FIRST EDUCATION

Robotics Engineering Explorations *FIRST*® Education

FIRST® Robotics Engineering Explorations

Student Guide — Programming Autonomous Robots

Unit 4

TABLE OF CONTENTS

Activity 1: Autonomous Functionality

Driving Questions

- How can we control a robot when it is not driver controlled?
- What do we need to know to make our robot move autonomously?
- How do we write instructions that will make our robot complete a task autonomously?

What Will I Be Doing?

- I will learn about dead reckoning.
- I will write a program that makes my robot move forward and stop without using the gamepad.
- I will use dead reckoning to drive my robot back and forth between two points autonomously.
- I will write autonomous instructions for my robot to complete a task for the ball game.

Getting Started

- So far, you've been using your gamepad to control your robot as it moves around, but most robots aren't operated by remote control. When a robot works without a human's direct control, it works autonomously. Before getting your robot to complete complex tasks, you should know how to create instructions for moving your robot around. In the last unit, you created a set of step-by-step instructions that guided your teacher through a task. In this activity, you will create similar instructions that guide your robot's movement.
- When writing your instructions for your teacher in the previous unit, you didn't have to worry about them running into anything because they could see their environment. When real-world robots complete complex tasks, they usually rely on sensors to guide their movements. The rovers that NASA sends to Mars have sensitive cameras and sensors that allow them to navigate the surface of the red planet without running into rocks or driving into ditches.
- But what would happen if you were driving a complex machine such as a plane and lost your high-tech sensors? How would you know how far away your destination was or how quickly you were going? Plane pilots and ship captains use a technique called dead reckoning to navigate when they can't rely on sensors.
- Research dead reckoning and explain how pilots and sea captains would use the process to navigate in your Engineering Notebook. When you understand how pilots and captains use dead reckoning, think about how to use it to help you create an autonomous program for your robot and record your ideas in your Engineering Notebook.

WHAT'S NEXT?

- Gather your team.
- Grab your supplies (Engineering Notebook, robot, pens and pencils, tape, and laptop)

HOW WILL I DO IT?

- After you've learned about dead reckoning, you will apply the concept to your robot. The first thing you will be doing is moving your robot between two points without using a gamepad.
- After you've created a program that moves your robot autonomously, you will be adding it so your robot can travel from one point to another and then return to the position where it started. After you've created and tested your program, you will take what you've learned and start thinking about the program that would allow your robot to solve your problem autonomously.

Task 1: From Point A to Point B

BRAINSTORM AND EXPLORE:

- You've created a few programs for your robot, but they used your gamepad to drive. Before you start, think about the complex autonomous algorithms you'll need to know to make your robot move from one place to another without using a gamepad. There are a few different ways to get your robot to move autonomously. Before you start building algorithms, think about what each section of your program does and what happens when you start or enable the robot program from the Driver Station.
- An essential part of understanding how to program autonomously is understanding how the program is structured. You may often start with a template or infrastructure that sets up the robot's behavior. In many of the *FIRST*® programming tools, some code contains hardware variables and fail-safes for how the robot communicates with the Driver Station. This section of code tells the robot when to go and stop. It also has sections where you can set data for the robot to start with. This is called initialization, which is helpful for things such as setting a motor direction, resetting a data value, or running in a particular mode.

- Your teacher has provided you with links to programming tools for your robot. Explore the tools to determine how to ensure your robot knows when it is performing the code you designed in autonomous versus driver-controlled mode. Also, look for any examples of how you might create code for the robot to use in initialization.

TEST AND IMPROVE:

- Place your robot on the ground and use tape to mark the position of the front wheels. Then, mark a point around ten feet away. Start a new program and name it “your team’s name Autonomous1.” Find a way to move your robot between the two points as accurately as possible. When you’ve finished constructing your program, save it and test it. If you are encountering errors in your program during your test, use the troubleshooting process and make changes to your code.
- When you are happy with how your program works, record your program in your Engineering Notebook. Remember to save your program.

Task 2: From A to B to A to C

DESIGN AND PROTOTYPE:

- Now that you can make your robot move between two points by itself, it’s time to find a way to give yourself more control. Using dead reckoning, pilots and captains can navigate between locations using information such as known distances, speed, and time. Because you know how to move between your starting and finish points, you already know quite a few things. You know what power and time the motors and wheels might take to move between the two points. Using the information you gathered during the first task, find a way to make your robot move from your start point to the finish point and then back to the start point. When the robot gets back to the start point, find a way to get it to move to a point halfway between the start and finish.
- Using your program tools, choose an algorithm that enables you to set your robot’s motors to run for a time and then stop. Consider how the data might help you accurately determine the distance your robot is traveling. For example, you can determine the distance it will travel for each rotation using the wheel’s circumference. How can knowing this help you improve the accuracy of your dead reckoning distance?

TEST AND IMPROVE:

- Before working on your program, mark the halfway point between your two points using tape. Create a program and use dead reckoning to move your robot from the starting point to the finish point and then back to where you started. When you’ve gotten your robot back to the starting point, get your robot to move to the midpoint.
- Your robot should move between the points in the following order: start point -> endpoint -> start point -> midpoint.
- When you’ve finished building your program, test it out. If your program doesn’t work
- the way you want it to, use the troubleshooting process to make improvements. When you’re happy with how your program works, record a copy of the algorithms you’ve used in your Engineering Notebook.

Task 3: An Autonomous Ball Game Robot

BRAINSTORM AND EXPLORE:

- The autonomous program you’ve written deals with movement in a straight line, but what would you do if you needed to write a much more complex autonomous program? Dead reckoning is only one tool you can use when creating an autonomous program. In this unit, you will learn how to control your motors better autonomously. You will also add two types of sensors to your robot that will provide it with real-time information as it moves.
- When you were designing your ball game robot, you probably pictured scoring points in the game. The ball game task you chose would need an autonomous program that is far more complicated than moving between two set points. What would the autonomous algorithm for your ball game robot look like?

IN YOUR ENGINEERING NOTEBOOK, DOCUMENT AND ANSWER THE FOLLOWING PROMPTS:

- What sensors would your robot need to address your ball game scoring, and how would they be used in an autonomous program?
- What distance would your robot need to move, and how would it know where it was while in motion? How fast would your robot need to move?
- Would you use dead reckoning to help with your robot’s navigation?

Reflection

- How would your ball game robot use dead reckoning?
- How do airplane pilots and ship captains use dead reckoning?
- How did you use dead reckoning in this activity to help your robot navigate?
- Were you able to get your robot to move autonomously the way you wanted it to?
- Is building an autonomous algorithm more challenging than building a driver-controlled algorithm? Why?

Checkpoint

In your Engineering Notebook:

- Record your responses to the Engineering Notebook prompts in Tasks 1-3.
- Record your responses to the reflection questions and your programs.

Activity 2: Better Control Through Encoders

Driving Questions

- What are encoders?
- How can we use the encoders built into our motors to gain better control of our robot's movement?
- What algorithms do we need to use to take advantage of our encoders?

What Will I Be Doing?

- I will learn about encoders:
  - What do they do?
  - How do they do it?
- I will use the telemetry data my encoders can send to measure a distance.
- I will create an algorithm that moves my robot by a set number of encoder values called ticks.
- I will design an autonomous algorithm that uses encoders to move my robot across the room, turn around, and return to its starting position.

Getting Started

- Up to this point, you've built your robot but haven't attached any sensors to increase its precision. In the last activity, you tested how long it took for your robot to move from one position to another using dead reckoning. Instead of relying on the time it takes your robot to move a distance with a set motor power, you can use your motors to take measurements of a distance using a component built right into the motor called an encoder.
- A motor encoder records the number of times the motor's axle has turned. The encoder can send that information to you through telemetry displayed on your Driver Station. Encoders measure the distance a motor's axle rotates in "ticks" units. Using telemetry, you can measure a distance by seeing how many ticks, or motor rotations, it takes for your robot to move from one position to another.
- When you set your motors to run at a set power, you might notice that your robot doesn't drive perfectly straight. Depending on the surface your robot is operating on and your robot's weight, one of your wheels may spin faster than the other. Using encoders, you can control the number of times your motor's axles turn, and, by doing so, you can make sure that both of your wheels are turning at the same rate.

IN YOUR ENGINEERING NOTEBOOK, DOCUMENT AND ANSWER THE FOLLOWING PROMPTS:

- How do you think encoders can track how many times your motor's axle is turning?
- Do you think cars and trucks use encoders? If so, why? If not, why not?
- How could you use encoders to improve the performance of your ball game robot?

WHAT'S NEXT?

- Gather your team.
- Grab your supplies (Engineering Notebook, robot, laptop and Driver Station, access to the programming tools, tape)

HOW WILL I DO IT?

- You will be using some of the algorithms in your programming tools that involve the encoders. To measure a distance, you will be resetting your encoders so they are counting up from zero and then getting your robot to move slowly toward a point. When you've gotten to the point you've marked using tape, you will go back and change your algorithm so your robot can autonomously move the same distance using encoders and the encoder value identified while you were running telemetry. In the final task, you will be using angular velocity algorithms to make your robot move across the room, complete a three-point turn, and return to where it started. Make sure you have utilized links in your programming tools on using encoders before you start working your way through the tasks.

Task 1: The Encoder Values

IDENTIFY THE PROBLEM:

- Another way of using dead reckoning is knowing the exact distance between two points.
- Using encoders, you can measure a distance using encoder values. You should set up telemetry algorithms to send the encoder value to your Driver Station while driving.
- Use tape to mark two points on the floor, one for your robot to start and one to stop at. The start and stop points should be around six feet apart. Ensure that someone is paying close attention to the total number of ticks for your robot to move from its starting position to the end position.

DESIGN AND PROTOTYPE:

- Set up an autonomous program and name it, "your team's name encoder.data." In the initialization section of your code, use an algorithm to reset the encoder for both your left and right motor. Set up two programs that provide telemetry data to your Driver Station. The data should track the number of ticks of your motors as they move from one point to another. Use half power in your algorithm to limit mechanical error.

TEST AND IMPROVE:

- Test your program and ensure you get the proper data output from your encoders. If your data output doesn't start at zero, you may have forgotten to include an algorithm to reset encoder values. When you reach your stop point, press Stop/Disable in the Driver Station.
- Repeat the process two more times and find the average encoder value.

Task 2: Run to Position

DESIGN AND PROTOTYPE:

- Now that you know the distance between your two points and their encoder value equivalent, you will create a new autonomous program that directs your robot to the finish position using a target position algorithm. Each motor's full rotation has an equivalent encoder value, so the data received between your starting position and your finish point may seem significant. The average you calculated based on the three tests of your program from the last task will be the target position you input for this task.

Tip

- You will want your motor power setting to be fairly low so you can stop your robot as soon as it gets to the stop point.
- Record the program you created and the total number of encoder values between the two positions you used in your Engineering Notebook.

TEST AND IMPROVE:

- Create a new autonomous program and name it "your team's name RunToPosition." Your program needs to start by resetting your encoders, so your motors know to start from zero and count up to the position you found by averaging your results in the last task. You will also need to set your target position and motor power so your motors will move your robot to the position you've set.
- When you've finished building your program, save it and test it. If your program doesn't work the way you want it to, use the troubleshooting process and make changes. Record the specific algorithms you built in your Engineering Notebook when it works properly.

Task 3: Multiple Target Positions

IDENTIFY THE PROBLEM:

- Now that you can set your motors to run to one target position, you can use multiple target positions. Angular velocity is the time it takes for an object to rotate around an axis. Your goal in this task is to have your robot move from the start position to the end position and then turn around and return to where it started.
- You discovered the value of your motor at one full rotation. Using that number, you can make your robot do more than move in a straight line. In this task, you will add multiple target positions to the program you just made that enables the robot to rotate and then move forward. Instead of changing your robot's direction by changing your motor's power to negative, you will make your robot turn around using encoder values.

BRAINSTORM AND EXPLORE:

- You will need to think about your motors' position during the different stages of your robot's trip between your start and stop points. Instead of adding to the number of ticks during each step of your robot's trip, you could also have your encoders reset more than once. The target positions you set for your robot are up to you.

DESIGN AND PROTOTYPE:

- The program you built in the last task can move your robot from the start position to the end position, so the first part of your trip is already taken care of. You will need to get your robot to turn around to face the starting position and then move back to where it started by adding onto the algorithms you already have. It may take several practice runs to figure out all the positions you will need to get your robot to move correctly, so pay attention to the telemetry your robot is sending you during its trips. Have one team member keep track of the encoder value your robot reports while it is turning to ensure that all your target positions are accurate.
- You will need to break your program into a few different parts for each target position you are setting. One way to break your program up is by telling your robot to finish getting to the target position you've set before moving on to the next. Another way to break down the algorithm is to program your robot to move until the motors have reached their target. Your teacher provided you with resources to support your learning with examples of how you might accomplish this.

TEST AND IMPROVE:

- There are many ways to use target positions, so the algorithm you build may look very different from the algorithm another team uses to complete this task. When you finish building your program, save it and test it out. If your robot isn't moving the way you want it to, use the troubleshooting process to make changes to your algorithm.
- When you are happy with how your robot moves, record your program in your Engineering Notebook.

Reflection

- How could encoders help your ball game robot complete its job?
- Could you have used encoders in the driver-controlled program you built? If so, how?
- Why would you want to use target positions to control how your robot moves instead of relying on motor power settings?
- What sort of telemetry could your encoders send other than the position of your motor?

Checkpoint

In your Engineering Notebook:

- Answer the questions from the Getting Started section of the activity.
- Record your programs from Tasks 1–3.
- Record the average encoder value your motors moved between your start point and end point.

Activity 3: Robot Senses

Driving Questions

- What are sensors, and how can we use them to improve our robot?
- What sensors that are available in our kit of parts?
- What additional sensors will be most useful in the ball game challenge?

What Will I Be Doing?

- I will learn about sensors that robots use to gather information from their environment.
- I will examine the available sensors that come in my kit of parts.
- I will investigate the types of data that sensors provide.
- I will create algorithms using sensor data to create a robot action.

Getting Started

- In the last unit, you learned that machines could send information to the Driver Station using telemetry. Telemetry sends real-time data from a machine to the person operating it.
- But how does a machine gather the information that it sends? Think back to the Mars rover that your team helped to get moving. How did it know it was stuck? The rovers that space agencies launch into space are incredibly complex. They are built using dozens of devices that gather information to be sent back to Earth. But how do the machines we use daily know about the world around them? How does a robotic vacuum see where the walls are in your house?
- People use their senses to gather information from the world around them, but machines don't have eyes to see; they have sensors! There are many types of sensors, and they are designed to gather all sorts of information.
- Refer to the programming tools provided by your teacher to investigate the sensors available in your kit of parts.

IN YOUR ENGINEERING NOTEBOOK, DOCUMENT AND ANSWER THE FOLLOWING PROMPTS:

- What type of sensors would a driverless car need to navigate a busy street?
- What type of sensors does a voice-activated assistant need to understand your question?
- What type of sensors does a GPS need to give you directions from where you are?

WHAT'S NEXT?

- Gather your team.
- Grab your supplies (Engineering Notebook, robot, laptop, kit of parts)

HOW WILL I DO IT?

- Before you build algorithms that test out your sensors, you should connect them to your control system and complete the hardware setup. Refer to the programming tools and hardware to set up a sensor that you can use to gather information from your surroundings and improve your robot's performance.

Remember

After you've named your sensor, you need to save your configuration!

Task 1: The Sensor

IDENTIFY THE PROBLEM:

- Just as you set up the motors in your programming platform, you will also want to set up your sensors. Refer back to your electrical diagram and the resources for your control system. Identify where your sensor should be plugged into your control system. Many sensors have a specific data protocol or a way to send data corresponding to where the sensor should be plugged into your control system. Consider this as a power adapter that you might have to plug a USB cable into the wall. Different power adapters provide different amounts of power to charge your device accurately. Data ports and data protocols work the same way.

- Many sensors use different types of data. For instance, the encoders you used in the last lesson used ticks and sent those numbers, including decimals. A decimal number is one form of data. Another form of data that a sensor might send is whether a condition is true or false. This is known as Boolean data. Sensors such as a touch or switch might send true or false, while a color or proximity sensor will send a specific value or number. Data types can help you understand why you might use specific algorithms to set up the hardware in your program. Consider the gamepad you might use for driving the robot; buttons on the gamepad will send a true or false value to the Driver Station, while the joysticks will send a number value in decimals ranging from -1 to 1.

DESIGN AND PROTOTYPE:

- After you have determined that your sensor is plugged into the correct port on your control system, the next step is for your programming tools to know what is plugged into the port and to ensure it knows the device's name. This is the process you used in hardware setup to ensure the operating system can recall or use the hardware within your algorithms. Refer to your teacher and your programming tools if you do not remember how to set up your hardware.
- Using your programming tools and strategy for scoring points in the ball game, identify the sensor you can use to help you best achieve your game strategy. This might be a sensor that tells you when a ball is in a particular position, when the robot has reached a specific position on the field, or if the robot can identify an image that gives a location. Using your programming tools, identify algorithms that will enable you to use the sensor on your robot for your ball game.

TEST AND IMPROVE:

- When you have discovered an algorithm that will help you, it is time to put the algorithm into your program and test it. Remember, most algorithms require fine-tuning and troubleshooting, so it might not work the first time.
- Consider adding data feedback that helps you determine if the sensor works correctly and what data you can gather from it. This is similar to the process you use with encoders.
- Record the telemetry data you collected from your sensor under different testing conditions in your Engineering Notebook.

Task 2: Robot Decisions

DESIGN AND PROTOTYPE:

- Now that you have data collected from your sensor, it is time to write an algorithm enabling your robot to act based on decisions. If-then statements or conditionals are used in programming to enable different conditions to happen on your robot based on the data it collects. Use your programming tools to discover how you might write an if-then algorithm for your program.
- Because you have chosen a sensor in the previous activity, it is time to prototype or test your algorithm with your teammates. Write directions for your teammate to act out what you want the robot to do. This can be essential in thinking through the algorithm's development process. Tell your teammate what process they should follow to make a decision based upon the sensor data. Think of the different scenarios the robot might go through:
  - If it sees an optimal value, what will happen?
  - What will happen if it sees a value that enables the robot to see it is completely off track?
  - What value will tell you if the robot is back on track?
  - What could the default value be that will ensure that the robot is safe until it is given another algorithm?

TEST AND IMPROVE:

- In your driver-controlled program, you may have used an algorithm that caused an action if a specific button was pressed. The same thing can be done using sensor data. If a certain value, such as a switch, is pressed on the robot, it can act independently without your input. Use the data you collected in the previous tasks to get your robot to decide based on the data it receives from the sensor you chose for your robot.
- Now that your algorithm is developed, it is time to test and troubleshoot it. Understanding how the robot responds to the algorithm you developed requires you to understand what data it is receiving and what it is doing based on that data. This is why telemetry algorithms are so helpful; they can help you identify how the robot responds to the data in real time. In the troubleshooting process, you will want to be able to identify if the robot is not completing a task. Is there an issue with the hardware or software? Gaining feedback can be essential to working through this process.

Task 3: Sensors, Data, and Calculations

IDENTIFY THE PROBLEM:

- In the previous task, you used an if-then statement to get your robot to decide based on a condition, such as if a button is pressed, and then stop. There are other ways that you can design algorithms for your robot to sense and respond to its environment based on the data it can collect.

- Data is often collected in the form of values or numbers. Just as in math class, you can do calculations using these numbers or make decisions based on a range of numbers; these are called operators. Operators can enable you to do calculations such as add, subtract, multiply, or divide. But a robot can make decisions on these numbers when you can evaluate data between numbers such as greater than, less than, or equal to.
- For instance, if you wanted a robot to respond to a specific range such as color or light. Colors have values in many different ranges, but you might want it to respond to various colors in the blue range. You might need a range of values to allow the robot to respond to various blues.
- Using your programming tools, research how you might set up an algorithm to respond to a range of values. Using your ball game strategy, identify how you are using a range of values that might help your robot score more points by achieving a specific action.
  - A pseudocode or basic directions for this type of algorithm might look like this:
    - Blue value = value <210 and >240
    - The robot should stop if the color sensor value is = to blue.

DESIGN AND PROTOTYPE:

- Develop and test an algorithm that compares the data it is collecting to respond to a value. Refer to the programming tools provided by your teacher, use any sample programs and sensors that enable you to improve your ball game robot by comparing data and then responding to that data.

Reflection

- When designing your robot, how did you imagine it would get information from the world around it?
- Now that you've used sensors on your robot, would you change the design of your ball game robot?
- How does initially collecting telemetry data help you improve your algorithm to use that data?
- How does using conditionals and operators enable you to create algorithms with more flexibility for your robot to perform more accurately?

Checkpoint

In your Engineering Notebook:

- Record your answers to the questions from the Getting Started section of the activity.
- Record your responses to the Engineering Notebook prompts in Tasks 1-3.
- Record your responses to the reflection questions.
- Record programs and wiring configuration details.

Activity 4: Collision Avoidance

Driving Questions

- How can we store data in our program?
- How can we reuse and call stored data?
- What are variables and how do we use them in our program?

What Will I Be Doing?

- I will create my own type of data called a variable.
- I will discover algorithms that enable me to use variables to improve data transfer in my program.
- I will use variable algorithms to program my ball game robot to avoid obstacles.

Getting Started

- A robot vacuum has sensors that help it avoid obstacles in your home. Newer versions have a way for it to store information about your room that it can reference and then refer back to so it can avoid obstacles more efficiently. You can store data in your program using variables.
- A variable can be defined as something that can change. For instance, look at our previous example of blue value = value <210 and >240. We could create a variable that could change and name its color. The color variable could have multiple options, such as red, blue, or green. The color could vary based on the data that the sensor receives.
- A variable can also be a way of storing a name or value that doesn't change. For instance, when you completed your hardware setup and named your hardware, such as left drive, you created a variable used throughout the program, which doesn't change.
- Research different algorithms that use variables in your programming tools. Consider how you might create a variable for the sensor data you used in the previous lesson.

IN YOUR ENGINEERING NOTEBOOK, DOCUMENT AND ANSWER THE FOLLOWING PROMPTS:

- What is a variable?
  - Where have you seen variables used?
  - What are they used for?
- How could you have used variables in a previous program?
- How could you use variables to store information on your robot that will help you be more successful with your ball game robot?

WHAT'S NEXT?

- Gather your team.
- Grab your supplies (Engineering Notebook, robot, and laptop).

HOW WILL I DO IT?

- You will develop a pseudocode, or written directions, for how you might use a variable in your program. Then, using your programming tools, research and choose a program template that utilizes variables similar to your pseudocode. The pseudocode should use variables that change or those that do not change and help you store data to avoid obstacles in your ball game.

Task 1: Obstacle Ahead

DESIGN AND PROTOTYPE:

- You have a variety of sensors and algorithm templates to choose from to improve your robot performance, avoid obstacles, and operate more efficiently in your ball game. Your teacher will provide some resources to guide you in your options. You can also use other sensors and algorithms you have discovered that you want to test.
- Before integrating other resources into your code, it is essential to consider the flow or process for improving communication. Programmers frequently use flow charts to illustrate the process they would like the algorithm to execute.

TEST AND IMPROVE:

- After you have developed a flow chart, have a team member act it out, similar to the previous activity where you acted out the ball game. This unplugged process of testing out the flow of how you would like to avoid obstacles using variables will ensure that the process will work before being implemented into your programming tools.

Task 2: Putting Process into Action

DESIGN AND PROTOTYPE:

- You were introduced to using variables and flow charts in the last activity. Now that you have completed some initial testing to determine if your thought process works through flow charts and are acting out those flow charts, it is time to put the process you tested into action. Use a sample algorithm or program to enable you to use the chosen sensor. The sample algorithm will help you utilize the code libraries to implement the sensor on your robot.
- Integrating new code into the robot programming tools can be challenging and require troubleshooting skills. You learned some of these previously, so additional strategies you can use when integrating new code include:
  - Use your programming resources to understand the best place to add code within the current code infrastructure. Your robot programming tools have a code infrastructure set up to run in specific modes and have a way to check for errors to keep the programming tools key.
  - Using telemetry to understand what is happening in the program versus what you see in the hardware can enable you to understand whether it is in your code or on the robot hardware.
  - Understanding where your code should be placed within the programming environment can help you overcome errors built into the code.
  - Use decomposition to understand where and how the data is being stored, processed, and transferred from the hardware setup, the program infrastructure, the output to the Driver Station, and to your robot. Tracing the path of where the hardware is set up, what you're doing with the hardware data, and then how it is transferred can increase your troubleshooting skills.

TEST AND IMPROVE:

- In your Engineering Notebook, create a testing plan for how you will test your algorithms using stored variables and the robot's response to avoid an obstacle in your ball game. Creating testing data can help you go back and use computational thinking to understand patterns in the data, decompose the problem, and further develop the algorithm to be more efficient.

Task 3: Iterate and Improve

TEST AND IMPROVE:

- Continue to iterate and improve your algorithm. The Engineering Design Process often has a measure or criteria that you use to determine if your end product or process is effective. Think about what criteria are considered a fully working and operating robot. Consider adding specifics, such as actions done within a certain amount of time or a certain number of tries. This can help you measure the performance of your algorithm and how it can help you meet your goals.
- Continue to improve and test your robot and algorithms so that your robot can complete the task of avoiding obstacles in your ball game to score points.

IN YOUR ENGINEERING NOTEBOOK, DOCUMENT AND ANSWER THE FOLLOWING PROMPTS:

- What are variables?
  - What are some examples of ways variables could be used in a program?
  - Could you have used variables in any of the algorithms you created before this activity?
- Is using variables in this program the simplest way to achieve your design criteria?
  - If so, why?
- How could using sensors and variables improve your robot's ability to compete in the ball game?

Reflection

- Does your ball game robot need to be able to detect obstacles?
- What sort of obstacles would your ball game robot need to navigate around?
- How would you use the new logic, math, and variable blocks in a program for your ball game robot?
- How can you use operators and logic to enable the robot to make better decisions?
- What are variable blocks, and how do they work?
- Did you have difficulty using the algorithms you were introduced to in this activity?
  - Which algorithms did you have trouble with?
  - Were you able to successfully use all the algorithms?
  - If you encountered problems using the algorithms, how did you work around them?

Checkpoint

In your Engineering Notebook:

- Record your responses to the questions from the Getting Started section of the activity.
- Record your responses to the questions from Task 1–3.
- Record the programs you created in Tasks 1–3.