FIRST® Robotics Engineering Explorations

Teacher Guide – Make It Move

Unit 3

TABLE OF CONTENTS

Activity 1: Configure It Out

Driving Questions

- Now that the robot is built and wired, how do we drive it around?
- How can we communicate with the robot so it can do its job?

Objectives

- Teams will install software tools for programming and hardware configuration.
- Teams will establish the wireless connection between the operating device and the robot.
- Teams will configure the software to recognize the hardware appropriately and do an out-of-the-box test of the hardware.

Materials

Each team will need:

- Fully built and wired robot
- Gamepads
- Driver Stations
- Laptop (See the following directions for laptop specifications.)
- Engineering Notebook
- Links to software download and hardware configuration

Getting Started

BEFORE THE START OF CLASS:

- Review the laptop requirements for the program you are using:
  - Computer requirements for *FIRST*® programs – *FIRST* Tech Challenge Docs 0.2 documentation (firstinspires.org)
- Provide students with links to software and hardware setup for the program you are implementing.
  - XRP robot controller is composed of a Raspberry Pi Pico W microcontroller with built-in Wi-Fi and Bluetooth. If the WPILib and *FIRST* Robotics Competition tools are not an option, the XRP can be programmed using the XRP programming platform. As a Driver Station, students will need to configure the web application for control found in the XRP Users Guide – Creating a dashboard.
  - XRP Users Guide – Two robots in one
  - XRP can also be used with the *FIRST* Robotics Competition WPILib. An Intro to Java Course is located in Thinkscape. You can access the course and add students using this video:
    - Using Thinkscape to Access *FIRST* Education Learning Content YouTube Video
- In *FIRST* Robotics Competition, the robot controller is called the NI. roboRIO 2.0. It has a separate wireless radio and uses a computer and programming tools as the Driver Station. The program running on the computer controlling the robot is the *FIRST* Robotics Competition Driver Station and *FIRST* Robotics Competition PC Dashboard.
  - *FIRST* Robotics Competition WPILib – Zero to Robot Step 2 (Complete Steps 2, 3, and 4.)
  - Note that you can choose a programming application you would like and that the WPILib tools require a PC. The programming tools require VS Code or LabVIEW installation, which will not work on a Chromebook.

Tips

- Provide students with links to software and hardware setup.
- Teams need to the robot with a charged battery and Driver Station components.

- *FIRST* Tech Challenge robot controller is called the Control Hub as the robot controller. It also has a Driver Hub that establishes the wireless connection for control and is considered the Driver Station. The Driver Hub has a Java-based SDK, or app that runs the robot program for control.
  - REV Duo Docs – Control Hub User Guides
  - Connect to the Robot Controller
  - Updating Wi-Fi Settings
  - Connecting Driver Station to Control Hub
  - Wiring Diagram
  - Next Steps
  - FIRST Tech Challenge Docs – Hardware and Software Configuration
- You may need to test the ability to connect devices wirelessly in your building. Occasionally, you can run into issues with Wi-Fi blockers on school networks.
- Ensure you and your students have access to the necessary hardware (laptop specifications) and wireless network testing.

DURING CLASS:

- Have students get into their teams at the start of class.
- Remind teams to keep important information in their Engineering Notebooks.
- Provide groups with resources for setting up software platforms and hardware configurations.
- Have teams discuss the outcomes of the Getting Started activity as time allows.
- They must ensure communication between the TV, the game console, and the controllers. Showing students a physical gamepad and added parts might be helpful – prompt students to explore the parts. Guide them to brainstorm ways the pieces connect and why they are connected this way.

Student Tasks

TASK 1: INSTALLING THE PROGRAMMING TOOLS (IDE)

Brainstorm and Explore:

- Teams need to discuss how they will control the robot they have built.
- Encourage teams to think about which parts of the Getting Started activity and their robot are alike.

Design and Prototype:

- XRP: XRP User Guide – XRP Code
- *FIRST* Tech Challenge: FIRST Tech Challenge Docs – Writing an OpMode
- *FIRST* Robotics Competition: *FIRST* Robotics Competition WPILib – Control System Software
- The gamepad executes the game program, stores your data, displays the gaming events on the TV, and lets you communicate with the game using controllers. It acts as the central hub that controls all communications and activities in the game.

TASK 2: PAIRING THE DRIVER STATION WITH THE ROBOT CONTROLLER

Design and Prototype:

- Like the game console and the TV, teams need both the Driver Station and the robot controller to recognize one another before communicating. This process is considered establishing a wireless connection, such as connecting to your router in your home for the Internet.
- Teams will think about pairing the robot controller and Driver Station and write notes in their Engineering Notebook about the process.
  - XRP: XRPCode – Integrated Development Environment
    - Note: If using the WPILib with the XRP, use the simulator to run the application on a computer through Wi-Fi.
  - *FIRST* Robotics Competition WPILib – Robot Simulation Tools
  - *FIRST* Tech Challenge: *FIRST* Tech Challenge Docs – Pairing Driver Station to Robot Controller
  - *FIRST* Robotics Competition: *FIRST* Robotics Competition WPILib – Zero to Robot Step 3

Test and Improve:

- Teams will connect their robot controller and Driver Station.
- Teams should be given the resources linked above and follow them step by step. If they skip steps, they often have communication issues, and the robot controller and Driver Station will not connect.

Career Connection

These skills relate to the types of skills a cable technician or auto mechanic use every day.

TASK 3: CONFIGURING THE MOTORS

Design and Prototype:

- This task aims to get students to configure their motors within their programming environment. Students need to understand that without declaring motor ports and data types, their motors may not work at all, or they may work incorrectly.
  - XRP
    - This task can be skipped for the XRP. It automatically contains all configurations for the needed hardware.
  - *FIRST* Tech Challenge: *FIRST* Tech Challenge Docs – Hardware and Software Configuration
  - *FIRST* Robotics Competition: *FIRST* Robotics Competition WPILib – Zero to Robot Step 4 Test Drivetrain Program
- Allow students time to explore the connection between how the programming tools send the correct signal to data to the hardware to make it operate.

Test and Improve:

- Teams will work together to create and name a configuration.
- Teams should use the resources provided to ensure they can send the correct signals to motors.
- Teams will create a plan for documenting and storing important information and record it in their Engineering Notebooks.

TASK 4: BALL GAME COMMUNICATION

Brainstorm and Explore:

- This activity aims to help teams apply their knowledge in wireless connectivity and configuring hardware.
- Give teams time to research and think about ways to give their ball game robot-specific commands.

Design and Prototype:

- Teams should create a plan and put it in their Engineering Notebooks. They should note why pairing and configuration are essential to their robot design.

Guiding Questions

- How is a wireless connection different from connecting with a wire? Why do you need to do both to make a configuration work?
- Where can you find examples of configuration in the real world?
- What needs to happen so your device can communicate with your robot?
- Describe series of steps an electronic message takes as it is sent from Driver Station to your robot motors.
- What strategies can you use going forward as you learn to code?

Suggested Extensions/Modifications

- Modification: Provide students with time to research pairing and configuration.
- Encourage students to share what they know with their peers. Encourage students to use the “Share Something, Learn Something” strategy. This strategy has students share a key idea they have learned about the topic of creating a wireless connection and hardware setup.

Teacher Reflection Questions

- Were my students able to make a connection between this activity and how it could help address their ball game?
- Did they successfully get wireless connectivity and hardware setup on their robot in preparation for programming it?

Student Artifacts

In their Engineering Notebook:

- Document their Getting Started design ideas.
- Record their wireless connection and hardware setup notes.
- Update the wiring diagram and notes. Explanation of how this process is essential to the ball game they are trying to solve.

Checkpoint

- Have students explain downloading, wireless connectivity, and hardware setup.
- Have students explain what the hardware on their robot will need to do in their ball game.

Activity 2: Programming Is Everywhere

Driving Questions

- How can we use a gamepad to move our robot?
- How and why should we use a similar strategy with our ball game robot?

Objectives

- Teams will program their robot to be operated using a dashboard and controls.
- Teams will participate in a race to test how well their robot can move and how fast it can go.

Materials

Each team will need:

- Fully assembled, wired, and configured robot
- Gamepad
- Computer
- Driver Station
- Engineering Notebook
- Links to tools for setup and programming

For the drag race:

- Masking tape
- Timers
- Metersticks
- Racing Bracket Diagram
- Racing Rubric

Getting Started

BEFORE THE START OF CLASS:

- Determine the programming language you want students to use, given experience, robot program, and focus.
  - XRP: XRP User Guide – Creating a Dashboard
  - *FIRST* Tech Challenge Docs – Programming Resources
    - REV Duo Programming – Hello Robot
  - *FIRST* Robotics Competition Programming Resources
    - *FIRST* Robotics Competition WPILib – Programming Basics
- Familiarize yourself with key processes: writing a program, saving a program, initializing the program, and running the program.
- Write and save a program to familiarize yourself with the process and to identify possible challenges teams may encounter as they work.
- Prepare the space being used for the drag race before students arrive.

DURING CLASS:

- Have teams get into their groups at the start of class.
- Instruct teams to keep important information in their Engineering Notebooks.
- Provide teams with the programming resources provided above.
- Allow teams to discuss the outcomes of the Getting Started activity as time allows.
- Present teams with the scenario in their activity. They must think of other cases where they are programming something in their lives. If they get stuck, encourage them to think about how they use their cellphones and apps or when they set an alarm to wake up. There might even be cases that are not electronic but could still signify something is being programmed.

Student Tasks

TASK 1: TESTING YOUR FIRST ROBOT PROGRAM

Design and Prototype:

- Direct teams to refer to the programming tools you have chosen for your students to use. They will want to look for guides within the programming environment that allow them to test a Tank Drive program.
  - XRP
    - XRPCode – Integrated Development Environment
  - *FIRST* Tech Challenge
    - REV Duo Docs – Robot Navigation: Blocks
    - *FIRST* Tech Challenge Docs – Blocks Tutorial
  - *FIRST* Robotics Competition
    - *FIRST* Robotics Competition WPILib – Zero to Robot Step 4
- Students will refer to the programming resources you have provided them with. It is their job to create a new program and replicate the code. If they are having trouble, refer them to:
  - The programming tools and resources have all the information students should use.
  - Common troubleshooting techniques and questions:
    - Ensure students follow the guides step by step.
    - Ensure the hardware ports match the guides or the program is changed to match the port.
    - Ensure that if students are using a text-based language, they have checked for syntax errors.

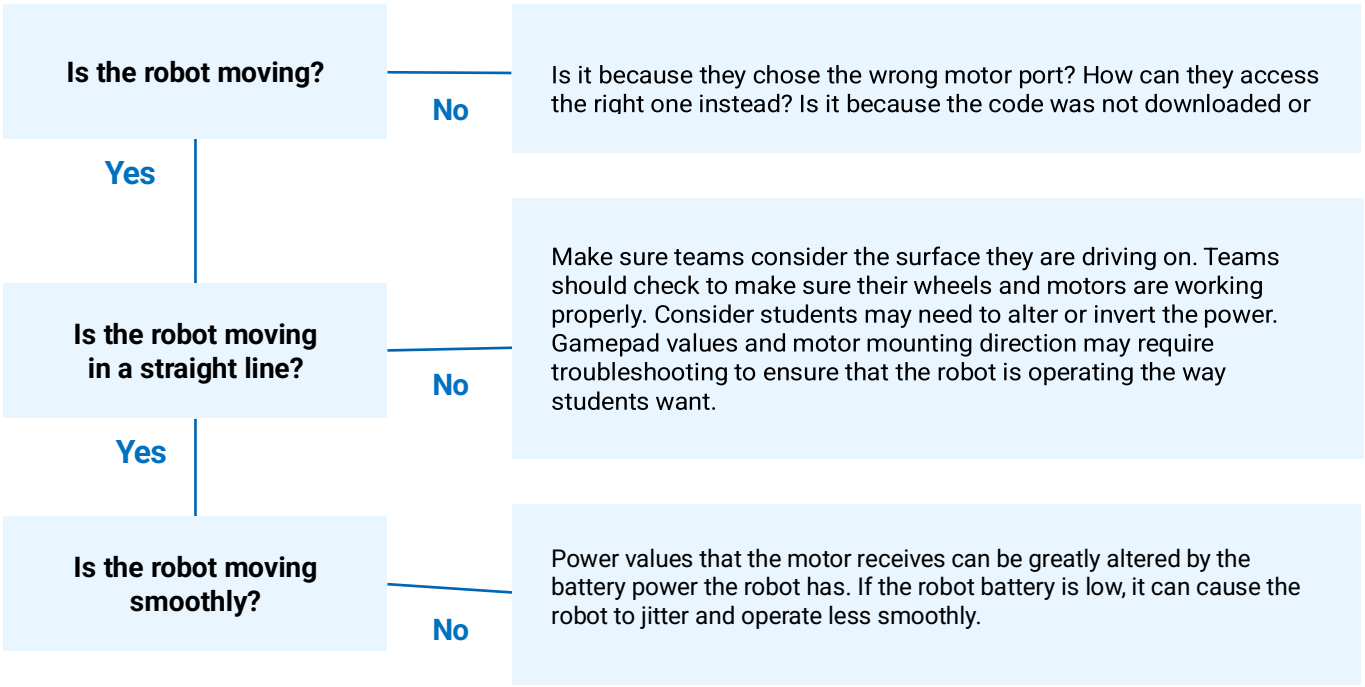
TASK 2: DRIVING THE ROBOT

Design and Prototype:

- Teams will test out their program, but it is up to them to ensure the code works the way they want it to.
- Instruct teams to document any issues they encounter as they work.

Test and Improve:

- Encourage students to keep the following criteria in mind when driving their robot:



TASK 3: DRAG RACE ON!

Design and Prototype:

- Create space on the classroom floor, in a hallway, or in the school gym to prepare for the race.
- Setting up the space in advance will allow teams to plan and practice more. Provide materials needed to set up the two racing lanes. Masking tape will help differentiate the lanes and the Start and Finish lines. You can use metersticks to measure the length of the track. It is essential to make sure the track isn't too short.
- Drag races require two robots to race each other. If you use one *FIRST* Robotics Competition Robot with two different drivers, drive the robot on the drag strip and compare time. Think about what you want the rules of the game to be.

Test and Improve:

- Teams will now apply what they have learned about creating a robot program and driving a robot to a real-world scenario: drag racing.
- Give teams time to explore the connection between their task and an actual drag race. If possible, show teams a video of an actual drag race. Ask questions such as:
  - How does friction affect the ability of a vehicle to take off and move?
  - Can the center of gravity affect the friction or contact the wheels have with the surface?
  - Races: Is it one race per team robot, or is it a best two out of three scenario, for example?
  - Winning/Elimination: Does the winner from one round play the winner from the next round? What happens to the teams that lose a round? Using a bracket diagram is a good idea so teams can organize each race.
  - Have students look at the following questions as they are testing:
    - What was the time it took from start to finish?
    - Do the motors have enough power to allow the robot to accelerate quickly?
    - Do the motors have too much power, causing them to spin too quickly?
    - Was the robot driving straight?

The championship team should be determined based on the following criteria:

- Does your Engineering Notebook contain the name of your program, the team notes about the task, and a completed reflection?
  - Is your robot functional (moving)? Does your robot move in a straight line?
  - How many races has your robot won? Which robot is the fastest?

Guiding Questions

- Where can you find examples of programming in the classroom?
- What are the qualities your motors will need to perform correctly?
- What communication or data is being transferred from the gamepad to the motors?
- How can you change your code to improve your robot's performance?
- How do you improve motor performance through programming your robot?

Suggested Extensions/Modifications

- Extension: Provide teams with opportunities to create their unique program designed for a specific task, such as maximum speed.
- Extension: Suggest a reflection of the Getting Started activity. Just like programming something in real life, such as recording a show on a cable box, you must ensure it is recorded properly when you go back and watch it. If it didn't, what can you do to ensure it correctly records the next time? Is it the robot that has the issue, or does your code need improvement?

Teacher Reflection Questions

- Were my students able to make a connection between this activity and how it could help complete the ball game?
- Did they pair and configure their robot successfully in preparation to program it?

Student Artifacts

In their Engineering Notebook:

- Programming links and screenshots of student programs
- Completed Racing Rubric with race results
- Documentation of everyday opportunities for programming
- Notes about enhancing robot functionality
- Record of driving strategies using a dashboard and controller

Checkpoint

- Have students investigate how to create and copy a program and how to use their programming tools.
- Have students explain why programming is an important part of their robotic solution for their ball game robot and how it enhances the robot's functionality.

Activity 3: Troubleshooting Is Everywhere

Driving Questions

- How can we troubleshoot a program?
- Why is troubleshooting important?

Objectives

- Teams will investigate a program that isn't doing what it should. Then, they will fix it.
- Students will use their Engineering Notebooks to record what the precise problem is.
- Students will document the steps they take to fix the problem they have identified.

Materials

Each team will need:

- A fully assembled, wired, and configured robot
- Robot
- Driver Station
- Links to programming and reference guides
- Engineering Notebook
- Gamepad
- Computer

Getting Started

BEFORE THE START OF CLASS:

- Have links available to the programming guides for teams to reference as needed.
- Learn how the robot programs use a specific configuration: hardware location, signal (data) to hardware, and hardware response.

DURING CLASS:

- Have teams get into their groups at the start of class.
- Instruct teams to keep important information in their Engineering Notebooks.
- Provide teams with links to the programming guides.
  - XRP
    - XRPCode – Integrated Development Environment
  - *FIRST* Tech Challenge
    - REV Robotics – Programming Drivetrains
    - *FIRST* Tech Challenge Docs
  - *FIRST* Robotics Competition
    - *FIRST* Robotics Competition WPILib
    - *FIRST* in Michigan – Virtual Robotics Studio
- Allow teams to discuss the outcomes of the Getting Started activity as time allows.
- Present teams with the scenario in their activity. They must think of other cases where they had to troubleshoot a device. If they get stuck, encourage them to think about other scenarios, such as times when:
  - Their computer wouldn't connect to the Internet.
  - The display from their game console wouldn't show on the TV screen.
  - The TV cable service was interrupted.

Student Tasks

- In the previous activity, teams copied and tested a program that allowed them to drive their robot using a gamepad.
- In this activity, students will troubleshoot motor control and gamepad use values. Teams will practice *Coopertition*® and *Gracious Professionalism*® by helping each other troubleshoot code.

Remember

Troubleshooting is like strength building. You might start at minimal weights, but the more you practice and challenge yourself, the better you get at it.

TASK 1: TEST AND DOCUMENT

Test and Improve:

- By now, students should have opened and tested a basic drive/drivetrain test program, saved it, and recorded the name in the Engineering Notebook.
- Teams will test the program by attempting to run and drive the robot. As the teams do this, please encourage them to check the prompts they are getting on their Dashboard/Driver Station.
- These prompts may provide clues to the source of the problem!
- Direct teams to record the Dashboard/Driver Station prompts in their Engineering Notebook.

Career Connection

Documenting error prompts and their fixes is essential to robotics technicians, programmers, coders, and vocations involving a technical skill.

TASK 2: INVESTIGATE

Brainstorm and Explore:

- In this task, it is crucial that teams independently try and discover the source of the problem. This is what troubleshooting is all about. Ask teams guiding questions that help them achieve their objective.

For example:

- What is the Driver Station telling you when you attempt to run the program?
  - Does the prompt give you any hints about what the problem might be?
- At which step do you think your robot is failing?
  - Is it in the wireless configuration, program code, or hardware setup?
  - Is it a combination of multiple steps?
  - Have you checked each step to see if it does its job correctly?
- Students can also draw upon their previous experience running their robot in the drag race. Comparing and contrasting the previous programs they tested and those not working can help troubleshoot. Students should refer to the programming documentation to understand how data/communication is transferred in the program.
- Students will be altering their code so that the robot does not operate the way they want it to. They then will exchange robots and programs (or have students move) with another group (alliance). They will try to troubleshoot the alliances code for five minutes. At the end of five minutes, they can then work together to share and discuss what has changed and how they fixed the problem.

TASK 3: SOLVE AND DOCUMENT

Design and Prototype:

- By now, teams should have listed one or many possible causes for the robot's malfunctioning.
- Encourage them to think through the causes and consider the possible process for fixing the problem.
- Direct teams to keep all their notes and ideas in their Engineering Notebooks.

Test and Improve:

- Teams should test their solution and observe the results.
- If the problem persists, teams can repeat the same process for what they think is another possible cause.
- Regarding troubleshooting, what teams learn is sometimes more important than the solution. Encourage students to document their solutions and takeaways from this experience in the Engineering Notebook.
  - What technical skills were necessary? Which personality traits helped the most?

Questions such as these might help:

- What is the relationship between a hardware setup and the program?
- How did character traits and skills such as patience, persistence, and resourcefulness play a role in troubleshooting?

- What is the best way to become better at troubleshooting?

Guiding Questions

- The teams did the troubleshooting for the robot. When it comes to the ball game robot, who will do the troubleshooting?
- Why is it important to provide a source of troubleshooting for the ball game robot?
- How can you change your code to improve your robot's performance? What must change?
- How can you apply previous experiences in troubleshooting to the current situation?

Suggested Extensions/Modifications

- Extension: Encourage teams to imagine situations where their robot malfunctions. How will this impact the playing field or the people around it? Might there be safety hazards? To mitigate hazards and damage, what fail-safe modes should be preprogrammed into the robot?
- Modification: Create alliances among teams. Pair strong teams with less proficient teams and let them work together to troubleshoot a working solution before other alliances.

Teacher Reflection Questions

- Were my students able to make a connection between this activity and how it could help address their ball game robot?
- Were they able to see the relationship between a program and hardware setup?
- Are those who didn't reach their objectives encouraged to try troubleshooting again?

Student Artifacts

In their Engineering Notebook:

- Documentation of the symptom(s), cause(s), and solution(s) to the problem.
- Documentation of the soft skills and technical skills in use.
- Documentation of crucial troubleshooting strategies they have learned.

Checkpoint

- Have students explain how a hardware setup and a program should be compatible.
- The problem and the team solution were documented in each student's Engineering Notebook.
- Documentation of the takeaways is provided in the Engineering Notebooks.

Activity 4: Think like a Robot

Driving Questions

- How can we write directions for a person to do exactly what we want and nothing else?
- How can we get our robot to do exactly what we want and nothing else?
- Why is this important?

Objectives

- Teams will create instructions that direct their teacher or a team member to do a task.
- Teams will create, outline, and revise instructions for a teammate to complete a task while also considering possible obstacles the teammate may encounter.
- Teams will consider the different tasks their robot could perform and then write instructions that would help it complete its assigned task.

Materials

- Each team will need:
- Engineering Notebook
- Self-Reflection Rubric
- A pen or pencil

Getting Started

BEFORE THE START OF CLASS:

- Write on a sticky note or print a card one of the following tasks:
  - Write the words "*Gracious Professionalism*" on the board.
  - Fold a paper airplane.
  - Walk in a straight line across the room.
  - Draw a picture of a cat on a piece of paper.
  - Tie a shoe.
  - Pat your head and rub your stomach in a circle at the same time.
- Each team should be given one random task card.
- Prepare the teaching space to create obstacles that might interrupt a series of instructions. This could include obstacles that might complicate a team's instructions.
- Familiarize yourself with key terms: loops, functions, conditionals, parameters, timed or linear based, iterative or command based.
- Prepare a set of materials students need to retrieve and put away during Task 2, such as:
  - Rulers
  - Textbooks
  - Tools
  - Posters
  - Chairs
  - Highlighters
  - Erasers
  - Computer mouse

DURING CLASS:

- Have teams get into their groups at the start of class.
- Instruct teams to keep all their work in their Engineering Notebooks.
- Allow teams to discuss the possible alternatives to their successful instructions as time allows.
- Introduce the concept of algorithm development within computational thinking. Computational Thinking and the Engineering Design Process
  - Algorithm development is the step-by-step instructions that programmers need to write for devices, computers, programs, and robots to be able to do their job. Teams need to think of cases where they use step-by-step instructions in their lives. If teams get stuck, encourage them to think about the steps required to perform a task, like folding a towel or brushing their teeth.

Student Tasks

TASK 1: PLOTTING A COURSE

Design and Prototype:

- Give each team a random task card from this list:
  - Write the words *Gracious Professionalism* on the board.
  - Fold a paper airplane.
  - Walk in a straight line across the room.
  - Draw a picture of a cat on a piece of paper.
  - Tie a shoe.
  - Pat your head and rub your stomach in a circle at the same time.
- Explain that teams should create step-by-step instructions that you or another team member will follow to complete the task on the card.
- Teams must create instructions that are extremely specific. For instance, simply telling you to walk or write does not fit the computational thinking criteria. Teams must break their complex instructions into smaller, simpler steps, such as:
  - Place your right foot a set distance forward.
  - Place your left foot a set distance beyond the right foot.
- Students should record each step of their instructions in their Engineering Notebook.
- Each step of the student's directions should be clear and lead directly to the next instruction.

Test and Improve:

- Give teams time to work and then ask them to read instructions to you or a team member without telling you the task.
- Follow their instructions exactly.

Career Connection

Documenting programmers often need to include a wide range of instructions to complete a seemingly small task such as:

- When a user presses "Play," start the movie.
- When a user presses "On," turn on the lights.

TASK 2: NAVIGATION

Design and Prototype:

- Teams should write instructions for retrieving a misplaced object and returning it to its correct location in the classroom.
- Teams will take turns providing instructions to a member of their team.
- Instruct teams to use their Engineering Notebook to document any issues they encounter as they work. Issues may include:
  - Incomplete or unclear directions
  - Obstacles in their path
  - Intersecting with another team's path
- Be sure to tell the teams which area they will be working in, which will help eliminate confusion and allow teams to focus on their task.

Test and Improve:

- Teams will now apply what they have learned about computational thinking to a more complex task.
- The object the students are tasked with retrieving should be easily sortable, i.e., a book to be placed in alphabetical order or a LEGO® component that belongs with similar pieces.

TASK 3: ROBOT ALGORITHMS

Design and Prototype:

- Teams will pick a role they think their robot might fill in their ball game.
- When the teams have decided what sort of task they will focus on, they should write a very basic set of instructions.
- Their instructions should consider the following:
  - What power or speed will the robot move at?
  - How will it know how far it has gone?
  - How will it know to correct errors?
  - What tells the robot when it is finished?

- Teams should also consider obstacles their robot may face in carrying out the tasks.

Test and Improve

- Teams will then have a team member act out the directions they plan to give the robot. They should consider:
  - Did the robot/human operate at the desired speed?
  - Did the teammate know how far to go and when to change direction?
  - Was they able to make corrections if they went off-path?
  - Did they know when they were finished?

Guiding Questions

- Why do the instructions programmers write need to be so specific?
- What sort of tasks do you perform on a daily basis that can be broken into small sets of instructions?
- How would your instructions differ if they were being performed by a robot?
- What are some common obstacles that programmers need to consider when writing code?

Suggested Extensions/Modifications

- Extension: Provide teams with random obstacles during Task 2.
- Extension: Encourage teams to imagine what the first set of coded instructions may have been written for. What would that code have looked like?
- Extension: If teams finish Task 1 or Task 2 quickly, ask teams to pair up and take turns creating different objectives for others to write instructions for.
- Extension: Create a route from one end of the classroom with various obstacles or distractions. The route can be the same for all teams, or each team can have its own. If each team has their own route, ensure that the challenge level for all teams is similar. The task can be set up to practice *Coopertition* between teams.
- There are two ways teams can win. One way is the time it takes for a team to create a set of successful instructions. Another is completing the task successfully.
- Modification: Create alliances among teams. Pair strong and less proficient teams and let them work together to write instructions.

Teacher Reflection Questions

- Were my students able to make a connection between this activity and how computational thinking and algorithms could be useful in addressing their ball game?
- Were students able to create simple instructions to complete tasks?

Student Artifacts

- In their Engineering Notebook:
- Getting Started: A list of everyday tasks that can be broken into step-by-step instructions.
- Task 1: A set of instructions for directing their teacher to complete a task from the task list.
- Task 2: A set of instructions for directing their teammate to retrieve and replace a random classroom object.
- Task 2: An outline of the steps needed to retrieve an object from across the room and place it where it belongs.
- Task 3: Documentation of possible algorithms in the ball game.

Checkpoint

- Have students explain how computational thinking and step-by-step instructions relate to robotics.
- Have students explain why programming is an important part of their robotic solution for their ball game and how it enhances their robot's functionality.

Activity 5: Let's Get Moving

Driving Questions

- Can we program our motors to change the way the robot moves?
- What role will the programming of the motor play in their robot design?

Objectives

- Teams will learn how to:
  - Rotate the direction of a motor.
  - Set the stopping of the motor (immediate brake vs rolling to a stop).
  - Set the power of the motor.
- Teams will use motor code to create a program so their robot can complete a challenge.
- Teams will use their Engineering Notebook to document their findings and other important information.

Materials

Each team will need:

- A fully assembled, wired, and configured robot
- Driver Station
- Gamepad
- A computer to access programming tools.
  - XRP
    - XRPCode – Integrated Development Environment
  - *FIRST* Tech Challenge
    - *FIRST* Tech Challenge Docs
    - REV Robotics – Hello Robot – Robot Control
  - *FIRST* Robotics Competition
    - *FIRST* Robotics Competition WPILib – Hardware APIs
    - *FIRST* in Michigan – Virtual Robotics Studio
- Masking tape to mark starting and midpoint locations for Task 4

Getting Started

BEFORE THE START OF CLASS:

- Choose an area of the class to set up a challenge course for Task 4. Use masking tape to mark the starting point a distance from a wall and a midpoint between the starting point and the wall.
- Familiarize yourself with key terms used in the programming environment: *power*, *direction*, *data types*, *variables*, *IDE structure* (OpMode, Command-Based Programming, Timed Robot)
  - XRP
    - XRP User Guide
  - *FIRST* Tech Challenge
    - *FIRST* Tech Challenge Docs
    - REV Robotics – Hello Robot – Robot Control
  - *FIRST* Robotics Competition
    - *FIRST* Robotics Competition WPILib – Hardware APIs
    - *FIRST* in Michigan – Virtual Robotics Studio

DURING CLASS:

- Have teams get into their groups at the start of class.
- Instruct teams to keep important information in their Engineering Notebooks.
- Allow teams some time to discuss the examples of motors in their Getting Started section.
- Ask students to think of examples of different motors they may encounter in the real world. If they get stuck, encourage them to think about motors they may have seen in the past, such as:
  - Machines at a construction site
  - Different cars they see on the road
  - Electric appliances such as hairdryers
  - Large machines that have moving parts, such as trains and planes
  - Battery-powered motors found in some toys

Note

The variety of programming options available to the teams might seem overwhelming to them at first. Encourage teams to test the sample programs before making program changes. Then, make small changes at a time to prevent overload.

Student Tasks

TASK 1: EXPERIMENT AND LEARN ABOUT MOTORS

Test and Improve:

- Teams will create a copy of their program and name it “their team name motor test 1.”
- Students will be working with the part of the program that affects the direction the motors rotate and the level of power the motors use to rotate.
- Students will learn how to use different variables to alter their motor direction by experimenting with their programming environment.
- The teams will first test different ways the motors can be directed to rotate. They will experiment with rotating the motors forward and reverse in the same and alternate directions.

Design and Prototype:

- Teams will experiment with different number settings linked to the motor power.
- Negative numbers will move the motors in reverse. Numbers between 0 and 1 or 0 and -1 will move the motors more slowly. 1 is the fastest setting for moving motors forward, and -1 is the fastest they can move in reverse.
- If the motor direction was initially set to reverse, a negative number will cause it to rotate in the direction of the forward arrow instead of reverse.
- Have teams document and answer the following prompts in their Engineering Notebooks:
  - What they learned, problems they ran into, and how they solved them.
  - Where they found the code that sets the power of their motors.
  - Why do they think the program is written that way? In other words, how is data transferred in the program from one place to another? How might they read the program flow to understand how it is being transferred to the robot?

Tip

On the robot, the motors driving the wheels are installed opposite one another.
Rotating the motors in the same direction will cause the robot to spin in place instead of moving forward or backward. This makes it important to reverse the direction of one of the motors.

- Teams will then set their motors to rotate at different power settings.
- Teams will discuss and plan how to change the motors.
- They should consider improving the process to eliminate human error in control.
- Record how the number attached to a power block changes the way your robot moves.

TASK 2: INVESTIGATE: BRAKING OR ROLLING?

Brainstorm and Explore:

- Teams will create another copy of their previous program and experiment with time and motor control algorithms.
- *FIRST* has technology for motors that can dictate the behavior of the motors.
- *FIRST* Tech Challenge
  - Blocks Tutorials – Using Encoders
- *FIRST* Robotics Competition has encoders that can be used on motors to improve performance.
  - *FIRST* Robotics Competition WPILib – Using Encoders
- In the instructions they wrote for the teacher in Activity 4, teams had to consider the stopping and starting points.

Tip

Braking mode is crucial to autonomous robotic operation using sensors. Programing your robot to stop if it encounters a block may fail if the robot rolls to a stop upon detecting an obstacle.

- When teams work with motors, the instructions they write will guide their robot using measures of power, direction, and position rather than telling them to stop at a specific desk.
- They will have to consider the instructions they give to the robot when the motors have power and when they don't.
- Motor variables give the robot two options when the power stops flowing to the motors: they can either brake or float.
  - Brake: The motors will break and stop the robot from moving further.
  - Float: The robot will continue coasting after power stops flowing to the motors.

Test and Improve:

- Teams will experiment with the variables and discover that telling their robot to brake when there is no power gives them more control over where the robot comes to rest.
- Have teams document and answer the following prompts in their Engineering Notebooks:
  - Record which algorithm controls the behavior of their motor when it is zero-powered.
  - Describe how they can use this block to make their robot roll to a stop when the robot operator stops pushing the gamepad stick.
  - Describe how they can use this block to make their robot brake to a stop when the robot operator stops pushing the gamepad stick.

Tip

Seeing different uses of each programming algorithm will be a big help to students when it comes time to choose from the assortment of options during tasks in later activities.

TASK 3: ROBOTS! ON MY COMMAND

Design and Prototype:

- Teams will examine the ways to monitor time on their programming tools.
  - *FIRST* Robotics Competition
    - *FIRST* in Michigan – Virtual Robotics Studio
  - *FIRST* Tech Challenge
    - *FIRST* Hour of Code Playlist

Test and Improve:

Teams will:

- Attempt to move the time algorithm to different locations in their program.
- Remove the time algorithm.
- Teams will see that altering and moving a time block can affect the way a program runs and the task the robot completes.
  - What sort of instructions did they give that were directed at moving?
  - What sort of instructions were given to prepare for unforeseen obstacles?

TASK 4: DESIGN, IMPLEMENT, AND COMPETE!

Identify the Problem:

- Show teams the area you've marked with tape before class and tell them the objective of the challenge:
- Teams need to create their program based on what they've learned in Tasks 1, 2, and 3.
- Remind them to reference the links you have provided during the activity.
- Understanding different variables such as zero-power behavior, power, or encoder use can be essential for students learning to develop more advanced algorithms.

Design and Prototype:

- Teams will create a strategy, plan, and design.

Test and Improve:

- Have teams take turns testing their programs.
- All teams should pay attention to every teams' trial to learn from their peers' successes and mistakes.
- There are many ways to write a program to complete the task successfully.
- Ask successful teams to share their instructions after every team has tried the challenge.
- The following questions might help guide teams as they create their programs:
  - What algorithm is the easiest to use when moving a small distance and then stopping?
  - How can we ensure our robot stops when it gets to the midpoint?
  - What algorithm gives us the best control over our robot when moving small distances?
- After all teams have finished the challenge, have them record their program in their Engineering Notebooks.

Guiding Questions

- What sort of real-world situations would require complex algorithms?
- What are some reasons a robot might need a motor to coast or not have power in a configuration?
- How can you change your program to improve your robot's accuracy when stopping? What must change?
- How can you apply previous experiences with troubleshooting to the current situation?

Suggested Extensions/Modifications

- Extension: Ask teams to use as many algorithms as possible to complete Task 4. What blocks or text code were easy to integrate? What blocks or text code required more thought?
- Extension: Ask students to use as few algorithm options as possible. What sort of challenges do you face when your options are limited?
- Modification: Create alliances among teams. Pair strong teams with less proficient teams and let them work together to race and defeat other alliances.

Teacher Reflection Questions

- Were my students able to make a connection between this activity and how different algorithms could be helpful in addressing their robot challenge?
- Are any of the motor algorithms more challenging to students than others? If so, why?

Student Artifacts

In their Engineering Notebook:

- All reference guides
- Document a summary of each of their team's codes or algorithms in Tasks 1-4.
- Record essential troubleshooting strategies they used if they encountered a problem putting new codes into their program.
- Record how their robot moved when they added the algorithms.
- Record the algorithm their team used to complete the challenge.
- Record their thoughts on the types of motors and their uses that will be used in the ball game.

Checkpoint

- Have students list the different algorithms they experimented with under the utilizing motors.
- Have students briefly describe each of the different motor algorithms they used.
- Inspect the programs the teams created in each task for assessment.
- Check Engineering Notebooks.

Activity 6: Information Exchange

Driving Questions

- How do we retrieve data from our robot?
- What sort of data can we get from our robot?
- How can we use data feedback to improve our robot?

Objectives

- Teams will learn how to get data from the robot to the Driver Station.
- Teams will talk about the ways data feedback is used in the world.
- Teams will discuss how data feedback works and why machines use it.
- Teams will discover different ways their robot can use data and how data feedback can improve the ways they use their robot.
- Teams will explore the different data output algorithms that are in their programming tools.
- Teams will discuss different ways their robot performance can be improved by data feedback.

Materials

Each team will need:

- A fully built and wired robot
- Driver Station
- Laptop
- Gamepad
- Engineering Notebook
- Pairing and Configuration Guide

Getting Started

BEFORE THE START OF CLASS:

- Ensure you have links to the programming tools for your program:
  - *FIRST* Robotics Competition
    - *FIRST* in Michigan – Virtual Robotics Studio
  - *FIRST* Tech Challenge
    - *FIRST* Tech Challenge – Blocks Tutorials Playlist
    - REV Duo Documentation – Hello Duo
- Acquaint yourself with some of the different kinds of information students can program their robots with to get data from the robot to the Driver Station.
- Take some time to learn about the rovers and probes that have been sent to Mars.
- Familiarize yourself with the various data feedback rovers designed to send back to Mission Control.
- Read through the Getting Started activity in the student guide.
- Find a few instances of data output around the classroom. Examples could include temperature readings from a thermostat or the battery level on a mobile device or laptop.

DURING CLASS:

- Have students get into their teams.
- Remind students to keep important information in their Engineering Notebooks.
- Remind groups to use their programming tools.
- Take a minute or two to talk about rovers and probes that have been sent to our moon and Mars.
- Ask questions such as:
  - Why do space agencies send machines to other planets?
  - What sort of information are space agencies looking to gather? What forms of information can space agencies collect from probes and/or rovers?

- Data feedback can also be called telemetry. It is the process of a device or machine recording information and then sending or displaying that information. Ask students to think of a few examples of telemetry. Students may talk about cars that use a speedometer to display speed or a fuel gauge to show how much gas is left in a tank. Telemetry is more common than gauges in cars. Some other examples of telemetry are:
  - The GPS location that a phone can provide
  - The number of steps measured by a pedometer or smart device
  - A heart rate monitor or other medical machines that gather information in a hospital
  - An airplane's altitude, speed, pitch, yaw, and cabin pressure
  - A depth finder on a boat that displays how deep a body of water is
  - Sonar and Radar devices that provide the location of objects in the sky or underwater
  - Humidity, temperature, wind speed, and UV index measurements that weather stations record and disseminate
- As technology becomes more prevalent, much of the information we depend on comes from telemetry. We check the weather in the morning, can see how far away a bus is from our stop, and can even see how far away a delivery driver is with a pizza.
- Present your students with the scenario in the Getting Started section of their activity. In the Getting Started section, students are asked to picture themselves in NASA's Mission Control watching the landing of a Mars rover. After the rover moves a short distance, it becomes stuck. Teams will use the situation to acquaint themselves with the concept of telemetry.
- If students have trouble thinking of obstacles a rover on Mars may encounter, have teams spend a few minutes researching past missions to Mars. Looking up images or watching a video will help students visualize some issues a rover might encounter.

Student Tasks

TASK 1: SAVE THE ROVER

Brainstorm and Explore:

- At this point, teams have had an opportunity to brainstorm examples of different forms of data feedback.
- Their first task is to use what they've discovered through their discussions to devise ways to free the rover based on information it may be sending back to Earth.
- Students will pick a reason for the rover to have stopped that could be identified using the data being sent to Mission Control.
- It is up to each team to think of a problem that could be solved using the help of data feedback. Some examples of ideas they may come up with include a buried rock or obstacle that the rover cannot drive over, a steep incline that the rover can't climb, or an issue with one of the rover's motors/tires. A buried rock or obstacle could be recorded in either photo or video format using a camera. A steep incline could be measured using gyroscopes and tilt sensors. If the rover's tires are not moving, the motors could send information about the rate at which they are rotating.

Test and Improve:

- Students might come up with an example of wheel rotation. If the wheels are moving forward but cannot gain traction, they might need to create instructions that direct the robot to change the direction the motors are spinning. They may suggest that the rover sends information from a proximity detector, indicating it has become stuck against a large object such as a rock. They need to create instructions that tell the rover to reverse and find a new path forward. Teams should relate their solution directly to a type of telemetry. Using the same form of telemetry, teams should be able to check if their instructions could get their rover moving.

TASK 2: WITH GREAT POWER COMES GREAT TELEMETRY

Design and Prototype:

- Students should look for code examples that display motor power. Consider helping students follow the data from their hardware (left motor) through the robot controller and then display it on the Driver Station.
- Examples of code that can display motor/servo data:
  - XRP
    - XRP-Creating a Dashboard
  - *FIRST* Tech Challenge: Telemetry Data
    - *FIRST* Tech Challenge Docs – Controlling a Servo
  - *FIRST* Robotics Competition: Shuffleboard
    - *FIRST* Robotics Competition WPILib – Telemetry
    - *FIRST* Robotics Competition WPILib – Robot Telemetry with Sendable
    - *FIRST* Robotics Competition WPILib – Shuffleboard

Test and Improve:

- After selecting the program, teams should test it. If their data feedback code is set up correctly, when they save and run it, the power level for each motor will be displayed on their Driver Station.
- The data they are retrieving and sending to the Driver Station should be displayed.

- This is also a great time to point out different types of data (string, integer, double, float, and so on). These data types are often covered in programming courses and are just applied here.
- Have teams document and answer the following prompts in their Engineering Notebooks:
  - Where is the telemetry displayed?
  - What format the telemetry is displayed in?
  - What can they learn about their robot based on the telemetry it is sending?

TASK 3: POWER ON AND POWER OFF TELEMETRY

Design and Prototype:

- This task aims to get teams to use what they learned in the last activity and apply their new knowledge of telemetry.
- They should gather data to tell them the condition or state of a motor, for example if power = 0 then robot is stopped. This data should be sent back to the Driver Station.

Test and Improve:

- Students should code to their telemetry program to set their robots to determine the state or condition of a motor position. The telemetry tools students use in this task are not prescriptive, so it takes some experimentation. Rather than sending a number, the information the robot will send will be in text form, so they will need to use the telemetry methods that express that type of data.
- When teams have their program running correctly, ask teams to record the differences between the telemetry data they used in Task 2 and this task.
- Have teams document and answer the following prompts in their Engineering Notebooks:
  - Issues they had to troubleshoot and their solutions
  - The name of the program and title that correctly displayed the power and stop mode data on their Driver Station
  - Comparison of the telemetry data from Task 2 and Task 3

TASK 4: THE MORE TELEMETRY, THE BETTER

Brainstorm and Explore:

- Ask teams to think of an example of a task their ball game robot might be given. In that role, what might the most helpful type of telemetry be?
- Ask students to think of their robot and consider the following:
  - How does the robot collect information about its performance?
  - What sort of telemetry can the robot send?
  - Are there any game-specific tasks that can be performed better using telemetry data?
- Time permitting, allow students to experiment with different types of telemetry data. Remember to reserve some time for reflection, checkpoint, and clean up at the end of class.

Test and Improve:

- Teams should try to edit the program they have created to send information that can provide feedback that will help them make changes.
- As they attempt to set up more telemetry code, students will be exploring the different options they have in the programming tools. The more opportunities students have to navigate the menus, the more familiar they will become with the options available.
- Before class ends, ask teams to document how they have used telemetry and any telemetry they found most helpful in their Engineering Notebooks. Not all the telemetry data they've created in this task must work immediately. The objective is to experiment with the possible telemetry options available to them.
- Explain to teams that experimentation does not always mean determining the correct task-performing method. Discovering what does not work can be as important as learning what does.
- Model creative problem-solving and persistence. This is a great way to inspire the same behavior in your teams.

Guiding Questions

- In past activities, where would telemetry have been helpful?
- Aside from the Mars rover example in this activity, what sort of problems would telemetry be used to solve in the real world?
- What component of the robot is providing the data being displayed in the Driver Station?
- What are some instances of data you may have come across in the past few days?
- Are there forms of data that you rely on every day?

- What would you do if you didn't have access to the data you use on a daily basis?
- Does the location of telemetry data in the program code matter?
- What part of the code libraries do the telemetry data need to appear in?

- If you notice a team using creativity and persistence to solve a problem, ask them to share what they are doing with the rest of the class.

Suggested Extensions/Modifications

- Extension: Challenge students to list 10 different types of telemetry a Mars rover might send back to Earth.
- Extension: Ask students what instruments the Mars rover would use to collect telemetry.
- Extension: Ask students to create a list of types of information their robot might record to be more helpful.
- Modification: Ask students to record questions about the telemetry a Mars rover might send and give them time to research past rovers. Students should record what they found in their research and their initial questions in their Engineering Notebooks.
- Modification: If a team is having difficulty thinking of different types of telemetry, pair them with another group so that they can brainstorm together. Prompt them with examples of telemetry they can find in the classroom, such as a thermostat or devices that display battery levels.

Teacher Reflection Questions

- Were my students able to make a connection between this activity and how telemetry could help address their ball game?
- Were students engaged with the Mars rover scenario? If not, what scenarios would be more easily relatable for future conversations?

Student Artifacts

Teams should have documented the following in their Engineering Notebook:

- Getting Started brainstorming ideas.
- Step-by-step instructions for Task 1.
- Explanation of the meaning of telemetry applications.
- The name of the program they created to experiment with telemetry and the code they could integrate successfully.
- Ideas about how telemetry can improve their robot's design.
- Explanation of how telemetry could be helpful in the ball game they are trying to solve.

Checkpoint

- Have students explain the meaning of telemetry and the data it provides.
- Have students provide examples of data in the real world and their lives.
- Have students explain how data could help solve their ball game.

Activity 7: I'm in Complete Control

Driving Questions

- What sort of tasks can we make our robot do?
- Can we create a set of instructions based on what we want our robot to do?
- How do we translate a set of instructions for completing a job into a program for our robot?

Objectives

- Teams will decide on a task for their robot to perform that will illustrate the level of control they have over its actions.
- Teams will work to break down the chosen task into step-by-step instructions.
- Teams utilize programming libraries and edit them so their robot can perform the specified jobs.
- Teams will use the sample code they are familiar with to build their program.
- Teams will document important information in their Engineering Notebooks.

Materials

Each team will need:

- A fully assembled, wired, and configured robot
- Programming tools and Driver Station
- Gamepad
- Laptop
- Engineering Notebook
- Reference guides
- Masking tape for teams to mark positions in the room

Getting Started

BEFORE THE START OF CLASS:

- Make sure the programming templates and tools are accessible to teams.
- Familiarize yourself with crucial vocabulary: iteration, innovate.
- Set up the ball game field designed previously.
- Spend some time with programming tools; explore sample codes not previously used.

DURING CLASS:

- Give teams time to discuss the Getting Started section of the activity.
- Allow teams some time to discuss domestic and industrial robots:
  - Do any students have a robot in their home? What is its purpose?
  - Do teams think having robots in the home/workplace is good? Why?
  - Do teams think that robots will become more prevalent in the coming years? Why or why not?
- Provide teams with the objects utilized in the ball game.
- Instruct teams to keep important information in their Engineering Notebooks.
- Provide teams with programming tools as needed.
- Present teams with the scenario in the Getting Started section of their activity. They should think of some examples of different jobs that robots perform in the real world currently or will in the future, such as:
  - Assembly lines
  - Automatic vacuum cleaners
  - Self-driving lawn mowers
  - Learning to code
  - Delivering packages
  - Piloting a self-driving car

- In previous activities, teams were given instructions on what code to add to a program and an explanation of what each algorithm does. In this activity, teams will use what they've learned to create their program to perform a chosen task.
- Teams are being introduced to the process of iteration. Teams will create, test, and then create iterations of their program that improve their robot's ability to complete its job.
- While teams have all the tools they need to build a program, they will likely need guidance during the activity's test, iterate, and improve section. Each task represents a different step in the development process, and providing teams with prompts and tips will help steer them in the right direction if they are struggling.

Student Tasks

TASK 1: WHAT AM I DOING?

Brainstorm and Explore:

- Each team will select or create tasks that demonstrate the level of control they have over their robots. Remind teams that the goal of the activity is not simply doing a job but showing how much control they have over their robot as it completes the job. Teams should draw on the knowledge they developed thus far in the course.

> **Note**
> The entire process, from deciding on a task for their robots to perform to building the program that will allow them to carry out the task, should be student directed.

In past activities, teams:

- Learned how to control their robot using the sticks on their gamepad. Teams have competed in a drag race against other teams and have had an opportunity to become familiar with how quickly their robot moves forward.
  - Used computational thinking to break tasks into step-by-step instructions.
  - Experimented with the direction of spin, power, and stop motor algorithms.
  - Learned the definition and purpose of telemetry and experimented with their robot's data feedback capabilities.
- If teams are having trouble deciding on a task for their robot, provide them with an example of a real-world robot that performs a task and break down how it completes it:
  - Robotic vacuums move around a house or building and clean using a built-in vacuum. The robots have to avoid colliding with objects and be able to navigate a room without getting stuck in a corner.
- Provide teams with the option to select from the objects you prepared before class. They may want to choose one if they are thinking of a task that involves delivering or manipulating something.
- In the last activity, teams were introduced to data feedback and given a scenario where telemetry had a practical application. In this activity, gaining feedback from the robot will allow teams to show that they can make a robot do what they want.
- After teams have decided on tasks, they should think about what completing the tasks entails.
  - If they are delivering an item, where will the robot start when it begins?
  - Where will the robot be when it finishes?
  - Should it make more than one stop along the way?

Design and Prototype:

- Teams will create a set of instructions that will guide their robot to complete their task. When they have a set of instructions, they will test them by having a team member follow the steps. If teams could not think of many details involved in completing their task when they decided on it, they will likely discover some when writing and testing their instructions.

> **Note**
> To provide teams with opportunities to innovate, provide them with objects to act as obstacles in the area they will be performing their tasks.

- These examples of prompts may help the students add detail to their instructions:
- How many steps does it take to complete your task? Can they be simplified?
- What sort of data would be available from your robot when you are working on completing the job?

Test and Improve:

- Teams should use tape to mark important locations to be aware of during the task, such as a line of tape to place the robot's front wheels on at the starting point and a line indicating where the robot will be when it completes the task. Encourage teams to record their instructions and any errors and corrections they make to them in their Engineering Notebooks.
- The troubleshooting of errors and the corrections they make to their instructions are the first steps in the test, iterate, and improve process.

TASK 2: TURNING INSTRUCTIONS INTO A PROGRAM

Design and Prototype:

- Teams should now have a relatively detailed breakdown of their task. As teams have learned over the past several activities, step-by-step instructions won't be enough to get their robot to do what they intend.
- The instructions they've created need to be translated into an algorithm that provides their robot with all the necessary information to complete the task.
- Teams should make a copy of their previous program and make additions and edits.
- The name of the copy they make should reflect the tasks they have their robot complete.
- Remind teams of all the different block/text code they have used so far and how they gave their robots the information necessary to complete the previous tasks.

Test and Improve:

- As teams create their new program, they will use algorithms from previous activities but will also have the option of looking for block/text code they haven't used yet.
- Ask teams to use comments in their program to indicate places they have used algorithms they haven't used before this activity.
- As they build their program, prompt them to look at the different options under the programming tools available encoders. In Task 3, they will be testing their program, so now is an excellent time to think of all the algorithms that might help them demonstrate their control over their robot.
- Have teams document and answer the following prompts in their Engineering Notebooks:
  - Document all the instructions they came up with for Task 1.
  - Document their program.
  - Take a screenshot of their program.
  - Document any algorithms they tried and then write a short description of what you want them to do. Describe why you've decided to use them.

TASK 3: TEST, ITERATE, AND IMPROVE

Test and Improve:

- Now that teams have a program designed to complete their task, it is time for them to test, iterate, and improve.
- While teams are using the programming tutorials, they will likely result in some issues in their code, especially if they are adding ones they haven't used before.
- In past activities, teams have used their troubleshooting abilities to find algorithm errors and fix mistakes that may have occurred along the way.
- During this task, they will use their troubleshooting skills to improve their program instead of simply making corrections.
- As teams begin testing their program, they might encounter issues getting their robot to complete its job. Some teams might not be able to move their robot at all because of the changes they've made to the program. While they are testing their program, ask teams:
  - Where are the algorithms you've added to the original template?
    - If teams have trouble making their robot move, they should compare their program to the original template.
  - What are they setting up hardware in initialization correctly?
    - The initialization section of the program is used to provide information that the robot needs to know before it starts following the algorithms that run after starting or enabling the program in the Driver Station.
  - What algorithms are in the code section that runs after the starting or enabling?
    - Refer to the previous programming tools to identify where hardware setup occurs and how to initialize hardware parameters.
  - Is your robot able to stop accurately?
    - Using items such as encoders can help teams operate their robots more smoothly. Refer to the previous programming tools to guide students in using encoders.

- If you're moving an item, does it fall off? Can you move the item from the starting point to the stopping point as fast as possible?
  - Encourage students to consider both mechanical and software means of controlling an object. Often, you can find a perfect combination of both that will create the perfect combination of mechanical control and software feedback to increase efficiency.
- If you are using new or different inputs on the gamepad, what algorithms are they attached to in your program?
  - The buttons and triggers on the gamepad provide a different kind of input than the gamepad sticks. The gamepad sticks can provide numbers anywhere between -1 and 1 depending on how far off they are from their center, but the buttons and triggers can only register as on or off. Refer to the programming tools for identifying the inputs and transferring the correct data into the algorithm.
- Does your team need to pick up or adjust your robot while performing its task?
  - Teams should be able to create a program that allows them to complete the job they have decided on without physically intervening in its progress. If a team has to pick up or reorient their robot while working to complete its job, they have likely not included enough detail in their program. For example, if teams cannot make their robot carry an object without the object falling, ask if they could adjust their robot's speed to fix the problem. If teams are unable to complete the robot's job without colliding with an obstacle, ask them to find a path in their testing space that is further away from the obstacle. Being able to make their robot do exactly what they want it to be more important than having their robot move quickly.

- Have teams demonstrate their best program.
- Encourage the teams to make notes and evaluate their work as well as the work of other teams.
- Have teams document and answer the following prompts in their Engineering Notebooks:
  - Program improvement ideas.
  - Speed improvement ideas.
  - Program simplification ideas.
  - Program reverse direction ideas.

Suggested Extensions/Modifications

- Extension: Ask teams to use as many gamepad buttons as they can.
- Extension: If teams use their gamepad to guide their robot through its job, ask if they can edit their program to make their robot complete the task independently. If they can, what changes need to be made? If they can't, why not?
- Modification: Create alliances among teams. Ask teams to pair up and work to complete the same job.

Guiding Questions

- What are some examples of jobs where robots are used in the real world?
- What are some examples of jobs that they may be used for in the future?
- What did the programs you've created in the past few activities allow your robot to do?
- Why do you use the gamepad sticks to control your robot's movement?
- What are some other ways you could use our gamepad to control your robot?
- What part of the code libraries do the telemetry data need to appear in?
- What skills did you learn that you can use as you go forward?

Teacher Reflection Questions

- Are teams using any algorithms that weren't used in previous activities? Which algorithms were common between the programs of all teams?
- Are any of the algorithm options more challenging for teams than others? If so, why?

Student Artifacts

In their Engineering Notebook:

- Programming Tools
- Examples of jobs that robots perform now and jobs they may perform in the future.
- Document the step-by-step instructions for completing the jobs teams have selected for their robot.
- Documentation of the program the teams created in Task 2.
- Documentation of the iterations that teams created during Task 3.
- Documentation of any new algorithms that teams used for their program.

- Documentation of their ideas from Task 3.
- Reflection of successes and challenges they faced during this activity.

Checkpoint

- Have teams provide examples of jobs that robots can perform now and jobs they might perform in the future.
- Have teams describe the final version of the program they created to complete their robot's job.
- Have teams discuss the purpose and application of the test, iterate, and improve the process used in this activity.
- Document takeaways provided in Engineering Notebooks.

Activity 8: The Big Race

Driving Questions

- Can we control our robot well enough to drive it in a relay race against another team?
- Can we use the Engineering Design Process to create the best possible program for winning a relay race?
- Why is the level of control we have over a robot important, and how does it apply to our ball game?

Objectives

- Teams will use the program from the past activity to test drive the course set up by the teacher.
- Every member of the team will practice *Coopertition* and *Gracious Professionalism* as they take a turn driving through the course.
- Teams will create a program suited for a relay race around a relay course.
- Teams will test their relay program by driving around the course once.
- Teams will improve the program through iteration based on how their robot performed during the test lap.
- When their program is ready, every team member will take a turn driving through the obstacle course as quickly and accurately as possible.
- Teams will reflect on the importance of control when driving their robot.
- Teams will brainstorm and document:
  - Common mistakes people make when taking their driving test.
  - Suggestions a driving pamphlet might give new drivers to prepare for their test.
  - Real-world rules that are important when driving.

Materials

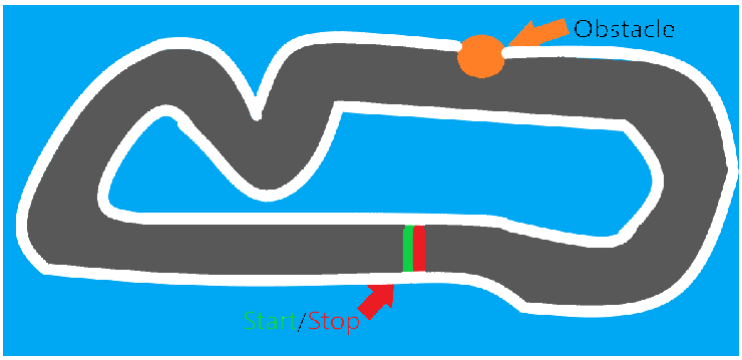
Each team will need:

- A fully assembled, wired, and configured robot
- Driver Station and laptop
- Gamepad
- Timekeeping methods include stopwatches or clocks
- A method for timing other teams as they race their robots

Getting Started

BEFORE THE START OF CLASS:

- Prepare a few courses for teams to race their robots through. The more courses you can create, the less time teams need to wait for a turn to test and race their robots. If you have multiple courses, try and make them all as similar as possible.
- The courses should be as large as possible.
- The courses should have at least two turns.
- Each course should have an obstacle for the teams to navigate around. The obstacle should be placed close to one of the sides of the course so students can steer around it with little difficulty. An obstacle could protrude out from the side boundary at a place in your chosen course.
- The courses should have clearly defined boundaries.
- The courses should be a closed circuit.
- There should be a starting point in the course that is marked. During the test run in Task 1 and the relay race in Task 4, students will pass the gamepad to a new team member each time their robot crosses the start line.



DURING CLASS:

- Present teams with their Getting Started activity and questions. Provide teams with some time to answer the questions in their Engineering Notebooks.
- Teams will be racing through the courses twice. The first is an opportunity for them to take notes on the best ways to complete the course.
  - After teams have tested their robots in the course, they will create a program for the relay race.
  - Students will draw from the previous activities in this module when building their robot and program.
  - Teams might often find that there is a combination of errors that can occur between hardware and software. For example, a loose wheel or motor can affect how the program runs. Guide students to use *Gracious Professionalism* while troubleshooting these issues.
- Teams can drive one lap of the course to test and improve the program their team created.
- In Task 4, team members will take turns running through the course until every member has gone. When a team is racing, have another team time them and make a note anytime a robot hits an obstacle or drives outside of the course boundaries. Driving outside the course boundary or encountering an obstacle will be a fault. For each fault a team collects, five seconds will be added to their final time. The team that completes a course with the best time wins that course. If time permits, teams can try multiple courses.

Student Tasks

TASK 1: TEST RUN

Brainstorm and Explore:

- As students test the course, they should remember what a driver would study before their driving test, such as slowing down when making a turn.
- Teams will decide on the order which their team members will drive in and then take turns guiding their robot through the course.

Test and Improve:

- Teams will create a rough sketch of the course they are driving in their Engineering Notebooks.
- Teams will mark the location of their faults on their sketch of the course.
- Teams will discuss the cause of any collisions and possible methods of avoiding collisions in the future.
  - For example, a team might note that an obstacle is very close to the outside boundary of the course, so their robot will need to slow down and keep to the inside boundary when passing the obstacle.
  - Teams will discuss the cause of their robot driving outside the course boundary and propose possible solutions for avoiding exiting the course while racing in the final task.
  - For example, if members of a team notice that their robot continues to go outside of the boundary in the same place, they may suggest slowing down when they near that section of the course.
- A description of the issues a team encounters as well as possible solutions to those issues should be recorded in their Engineering Notebook.

TASK 2: A PROGRAM MADE TO WIN

Identify the Problem:

- Teams will discuss the areas of the course that were difficult for them and list some methods for navigating those areas.
- Teams will decide what data types, if any, would be helpful while they are racing.
- Teams will discuss algorithms they've used in previous activities and ask themselves if any others may be better suited to a relay race.

Design and Prototype:

- Teams will create a new program and name it, "their team name Relay Team__."
- Teams will analyze their findings from Task 1 and build their program to suit a relay race through the course.
- Teams will consider the programs they used in previous activities. If they think any part of a program they've used before could be helpful during the race, they may draw from it for inspiration.
- Have teams document and answer the following prompts in their Engineering Notebooks:
  - A copy of their program.
  - The name of any program from previous activities that they drew inspiration from and an explanation of why they found them helpful.
  - The components of other algorithms that they integrated into their program.

TASK 3: TEST, ITERATE, AND IMPROVE

Test and Improve:

- Teams will select one team member to test out their program.
- The team member performing the test is only allowed one lap around the course.
- Before teams take their test lap, they should review the problematic areas of the course that they noted in their sketch from Task 1.
- Teams will make a second sketch of the course to take notes on areas where their robot has difficulties.
- Remind teams that they will only have one lap to test their robot. They should think of specific questions they want their test to answer.
- When they have finished their test lap, teams should return to their programming tools and record any changes or modifications they should make.
- As their team members take their test lap around the relay course, teams will record notes on the robot's performance. Remind students that the more detail they have in their notes, the more information they will have to work with while improving their OpMode program.
- Have teams document and answer the following prompts in their Engineering Notebooks:
  - Detailed notes on their robot performance.
  - A second sketch of the relay course with notes on where their robot encountered difficulties.
  - A reflection on where their robot was successful.
  - A reflection on where their robot could use improvement.
  - A breakdown of the troubleshooting process to address the difficulties they noticed during their test run.
  - The improved iteration of their robot and program.

TASK 4: TIME TRIALS

Test and Improve:

- Allow teams to pair up and create an alliance.
- While one team in the alliance is racing, the other team will be timing and keeping track of faults.
- Be sure each alliance has a device to keep track of time accurately.
- Each pair of teams will run the relay race one at a time.
- When every racing team member has completed their lap, the team that is timing will mark their final time.
- Teams will add up the number of faults from their race and multiply it by five seconds. The total number will then be added to their final time.
- After the faults are added, the team with the fastest time on the course wins.
- If time permits, allow teams to try other courses.
- You may also allow alliances to combine times and race other alliances. The best-combined alliance time wins!

Suggested Extensions/Modifications

- Extension: Ask students to include at least two forms of data in their program.
- Extension: Ask students to integrate at least two gamepad buttons into the program.
- Extension: Challenge students to complete the race without incurring a single fault.
- Modification: Pair up teams with trouble creating a working robot and have them collaborate.

Guiding Questions

- What parts of the relay course can your robot navigate easily?
- What parts of the course are more difficult?
- What parts of the robot worked well with the relay course?
- What parts of the robot need improvement?

Teacher Reflection Questions

- Did my students engage in the Getting Started Activity?
- Compared to earlier activities, have my students demonstrated improved control over their robot?
- Were teams able to create robots with a program better suited to the course?

Student Artifacts

In their Engineering Notebook:

- Notes on their team's performance in the first task using their previous program.
- A sketch that marks locations on the course where their team had difficulty in Task 1.
- An outline for the first version of each team's program
- A record of some Op Modes from past activities that teams drew from while building their current program.
- A sketch that marks locations on the course where teams had difficulty in Task 3.
- An outline of the problems the first iteration of a team's robot had during their trial laps.
- Their team's final time for completing the relay race with and without the additional time added for faults.
- Ball game robot design ideas.

Checkpoint

- Were teams able to decide on the order in that team members would drive in during Task 1 and Task 4?
- Were teams able to use their Engineering Notebooks to find algorithms from past activities from which to draw inspiration?
- Were teams able to demonstrate reasonable control over their robot during the relay race?