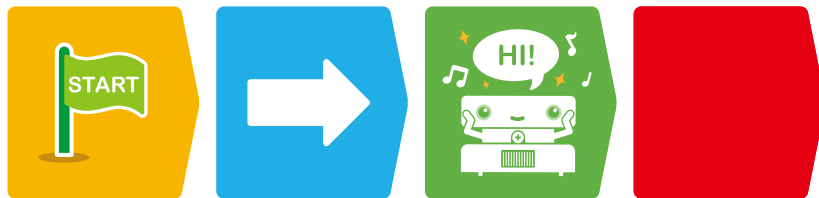




# CODING & ROBOTICS



## Lesson Plan and Experiment Manual

## IMPORTANT INFORMATION

### Safety Information

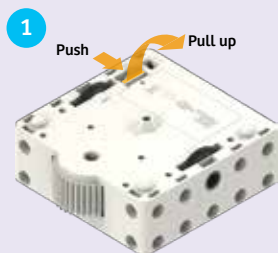
**WARNING!** Not suitable for children under 3 years. Choking hazard — small parts may be swallowed or inhaled.

Keep the packaging and instructions as they contain important information.

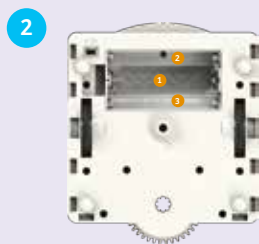
Store the experiment material and assembled models out of the reach of small children.

### Inserting and Replacing the Batteries

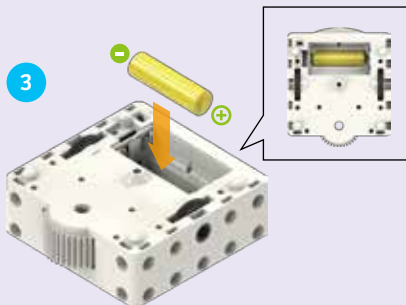
An adult must insert and replace the batteries inside the robotic base unit. Here are the instructions for inserting and replacing the batteries. You will need three AA batteries.



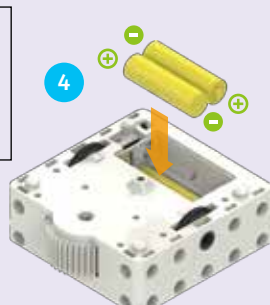
1. Push the tab on the battery compartment lid inward and then pull up on the tab to open the battery compartment.



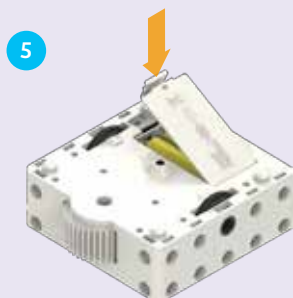
2. Look at the markings inside the compartment to determine the correct battery orientations.



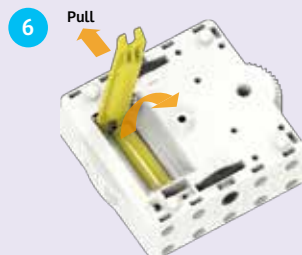
3. Install one AA battery into the lower level of the battery compartment first.



4. Then, install two more AA batteries in the upper level of the compartment.



5. Push the lid back onto the battery compartment.



6. To remove the batteries, you can use the included part separator tool to pry them up.



**WARNING:**  
**CHOKING HAZARD** — Small parts.  
Not for children under 3 yrs.

### Safety for Experiments with Batteries

- » To operate the models, you will need three AA batteries (1.5-volt, type AA/LR6) or three AA rechargeable batteries (1.2-volt, type AA, HR6/KR6), which could not be included in the kit due to their limited shelf life.
- » The supply terminals are not to be short-circuited. A short circuit can cause the wires to overheat and the batteries to explode.
- » Different types of batteries or new and used batteries are not to be mixed.
- » Do not mix old and new batteries.
- » Do not mix alkaline, standard (carbon-zinc), or rechargeable (nickel-cadmium) batteries.
- » Batteries are to be inserted with the correct polarity. Press them gently into the battery compartment.
- » Always close battery compartments with the lid.
- » Non-rechargeable batteries are not to be recharged. They could explode!
- » Rechargeable batteries are only to be charged under adult supervision.
- » Rechargeable batteries are to be removed from the toy before being charged.
- » Exhausted batteries are to be removed from the toy.
- » Dispose of used batteries in accordance with environmental provisions, not in the household trash.
- » Be sure not to bring batteries into contact with coins, keys, or other metal objects.
- » Avoid deforming the batteries.

As all of the experiments use batteries, have an adult check the experiments or models before use to make sure they are assembled properly. Always operate the motorized models under adult supervision.

After you are done experimenting, remove the batteries from the battery compartment. Note the safety information accompanying the individual experiments or models!

### Notes on Disposal of Electrical and Electronic Components

The electronic components of this product are recyclable. For the sake of the environment, do not throw them into the household trash at the end of their lifespan. They must be delivered to a collection location for electronic waste, as indicated by the following symbol:



Please contact your local authorities for the appropriate disposal location.

## CURRICULUM CORRELATION

The Computer Science Teachers Association (CSTA) is a membership organization with the mission of empowering, engaging, and advocating for K-12 computer science teachers worldwide. The CSTA has developed the **CSTA K–12 Computer Science Standards** to “delineate a core set of learning objectives designed to provide the foundation for a complete computer science curriculum and its implementation at the K–12 level.” The following table is Thames & Kosmos’ analysis of how the lessons in this kit correlate to the CSTA standards.

Curriculum Identifier	Grades	Standard Description	Concept	Correlation to this Kit
1A-AP-08	K–2	Model daily processes by creating and following algorithms (sets of step-by-step instructions) to complete tasks.	Algorithms	In Lesson 1, children create an algorithm to make a sandwich. In many lessons, children write programs to model the daily activities of the characters described in the stories.
1A-AP-09	K–2	Model the way programs store and manipulate data by using numbers or other symbols to represent information.	Variables	Covered in the lessons that use the number cards, especially the math lessons
1A-CS-03	K–2	Develop programs with sequences and simple loops, to express ideas or address a problem.	Control	Covered in all lessons, especially those involving simple loops
1A-AP-11	K–2	Decompose (break down) the steps needed to solve a problem into a precise sequence of instructions.	Modularity	Covered in all lessons due to the step-by-step nature of the code cards, and especially in the lessons involving subroutines (functions)
1A-AP-12	K–2	Develop plans that describe a program’s sequence of events, goals, and expected outcomes.	Program Development	Covered by lessons in which children create programs on their own, especially Lesson 8
1A-AP-14	K–2	Debug (identify and fix) errors in an algorithm or program that includes sequences and simple loops.	Program Development	This will naturally come up in the course of testing and correcting the arrangement of code and map cards to get the robot to work as expected.
1A-AP-15	K–2	Using correct terminology, describe steps taken and choices made during the iterative process of program development.	Program Development	This can happen when parents or teachers engage with the children, asking them to describe their specific programs and how they developed them.
1B-AP-08	3–5	Compare and refine multiple algorithms for the same task and determine which is the most appropriate.	Algorithms	There are many lessons in which more than one program is suggested as a solution. For example, Lesson 9 shows four ways to write the code.
1B-AP-09	3–5	Create programs that use variables to store and modify data.	Variables	This is modeled in Math Lesson 4.
1B-AP-10	3–5	Create programs that include sequences, events, loops, and conditionals.	Control	Covered by any lesson in which a conditional statement is used, starting with Lesson 18.
1B-AP-12	3–5	Modify, remix, or incorporate portions of an existing program into one’s own work, to develop something new or add more advanced features.	Modularity	This is inherent to all lessons in which the child first follows the instructions to lay out the code cards as shown in the manual and then makes modifications to the code.

## INTRODUCTION

### Dear Parents, Teachers, and Other Supervising Adults,

This kit is designed to teach children the basic principles of programming in a fun, interactive, and experimental way. With your help, children can follow the lessons in this manual to learn how the robot works. They will see the effects of the different code cards on the robot's behavior. They can learn how to get the robot to do what they want it to do through a trial-and-error process.

Children will likely need help from a parent or adult while using this product. They may need your help interpreting terms that are new to them. They may need your help assembling difficult models or programs. And they may need your help explaining what the robot is doing and why.

Computer programming is complicated! Some of the lessons or descriptions in this manual may be too complicated for some children. In general, this manual is written with the intention of a parent, teacher, or adult reading it and then explaining it to the child, which may require interpretation or simplification to tailor the content to a specific child.

This product is intended to grow with your child. It is designed for children of 4 to 8 years in age. Starting out, younger children can play with the simplest programs. As they get older and understand more, they can use the more advanced features, like functions and conditional statements.

Robotics is a complex and technical field. Getting the robot to function exactly as expected is a process, often requiring many attempts. Each step of the way, your child is learning.

At the end of the manual, all the functions of the robot and cards are described. There is a glossary of terms on the back cover of this manual. If your child does not know the words "programming," "coding," or "robotics" yet, please review those definitions with them before starting.

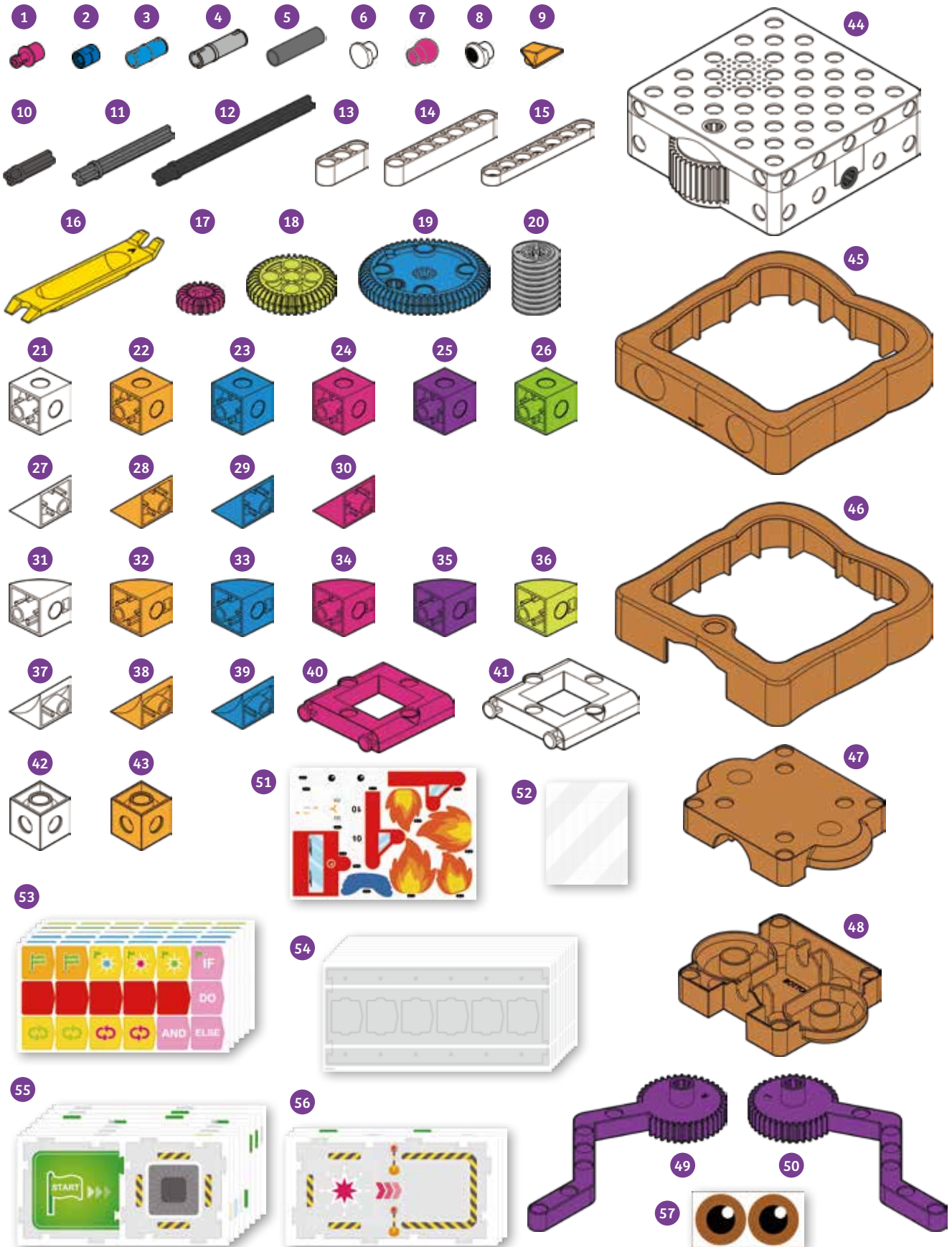
We hope you and your child have fun and learn a lot about coding and robotics while playing with this kit!

## TABLE OF CONTENTS

<b>Safety Information . . . .</b>	<b>Inside front cover</b>
<b>Introduction . . . . .</b>	<b>1</b>
<b>Kit Contents . . . . .</b>	<b>2</b>
<b>Getting Started . . . . .</b>	<b>4</b>
<b>Robotic Base Unit . . . . .</b>	<b>4</b>
<b>Code Cards . . . . .</b>	<b>5</b>
<b>Map Cards . . . . .</b>	<b>5</b>
<b>Basic Operation . . . . .</b>	<b>6</b>
<b>Lesson 1: Peanut Butter and Jelly Coding . . . . .</b>	<b>8</b>
<b>Chapter 1: Sammy and Foodville . . . . .</b>	<b>9</b>
<b>Lessons 2 – 8 . . . . .</b>	<b>12 – 17</b>
<b>Coding Concepts: Sequences, Loops, and Functions . . . . .</b>	<b>18</b>
<b>Chapter 2: The Adventures of Pippy the Mouse . . . . .</b>	<b>19</b>
<b>Lessons 9 – 12 . . . . .</b>	<b>21 – 24</b>
<b>Chapter 3: Arty's Party in the Park . . .</b>	<b>25</b>
<b>Lessons 13 – 16 . . . . .</b>	<b>28 – 31</b>
<b>Coding Concepts: Conditional Statements and Events . . . . .</b>	<b>32</b>
<b>Chapter 4: Robbie's Robotic Soccer Game . . . . .</b>	<b>33</b>
<b>Lessons 17 – 20 . . . . .</b>	<b>37 – 38</b>
<b>Chapter 5: Robotic Fire Truck . . . . .</b>	<b>39</b>
<b>Lessons 21 – 24 . . . . .</b>	<b>42 – 45</b>
<b>Chapter 6: Robotic Factory Floor . . . . .</b>	<b>46</b>
<b>Lessons 25 – 28 . . . . .</b>	<b>50 – 52</b>
<b>Chapter 7: Sammy's Ultimate Adventure . . . . .</b>	<b>53</b>
<b>Lessons 29 – 30 . . . . .</b>	<b>52 – 53</b>
<b>Math Lesson Mode . . . . .</b>	<b>56</b>
<b>Tech Specs . . . . .</b>	<b>59</b>
<b>Code Card Definitions . . . . .</b>	<b>59</b>
<b>Map Card Overview . . . . .</b>	<b>61</b>
<b>Combining Light Cards . . . . .</b>	<b>62</b>
<b>Code Graphics . . . . .</b>	<b>63</b>
<b>Background Music On/Off . . . . .</b>	<b>63</b>
<b>Math Programs . . . . .</b>	<b>63</b>
<b>Glossary . . . . .</b>	<b>Back cover</b>

# KIT CONTENTS

## What's inside your kit:



## KIT CONTENTS

## Checklist: Find, Inspect, Check off

✓	No.	Description	Qty.	Item No.
<input type="radio"/>	1	Shaft plug	8	7026-W10-H1K
<input type="radio"/>	2	Short anchor pin	30	7344-W10-C2B
<input type="radio"/>	3	Joint pin	4	7413-W10-T1B
<input type="radio"/>	4	Long joint pin	2	7413-W10-U1S
<input type="radio"/>	5	Tube, 30-mm	2	7400-W10-G1D
<input type="radio"/>	6	Button pin	10	7061-W10-W1W
<input type="radio"/>	7	Ball pin	1	7128-W10-E1K
<input type="radio"/>	8	Eye pin	8	7128-W22-2
<input type="radio"/>	9	Trapezoid pin	1	7128-W10-E4O1
<input type="radio"/>	10	Motor axle	2	7026-W10-L1D
<input type="radio"/>	11	Axle, 60-mm	1	7413-W10-M1D
<input type="radio"/>	12	Axle, 100-mm	1	7413-W10-L2D
<input type="radio"/>	13	3-hole wide rounded rod	2	7404-W10-C1W
<input type="radio"/>	14	7-hole wide rounded rod	2	7404-W10-C2W
<input type="radio"/>	15	7-hole flat rounded rod	2	7404-W10-C3W
<input type="radio"/>	16	Part separator tool	1	7061-W10-B1Y
<input type="radio"/>	17	Small gear	3	7026-W10-D2K
<input type="radio"/>	18	Medium gear	4	7408-W10-D2YG
<input type="radio"/>	19	Large gear	2	7026-W10-W5B
<input type="radio"/>	20	Worm (for worm drive)	1	7344-W10-A1S1
<input type="radio"/>	21	Cube block, white	16	880-W10-A1W
<input type="radio"/>	22	Cube block, orange	12	880-W10-A1O3
<input type="radio"/>	23	Cube block, blue	20	880-W10-A1B2
<input type="radio"/>	24	Cube block, pink	7	880-W10-A1K1
<input type="radio"/>	25	Cube block, purple	8	880-W10-A1P1
<input type="radio"/>	26	Cube block, green	7	880-W10-A1G1
<input type="radio"/>	27	Triangle block, white	4	880-W10-S1W
<input type="radio"/>	28	Triangle block, orange	4	880-W10-S1O3
<input type="radio"/>	29	Triangle block, blue	2	880-W10-S1B2

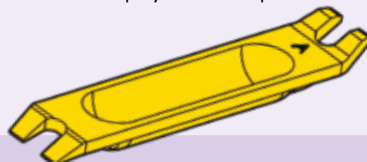
✓	No.	Description	Qty.	Item No.
<input type="radio"/>	30	Triangle block, pink	4	880-W10-S1K1
<input type="radio"/>	31	Convex block, white	18	880-W10-R1W
<input type="radio"/>	32	Convex block, orange	16	880-W10-R1O3
<input type="radio"/>	33	Convex block, blue	12	880-W10-R1B1
<input type="radio"/>	34	Convex block, pink	10	880-W10-R1K1
<input type="radio"/>	35	Convex block, purple	4	880-W10-R1P
<input type="radio"/>	36	Convex block, light green	12	880-W10-R1YG
<input type="radio"/>	37	Concave block, white	4	880-W10-D1W
<input type="radio"/>	38	Concave block, orange	6	880-W10-D1O3
<input type="radio"/>	39	Concave block, blue	3	880-W10-D1B2
<input type="radio"/>	40	Square frame, pink	2	7411-W10-F1K
<input type="radio"/>	41	Square frame, white	2	7411-W10-F1W
<input type="radio"/>	42	6-hole cube block, white	3	880-W10-N1W
<input type="radio"/>	43	6-hole cube block, orange	2	880-W10-N1O3
<input type="radio"/>	44	Robotic base unit	1	7442-W85-A
<input type="radio"/>	45	Sammy's crust, top	1	7442-W10-G1T1
<input type="radio"/>	46	Sammy's crust, bottom	1	7442-W10-G2T1
<input type="radio"/>	47	Sammy's gearbox, top	1	7442-W10-F1T1
<input type="radio"/>	48	Sammy's gearbox, bottom	1	7442-W10-F2T1
<input type="radio"/>	49	Sammy's arm, right	1	7442-W10-H2P
<input type="radio"/>	50	Sammy's arm, left	1	7442-W10-H1P
<input type="radio"/>	51	Die-cut graphics sheet	1	K16#7442
<input type="radio"/>	52	Map card straps (6)	1	K41#7442
<input type="radio"/>	53	Code cards (108)	1	K16#7442-3
<input type="radio"/>	54	Code card frames (10)	1	K16#7442-4
<input type="radio"/>	55	Map cards (16)	1	K16#7442-1
<input type="radio"/>	56	Base map cards (3)	1	K16#7442-2
<input type="radio"/>	57	Sammy's eye sticker sheet	1	R20#7442

## Batteries Not Included

You will also need 3 AA batteries (1.5-volt, type LR6). For classroom use or extended home use, **rechargeable** AA batteries (1.2-volt, type HR6/KR6) are recommended. While 3 batteries are in use inside the robot, you can keep 3 additional batteries charging in a charger, and then swap them out as needed.

## How to Take the Models Apart

If any of the building pieces are stuck together and too hard to separate with your bare hands, use the **part separator tool** like a crowbar to pry them apart.



## Missing Parts?

If you are missing any parts, please contact Thames & Kosmos customer service.

US: techsupport@  
thamesandkosmos.com  
UK: techsupport@  
thamesandkosmos.co.uk

## GETTING STARTED

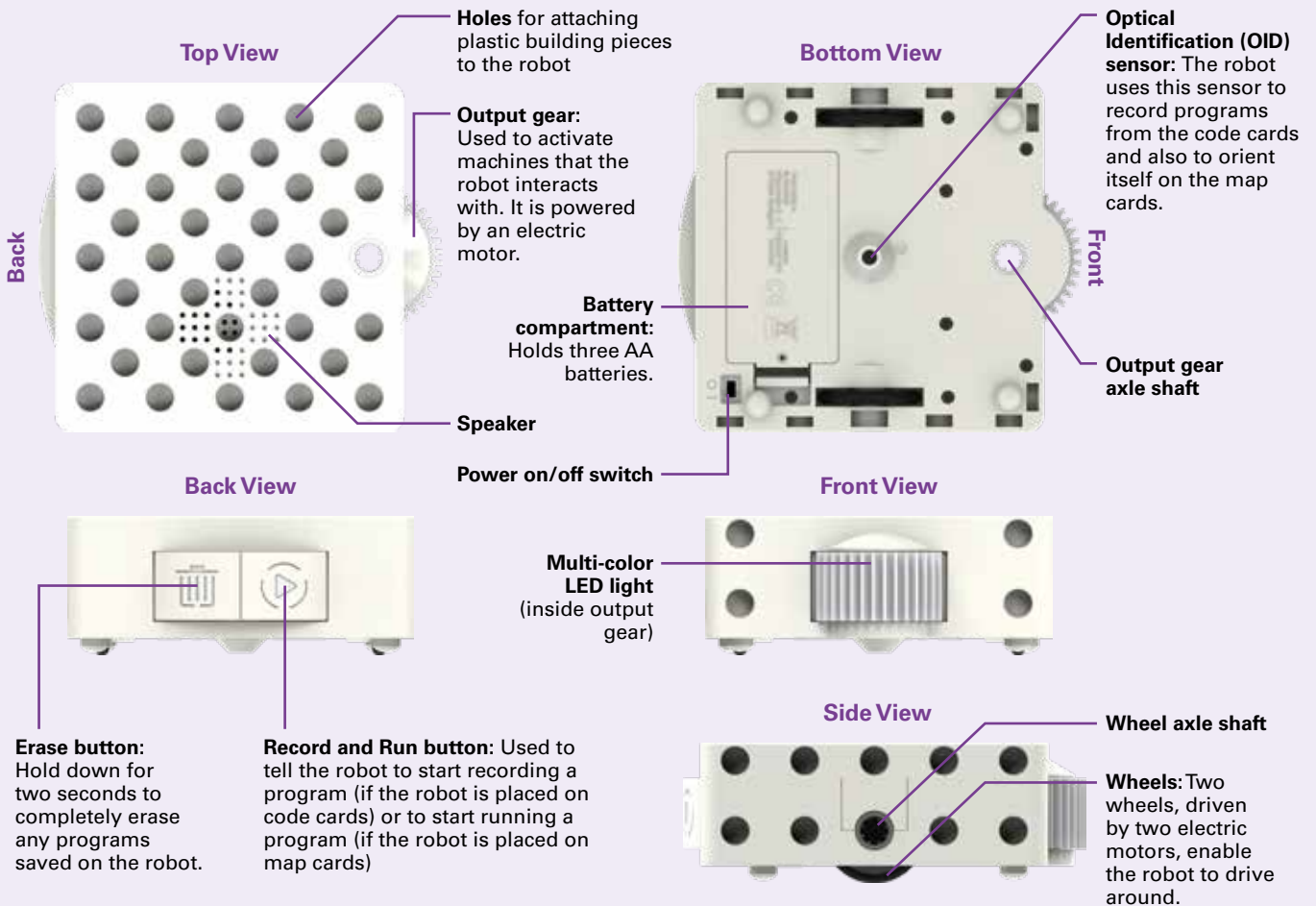
### Getting Started

Welcome to Kids First Coding & Robotics! First, let's take a look at the main parts of this kit: the **robotic base unit**, **code cards**, and **map cards**.

#### Robotic Base Unit

This is the base for all the robots you can build with this kit. Step-by-step instructions starting on page 9 show you how to assemble the plastic building pieces in the kit onto the robotic base unit, or into

other models that can be used alongside the robots you construct. The robotic base unit is packed with cool functionality! Here's an overview of all of its features:



#### Normal Mode vs. Math Mode

The robot has two modes. Out of the box, the robot is in **normal mode** by default. Lessons 1–30 use normal mode only. The robot has to be set to **math mode** to do the math lessons at the end of the manual. Learn how to use math mode starting on page 56.

#### Overview of Normal Mode Operation

In normal mode, when you slide the robot's power switch to the on position, the robot stands by for recording. You can then have the robot record a program. After the robot successfully records a program, the robot stands by to begin execution of the program. Place the robot on the Start map card, and the program will begin to run. When the program ends, the robot stands by to either run the same program again or record a new program.

## GETTING STARTED

### Code Cards

To program the robotic base unit, you don't need a computer or a tablet — all you need are the **code cards** and the **code card frames!** There are 61 different code cards. The kit includes multiple copies of some of the cards. There are 108 double-sided code cards, for a total of 216 sides.

You write a program by laying out a sequence of code cards in the frames. Then, the robot drives over the code cards one by one. While it does this, the **OID sensor** on the bottom of the robot scans a small pattern of dots that you can barely see printed on the cards. The robot's microprocessor is preprogrammed to translate this pattern into instructions it can follow.



Every program always starts with a **Start** code card.



Every program always ends with an **End** code card.



There are also cards that make the robot **move**.



There are cards that make the robot's **output gear turn**.



There are cards that tell the robot to make **sounds**.



There are cards that tell the robot to **light up** in a certain way or with a specific color.



There are **number** cards which **repeat** the card immediately before it a number of times.

There are **simple loop cards, function cards, conditional cards,** and **event cards.** You will learn about all of these later in the manual.



You will learn the rules and behaviors for the other code cards as you follow the instructions for the 30 lessons in this manual.

See page 59 for a **complete list of all the code cards** included in the kit and their functions.

### Map Cards

The robot always plays out (or **runs**) its programs on the **map cards.** The map cards also have OID patterns of dots printed all over them. The robot uses the OID sensor to read these patterns, which tell it which map card it is on and helps it orient itself and move in the correct directions on the map cards.

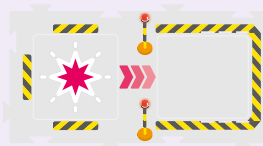
For every program you write, you always lay out a grid of map cards for the robot to run its program on.



The robot always starts its program on the **Start** map card.



Some map cards represent events that **trigger special functions.**



A few of the map cards are bigger than the others. These are called **base map cards.** You attach certain models to these cards using the **map card straps** so that the robot can interact with the models.

There are a total of 38 different map cards in the kit, including two Start cards, six base map cards, and four event map cards. The map cards are **double sided**, so there are actually just 19 separate cards, each with one map card on the front and one on the back.

The map cards have **interlocking tabs** like a jigsaw puzzle to keep them together. Please note that you either have to use the front sides or the back sides of the map cards at one time, because the tabs will only match up correctly if all of the cards are flipped to their compatible sides.

See page 61 for a **complete list of all the map cards** included in the kit and their functions.



**A close-up of a code card showing the OID pattern of dots**



## BASIC OPERATION

### Basic Operation

#### Turning the Robot On and Off

1. Make sure that a parent or adult has installed the **batteries** as described on the inside front cover of this manual.
2. Slide the **power switch** located on the bottom of the robot base unit to the on position.
3. The robot will light up and play its **startup** sounds.
4. The robot is now **standing by for recording**. It will pulse with a blue light.

When you are not using the robot, turn the power off by sliding it into the off position to **save battery energy**. Programs are erased when you turn the robot off.

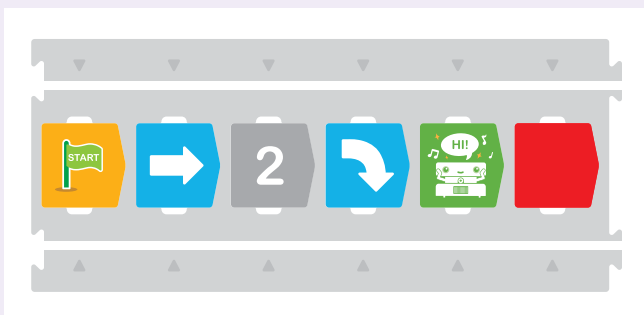
If you don't use the robot for five minutes, it will automatically go into a **sleep mode**. Programs are preserved while the robot is sleeping. You can press either button to **wake up** the base unit.

When the batteries are running low, the robot will alert you with a flashing orange light and play a **low-power indicator** sound.

#### Recording a Program

You **program** the robot by **laying out a series of code cards** for the robot to drive over and record. Here's how it works:

1. Make sure the robot is **turned on** and standing by for recording.
2. **Lay out a series of code cards** in the code card frame(s). A main program can have up to 30 code cards, not including the Start and End code cards.



Subroutine programs, or **functions**, are introduced in Lesson 11. Functions can have up to 15 code cards.

If your table is too short to place all of the frames in a row, no problem! **You can record any program in segments**. The robot won't stop recording until it scans the End code card. Therefore, you can scan one row of code, and the robot will pause at the end. Then you can move the robot to another row, and the robot will automatically continue recording.

If your robot scans the maximum number of cards but did not scan an End card, the robot will automatically end the program.

3. **Place the robot directly above the Start code card** at the start of the frame, facing toward the rest of the code.
4. **Press the Record button**.
5. The robot will pulse with red light, its Record button will pulse green, and it will play music indicating it is recording. At the same time, the robot will drive forward over the code cards, one at a time, **scanning and recording** the program.
6. For each successfully recorded code card, the robot will play a sound.
7. If the robot encounters any **problem** while recording, it will **flash orange and red** and play an **error sound**. This could happen if the robot is facing the wrong direction or if the code cards were laid out in an incorrect order.
8. When the robot reaches the End card and scans it, the robot will stop moving and play a **finished-recording** sound.
9. The robot will now be **standing by to run** its program. Its Record button light will now be solid green.
10. If there is a subroutine function to program, place the robot on the Function Start card and press the Record button. **The robot remembers one main program and up to three functions at one time in its memory.**

#### Running a Program

Once a program has been recorded, you can **run the program**. Here's how:

1. **Place the robot on the Start map card**, facing the direction of the arrows.

## BASIC OPERATION

2. **Press the Run button.** Record and Run are the same button. The robot knows whether to record or run based on whether it is sitting on a code card or a map card.
3. The robot will now **run the program.** The robot will first move around a little on the Start map card to **orient itself.** This is important so that it stays aligned to the map cards throughout the program. If at this time there is no main program recorded on the robot, it will flash between red and orange and play a warning sound. As the robot runs the program, it will play its running background music, unless the program tells it to play other music.
4. When the robot reaches and scans certain map cards, such as Event or Base map cards, it may trigger **special behaviors or functions.**

After running a program, the robot still remembers the programs; the **programs are not automatically erased** after running. You can **run the program again**, or record or overwrite the program or function.

### Overwriting Programs

The robot can only hold one main program and one of each subroutine functions at a time. If you have the robot record a new program or function (starting with the Start card or one of the Function Start cards) when there is already a program or function saved, the robot will **overwrite the old main program or function.** This means the old program is erased and the new one replaces it.

If you want to revise the main program or a function, you can overwrite them one at a time; the other programs are saved.

### Erasing Programs

To **completely erase** all of the programs on the robot (and quit Math Mode), **press and hold the Erase button for two seconds or longer.** The robot's light will flash red for a few seconds and then stop, indicating that the program memory has been cleared.

## Lessons

The best way to learn what all the code cards do and how they work together is by following the **lessons** in this manual.

For each lesson, you **first build some models.**

The step-by-step assembly instructions are printed before the lessons in which they are used. Then, you **lay out the grid of map cards** exactly as shown in the lesson, and also the **series of code cards.** Then **record and run** the program and observe what the robot does. Did it all work perfectly? Congratulations! If not, you should go through a debugging process to fix the physical model, the code cards, and/or the map cards until it works perfectly!

## Debugging

**Debugging** is the process of finding and preventing unwanted behavior in a program, computer, or robot.

When and where did the robot start to do something other than what you wanted it to do?

Could the code cards have caused this? If so, try checking the code cards against the instructions and making changes.

Could the map cards have caused this? If so, try checking the map cards against the instructions and making changes.

Could the model construction have caused this? If so, try checking the model against the instructions and making changes.

You can look at the technical information in the back of this manual to find more **troubleshooting tips.**

## Background Music

If you want to **turn the background music off or on**, scan the Background Music code graphic on page 63. The background music is on by default. The graphic looks like this:



## Need Help?

Contact Thames & Kosmos  
Technical Support!

### United States

Email: [techsupport@thamesandkosmos.com](mailto:techsupport@thamesandkosmos.com)  
Web: [thamesandkosmos.com](http://thamesandkosmos.com)

### United Kingdom

Email: [techsupport@thamesandkosmos.co.uk](mailto:techsupport@thamesandkosmos.co.uk)  
Web: [thamesandkosmos.co.uk](http://thamesandkosmos.co.uk)

Why is Sammy shaped like a **peanut butter and jelly sandwich**? Sammy was inspired by a classic activity that is used to introduce students to computer science. In this activity, students are asked to write a program, or a series of instructions, for another student or the instructor to follow to make a peanut butter and jelly sandwich.

This activity teaches many concepts from **computer science**. Students learn to write precise and thorough instructions. They learn how computers do only what they are programmed to do. And they learn about the process of debugging, or the repetitive process of finding errors in a program, correcting them, and then retesting the program.

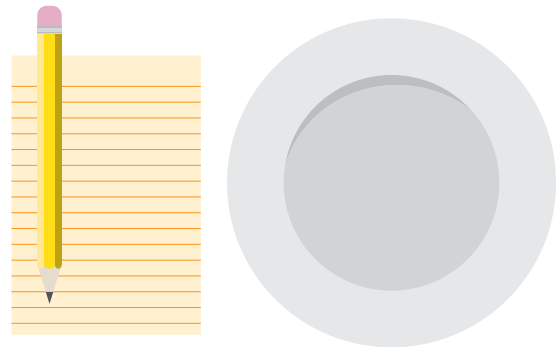
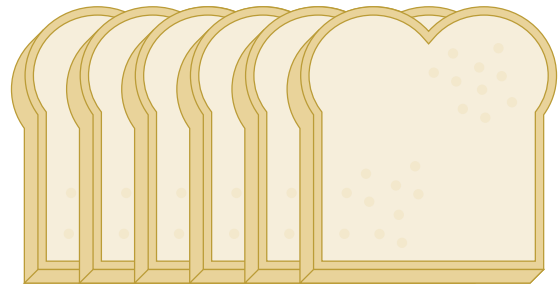
You can do a simple version of this activity here. Obviously, don't attempt this if you or your child have any allergies or dietary restrictions that would cause any issues with it.

### You will need:

*Package of sliced bread, jar of peanut butter, jar of jelly, dull knife, plate, paper, pen or pencil*

### Here's how:

1. Set out and review the materials needed to make a peanut butter and jelly sandwich with your child.
2. Ask your child to tell you the steps to making a peanut butter and jelly sandwich. Write down each step as your child dictates them to you.
3. When you are done writing the instructions, start following the instructions, one step at a time.
4. Take the instructions as literally as possible. For example, if the instruction is to put the peanut butter on the bread, you might take the jar of peanut butter and put it on the package of bread. If the instruction is to pick up the knife, you could pick it up by the blade instead of the handle. If an instruction is impossible to perform, you could freeze up and state that an error has occurred.
5. Go through the instructions step by step and try to debug them with your child. In the end, you should have a peanut butter and jelly sandwich that you can eat together as a snack.

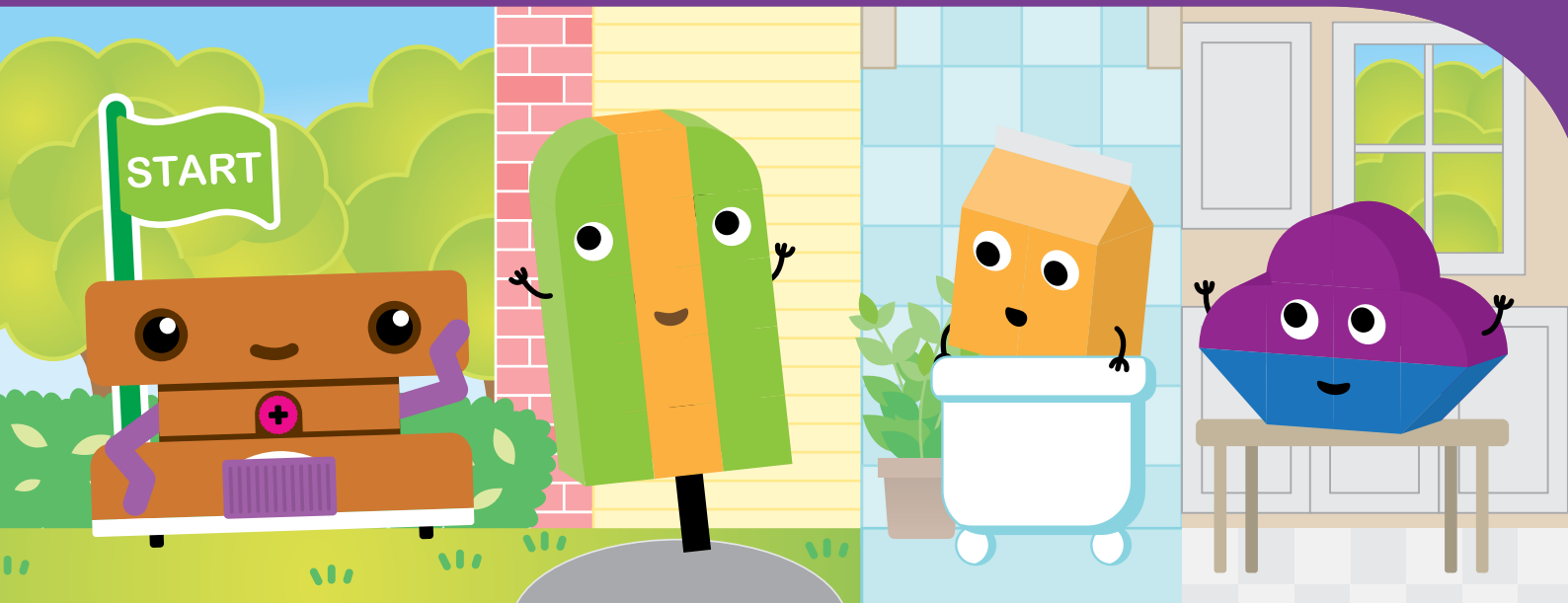


This activity is a simple model for demonstrating how a computer program works to control a robot. The list of instructions is the computer program and you played the role of the robot.

As you probably observed, a program's instructions must be performed in a **sequence**, or a specific order. They must be written in a language, or **code**, that the robot's computer can understand. They must describe everything that the programmer wants the robot to do. The robot will not do anything other than what is programmed.

In the process of programming and testing programs, it is usually the case that the program behaves in an unexpected or unintentional way. This is a **bug**. Bugs can be fixed by locating which segment of code is causing it, changing it, and testing it until the desired behavior is achieved.

Now, let's start using the robot!



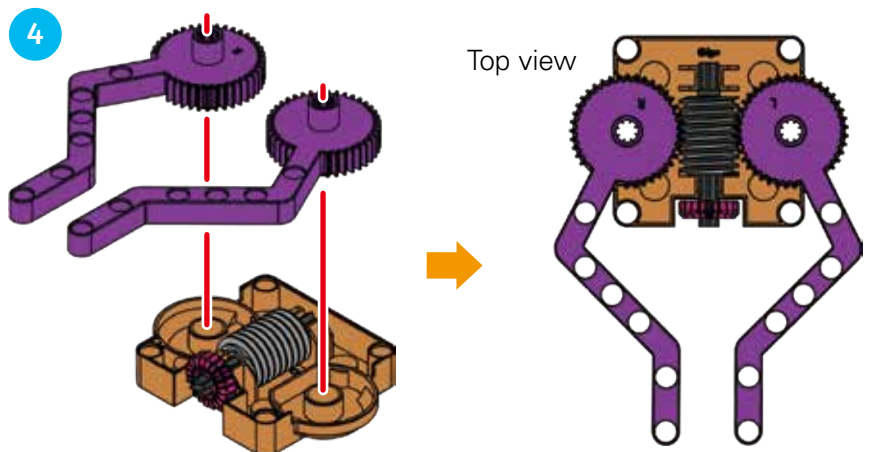
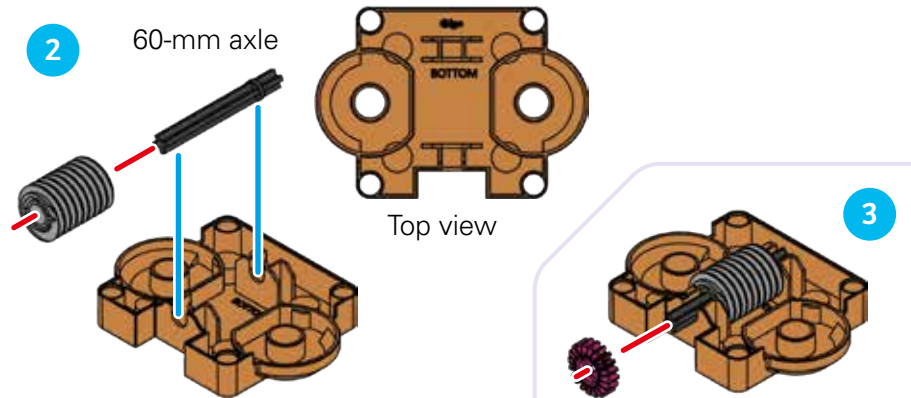
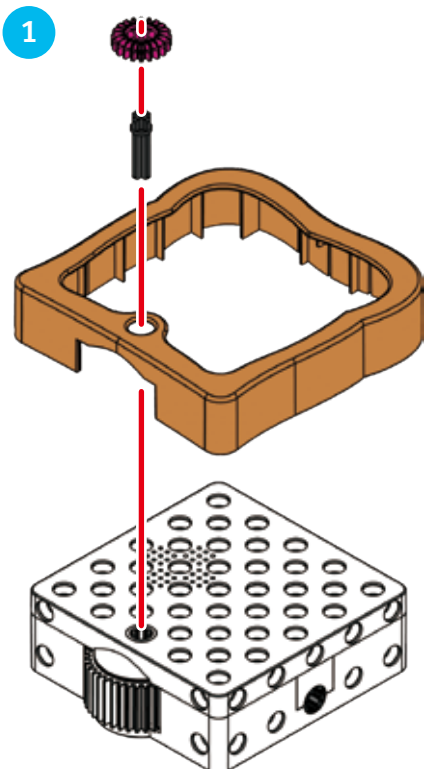
## Chapter 1: Sammy and Foodville

Sammy is a robot. Robots come in all shapes and sizes. Sammy happens to be shaped like a peanut butter and jelly sandwich! Sammy has wheels powered by an electric motor inside its robotic base unit. Sammy also has purple arms connected to gears that are connected to an electric motor.

In this chapter, you will build and program Sammy to visit other food-friends that live in Foodville. First, follow the instructions below to assemble Sammy. Then, build Sammy's food-friend Hammy the Hamburger. Then, follow the instructions for the lessons starting with Lesson 1 to start coding!

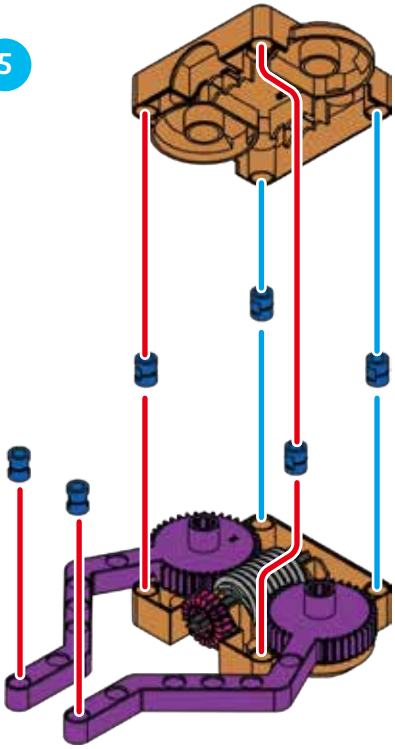
### BUILD

### SAMMY

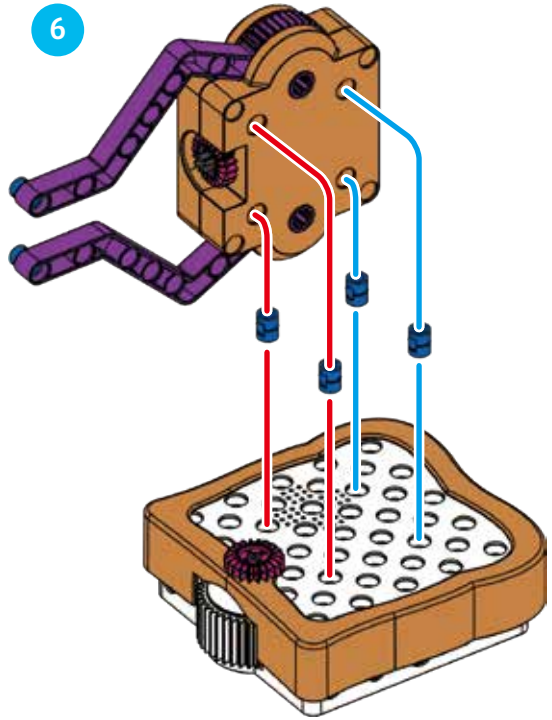


CONTINUED ...

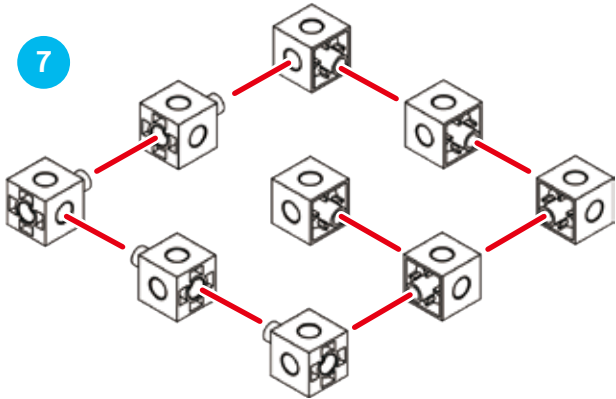
5



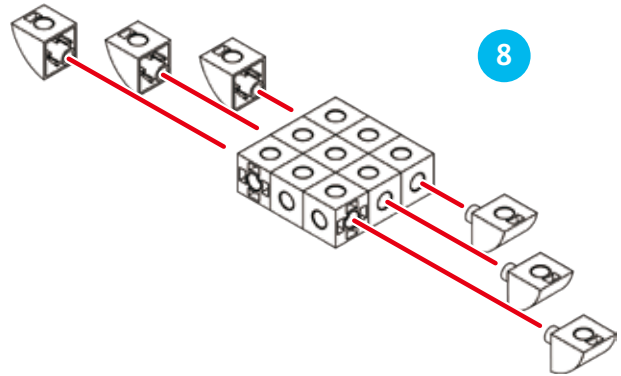
6



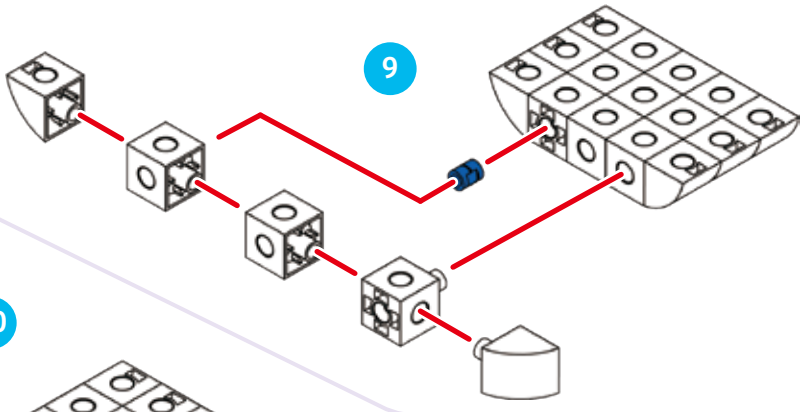
7



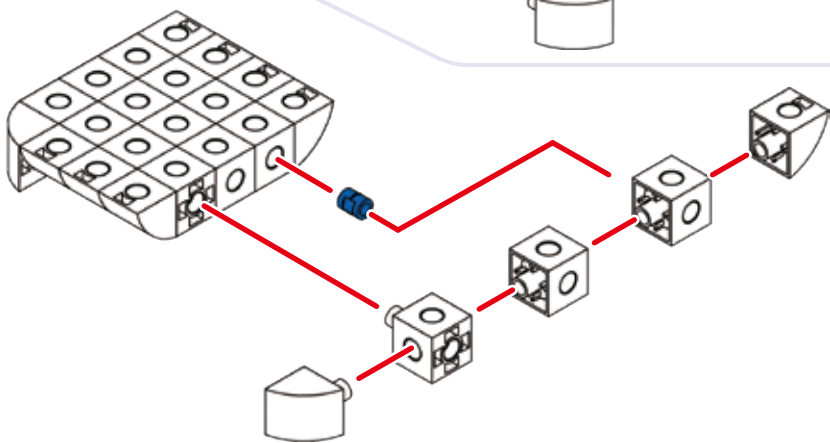
8



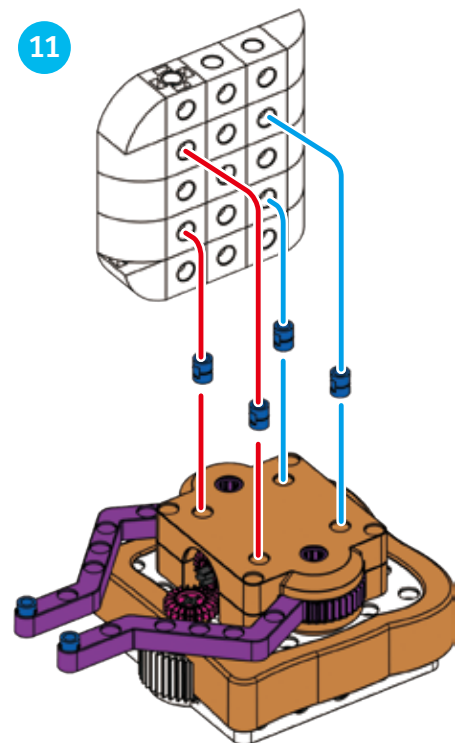
9



10

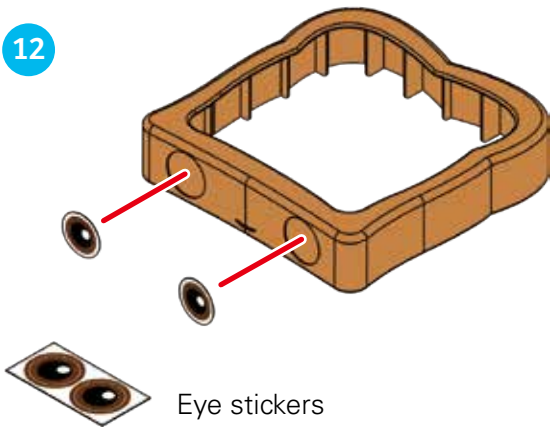


11

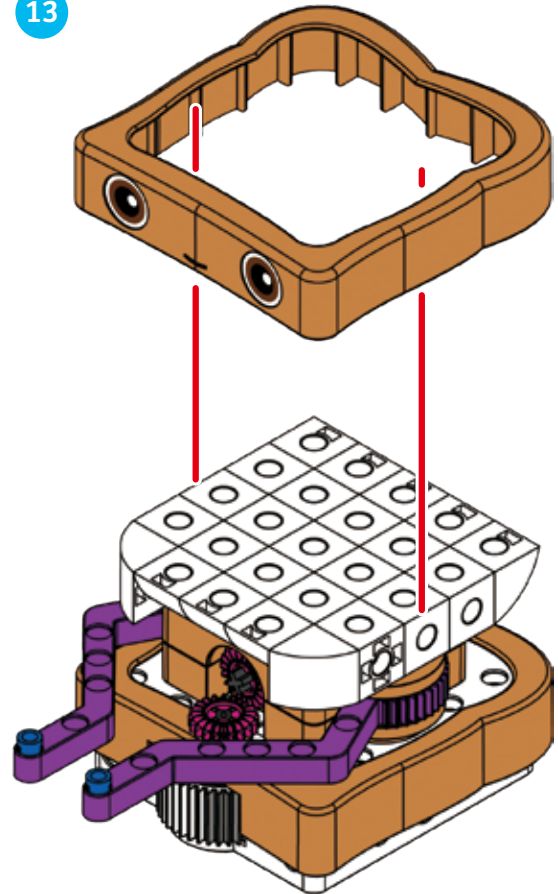


CONTINUED ...

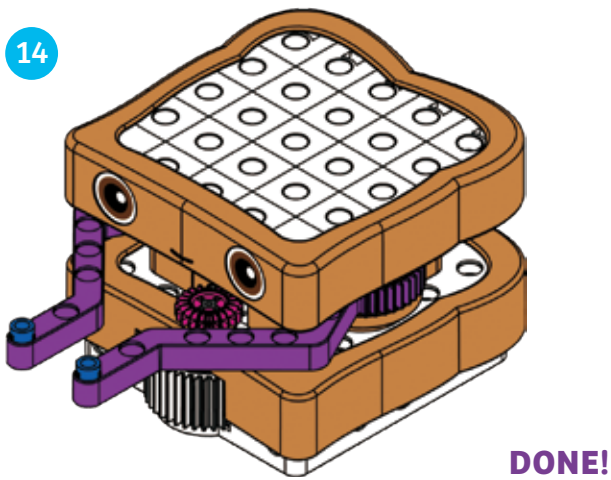
12



13



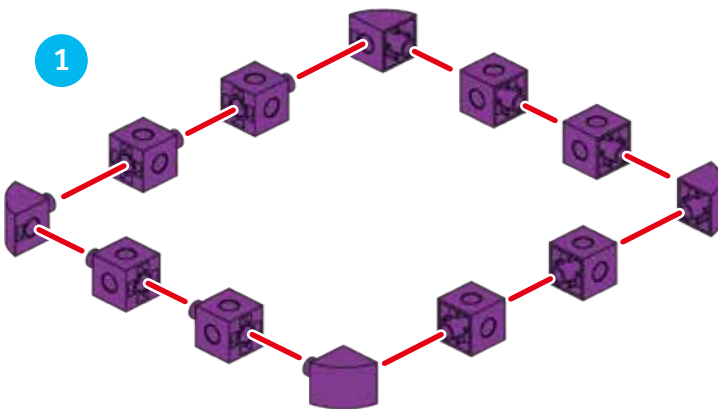
14



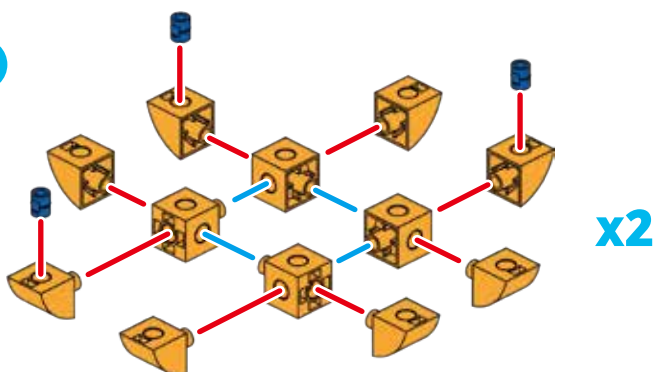
**BUILD**

**HAMMY**

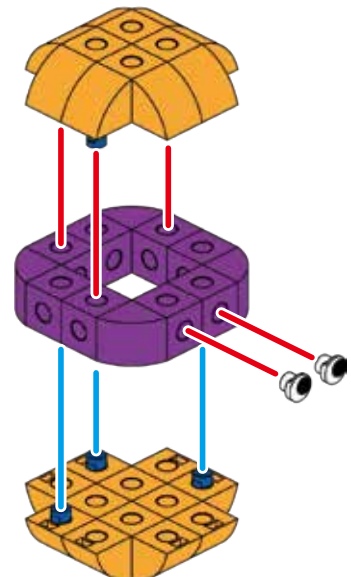
1



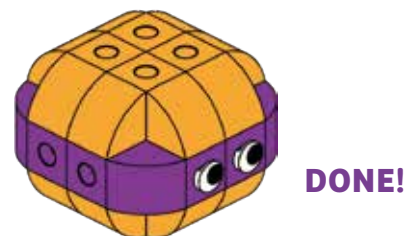
2



3



4



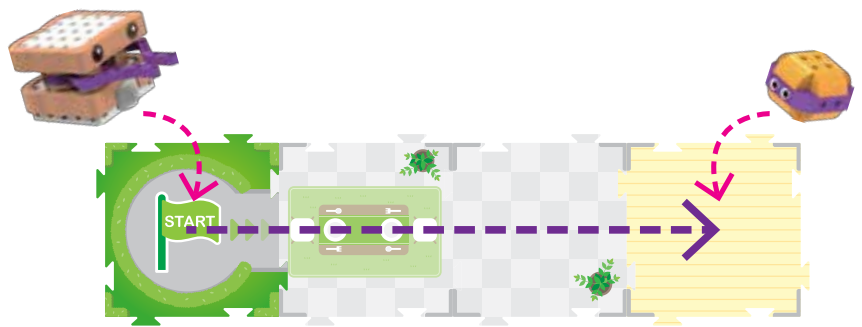
**STORY**

Sammy is going to visit Hammy in Hammy's house. Sammy has to pass through two rooms to get to the room Hammy is in. Can you program Sammy to do this?

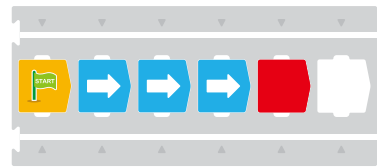
**HERE'S HOW**

Before starting, make sure you have read the introductory instructions for using the robotic base unit on pages 4 through 7.

1. Set up the map cards as shown. Place Hammy on the map card as shown.
2. Put the code cards into the code card frame in the order shown.
3. Turn Sammy on with the switch on the bottom.
4. Place Sammy on the Start code card. (You can align the axle hole of the robot's wheels with the dark gray arrows on the code card frame.) Press the Record button. Wait for Sammy to finish recording the program.
5. Place Sammy on the Start map card. Press the Run button (which

**MAP**

**CODE**


Put the code cards into the code card frame in this order.



is the same as the Record button).  
6. Watch Sammy drive through the house and get to Hammy. Did everything work as you expected?

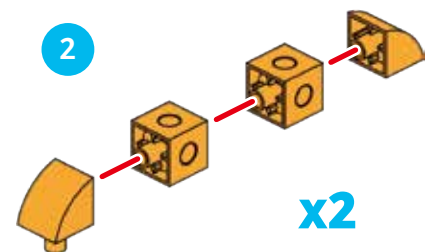
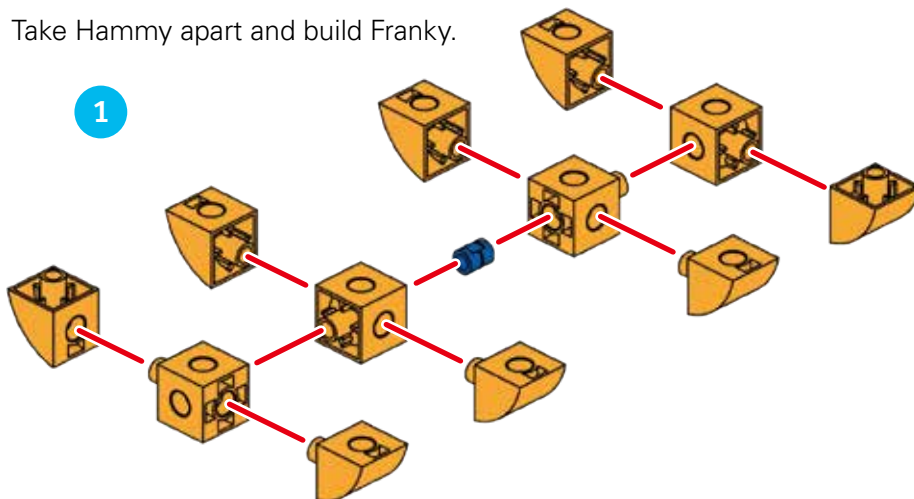
**WHAT'S HAPPENING**

The robot scans a Start code card, then three Move Forward code

cards, then an End code card. This results in a simple program that moves Sammy forward three map cards. Note how the robot always moves around a little to orient itself on the Start map card before running the program.

**BUILD**
**FRANKY**

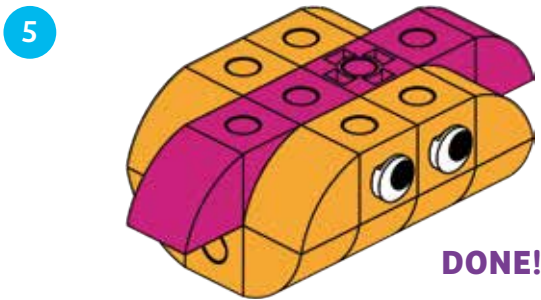
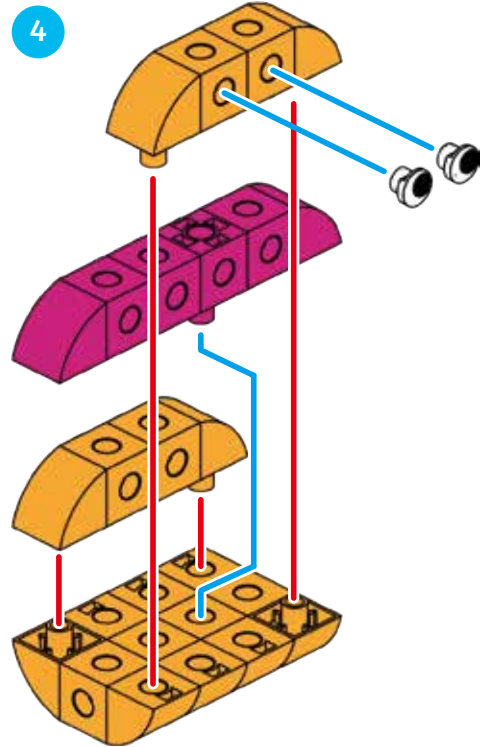
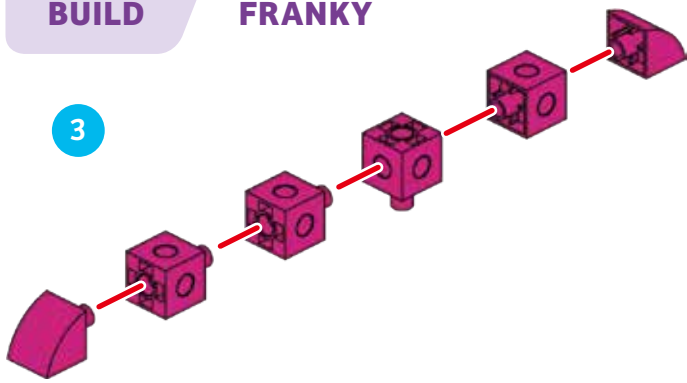
Take Hammy apart and build Franky.



**CONTINUED ...**

**BUILD**

**FRANKY**



**DONE!**

**LESSON 3**

**FRANKY'S WAKE-UP CALL**

**STORY**

Now Sammy is going to wake up Franky, who has overslept. Being a hot dog, naturally Franky's house is longer than Hammy's. Program Sammy to drive into the house to get to Franky and then drive back outside again.

**HERE'S HOW**

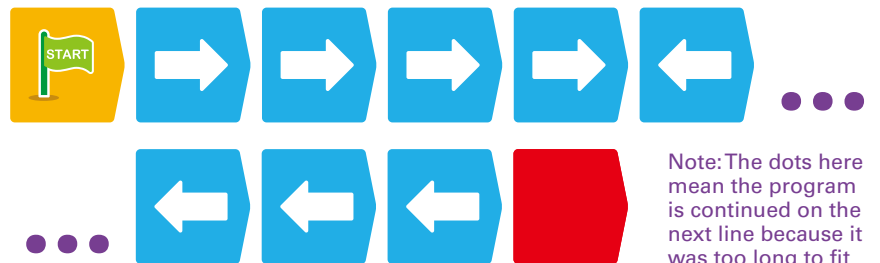
Set up the map cards, models, and code cards as shown. You will need two code card frames for this program. Record the program and then run the program. If anything didn't work properly, try it again or try debugging the cards.

**Note: The Here's How sections will no longer be repeated in the following lessons, because it is always the same basic process. Only special instructions will be called out.**

**MAP**



**CODE**



Note: The dots here mean the program is continued on the next line because it was too long to fit on one line.

**WHAT'S HAPPENING**

This program uses a sequence of four Move Forward cards and four Move Backward cards. This results in a program that moves Sammy

forward four map cards and then backward four map cards.



**A LONGER WAKE UP**

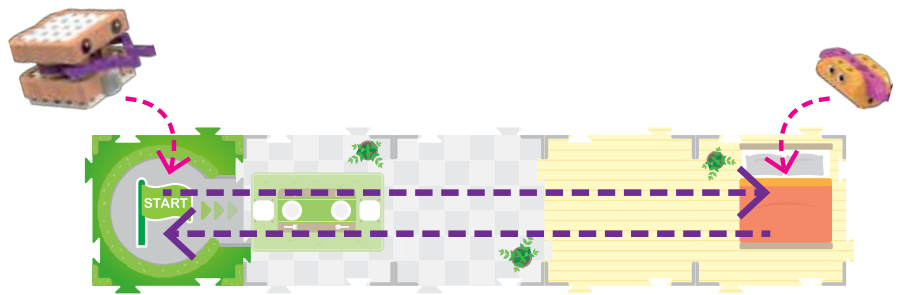
**STORY**

Franky didn't wake up the first time Sammy visited. Sammy didn't wait long enough to make sure Franky woke up. This time, program Sammy to pause for a second in Franky's room.

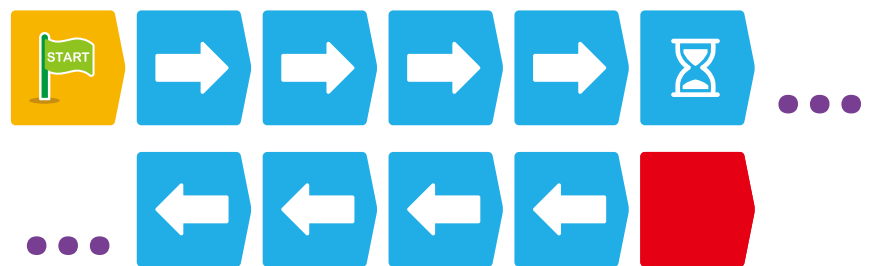
**WHAT'S HAPPENING**

This time, you have inserted a Pause Movement card in between the Move Forward and Move Backward cards. This stops Sammy's movement for one second when Sammy is in Franky's room.

**MAP**



**CODE**



**A LONGER PAUSE WITH SHORTER CODE**

**STORY**

Oh boy. Franky still didn't wake up! And laying out all these code cards is taking a long time. Is there a way to make Sammy pause for a longer time and also use fewer cards to do the same thing?

**WHAT'S HAPPENING**

The number cards execute the code card immediately before them by the specified number of times. The first Number 4 card executes the Move Forward action four times. The Number 2 card executes the Pause Movement card two times, pausing the robot for two seconds instead of one. The second Number 4 card executes the Move Backward card four times.

**MAP**



**CODE**



● ○ ○ **LESSON 6**

**TURNING A CORNER**

**STORY**

Waking up Franky has made Sammy exhausted! Sammy wants to go home to bed, which is around a corner in its house. What's the shortest program you can write to get Sammy there?

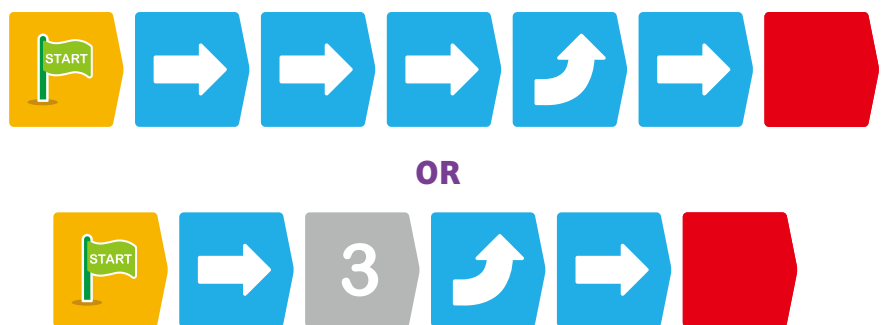
**WHAT'S HAPPENING**

In this program, you are using the Turn Left code card for the first time. First, Sammy moves forward three map cards. Then, the Turn Left code card rotates Sammy 90 degrees (a quarter of a full circle) so it is facing the bedroom. Finally, the last Move Forward card moves Sammy into the bedroom. A shorter version of the program using a number card is also shown.

**MAP**



**CODE**



● ○ ○ **LESSON 7**

**TOURING A NEW HOME**

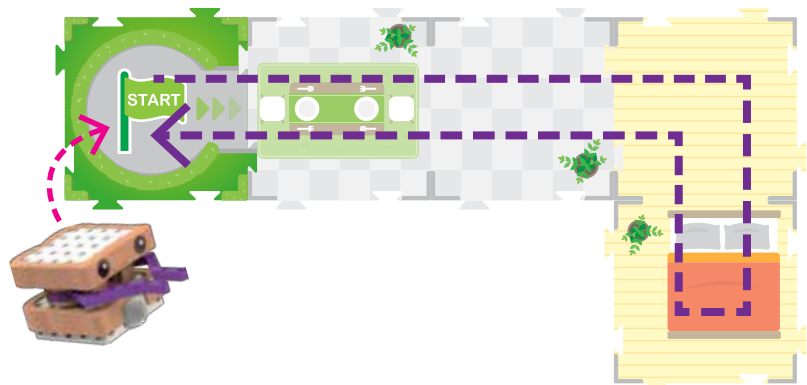
**STORY**

Sammy wants to take a tour of another house, which has a different layout than Sammy's own home. Can you write a program to move Sammy through the entire house and then back to the Start map card again?

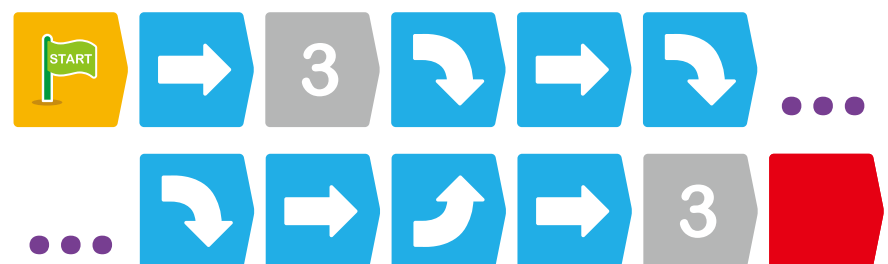
**WHAT'S HAPPENING**

In this program, you are using both Turn Left and Turn Right code cards. First, Sammy moves three map cards forward. Then, the robot turns right and moves forward one more map card into the bedroom. Then, it turns right two more times, moves out of the bedroom, turns left, and moves all the way out of the house.

**MAP**



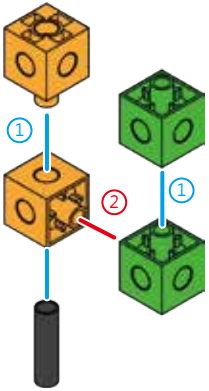
**CODE**



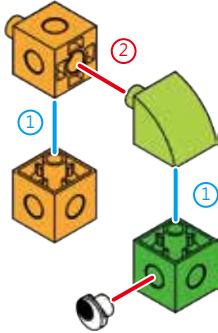
**BUILD**

**POPSY**

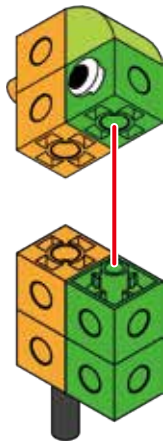
1



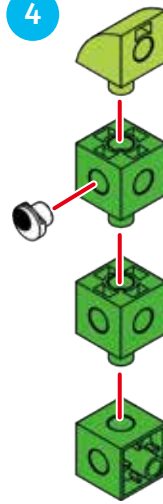
2



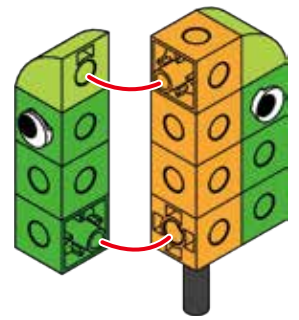
3



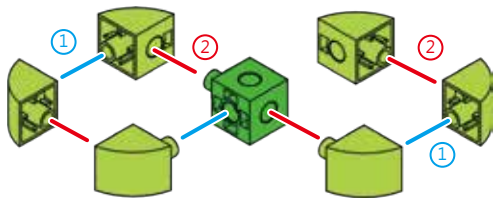
4



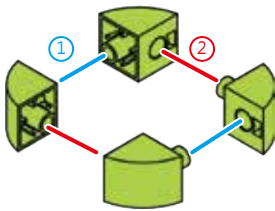
5



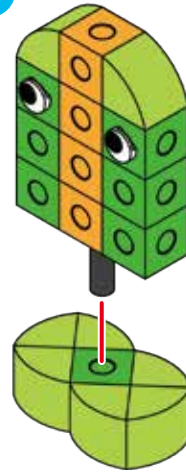
6



7



8



9

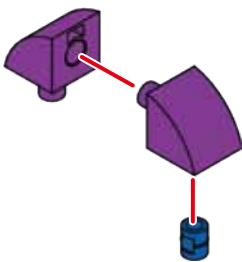
**DONE!**



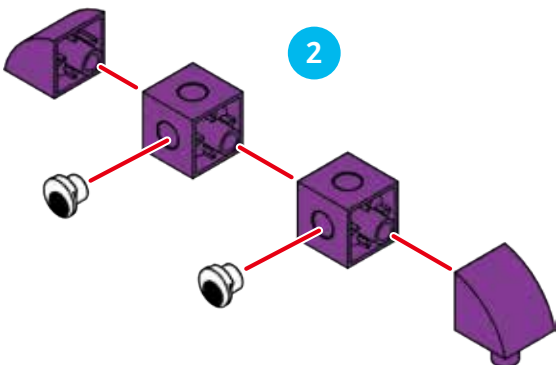
**BUILD**

**PUDDINGTON**

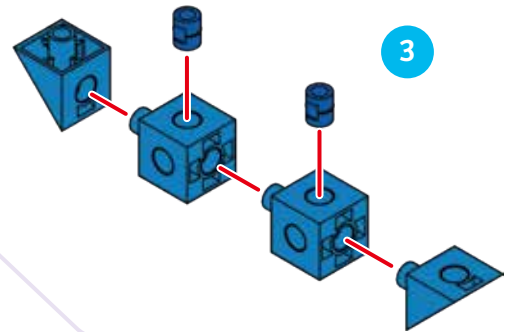
1



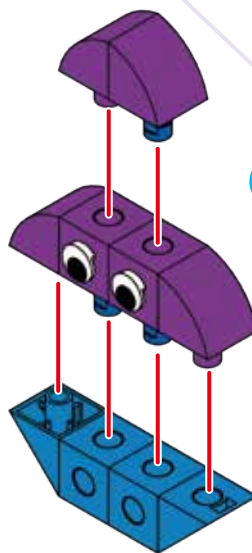
2



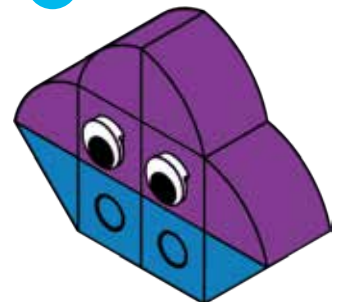
3



4



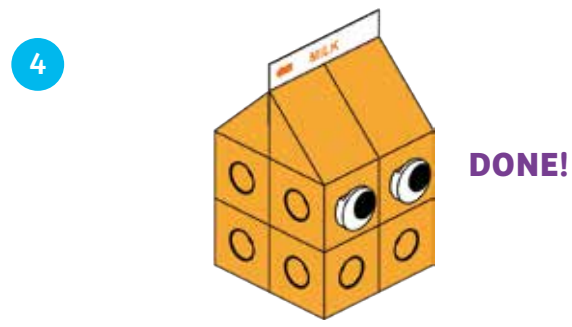
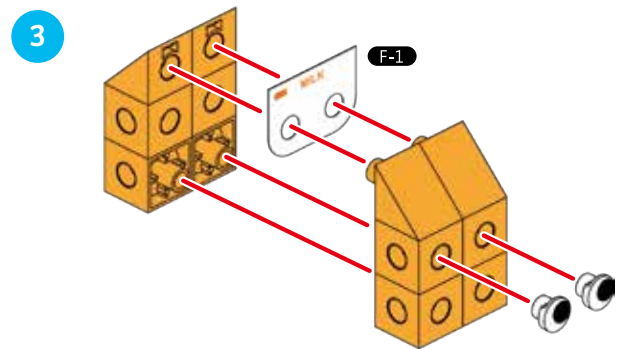
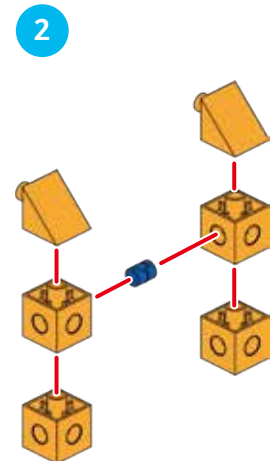
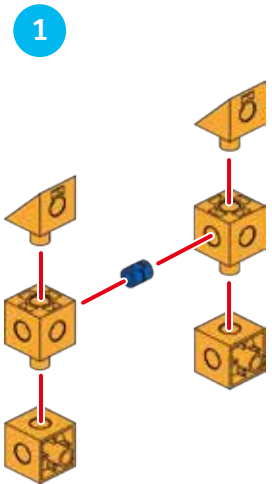
5



**DONE!**

**BUILD**

**LIL' MILK**



● ● ● **LESSON 8**

**SAMMY'S FRIEND-FILLED DAY**

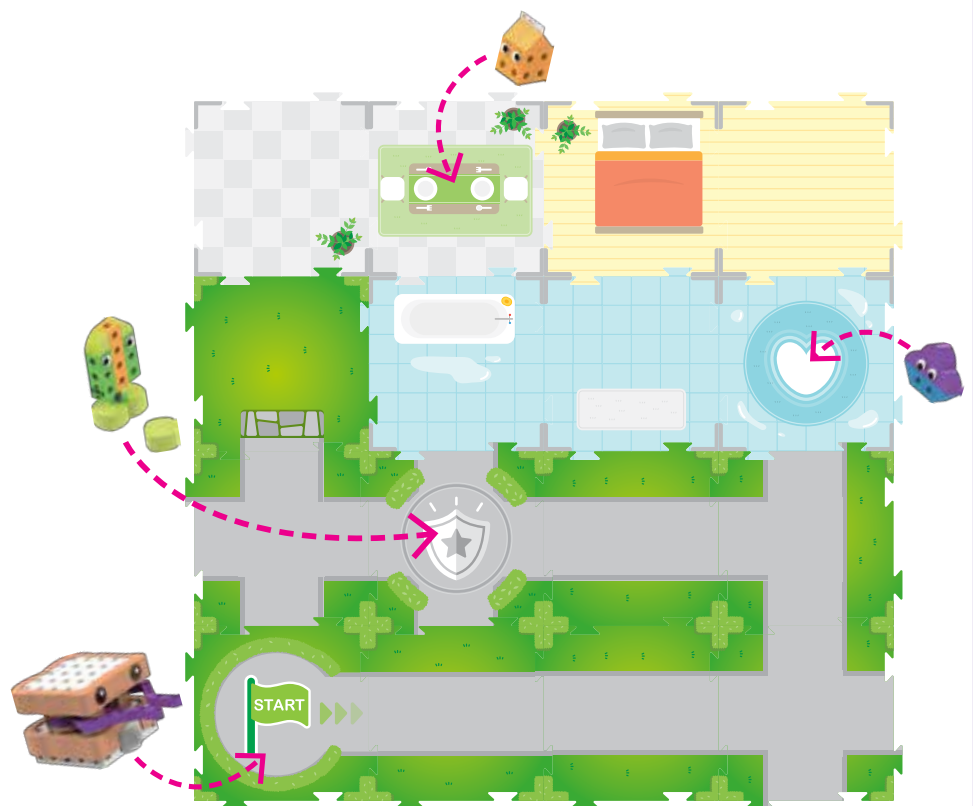
**STORY**

Sammy wants to visit three food-friends that are currently located at different spots around town. Place Puddington, Popsy, and Lil' Milk on the map as shown.

**CODING CHALLENGE**

Can you write a program to make Sammy move around the map to visit all three of the food-friends? The specific program for completing this lesson is not printed in this manual, so you have to figure it out for yourself. There are many possible solutions. First, plan out what you want the robot to do and where you want it to go. Then, write a code to make the robot follow your plan. Try to use the fewest number of code cards as possible to achieve your goal of bringing Sammy to the three food-friends.

**MAP**



## Sequences

In Lessons 1–8, you learned about and experimented with sequences. A sequence is one of the most important structures (or arrangements of elements) in computer programming. A **sequence** is a set of steps or commands arranged in a specific order. Computers run through the steps of a sequence in order, executing one at a time, for the purpose of performing a specific task that the sequence was created to perform.

When you wrote the steps to make a peanut butter and jelly sandwich, you created a sequence. When you arranged the code cards in order to tell Sammy the robot what to do and when to do it, you created a sequence.

All computer programs are sequences. A computer or robot will likely not perform the task intended by a programmer if the steps of the program are not in the correct order. All the steps of a program must be in a language that the computer can understand.

## Loops

In the next chapter, we will introduce the concept of the loop. A loop is another one of the most important structures in computer programming. A **loop** is a set of steps that repeats a number of times. Loops can be programmed to repeat a set number of times, forever, only while something else is happening, or until another thing happens.

Loops are important because programmers often need to have certain tasks or operations repeated a number of times. If the programmer had to write the same code over and over again, the code would be long and messy. In coding, programmers are usually trying to create the shortest, cleanest, simplest code that still works properly.

In the coding language in this kit, there is something called a **simple loop**.

### How to Use Simple Loops

A simple loop runs the sequence of code cards laid out inside the loop a set number of times. There are **two sets of simple loop cards** in this kit: **green and red**. This means you can use up to two loops in the same program.

To set up a loop, you must always use **two loop cards** of the same color (either two green loop cards or two red loop cards). One loop card is placed at the **start** of the loop and the other is placed at the **end** of the loop. A **number card** must be placed immediately after the first loop card. This number card indicates how many times the other code cards placed after it but before the second loop card will be **executed (run)**.



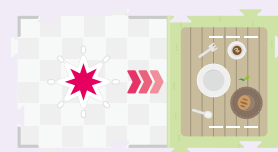
You cannot place more than one number card after the first loop card. You cannot place a number card after the second loop card. Both of these placements will result in an error. You can nest one loop inside another.

You can experiment with simple loops in Lesson 9.

## Functions

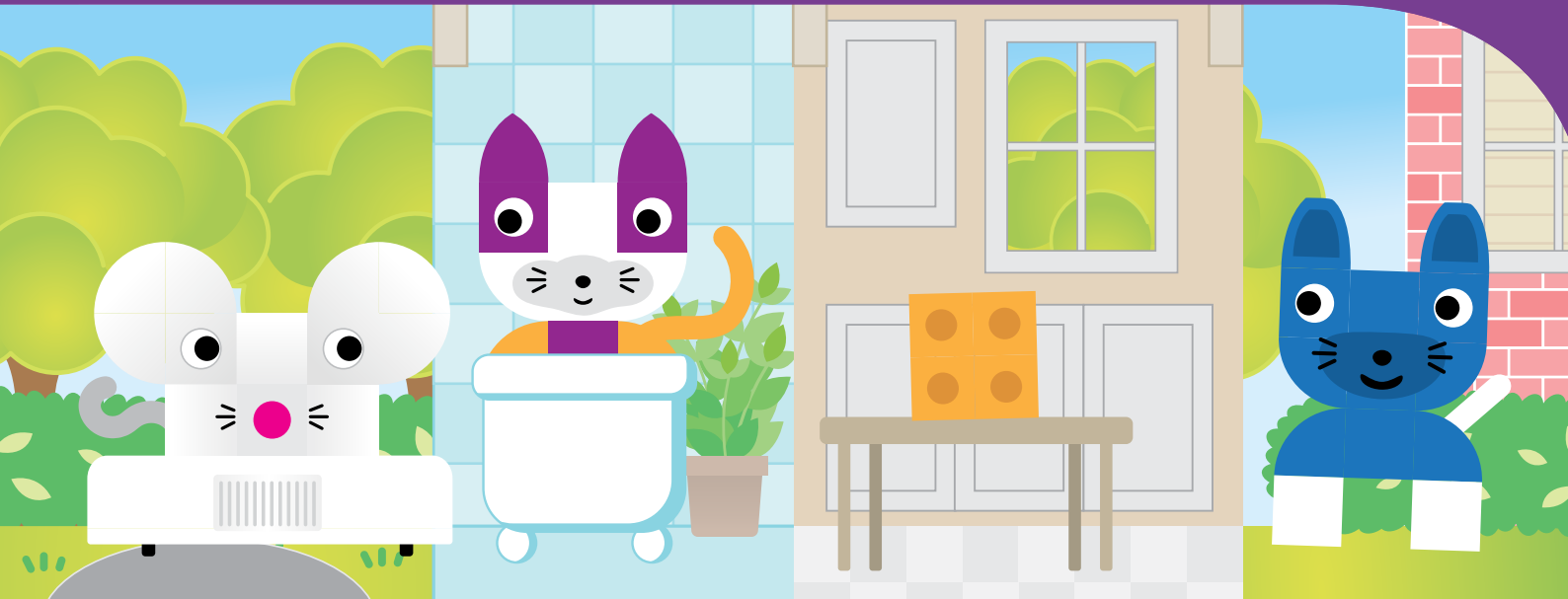
Another critical programming structure is the function. Functions will first be used in Lesson 11. A **function** is a set of steps that can be used again and again in a larger program. A function is written once and given a name or label. Then the function can be called upon in the program whenever it is needed, eliminating the need for the code of the function to be written more than once. This helps keep the code short and clean. Many functions are built into programming languages and computers already. Functions are also sometimes called subroutines.

In the coding language in this kit, functions are demonstrated with the **red, green, and blue function cards**. These functions are always used with the **base map cards**. For example, the red function is performed when the robot scans the red function base map card.



**The red function card is always used with the red base map card.**

You can learn how to use these functions in Lesson 11.



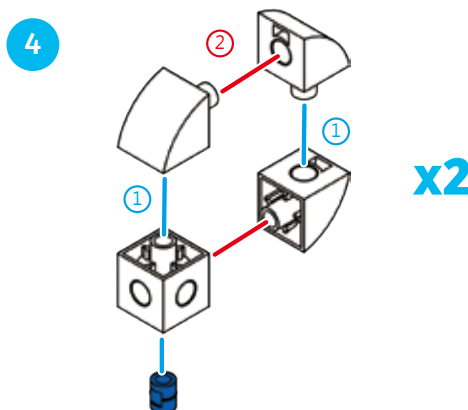
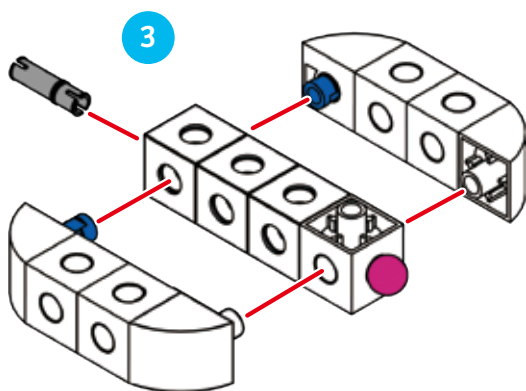
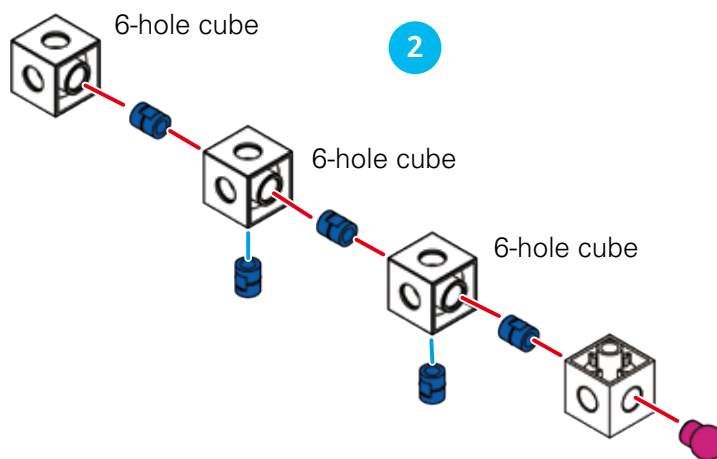
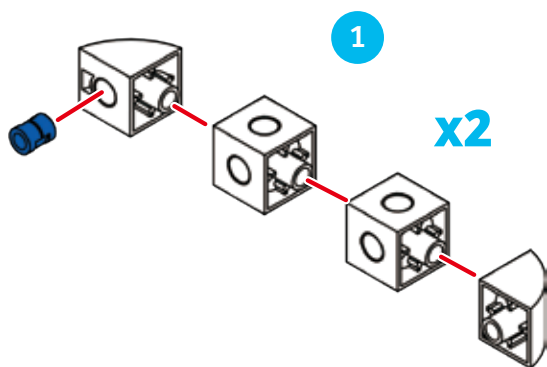
## Chapter 2: The Adventures of Pippy the Mouse

Pippy is a mouse who loves cheese. She is always trying to find cheese that people have left out. But she has to be careful, because Purry the Cat and Barker the Dog might be hiding in the bathtub or patrolling the neighborhood streets! Pippy will have to scurry past them and not get caught.

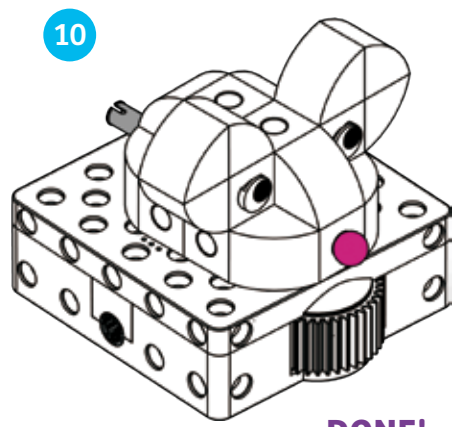
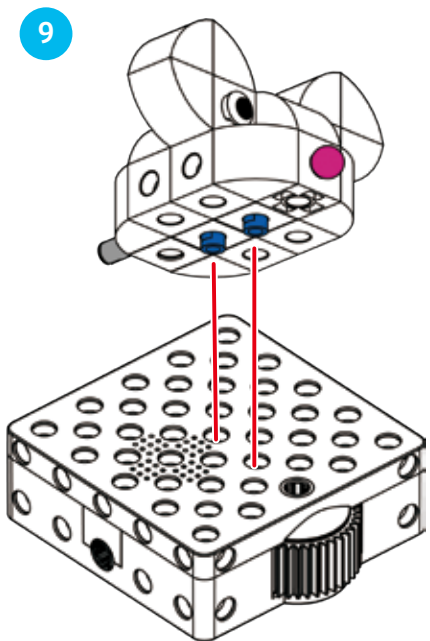
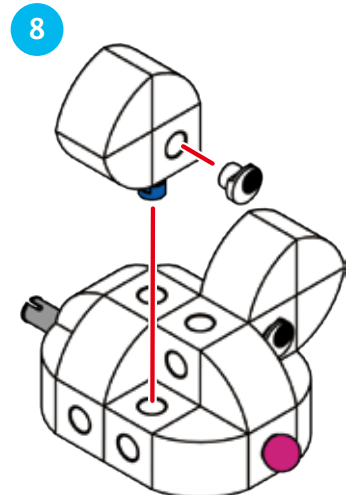
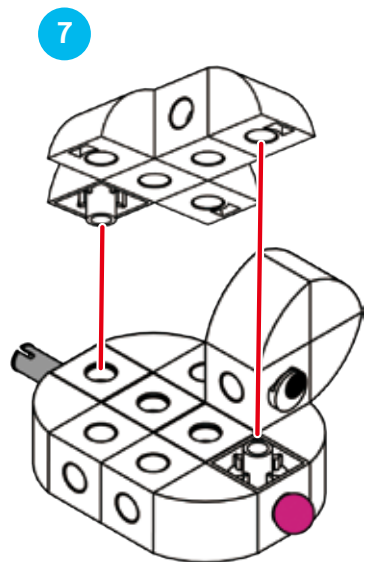
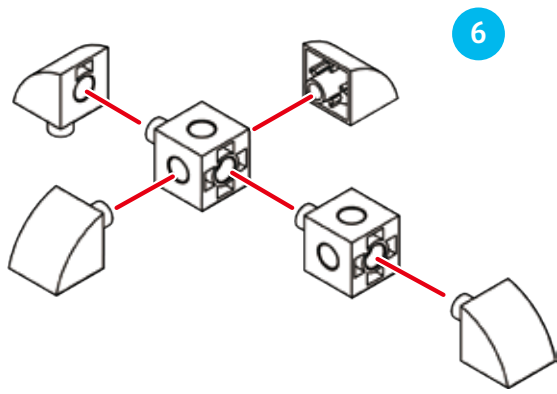
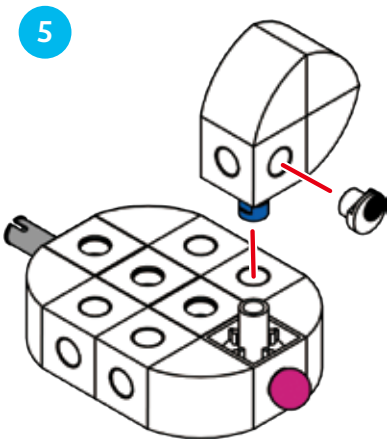
In this chapter, you will first build Pippy and two cheeses. Then, you can try Lessons 9, 10, and 11 to code Pippy to find the cheeses. You will learn about simple loops and functions. Then, build Purry and Barker and conduct Lesson 12 to finish the chapter.

### BUILD

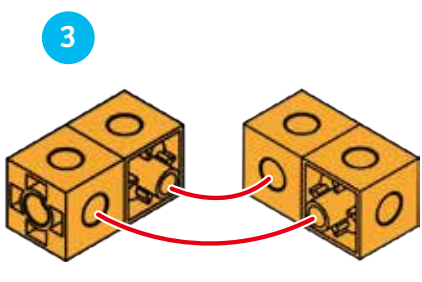
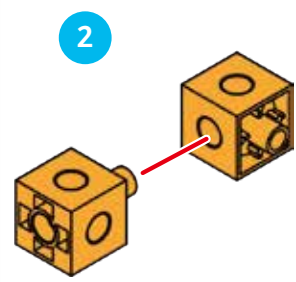
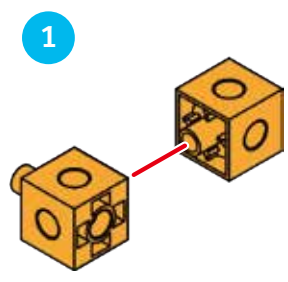
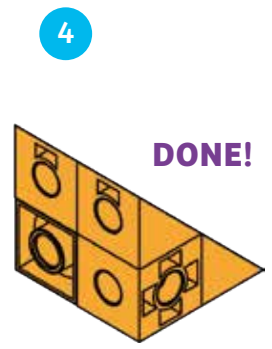
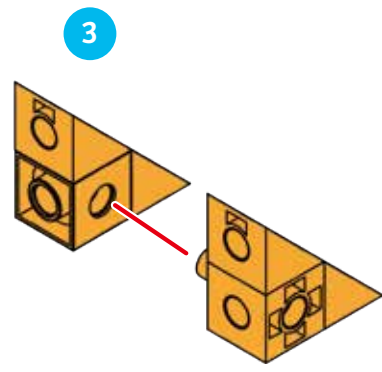
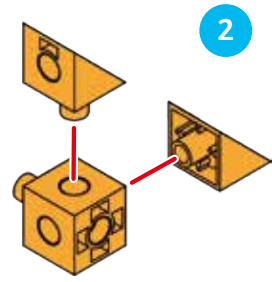
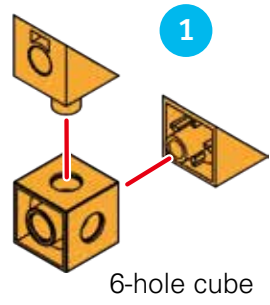
### PIPPY



CONTINUED ...



**BUILD TWO CHEESES**



● ○ ○ **LESSON 9**

**PIPPY IS LOOPY FOR CHEESE**



**STORY**

Pippy is looking for a yummy piece of cheese. She thinks there might be one on the table in the dining room. Place the cheese on the dining room map card. Can you code Pippy to find the cheese? What is the fewest number of code cards you can use to get Pippy to the cheese? Try using only Move Forward cards, number cards, and/or simple loop cards.

**CODE**

- A.
- B.
- C.
- D.

**WHAT'S HAPPENING**

- A. In this example, five Move Forward cards make Pippy move forward five map cards to the cheese.
- B. In this example, the Number 5 card executes the Move Forward card five times, bringing Pippy to the cheese.
- C. In this example, the Green Simple Loop is executed five times because of the Number 5 card. The loop is defined as one Move Forward card, so Pippy is moved forward five map cards to the cheese.
- D. In this example, the Green Simple Loop is only executed once, but the Number 5 card repeats the move forward command five times.

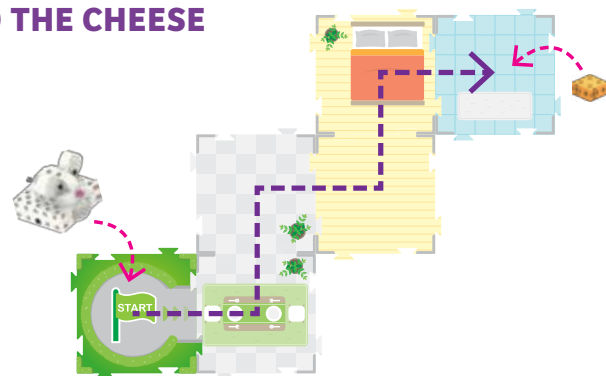
● ● ○ **LESSON 10**

**ZIG-ZAG TO THE CHEESE**

**STORY**

Again, Pippy is looking for cheese. For some reason, the cheese is in the bathroom this time. Can you write a code to bring Pippy to the cheese? Can you use a loop to do it efficiently?

**MAP**



**WHAT'S HAPPENING**

If you use the simple loop, you can repeat the action required to perform one zig-zag maneuver.

**CODE**

- A.
- B.



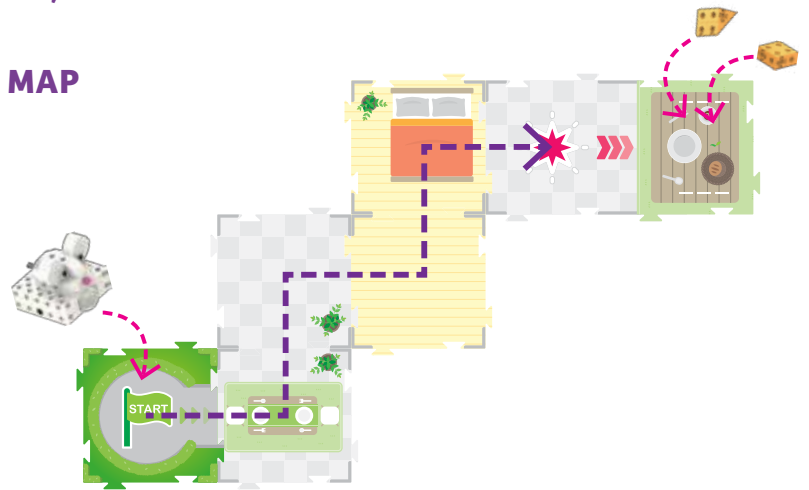
**STORY**

Pippy smells some cheese on the picnic table. It smells strong, so it must be two pieces of cheese! Can you program Pippy to first zig-zag through the house and into the backyard, and then to spin around in a circle when she reaches the cheese?

First, record the main program. Then, lift the robot up and record the function starting with the Red Function Start card. The robot will save both the main program and the function in its memory. Then run the program on the map.

**WHAT'S HAPPENING**

The zig-zag program works the same way it did in the previous lesson. But this time, there is a base map card (the card with the red star on it) at the end of the path. Because you recorded a program for the Red Function by placing code cards, the robot will run the Red Function when it reaches the base map card with the red star on it.

**MAP**

**CODE**
**MAIN PROGRAM:**

**RED FUNCTION:**


OR


**How to Use Functions**

Lesson 11 demonstrates how to use the three Function Start code cards along with the corresponding base map cards. The Red, Green, and Blue Functions all work the same way.

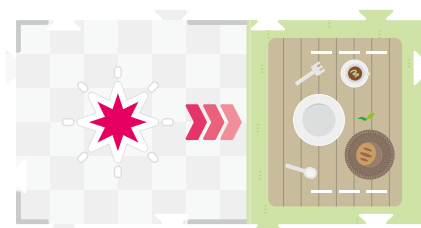
You can have up to 15 code cards in a function. The **Move Forward** and **Move Backward** code cards **don't work in functions**; if you try to use them, you will get an error.

The **Turn Output Gear** and **Pause Output Gear** code cards don't work in the main program. You can use these **only in functions or conditional statements**.

Functions are programmed with the **Function Start cards**:



A function runs when the robot scans the **star** on the corresponding base map card, when it is **facing in the same direction** as the three arrows on the base map card, and when there is a corresponding **function program** recorded in its memory.



The robot must be programmed to be facing the direction of the interaction position (i.e., facing in

the same direction as the three arrows). The robot can either enter the map card already facing this direction, or it can be turned with a turn code card to face this direction after entering the card.

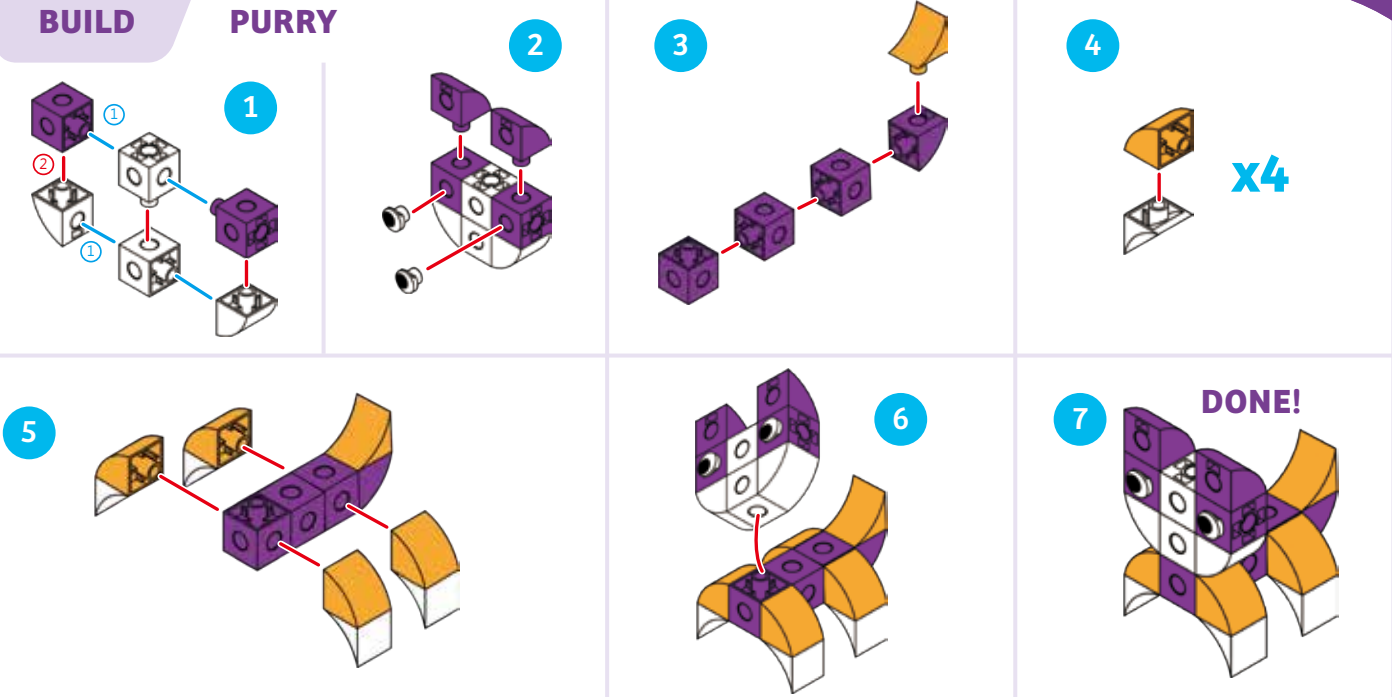
When the robot scans the base map card, it first orients itself on the star. Then, the robot advances toward the interaction point following the three arrows. Then, the function runs. Finally, the robot backs up to the star again.

When you want to use the output gear with a model on the base map card, you need to secure the model with these plastic straps, so it stays in place:



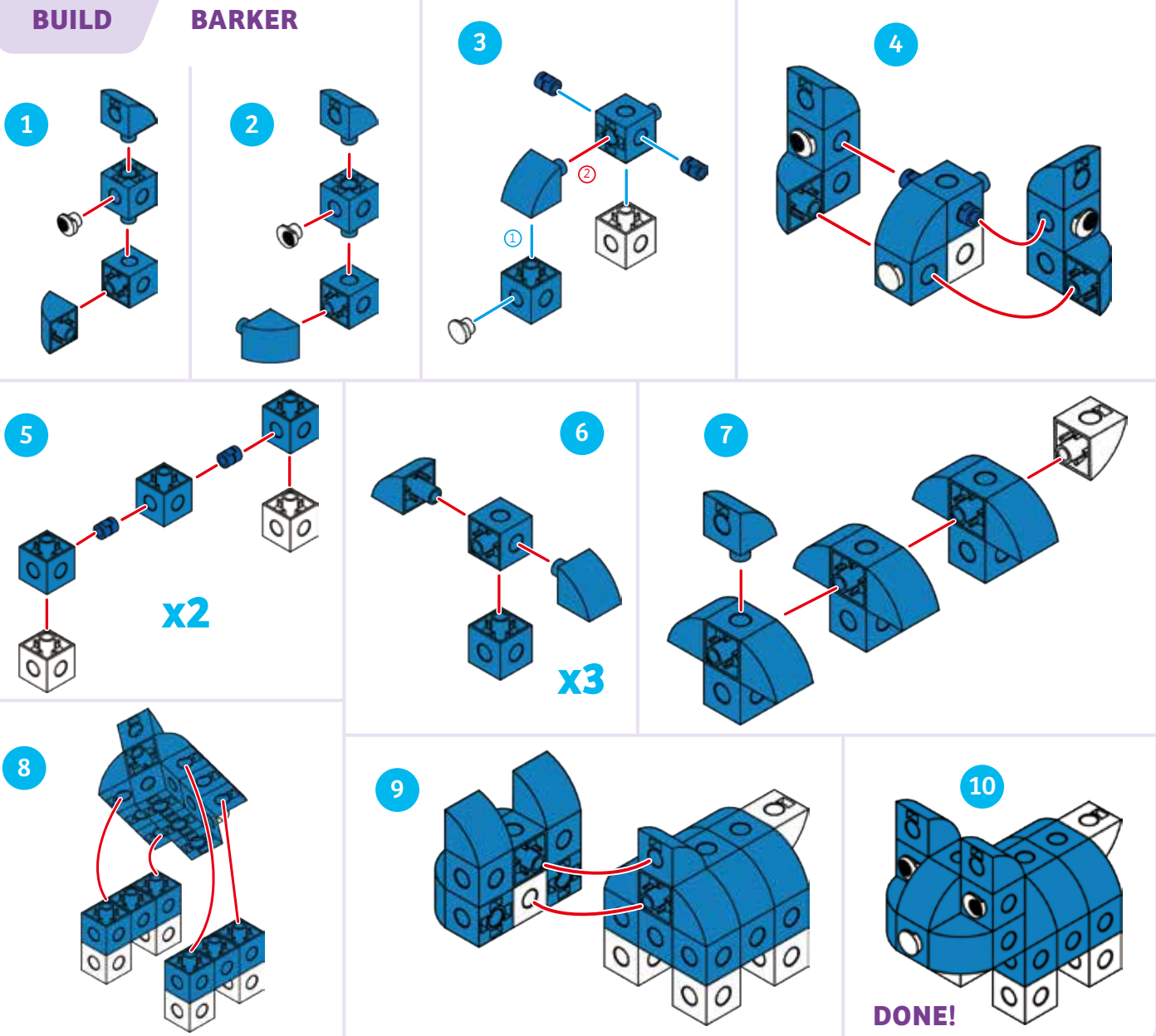
**BUILD**

**PURRY**



**BUILD**

**BARKER**



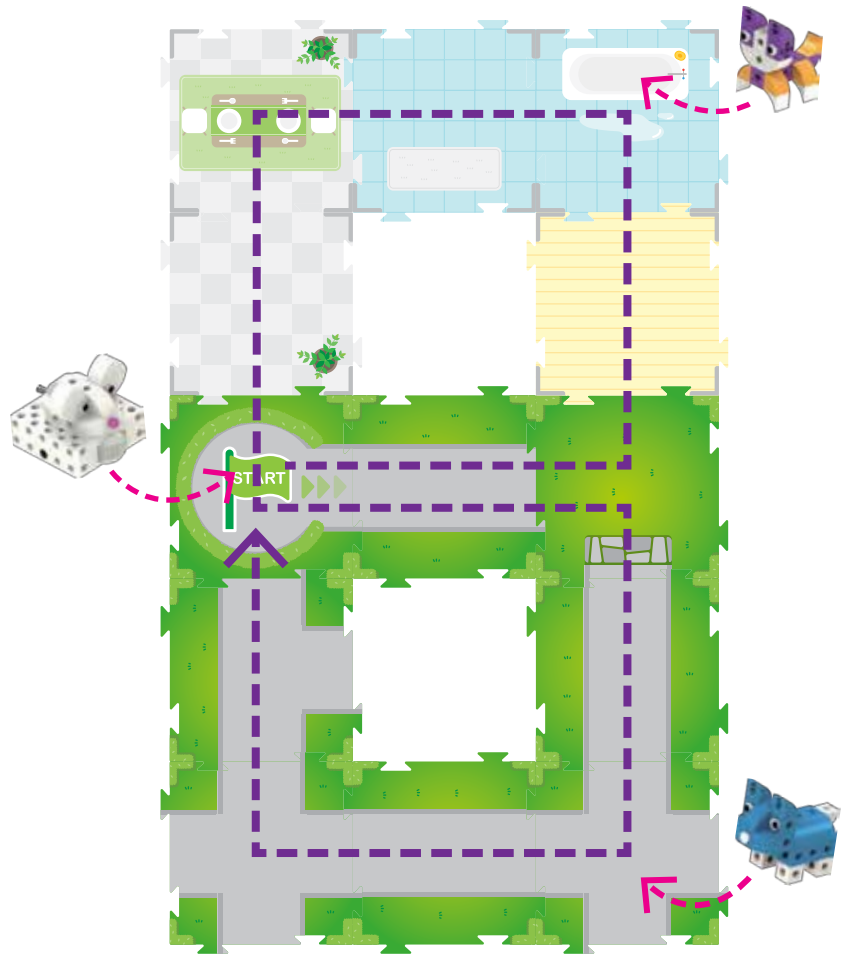
**STORY**

Pippy has lots of energy from all the cheese she has been eating. Now, she wants to play one of her favorite games: running around the house and the neighborhood, running right by Purry and Barker, and trying not to get caught. Can you program Pippy to run in a figure-eight shape around the map, past Purry and Barker, and back to the start again? Can you do it with two loops, to use fewer code cards?

**WHAT'S HAPPENING**

Three examples of programs that will complete this lesson are pictured below. Example A uses no loops and is almost twice as long as the other two. Examples B and C both use two loops in similar ways. The difference between Examples B and C is that Example C uses number cards to repeat the Move Forward commands.

**MAP**



**CODE**

A.

OR

B.

OR

C.



## Chapter 3: Arty's Party in the Park

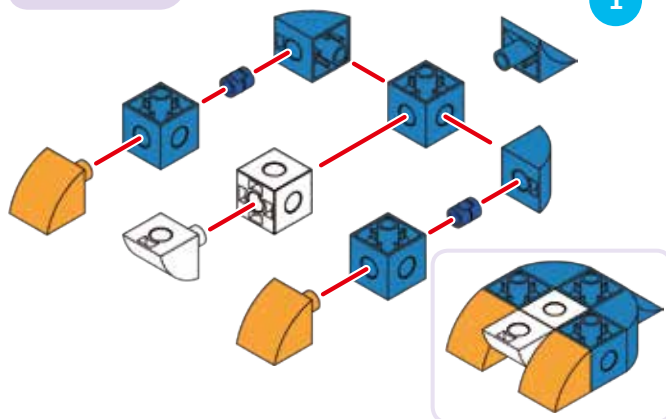
Arty is a penguin. Arty didn't like the cold weather in Antarctica, so he moved to a pleasant park with grass and trees. Arty lives in the park with his friend Tucker the Turtle.

In this chapter, first build Arty and Tucker, and do Lesson 13 with them. Then, build Gully and do the rest of the lessons with all three models. You will start to use more complex functions in this chapter. You will also be introduced to the output gear functionality, so you can make the robot interact with models placed on the base map cards. Mechanical outputs are a fundamental aspect of robotics.

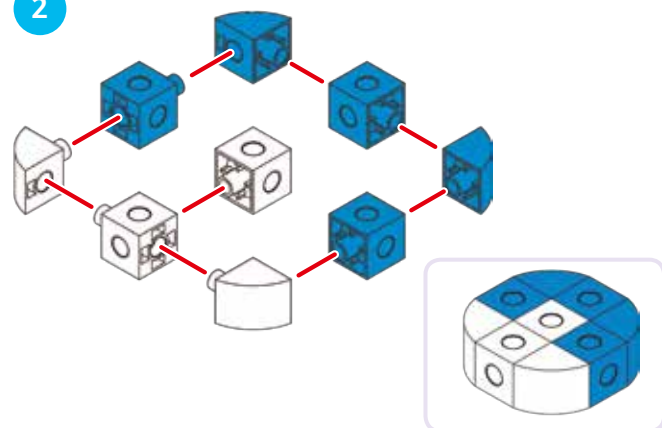
### BUILD

### ARTY

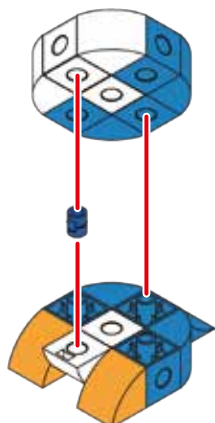
1



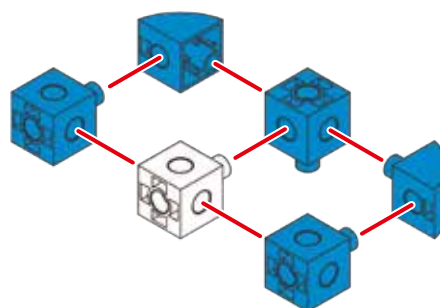
2



3

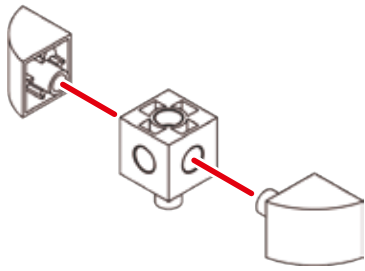


4

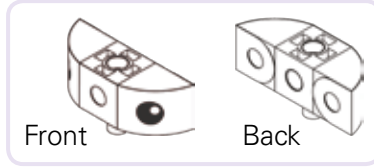
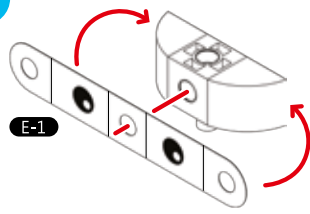


CONTINUED ...

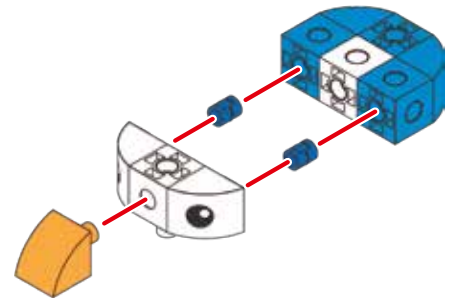
5



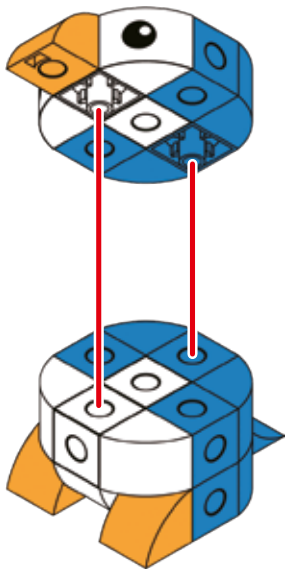
6



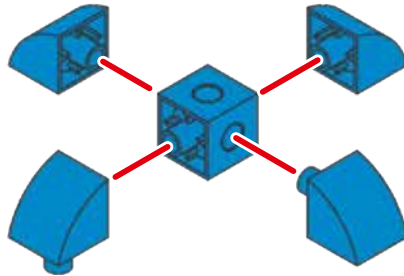
7



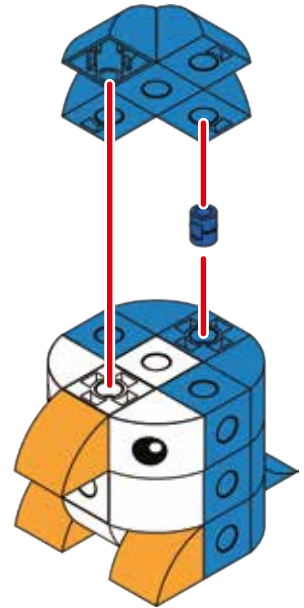
8



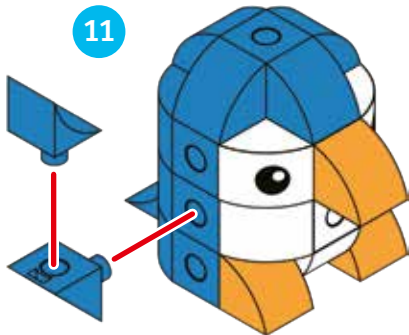
9



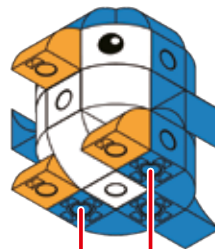
10



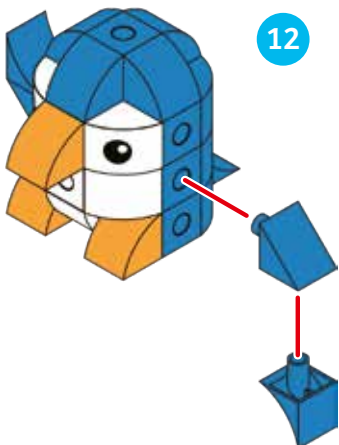
11



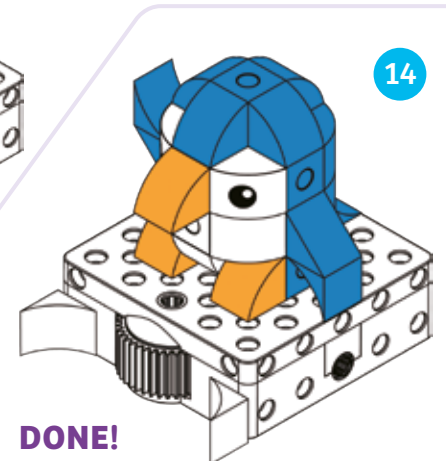
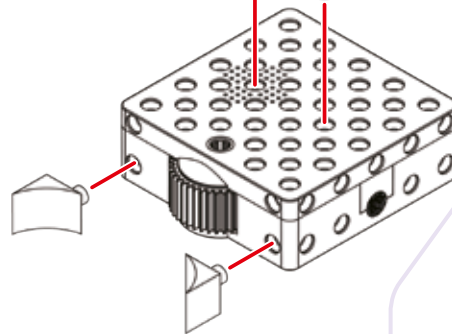
13



12



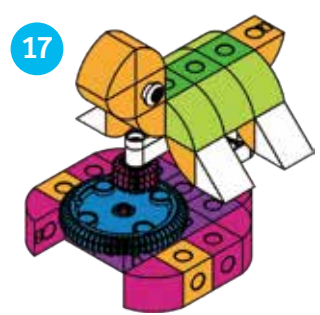
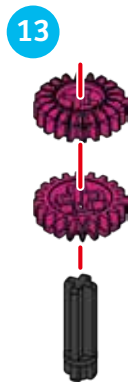
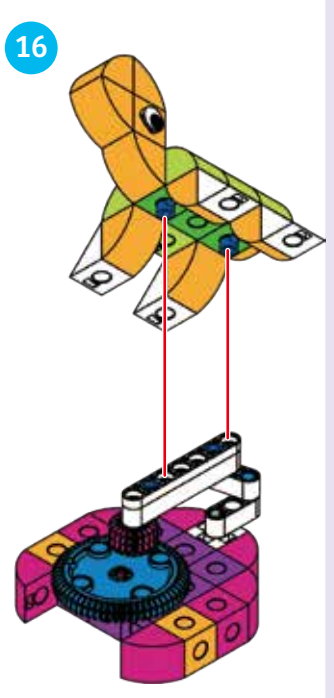
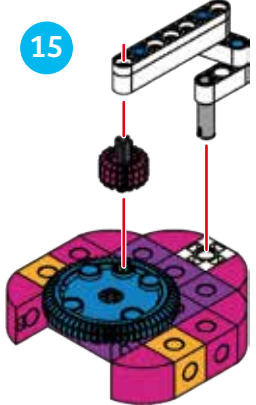
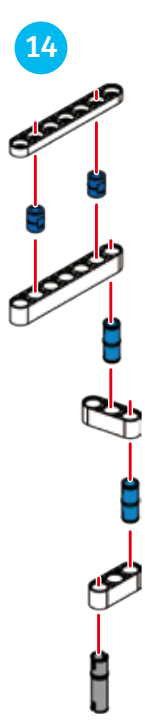
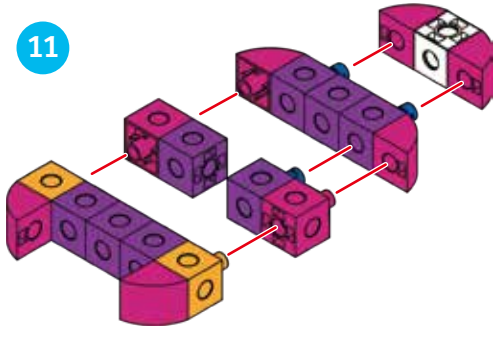
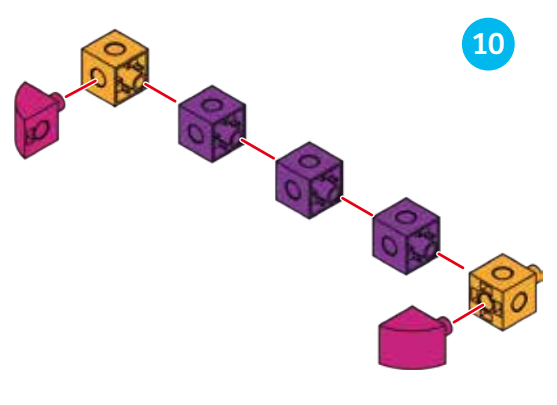
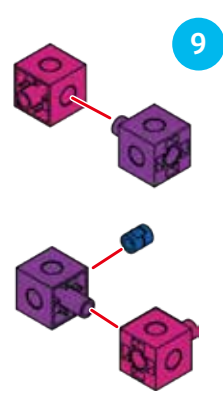
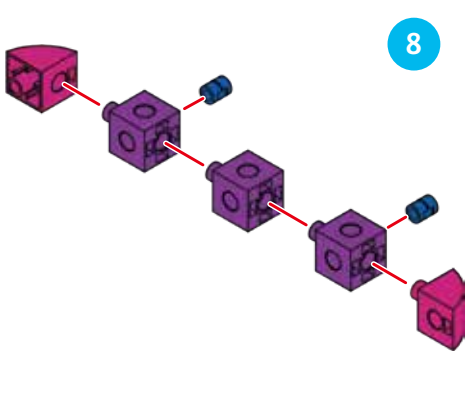
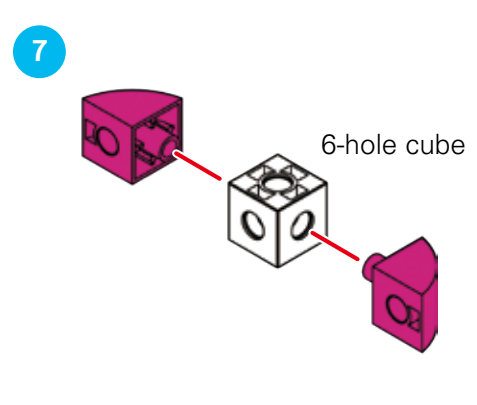
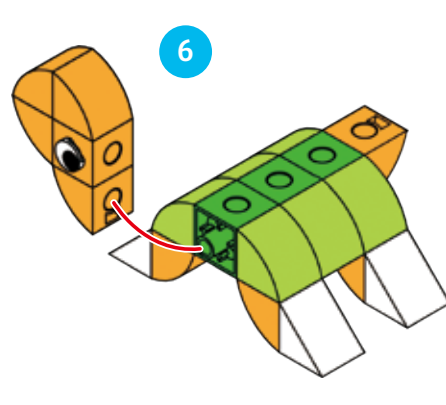
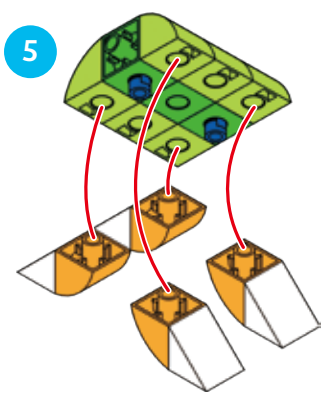
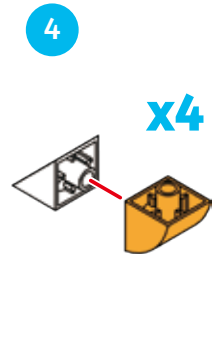
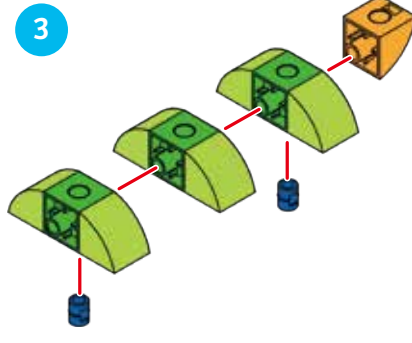
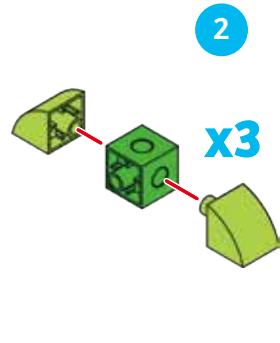
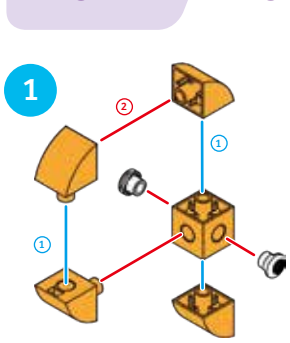
14



**DONE!**

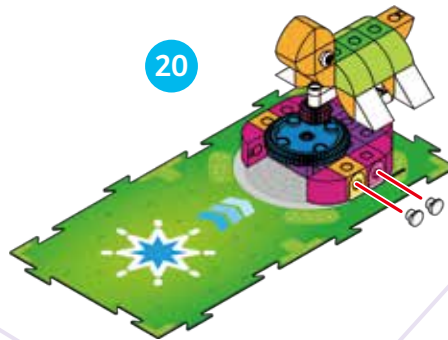
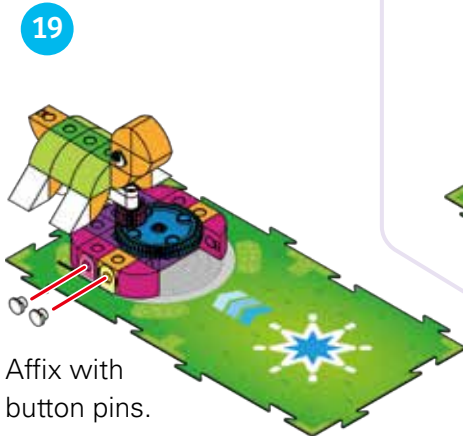
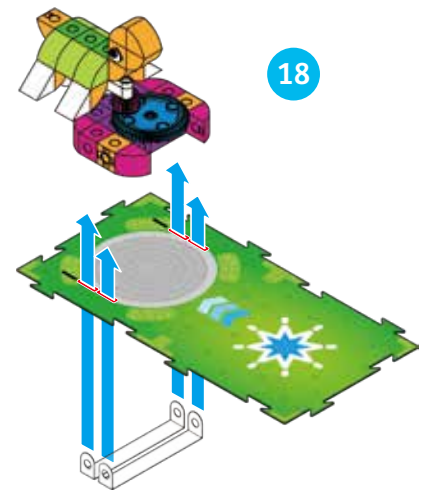
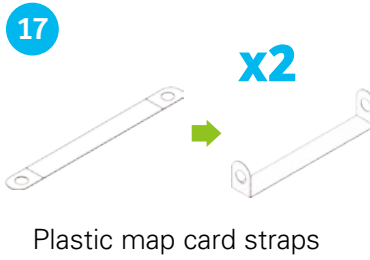
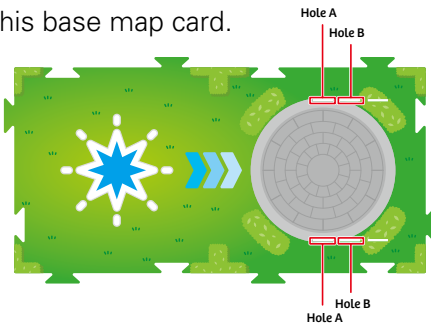
**BUILD**

**TUCKER**



**CONTINUED ...**

Now attach the turtle to this base map card.

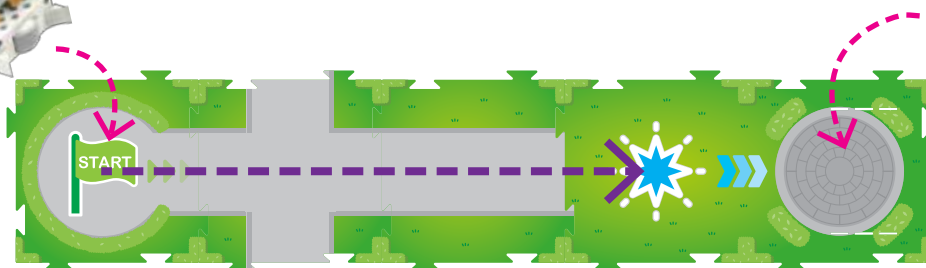


## LESSON 13

## ARTY DANCES WITH TUCKER



### MAP



### STORY

Arty wants to visit his friend Tucker. Can you write a program to make Arty drive to Tucker, and when he gets there, to perform a function to spin Tucker around in circles?

### WHAT'S HAPPENING

The main program brings Arty to the base map card with the blue star on it. When the robot scans the base map card, the robot automatically moves into position and runs the Blue Function code, which instructs the robot to turn the output gear first clockwise and then counterclockwise. The gear meshes with the gear connected to Tucker, so this makes Tucker turn around as well.

### CODE

#### MAIN PROGRAM:

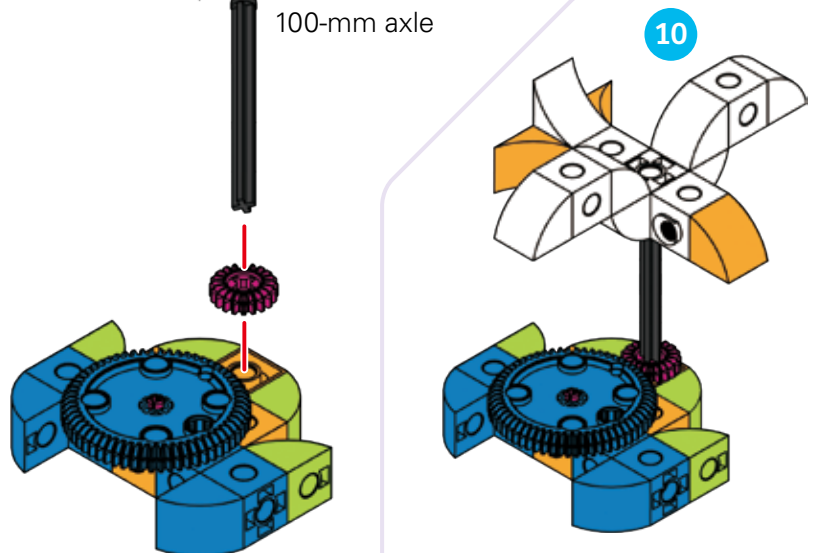
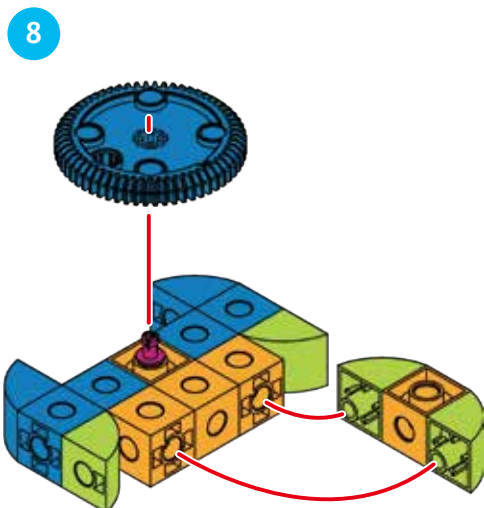
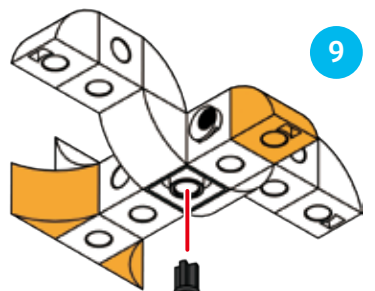
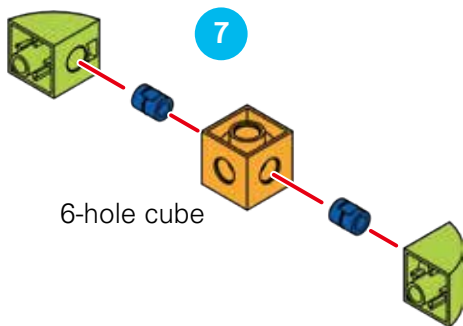
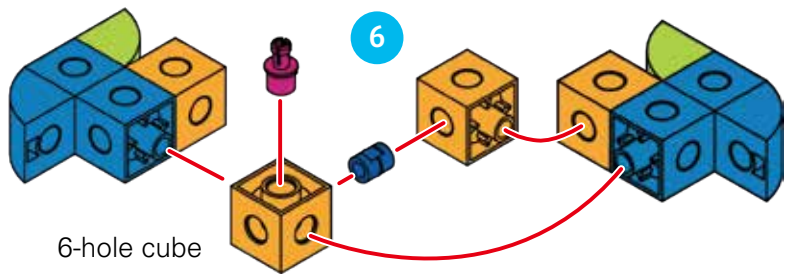
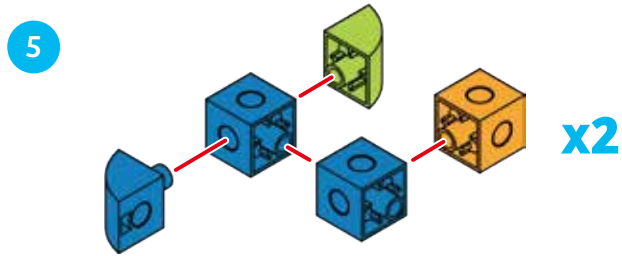
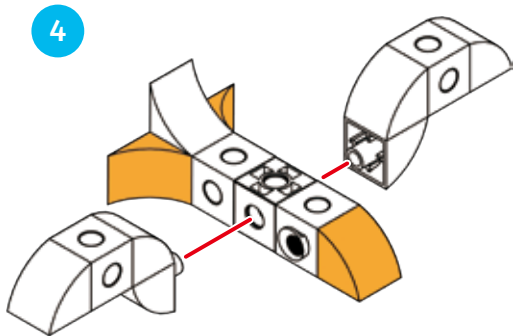
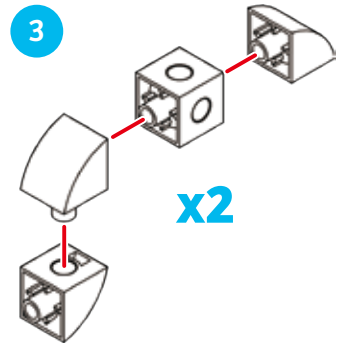
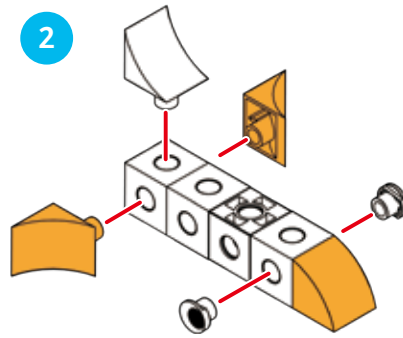
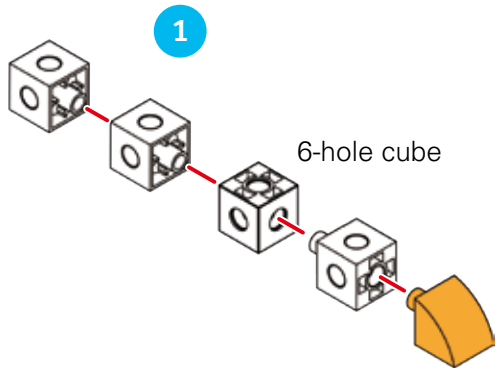


#### BLUE FUNCTION:



**BUILD**

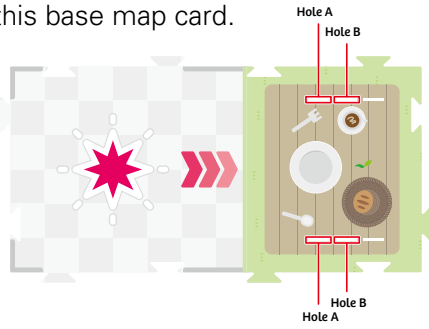
**GULLY**



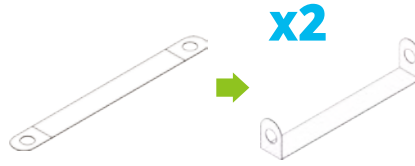
**CONTINUED ...**



Now attach the bird to this base map card.

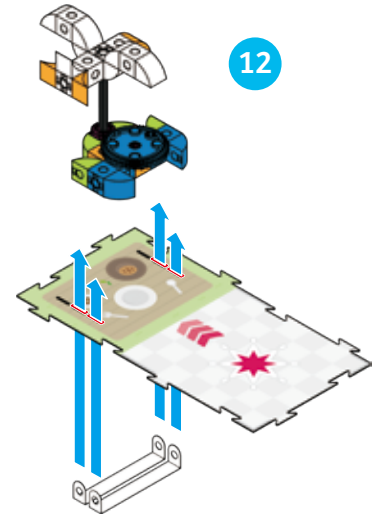


11

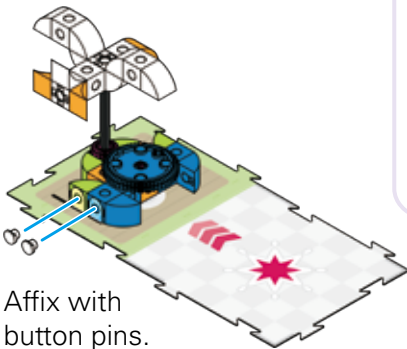


Plastic map card straps

12

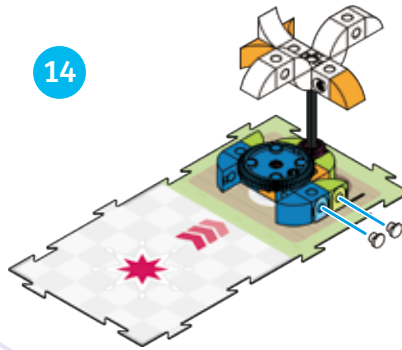


13

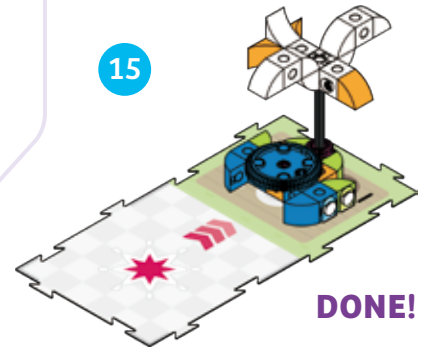


Affix with button pins.

14



15



**DONE!**

● ● ● **LESSON 14**

**COMMOTION IN THE PARK**

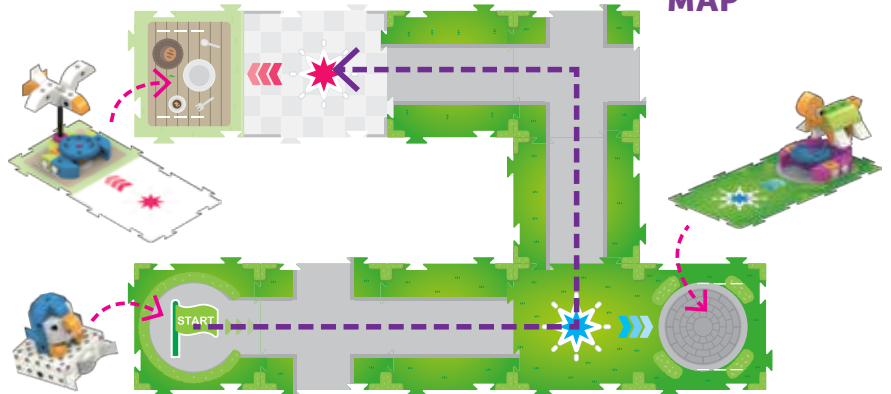
**STORY**

Arty visits Tucker, who dances around because he's so happy to see Arty. Then Arty goes to his picnic table, and sees that Gully is trying to grab his fish dinner! He does an elaborate dance to shoo Gully away from his food.

**WHAT'S HAPPENING**

The main program brings Arty to Tucker, who spins around just like in the previous lesson. Then, the main program moves Arty along to the base map card with the red star on it. Here, the Red Function causes Arty to perform a clockwise and counterclockwise spinning action two times. This causes Gully to spin around, as if to fly away from Arty's dinner.

**MAP**



**CODE**

**MAIN PROGRAM:**



**BLUE FUNCTION:**



**RED FUNCTION:**



● ● ○ **LESSON 15**

**ARTY STARTS SQUAWKING**

**STORY**

Arty visits Tucker again, but this time Arty talks to Tucker when he sees him. Do you recognize part of Lesson 14 that can be repeated in a simple loop? Repeat Lesson 14, but this time try using a simple loop in the main program. Also, try adding a sound code card to the Blue Function.

**WHAT'S HAPPENING**

The addition of the Play Sound: Penguin code card causes the robot to play a penguin sound when the Blue Function runs.

**MAP**

Use the same map as in Lesson 14.

**CODE**

```

[START] → → → [Loop: 2] [Turn Right] → → [Loop: 2] [STOP]
[START] [Play Sound: Penguin] [Loop: 2] [Turn Right] [STOP]
[START] [Play Sound: Starburst] [Loop: 2] [Turn Right] [Loop: 4] [Turn Right] [Loop: 4] [Loop: 2] [STOP]
    
```

● ● ○ **LESSON 16**

**ARTY'S COMPLETE TRIP**

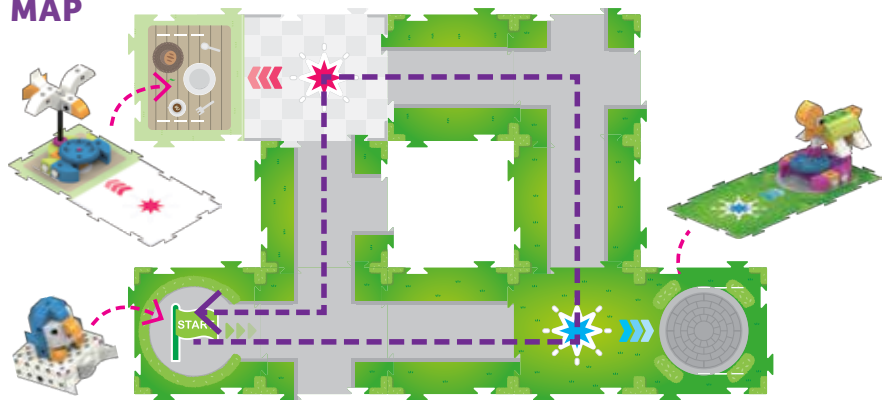
**STORY**

This time, Arty wants to go back to his starting point after he gets Gully to leave his food alone. Repeat Lesson 15, but this time add a map card that allows Arty to get from the base map card back to the start. Edit the main program to bring Arty back to the start. And also, add some more penguin sounds to the interaction Arty has with Gully.

**WHAT'S HAPPENING**

The addition of the Play Sound: Penguin code card to the Red Function code causes the robot to play a penguin sound when the Red Function runs.

**MAP**



**CODE**

```

[START] → 3 [Loop: 3] [Turn Right] → → [Loop: 2] [Turn Right] [STOP]
[START] [Play Sound: Penguin] [Loop: 2] [Turn Right] [STOP]
[START] [Play Sound: Starburst] [Loop: 2] [Turn Right] [Loop: 4] [Turn Right] [Loop: 4] [Loop: 2] [Play Sound: Penguin] [STOP]
    
```

## Conditional Statements

Computers and robots often have to make **decisions** to perform tasks or execute programs correctly. Programmers use conditional statements to give computers and robots the ability to make decisions. For example, imagine a robot that makes peanut butter and jelly sandwiches. If its program calls for chunky peanut butter, then the robot will use chunky peanut butter. If its program calls for smooth peanut butter, then the robot will use smooth peanut butter.

A **conditional statement** is a portion of a program that tells the computer or robot to perform different instructions depending on whether or not a specified condition, or set of conditions, is **true or false**. Conditional statements are often just referred to as **conditionals**.

Conditional statements are also known as **if-then** statements: If a condition is true, then the program will run the instructions. If the condition is not true, then the program will not run the instructions.

There are also **if-then-else** conditionals. In this case, if the condition is true, then it will run one set of instructions. If the condition is not true, it will run a different set of instructions.

Programmers can also combine conditions in different ways. For example, a program could require two conditions be true before a set of instructions is run. This is called an **And** operation, because both the first condition **and** the second condition must be true for the whole statement to be true. Another example is the **Or** operation. In this case, either the first condition **or** the second must be true for the whole statement to be true. There are more types of operations like these, but the coding language in this kit only covers And and Or operations in conditional statements.

In the coding language in this kit, conditional statements are always triggered by **event cards**.

## Events

In programming, an **event** is an occurrence or an interaction that can be recognized by the computer or robot. Robots often use **sensors**, which are electronic devices that can detect changes in the

robot's environment or state. For example, your robot's OID optical sensor can detect the patterns on the code cards. The robot can then be programmed to react to the patterns that it senses.

In this kit, there are **four event cards** with different symbols on them. With the conditional statements, the robot can be programmed to perform specific actions if it senses these event cards with its optical sensor.

## How to Use the Condition and Event Cards

Conditional statements are set up as subroutines, separate from the main program, in the coding language in this kit.

A conditional statement must start with the **If** card. The If card has a little green flag on it to signify that it is the start of a subroutine. The If card must be followed by one, and only one, of the four event cards.

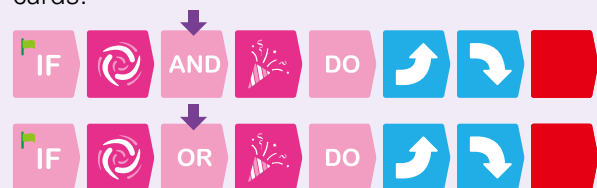


A conditional statement also must have the **Do** card. This could also be called a Then card. The Do card must be followed by the code cards for the code you want the robot to run in case the If statement is true — in other words, if the robot scans the event card(s) needed for the If statement to be true.

You can use the **Else** card after the Do statement to tell the robot what to do if the If Statement is not true. You can have up to 15 code cards after the Do card and 15 code cards after the Else card.



You can also add the **And card** and the **Or card** to the If statement. When used, these cards must be followed by one, and only one, of the four event cards.



You can experiment with conditional statements and events starting in Lesson 18.



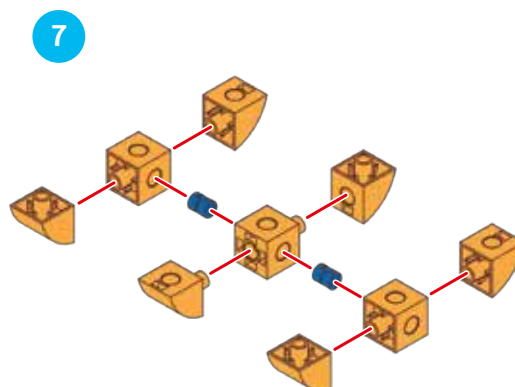
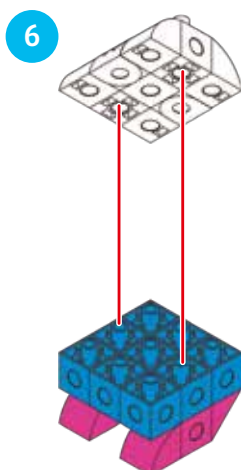
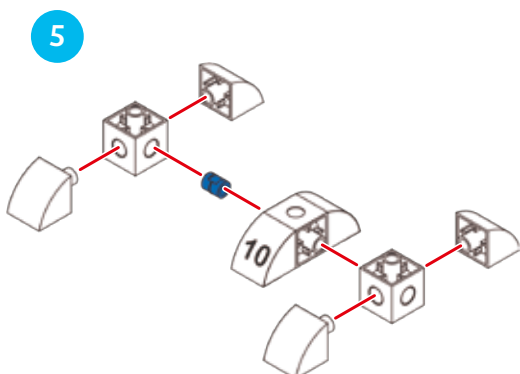
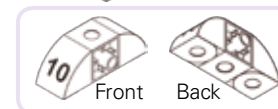
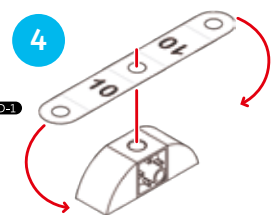
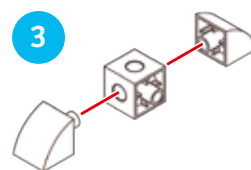
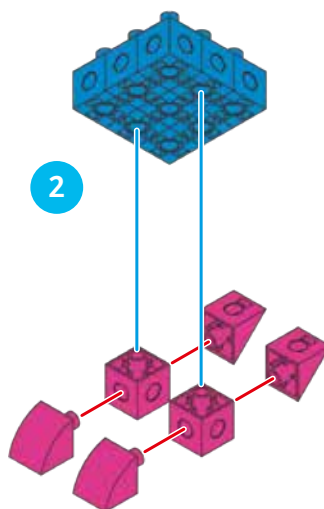
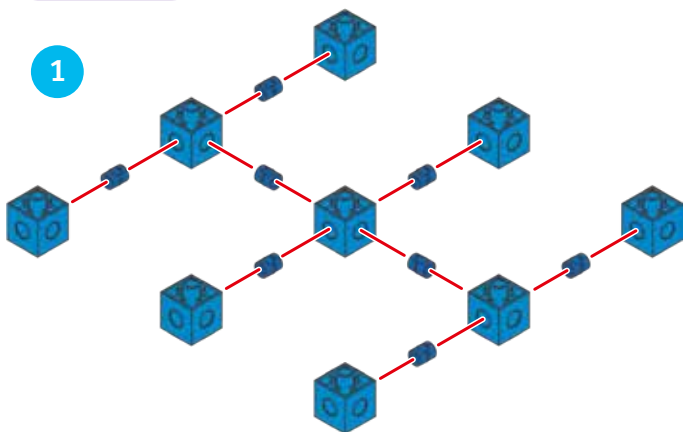
## Chapter 4: Robbie's Robotic Soccer Game

Robbie is a robotic soccer player. He uses a ball-launching mechanism to roll the ball into the goal when he gets close enough. You can program Robbie to shift from side to side when he encounters opponents on the field to try to fake them out.

In this chapter, you will first build Robbie, the goal, the ball, and some opponent players. Then, you will program Robbie to move the ball to the goal. Conditional statements and events are introduced.

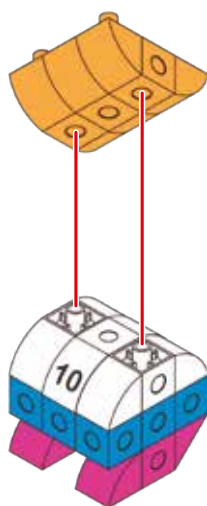
### BUILD

### ROBBIE

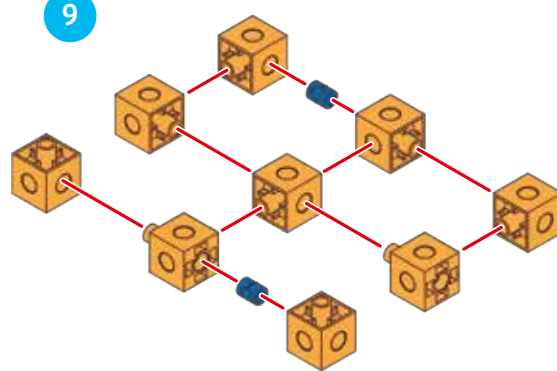


CONTINUED ...

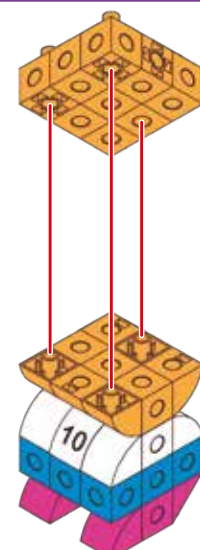
8



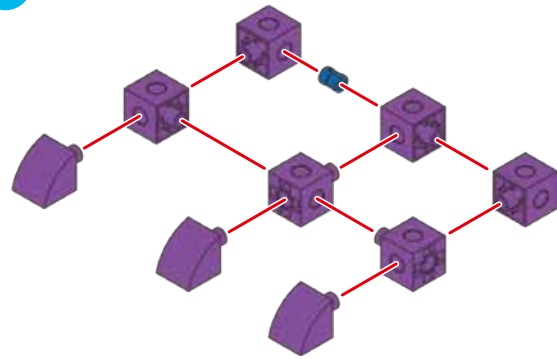
9



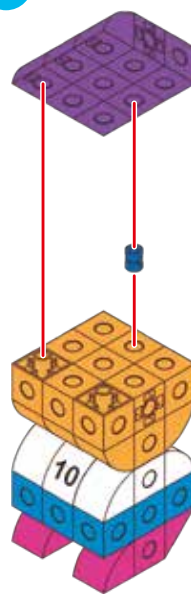
10



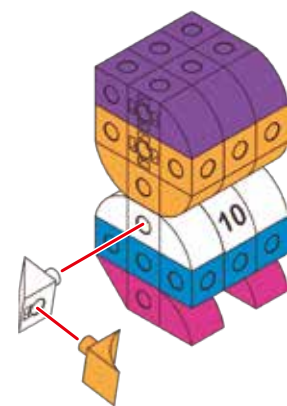
11



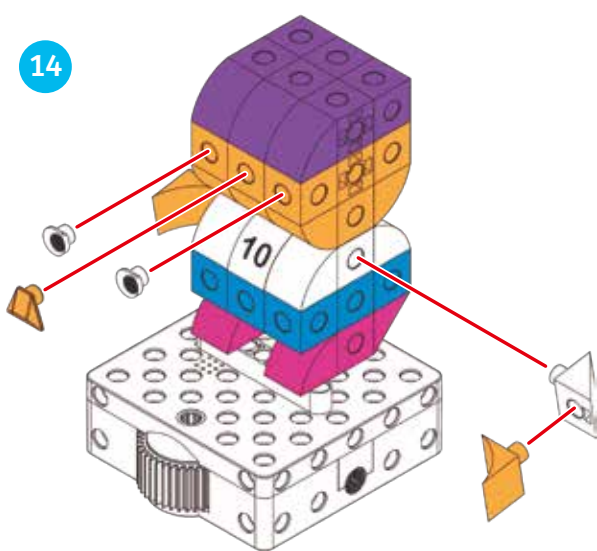
12



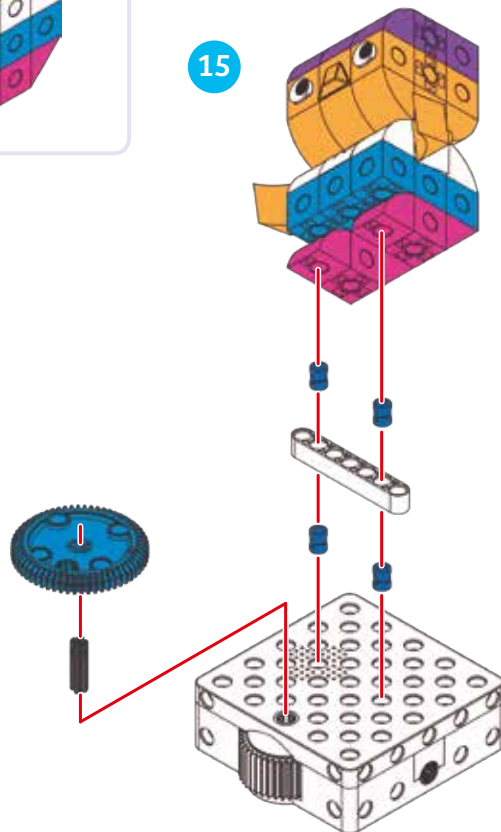
13



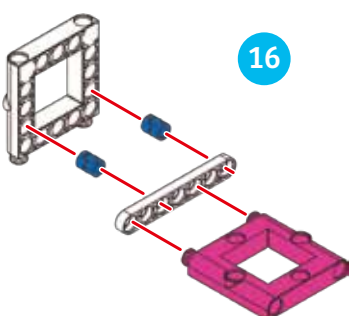
14



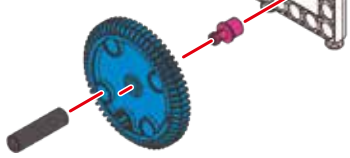
15



16

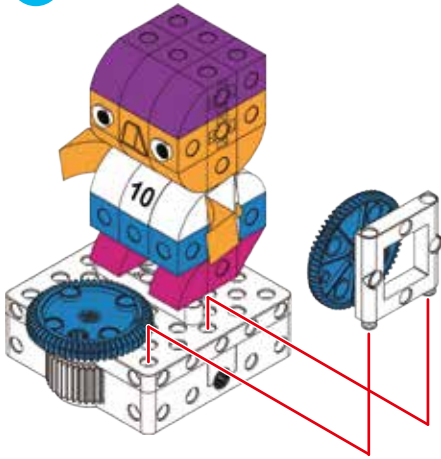


17

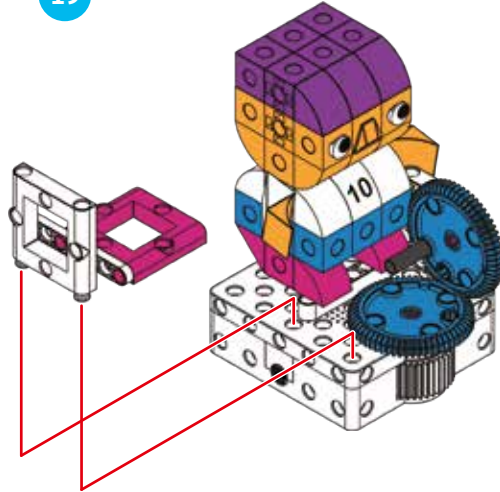


CONTINUED ...

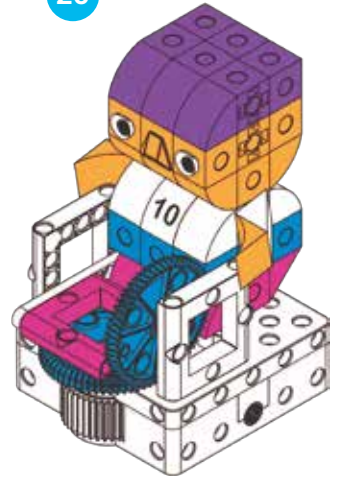
18



19



20

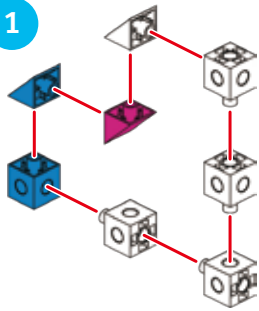


**DONE!**

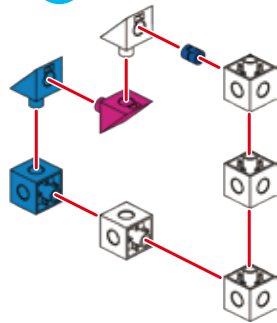
**BUILD**

**GOAL**

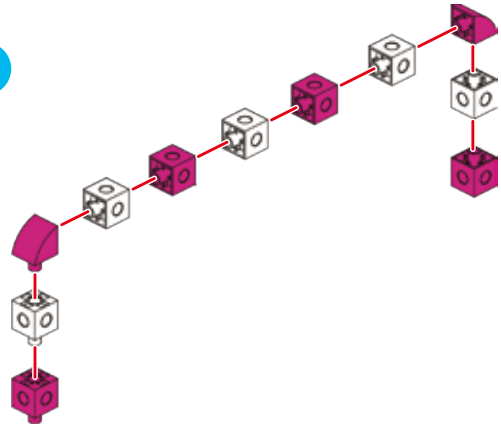
1



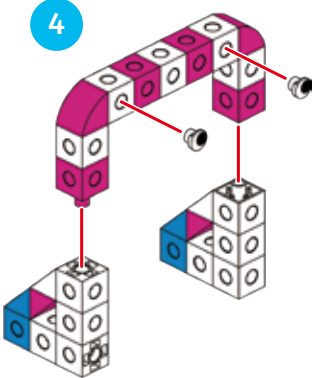
2



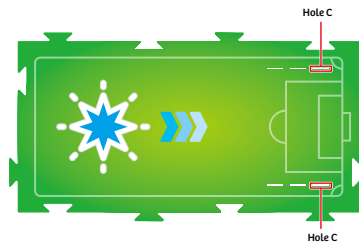
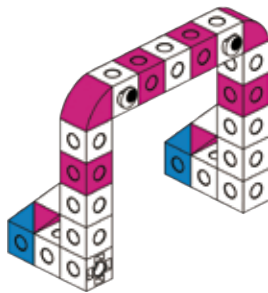
3



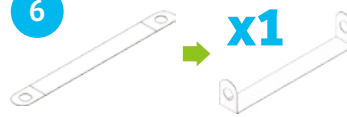
4



5

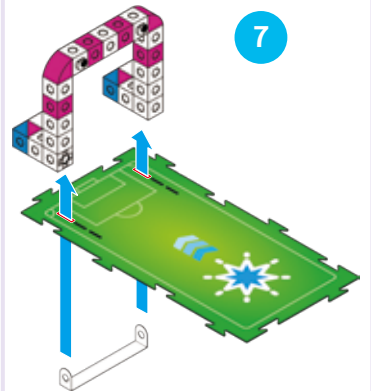


6



Plastic map card straps

7



8

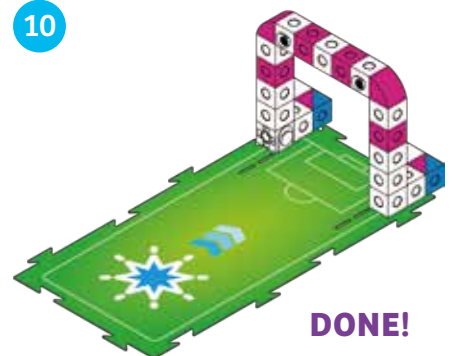
Affix with button pins.



9



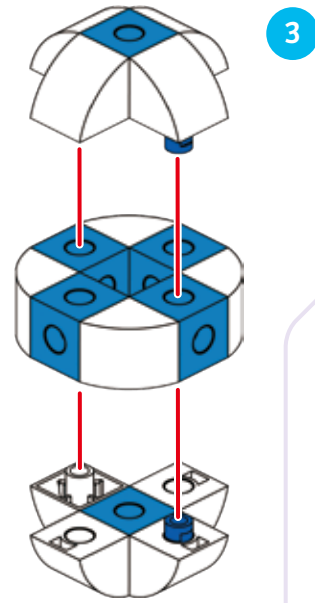
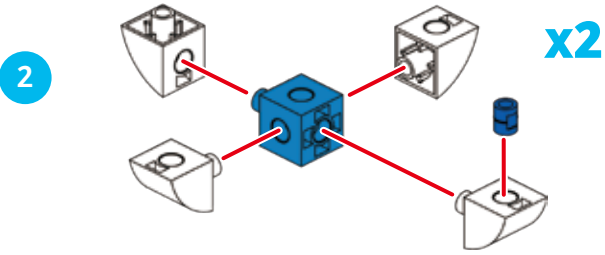
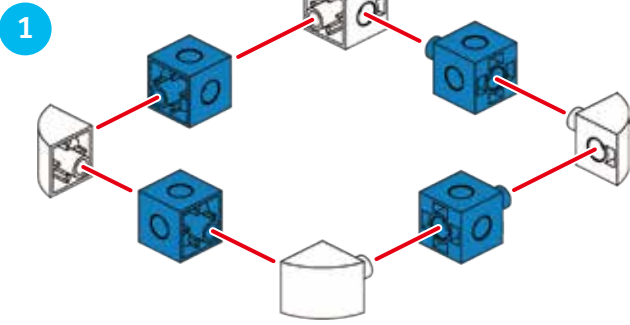
10



**DONE!**

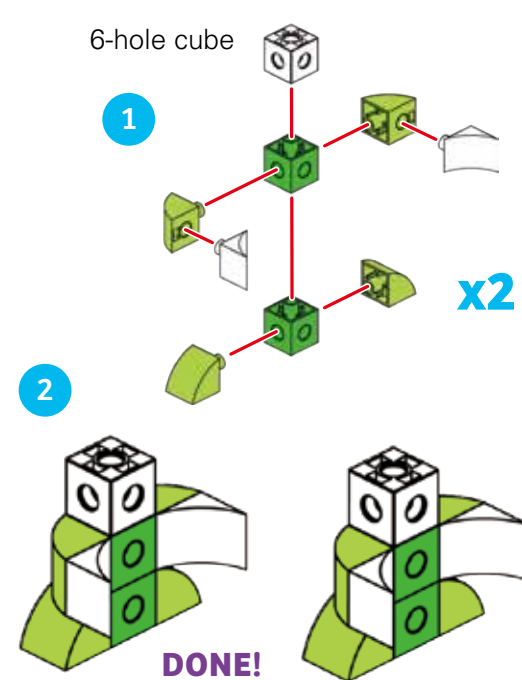
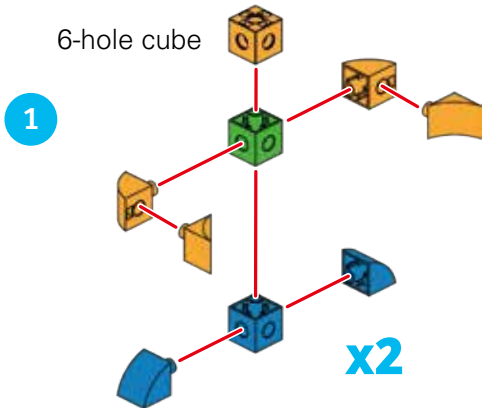
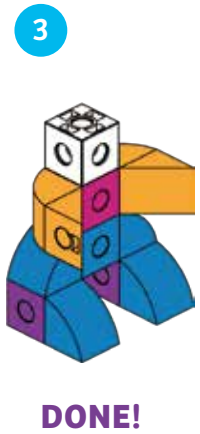
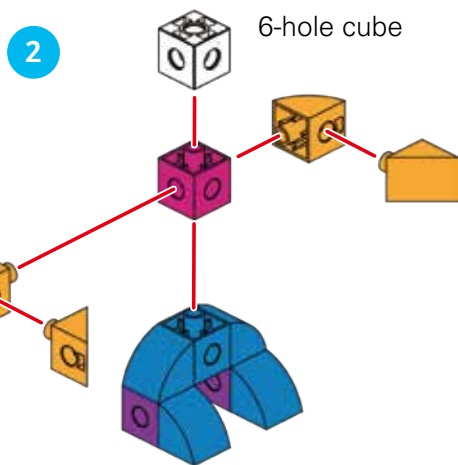
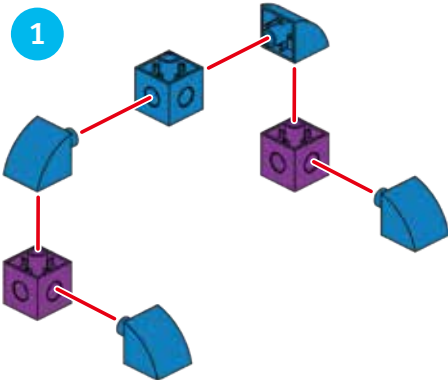
**BUILD**

**SOCCER BALL**



**BUILD**

**SOCCER PLAYERS**



● ● ● LESSON 17

STRAIGHT SHOT

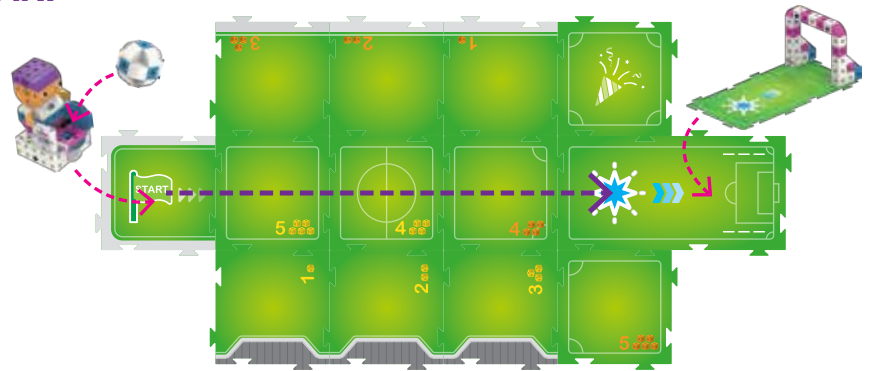
STORY

During soccer practice, Robbie tries running straight down the full length of the field and kicking the ball into the goal. Can you program Robbie to move to the goal and launch the ball? Load the ball onto Robbie's launching mechanism at the beginning.

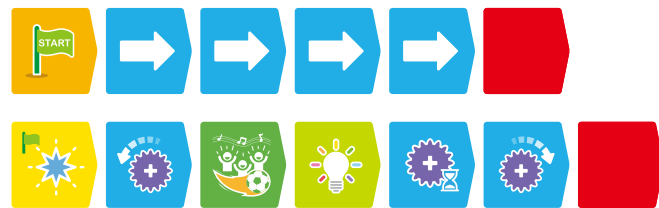
WHAT'S HAPPENING

The main program moves Robbie to the base map card with the goal on it. The Blue Function launches the ball, plays a cheering sound and lights up the light. You may have to try a few times to make a goal.

MAP



CODE



● ● ● LESSON 18

ROBBIE'S GAME-DAY DECISIONS

STORY

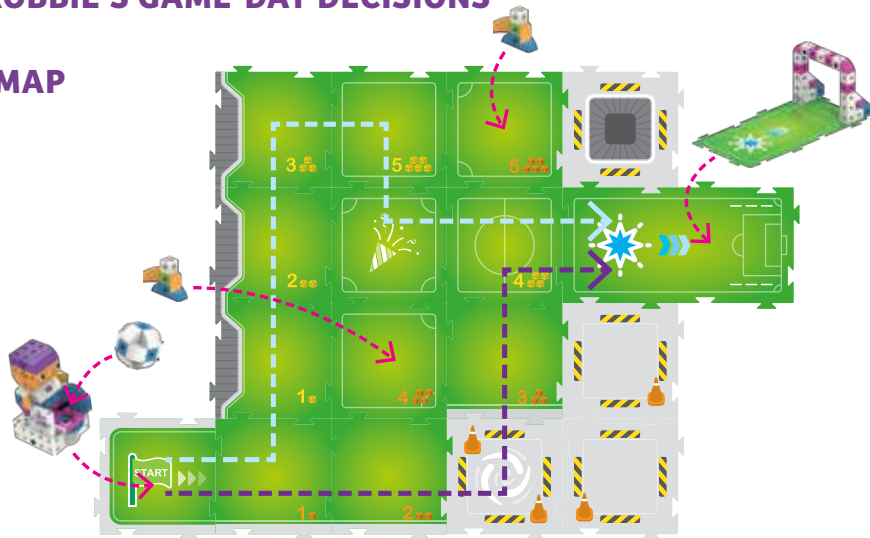
During the soccer game, Robbie has to move the ball from one side of the field to the other, avoiding the opponents on the field. Place the two opponents on the map cards as shown. Can you program Robbie to get to the goal?

WHAT'S HAPPENING

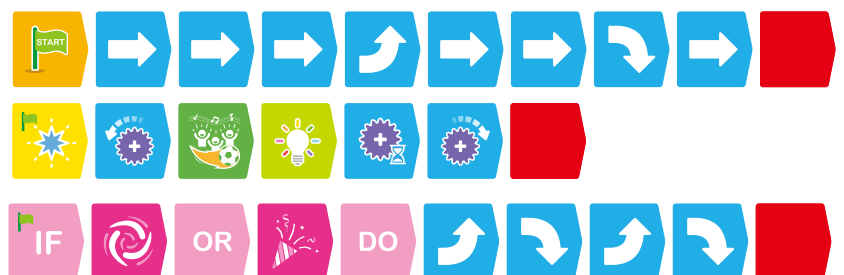
The main program follows the purple line to the goal. The Blue Function works like it did in the previous lesson to release the ball.

Now, there is a conditional statement in the code. This says: If the robot scans the Event 3 or Event 2 cards, then the robot should turn left and right two times, in a fake-out maneuver. Then the main program continues. The opponents are placed near the event cards.

MAP



CODE



CODING CHALLENGE

Can you program Robbie to follow the light blue line to the goal?



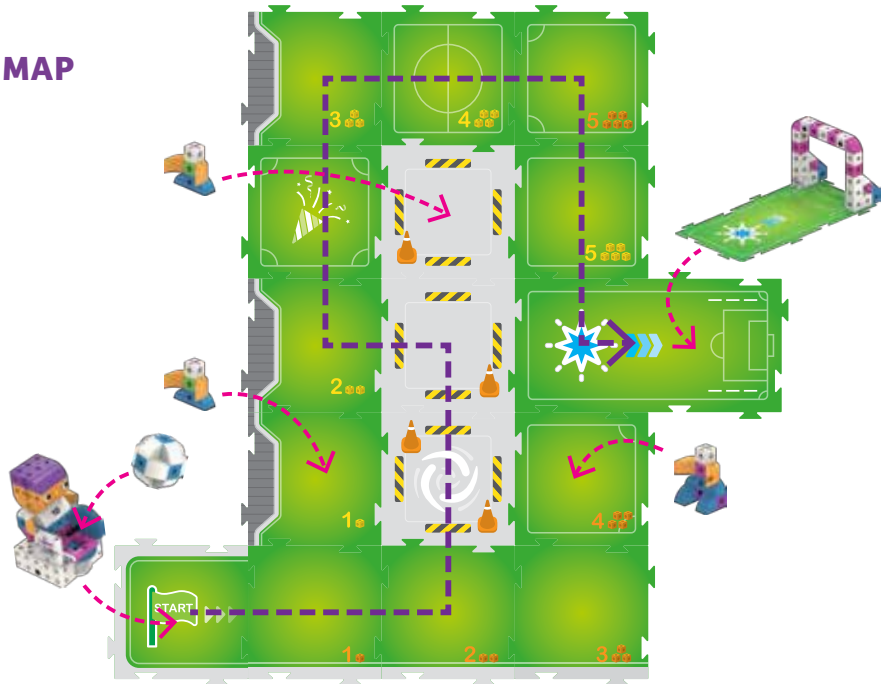
**STORY**

Now there are three opponents on the field. Robbie has to maneuver his way past them to get to the goal.

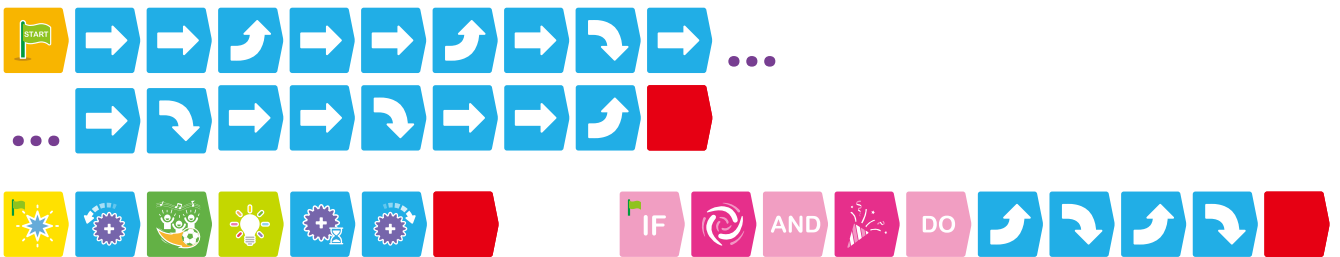
**WHAT'S HAPPENING**

The main program moves Robbie from the start to the goal, avoiding the opponents. The Blue Function works the same as in the previous lesson. But this time, the conditional statement has an And card in it, requiring the robot to have scanned both event cards before the statement is true.

**MAP**



**CODE**



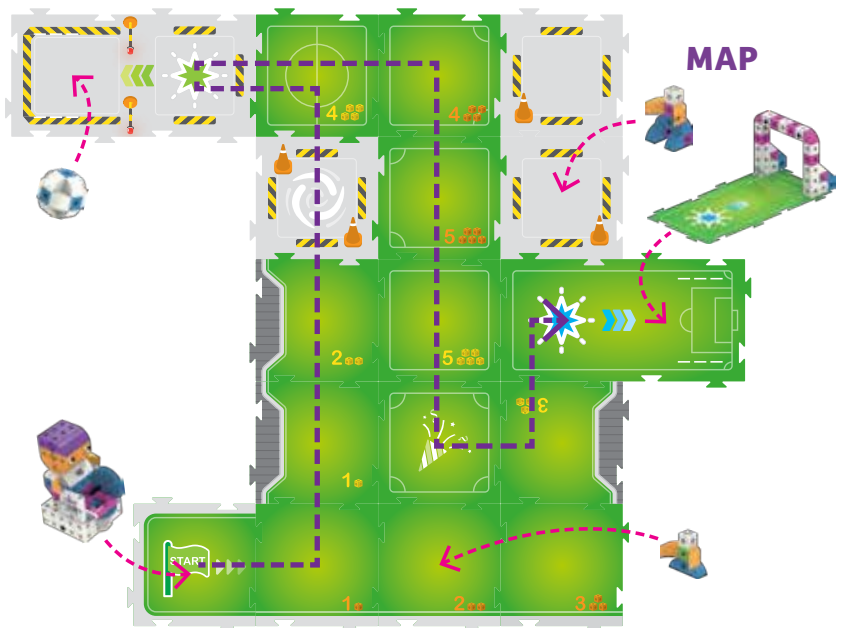
**STORY**

It's the day of the big game. Robbie has to run to a spot on the field where he can pick up the ball. When he gets there, he pauses and you can place the ball on Robbie's launcher. Then Robbie runs to the goal, shoots, and ... !

**CODING CHALLENGE**

Write a main program to move Robbie to the goal. The Blue Function and the conditional statement will be the same as before. You will need to add the Green Function, which tells Robbie to pause at the base map card with the green star on it.

**MAP**



**CODE**

**GREEN FUNCTION:**





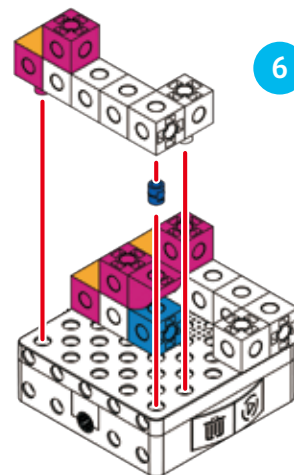
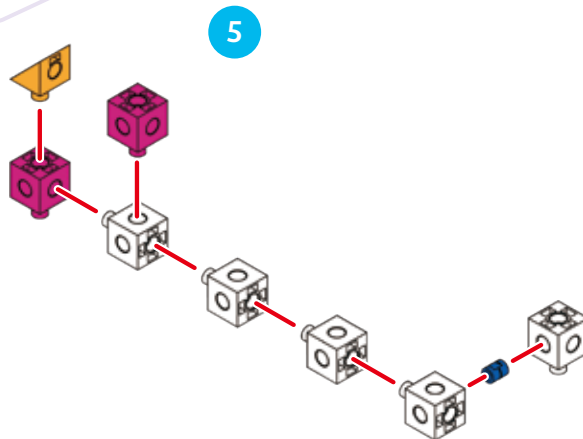
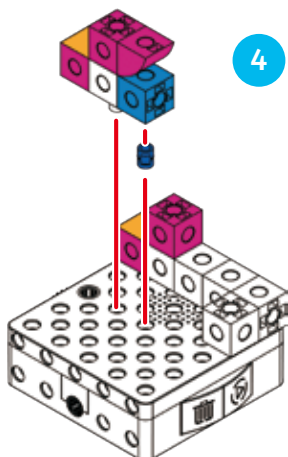
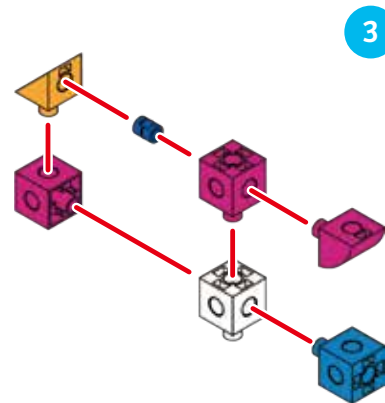
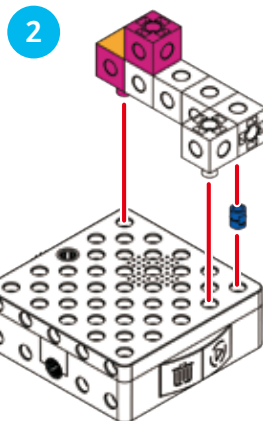
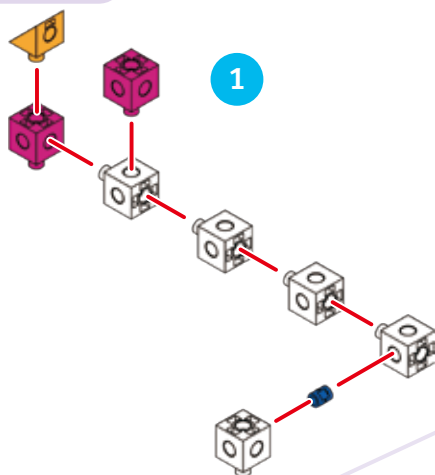
## Chapter 5: Robotic Fire Truck to the Rescue

This robotic fire truck can drive through town to a fire and extinguish the fire. In reality, the fire truck does move, but it doesn't actually spray water. Instead, it plays a sound effect of splashing water. The fire truck also has emergency lights and siren sounds.

In this chapter, you will build the fire truck, a cat and a bird in need of rescue from a fire, some townspeople and some barriers. You will get experience with more complex programs, using many light and sound effects.

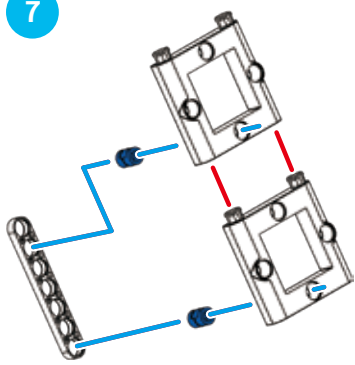
### BUILD

### FIRE TRUCK

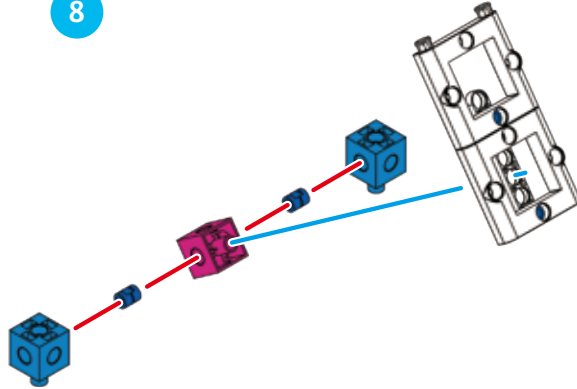


CONTINUED ...

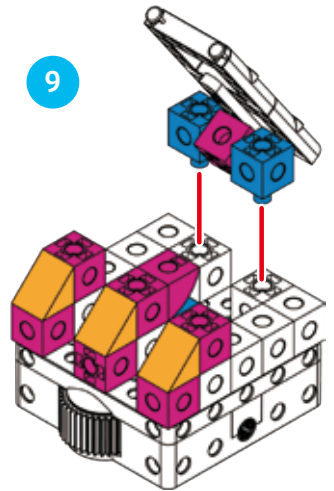
7



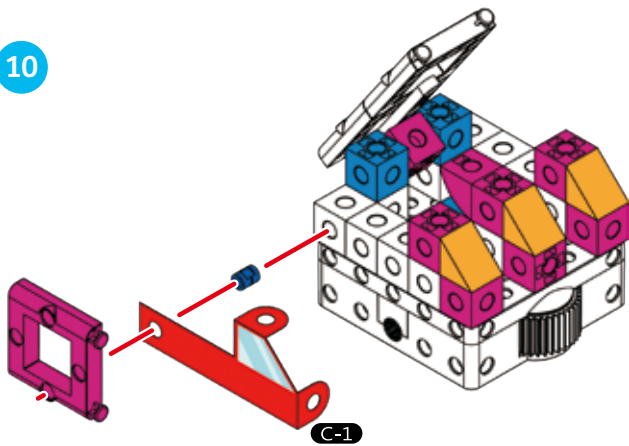
8



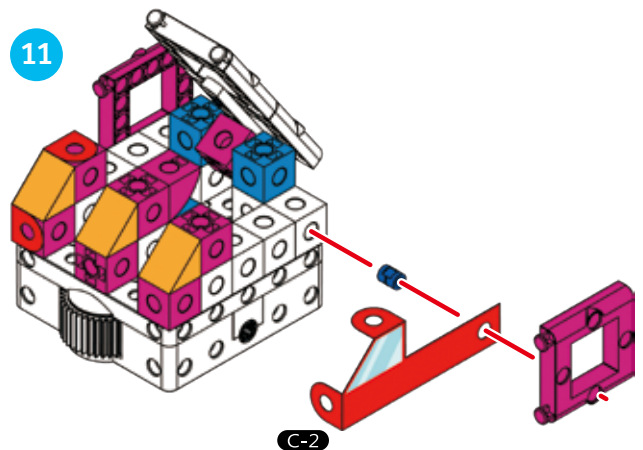
9



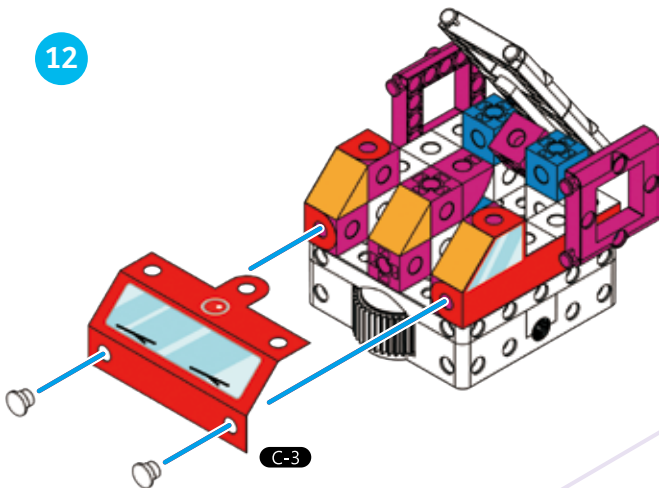
10



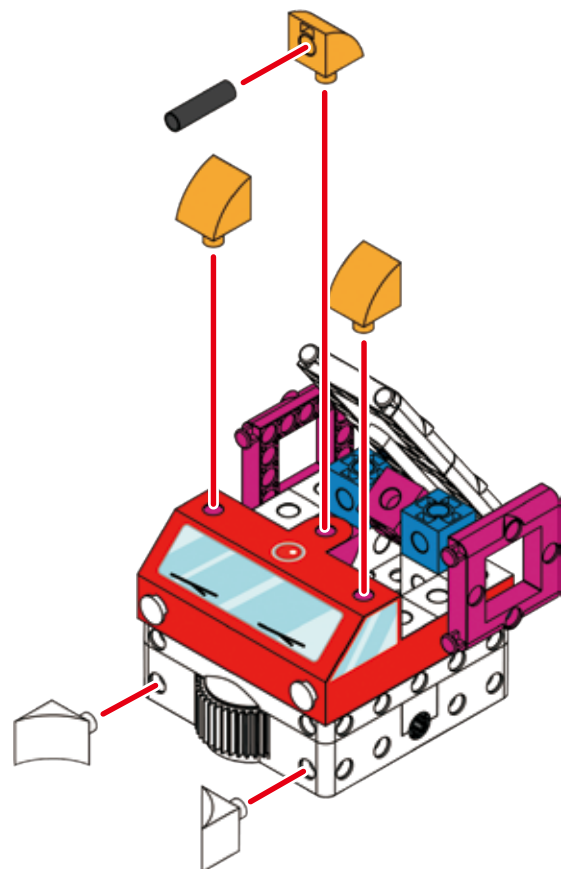
11



12

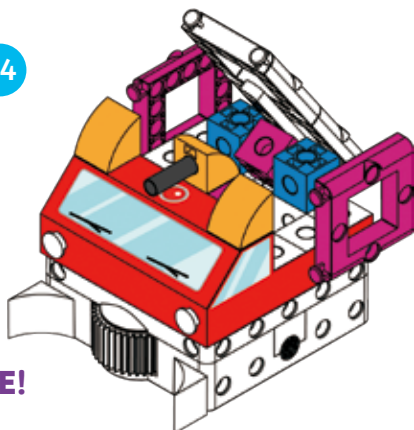


13



14

**DONE!**



**BUILD**

**CAT IN NEED OF RESCUE**

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

6-hole cube

6-hole cube

Now attach the cat to this base map card with plastic straps.

**BUILD**

**PEOPLE**

1

2

**DONE!**

**1**

2

**DONE!**

**BUILD**

**BARRIERS**

1

x2

2

**DONE!**

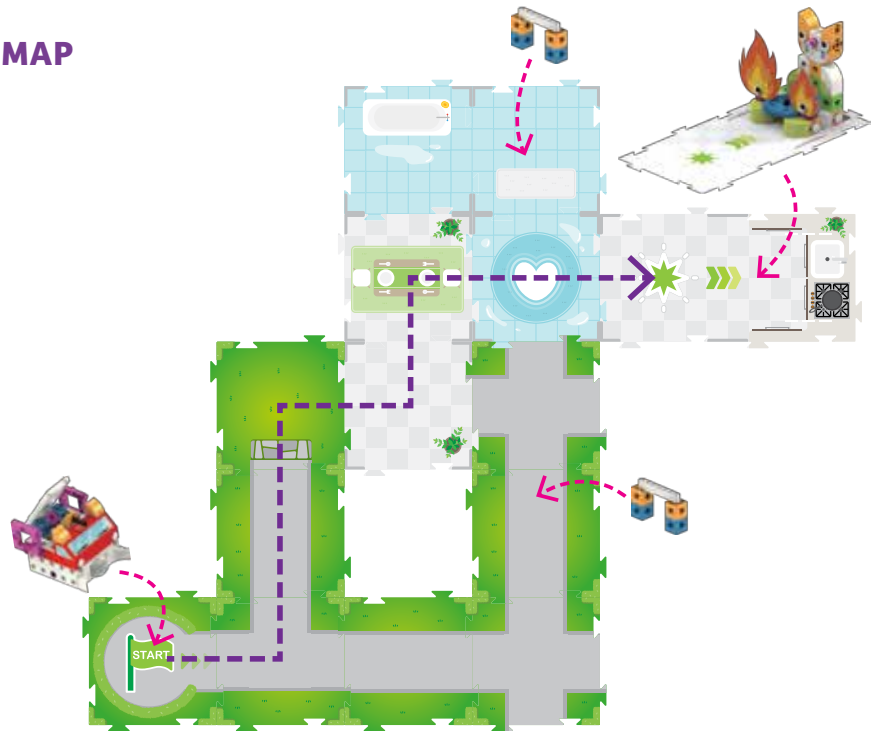
**STORY**

The fire truck must move from the start to the kitchen, avoiding the barriers, and put out the fire.

**WHAT'S HAPPENING**

The main program moves the fire truck to the base map card with the green star. The Green Function activates the output gear to "put out the fire" and plays the fire hose sound. The conditional statement tells the robot to flash red and blue lights until the robot scans the Event 4 card. Then, if the robot scans Event 4, it plays the siren sound and flashes its emergency lights in red and blue.

**MAP**



**CODE**



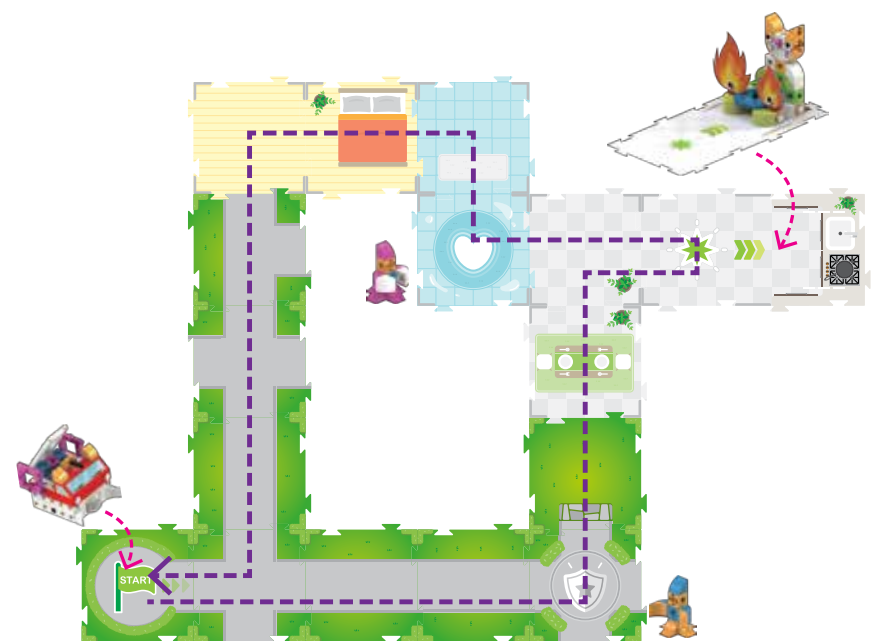
**STORY**

This time, the fire truck must return to the start after rescuing the cat.

**WHAT'S HAPPENING**

The main program moves the fire truck to the base map card. The Green Function works the same as in the previous lesson. This time, the conditional statement tells the robot to play a "Huh?" sound, to ask the townspeople where the fire is, if the robot scans the Event 1 or Event 4 cards. Otherwise, it plays a siren sound and flashes its lights.

**MAP**

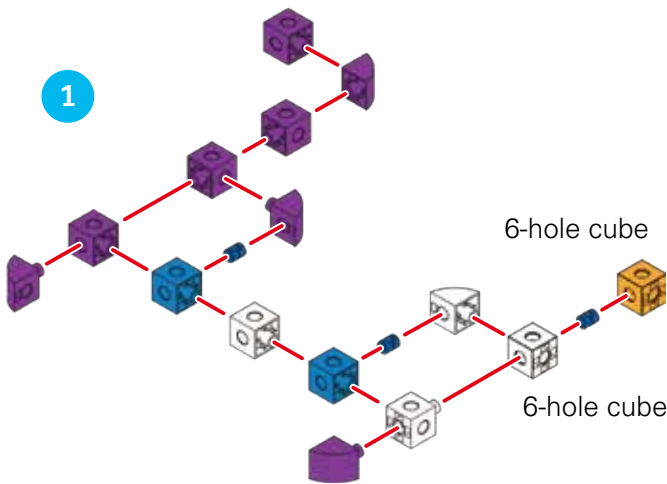


CODE

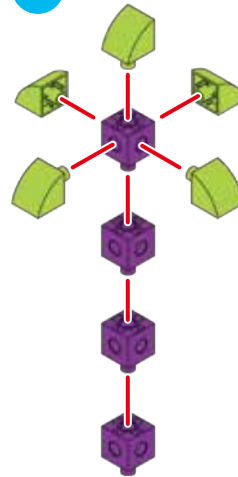
BUILD

BIRD IN NEED OF RESCUE

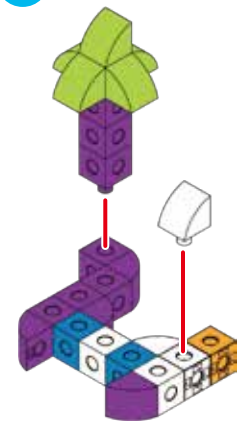
1



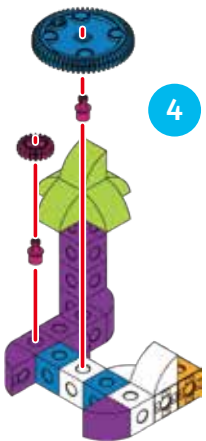
2



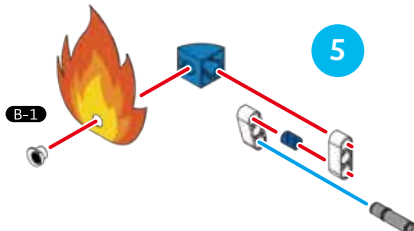
3



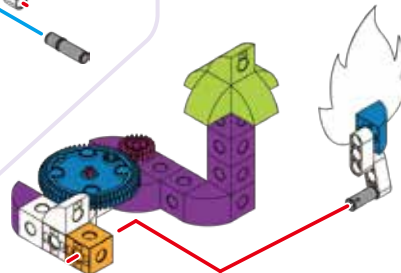
4



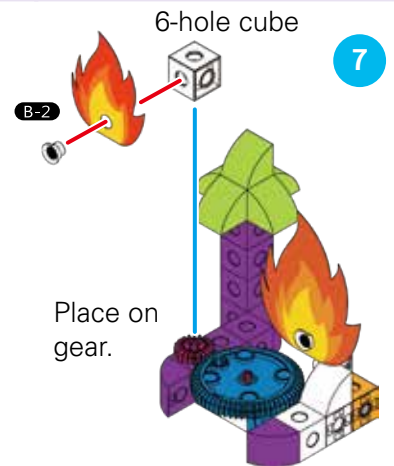
5



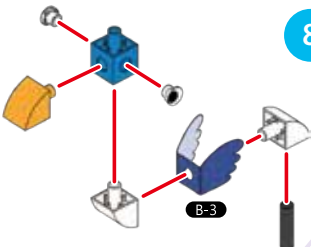
6



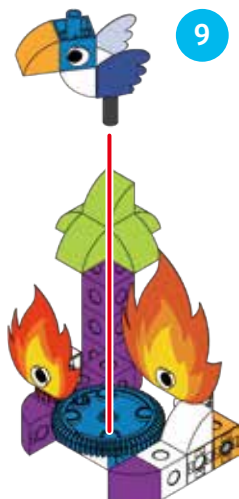
7



8

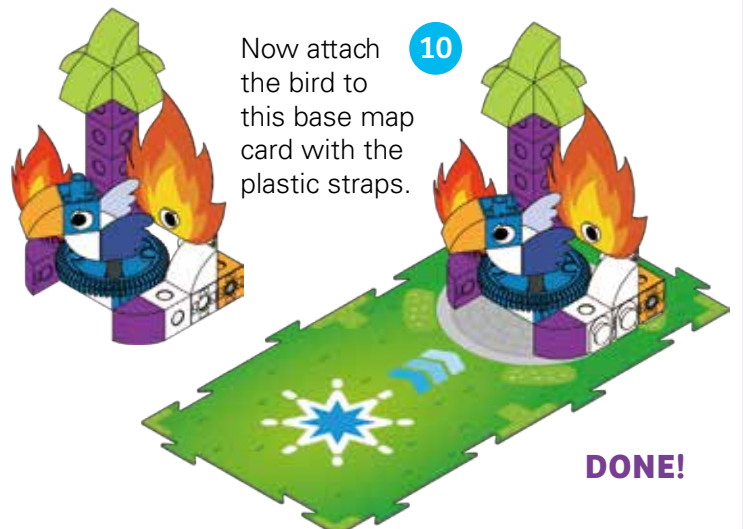


9

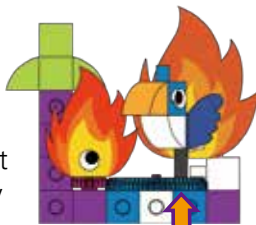


10

Now attach the bird to this base map card with the plastic straps.



Front view





STORY

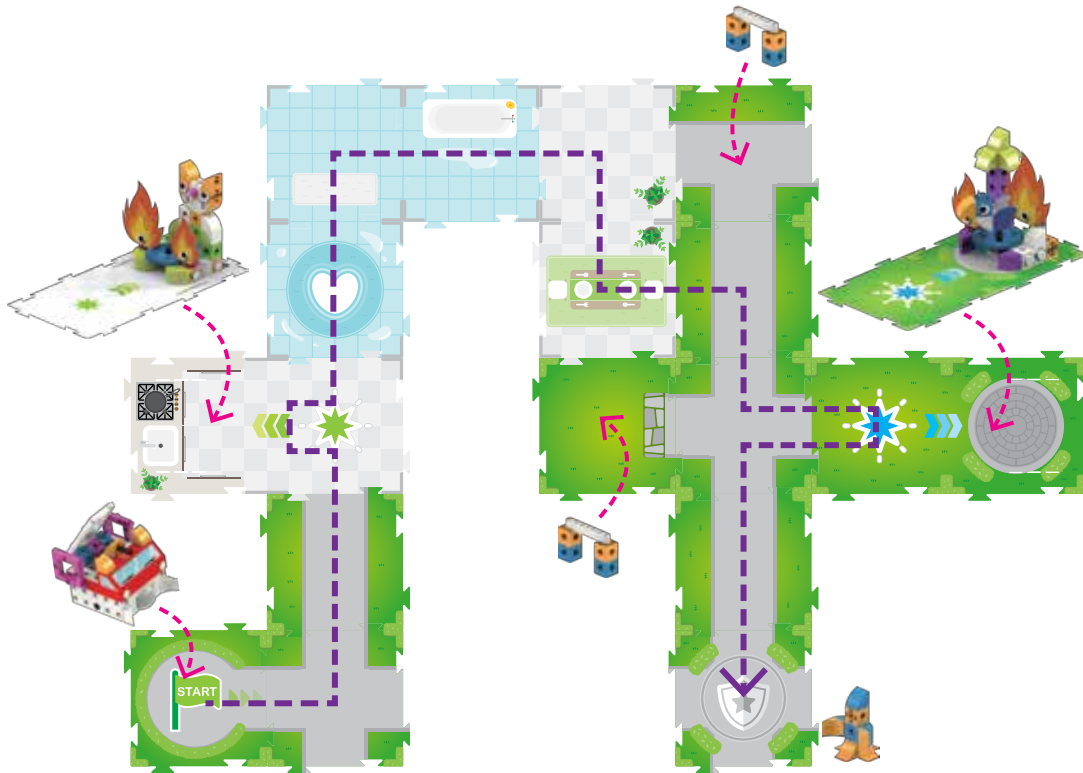
This time, the fire truck must first rescue the cat and then rescue the bird, avoiding barriers along the way. After the robot puts out each fire, it plays a cheering sound.

WHAT'S HAPPENING

The main program moves the fire truck to the base map cards with the green star, then the blue star, and finally the Event 1 map card. The Green and Blue Functions work similarly to how

the Green Function works in the previous lessons. The conditional statement plays the cheering sound if the robot scans the Event 4 and Event 1 map cards.

MAP



CODE

```

[START] → ↻ → → ↻ ↻ → → ↻ → → ↻ → → ...
... ↻ → ↻ → ↻ → ↻ → ↻ → ↻ → ↻ → ↻ → ↻ → [Red]

[Green Star] [Blue Star] 4 [Blue Star] 4 [Cheering Sound] [Red]

[Green Star] [Blue Star] 6 [Cheering Sound] [Red]

IF [Heart] AND [Shield] DO [Green Star] [Blue Star] ELSE [Fire Truck] [Cheering Sound] [Lightbulb] ...
... [Fire Truck] [Cheering Sound] [Lightbulb] [Red]

```

● ● ● LESSON 24

FIRE RESCUE OBSTACLE COURSE

STORY

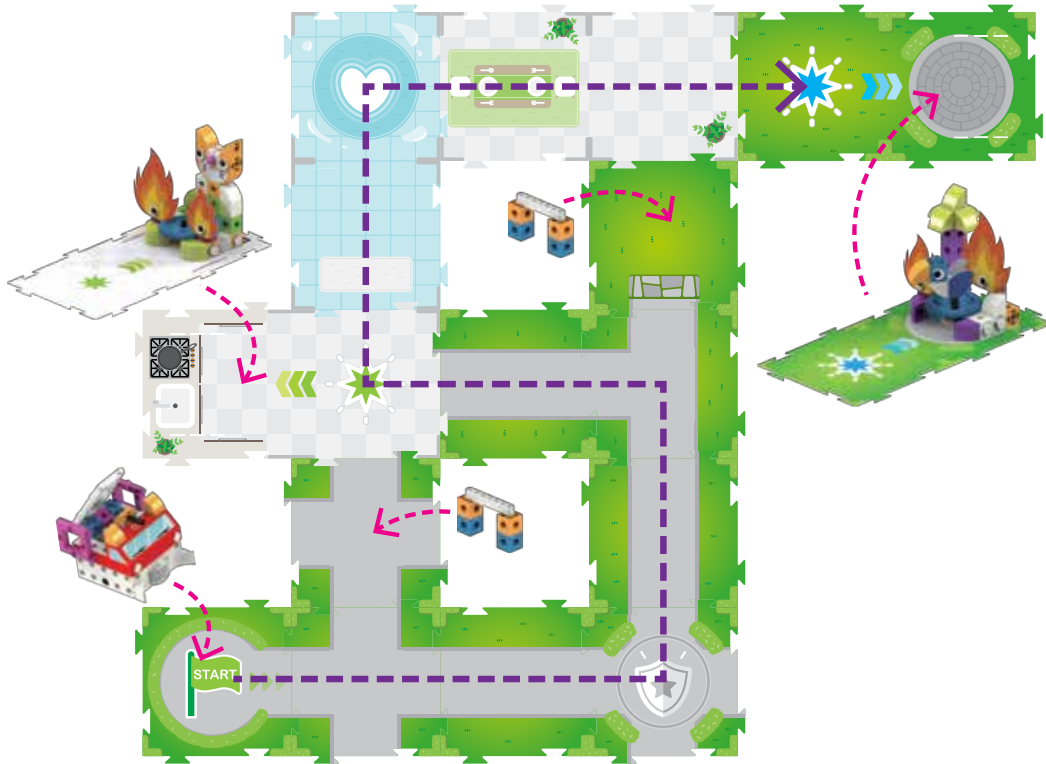
There are barriers all over town. The fire truck must first rescue the cat and then rescue the bird, avoiding the barriers along the way. After the robot puts out each fire, it plays a cheering sound.

WHAT'S HAPPENING

The main program moves the fire truck to the base map cards with the green star and then the blue star. The Green and Blue Functions work the same as in the previous lesson. The conditional

statement plays the siren sound and flashes the emergency lights in red and blue if the robot scans the Event 1 or Event 4 map cards. Otherwise, it plays a red-blue light effect.

MAP

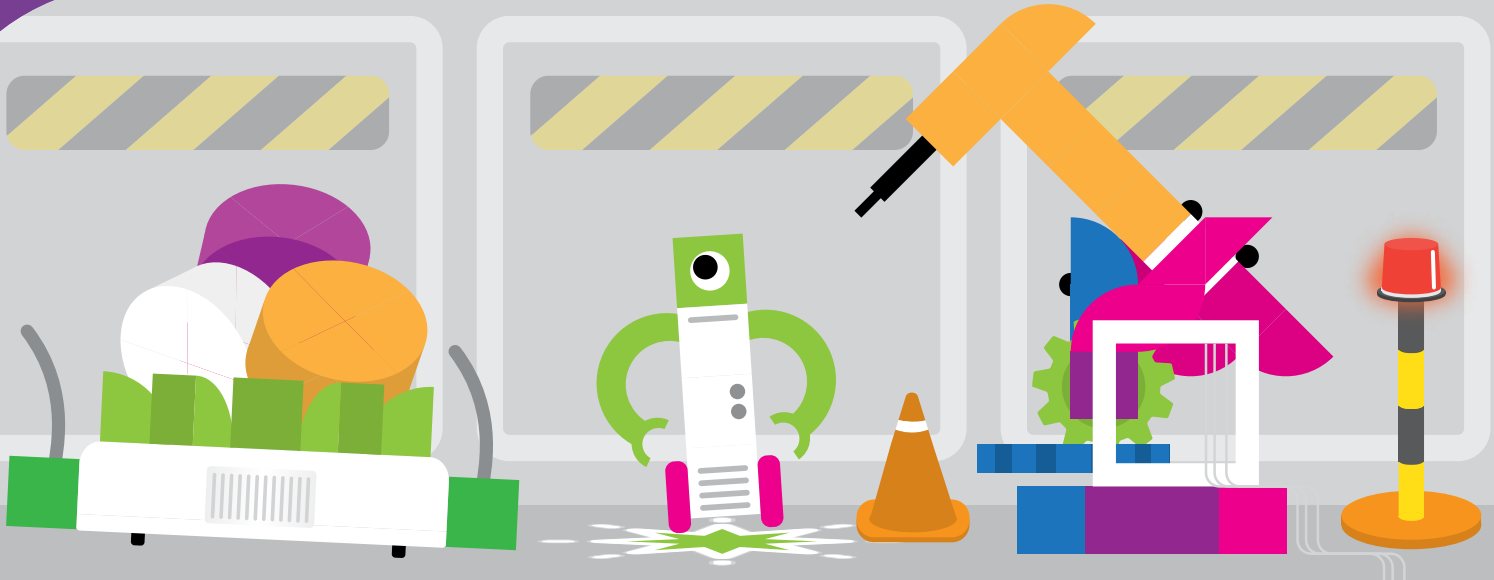


CODE

```

[START] → 3 [Turn Right] → → → [Turn Right] → → [Turn Left] → → → ...
... → 3 [Red]
[Green Star] [Blue Star] 4 [Blue Star] 4 [Cheer] [Red]
[Blue Star] [Blue Star] 6 [Cheer] [Red]
[IF] [Shield] [OR] [Heart] [DO] [Siren] [Light] [Light] [Siren] [Light] ...
... [ELSE] [Light] [Light] [Light] [Light] [Red]
    
```



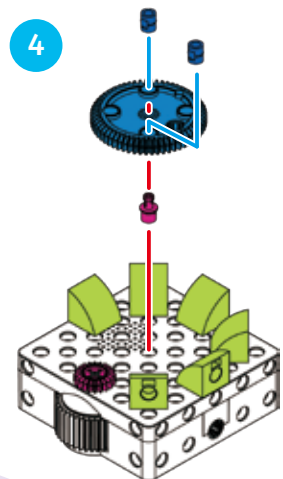
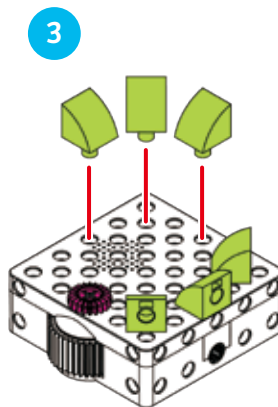
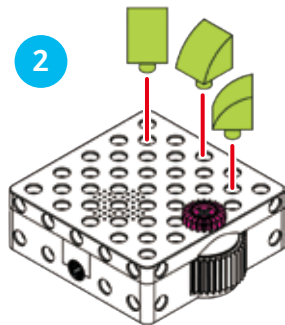
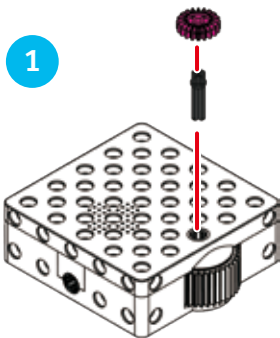


## Chapter 6: Robotic Factory Floor

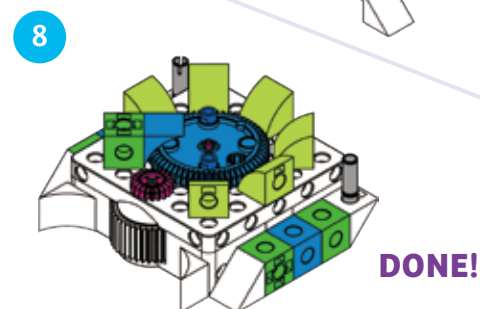
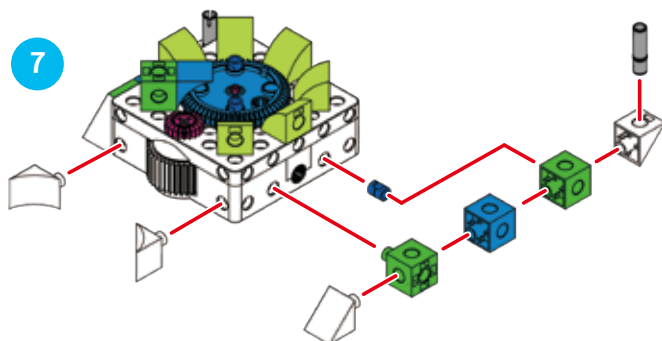
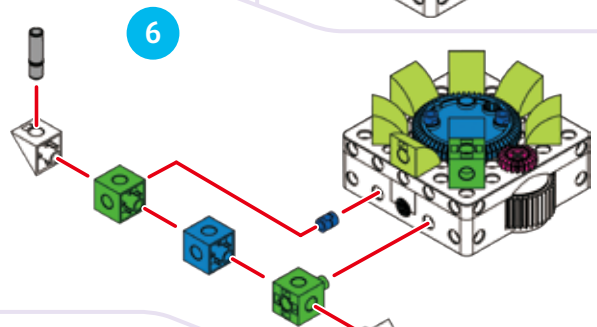
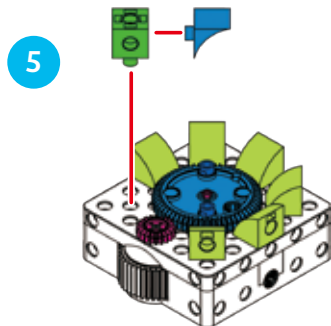
Let's put everything we learned about coding and robotics to the test with some complex mechanical models. In this chapter, you will build a factory robot that moves products around the factory for processing.

### BUILD

### FACTORY BOT

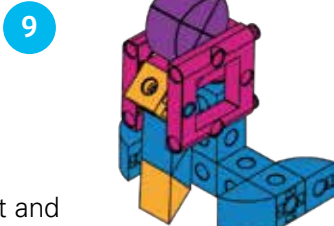
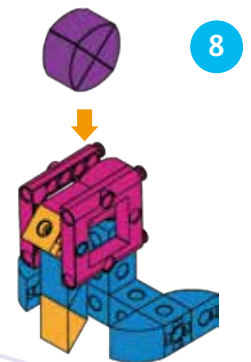
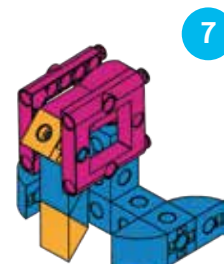
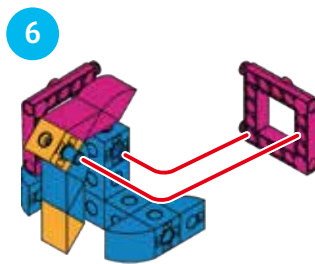
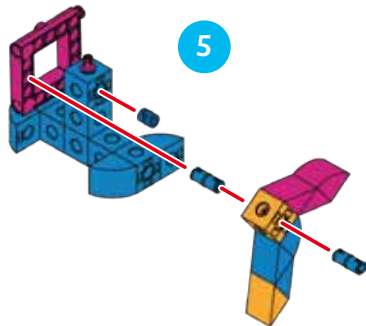
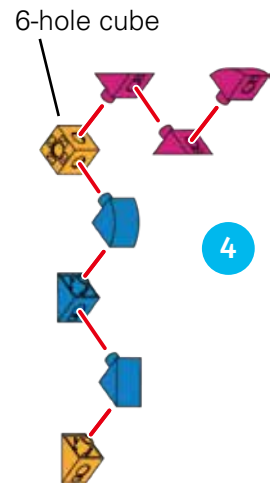
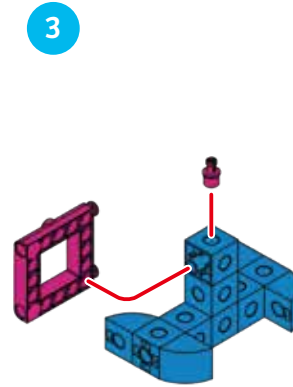
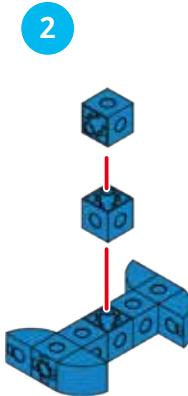
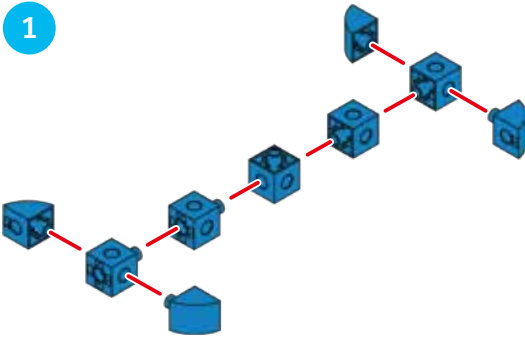


Top view

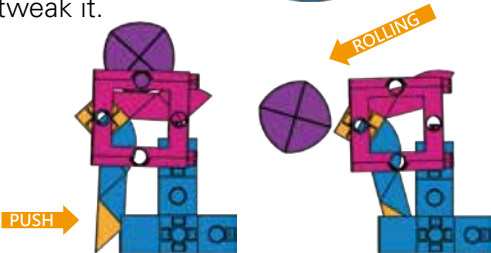


**BUILD**

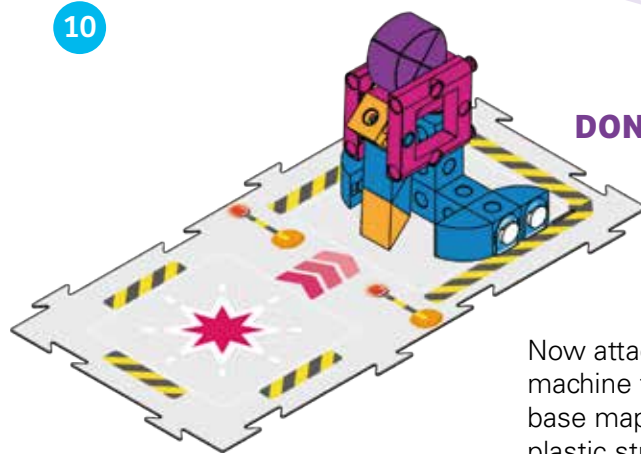
**FACTORY LOADER MACHINE**



Test it and tweak it.



10

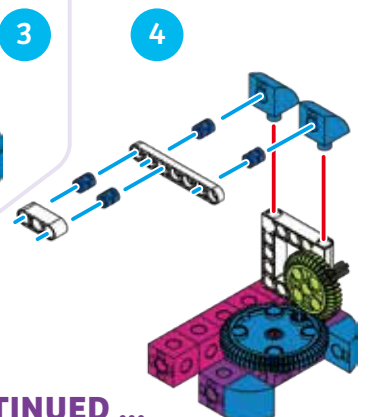
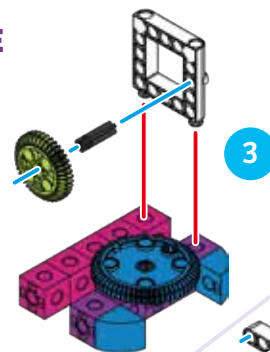
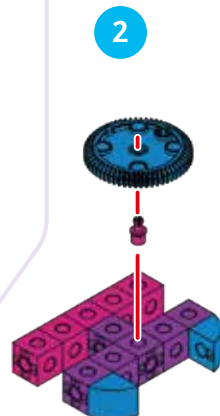
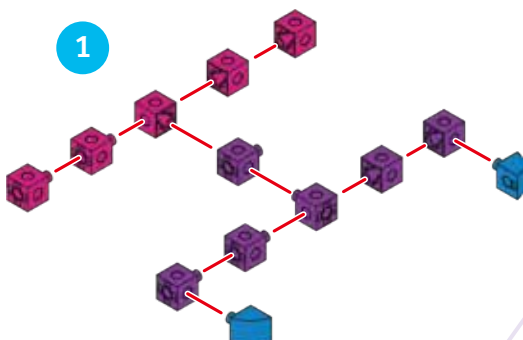


**DONE!**

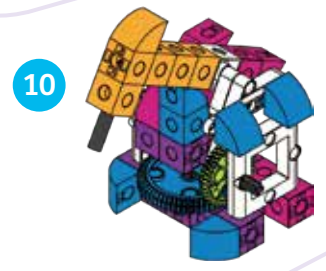
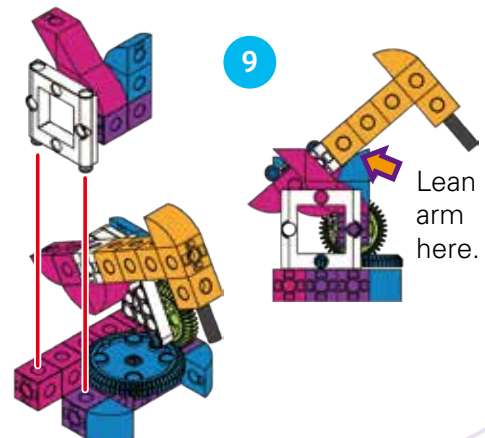
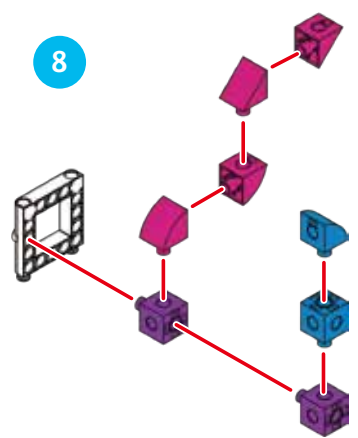
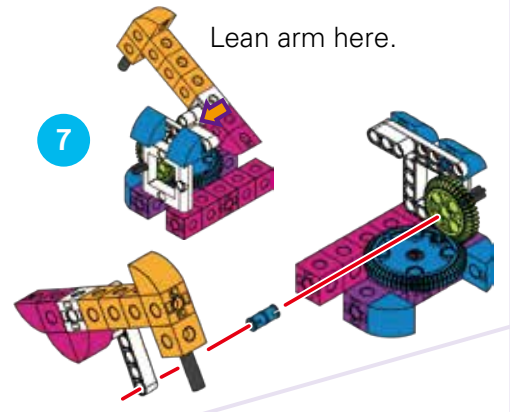
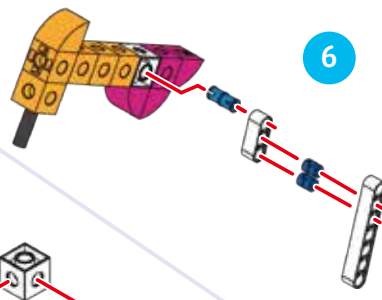
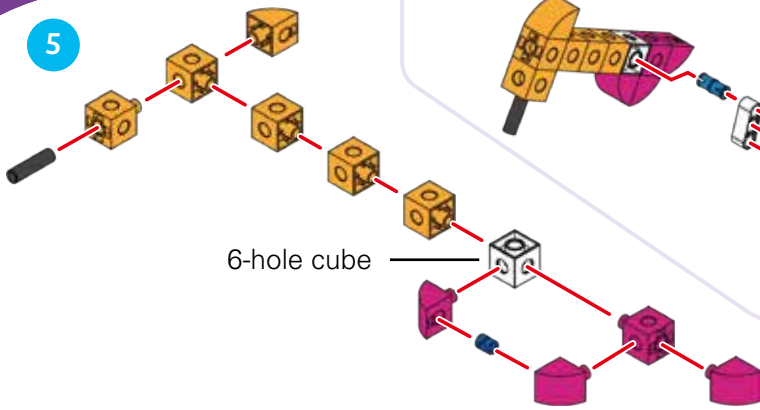
Now attach the machine to this base map card with plastic straps.

**BUILD**

**FACTORY SCREWDRIVER MACHINE**



**CONTINUED ...**

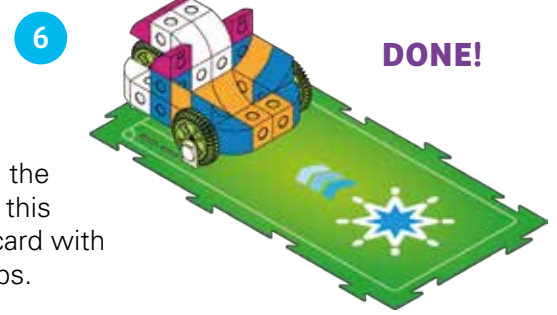
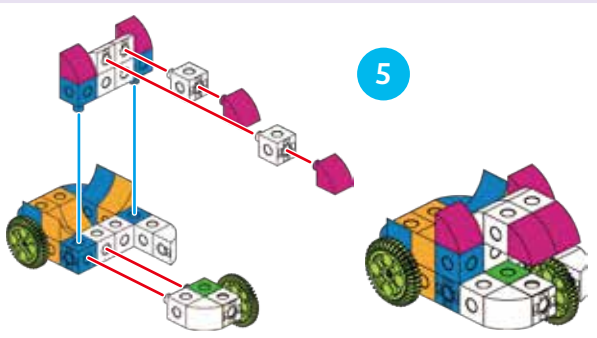
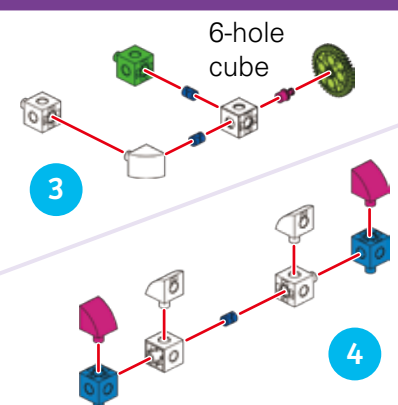
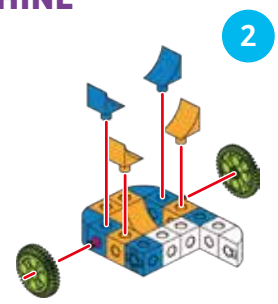
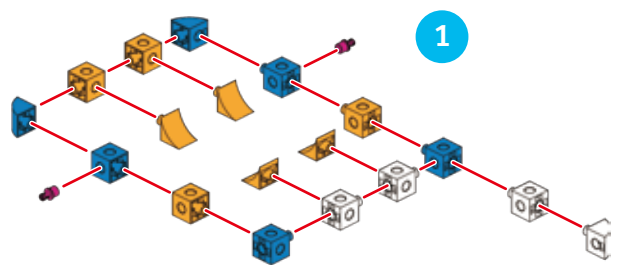


**DONE!**

Now attach the machine to this base map card with plastic straps.



**BUILD FACTORY DELIVERY MACHINE**

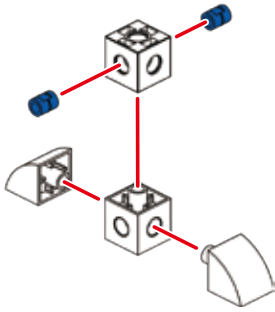


Now attach the machine to this base map card with plastic straps.

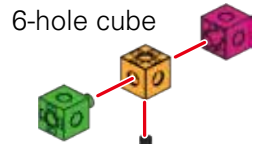
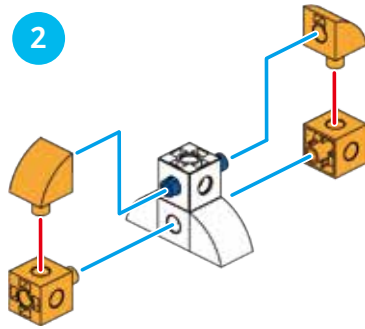
**BUILD**

**FACTORY CONTROL LIGHTS**

1 6-hole cube

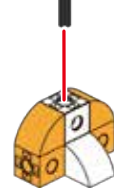


2

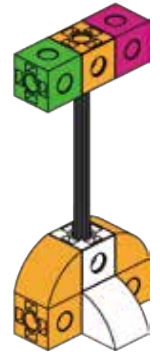


3

100-mm axle



4

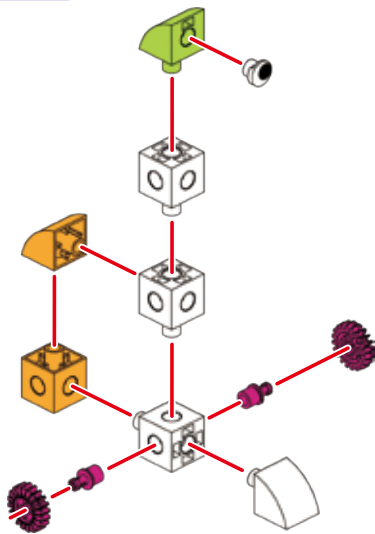


**DONE!**

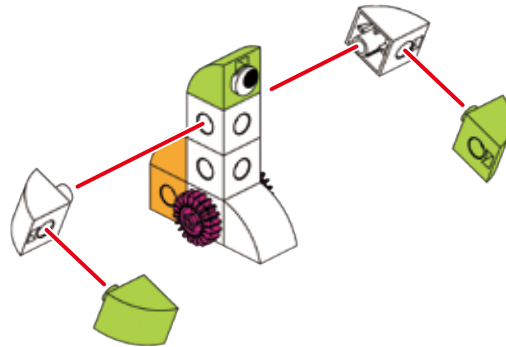
**BUILD**

**FACTORY HELPER BOTS**

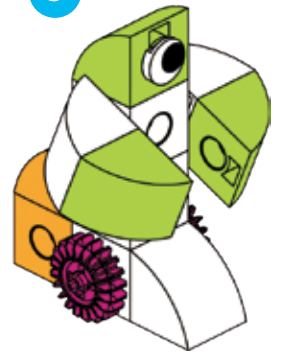
1



2

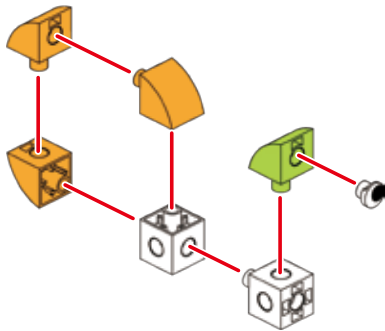


3

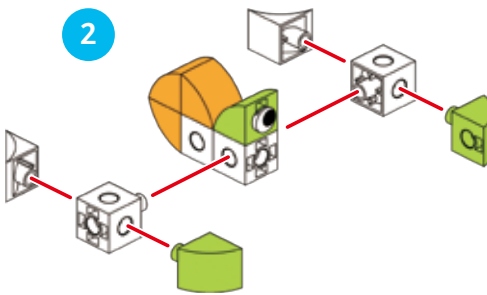


**DONE!**

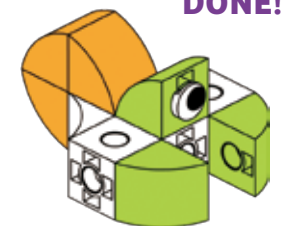
1



2



3



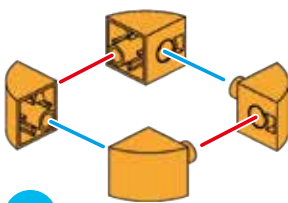
**DONE!**

**BUILD**

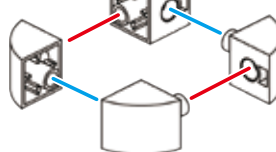
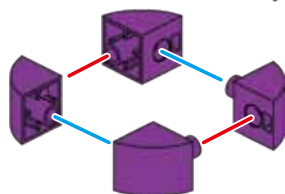
**FACTORY PRODUCTS**

3

x2



2



**DONE!**

PICK AND PACK

STORY

The robot must move to the loader machine to pick up a product, and then bring it to the helper bot at the green star and release it.

WHAT'S HAPPENING

The main program moves the robot around the factory floor, first to the red star and then to the green star. The Red Function causes the robot to move forward and trigger the loader machine to drop a product into the robot's basket. The Green Function causes the robot to rotate its output gear and release the product from its basket.

MAP



CODE



PICK, PROCESS, AND INSPECT

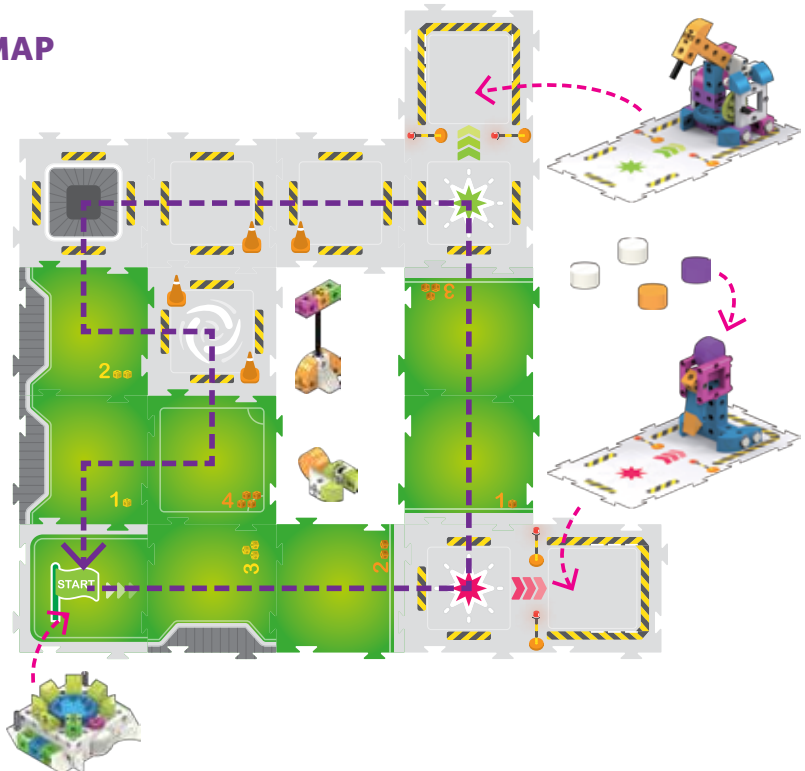
STORY

Now, the robot must move to the loader machine to pick up a product, then move to the screwdriver machine for processing, then have the product inspected, and go back to the start.

WHAT'S HAPPENING

The main program moves the robot around the factory floor, from station to station. The Red Function activates the loader and the Green Function activates the screwdriver and plays factory sounds. If the robot scans the Event 3 card, then a factory inspection program is performed.

MAP



CODE

```

[START] [Loop 3] [Move Right] [Move Right] [Move Right] [Turn Right] [Loop 3] [Move Right] [Turn Right] ...
... [Loop 2] [Move Right] [Turn Right] [Loop 2] [Move Right] [Turn Right] [Move Right] [Stop]
[Starburst] [Rabbit] [Lightbulb] [2] [Stop] [Starburst] [Gear] [3] [Screwdriver] [Lightbulb] [Stop]
[IF] [AND] [DO] [Lightbulb] [Lightbulb] [Screwdriver] [ELSE] [Rabbit] [Lightbulb] [Stop]
    
```

LESSON 27

PICK, PROCESS, INSPECT, RE-PROCESS, AND INSPECT

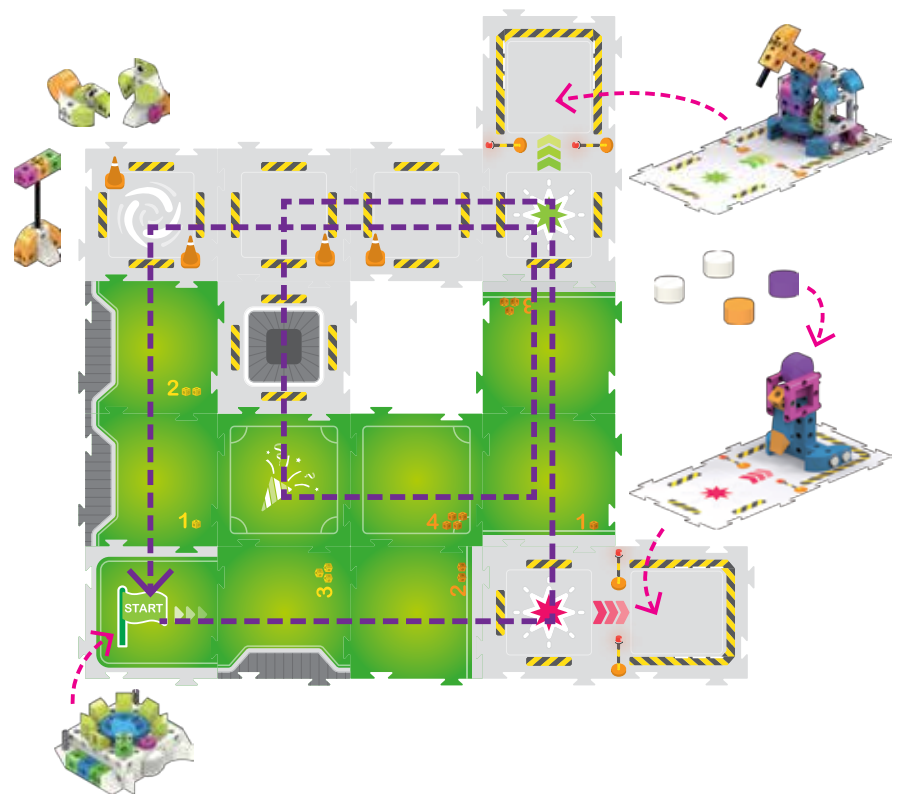
STORY

This time, the robot must loop back around to perform a second processing operation at the screwdriver machine before the product can pass inspection.

WHAT'S HAPPENING

The main program moves the robot around the factory floor two times, first to the red star and then to the green star. The Red and Green Functions work similar to how they work in the previous lesson. The conditional statement says that if the robot scans both the Event 2 and Event 3 cards, then the robot will play a cheering sound and light up with a light effect. The conditional uses an And operator, meaning it must scan both events before the condition is true.

MAP



CODE

```

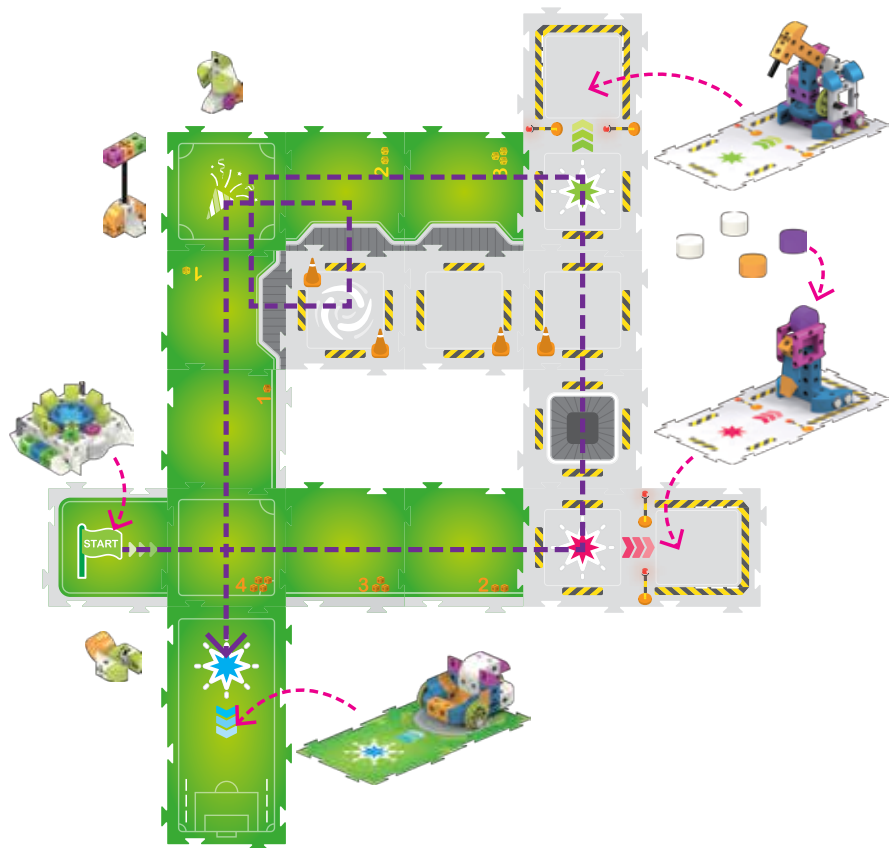
[START] [Loop 2] [Move Right] [Move Right] [Move Right] [Turn Right] [Loop 2] [Loop 2] [4] [Move Right] [Move Right] [Turn Right] ...
... [Loop 2] [Move Right] [Move Right] [Move Right] [Turn Right] [Move Right] [Move Right] [Move Right] [Stop]
[Starburst] [Rabbit] [Lightbulb] [2] [Stop] [Starburst] [Gear] [4] [Screwdriver] [Lightbulb] [Lightbulb] [3] [Stop]
[IF] [AND] [DO] [Lightbulb] [Lightbulb] [Screwdriver] [ELSE] [Rabbit] [Lightbulb] [Stop]
    
```

**STORY**

Finally, the robot must perform an entire production process, from picking up the product all the way to delivering it to the delivery machine.

**WHAT'S HAPPENING**

The main program moves the robot around the factory floor, first to the red star, then to the green star, then through an inspection process, and finally to the blue star where the delivery machine is waiting. The Red and Green Functions work somewhat similar to how they work in the previous lesson. The conditional statement says that if the robot scans both the Event 2 and Event 3 cards, then the robot will play an "Ahh" sound and give the product a green light. The Blue Function is added to rotate the product at the delivery machine, play a cheering sound, and display a light effect.

**MAP**

**CODE**

```

START → → → → → ↻ → → → ↻ → → ...
  
```

```

... ↻ 5 → ↻ → 4 ← ↻ ↻
  
```

```

★ 📺 💡 2
  
```

```

★ 🌀 5 🤖 🕊️ 💡
  
```

```

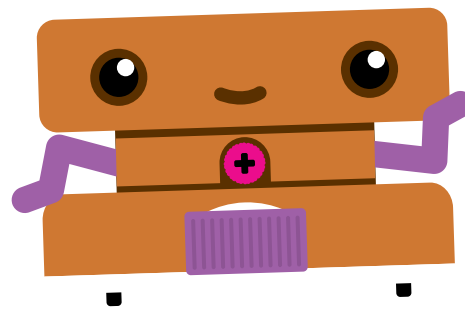
★ 🌀 8 🎮 🌍 3
  
```

```

IF ★ AND 🌀 DO 🎮 💡 4 ELSE 🕊️ 💡
  
```

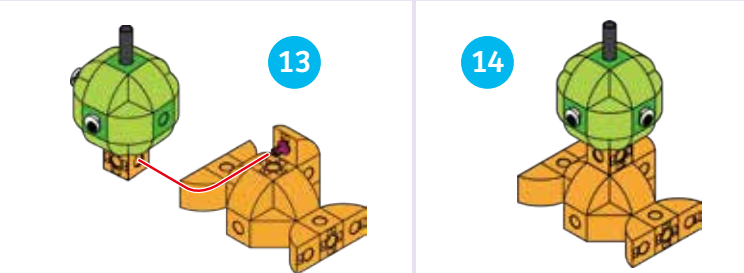
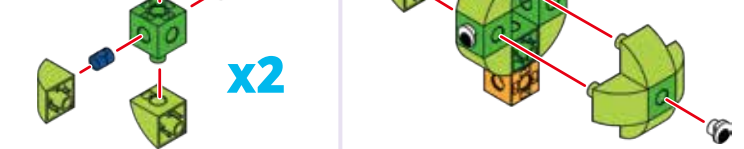
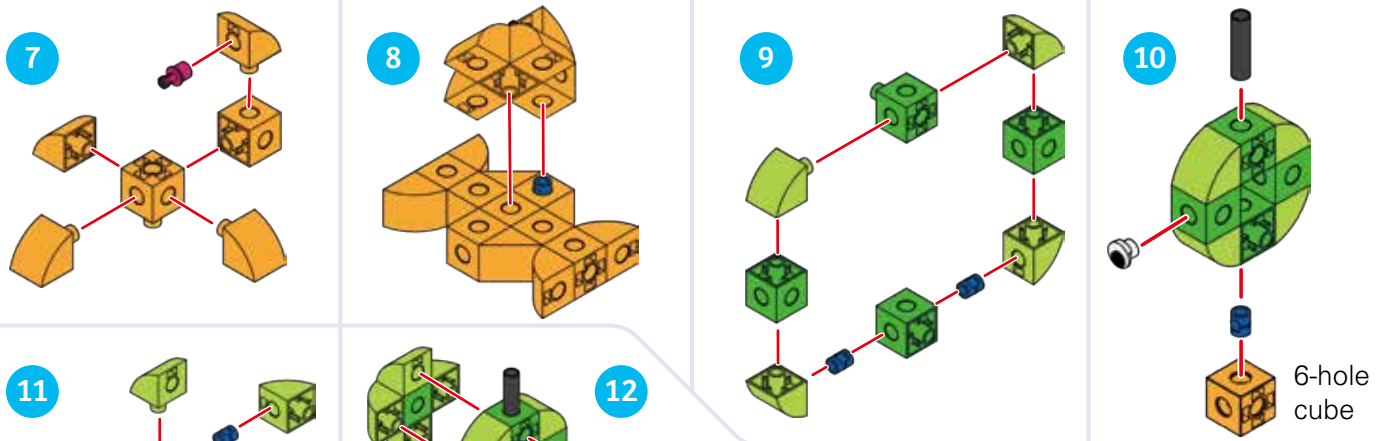
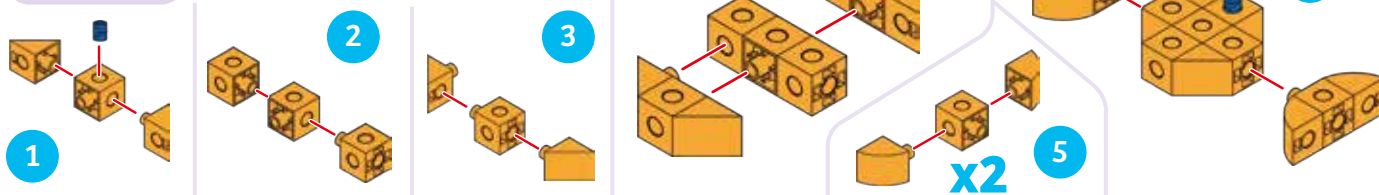
# Chapter 7: Sammy's Ultimate Adventure

In this final chapter, we will use everything we've learned about coding and robotics in two final lessons with Sammy. Now, you have learned enough to activate Sammy's arms using functions, so it can interact with other models. You will need to reassemble Sammy following the instructions on page 9.



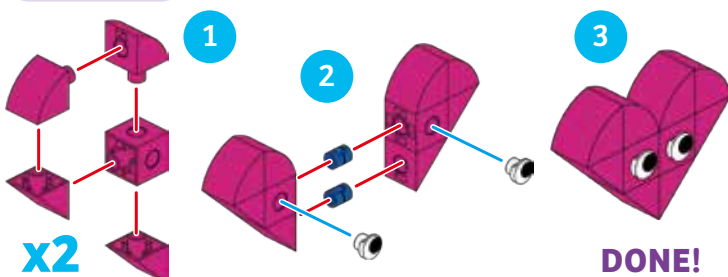
### BUILD

### GRANNY SMITH



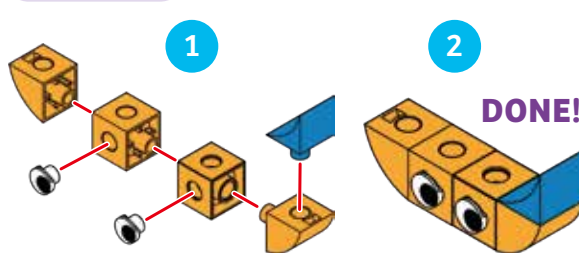
### BUILD

### JIGGLY



### BUILD

### NANA





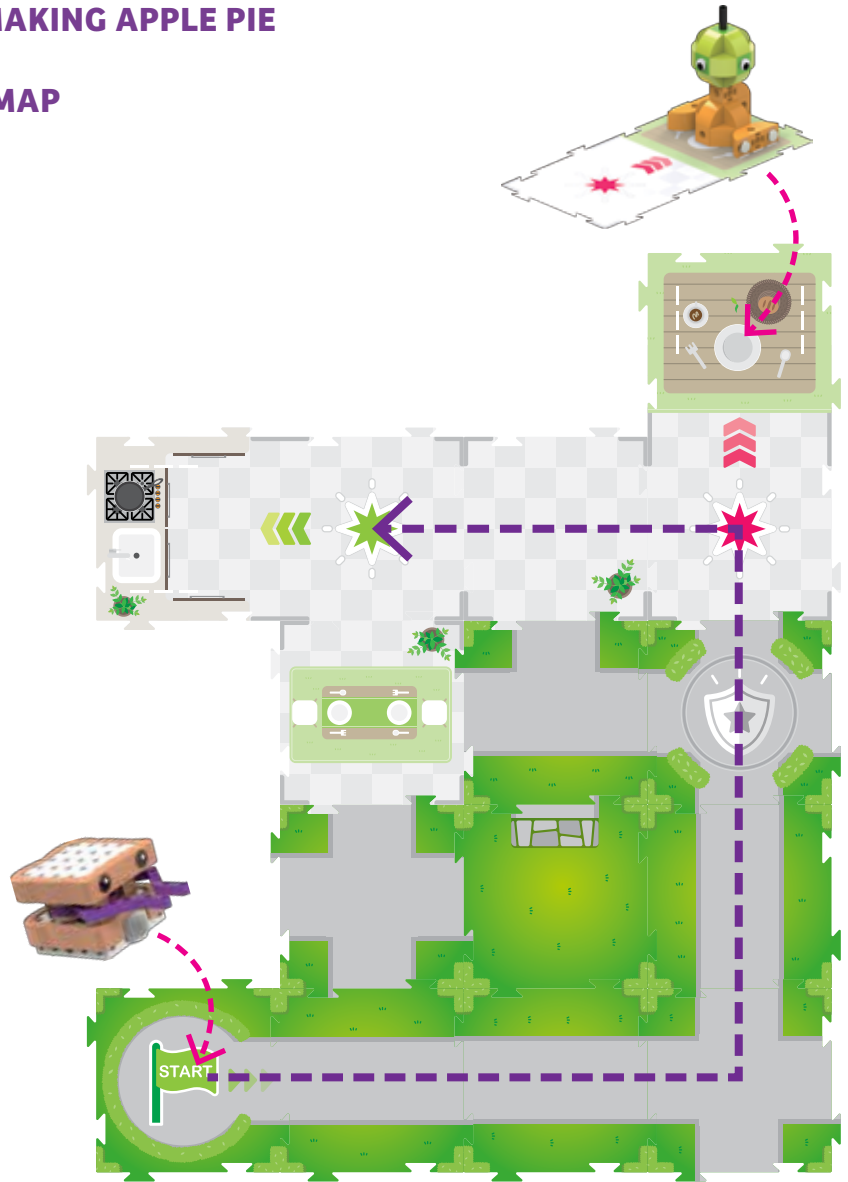
**STORY**

Sammy wants to make an apple pie. Sammy drives through town, opens up its arms, picks up an apple from the table, carries it to the kitchen, and releases it in the kitchen. Start Sammy with its arms together, and place the apple on the pedestal.

**WHAT'S HAPPENING**

The main program moves Sammy through the streets, to the event card, then to the red star, and finally to the green star. The conditional statement says that if Sammy scans the Event 1 card, then Sammy should open up its arms and say "Hi!" The Red Function makes Sammy's arms close to pick up the apple and play an "Ahh" sound. The Green Function causes Sammy to release the apple in the kitchen and play a "Huh?" sound. This may take multiple tries before you get it to work correctly. That is normal in coding and robotics. Keep trying until you get it to work!

**MAP**



**CODE**

```

[START] → → → ↻ → → → ↻ → → [STOP]
[IF] [Shield] [DO] [Hi!] [Turn 90°] [3] [STOP]
[Star] [Turn 90°] [4] [Kitchen] [STOP]
[Star] [Turn 90°] [2] [Kitchen] [STOP]
    
```

● ● ● LESSON 30

SAMMY'S ULTIMATE ADVENTURE

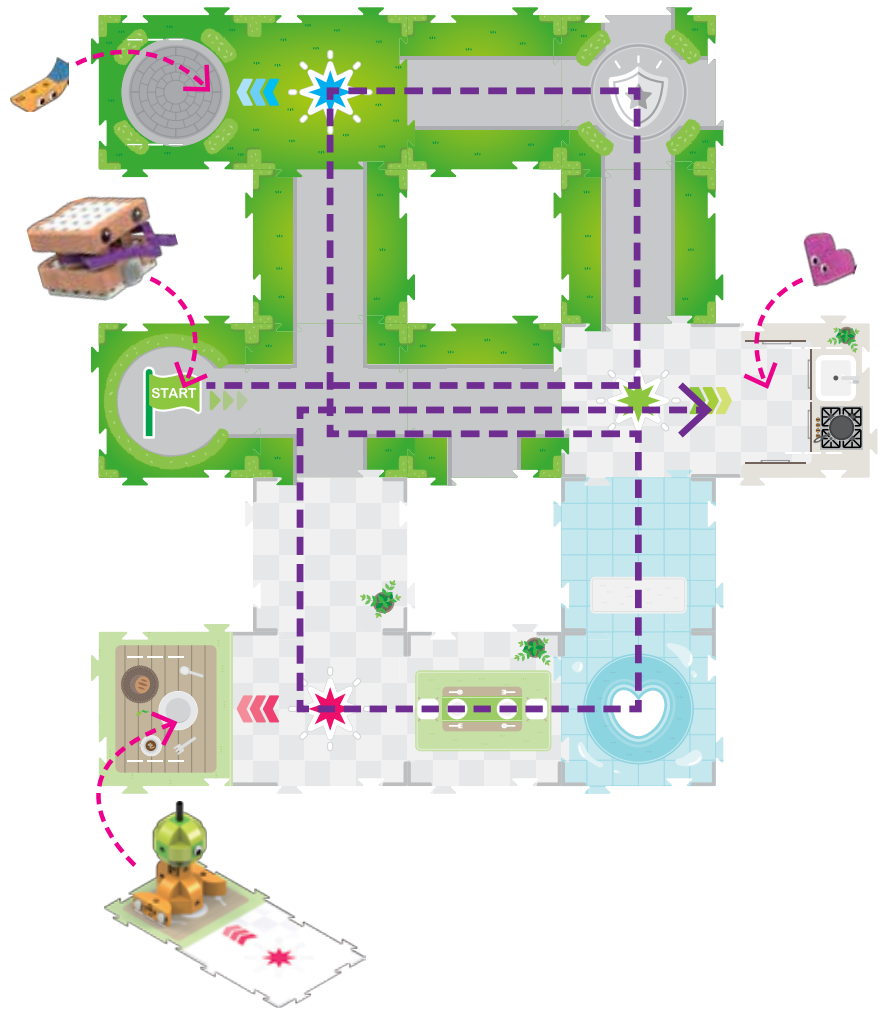
STORY

In this final lesson, try to use as many of the types of code cards as possible. Sammy is looking for Granny Smith the Apple to bring to visit Jiggly the Gummy Heart. Sammy looks in a few places first, finding nothing on the street or in the pool, and finding Nana the Banana in the park. Finally, Sammy finds Granny Smith and takes it to Jiggly.

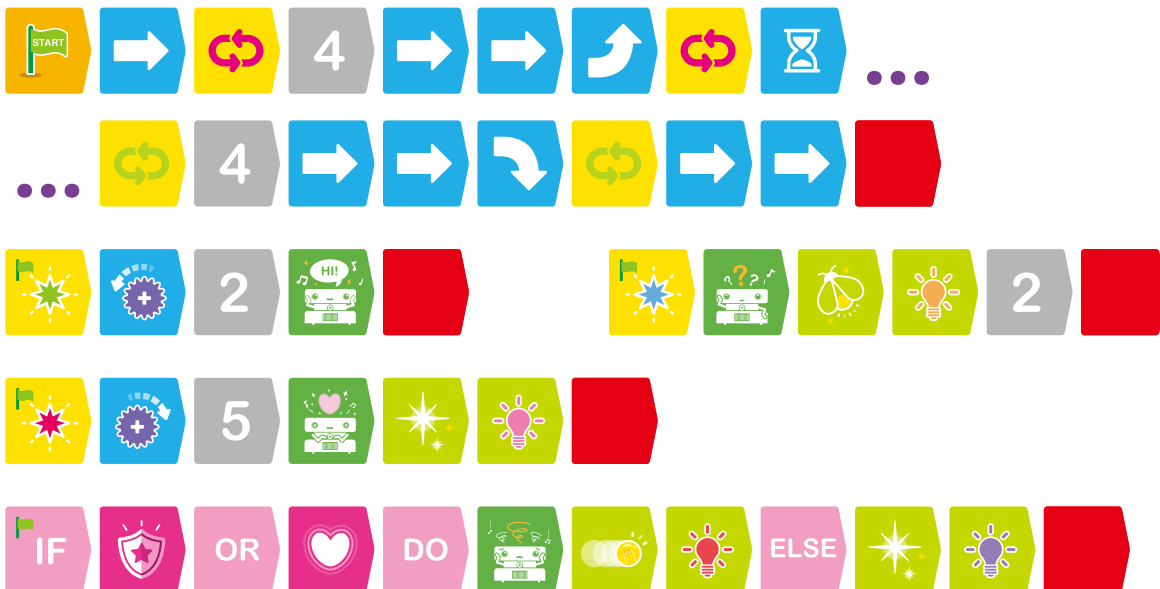
WHAT'S HAPPENING

The main program moves the robot around the map. The Green Function makes Sammy say "Hi!" and opens its arms the first time. The conditional statement causes Sammy to say "Aargh" and display a red light when it scans the Event 1 or Event 4 cards. The Blue Function causes Sammy to say "Huh?" to Nana and flash orange. The Red Function causes Sammy to pick up Granny Smith. The third time the Green Function is triggered, Sammy opens its arms again, releases Granny, and says "Hi!" again.

MAP



CODE



## MATH LESSONS

### Math Lesson Mode

The robotic base unit can be switched into special modes to teach specific math lessons. In these modes, the robot behaves differently than in its normal operating mode. You switch the robot into these modes by scanning the additional OID control graphics printed on pages 63 and 64.

These are just like code cards, but they are printed into the manual instead of onto separate cards.

In math mode, you program the robot the same way as before, but this time with the goal of solving the stated math problem. In math mode, when the robot reaches the end of its program, it will play music and light up depending on whether the final solution was right or wrong: Harp music and multi-colored lights will play if the solution was correct. Tuba music and red-orange lights will play if the solution was incorrect.

Math mode uses the map cards with the orange and yellow cubes printed on them. These cards represent the numbers 1 through 5 in orange and in yellow.



To complete each math lesson, write a code to solve the stated problem by moving the robot to specific numbered map cards and finally to the blue, red, or green function base map card.

Note: You cannot use function code cards or conditional code cards in math mode. The robot will not react to event cards in math mode.

To exit math mode, press and hold the Erase button for two seconds.

### Algorithms

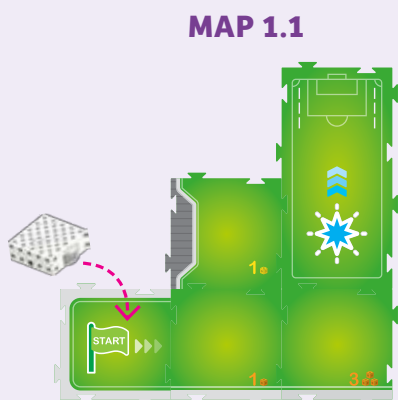
You may hear the word “algorithm” used in connection with computers and programming. An **algorithm** is a part of a computer program that is used to solve a stated problem using a sequence of calculations or steps. An algorithm is a step-by-step way of solving a problem. With these math lessons, you are creating simple algorithms to solve the stated math problems.

## MATH 1

### FIND CUBES OF THE SAME COLOR

#### PROBLEM 1

Scan the “Lesson 1” code graphic on page 63. Program the robot to drive only on map cards with numbers of the same color printed on them, and end up on the blue star.



#### MAP 1.2



#### MAP 1.3



Can you find two different ways to solve the problem for this map?

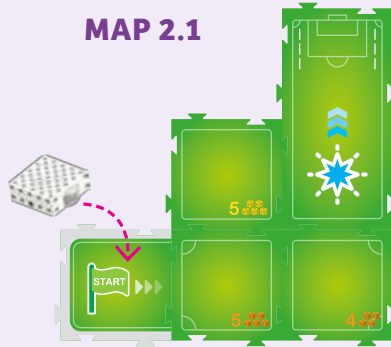
MATH 2

FIND CUBES OF EQUAL VALUE

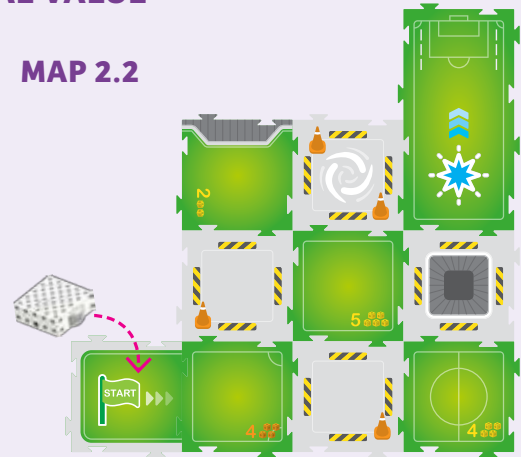
PROBLEM 2

Scan the "Lesson 2" code graphic on page 63. Program the robot to drive only on map cards with numbers of the same value (or amount) printed on them, and end up on the blue star.

MAP 2.1

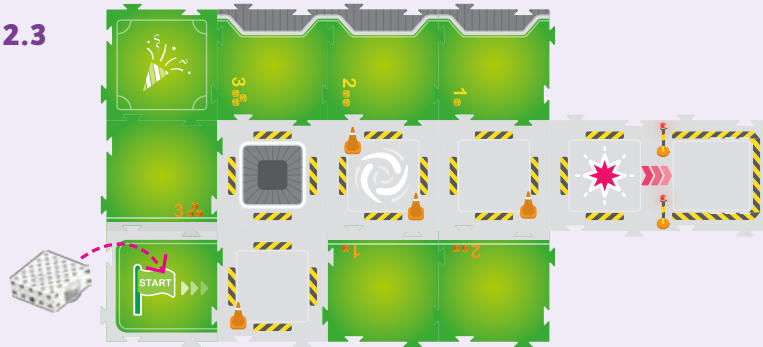


MAP 2.2



MAP 2.3

Can you find three different ways to solve the problem for this map?



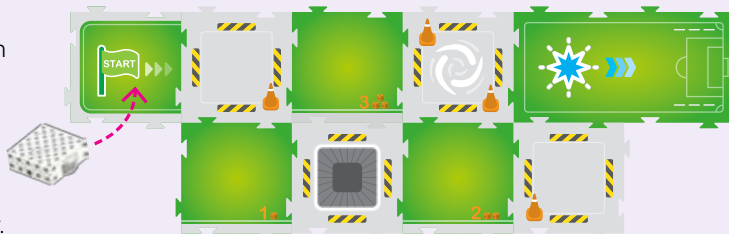
MATH 3

FIND CUBES IN SEQUENCE

PROBLEM 3.1

Scan the "Increasing Value" code graphic on page 63. Program the robot to drive on map cards with numbers of **increasing value** printed on them, ending at the blue star.

MAP 3.1



PROBLEM 3.3

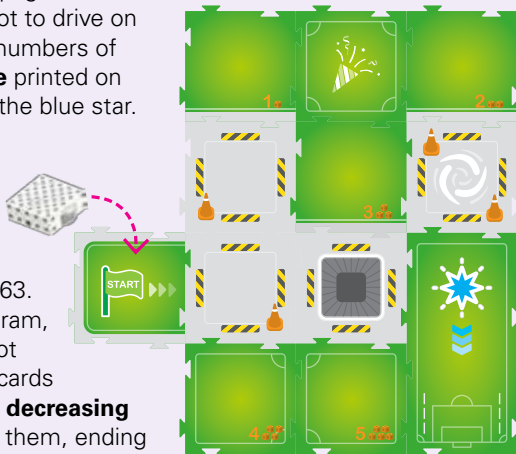
Scan the "Increasing Value" code graphic on page 63. Program the robot to drive on map cards with numbers of **increasing value** printed on them, ending at the blue star. Can you find two different ways to solve the problem for this map?

PROBLEM 3.2

Scan the "Increasing Value" code graphic on page 63. Program the robot to drive on map cards with numbers of **increasing value** printed on them, ending at the blue star.

MAP 3.2

Now, scan the "Decreasing Value" code graphic on page 63. In a second program, program the robot to drive on map cards with numbers of **decreasing value** printed on them, ending at the blue star.

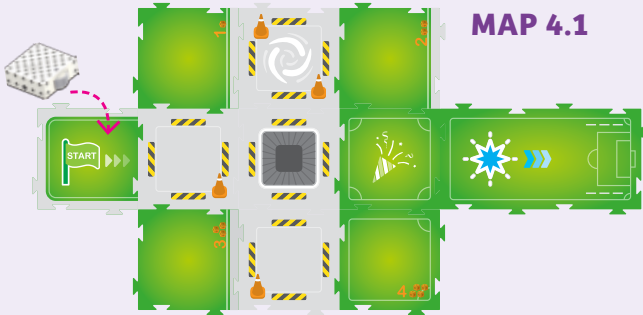


MAP 3.3



PROBLEM 4

Scan a number graphic on page 64. Write a program to drive the robot on map cards with numbers printed on them that **add up to** the number you scanned, ending at the blue star.



MAP 4.2



Variables

In this lesson, you are working with variables. A **variable** is a quantity in a calculation or program that is assumed to vary, or change, or be capable of varying in value. Each time you set the number with the code graphics on page 64, you are setting a new variable for the program.

MAP 4.3



MATH

SOLUTIONS

There are multiple solutions to most of the math lessons. Here are examples of correct solutions for each math lesson.

- 1.1
- 1.2
- 1.3 a Finding orange numbers:
- 1.3 b Finding yellow numbers:
- 2.1
- 2.2
- 2.3 a Finding both 3's:
- 2.3 b Finding both 1's:
- 2.3 c Finding both 2's:
- 3.1
- 3.2 a

- 3.2 b
- 3.3 a
- 3.3 b
- 4.1 Example adds up to 5:
- 4.2 a Example adds up to 15:
- 4.2 b Example adds up to 15 via alternate path:
- 4.3 a Example adds up to 10:
- 4.3 b Example adds up to 20:
- 4.3 c Example adds up to 30:

## TECH SPECS

## Code Card Definitions

Each code card itself represents a function or chunk of code that tells the robot's motors, light, and speaker what to do. Here are the specs for each code card and how many are included in the kit, counting both sides.

Image	Name Description	Qty.
	<b>Start</b> Every main program must begin with this card. Only used in the main program.	4
	<b>End</b> Every program, including main and function programs, must end with this card.	10
	<b>Red Function Start</b> The red function program must start with this card. This function is called when the robot scans the matching base map card (red star).	2
	<b>Green Function Start</b> The green function program must start with this card. This function is called when the robot scans the matching base map card (green star).	2
	<b>Blue Function Start</b> The blue function program must start with this card. This function is called when the robot scans the matching base map card (blue star).	2
	<b>If (Conditional Element)</b> This is the start card for a conditional (if-then) function. When the robot scans an Event card that satisfies the condition, the function runs.	2
	<b>Do (Conditional Element)</b> This card can only be used with the If card in a conditional function. If the condition is satisfied, the sequence after the Do card runs.	2
	<b>Else (Conditional Element)</b> This card can only be used with the If card in a conditional function. If the condition is not satisfied, the sequence after the Else card runs. Note: Move Forward, Move Backward, Turn Right, Turn Left, and Pause Movement cannot be used in the Else statement after the Else card.	2
	<b>And (Conditional Element)</b> This card can only be used with the If card in a conditional function. When used, two conditions must be met for the function to run.	1
	<b>Or (Conditional Element)</b> This card can only be used with the If card in a conditional function. When used, either one of two conditions can be met for the function to run.	1
	<b>Event 1 (Conditional Element)</b> Used in a conditional function, this card defines the condition that must be met for the function to run. The robot must scan the matching map card.	1
	<b>Event 2 (Conditional Element)</b> Used in a conditional function, this card defines the condition that must be met for the function to run. The robot must scan the matching map card.	1
	<b>Event 3 (Conditional Element)</b> Used in a conditional function, this card defines the condition that must be met for the function to run. The robot must scan the matching map card.	1

Image	Name Description	Qty.
	<b>Event 4 (Conditional Element)</b> Used in a conditional function, this card defines the condition that must be met for the function to run. The robot must scan the matching map card.	1
	<b>Green Simple Loop Start/End</b> These two cards allow you to repeat a sequence of code placed between them a specific number of times, defined by a number card.	4
	<b>Red Simple Loop Start/End</b> These two cards allow you to repeat a sequence of code placed between them a specific number of times, defined by a number card.	4
	<b>Move Forward</b> This card tells the robot to move forward one map card. It can only be used in the main program. It can be repeated with a number card.	24
	<b>Move Backward</b> This card tells the robot to move backward one map card. It can only be used in the main program. It can be repeated with a number card.	24
	<b>Turn Right (Clockwise)</b> This card tells the robot to turn 90 degrees to the right. It can be repeated with a number card.	18
	<b>Turn Left (Counterclockwise)</b> This card tells the robot to turn 90 degrees to the left. It can be repeated with a number card.	18
	<b>Pause Movement</b> This card tells the robot to pause for one second. It can only be used in the main program. It can be repeated with a number card.	4
	<b>Turn Output Gear Clockwise</b> This card tells the robot to turn its output gear clockwise for one second. It can only be used in a function. It can be repeated with a number card.	5
	<b>Turn Output Gear Counterclockwise</b> This card tells the robot to turn its output gear counterclockwise for one second. It can only be used in a function. It can be repeated with a number card.	5
	<b>Pause Output Gear</b> This card tells the robot to pause turning its output gear for one second. It can only be used in a function. It can be repeated with a number card.	4
	<b>Play Sound: Hi!</b> This card tells the robot to make a "Hi!" sound. It can be repeated with a number card.	2
	<b>Play Sound: Ahh</b> This card tells the robot to make an "Ahh" sound, as if happy. It can be repeated with a number card.	2
	<b>Play Sound: Huh?</b> This card tells the robot to make a "Huh?" sound, as if questioning. It can be repeated with a number card.	2
	<b>Play Sound: Aargh</b> This card tells the robot to make an "Aargh" sound, as if frustrated. It can be repeated with a number card.	2
	<b>Play Sound: Mouse</b> This card tells the robot to squeak like a mouse singing a little song. It can be repeated with a number card.	2
	<b>Play Sound: Penguin</b> This card tells the robot to make the sound of a squawking penguin. It can be repeated with a number card.	2

## TECH SPECS
































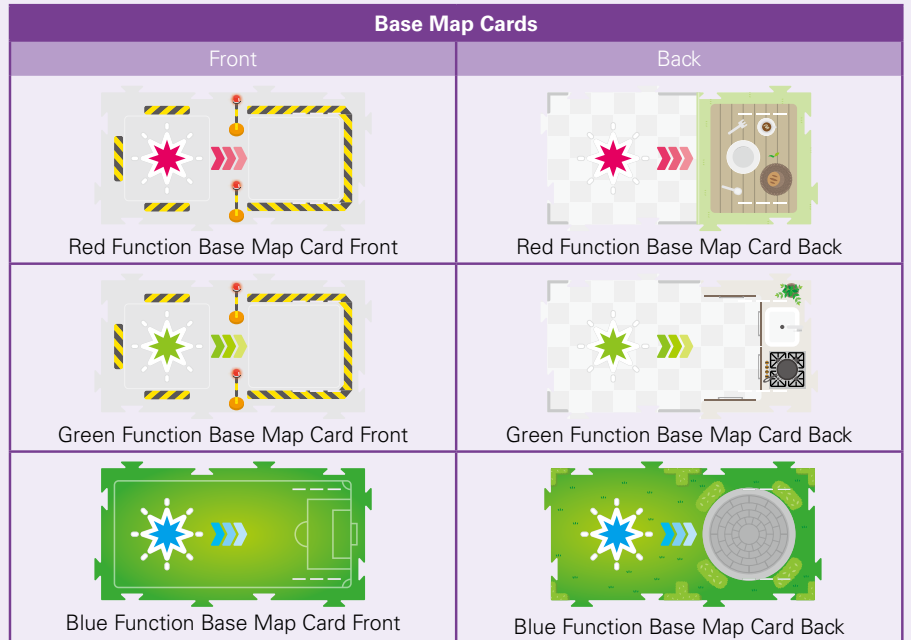
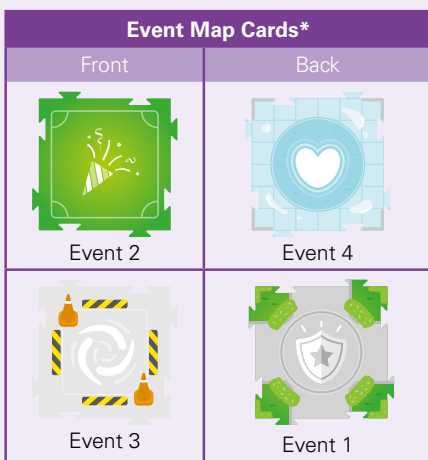
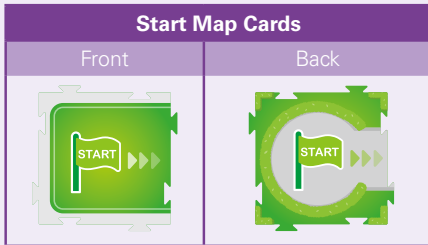
Image	Name Description	Qty.
	<b>Play Sound: Cheering</b> This card tells the robot to play the sound of a cheering crowd. It can be repeated with a number card.	2
	<b>Play Sound: Factory</b> This card tells the robot to play the sound of machines in a factory. It can be repeated with a number card.	2
	<b>Play Sound: Fire hose</b> This card tells the robot to play the sound of a fire hose spraying water. It can be repeated with a number card.	2
	<b>Play Sound: Siren</b> This card tells the robot to play the sound of an emergency vehicle's siren. It can be repeated with a number card.	2
	<b>Light Color: Blue</b> This card tells the robot to change the color of the light inside its output gear to blue for one second. It can be repeated with a number card.	2
	<b>Light Color: Purple</b> This card tells the robot to change the color of the light inside its output gear to purple for one second. It can be repeated with a number card.	2
	<b>Light Color: Pink</b> This card tells the robot to change the color of the light inside its output gear to pink for one second. It can be repeated with a number card.	2
	<b>Light Color: Red</b> This card tells the robot to change the color of the light inside its output gear to red for one second. It can be repeated with a number card.	2
	<b>Light Color: Orange</b> This card tells the robot to change the color of the light inside its output gear to orange for one second. It can be repeated with a number card.	2
	<b>Light Color: Yellow</b> This card tells the robot to change the color of the light inside its output gear to yellow for one second. It can be repeated with a number card.	2
	<b>Light Color: Green</b> This card tells the robot to change the color of the light inside its output gear to green for one second. It can be repeated with a number card.	2
	<b>Light Color: Rainbow</b> This card tells the robot to cycle through seven colors of light in its output gear for half a second each. It can be repeated with a number card.	2
	<b>Light Effect: Disco Strobe</b> This card tells the robot to light up its output gear in a very-fast, on-off flashing pattern, in a purple color by default, and for three seconds. It can be repeated with a number card.	2
	<b>Light Effect: Emergency Vehicle Light</b> This card tells the robot to light up its output gear in a pattern like an emergency vehicle's light, in a purple color by default, and for three seconds. It can be repeated with a number card.	2
	<b>Light Effect: Falling Star</b> This card tells the robot to light up its output gear in a fast-slow-fast flashing pattern, in a purple color by default, and for three seconds. It can be repeated with a number card.	2
	<b>Light Effect: Twinkling Star</b> This card tells the robot to light up its output gear continuously with a little twinkle in the middle, in a purple color by default, and for three seconds. It can be repeated with a number card.	2

Image	Name Description	Qty.
	<b>Light Effect: Firefly</b> This card tells the robot to light up its output gear in a pattern like a firefly's light, in a purple color by default, and for three seconds. It can be repeated with a number card.	2
	<b>Light Effect: Slow Blink</b> This card tells the robot to light up its output gear in a slow, on-off blinking pattern, in a purple color by default, and for three seconds. It can be repeated with a number card.	2
	<b>Light Effect: Medium Blink</b> This card tells the robot to light up its output gear in a medium-speed blinking pattern, in a purple color by default, and for three seconds. It can be repeated with a number card.	2
	<b>Light Effect: Fast Blink</b> This card tells the robot to light up its output gear in a fast, on-off blinking pattern, in a purple color by default, and for three seconds. It can be repeated with a number card.	2
	<b>Light Effect: Speeding Up</b> This card tells the robot to light up its output gear in an increasingly fast blinking pattern, in a purple color by default, and for three seconds. It can be repeated with a number card.	2
	<b>Light Effect: Slowing Down</b> This card tells the robot to light up its output gear in a decreasingly fast blinking pattern, in a purple color by default, and for three seconds. It can be repeated with a number card.	2
	<b>Number Cards 1 through 9</b> These cards tell the robot to repeat a code card's instructions by the number of times printed on the number card when the number card is placed immediately after the code card in the sequence. This only works with the following code cards:	2
	<ul style="list-style-type: none"> <li>• Simple Loop Start (but not Simple Loop End)</li> <li>• Move Forward and Move Backward</li> <li>• Turn Right and Turn Left</li> <li>• Pause Movement</li> <li>• Turn Output Gear Clockwise and Turn Output Gear Counterclockwise</li> <li>• Pause Output Gear</li> <li>• All Play Sound cards</li> <li>• All Light Color cards</li> <li>• All Light Effect cards</li> </ul>	2
		2
		2
	You cannot place more than one number card consecutively (one after another without interruption) in a program, or the robot will give you an error.	2
	<ul style="list-style-type: none"> <li>• Number 1: Execute preceding code card 1 time</li> <li>• Number 2: Execute preceding code card 2 times</li> <li>• Number 3: Execute preceding code card 3 times</li> <li>• Number 4: Execute preceding code card 4 times</li> <li>• Number 5: Execute preceding code card 5 times</li> <li>• Number 6: Execute preceding code card 6 times</li> <li>• Number 7: Execute preceding code card 7 times</li> <li>• Number 8: Execute preceding code card 8 time</li> <li>• Number 9: Execute preceding code card 9 times</li> </ul>	2
		2
		2
		2

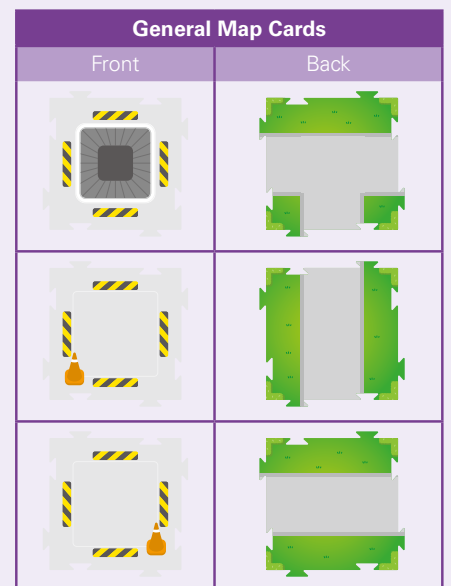
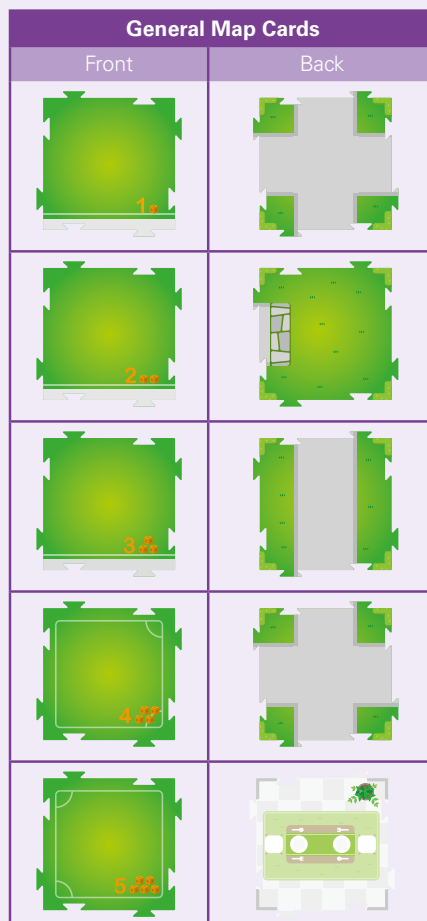
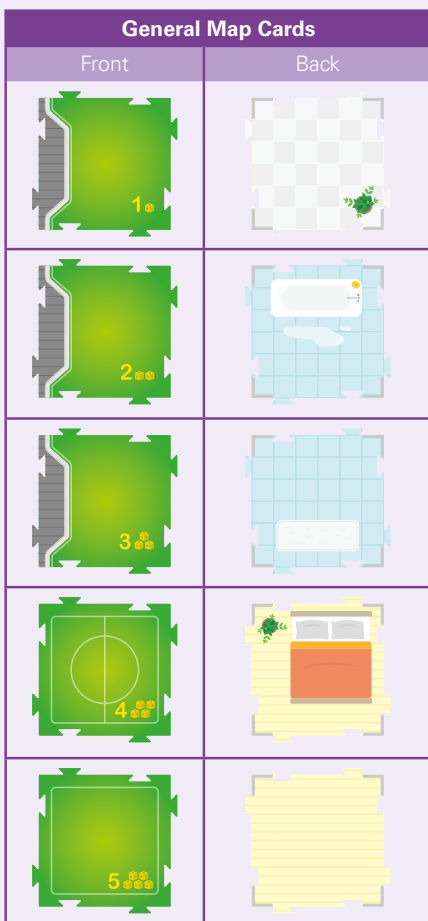
**TECH SPECS**

**Map Card Overview**

There are four basic types of map cards included in this kit. The map cards are not all interchangeable, as each one has a special OID pattern printed on it. You need to make sure you are using the correct map cards in the correct places. Here is an overview of all the map cards.



\*Note: Event map cards cause the robot to perform a default action when they are scanned and no matching event code card has been used.

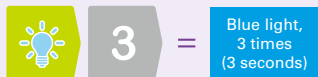




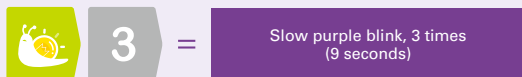
## Combining Light Cards

The light color and light effect code cards can be combined together in the program to make more complex results.

Place a number card after a light color card to change the number of times the light color runs, making the light stay on longer.



Place a number card after a light effect card to change the number of times the light effect runs, making the light effect run longer.



### Examples

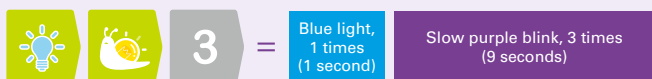
If you combine the cards together as follows, the light effect runs two times and then the light changes to blue and runs three times, which is about three seconds in this case.



If you combine the cards together as follows, the color of the light effect will be blue instead of the default purple and it will run three times, or about nine seconds in this case. When the light effect card comes before the light color card, the color of the light effect changes.



If you place the cards as follows, the light effect will not be combined with the light color. The blue light will shine for one second and then the light effect will run three times in the default purple color. When the light effect card comes after the light color card, the two cards do not combine.



Not sure how to interpret all this? When in doubt, try it out!

## Troubleshooting Tips

### If your robot isn't recording:

- Make sure you are starting your program with a Start, Function Start, or If code card.
- Make sure your robot's batteries are charged and the robot is not giving you the low-power indicator alert.
- Make sure the robot is facing the correct direction, following the arrows on the code card frame.

### If your robot is acting funny or not working properly:

- Make sure the batteries are sufficiently charged. When the batteries are running low, the robot will alert you with a flashing orange light and play a low-power indicator sound.
- Dust, stains, or fading on the surface of the OID cards may interfere with the reading of the OID codes. Please keep the cards clean and dry.
- If your robot can't record a function start code card or an If code card, the robot might be in math mode. Hold down the Erase button for two seconds to go back to normal mode.

### If your robot is flashing orange and stopping in the middle of a line of code cards:

- If the robot encounters any problem while recording, it will flash orange and red and play an error sound. Check the code cards and make sure they are in the correct order.

### If your robot does unexpected movements when it starts to run a program:

- This is normal. The robot is calibrating its position. If you place the robot in the center of the Start map card, it will have a shorter calibration time.
- The calibration process improves the precision of the robot's movements. Do not move the robot during the calibration process.
- The robot runs a quick calibration when you press the Run button. The robot runs a standard calibration the first time you press the Run button after turning the robot on, or when you press and hold the Run button for two seconds. The main program is executed after calibration.

## Need Help?

Contact Thames & Kosmos Technical Support!

### United States

Email: [techsupport@thamesandkosmos.com](mailto:techsupport@thamesandkosmos.com)  
 Web: [thamesandkosmos.com](http://thamesandkosmos.com)

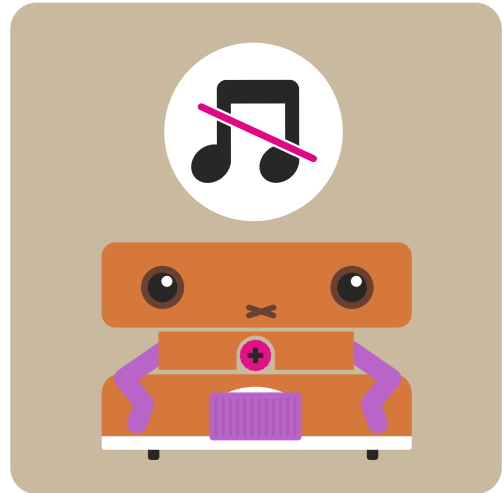
### United Kingdom

Email: [techsupport@thamesandkosmos.co.uk](mailto:techsupport@thamesandkosmos.co.uk)  
 Web: [thamesandkosmos.co.uk](http://thamesandkosmos.co.uk)

# Background Music



**ON**

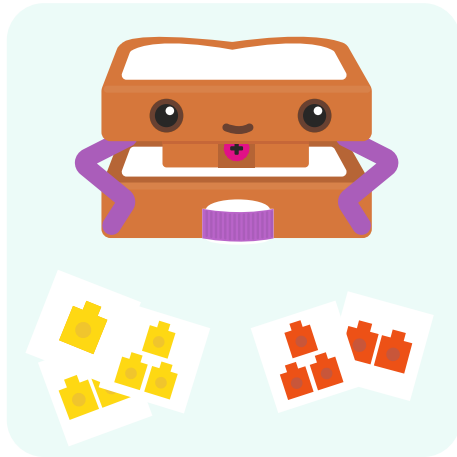


**OFF**

# Math Programs

## Lesson 1

Find cubes of the same color



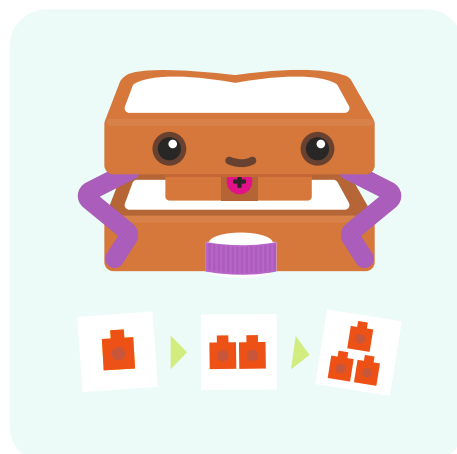
## Lesson 2

Find cubes of equal value

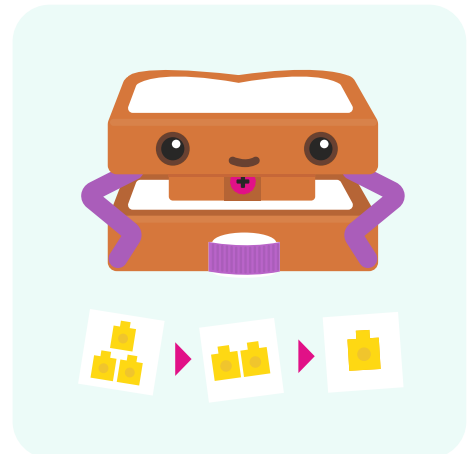


## Lesson 3

Find cubes in sequence



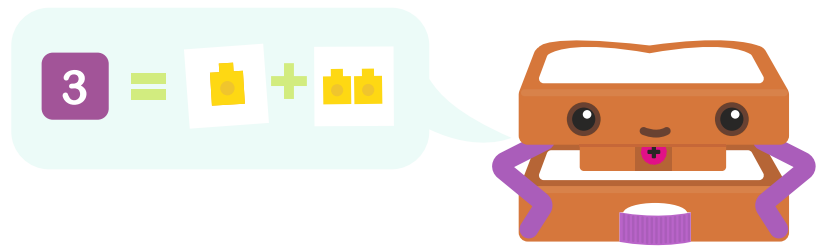
**Increasing value**



**Decreasing value**

# Lesson 4

Find cubes adding up to a variable



1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

## GLOSSARY

**Algorithm:** a part of a computer program that is used to solve a stated problem using a sequence of calculations or steps. An algorithm is a step-by-step way of solving a problem.

**Bug:** an error in a program that causes unexpected or unwanted behavior.

**Code:** In general, a code is a system of words, letters, or symbols that represent other words or meanings. In robotics, "code" refers to the language of the program or a segment of the program.

**Command:** an instruction given to a computer that a computer can interpret and act on.

**Conditional statement:** a portion of a program that tells the computer or robot to perform different instructions depending on whether or not a specified condition, or set of conditions, is true or false. Conditional statements are also called if-then statements or just conditionals for short.

**Debugging:** the process of finding and preventing unwanted behavior in a program, computer, or robot.

**Event:** an occurrence or an interaction that can be recognized by the computer or robot.

**Function:** a set of steps that can be used again and again in a larger program. A function is written once and given a name or label. Then the function can be called upon in the program whenever it is needed, eliminating the need for the code of the function to be written more than once.

**Loop:** a set of steps that repeats a number of times. Loops can be programmed to repeat a set number of times, forever, only while something else is happening, or until another thing happens.

**Program:** a sequence of instructions that tells a computer or robot what to do.

**Programming:** the process of writing a program. Also called coding.

**Robot:** a mechanical agent controlled by a computer program. Robots can be programmed to perform all sorts of tasks and movements. Robots can assemble cars, play soccer, vacuum floors, deliver packages, map terrain, climb mountains, entertain people, cook dinner, and do countless other things. Robots use sensors to sense their environments, and motors, lights, speakers, and other output devices to interact with their environments.

**Robotics:** the branch of technology that deals with robots!

**Sensor:** an electronic device that can detect changes in a robot's environment or state.

**Sequence:** a set of steps or commands arranged in a specific order. Computers run through the steps of a sequence in order, executing one at a time, for the purpose of performing a specific task that the sequence was created to perform.

**Variable:** a quantity in a calculation or program that is assumed to vary, or change, or be capable of varying in value.

1st Edition 2018 Thames & Kosmos, LLC, Providence, RI, USA  
Thames & Kosmos® is a registered trademark of Thames & Kosmos, LLC.

This work, including all its parts, is copyright protected. Any use outside the specific limits of the copyright law without the consent of the publisher is prohibited and punishable by law. This applies specifically to reproductions, translations, microfilming, and storage and processing in electronic systems and networks. We do not guarantee that all material in this work is free from copyright or other protection.

Technical Product Development: Genius Toy Taiwan Co., Ltd., Taichung, Taiwan, R.O.C.  
Text and Editing: Ted McGuire  
Additional Graphics and Packaging: Dan Freitas  
Product Development Support: Camille Duhamel, Franckh-Kosmos Verlags-GmbH & Co. KG  
Manual Layout: Mark Geary

Manual design concept: Atelier Bea Klenk, Berlin  
Manual illustrations: Genius Toy Taiwan Co., Ltd., Taichung, Taiwan, R.O.C., and Thames & Kosmos  
Manual photos: Thames & Kosmos

The publisher has made every effort to locate the holders of image rights for all of the photos used. If in any individual cases any holders of image rights have not been acknowledged, they are asked to provide evidence to the publisher of their image rights so that they may be paid an image fee in line with the industry standard.

Distributed in North America by Thames & Kosmos, LLC, Providence, RI 02903  
Phone: 800-587-2872; Web: [www.thamesandkosmos.com](http://www.thamesandkosmos.com)

Distributed in United Kingdom by Thames & Kosmos UK, LP, Cranbrook, Kent TN17 3HE  
Phone: 01580 713000; Web: [www.thamesandkosmos.co.uk](http://www.thamesandkosmos.co.uk)

We reserve the right to make technical changes.

Printed in Taiwan / Imprimé en Taiwan