FIRST® Robotics Engineering Explorations

Student 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?

What Will I Be Doing?

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

Getting Started

- You are the luckiest person ever! You got a brand-new 65" HDTV for your birthday and the newest, best, and fastest video game console on the market, complete with top-of-the-line controllers. There's only one small problem. Your friends are coming over to play your favorite game on your new console, and nothing is ready. What do you do? Design a plan to get your TV and game console working together and save yourself from total embarrassment!
  - Draw a diagram that shows how all the pieces connect.
  - Identify setup options you might need to do.
  - What type of software does the console run on? How is it managed?
  - Why is it important that all the pieces communicate with one another?

WHAT'S NEXT?

- Gather your team.
- Grab your supplies (Engineering Notebook, robot, computer, Driver Station, gamepad, kit of parts).
- Get started!

HOW WILL I DO IT?

- You will install the software tools needed to program and run your robot.
- You will design an electrical diagram that will help you communicate and troubleshoot your drive system.

Task 1: Installing the Programming Tools (IDE)

BRAINSTORM AND EXPLORE:

- How can you control the robot you have built and wired?
- Think about the TV and game console you worked on at the start of this activity. Which part acts like the brain and runs the software?

DESIGN AND PROTOTYPE:

- Before going any further, download the programming tools from your teacher's links.
- Follow the steps your teacher tells you to complete.

Task 2: Pairing the Driver Station with the Gamepad

DESIGN AND PROTOTYPE:

- Now that you have the programming tools, you can utilize the information to connect your robot to the programming device.
- Think about how you can pair them:
  - What allows the gamepad to communicate with your robot?
  - Does your gamepad communicate with your robot through wires or is it wireless?
  - Which technology would you use to establish the connection between your robot and the programming device?

TEST AND IMPROVE:

- Try to design a strategy that will allow you to connect and pair your gamepad with your programming device and Driver Station. Follow the steps on the links provided by your teacher if you get stuck.
- Keep these questions in mind while you are pairing:
  - How can you check that the Driver Station is connected to the gamepad's Wi-Fi?
  - What applications are the two devices using to connect? How is the connection verified in the Driver Station?

Task 3: Configuring the Motors

DESIGN AND PROTOTYPE:

- You have already wired your motors to your gamepad. Just like your gaming console needs to know where the gamepads are plugged into, the gamepad needs to know where to send an electrical signal (port, power, and other data). This configuration process is achieved via the hardware setup. In your programming tools, you should specify which motor is plugged into which port and declare a motor name, which becomes a form of data that can be utilized in your programming tools. Your teacher can provide the needed links to set up your hardware and declare your motor names. Refer to the wiring diagram you created in previous lessons and ensure it matches your configuration.

TEST AND IMPROVE:

- Work with your teammates to create a setup that tells your robot which device is connected and in which port. Follow the steps provided by your teacher. Be sure to save your hardware setup, give it a name, and label your wiring diagram with the file name so it can be referred to for troubleshooting.
- Make sure to create a plan that considers these things:
  - How can you keep a record of this hardware setup?
  - Where will you access this information if you need it later?

Task 4: Ball Game Communication

BRAINSTORM AND EXPLORE:

- You are designing to compete in the ball game that you designed. Your robot needs to have effective communication from your program to the motors to score points. Now that you have paired and configured your robot, think about how you can apply what you just learned to your robot design.

DESIGN AND PROTOTYPE:

- How would you control the parts of this robot and communicate with this robot?
- What actions or steps will your robot need to do to score points?
- What priority might you give to each action to score the most points? (For instance, do you choose one thing you can do well many times, or do you go for a single shot worth the most points?)
- Consider taking measurements from your game so you can record the distances the robot might need to travel or the height it might need to reach.
- How might the measurements affect the data you send to the motors, such as the number of rotations or electrical power needed?

Reflection

- Did you get wireless connectivity and hardware set up correctly for your competition robot?
- Was there anything that could have been done differently to simplify the process?
- What hardware actions will the competition robot need to perform to score points in the ball game?
- What information do you need to document for future reference?

Checkpoint

In your Engineering Notebook:

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

Cleanup Tips

- Ensure you have a place for your laptop.
- Plug in your Driver Station and robot battery and label them for your team.

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?

What Will I Be Doing?

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

Getting Started

- You experience some programming every day. You are programming the box when you set your TV to record your favorite show. You are programming the phone when you block a number or add a number to your favorites. In all these cases, you carry out a list of specific tasks on an electronic device (by pressing specific buttons) so that it starts behaving the way you want it to.
  - What other everyday experiences involve programming?
  - List as many everyday programming experiences as you can in your Engineering Notebook.

WHAT'S NEXT?

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

HOW WILL I DO IT?

Programming a robot to perform a particular behavior involves the following:

- Write a program: You will list the tasks you want the robot to do in the specific order you want them done. These can become robot actions you can program in your programming tool.
- Save programs: You will have multiple programs saved to your robot's memory, enabling you to choose different programs for robot actions you hope to achieve in your game.
- Initialize the program: You will load a program on your Driver Station and call certain variables, such as hardware data, before running the program. This is called initializing.
- Run the program: You will use a dashboard with a gamepad or application that enables the robot from a remote location.

Tip

Make sure your robot program matches your robot hardware setup.

Task 1: Testing Your First Robot Program

DESIGN AND PROTOTYPE:

- Start a program on your computer: Follow the steps in the links from your teacher.
- Create a program that will enable you to drive your robot using the gamepad for your system.
- The first step will allow you to test your hardware setup by moving the motors using the gamepad.
- Sometimes, you will end up with a motor going in the wrong direction because the data you send differs from what you thought.
- Conduct a test with your robot wheels not touching the ground.
- Ensure the communication path is working the way you want it to. For example, when you push forward, the designated motors move forward.
- You may often have to invert motors or the programming power so that it completes the communication pathway as intended.

Check It Out

In a way, your robot is a lot like the TV box and phone you thought about earlier. It will do exactly what you program it (tell it) to do. The tasks you can program your robot to do involve sending data from the program to the robot hardware you configured in the last activity. Your robot can do many tasks. The combination of these task is what will help you do well in your ball game challenge.

Task 2: Driving the Robot

DESIGN AND PROTOTYPE:

- Make sure your robot and dashboard/Driver Station are configured wirelessly. Initialize the program you just saved. If you get stuck, check out your programming tools. Now comes the fun part. It's time to drive your robot!
- As you drive, think about these things:
  - Are you happy with how it performs?
  - Does it go in a straight line?
  - Is it easy to maneuver?
  - What can you improve?
  - How can you improve it?

TEST AND IMPROVE:

- Use your Engineering Notebook to track your progress and brainstorm ways to improve your robot's movement.
  - How can you improve the way your robot moves?
  - Is there any driver error?
  - Where is your robot having trouble moving?
  - Does your robot design need to change?

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

- Ensure you have recorded your program name, observations, improvements, and test results in your Engineering Notebook.

Task 3: Drag Race On!

DESIGN AND PROTOTYPE:

- Now, it's time to test your skills. Have you ever wanted to race your friends? Now is your chance!
  - Work with other teams to set up a drag race strip. Your teacher will help coordinate building the strip and organizing the race.
  - Your teacher will give you the rules of the game and tips to help you evaluate your robot's performance.

TEST AND IMPROVE:

- For each round of the drag race, you compete in, record the following:
  - 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?

Reflection

- Think about your ball game robot; what criteria will your robot need to meet to enable the drive system to score in the game efficiently?
- Given your testing data, what improvements might you make to improve your robot performance for the ball game?

Checkpoint

In your Engineering Notebook:

- Save your program steps and iterations.
- Testing data and answers to reflection questions.

Cleanup Tip

Make sure your robot program Keep your Engineering Notebook organized! This will help you find important information, design notes, and ideas later.

Activity 3: Troubleshooting Is Everywhere

Driving Questions

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

What Will I Be Doing?

- I will investigate a program that isn't doing what it should. Then, I will fix (troubleshoot) it.
- I will use my Engineering Notebook to record what the precise problem is.
- I will document the steps I take to fix the problems I have identified.

Getting Started

Troubleshooting a system involves:

- Knowing how the system is supposed to operate when it is working correctly.
- Discovering the source of the problem.
- Creating a solution and testing it to see if it fixes the problem.
- Keep troubleshooting until all the problems you have found have been fixed!
  - Troubleshooting is everywhere! For example, what if your phone battery suddenly was taking forever to charge? Before you send it for an expensive repair, you might try using a new charger. It makes sense if the battery charges quickly, assuming the problem is in the original charger. You have gone through the troubleshooting process by identifying a problem (the long time it took to charge) and creating a solution (trying a new charger).

Discuss with your teammates:

- A situation you experienced that required troubleshooting.
- How did you discover the source of the problem?
- Ways you tried to fix the problem.
- Were you successful?

WHAT'S NEXT?

- Gather your team.
- Grab your supplies (Engineering Notebook, reference guides, robot, Driver Station, laptop, and gamepad).
- In the last activity, you may have encountered problems where the power was too great or running in the wrong direction.
- In this activity, you will modify the current direction of a motor, so the gamepad doesn't communicate correctly with the robot.
- You will then trade with an alliance partner in your class.
- Take five minutes to try to find out how they changed their robot, so it doesn't operate correctly.
- Then, between the alliances, discuss the process of troubleshooting. If you cannot fix the problem, consult with them to see what they changed and how to fix it. You will also do this with them and share what you changed and how you solved the problem.
- Save your program without making any changes to it. Record the name in your Engineering Notebook.

HOW WILL I DO IT?

- You need to decide whether your drive program is working correctly before you do any troubleshooting.
- The drive program you created in this activity is supposed to help you use the gamepad to drive your robot.
- You will test this drive program for problems. Then, you will investigate to find the source of the problem.
- You will design a fix and test it. You will also document the problem, its causes, and your solution in your Engineering Notebook.

Career Connection

Troubleshooting skills for technicians, engineers, and programmers are as important as investigative skills are for a detective.

Task 1: Test and Document

TEST AND IMPROVE:

- Test your program.
  - Are you able to drive the robot? If not, what is happening instead? What should your robot be doing?
  - Document what is happening in your Engineering Notebook. Ensure you write down any warning or error prompts in your program.
- Is your wireless connection working properly?

Task 2: Investigate

BRAINSTORM AND EXPLORE:

- What could have gone wrong?
  - What errors are possible?
  - What should your robot be doing?
  - What is actually happening?
- Look for the simplest possibilities first. Ask yourself:
  - What do I want to happen?
  - What is happening? Do you have any response? If so, what data (power) do you think the motor receives?
  - What could cause the difference in communication/data (power, hardware setup, or signal direction (+ or -))?
  - Is the gamepad connected properly?
  - Am I using the correct hardware setup?
  - Did I initialize my program?
- If everything seems normal, you should then dig deeper. You've tested a working program before.
  - What is different about the new program?
  - What could be causing it to fail?

Task 3: Solve and Document

DESIGN AND PROTOTYPE:

- By now, you know the problem and its possible source. With your teammates, discuss a possible solution.
- Have you identified the problem and its possible source?
- Create some solutions to try.
- Keep your ideas in your Engineering Notebook.

TEST AND IMPROVE:

- Choose one of your team's solutions and test the program again.
  - Is it working?
  - What you learn is more important than what you win.

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

- What might be causing the problem?
- Where or when is the problem happening?
- What fixed the problem?
- Which technical skills are essential when you are troubleshooting?

Reflection

- Think about your troubleshooting experience in this activity.
  - Why is troubleshooting important?
- Think about your robot.
- How important is it to have a technician who can troubleshoot it?
  - What other ways can you provide troubleshooting support for your design?
  - In your Engineering Notebook, write down why a robot technician needs to be a troubleshooting whiz!

Checkpoint

In your Engineering Notebook:

- Document the problem, its cause, and solutions for the program. Make sure you indicate whether you succeeded in troubleshooting.
- Document the technical and personality skills needed to troubleshoot successfully.
- Record essential troubleshooting strategies you learned.

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?

What Will I Be Doing?

- I will create a set of instructions directing a person to perform a task.
- I will work with my team to create a set of instructions that directs a teammate to move from one side of the room to the other and then perform a task.
- I will consider the different jobs my robot could perform and write instructions that would help it get the job done.

Getting Started

- Robots, computers, and even our phones need instructions to do all the amazing things they do. When you want to make a phone call, the phone has to show you a home screen with an icon. After you press the icon to make a call, the phone must display all the numbers you need to dial. If you want to call one of your contacts, the phone has to display all the names saved in your phone with all your contacts' phone numbers. When you pick a contact, the phone has to show you yet another screen and make the call. No matter how smart our phones become, they still need instructions to do what you want them to. Just like making a call on a cell phone, when you program a robot to move across a room, there are many steps the robot has to take to make the trip.

Think about these things as you complete this activity:

- How many steps does it take to use a phone to make a call?
- How can you break up the daily things such as brushing your teeth, getting to school, or getting a book out of a backpack into step-by-step instructions?

WHAT'S NEXT?

- Gather your supplies (Engineering Notebook, Self-Reflection Rubric, pen, or pencil).
- Gather your team.

HOW WILL I DO IT?

- If a person was a robot and couldn't understand instructions, how would you be able to get them to do anything?
- Create a series of directions to guide a person to complete the task you've been given. Creating directions is just like a programmer creating a program that allows your phone to make a call.
- Decide how many steps you need to give your teacher or another classmate enough information to complete their task.
  - Determine how much detail your instructions need so the person understands exactly what you want them to do.
  - Even programmers make mistakes. If your instructions don't work the first time, decide which of your steps works and which does not.

Task 1: Plotting a Course

DESIGN AND PROTOTYPE:

- Write out your instructions one step at a time. Remember, no detail is too small!
- Run through your instructions with your group.
- Ensure your instructions are detailed enough for your teacher or a fellow team member to follow. Remember, you can't add anything or help them if something is unclear. They might not be able to complete their task if your instructions are hard to understand.

TEST AND IMPROVE:

- Now, it's time to give the person instructions. When the person is ready, read your instructions from your Engineering Notebook one step at a time. Pay attention to the instructions other teams are giving. Other teams may have some ideas your team could learn from!

- As the person is completing the directions, think about the following:
  - Are you happy with your instructions?
  - Is your teacher or classmate moving quickly or slowly?
  - Are your instructions too vague? Too complicated?
  - What can you improve?
  - How can you improve it?

Tip

Make sure you write down each of the steps in your instructions. Think about ways you can break complex instructions into smaller, easier to understand bits of information.

Task 2: Navigation

DESIGN AND PROTOTYPE:

- Now, it's time to take what you've learned and apply it to something a bit more complicated. Your team must come up with instructions for one of your team members. Across the room, there is an object that is out of place. It is your job to write instructions to get that object back to where it belongs.
- Investigate the path with your team. Consider what your teammate should do to get from the starting point to the object.
- The route to the object will be more complex than the task you did earlier. What sorts of obstacles stand between the starting point and the object?

TEST AND IMPROVE:

- When you are confident in your instructions and your teammate is ready at the starting point, test out the directions your team has created.
- If your first set of instructions doesn't work, remember that mistakes in programming happen. Review your instructions in your Engineering Notebook and decide what worked and what might need to be fixed and then try again.

Task 3: Robot Algorithms

DESIGN AND PROTOTYPE:

- Think of your robot design.
- Think of some examples of tasks your robot could perform in the ball game and pick the ones your team thinks are most important.
- After your team has decided on a task for the robot, think of instructions your robot might need. 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?
- Conduct research using your programming tools on how to write the algorithms in your programming environment.
- Record all this information in your Engineering Notebook.

TEST AND IMPROVE:

- Before testing the algorithm on the robot, it can be helpful to have members of your team act as robots performing the task.
- Have a team member test out your algorithm acting as a robot. Consider the following questions:
  - Did the robot/human operate at the desired speed?
  - Did your teammate know how far to go and when to change direction?
  - Were they able to make corrections if they went off-path?
  - Did they know when they were finished?

Reflection

- When creating instructions for your teammates, what were the most common/repeated types of instruction?
- Is there a way to simplify the parts of your instructions that were repeated often?
- Think about your robot. You may need the robot to deal with obstacles your team could not plan for in advance.
  - What obstacles might your robot run into?
  - What are some ways you could avoid obstacles? Think about your robot.
- Could your team provide instructions to your teacher/teammate that solved problems quickly?
  - Is there a way to change your instructions to speed up the process?

Checkpoint

In your Engineering Notebook:

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

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?

What Will I Be Doing?

- I 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.
- I will use motor code to create a program so our robot can complete a challenge.
- I will use my Engineering Notebook to document our findings and other important information.

Getting Started

- When you think of motors, what comes to mind? Some motors, such as cars or airplane propellers, are designed to drive things forward. Some motors are designed to help lift things, such as construction cranes or the winch on the back of a tow truck. Most machines use motors to make things move.

- What kinds of vehicles use motors to move forward and backward?
- What sorts of machines or electric appliances use motors for something other than physically moving from one place to another?
- A record player uses a motor to spin records around and play music. What small machines can you think of that may also use a motor?

Discuss with your teammates:

- When you used a gamepad to drive your robot, what do you think the motors were being programmed to do?
- What jobs could you perform with your robot if you had better control of your motors?
- Without a motor, your robot isn't going to do anything! What else can you do with a motor?

Career Connection

Designing the right motor for the right job is very important. For example, electronic engineering technicians use high-precision motors to control the spin of the hard drive inside your computer. That is partly why you can save and run computer games and other software.

On the other hand, mechanical engineering technicians use low-precision but powerful engines to generate high towing power. Clean energy engineers are always researching ways to create affordable motors that are environmentally friendly.

WHAT'S NEXT?

- Gather your team.
- Grab your supplies (Engineering Notebook, reference guides, robot, Driver Station, laptop, and gamepad).
- Access the gamepad and start a new OpMode.

Tip

Remember to record the name of your program and hardware setup in your Engineering Notebook.

HOW WILL I DO IT?

- In this activity, you are going to modify one of the programs you have already used.

You are going to:

- Change some block code or text code.
- See how the changes affect the way your robot operates.
- Record the changes in your Engineering Notebook.
- Understand how to program robot motors to be able to do certain tasks.

Task 1: Experiment and Learn about Motors

TEST AND IMPROVE:

- Create a copy of your last program that your team created in Activity 2.
- Name the copy "your team's name motor test 1."
- Change the direction of the rotation.
- Save the program and run it. What does your robot do when you try to drive it forward using your gamepad?
- Remove the segment of code you changed.
- Add another code segment that changes the direction of both motors.
- Set the motor directions so that your program will work.

Tip

Remember, when you add or change blocks in your OpMode, it might make your robot behave in ways you don't want it to, or it may not move at all!

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

- Document what you learned, problems you ran into, and how you solved them.
- Where do you find the code in your program that sets the power of your motors?
- Why do you think this code was placed there? In other words, what role did it play in helping you drive the robot around using the gamepad?

DESIGN AND PROTOTYPE:

- Now you know how setting the direction of the motor rotation affects your robot. It's time to see how moving the gamepad powers the motors forward and in reverse.
  - Can you locate the code segment in your program that sets the motor's power?
  - How is the code structure designed to send power to the motor from the gamepad?
  - How can you automate the control of the motor to eliminate human error in control?

Tip

- Open the program you just created.
- Look in the program and find the segment of code that set the direction of drivetrain motors.
- What do you think the initialization and declaring of variables or parameters do?
- Which motor do you think each loop controls?

TEST AND IMPROVE:

- Experiment using different numbers in the power or effort value for the motors.
  - How does the number impact motor movement?
  - Consider the data: is it a decimal or whole number?
  - How is the motor receiving that data?
  - How could the battery power affect the output of that data?

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

- Record what happens when you alter the code in the Test and Improve portion.
- Record how the number attached to a power code changes your robot's movement.
- Record any troubleshooting techniques used in getting the code to work.

Task 2: Investigate: Braking or Rolling?

- Imagine you are navigating a narrow, winding path with your robot. It must immediately stop when you let go of the drive gamepad. You can't risk having it roll to a stop. How could you program that? Think about that as you dive into Task 2!

BRAINSTORM AND EXPLORE:

- What happens if you move your robot forward using the gamepad and then stop pushing the gamepad sticks? Does the robot stop right away or keep rolling? Research a way within your programming tools to create a brake behavior for your motors. If you don't find it in the programming tools, search online. *FIRST*® has a wonderful online community of other teams with tutorials to help you solve problems such as these.

TEST AND IMPROVE:

- Can you use a program to make your robot roll or stop when the gamepad sticks are not pushed forward or backward?
- What options do you have when telling the robot what to do when zero power flows to your motors?

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

- Record which algorithm controls the behavior of your motor when it is zero-powered.
- Describe how you can use this block or text code to make your robot roll to a stop when the robot operator stops pushing the gamepad stick.
- Describe how you can use this block or text code to make your robot brake to a stop when the robot operator stops pushing the gamepad stick.

Task 3: Robots! On My Command!

DESIGN AND PROTOTYPE:

- In many different robot tasks, you will want to be able to identify the algorithm that specifies time. Each programming language and IDE can handle this a little differently. Research your programming tools.

With your team, discuss:

- When might you need a timed block?
- How will moving a time algorithm affect how your robot completes a task?

TEST AND IMPROVE:

- Try changing where the algorithm is in your program and save your project. Use your Engineering Notebook to document what happens.
- Does it matter where the wait time algorithm is in your program?
  - Do you need a time algorithm at all? Why?
  - If changing where the time algorithm stops your program from working the way you want it to, use your troubleshooting skills.

Task 4: Design, Implement, and Compete!

IDENTIFY THE PROBLEM:

- Now that your team is familiar with the different algorithms to operate motors, it is time to put what you've learned into practice! Your teacher has set up a challenge area. A starting point is on one side of the room, and a midpoint is between the starting position and a wall.
  - Create a program to move your robot from the starting point to the midpoint.
  - The team that stops their robot closest to the midpoint wins!

Tip

When you use troubleshooting to fix something, make sure you record the problems and solutions in your Engineering Notebook.

DESIGN AND PROTOTYPE:

- Create a strategy, plan, and design.
- Think about these questions as you work:
  - How will you update your program to help your team win the challenge?
  - Would a fast robot help or a slow one?
  - What should the zero-power behavior of your motors be?

TEST AND IMPROVE:

- When it is your team's turn to complete the challenge, ensure you take notes about your robot's performance in your Engineering Notebook.
- Be sure to take a video of your robot as it performs if you can! A video is a great way to review later as you work to build more complicated algorithms.
  - Did your algorithm work the way you intended it to?
  - What might you need to change?

Tip

As other teams take their turns completing the challenge, pay close attention to their strategies. There are a lot of ways to design an OpMode to complete the challenge.

Reflection

- Which of the algorithms your team created in this activity did you find to be the most useful?
- Which algorithm steps were the most difficult to work with?
- Think of some additional ways your ball game robot could use motors.
  - What different tasks might require your robot to move in various ways?
  - Could your robot use motors for more than just moving around a room?
- Record your thoughts in your Engineering Notebook.

Checkpoint

In your Engineering Notebook:

- Document a summary of each of your team's algorithms in the activity.
- Record essential troubleshooting strategies you used if you encountered a problem putting algorithms into your program.
- Record how your robot moved when you added the algorithms.
- Record the program name your team used to complete the challenge.
- Record your thoughts on different types of motors and their uses in your ball game robot design.

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?

What Will I Be Doing?

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

Getting Started

- You and your team have been invited to watch a new robotic rover land on Mars! From your front-row seats in Mission Control, you watch as a lander drops to the planet's surface and releases the rover onto the alien world.

- On a big screen at the front of the room, you see the rover start to drive to a nearby crater when, all of a sudden, it stops moving. You can see technicians speaking anxiously to each other, and you realize the rover is stuck! A scientist approaches you and asks if your team could help them get the robot moving again.

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

- Think about what obstacles the rover may have encountered that could have stopped it from moving.
- What kind of information could the rover send back to Earth that would help your team find a solution for getting it moving again?

WHAT'S NEXT?

- Gather your team.
- Grab your supplies (Engineering Notebook, robot, Driver Station, computer, gamepad, programming tools.
- In the first task, your team will discuss the getting started scenario and find a way to free the rover.
- After the first task, your team will experiment with data output that can be sent to the Driver Station.

Key Vocabulary

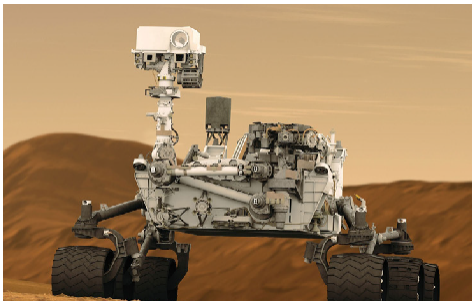
Machines that perform important jobs often need a way of providing information to the people using them. When a machine gathers data and sends or displays it, the information is called **telemetry**.

HOW WILL I DO IT?

- Telemetry can be simple, such as the speed of a car displayed on a speedometer, or sophisticated, such as the GPS location of a plane flying across a country displayed on a detailed map. In complex machines, there is more information to gather and display. In this activity, your team will use data output to save the Mars rover, discover how to get your robot to send telemetry by modifying your program, and experiment with the different types of telemetry your robot can send.

Career Connection

There are many types of engineers, technicians, and scientists needed to make space travel and exploration possible. Aerospace technicians work to design and build the shuttles that get people into space. Mechanical engineers and robotics technicians design, create, operate, and maintain the amazing machines that explore our solar system and beyond.



Task 1: Save the Rover

BRAINSTORM AND EXPLORE:

- Now that you have discussed some different forms of telemetry, talk to your team about the information the rover might be sending back to Earth. Remember that there are many kinds of data, and a Mars rover is a complex machine. The rover has all kinds of sensors and mechanisms to gather information and send it back to Mission Control.
  - What forms of data could the rover be sending?
  - Imagine some of the data Mission Control would have available. Could you use the telemetry the rover is sending to determine why it is not moving?

TEST AND IMPROVE:

- Using the data you thought about in Getting Started, write step-by-step instructions that would help get the rover moving again.
  - Refer to instructions from your teacher on the programming resources you will be using.
- Record your thoughts and step-by-step instructions in your Engineering Notebook.

Task 2: With Great Power Comes Great Telemetry

DESIGN AND PROTOTYPE:

- Use the resources you have in your programming tools to design and conduct two tests for your robot. For instance, create a test that allows you to analyze the power of two different motors.
- Create a test that will give you valuable feedback about the data you are collecting. For example, does the data differ between the motors?

Tip

Remember past activities where your team has had to think of obstacles you needed to program around.

TEST AND IMPROVE:

- Save your program and use your gamepad to test your team's telemetry and data feedback. Remember to use the troubleshooting process if your program isn't working as you want it to.

Keep these questions in mind while you are pairing:

- Where do I need to put the telemetry code to ensure the data I want is being sent?
- Where is the telemetry being displayed?
- How detailed is the telemetry your robot is sending?
- What can your team learn about your robot from the telemetry you've gathered?
- How can you use that information?

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

- Where is the telemetry displayed?
- What format is the telemetry displayed in?
- What did you learn about your robot based on the telemetry it is sending?

Task 3: Power On and Power Off Telemetry

Design and Prototype:

- In the last activity, you learned about different programs for your motors and what they should do if you stop sending power to them.
  - Like the power level, your robot can send telemetry to display what the motors will do when there is no power flowing.
  - You will need to use a different code to send the stop mode than the one you used to send the power level because the data you send will not be displayed in numbers. Set a stop mode for each of your motors and use telemetry to the Driver Station to display what behavior each motor is set to.

TEST AND IMPROVE:

- Save your program.
- Add code to your program using telemetry that determines the state or condition of a motor position. If your telemetry data isn't working how you want it to, use your troubleshooting skills to discover why and record the issue and your solutions.
- Record the program and title when your program displays power or position and stop mode data correctly on your Driver Station.
- Compare your programs from Task 2 and Task 3.
  - What do you notice?
  - Are there key similarities?
  - Are there key differences?

Tip

When you are experimenting with the different telemetry blocks, they may not behave the way you want them to. Experimenting isn't about discovering the right way to do things. Discovering what processes don't work can be just as important as figuring out which ones do.

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

- Make sure you have the following before moving on:
  - Issues you had to troubleshoot and your solutions.
  - The name of the program and title that correctly displayed the power and stop mode data on your Driver Station.
  - Comparison of the telemetry data from Task 2 and Task 3.

Task 4: The More Telemetry, the Better

BRAINSTORM AND EXPLORE:

- Think about your robot. Your ball game may require the robot to be a very complex machine. Based on what you've learned in this activity, think about the different sorts of telemetry your robot might need to score the most points.
- Think about your 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?
- Record your ideas in your Engineering Notebook.
- Your team has altered your program to send two different types of telemetry, but many options are available in your programming tools. Experiment with the different telemetry options and see what other forms of telemetry you can get your robot to send!

TEST AND IMPROVE:

- Try to edit the program you have created to send as much information as possible.
- Explore the different options available in the programming tools. Test any tools you think will help you score more points in the ball game.

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

- Document the different ways you have used telemetry.
- Document the telemetry information your team found the most helpful.
- Remember innovation! Use creativity and persistence to solve problems.

Reflection

- What types of data do you use in your personal life?
  - How do you use it?
  - Which types of data do you use the most often?
  - Are there any types of data that you use that would be difficult or impossible to live without?
- Did you find a way to get the Mars rover working again? Could you improve the instructions you wrote to free it?
- Could you use telemetry in our program to display your motor's power levels and stop mode in the Driver Station?
- Where in your program did you put the telemetry code? Could you have put it somewhere else in the code?
- What information do you need to document for future reference?

Checkpoint

In your Engineering Notebook:

- Document your thoughts on the questions from the Getting Started section.
- Document the step-by-step instructions you created to free the Mars rover in Task 1.
- Summarize your definition of telemetry.
- Document the name of your telemetry program and the code used.
- Document how telemetry could improve the way your ball game robot is designed.
- Document an example of how telemetry could help increase the scoring in the ball game.
- Save all the guides you used in this activity to reference in future activities.

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?

What Will I Be Doing?

- I will create and then complete a task to show I can control what my robot does.
- I will develop a set of step-by-step instructions for completing the task I created.
- I will utilize sample algorithms and integrate them into my program structure.
- I will take all of the algorithms I've learned about and use them to build my program.
- I will think about all the parts of the task my robot will need to know about:
  - I will test my instructions using a teammate first. I will then create a list of changes that could be made based on how my teammate did.
  - I will test, repeat, and improve my program until it can make my robot do its job as accurately as possible.
  - I will use my Engineering Notebook to record my process and reflect at the end of the activity.

Getting Started

- You live in a world where robots do all sorts of tasks. People have used robots to manufacture cars and work on assembly lines for decades. Robots have been used to explore the ocean floor and planets millions of miles away. Today, people have robots to vacuum their floors and mow their lawns. Soon, self-driving cars and robots will deliver packages and food right to your home.

- Your class has been invited to a technology conference to show your robotics skills. Your team has been given your booth, where guests at the conference will be coming to see what you can do.
- As your team sets up, you hear a rumor that one of the big companies at the convention is looking to invest one million dollars in the team with the best presentation! To make an impression on the guests at the tech conference, your team needs to find a way to make your robot do something that stands out and shows you can make your robot do what you want it to.

Talk with your team:

- What are some real-world examples of jobs that robots are used for today?
- How could your robot score the most points in the robot game?
- What tasks could your robot perform to show your team the robot will do exactly what you want it to?
- What sort of data can your robot send? How many types of data can you get your robot to send?
- What obstacles would a robot face if driving without using the gamepad?
- What parts of the gamepad do you use to control your robot? Why do you use those parts instead of some of the others?
- Can you edit your program to use more of the gamepad?
- Can your robot finish a task without the use of a gamepad?

WHAT'S NEXT?

- Gather your team.
- Grab your supplies (Engineering Notebook, reference guides, robot, Driver Station, gamepad, and laptop).
- With your team, brainstorm and discuss different jobs your robot can do:
  - What have you gotten your robot to do?
  - What sort of tasks could your robot do that would show you can make your robot do what you want?
  - What do you think your team could do that other teams couldn't?

HOW WILL I DO IT?

- Every team will create a task that best shows their ability to control their robot. The task should be a task that will help you score points in the ball game. After you have decided on what your robot is going to do:
  - Make a set of instructions that tell your robot how to do it.
  - Try them out by having a team member follow the instructions.
  - Edit your instructions to make sure they are as straightforward as possible.
  - Make a copy of the previous program and start editing it.

Task 1: What Am I Doing?

BRAINSTORM AND EXPLORE:

- Your team first needs a task for your robot to perform. At the end of this activity, your team should be able to prove that you can make your robot do what you want, so picking the right task is essential. Picking a simple task for your robot to complete is also challenging enough to show off your coding skills can be tricky, so take some time to discuss some options with your team.
  - In the programs you've made in past activities, what tasks were you able to accomplish?
  - What are some examples of real-world robots performing tasks in workplaces or homes? What do those robots need to do to complete those tasks?
  - Which task do you want your robot to do in the ball game? How many points will it score?
  - Could the data our robot can send help you complete a job?

DESIGN AND PROTOTYPE:

- After your team has decided on a task and recorded it in your Engineering Notebook, it's time to turn it into a language your robot can understand. To complete the task, your robot must work through all the steps involved. While you are writing the steps, keep the following in mind:
  - How is your robot going to move?
    - What directions?
    - How fast?
    - Where will it be starting from?
    - Where does it need to stop?
    - How many times will it need to stop?
  - If your task involves delivering or moving an object, where will the object be on the robot as it moves?
- How will your robot avoid obstacles?
- Marking essential locations in the test area using some tape might be helpful. Knowing where your robot will start, and any spots you'll be stopping along the way, might be helpful when testing your robot later on. Using a piece of tape to mark where the front wheels of your robot will be before it begins its task will help you ensure you are always starting from the same place.

Tips

- Use what you've learned in past activities and make changes or improvements to complete your goal for today. When you're ready, save your program and test it.
- The different versions of the programs you make while you are testing and improving are called iterations.

TEST AND IMPROVE:

- When you've finished writing your instructions, have a team member test them by acting them out. If your instructions don't work how you want them to, use the troubleshooting process to find ways to improve them. Make sure that the task you've picked, the instructions you've created, and a list of obstacles you'll need to avoid are all recorded in your Engineering Notebook.

Task 2: Turning Instructions into a Program

DESIGN AND PROTOTYPE:

- You have a set of instructions, and now it's time to turn those instructions into something your robot can use.
  - Create a copy of your previous program.
  - Name it based on the task your robot will be completing.
  - Build an algorithm to give your robot all the necessary information to complete your task. Your team's algorithm(s) should show that you can make your robot do whatever you want.
  - Use comment blocks to mark places in your program where you use algorithms you haven't used before.

Remember

You want to show how much control you have over your robot, and the more detail you have in your instructions, the easier it will be.

- Think of the following questions when you are building your program:
  - Which algorithm will give you the most control over how your robot moves forward and backward?
  - Which algorithm determines the speed of your robot?
  - Do you want to control your robot with your gamepad's sticks?
  - Do you want to use the gamepad to control your robot's movement?

TEST AND IMPROVE:

- The instructions your team created could tell your teammate how to complete your task. Use the algorithm you've used in past activities to give your robot all the information it needs to complete its task. Now is also an excellent time to look into the libraries where you get your algorithms to see if there is anything new you can use.
  - What data have you been able to send in past activities?
  - What parts of the gamepad have you used to control your robot?
  - What algorithms have proved to be most effective?
  - Does your task require you to have multiple algorithms to execute the task?

IN YOUR ENGINEERING NOTEBOOK, ANSWER THESE QUESTIONS AND DOCUMENT YOUR RESPONSES.

- Document all the instructions you came up with for Task 2.
- What format is the data and telemetry displayed in?
- Document any algorithms you are trying and then:
  - Write a short description of what you want them to do.
  - Describe why you've decided to use them.
  - Take a screenshot of your algorithm to include in your notes.

Task 3: Test, Iterate, and Improve

TEST AND IMPROVE:

- You've built your first program; well done! Creating a set of algorithms to complete a task is an enormous part of the job for programmers.
  - The next step is testing your program to see what it can do!
  - Your team probably tried some new things when you were writing this program, so remember to use your troubleshooting skills when you are testing it out.
- Start your program and try to complete the tasks you created for your robot.
- Don't worry if your program isn't working as you want! Testing new algorithms to integrate hardware and software components can be tricky.
- After each test, make a copy of your program, add a number to the copy's name to show which iteration version it is, and make changes to improve its performance.

Career Connection

Testing, correcting, and trying again are a big part of creating code. Video game designers usually go through dozens of iterations of their game before they get to a finished product. When a game designer thinks their game is almost ready, they will sometimes release a beta version for people to test. Designers use the feedback from the people that play a beta to fix problems in their game.

Keep these things in mind while you are testing:

- Where are the algorithms you've added to the original program template?
- What parts of your algorithm allow you to initialize hardware?
- Which algorithms are executed when the robot is "started" from the Driver Station?
- Is your robot able to stop accurately?
- If you're moving an item, does it fall off? Can you move the item from the starting point to the stopping point as quickly as possible?
- If you are using new or different inputs on the gamepad, what algorithms are they attached to in your program?
- Does your team need to pick up or adjust your robot while performing its task?
- Share your robot's performance with the class when your teacher tells you. Pay attention to each team's performance.

Tip

Testing and improving will bring you closer the best program for the tasks you've set for your robot. Even if you can complete your task the first time you use your program, there is always room to improve.

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

- Program improvement ideas.
- Speed improvement ideas.
- Algorithm simplification ideas.

Reflection

- Did each of your program iterations improve the performance of your robot? Why? Why not?
- How many iterations did your team go through before you reached a working program?
- What was the end goal of the tasks you created for your robot? Were you able to achieve the end goal?

Checkpoint

In your Engineering Notebook:

- Record your responses to the prompts in the Getting Started section of the activity.
- Record your responses to the Engineering Notebook prompts in Tasks 1-3.
- Record your answers to the reflection questions.

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?

What Will I Be Doing?

- I will use the program from the past activity to test drive the course set up by the teacher.
- Every member of my team will practice *Coopertition*® and *Gracious Professionalism*® as they take a turn driving through the course.
- I will create a program suited for a relay race around a relay course.
- I will test my relay program by driving around the course once.
- I will improve the program through iteration based on how my robot performed during the test lap.
- When our program is ready, every team member will take a turn driving through the obstacle course as quickly and accurately as possible.
- I will reflect on the importance of control when driving my robot.
- I 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.

Getting Started

- It's time for you to take your driving test! You've taken a driving course and have been practicing driving with an older driver in the car for months. If you pass the test, you can finally drive without supervision. To prepare, you have been reviewing a pamphlet from the driving school you attended. The pamphlet lists some tips and common reasons people fail their tests.
- Discuss the following questions with your team and record your thoughts in your Engineering Notebook:
  - What are some common mistakes people might make while taking their driving test?
  - What are some tips a driving pamphlet might suggest to people before they take their test?
  - What are the most important rules to remember when driving a car?

WHAT'S NEXT?

- Gather your team.
- Grab your supplies (Engineering Notebook, robot, Driver Station, laptop, and gamepad).
- Decide on the order team members will race.

Tip

Try to avoid driving outside the course boundaries or colliding with any obstacles. During the final task, exiting the course or colliding with an obstacle will earn your team a fault. Every fault your team earns during the relay race will add five seconds to your final time.

HOW WILL I DO IT?

- Each team member will take turns driving through a course your teacher has set up.
- Before your team begins driving, you will make a rough sketch of the course in your Engineering Notebook.
- When driving through the course for the first time, remember your answers from the Getting Started activity.
- While your team members are driving:
  - Mark any areas where your robot has trouble using the sketch you created.
  - Reflect on any issues your team faced during the test.

- When all your team members have had a chance to take a test drive:
  - Create a new program.
  - Use the notes you took during your test drive to build a program for a relay race around the course.
  - Then, your team will have one lap around the course to test and improve your robot and algorithms.
- When your robot and program are ready, your team members will take turns racing through the course as fast as you can. The team with the best time on the course wins!

Task 1: Test Run

BRAINSTORM AND EXPLORE:

- To better understand what you will be doing in the relay race, you will have a chance to practice driving through the course you will be racing on. In the Getting Started scenario, your team discussed some mistakes people might make during their driving test. In this task, you can see the difficult parts of the track so you can plan ahead and avoid making mistakes while driving in the relay race at the end of the activity.
  - You should choose the program you feel will best help you achieve the task while driving through the course for the first time.
  - Before you start, draw a rough sketch of the course in your Engineering Notebook.

TEST AND IMPROVE:

- Consider involving other team members to take turns driving and record each time your robot collides with an obstacle or drives outside of the course boundaries.

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

- Mark the location of any obstacles in the course on the sketch you made.
- Record any parts of the course where your robot drove outside the boundaries and received a fault on your sketch.
- Discuss why your robot drove off the course or collided with an obstacle and brainstorm possible solutions.
- Record any issues your team had while driving and examples of what you might do differently during the relay race in your Engineering Notebook.

Task 2: A Program Made to Win

IDENTIFY THE PROBLEM:

- Using the sketch your team created during Task 1, discuss what algorithms designed to win a relay race might look like:
  - What are the limitations of your current robot?
  - What sorts of data would be helpful during a relay race? Why?

DESIGN AND PROTOTYPE:

- Create a new program for your team to use during the relay race and name it “your team’s name Relay Team__.” Add any new algorithms that might give you data to help improve the mechanical performance of the robot.

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

- Document your relay program and iterations
- If you used pieces of an algorithm you used in a past activity to create your relay program, explain why.

Task 3: Test, Iterate, and Improve

TEST AND IMPROVE:

- One team member can test the relay program you’ve created, but they only have one lap around the track to do it.
- Before you take your test lap, look at the sketch you made during the first task.
- Go over the problems you faced and the areas of the course you found difficult.
- You only have one lap to test your program, so make it count!
- Before you start the test lap, make another sketch of the track in your Engineering Notebook so you can mark trouble areas.
- As your robot is driving through its test lap, take as many notes on your robot’s performance as you can. Consider recording data from the Driver Station and the robot so the data can be analyzed.
- The more detail you have on the robot’s strengths and problem areas, the easier it will be to make changes before it’s time to race.
- When your team is ready, make a new iteration of your program, fix mechanical issues, and make improvements based on what you saw in your test lap.

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

- Record detailed notes on your robot's performance to indicate if the performance is hardware-related or software-related.
- Document a second sketch of the course you will be racing on.
- On your second sketch, mark the places where the robot had difficulty avoiding an obstacle or staying within the track's boundaries.
- Reflect on where your robot worked well.
- Reflect on areas where your robot had trouble and use the troubleshooting process to try and make improvements.
- Record the program iteration, including any hardware and software changes you've made.

Task 4: Time Trials

TEST AND IMPROVE:

- You have taken turns driving on the track, created a program for the race, and used the Test and Improve step to ensure you have the best possible program. Now it's time to race!
  - Instead of having all teams simultaneously race on the same track, you will compete in time trials.
  - Pair up with another team to create an alliance.
  - You will be timing their performance, and they will keep time for you.
  - When you are ready, ask the timekeeper to start the timer and begin the race.
  - The other team will keep track of your robot and note any time you collide with an obstacle or drive outside the track's boundary.
  - Remember that leaving the track boundaries or colliding with an obstacle is a fault.
- Using the order your team decided on at the beginning of the activity:
  - Take turns racing your robot around the course.
  - Make sure your team is ready in its order so you can pass the gamepad to the next person as soon as you finish your lap.
  - When you have finished driving, multiply the number of faults your team had during the race by five seconds and add the time your team's final time.

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

- Record the total time it took for all team members to finish the course.
- Record the number of faults your team made during your race and the total time the faults added to your team's time.
- Record the name of your alliance partner.

Reflection

- How can you apply what you've learned during this activity to your ball game robot design?
- What about the final version of your program worked well for your team in the final task?
- Were there any parts of your final robot that didn't work how you wanted them to?

Checkpoint

In your Engineering Notebook, make sure you have:

- Record your responses to the prompts in the Getting Started section of the activity.
- Record your responses to the Engineering Notebook prompts in Tasks 1-3.
- Record your answers to the reflection questions.