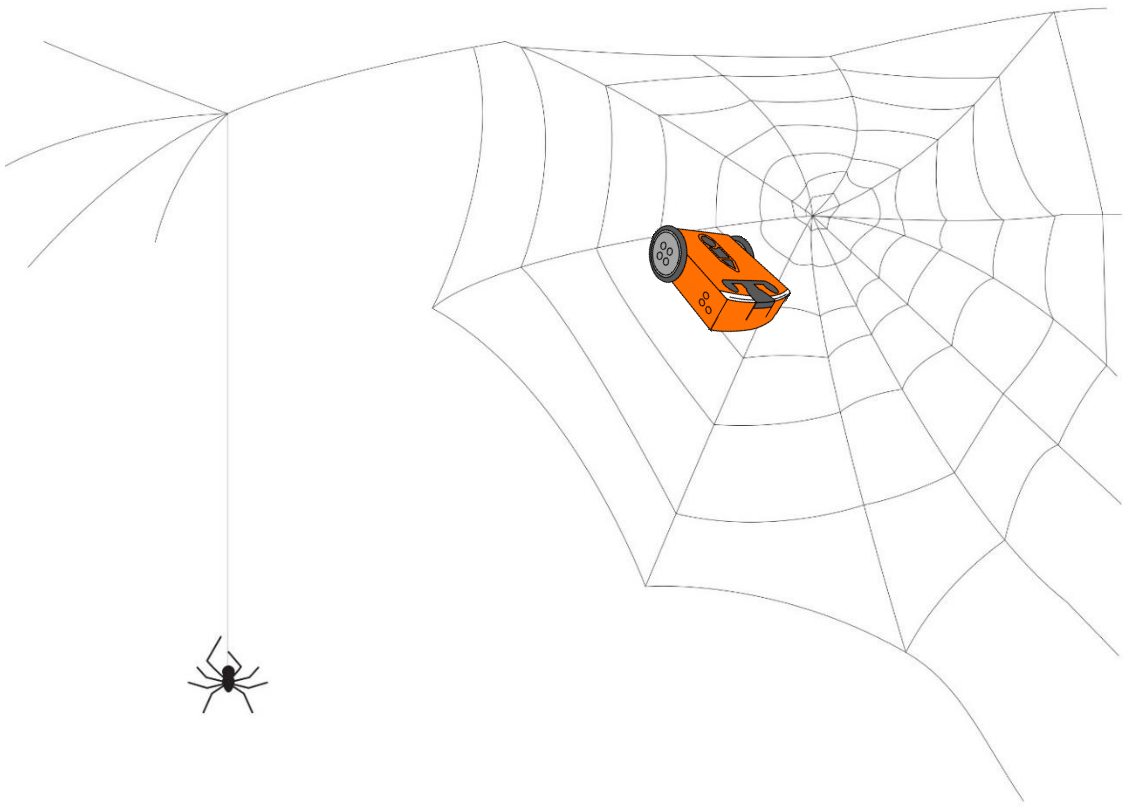
Edison and the spiralling spider trap

Teacher's notes



Doc 3.2.4.5.2.1.1 Rev 1.1

Contents

Go ahead – show off!

We love seeing how classrooms use Edison! If you and your students want to share your Hour of Code Edison EdVenture, be sure to tag us into the fun!

	@meetedison	twitter.com/meetedison
	@meet_edison	instagram.com/meet_edison
	@meetedison	facebook.com/meetedison

About this lesson and guide

This guide offers teachers and instructors overview information, facilitation recommendations and other supporting information for the *Edison and the spiralling spider trap* lesson available at https://meetedison.com/robotics-lesson-plans/spiralling-spider-trap/

Do you need to read this whole guide to run the lesson? **Absolutely not!**

Once the robots and programming devices are set up[1], you can start learning along with your students! The student sheets for this lesson have been designed to allow students to work through the different parts of the lesson independently, learning about the Edison robot, how to use the EdScratch programming environment, and the key computer science learning objectives of the lesson. This guide simply offers further information for teachers and instructors to help make using this lesson easy and fun.

Each part of the lesson is included in this guide along with any relevant supporting information for that part. Supporting information is divided into the following sections:

Overview
General information about the section and key learning objectives covered.

Delivery recommendations
Suggestions for how you can cover the lesson section if you want to run the lesson in a more facilitator-led capacity.

Tips and tricks
Helpful hints and ways to overcome common issues students may encounter.

Creative Commons licence attribution details

Developed and written by: Kat Kennewell
Illustrations by: Jin Peng

The *Edison and the spiralling spider trap* lesson set was developed using resources from the the EdScratch Lesson Plans Set[3].

[1] The *Getting started with Edison and EdScratch* set available at https://meetedison.com/robotics-lesson-plans/edison-designer has step-by-step help for setting up your robots and programming devices. If you are new to Edison or EdScratch, it is recommended you start with that guide.
[2] http://creativecommons.org/licenses/by-sa/4.0/
[3] https://meetedison.com/robot-programming-software/edscratch/#EdScratch-resources

Lesson overview

Introduce the key computational concept of variables using Edison robots and the Scratch-based programming language EdScratch[4]. Variables, which can be created and utilised through the special 'Data' category in EdScratch, are introduced and then explored in this lesson. The concept of using maths in programming, and the practice of tracing code are also explored and practised.

Grade levels	Difficulty	Duration
Year 6+	Intermediate	55+ minutes

Computer science and computational thinking topics	☐ Variables ☐ Code tracing ☐ Computations in programs
Tie-ins to other subjects	☐ Mathematics: algebra and geometry

Prerequisite knowledge

To be successful with this activity, it is recommended that students:

- Have used Edison and EdScratch[5]
- Understand sequence and sequential programming
- Understand definite loops ('for' loops)
- Are comfortable using addition, multiplication and degrees as a measurement for angles

Supplies you need

- Full set of Edison robots[6] and EdComm programming cables
- Full set of prepared programming devices (computers or tablets)
- 4x AAA batteries per robot
- Print-outs or digital copies of the student sheets
- **Optional**: various 'maker-space' craft materials [such as string, sticky tape, Blu Tack, paper, recycled materials, and other similar materials] for the 'sticky spider thread' bonus challenge

[4] https://meetedison.com/robot-programming-software/edscratch/
[5] The *Getting started with Edison and EdScratch* set available at https://meetedison.com/robotics-lesson-plans/spiralling-spider-trap/ has a step-by-step activity for introducing Edison and EdScratch. If this is your first time using Edison or EdScratch, start with that guide and activity.
[6]This activity assumes Version 2.0 Edison robots. If you have Version 1 Edison robots, you will need to adjust the activity to have the robot turn using seconds as the input parameter rather than degrees. Learn more at https://meetedison.com/meet-edscratch-edison-robots-scratch-language/#V1-EdScratch

Aren't variables hard? A special note about this lesson

The primary focus of this lesson is variables and data, two of the most fundamentally important parts of general-purpose programming languages. It's not uncommon for students (and instructors!) to feel like these are 'hard' areas of programming, especially if this is your first time working with variables. Learning about variables and maths in programming is an important step in meaningful computer science education. Just like anything that is brand new, becoming comfortable with maths and variables in programs can take some time. Encourage students to be patient with themselves as they work through the activity. They will soon see the creative potential these new skills unlock inside programming!

Some great advice from the Hour of Code team

It's okay not to know! Respond to student questions and struggles with phrases like:

- "I don't know. Let's figure this out together."
- "Technology doesn't always work out the way we want."
- "Learning to program is like learning a new language; you won't be fluent right away."

And don't forget to have fun! (^_^)

Part 1: What are variables?

Overview

The concept of variables, one of the fundamental components of code, is introduced in this section. The student sheets explain how variables work, then asks students to trace a program using a variable to understand the value of that variable at different points in the program. Understanding how variables work and how the value of a variable can change will allow students to create their own programs using variables.

Delivery recommendations

- Recommended time: 25 minutes
- Answer key to the student sheet code trace table:

In loop #	Starting value of DriveLength	Wait block input value (in milliseconds)	New value of DriveLength
1	1	200	2
2	2	400	3
3	3	600	4
5	5	1000	6
7	7	1400	8
10	10	2000	11

- To ensure students are understanding the concept of variables, you can ask them to answer the following two questions as a part of this section:
  - What does Edison do when you run this program? What 'shape' does the robot drive in?
  - Look at the code in the program. Explain why the robot drives in the pattern you see when you run the program in Edison. What in the code makes the robot move in that pattern?

Tips and tricks

- Only variable names using legal characters will be able to be compiled and sent to Edison. If students get error messages when they try to download a program with variables, check the variable's name.

- For a deeper exploration on using Edison's 'set motors' blocks, refer to activity U2-2.5 *Let's explore Edison's motors* in the free EdScratch Lesson Plans Set[7].
- When it comes to expressions and any mathematics in EdScratch programs, it is important to note that Edison can only handle integers. Likewise, Edison can only work in 'real' mathematics, so if students put in computations which, for example, divide by zero, they may get errors, or the robot may behave in unexpected ways.
- If an operator input parameter is left blank in an EdScratch program, it will be read as '0' (zero) by the robot.
    - The operator may display NaN in this case, which stands for 'not a number'.
- Whenever students want to use a variable or an operator block inside a 'wait' block in EdScratch, they must use the 'wait ( ) milliseconds' block. (The other wait block cannot except bubble-shaped blocks as input parameters). The reason this block is in milliseconds instead of seconds has to do with a limit regarding how Edison can compute math. Essentially, Edison doesn't know what decimals are. Therefore, 0.3 seconds when stored in Edison is rounded and becomes 0 seconds. This makes doing any sort of maths wildly inaccurate. By using milliseconds, we can represent 0.3 seconds as 300 milliseconds, which is a number Edison understands. This enables computations to be performed without gross rounding errors.
    - The other inputs which are in seconds in EdScratch can accept decimals in the input parameters. The compiler that converts the EdScratch code before it is sent to Edison converts all time inputs in these blocks into milliseconds before the values are sent to Edison.
- If your students have already studied basic algebra, you may choose to demonstrate that the values table in this worksheet can be filled out by thinking of the three columns as n, n*200, and n+1.
- Due to minor mechanical differences in the motors and encoders inside different Edison robots, students may need to adjust the input parameters of the drive block related to the turn in this program to suit their robots. To improve accuracy, students can also add either a 'stop motors' block or a 'wait' block with a sort input value (e.g. .2 secs) in-between the drive commands (including at the bottom of the code inside the loop).
- If students are struggling to see the pattern Edison makes when running this program, try attaching a pen to Edison so that the robot 'draws' the shape as it moves. To make attaching a pen quick and easy, consider making a 3D-printed pen holder, such as the one available at https://www.thingiverse.com/thing:2949946

[7] https://meetedison.com/robot-programming-software/edscratch/#EdScratch-resources

Part 2: How do you program a spiralling spider trap?

Overview

Building on the programming concepts explored in part 3, this section asks students to re-imagine the ‘spiral-out’ program to be a spiral-in program instead. Designed to be a stepping-stone activity to help students explore using variables and computations in self-created programs, this activity challenges students to think through the logic of the ‘spiral-out’ program in order to modify and reapply its main components.

The extra ‘bonus challenge’ attached to this activity offers students an opportunity to mix computer programming and physical engineering to turn their robots into trap-laying, spiralling spiders.

Delivery recommendations

- Recommended time: 30 minutes (**Optional:** plus an additional 30-45 minutes for the bonus challenge)
- An example programming solution can be seen at https://www.edscratchapp.com?share=rDBEwzbR
- The bonus challenge brings in engineering to physical computing, but can take a while. If you have the opportunity, consider making this activity its own lesson to allow students the chance to explore the design-build-test cycle as they design their spider silk solution.
    - Engineering a way to leave the ‘spider silk’ AND get that ‘silk’ to stay in place (rather than get pulled along and out of shape by the robot as it moves) is not a fail-proof activity. Many factors; including the types of materials that students use, the way they attach (or don’t attach) the string (or other substance) to the robot and driving surface, the speed at which the robot drives, etc; will affect how successful they are in this bonus activity. Students may not be able to create a way of completing this extra challenge ‘perfectly’. That’s fine! The key learnings will come from the process of experimentation and problem-solving, not the final outcome.
    - If students want to see the shape their robot makes but skip the string, consider having them attach a pen to the robot instead.

Tips and tricks

- If students are feeling stuck, suggest they review the program in section 3 to help begin to work out their ‘spiral-in’ programs.

- Remind students to use good variable names and avoid illegal characters in their variable’s name.

What should I do if a student finishes the lesson early?

- Encourage students to make another shape using variables and Edison.
- Or, ask students who finish early to help classmates who are having trouble with the activity.