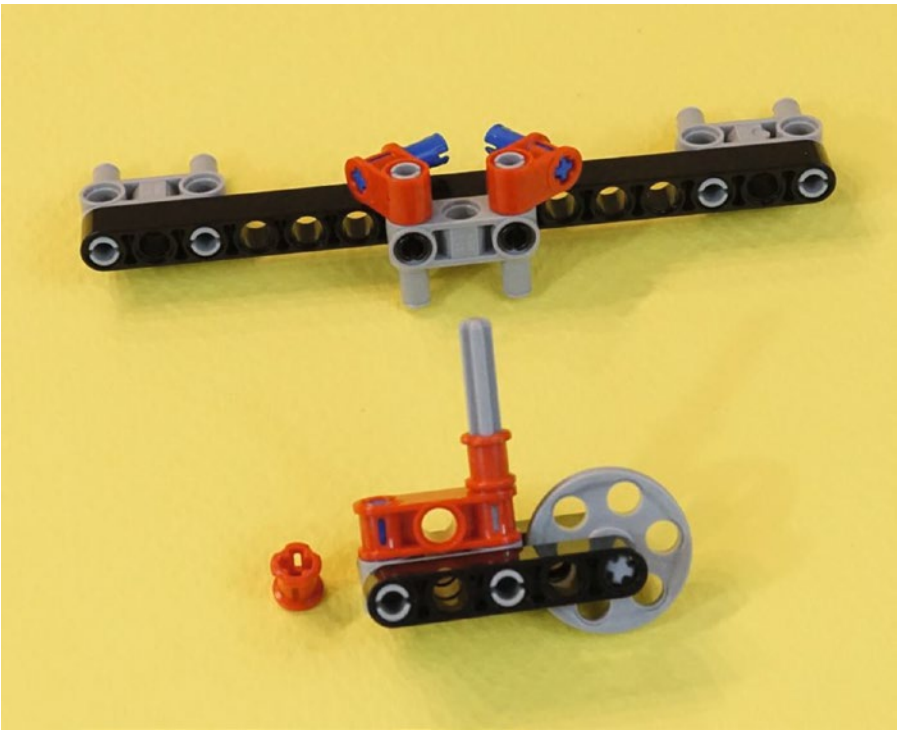


Figure 19-34. *All of the parts are pressed together. The axle will go in the center of the middle gray connector.*

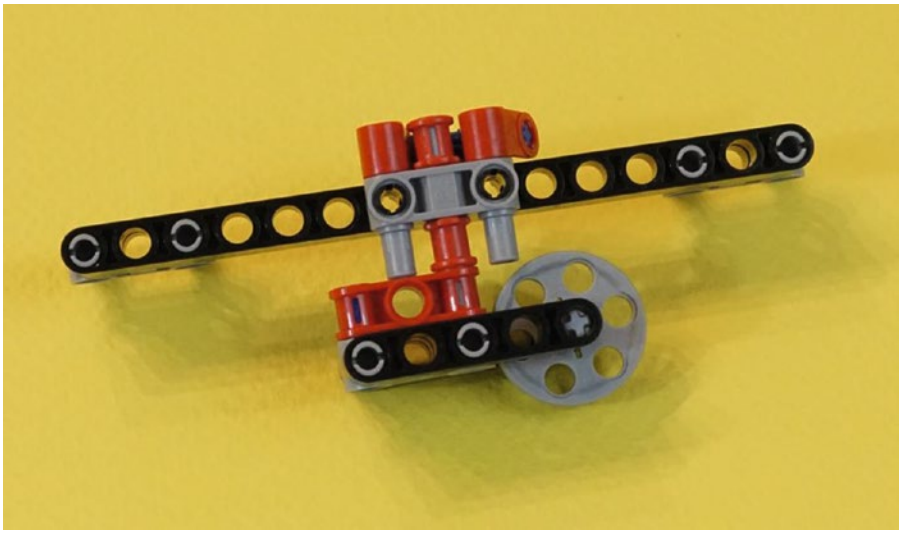


Figure 19-35. *The caster wheel is in place, held in by that remaining red collar. It will swivel freely as the robot makes its turns.*

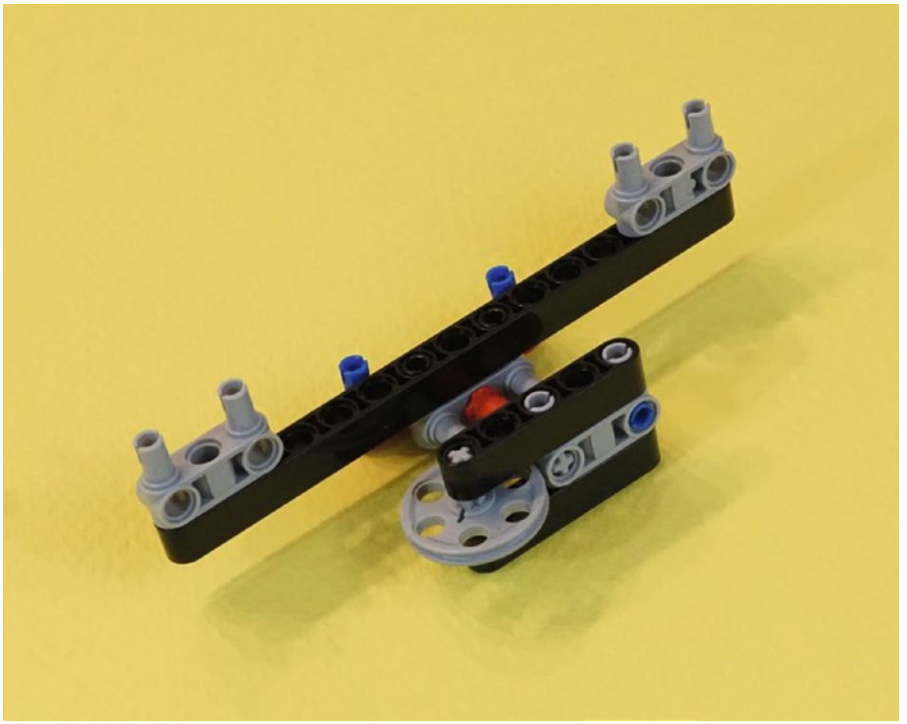


Figure 19-36. *Another view of the completed caster frame assembly. Set it aside until Figure 19-47, so you can complete the rest of the neck assembly.*

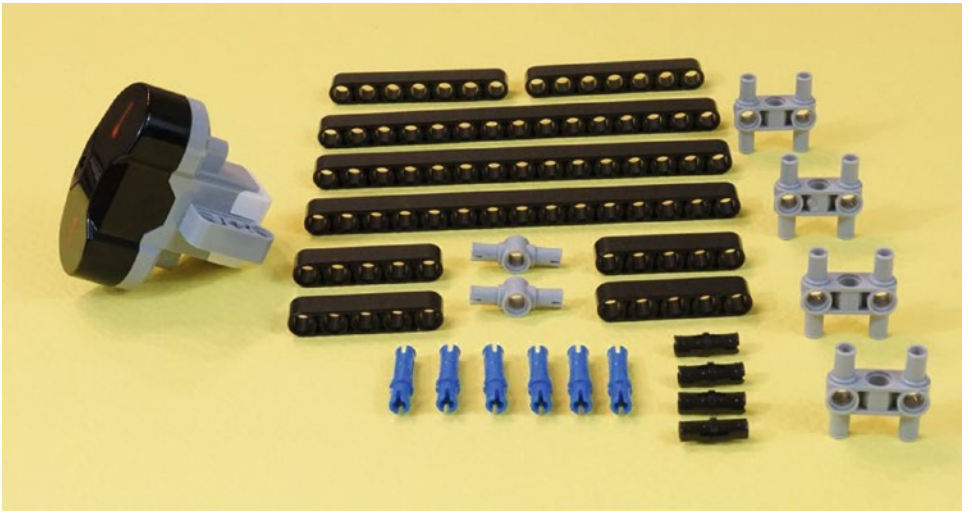


Figure 19-37. *Gather all of these parts for the remainder of the Neck/Infrared Sensor assembly*

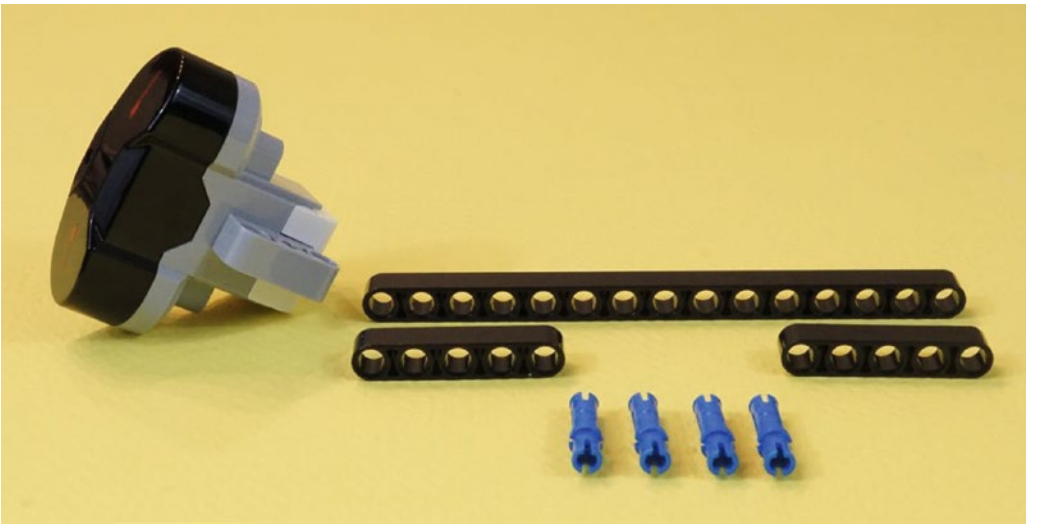


Figure 19-38. *From the parts in Figure 19-37, select this 15-hole beam, two five-hole beams, four long blue connectors and the Infrared Sensor.*

Figure 19-39. *Begin the assembly of the head. These four connectors will fasten the two five-hole beams to the 15-hole beam.*

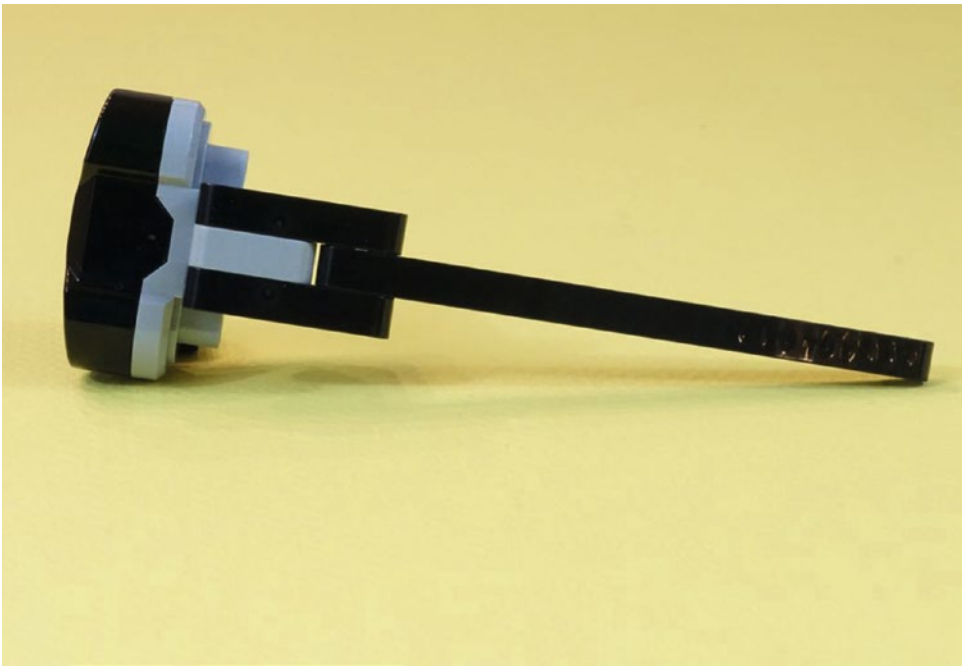


Figure 19-40. *Push the connectors together, and your Infrared Sensor neck and head are ready for the next step*

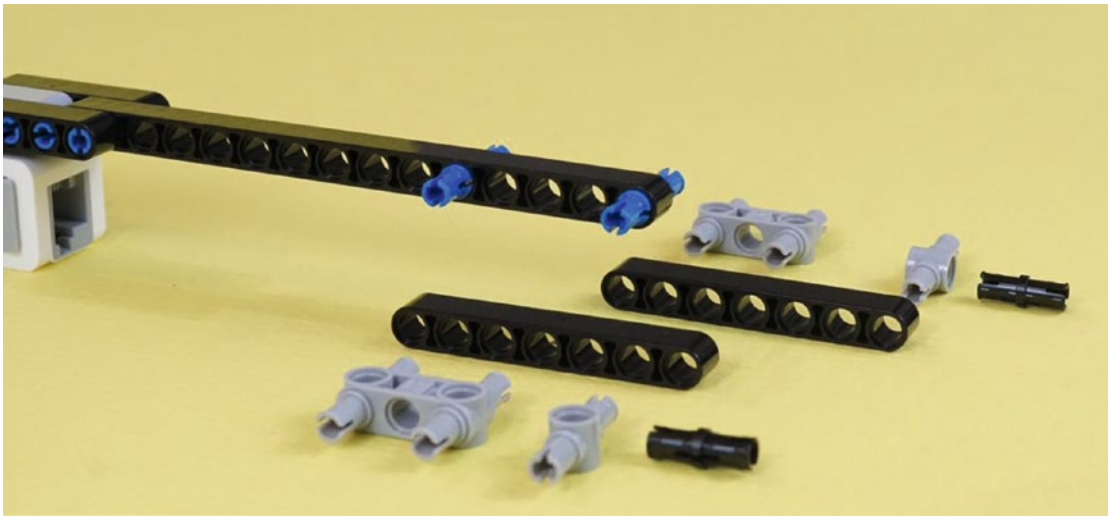


Figure 19-41. *From the remaining parts in Figure 19-37, take two long blue connectors, two seven-hole beams, and the other parts shown here. Insert the long blue connectors into the 15-hole beam as shown.*

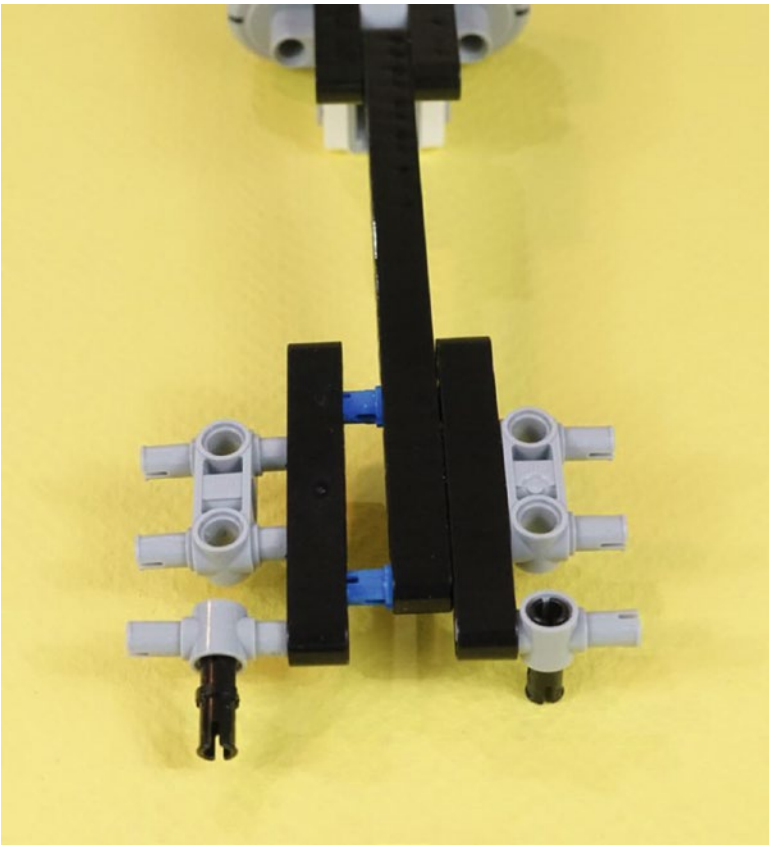


Figure 19-42. *Assembling the Infrared Sensor neck. Press the pieces together, which will then look like Figure 19-43.*

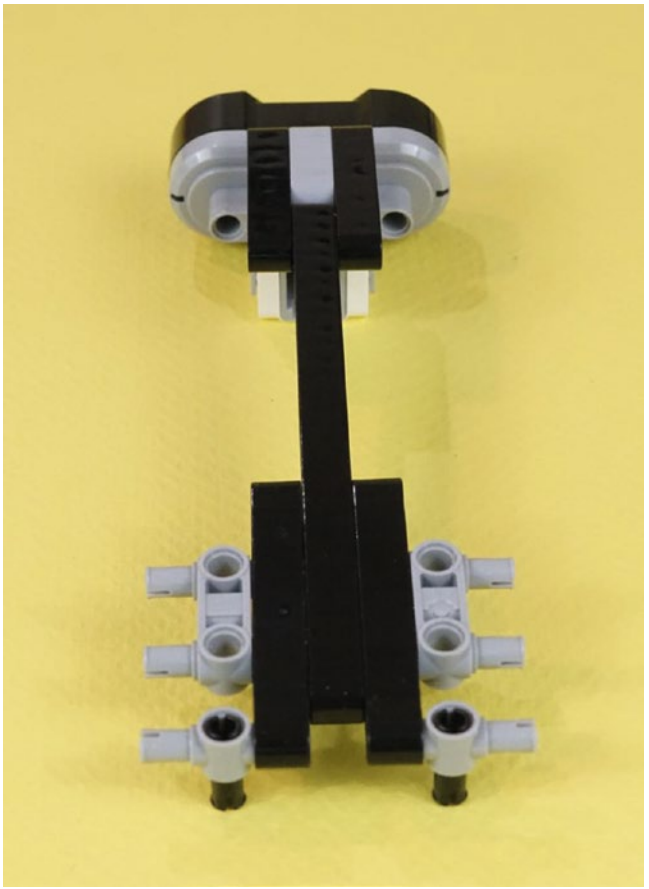


Figure 19-43. *The assembled Infrared Sensor neck*

Let's take a look at all of the parts so far. In the middle of Figure 19-44, we see the completed Infrared Sensor neck assembly. Behind that is the caster wheel assembly. In the foreground are the remaining parts from Figure 18-37. They are the two 15-hole beams, two five-hole beams, two gray double connectors, and two black pins. As they are assembled, they too will be symmetrical, as are most parts in this bot.

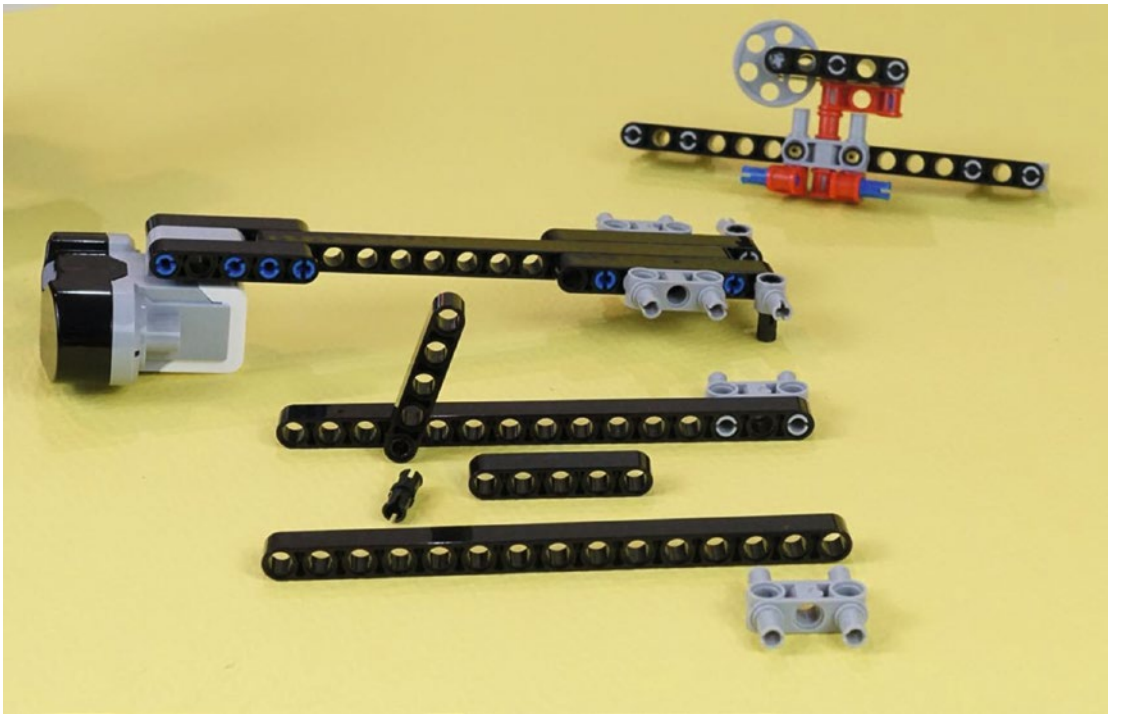


Figure 19-44. *The completed neck and caster wheel, ready to go. In the foreground, you'll use the remaining parts from Figure 19-37. Begin to assemble them as shown, so you get two long support assemblies. The loose pieces in the foreground will assemble to a mirror image of the piece behind them.*

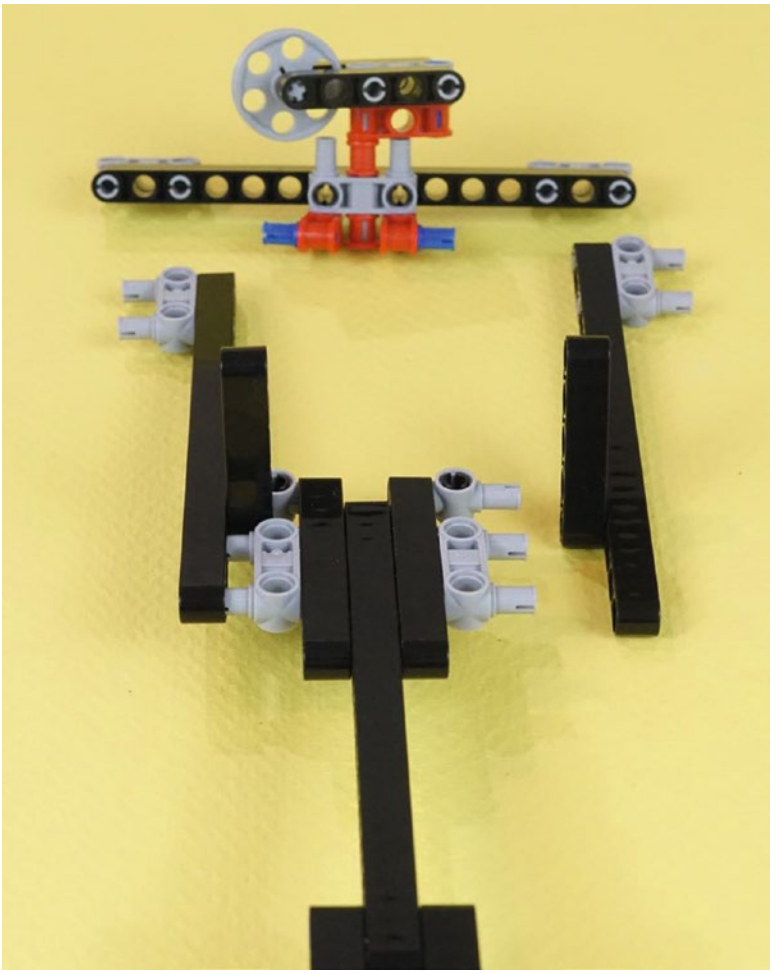


Figure 19-45. *Fasten the two long support assemblies to the Infrared Sensor neck assembly. The five-hole beams point up as shown, and the two black pins at the back of the neck point down.*

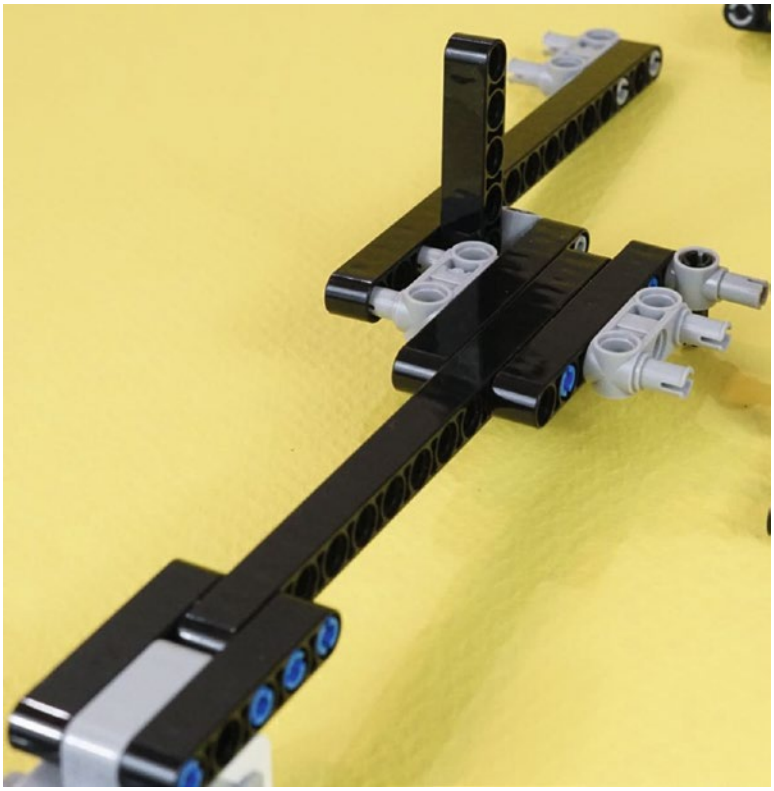


Figure 19-46. *A side view of the same pieces as shown in Figure 19-45. The five-hole beam fits in between the double gray connector and the end gray connector. Press this left side all the way in.*

This next step, shown in Figure 19-47, is one of those steps that "gets interesting." No new parts coming in. Instead, we're connecting parts that have been hanging around for a while. This is usually one of the more satisfying steps in building a bot using directions. You have all these partially built pieces with connectors sticking out. They need a place to go! And in Figure 19-47, it's happening. Then, the completed piece is shown in Figures 19-48, 19-49, and 19-50 ↓ three views of the completed piece, from the back, bottom, and side.

Multiple, completed views can be very useful as you build. For one thing, you might skip ahead to see what you're building. Secondly, they're handy if you're ever thinking, "Hang on, how was this step supposed to work?" You can double-check your assembly steps by referring to the different views.

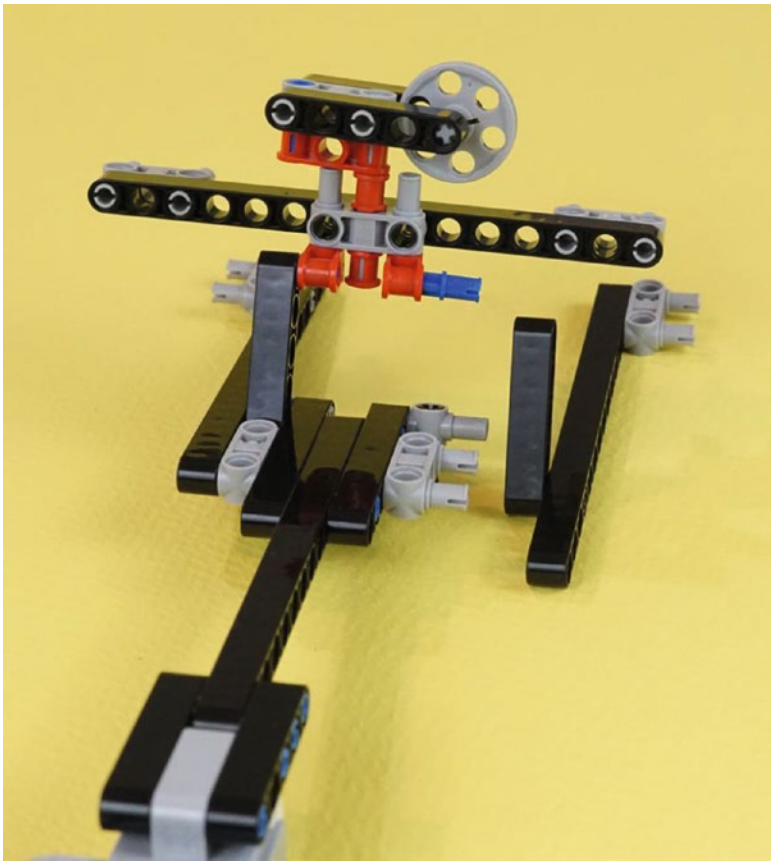


Figure 19-47. *Now that caster frame piece joins in! Press the right-side long beam into the neck and caster assembly in the same manner as the other side. Figure 19-48 shows a view of the completed item from the back side.*

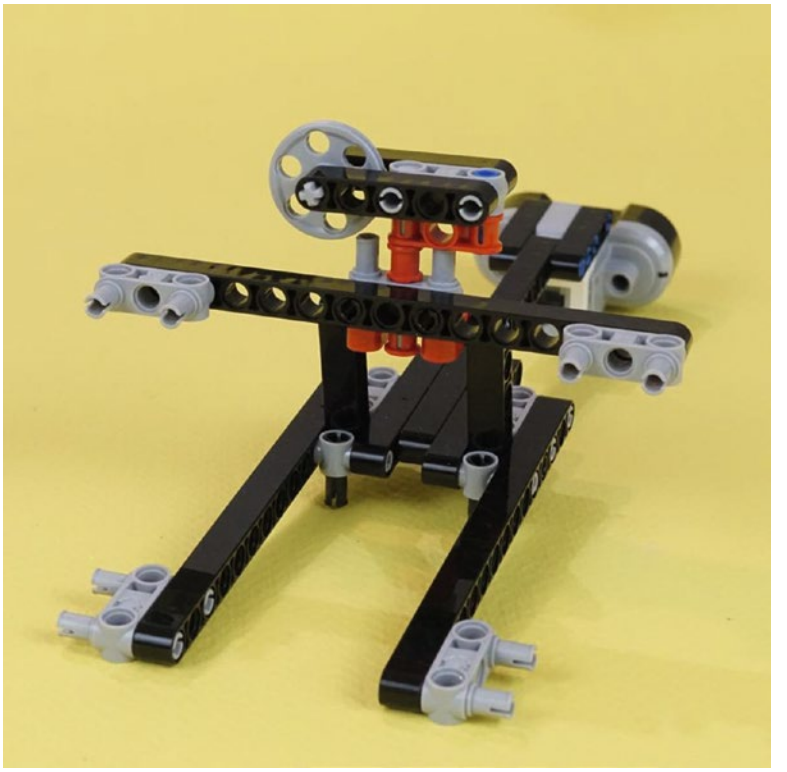


Figure 19-48. *Turn it around to see the back side of the Neck/Infrared Sensor•caster wheel and frames*

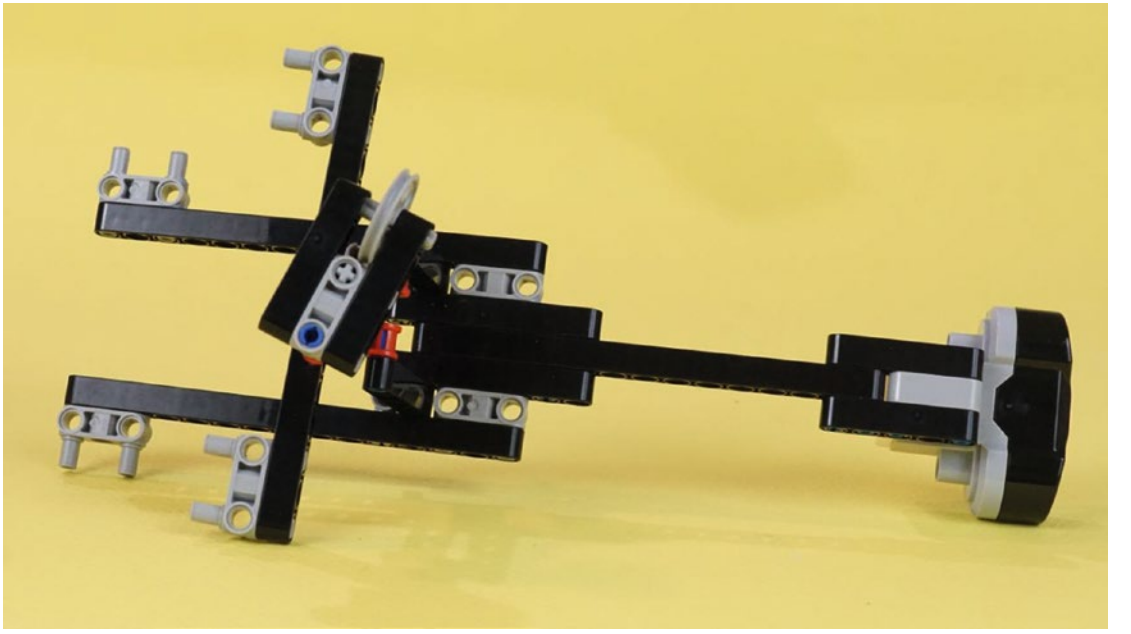


Figure 19-49. *Bottom side of the Neck/Infrared Sensor caster wheel and frames*

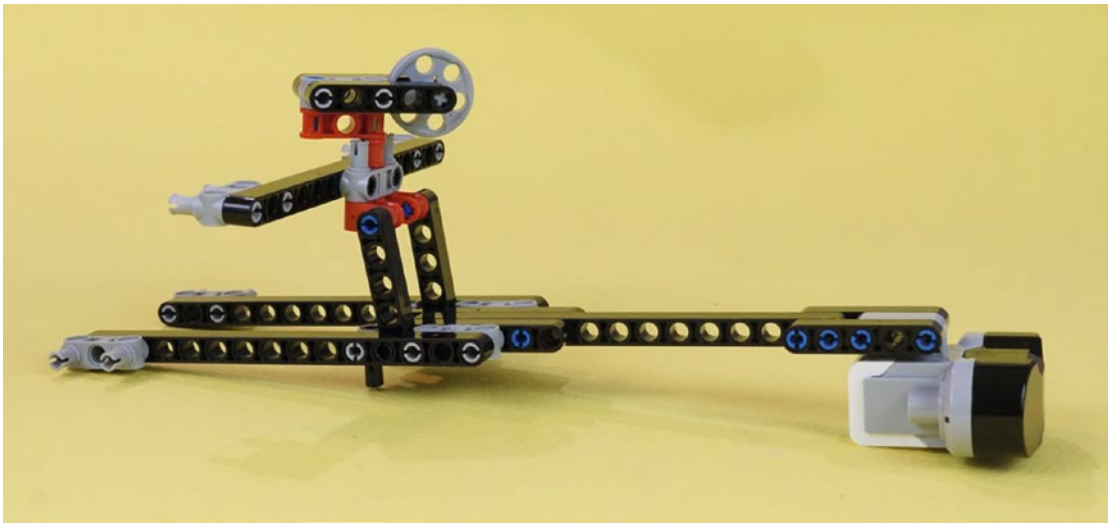


Figure 19-50. *Side view of the completed Neck/Infrared Sensor frame assembly with the caster wheel*

Third Section: Main Body and Motors

The PushBot is almost complete. Figures 19-51 through 19-72 demonstrate the steps needed to assemble the final bot.

Start with the Neck/Infrared Sensor assembly and Brick, as shown in Figure 19-51. In this figure, the Brick's "top" (where the USB port is located) is toward the bottom of the picture, although it's not visible in these shots. Figures 19-51 and 19-52 are different from most of the rest in this book. They are photographed to call attention only to the connectors that are about to be attached. The rest of the image is out of focus on purpose. Most other pictures in the book are photographed with all the parts in focus. But in the case of Figures 19-51 and 19-52, it helps to highlight the specific connection without other distracting details.

In Figure 19-53, the major parts are once again in focus and you can see how they are attached. In that figure, notice that the bottom of the Brick is toward the bottom on the picture. The bottom is the side with the A-B-C-D motor ports and the other USB port marked PC.

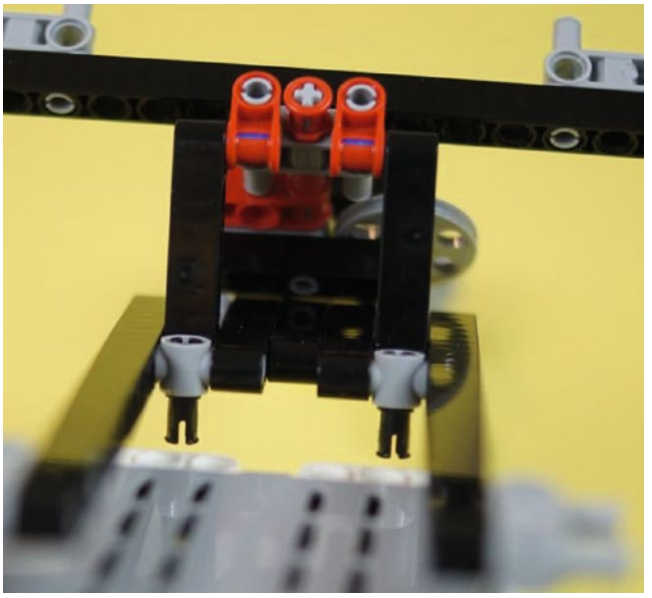


Figure 19-51. *Prepare to insert the two small black connectors in the top of the Brick as shown*

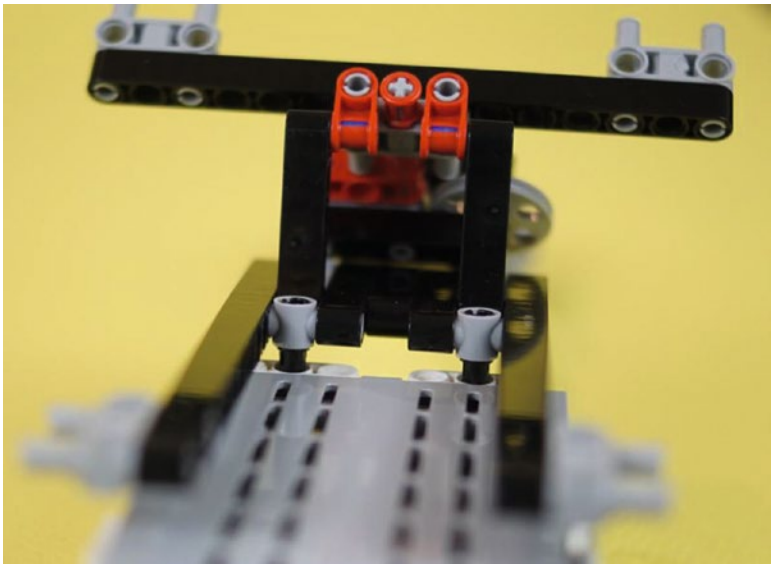


Figure 19-52. *The two connectors, partially inserted into the Brick. Press them in all the way and your bot will look like Figure 19-53.*

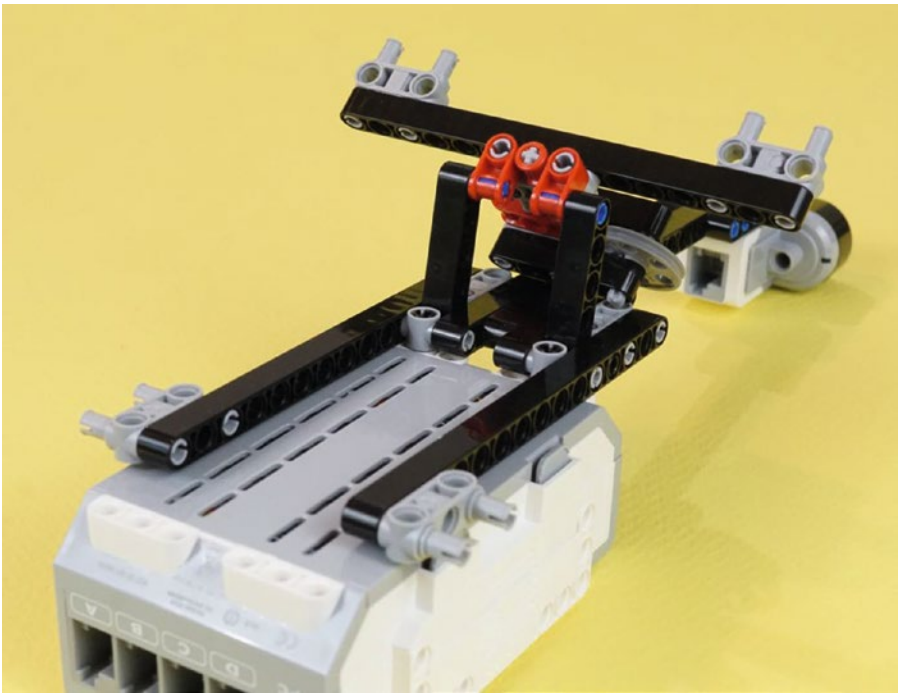


Figure 19-53. *The neck assembly is attached. The "bottom" of the Brick, motor ports A-B-C-D, is toward the bottom of the picture.*

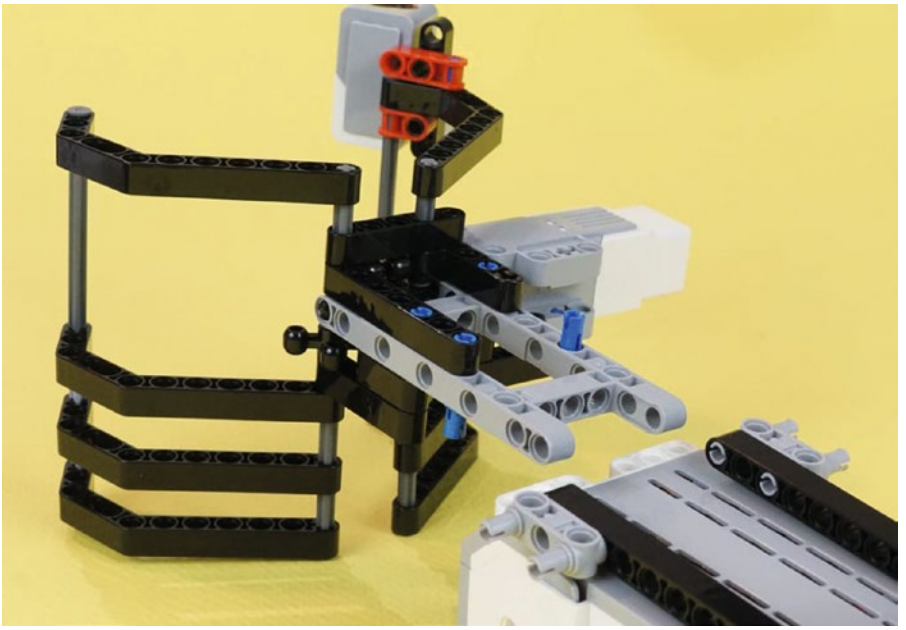


Figure 19-54. *Now the Figurine Jaw Cage is about to join in. Those long blue connectors on the bottom will engage the bottom of the Brick.*

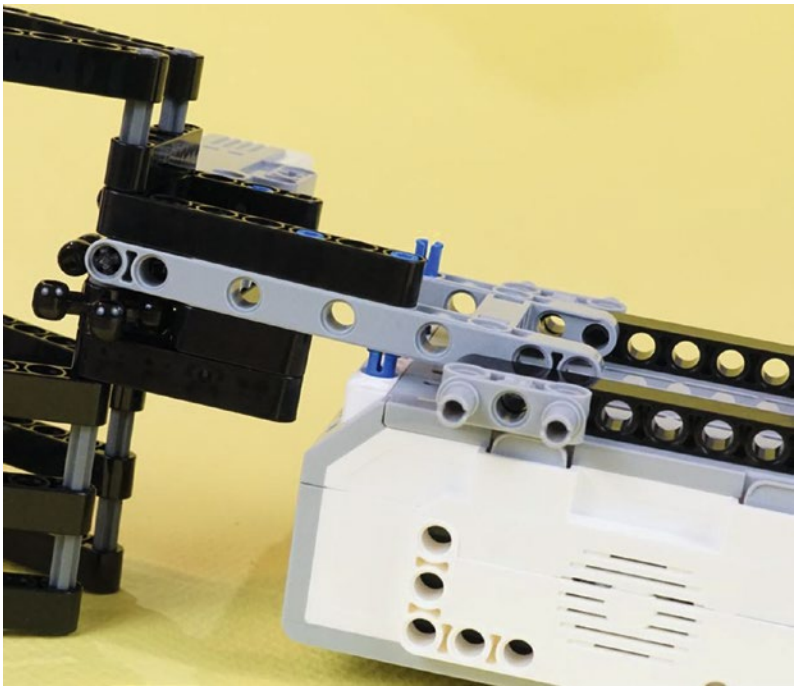


Figure 19-55. *Side view of how the long blue connectors go in. Press them all the way down.*

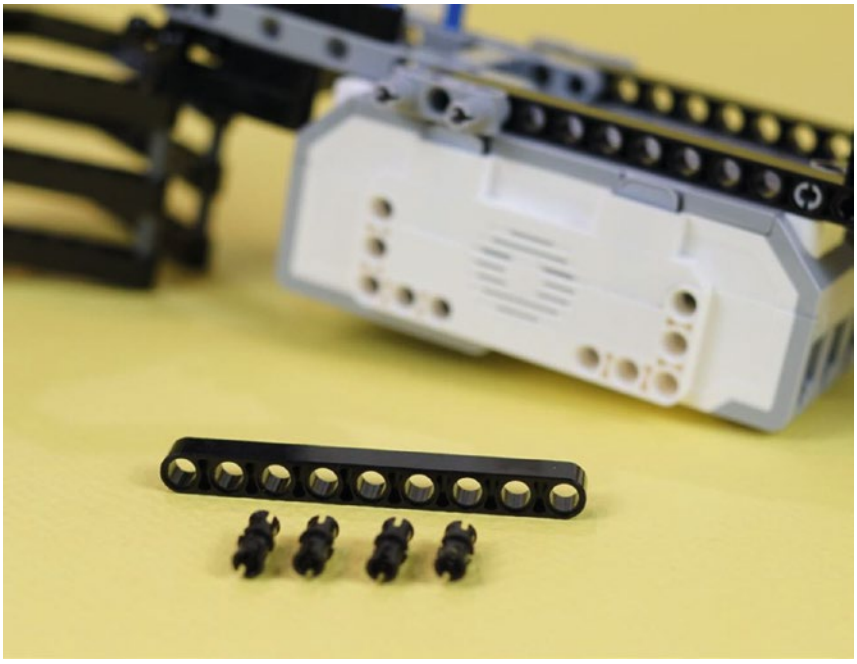


Figure 19-56. *Five new parts ➤ a nine-hole beam and four short black connectors*

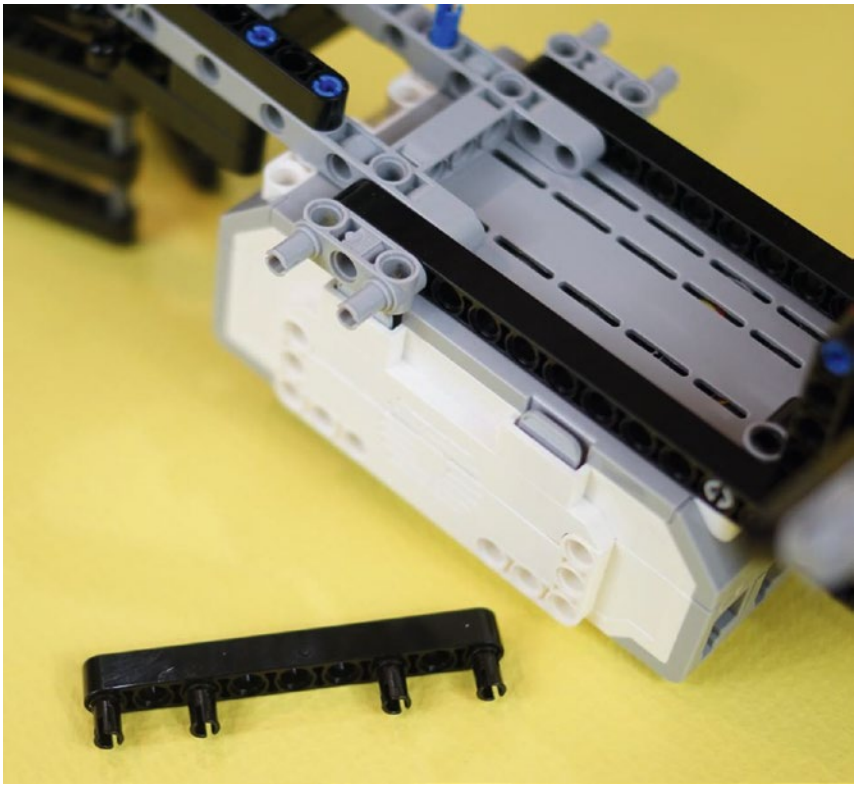


Figure 19-57. *Assemble the new parts like this. They create a reinforcing beam for the Figurine Jaw Cage.*

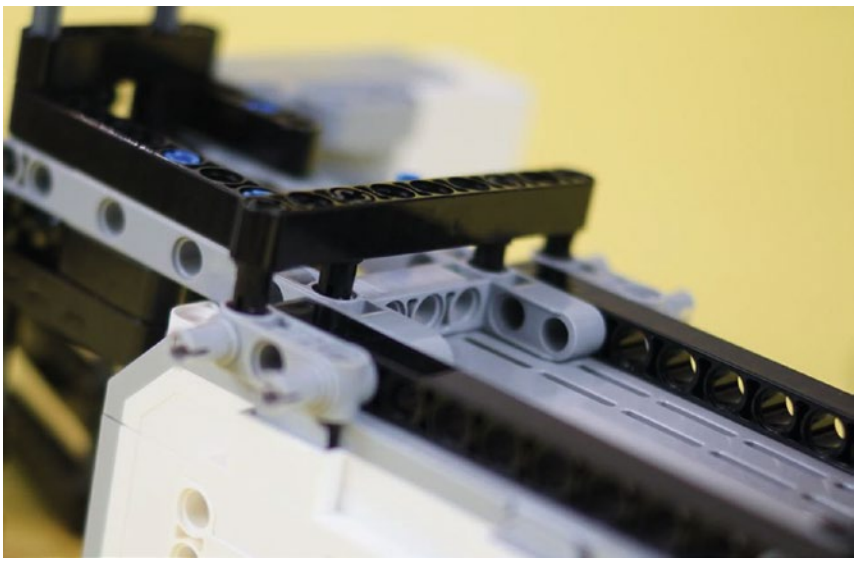


Figure 19-58. *This is where the reinforcing beam goes. It helps hold the neck assembly, Figurine Jaw, and Brick all together.*

Engineering: Prototypes and Parts You Don't "Need"

Figures 19-59 and 19-60 show the mounting of a long axle that reinforces the whole structure. You don't "need" these parts. And, if you have to change the batteries, they're one more thing you have to push out of the way to get the old batteries out. But they do make the structure stronger.

It's worth taking a moment to ask how that axle got into the design. And that gets us to thinking about *prototypes*. When you start building, you usually want to get something running so you can test it and see if it even has a realistic chance of solving the problem. That's what happened with this bot. When it became clear that the wheels and caster were reliable, I sat down to design a jaw. That took quite a few iterations, and each one became more compact and more reliable than the one before. Then I added as many long and short pins as possible, since strength is very important for a motorized part such as the jaw. So I would not call those dense pins "parts I don't need."

In the case of the reinforcing axle shown in Figure 19-60, the robot runs fine without it. There is no motorized part dependent on this for strength. It does have two advantages, though. One, it makes the robot more sturdy, particularly when I pick it up and set it on the course. Two, it has sort of an esthetic value ↓ "it's there because it *can* be there."

With that thought, we get to another engineering concept ↓ "gold plating!" Once you've finished something and it does the job it needs to do, there's a tendency to sit around admiring it and thinking "What else can I add to my robot?" Experienced engineering managers are used to seeing this and sometimes step in to say, "Hey team, this design needs to be frozen. No more work on it. We go with what we've got."

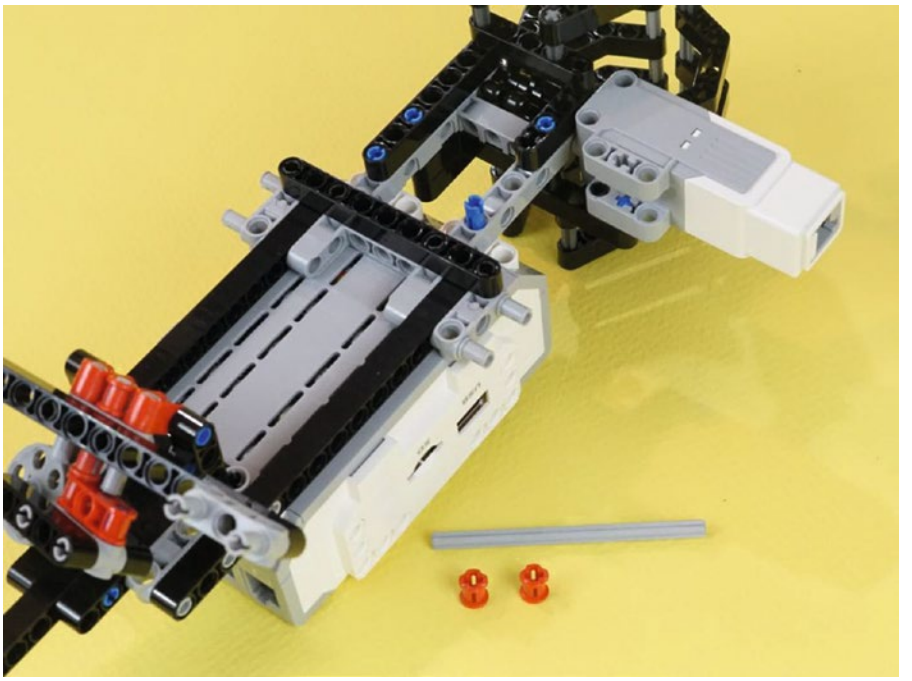


Figure 19-59. *Here are the two new parts for reinforcing the frame ➤ a long axle and two collars*

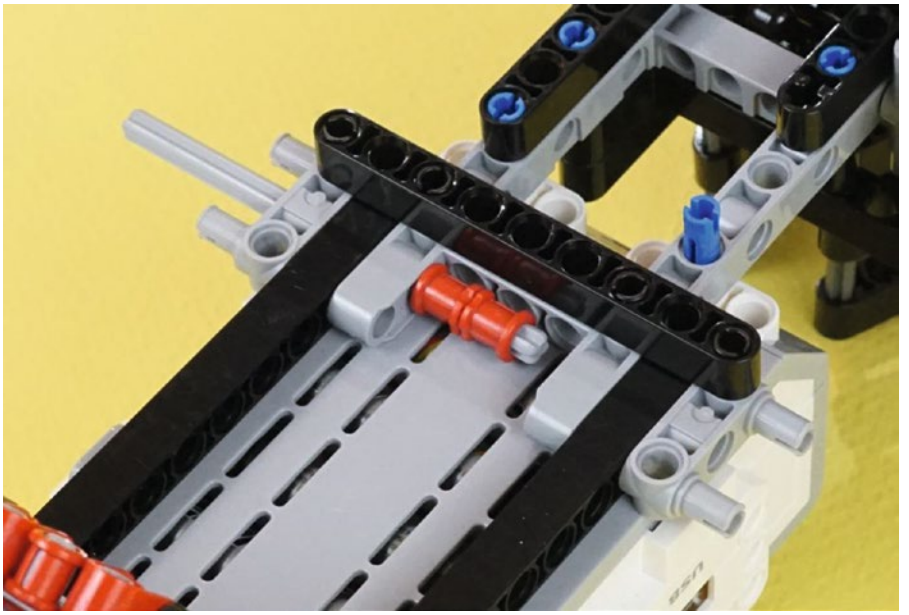


Figure 19-60. *Insert the axle, engage the two red collars, and push the axle the rest of the way in. Then press the collars as far apart as possible, for maximum strength.*

The neck assembly and Figurine Jaw are mated to the Brick. Now we need to build a couple of large motors with wheels and mounts. They are mirror images, as often seen on LEGO driving bases.

Figure 19-61. *All the parts to make one large motor drive wheel assembly. You'll make two of these, as mirror images. The axle is a six-hole length.*

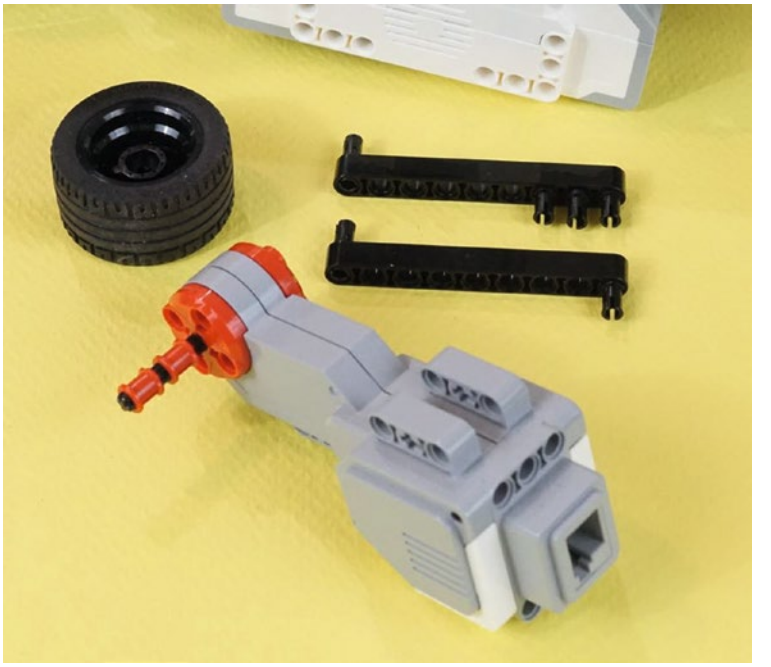


Figure 19-62. *The axle goes into the large motor, along with two collars. Press the tire onto the axle. Put the six pins into the two nine-hole beams.*

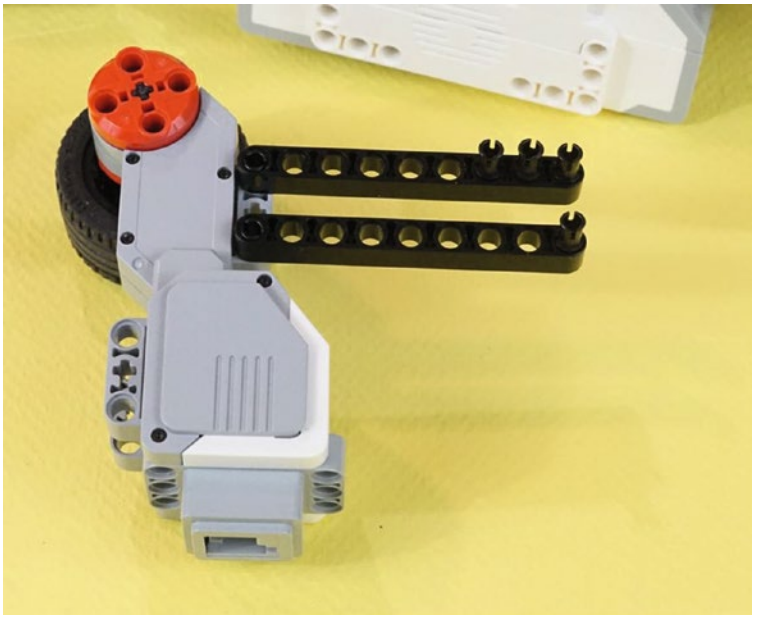


Figure 19-63. *Place the beams on the large motor as shown. Again, you'll make two of these. The other one will be a mirror image. Both mirror-image large motors are shown in Figure 19-64.*

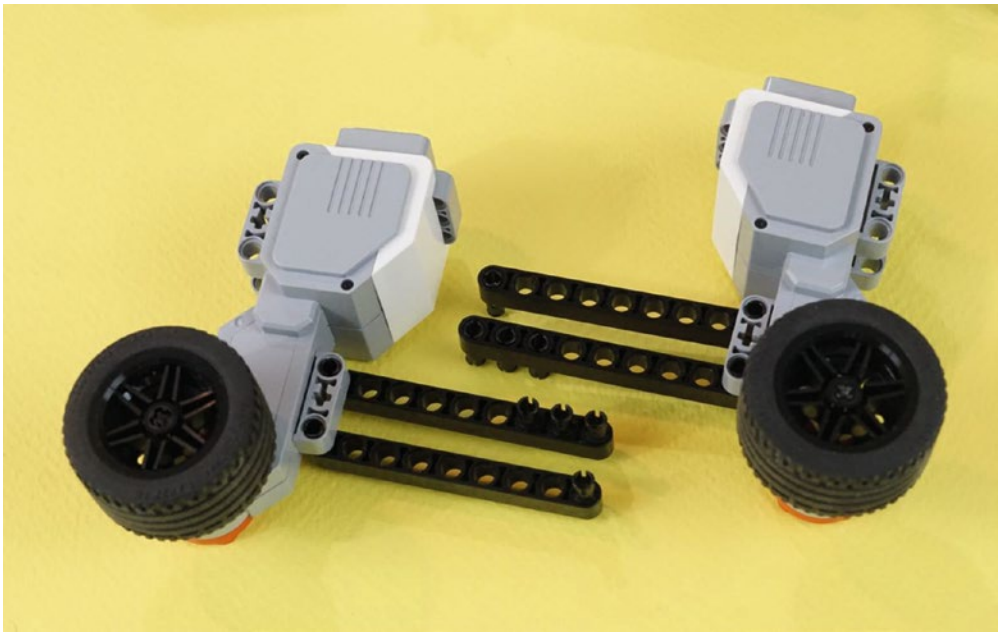


Figure 19-64. *Both of the motors with frame pieces, ready to be attached to the Brick*

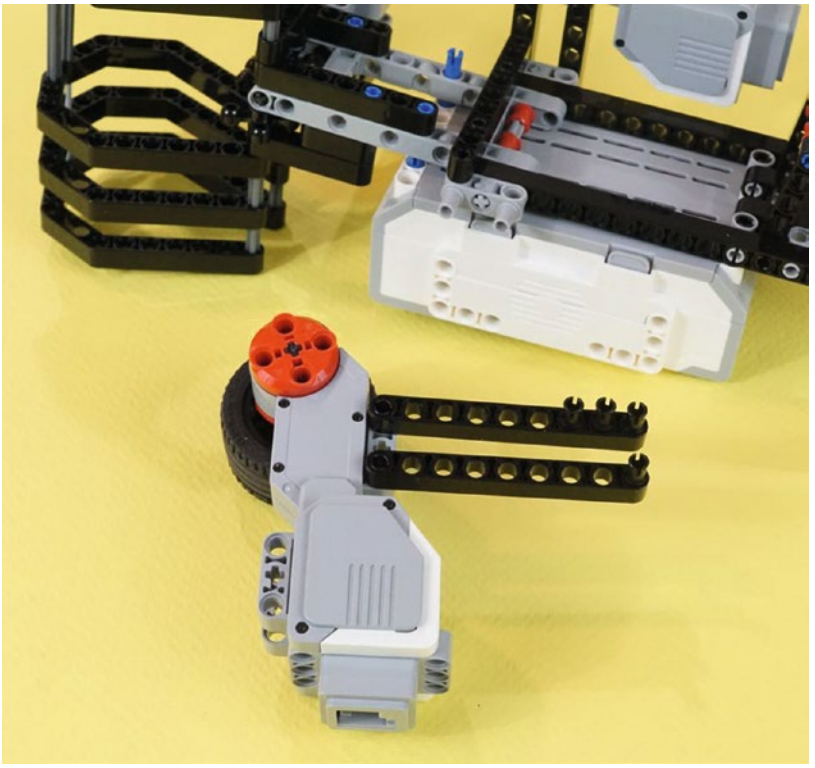


Figure 19-65. *Lift this motor up and press it into the Brick and frame pieces, as shown in Figure 19-66*

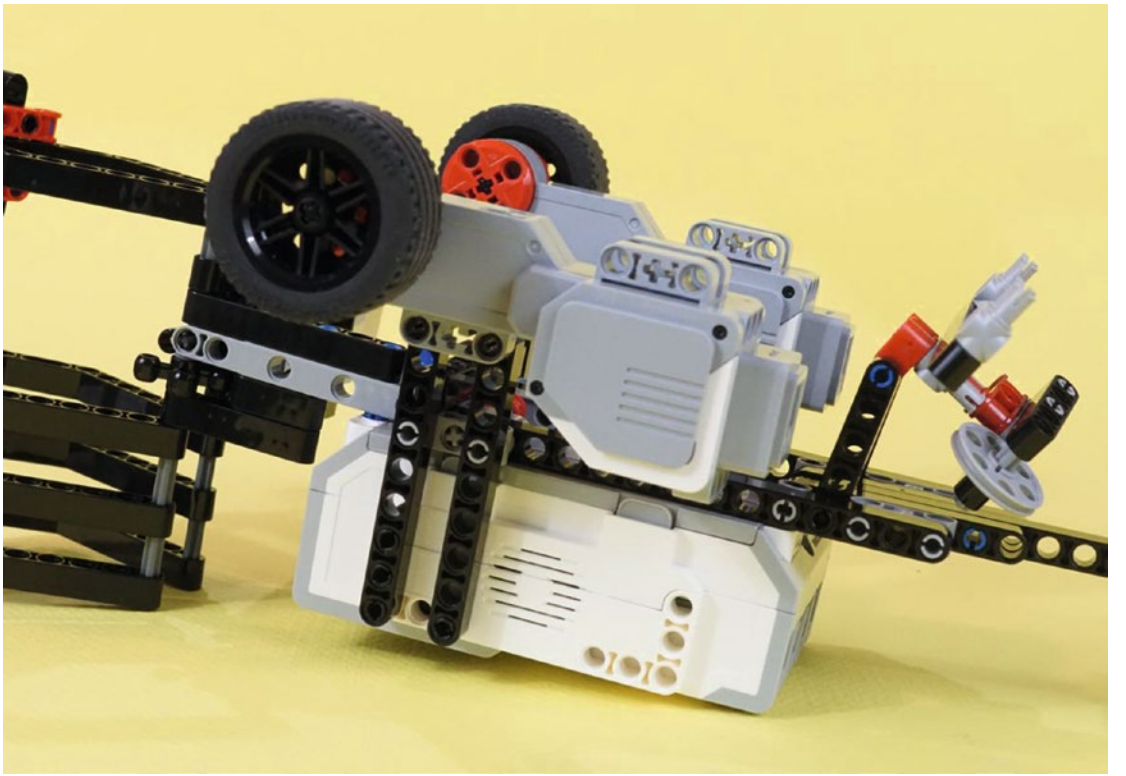


Figure 19-66. *A lot of connecting happens here. The four black pins on the motor's nine-hole beams go into the Brick itself. At the same time, that double gray connector sticking out from the frame under the bot connects with each beam.*

Both large motors should now be attached to the bot, with all pins pushed fully in. And now, one more shallow-focus picture to call attention to one specific component. The last thing we need to do is hook up the beam holding the caster wheel to the two large motors. In Figure 19-67, we see the double gray connector standing out against a soft background. It needs to go into that motor just behind it. This is shown in the normal, fully-focused, picture in Figure 19-68.

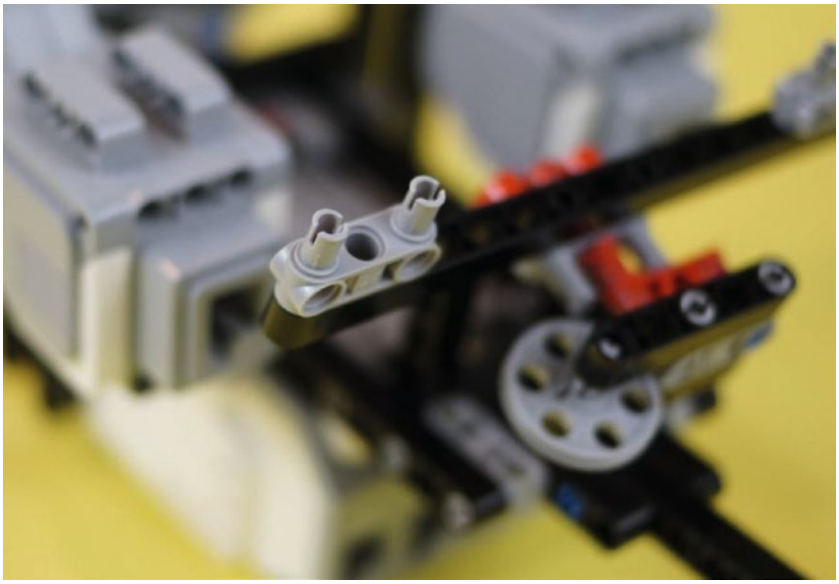


Figure 19-67. *This double gray connector will go into the motor closest to it. Same with the one on the other side.*

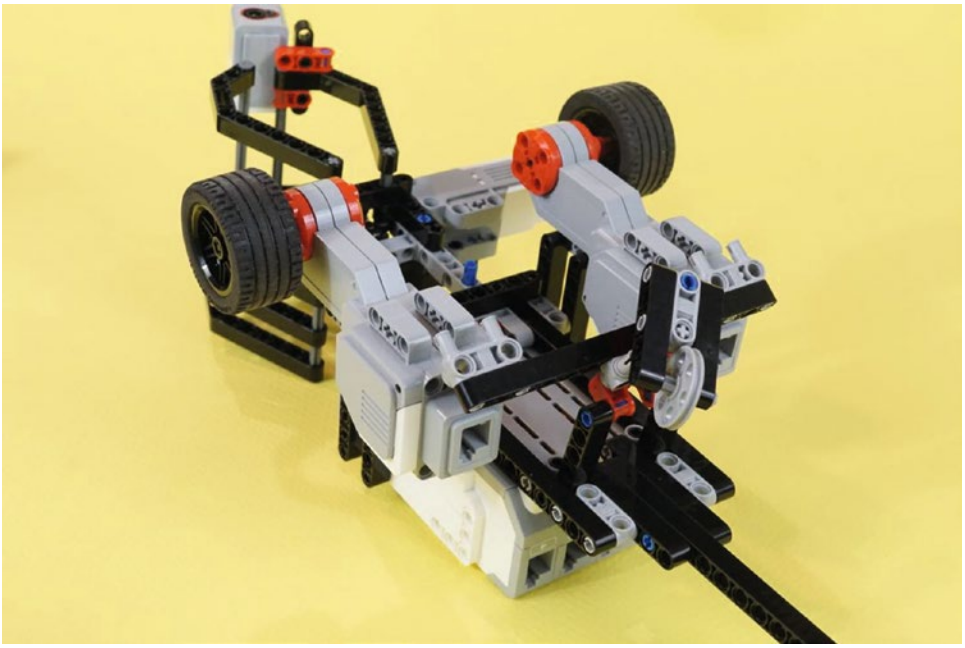


Figure 19-68. *The two double gray connectors on the long beam are ready to engage the large motors*

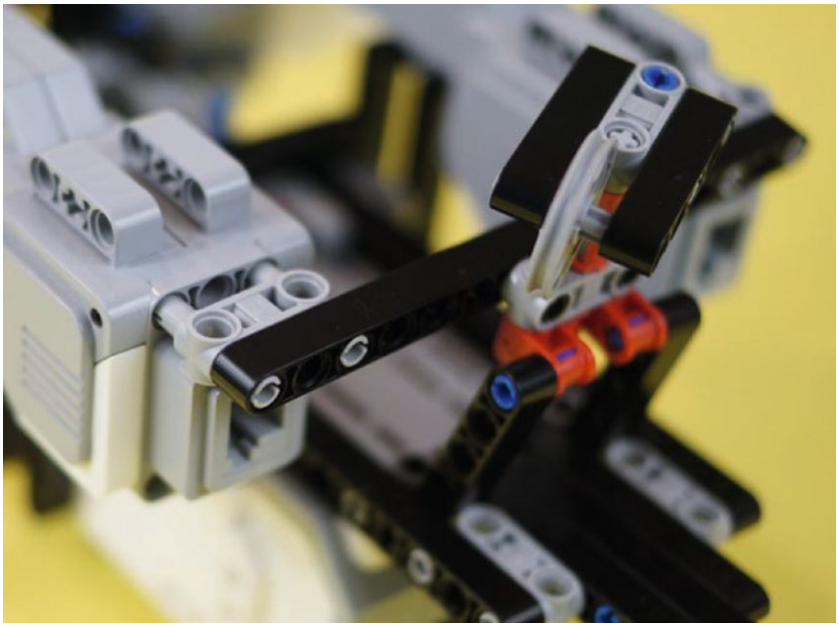


Figure 19-69. *This double gray connector engages the large motor next to it. Do the same with the one on the other side. Press the connectors all the way in.*

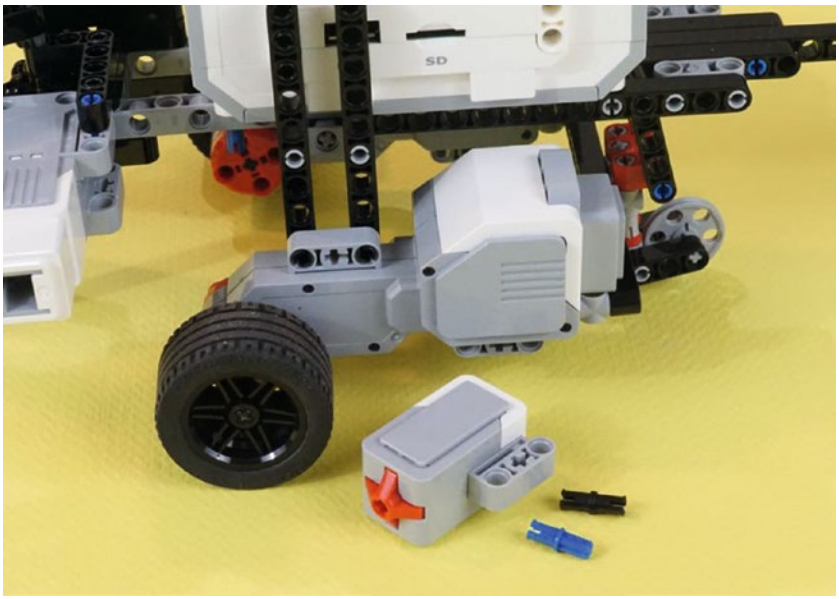


Figure 19-70. *The last parts! Gather up the Touch Sensor, a short blue axle-pin, and a black pin*

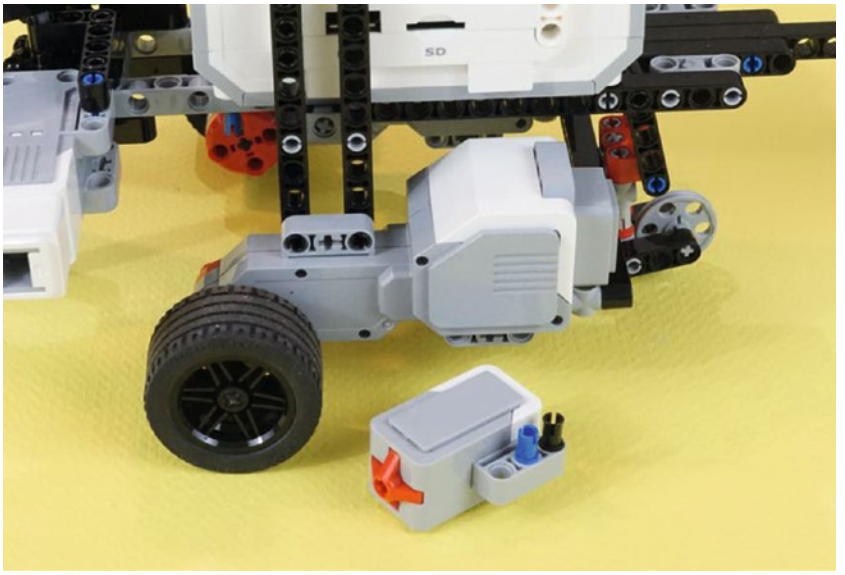


Figure 19-71. *Insert the short blue axle-pin and the black connector pin into the Touch Sensor*

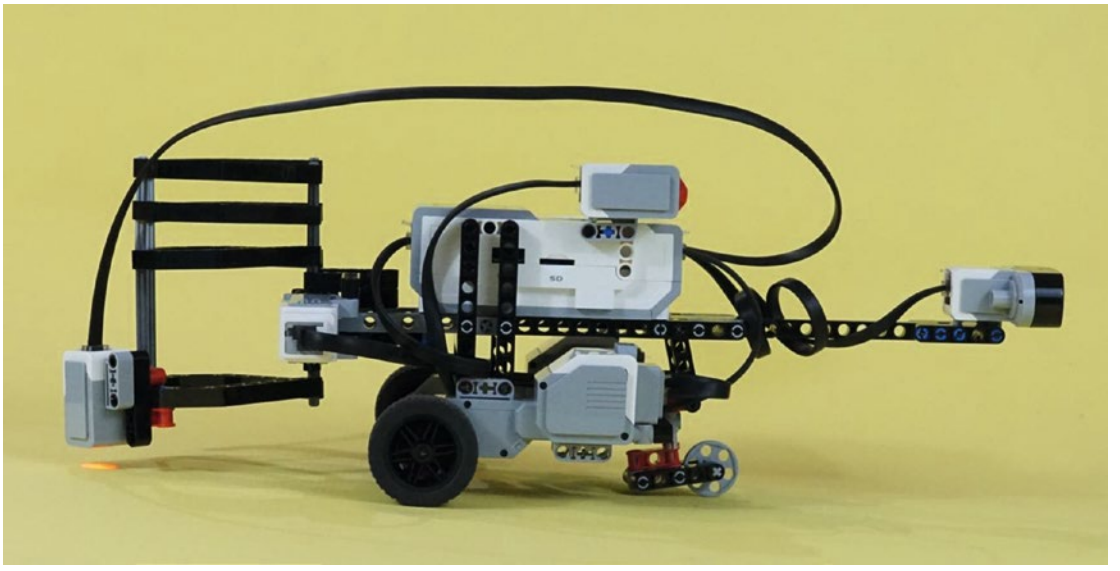


Figure 19-72. *Find a convenient spot on the Brick to mount the Touch Sensor. Then wire the three motors and the three sensors. All six wires from the EV3 retail kit are used in this design.*

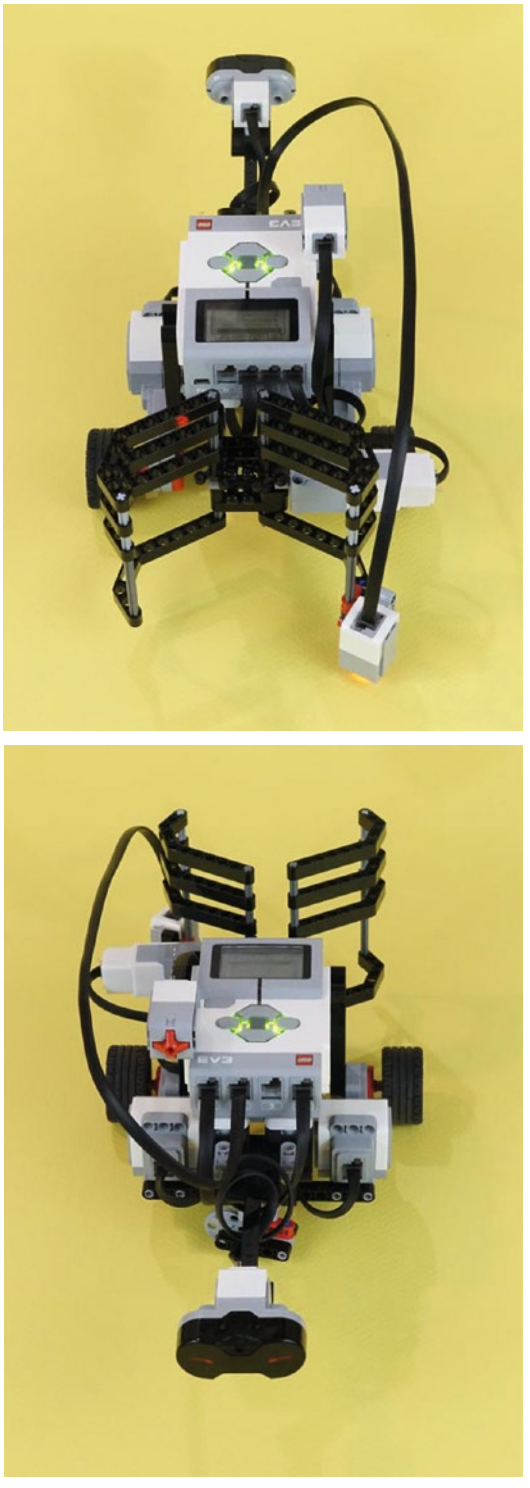


Figure 19-73. *On the left, motor ports A, B, and C are used. On the right, sensor ports 1, 2, and 4 are used*

Wiring the PushBot will use every wire found in the retail kit. The Medium Motor goes on port A. The Large Motors go on ports B and C. Wire the Touch Sensor to port 1, the Color Sensor to port 2, and the Infrared Sensor to port 4. Figure 19-73 shows this setup.

In Chapter 20, I show you how I intend to program this PushBot to complete its tasks. When you've wired up this PushBot (or your own version), continue reading. You're almost ready to send the bot into the burial chamber . . .

CHAPTER 20

PushBot: Program It

The PushBot has a *lot* to do! Because of this, I want to be absolutely certain that my program works, and I'll be testing it often. To make it easier on myself, I'm going to break down the program into three sections. The first section covers getting the bot to the proper location in front of the first figurine. The second section covers the programming blocks used to put the first three figurines into their proper locations. And the final section is devoted to pushing the last figurine up the ramp and onto the Mayan sarcophagus.

If you've built your own version of the PushBot, your program will most likely not match mine exactly. It really depends on how you've built your bot to complete the challenge.

In the next three sections, I walk you through the programming of this last bot so the team can finally enter King Ixtua's burial chamber and see what it contains. So, without any further delay, let's get this bot programmed and ready to go.

Getting the PushBot Into Position

In the LEGO MINDSTORMS EV3 software, click on the + tab at the upper left and then double-click on the blue Program name. Then type **PushBot**. Your screen should look like Figure 20-1.

Figure 20-1. *The beginning of the program for the PushBot*

Note To have more workspace visible on your screen, minimize the Document Your Work area on the right by clicking the EV3 Button symbol at the far right of the top tab bar. (This symbol is shaped like a little stop sign—that is, an octagon. It resembles the buttons on your EV3 Intelligent Brick.)

Once again, I am using the Touch Sensor as a Start button for the PushBot. You can choose to use one of the Brick's buttons, but I like the simplicity of running the program and having it wait until I press the Touch Sensor. I've placed the Touch Sensor in an easy-to-reach place on top of the Brick.

M. Bell and J. F. Kelly, *LEGO® MINDSTORMS® EV3*, DOI 10.1007/978-1-4842-2262-1_20

The first block I've dropped in my program is a LOOP block (see Figure 20-2). It will wait for the Touch Sensor to trigger.

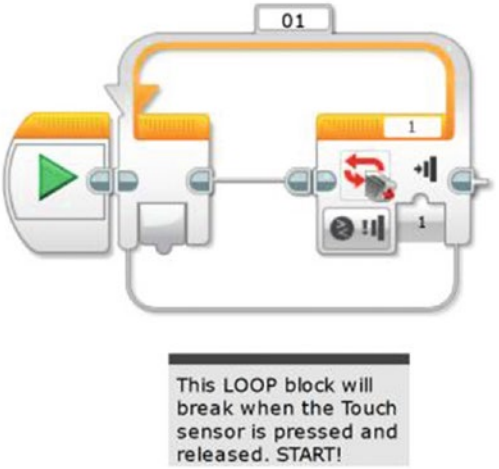


Figure 20-2. *This LOOP block will break when the Touch Sensor is triggered. The sensor is on Sensor Port 1.*

The first task my bot must perform is "Move from pedestal to first figurine without touching it." I'm going to take advantage of the fact that I can place my PushBot initially on the floor and point it directly at the first figurine. After that, I have a couple of different methods available for getting the bot into position.

I know roughly the distance between the pedestal and the first figurine, so as long as I program the bot to stop before it touches the figurine, I should be safe. If you look back at Figure 17-2, you see that the estimated distance is seven feet. I have measured my PushBot and it is just about 16 inches long. Take a look at Figure 20-3. If I place my bot as shown and point it toward the first figurine, it will need to move forward 5.5 feet and then stop to avoid touching the figurine. So, for the first method, I could program a MOVE STEERING block with the correct number of rotations (or degrees) to move it 5.5 feet.

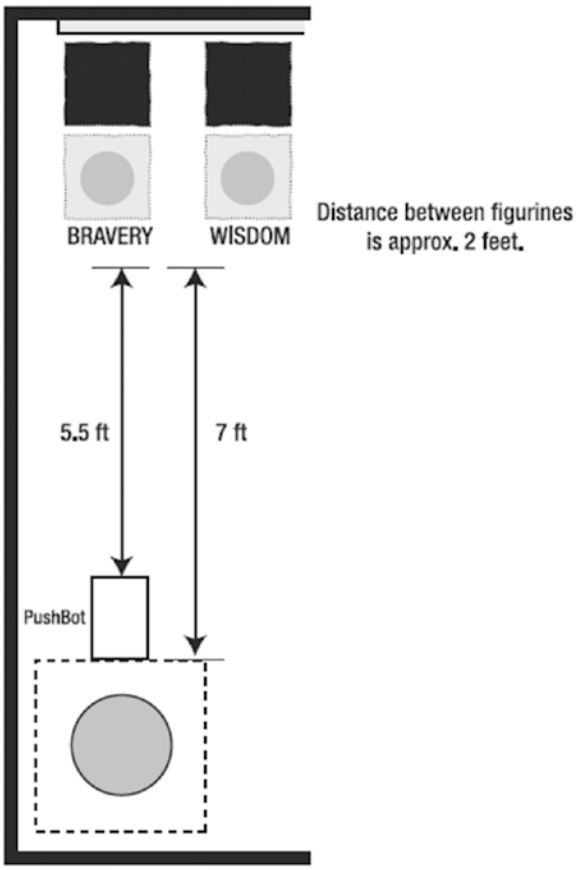


Figure 20-3. *The PushBot's initial movement will be 5.5 feet forward, based on the Infrared Sensor stopping in front of the first figurine*

But I want to use the second method that relies on the Infrared Sensor. Why do I want to do this? Well, if my measurements are off by just an inch or two, I might accidentally bump one of the figurines. I've noticed that when the batteries get low in my bot, sometimes strange things can happen with the motors. What I'd prefer is to use the Infrared Sensor to detect the figurine and stop the bot a safe distance away. And to accomplish this, I'll simply program my bot to move slowly toward the figurine, constantly checking to see whether it detects an obstacle (a figurine).

In Figure 20-4, you'll see that I've dropped in a LOOP block that is configured to break when the Infrared Sensor is triggered. Since I haven't tested it yet, I'll set the Infrared Sensor to be triggered when it detects an obstacle less than 30 centimeters (12 inches) away. Remember, this might change once I test the sensitivity of the Infrared Sensor.

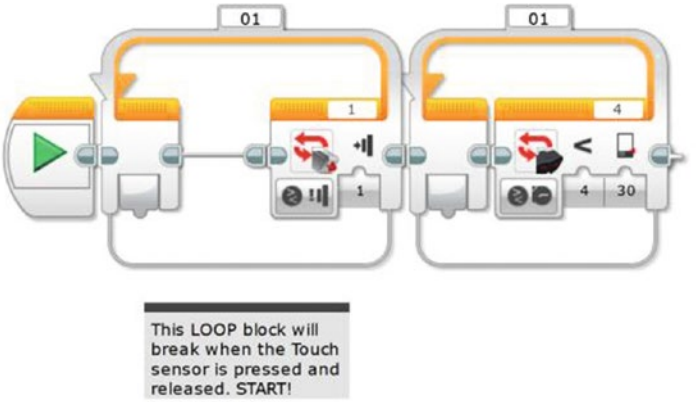


Figure 20-4. *This second LOOP block will break when the Infrared Sensor is triggered. It is on Sensor Port 4*

Before I perform an initial test of my PushBot, I'll place a `MOVE STEERING` block inside the `LOOP` and tell it to spin motors B and C. I can choose to set the Duration of the motors either to Unlimited or for a specific number of degrees or rotations. I will test both methods to determine which is the safest way to approach the figurine. You can see the `MOVE STEERING` block in Figure 20-5, and this block is initially configured to spin for one rotation. I am doing this because I want the bot to move forward, the Infrared Sensor to test for distance, and then the bot to move forward one rotation again. This will continue until the Infrared Sensor is triggered. (I'll also test it with the motors spinning constantly until the Infrared Sensor is triggered.)

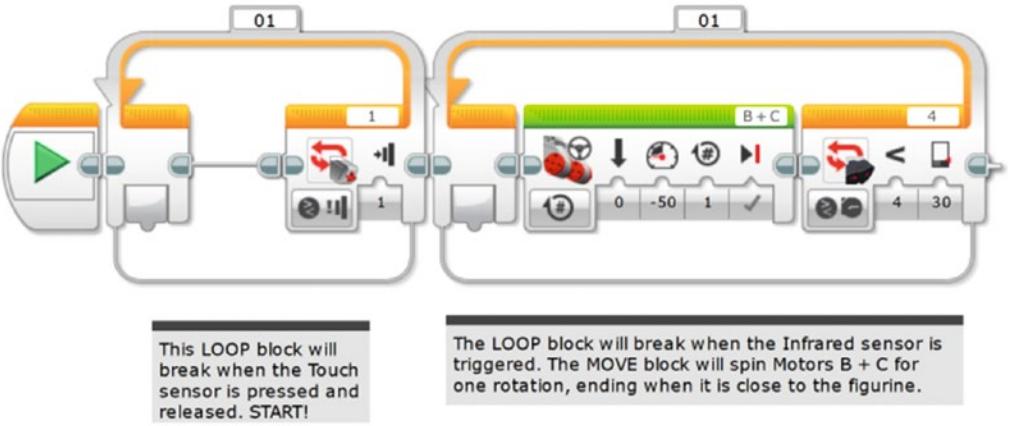


Figure 20-5. *A MOVE STEERING block for the PushBot*

I place a bottle of water about nine feet away from the PushBot. I make sure the cage jaws are open. I push the Touch Sensor (Start button) and off it goes. Results? When I configure the MOVE STEERING block's Duration to Unlimited, the bot stops much nearer to the bottle than when I configure it for a duration of one rotation. I also am forced to increase the sensitivity of the Infrared Sensor to 30 centimeters or less. After testing again, I find these settings are acceptable, and the PushBot is about 3 to 5 inches from the bottle.

Now that the bottle is found, I need to orient the PushBot to push the bottle. My PushBot design requires the bot to spin around and move toward the bottle until I tell it to stop. It will then close the cage mechanism and hold the bottle as the bot pushes it forward. The bot will stop when the black pressure plate (obsidian rock) triggers the Color Sensor. Let's take all of this one step at a time.

When the PushBot reaches the bottle, I need it to turn 180 degrees so the cage is facing the bottle. How do I do this? Simple. I first back the bot up to give it room to turn—one rotation of motors B and C should do it. I use a MOVE STEERING block to do this, as shown in Figure 20-6.

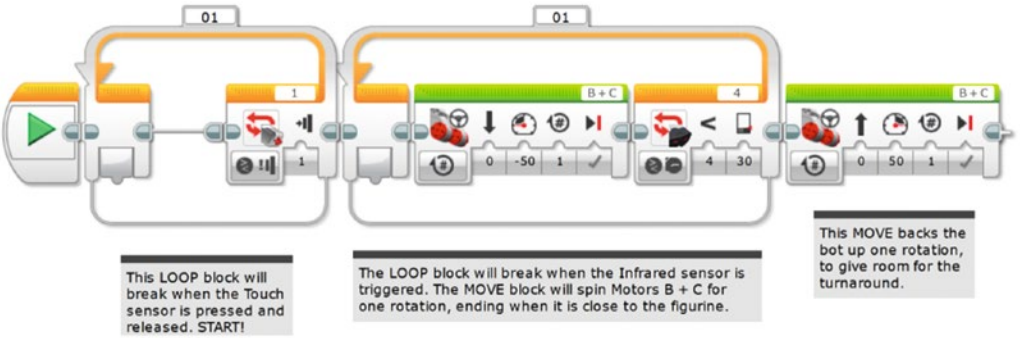


Figure 20-6. *The PushBot will first back up about five inches*

Next, I will turn *only* large motor B so the bot turns 90 degrees to the left. I will follow this with a LARGE MOTOR block that turns *only* large motor C so the bot turns 90 degrees to the right. At this point, the PushBot will be facing the reverse direction. This is illustrated in Figure 20-7.

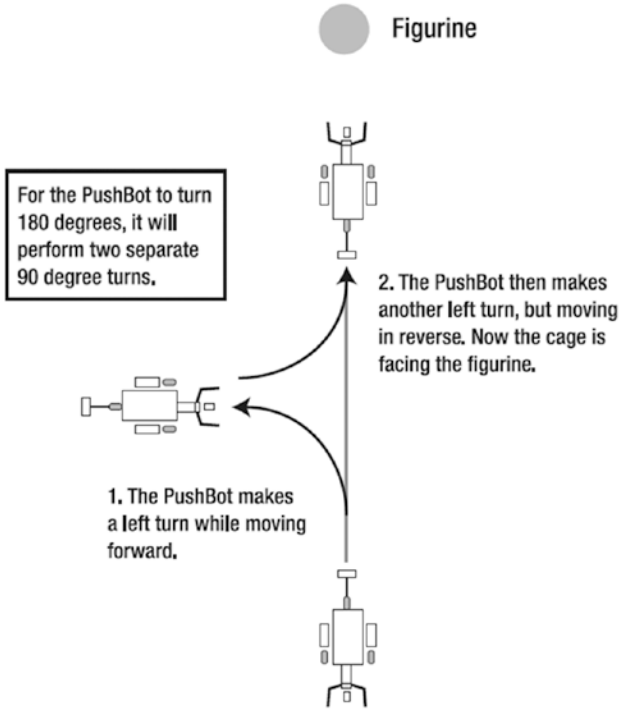


Figure 20-7. *The PushBot can turn 180 degrees by a pair of LARGE MOTOR blocks*

I used the method described in Chapter 4 for obtaining the correct number of degrees for large motors B and C to turn. For my PushBot, the right turn (motor B) made in Figure 20-7 requires -600 degrees. For the left turn (motor C), the motor is also configured for 600 degrees (the negative sign simply tells me to configure motor B for 600 degrees but to also change the spin direction). You cannot use a negative number as a duration, so you must enter the same value (600) but select the opposite spin direction. Figure 20-8 shows the `LARGE MOTOR` block for the right turn.

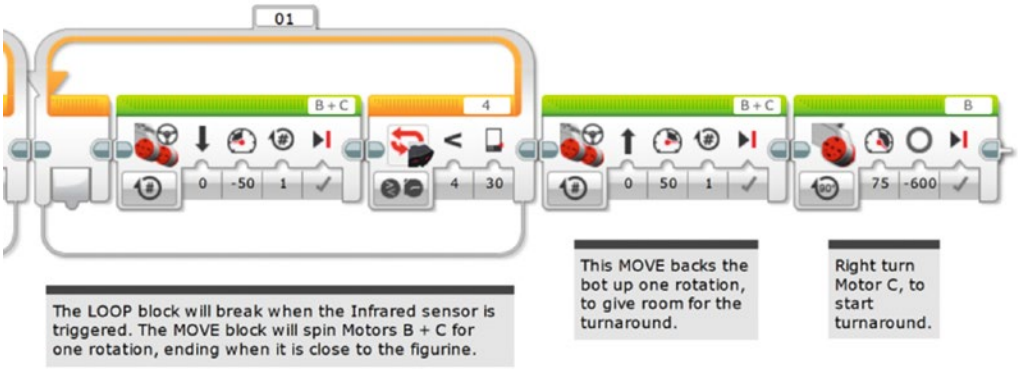


Figure 20-8. *This LARGE MOTOR block makes the PushBot perform a right turn*

I follow this with another LARGE MOTOR block for the left turn in Figure 20-9.

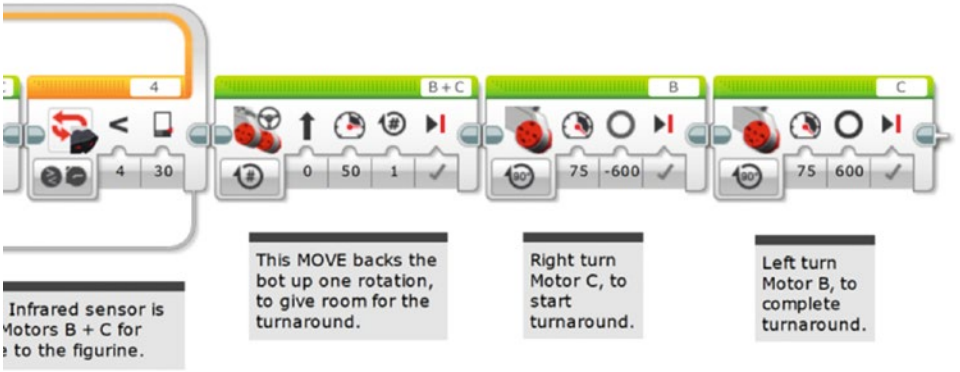


Figure 20-9. *And this LARGE MOTOR block makes the PushBot turn left*

I save and download the program to my PushBot for testing. And after testing, it is lined up properly with the bottle.

Now, let's pause here for just a minute and look at where my PushBot is located. Currently, the bot has spun around and is facing the first figurine with the cage in the opened position. What happens next? Let me break it down in a small list:

1. Approach figurine 1.
2. Close the cage around figurine 1.
3. Push figurine 1 forward onto the pressure plate and stop.
4. Open the cage and reverse to the starting position.
5. Turn right, move forward a short distance, and then turn left.
6. PushBot is now in front of figurine 2.

Do you see the pattern? It will do the exact same steps for figurine 2 and then end up facing figurine 3. It will perform the steps for figurine 3, with the only difference being that after it opens the cage (Step 4) and reverses direction, it will need to be directed to the figurine near the ramp.

Because of these repeated steps, I'm going to use a LOOP block that will allow me to perform these steps three times, once for each of the first three figurines.

Positioning Three Figurines

Before my bot locates and begins pushing the final figurine up the ramp, I'll need it to be stopped in front of figurine 3. Knowing this information will help me determine what I should place inside the LOOP block I'm going to add for the actions needed to push figurines 1, 2, and 3 into position.

But first, I'll drop in the LOOP block (see Figure 20-10) and configure it to execute any blocks placed inside it one time. (I'll later change this to three times after testing so it will perform the same actions for three bottles.)

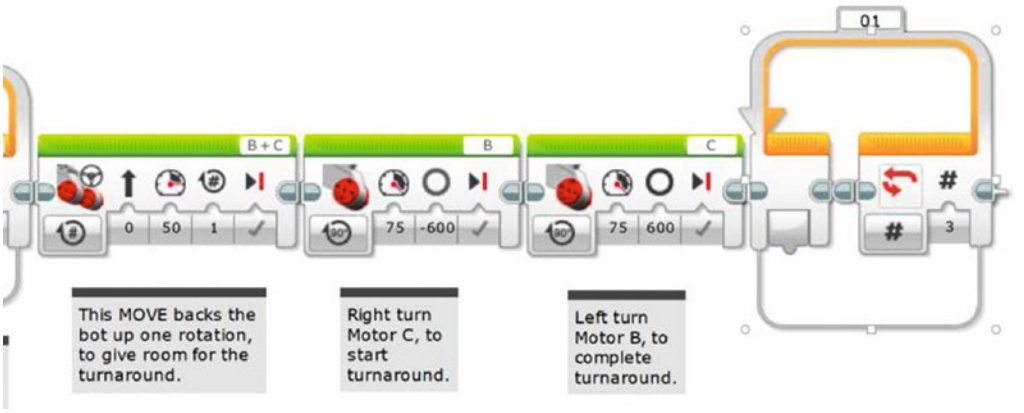


Figure 20-10. *The LOOP block for positioning the first three figurines. For testing, set the LOOP counter to 1. For running it on the real course, set the counter to 3, as shown.*

Now, the first thing I need the PushBot to do is move slowly toward the figurine and stop when the Infrared remote detects my signal. I'll use another `LOOP` block and configure it to break when the Infrared remote is triggered (see Figure 20-11).

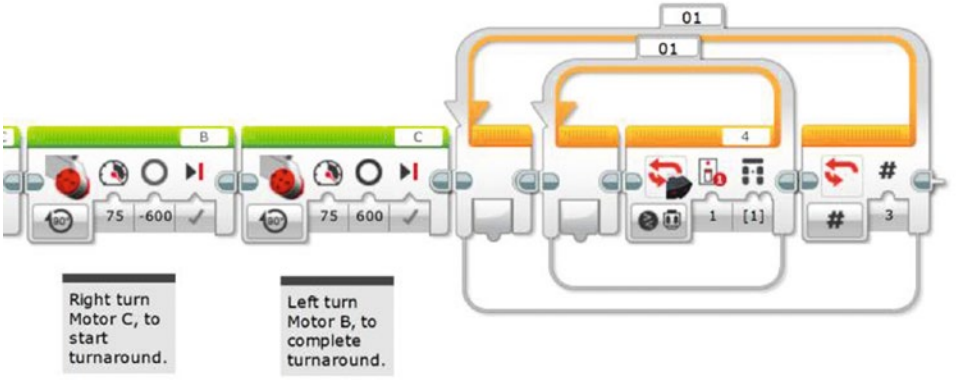


Figure 20-11. *This inner LOOP block will break when the Infrared Remote signal triggers the Infrared Sensor*

I'll add a `MOVE STEERING` block that will slowly move the bot toward the bottle. Figure 20-12 shows that I've configured it to spin large motors B and C for a quarter rotation (0.25).

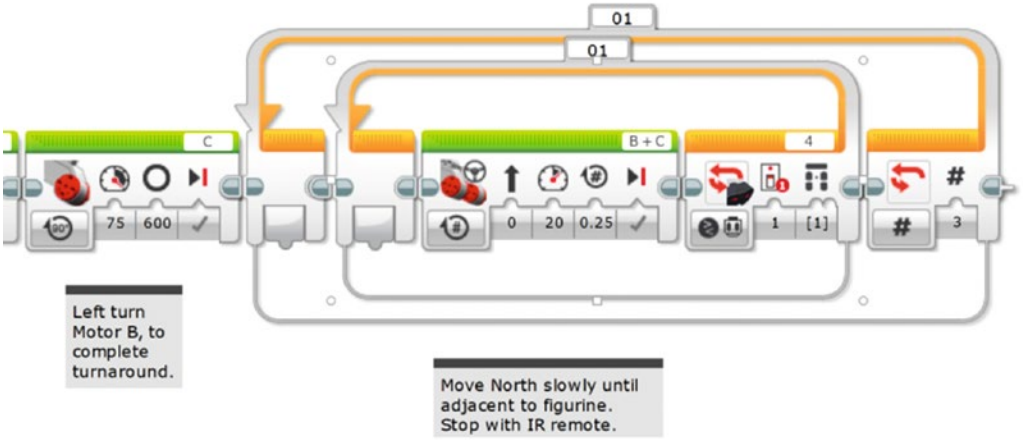


Figure 20-12. *The MOVE STEERING block will spin motors B and C in quarter rotations*

Now it's time to download the program and test it. I took the default Infrared Sensor setting, which is a value of 1. That corresponds to the upper-left button on the Infrared remote.

The bot did a perfect turn and approached the bottle slowly until the Infrared remote button-press exited the loop. Now I want the cage to close and hold the bottle. This is done with a `MEDIUM MOTOR` block, as shown in Figure 20-13.

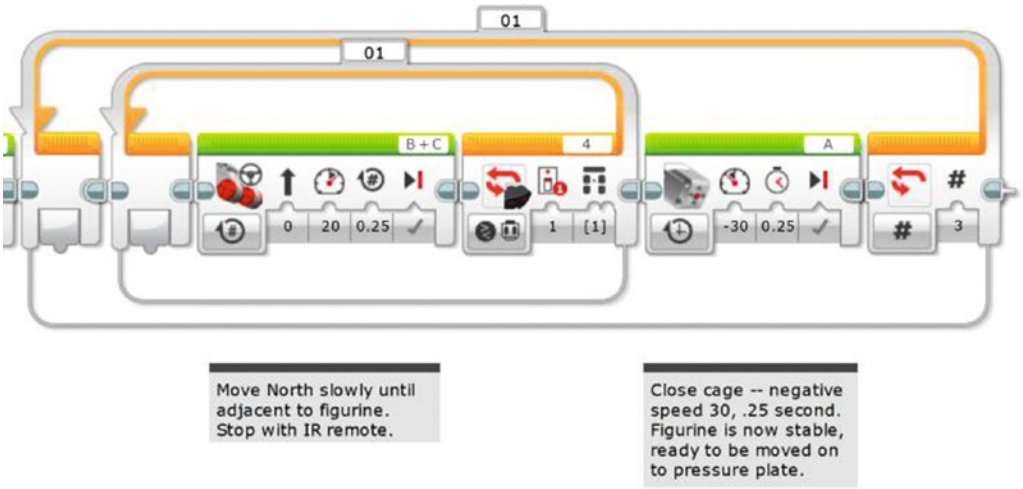


Figure 20-13. *The MEDIUM MOTOR block closes the cage. Using time instead of degrees means the motor won't jam, even if it cannot complete the Degree move as commanded.*

I configured the MEDIUM MOTOR block to spin motor A –65 degrees. In Figure 20-13, I originally entered a value of 65 for the number of Degrees and I also configured medium motor A to spin in reverse (to take into account the negative value of the rotation). This was the proper number of degrees required to move the cage from an open position to the closed position. I obtain this value using the View option on the Brick; refer to Chapter 4 for a review on how to obtain this reading.

Yet later, I ended up with the programming shown in Figure 20-13. That is, a negative speed of 30 for a time duration of 0.25 seconds. See below for an explanation of why I made this change.

After the cage closes, I've added another Medium Motor block that turns off the power to motor A (see Figure 20-14). This saves battery power.

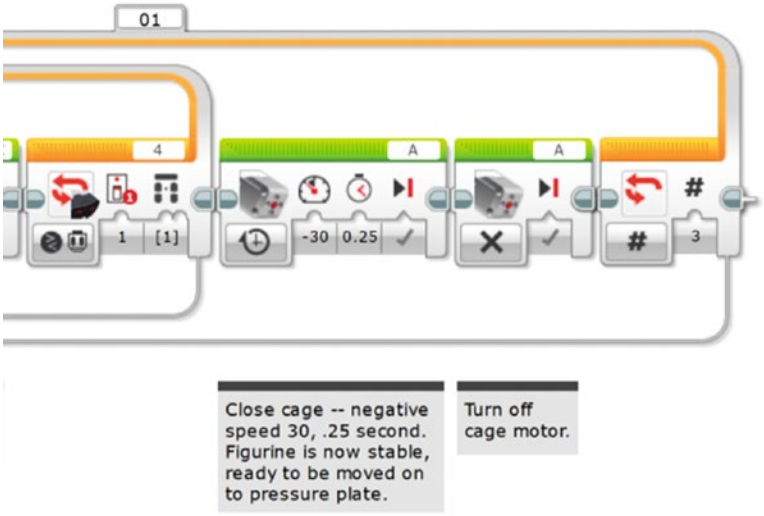


Figure 20-14. *This second MEDIUM MOTOR block turns off Medium Motor A, saving battery power*

Now that the cage is closed, I need the bot to slowly begin pushing the figurine (or in our tests, a bottle) until it detects the black obsidian rock floor. For testing, I've taped a black square of paper to the floor behind the bottle. I'll use a LOOP block that will break when the Color Sensor is triggered (see Figure 20-15). As I have done before, I test the Color Sensor on the black paper to obtain the correct setting (following the instructions in Chapter 4 to use the View option on the Brick).

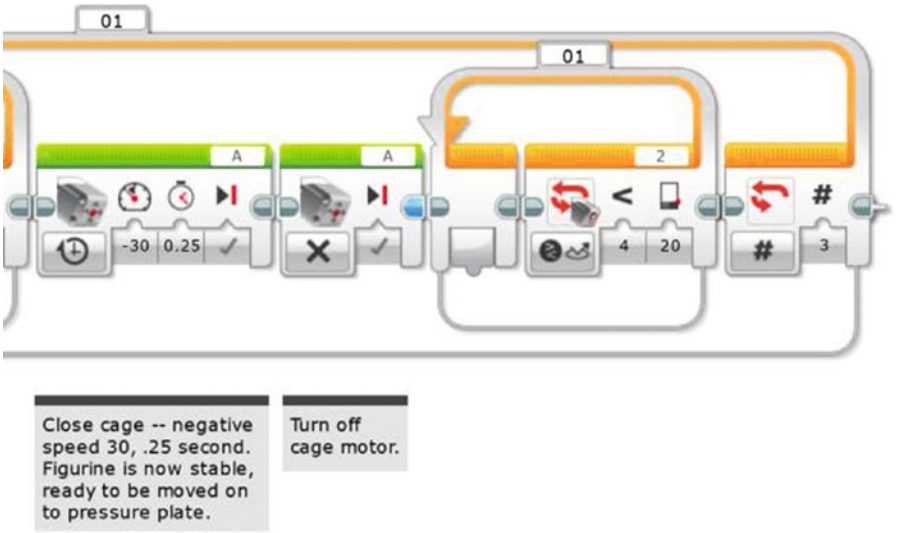


Figure 20-15. *The LOOP block will break when the Color Sensor is triggered*

My black paper gives me a value of 5; my floor gives me a value of 89. I will set the Color Sensor to trigger if the value drops below 20. Your values will differ, so testing is absolutely required.

I need to drop in a `MOVE STEERING` block to perform the pushing action (see Figure 20-16). I'll have it push forward 2.75 inches at a time (half a rotation). And, I'll add a half second delay to make sure the Color Sensor is able to detect the black paper.

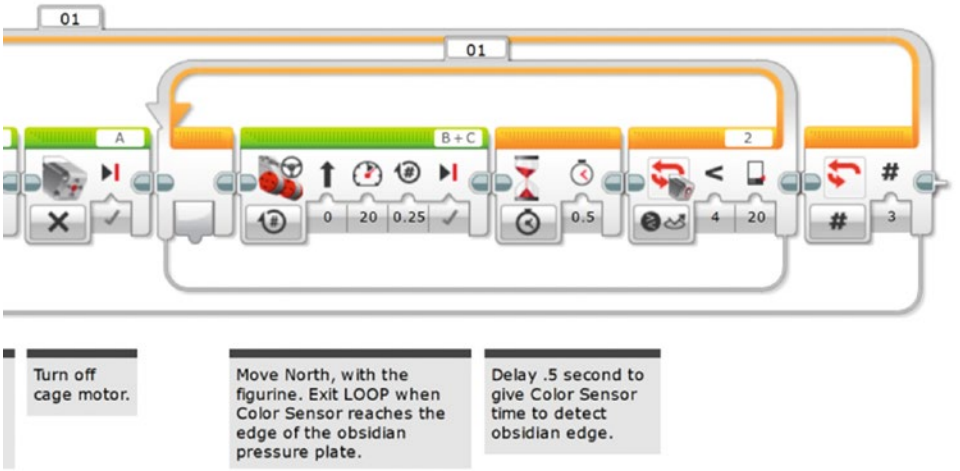


Figure 20-16. *The MOVE STEERING block pushes the figurine forward in quarter rotations. The WAIT block makes sure the Color Sensor has time to "see" the obsidian pressure plate (that is, the black paper)*

■ **Note** Observe that Large Motors B and C are frequently changing spin direction. In some instances you might think your robot will move forward, but it moves backward. When I had the PushBot spin around 180 degrees, everything changed! Now when I want Large Motors B and C to spin forward, I have to configure their directions the other way. It can get confusing, but that's why we test.

Now it's time to download and test. If everything works as planned, my bot should approach the bottle, stop when the bottle is detected, back up a bit and spin around, approach the bottle until I tell it to stop, close the cage, and then push the bottle forward until the Color Sensor is triggered.

Does it work? Not perfectly. That's the great thing—the essential thing—about testing frequently. I do *not* want to reach the end of my program only to find that I made a mistake very early! I originally find a problem with the cage—it jams and interferes with the rest of the program running. This problem is easy to fix. Medium Motor A was programmed to spin 65 degrees, but sometimes it doesn't spin all the way closed. The motor stalls and the program doesn't continue until motor A finishes its movement and closes the cage completely. My fix is to simply have motor A spin for 0.25 second. I'll leave in the `MEDIUM MOTOR` block that stops motor A, just in case I later decide to switch back to closing it a specified number of degrees or rotations. Other than that, the PushBot works well.

It pushes the bottle onto the black paper (pressure plate) and stops. But it is only on the edge of the paper. So I'll add another `MOVE STEERING` block after the `LOOP` to advance the figurine past the edge, onto the center. That one rotation gives me 5.5 inches. Now I just need to get it to open the cage, back up a reasonable distance, and perform a couple of movements to put it in front of figurine 2.

First, I'll make that one-turn advance with a `Move Steering` and then open the cage with a `MEDIUM MOTOR` block (see Figure 20-17).

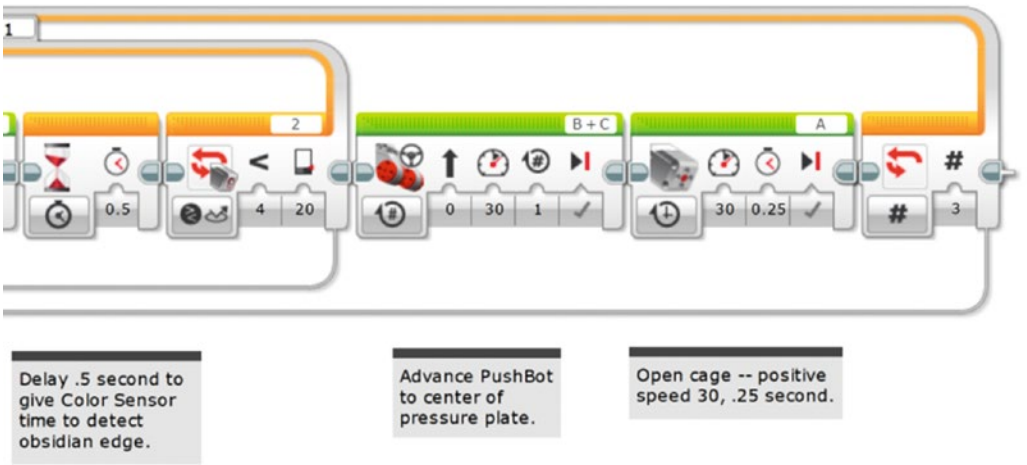


Figure 20-17. *The first MOVE STEERING block centers the figurine, and the second MEDIUM MOTOR opens the cage*

Next, I'll get the PushBot to reverse direction a reasonable distance. Referring back to Figure 17-2, I think a safe distance for the bot to pull back would be about three feet. I'll test it and tweak that value if I find the bot isn't pulling back far enough. Figure 20-18 shows the back-up-South `MOVE STEERING` block I've added. (To configure the distance, I've converted 3 feet to 36 inches. I divide that by 5.5 inches—the circumference of the tires—and I get a value of approximately 6.5 rotations.)

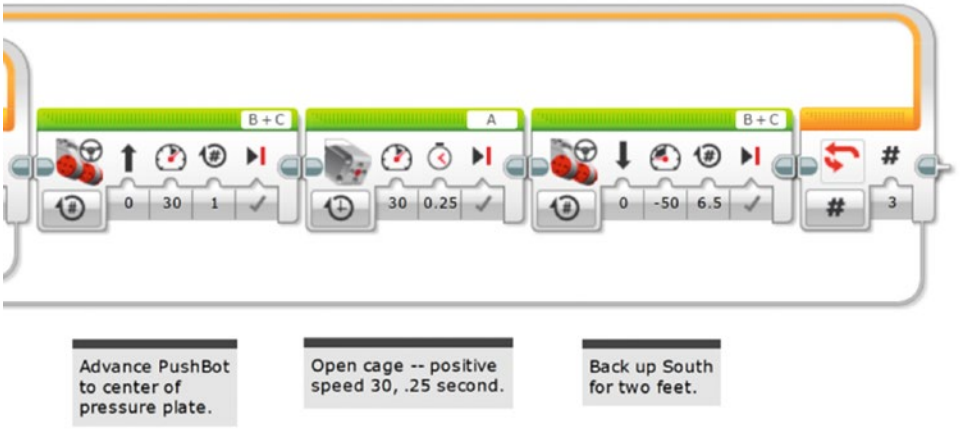


Figure 20-18. *This MOVE STEERING block will get the PushBot to move away from the figurine, in the southerly direction*

The last few items I need to take care of will get the PushBot in front of the next figurine. The bot needs to turn right, move forward a small distance (about two feet), and then turn left. This will position it facing the next figurine.

Then let's make the bot perform a right turn (see Figure 20-19).

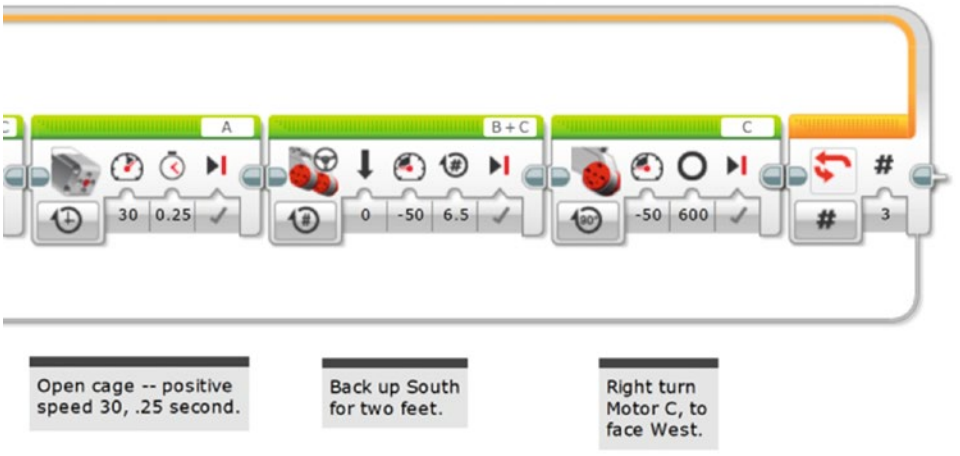


Figure 20-19. *This LARGE MOTOR block turns the PushBot right, facing West*

Next, Figure 20-20 shows the MOVE STEERING block needed to move the PushBot forward two feet in a westerly direction. Keep in mind that this is one of those configurations that might need to be changed; you won't know until you test your bot with three figurines properly placed.

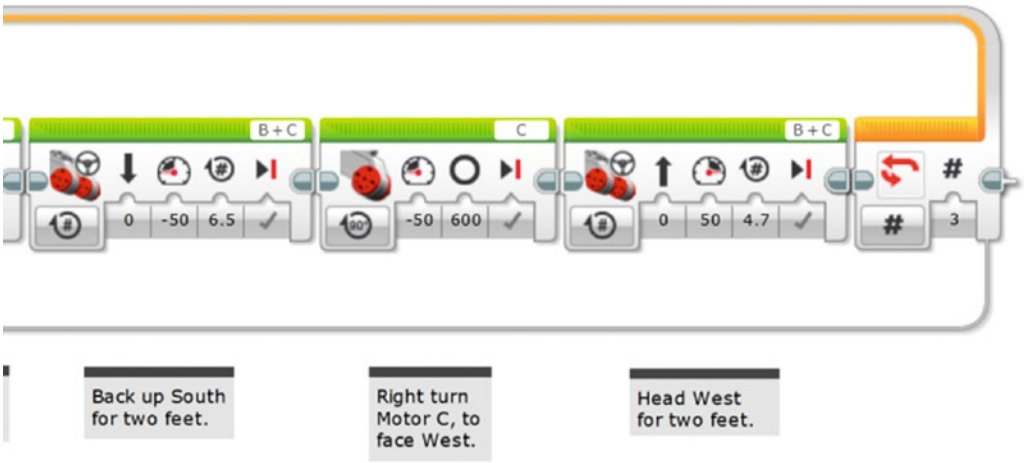


Figure 20-20. *This MOVE STEERING block sends the PushBot forward, heading West approximately two feet*

And, finally, I need the bot to spin to the left (see Figure 20-21).

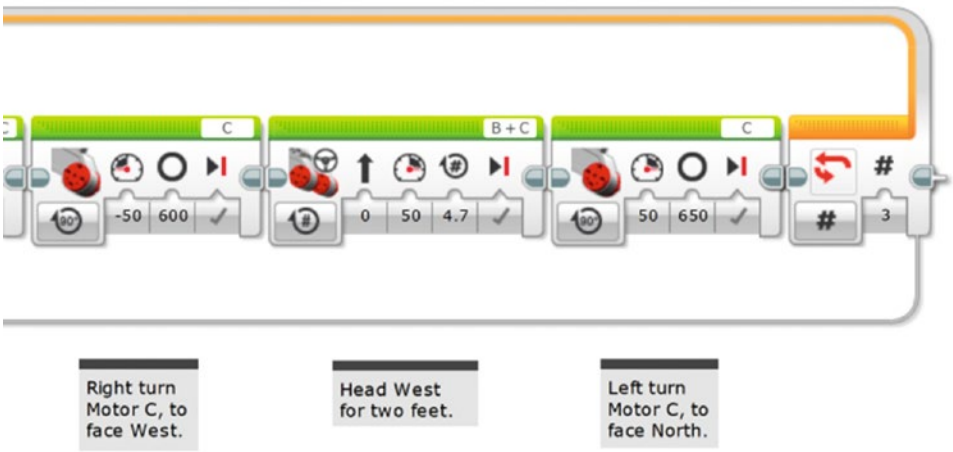


Figure 20-21. *This LARGE MOTOR block turns the PushBot left, facing North once again*

Time to test; let me describe my testing environment. I place three water bottles four feet from a wall in my living room. The water bottles are two feet apart as well. I then fasten three squares of black paper about four inches behind each bottle (the paper is taped down so it can't be pushed along with the bottle). I then place my PushBot about nine feet away from the first bottle, pointed directly at the bottle with the Infrared Sensor. The cage is fully open. And here's what happens.

I have success on the first figurine, but overshoot the second figurine by about six inches. I fix this by reducing the number of durations large motors B and C spin (in Figure 20-20) from my earlier value of 6 to 4.7, as is shown in the figure. I also have a problem with the cage closing properly but opening too wide, and the arms rub the tires. I reduce the time for the cage to open from 1 second to 0.25 seconds (a quarter of a second). On the second test, I have perfect success—all three figurines are located and pushed to their proper locations without tipping over.

At this point, I'd like to remind you that your settings will probably be different. Motors and sensors all have varying sensitivity, so testing is an absolute requirement at this point. Don't stress about this part of building robots; this is how you improve your skills and gain insight into the way robots work in different environments.

Figure 20-22 shows the approximate location of the PushBot at this point. I've got to get it moving so it can push the fourth figurine up the ramp and onto the sarcophagus.

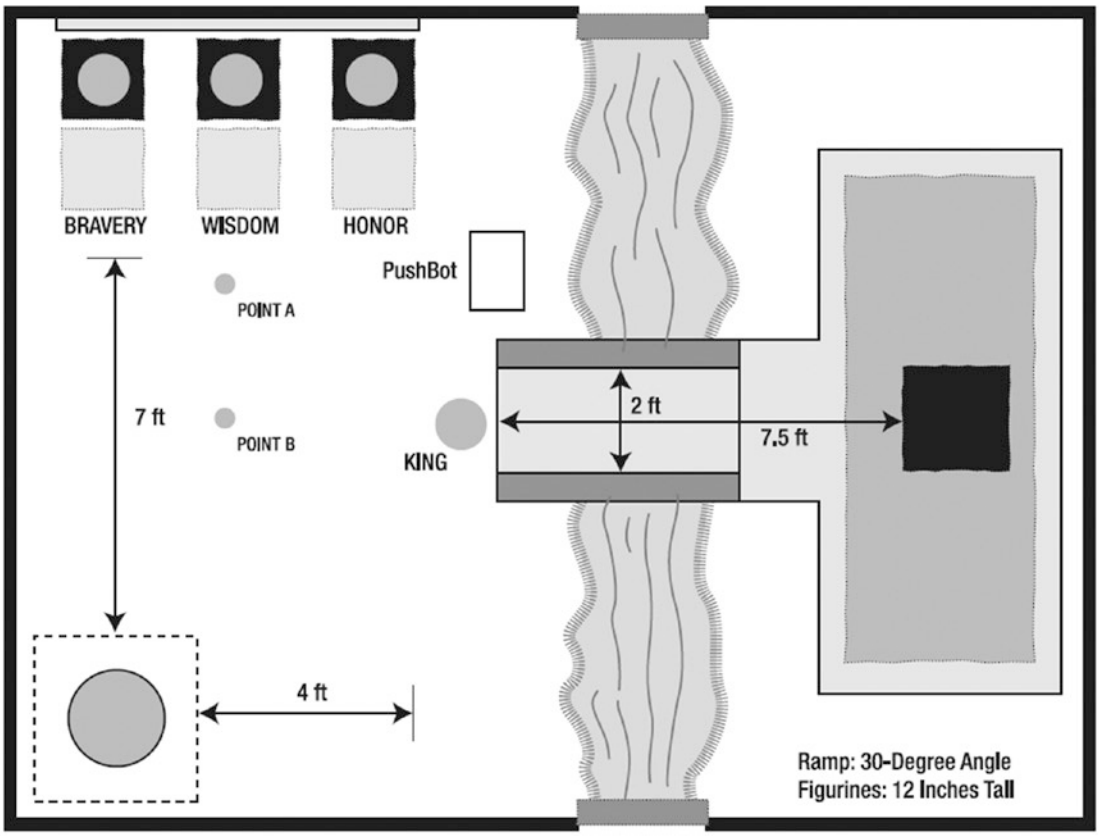


Figure 20-22. *The PushBot should be located above the center of this diagram as shown*

The Final Figurine

Now to what I consider the easy part. Okay, maybe not easy, but definitely easier than trying to push three figurines onto pressure plates. I have some rough distances to work with from Figure 20-22, so I'll use these measurements, along with the Infrared remote, to position the PushBot for its final task.

First, I have to reorient the bot. Since it is pointing away from the fourth figurine, I first have the bot spin to the right and then move to point A in Figure 20-22. The cage will be pointing away from the sarcophagus when it reaches point A. I'll drop a `MEDIUM MOTOR` block in for the turn to the right, facing East (see Figure 20-23).

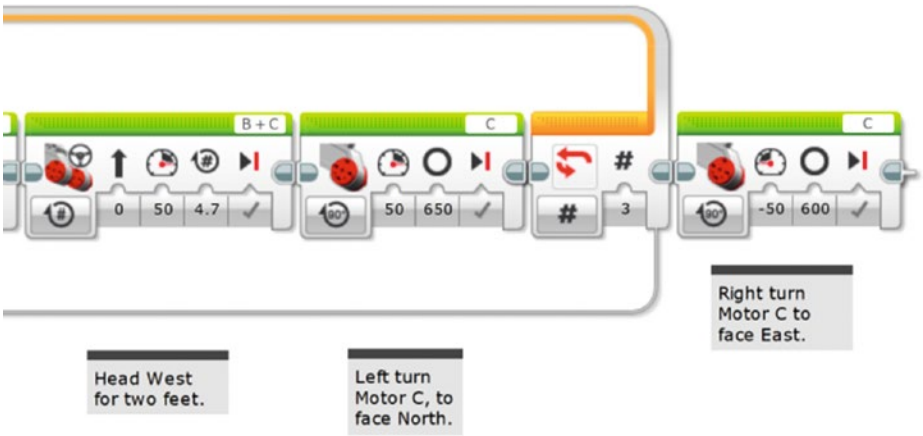


Figure 20-23. *This MEDIUM MOTOR block will spin the bot to the right in a clockwise direction*

Next I'll add another `MOVE STEERING` block that will take the bot to point A—approximately four feet (see Figure 20-24).

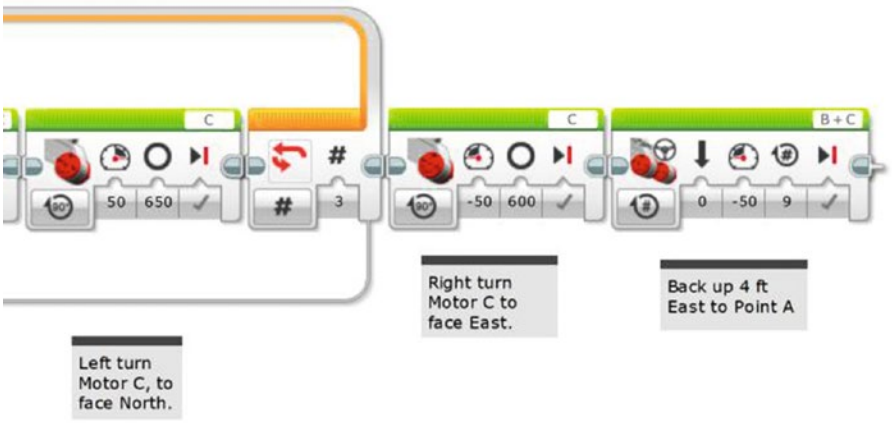


Figure 20-24. *This MOVE STEERING block moves the bot to point A*

My bot moves back nine rotations (approximately four feet) and stops. It's now at point A and I'm ready to get it to point B. It's important for me to have the bot centered with the figurine and the ramp. To do this, I'll perform plenty of tests using the Infrared Remote to get it to stop and turn at precisely the right point where I want it.

In goes another `MEDIUM MOTOR` block (see Figure 20-25) to get the bot to turn right (the bot's cage will be facing away from the first three figurines).

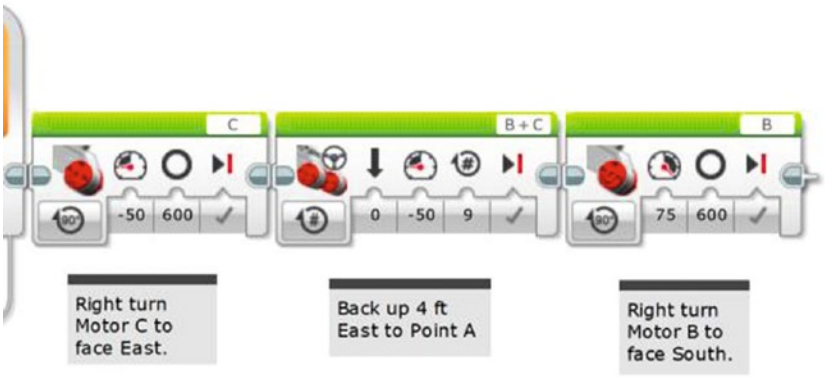


Figure 20-25. *This MEDIUM MOTOR block turns the PushBot toward point B, heading South*

Now I want the bot to slowly move down to point B. I say slowly because I'm going to watch it and trigger the Infrared remote to stop the bot when it reaches the center point. Again, this will take some practice to determine exactly where the bot should be when I yell "Stop!" And when I yell, I will also press that upper-left button on the Infrared Remote. The `LOOP` block, shown in Figure 20-26, will break when the Infrared remote is triggered.

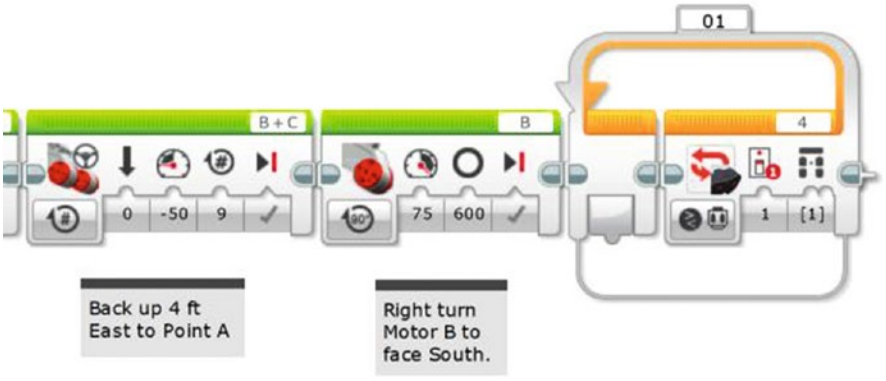


Figure 20-26. *This LOOP block will break when the Infrared Remote is triggered*

In Figure 20-27, I add the MOVE STEERING block that will slowly spin motors B and C. Depending on how fast and accurately I think I can push the button on the remote, I could also add a WAIT block configured for one or two seconds to give me time to watch as it approaches point B. In this case, I'm simply relying on the MOVE STEERING block with its fairly small 0.25 rotation to allow enough time to correctly trigger the turn toward the king figurine. Your own tests may show that the WAIT block would be a good thing to have.

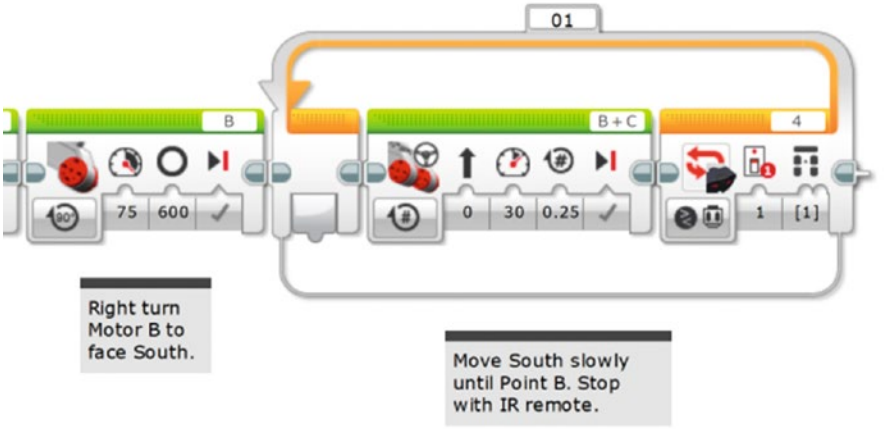


Figure 20-27. *The MOVE STEERING block is added to get the bot to point B. A WAIT block could be added after the MOVE STEERING block to gain better control.*

Once the PushBot reaches point B, it needs to make a left turn (cage facing the king figurine). I add a LARGE MOTOR block for this left turn (see Figure 20-28).

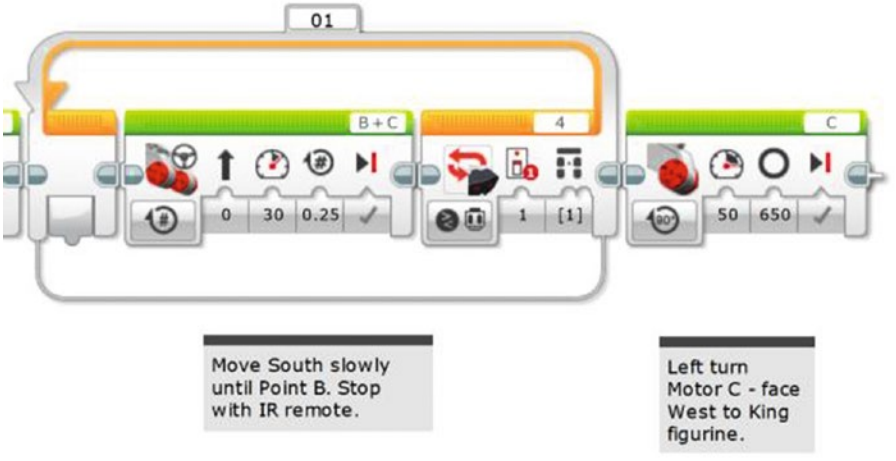


Figure 20-28. *This LARGE MOTOR block turns the bot to the left with the cage facing the figurine*

And now I need the bot to move (slowly) toward the fourth figurine. Once again, I'll use the Infrared remote to tell the bot when to stop and close the cage. I add a LOOP that is configured to break when the Infrared Remote signal is received (see Figure 20-29).

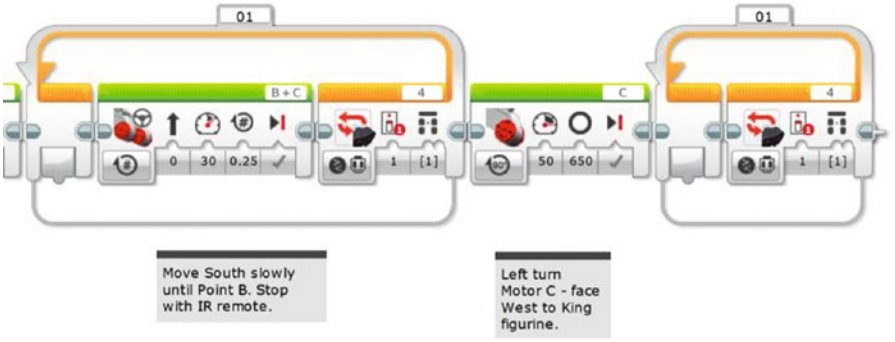


Figure 20-29. *This LOOP block will break when the Infrared Remote triggers the Infrared Sensor*

A MOVE STEERING block is added inside the LOOP block to move the bot toward the figurine in quarter rotations (see Figure 20-30).

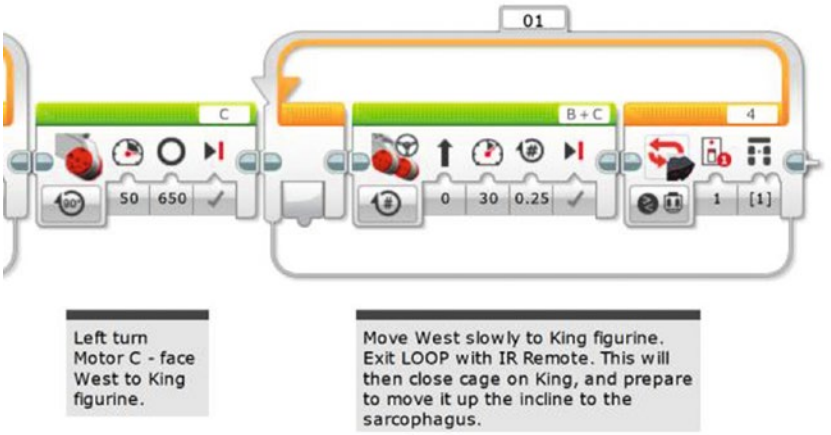


Figure 20-30. *This MOVE STEERING block moves the bot toward the final figurine*

When the Infrared Sensor is triggered by the remote, a MEDIUM MOTOR block will close the cage on the final figurine (see Figure 20-31).

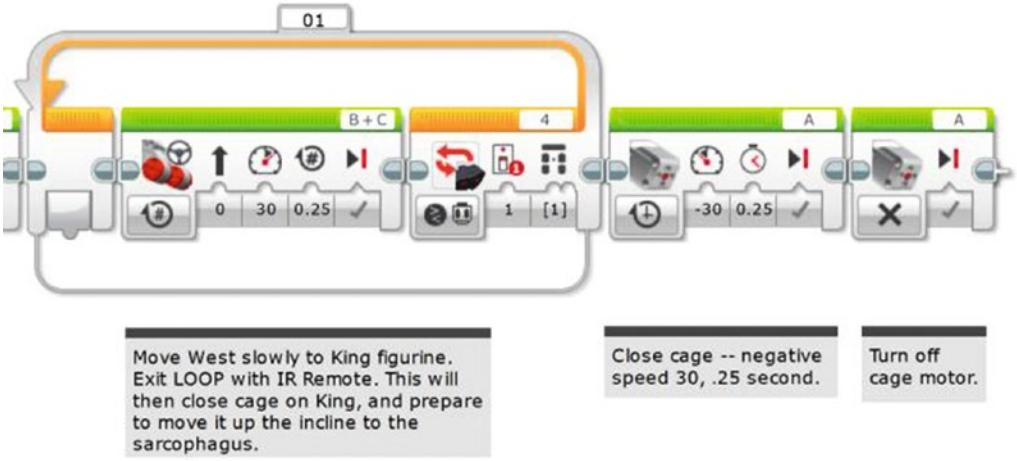


Figure 20-31. *This MEDIUM MOTOR block closes the cage on the final figurine. Then that motor is turned off.*

I add a final LOOP that will break when the Color Sensor is triggered by the black pressure plate on top of the sarcophagus (see Figure 20-32).

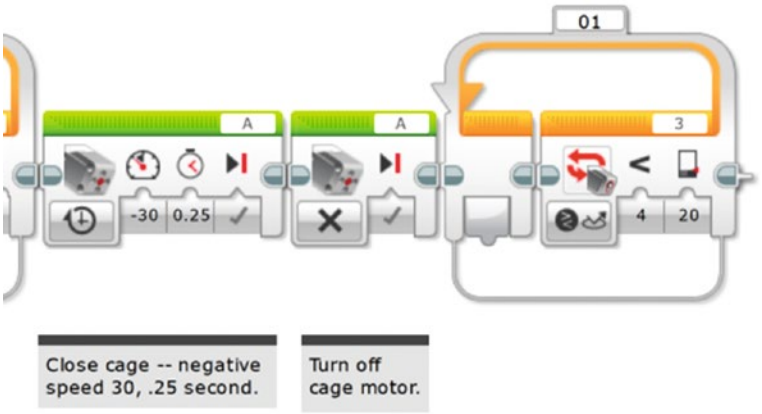


Figure 20-32. *This final LOOP block breaks when the Color Sensor is triggered*

This last MOVE STEERING block will spin motors B and C in half rotations until the Color Sensor is triggered and the bot stops (see Figure 20-33).

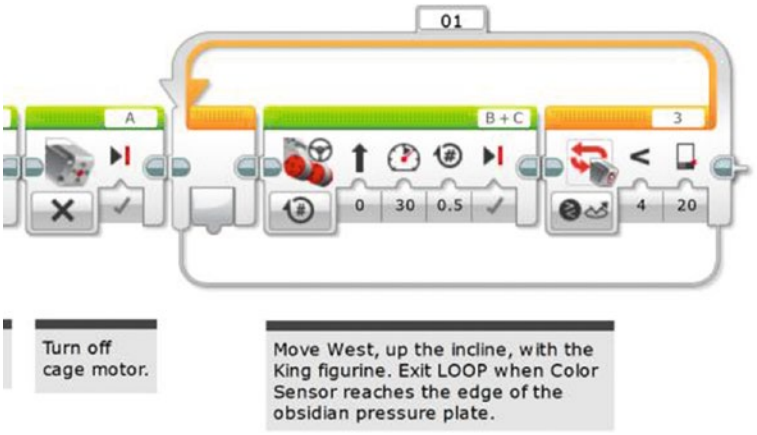


Figure 20-33. *This MOVE STEERING block stops spinning when the Color Sensor is triggered*

When the PushBot reaches the edge of the black pressure plate, the Color Sensor triggers the bot to stop moving. Then we need that last little push to get it to the center of the plate. The final `MOVE STEERING` block pushes it one more rotation, as shown in Figure 20-34, and winds up at the `STOP` command.

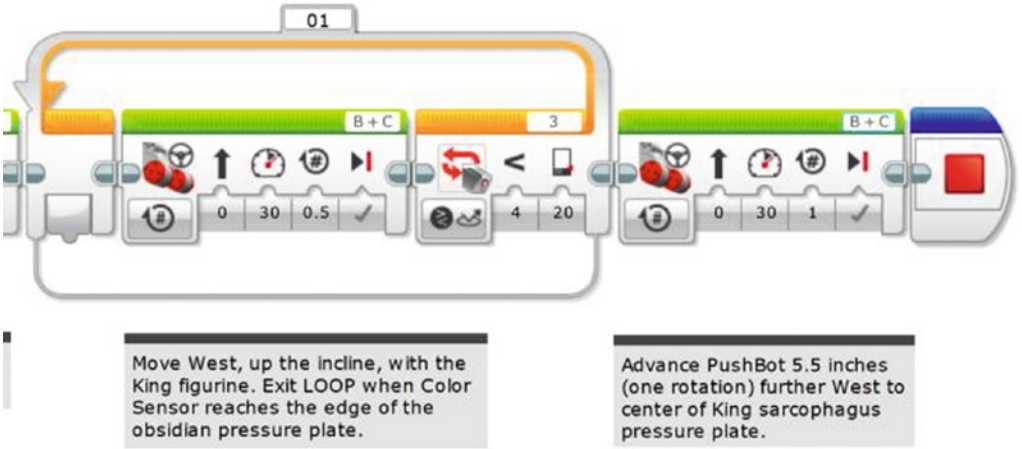


Figure 20-34. *The MOVE STEERING block in the LOOP stops spinning when the Color Sensor triggers. The final MOVE STEERING block gets it to the center of the sarcophagus pressure plate.*

At this point, the final figurine should be resting on top of the sarcophagus, in the center of the pressure plate.

If your robot's program succeeded, congratulations! If not, you'll need to test your bot quite a few times to make certain it works as expected. Set up your test environment and run as many as you can. When you're confident of your bot's abilities, run it one more time. If the bot accomplishes all the tasks, pat yourself on the back and get ready . . .

. . . It's time to investigate King Ixtua's burial chamber.

The story concludes in Chapter 21 . . .

CHAPTER 21

Discovery, Secret, and Home

Location: Southwest Guatemala

Weather Conditions: 86° Fahrenheit, Humidity 45%, Rain 50%

Day 7: Base Camp, King Ixtua's Tomb, 11:05 AM

The past few days had been a flurry of activity, with Evan assisting Uncle Phillip, Max, and Grace to take pictures and measurements, draw sketches, and weigh artifacts. Evan had never had this much fun in his life. He had taken part in opening King Ixtua's tomb and helped discover the king's burial chamber.

Evan sat enjoying a cold drink in the tent and stared at the massive collection of photographs that had been pinned up on every imaginable surface. Uncle Phillip sat across from him, enjoying a drink and a break from the heat.

"Well, Evan, have you enjoyed your time here?" asked Uncle Phillip.

Evan looked at his uncle with a shocked look on his face. "Are you *kidding*?" he replied. "All I want to know is where do we find another tomb?"

Uncle Phillip laughed. "I wish it were that easy. But I do have something you might find interesting," he said. "Put these protective cotton gloves on first."

Evan put on the soft gloves and watched as his uncle placed a small scroll on the table. He nodded for Evan to take a look.

Evan picked up the scroll and unrolled it. The surface was covered with Mayan glyphs. "What is this? What does it say?" asked Evan.

"That is a scroll that we found inside King Ixtua's sarcophagus. The fact that it wasn't in the king's library makes it something special. Grace has the better translation skills, so I've asked her to come and join us, but if that scroll contains what I think it does, you may get your wish," replied Uncle Phillip.

Evan stared back at the scroll. He carefully rolled it back up and placed it on the table.

"Uncle Phillip, this has been the best summer I've ever had," said Evan. "I don't think I've told you yet, but thanks for inviting me."

The tent flap separated and Grace entered carrying some books and a camera.

Uncle Phillip smiled. "Evan, you're welcome. But I have to say, if you had *not* come along, I don't honestly know if we would have been able to continue. So let me thank you for making the trip."

Grace nodded and set the items on the table. "Yeah, Evan. Max and I are really happy you came along, too. None of us has the proper skillset to train a monkey," she said with a smile.

"Have a seat, Grace," said Uncle Phillip. "Evan and I were just talking about this little scroll here. Can you take a look?"

Grace unrolled the scroll and studied it for a moment. She opened one of the books on the table and spent a few more seconds searching for something. She flipped a few more pages and then suddenly stood up. "This is unbelievable!" she yelled.

M. Bell and J. F. Kelly, *LEGO® MINDSTORMS® EV3*, DOI 10.1007/978-1-4842-2262-1_21

Uncle Phillip laughed out loud. “I thought so!” he replied.

“What is it?” asked Evan. Grace and Uncle Phillip were laughing together. Evan found himself laughing with the group.

Max stuck his head in the tent. “What’s all the noise in here?” he asked, looking at Evan.

Evan shrugged. “I have no idea.”

“Sit down, Max. I’ll let Grace explain,” said Uncle Phillip.

Max and Grace both sat down and Evan looked over at Grace. She held up the scroll and pointed at a small glyph. “I know we were laughing, but this is not the punch line to a joke,” she said. “This symbol right here is ‘quetz’la’ki,’ which translates to King’s Treasure.”

Max looked at Grace and then at Uncle Phillip. “Let me get this straight. Are you telling me that’s a treasure map?”

Uncle Phillip nodded with a smile. “I wasn’t certain until Grace translated, but I do recognize the other symbols as measurements and natural landmarks,” he said.

Grace pointed at some more symbols. “This scroll contains directions from this tomb to the location of King Ixtua’s treasure repository.”

“Are we all going to be rich?” asked Evan.

Uncle Phillip patted Evan on the back. “Not a chance, Evan. Anything we find belongs to the government of Guatemala. And that’s assuming this repository hasn’t already been found and looted,” he replied.

“But if it hasn’t?” Max asked.

“Well, as the team that discovered its location, the Guatemalan government will give us complete access and allow us to catalogue anything we find. But remember, the historical data from the scrolls and artifacts in King Ixtua’s tomb is going to keep us busy for years,” said Uncle Phillip.

Evan looked at his uncle, a little confused. “I have a question. Do you think the repository has traps like the king’s tomb?”

“I’m absolutely certain of it,” replied Grace. “This scroll matches the ones we have that were written by Tupaxu. Knowing Tupaxu, the repository probably has even more complex traps.”

“Well, I have to go back in a few days and start school,” said Evan. “I’ll be happy to leave my robotics kit, but I don’t think I’ll have time to explain everything about it.”

Uncle Phillip leaned forward and looked at Evan. “We probably won’t be able to start looking for this repository for five or six months. And we have another six months’ of paperwork to fill out for the Guatemalan government for King Ixtua’s tomb. Trust me, Evan. We won’t be able to tackle the repository for at least a year,” he said.

Evan smiled. “Are you saying what I think you’re saying?”

“Yes, Evan,” said Uncle Phillip. “I think it’s a safe bet that we might need your expertise again next summer, maybe summer after next, if we find King Ixtua’s treasure room. Think you might be interested?”

“Are you serious?” yelled Evan. “The only problem I can think of is all my friends will want to come along. I have three friends—Sean, Courtney, and Aaron—who have robotics kits, too.”

“Well, let’s take it one step at a time. You’ve got school and we’ve got a lot of paperwork to complete and reports to write. We’ll stay in touch with you during the year, and if everything works out, maybe you can join us next summer if we find the repository,” said Uncle Phillip, standing up. “Now, how about we have a nice lunch together and talk more about this treasure room.”

Max, Grace, and Evan all nodded.

“Okay, then,” said Uncle Phillip. “Lunch is on me. Grilled vegetables and lemonade in the next tent. Let’s go.”

Evan stood with the team, his mind racing with images of what kinds of traps and challenges the king’s treasure room would contain. He also had a new idea for a bot that he planned on designing on the flight home. It was going to be a busy school year. He made himself a promise to study more for his history class and walked out of the tent.

THE END?

More at http://ninevolt.ninja.

APPENDIX A

THE MINDSTORMS Community and EV3 Web Sites

LEGO has created a powerful tool for you to share your robotic creations with the world! It's a web site called MINDSTORMS Community, which I know you're going to love. This is an official LEGO web site, so it's totally safe for kids. The MINDSTORMS Community is constantly growing and changing, so some features might not be available when you visit. But you'll enjoy seeing all the other robots that people are submitting, and hopefully you'll start submitting your own. This appendix is a brief overview of how it works.

LEGO Club User Account

First, you create a LEGO Club user account (if you haven't already) so that you can access the MINDSTORMS Community site. Visit `http://club.lego.com` and click the Sign-Up button, shown in Figure A-1 (note that the background image on the button changes every few seconds).

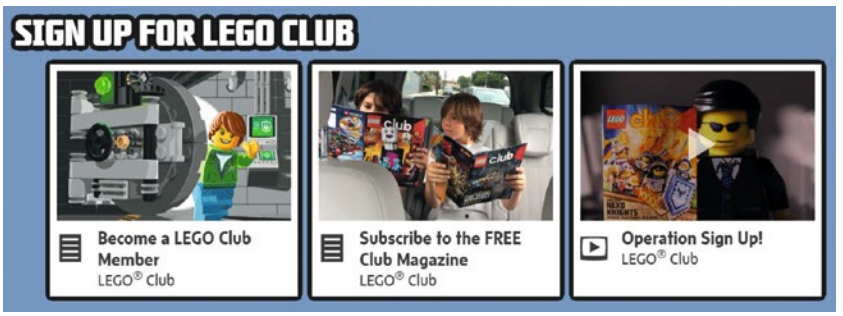


Figure A-1. *Create a user account on LEGO's web site*

If you are already a member, enter your username and password at the upper right and click Enter. If you're not a member yet, click the Sign Up button at the center of the screen. Enter your information, read the LEGO ID Terms of Service, and work your way to the final screen. After you've created your account, visit `http://mindstorms.lego.com` and log in to the MINDSTORMS EV3 page.

M. Bell and J. F. Kelly, *LEGO® MINDSTORMS® EV3*, DOI 10.1007/978-1-4842-2262-1_22

Community Gallery

When you are logged in to the MINDSTORMS Community site, as shown in Figure A-2, you can click Gallery in the menu bar at the top of the screen.

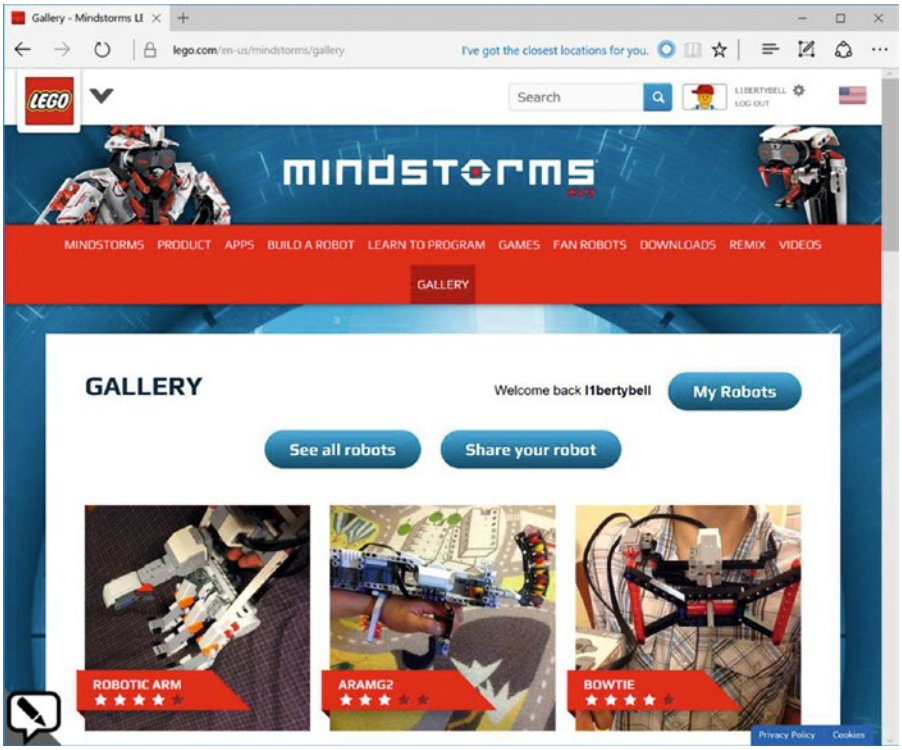


Figure A-2. *The MINDSTORMS Community Gallery screen*

The following are a few things that you can do from this page:

- *My robots:* You can view projects you have uploaded, and other club members can provide comments and reviews of your designs. You can also choose a picture for each of your projects and read comments others have left for you.
- *Share your robot:* You can upload pictures, videos, and notes about your robotic creations. Share your new robots with the world and give others a chance to test your designs.
- *See all robots:* If you're looking for design ideas, you'll have plenty of robots to examine. The "community" of robot designers is growing every day, so you'll need to check back often to see what's new.

Now, if you're wondering how you add steps to your projects, let me give you a small glimpse of how this works. In Figure A-2, if you click the Share Your Robot button, you can upload building instruction photos for your robots, one at a time. You can also share your EV3-G programs. If you post building instructions, other members will be able to build and test your designs, and maybe suggest improvements or upgrades!

■ **Note** Since you are posting your designs on the Internet, don't be worried or upset if others take your designs and modify them. That's just good robot design—and you should do the same! Take a design and change it, add new sensors or more motors, and definitely provide feedback on the MINDSTORMS Community site for the original designer and other members.

The MINDSTORMS Community is a great way for you to work with robot fans from around the world. Think about how much fun you'll have working on a robot project with friends from Europe, Japan, Australia, or other countries. You'll find teams of robot designers discussing project ideas and then submitting their designs. You'll be able to participate in competitions (such as "Design an EV3 robot that can change a light bulb") and even create your own.

The EV3 robotics community is going to give you plenty of ideas, and you'll never run out of robots to experiment with and program and test. Have fun!

Online Reference and Support

You should have received a printed *TRACK3R Instruction Guide* with your LEGO MINDSTORMS EV3 retail kit. The detailed PDF of the full manual is available online at https://www.lego.com/en-us/mindstorms/downloads/user-guide. Building instructions for the other kit robots are readily found at https://www.lego.com/en-us/mindstorms/build-a-robot. But that's not the end of the reference material available for your robotics kit. The amount of information available online is substantial and growing fast.

There is no way for me to include everything available for your EV3 reading pleasure. There are new web sites, blogs, and training materials popping up every day. From videos to pictures to new operating systems and software, the EV3 world has enough to keep you busy for quite a while. Let's look at a few.

Web Sites

When it comes to web sites, there are plenty. The web sites I'm including here have proven to be kid-friendly:

- http://www.mindstorms.com: The official web site of the LEGO MINDSTORMS EV3. This site offers videos, building instructions, plenty of cool downloads, interviews with EV3 designers and demos of their designs, links to blogs and other web sites, and access to online forums. The site is always changing, so check in frequently to see what's new.
- http://www.education.lego.com: The companion web site for LEGO Education. This web site has interesting articles to read (a good source for future bot ideas) and offers challenges that will improve your robot design skills.
- http://mynxt.matthiaspaulscholz.eu/lego-mindstorms-ev3.html: Matthias Paul Scholz lives in Germany and has worked with me on many projects, including TheNXTStep blog (see the "Blogs and Forums" section next). His web site includes videos, pictures, and instructions for many strange and unusual bots that he's designed. He also provides links to other web sites and blogs.

- http://www.firstlegoleague.org: FIRST LEGO League is a series of robotic competitions for students ages 9 to 14. There are local competitions that culminate in an annual international competition, with teams from around the world coming together to display some excellent robotic design and programming skills.
- http://www.philohome.com: Phillipe Hurbain's web site covers some fairly advanced robotic subjects, but it's always fun to visit. Philo (his nickname) likes to take his Bricks apart, so don't try and copy him when he does. It could damage your EV3.

There are *plenty* more web sites that focus on the EV3, and new ones are popping up all the time. You might want to even consider starting your own web site on the EV3!

Blogs and Forums

Blogs have been growing in popularity for a few years now, and the number of blogs related to the EV3 continues to increase. Forums and message boards are popular places for you to share your comments, questions, and robot designs with other EV3 fans. Be aware that most forums require you to register with a username and a password to use the site. Be sure to read the rules for what you can post and what you cannot post. Violating the rules can get you banned from using the forum in the future.

Here are some popular, frequently updated blogs and forums that are focused on the EV3 robotics kit, related products, and news:

- http://thenxtstep.com: I contribute to this blog with a handful of EV3 colleagues. We cover news and products related to the EV3 and more. It's a good place to check for updates on new products as well as information on new web sites, forums, and blogs. (The name *thenxtstep* reflects this site's origins in the earlier LEGO NXT world. But it does actively deal with our EV3 work!)
- https://community.education.lego.com: This forum focuses on the MINDSTORMS Education EV3 kit and includes news and updates pertaining to MINDSTORMS EV3 and the classroom. You need your LEGO ID to log into this site.

As with web sites, there are many more blogs out there dedicated to the MINDSTORMS EV3 kit. You might even want to consider starting your own; visit http://www.blogger.com or http://wordpress.com to learn how to create your own blog, absolutely free. Post news of your EV3 bot designs with pictures and video and you may find yourself developing a growing base of fans who visit your blog often.

APPENDIX B

Robot Commander Remote Control App

LEGO has helpfully given us an app that can operate every aspect of your EV3 robots. It can control motors, both large and medium. It can give you a joystick or slider bars to control motors singly or in combination. It can also report the status of your sensors. This is exceptionally useful for testing a new design without writing any software.

Downloading and Running the Robot Commander App

Figure B-1 shows a screenshot from the LEGO Robot Commander app download page. As shown at the bottom, you can download the Android version from Google Play, or the iPhone version from the App Store. These downloads are free.

The app comes with pre-configured controls for the five Kit Robots shown on the box with the EV3 Retail Kit. However, for our purposes, we want to be able to create custom command screens for the robots in this book, or others you may build.

Figure B-1 shows two possible user-configured control screens. The center smartphone screenshot demonstrates how you can position the control elements anywhere on the smartphone screen. In this case, a dual motor control is at the upper left, and a slide bar is below that. The dual-motor control would typically operate both large motors, possibly as regular wheels, a tracked vehicle, etc. The slide bar would then operate the medium motor, say as a gripper jaw as seen with the PushBot.

M. Bell and J. F. Kelly, *LEGO® MINDSTORMS® EV3*, DOI 10.1007/978-1-4842-2262-1_23

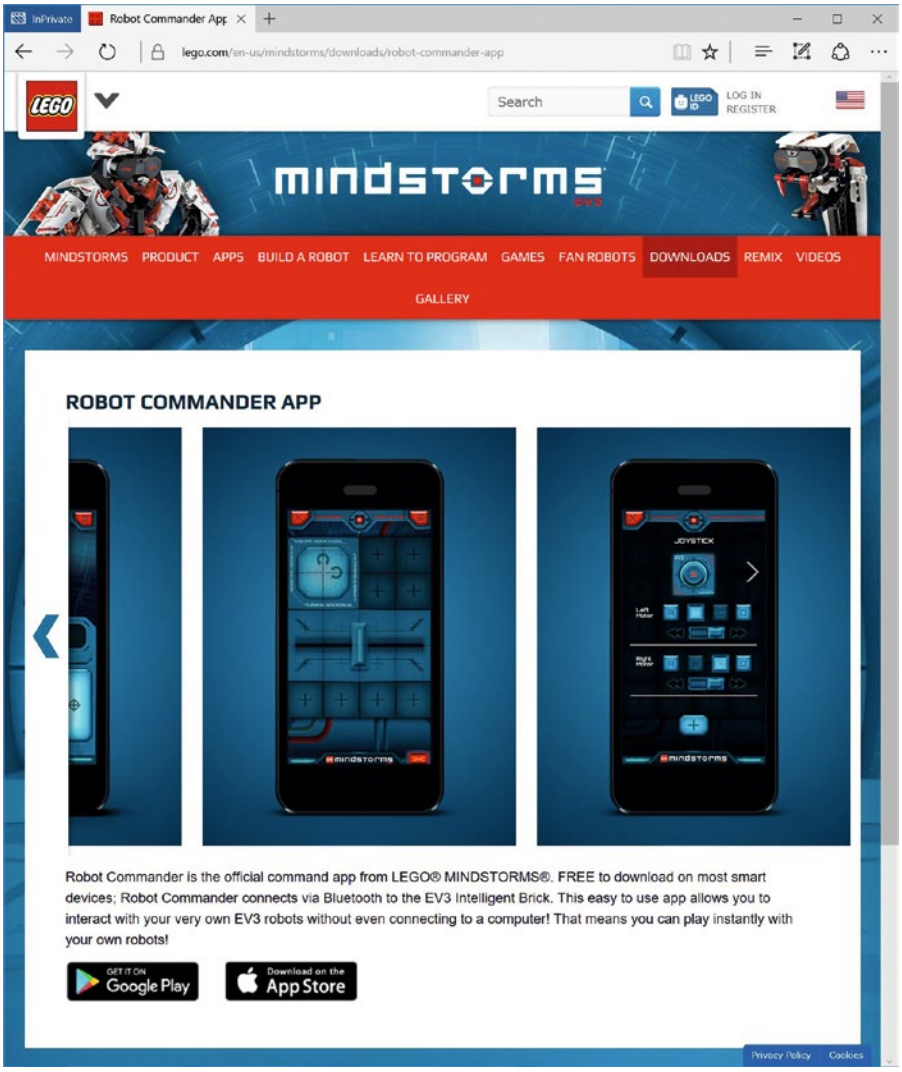


Figure B-1. *A screenshot from the LEGO Robot Commander download page. Two configuration screens are seen in this view.*

The Perfect Hardware-Testing App

The Robot Commander app played an important role in the design of the PushBot. In Figure B-2, we see the PushBot from the Grabber Jaw end. This design "looks good"—that is, it *should* be able to close the jaw on a Mayan Figurine (or a bottle of water). And the bottle *should* be able to fit nicely in the jaws. And the bot *should* be able to push the bottle on to a pressure plate.

But will it? When you do this kind of testing, the Robot Commander app is (you guessed it) your friend. As you design a bot, you can test it as soon as your motors are fastened and wired up. You configure the app with controls and try it! First, create a joystick for the drive motors and a slide bar for the Grabber Jaw. Then turn Bluetooth on for your bot. Synch them up and the app is in control.

I tried a few different hardware design tweaks for the PushBot. I tested each one (drive, grab, push, release, back up) and made hardware improvements each time. And the design you see in Figure B-2 works great. At that point, I sat down and began working on the actual downloadable software as described in Chapter 20. I knew the hardware would work if the software were designed correctly.

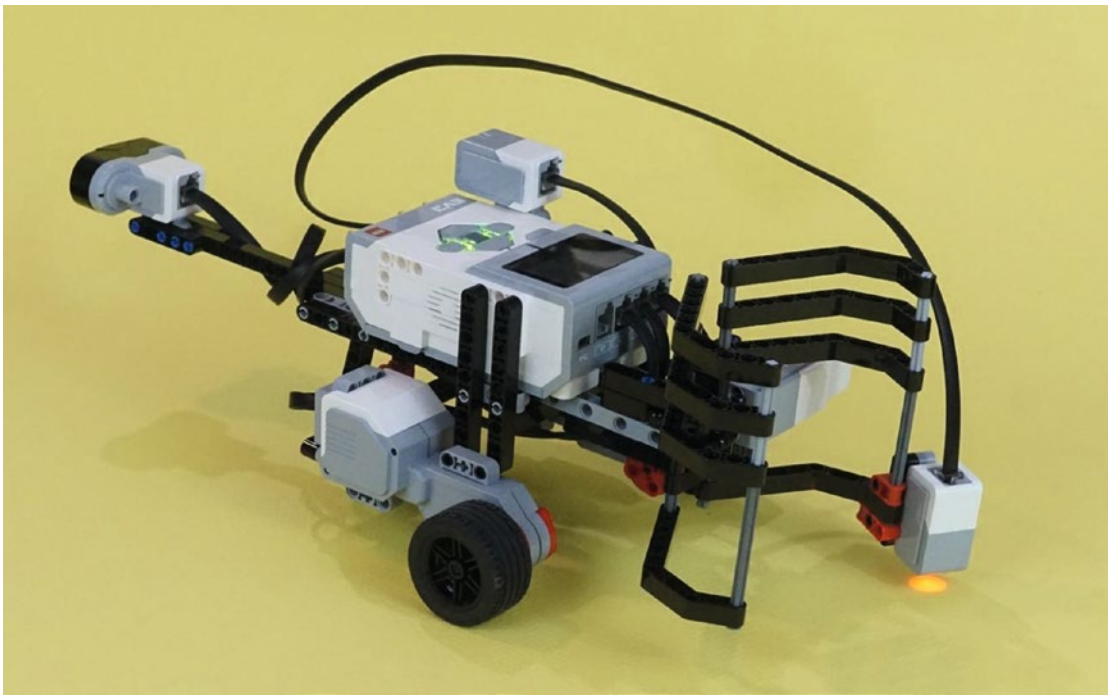


Figure B-2. *The PushBot after being tested and hardware-validated with the Robot Commander app*

Engineering: Isolating Design Issues in Testing

And now for our final Engineering sidebar. The Robot Commander offers us a valuable service in design and testing. If we simply built the PushBot and downloaded software to run it, we'd probably find something not working. And we might have to puzzle out whether the problem was hardware or software. But the Robot Commander has isolated the hardware from the software. So we can build a bot that *should* work and verify the hardware before writing any software.

So if something doesn't work in software testing, we can dig into the software problem with confidence the hardware is okay. After all, remember all those times we said "The hardware *should* work?" Well, we saw it work! And the Robot Commander app was the key to that testing.

APPENDIX C

Kit Organization: Where Do All Those Parts Go?

Where do you put all of those parts? A multi-compartment box is ideal, so parts can be kept in categories. Figures C-1 and C-2 show a top and bottom stacking storage box with places for everything.

The Top Tray for the Small Parts

In the top section of the box (see Figure C-1), the categories are chosen to keep things in small enough groups to fit in each compartment. Also, the parts in each group relate to each other. Table C-1 describes the contents of each section.

Table C-1. *How the Small Parts Are Organized*

Connector Pins: Short black, long blue, and long red	**Short Red Connectors:** Any connectors with axle holes as well as pin holes	**Odd Connectors**: Don't seem to go anywhere else	**Axles**: Including collars, short axle pins, and nailhead axles
Gears: Regular, worm, and elastic band	**Double Connectors**: Wide and narrow	**Angle Beams**: Tees, ells, and angled beams	**Straight Beams**: From 3-hole up to 15-hole

M. Bell and J. F. Kelly, *LEGO® MINDSTORMS® EV3*, DOI 10.1007/978-1-4842-2262-1_24

Figure C-1. *An eight-compartment tray for all of the small parts*

This is one suggested system for storing the parts. The Axle category is surprisingly useful, since all axles as well as all collars are here. You'll be using these constantly.

Bottom Tray for the Large Parts

Figure C-2 shows the large EV3 parts. (The top tray is off to the side). The categories are not as specific as those in the top tray. Since the parts are larger, it is easy to see what is where. One worthwhile idea is to keep all wheel-related parts together, since some of them are small and need a home where you can reliably find them.

Figure C-2. *A six-compartment tray for all of the large parts*

A Secure Lid Is Your Friend

Finally, Figure C-3 shows a secure-fitting lid for the box with both trays inside. If at all possible, find a container with a good lid. Sooner or later, you'll accidentally kick the kit or knock it over. At that point, that lid is your *friend*!

Figure C-3. *The lid that shall bind them all together. Securely.*

APPENDIX D

Building Instructions for Bots

If you build a unique bot of some sort, someone with an EV3 kit might ask you for instructions on building a duplicate. There are numerous methods for demonstrating how to build a robot that you have designed. One easy method is to simply digitally record yourself building it, talking as you go and showing to the camera the pieces you are using and where you place them. I'd like to also introduce to you something called CAD (computer aided design) software. This type of software allows you to create accurate drawings of your robot designs as well as step-by-step instructions that can be printed or viewed on a computer screen; examples of CAD programs used for LEGO creations include LDraw (http://www.ldraw.org), MLCad (http://www.lm-software.com/mlcad), and from LEGO, the LEGO DIGITAL DESIGNER (http://ldd.lego.com).

For this book, I've been using photographs taken with a digital camera. With the digital camera, I can immediately upload the photo to my computer (a laptop in this case) and view the image. If it's blurry or doesn't show quite what I wanted to capture, I can delete it and take another.

This appendix is a short tutorial containing some tips and suggestions I want to pass along. I've learned a lot from photographing this book's bots and I'm hoping some of my experiences can help you if you choose to create building instructions (BI) from photos of your own bots.

What Will the Background Be?

One of the biggest mistakes I made early on was to photograph the construction of my bots against a white background. After converting the photos to black-and-white images, what I found was that the colors of most of the EV3 parts (off-white, light gray, dark gray) just didn't show up very well when I placed a white posterboard under the parts. Take a look at Figure D-1. On the left is a bot against a white background, and on the right is the same bot against a yellow background. After converting the photograph to a black-and-white image, which do you think looks better?

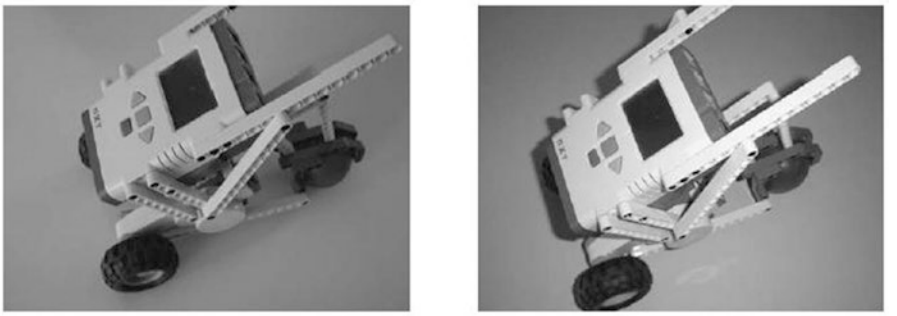


Figure D-1. *The background can make all the difference*

M. Bell and J. F. Kelly, *LEGO® MINDSTORMS® EV3*, DOI 10.1007/978-1-4842-2262-1_25

I also tried photos using blue, green, red, and gray posterboard. What I found was that yellow or light-blue posterboard worked best. Whether you convert the pictures to black-and-white or not, photographing your bot's assembly against a colored background instead of white works much better. Even better is to use a colored background with some texture.

Step by Step Picture-Taking Strategy

I'll jump straight to the secret for taking great BI photographs: Build your bot *first*. Get it perfect, the way you like, and make sure it works, and then start taking pictures as you take the bot apart.

You're right, this is not a big secret. You'll photograph the bot as it's disassembled and then reverse the photographs. Simple. But I do have a couple of suggestions for you:

- *Your first photograph should be of the completed bot:* When you are done, you'll reverse the picture order, so this will actually be your last building instruction photo. Whatever angle you use to take a picture, that is the angle you need to keep for the next photo.
- *Without moving the bot, remove a part and set it down close to the bot:* Take the photograph (from the same angle as the first) so that it shows the bot and the removed part, and make sure that where the part was removed is visible in the new photograph. Look at Figure D-2. On the left side is the final bot, and on the right you can see that a part has been removed. You should be able to determine from the photo on the left where the parts on the right will be placed. Good building instructions always show you a part in one step that you'll use in the next step (which is why the photo on the left has two new parts next to it).

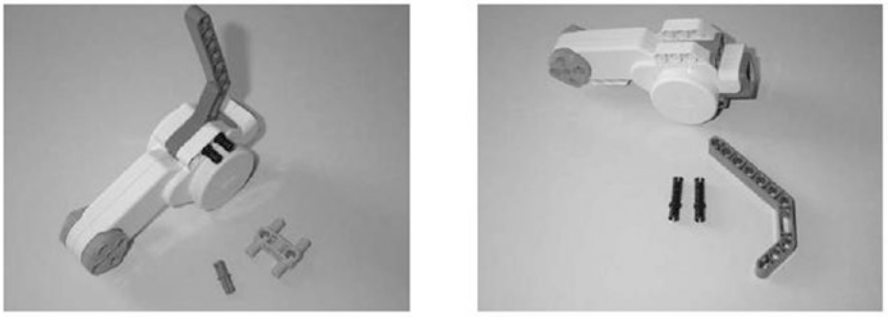


Figure D-2. *Look at the image on the left to determine the placement of the part on the right*

- Remove multiple parts if they are all visible in the most recent photograph: If you have three 15-hole beams and two small black connectors that can be immediately removed and are not hidden by other pieces, feel free to take all of them off. Place them in an organized fashion near the bot (or what's left of it) and take a picture. As long as the locations of all the parts you are removing are visible in the previous photo you took, everything should be fine.

As you take more and more photographs, you'll find even better methods for taking photographs and discover things to avoid. For example, you can't remove a part on a nonvisible side of the bot that you just photographed. If you take off a nonvisible part, place it next to the bot, and then photograph it, how will others using your instructions know where to place the part? They'll see the new part to add, but when they look at the next photograph, the part you removed will not be visible.

After you've completely disassembled your bot, take the pictures and put them in reverse order. Using a digital camera and uploading the photos to your computer, rename the images Step 1, Step 2, etc. The first image will show the first piece (or first few pieces) of your bot being placed or connected together.

Now, use your new building instructions and try to build your bot. If you took good photographs, you should be able to build your bot again. If anything is confusing, take a picture to capture the proper placement of a part or two where you found the problem. You can give the picture a name such as Step 3b if it fits between Steps 3 and 4.

When you are happy with your building instructions, you can use image editing software to add text to your pictures if you like. Then, upload the images or print them out and share them with others. You can also post your steps to the MINDSTORMS Community (see Appendix A). And now your bot design can be re-created anytime.

After taking the picture, inspect it closely in the LCD screen of the camera and delete any picture that is not in sharp focus or doesn't clearly show what you want to show. It's digital, so you can always take another picture, but the model will never be at this particular step again, so check now, and don't regret it later.

Lighting, Camera, and Lens Choices

Lighting: If at all possible, photograph in strong, indirect light. The built-in flash on most cameras tends to "flatten" the image, removing or reducing depth cues. If you can use a strong, even light source from a different direction than the camera, you have much more control over the lighting and the depth cues in the picture. An inexpensive lighting kit can be as little as $50. Lights, stands, and two white umbrellas make this indirect illumination much easier. Figure D-3 was shot with umbrellas, one to the right and one to the left.

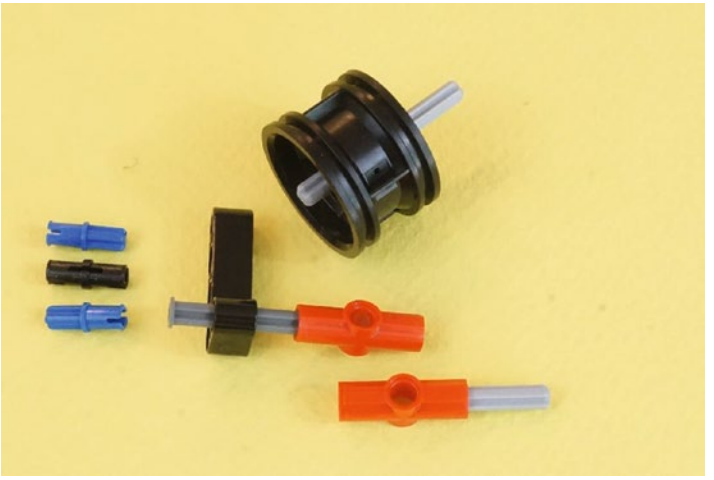


Figure D-3. *This image is well lit from the sides, F22, excellent depth of field. But is there some dust?*

Depth of Field: For most photos, you'll need a great depth of field. This means you might shoot at F22 so that items in the foreground are in sharp focus, along with items in the background. A step-up in quality also results from using a non-zoom lens. The photos in this book were taken with a 35mm equivalent lens. This relatively wide-angle lens gives greater depth of field than a 50mm or longer lens.

Exposure: You may want to overexpose your photos for greater detail in dark areas. All of the photos in this book, including Figure D-4, were shot with automatic exposure set to 0.7 stop (that is, 0.7 EV) overexposure. By using an automatic camera, the 0.7 EV boost is consistent across photos. This overexposure, as an example, makes the black details on the wheel hub of Figure D-3 much clearer.

Dust on the Sensor: An unwelcome phenomenon can show up when you're shooting at F20 or F22—dust on the digital sensor. We don't notice this dust at stops of F2 or F4, but as the aperture closes down the dust becomes more visible. In Figure D-3, we see a normal shot. But in Figure D-4, an enlarged portion of Figure D-3, the dust is more visible. This can be tricky to clean off the sensor but may be essential if you want to avoid a lot of Photoshop after-the-fact cleanup. The camel hair photo brush and canned air "can" be your friends.

Figure D-4. *Enlargement of Figure D-3. What are those little gray dots doing there? Very possibly, they're dust on the sensor.*

Index

A, B

Base camp, King Ixtua's Tomb, 313

C, D

Community gallery, 316

E, F

ExploroBot, 7, 41
  construction, 17
    assemble all parts, 33
    bot body and motors, 21
    infrared sensor
      and neck, 18
    intelligent brick, 24
    rear-wheel assembly, 26
    reinforcement strut, 26, 31
  description, 9
  into the tunnel, 45
    comment tool, 50
    large motor block, 51
    LOOP block, 46–47
    motor encoder, 49
    move steering block, 48–49
    nested Loops, 47
    WAIT block, 50
  issues, 11
  LEGO MINDSTORMS EV3
      software, 41
  limitations and constraints, 11
  LOOP block, 45
  Mindstorm, 13
  motor rotations
      measurement, 53
  move steering block, 43
  out of tunnel, 51
    LOOP blocks, 52
    move steering blocks, 52
  sketches, 15
  task list, 10, 43
  WAIT block, 45

G, H

GrabberBot, 175, 183, 217
  connectors, 214
  description, 175
  down the tunnel, 217
    LOOP block, 218
    medium motor block, 227
    MOVE block, 228
    move block controls, 220
    move tank block, 222
    parallel paths, 228
    parallel process, 225
    rotations and degrees, 219
    test environment, 221
    touch sensor, 218
    WAIT block, 224, 230
  elastic latch, 213
  Grabbber assembly, 194
  infrared sensor, 210
  Jaw open, 216
  Jaw shut, 213
  limitations and constraints, 178
  Mindstrom, 178
  sensor port, 215
  sketches, 181
  tank-tread motor, 184
    15-hole beams, 189
    assembly, 188, 191
    components, 184
    gear-wheel assembly, 186
    nailhead axles, 185
    reinforcing beams, 193
    short pins, 185
  task list, 176
  touch sensor, 212

I

Inside King Ixtua's Tomb, 57
  Evan's solutions, 60
  reception room sketch, 58
  Vine challenge, 59

M. Bell and J. F. Kelly, *LEGO® MINDSTORMS® EV3*, DOI 10.1007/978-1-4842-2262-1

J

Jaw Cage mechanism, 244

K

King Ixtua's Library, 169
  burial chamber located, 170
  Max's solution, 174
  scroll challenge, 172
  throne room, 169
King Ixtua's Throne Room, 231
  burial chamber, 232
  Evan's solution, 235
  figures, 233
  final challenge, 234
King Ixtua Tomb excavation, 1
  Evan's solution, 4
  tunnel challenges, 3
  tunnel drawing, 2
Kit organization, 323
  bottom tray, 324
  secure-fitting lid, 325
  top tray, 323

L, M, N

LEGO Club user account, 315

O

Online reference and support, 317
  blogs and forums, 318
  web sites, 317

P, Q

Photographs
  background, 327
  depth of field, 330
  dust on the sensor, 330
  exposure, 330
  lighting, 329
  picture-taking strategy, 328
Planning and design process, 7, 61
  ExploroBot, 7
    description, 9
    limitations and constraints, 11
    Mindstrom, 13
    sketches, 15
    tasklist, 10
  GrabberBot, 175
    description, 175
    limitations and constraints, 178
    Mindstrom, 178
    sketches, 181
    task list, 176
  PushBot, 237
    description, 237
    limitations and constraints, 240
    Mindstorm, 241
    sketches, 242
    task list, 239
  SnapShotBot, 117
    description, 118
    limitations and constraints, 126
    Mindstorm, 126
    sketches, 128
    task list, 119
  StringBot, 61
    description, 62
    limitations and constraints, 64
    Mindstrom, 66
    sketches, 69
    task list, 63
PushBot, 237, 243, 291
  description, 237
  engineering
    difference between designing and explaining, 244
    double gray connector, 285–286
    mirror-image large motors, 282
    prototypes and parts, 280–281, 284, 289
    rear-wheel designs and constraints, 260, 262, 270, 272
    reinforcing frame, 280
    stiffness II, 252–253, 257, 259
    Touch Sensor, 286–287
  figurine jaw cage/medium motor mechanism, 244, 252
    bottom-axle sub-assembly, 247–248
    five-hole beam, 250
    L-beam, 249
    L-beam with axle and gear, 251
    medium motor mounted, 246
    nailhead axle, 251
    partially assembled parts, 254
    pins and axles, 245
    rounded gears, 246
    sub-assembly, 254
    with Color Sensor, 256–258
  final figurine, 306
  into position, 291
    infrared sensor, 294
    large motor block, 295–296
    Loop block, 292
    move steering block, 294
  limitations and constraints, 240

main body and motors, 275, 280–281, 284, 289
black connectors, 278
blue connectors, 278
reinforcing beam, 279
Mindstorm, 241
neck/infrared sensor frame assembly, caster wheel, 260, 270, 272
blue connectors, 267
bottom side, 274
five-hole beams, 263, 271–272
gathering parts, 265
gray double connector, 262–263
middle gray connector, 264
three double connectors, 262
sketches, 242
task list, 239
three figurines position, 297
Large Motor block, 304
LOOP block, 297, 300
LOOP block, 301
Medium Motor block, 300, 302
move steering block, 298, 303–304

R

Robot Commander app, 319
pre-configured controls, 319
testing, 320
design issues, 321
hardware design, 321
user-configured control, 319

S

SnapShotBot, 117, 129, 155, 260
angel crank, 165
description, 118
infrared sensor, 162
Infrared sensor, 155
large motor block, 156, 164
limitations and constraints, 126
loop block, 158, 160, 163
medium motor block, 157
Mindstorm, 126
move steering block, 155
peg-leg, 129
camera frame, 146
camera trigger, 138
color sensor, 135
concept of *stiffness*, 134
corner reinforcements, 142
gear trade speed, power, 143
Gear Yoke, 139, 141
hardpoints, 131, 135
IR sensor, 137
testing, 149
sketches, 128
steering block, 159, 164
task list, 119
test environment, 161
wait block, 157
StringBot, 61, 71, 93
construction, 71
assemble all parts, 89
carrier and pebble, 85
intelligent brick, 89
left-side motor, guides and rubber wheel, 77
right-side motor clips, 91
right-side motor, IR Sensor, 73
string up, 92
wire assignments, 90
description, 62
limitations and constraints, 64
LOOP block, 97, 104
Medium Motor block, 102–103
Mindstrom, 66
MOVE block, 98, 104
sketches, 69
STOP block, 105
STOP block, 96
SWITCH block, 96
task list, 63, 94
testing, 106
WAIT block, 99, 101
WAIT block, 100

T, U, V, W, X, Y, Z

Tomb door, 55
Tomb reception area, 109
challenges, 112
Grace's Solution, 114
King's library, 110