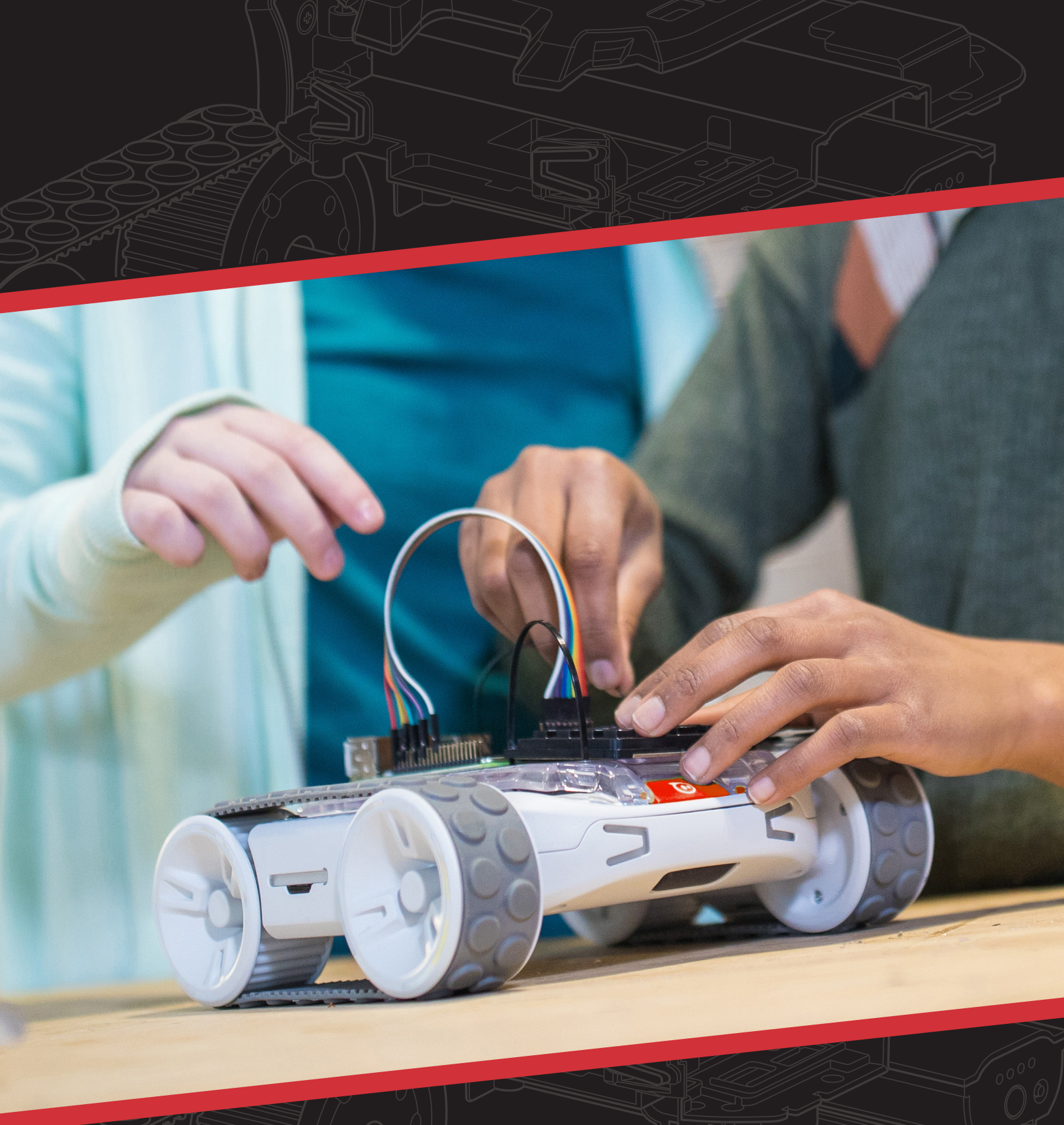


sphero®

RVR+™

EDUCATOR GUIDE

sphero®

RVR+™

EDUCATOR GUIDE

For information about this title or to order other books and/or electronic media, contact the publisher:
Sphero, Inc
education@sphero.com
www.sphero.com

Library of Congress Control Number: 978-1-7331447-6-6

Table of Contents

Introduction

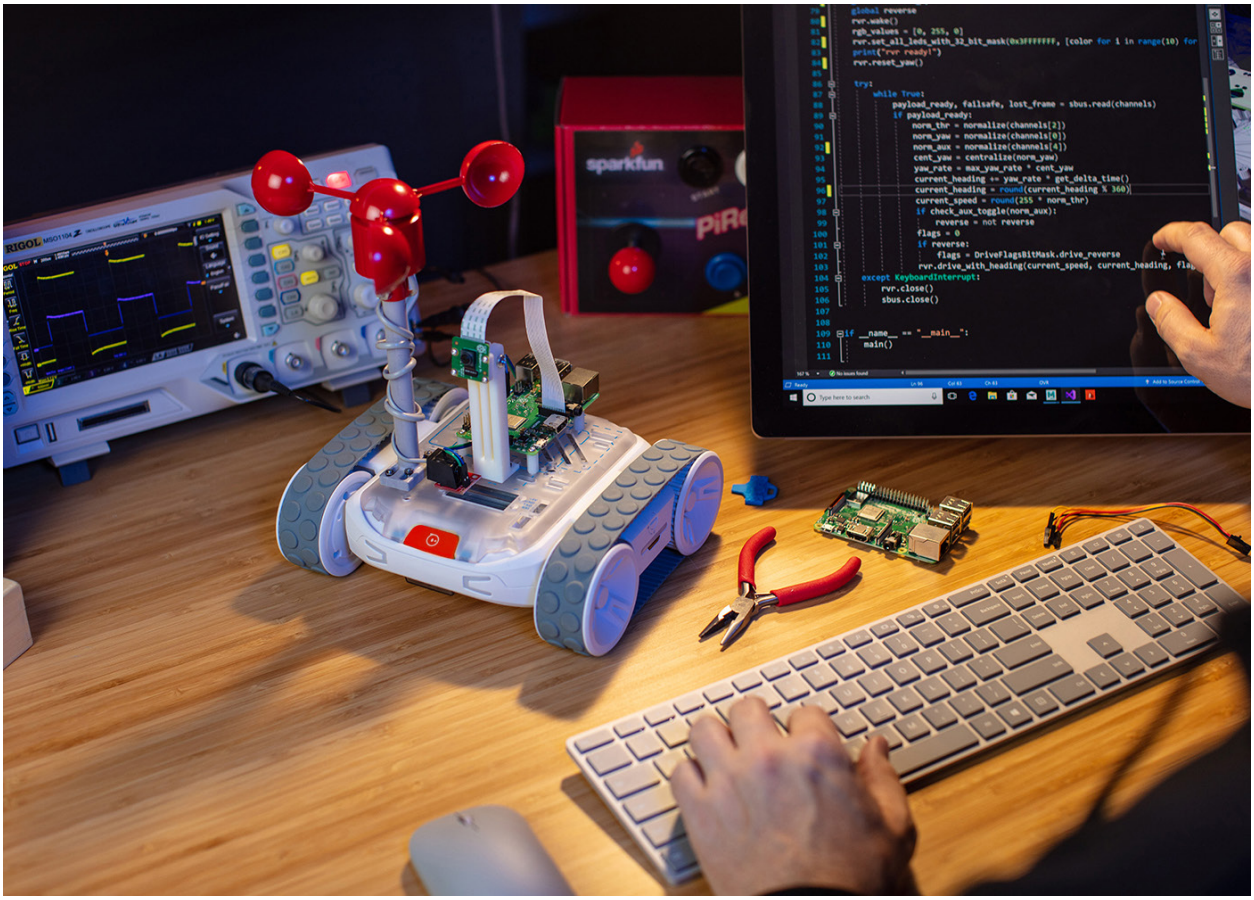
Welcome: Meet Sphero RVR+

RVR+ is Sphero's revolutionary take on the programmable robot and is optimized for middle and high school classrooms and makerspaces. It's drivable right out of the box, packed with a diverse suite of sensors, and built for customization. RVR+ is expandable and made for novice to advanced programmers. Students learn coding skills and computer science basics with the free Sphero Edu app and advanced programmers can level up with the Public Sphero SDK to program with Python and connect third-party hardware.

- **Build Coding & Computer Science Skills**: Program in the Sphero Edu app with draw & drive, block programming, or JavaScript on any computing device.
- **Level Up:** Use the Sphero Public SDK and API libraries for more advanced hackers and makers.
- **Build on It:** Build mechanisms and inventions on top of RVR+ to expand its abilities.
- **Collect Data:** RVR+ features advanced onboard sensors—including a gyroscope, accelerometer, color sensor, and light sensor.
- **Drive Anywhere:** With RVR+'s powerful motor, enhanced gearbox, and high torque, RVR+ can traverse all kinds of terrain.
- **Code with Confidence:** Simple enough for a novice to get started, yet advanced enough to create anything your imagination can come up with.
- **Expand on It:** The customizable expansion plate allows connection of most third-party hardware including Raspberry Pi, Arduino, micro:bit, and Sphero's own littleBits.

Sphero Mission

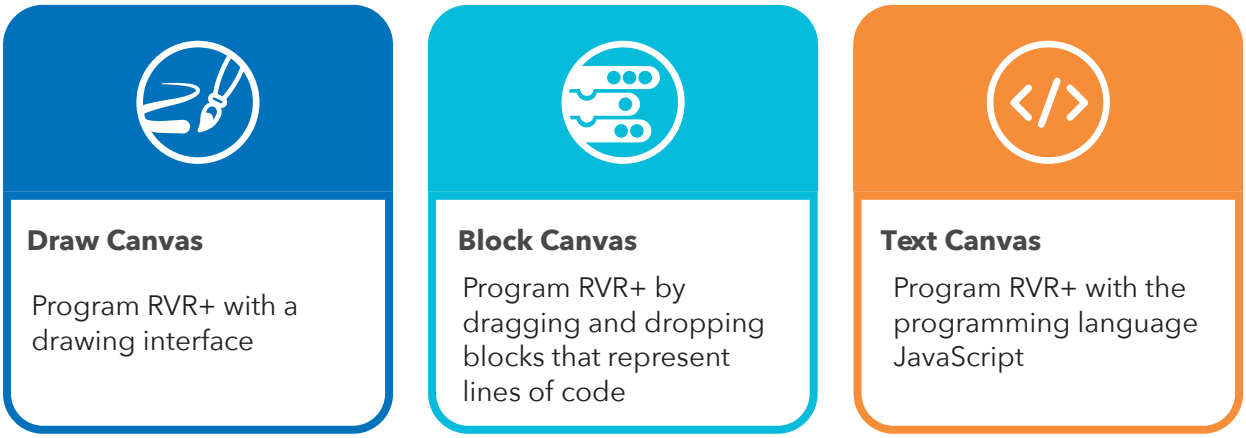
Sphero is transforming PK-12 education with accessible tools that encourage exploration, imagination, and perseverance through STEAM and computer science. With the help of educators around the world, we are empowering learners of all backgrounds and abilities to discover their interests and passions while equipping them with the skills they need to be the world's future Changemakers.



Why RVR+?

Standards-Aligned Learning for Grades 5 and Up

RVR+ can be used to teach computer science concepts and supplement subject matter in any content area. Sphero provides extensive learning activities that are aligned to Common Core ELA and Math, NGSS, CSTA, ISTE, and other national and international standards for easy integration into curriculums. You, as a teacher, don't have to be a programming expert to integrate RVR+ into your classroom instruction. The Sphero Edu app offers three different coding "canvases"—Draw, Blocks, and Text – that move from beginner to more advanced coding skills. The three coding canvases make it easy for you to progress as a programmer alongside your students.



RVR+ is truly a low-floor, high-ceiling robot. For young learners, RVR+ makes getting started on their programming journeys engaging and fun. But with advanced sensors, JavaScript Text Canvas, the ability to hack your RVR+ with third-party hardware, and the opportunity to add inventions on top, RVR+ ensures there is no limit to the possibilities for more advanced students.

Computer Science and Engineering for All Students

RVR+ was designed to make learning the fundamentals of programming, computer science, and engineering accessible and fun for all students, regardless of their backgrounds or abilities.

As we continue to shift to a more digitized future in both our work and everyday lives, we will need more workers in engineering and computer-related occupations. The U.S. Bureau of Labor Statistics predicts that the number of jobs in engineering will increase by 6.5%, while jobs in the computer industry will increase by 13% from 2020–2030.

Yet, the computing and engineering fields still struggle to attract and retain a workforce that accurately represents the gender, racial, ethnic, and socioeconomic diversity of our communities. A more diverse workforce—along with the rich perspectives and backgrounds that come with it—is essential to solving the problems of tomorrow.

RVR+ can make students' initial programming and engineering experiences hands-on and visual. Drag a roll block onto the canvas, execute the program, and immediately observe the effect your program has on the robot.

Consider how the following active uses of RVR+ make learning computer science and engineering accessible and tangible:

- using trial and error to program RVR+'s path through a maze
- communicating with a learning partner to adjust the values in a comparator
- using RVR+ to visually see how velocity and acceleration are affected in different terrains
- prototyping a mechanism on top of RVR+ to gather objects from the floor

Compared to more passive instructional methods, such as watching a video tutorial or listening to a teacher, learning with RVR+ places the students squarely in the center of the learning process.

Getting Started

The Robot

RVR+ is as tough as it is accurate. Its variety of features will help learners in grades 5 and up develop programming and computational thinking skills. RVR+'s high torque drive motors, combined with its advanced sensors and enhanced gearbox, allow for powerful and accurate movement. RVR+ is equipped with RGB LEDs that are individually programmable. RVR+'s onboard sensors allow it to gather data during program execution that can be used to program interesting logic in the free Sphero Edu app. Here is an overview of RVR+'s sensors:

- **Accelerometer**: The accelerometer measures linear acceleration and can be used to detect changes in speed and collisions.
- **Gyroscope**: The gyroscope measures twisting or rotational movement around the pitch (x-axis), roll (y-axis), and yaw (z-axis).
- **Light Sensor**: The light sensor reads the light intensity (luminosity) in your environment from 0–100,000 lux, where 0 lux is full darkness and 30,000-100,000 lux in direct sunlight.
- **Infrared (IR) Sensors**: The IR sensors can be used to send and receive messages between RVR+s (and other Sphero robots) within a 4-meter range.
- **Color Sensor**: The color sensor allows RVR+ to drive over colors and recognize and respond to the color it is seeing.

RVR vs. RVR+

If you are familiar with the original RVR, you'll notice that RVR+ has the same basic functions but with some major upgrades!

- **Which one do I have?** RVR+ is easily identifiable by the red button on the front.

RVR

RVR+

RVR+ upgrades at a glance:

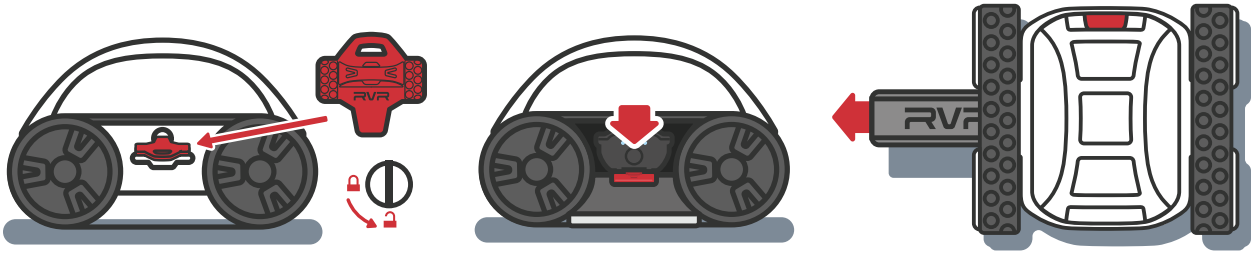
Under-the-hood improvements mean the RVR+ experience will be similar to the original—but with some game-changing improvements.

- **Enhanced Gearbox** - The gearbox has been redesigned to improve torque, payload capacity, and durability to take your inventions even further.
- **Higher Torque** - Fifty percent higher torque yields much more accurate movements and sensor data.
- **Improved Color Sensor** - A totally new color sensor enhances your ability to incorporate color tiles into your RVR+ programs.
- **Rechargeable Battery** - Improved rechargeable and swappable battery to keep programs, inventions, and classroom projects running without any downtime.

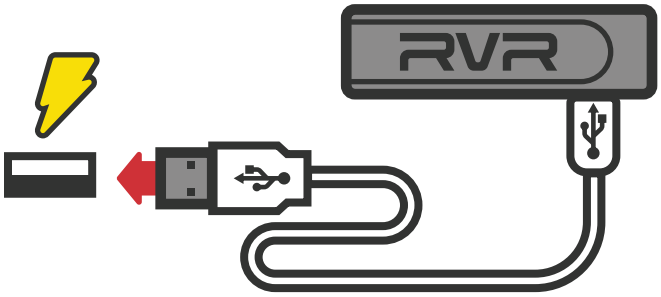
Operation

Charge a RVR+ Battery:

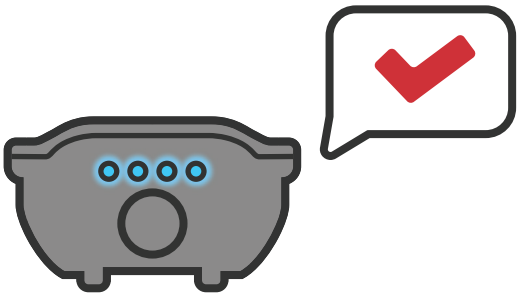
1. Remove the battery from RVR+, using the included red key.



2. Insert the USB-C charging cable into the RVR+ battery charging port. (Wall adapter not included. You will need a USB wall adapter or USB outlet.)

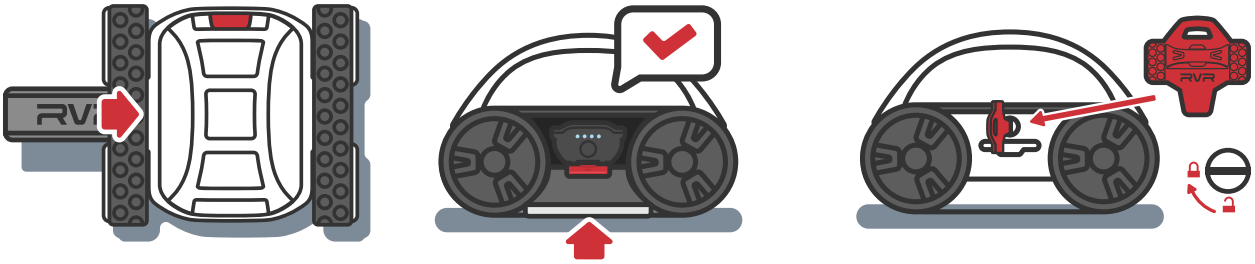


3. Wait until all four blue LEDs are illuminated to indicate a full charge. If blue lights are off, press the button under the LED lights, to show charge level.

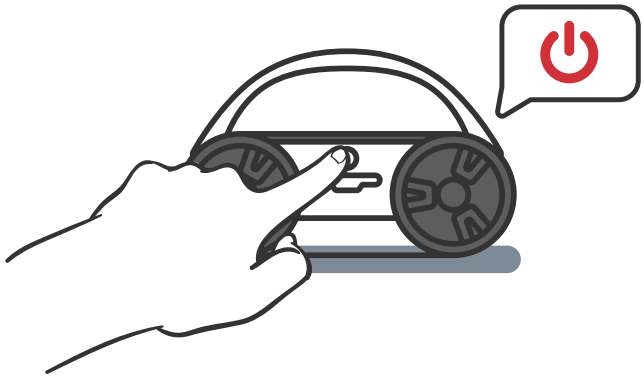


Turn on a RVR+:

1. Ensure you have a charged battery, and secure the battery into RVR+ using the included red key.



2. Press the power button on the side of RVR+. The LEDs will illuminate, indicating RVR+ is powered on.

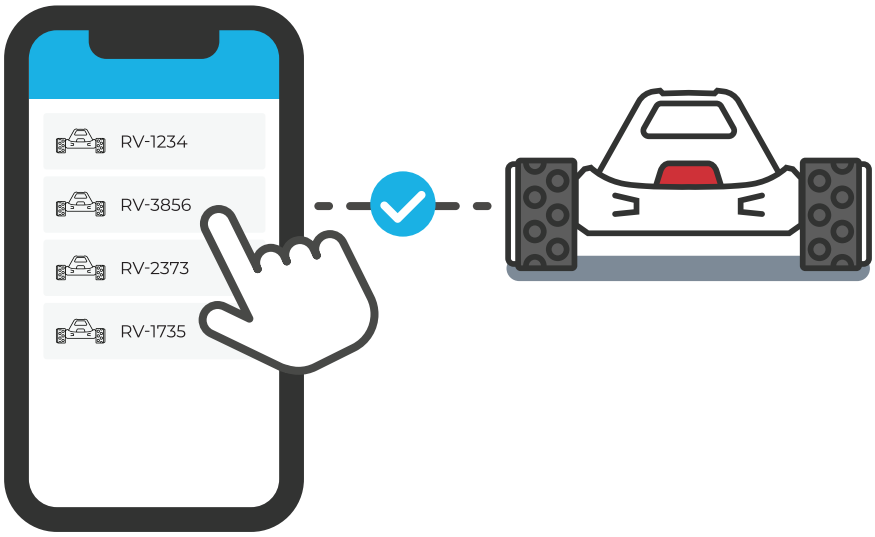


Connect RVR+ to the Sphero Edu app:

1. Download the Sphero Edu app on iOS, Android, Kindle, Mac, Windows, or Chrome: sphero.cc/edu-d

2. Open the Sphero Edu app and find the 'Connect Robot' button.

3. Hold your device near your RVR+.
4. Select RVR/RVR+ from the list of robots and select your robot name from the list to connect. Your robot name (RV-XXX) will be printed on the bottom of your RVR+. This is the name you will see in the Sphero Edu app.



Caring for RVR+

RVR+ is manufactured to be durable and tough. We want you to be able to test the limits of the terrain that RVR+ can maneuver in, but also remind you that because RVR+ has exposed ports on the top, it is not recommended to use RVR+ in a wet environment. RVR+ has proven it's a capable and durable robot based on durability tests in the product development process, but it's still not recommended to test its capability for drops and falls from elevated surfaces. It is best to test your programs on the floor instead of a desk.

To clean RVR+, you can use your preferred cleaning product on a towel and/or disinfecting wipes. Do not spray directly onto RVR, and avoid the open ports while wiping down. RVR+'s treads can be removed and cleaned under running water. Be sure not to use any harsh solvents or anything abrasive or sharp to clean them. Allow RVR+ to fully dry before use.

Battery Care:

If you are not using your RVR+ for a few weeks or more, like over winter or summer break, use the following storage tips:

1. Turn off RVR+.
2. Remove the battery and leave it unplugged. For battery longevity, it is recommended to store batteries at about 50% charge in a temperature-controlled environment.
3. Close and lock the battery door with the key to ensure no debris gets in the battery compartment.

The Sphero Edu App

The Sphero Edu app is designed to allow you and your learners to take advantage of RVR+'s features and grow with their robot as their understanding of programming develops.

The Sphero Edu app includes many features, including:

- standard-aligned learning activities for a variety of skill levels and content areas
- pre-created programs for RVR+. Execute the program as is or use it as a starter for coding your own!
- Draw, Block, and Text Canvases for creating programs

Compatible Devices

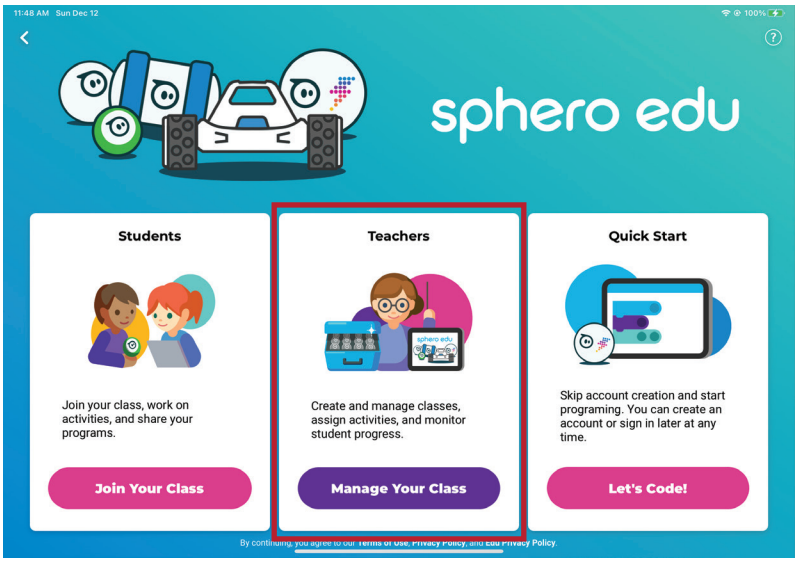
To get started, the app will need to be downloaded on student learning devices. The app is available for free in the iOS, Google Play, MacOS, Microsoft, and Amazon app stores for use on any type of classroom device. For use on Chromebooks, you will need to install the Sphero Edu Android app from the Google Play store.

Download the app for your device:

sphero.cc/edu-d

Connecting as an Educator

To get started as a teacher with the Sphero Edu app, select "Teachers" the first time you open the app.



You'll be guided through account creation. After you've created your account, you'll have access to all the same programming tools as your students. You'll also be able to log in to your account at edu.sphero.com to set up classes and assign activities to your students.

CLASSROOM TIP: It's a good idea to set up your class and invite students before they use the app. If students create accounts before a class is made, they will have to enter a guardian email for account approval.

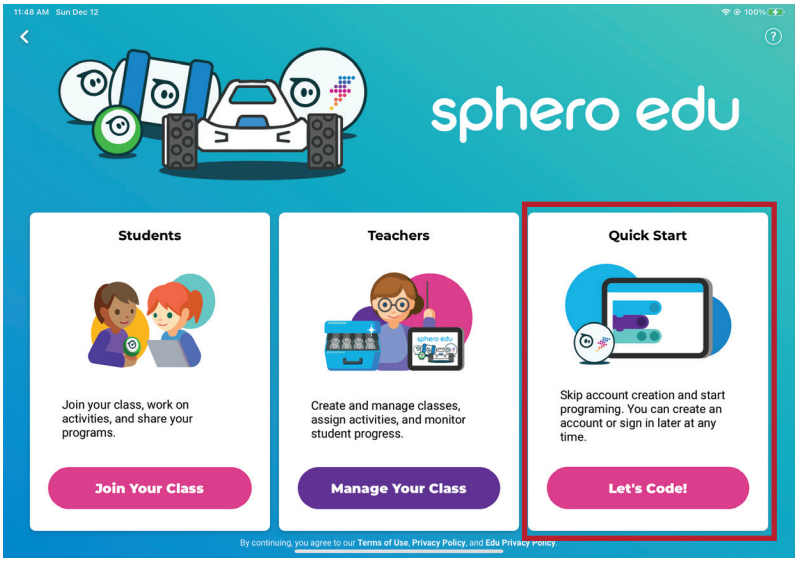
Learn more with the link below about how to set up and manage your Sphero classroom:

sphero.cc/gettingstarted

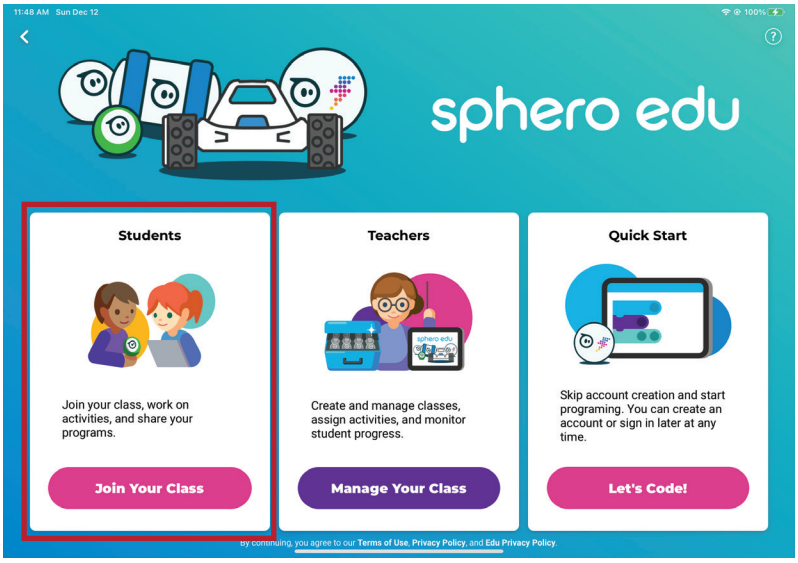
Connecting as a Student

Once the app is installed, students have two options for connecting their devices with RVR+ and getting started with programming:

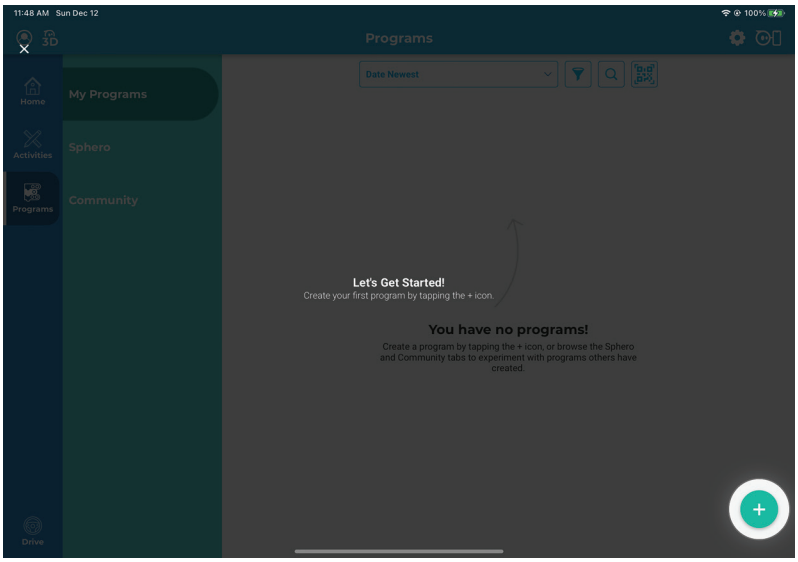
1. Select "Let's Code" in the Quick Start box to jump right in without creating an account. Students will be able to create and execute programs. However, none of their work will be saved to the cloud to be available from another device.



2. Select "Join Your Class" in the Students box to open the app with a class code (if you've already created a class) or sign in with a Google, Clever, or Sphero account. This method will allow students to save their work to the cloud.



3. Once the app opens, choose “Programs”, then select the plus symbol in the low right corner to create program.



Classroom Lessons

Your RVR+ Multi Pack is your STEAM classroom on wheels. But now that you have it, how do you get started with STEAM learning? The eight out-of-the-box, unpack-and-go activities included in this Educator Guide will help you and your students get up and running quickly with RVR+, no previous experience required. Each lesson is designed for students of all experience levels and can be completed in one 45–60 minute class period. Complete them in order or skip around based on your students' comfort level with RVR+, engineering, and programming. While students need to have the Sphero Edu app installed on their devices, they will not need to create or sign into accounts to work on the programs for these classroom activities.

BEGINNER	PROGRAMMING FOCUS
Lesson 1: Drive to Thrive	drive controls
Lesson 2: Block Basics	movement blocks
Lesson 3: Hypotenuse Measure	movement, strings, and location sensor blocks

INTERMEDIATE	PROGRAMMING FOCUS
Lesson 4: Color Events	movement, lights, strings, and event blocks
Lesson 5: Automatic Headlights	movement, lights, and light sensor blocks
Lesson 6: Ramping Up	movement, pitch orientation sensor blocks

ADVANCED	PROGRAMMING FOCUS
Lesson 7: Engineer an Apple Picker	JavaScript movement
Lesson 8: Variable Movement	loops and variable blocks

Before teaching:

- Read through the educator instructions for the activity, try out the challenge, and anticipate obstacles that your students will encounter.
- Gather all related classroom materials and prepare the activity area as directed in the instructions.
- Copy and/or prepare to project the student-facing instructions for the lesson.
- Make sure all robots and programming devices are fully charged.
- Plan for two students per RVR+ robot.

During teaching:

- Use the Exploration and Skills Building steps to launch the activity and introduce foundational skills.
- Give students the opportunity to engage with the Challenge without guidance. Circulate the learning space to celebrate teamwork, mistakes, and problem solving.
- Encourage students to extend their learning with the Extended Challenge options.

After teaching:

- Reflect on student learning. What was easy for students? What was more difficult? Was there an artifact of student learning that you can share with the whole class to address a common challenge or showcase a programming concept or problem solving strategy?
- Plan your next lesson. Use another lesson in this Educator Guide or try one of the suggested Sphero activities in the Go Further section.

Lesson 1: Drive to Thrive

Overview
RVR+ is your tough and tenacious tankbot, powerful enough to move cardboard boxes around your classroom with ease. In this lesson, students learn to drive RVR+ and control its movements.

Level
Beginner

Learning Objectives
1. I can control RVR+'s movement—including speed and direction—with the Sphero Edu app drive controls.
2. I can use RVR+ to push an object through an obstacle course while maximizing speed and efficiency.

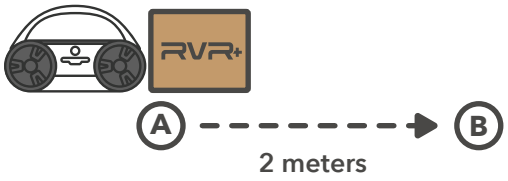
CS Standards
- 1B-AP-08 Compare and refine multiple algorithms for the same task and determine which is the most appropriate.
- 2-AP-13 Decompose problems and subproblems into parts to facilitate the design, implementation, and review of programs.

Content Connections
- Technology

Materials
- RVR+
- programming devices with the Sphero Edu app installed
- small box (tissue, RVR+ box, or other)

Preparation
- Gather small, cubic boxes for RVR+ to push around. These could be boxes RVR+ shipped in or other boxes like a tissue box.
- Set up four to five workspaces around the room. Each workspace will need two floor markers (like post-it notes or scrap pieces of paper), placed approximately 2 meters apart.
- Preview and prepare the Student Instructions (page 26).

Setup



Exploration

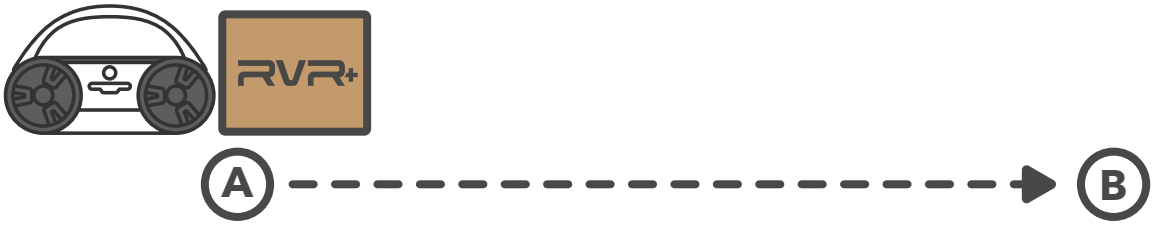
1. Introduce RVR+ to students. Point out and discuss the utility of some of the following features:
   - battery compartment
   - tank treads
   - color sensor
   - IR message beacons
2. Ask students to connect their driving and programming device to RVR+.
   - **Press the power button** on the side of RVR+.
   - **Open the Sphero Edu app** on the programming device.
   - **Select the drive icon** in the lower left corner of the app.
   - **Select Sphero RVR/RVR+** as the robot type.
   - **Select the appropriate RVR+** from the list using the RVR+ ID (RV-XXXX) printed on the underside of RVR+.

TEACHER TIP: Students do not need a Sphero Edu account in order to get started with RVR+. Instruct students to select the Let's Code button in the Quick Start menu to get started right away. Since this lesson only involves driving, it is not important for students to be able to save their progress.

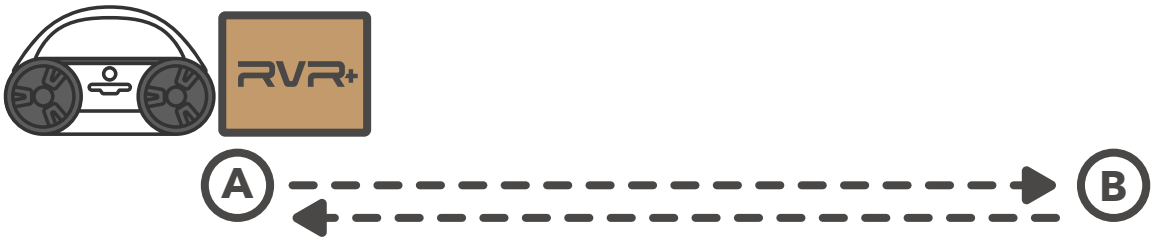
3. Give students three to five minutes to practice driving back and forth from point A to point B and explore the drive controls including aiming their RVR, adjusting the speed, and changing the LED color.

Skills Building

1. Explain that RVR+'s tank treads and enhanced drivetrain make it accurate and powerful! They will explore some of these features in this activity.
2. Challenge students to move a box from point A to point B with their drive controls.
   - Place a box in front of RVR+.
   - Push the box with the front of RVR+.
   - Stop at point B.



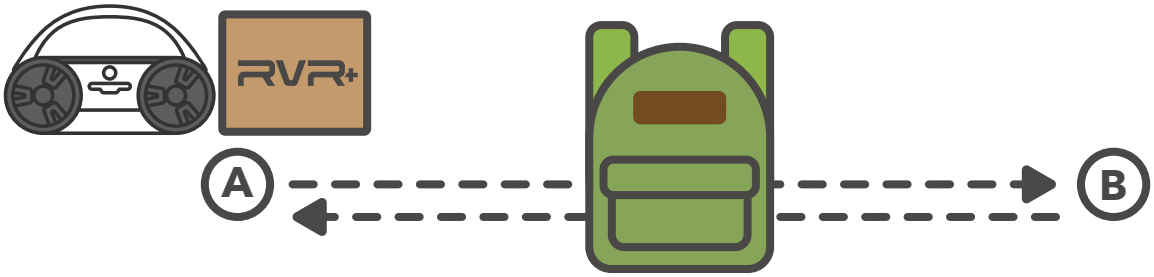
3. Challenge students to move a box from point A to point B and then back to point A.



4. Discuss the following questions with students to help them summarize their learning about driving RVR+.
   - *What is your favorite driving speed? Why?*
   - *What is the most efficient way to push an object back and forth from A to B? What is the fewest number of turns needed?*

Challenge

1. Introduce the next challenge.
   - Place an object like a backpack in between point A and point B.
   - Use RVR+ to push the box back and forth to point B while avoiding the object.



2. As time allows, ask students to see who can complete the course the fastest or using the fewest amount of turns.
   - There are many different ways to accomplish this challenge. Compare and contrast different strategies that students used during the activity.

TEACHER TIP: While driving is fun, in most classroom situations, students will be programming RVR+ to move with blocks or JavaScript. Help students make a connection to programming by asking them to sequence the drive instructions for accomplishing the driving challenges (e.g. "Drive forward 1.75 meters, turn left, drive forward 0.25 meters, turn right, etc.")

Extended Challenge

1. Ask students to get creative and make their own obstacle courses for their classmates to solve. Here are some possibilities:
   - Add point C and point D so students have to push the box to more destinations.
   - Add more obstacles that students have to avoid.

Lesson 1: Drive to Thrive

RVR+ is your tough and tenacious tankbot, powerful enough to move boxes around your classroom with ease. In this lesson, you'll learn to drive RVR+ and control its movements.

Learning Objectives

1. I can control RVR+'s movement—including speed and direction—with the Sphero Edu app drive controls.
2. I can use RVR+ to push an object through an obstacle course while maximizing speed and efficiency.

Setup

Exploration:

Connect RVR+ to your programming device and practice driving RVR+ around your classroom. Can you change the speed? The color?

Skills Building:

Use RVR+ to first push a box from point A to point B. Then figure out how to turn around and push the box back to point A.

Challenge:

Place a backpack or other obstacle directly between point A and point B. Then use RVR+ to push the box around the obstacle back and forth between the two points.

Extended Challenge:

Ready for more? Get creative and make your own obstacle courses for your classmates to solve.

Lesson 2: Block Basics

Overview
Your students have graduated from RVR+ driving school. It's time to level up their skills with programming on the Block Canvas. In this lesson, students learn to control RVR+'s movements with precision using the **roll** and **drive blocks**.

Level
Beginner

Learning Objectives
1. I can control RVR+'s movement with **roll** and **drive blocks**.
2. I can program RVR+ to move around an object in a variety of paths.

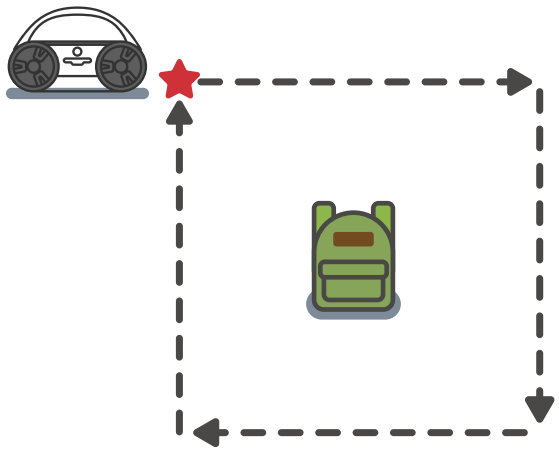
Vocabulary
- **Heading**: Input to program the direction RVR+ is pointed during or before a roll, between 0° and 360°
- **Duration**: Input to program the amount of time that RVR+ will execute a movement like a roll or a spin, in seconds
- **Speed**: Input to program how quickly (or slowly) RVR+ will move, from 0 to 255
- **Distance**: Input to program how far RVR+ will move, measured in centimeters

CS Standards
- 1B-AP-10 Create programs that include sequences, events, loops, and conditionals.
- 2-AP-13 Decompose problems and subproblems into parts to facilitate the design, implementation, and review of programs.
- 2-AP-17 Systematically test and refine programs using a range of test cases.

Content Connections
- Computer science

Materials
- RVR+
- Programming devices with the Sphero Edu app installed
- Classroom objects like backpacks or binders

Preparation
- Ensure that each student group has a 2 by 2 meter area of floor space to program RVR+.
- Preview and prepare to share the student instructions (page 33).

Setup


Program QR Code
sphero.cc/RVR2

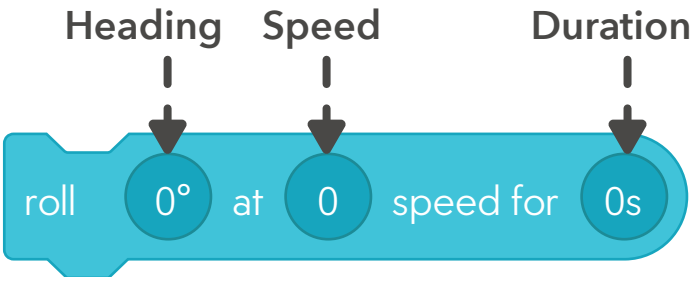
Exploration

1. Ask students to scan the QR code. This will open a Block Canvas in the Sphero Edu app.
   - Give students two to three minutes to look around and explore.
   - Introduce or review the following key features of the Block Canvas: **aim** button, **drive** button, more options under **three dots, block library** at the bottom of the screen, and the **on start program block.**
2. Introduce the activity. Students will program RVR+ to roll around an object like a backpack, textbook, or binder in a variety of paths.

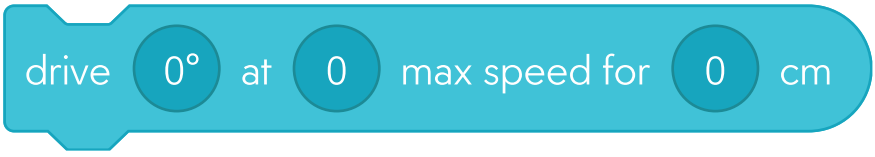
TEACHER TIP: Don't worry about students "breaking" anything on the Block Canvas. They can always use the undo button to get back to where they started. Also, students can create a new Block Canvas either within the Sphero Edu app or by rescanning the QR code.

Skills Building

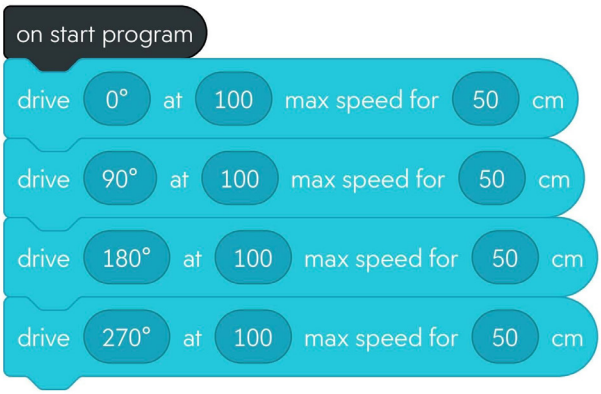
1. Explain that before students can program their RVR+ to roll around the backpack, they need to learn how to use a roll block to make RVR+ drive in a straight line.
2. Discuss the three inputs in the **roll block**.
   - **Heading** is the direction RVR+ is pointed between 0° and 360°. 0° is the aim direction.
   - **Speed** is a number between 0 and 255 that controls how fast RVR+ rolls.
   - **Duration** is the time in seconds for a roll.



3. Discuss the three inputs in the drive block
   - **Heading** is the direction RVR+ is pointed between 0° and 360°. 0° is the aim direction.
   - **Speed** is a number between 0 and 255 that controls how fast RVR+ rolls.
   - **Distance** is the distance RVR+ will travel in cm.



4. Ask students to drag a **roll block** under the **on start program block** to make RVR+ roll in a straight line past their obstacle.
5. Ask students to make RVR+ drive past their obstacle with a **drive block**.
6. Discuss the following with students:
   - *Can you recreate the same RVR+ movement with **roll** and **drive blocks**?*
   - *Which inputs will be the same? Which ones will be different?*
7. Challenge students to use four **drive blocks** to make RVR+ drive around their objects in a square. Student blocks will look something like the following although speeds and durations may vary.



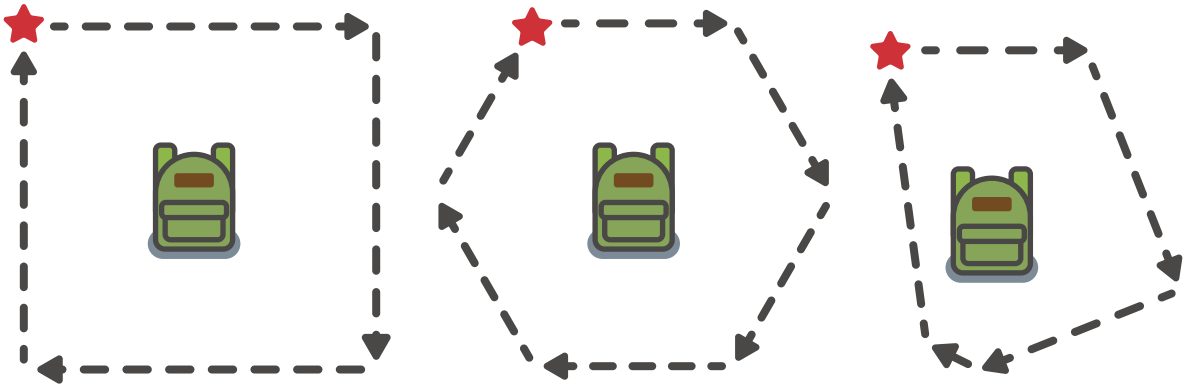
8. Direct students to select the three dots in the upper right-hand corner of the app and choose "Sensor data". They will be able to see all kinds of sensor data from RVR+, from the program that was most recently run.

TEACHER TIP: If you are using a Windows device this sensor data will not be available. You will have to download a CSV file of the data and graph it in a database software (Microsoft Excel, Google Sheets, etc.). However this feature does work on all iOS and Android phones/tablets, as well as Chromebooks and macOS laptops.

TEACHER TIP: Encourage students to use an **exit program block** from the Controls category of the block library so that their program stops executing when RVR+ finishes its path.

Challenge

1. Challenge students to make other paths around the object and end up back where they started. Use the following prompts to get them started:
   - *Can you make RVR+ roll around the object in the opposite direction?*
   - *Can you make RVR+ trace a regular hexagonal shape around the object?*
   - *Can you make RVR+ trace a path with different lengths for every side?*



2. Encourage students who finish early to create and solve their own challenges.

Extended Challenge

1. Once your students are confident using **roll** and **spin blocks**, they'll be ready for more. Prompt them to add some of the following challenges:
   - Use **blocks in the Lights category** of the block library to **change RVR+'s LEDs** at every corner.
   - Play the **spaceship sound** while RVR+ is rolling around the object.
   - Make RVR+ **roll around the object in a circle**.

TEACHER TIP: Students may need a hint on the last challenge. To make RVR+ roll in a circle, students will need to use a **speed 150 block**, a **spin 360° for 3s block**, and a **stop block**.

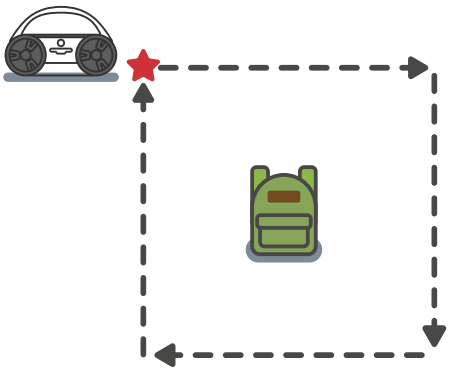
Lesson 2: Block Basics

You've graduated from RVR+ driving school. It's time to level up your skills with programming on the Block Canvas. In this lesson, you'll learn to control RVR+'s movements with precision using the **roll** and **drive blocks**.

Learning Objectives

1. I can control RVR+'s movement with **roll** and **drive blocks.**
2. I can program RVR+ to move around an object in a variety of paths.

Setup



Program QR Code

sphero.cc/RVR2

Exploration:

Open a Block Canvas and take a look around. *What blocks do you see? What do you think they do?* Connect to your RVR+ and get ready to start programming.

Skills Building:

Practice using **drive blocks** to program a path around an object like a backpack or a binder. Review the location data to see your path.

Challenge:

Make new paths around RVR+ around the object.

- *Can you make RVR+ move around the object in the opposite direction?*
- *Can you make RVR+ trace a regular hexagonal shape around the object?*
- *Can you make RVR+ trace a path with different lengths for every side?*

Extended Challenge:

Ready for more? Try some of the following:

- Use **blocks in the Lights category** of the block library to **change RVR+'s LEDs** at every corner.
- Play the **spaceship sound** while RVR+ is rolling around the object.
- Make RVR+ **roll around the object in a circle.**

Lesson 3: Hypotenuse Measure

Overview

The Pythagorean theorem, $A^2 + B^2 = C^2$, is one of the most fundamental rules of geometry. In this lesson, students will use RVR+'s location sensor and the build string block to gather data and verify the length of side C, the hypotenuse in a right triangle.

Level

Beginner

Learning Objectives

1. I can collect information from the RVR+ location sensor.
2. I can build string blocks to program RVR+ to announce data.
3. I can use RVR+ to explore geometric principles like the Pythagorean theorem.

Vocabulary

- **Location sensor**: Measures RVR+'s x-axis, y-axis, and total distance from its point of origin, in centimeters
- **String**: A sequence of characters that combine letters, numbers, and other characters in a computer program

CS Standards

- 2-DA-08 Collect data using computational tools and transform the data to make it more useful and reliable.
- 2-AP-17 Systematically test and refine programs using a range of test cases.

Content Connections

- Math
- Computer science

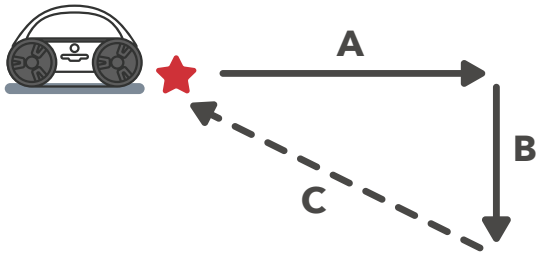
Materials

- RVR+
- Programming devices with the Sphero Edu app installed
- Meter sticks or tape measures
- Pennies, paper scraps, or small items to mark RVR+ position

Preparation

- Plan for student work areas around your classroom. Each student group will need a 3-by-3 meter area of floor space.
- If your students have little prior knowledge about the Pythagorean Theorem, plan to do a more in depth introduction in the Exploration steps of the lesson plan.
- Preview and prepare to share the student instructions (page 39).

Setup



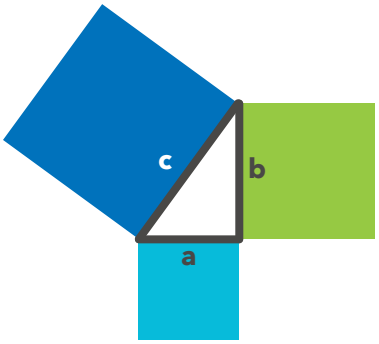
Program QR Code

sphero.cc/RVR3

Exploration

1. Introduce or review the Pythagorean theorem: **For right triangles, the square of side A plus the square of side B is equal to the square of the hypotenuse, side C.**

$$A^2 + B^2 = C^2$$



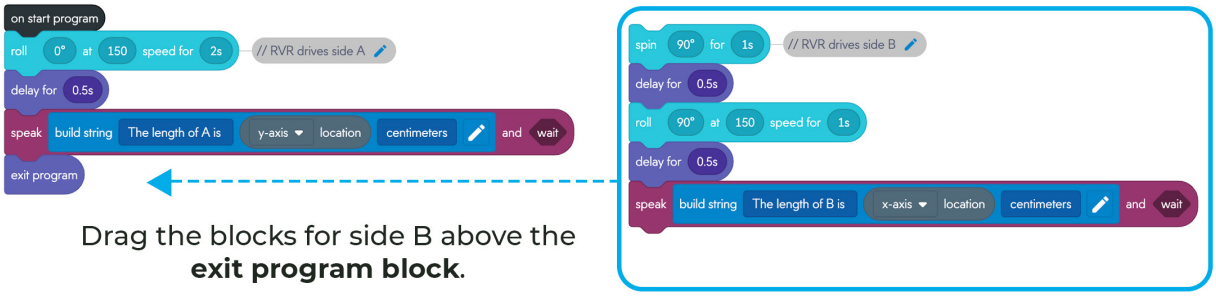
   - If possible, show a quick video like this one (sphero.cc/567405) to get students thinking about the history of the Pythagorean theorem and its usefulness in our everyday lives.
   - Use a 3-4-5 triangle to demonstrate the theorem and how to solve for C.

$$\begin{aligned} 3^2 + 4^2 &= C^2 \\ 9 + 16 &= C^2 \\ 25 &= C^2 \\ 5 &= C \end{aligned}$$

2. Explain that you'll be verifying the Pythagorean theorem with RVR+. In the process, students will be learning how to gather data from RVR+'s sensors and speak the data with a **build string operator**.
3. Tell students that RVR+ is equipped with a location sensor. This can report distance from the starting point—or point of origin— in three different directions:
   - **Y-axis measures forward (+) or backward (-) distance** from the origin, in centimeters.
   - **X-axis measures right (+) or left (-) distance** from the origin, in centimeters.
   - **Total distance measures the linear distance** from the origin in centimeters, which will always be positive.

Skills Building

1. Direct students to use the following steps to open and run the program.
   - Scan the QR code to open the program in the Sphero Edu app.
   - Mark a start position on the floor with a penny, paper scrap, or other small items.
   - Place RVR+ on top of the marker.
   - Press **Start**.
   - Mark RVR+'s stopping point with another small item.
2. Ask students to measure the distance from the starting to the stopping point with a meter stick or tape measure to ensure accuracy.
3. Discuss with students:
   - What is the length of side A?
   - Does the distance announced by the RVR+ program match your measured distance?
4. Direct students to snap the blocks with the comment "RVR+ drives side B" immediately above the **exit program block**, place RVR+ back on its starting point, and execute the program again.



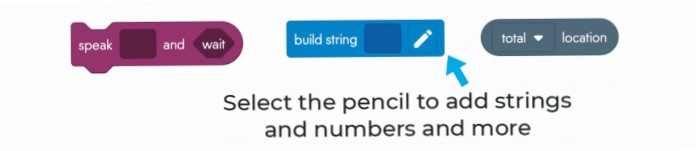
Drag the blocks for side B above the **exit program block**.

TEACHER TIP: Students can add their own comments to coding blocks in the Sphero Edu app by right-clicking or long-pressing on a block. Code comments help programmers explain what their code is meant to do and is an invaluable tool for troubleshooting errors and communicating with other programmers.

5. Ask students to measure side A and side B with a meter stick or tape measure.
6. Discuss with students:
   - *What is the length of side B?*
   - *Does the distance announced by the RVR+ program match your measured distance?*
   - *How can you now find the length of side C?*

Challenge

1. Discuss the **build string operator block** inside of the program's **speak block**.
   - A **string** is a sequence of characters that combine letters, numbers, and more.
   - The **build string operator** allows the program to speak text strings that don't change (i.e. "The length of A is") as well as numerical values (i.e. y-axis location).
2. Challenge students to build a new string that reads:
   - **speak build string "The length of C is" total location "centimeters" and wait.**
   - Students can build the block from the three blocks shown here. The color of the block will help students find the category it came from.

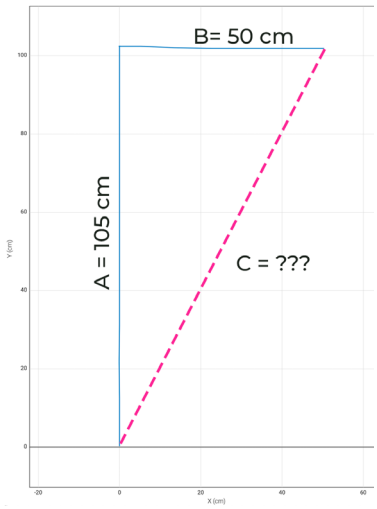


3. Direct students to place the new **speak block** above the **exit program block** and run their program again.
4. Ask students to verify the length of side C in three ways:
   - using the value of the total location sensor at the end of the program
   - measuring the distance from RVR+'s starting to ending position
   - calculating the length using the equation $A^2 + B^2 = C^2$

TEACHER TIP: Encourage students to review the location data after they execute a program by selecting the three dots in the upper right corner of the app. This data provides a clear view of RVR+'s path and the right triangle that students are programming.

5. Discuss with students:
   - *Does the length of side C reported by the Sphero Edu program match your measurement and your Pythagorean theorem calculations?*

Location Sensor Data



Extended Challenge

1. Once your students have explored the location sensor and build string operator, they'll be ready for more. Suggest some of the following challenges:
   - Change the speed and duration of side A and side B **roll blocks** and calculate the side lengths of more triangles
   - Add a third **roll block** to make RVR+ roll back to its starting position.

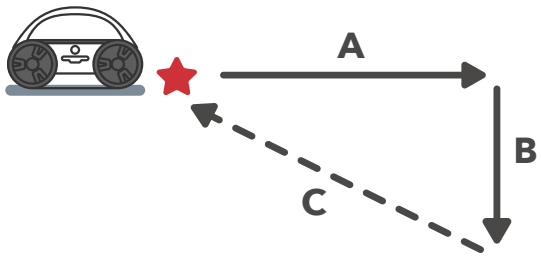
Lesson 3: Hypotenuse Measure

The Pythagorean theorem, $A^2 + B^2 = C^2$, is one of the most fundamental rules of geometry. In this lesson, you'll use RVR+'s location sensor and the build string block to gather data and verify the length of side C, the hypotenuse in a right triangle.

Learning Objectives

1. I can collect information from the RVR+ location sensor.
2. I can use build string blocks to program RVR+ to announce data.
3. I can use RVR+ to explore geometric principles like the Pythagorean theorem.

Setup



Program QR Code

sphero.cc/RVR3

Exploration:

Review the Pythagorean theorem, $A^2 + B^2 = C^2$, and think about how you can use RVR+'s location sensor to explore right triangle side lengths.

Skills Building:

Execute a block program where RVR+ traces side A of a right triangle. Compare your own measurement of side A to data collected by RVR+. Add block to the program to make RVR+ trace side B. Measure side B and compare again to RVR+'s data. How accurate is RVR? How accurate is your measurement?

Challenge:

Use a build string operator to make RVR+ speak the total location distance from the point of origin. Find the length of side C in three ways:

- Using the value of the total location sensor at the end of the program
- Measuring the distance from RVR+'s starting to ending position
- Calculating the length using the equation $A^2 + B^2 = C^2$

Extended Challenge:

Ready for more? Try some of the following:

- Change the speed and duration of side A and side B **roll blocks** and calculate the side lengths of more triangles
- Add a third **roll block** to make RVR+ roll back to its starting position.

Lesson 4: Color Events

Overview
Visible light is made up of waves of varying lengths which the eye detects as color. While RVR+ doesn't have eyes, it does "see" a wide range of colors with its underside sensor! In this lesson, students will learn the capabilities of RVR+'s color sensor and how to program it.

Level
Intermediate

Learning Objectives
1. I can program with an on color event.
2. I can retrieve values from the RVR+ color sensor in the Sphero Edu app.
3. I can explain how the RVR+ color sensor works.

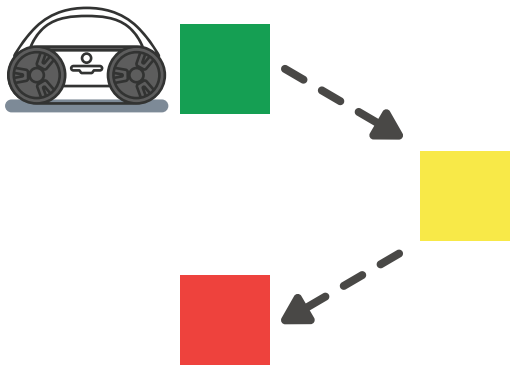
Vocabulary
- **Color sensor**: Measures the numerical RGB values of any color that is directly underneath the sensor on RVR+
- **String**: A sequence of characters that combine letters, numbers, and other characters in a computer program

CS Standards
- 1B-CS-02 Model how computer hardware and software work together as a system to accomplish tasks.
- 1B-AP-10 Create programs that include sequences, events, loops, and conditionals.
- 1B-AP-12 Modify, remix, or incorporate portions of an existing program into one's own work, to develop something new or add more advanced features.

Content Connections
- Science
- Computer science

Materials
- RVR+
- Programming devices with the Sphero Edu app installed
- Color tiles from RVR+ with additional colored pieces of paper

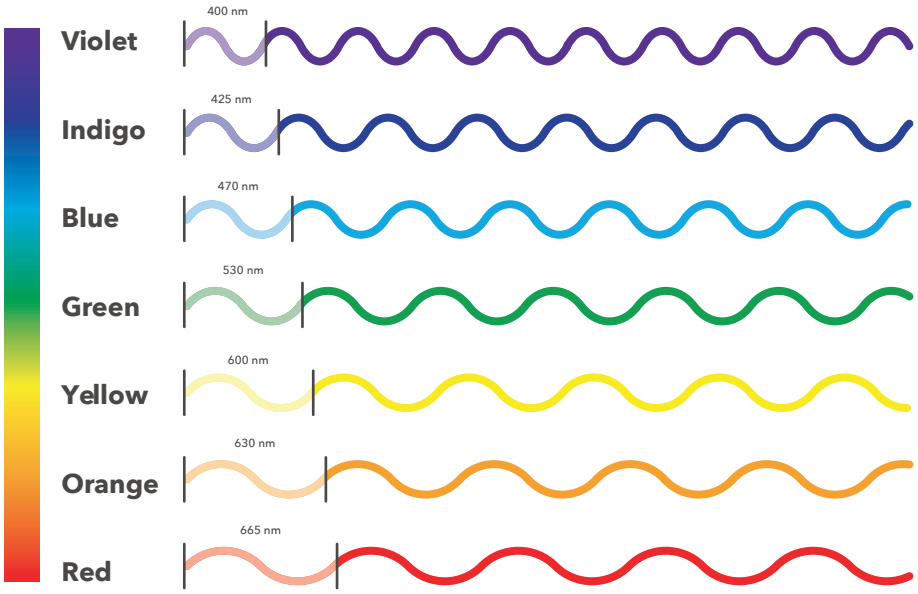
Preparation
- Plan for student work areas around your classroom. Each student group will need a 3-by-3 meter area of floor space.
- Preview and prepare to share the student instructions (page 46).

Setup


Program QR Code
sphero.cc/RVR4

Exploration

1. Introduce the concept that colors are seen by our eyes. But on a nano level, colors are really created by the wavelength of the light waves that are reflected back after bouncing off objects. Show students this graphic or a similar one to see what wavelengths correspond to each color.



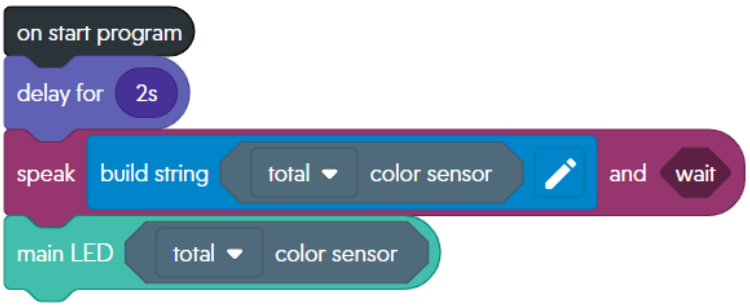
2. Explain that every color can be made by different combinations of the three primary colors: red, green, and blue. This is commonly referred to as RGB. In coding to make different colors, we use a value from 0-255 for each of these three colors to create any color we want. Here are some examples:
   - R=0 G=0 B=0 corresponds to black
   - R=255 G=255 B=255 corresponds to white
   - R=255 G=0 B=0 corresponds to red
   - R=255 G=0 B=127 corresponds to pink

3. Explain that the color sensor on RVR+ works in a similar way. It shines light onto objects and uses the reflection it gets back to interpret what color it is seeing. RVR+'s color sensor is located on the bottom of the RVR+. While the color sensor can interpret any color in RGB values, it can also identify a handful of common colors by name. Below is an example of some of the colors RVR+ recognizes, for a full list visit https://sphero.cc/Color-Chart

Light Blue	**R: 175, G: 175, B: 255**
Cyan	**R: 000, G: 255, B: 255**
Purple	**R: 127, G: 000, B: 255**
Magenta	**R: 255, G: 000, B: 255**
Pink	**R: 255, G: 000, B: 127**

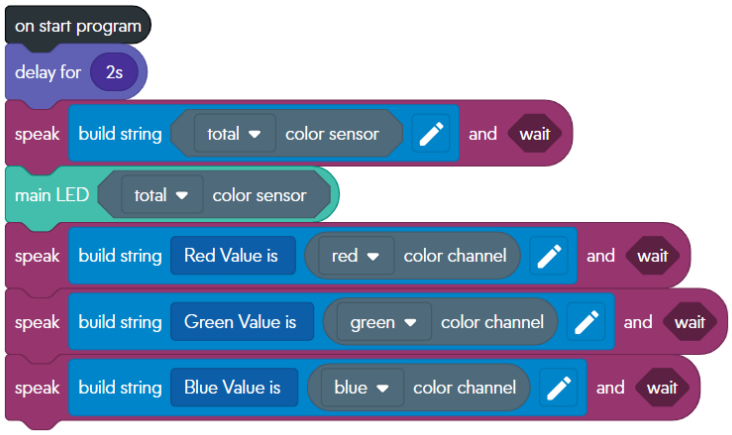
Skills Building

1. Direct students to use the following steps to open and run the program:
   - Scan the QR code to open the program in the Sphero Edu app.
   - Find an object to place RVR+ on. All groups should try different objects with different colors.
   - Have students place RVR+'s color sensor over their chosen object and select **Start**.
2. Discuss as a class what objects groups used and what color RVR+ identified them as. If there were objects that didn't work well, have students find others and run the program again.
3. Remind students that RVR+ is able to identify colors by their RGB values and those values can be used in other ways as well.
   - Have students drag a **main LED block** under the **speak block**.
   - In the sensors tab, have student find the **total color sensor block** and drag it inside the **main LED block.**
   - Have students run their program again with their object and have them observe RVR+'s LEDs.



4. Instruct students to add **three speak blocks** to their program, and to put a **build string operator** in each **speak block**. This will allow them to hear the RGB values of each color once we put some more blocks in.
   - Modify the **build string block** to include a string and then a number.
   - In the string portion of the first **speak block** type "Red value is" then insert the **color channel block** and select "red".
   - Repeat this process for green and blue.
   - Run the program again and listen to the RGB values.

     Are they what you expected?

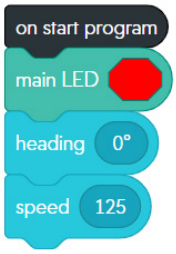


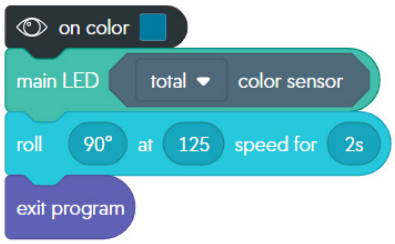
TEACHER TIP: Note that the **color sensor block** will read the name of the color and the **color channel block** will read the numeric RGB value. This may be worth pointing out to students ahead of time or useful when helping them troubleshoot.

If your students have experience with loops, you can extend this portion of the activity by putting all the code in a **loop forever block** and see which team of students can go around the room and get RVR+ to read out the most colors in a set amount of time.

Challenge

1. Discuss with students that RVR+ doesn't just recognize color, it can respond and run commands based on what color the sensor "sees". This part of programming is called **events**.
2. Have students remove the blocks from the previous step from under the **on start program block**. Ensure that each group has a set of color tiles or a variety of colored paper.
   - Choose one color tile or paper.
   - Calibrate RVR+ with the color tile using the **on color event blocks**.
     i. Drag the **on color event** onto your program from the **events** tab.
     ii. Click on the white square and place the RVR+ color sensor over the first color tile.
     iii. Click set color.
   - Connect a **heading** and **speed block** under **on start program** (0° heading, 125 speed).
   - Connect a **roll block** under the **on color event** (90° heading, 125 speed, 2s).
   - Connect a **main LED block** under each **roll block.** Make the color under **on start program** red, and under **on color** match the **total color sensor.**
   - Put the color tile on the floor, and aim RVR+ towards it about 2 ft away.
   - Start the program and watch what happens!





3. Instruct students that they are now going to use all their colored tiles or paper to construct a maze for RVR+. They will need to adhere to the following parameters. Their maze must:
   - use at least four color tiles, and four on **color events**
   - have RVR+ make at least three changes in direction
   - include one color should tell RVR+ to stop

TEACHER TIP: If students are having a hard time getting started, suggest that they make a shape with their color tiles, and then have RVR+ drive in that shape using the tiles to know when to turn.

Extended Challenge

1. For a harder challenge, have each group set up a maze, then swap mazes with another group, and write a program to solve the other group's maze.
   - Ensure each group has a clear starting and ending point.
   - Challenge students to find multiple ways to drive through each maze.

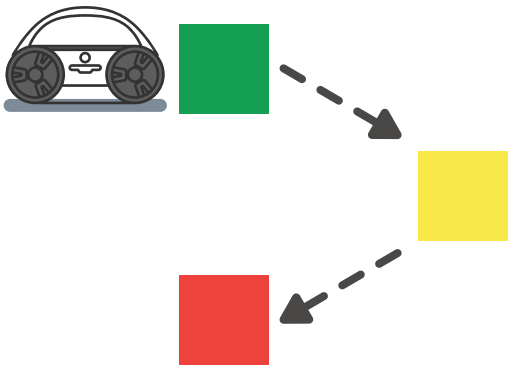
Lesson 4: Color Events

Visible light is made up of waves of varying lengths which the eye detects as color. While RVR+ doesn't have eyes, it does have a sensor that is able to detect a wide range of colors. In this lesson you'll learn the capabilities of RVR+'s color sensor and how to program it.

Learning Objectives

1. I can program with an **on color event**.
2. I can retrieve values from the RVR+ color sensor in the Sphero Edu app.
3. I can explain how the RVR+ color sensor works.

Setup



Program QR Code

sphero.cc/RVR4

Exploration:

Review the basics of light and color, and learn about RGB values. Review the colors that RVR+'s color sensor can recognize.

Skills Building:

Execute a block program that uses the values read by RVR+'s color sensor to "see" colors. Add blocks to the program to have RVR+ change colors based on the color it is seeing, and then read the RGB values out loud. *Can you predict what values RVR+ will see on different items?*

Challenge:

Use events and color tiles to make a maze for RVR+ to navigate. By placing color tiles in a certain place and programming RVR+ with **event** and **roll blocks** you can get RVR+ to solve even the most challenging maze.

Extended Challenge:

- Swap mazes with another group and see if you can use your programming skills to solve each other's maze.

Lesson 5: Automatic Headlights

Overview
If it is too dark to drive, RVR+ needs to turn on its headlights. In this lesson, students use a function, a comparator, and an **if else block** to turn on RVR+'s LEDs when the robot detects light levels below a certain threshold.

Level
Intermediate

Learning Objectives
1. I can program a RVR+ to turn on its headlights when the light level is below a certain threshold.
2. I can use a comparator to form a condition in an **if else block**.
3. I can create and call a function in a program.

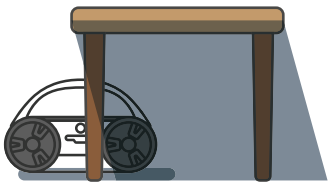
Vocabulary
- **Comparator**: Allows you to define conditions by comparing two values (usually a sensor to a number)
- **Function**: Groups of blocks or statements that can be called, or reused, throughout a program
- **Luminosity**: The light intensity from 0–100,000 lux, where 0 lux is full darkness and 30,000 to 100,000 lux is direct sunlight

CS Standards
- 2-DA-08 Collect data using computational tools and transform the data to make it more useful and reliable.
- 2-AP-12 Design and iteratively develop programs that combine control structures, including nested loops and compound conditionals.
- 2-AP-14 Create procedures with parameters to organize code and make it easier to reuse.

Content Connections
- Computer science

Materials
- RVR+
- Programming devices with the Sphero Edu app installed

Preparation
- Students will be programming RVR+'s path around your learning space to test and respond to light levels. Prepare your classroom with a variety of different light levels.
- Use RVR+'s light sensor to test light levels around your classroom.
- Preview and prepare to share the student instructions (page 54).

Setup


Program QR Code
sphero.cc/RVR5

Exploration

1. Tell students that they will jump right into the code in this lesson. Prompt students to do the following:
    i. **Scan the QR code** to open the program.
    ii. **Connect their RVR+** to their programming device.
    iii. Select **Start**.
    iv. Walk around the learning space carrying their RVR+ and listen to the speaker on the programming device.
2. Give students 2-3 minutes to walk around and explore.
3. Discuss with students:
    i. *What type of information is RVR+ gathering?*
    ii. *What was the lowest value detected? The highest?*
    iii. *Which areas of the classroom corresponded to low values? High values?*
4. Explain the program is using a speak block to read the light level, or luminosity, detected by RVR+'s onboard light sensor.
    i. **Luminosity is the light intensity from 0-100,000 lux, where 0 lux is full darkness and 30,000 to 100,000 lux is direct sunlight.**
    ii. The range in the majority of settings is much more narrow. Your students should already know the range in the classroom from the exploration exercise.
5. Tell students that, in this lesson, they will use the light sensor to automatically turn on RVR+'s headlights when it is dark.

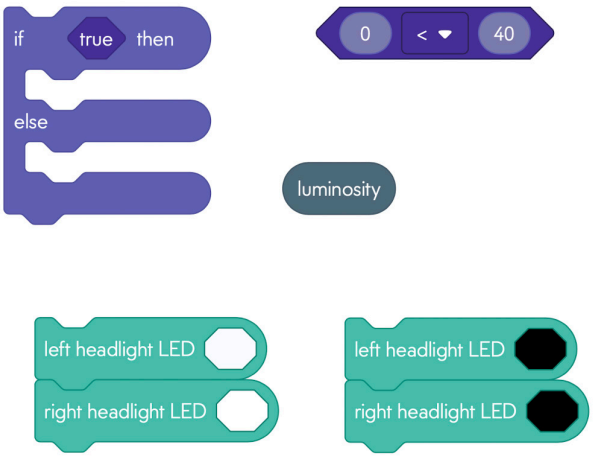
TEACHER TIP: As time allows, encourage students to remove the RVR+ cover and look for the light sensor on the RVR+ circuit board. It is located on the circuit board behind the front right headlight. It is found inside the cylinder post that is just underneath the cutout on your RVR+ cover plate.

Skills Building

1. Together with students, analyze the small number of blocks in the program.
   - *Which blocks are attached to on start program?*
   - *What is the green define checkLights block off to the side?*
2. Explain that the green block is a **function**.
   - **Functions are groups of blocks or statements that can be called, or reused, throughout a program**.
   - The **define checkLights block** is where the programmer snaps blocks that will be executed when the function is called in the main program.

   - The **green checkLights block** is used in the main program to call or execute the blocks.

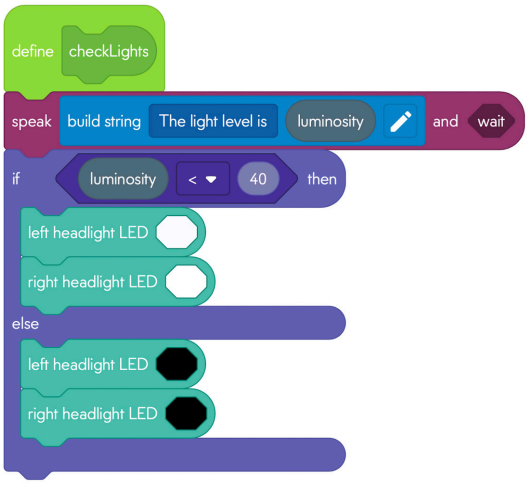
3. Direct students to look at the scattered blocks underneath the **checkLights block**.



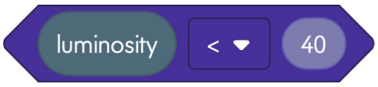
   - Tell students that when these blocks are assembled correctly and attached under the **speak block**, RVR+ will **turn on the headlight when the luminosity is less than 40** and **turn off the headlights when it is not**.

- Give students approximately five minutes to explore and try to figure out the solution on their own.

4. Review the solution with students.



5. Draw students' attention to the purple hexagonal **comparator block** in the top of the **if else block**.



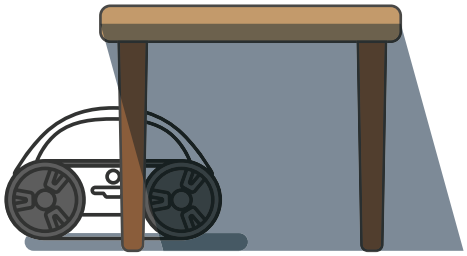
- Programmers use **comparators to define conditions by comparing two values, in this case the luminosity to a number**.
- In this case, if **luminosity is less than 40**, RVR+'s headlights turn on. If **luminosity is not less than 40**, the headlights turn off.

TEACHER TIP: The comparator checks if luminosity is less than 40. Make sure to revise the 40 lux value for the light levels in your learning space. You may need to adjust it up or down depending on your current conditions.

Challenge

1. Give students a programming challenge:

   Program RVR+ to roll around the room, check for light levels, and automatically turn on and off its headlights three times.



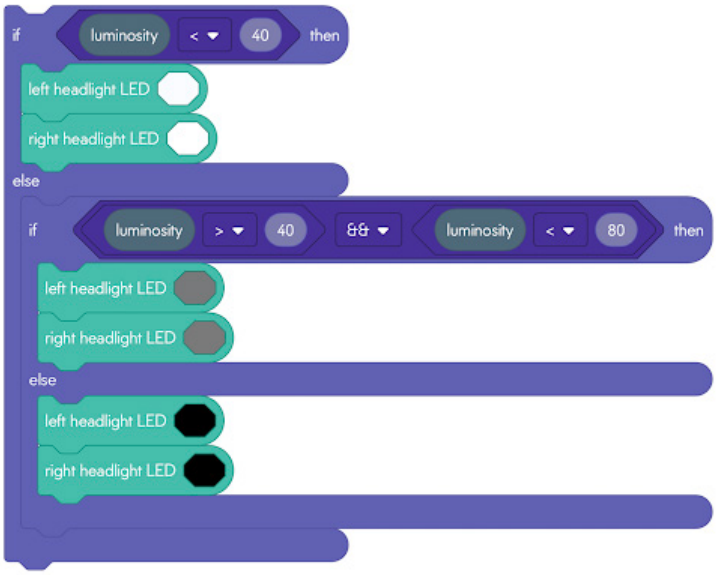
2. As needed, provide the following tips:
   - Remove the **loop forever block** from **on start program**.
   - Students only need **roll blocks**, **delay blocks**, and the **checkLights function block**.
   - Insert a **checkLights function block** at the end of every **roll block**.

Extended Challenge

1. Once your students have programmed RVR+'s path through the light and dark regions of your classroom, they'll be ready for more. Suggest one of the following challenges:
   - Add a "turn headlight on" sound and a "turn headlight off" sound to the **checkLights function**.
   - Figure out to turn your headlights to 50% brightness when light levels are between two levels. Hint: You'll need a new **if else block**, more **comparators**, and an **AND (&&) logic block**.

TEACHER TIP: Students may need help with the second extended challenge. To accomplish this they will need to nest a second **if else block** inside of the **else condition** of the first **if else block** and then create a **comparator** to test if the light level is both **greater than one threshold AND (&&) less than another threshold**.

Possible Solution:



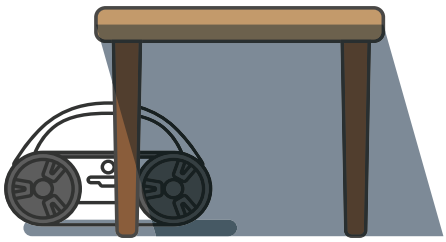
Lesson 5: Automatic Headlights

If it is too dark to drive, RVR+ needs to turn on its headlights. In this lesson, you'll use a **function**, a **comparator**, and an **if-else block** to turn on RVR+'s LEDs when the robot detects light levels below a certain threshold.

Learning Objectives

1. I can program a RVR+ to turn on its headlights when the light level is below a certain threshold.
2. I can use a **comparator** to form a condition in an **if else block**.

Setup



Program QR Code

sphero.cc/RVR5

Exploration:

Open the program, select **Start**, and use RVR+ to check the light levels around your classroom. *What are the highest and lowest levels that you can find?*

Skills Building:

Assemble the blocks underneath the define **checkLights function** on the block to make RVR+ turn on its headlights when light levels are below 40 and turn off its headlights when light levels are not below 40.

Challenge:

Program RVR+ to roll around the room, check for light levels, and automatically turn on and off its headlights three times. You only need to use **roll blocks**, **delay blocks**, and **checkLights function blocks**.

Extended Challenge:

Ready for more? Try some of the following challenges:

- Add a "turn headlight on" sound and a "turn headlight off" sound to the checkLights function.
- Figure out how to turn your headlights to 50% brightness when light levels are between two levels. Hint: You'll need a new **if else block**, more **comparators**, and an **AND (&&) logic block**.

Lesson 6: Ramping Up

Overview

RVR+ is your tough and tenacious tankbot that can drive on not only flat surfaces but up ramps and inclines! In this lesson, students will explore how RVR+'s IMU sensor can measure its location and distance along all three axes (X, Y, Z) and how we can use that to help design accessible ramps.

Level

Intermediate

Learning Objectives

1. I can explain the difference between roll, pitch, and yaw on RVR+.
2. I can use RVR+'s IMU sensor to gather data.

Vocabulary

- **Pitch**: Rotation about the lateral axis that is made by drawing a line from one wheel tread to another, i.e. the forward and backward tilt of RVR+
- **Roll**: Rotation about the longitudinal axis that is made by drawing a line from the front to back of RVR, i.e. the left and right tilt of RVR+
- **Yaw**: Rotation about the vertical axis that is made by drawing a line from top to bottom through RVR+, i.e. the clockwise and counterclockwise spin/twist of RVR+

CS Standards

- 1B-DA-07 Use data to highlight or propose cause-and-effect relationships, predict outcomes, or communicate an idea.
- 2-CS-02 Design projects that combine hardware and software components to collect and exchange data.
- 2-AP-17 Systematically test and refine programs using a range of test cases.

Content Connections

- Engineering

Materials

- RVR+
- Programming devices with the Sphero Edu app installed
- Materials that RVR+ can drive on that can create an incline; Examples include binders, textbooks, plywood, or thick cardboard

Preparation

- Plan for student work areas around your classroom. Each student group will need a 3-by-3 meter area of floor space.
- Gather materials for students to construct ramps for RVR+ to drive on.
- Preview and prepare to share the student instructions (page 60).

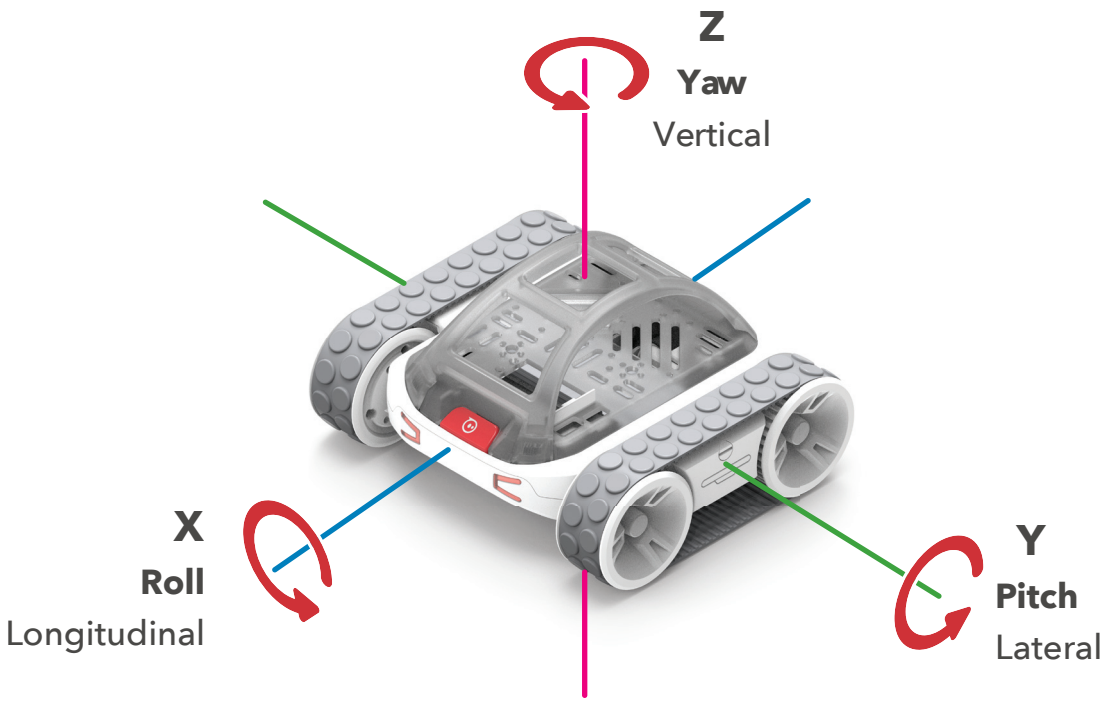
Setup

Program QR Code

sphero.cc/RVR6

Exploration

1. Review the mathematical terms for the three directions of movement. The longitudinal axis typically represents left to right movement, the lateral axis typically represents up and down movement, while the vertical represents rotational motion. However when referring to objects moving in these directions the industry standard is to use the terms **roll, pitch, and yaw**. Use the image below (or similar) to help students identify which movement is represented by each term.



2. Instruct students to rotate their RVR+'s to show the movement associated with **roll, pitch, and yaw** and ensure all students are modeling the movement correctly.

Skills Building

1. Explain that students will specifically be looking at **pitch** in this activity, which will measure the angle of inclines that RVR+ is driving up or down. The program they are going to run in the next step will speak the angle of the **pitch** every two seconds, five times in a row. They will use this program to get an idea of what different **pitches** look like.
   - Instruct students to open the program associated with the lesson.
   - Connect the RVR+ to the programming device and while the RVR+ is on the ground select **Start**.

- Once the program has read the **pitch**, pick up the RVR+ and tilt it forward, barkwards, upside down, etc. and listen to the values the program returns for the **pitch**.

2. Discuss the following with students:
   - *When was the pitch angle positive?*
   - *When was the pitch angle negative?*
   - *What values did you notice when RVR+ was upside down?*
3. Instruct students to use the materials provided to construct a ramp for RVR+ to drive up. The ramp does not need to be any specific length or height.
4. Have students measure the angle of the ramp by setting RVR+ on the ramp and running the program again. Have them record their **pitch**.
5. Instruct students to do the following to create another program:
   - Drag the blocks off the **on start program block**, but don't delete them.
   - Drag a **drive block** on the **on start program block**.
   - Modify the **drive block** so that RVR+ consistently drives up the ramp and stops at the top under the following 2 conditions:
     i. The slowest speed you can get RVR+ to move and still climb the ramp.
     ii. The fastest speed you can get RVR+ to climb the ramp and reliably stop at the top
6. Discuss the following as a class:
   - *Which group had the largest angle? What was their minimum speed compared to other groups?*
   - *Which group was able to get the fastest reliable speed? What was the angle of their ramp compared to other groups?*

TEACHER TIP: Consider making a class chart showing the ramp angle, minimum speed, and maximum speed of each group and displaying it throughout the activity. You would expect that groups with a higher pitch, will also have a higher minimum speed.

Challenge

1. Introduce a design challenge:

   Design and build a ramp that is within Americans with Disabilities Act (ADA) standards, and find a safe traveling speed for RVR+ to ascend and descend the ramp.

2. Provide students with background and constraints:
   - The ADA ensures that people with disabilities have access to the same goods, services, and treatment as all American citizens. One of the accommodations included in this act is that businesses and institutions must have a ramp to allow wheelchairs to access any place that needs stairs to get to.
   - The ADA specifies that the ramps must have a 1:12 slope ratio.
3. Tell students that once their ramp is complete, they should measure the pitch angle of the ramp and create a program that has RVR+ drive up the ramp, turn around, and drive back down the ramp at what you would consider a safe speed.

TEACHER TIP: If you are short on materials to build ramps, consider building one ramp for the entire class to use that has the ADA specified slope.

Extended Challenge

If you have extra time, consider visiting ramps throughout your school and measuring the pitch of RVR+ when it is sitting on the ramp. You can also bring a tape measure with you and have the students calculate the slope of the ramp to ensure it meets the 1:12 slope ratio standards.

Lesson 6: Ramping Up

RVR+ is your tough and tenacious tankbot that can drive on not only flat surfaces but on inclines as well! In this lesson, you will explore how RVR+'s IMU sensor can measure its location and distance along all three axes (X, Y, Z) and how we can use that to help design accessible ramps.

Learning Objectives

1. I can explain the difference between roll, pitch, and yaw on RVR+.
2. I can use RVR+'s IMU sensor to gather data.

Setup

Program QR Code

sphero.cc/RVR6

Exploration:

Explore the terms roll, pitch, and yaw with your classmates. Use the QR code to open the program, select **Start**, and use RVR+ to learn more about pitch and what values it can have.

Skills Building:

Construct a ramp out of the provided materials and use RVR+ to measure the angle of the ramp. Write a program to have RVR+ drive up the ramp at the slowest and fastest speeds possible. Have students compare their results.

Challenge:

Construct a ramp that meets the Americans with Disabilities Act (ADA) criteria for accessibility, and program RVR+ to drive up, turn around, and drive back down.

Extended Challenge:

Have some time to explore the school? Take RVR+ with you and find ADA ramps in your school. Set RVR+ on each ramp and have it read the pitch to you. *Do the ramps in your school meet ADA criteria?*

Lesson 7: Engineer an Apple Picker

Overview
RVR+ is more than a programmable robot, it has a payload capacity of 1 kilogram! In this activity RVR+ will turn into an apple picking vehicle. First, students will build a mechanism on top of RVR+ that will collect the apples, and then they will program RVR+ with JavaScript to pick up and deliver the apples so they can be packaged.

Level
Advanced

Learning Objectives
1. I can program RVR+ with JavaScript on a Text Canvas.
2. I can make RVR+ drive using await roll commands.

Vocabulary
- **Text Canvas**: The space provided in the Sphero Edu app for writing and executing JavaScript programs
- **drive to distance**: The test-based command equivalent of a roll block

CS Standards
- 1B-AP-15 Test and debug (identify and fix errors) a program or algorithm to ensure it runs as intended.
- 2-AP-17 Systematically test and refine programs using a range of test cases.

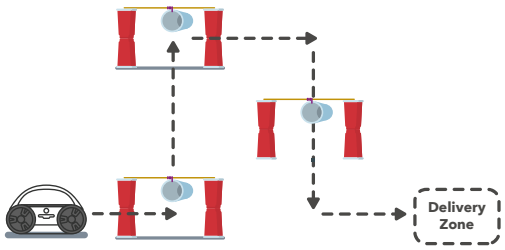
Content Connections
- Engineering

Materials
- RVR+
- Programming devices with the Sphero Edu app installed
- Supplies to build arches and mechanisms on RVR+

Arches - Each group will need:
- 12 - 16 oz. cups
- 3 - 5 oz. cups
- 3 pipe cleaners
- 3 cross bars (Anything sturdy will do, cardboard, foam board, craft sticks)

Suggested Materials for RVR+ mechanisms
- Craft sticks
- Pipe cleaners
- Cardboard or foam board
- Tape
- Cable ties
- Paper clips
- Other craft materials

Preparation
- Plan for student work areas around your classroom. Each student group will need a 3-by-3 meter area of floor space.
- Gather materials needed and build a sample tree and apple model so studdents can reference it.
- Preview and prepare to share the student instructions (page 65).

Setup


Program QR Code
sphero.cc/RVR7

Exploration

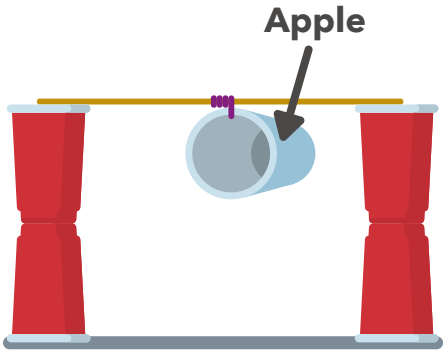
1. Introduce the different top plates RVR+ came with. Ensure students know how to swap them, and discuss the pros and cons of building a mechanism on top of each. Examples may include:
    i. **Cover Plate** - Flat surface, but nothing to solidly attach to
    ii. **Developer/Mounting Plate** - Flat surface, includes cutouts to easily attach mechanisms
    iii. **Roll Cage** - Provides additional sturdy height, could get in the way if RVR+ is driving under objects
2. Show students their available supplies.
3. Allow students a few minutes to brainstorm how that would attach and build a mechanism on top of RVR+ with the provided supplies.

Skills Building

1. Instruct students to open the program associated with this activity.
    - Students will see one line of code `drive to distance(0, 150, 2)`.
    - Explain to students that this is a JavaScript version of a roll block and the numbers represent the same values in the same order as a roll block, so the command is written as `drive to distance(heading value, speed value, distance value)`.
    - Instruct students to run their program to see what RVR+ does.
2. Instruct students to add three lines of text code to make RVR+ drive in a square and to test their code until they are successful.
    - If students need support, you can explain that their program will be four total lines of code that are all the same except the heading changes: 0 → 90 → 180 → 270.

Challenge

1. Ask students to extend the game area one more level.
   - Set up your trees and apples using the picture.
   - Build a mechanism on top of RVR+ to retrieve the apples from each tree.
   - Program RVR+ with JavaScript to navigate the course and retrieve the apples.



Extended Challenge

1. Have more time? Try some of these ideas to challenge students further:
   - Increase the height of the apples by one cup.
   - Set up the course differently so that none of the turns are 90 degree turns.
   - If students have text-based programming experience, see if they can program RVR+ to navigate the course using the distance sensor instead of roll commands.

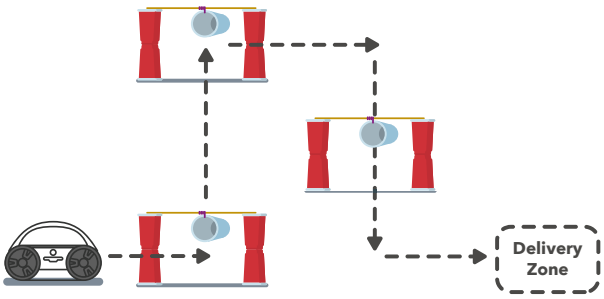
Lesson 7: Engineer an Apple Picker

RVR+ is more than a programmable robot, it has a payload capacity of 1 kilogram! In this activity RVR+ will turn into an apple-picking vehicle. First, you will build a mechanism on top of RVR+ that will collect the apples, and then you will program RVR+ with JavaScript to pick up and deliver the apples to get them packaged up.

Learning Objectives

1. I can create a text programming canvas.
2. I can make RVR+ drive using drive to distance commands.

Setup



Program QR Code

sphero.cc/RVR7

Exploration:

Investigate RVR+'s top plates and roll cage. Brainstorm ways that you would be able to attach a mechanism to the top of RVR+ with the materials provided by your teacher.

Skills Building:

Learn how to create a text-based program using JavaScript in the Sphero Edu app. Make RVR+ drive in a square using JavaScript commands instead of blocks.

Challenge:

Build a mechanism on top of your RVR+ that grabs the apples from the trees. Create a JavaScript program that ensures RVR+ collects and delivers the apples to the delivery zone.

Extended Challenge:

If you have more time, try some of the following:

- Increase the height of the apples by one cup.
- Set up the course differently so that none of the turns are 90 degree turns.
- *Experienced with JavaScript?* Program RVR+ to navigate the course using the distance sensor instead of drive to distance commands.

Lesson 8: Variable Movement

Overview

RVR+ comes with an Inertial Measurement Unit (IMU) that measures everything you could want to know about RVR+'s movement including location, distance, velocity, acceleration, and more. In this lesson, students will use the IMU sensor with loops and variables to experiment with RVR+'s movement.

Level

Advanced

Learning Objectives

1. I can create a variable and use it in a program.
2. I can use a loop in a program.
3. I can access data from RVR+'s IMU sensor and use it in a program.

Vocabulary

- **Variable**: Information—numbers, strings, boolean values, and colors—that can change during a program
- **Loop**: The action of repeating over and over again, either indefinitely, for a set number of times, or until a condition is met

CS Standards

- 1B-AP-09 Create programs that use variables to store and modify data.
- 1B-AP-10 Create programs that include sequences, events, loops, and conditionals.
- 2-AP-11 Create clearly named variables that represent different data types and perform operations on their values.

Content Connections

- Science
- Computer science

Materials

- RVR+
- Programming devices with the Sphero Edu app installed

Preparation

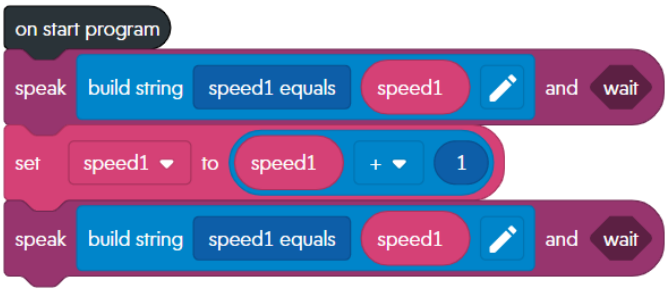
- Plan for student work areas around your classroom. Each student group will need a 3-by-3 meter area of floor space.
- Preview and prepare to share the student instructions (page 71).

Program QR Code

sphero.cc/RVR8

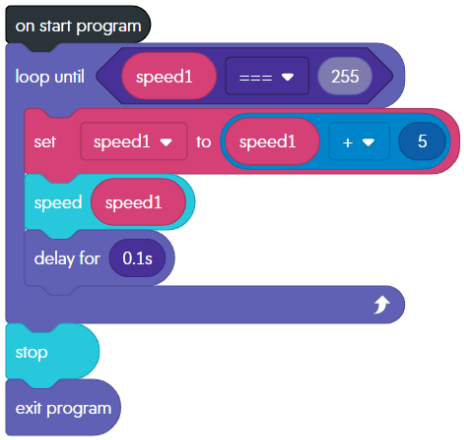
Exploration

1. Introduce students to the concept of variables. A variable defines a value that can be used throughout a program and can be used to do things like count or compare values. There are four types of variables: number, string, boolean, and color. You must declare what type of variable you want when you create the block and you can not change it later.
2. Have students scan the QR code to open a blank Blocks canvas in the Sphero Edu app and walk them through the steps of creating a variable:
    i. Scroll to the Variables tab at the bottom of the canvas.
    ii. Select Create a Variable.
    iii. Type in Variable name "speed1".
    iv. Select number as the variable type.
    v. Select the check mark.
3. Have students look at the Variable tab now, they should see a block for their variable and a block that they can use to set their variable to a value.
4. Instruct students to drag a **speak block** under on start program and a **build string block** inside of it.
    - Edit the **build string block** by pressing the pencil icon and add a number.
    - Type "speed1 equals" into the string and drag the **speed1 variable block** into the number.
    - Drag the **set speed1 block** in next, and drag in the **addition operator block**. Set it to equal speed1 + 1.
    - Duplicate the **speak block** and place the second **speak block** under the **set block**.
    - Run the program and listen to the values of your variable at the beginning and end of the program.



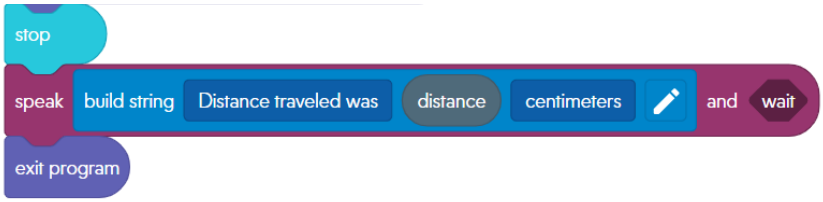
Skills Building

1. Instruct students to drag these blocks off the on start program block, and explain that students are going to make a program that directs RVR+ to increase its speed using the speed1 variable and a loop.
   - Drag a **loop until block** under on start program.
   - Drag a **comparator block** into the loop and set it to speed1 === 255.
   - Inside the **loop until block**, drag in the following blocks:
     - Set speed1 to speed1 + 5
     - **Speed block** with **speed1 variable block** inside
     - Delay for 0.1s
   - After the **loop until block**, drag a **stop block** and **exit program block**.



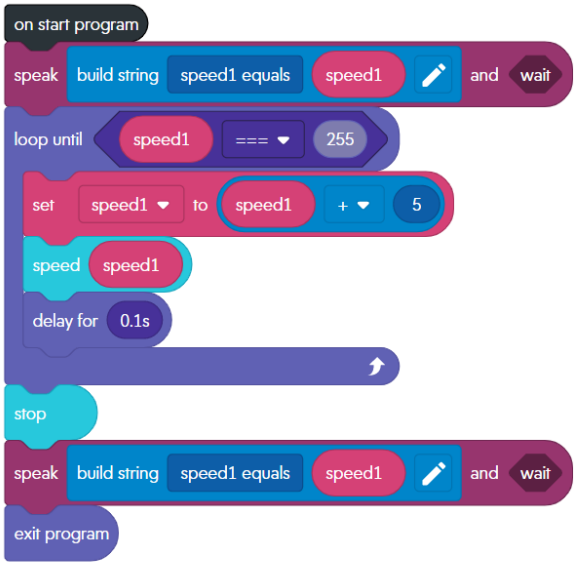
2. Have students put their RVR+ on the floor with about 6 ft. of clear space in front of it and run the program. Discuss the following:
   - Did RVR+ do what you expected? If not, what did you think it was going to do?
   - What role did the variable play in this program? Was it used to compare values? Was it used to count? Was it used for both?
3. Remind students that RVR+'s IMU sensor measures all sorts of data about RVR+'s movement, including how far it travels. Have students drag a **speak block** between the stop and exit program blocks.
   - Drag a **build string** in the **speak block** and edit it to have a string, number, and string.

- Enter the following values:
  - String 1: "Distance traveled was"
  - Number: Insert **distance block** from the sensors tab
  - String 2: "centimeters"



4. Run the program again and have groups compare the distance values to each other. Consider making a chart in a visible area of the classroom.

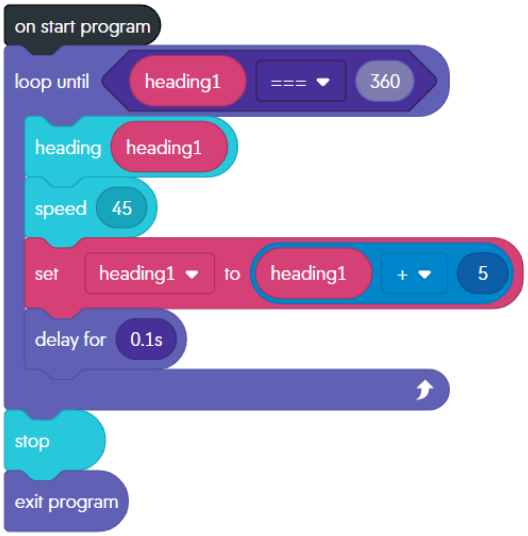
TEACHER TIP: If you feel like your students need more understanding of what the value of the speed1 variable is throughout the program, have them add the speak blocks from the Exploration step between the stop and exit program blocks.



Challenge

1. Challenge students to create a program that makes RVR+ drive in a circle by creating a variable for the heading and increasing it inside a loop.

TEACHER TIP: If students are struggling see the sample code below.



Extended Challenge

1. Do you have extra time or more experienced programmers? Try adding a variable for color and increasing or decreasing the RGB value as the loop runs.

Lesson 8: Variable Movement

RVR+ comes with an Inertial Measurement Unit (IMU) that measures everything you could want to know about RVR+'s movement including location, distance, velocity, acceleration, and more. In this lesson, you'll use the IMU sensor with loops and variables to experiment with RVR+'s movement.

Learning Objectives

1. I can create a variable and use it in a program.
2. I can use a loop in a program.
3. I can access data from RVR+'s IMU sensor and use it in a program.

Program QR Code

sphero.cc/RVR8

Exploration:

Learn what a variable is and how to use it in the Sphero Edu app. Create a program that defines a variable, changes its values, and speaks its value.

Skills Building:

Take your knowledge one step further by using a variable inside a loop to incrementally increase the speed that RVR+ is driving. Also use data from the RVR+'s IMU sensor to see how far RVR+ has traveled.

Challenge:

Now that you have used variables and loops to increase RVR+'s speed, see if you can make a program that makes RVR+ drive in a circle by creating a variable for the heading and increasing it inside a loop.

Extended Challenge:

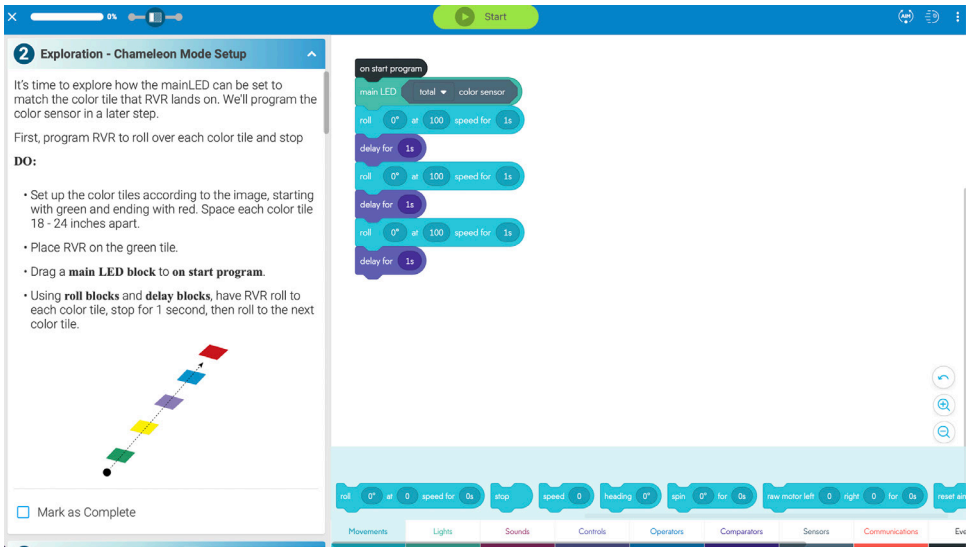
Do you have extra time or more experienced programmers? Try adding a variable for color and increasing or decreasing the RGB value as the loop runs.

Go Further

The Sphero Edu App

After your students have completed the lessons in this guide, they'll be ready to continue learning with other activities in the Sphero Edu app.

Activities in the Sphero Edu app allow students to follow media-rich instructions while programming their RVR+ on a Block or Text Canvas in a side-by-side environment.



Teachers logged in with an educator account can use the educator notes to support students as they work through an activity.

EDUCATOR TIP:

Although we didn't cover it in this activity, the RVR also has capabilities to use Infrared communications. This allows it to communicate to other RVRs or BOLT robots. This also gives an integration to using littleBits and the IR Trigger. Previous activities have alluded to this, and it's a great opportunity to showcase another capability of RVR if the students want to engage in another sensor block with RVR.

Whether your students are entirely new to computer science or ready for a more advanced challenge with RVR+, the Sphero Edu app can help them on their programming and engineering journeys. Refer to the RVR+ & littleBits section of the Getting Started with RVR+ Activity Guide to find the right activities for your students.

sphero.cc/RVR-Lessons

RVR+ & Sphero littleBits

Integrating littleBits is the easiest way to expand on RVR+ and get started with teaching and learning engineering principles and design. littleBits are colorful modular electronic components that magnetically snap together to build fully-fledged inventions. The Bits help make learning about circuits and electronics hands-on and fun. Utilize RVR+'s onboard power source to power all student littleBits inventions and make them mobile!

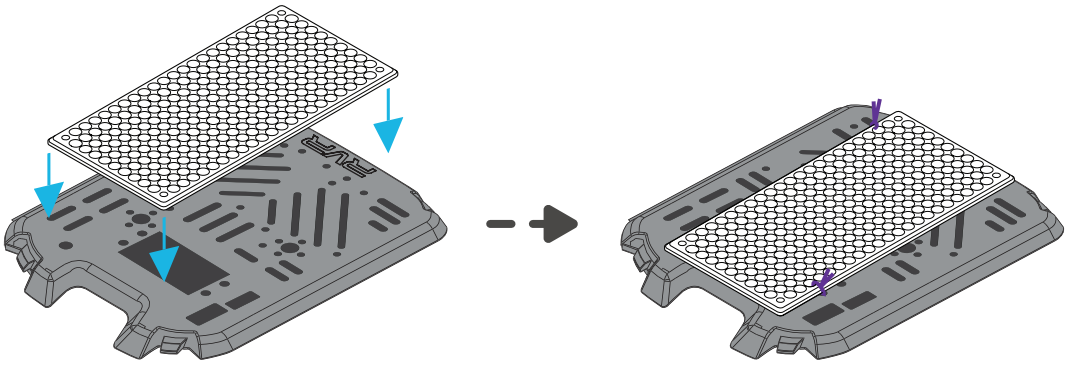
littleBits RVR+ Topper Kit

The littleBits topper kit comes with 14 individual Bits and nine accessories. Whether building inventions on top of RVR+ or off to the side, the kit combines the best of littleBits to maximize the utility of RVR+'s mobility and onboard sensors capabilities. Connect littleBits' newest Bit—the micro:bit adapter—to program both littleBits and RVR+ in Microsoft MakeCode. micro:bit is not included and not sold by Sphero.

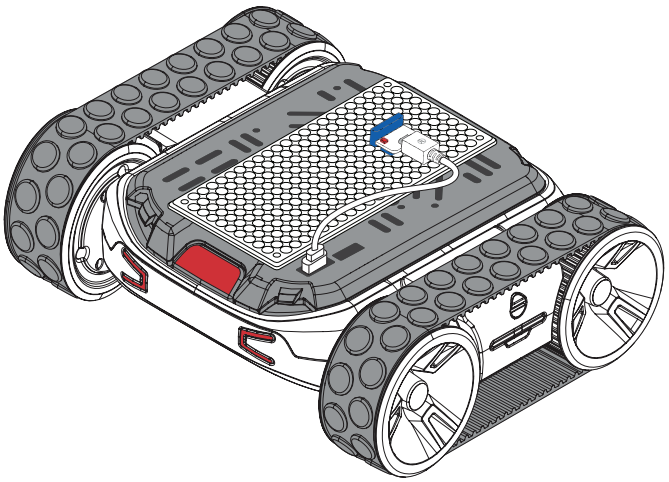
Learn more: sphero.cc/RVR-Topper

Getting Started

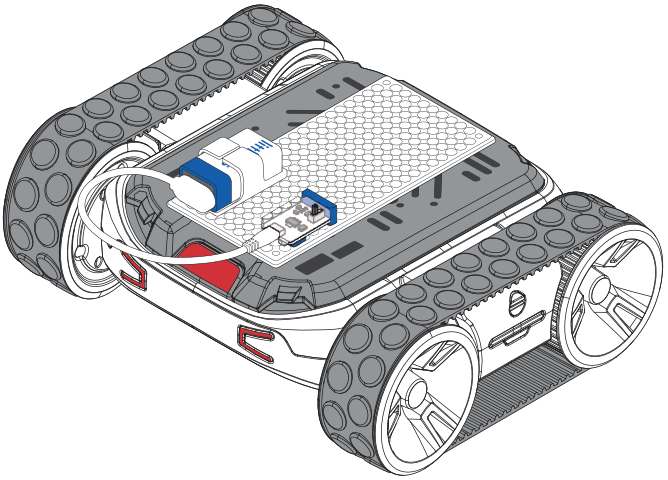
1. Attach the littleBits mounting board to the RVR+ mounting plate with twist ties.



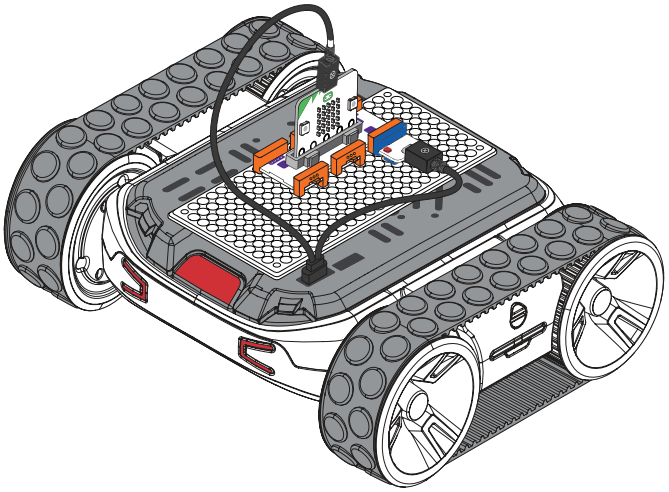
2. Power the littleBits.



- Use the p3 power Bit or RVR+'s USB port.



- Use the p7 power Bit and a 9v battery.



- Use the micro:bit adapter and p3 + USB input from micro:bit.

3. Invent and build with littleBits and RVR+.

Next Steps

Looking for lessons to use with students, refer to the RVR+ & littleBits section of the Getting Started with RVR+ Activity Guide.

sphero.cc/RVR-Lessons

RVR+ & micro:bit

RVR+ is an expandable platform, featuring 4-pin UART connectors and a USB port. Adding a BBC micro:bit to your RVR+ setup is the perfect first step for students to level up their learning.

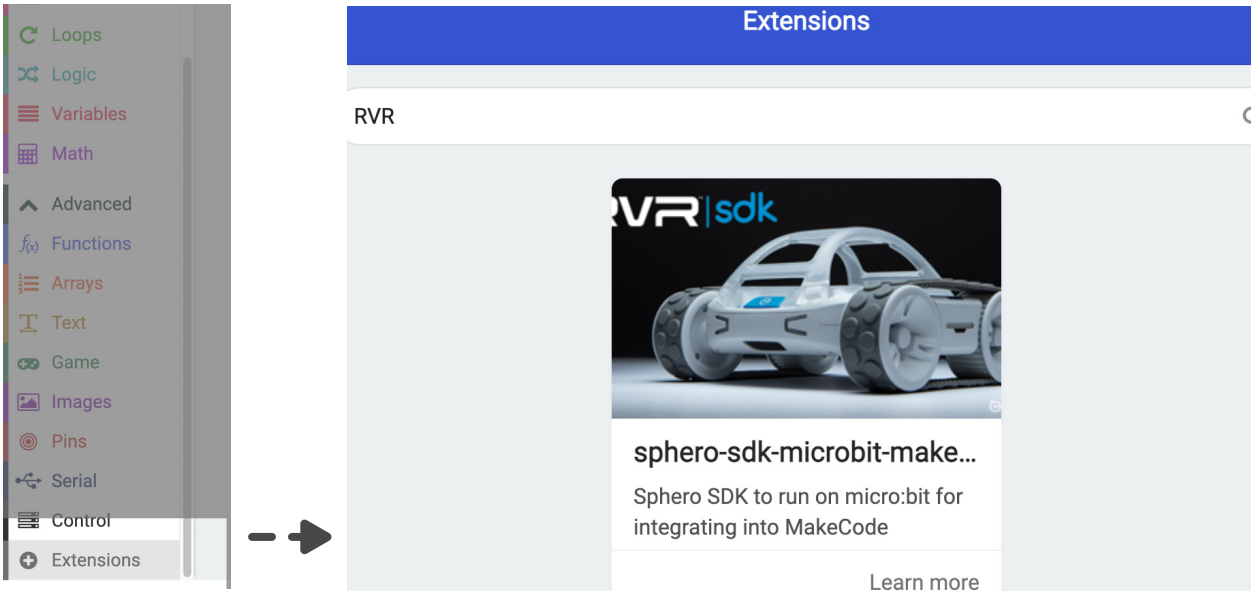
The micro:bit is a tiny computer designed to teach students how to program hardware with blocks, JavaScript, or Python. The circuit board is small enough to fit in the palm of your hand yet includes a 5 × 5 LED display, 3 buttons, a speaker, a variety of sensors, radio communications, pins for inputs and outputs, and more. Fun on its own, micro:bit is even more fun with RVR+.

Micro:bit can be programmed with Microsoft's web-based Make Code editor or in online or installed Python editors.

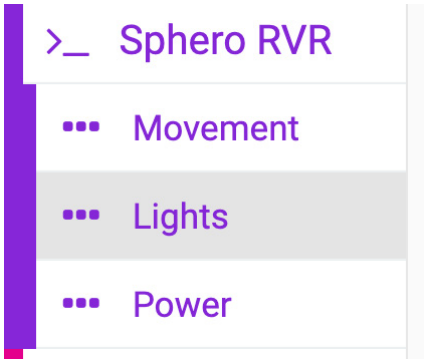
Interested in buying a micro:bit? Visit microbit.org/buy to find a reseller. micro:bit is not sold by Sphero.

Getting Started

1. Go to the MakeCode website: https://makecode.microbit.org.
2. Select "New Project" and name your project.
3. Add the RVR+ extension.
   - Select "Extensions" under the "Advanced" accordion menu.
   - Search for "RVR".
   - Select the Extension to add it to your project.

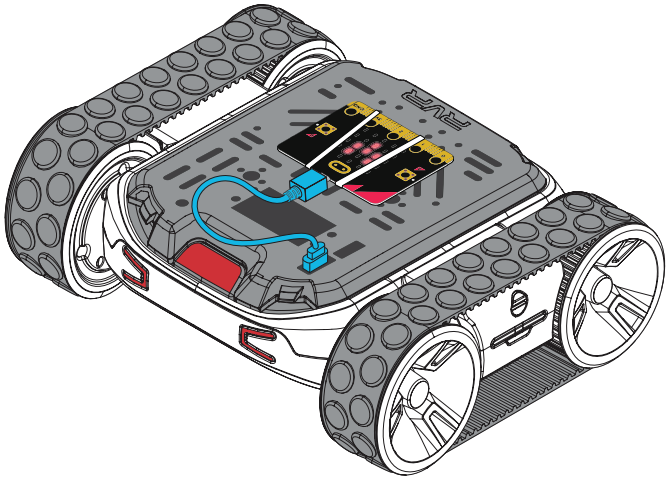


4. Use the blocks under the Sphero RVR+ menu to program RVR+'s movement, lights, and power.



5. When your program is ready, flash—or transfer—the program to your micro:bit.
   - Plug your micro:bit into your programming device.
   - Select "Download."
   - Drag the program to your micro:bit or connect MakeCode to your micro:bit for one click downloads.

6. Connect micro:bit to your RVR+.
   - Secure your micro:bit to the RVR+ mounting plate with twist ties. Don't cover any of the buttons or the 5 × 5 LED matrix on top of the micro:bit.
   - Plug the micro:bit into the USB port of your RVR+ with a USB to micro USB cord Make sure your cord transfers data as well as power.
   - Turn on your RVR+ and watch your program execute!



Example: Thermometer RVR+

Don't want to start a MakeCode project from scratch for RVR+ and micro:bit? Download an already programmed .hex file (sphero.cc/Thermometer-RVR), then import it into MakeCode! The program uses micro:bit's temperature sensor to set the color of RVR+'s LEDs. Remix the program to make it your own. For example, have fun trying some of the following:

- Make RVR+ spin in a circle when the temperature is above the threshold.
- Make RVR+ go to sleep when the temperature is below the threshold.
- Play "awake" and "asleep" sounds from the micro:bit.

Next Steps

Looking for lessons to use with students, refer to the RVR+ & micro:bit section of the Getting Started with RVR+ Activity Guide.

sphero.cc/RVR-Lessons

RVR+ & Raspberry Pi

Once your students have explored what they can do with their RVR+, littleBits, and micro:bit, they'll have the know-how and the inspiration to do a lot more. Connecting a Raspberry Pi to RVR+ with a USB cord and jumper wires is the perfect next step. With a fully fledged operating system—Raspberry Pi OS—students are essentially attaching a powerful and flexible computer on top of the RVR+, opening up endless possibilities with RVR+'s onboard sensors and other third party hardware.

We are looking forward to seeing all the fun you're able to have by connecting your Raspberry Pi to your RVR+.

RVR+ is compatible with a Raspberry Pi 3, 3B+ or Zero W. Interested in purchasing a Raspberry Pi? Visit raspberrypi.com/products/ to find a reseller. Sphero does not sell Raspberry Pi.

Getting Started

The Sphero SDK documentation (sphero.cc/Pi-SDK) will get you up and running with your RVR+ and Raspberry Pi quickly. You can start with a preconfigured version of Raspberry Pi OS with the Sphero SDK already installed or you can add the SDK on your own.

Below is a summary of the steps. Refer to the full SDK documentation at https://sdk.sphero.com/ for further details.

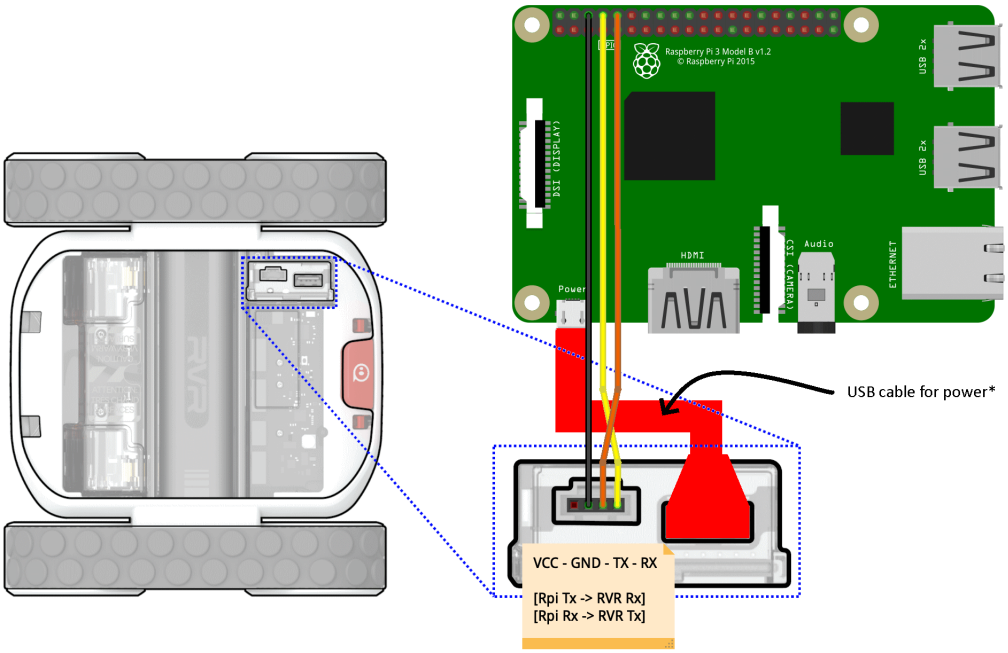
You'll need:

- Raspberry Pi 3, 3B+ or Zero W
- USB keyboard and mouse for the Raspberry Pi
- HDMI monitor and cable to connect it to your Raspberry Pi (Zero W require specialized cables or adapters)
- USB power source and cable for the Pi
- Three female-to-female jumper wires

The steps below outline how to connect your RVR+ to a Raspberry Pi in more detail:

1. Download the Sphero version of the latest Raspberry Pi OS. This version is preconfigured with the Sphero SDK. Write it onto a micro SD card.
2. Insert the card into your Pi, connect your Pi to a monitor, keyboard, and mouse, and set up your Raspberry Pi OS.
3. Open your Raspberry Pi Terminal and learn some bash commands. Developing some fluency with Terminal and bash commands is key for navigating the file structure on your Pi and writing and executing Python programs on RVR+.
4. Set up Secure Shell Protocol (SSH) so you can program your Raspberry Pi and RVR+ from a remote computer.

5. Disconnect your Pi from the monitor, keyboard, and mouse and attach it to the RVR+ mounting board. Connect the Pi ground and serial pins to those on RVR+ as shown in the wiring diagram.



6. Run your first program: set_multiple_leds.py. If the lights on your RVR+ turn red and blue, your setup is working!

Next Steps

Looking for example programs to use as inspiration for your students? Refer to the Python samples in the Sphero SDK documentation (sphero.cc/Pi-Samples) including the following projects:

- Keyboard Control: Drive RVR+ with the WASD keys on your keyboard.
- RVR+ // BOLT: Communicate with BOLT with infrared messages.
- Ultrasonic RVR: Turn your RVR+ into an autonomously driving robot that avoids collisions with ultrasonic sensors.

Resources

Glossary

Heading: Input to program the direction a Sphero robot is pointed during or before a roll, between 0° and 360° (Lesson 2)

Duration: Input to program the amount of time that RVR+ will execute a movement like a roll or a spin in seconds (Lesson 2)

Speed: Input to program how quickly (or slowly) a Sphero robot will move (Lesson 2)

Location sensor: Measures RVR+'s x-axis, y-axis, and total distance from its point of origin, in centimeters (Lesson 3)

String: A sequence of characters that combine letters, numbers, and other characters in a computer program (Lesson 3)

Comparator: Allows you to define conditions by comparing two values (usually a sensor to a number) (Lesson 5)

Function: Groups of blocks or statements that can be called, or reused, throughout a program (Lesson 5)

Luminosity: The light intensity from 0-100,000 lux, where 0 lux is full darkness and 30,000 to 100,000 lux is direct sunlight. (Lesson 5)

Color sensor: Measures the numerical RGB values of any color that is directly underneath the sensor on RVR+ (Lesson 4)

Pitch: Rotation about the horizontal axis that is made by drawing a line from one wheel tread to another, i.e. the tilt of front of RVR+ up and down (Lesson 6)

Roll: Rotation about the horizontal axis that is made by drawing a line from the front to back of RVR, i.e. the rotation of RVR+ when the wheels are driving on uneven surfaces (Lesson 6)

Yaw: Rotation about the vertical axis, i.e. left and right turns (Lesson 6)

Text canvas: A programming canvas in the Sphero Edu app that allows users to program in JavaScript (Lesson 7)

Await roll: The text-based command equivalent of a roll block (Lesson 7)

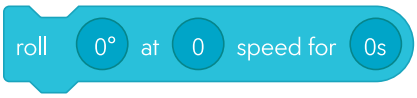
Drive to distance: The text-based equivalent of a drive block (Lesson 7)

Variable: Information—numbers, strings, boolean values, and colors—that can change during a program (Lesson 8)

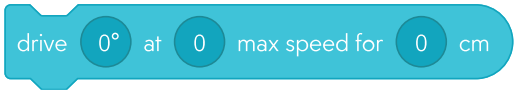
Loop: The action of repeating over and over again, either indefinitely, for a set number of times, or until a condition is met (Lesson 8)

Block Library

Movements



Roll: Combines heading, speed, and duration to make the robot roll.



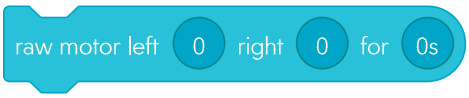
Drive: combines heading, speed, and distance to make the robot drive.

Stop: Sets the speed to 0 to stop the robot when a Speed block is used.

Speed: Sets the target speed of the robot from -255 to 255. Positive speed is forward, negative speed is backward, and 0 is stopped.

Heading: Sets the target direction the robot rolls. Assuming the robot is aimed with the blue tail light facing you, then 0° is forward, 90° is right, 270° is left, and 180° is back.

Spin: Spins the robot for a given number of degrees over time, where 360° is a single rotation.



Raw Motor: This command disables the stabilization and will cause the robot to jump when both motors are set to full power. Controls the electrical power sent to the left and right motors independently, from -255 to 255, for a duration of seconds.

Reset Aim: Resets the heading calibration (aim) angle to use the current front-facing direction of the robot as 0°.

Lights

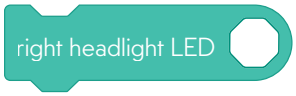
Main LED: Changes the color of the main LEDs. Set this using the color wheel and brightness slider, or the exact RGB (red, green, blue) values, from 0 to 255.

Right LED: Changes the color of the right LED. Set this using the color wheel and brightness slider or the exact RGB (red, green, blue) values, from 0 to 255.

Left LED: Changes the color of the left LED. Set this using the color wheel and brightness slider or the exact RGB (red, green, blue) values, from 0 to 255.

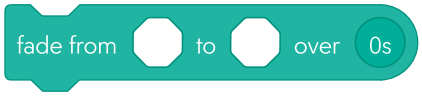
Back LED: Changes the color of the back LED. Turns on the back LED. This is limited to blue only. You can adjust the brightness by tapping on the number.

Front LED: Changes the color of the front LED. Set this using the color wheel and brightness slider, or the exact RGB (red, green, blue) values, from 0 to 255.

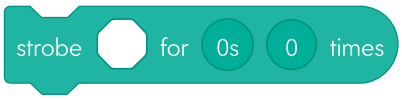


Right Headlight LED: Changes the color of the right headlight LED. Set this using the color wheel and brightness slider or the exact RGB (red, green, blue) values, from 0 to 255.

Left Headlight LED: Changes the color of the left headlight LED. Set this using the color wheel and brightness slider or the exact RGB (red, green, blue) values, from 0 to 255.

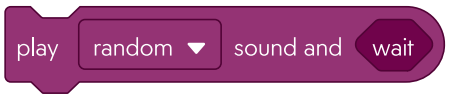


Fade: Changes the main LEDs from one color to another, for a duration of seconds. This command works as a standalone command and cannot be used to fade main LEDs while using a roll command.

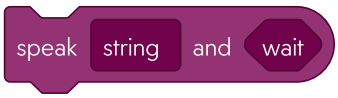


Strobe: Blinks the main LEDs for a period of seconds (that includes lights ON and OFF) and a count of cycles. A short period will produce a fast blink whereas a long period will produce a slow blink.

Sounds



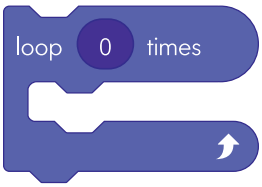
Sound: Plays a sound from your programming device (not from the robot) that you select from the list. Toggle the Randomize option to generate a random sound or select one from Sphero's library.



Speak: Speak strings (numbers and words) from your programming device. Type what you want your robot to say.

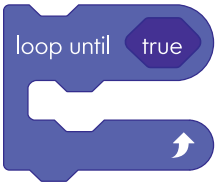
Controls

Delay: Delays execution of the next block for a number of seconds.



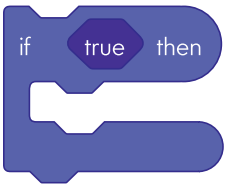
Loop: Repeats the blocks within for the number of loops specified, also known as a "for" loop.

Loop Forever: Repeats the blocks within forever, also known as a "while 1" loop. You can use this to constantly evaluate conditions.

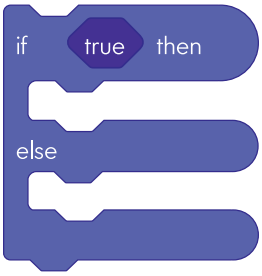


Loop Until: Repeats the blocks within until a condition is met, also known as a "while" loop. Drag any comparator into the "true" field to create a condition. Use "and" and "or" to combine or exclude conditions, which must be added before other comparators.

Exit Program: Stops all code and ends the program; the same as hitting the Stop button.

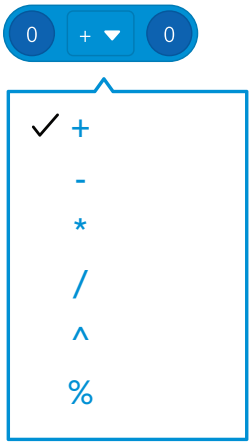


If Then: Calls the "if" blocks if the given condition is true. Use "and" and "or" to combine or exclude conditions, which must be added before other comparators.



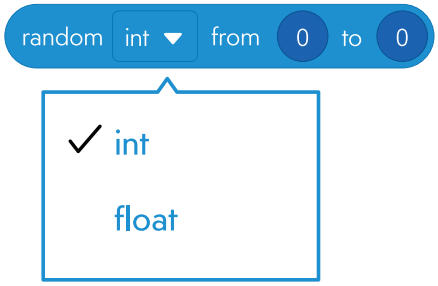
If Then Else: Calls the "if" blocks if the given condition is true; otherwise calls the "else" blocks. Use "and" and "or" to combine or exclude conditions, which must be added before other comparators.

Operators



Basic Operators: Adds a mathematical action to calculate or evaluate a set of data or given value.

Build String: Combines multiple values into a single string. Those values can be numbers (variables, parameters, or sensors), strings, booleans, or colors.



Random Int: Generates a random integer value (nearest whole number) within the given minimum and maximum.

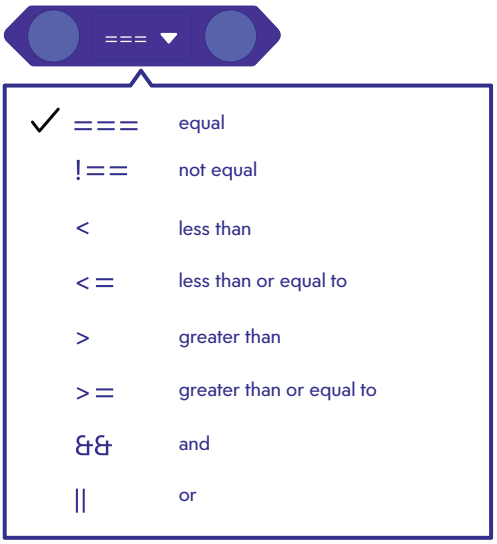
Random Float: Generates a random float value (including decimals) within the given minimum and maximum.

Color Channel: Gets the value of the red, green, or blue color channel for a given RGB value.

Color Mixer: Returns a new color by modifying a single channel (red, green, blue) of a given color.

Random Color: Sets a random color when placed in a color block.

Comparators



Comparators: Comparators are blocks of code that compare two things (Sensor data, operators, strings, etc.). Comparators allow us to create logical checks, which, when "true," will allow the program to perform a certain action or event. If a comparison is true and logical, then the code will be executed.

Sensors

Accelerometer - Total: The accelerometer sensor measures the change in force in the robot.

- Total: The combined vector acceleration of all three axes, from of 0 to 14 g's.
- X-Axis: The left-to-right acceleration, from -8 to 8 g's.
- Y-Axis: The forward-to-back acceleration, from -8 to 8 g's.
- Z-Axis: The up-and-down acceleration, from -8 to 8 g's.

Orientation: The orientation sensor measures the orientation of the robot.

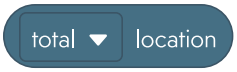
- Pitch: The forward or backward tilt angle, from -180° to 180°.
- Roll: The left or right tilt angle, from -90° to 90°.
- Yaw: The spin (twist) angle, from -180° to 180°.

Gyroscope: The gyroscope sensor measures the rate of spin in the robot in degrees per second.

- Pitch: The rate of forward or backward spin, from -2,000° to 2,000° per second.
- Roll: The rate of left or right spin, from -2,000° to 2,000° per second.
- Yaw: The rate of sideways spin, from -2,000° to 2,000° per second.

Velocity - Total: The velocity sensor measures the speed in centimeters per second.

- Velocity Total: The combined vector speed of both axes which will always be a positive value, in centimeters per second.
- Velocity - X: The right (+) or left (-) speed, in centimeters per second.
- Velocity - Y: The forward (+) or back (-) speed, in centimeters per second.



Location - Total: The distance from the location of the program start, which will always be a positive value, in centimeters.

Distance: The total distance traveled, in centimeters.

Speed: Target speed value, from -255 to 255.

Heading: Target directional angle of the robot. Assuming you aim the robot with the blue tail light facing you, then 0° heading is forward, 90° is right, 180° is backward, and 270° is left.

Main LED: The RGB color of the main LEDs, from 0 to 255 for each color channel.

Time Elapsed: The amount of time with which the program has run (in seconds).

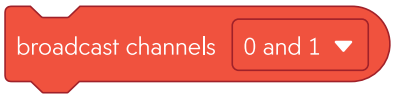
Luminosity: The light intensity from 0 - 100,000 lux, where 0 lux is full darkness and 30,000-100,000 lux is direct sunlight.

Luminosity: Returns which channel the last infrared message was received on.

Color Sensor: Color value from 0 to 255 for all channels (red, green, blue) of the color sensor.

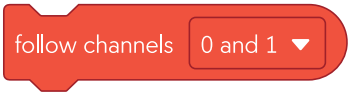
Color Channel: Number value from 0 to 255 for the selected color channel (red, green, or blue) of the color sensor.

Communications



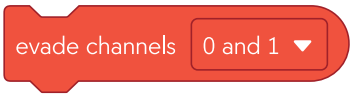
Broadcast: Sets the IR emitters to broadcast on two specified channels, from 0 to 7. The broadcaster uses two channels so the following or evading BOLTs can detect these messages on their IR receivers with a sense of relative proximity to the broadcaster. You can't use a channel for more than one purpose at a time.

Stop Broadcast: Stops the broadcasting behavior.



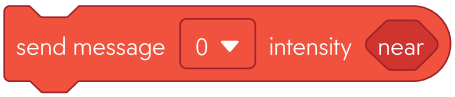
Follow: Sets the IR receivers to look for broadcasting BOLTs on the same channel pair, from 0 to 7. Upon receiving messages from a broadcasting BOLT, the follower will adjust its heading and speed to follow the broadcaster.

Stop Following: Stops the following behavior.



Evade: Sets the IR receivers to look for broadcasting BOLTs on the same channel pair, from 0 to 7. Upon receiving messages from a broadcasting BOLT, the evader will adjust its heading to roll away from the broadcaster.

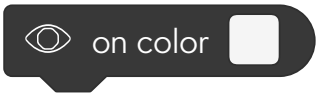
Stop Evading: Stops the evading behavior.



Send Messaging: Sends a message on a given IR channel, from 0 to 7, at a set intensity, from one to 64. Intensity is proportional to proximity, where one is the closest, and 64 is the farthest. You can't overlap sending messages with broadcasting, following, or evading behaviors on the same channels.

Events

On Message Received: Conditional logic called when an infrared message is received on the specified channel.



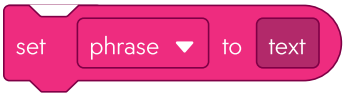
On Color: Conditional logic called when the color sensor reads a given RGB value, defined as 0 to 255 for the red, green, and blue channels. The color for this block is set using the color sensor on the Sphero RVR+.

Variables

Number Variable: A stored number value that can be assigned and operated on. Values can be integers (whole numbers) or floating point values (numbers with decimals).

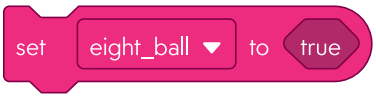
Set Number: Sets the number value for the selected number variable.

String Variable: A stored string value that can be assigned and operated on. Values can be words or sentences, such as "Hello World!"



Set String: Sets the string value for the selected string variable.

Boolean Variable: A stored boolean value that can be assigned and operated on. Values can be true or false.



Set Boolean: Sets the boolean value for the selected boolean variable.

Color Variable: A stored color value that can be assigned and operated on. Values are made up of Red, Green and Blue channels that range from 0 to 255.

Set Color: Sets the color value for the selected color variable.

Functions

Function: Functions allow you to define a reusable group of blocks and then call those blocks from anywhere within the main program. Parameters are like variables, but they are local to the function and help quickly change inputs without modifying the code.

sphero®

RVR+™