

Unit 2: Motors

A LEGO® Education Unit

Unit Introduction

This unit allows students to explore essential computer science principles and programming concepts of the text-based coding language, Python, while creating models that move autonomously. Students will investigate how robots are motorized and can help automate actions. The lessons are designed in an order that allows students to progress in their skills and knowledge in the following areas:

- Program single and multiple motors run using variables for time, speed, and degrees
- Program a motor to move to a specific position using the shortest path
- Build and program a model that automates a task
- Use and modify existing code to explore new ideas
- Create pseudocode to support creating algorithms
- Utilize code comment features to documents parts of a program
- Define and decompose a problem

Unit Learning Promise

In this unit, students will explore how to control motors in different ways, how to effectively utilize motors in robots, and how to utilize motors to automate tasks. Students will utilize pseudocode to support creating algorithms and code comments to document their programs.

Investigation Questions:

How do robots move? How can automation make tasks easier? Why is documenting programs important?

Unit Lessons

Lesson 1	Lesson 2	Lesson 3	Lesson 4	Lesson 5	Lesson 6
Making Moves with Motors	New Moves with Motors	Automating Action	Hopper Run	Race Day	Ideas to Help
Time: 45 minutes	Time: 45 minutes	Time: 45 minutes	Time: 45 minutes	Time: 45 minutes	Time: 45 minutes

Assessment: We recommend assessing students on various skills throughout the unit.

- Use the progression of lessons as an opportunity to provide on-going feedback to prepare students for success for the open-ended project at the end of the unit.
- Each lesson includes a recommendation for teacher observations, student self-assessment, evaluation of success.

Unit Standards

CSTA
2-CS-02 Design projects that combine hardware and software components to collect and exchange data.
2-AP-10 Use flowcharts and/or pseudocode to address complex problems as algorithms.
2-AP-13 Decompose problems and subproblems into parts to facilitate the design, implementation, and review of programs.
2-AP-16 Incorporate existing code, media, and libraries into original programs, and give attribution.
2-AP-17 Systematically test and refine programs using a range of test cases.
2-AP-19 Document programs in order to make them easier to follow, test, and debug.
1B-IC-20 Seek diverse perspectives for the purpose of improving computational artifacts.

Integrated Standards

NGSS		
MS-ETS1-4 Develop a model to generate data for iterative testing and modification of a proposed object, tool or process such that an optimal design can be achieved.		
Common Core English Language Arts (ELA)		
6 th Grade	7 th Grade	8 th Grade
SL.6.1 Engage effectively in a range of collaborative discussions (one-on-one, in groups, and teacher-led) with diverse partners on grade 6 topics, texts, and issues,	SL.7.1 Engage effectively in a range of collaborative discussions (one-on-one, in groups, and teacher-led) with diverse partners on grade 7 topics, texts, and issues,	SL.8.1 Engage effectively in a range of collaborative discussions (one-on-one, in groups, and teacher-led) with diverse partners on grade 8 topics, texts, and issues,

building on others' ideas and expressing their own clearly	building on others' ideas and expressing their own clearly	building on others' ideas and expressing their own clearly
SL.6.2 Interpret information presented in diverse media and formats (e.g., visually, quantitatively, orally) and explain how it contributes to a topic, text, or issue under study	SL.7.2 Analyze the main ideas and supporting details presented in diverse media and formats (e.g., visually, quantitatively, orally) and explain how the ideas clarify a topic, text, or issue under study	SL.8.2 Analyze the purpose of information presented in diverse media and formats (e.g., visually, quantitatively, orally) and evaluate the motives (e.g., social, commercial, political) behind its presentation
SL.6.4 Present claims and findings, sequencing ideas logically and using pertinent descriptions, facts, and details to accentuate main ideas or themes; use appropriate eye contact, adequate volume, and clear pronunciation	SL.7.4 Present claims and findings, emphasizing salient points in a focused, coherent manner with pertinent descriptions, facts, details, and examples; use appropriate eye contact, adequate volume, and clear pronunciation	SL.8.4 Present claims and findings, emphasizing salient points in a focused, coherent manner with relevant evidence, sound valid reasoning, and well-chosen details; use appropriate eye contact, adequate volume, and clear pronunciation
RST.6-8.3 Follow precisely a multistep procedure when carrying out experiments, taking measurements, or performing technical tasks	RST.6-8.3 Follow precisely a multistep procedure when carrying out experiments, taking measurements, or performing technical tasks	RST.6-8.3 Follow precisely a multistep procedure when carrying out experiments, taking measurements, or performing technical tasks.
L.6.6 Acquire and use accurately grade-appropriate general academic and domain-specific words and phrases; gather vocabulary knowledge when considering a word or phrase important to comprehension or expression	L.7.6 Acquire and use accurately grade-appropriate general academic and domain-specific words and phrases; gather vocabulary knowledge when considering a word or phrase important to comprehension or expression	L.8.6 Acquire and use accurately grade-appropriate general academic and domain-specific words and phrases; gather vocabulary knowledge when considering a word or phrase important to comprehension or expression

Making Moves with Motors

Grade 6-9

45 minutes

Beginner

Making Moves with Motors

Students will work with individual motors to create a dance party.

Questions to investigate

- How can engineers and programmers work together to make something move?

Materials needed

- SPIKE Prime sets ready for student use. Prior to the first lesson, please visit the following website for help with set up, kit organization and SPIKE App <https://education.lego.com/en-us/start/spike-prime/intro>
- Devices with the SPIKE App installed.
- Student journals

Prepare

Ensure SPIKE Prime hubs are charged, especially if connecting through Bluetooth.

1. Engage

Engage students in thinking about how motors move. Have students stand to act as motors. Have students create a line and ask them to walk forward 5 steps. Have them try this a couple of times. Ask students why everyone does not move the same distance.

- Ask students to raise their right arm when you say “Raise”. Wait a couple of seconds and say “Raise.” Why didn’t everyone move at the exact same time?
- Have them do this a couple of times. Ask students why everyone does not move the same way.

KEY OBJECTIVES

Students will:

- Program motors to turn individually using parameters of time and speed
- Create a robot dance party

STANDARDS

CSTA

2-CS-02 Design projects that combine hardware and software components to collect and exchange data.

2-AP-10 Use flowcharts and/or pseudocode to address complex problems as algorithms

2-AP-16 Incorporate existing code, media, and libraries into original programs, and give attribution.

2-AP-17 Systematically test and refine programs using a range of test cases.

2-AP-19 Document programs in order to make them easier to follow, test, and debug.

VOCABULARY

Pseudocode

Initialize

Motor

Speed

- Ask students to clap when you say “clap.” Wait a couple of seconds and say clap. Have them try this a couple of times. Ask students why everyone does not clap at the same time.
- Discuss how a group of robots programmed with the same code would react given the same command simultaneously.
- Discuss how a robot moves and why movement for a given command would be the same every time. Ask students when this would be useful or helpful.

2. Explore

Students will explore working with motors and ways to code movement.

Direct students to the **START** section in the SPIKE App. Here students can access **2. Motors and Sensors**. This Getting Started lesson provides students with an early experience in building and coding with SPIKE Prime.

Use this lesson to guide students in building the motor modules. Students should complete the building instructions **through step 19 only**. You will **not** need to complete the rest of this lesson. Direct students to leave the Motors and Sensors lesson.

Open a new project in the Python programming canvas. Ask students to erase any code that is already in the programming area. Students should connect their hub.

Plug the large motor into port C of the hub.

Before students can use motors, the motors must be initialized. That means you are naming the motors in a way that clarifies for you and the program which motor(s) you want to use and where they are located in the hub. Without this step, the motor will not work.

For example,

A motor in Port A would be initialized as follows:

```
motor = Motor('A')
```

Run Single Motor for Time

Running a motor for time allows you to set how long the motor will run.

Brainstorm with students how to program the motor to move. Discuss what information the software needs to run the hardware correctly.

Write a pseudocode program for making the motor run for 2 seconds with students. Pseudocode is writing in words what you want the program to do.

An example could be:

- Import motor

- Initialize motor
- Turn motor on
- Move clockwise for time 2 seconds

Note: The code written in steps does not need to match exactly what will be put in the program. Writing pseudocode helps students determine what is needed.

Provide students with the sample code below to move the motor. Have students type the code into the programming canvas. (Students can copy and paste the sample code from the **Knowledge Base Getting Started Part 2: Controlling the Motors** section under **Run Single Motor for Time**.)

```
from spike import Motor

# Initialize the motor
motor = Motor('C')

# Rotate clockwise for 2 seconds at 75% speed
motor.run_for_seconds(2.0, 75)
```

Troubleshooting Tip: Ensure students have the motor plugged into the C port or change the Motor('C') to the correct port.

Discuss what the 2.0 and 75 represent in the last line of code. The code comment provides a hint.

- Prompt students to put different numbers in for these to values to investigate how it changes how the motor moves.
- Allow students to share the new examples they create and discuss ways to program the motor.

Remove the large motor from the hub.

Run Multiple Motors

Prompt students to add the two medium motors to their hub. Having multiple motors attached will allow students to run more than one motor at a time. Ask students to consider what type of robots they could create that would need more than one motor. What function could multiple motors serve?

Brainstorm with students what they need to add to their program to allow for two motors to run. The task is to make one motor run (in port C) and then the second motor run (in port B).

Remind students that they can plug the motors into any port. However, the program needs to match the chosen port, which might not match the sample program.

Ask students to add to the pseudocode used previously. Add wording to make the second motor turn on after the first motor.

Have students use the pseudocode as inspiration to help write their new program that will run both motors.

```
from spike import Motor

# Initialize the motor
Motor_c = Motor('C')
Motor_b = Motor('B')

# Rotate clockwise for 2 seconds at 75% speed
Motor_c.run_for_seconds(2.0, 75)
Motor_b.run_for_seconds(2.0, 75)
```

3. Explain

Discuss with students how the program worked. What different combinations of the variables time and speed did students choose?

Ask students questions like:

- Explain the difference between importing the library for the medium motor and the large motor.
- What is required when initializing more than one motor? How do you differentiate between them?
- What variables for the motor can you change to make it move in different ways?
- How did the two motors move? Did they move at the same time? Why or why not?
- How can you troubleshoot errors that occur when working with motors?

Note: Explain to students that the number specified for seconds is a “float.” A float type parameter can be a decimal number.

Explain to students that the number specified for speed is an integer type input. An integer type input can be a positive or negative whole number ranging from -100% to 100%.

4. Elaborate

Dance Party

Challenge students to create a short program to run the 2 medium motors to create a dance party. Play some music to get kids creative. For example, students could have both motors moving in one direction or have each motor moving in opposite directions.

- Allow students to add additional bricks to their motors to create something new.
- Prompt students to add sounds and hub lights to their program to complete the dance party.
- Remind them to think about which libraries need to be imported to support their programs and where to find help in the **Knowledge Base** as needed.

Students should share their dancing robots and explain their programs.

Disconnect the motors and return all parts to the correct bin location.

5. Evaluate

Teacher Observation:

Discuss the program with students.

Ask students questions like:

- What happened on your hub when you ran the program?
- What libraries did you use to run this program?
- How can engineers and computer programmers work together to create a way to communicate ideas to others?
- How can engineers and programmers work together to make something move?

Self-Assessment:

Have students answer the following in their journals:

- How do you determine which motor you are programming to move if multiple motors are attached to the hub?
- What characteristics of a good teammate did I display today?
- Ask students to rate themselves on a scale of 1-3, on their time management today.
- Ask student to rate themselves on a scale of 1-3, on their materials (parts) management today.

New Moves with Motors

Grade 6-9

45 minutes

Beginner

New Moves with Motors

Students program their motors to move to exact positions and for exact degrees.

Questions to investigate

- How do robots move precisely?

Materials needed

- SPIKE Prime sets
- Devices with the SPIKE App installed.
- Student journals

Prepare

- Ensure SPIKE Prime hubs are charged, especially if connecting through Bluetooth.
- Ensure students have built the Getting Started 2: Motors and Sensors model through step 19 utilized in the Making Moves with Motors Lesson.

1. Engage

Using a tire from the SPIKE Prime set, ask students to think of different ways to measure how far the tire would drive if it went around one rotation. Students should come up with several methods to determine this distance. Prompt student to think about the wheel as a circle. Discuss with them how to calculate the circumference to determine the distance. Ask students how they could measure the circumference of the wheel using a flat surface if they are struggling.

Ask students to imagine they have a robot with wheels. They need to code the motors to move a certain distance. Ask students how could they program the motor to move other than for time? How would this help move the robot more accurately? Discuss ideas with the students about how to move only $\frac{1}{2}$ a turn (0.5 rotation), a full turn (1 rotation), 2 turns (2 rotations).

KEY OBJECTIVES

Students will:

- Program a motor to move to position using the shortest path.
- Program a motor to move to a specific position.
- Program a motor to move a defined number of degrees.

STANDARDS

CSTA

2-CS-02 Design projects that combine hardware and software components to collect and exchange data.

2-AP-10 Use flowcharts and/or pseudocode to address complex problems as algorithms

2-AP-16 Incorporate existing code, media, and libraries into original programs, and give attribution.

2-AP-17 Systematically test and refine programs using a range of test cases.

2-AP-19 Document programs in order to make them easier to follow, test, and debug.

VOCABULARY

Degrees

Position



Note: The wheel has a diameter of 5.6 cm (2.2 in.) and travels a distance of 17.6 cm (6.9 in) per rotation or a complete 360° degree circle.

2. Explore

Students will explore how to program motors to move using degrees.

Run Single Motor for Degrees

Running a motor for degrees allows you to determine how far the motor will turn in degree increments – $1/360^{\text{th}}$ of a rotation. Brainstorm with students how they could program the motor to move.

Students should think about what information the software needs to run the hardware correctly. Write a pseudocode program with students for making the motor run for one complete circle.

An example could be:

- Import motor
- Initialize motor
- Turn motor on
- Move motor 360 degrees

Note: The code written step-by-step does not need to match exactly what will be in the program. Pseudocode is used to help students think through what steps are needed so programming can be coded linearly.

Direct students to open a new project in the Python programming canvas. Ask students to erase any code that is already in the programming area. Students should connect their hub and plug a motor into the C port of their hub.

Provide students with the sample code to move the motor. Ask students to run the program. Students can also copy and paste this code from the **Knowledge Base Getting Started Part 2: Controlling the Motors** section into the programming canvas.

```
from spike import Motor

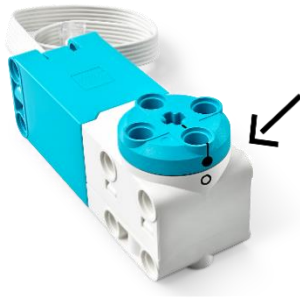
# Initialize the motor
motor = Motor('C')

# Rotate the motor 360 degrees clockwise
motor.run_for_degrees(360)
```

Discuss with students how the motor moved. Help students identify the marker on the motor that allows you to set the position to 0 degrees. This will allow students to measure how far the motor moves.

Ask students to set the mark to the 0 position. Run the program again. Ask students if the marker return to the same position.





Prompt students to put different numbers into the degree values to see how the motor moves. Have them try large and small numbers.

Ensure students specifically try to run the motor using a negative value. Ask them what changed. (The direction reversed.)

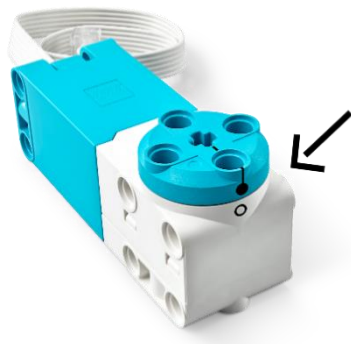
Ask students to enter a number larger than 360. What happens?

```
# Run the motor 360 degrees clockwise, at 30% speed  
motor.run_for_degrees(-360, 30)
```

Discuss what students discover after their investigations.

Run Single Motor to Position

Students will explore how to run a single motor to a position.



Have students locate the position marker again and ensure it is lined up. When aligned, the motor is in the 0-degree position. Discuss with students how to program the motor to move to a specific position (the motor will stop at an exact position). Prompt students to change their pseudocode to use position instead of degrees.

Provide students with the sample code to move the motor. Ask students to type this program into the programming canvas. Students can also copy and paste this code from the **Knowledge Base Getting Started Part 2: Controlling the Motors** section. Ask students to run this program.

```
from spike import Motor

# Initialize the motor
motor = Motor('C')

# Place the motor in position "0," aligning the markers
motor.run_to_position(0, 'shortest path', 75)
```

Discuss the program with students to identify what happens when they run the code. If students start at the 0 degrees position, the motor will not move.

Ask students to move the motor so it is no longer in the 0 degrees position. Run the program. The motor should move to the 0 degrees position. Which direction did it move to get there?

Ask students to run the program several more times with the motor starting in different positions each time. Discuss what happens. Students should identify that the motor is not moving the same each time. The motor moves in the direction that makes the shortest path back to the 0 degrees position.

Review the lines of code together to highlight why the motor is moving in this way.

What is the longest move a motor can make to return to 0 position?

Ask students to change the code to make the motor stop in different positions. You can provide this sample code to them and then let them explore ideas of how they might change the code. Remind students that when exploring there is a high likelihood of a receiving an error message in the bottom console and the program not executing.

```
from spike import PrimeHub, Motor
from spike.control import wait_for_seconds

# Initialize the motor
motor = Motor('C')

# Run the motor to different positions, at different speeds
wait_for_seconds(1)
motor.run_to_position(0, 'shortest path', 30)
wait_for_seconds(1)
motor.run_to_position(90, 'clockwise', 100)
```

3. Explain

Allow students to share the new example programs they created for degrees and position and discuss ways to program the motor. Discuss how the motor can be set to move for degrees and when this might be useful in a program.

- How does this program work?
- When would it be useful to program a motor to move using degrees or to position?
- Why does using degrees or to position allow for more precision in movement than programming for seconds?
- How did the motor move when you put a number in larger than 360?
- How can you calculate how many degrees you will need to program?
- When using the shortest path in your program for position, why does the motor not move if you start at the 0 degree position?
- How can you program the motor to move in the opposite direction?

4. Elaborate

Work together to work on debugging. Provide students with each of the following codes. Discuss what is wrong with each code. Make sure students run each code and review the error message in the console.

1. What is missing?

```
from spike import Motor

# Rotate the motor 360 degrees clockwise
motor.run_for_degrees(360)
```

Students should identify that the motor is not initialized and therefore cannot run for degrees. The error message in the console is

```
File "lib/hub/flash/programrunner/__init__.py", line 1, in start_program
File "./projects/31408.py", line 4
NameError: name 'motor' isn't defined
```

Students need to add `motor=Motor('C')` after the import line. Have students add this in and run the program again to check their program.

2. What number is wrong and why?

```
from spike import Motor

# Initialize the motor
motor = Motor('C')
```

```
# Run motor for 2 seconds at 50% speed
motor.run_for_seconds(2.0, 150)
```

Students should identify that the motor power is set to 150 which is outside the allowable 100% range. The program, however, will still run with no error message in the console. The power will revert back to 100% and run the program.

3. What is wrong?

```
from spike import PrimeHub, Motor
from spike.control import wait_for_seconds

# Initialize the motor
motor = Motor('C')

# Run the motor to different positions, at different speeds
wait_for_seconds(1)
motor.run_to_position(0, 'shortest path', 30)
wait_for_seconds(1)
motor.run_to_position(90, 'clockwise', 100)
```

Students should recognize that the word shortest is misspelled in line 9. The error message indicates that there is an issue in line 9 and that the error is a value that is not allowed.

File "lib/hub/flash/programrunner/__init__.py", line 1, in start_program

File "./projects/23262.py", line 9

File "lib/hub/flash/spike/motor.py", line 1, in run_to_position

ValueError: direction is not one of the allowed values: ('shortest path', 'clockwise', 'counterclockwise').

5. Evaluate

Teacher Observation:

Discuss the program with students.

Ask students questions like:

- What are different ways you can program your motors to move?
- Which methods allow for more precise movements?
- What are ways you can program to move using position?

Self-Assessment:

Have students answer the following in their journals:

- What did you learn today that might be helpful in programming robots to move with precision?
- What characteristics of a good teammate did I display today?



- Ask students to rate themselves on a scale of 1-3, on their time management today.
- Ask student to rate themselves on a scale of 1-3, on their materials (parts) management today.



Automating Action

Grade 6-9

45 minutes

Beginner

Automating Action

Students will add a motor to a model to automate the action.

Questions to investigate

- How can a mechanism be motorized to add automation? How can automation make tasks easier?

Materials needed

- SPIKE Prime sets
- Devices with the SPIKE App installed.
- Student journals

Prepare

Ensure SPIKE Prime hubs are charged, especially if connecting through Bluetooth.

1. Engage

Engage students in a discussion about automation. Prompt students to provide ideas about what automation is and provide examples of everyday items that provide automation. Consider showing images or videos as needed for students to understand how automation happens.

Show students the video from the SPIKE Prime **Extra Resources - Ideas, the LEGO Way lesson**. This lesson is located at <https://education.lego.com/en-us/lessons/prime-extra-resources/ideas-the-lego-way#ignite-a-discussion>. Use this video to prompt a discussion about what could be automated in the model shown. Discuss ideas for adding a motor to automate the movement provided by the two people in the video.

2. Explore

KEY OBJECTIVES

Students will:

- Build and program a model that automates a task.

STANDARDS

CSTA

2-CS-02 Design projects that combine hardware and software components to collect and exchange data.

2-AP-10 Use flowcharts and/or pseudocode to address complex problems as algorithms

2-AP-16 Incorporate existing code, media, and libraries into original programs, and give attribution.

2-AP-17 Systematically test and refine programs using a range of test cases.

2-AP-19 Document programs in order to make them easier to follow, test, and debug.

NGSS

MS-ETS1-4 Develop a model to generate data for iterative testing and modification of a proposed object, tool or process such that an optimal design can be achieved.

VOCABULARY

Automate

Challenge students to add a motor to the tower arm in order to automate the movement to the character rather than needing to move it manually.

Students will design an idea for adding a single motor to the tower arm. Students should sketch an idea for adding the motor then write out their pseudocode for how to program the tower arm to move. Students need to consider the design of the tower arm, understanding how it can move, in order to program the movement to be automated.

Open a new project in the Python programming canvas. Ask students to erase any code that is already in the programming area. Students should connect their hub.

Students will program the movement of the arm. Use the pseudocode to help create the program. Remind students to test their program several times in order to ensure it moves as expected. Ask students to add code comments using `#` to explain the steps of the program.

3. Explain

Students should share their final design and program, explaining how the program automates the movement.

Ask students questions like:

- How did you program your model? Ask students to share their program comments to explain.
- What decisions did you have to make while adding the motor?
- What decisions did you have to make when programming your movement?
- How does adding the motor to automate the movement make the task easier?
- What were areas that you had to debug or troubleshoot?
- What was difficult about this challenge?

4. Elaborate

Have two groups of students come together to form a larger group.

Group 1 will present their model and program to Group 2. Group 2 will then present their model and program to Group 1.

Both groups should focus on the code comments provided in each program. The groups will work together to make sure the code comments clearly outline each program.

Working together, groups should create basic user instructions for their automation. Each group can add these instructions to the top of their program using a code comment.

5. Evaluate

Teacher Observation:

Discuss the program with students.

Ask students questions like:

- How were you able to motorize your mechanism?
- What decisions did you need to make in order to add the motor?
- Why are code comments, including user instructions, important to create?

Self-Assessment:

Have students answer the following in their journals:

- What did you learn today about adding automation to movement?
- When can automation be helpful?
- What characteristics of a good teammate did I display today?
- Ask students to rate themselves on a scale of 1-3, on their time management today.
- Ask student to rate themselves on a scale of 1-3, on their materials (parts) management today.

Hopper Run

Grade 6-9

45 minutes

Beginner

Hopper Run

Students build and program a robot without wheels to go forward.

Questions to investigate

How does the design of a robot determine how it can move?

Materials needed

- SPIKE Prime sets
- Devices with the SPIKE App installed.
- Student journals

Prepare

Ensure SPIKE Prime hubs are charged, especially if connecting through Bluetooth.

1. Engage

Engage students to think about how to move when using more than one motor. Put students in groups of 4. Have them link together either by holding arms or a ruler/stick between them. Ask groups to move together in unison in a line. Discuss what happens.

Challenge students to become synced in the way they move (stepping forward together at the same time, for the same distance). See how students communicate together to make these moves happen. Discuss as a group.

2. Explore

Students will build a hopper model to investigate different ways to move using the motors.

Direct students to the **BUILD** section in the SPIKE App. Here students can access the building instructions for **Hopper**. Ask students to build the model. The building instructions are also available at <https://education.lego.com/en-us/support/spike-prime/building-instructions>.

KEY OBJECTIVES

Students will:

- Program two motors to move simultaneously.
- Build and program a robot without wheels to move forward.

STANDARDS

CSTA

2-CS-02 Design projects that combine hardware and software components to collect and exchange data.

2-AP-10 Use flowcharts and/or pseudocode to address complex problems as algorithms

2-AP-13 Decompose problems and subproblems into parts to facilitate the design, implementation, and review of programs.

2-AP-16 Incorporate existing code, media, and libraries into original programs, and give attribution.

2-AP-17 Systematically test and refine programs using a range of test cases.

2-AP-19 Document programs in order to make them easier to follow, test, and debug.

VOCABULARY

Motor Pair

Direct students to open a new project in the Python programming canvas. Ask students to erase any code that is already in the programming area. Students should connect their hub.

Get Moving

Challenge students to have their hopper run forward for 5 seconds.

Discuss with students how the hopper model will move using the two motors attached to move simultaneously. Ask students to write a pseudocode program to explain how they need to program hopper to move forward. Discuss the pseudocode examples students create.

Example pseudocode:

- Import motors
- Initialize motors
- Start motors
- Move for 5 seconds
- Stop motors

Note: This is a sample of what students might be thinking and does not represent an actual code.

Share this sample program with students. Students will need to type this program into the programming canvas.

```
from spike import PrimeHub, MotorPair
from spike.control import wait_for_seconds

# Initialize the hub and motor
hub = PrimeHub()
hop_motors = MotorPair('E', 'F')

# Run motors for 5 seconds then stop
hop_motors.start()
wait_for_seconds(5)
hop_motors.stop()
```

Note: Remind students to watch for errors in the console. Students can reference the line from the error message to pinpoint where a typing error might have occurred.

Review how the hopper model moved with this code. Then, take students to the **Knowledge Base** and open the **Motor Pairs section**. This section is an in-depth review of how to program motors together.

Review the start() section with students. Note to students that they can set variables for steering and speed. Both are integer type inputs (whole numbers) with a range from -100 to 100.

Challenge students to create a 3-2-1 countdown on their hubs and the hopper to travel a distance of 50 cm. Remind students to consider the needed libraries and the time needed to move 50 cm when writing their program.

Sample program:

```
from spike import PrimeHub, MotorPair
from spike.control import wait_for_seconds

# Initialize the hub and motor
hub = PrimeHub()
hop_motors = MotorPair('E', 'F')

# Countdown
hub.light_matrix.write('3')
wait_for_seconds(1)

hub.light_matrix.write('2')
wait_for_seconds(1)

hub.light_matrix.write('1')
wait_for_seconds(1)

# Run motors for 50cm then stop
hop_motors.start()
wait_for_seconds(7)
hop_motors.stop()
```

3. Explain

Discuss with students how they were able to move their models and review the different code combinations together.

Ask students questions like:

- How did your program work?
- How did programming two motors differ from programming one motor?
- How does the design of the hopper model determine how it is able to move?
- What was difficult about getting hopper to move?
- What was difficult about moving hopper to 50 cm when programming with seconds?

4. Elaborate

Investigate moving two motors using distance and turning.

Introduce three new lines of code to students:

```
hop_motors.start(-20,50)
hop_motors.start(90,30)
hop_motors.move_tank(10, 'cm', left_speed=25, right_speed=75)
```

Allow students time to explore these new lines of code and investigate how they can move their hopper model in various ways.

Discuss with students how they were able to move their models and review the different code combinations together.

Ask students questions like:

- How did each line of code work?
- How did your program allow the motor to move in a different direction?
- What do the two numbers after start represent?
- What are different ways you can set the speed of the motors?
- What happened when you set the motors to move for cm?
- When might you use these types of codes?

Point out to students that the numbers used in the line `hop_motors.start` are for steering and speed respectively. Both numbers are integer types meaning that only whole numbers ranging from -100 to 100 can be used.

Review this line of code specifically with students to discuss how to move the motors differently. Point out that 10 represents a float type because a decimal can be used. However, the units, in this case centimeters, is a string type since it is text and needs to be put in quotation marks as a result.

```
hop_motors.move_tank(10, 'cm', left_speed=25, right_speed=75)
```

5. Evaluate

Teacher Observation:

Discuss the program with students.

Ask students questions like:

- How were you able to program multiple motors to move the hopper model in a variety of ways?
- How does the design of the hopper model determine how it can move?
- What was difficult about programming hopper?

Self-Assessment:

Have students answer the following in their journals:

- What did you learn today about programming multiple motors?
- How does the design of a robot determine how it moves?
- What characteristics of a good teammate did I display today?
- Ask students to rate themselves on a scale of 1-3, on their time management today.
- Ask student to rate themselves on a scale of 1-3, on their materials (parts) management today.

Race Day

Grade 6-9

45 minutes

Beginner

Race Day

Students program their model to move through a course considering the best use of motor controls.

Questions to investigate

How does the task to complete determine the way a robot needs to move? Why is documenting moves important?

Materials needed

- SPIKE Prime sets
- Devices with the SPIKE App installed.
- Student journals

Prepare

- Ensure SPIKE Prime hubs are charged, especially if connecting through Bluetooth.
- Ensure students have built the Hopper model, which was used in the Hopper Run lesson.

1. Engage

Ignite a discuss with students about different types of racetracks – tracks for cars, tracks for people, tracks for bicycles, etc. There are many different types of tracks that you can race across. Consider showing videos or images of different types of racetracks.

Challenge students to create a track for the race that includes straight areas and turns.

Each group should design a basic racecourse for Hopper to move through. The course should have at least 5 steps including straight moves and turns.

2. Explore

Students will program their Hopper to run the course.

KEY OBJECTIVES

Students will:

- Create a program to move through a series of steps and turns.
- Utilize motor pair in multiple ways.

STANDARDS

CSTA

2-CS-02 Design projects that combine hardware and software components to collect and exchange data.

2-AP-10 Use flowcharts and/or pseudocode to address complex problems as algorithms

2-AP-13 Decompose problems and subproblems into parts to facilitate the design, implementation, and review of programs.

2-AP-16 Incorporate existing code, media, and libraries into original programs, and give attribution.

2-AP-17 Systematically test and refine programs using a range of test cases.

2-AP-19 Document programs in order to make them easier to follow, test, and debug.

VOCABULARY

Discuss with students how Hopper can be programmed to move in the needed ways to run along the trail. Identify different ways that they could get the motors to move without changing the design of the model.

Prompt students to think of different ways that each motor can move to make the model go straight or turn.

- For example, motor E might move quickly while motor F moves slowly using a Motor Pair tank move.
- For example, one motor could move using degrees while the other is stopped.

Ask students to write a pseudocode program to explain the needed steps and programming elements to complete the race.

Open a new project in the Python programming canvas.

- Ask students to erase any code that is already in the programming area.

For this challenge, tell students to include a code comment using the # in their code for each step of the track moves to explain the movement of the model (i.e. moving straight, way it turns, moving backwards).

Encourage students to complete one step of the race at a time. Testing and iterating on the program will be important during this challenge. Remind students to watch their console for error messages and to reference the **Knowledge Base** as needed for help.

3. Explain

Allow students to share their final programs and how they programmed Hopper.

Ask students

- How did you program your Hopper to move through the different parts of the race?
- What debugging issues did you have? Did you have any error messages during programming?
- What was difficult about this challenge?

Ask students to review the code comments used in their program to determine if the code was documented well and is easy to follow. Discuss several examples as a class to think about best practices in documenting.

4. Elaborate

Allow students to try racing on other courses developed by other groups. Students will need to create a new program in order to complete the new course. Remind students to test their program several times in order to ensure it moves as expected from the pseudocode. Ask students to add code comments using # to explain the steps of the program.

5. Evaluate

Teacher Observation:

Discuss the program with students.

Ask students questions like:

- How did you approach programming each step for the challenge?
- Why was documenting each step of your program important?
- Why is testing your programming at each step important?

Self-Assessment:

Have students answer the following in their journals:

- Why is checking each step of a long process one way to save time debugging?
- What was difficult about this challenge?
- What characteristics of a good teammate did I display today?
- Ask students to rate themselves on a scale of 1-3, on their time management today.
- Ask student to rate themselves on a scale of 1-3, on their materials (parts) management today.

Ideas to Help with Race Day

Grade 6-9

30-45 min.

Intermediate

Ideas to Help

Practice giving and using feedback from others.

Questions to investigate

- How can input from others help me make a better design and program?

Materials needed

- SPIKE Prime sets ready for student use. Prior to the first lesson, please visit the following website for help with set up, kit organization and SPIKE App <https://education.lego.com/en-us/start/spike-prime/intro>
- Devices with the SPIKE App installed
- Student journals
- Models from the Leading the Team lesson

Prepare

- Ensure SPIKE Prime hubs are charged, especially if connecting through Bluetooth.
- Ensure students have their built model and race course from the Race Day lesson.

1. Engage

Review the model for providing feedback with students.

Explain to students the following guidelines for giving feedback. Consider posting the guidelines for student reference.

- Feedback is not doing something for someone else.
- You should not rebuild a model for someone else.
- You should not type into someone's program.
- You should ask questions of each other.
- You should share your ideas and show your own programming, explaining why and how you did something.
- You should be encouraging and helpful to others and not provide negative or mean comments.

KEY OBJECTIVES

Students will:

- Give specific feedback on a peer's project.
- Explore how to use feedback to improve a project.

STANDARDS

1B-IC-20 Seek diverse perspectives for the purpose of improving computational artifacts.

VOCABULARY

Feedback, Specific, Positive, Negative

2. Explore

Have students work together to provide feedback to each other about the Race Day program.

Have two teams work together to provide feedback to each other. Teachers should model the process and what specific feedback looks and sounds like.

Review the procedure with students. Then have students take turns providing feedback.

- Team B will show their working model and explain their program.
- Team A provides feedback while Team B takes notes in their journal.
- Then teams can switch roles. Team A will show their working model and program while taking notes and listening to the feedback from Team B.

Feedback should include:

1. Tell something they really like. This could be the model, program, or design.
2. Tell something that worked well.
3. Share something the group could try differently.
4. Share anything that is confusing, did not work or that could be improved,
 - Remind students to be kind and clear in explaining why it is not clear or could be improved.
 - Let the team receiving the feedback ask questions as needed for more clarity.
 - The team giving feedback can also share ideas for improvement.

Teacher tip – Model providing feedback for the class frequently to help them learn to use positive language instead of negative language when providing feedback. Also practice taking feedback and thinking about how to use it rather than becoming defensive.

3. Explain

Have students discuss what they learned from their feedback session.

Ask students questions like:

- What did you notice in program that worked well?
- What ideas did you get from others?
- What is something you can do with your feedback?

4. Elaborate

Students should incorporate the feedback they were given.

Give students time to modify their program based on the feedback they received. Have students document their changes in their journal.

Allow students to share their updated their programs. Ask students to share what changes they incorporated and how they were able to make the changes.

5. Evaluate

Teacher Observation:

Discuss the program with students.

Ask students questions like:

- How did you use the feedback given?
- How did it feel to give feedback to others? And to receive it?
- How did you work to provide good feedback today?

Self-Assessment:

Have students answer the following in their journals:

- What did you learn today about providing good feedback?
- What did you learn today about how feedback can help in your work?
- What characteristics of a good teammate did I display today?
- Ask students to rate themselves on a scale of 1-3, on their time management today.
- Ask student to rate themselves on a scale of 1-3, on their materials (parts) management today.