FIRST® Robotics Engineering Explorations

Teacher 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?

Objectives

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

Materials

Each team will need:

- Engineering Notebooks
- Pens or pencils
- Tape to mark the start and stop positions for the robot
- Robot
- Laptop
- Controller

Getting Started

BEFORE THE START OF CLASS:

- Familiarize yourself with the concept of dead reckoning. In the context of students' robots, dead reckoning means using their knowledge of the distance between two locations as a basis for navigation.
- Take some time to learn about the difference between the process of building a driver-controlled and autonomous robot operation.
- Ensure their code is in the proper area for their programming tools. The programming tools used previously have directions for students to ensure they are placing code in the proper location for their robot operation.
  - *FIRST*® Tech Challenge
    - REV Robotics Programming Drivetrains
    - *FIRST* Tech Challenge Docs
    - Blocks Moving Forward and Backward
  - *FIRST*® Robotics Competition
    - *FIRST* Robotics Competition WPILib
    - *FIRST* In Michigan – Virtual Robotics Studio
- To get their robot to move between two points, teams will set up their program so that when the Driver Station is initialized, the motor power will be set, and the robot will move for a predetermined length of time or until they press Stop in the Driver Station. To return their robot to its starting location, teams can set their robot's motor to reverse for the same length of time they moved forward.

DURING CLASS:

- Have teams mark the position their robot's front wheels will start from and where they will stop using tape.
- Ensure every team has enough space to move their robot between two positions without colliding with another team's robot.
- Ask teams to name their program "their team's name Autonomous1."
- As teams are introduced to dead reckoning, they might have difficulty applying the concept to their robot. The landmarks teams will use in this activity will be marked using tape. They will be thinking of their movement in terms of the power their motors are set to and the length of time the motors are running.

Student Tasks

TASK 1: FROM POINT A TO POINT B

Brainstorm and Explore:

- As this is the first-time teams will be working with a program that doesn't use their gamepad, they might find it difficult to decide how to get their robot moving.
- Before they move on to working on a program, they should look at each part of their program.
- Prompt teams with questions that get them to focus on how the Driver Station and robot connect with the different parts of their program.
  - Why is having a way to initialize data before starting the robot valuable?
  - How can initializing data increase troubleshooting skills in code in the future?
  - What happens when you play/enable your robot code?
  - What happens when you stop your robot code?

Test and Improve:

- Ask teams to put their robots on the ground and mark the position of their front wheels on the floor with tape.
- Teams should then mark a spot on the floor approximately 10 feet from their robot's front wheels.
- Teams will create a program and title it "their team's name Autonomous1."
- There are a few different ways teams can set their robot to move autonomously between the two points they marked on the floor. By creating a program where their motor's power is set to 1 as soon as they press start/enable in the Driver Station, their robot will begin moving and stop when they touch the Stop button in the app.
- Teams may also try and experiment with the motor algorithms that set the rotation of their wheels. Those algorithms will be explored in greater detail in the next activity.
- After teams have tested their program and made necessary corrections, they should record their algorithms in their Engineering Notebooks.

TASK 2: FROM A TO B TO A TO C

Design and Prototype:

- By the end of Task 1, teams should have a program to move their robot between the two points marked with tape.
- During their research into dead reckoning, teams should learn that one way to navigate is through using set distances and speeds rather than sensor information.
- By creating a program that moved their robot to a set point, teams learned about the power they needed to send to their motors to get their robot from the starting point to the finish point and the length of time the motors needed to be running.
- By taking the settings they used in Task 1, teams can use dead reckoning to determine how to move their robot freely between the starting and ending points they've marked on the floor.
- Teams will add a third point on the floor between their start and finish points that they will be navigating to after moving back and forth between their first two points.
- Before teams start working on their program, they should make a point halfway between their start and finish points using tape.

Test and Improve:

- Teams should create an algorithm that sets their motor power to run from their starting position to the endpoint and then stop.
- From the endpoint, teams should have their robot reverse until it reaches the starting position and then stop. For this task, teams should have their robot move backward rather than turn around.
- From the starting position, teams should have their robot move toward the midpoint marked between the start and stop points.
- When students have finished coding and testing, they should record their OpMode in their Engineering Notebooks.

TASK 3: AN AUTONOMOUS BALL GAME ROBOT

Brainstorm and Explore:

- Using dead reckoning in Task 2, teams should have navigated the distance between the start and finish points they marked using tape. Teams should understand that dead reckoning is a technique used when you have limited sensors available.
- Before moving on to more complex autonomous functions, teams will consider what an autonomous program designed to solve their ball game challenge would look like.

- When teams were designing their ball game, they should have included points they could score by completing tasks autonomously. To keep autonomous tasks in the context of their ball game challenge, they should look back at the notes they took on their problem and reexamine how they would want their robot to solve it.
- Before teams think about a program that would solve their ball game, ask them to look back at the notes they took on their game challenge. The work teams have put into programming their robot over the last unit, combined with the basic autonomous program they made in this activity, may inspire them to think differently about the ball game challenge.
- When teams have reacquainted themselves with their ball game, they should start decomposing it into smaller tasks.
- When teams have broken their ball game robot's job into smaller tasks, they will begin thinking of those tasks in terms of autonomous period.
- Teams will answer the following questions for each of the tasks they broke their ball game challenge into:
  - What sensors would your robot need to address your ball game scoring, and how would they be used in an autonomous program?
  - What distances would your robot need to move, and how would it know where it was while it was moving?
  - How fast would your robot need to move?
  - Would you use dead reckoning to assist with your robot's navigation?
  - Would your robot be moving at the same speed the whole time?

Guiding Questions

- How can I control a robot without using a controller?
- What is dead reckoning? Who uses it? How does it work? How can it help our robots navigate?
- What is an autonomous program?
- What are the benefits of an autonomous mode over a driver-controlled mode?
- Do you think that autonomous programming is more challenging to create than the driver-controlled algorithms you've been building?

Suggested Extensions/Modifications

- Extension: Have students explain how they could use dead reckoning in their personal lives.
- Extension: Ask students to add to their program in Task 2. Teams may want to increase the number of stops they make or adjust their robot's speed, so it completes its trip as quickly as possible.
- Extension: Ask students to have their robot turn around in Task 2 rather than reverse when it reaches the endpoint.
- Modification: If a team is having trouble creating a program for either of the first two tasks, pair them with another team so they can work together.
- Modification: If teams are struggling to create autonomous programs, ask them to break what they are doing into the exact steps they made for the teacher when creating instructions for completing simple tasks.

Teacher Reflection Questions

- Do my students understand dead reckoning after researching the concept in the Getting Started section of their activity?
- Could my students set up an autonomous program in the programming environment?
- Were my students able to select the autonomous program they created from the Driver Station?
- Do my students understand how dead reckoning can be used to aid in the navigation of their robot?
- Do my students understand the purpose of autonomous programming?

Student Artifacts

- In their Engineering Notebook:
- Explain how airplane pilots and ship captains use dead reckoning.
- Explain how dead reckoning could be used to help a robot navigate.
- Record their programs for Tasks 1–2.
- Answer the following questions about their ball game robot's autonomous program from Task 3:
  - What sensors would your robot need to address your ball game scoring, and how would they be used in an autonomous program?
  - What distances would your robot need to move, and how would it know where it was while it was moving?

- How fast would your robot need to move?
- Would you use dead reckoning to help with your robot's navigation?

Checkpoint

- Have students explain dead reckoning.
- Have students explain an autonomous program.
- Have teams explain the process of building an autonomous program.

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?

Objectives

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

Materials

Each team will need:

- Engineering Notebooks
- Robot
- Laptop and Driver Station
- Access to the programming tools
- Tape to mark positions on the floor

Getting Started

BEFORE THE START OF CLASS:

- View resources in the programming tools on how to use encoders.
  - XRP
    - XRP – Driving for A Distance
  - *FIRST* Tech Challenge
    - Blocks – Using Encoders
    - Computational Thinking – Simple Operations Motor OpMode
  - *FIRST* Robotics Competition
    - *FIRST* Robotics Competition WPILib – Zero to Robot – Programming Your Robot
    - *FIRST* Robotics Competition WPILib – Digital Rotary Sensors
    - *FIRST* In Michigan – Virtual Robotics Studio
- Try to create an open space for each team to drive their robot.

DURING CLASS:

- Share with students the resources on using encoders provided above.
- Ask teams to explore algorithms that use encoders.
- There are more than a few ways to use encoder algorithms to work through the tasks for this activity, so encourage teams to test out different algorithms. Begin with the simpler algorithms and then move to more advanced ones.
- In the students' Getting Started section of the activity, they are introduced to the fundamental functions of encoders. They will learn that encoders track the way that a motor's axle spins, and that the information can be sent as telemetry. For this activity, teams will be testing out some of the basic functions of their encoders, but they should be encouraged to experiment with some of the more complex algorithms if they finish their tasks quickly.

Student Tasks

TASK 1: THE ENCODER VALUES

Identify the Problem:

- Teams will use the encoders on their motors to track the number of wheel rotations their robot needs to cycle through to get from one position to the other.
- In the last activity, teams used dead reckoning by way of the power they were putting into their motors and the time it took for their power setting to move their robot to their stop point.
- This task aims to learn how a team's motor encoders interpret the spin of their axle using specific data or values.

Design and Prototype:

- Teams should mark a starting location and an endpoint on the ground in tape and place their robot on the tape at their first point.
- Teams should set up an autonomous program to track the encoder data their motors turn between two points. Record information from the Driver Station as they are completing the task.
- Teams will create an autonomous program named "their team's name encoder.data."

Test and Improve:

- Teams should set their motors to about half power and add telemetry data so they can monitor the position of their encoder data as it moves toward the stop position.

TASK 2: RUN TO POSITION

Design and Prototype:

- Teams should now know the average encoder value for moving their robots from their starting to finish positions.
- Using the programming tools, they should be able to utilize sample encoder programs that allow them to collect the encoder data and send it to the dashboard of their Driver Station through telemetry data.
- Remind teams that each full rotation of a motor will have a specific encoder value depending on the motor and encoder. As the endpoint that teams are driving their robot is likely more than one full rotation of their motors away, the target position they set will be a large number.

Test and Improve:

- Teams should create a new autonomous program and name it "their team's name RunToPosition."
- The algorithm they create will need to be able to stop and reset both of their motors, so the motors are counting up from zero.
- Teams will set their target position to the number of ticks they find by averaging the distance between their start and stop positions in the last task.

TASK 3: MULTIPLE TARGET POSITIONS

Identify the Problem:

- Now that teams know how to move their robots using Target Position algorithms. They will be adding onto their program so that their robot moves from their starting position to the end position and then turns around and returns to where it started.
- Teams should become familiar with the definition of angular velocity, which is the time it takes for an object to rotate around an axis.
- Setting multiple target positions and refining the algorithms for accuracy will be challenging, which may require additional time.
- This task aims to have their robot turn using a position defined by encoder data rather than using a negative power setting on one of their motors.

Test and Improve:

- The algorithms that teams built in the last task already take care of the first part of the trip their robot is taking in this task.
- Utilize the links to the programming resources to help students troubleshoot algorithms.
- Consider talking through the algorithm with the students to help them think about the sequential steps needed to break down the problem. Each programming environment will have a slightly different approach.
- Utilize if-then statements to help students check if a target position has been met before moving on to the next statement.
- Using what they learned in the resources, most students may try adding more target positions imminently after the algorithms they used in the last task.
- If teams add more target position blocks without trying to make new measurements, their robot probably won't move the way they want it to.

- Teams may want to consider resetting their motor's encoders once they reach the end position to start from zero before setting new target positions.
- Remind teams that their robot should be turning based on a target position and not a negative value in motor power.

Guiding Questions

- What is an encoder? What does it do? How can you use them to control your robot?
- How can you use encoders to improve the way your ball game robot works?
- What is an encoder value?
- What encoder data is equivalent to on full motor or wheel rotation?
- How do you get your encoders to start at zero when you start your algorithm?

Suggested Extensions/Modifications

- Extension: In Task 3, students program their robot to spin around twice at the endpoint before returning to the start point.
- Modification: Ask students to take detailed notes on the programming tools they utilize and create a quick guide for reference.

Teacher Reflection Questions

- Do my students understand the purpose and function of encoders?
- Did my students understand the information they were presented with the programming resources?
- Were my students able to measure the distance between their start and stop points using their encoders?
- Were my students able to create an autonomous program that guided their robot to a target position in Task 2?
- Were my students able to create an autonomous program that guided their robot to move to a point, spin around, and return to the point where it started?

Student Artifacts

- In their Engineering Notebook:
- Record their responses to the questions from the Getting Started section of the activity.
- Record their programs from Tasks 1–3 sections of the activity.
- Record the average encoder value their motors moved between their start point and end point.

Checkpoint

- Have students explain the purpose and function of an encoder.
- Have students explain what encoder data is.
- Have teams explain how to use telemetry to measure a distance.
- Have teams explain the purpose of using encoders to reach a target position.

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?

Objectives

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

Materials

Each team will need:

- Engineering Notebooks
- Robot
- Laptop
- Kit of parts

Getting Started

BEFORE THE START OF CLASS:

- Spend some time learning about the types of sensors teams will use.
- You may want to try using one of the sensors to understand the process students will go through.
- If you are having trouble connecting any of the sensors in the team's kit of parts, refer to the programming resources to take a deeper dive into the parts.

DURING CLASS:

- Direct teams to explore the sensors available.
  - XRP
    - XRP – Sensors
  - *FIRST* Robotics Competition
    - *FIRST* Robotics Competition WPILib – Hardware – Sensors
  - *FIRST* Tech Challenge
    - *FIRST* Tech Challenge – Docs – Blocks Using Sensors
    - Class Pack Color Sensor Blocks Activity
    - Class Pack Encoder Drive Activity
    - Class Pack – Touch Sensor Activity
- Encourage students to test sensors without integrating them into the robot. They will need to understand more about how the sensor collects data to ensure proper integration on the robot.
- Ensure teams are connecting their sensors to the correct ports of their control system.
- Teams should complete hardware setup for their sensors before using them in a program.
- In the Getting Started section, teams will be introduced to sensors. They are asked to think about sensors used by driverless cars, voice-activated assistance, and GPSs in the world.

Student Tasks

TASK 1: THE SENSOR

Identify the Problem:

- Students will learn additional information about sensors and the data they utilize.
- They will choose a sensor to help them score more points in the ball game challenge.
- They will determine the correct electrical connection and hardware setup for their sensor.
- Have students refer to the programming tools to ensure they correctly set up their sensors.

Design and Prototype:

- Students will discover algorithms that will help them and then put the algorithm into their program and test it. Remember, most algorithms require fine-tuning and troubleshooting, so it might not work the first time.
- Students should consider adding data feedback that helps them determine if the sensor works correctly and what data they can gather from it. This is similar to the process they used with encoders.
- They should ensure they record the telemetry data collected from the sensor under different tests.

TASK 2: ROBOT DECISIONS

Design and Prototype:

- Students will learn about conditionals such as if-then statements.
- They will use them to enable different conditions on your robot based on the data it collects.
- Have students write out pseudocode and then act out it to test their algorithmic thinking before they start programming. This can help reduce frustration when they can't determine if the errors are in hardware or software. They can troubleshoot the overall thinking process, allowing them to refine the data in the algorithms versus the overall algorithm.
- Guide students with these questions:
  - 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?
  - Students will use programming tools to discover how they might write an if-then algorithm for their program.

Test and Improve:

- In their driver-controlled program, they may have used an algorithm that caused an action if a 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.
- Students will use the data collected in the previous tasks to get their robot to decide based on the data it receives from the sensor they chose for their robot.
- Now that students have algorithms developed, it is time for them to test and troubleshoot it.
- They should narrow down how the robot responds to the algorithm they developed.
- They should focus on what data it is receiving and what it is doing based upon 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, students will want to be able to identify if the robot is not completing a task. Is it through hardware or software? Gaining feedback can be essential to working through this process.

TASK 3: SENSORS, DATA, AND CALCULATIONS

Identify the Problem:

- Students will learn more about operators and conditionals and how a robot can make decisions using algorithms that do calculations and then evaluate the data and plan based on it.
- Many of the programming tools have these built in. Point out to students where conditionals are occurring, the program is making a calculation, and where you can use items such as greater than, less than, or equal to.
- This takes the process of data and turns it from numbers into a true or false statement that the robot can use to make the decision.

Design and Prototype:

- Students will develop and test an algorithm that compares the data it is collecting to respond to a value.
- Refer students to the programming tools for the lesson and guide them to use any sample programs and sensors that enable them to improve their ball game robot by comparing data and then responding to that data.

Guiding Questions

- What was the process for the hardware setup for motors with encoders?
- How do you add the sensors you attached to your wiring diagram?
- Why is updating your wiring diagram important?
- How is the data the sensor provides helpful in enabling the robot to make decisions?
- How does the data help you improve algorithmic thinking skills in designing robot programs?

Suggested Extensions/Modifications

- Extension: Ask teams to find multiple programs to send telemetry from their sensors to the Driver Station.
- Extension: Ask teams to explain the importance of a wiring diagram.
- Extension: Ask teams to think about the best position for mounting their sensors on their robot and explain why they chose those based on the data the sensor is collecting.
- Modification: Provide students with specific programming tools to help them focus on one specific sensor.
- Modification: Ask teams that have successfully configured their sensors to assist teams with difficulty.

Teacher Reflection Questions

- Do my students understand the purpose and function of sensors?
- Did my students remember how to configure electronic components?
- Were my students able to configure their sensors properly?
- Were my students able to set up programs that tested their sensors?
- Were all teams' sensors functioning correctly?
- If the sensors were not functioning correctly, was the issue caused by a mistake in the configuration and setup or algorithm development?
- Were my students able to add their sensors to their wiring diagrams?

Student Artifacts

- In their Engineering Notebook:
- Record their answers to the questions from the Getting Started section of their activity.
- Record the port they used to connect their touch sensor and the name they gave it in their configuration and update their wiring diagram to include the sensor.
- Record the OpMode they created to send telemetry from their touch sensor to their Driver Station app.
- Record the port they used to connect their sensor and the name they gave it in their configuration and update their wiring diagram to include the sensor.
- Record the program they created to send telemetry from their sensor to the Driver Station.

Checkpoint

- Ask teams to explain how the sensors they've attached worked.
- Ask teams to explain the importance of a proper configuration.
- Ask teams to describe the types of telemetry they could get from their sensors.

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?

Objectives

- Teams will create their type of data called a variable.
- Teams discover algorithms that enable them to use variables to improve data transfer in their program.
- Teams will use variable algorithms to program their ball game robot to avoid obstacles.

Materials

Each team will need:

- Engineering Notebook
- Robot
- Laptop

Getting Started

BEFORE THE START OF CLASS:

- In this activity, teams will work with new types of algorithms that use variables in addition to the sensors introduced in the last activity. Take some time to familiarize yourself with the variables.
- Many of the example programs in the programming tools use variables. Understanding the variable type and how it stores data is a vital programming principle in computer science for students to understand. This will enable them to transfer these skills to different programming languages.
- Explore through the programming tools and choose examples that have variables that you could use to demonstrate these to students:
  - XRP
    - XRP WPILib
  - If using Python programming and variables, open this reference in a new tab. If using blocks, the process can use the *FIRST* Tech Challenge resources.
  - *FIRST* Tech Challenge Examples:
    - Computational Thinking Writing Software Playlist
  - *FIRST* Robotics Competition Examples
    - *FIRST* Robotics Competition WPILib – Examples and Tutorials
- Teams will need a clear path toward an obstacle. A wall, overturned desk/table, or other large objects will work best.

DURING CLASS:

- As teams build and test their program, and ensure they save their work.
- Ask teams to keep track of the algorithms they create in their Engineering Notebooks.
- Encourage teams to work together if they are having difficulty using the algorithms.
- When students direct their robots towards obstacles, ensure they use a low motor power setting. They risk damaging their sensors if they run their robots at full speed towards a wall.
- Teams should also be aware that their sensors may have trouble gathering needed data. They should consider several factors (environmental, such as light, hardware setup and placement, algorithm development, and code placement).
- In the Getting Started section of their activity, teams are introduced to the algorithms they will be using. In Task 1, they will be working with comparison and operator algorithms. As they get to Task 3 and Task 4, they will work with variables. New algorithms can be challenging to work with, so encourage teams to use *Gracious Professionalism*® and assist others when they can.

Student Tasks

TASK 1: OBSTACLE AHEAD

Design and Prototype:

- Students will choose from the variety of sensors available to them in their kit of parts that will best help them achieve more points in the ball game challenge by avoiding certain obstacles.
- Students will develop a flow chart of how they hope to improve their robot processing to score more points.
- Students will learn troubleshooting skills using iteration and testing to improve and perfect their code design.

Test and Improve:

- Students will do an unplugged activity to act out the process they developed in their flow chart above. This will help them test the algorithm development before implementing it into their code.

TASK 2: PUTTING PROCESS INTO ACTION

Design and Prototype:

- Students will now use program templates to write algorithms that put their flow and process into machine code that the robot can understand using their programming tools.
- Students should work through a troubleshooting process as they implement their algorithms. Encourage them to start with small changes, test to see if they work, and then implement larger ones.
- Ask student questions to ensure they are:
  - Understanding within the programming tools and templates where the sample code goes within their current programming structure. Are you putting your algorithm in the correct spot?
  - How is the data being recalled and transferred in the program? Are they recalling the right data? Are there any syntax errors? Is the flow of the program completing the same process that they tested when they acted it out?
  - How are they using telemetry and data collection in the process to help them in their troubleshooting?
  - Is something in the programming platform changing the flow they thought they could achieve?

Test and Improve:

- Checking with students during testing ensures they save programs that work with a new name.
- Students often change code and forget to save it or document it. This leads to decreased success and increased frustration. Developing guidance of saving often and documenting what works and what doesn't will help them be more successful and have a reference to always fall back on to help them have little success.

TASK 3: ITERATE AND IMPROVE

Test and Improve:

- Students will continue to iterate and improve their algorithms.
- The engineering design process often has a measure or criteria that you use to determine if your end product or process is effective.
- Guide students to develop success criteria. Ensure they have specifics, such as actions done within a certain amount of time or a certain number of tries. This can help them have a guide and reward for your improvements.
- Ensure students document the testing process and measure the successes and lessons learned.

Guiding Questions

- What is the purpose of a logic algorithm?
- What is the purpose of operators?
- Why do you need to use logic and operators that integrate sensors?
- What data is gathered from sensors?
- Where have you heard the term 'variables' used before? How were they used?
- Could you have used variables in any of your previous programs? If so, why?

Suggested Extensions/Modifications

- Extension: In Task 1, find a way to avoid the obstacle without using variables.
- Extension: In Task 2, find a way to get your robot to reset to its original position after it's stopped at the obstacle.
- Extension: In Task 3, find a way to have your robot turn around 180 degrees and return to its original position rather than reverse.
- Modification: Have teams that are having trouble with Task 1 and Task 2 work together.
- Modification: If teams are having difficulty using variables in Task 3, ask them to find a way to create a program that allows their robot to detect an object and back away without the use of variables.

Teacher Reflection Questions

- Do my students understand the purpose of comparison logic blocks and operators?
- Do my students understand how to use their sensors?
- Were my students able to create functional programs for each task?
- Do my students understand what a variable is?
- Do my students understand why they are using variables to create a sequence?
- Were my students able to successfully use all the algorithms that were introduced in this activity?

Student Artifacts

- In their Engineering Notebook:
- Record their responses to the questions from the Getting Started section of the activity.
- Record their responses to the questions from Tasks 1–3.
- Record the program they created in Tasks 1–3.

Checkpoint

- Ask teams to explain how they can measure data using sensors.
- Ask teams to explain each of the algorithms that were introduced in this activity (logic comparison along with and blocks, math blocks, and variable blocks).