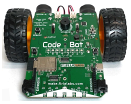


MISSION 7: Hot Pursuit
Lesson 3 (Objectives 8-11)

Time Frame: 45-50 minutes

Project Goal: Students will use proximity sensors to program the 'bot to track and chase an object.

Learning Targets

- I can apply previous knowledge of the motors to rotate and face an object moving in front of CodeBot.
- I can follow an algorithm to track an object.
- I can apply a safety feature by toggling a Boolean variable.
- I can follow an algorithm to chase an object.
- I can apply abstraction to the program by defining and calling functions.

Key Concepts

- The previous concepts of controlling CodeBot's motors can be applied and used with proximity sensors. This includes enabling the motors and supplying power to the LEFT and RIGHT motors.
- Both prox.detect() and prox.range() return a tuple. The first returns a tuple of bools and the second a tuple of integers. You can access the LEFT sensor value with p[LEFT] and the right value with p[RIGHT].
- The logical operator "not" takes only one argument. It can be used to toggle a Boolean variable and useful for turning something on/off.
- Functions are a form of procedural abstraction, a fundamental concept used in programming and STEM.

Assessment Opportunities

- Mission 7 Lesson 3 Mission Log
- Submit completed program ***HotPursuit***
- Mission 7 Obj. 8-11 Review Kahoot!

Success Criteria

□ Access the LEFT and RIGHT proximity sensor detection reading

□ Use motor controls in an if statement to track an object

□ Toggle the motors on and off using a Boolean variable and the logical operator "not"

□ Modify the if statement to chase an object

□ Define a function for driving the 'bot

□ Define a function for toggling the motors

Teacher Materials in Learning Portal

- Mission 7 Lesson 3 Slides
- Mission 7 Lesson 3 Mission Log
- Mission 7 Lesson 3 Mission Log Answer Key

Additional Resources

- Mission 7 Obj. 8-11 Review Kahoot!
- HotPursuit_obj10 sample code (learning portal)
- HotPursuit_obj11 sample code (learning portal)
- HotPursuit_ext3 sample code (learning portal)
- HotPursuit_ext4 sample code (learning portal)

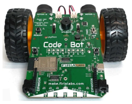
Vocabulary

- **Toggle:** Flip from True to False, or False to True; on to off or off to on.
- **Not:** A logical operator that is used to toggle a Boolean variable.
- **Abstraction:** The process of taking away or removing characteristics from something in order to reduce it to a set of essential characteristics.

New Python Code

```
if p[LEFT] and p[RIGHT]:
    motors.run(LEFT, 40)
    motors.run(RIGHT, 40)
```

Control the motors if both proximity sensors detect an object.



<table>
<tr><td><pre>elif p[LEFT]:
   motors.run(LEFT, 0)
   motors.run(RIGHT, 20)
elif p[RIGHT]:
   motors.run(LEFT, 20)
   motors.run(RIGHT, 0)</pre></td><td>Turn the ‘bot either left or right if only the LEFT or RIGHT proximity sensor detects an object.</td></tr>
<tr><td><pre>go_motors = False</pre></td><td>Define a Boolean variable that will toggle the motors enabled on/off.</td></tr>
<tr><td><pre>if buttons.was_pressed(0):
   go_motors = not go_motors</pre></td><td>Use the “not” logical operator to toggle the value of the Boolean variable. It will change from True to False or False to True.</td></tr>
<tr><td><pre>motors.enable(go_motors)
leds.user_num(go_motors)</pre></td><td>Use a Boolean toggle variable to turn on/off the motors and turn on/off a user LED.</td></tr>
</table>

Real World Applications

Review the real world applications from Lesson 1. By the end of this mission you can see more clearly how proximity sensors, auto-calibration and movement are used inside many electronic objects you might use every day. Brainstorm even more devices, or discuss how they use these systems.

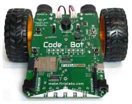
- touchless faucets, soap dispensers and hand dryers
- automatic doors
- vehicle navigation and safety systems
- factory automation systems

Teacher Notes:

- In CodeSpace, objective 11 goes into quite a bit of math and can be complicated. I haven’t found it to be very useful in improving the overall program. The slides introduce an alternative to objective 11 by discussing abstraction and having students write two more functions from their current program. This is the only time when the coding in the slides will not meet the goals of the objective. But it is the last objective, so after students finish you can just unlock the Mission and students can continue with the mission pack.
- Although movement is introduced in objective 8, the ‘bot doesn’t move forward until objective 10. Students can test Obj. 8 and 9 on their desks or on the floor.
- In previous programs, a button press was used to “wait” as a safety feature. Obj. 8 starts without a safety feature. It is added during objective 9..
- The slide deck version of objective 11 introduces the concept of abstraction. A short video from Code.org is used to give an example. You might find your own video or examples of abstraction as an alternative.

Extensions / Cross-Curricular:

- After Obj 7, the code can still throw an error if the ‘bot is not under a surface. Fix the bug by using an if statement.
- Add code to debounce button presses.
- Define a variable for the speed, and use the variable when calling the drive() function. You can use a math operation with the variable for turning.
- **STEM:** Have a lesson on abstraction. Let students come up with and present their own examples of abstraction.
- **ENGLISH LANGUAGE ARTS:** Have students write a story about an electronic device that uses proximity sensors and moves. It can be about how useful and helpful it is, or about how things might go wrong.
- **ENGLISH LANGUAGE ARTS:** Have students write an opinion piece about technology and its effects on society.
- Supports **language arts** through reading instructions, guided notes, and reflection writing.



Preparing for the lesson:

- Look through the slides. Decide what materials you want to use for presenting the lesson. The slides can be converted to Google Slides. They can be projected on a large screen.
- Look over objective 11 in CodeSpace and in the slide deck. They are different! You can skip Obj. 11 altogether, or decide which one you want students to complete.
- Be familiar with the mission log assignment and the questions they will answer. Prepare the assignment to give through your LMS.
- Have several objects of different shapes, sizes and colors for the ‘bot to follow.
- Have small paper cups for the ‘bot to follow. I suggest paper cups, not plastic, about the length of the front of the ‘bot.
- If you have a word wall, or another form of vocabulary presentation, prepare the new terms.

Lesson Tips and Tricks:

💡 **Teaching tip:**

You can use a variety of discussion strategies to get the most engagement from your students. For example, you can have students write their answers before asking anyone for an answer. You can use one of many think-pair-share methods.

🧑‍🤝‍🧑 **Pre-Mission Warm-up:** -- slide 2

Students can write in their log first and then share, or discuss first and then write in their log. The warm-up questions review CodeBot movement. Students can share their answers, or compare with each other.

- Question: What code must be included before the motors of CodeBot will run?
- Question: What code will turn the ‘bot to the left?

💻 **Mission 7 Lesson 3 Activities:**

The Chrome browser works best, but other browsers also support CodeSpace. Each student will complete a Mission Log. Students could work in pairs through the lesson, or they can work individually. You should continue with the same structure as the first two lessons.

💡 **Teaching tip: Mission Introduction** -- slides 3-6
This mission is divided up into three lessons. This lesson completes the last two goals. Students answer one question in their mission log.

💡 **Teaching tip: Objective #8** -- slides 7-10
These slides show how to access the LEFT and RIGHT proximity sensor readings. It is similar to accessing them from the prox.range() function with sensed. An example is given. Students should answer the questions in the Mission Log.

💡 **Teaching tip: Objective #8** -- slides 11-12
These slides go over the algorithm for tracking an object and show the code that goes with each step. It is fairly straightforward.

💡 **Teaching tip: Objective #8** Activity -- slides 13-17
Students open their program HotPursuit and add code. All the cod was given with the algorithm, but it is repeated here. Slide 17 suggests experimenting with different speeds for tracking. It is optional.

NOTE: Students don’t include a safety feature in their code at this time. But the ‘bot doesn’t move forward, so there shouldn’t be any danger. The safety feature is added in the next objective.

💡 **Teaching tip: Objective #9** -- slides 18-20
Students learn about “not” and how to toggle a Boolean variable. They practice this concept with problems on their mission log.

💡 **Teaching tip: Objective #9** -- slides 21-22
These slides go over the algorithm for toggling a Boolean variable and show the code that goes with each step. It is fairly straightforward.

💡 **Teaching tip: Objective #9 Activity** -- slides 23-27
Students modify their code by adding another button press for toggling. This is the safety feature. The slides walk them through the coding and testing process.

💡 **Teaching tip: Objective #10** -- slide 28
Students modify the program to chase instead of just track.

💡 **Teaching tip: Objective #10 Activity** -- slides 29-31
Students modify the if statement for moving forward. They also need to test the program with objects of different shapes, sizes and colors. They write the results on the mission log. This is a good time to try the paper cup on its side to see if the ‘bot follows it as it rolls.

NOTE: The slides give instructions for a different Objective 11. Students will write different code than what is in CodeSpace. It has less math, no complicated algorithm, and introduces abstraction. If you are following the instructions in CodeSpace, then do not use the slides from here on out.

💡 **Teaching tip: Objective #11** -- slide 32
Introduction to the modified objective 11.

💡 **Teaching tip: Objective #11** -- slides 33-36
These slides go over abstraction. A short video can be shown that explains abstraction, and examples are given. You can skip all this information about abstraction and just get to the code, or you can take some time to explain and discuss abstraction.

💡 **Teaching tip: Objective #11 Activity** -- slides 37-43
These slides give the instructions for defining two more functions using pre-existing code. NOTE: The function on slide 42 needs the global statement for its variable. The slide talks about this but doesn’t show it. Make sure students add this line of code.

💡 **Teaching tip: Extensions** – slides 44-47
Three extensions are given for this lesson.

- The first extension is the same as Lesson 2 – add a ‘bug’ fix to cal_power().
- The second extension is to include the code for debouncing a button press. Sample code that includes these two fixes in included in the learning portal (HotPursuit_ext3)
- The third extension is to define a variable for speed and use it as an argument in the drive() function calls. They can simplify the need for more variables by using a math relationship between speed (moving forward) and turning. An example of this is given slide 47. Sample code that includes all three extensions is available in the learning portal (HotPursuit_ext4).

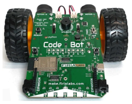
Optional: 🗝️ Mission 7 Obj 8-11 Kahoot! Review. A review Kahoot! Is available for these four objectives.

🧑‍🤝‍🧑 **Post-Mission Reflection:**

The post-mission reflection asks students to review what they learned during this lesson. Answers can vary widely, depending on each student’s experience.

You can use a cross-curricular activity for a post-mission activity.

End by collecting the Mission 7 Lesson 3 Log.



SUCCESS CRITERIA:

- □ Access the LEFT and RIGHT proximity sensor detection reading
- □ Use motor controls in an if statement to track an object
- □ Toggle the motors on and off using a Boolean variable and the logical operator “not”
- □ Modify the if statement to chase an object
- □ Define a function for driving the ‘bot
- □ Define a function for toggling the motors