



# EdScratch lesson plans

## Teaching guide and answer key



The EdScratch Lesson Plans Set by [Kat Kennewell](#) and [Jin Peng](#) is licensed under a [Creative Commons Attribution-ShareAlike 4.0 International License](#).

## Contents

About this guide .....	3
What's in this guide.....	3
Creative Commons licence .....	3
How to use this guide .....	4
Understanding the activity types .....	4
Reading the activity overview .....	5
Using the answer key .....	6
Supplies you will need .....	7
Frequently asked questions .....	8
Before you start.....	12
Setting up your programming devices for EdScratch.....	12
Get Edison ready .....	13
Unit overview.....	14
Unit 1: Get started .....	17
Lesson 1: Meet Edison .....	18
Lesson 2: Meet EdScratch.....	32
Unit 2: Move it! .....	37
Lesson 1: Sequence .....	39
Lesson 2: Inputs and outputs.....	47
Unit 3: Got loops? .....	65
Lesson 1: Loops .....	67
Lesson 2: Interrupts .....	80
Unit 4: What if.....	89
Lesson 1: Conditionals .....	91
Lesson 2: Sensing .....	101
Unit 5: Versatile variables.....	121
Lesson 1: Maths and data in EdScratch .....	122
Unit 6: Inventor's time!.....	140
Lesson 1: Design, build, test, repeat.....	141
Appendix 1: Blank digital display number .....	149
Appendix 2: Calibrate obstacle detection .....	150

## About this guide

This guide offers teachers and instructors overviews, delivery recommendations and other supporting information for the EdScratch lesson activities available at <https://meetedison.com/robot-programming-software/edscratch/>.

The EdScratch activities are designed to allow students to work independently. As students work through the activities, they will develop a familiarity with computational thinking and fundamental computer science concepts, mastering a range of related skills as they progress. This guide outlines more about the concepts to be taught, approaches to be taken, skills to develop and processes to be learned. Here you will find strategies, ideas, and further information to help make teaching using Edison and EdScratch effective, easy and fun.

## What's in this guide

The EdScratch lesson activities are organised into six units – from an initial preparatory unit to a culminating project-based unit – with each unit containing one or two lessons. Each lesson contains a mix of structured and open-ended activities that introduce the key concepts and learning objectives while engaging students in active exploration of the learnings.

This guide provides an overview and supporting information for all of the EdScratch units, lessons and activities. In this guide you will find:

- an overview of each unit, including the key learning objectives of that unit,
- an overview of each lesson with a list of all activities in that lesson,
- a dedicated section per activity detailing:
  - the activity's purpose and objectives,
  - supporting information including estimated time requirements, supply requirements, and tips for delivery, and
  - an answer key including recommendations on assessing student work.

## Creative Commons licence

These teaching resources have been released under a Creative Commons licence. You are free to use these resources as they are, translate them, share them or use them as the base to develop your own customised lessons.

## Licence and attribution details

The EdScratch Lesson Plans Set is comprised of the EdScratch lesson activities and this guide. The collection is licensed under a [Creative Commons Attribution-ShareAlike 4.0 International License](https://creativecommons.org/licenses/by-sa/4.0/).

Developed and written by: Kat Kennewell

Illustrations by: Jin Peng

## How to use this guide

This guide is organised around the activities, lessons and units that make up the EdScratch lesson set.

The EdScratch lesson activities are divided into six units, each of which has one or two lessons. Each lesson contains a number of activities which are all related to that unit's theme and tied to the learning objectives of that unit.

## Understanding the activity types

The EdScratch lessons are comprised of activities designed to gradually introduce and reinforce computational thinking, computer science and problem-solving skills. All lessons include base activities which introduce key ideas and contain tasks requiring students to apply what they have learned. Supplemental extension activities, designed to further explore and expand on the core learnings of each lesson, are also included.

There are three types of lesson activities: 'Let's explore' activities, 'Change it up' activities, and 'Challenge up' activities.

### Let's explore activities – core learning

All lessons include 'Let's explore' activities. These activities are the base activities which introduce new concepts and explain key ideas. 'Let's explore' activities include explanations and guidance around the core learnings, then have tasks for students to complete. The tasks allow students to explore the concepts and skills being introduced.

### Change it up activities – extension activities

'Change it up' activities are extension activities that reinforce skills and concepts that have already been introduced. While 'Change it up' extension activities do not introduce new material, they do offer new projects or tasks to try. 'Change it up' activities are generally less structured than 'Let's explore' activities but still include sufficient guidance and hints to help students undertake the tasks or projects.

### Challenge up activities – extension activities

Similar to 'Change it up' activities, 'Challenge up' activities are also extension activities which expand on and reinforce the material that has been previously introduced. The tasks and projects outlined in 'Challenge up' activities are generally more open-ended and offer less structured guidance than 'Let's explore' or 'Change it up' activities.

## Reading the activity overview

Every EdScratch lesson activity has a dedicated section included in this guide. Per activity you will find the following information:

- the activity's ID number and name,
- an estimate of the activity's size,
- delivery recommendations (if any),
- a list of the supplies and resources needed for the activity,
- an overview of the activity,
- any tips or tricks to help in delivering the activity, and
- an answer key with assessment suggestions (as needed).

## A note on activity size estimates

The size estimates (given as 'small', 'medium', 'large', or 'project') are indications of the size of the activity compared to other activities in the EdScratch lesson set. The time required to complete an activity can vary greatly depending on each students' age, experience and interest in the activity.

As a general rule:

- 'small' activities will require less than one standard teaching block (less than 45 minutes)
- 'medium' activities should be able to be completed within one standard teaching block (45 minutes or less)
- 'large' activities may take students more than one full teaching block (approximately 45 minutes – 90 minutes)
- 'projects' are complex activities which will generally need multiple teaching blocks (minimum of 90 minutes)

It is recommended to work through a few introductory activities with your students to gauge how long they will need, on average, to complete activities of each size.

## Using the answer key

All of the EdScratch activities have tasks or projects to be completed. Many of the worksheets also require students to note down elements of their experience or to provide answers to questions in various formats. For every activity which has students provide answers to a question or questions, you will find an answer key in that activity's section.

The answer key lists both the question number and question type, then provides the answer (or sample answer) along with marking notes, as required.

*Example answer key:*

Question	Type	Sample answer	Marking notes
1	SE	<i>When I turned Edison on, the red LEDs came on and started flashing. Edison also made a chirping noise one time.</i>	There are two key things to look for in students' answers: - The robot's red LED lights turn on and begin to flash. - The robot makes a noise.

## Answer types

The answer type column identifies what style of response the question has asked students to provide. There are three possible answer types:

- **Exact answer (EA):** a question which has an exact solution which the student must provide.
- **Result code (RC):** a question which asks students to capture the code they have as a result of their programming, where a variety of code solutions are possible.
- **Student experience (SE):** a question which asks students to capture their experiences or outcomes as a result of their experimentation and programming, where a variety of responses are possible.

It is valuable to note the answer type when marking students' work. When marking result code (RC) or student experience (SE) type answers, it is important to remember that many answers are possible.

## Supplies you will need

All lesson activities require a set of V2.0 Edison robots and internet-connected programming devices (desktops, laptops, Chromebooks or tablets). Be sure to read the '[Before you start](#)' section of this guide carefully to learn how to set up your programming devices to work with EdScratch. Ideally, you will want to have an equal number of Edison robots and programming devices.

The majority of the lesson activities are designed to work in a 1:1 student-to-robot ratio. However, some of the activities are best suited to pair or group work, with multiple students sharing one or more Edison robots. Learning about robots can initially be an overwhelming task for some students, especially if they are new to computer science, so having a supportive partner can often be helpful.

The majority of the 'Let's explore' base activities can be completed using just basic supplies along with the student worksheets and activity sheets included in the EdScratch lesson activities set. Many of the 'Change it up' and 'Challenge up' extension activities require additional supplies to enable students to get the most out of physical computing. The following lists detail the basic supplies and key additional resources you need to get the most out of the EdScratch lesson activities.

### Basic supplies:

- Full set of Edison robots and EdComm programming cables
- Full set of programming devices (computers or tablets)
- 4x AAA batteries per robot (please see '[Get Edison ready](#)' in this guide for more information on batteries)
- Print-outs of student worksheets and activity sheets

**Note:** Most lessons will list '**basic supplies set**' as an item in the 'resources needed' section of the activity overview. A 'basic supplies set', in this instance, means a full set of Edison robots, EdComm programming cables and programming devices as well as batteries per robot.

### Additional resources:

- Dark coloured tape (e.g. black electrical tape)
- Torches (flashlights)
- Opaque objects to use as obstacles and walls for 3D mazes
- TV or DVD remote controls
- EdCreate kits and any other LEGO brick compatible building system parts
- Various 'maker-space' craft and build materials [such as coloured construction paper, large-sized paper (e.g. butcher's paper), glue, felt, cardboard, pipe cleaners, recycled materials, and other similar materials]

## Frequently asked questions

The following section includes some of the most common questions about using the EdScratch lesson activities set. Additional information about using the Edison robots, including troubleshooting help, is available at the Meet Edison website:

[www.meetedison.com](http://www.meetedison.com)

### **Q: How much do students need to already know about coding and robotics to use these lessons?**

A: No prior knowledge of either coding or robotics is needed to complete the EdScratch lessons. If you or your students are new to coding and robotics, you may find working through the lessons in the order they are presented in this guide to be the best way to build up your knowledge and get the most out of these activities.

### **Q: Do students need to know Scratch to use these lessons? Do these lessons tie into Scratch?**

A: The lessons assume that students have not used a Scratch programming environment previously. If students are familiar with Scratch, however, they will likely find using the EdScratch environment fairly straightforward.

The EdScratch lessons cover core coding and computer science concepts that can also be taught or reinforced using MIT's Scratch programming environment. However, EdScratch and Scratch are separate programming languages – programs written in one language are not compatible with the other. If you are also using Scratch with your students, you may want to do activity [U1-2.1b Change it up: Does EdScratch = Scratch?](#) (unit 1, lesson 2) to explore this concept further.

### **Q: Do students need to do all the activities in a unit to cover the learning objectives of that unit?**

A: The EdScratch lessons are designed to gradually introduce and reinforce key computational thinking, computer science and problem-solving skills. All lessons include 'Let's explore' activities. These activities introduce new concepts, terminology and explain key ideas. These core learnings are then expanded and reinforced through the 'Change it up' and 'Challenge up' extension activities in the lesson. This modular design is intended to allow for more flexible teaching and learning through a mix of whole-class, small groups, pair work and individual study.

Having students complete all of the 'Let's explore' activities in a unit will ensure that they are introduced to all of that unit's key computational thinking and computer

science learning objectives. You may find that using at least a few of the ‘Change it up’ or ‘Challenge up’ extension activities in a unit will help students cement their understanding of important topics and build up skills. Including some of the ‘Change it up’ and ‘Challenge up’ extension activities will also ensure students get exposure to real-world problem-solving and robotics project opportunities.

**Q: Are the solutions in the answer key the only correct answers?**

A: In some cases, there is a single correct answer, but in most cases many solutions are possible. Programming is creative – don’t be too quick to dismiss your students’ ways of thinking. What might appear to be ‘incorrect’ can be an acceptable solution, even when a more elegant solution is also possible.

**Q: Can I modify the suggested activities and projects?**

A: Absolutely! Tools, such as Edison and EdScratch, can be powerful motivators on their own. However, to get the most out of any ‘cool tool’, it needs to relate directly to the interests of the students using it. Choosing extension activities and projects that are the most relevant to your students’ lives and interests will help them to get the most out of the lessons overall.

**Q: Do I need to follow the units in order? Do I need to do the lessons and activities inside a unit in order?**

A: The EdScratch lessons are designed to allow for flexible teaching. You may choose to rearrange the content to best suit your students’ needs. For example, you may want to move some units, lessons or activities around to match better with other areas of study your students are engaged in at the time. If your students have used Edison robots before (for example, with the [EdBlocks programming environment](#)) you may also choose to skip some activities.

However, practical skills in computer science need to be built up from some key foundational concepts, and some elements of the lessons are progressive. As such, you may find it easiest to work through the lessons and activities sequentially.

**Q: Should the whole class work on the same activity together? Can students jump ahead?**

A: How you teach the content is really up to you! All of the lessons are designed to enable independent learning by your students. You may choose to let students

progress through all or some of the material at their own pace. Units make good stopping points, so you may choose to let students work independently through a unit, but no further, for example.

Depending on your students' ages, abilities and familiarity with Edison robots and Scratch programming environments, you may find that some students complete activities more quickly than others. Making 'Change it up' and 'Challenge up' extension activities available for students to work through at their own pace can help keep the entire class engaged but on the same topic.

As programming is, at its heart, about participation, collaboration, creativity and sharing, having students who finish an activity more quickly work together with others who require additional time is another great way to keep the whole class engaged and progressing together.

**Q: What reading level do students need to read at to use the student worksheets independently?**

A: The original set of student worksheets, written in Australian English, are designed for independent use by students in Year 5 (10-11 years old) and above. Overall, the complete worksheet set has an average reading level of 6.6 on the Flesch-Kincaid Grade Level scale with later units averaging slightly higher reading requirements than earlier units.

*Please note: Translated and modified versions of these lessons may have different reading level requirements.*

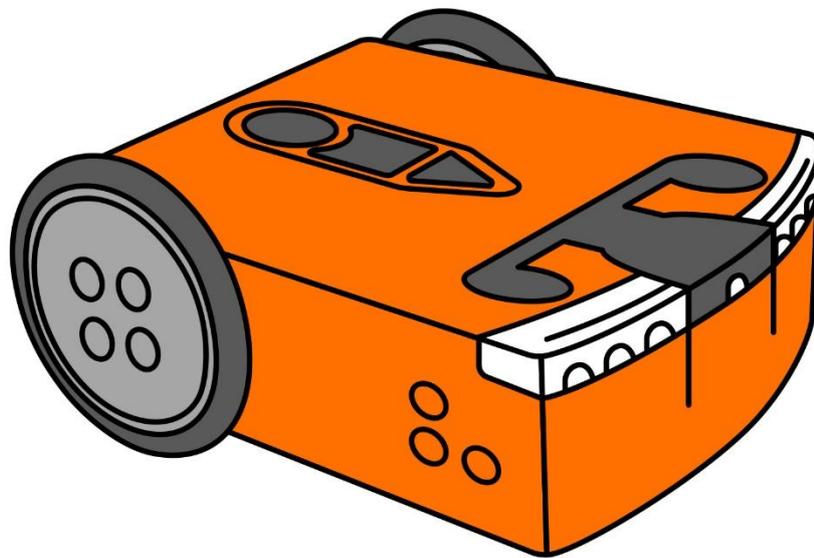
**Q: I have Version 1 Edison robots. Can I use EdScratch and these lessons?**

A: As a general rule, you can use Version 1 Edison robots which have the latest firmware update (available at <https://meetiedison.com/edison-robot-support/firmware-update/>) with EdScratch and the activities in this EdScratch lesson set. However, there are a few limitations, especially when it comes to Edison's motor outputs. When using EdScratch blocks from the 'Drive' category in EdScratch with Version 1 robots, only use 'seconds' as the distance units input parameter. Using 'cm', 'inch', or 'degrees' as the distance units input parameter will not work with Version 1 robots and can result in some bizarre behaviour from the robot. In all activities and projects where students use 'Drive' category blocks, be sure to remind students to always and only use 'seconds' as the distance units input parameter.

It is also recommended that you avoid using the sensing and event EEdScratch blocks related to 'drive strain' with Version 1 robots as these blocks can cause the

robots to behave erratically. (*Note:* none of the activities in these lessons use the 'drive strain' sensing or event blocks.)

What's the reason behind these limitations? The key cause has to do with some physical differences between Version 1 Edison robots and V2.0 Edison robots. Edison V2.0 robots have wheel encoders which allow the robots to travel specific distances at exact speeds. These encoders enable the robots to use distance units other than time, including centimetres, inches and degrees. Version 1 Edison robots do not have encoders and therefore are not capable of the precision driving required to use these other inputs.



**Love these lessons? Hate them? Have an idea for a lesson activity?**

The team behind EdScratch and Edison would love to hear from you! You can share your feedback and ideas with us through the contact form on our website at [www.meetedison.com/edison-robot-support/contact-us/](http://www.meetedison.com/edison-robot-support/contact-us/)

## Before you start

Before using Edison with your students, you will need to set up the programming devices, (i.e. the computers or tablets) you will be using with the EdScratch app and get the Edison robots ready.

Additional details can be found in the free *Getting started with Edison guide* available at <https://meet Edison.com/content/Get-started-with-Edison-guide-English.pdf>

## Setting up your programming devices for EdScratch

Depending on the type of programming devices you are using, there are a few things you will need to do to prepare them to be able to work with the EdScratch app.

If you are using computers running Windows operating systems, you will need to disable sound enhancements. Please go to <https://meet Edison.com/edison-robot-support/trouble-shooting/#soundenhancements> to find step-by-step video guides showing you how to disable sound enhancements for standard Windows sound enhancements as well as the most common third-party software programs.

To be able to program Edison, most devices will need the volume turned up to maximum or 100%. As many devices have built-in safety settings that reduce the volume when an audio device is connected using the headphone jack, it is also important to double check that your volume is turned all the way up after plugging in the EdComm programming cable to your device.

Additional details specific to different types of devices (Windows, Macs and tablets) can be found in the free *Getting started with Edison guide* available at <https://meet Edison.com/content/Get-started-with-Edison-guide-English.pdf>

## Get Edison ready

To get Edison ready for use, you need to:

1. Open the battery compartment at the back of Edison and remove the EdComm programming cable.
2. Insert 4 'AAA' batteries. Please refer to the picture to ensure that the batteries are inserted correctly. Be sure to reclose the battery case by clipping the battery cover back on.

*Please note:* Low or flat batteries can cause a range of issues with Edison. For this reason, please be sure always to use fresh, fully charged batteries in your robots.

*Choosing batteries:* If using disposable batteries with Edison, only ever use alkaline batteries.

(These are the most common standard AAA batteries you will find in just about any shop.) If you are using rechargeable batteries with Edison, only use nickel metal hydride (NiMH) rechargeable batteries. Never use lithium rechargeable, heavy-duty disposable, super heavy-duty disposable or carbon zinc batteries.

3. To turn Edison on, flip the robot over. Slide the power switch to the 'on' position, as shown in the picture. This will turn Edison on, and the red LED lights will start flashing.



Ensure the batteries are in the right way.



Push the switch towards the 'on' symbol.

*Please note:* While Edison will turn off automatically if not used after five minutes, we recommend you turn the robots off manually when not in use.

## Unit overview

A summary of each unit, including that unit's learning objectives, follows.

### Unit 1: Get started

Prepare for your EdVenture! This initial unit, which includes two lessons with a total of four base activities and ten extension activities, introduces students first to the Edison robot and then to the EdScratch programming environment. Students gain familiarity with the Edison robot through a series of activities including programming the robots using barcodes. They then learn the basics of the EdScratch programming environment in preparation for the next unit.

The learning objectives for this unit are that students will:

- develop a working familiarity with the Edison robot, including its key physical features (power switch, sensors and buttons),
- be able to program an Edison robot using barcodes, and
- know how to access EdScratch and how to download a program from EdScratch to Edison.

### Unit 2: Move it!

Focus in on the key computational concept of sequence in this unit which includes two lessons with a total of eight base activities and 13 extension activities. Students explore Edison's abilities to move using its motors, plus use the robot's LEDs and sound-producing buzzer through a range of activities. Computer programming fundamentals including inputs, outputs, bugs and debugging are introduced. Students begin to develop their familiarity with programming Edison in EdScratch and with using Edison as the base for creative robotic builds.

The learning objectives for this unit are that students will:

- be introduced to the idea of computational thinking and begin to use computational frameworks when approaching tasks,
- become familiar with the concept of sequence,
- learn what inputs, outputs and input parameters are,
- explore EdScratch's core output blocks with Edison, including the 'Drive', 'LEDs' and 'Sound' categories in EdScratch,
- be able to create and modify sequential programs for Edison in EdScratch using basic output blocks and those blocks' input parameters, and
- begin to explore robotics applications to real-world situations through projects.

### Unit 3: Got loops?

Examine the key computational concept of loops and explore different ways loops can be used to control how the Edison robots behave in this unit consisting of two lessons with a total of six base activities and 14 extension activities. The topic of programming logic is examined more closely, including how control structures can affect the flow of code. Students continue to discover the EdScratch environment, learning more about the various block types in EdScratch and ways in which these blocks can be used to create with Edison.

The learning objectives for this unit are that students will:

- be introduced to the computational thinking concept of repetition through the computer programming structure of loops,
- develop a working understanding of the difference between definite and indefinite loops,
- combine the concepts of sequence and loops into working programs in EdScratch,
- be introduced to the programming concept of interrupts,
- experiment with new blocks in the 'Control', 'Comments' and 'Events' categories in EdScratch, and
- expand on their problem-solving abilities by creating and programming robotics projects.

### Unit 4: What if...

Explore selection and branching in computer programming through the key computational concepts of conditionals and events in this unit's two lessons with a total of nine base activities and 15 extension activities. Key computer programming skills, such as developing pseudocode, are introduced to help students further their problem-solving abilities as they unlock the Edison robots' various sensor capabilities. Students learn about algorithms and use this understanding to create programs enabling more autonomous behaviour from the robots. Conditionals, sensing, interrupts and event-based programming are brought to life through the physical computing activities in this unit.

The learning objectives for this unit are that students will:

- be introduced to the computational thinking concept of conditional selection (i.e. branching),
- explore how inputs work with Edison using new blocks in the 'Control', 'Sensing' and 'Events' categories in EdScratch,
- experiment using Edison's sensors and input capabilities in working programs,

- combine the concepts of sequence, loops and selection into working programs in EdScratch,
- develop a practical understanding of how to use pseudocode and comments to plan, track and debug programs,
- be introduced to the idea of algorithms in computer programming and learn how algorithms differ from programs, and
- expand their understanding of robotic applications through projects using sensing and inputs.

### Unit 5: Versatile variables

Dive into the key computational concepts of variables, data and expressions while applying prior learnings from previous units. This unit's one lesson, which includes a total of four base activities and eight extension activities, rounds out students' exploration of the EdScratch environment. Earlier concepts are revisited and expanded on using the additional flexibility afforded by including variables and operators to manage data within their programs.

The learning objectives for this unit are that students will:

- be introduced to the computer science fundamentals of variables and data,
- explore how data and variables can be used with Edison using new blocks in the 'Data' and 'Operators' categories in EdScratch,
- further explore Edison's sensors and input capabilities using variables and operators to refine and modify behaviours in working programs, and
- apply computer science fundamentals such as tracing and debugging to work through projects using sensing and variables.

### Unit 6: Inventor's time!

Put all of your Edison and EdScratch knowledge into action! This unit includes one lesson with a total of two base activities and five extension activities with a major focus on projects. By designing and developing projects of their own using iterative cycles of planning, making and testing, students put the key computational thinking, problem-solving, programming, and physical computing concepts they have learned to work in this culmination unit.

The learning objectives for this unit are that students will:

- learn about the design-build-test cycle and strategies, such as decomposition and iterative testing, for physical computing problem-solving, and
- demonstrate their understanding of key computational thinking and computer science principals through open-ended projects.

## Unit 1: Get started

Prepare for your EdVenture! This initial unit introduces students first to the Edison robot and then to the EdScratch programming environment. Students gain familiarity with the Edison robot through a series of activities including programming the robots using barcodes. They then learn the basics of the EdScratch programming environment in preparation for the next unit.

### Learning objectives

Students will:

- develop a working familiarity with the Edison robot, including its key physical features (power switch, sensors and buttons)
- be able to program an Edison robot using barcodes
- know how to access EdScratch and how to download a program from EdScratch to Edison

**Key ideas:** hardware versus software, Edison's buttons and sensors, EdScratch terminology and basic features

### Lessons and activities in this unit

This unit includes two lessons with a total of four base activities and ten extension activities.

#### Lesson 1: Meet Edison

- [U1-1.1 Let's explore our Edison robots](#)
  - o [U1-1.1a Change it up: Bricks, blocks and Edison](#)
- [U1-1.2 Let's explore barcode programming](#)
  - o [U1-1.2a Change it up: Sumo wrestling](#)
  - o [U1-1.2b Change it up: Make your own barcode?](#)
  - o [U1-1.2c Change it up: TV remote control barcodes](#)
  - o [U1-1.2d Challenge up: Edison soccer](#)
  - o [U1-1.2e Challenge up: Build and control the EdTank](#)
  - o [U1-1.2f Challenge up: Build and control the EdDigger](#)
  - o [U1-1.2g Challenge up: Build and control the EdRoboClaw](#)

#### Lesson 2: Meet EdScratch

- [U1-2.1 Let's explore the EdScratch environment](#)
  - o [U1-2.1a Challenge up: Download another!](#)
  - o [U1-2.1b Change it up: Does EdScratch = Scratch?](#)
- [U1-2.2 Let's explore warning messages](#)

## Lesson 1: Meet Edison

This initial lesson serves as an introduction to the Edison robots. Developing an understanding of Edison’s parts and capabilities will help students use Edison effectively in all the activities in every unit.

In this lesson, students will ‘meet’ the Edison robot and learn about the robot’s key features, including the power switch, button locations and functions, sensor locations and functions, and the robot’s removable parts. Students will also learn about programming the robots using barcodes.

This lesson has a total of two base activities and eight extension activities:

- [U1-1.1 Let’s explore our Edison robots](#)
  - o [U1-1.1a Change it up: Bricks, blocks and Edison](#)
- [U1-1.2 Let’s explore barcode programming](#)
  - o [U1-1.2a Change it up: Sumo wrestling](#)
  - o [U1-1.2b Change it up: Make your own barcode?](#)
  - o [U1-1.2c Change it up: TV remote control barcodes](#)
  - o [U1-1.2d Challenge up: Edison soccer](#)
  - o [U1-1.2e Challenge up: Build and control the EdTank](#)
  - o [U1-1.2f Challenge up: Build and control the EdDigger](#)
  - o [U1-1.2g Challenge up: Build and control the EdRoboClaw](#)

### Activity U1-1.1 Let’s explore our Edison robots

<b>Activity size</b>	Small
<b>Delivery recommendations</b>	Strongly recommended if your students are new to Edison robots
<b>Resources needed</b>	Edison robots with batteries, EdComm cables, worksheet U1-1.1

#### Overview

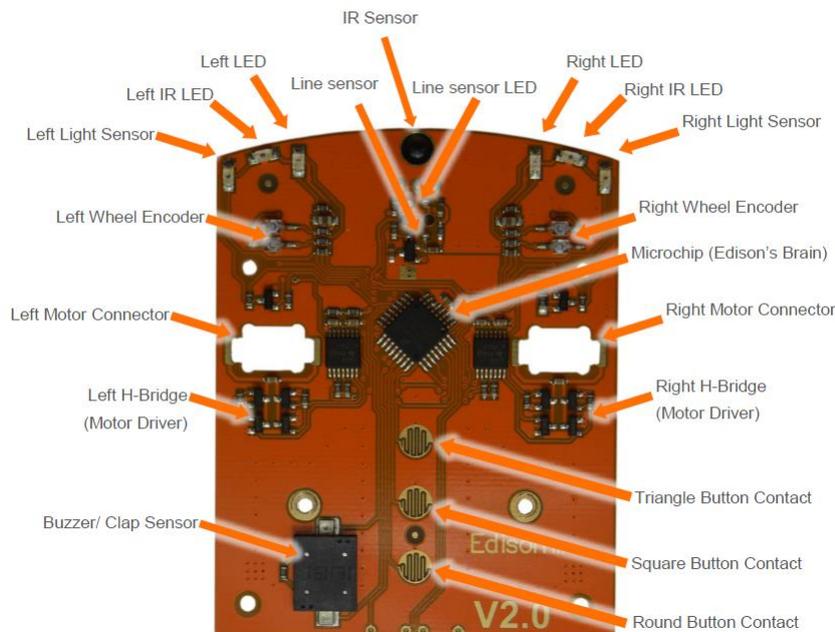
This lesson introduces students to the Edison robot, including the location of the robot’s buttons and sensors. Students familiarise themselves with their robot and its parts. Establishing a working understanding of the robot in this way will help set students up for success in future lessons.

#### Tips and tricks

- It may be helpful to have students keep this activity sheet for their review of Edison’s components in future lessons.
- Even though students don’t need to use the EdComm cables with programming devices such as computers or tablets in this lesson, you may wish to show them how to connect the cables to the headphone jacks of the programming devices they will be using in future lessons at this time.
- Some of the robotics projects in later lessons will require students to remove the wheels and skid from Edison. Understanding that this is possible can help

open up students' imaginations to what the robot might be able to become. It is recommended that students practice at least removing the wheels. If you want students to practice removing the skid as well, consider having them do so over something, like a plastic tray, so that if they drop the skid, it is easier to locate.

- If your students are interested in learning more about what is on Edison's motherboard, you can download the full-size motherboard layout guide at <https://meet Edison.com/content/Edison-Motherboard-layout-V2.pdf>



### Answer key

Question	Type	Sample answer	Marking notes
1	SE	<i>When I turned Edison on, the red LEDs came on and started flashing. Edison also made a chirping noise one time.</i>	There are two key things to look for in students' answers: - the robot's red LED lights turn on and begin to flash, and - the robot makes a noise.

### Activity U1-1.1a Change it up: Bricks, blocks and Edison

<b>Activity size</b>	Small
<b>Delivery recommendations</b>	If students have never used LEGO bricks with Edison before, this is a fun activity to help get them excited
<b>Resources needed</b>	Edison robots, EdCreate kits or any other LEGO brick compatible building system, worksheet U1-1.1a

### Overview

Edison is so much more than just a 'car' style programmable robot. Using Edison as the building block base in robotics projects is one of the best ways to see the power of programming and robotics come to life. This fun STEAM (science, technology,

engineering, arts, and maths) activity is designed to be a low-risk task that gives students a chance to experiment with ‘engineering’ using Edison. The free-form building task encourages kids to experiment, tap into their creativity, and begin to explore some of the tactile engineering components of physical computing.

### Tips and tricks

- Edison works with any LEGO brick compatible building system, including the EdCreate robot creator’s kit. Bricks can be attached to the top and bottom of the Edison robot and pegs can be attached to the robot’s sides. When Edison’s wheels are removed, cross axles can be used in the powered sockets.
- You may want to add mini challenges to this activity such as ‘build as high as you can’ or ‘make sure Edison’s buttons are still accessible’.
- You may choose to get students to include why they created the design they did in their answer.

### Answer key

Question	Type	Sample answer	Marking notes
1	SE	<i>My Edison had a big tower on the top of the robot.</i>	This is a free-form activity.

### Activity U1-1.2 Let’s explore barcode programming

<b>Activity size</b>	Medium
<b>Delivery recommendations</b>	Strongly recommended if your students are new to Edison robots
<b>Resources needed</b>	- Edison robots with batteries, worksheet U1-1.2, activity sheet U1-1, torches (flashlights), objects for making obstacles - <i>Optional:</i> EdMats, activity sheet U1-2, supplies for making your own lines/borders

### Overview

Students explore some of Edison’s sensors in action using pre-set programs. Seeing what the robots are capable of without needing to do any coding is a great way to deliver early success and get students motivated for upcoming activities.

This activity includes five programs stored in Edison’s memory, which are accessed using special barcodes. Details on each barcode-activated program follow.

### Program 1: Clap controlled driving

*What’s happening:* Edison’s sound sensor is responding to loud sounds, such as claps. Edison will turn right when it detects one clap or drive forward about 30cm if it detects two claps.

### Tips and tricks

- The robots may struggle to detect and differentiate sounds when there is a high level of background noise. Having students tap a finger near the sound sensor on the top of their Edison robot will have the same effect as clapping.

### Program 2: Avoid obstacles

*What's happening:* The avoid obstacles program uses the Edison robot's infrared (IR) light LEDs and IR sensor to detect objects directly in front of the robot. Once the pre-set program is activated, the Edison robot will drive forward, turning as needed to avoid obstacles it encounters.

### Tips and tricks

- Obstacles need to be opaque but not too dark (e.g. not black) and at least as tall as Edison for the robot to detect them.

### Program 3: Follow a torch

*What's happening:* In this program, Edison's two light sensors at the front-left and front-right of the robot take light readings of the amount of visible light each one detects. The readings are then compared to each other. If the level of light detected by the right sensor is higher than the level of the left sensor, then Edison's left motor is driven forward, turning Edison right, towards the light. This movement will continue until the level of light detected by the left sensor has the greater value. At that time, the left motor will stop, and the right motor will be driven forward, driving Edison, once again towards the light.

### Tips and tricks

- You will need a torch (flashlight) and a flat surface located away from any other sources of bright light, such as sunlight or overhead fluorescents to run this program.
- Once Edison 'sees' the bright source of light, the robot will drive towards it. By moving the torch, you can control where Edison drives.

### Program 4: Follow a line

*What's happening:* The line tracking program uses the Edison robot's reflected light sensor to detect differences between dark and light surfaces beneath the robot. Once the pre-set program is activated, the Edison robot will drive until it finds a dark coloured line, then follow that line. In this program, Edison's line tracking sensor shines light from its red LED on to the surface beneath the robot. The sensor then measures the amount of light that is reflected back to the robot. White surfaces reflect a lot of light, giving a high light level reading while black surfaces reflect very little, giving a low light level reading. Edison adjusts direction according to these light

level readings. When Edison is off the line, it turns right to get on the line. However, when Edison is on the line, it turns left to get off the line. This functionality is why Edison ‘waddles’ back and forth at the edge of the line.

### Tips and tricks

- The difference between the dark and light surfaces needs to be easily understood by the robot. Either use the thick, dark lines on the activity sheet, those on an EdMat (free download available at <https://meetedison.com/edmat/>) or make a track for Edison to follow by drawing a dark (e.g. black) line approximately 1.5cm (0.6 inches) wide on a white background.
- Make sure students start by placing Edison next to the black line, but not on top of it, so that the line tracking sensor starts on a white surface. Always start the robot on a white surface when using the line tracking sensor.
- If Edison’s wheels catch the edge of the activity sheet’s paper, this can throw the robot off slightly. You can fix this by taping the activity sheet down or replicating the activity sheet’s pattern on a larger piece of paper.

### Program 5: Bounce in borders

*What’s happening:* Like the line tracking program, the bounce in borders program uses the Edison robot’s reflected light sensor to detect differences between dark and light surfaces beneath the robot. Once the pre-set program is activated, the Edison robot will drive until it encounters a dark coloured line. It will then turn around and drive in a different direction without crossing that line.

### Tips and tricks

- The difference between the dark and light surfaces needs to be easily understood by the robot. Either use the thick, dark lines on the activity sheet, those on an EdMat (free download available at <https://meetedison.com/edmat/>) or make a track for Edison to follow by drawing a dark (e.g. black) line approximately 1.5cm (0.6 inches) wide on a white background.
- Make sure students start by placing Edison inside the black line so that the line tracking sensor is on a white surface. The robot can start near the black line but not on top of it. Always start the robot on a white surface when using the line tracking sensor.

### Activity U1-1.2a Change it up: Sumo wrestling

<b>Activity size</b>	Small
<b>Delivery recommendations</b>	Complete activity U1-1.2 prior to this activity
<b>Resources needed</b>	<ul style="list-style-type: none"> <li>- At least 2 Edison robots with batteries, worksheet U1-1.2a, activity sheet U1-2</li> <li>- <i>Optional:</i> EdMats, supplies for making your own sumo ring</li> </ul>

#### Overview

This activity uses one of Edison’s special barcodes to activate the sumo wrestling program. The program is a favourite with students and a good way to begin to create an atmosphere of collaboration when it comes to using the robots.

The sumo wrestling program combines aspects of two of Edison’s other programs – bounce in borders and obstacle detection. For this program to work, you need to place at least two Edison robots on a white-coloured surface with a black-coloured outline. The obstacle detection part of the program helps each Edison robot find the other robots while the line detection part of the program helps Edison detect and react to the dark-coloured border.

#### Tips and tricks

- You will need at least two Edison robots for this activity. Both robots need to scan the sumo wrestling barcode to activate the program.
- Make sure the sumo ring is large enough for all the robots to drive around inside. If the ring is too large, however, it will take longer for the robots to find each other.
- This program uses Edison’s line tracking sensor. If students are struggling with a line tracking program, check that the line they are using is a very dark colour, such as black, and approximately 1.5cm (0.6 inches) wide. Also, make sure that the background is white or another highly reflective colour. The EdMat (free download available at <https://meetiedison.com/edmat/>) works very well for this activity.
- Make sure students start by placing Edison inside the black line so that the line tracking sensor is on a white surface. The robot can start near the black line but not on top of it. Always start the robot on a white surface when using the line tracking sensor.

### Activity U1-1.2b Change it up: Make your own barcode?

<b>Activity size</b>	Small
<b>Delivery recommendations</b>	- Recommended if students are new to programming - Complete activity U1-1.2 prior to this activity
<b>Resources needed</b>	Worksheet U1-1.2b

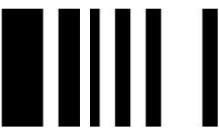
#### Overview

This offline activity reinforces how barcode programming with Edison works. Students learn the connection between how barcodes function (i.e. by activating stored programs in the robot's memory) and why this means they cannot make their own barcodes. Students then embark on an exercise in imagination – creating their own pretend program and barcode for Edison.

#### Tips and tricks

- The worksheet starts with several questions about how barcodes work. You may want to discuss these as a group or have students think about them on their own, coming up with their answers before handing out the worksheets. Doing so is a good exercise in extrapolation.
- Having students create a 'fake' program is a good, low-risk task to get them to start thinking about what programs actually are. This activity gets students to start exercising computational thinking before being introduced to the concept explicitly.

#### Answer key

Question	Type	Sample answer	Marking notes
1	SE	<p><i>My program would have Edison put away my clean laundry. If Edison ran my program, the robot would drive to the laundry basket and pick up all the clean clothes. It would then drive to my closet and put all the clothes away.</i></p> 	<p>- If students have developed an understanding of the robot's capabilities, they may design a program which uses the robot's actual functionality. Otherwise, they may design a program that is quite fanciful, like in the example answer. Either is fine.</p> <p>- You may want to see if students' programs show a preliminary understanding of basic computational thinking, such as sequence, in their programs.</p> <p>- As the barcodes are only pretend, they do not need to look like real barcodes.</p>

### Activity U1-1.2c Change it up: TV remote control barcodes

<b>Activity size</b>	Small
<b>Delivery recommendations</b>	- Strongly recommend if students will be using remote controls with Edison in later projects - Complete activity U1-1.2 prior to this activity
<b>Resources needed</b>	Edison robots with batteries, worksheet U1-1.2c, activity sheet U1-3, TV or DVD remote controls)

#### Overview

This activity introduces using Edison robots with TV or DVD remote controls. This activity is the first in the lesson set using Edison’s IR messaging and can serve as a chance to explore that functionality.

Using remote controls with Edison gives students a chance to actively control their robots without needing to code. This activity can excite students to learn more ways they can program the robots.

#### Tips and tricks

- The TV/DVD remote control barcodes affect Edison while the robot is in standby mode – make sure students do NOT press the play (triangle) button after using the remote-control barcodes. Instead, press the paired button on the remote control to activate that action.
- Edison is compatible with around 75% of TV and DVD remote controls. If one of your remotes doesn’t happen to work with Edison, try a different remote, preferably a different brand. Alternatively, you can purchase an inexpensive ‘universal remote’ and set it to be a Sony DVD remote control, which works well with the robots.
- If multiple students are running programs using the remote-control codes in close physical proximity to each other, they may experience ‘cross-talk’ where one robot receives and reacts to the remote-control signals sent from a different group. Try spacing students out. You can also encourage them to select different buttons on their remotes to pair with the robots to differentiate each group from one another.

### Answer key

Question	Type	Sample answer		Marking notes
1	SE	<b>Program</b>	<b>Remote control button</b>	You may want to check that students have applied logic to their button selection to ensure they understand the robot's functionality.
		Drive forward	<i>Menu navigation wheel 'up'</i>	
		Drive backwards	<i>Menu navigation wheel 'down'</i>	
		Spin left	<i>Channel down</i>	
		Spin right	<i>Channel up</i>	
		Turn left	<i>Menu navigation wheel 'left'</i>	
		Turn right	<i>Menu navigation wheel 'right'</i>	
		Play beep	<i>Volume down</i>	
		Play tune	<i>Volume up</i>	

### Activity U1-1.2d Challenge up: Edison soccer

<b>Activity size</b>	Medium
<b>Delivery recommendations</b>	- Complete activity U1-1.2c prior to this activity
<b>Resources needed</b>	- Minimum of 2 Edison robots with batteries, worksheet U1-1.2d, activity sheet U1-3, minimum of 2 TV/DVD remote controls, materials for creating the pitch, goals and ball - <i>Optional:</i> EdCreate kits/LEGO bricks

#### Overview

This activity gives students a specific task to complete using the TV remote control barcodes with their Edison robots.

Students need to work together to be successful. The worksheet specifically instructs students to compete against each other, then hints that they will first need to collaborate to avoid cross-talk with the remotes. Although not explicitly stated, students also need to work together to agree on the creation of both the playing surface and rules for the soccer game. Encouraging, and pointing out this type of natural collaboration as it occurs, is a good opportunity to show problem-solving in action.

Creating the pitch, goals and ball are a good chance for low-risk experimentation in a 'real world' scenario.

#### Tips and tricks

- The TV/DVD remote control barcodes affect Edison while the robot is in standby mode – make sure students do NOT press the play (triangle) button

after using the remote-control barcodes. Instead, press the paired button on the remote control to activate that action.

- Edison is compatible with around 75% of TV and DVD remote controls. If one of your remotes doesn't happen to work with Edison, try a different remote, preferably a different brand. Alternatively, you can purchase an inexpensive 'universal remote' and set it to be a Sony DVD remote control, which works well with the robots.
- With multiple students running programs using the remote-control codes in close physical proximity to each other, they may experience 'cross-talk' where one robot receives and reacts to a remote-control signal intended for a different robot. Remind students to choose different buttons on their remotes to differentiate their commands.
- The size of the pitch and ball will affect student success in this activity. Encourage students to experiment with setting up a good working solution before having an 'official' match.
- An alternative to this activity is to allow students to build on their Edison robots using EdCreate or alternative compatible LEGO bricks to make contraptions to help control the ball. Think of this version of the activity as modified hockey or polo rather than soccer!

### Activity U1-1.2e Challenge up: Build and control the EdTank

<b>Activity size</b>	Medium (basic EdTank) Large (complete EdTank)
<b>Delivery recommendations</b>	- Complete activity U1-1.2c prior to this activity - While activities U1-1.2e, U1-1.2f and U1-1.2g are all different builds, the learning within the three is comparable. You may choose only to have students complete one of these three activities.
<b>Resources needed</b>	1 or 2 Edison robots with batteries per EdTank (depending on build), 1 TV/DVD remote per EdTank, worksheet U1-1.2e, 1 EdCreate kit per EdTank, EdTank build instructions set (available at <a href="https://meet Edison.com/content/EdCreate/EdBuild-EdTank-instructions.pdf">https://meet Edison.com/content/EdCreate/EdBuild-EdTank-instructions.pdf</a> )

#### Overview

Using Edison to create programmable robotic builds is one of the most exciting things students can do with the robots. The EdBuild projects using the pre-set instructions and EdCreate kits are an excellent way to give kids exposure to interactive engineering. Using the EdCreate kits and step-by-step instructions for creating an EdBuild helps ease students into building with Edison without needing to design the creation themselves.

### About the EdTank:

The EdTank is actually two builds in one: the basic EdTank and the complete EdTank with rubber band cannon.

The basic EdTank uses one Edison robot which can be programmed to drive forwards, backwards and turn right or left using barcodes and a TV or DVD remote control. The complete EdTank includes a second Edison robot which controls (fires) the rubber band cannon.

### Tips and tricks

- Students may find it easiest to first lay out all of the EdCreate pieces onto their work surface and organise the parts into groups of the same piece type and colour. This can help students identify and use the correct pieces as they work through the build.
- You will need to reset the cannon and reload a rubber band each time you fire the cannon. This should be done manually to ensure the new band is loaded properly and that the firing ‘pin’ is pushed completely back into the starting location.
- For best results, use the orange bands that come with your EdCreate kit as the ‘ammo’ in the cannon.
- Just like many real tanks, the EdTank’s design means it will be slow to turn left or right. When the EdCreate tracks are brand new, they may be extra grippy, which will make the EdTank turn even slower. You can reduce the grip by removing the tracks from the EdTank and lightly dusting them with talcum powder. Be sure to knock any excess powder off the tracks before putting them back onto the EdTank.

### Answer key

Question	Type	Sample answer	Marking notes
1	SE	<i>The EdTank didn't turn as well as the Edison robot does when it's in 'car' mode. I think this is because the tank has a wider turning radius and also it is heavier than the normal Edison.</i>	<ul style="list-style-type: none"> <li>- Any response that applies logic to any difference in driving is acceptable.</li> <li>- It is possible that the EdTank will turn as quick as the Edison robot depending on the driving surface and condition of the tracks.</li> </ul>

### Activity U1-1.2f Challenge up: Build and control the EdDigger

<b>Activity size</b>	Large
<b>Delivery recommendations</b>	- Complete activity U1-1.2c prior to this activity - While activities U1-1.2e, U1-1.2f and U1-1.2g are all different builds, the learning within the three is comparable. You may choose only to have students complete one of these three activities.
<b>Resources needed</b>	2 Edison robots with batteries per EdDigger, 1 TV/DVD remote per EdDigger, worksheet U1-1.2f, 1 EdCreate kit per EdDigger, EdDigger build instructions set (available at <a href="https://meetedison.com/content/EdCreate/EdBuild-EdDigger-instructions.pdf">https://meetedison.com/content/EdCreate/EdBuild-EdDigger-instructions.pdf</a> )

#### Overview

Using Edison to create programmable robotic builds is one of the most exciting things students can do with the robots. The EdBuild projects using the pre-set instructions and EdCreate kits are an excellent way to give kids exposure to interactive engineering. Using the EdCreate kits and step-by-step instructions for creating an EdBuild helps ease students into building with Edison without needing to design the creation themselves.

#### About the EdDigger:

The EdDigger is a remote-controlled excavator, or digger, with a scoop that you can drive around (forwards, backwards and spin right or left). The digger scoop of the EdDigger can lift and lower and can carry small objects, such as other parts from the EdCreate kit.

#### Tips and tricks

- Students may find it easiest to first lay out all of the EdCreate pieces onto their work surface and organise the parts into groups of the same piece type and colour. This can help students identify and use the correct pieces as they work through the build.
- The top robot connects to the bottom robot on the third row of studs from the front of the bottom robot. Thus, the top robot overhangs off the back of the bottom robot by approximately 2 cm.

#### Answer key

Question	Type	Sample answer	Marking notes
1	SE	<i>To scoop up objects I needed to have the EdDigger push the objects into something, like a wall, so that the objects would be pushed up into the bucket. Then I could raise the scoop. To drop objects out of the bucket, I first lowered the bucket with Edison stopped, then drove backwards, so the objects fell out.</i>	Any response that explains how they managed the scoop is acceptable.

### Activity U1-1.2g Challenge up: Build and control the EdRoboClaw

<b>Activity size</b>	Large
<b>Delivery recommendations</b>	- Complete activity U1-1.2c prior to this activity - While activities U1-1.2e, U1-1.2f and U1-1.2g are all different builds, the learning within the three is comparable. You may choose only to have students complete one of these three activities.
<b>Resources needed</b>	2 Edison robots with batteries per EdRoboClaw, 1 TV/DVD remote per EdRoboClaw, worksheet U1-1.2g, 1 EdCreate kit per EdRoboClaw, EdRoboClaw build instructions set (available at <a href="https://meetedison.com/content/EdCreate/EdBuild-EdRoboClaw-instructions.pdf">https://meetedison.com/content/EdCreate/EdBuild-EdRoboClaw-instructions.pdf</a> )

#### Overview

Using Edison to create programmable robotic builds is one of the most exciting things students can do with the robots. The EdBuild projects using the pre-set instructions and EdCreate kits are an excellent way to give kids exposure to interactive engineering. Using the EdCreate kits and step-by-step instructions for creating an EdBuild helps ease students into building with Edison without needing to design the creation themselves.

#### About the EdRoboClaw:

The EdRoboClaw is a remote-controlled articulated robotic arm, which can be programmed using the TV/DVD barcodes and controlled with a standard TV or DVD remote control. You can drive the EdRoboClaw forwards, backwards and spin it right or left. You can also open and close the claw to pick up and carry an object, such as one of the EdCreate beams.

#### Tips and tricks

- Students may find it easiest to first lay out all of the EdCreate pieces onto their work surface and organise the parts into groups of the same piece type and colour. This can help students identify and use the correct pieces as they work through the build.
- The top robot connects to the bottom robot on the second row of studs from the front of the bottom robot. Thus, the top robot overhangs off the back of the bottom robot by approximately 1 cm.
- The claw is composed of 3 'fingers' – two parallel fingers which are stationary (made from grey beams) and the frontmost finger which moves. The row of gears in the articulated arm controls this forward finger, including its positioning relative to the stationary fingers. The alignment of the forward most two gears can affect how the moving finger sits relative to the stationary fingers when the claw is fully open. When the claw is fully open, and the EdRoboClaw is sitting on a flat surface (such as a table or desk), the moving finger should be high enough that one of the EdCreate grey beams can slide under it between the finger and the table. If the moving finger is not this high, try gently separating the front of the arm and rotating the frontmost gear by

one or two teeth clockwise independently of the next gear. You will be able to see this move the front finger. Reconnect the gears and the arm.

- The EdRoboClaw can pick up objects with some flat surfaces. Students may find that they are not as able to pick up and carry objects which are round, such as a pen. Try using the 7-hole long grey beam from the EdCreate kit.

Answer key

Question	Type	Sample answer	Marking notes
1	SE	<i>I couldn't carry my pen, but I could carry my granola bar and some of the EdCreate parts. I think that the best objects are flat in some places so they don't slip out of the claw.</i>	Any response that applies logic to the student's findings around objects is acceptable.

## Lesson 2: Meet EdScratch

The second part of the first unit introduces students to the EdScratch programming environment. Students learn how to access the EdScratch online application, explore the environment's main areas (including the 'bug box' and warning messages) and try downloading a test program from EdScratch to Edison.

This lesson has a total of two base activities and two extension activities:

- [U1-2.1 Let's explore the EdScratch environment](#)
  - o [U1-2.1a Challenge up: Download another!](#)
  - o [U1-2.1b Change it up: Does EdScratch = Scratch?](#)
- [U1-2.2 Let's explore warning messages](#)

### Activity U1-2.1 Let's explore the EdScratch environment

<b>Activity size</b>	Large
<b>Delivery recommendations</b>	Strongly recommended if your students are new to EdScratch
<b>Resources needed</b>	Basic supplies set, worksheet U1-2.1

#### Overview

While drag-and-drop coding in a Scratch-like environment may be familiar to some students, understanding the EdScratch environment's particular layout and functionality is critical for success using the programming language. This activity introduces the online application and basic layout of EdScratch to students, including the names of the main areas inside the programming environment. Students then practice downloading a program from EdScratch to their Edison robot.

Ensuring students can navigate inside EdScratch and understand how to download EdScratch programs to the robot will support their independent use of the environment in future lessons.

#### Tips and tricks

- Some devices, especially tablets, automatically lower the volume when they detect that an audio device, such as headphones, has been connected to the audio jack. The programming device may read the EdComm cable as 'headphones'. Make sure the volume on the computer or tablet is still turned all the way up after the EdComm cable is plugged in.
- While a program is downloading to Edison, Edison makes a whirring sound, similar to an old dial-up modem. Once the program downloads successfully, Edison will make a chirping beep. If the program fails while downloading, Edison will make a 'fail sound'. You can hear what the success and fail noises sound like at <https://meetiedison.com/edison-robot-support/trouble-shooting>

- Make sure students do not unplug the EdComm cable until after they hear the ‘success’ sound.
- When many students are downloading programs at the same time, you may experience slower internet speeds, causing the program to take longer to create the ‘Program Edison’ button in the pop-up box and for the program to download to Edison. Remind students to listen for the success sound before unplugging the EdComm cable to ensure they wait until the program fully downloads.

### Answer key

Question	Type	Answer/Sample answer	Marking notes
1	EA	<i>In the menu bar.</i>	
2	EA	<i>There are zero messages in the bug box.</i>	The correct <b>Test_program</b> does not have any warning messages, so nothing should appear in the bug box.
3	SE	<i>First, the robot beeps once, turns on the left LED and spins left almost all the way around. Then the robot beeps once, turns on the right LED and turns off the left LED and spins back to the start position. It repeats the whole thing five times.</i>	The main objective is that students download and run the correct program successfully. How they describe the program is of secondary importance.

### Activity U1-2.1a Challenge up: Download another!

<b>Activity size</b>	Small
<b>Delivery recommendations</b>	Complete activity U1-1.2c prior to this activity
<b>Resources needed</b>	- Basic supplies set, worksheet U1-2.1a - Additional supplies will be needed depending on the programs selected.

#### Overview

This activity encourages students to explore EdScratch a bit further independently. By selecting a different demo program than the Test\_program (which was used in the base activity), students practice the key skills required to download programs to their robots successfully.

The questions on worksheet U1-2.1a encourage students to begin to tie the on-screen code they see in the demo program of their choice to their robot’s real-world

actions. Understanding this connection is fundamental to being able to use computational thinking practices effectively when developing their own programs.

### Tips and tricks

- You can limit the supplies required for this activity to just the basic supply set and the worksheet by having students select the Moving\_with\_music or Clap\_controlled\_driving demo programs.
- If students are struggling with this activity, have them review worksheet U1-2.1.

### Answer key

Question	Type	Sample answer	Marking notes
1	SE	<i>Moving_with_music</i>	The program name should be one from the demos list.
2	SE	<i>I thought the robot would move to whatever music I put on. Instead, the robot played a song and moved to that song.</i>	Anything that answers the question is acceptable.
3	SE	<i>The robot played the notes that the program has in it. The robot also moved the same way as what the blocks said. It did the actions in the same order that the blocks are in, top to bottom.</i>	Ideally, the answer should demonstrate an understanding that the blocks on screen are related to the robot's actions. Depending on the program students' select, this connection may be less obvious.

### Activity U1-2.1b Change it up: Does EdScratch = Scratch?

<b>Activity size</b>	Small
<b>Delivery recommendations</b>	Recommended if students have used, or will be using, both Scratch and EdScratch
<b>Resources needed</b>	Internet-linked computers or tablets, worksheet U1-2.1b

### Overview

Many students will immediately 'recognise' EdScratch – especially if they have used MIT's Scratch programming language before. The EdScratch app was developed using the Scratch Blocks code base developed by MIT (building on the Blockly code base developed by Google). That's why there is such a strong similarity between Scratch and EdScratch. It's important for students to understand that the two are separate languages, however. This activity asks students to explore this for themselves.

### Tips and tricks

- This activity may not be as effective for students who have not used Scratch. You may choose to have it as an optional activity for any student curious about the relationship between the two languages.
- This activity is well suited to group or whole-class work, with students contributing to a list of similarities and differences.
- Even if your students are not using Scratch, this activity can still make for an interesting exploration of the fact that there are many different programming languages, each with its own capabilities and limitations.

### Answer key

Question	Type	Sample answer	Marking notes
1	SE	<p>1 – <i>The colours and the shapes of the blocks look the same in both languages.</i></p> <p>2 – <i>Both Scratch and EdScratch have a block pallet, a menu bar and an area for making programs.</i></p> <p>3 – <i>Some of the block categories, like 'sound' and 'control', are the same in both Scratch and EdScratch.</i></p>	Any three things that are similar or the same in the two languages are acceptable.
1	SE	<p>1 – <i>Scratch uses Sprites, like the cat, but EdScratch doesn't have Sprites.</i></p> <p>2 – <i>EdScratch has a bug box, but Scratch doesn't have one.</i></p> <p>3 – <i>The names of the block types are different in EdScratch and Scratch.</i></p>	Any three things that are different or appear different in the two languages are acceptable.

### Activity U1-2.2 Let's explore warning messages

<b>Activity size</b>	Small
<b>Delivery recommendations</b>	<ul style="list-style-type: none"> <li>- Strongly recommended if your students are new to EdScratch</li> <li>- Strongly recommended if your students are new to the idea of debugging in programming</li> </ul>
<b>Resources needed</b>	Basic supplies set, worksheet U1-2.2

### Overview

The bug box in EdScratch provides warning messages to users. These messages can flag different problems or potential issues. In this activity, students learn about the bug box and the two types of warning messages they may encounter in the bug box. The idea of bugs in programming is not explicitly introduced in this activity, however, nor is the practice of debugging.

The critical learning objective of this activity is that warning messages in EdScratch are not 'failures' or inherently bad things. Quite the opposite! The bug box and its

warning messages are wonderful tools to help discover and fix issues – critical problem solving and coding skills. This lesson activity aims to help students develop a good attitude towards the bug box, viewing it as a tool they can use, while laying the framework for debugging as a skill to be developed in future lessons.

### Tips and tricks

- Some students may fix the bugs before answering the questions. Reloading the demo program will bring back the original program and its bugs. Using the keyboard shortcut 'undo' (PC: Ctrl + z) (Mac: command + z) will also undo changes one by one.
- You can find a complete guide to the warning messages in EdScratch at <https://meetedison.com/robot-programming-software/edscratch/>. This guide includes all warning messages, what they mean and examples of when you may encounter them.

### Answer key

Question	Type	Answer/Sample answer	Marking notes
1	SE	<i>An error message popped up and said that the program cannot download because there is a red warning message.</i>	While a red message is displaying, the program will not download. Student answers must demonstrate an understanding of this.
2	SE	<i>I took the orange 'wait 1 sec' block out of the purple blocks.</i>	Removing the non-musical note block from the stack will fix the issue. Other fixes which result in the same outcome (such as deleting all blocks) are technically correct but are not ideal solutions.
3	EA	<i>The two blue 'drive' blocks ('forwards for 10 cm at speed 2' and 'backwards for 17 cm at speed 5') will not be programmed into Edison.</i>	This answer is based off the program without any changes applied.

## Unit 2: Move it!

Focus in on the key computational concept of sequence in this unit. Students explore Edison's abilities to move using motors, plus use the robot's LEDs and sound sensor/buzzer through a range of activities. Computer programming fundamentals including inputs, outputs, bugs and debugging are introduced. Students begin to develop their familiarity with programming Edison in EdScratch and with using Edison as the base for creative robotic builds.

### Learning objectives

Students will:

- be introduced to the idea of computational thinking and begin to use computational frameworks when approaching tasks
- become familiar with the concept of sequence
- learn what inputs, outputs and input parameters are
- explore EdScratch's core output blocks with Edison, including drive, LEDs and sound
- be able to create and modify sequential programs for Edison in EdScratch using basic output blocks and the input parameters of those blocks
- begin to explore robotics applications to real-world situations through projects

**Key ideas:** computational thinking, sequence, programming (coding), inputs and outputs, input parameters, bugs (errors), debugging, syntax

### Lessons and activities in this unit

This unit includes two lessons with a total of eight base activities and 13 extension activities.

#### Lesson 1: Sequence

- [U2-1.1 Let's explore how computers 'think'](#)
  - [U2-1.1a Change it up: Make a PBJ sandwich](#)
  - [U2-1.1b Change it up: Human robots](#)
- [U2-1.2 Let's explore going step-by-step in EdScratch](#)
- [U2-1.3 Let's explore driving Edison](#)
  - [U2-1.3a Challenge up: Maze madness](#)
  - [U2-1.3b Challenge up: Self-walking pet](#)

#### Lesson 2: Inputs and outputs

- [U2-2.1 Let's explore Edison's outputs](#)
  - [U2-2.1a Challenge up: Drive the maze safely](#)
- [U2-2.2 Let's explore input parameters](#)
  - [U2-2.2a Change it up: Teach Edison to count to 9](#)
  - [U2-2.2b Challenge up: Teach Edison to count to 9 out loud](#)
- [U2-2.3 Let's explore Edison's musical talents](#)
  - [U2-2.3a Change it up: Play a song in a round](#)

- [U2-2.3b Challenge up: You are the conductor](#)
- [U2-2.4 Let's explore bugs and debugging](#)
- [U2-2.5 Let's explore Edison's motors](#)
  - [U2-2.5a Challenge up: Spinning garden](#)
  - [U2-2.5b Challenge up: Spinning solar system](#)
  - [U2-2.5c Challenge up: Cartographer and navigator](#)
  - [U2-2.5d Challenge up: Writer and director](#)

## Lesson 1: Sequence

The main focus of this lesson is on the foundational computational concept of sequence. Students first learn what ‘computational thinking’ is and why it is a helpful framework for problem-solving and working with computers. Sequence, one of the most critical building blocks of computational thinking, is then introduced. Students begin to apply computational frameworks, especially sequential thinking, when approaching tasks. The activities in this lesson offer students ample opportunities to practice applying sequential logic, determining the exacting step-by-step instructions that they need to provide in order to complete objectives. Students also begin to create their own programs for Edison using the ‘Drive’ category of blocks in EdScratch.

This lesson has a total of three base activities and four extension activities:

- [U2-1.1 Let’s explore how computers ‘think’](#)
  - o [U2-1.1a Change it up: Make a PBJ sandwich](#)
  - o [U2-1.1b Change it up: Human robots](#)
- [U2-1.2 Let’s explore going step-by-step in EdScratch](#)
- [U2-1.3 Let’s explore driving Edison](#)
  - o [U2-1.3a Challenge up: Maze madness](#)
  - o [U2-1.3b Challenge up: Self-walking pet](#)

### Activity U2-1.1 Let’s explore how computers ‘think’

<b>Activity size</b>	Medium
<b>Delivery recommendations</b>	Consider delivering activity U2-1.1a prior to using this activity
<b>Resources needed</b>	Worksheet U2-1.1, activity sheet U2-1

#### Overview

In this offline activity, students are introduced to the ideas of computational thinking and sequence. Students do not need to fully understand what computational thinking entails at this point, but it is valuable to use the term explicitly. Using computational thinking makes coding much easier, but it is a bit different from how we normally approach and think about things in our lives. That’s why introducing ‘computational thinking’ as a way to think about things is a useful notion. Getting kids to ‘put on their computational thinking hats’ can help them approach problems differently, finding solutions they may otherwise overlook.

This activity has students practice following and giving good sequential instructions by being explicit about each action and going step-by-step, one action at a time. Developing a sequential approach to problem-solving is crucial for good computational thinking and success when programming in EdScratch.

### Tips and tricks

- Extension activities U2-1.1a and U2-1.1b are both designed to introduce sequence in a fun, engaging way. Both extension activity options are best done as a group or whole class as these activities generally need an instructor to run effectively. You may choose to run one or both of the extension activities before formally introducing the key concepts using this activity or use these activities to reinforce the concepts after this activity. For example, you could run activity U2-1.1a as a class, then have students work on activity U2-1.1 individually, then do U2-1.1b as a small group reinforcement activity.

### Answer key

Question	Type	Answer/Sample answer	Marking notes
1	EA	<i>The car.</i>	
2	EA	<i>The car.</i>	
3	EA	<i>The ice cream cone.</i>	
4	SE	<i>Move forwards 3 squares. Turn left. Move forwards 2 squares.</i>	Any answer which meets the question's criteria is acceptable.
5	SE	<i>Move forwards 2 squares. Turn left. Move forwards 2 squares. Turn right. Move forwards 1 square.</i>	Any answer which meets all of the question's criteria is acceptable.
6	SE	<i>Turn right. Move backwards 1 square. Turn right. Move backwards 1 square. Turn right. Move backwards 1 square.</i>	Any answer which meets all of the question's criteria is acceptable.

### Activity U2-1.1a Change it up: Make a PBJ sandwich

<b>Activity size</b>	Medium
<b>Delivery recommendations</b>	Consider delivering this activity prior to activity U2-1.1
<b>Resources needed</b>	Worksheet U2-1.1a, supplies for making the sandwich or alternative food
<b>NB</b>	This activity is best done as a group or whole class as it will likely need an instructor to run effectively.

### Overview

This two-part offline activity is designed to be a fun way to have things go wrong if and when instructions are not exactly right. Understanding that computers are super literal and require instructions that are both exacting and sequential is crucial for

success in coding. Helping build up students' understanding of this concept will help them better problem-solve as they create programs of their own.

### How to run task 1

The first part of this activity asks students to write out instructions to make a sandwich. This can be done individually or in pairs. The worksheet gives a few hints that this task may be more complicated than it first appears, but there's no need to belabour the point. Having some less-than-perfect instructions will make the second part of the activity more fun and help drive home the key message of the activity to all the students.

### How to run task 2

If you are using this activity as a whole class (recommended), this task should be done with the whole class watching. Select one of the student-written instructions sets either at random or cherry-pick one you can see has issues. Have a student volunteer read out each step one at a time. Follow each step EXACTLY as it is written. For example, if the instruction says 'put peanut butter on the bread' but doesn't say to open the jar – don't open the jar. Just put the whole jar onto the bread.

Early exposure to what can go wrong if our instructions aren't exacting enough for a computer (or robot) in this for-fun environment helps kids build up an understanding of the concept removed from the act of coding. This can help kids learn to consider sequential thinking and carefully analyse what is happening, enabling them to problem-solve issues they face when coding more effectively.

### Tips and tricks

- This activity is best done as a group or whole class as it generally needs an instructor to run effectively, especially for task 2.
- You can see an example of this type of activity being demonstrated at <https://www.youtube.com/watch?v=RjHzD2sfWcQ>
- You may choose to run this activity and/or extension activity U2-1.1b to kickstart your exploration of sequence before examining the key concept in detail using activity U2-1.1. For example, you could run activity U2-1.1a as a class, then have students work on activity U2-1.1 individually, then do U2-1.1b as a small group reinforcement activity.
- If your students are not familiar with peanut butter and jam sandwiches, or if you need to avoid peanut butter in case of nut allergies, switch out the food. Choose something your students will be familiar with that seems simple but also has multiple steps to make.
  - Alternative sandwich ideas: vegemite and cheese, cream cheese and cucumber, cream cheese and jam
  - Other foods: rice balls, soft tacos, yogurt parfaits

### Answer key

Question	Type	Sample answer	Marking notes
1	EA RC SE	<i>Take two pieces of bread. Add peanut butter to one piece of bread. Put jam on the other one. Put the peanut butter and jam together.</i>	Any instructions – even full of errors – are fine. Alternatively, you might have students ‘fix’ their answers and submit those to demonstrate an understanding of the key concepts of exacting and sequence.
2	SE	<i>It was not a sandwich – it was just jars of peanut butter and jam stacked on bread!</i>	Opinion based or descriptive answers are acceptable.

### Activity U2-1.1b Change it up: Human robots

<b>Activity size</b>	Medium
<b>Delivery recommendations</b>	Consider delivering this activity in conjunction with U2-1.1
<b>Resources needed</b>	Worksheet U2-1.1b, supplies for marking out the human robot grid
<b>NB</b>	This activity is best done as a group or whole class as it will likely need an instructor to run effectively.

#### Overview

This offline activity is designed to help students better understand the ‘robotic’ logic of sequence. Similar to activity U2-1.1, this activity has students use a grid to give and receive sequential instructions. In this activity, however, the students follow those instructions as ‘human robots’.

Understanding that computers are super literal and require instructions that are both exacting and sequential is crucial for success in coding. Helping build up students’ understanding of this concept will help them better problem-solve when they create programs of their own.

#### What to do

Set up a human-sized grid with a goal marker somewhere on the grid. Determine the start point and have students work out the instructions needed to get the ‘human robot’ to that goal. Have students work together to write down and then give the instructions to another student acting as the robot. Remind students that the robot can only follow the instructions that the robot receives exactly as they receive them. No fixing allowed!

### Tips and tricks

- This activity is best done in groups or as a whole class as it generally needs an instructor to set-up and run effectively.
- A great example of how to run this activity can be seen at <https://csunplugged.org/en/topics/kidbots/unit-plan/rescue-mission/>
- You may choose to run this activity in conjunction with activity U2-1.1a and activity U2-1.1. For example, you could run activity U2-1.1a as a class, then have students work on activity U2-1.1 individually, then do U2-1.1b as a small group reinforcement activity.

### Answer key

Question	Type	Sample answer	Marking notes
1	SE	<i>&lt;series of steps leading the robot to the goal&gt;</i>	Answers will depend on the grid and task you set.
2	SE	<i>Yes, but we forgot to include some of the turns, so we had to fix that first.</i>	

### Activity U2-1.2 Let's explore going step-by-step in EdScratch

<b>Activity size</b>	Small
<b>Delivery recommendations</b>	Recommend if students are new to EdScratch
<b>Resources needed</b>	Basic supplies set, worksheet U2-1.2

#### Overview

This activity gets students to apply the idea of sequence and sequential programming to the EdScratch environment. Exploring how changing the sequence of the blocks on the screen affects the robot's behaviour helps solidify the importance of sequence in programming.

The idea that each EdScratch block is one 'action' is introduced. Understanding this idea will help students apply sequence when building programs. Students also experiment moving blocks around and changing inputs inside blocks to gain confidence manipulating the EdScratch language.

### Tips and tricks

- If students are struggling to get blocks to move from the block pallet, make sure that they are dragging the blocks straight to the right into the programming area, not in another direction (such as up or down inside the blocks pallet itself).
- Blocks must be attached to the start block to be downloaded to Edison. Floating blocks not attached to a start block will not download to Edison.

- You can remove blocks you don't want by dragging them into the trash bin in the lower right corner of the programming areas or back into the block pallet.
- While the term 'input parameter' is not introduced until the next activity, beginning to refer to the options inside a block as 'input parameters' is a habit you may want to get into straight away.

### Answer key

Question	Type	Answer/Sample answer	Marking notes
1	EA	<i>First, the robot will drive forwards for 2.5 seconds at speed 5. Second, it will beep. Third, it will turn the left LED on. Fourth, it will spin left for 120 degrees at speed 10. Fifth, it will drive backwards for 2 seconds at speed 2.</i>	NOTE: Task 2 encourages kids to find and fix any errors in what they have written in either this answer or their program.
2	SE	<i>First, the robot will drive forwards for 2.5 seconds at speed 5. Second, it will drive backwards for 2 seconds at speed 2. Third, it will spin left for 120 degrees at speed 10. Fourth, it will turn the left LED on. Fifth, it will beep.</i>	Student answers should only move the existing blocks around – not add or change blocks.

### Activity U2-1.3 Let's explore driving Edison

<b>Activity size</b>	Medium
<b>Delivery recommendations</b>	Consider using activity U2-1.3a and this activity as a single lesson
<b>Resources needed</b>	- Basic supplies set, worksheet U2-1.3, activity sheet U2-2, activity sheet U2-3 - <i>Optional:</i> supplies for making your own tracks

#### Overview

Get the robots moving by applying sequential coding to the 'drive' category of blocks in EdScratch. Kids work through two programming tasks to get the robots to complete driving tasks – first, a straight track and then a mini maze.

The mini-maze task requires students to use computational thinking to create a sequential program to solve the maze. To complete the maze without 'cheating' students inevitably need to iterate different versions of their initial program concept, experimenting with different blocks and input parameters.

#### Tips and tricks

- You can also have students make their own straight tracks or mazes to complete on large paper or by using tape to mark out a track on the floor or desk.

- First attempts in programming are almost always ‘wrong’ – and that is perfectly okay! Persevering and changing programs iteratively, testing as they go, will get students to the right answer. Resilience is key!
- Due to minor mechanical differences in the motors and encoders inside different Edison robots, some robots may not turn to exactly 90 degrees when given the input of 90. Encourage students to try different values around 90 (e.g. 87 or 93) to find the input that works best for their Edison.

### Answer key

Question	Type	Sample answer	Marking notes
1	RC	<i>forwards for 26.5 cm at speed 5 OR forwards for 10.5 inches at speed 5 OR forwards for 1.05 seconds at speed 5</i>	Student answers will depend on the input parameters they select. Sample answers assume completion on the provided activity sheet using speed 5.
2	SE	<i>1. Go straight. 2. Turn left. 3. Go straight. 4. Turn left. 5. Go straight. 6. Turn right. 7. Go straight then stop.</i>	Students may miss steps in their initial plan – creating the program will show them any issues.

### Activity U2-1.3a Challenge up: Maze madness

<b>Activity size</b>	Small to medium (depending on the number of tasks attempted)
<b>Delivery recommendations</b>	Consider using this activity as a set of extra challenges for activity U2-1.3
<b>Resources needed</b>	- Basic supplies set, worksheet U2-1.3a, activity sheet U2-3 - <i>Optional:</i> supplies for making your own maze

#### Overview

Perfect for students who finish activity U2-1.3 quickly or for students needing additional task-based sequential programming practice, this set of 3 extension ideas offers variations on the maze-driving task from activity U2-1.3.

#### Tips and tricks

- If students are making their own mazes to complete, they will be most successful doing so on large paper or by using tape to mark out a track on the floor or desk.
- Due to minor mechanical differences in the motors and encoders inside different Edison robots, some robots may not turn to exactly 90 degrees when

given the input of 90. Encourage students to try different values around 90 (e.g. 87 or 93) to find the input that works best for their Edison.

- Encourage students to try both the ‘turn’ and ‘spin’ input parameters to see which works best.

### Activity U2-1.3b Challenge up: Self-walking pet

<b>Activity size</b>	Large
<b>Delivery recommendations</b>	This is a good activity to bring maker-space/crafting and coding together
<b>Resources needed</b>	Basic supplies set, worksheet U2-1.3b, maker-space/crafting supplies

#### Overview

This fun activity asks students to use their creativity to make Edison into something else – in this case, a pet which can walk itself.

The coding part of this activity can be quite simple: a program where the robot moves forward for a while, turns around and comes back is sufficient. The real challenge lies in building Edison into a pet.

A few hints as to how this can be accomplished are included in the student worksheet, but the sky is the limit! This is a creative activity for students to experiment with what will and will not work in terms of design, materials and construction.

#### Tips and tricks

- To download and run the program, students will need to access the EdComm cable attachment area and buttons. You may choose to caution students about this at the start of the activity or use any designs which do not cater for this as a chance to explore design planning.
- This activity can be adapted as a home economic sewing project.
- There is no ‘right’ answer, but here is what one sample solution looks like:



## Lesson 2: Inputs and outputs

This lesson begins students' exploration of the fundamental computer programming concepts of inputs (including input parameters) and outputs. Students engage with the robots' various outputs by creating and modifying sequential programs for Edison using a range of EdScratch blocks from the 'Drive', 'LEDs', 'Sound', and 'Control' categories. The idea of bugs in programming and the practice of debugging are introduced, preparing students to work with more complicated programming structures in future activities.

This lesson has a total of five base activities and nine extension activities:

- [U2-2.1 Let's explore Edison's outputs](#)
  - o [U2-2.1a Challenge up: Drive the maze safely](#)
- [U2-2.2 Let's explore input parameters](#)
  - o [U2-2.2a Change it up: Teach Edison to count to 9](#)
  - o [U2-2.2b Challenge up: Teach Edison to count to 9 out loud](#)
- [U2-2.3 Let's explore Edison's musical talents](#)
  - o [U2-2.3a Change it up: Play a song in a round](#)
  - o [U2-2.3b Challenge up: You are the conductor](#)
- [U2-2.4 Let's explore bugs and debugging](#)
- [U2-2.5 Let's explore Edison's motors](#)
  - o [U2-2.5a Challenge up: Spinning garden](#)
  - o [U2-2.5b Challenge up: Spinning solar system](#)
  - o [U2-2.5c Challenge up: Cartographer and navigator](#)
  - o [U2-2.5d Challenge up: Writer and director](#)

### Activity U2-2.1 Let's explore Edison's outputs

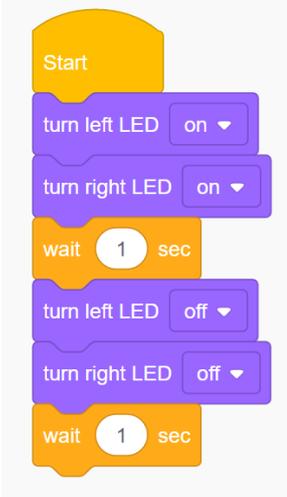
<b>Activity size</b>	Large
<b>Delivery recommendations</b>	
<b>Resources needed</b>	Basic supplies set, worksheet U2-2.1

#### Overview

The input-process-output cycle is introduced and then experimented with in this activity with an emphasis on observing the output results of sequential programming. Students begin to explore the 'LED' and 'Sound' categories of blocks in EdScratch and are introduced to their first control structure – the 'wait' block.

Beginning to understand that the input-process-output cycle is at play in every program helps students start to decompose programming into separate stages. This is an important skill for debugging in coding and for the computational thinking skill of decomposing problems into smaller chunks.

Answer key

Question	Type	Answer/Sample answer	Marking notes
1	EA	<ul style="list-style-type: none"> <li>• Turn Edison's lights off -- LEDs</li> <li>• Play a musical note -- Sound</li> <li>• Spin Edison right -- Drive</li> </ul>	
2	SE	<i>The robot did all the steps, but it was really fast, so it sort of seemed like it wasn't doing them in order but all at once.</i>	Students may note that the execution of the outputs is very quick.
3	RC		What works 'best' is subjective, but students should have a wait block in-between the LEDs being turned on and being turned off at a minimum.
Mini-challenge	n/a		The worksheet does not instruct students to share their answer, but you may choose to have students present their thinking or resulting code.

### Activity U2-2.1a Challenge up: Drive the maze safely

<b>Activity size</b>	Small
<b>Delivery recommendations</b>	Complete activities U2-1.3 and U2-2.1 prior to this activity
<b>Resources needed</b>	Basic supplies set, worksheet U2-2.1a, activity sheet U2-3

#### Overview

Students will need to put their understanding of sequence into careful practice in this activity. Writing a program that has the robot perform all the steps to complete the maze plus indicating turning direction using the LED lights, the buzzer or both before making each direction change is a great way to let students apply their understanding of sequence in a more creative programming challenge.

#### Tips and tricks

- Remind students that they can use 'wait' blocks to have the robot pause for a set amount of time.
- While the worksheet doesn't ask students to capture their program, you may choose to have students write down or demonstrate their program and explain their reasoning for using the blocks, inputs, and sequence they chose.
- Due to minor mechanical differences in the motors and encoders inside different Edison robots, some robots may not turn to exactly 90 degrees when given the input of 90. Encourage students to try different values around 90 (e.g. 87 or 93) to find the input that works best for their Edison.
- Encourage students to try both the 'turn' and 'spin' input parameters to see which works best.

### Activity U2-2.2 Let's explore input parameters

<b>Activity size</b>	Small
<b>Delivery recommendations</b>	
<b>Resources needed</b>	- Worksheet U2-2.2 - <i>Optional:</i> Basic supplies set

#### Overview

This offline activity explores the concept of input parameters, building on the basic definition introduced in activity U2-1.3. This activity helps students understand the concept of input parameters and establishes a framework for working out what any given input parameter is contributing to a program. Being able to think critically about input parameters helps students to understand what is actually happening in a program.

Input parameters contribute critical information to the inputs of programs. By explicitly examining input parameters, students gain a deeper understanding of the

inputs that they are using in their programs. The ability to link the input components of any given program to the expected outputs of that program is an important skill to develop for computational thinking and problem-solving.

### Tips and tricks

- You may want students to use the programs in the worksheet with their Edison robots to see the programs in action, helping to cement the connection between inputs and outputs.
- To further demonstrate the connection between input parameters' effects on outputs, have students adjust the input conditions of the worksheet programs in EdScratch, hypothesise on what those changes to the inputs will do to the outputs, and then test the programs with their robots to see the results.

### Answer key

Question	Type	Answer	Marking notes
1	EA	<i>seconds</i>	
2	EA	<i>7</i>	
3	EA	<i>Drive backwards for 6.5 seconds at speed 7.</i>	
4	EA	<i>4</i>	While this program has multiple blocks, only one of the blocks has input parameters.
5	EA	<i>What direction do you want the robot to go?</i>	Any answer that shows an understanding that this input parameter is related to direction is acceptable.

### Activity U2-2.2a Change it up: Teach Edison to count to 9

<b>Activity size</b>	Medium
<b>Delivery recommendations</b>	Complete activity U2-2.2 prior to this activity
<b>Resources needed</b>	Basic supplies set, worksheet U2-2.2a, activity sheets U2-4, U2-5, U2-6 and U2-7

### Overview

This extension activity offers programming challenges which put some of the skills students have been developing to use. Creating programs for the Edison robot to 'trace' (i.e. drive in the pattern) the digital display numbers of their choice requires the application of sequential thinking and an understanding of the input-process-output cycle.

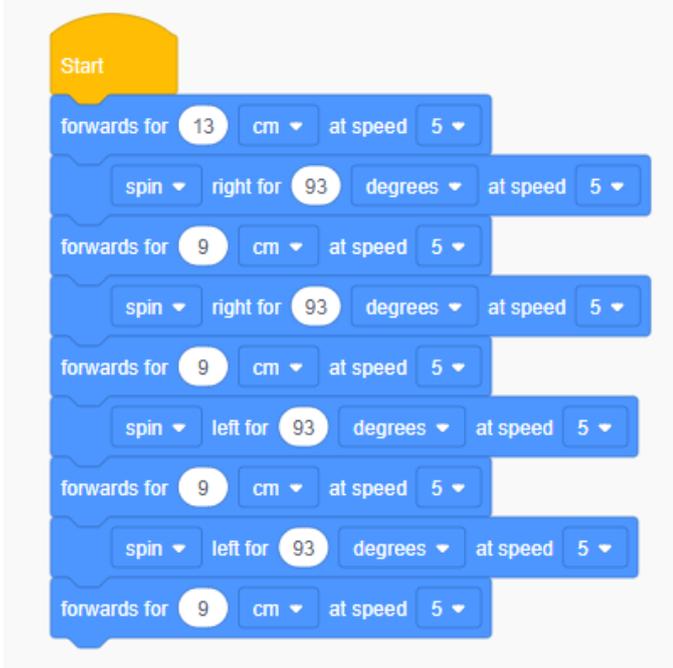
This activity calls attention to one particular input parameter (distance in the drive blocks) and asks students to identify patterns regarding the input parameter's effects on output. Students then extrapolate from what they observe, applying their understanding to a different program challenge.

### Tips and tricks

- Each segment in all of the digital display number activity sheets is the same length. However, when Edison turns, the robot does not turn on the spot but travels a bit. As a result, the distance parameter required to complete any segment after a turn will be smaller than the distance parameter needed to complete a straight segment.
- Due to minor mechanical differences in the motors and encoders inside different Edison robots, some robots may not turn to exactly 90 degrees when given the input of 90. Encourage students to try different values around 90 (e.g. 87 or 93) to find the input that works best for their Edison.
- Encourage students to try both the 'turn' and 'spin' input parameters to see which works best.
- It is possible to write a program for any of the numbers in the activity sheets without the robot needing to trace over the same segment twice. You may choose to mention this to help students envision the sequence they need to use in their programs.
- If Edison's wheels catch the edge of the activity sheet's paper, this can throw the robot off slightly. You can fix this by taping the activity sheet down or replicating the activity sheet's pattern on a larger piece of paper.

### Answer key

Question	Type	Sample answer	Marking notes
1	SE	2	

2	RC		
3	SE	<p><i>The distance input parameter for the first drive block is slightly longer than all the others. All of the other blocks use the same number for the distance input parameter. I think this means that any digital display number will only need one of the two distance input parameters: one for a straight segment and one for after a turn.</i></p>	<p>Ideally, student answers will note a correlation between the value needed in an input parameter and the repeated real-world distance the robot needs to travel.</p>
4	SE	7	
5	RC		<p>You may choose to see if students used the same distance input parameters in each program.</p>

6	SE	<i>The distance input parameter for the first drive block, which is straight, is the same in both programs. Any drive blocks following a turn are shorter, but the same as each other. The distance input parameter in the turns is always the same. The last block of the program for '7' is the same as the first drive block in each program plus the drive block after a turn because it is a combination of two segments.</i>	Ideally, student answers will note a correlation between the value needed in an input parameter and the repeated real-world distance the robot needs to travel.
---	----	--	---

### Activity U2-2.2b Challenge up: Teach Edison to count to 9 out loud

<b>Activity size</b>	Small
<b>Delivery recommendations</b>	It is recommended to complete activity U2-2.2a prior to this activity
<b>Resources needed</b>	- Basic supplies set, worksheet U2-2.2b, activity sheets U2-4, U2-5, U2-6 and U2-7 - <i>Optional:</i> supplies for making your own digital display numbers including appendix 1 (blank digital display sheet) from this guide

#### Overview

Bring multiple outputs together into a single sequential program in this semi open-ended challenge.

There's no real 'right' or 'wrong' way to complete this programming challenge. Any program that meets the criteria of 'drive the number' and 'count the number' is okay. This is an important concept for students to begin to understand. Programming is highly creative, and there are often many solutions possible. That's part of what makes computer science so fun and interesting!

#### Tips and tricks

- Students can use the Sounds outputs, the LEDs outputs, both or a combination of the two to count. (They can probably also use the Drive outputs to count if they can figure out a way to do it while still tracing their display number.) For example, if the student has selected the number 5, their Edison might drive a segment of the display number, beep, turn, then drive the next segment, beep, turn, etc. until it has driven all segments and beeped five times. Alternatively, the program could drive the whole path, then flash

the robot's left LED on and off five times. Any program where the robot drives the path and somehow signals the same value as that path is fine!

- If you want to highlight the idea that many different solutions are possible, split your students into groups for this activity and give each group the same digital display number. Have each group demonstrate their solution for the class. Discuss the differences and similarities in each group's program and highlight how, even if they are different, by meeting the criteria, they are still all correct.
- If you have students make their own numbers, you may want to give them a hint as to what segments they have to work with: a standard display grid is fully lit up when displaying the number 8. Alternatively, you can provide them with the blank digital display number worksheet found in appendix 1 in this guide.

### Activity U2-2.3 Let's explore Edison's musical talents

<b>Activity size</b>	Large
<b>Delivery recommendations</b>	
<b>Resources needed</b>	Basic supplies set, worksheet U2-2.3

#### Overview

Simple musical tunes are a great example of sequence in action. Using Edison to create music ties together sequence and inputs-outputs into programs that students love.

This activity is designed to mix the familiar with the unknown. By having students experiment using new blocks without explicit instructions or detailed explanations, they apply computational thinking practices and develop problem-solving mindsets.

The final task in this activity includes a semi open-ended programming challenge. This is a great opportunity for students to experience the creative side of coding.

#### Tips and tricks

- The song used in task one in the worksheet is *Mary Had a Little Lamb*. Use this share-code to access a program with more of the song:  
<https://www.edscratchapp.com?share=Eb12x3Dm>
- The third input parameter in the 'music note' block has three choices: ( - ), sharp, flat. The default ( - ) input means to play a normal note, not a sharp or flat.
- The 'tempo' block sets the tempo to be played. Only blocks which come after a 'tempo' block will be affected.
- Edison's default tempo is 'medium'. If no 'set music tempo to' block is used, the robot will play notes at the default tempo.

- Use this share-code to access a program with more of *The Hokey-Pokey* song: <https://www.edscratchapp.com?share=v0wMx5D5>  
**NB:** This version of the song uses loops.
  - You may want to wait to use this program until after students have studied repeating loops.
  - Repeat loops cannot be used inside 'play music in background' blocks. To use the longer version of the song in the background, duplicate the blocks inside the loop and add them into the program in sequential order instead.

### Answer key

Question	Type	Answer/Sample answer	Marking notes
1	SE	<i>Very fast</i>	
2	SE	<i>It made the song play a lot quicker compared to the original with the 'medium' input parameter.</i>	
3	SE	<i>The 'set tempo' block sets the speed or tempo of the music.</i>	Students should identify that the block sets the tempo of the music.
4	EA	<i>If the 'set tempo' block is at the end, it won't do anything. This is because Edison looks at the blocks one by one and does them in the order. If there are no music blocks after the tempo block, then the tempo block has no effect.</i>	Ideally, students will identify that sequential programming means things only go into effect in order. Consider having students try using the tempo blocks in different places in the program to test and reinforce this concept.
5	EA	<i>No, it played the music and then moved. I think this is because the blocks are in order with all the music first and then the drive blocks.</i>	Student answers should note that the robot doesn't move until after the music plays and identify sequence as the reason.

### Activity U2-2.3a Change it up: Play a song in a round

<b>Activity size</b>	Medium
<b>Delivery recommendations</b>	Complete activity U2-2.3 prior to this activity
<b>Resources needed</b>	Basic supplies set, worksheet U2-2.3a, sheet music or access for students to look up songs
<b>NB</b>	This activity is best done as a group or whole class.

#### Overview

Collaborating with other students on a programming project is at the heart of this activity. Highlighting the ‘sound’ outputs in their robots and using the ‘wait’ block control structure, students work together to get multiple robots to perform in harmony.

#### Tips and tricks

- Consider giving your students some song options or sheet music to use in this activity. Good options for songs in a round include:
  - *Row, Row, Row Your Boat*
  - *Three Blind Mice*
  - *Frère Jacques*
  - *Farmer in the Dell*
- Getting the timing just right so the robots are perfectly in harmony may be difficult as all robots will need to start at exactly the same time. Activity U5-1.4d *Change it up: The Edison chorus* addresses this problem by switching out the ‘wait’ blocks for IR messaging blocks. You may choose to wait to use activity U5-1.4d *Change it up: The Edison chorus* and this activity together, highlighting the advantages and disadvantages of each approach.
- If you want to get really musical, get your students to sing along with their Edison robots so that the students and the robots all play the song in a round together!

### Activity U2-2.3b Challenge up: You are the conductor

<b>Activity size</b>	Medium
<b>Delivery recommendations</b>	Complete activity U2-2.3 prior to this activity
<b>Resources needed</b>	Basic supplies set, worksheet U2-2.3b, sheet music or access for students to look up songs

#### Overview

This free-form activity asks students to bring their favourite tune to life by programming it in EdScratch for Edison to play. Music is a great way for kids to express themselves and their interests, and this activity lets them marry that enthusiasm with programming, helping showcase the arts in STEAM (science, technology, engineering, arts and maths).

### Tips and tricks

- Having students search online for their favourite music opens up opportunities to discuss safe search habits, copyright and attribution, and many other digital citizenship topics.
- Students with musical talent can choose to write their own original song to program instead of using an existing song.

### Activity U2-2.4 Let's explore bugs and debugging

<b>Activity size</b>	Medium
<b>Delivery recommendations</b>	Strongly recommended before students begin using programming control structures (including loops and conditionals) or sensing in programs.
<b>Resources needed</b>	Basic supplies set, worksheet U2-2.4

### Overview

Brace yourself – this lesson has a lot more text and a lot more definitions than most, but don't let that scare you away! Understanding bugs and debugging is one of the most critical skills you need your students to develop. Computational thinking is, in many ways, all about problem-solving. And problem-solving in computer science is all about debugging.

Students and teachers alike often struggle when moving from purely sequential programming to coding using more dynamic programming structures such as loops, branching and conditionals. A key reason people struggle is that they lack the debugging skill-set needed to find, identify and fix errors successfully.

The fundamentals of debugging are introduced in this lesson, and the practice is applied to the EdScratch environment. This mainly offline activity will begin to develop debugging capabilities in your students that are crucial for setting them up for success in activities in other units.

### Tips and tricks

- While technology and computer science have developed the unfortunate reputation of being 'for males,' the history of computer science is full of fascinating people both male and female. Grace Murray Hopper (mentioned in this activity), Ada Lovelace, Margaret Hamilton, and Hedy Lamarr are just some of the fabulous females who have contributed to making modern computing what it is today. Any of these ladies make brilliant subjects for research projects and wonderful inspiration for students of any gender.
- You can find a complete guide to the warning messages in EdScratch at <https://meetiedison.com/robot-programming-software/edscratch/>. This guide includes all warning messages, what they mean and examples of when you

may encounter them. Warning messages in EdScratch can help you resolve any syntax errors and also give hints to help fix many common logical errors.

- All syntax errors that can be made in EdScratch will generate a red warning message in the bug box. Not all red messages are syntax errors, however.
- When a program runs but doesn't do what you expected, chances are there is a logical error in the program. Get into the habit of asking students what they want their program to do, then having them read out the program they have written in the same flow that the robot will. This will often uncover the logical error.

### Answer key

Question	Type	Answer	Marking notes
1	EA	<p><i>Syntax error: the tempo block cannot go inside of the 'play music in background' grouping block. Only music note blocks can go in that block.</i></p> <p><i>Logical error: the problem is with the sequential logic the programmer used. The programmer needed to use the tempo block to set the tempo before the music blocks in order to have the music play at the very fast tempo.</i></p>	Any description that identifies and explains the two errors correctly is acceptable.
2	EA	<p><i>To get the program to work as described the following changes need to be made:</i></p> <p><i>1 – the distance input in the 'spin right' block needs to be 360, not 630. [logical error]</i></p> <p><i>2 – the speed inputs for the 'spin left' and 'spin right' blocks need to be the same value. [logical error]</i></p> <p><i>3 – the 'set tempo' block needs to be outside of the 'play music in background' grouping block [syntax error]</i></p> <p><i>4 – the 'set tempo block' needs to be before the 'play music in background' block [logical error]</i></p> <p><i>5 – the 'spin left' and 'spin right' blocks need to be after the 'play music in background' block [logical error]</i></p>	Student answers should identify and fix all five errors. The inclusion of the error type is not requested in the worksheet but may be helpful in reviewing the bugs with students.

### Activity U2-2.5 Let's explore Edison's motors

<b>Activity size</b>	Large
<b>Delivery recommendations</b>	
<b>Resources needed</b>	Basic supplies set, worksheet U2-2.5

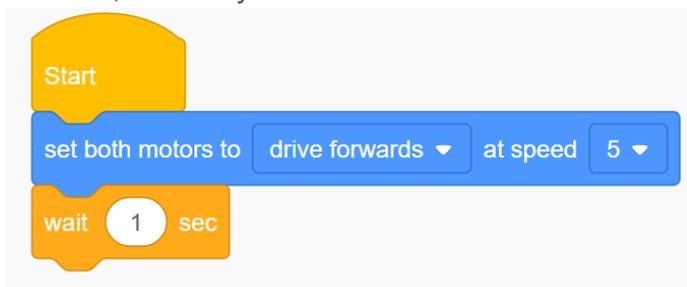
#### Overview

This lesson pulls together many of the key concepts from unit 2, including sequential programming, inputs and outputs, input parameters and debugging. Students further explore how Edison's motor outputs are controlled through drive blocks and learn that the motors can operate, and be controlled, independently. They are given a creative thinking challenge to imagine a use for this independent motor movement in a robotics application.

This activity also has students apply their understanding of how inputs (including information input through input parameters) affect outputs. They use this understanding to explore why some blocks in the language are dependent on other blocks. They are then challenged to apply their knowledge to get a program to work in an activity designed to push them, just a little bit, out of their comfort zone.

#### Tips and tricks

- The 'set motors' blocks only turn the motors on. These blocks require additional blocks to control the motor's activity by establishing a condition for the block. For example, this program says to turn the motors on, set to move forward, then says to wait in that condition for 1 second.



- The 'set motors' blocks can be a bit tricky to use at first, but they are actually very powerful blocks and are very helpful in many different robotics build projects using Edison.
- Even if you do not plan on using the 'set motor' blocks in other activities in this unit, this lesson activity is a good opportunity for students to experience, and overcome, difficulties in programming.

#### Answer key

Question	Type	Answer/Sample answer	Marking notes
1	EA	<i>The left wheel moves backwards.</i>	
2	EA	<i>The right wheel moves forwards.</i>	

3	SE	<i>I think that the 'set left motor' and the 'set right motor' blocks are the blocks that control only one motor because they say which motor is being affected.</i>			Students may not select the correct blocks, but they should have a reason to support the choice they make. Students may have discovered that the input parameters for the 'stop' block can also be set to affect only one motor at a time.
4	SE	<i>I would turn Edison into a fan, and the one motor would spin the fan blades.</i>			This is an exercise in the application of creative thinking. Any idea that uses one of Edison's motors is acceptable.
5	EA	<b>Information</b>	<b>In the program?</b>	<b>Value</b>	
		Direction	yes	forwards	
		Distance	yes	3	
		Distance units	yes	cm	
		Speed	yes	4	
6	EA	<i>The robot drove forwards for 3 cm at the speed of 4.</i>			
7	EA	<b>Information</b>	<b>In the program?</b>	<b>Value</b>	
		Direction	yes	forwards	
		Distance	no		
		Distance units	no		
		Speed	yes	5	

8	SE	<i>The robot didn't do anything. It couldn't because it did not have all the information it needed.</i>	The robot will not have the information to know how far to move, so will appear to do nothing. Student answers may note that there is a message in the bug box warning them that they need additional blocks.
9	SE	<i>First, I tried adding a 'set music tempo' block to the start of the program, but it still didn't move. I moved the 'tempo' block to the end of the program but that still didn't work. I looked at the other Drive blocks and saw that seconds was one of the distance units. That made me think of the 'wait' block, so I added that to the end of the program and it worked! I was getting really frustrated when I couldn't get the program to work, but once I did figure it out I was really happy!</i>	The key here is developing resilience. Any path students took and whether or not they got the program to work is secondary.

### Activity U2-2.5a Challenge up: Spinning garden

<b>Activity size</b>	Project
<b>Delivery recommendations</b>	- Complete activity U2-2.5 prior to this activity - This activity and activity U2-2.5b are both applied robotics projects. Consider using at least one as a robotics build project for this unit.
<b>Resources needed</b>	Basic supplies set, worksheet U2-2.5a, maker-space/crafting supplies, EdCreate kits/LEGO bricks

#### Overview

This open-ended project has students apply what they have learned about sequential programming, inputs and outputs, and debugging into a physical computing project. This robotics build project asks students to collaborate on the creation of a spinning garden. Students need to plan their garden, build the different elements to attach to the robots and program each robot to spin its attachment.

#### Tips and tricks

- You can see a version of a spinning garden project using Raspberry Pi boards on YouTube at <https://youtu.be/4Fs7y7gZlag?t=424> (Jump to the 7:00-minute mark to see the final spinning build in action.)

- Part of the idea of this project is to turn the robots onto their sides and use the robot's motors in a way other than to 'drive'. However, this project is also about unleashing students' creative thinking and problem-solving. If students decide to use the robots in ways other than with the robots on their sides, but are still using them to create a spinning garden, that is fine!

### Activity U2-2.5b Challenge up: Spinning solar system

<b>Activity size</b>	Project
<b>Delivery recommendations</b>	- Complete activity U2-2.5 prior to this activity - This activity and activity U2-2.5a are both applied robotics projects. Consider using at least one as a robotics build project for this unit.
<b>Resources needed</b>	Basic supplies set, worksheet U2-2.5b, maker-space/crafting supplies, EdCreate kits/LEGO bricks

#### Overview

This project has students apply what they have learned about sequential programming, inputs and outputs and debugging into a physical computing project. This robotics build project asks students to collaborate on the creation of a model solar system. Students need to plan their approach, build the different elements to attach to the robots and program each robot to spin its attachment.

#### Tips and tricks

- If you want your solar system to be a more accurate model, work with students to create the different scales needed to develop a more representative model. For example, determine what scale the planets are sized to and the scale for the distance each planet needs to be spaced from one another and the sun.
- If you want to allow for more creativity, have students design their own star system instead of making a model of the solar system. Students can create and name their own planets, and decide on the composition of each planet including features like it's dimensions, colouring, habitability, etc. Have students write stories about this star system and any inhabitants which reside inside of it.

### Activity U2-2.5c Challenge up: Cartographer and navigator

<b>Activity size</b>	Project
<b>Delivery recommendations</b>	- Recommended as a capstone activity following the completion of the Let's Explore activities from unit 2 - Both this activity and activity U2-2.5d are designed to be used as capstone project options for unit 2. Consider using at least one as a final project for this unit.
<b>Resources needed</b>	Basic supplies set, worksheet U2-2.5c, maker-space/crafting supplies for making the map

#### Overview

Get your students to apply sequence to a project where they design the set-up, the challenges, and the solutions. Students create a map large enough to use with their Edison robots, then design programming challenges to be solved using that map. After testing their challenges to make sure a solution is possible, they can then swap challenges with a classmate.

Designing the programming challenges and testing them for viability gets students to work with sequence and the other concepts from this unit in a new way. This project can be used as a 'showcase' for students to demonstrate their learnings from the unit.

#### Tips and tricks

- This project works well in groups or even as a whole class. Students can trade programming challenges with each other using the same or different maps.
- Consider linking this project to any relevant geography or world studies the students are doing by having students make maps of places they are learning about in other subjects.
- If students make their map using a grid system, it will be easier to navigate, but might not be as accurate. This is a good example of trade-offs in design and makes for a good example to discuss the topic.
- Does your class have a 'buddy class' of younger students? Try this variation: have the older students create a grid-system map and different driving challenges. (You may find it best to keep it simple and skip using 'Sound' and 'LED' elements for these challenges.) The 'cartographers' can then work with the younger students in the role of mentors, helping the other students to solve the challenges. The younger students can either use a push-button robot, program Edison using the [EdBlocks programming language](#) or help the older students program the robot using EdScratch. Whichever way you do it, giving your students the chance to help teach the concept of sequence will cement this key computation thinking concept for them.

### Activity U2-2.5d Challenge up: Writer and director

<b>Activity size</b>	Project
<b>Delivery recommendations</b>	- Recommended as a capstone activity following the completion of the Let's Explore activities from unit 2 - Both this activity and activity U2-2.5c are designed to be used as capstone project options for unit 2. Consider using at least one as a final project for this unit.
<b>Resources needed</b>	Basic supplies set, worksheet U2-2.5d, maker-space/crafting supplies for making the story map

#### Overview

Story maps are helpful tools both for writing stories and for breaking down the stories we read to show comprehension. Story maps are great examples of sequence in the real-world and that connection is the basis for this open-ended capstone project.

Students create a story using a story map. They 'illustrate' the story using Edison – by programming the robot to help bring the story to life! Students should use a combination of Edison's outputs using the robot's motors, LEDs and sounds to highlight key moments in their story, add lighting or sound effects or however else they like.

#### Tips and tricks

- A 'story mountain' style story map works well as a base design for this project as it creates a nice linear pathway. You can see an example of this style of layout here <https://www.tes.com/lessons/wlmfQ5vQh-8OPA/sequencing>
- This project lends itself well to group work.
- This project lends itself well to mixed media by filming presentations – a great 'showcase' opportunity.

## Unit 3: Got loops?

Examine the key computational concept of loops in this unit, exploring different ways loops can be used to control how the Edison robots behave. The topic of programming logic is examined more closely, including how control structures can affect the flow of code. Students continue to discover the EdScratch environment, learning more about the various block types in EdScratch and ways in which these blocks can be used to create with Edison.

### Learning objectives

Students will:

- be introduced to the computational thinking concept of repetition through the computer programming structure of loops
- develop a working understanding of the difference between definite and indefinite loops
- combine the concepts of sequence and loops into working programs in EdScratch
- be introduced to the programming concept of interrupts
- experiment with new blocks in the 'Control', 'Comments' and 'Events' categories in EdScratch
- expand on their problem-solving abilities by creating and programming robotics projects

**Key ideas:** loops (definite and indefinite), repetition, programming control structures, logic, events, interrupts, comments

### Lessons and activities in this unit

This unit includes two lessons with a total of six base activities and 14 extension activities.

#### Lesson 1: Loops

- [U3-1.1 Let's explore repeating steps](#)
  - [U3-1.1a Change it up: Drive a triangle](#)
  - [U3-1.1b Change it up: Drive a hexagon](#)
  - [U3-1.1c Challenge up: Choose your shape](#)
  - [U3-1.1d Challenge up: Drive a circle](#)
  - [U3-1.1e Change it up: Drive a square?](#)
  - [U3-1.1f Challenge up: Doodle-bot challenge](#)
- [U3-1.2 Let's explore loops and sequence](#)
- [U3-1.3 Let's explore forever loops](#)
  - [U3-1.3a Challenge up: Earworm](#)
- [U3-1.4 Let's explore stacking and nesting loops](#)
  - [U3-1.4a Change it up: Edison the designer](#)
  - [U3-1.4b Challenge up: Dance party!](#)

## Lesson 2: Interrupts

- [U3-2.1 Let's explore interrupting the main program](#)
  - o [U3-2.1a Change it up: Try a clap instead](#)
  - o [U3-2.1b Challenge up: Cheater bot](#)
  - o [U3-2.1c Challenge up: Pick one](#)
- [U3-2.2 Let's explore comments in coding](#)
  - o [U3-2.2a Challenge up: Create and comment](#)
  - o [U3-2.2b Challenge up: Share your comments](#)

## Lesson 1: Loops

The computational thinking concept of repetition is the primary focus of this lesson. Students explore repetition in programming and the idea of making code more efficient by using the computer programming structure of loops. Both definite and indefinite loops are introduced and explored using some of the loop blocks in the 'Control' section of EdScratch. Students increase their understanding of programming logic by combining sequential programming with loop structures in various ways to create programs for Edison using EdScratch.

This lesson has a total of four base activities and nine extension activities:

- [U3-1.1 Let's explore repeating steps](#)
  - o [U3-1.1a Change it up: Drive a triangle](#)
  - o [U3-1.1b Change it up: Drive a hexagon](#)
  - o [U3-1.1c Challenge up: Choose your shape](#)
  - o [U3-1.1d Challenge up: Drive a circle](#)
  - o [U3-1.1e Change it up: Drive a square?](#)
  - o [U3-1.1f Challenge up: Doodle-bot challenge](#)
- [U3-1.2 Let's explore loops and sequence](#)
- [U3-1.3 Let's explore forever loops](#)
  - o [U3-1.3a Challenge up: Earworm](#)
- [U3-1.4 Let's explore stacking and nesting loops](#)
  - o [U3-1.4a Change it up: Edison the designer](#)
  - o [U3-1.4b Challenge up: Dance party!](#)

### Activity U3-1.1 Let's explore repeating steps

<b>Activity size</b>	Medium
<b>Delivery recommendations</b>	Consider delivering this activity along with U3-1.1a and/or U3-1.1b as a single lesson
<b>Resources needed</b>	Basic supplies set, worksheet U3-1.1, activity sheet U3-1

#### Overview

Loops, the first programming control structure of this lesson, are introduced in this activity. Students first write a program to have Edison drive in a square using only sequential programming, observing the repeating pattern. Definite loops are then introduced, and students recreate their 'drive in a square' program using this more efficient coding method.

#### Tips and tricks

- If students are struggling to understand the loop concept, it can be helpful to draw out what is happening in this program on paper or the whiteboard. Write down the 'drive' and 'turn' commands in sequential order and wrap them in a loop. Write the numbers 1, 2, 3, and 4 next to the loop. Use arrows to show that the robot will do the two commands in order, then move back to the top of

the loop. Cross out the number 1 and then repeat the process until you have moved through all four repetitions of the loop.

- Loops make programming more efficient, but that doesn't mean that programs that don't use loops are wrong. Programming both the eight-block and three-block program, observing that both programs work equally well at getting Edison to drive in a square, is a good way for students to experience the idea that 'multiple solutions are possible in coding' for themselves.
- By first writing the eight-block program, then the three-block program, students can see how programming structures can make creating programs easier and more efficient.
- If Edison's wheels catch the edge of the activity sheet's paper, this can throw the robot off slightly. You can fix this by taping the activity sheet down or replicating the activity sheet's pattern on a larger piece of paper.
- Due to minor mechanical differences in the motors and encoders inside different Edison robots, some robots may not turn to exactly 90 degrees when given the input of 90. Encourage students to try different values around 90 (e.g. 87 or 93) to find the input that works best for their Edison.
- When using programs that run multiple 'drive' commands consecutively, students may note that their Edison robot's accuracy is reduced as more and more commands run. Adding a pause (using either a 'wait' with a very short input value or a 'stop motors' block as appropriate) between 'drive' commands allows the motors to fully stop moving, which will increase accuracy.

### Answer key

Question	Type	Answer/Sample answer	Marking notes
1	EA	8	If students only have seven blocks, their robot likely doesn't complete the final turn, which returns it to the starting position, as instructed on the worksheet.
2	SE	<i>There is a pattern to the blocks. Only two block types are being used with the same input parameters, and they repeat four times.</i>	Students should note a pattern of repetition regarding the blocks being used and/or the order of the blocks in the program.
3	EA	4	
4	SE	<i>The value needs to be 4 because a square has four sides and four corners. So, the code needs to drive each side and turn each corner a total of 4 times.</i>	Students will ideally note a correlation between the number of looped actions required and the shape's sides/angles.

### Activity U3-1.1a Change it up: Drive a triangle

<b>Activity size</b>	Small
<b>Delivery recommendations</b>	- Complete activity U3-1.1 prior to this activity - Consider delivering this activity along with U3-1.1 and/or U3-1.1b as a single lesson
<b>Resources needed</b>	Basic supplies set, worksheet U3-1.1a, activity sheet U3-2

#### Overview

Students practice using the definite loop programming structure to write an efficient program to get Edison to drive a triangle. This activity is designed to be used in conjunction with activity U3-1.1 to help students master the connection between the number of loops in a definitive loop and the resulting output.

#### Tips and tricks

- Working out the angle in degrees that the robot needs to turn to trace the triangle can be tricky. Remind students that the sum of the interior angles of a triangle is  $180^\circ$  and that the triangle on the activity sheet is an equilateral triangle.
- This activity lends itself well to a geometry extension lesson.
- If Edison's wheels catch the edge of the activity sheet's paper, this can throw the robot off slightly. You can fix this by taping the activity sheet down or replicating the activity sheet's pattern on a larger piece of paper.

#### Answer key

Question	Type	Answer/Sample answer	Marking notes
1	SE	3	The minimum number of blocks needed for a successful program is 3: a loop, a drive forward block and a turn block.
2	EA	3	
3	SE	<i>The value needs to be 3 because a triangle has three sides and three corners. So, the code needs to drive each side and turn each corner a total of 3 times.</i>	Students will ideally note a correlation between the number of looped actions required and the shape's sides/angles.

### Activity U3-1.1b Change it up: Drive a hexagon

<b>Activity size</b>	Small
<b>Delivery recommendations</b>	- Complete activity U3-1.1 prior to this activity - Consider delivering this activity along with U3-1.1 and/or U3-1.1a as a single lesson
<b>Resources needed</b>	Basic supplies set, worksheet U3-1.1b, activity sheet U3-3

#### Overview

Students practice using the definite loop programming structure to write an efficient program to get Edison to drive a hexagon. This activity is designed to be used in conjunction with activity U3-1.1 to help students master the connection between the number of loops in a definitive loop and the resulting output.

#### Tips and tricks

- Working out the angle in degrees that the robot needs to turn to trace the hexagon can be tricky. Remind students that the sum of the interior angles of a hexagon is  $720^\circ$  and that the hexagon on the activity sheet is a regular (meaning that all sides are equal) 6-sided shape.
- This activity lends itself well to a geometry extension lesson.
- If Edison's wheels catch the edge of the activity sheet's paper, this can throw the robot off slightly. You can fix this by taping the activity sheet down or replicating the activity sheet's pattern on a larger piece of paper.

#### Answer key

Question	Type	Answer/Sample answer	Marking notes
1	SE	3	The minimum number of blocks needed for a successful program is 3: a loop, a drive forward block and a turn block.
2	EA	6	
3	SE	<i>The value needs to be 6 because a hexagon has six sides and six corners. So, the code needs to drive each side and turn each corner a total of 6 times.</i>	Students will ideally note a correlation between the number of looped actions required and the shape's sides/angles.

### Activity U3-1.1c Challenge up: Choose your shape

<b>Activity size</b>	Medium
<b>Delivery recommendations</b>	Complete activity U3-1.1 and either U3-1.1a or U3-1.1b prior to this activity
<b>Resources needed</b>	Basic supplies set, worksheet U3-1.1c, supplies for making their own shapes

#### Overview

Students apply their understanding of the relationship between a definite loop and a regular shape by choosing a shape and then writing an efficient program that gets Edison to drive that shape. This activity asks students to articulate the patterns they have noticed about using definite loops with shapes and extrapolate from that to explore other shapes.

#### Tips and tricks

- To complete a shape using a 3-block program consisting of a definite loop which contains a 'drive' block and a 'turn' block, students must select a regular shape. However, if students decide not to use a regular shape, they can still attempt the challenge. You may want to run activity U3-1.2 first if students want to use non-regular shapes.

#### Answer key

Question	Type	Answer/Sample answer	Marking notes
1	EA	12	
2	SE	<i>My shape was an octagon, which has eight sides and angles. You always need to have the input parameter be equal to the number of repeating sides and angles that the shape has, so I knew that I would need an 8 as the repeat block's input parameter.</i>	Students will ideally note a correlation between the number of looped actions required and the shape's sides/angles.

### Activity U3-1.1d Challenge up: Drive a circle

<b>Activity size</b>	Small
<b>Delivery recommendations</b>	Complete activity U3-1.1 and either U3-1.1a or U3-1.1b prior to this activity
<b>Resources needed</b>	Basic supplies set, worksheet U3-1.1d, activity sheet U3-4

#### Overview

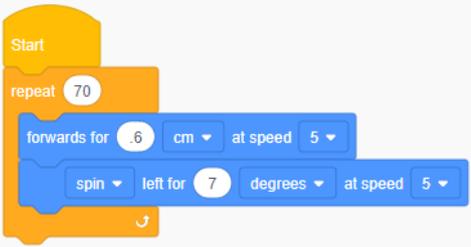
This activity is a great exercise in creative problem-solving and an exploration of the limits of the definite loop programming structure.

Students apply their understanding of the relationship between a definite loop and a regular shape in this challenge by extrapolating, using the pattern they have observed to create a program where Edison drives in a circle. This activity is designed to be used once students have realised that the number of loops required to get the robot to drive a shape is equal to the number of sides of that shape. To drive Edison in a circle, students need to expand on the idea and get a bit creative.

#### Tips and tricks

- If students are stuck, you can offer them the hint that ‘a shape with many very small sides can closely approximate a circle’.
- The input parameter in the repeat block has a value cap of 1000. If students put in a larger number, the block will automatically limit down to 1000.
- If Edison’s wheels catch the edge of the activity sheet’s paper, this can throw the robot off slightly. You can fix this by taping the activity sheet down or replicating the activity sheet’s pattern on a larger piece of paper.

#### Answer key

Question	Type	Sample answer	Marking notes
1	RC		
2	SE	<i>No, Edison isn't actually driving in a circle, but in a shape with lots of sides. It looks like a circle, but it's not a real circle.</i>	Students may be able to get the robot to approximate a circle closely. However, the robot is not driving in one continuous arch but making small forward movements and turning slightly each loop.

### Activity U3-1.1e Change it up: Drive a square?

<b>Activity size</b>	Small
<b>Delivery recommendations</b>	Complete activity U3-1.1 prior to this activity
<b>Resources needed</b>	Basic supplies set, worksheet U3-1.1e, activity sheet U3-1

#### Overview

Switch up the definite loop by introducing the random number input block. This activity adds a bit of variety to the definite loop and is a nice way to begin to showcase maths in coding. While the concepts are not explicitly introduced, the random number input is stored in Edison's memory as a variable value. Variables, which are introduced in unit 5 of these lessons, are used extensively when using Edison's sensors in programs. This activity is a nice way to plant the seeds of what is possible with data and Edison in your students' imaginations.

#### Tips and tricks

- 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 blocks in this program to suit their robots. To improve accuracy, students can also add a 'wait' block with a sort input value (e.g. .2 secs) in-between the drive commands.
- You can ask students to watch the program run on the worksheet and count the number of sides the robot drives. From this, they can determine the number of loops the program completed that run.
- While the number of loops the program will run is random, this is still a definite loop. The program will loop for a definite number of times being one value of the set of (1, 2, 3, 4, 5, 6, 7, 8, 9, 10).
- This activity lends itself well to a probability extension activity.
- If Edison's wheels catch the edge of the activity sheet's paper, this can throw the robot off slightly. You can fix this by taping the activity sheet down or replicating the activity sheet's pattern on a larger piece of paper.

#### Answer key

Question	Type	Sample answer	Marking notes
1	SE	<i>The first time I ran the program the robot drove around the square one full time plus it went 3 more sides, so it looped 7 times. The second time it only made it halfway around the square, so it only looped 2 times! Different things happen every time because the number of loops is random, not always 4.</i>	

### Activity U3-1.1f Challenge up: Doodle-bot challenge

<b>Activity size</b>	Large
<b>Delivery recommendations</b>	
<b>Resources needed</b>	Basic supplies set, worksheet U3-1.1f, EdCreate kits/LEGO bricks and/or any other maker space materials, felt-tip markers or equivalent

#### Overview

This challenge adds an engineering build element to the task of getting the robot to drive in a way that makes a shape. An open-ended challenge, this activity offers students very little guidance or limits. Student's apply their understanding of sequential programming and the definite loop control structure to get the robot to drive in a shape. Designing and then building a way to attach a pen to the robot so that this shape can be traced onto paper is the main challenge of this activity.

#### Tips and tricks

- Engineering a physical design that will put enough downward pressure on the pen to allow it to mark the paper while still enabling the robot to drive is challenging. Using felt-tip markers or another writing device that leaves ink behind it easily will make this challenge more achievable.
- This challenge can be run as an engineering design challenge. You can read more about using this challenge to teach engineering design at <https://meetedison.com/teach-engineering-design-with-edcreate/>

#### Answer key

Question	Type	Sample answer	Marking notes
1	SE	<i>Getting the pen attached to the robot was hard. My first plan didn't work at all, and I had to start over. My second plan used rubber bands to attach the pen to the side of the robot and that worked. I programmed the robot to draw an octagon, but when I went to run it, the rubber bands were over the wheel, which meant that Edison couldn't drive. I finally attached the pen to the back of Edison using some building beams and rubber bands and got the robot to drive. Edison drove in the octagon shape but not didn't draw it very well because there wasn't enough pressure on the pen to hold it down against the paper.</i>	Student answers should cover some of the challenges or problems they faced and how they dealt with those problems.

### Activity U3-1.2 Let's explore loops and sequence

<b>Activity size</b>	Medium
<b>Delivery recommendations</b>	
<b>Resources needed</b>	Basic supplies set, worksheet U3-1.2, activity sheet U3-5

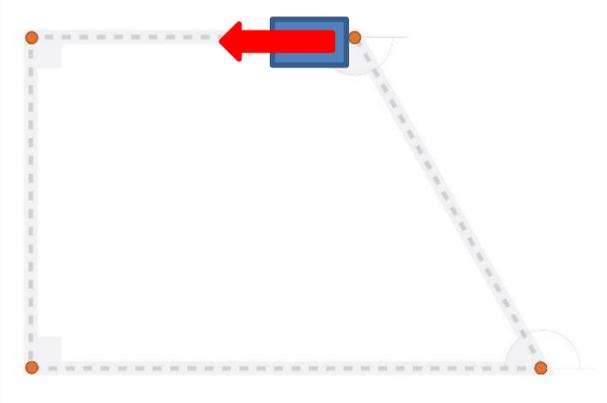
#### Overview

This activity has students write an efficient program that gets Edison to drive a non-regular shape. Students need to use both their mastery of the concept of sequence and their understanding of the definite loop structure to accomplish this task. Using a quadrilateral as the shape for Edison to drive is a great way to look at the limits of loops and reinforce the importance of sequence in programming. By choosing the correct starting point on a quadrilateral, a definite loop can still be used and will make the program more efficient. However, the program needs additional blocks outside of the loop in order to complete the shape.

#### Tips and tricks

- If students are struggling to determine what needs to go inside of the loop and what is outside of it, work through the quadrilateral's shape with your students. Look at what repeats in the shape and what does not. Use this to determine what steps can be inside the loop as well as the sequential order of all the code needed in the program.
- If Edison's wheels catch the edge of the activity sheet's paper, this can throw the robot off slightly. You can fix this by taping the activity sheet down or replicating the activity sheet's pattern on a larger piece of paper.

#### Answer key

Question	Type	Sample answer	Marking notes
1	SE		

2	SE	<i>I started the robot on the corner with the obtuse angle, pointing towards the corner with the right angle. That meant that the first two sides that the robot needed to drive were the same length and the first two turns were both 90 degrees. That is why the first part of my code is a loop that repeats twice. The rest of the program is individual blocks outside of a loop because the other sides and angles do not repeat on the shape, so cannot be used with a loop in the program.</i>	Student answers should note a correlation between the start point they used and where the program loops. <b>NB:</b> Some start points will not allow the program to use a loop.
---	----	---	---

### Activity U3-1.3 Let's explore forever loops

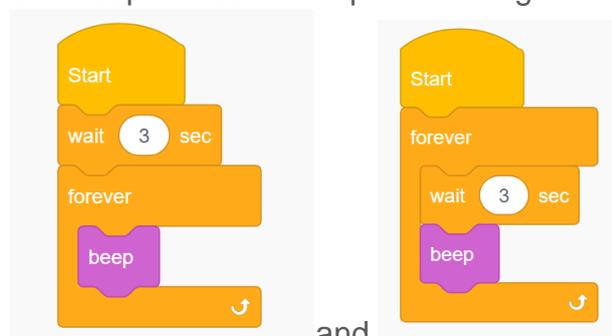
<b>Activity size</b>	Small
<b>Delivery recommendations</b>	It is recommended to introduce definite loops prior to this activity.
<b>Resources needed</b>	Basic supplies set, worksheet U3-1.3

#### Overview

This activity introduces the second major type of loop structure: the indefinite loop. While definite loops are generally easier to understand than indefinite loops, indefinite loops have a much wider range of application in computer programming. The 'forever' loop, which is an indefinite loop, is introduced and explored in this activity. Students use this new block in the semi open-ended egg-timer programming task, allowing them to apply their understanding of sequence and loop structures.

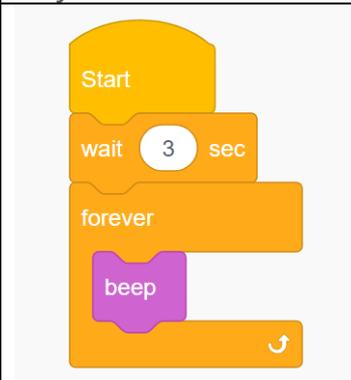
#### Tips and tricks

- Any program can be stopped at any time by pushing the stop (square) button. This will end a 'forever' loop and send the robot back into standby mode.
- You may want to encourage students to test what happens if they put both a 'wait' and a 'beep' block inside the 'forever' loop compared to having the 'wait' block outside and the 'beep' inside the loop. This is a great example of a



logical error: both and appear to be in the correct sequence of 'wait, then beep', but the two programs will behave completely differently.

### Answer key

Question	Type	Sample answer	Marking notes
1	SE	<i>I don't think you will be able to have Edison do anything else after the forever loop. For one thing, 'infinity' doesn't end, so this loop will never stop repeating. Also, the shape of the loop is different from other blocks – the bottom is square, so there's not a way to attach a next step block.</i>	Students answers will ideally note the block's function, shape or both as support for their answer.
1	RC		

### Activity U3-1.3a Challenge up: Earworm

<b>Activity size</b>	Small
<b>Delivery recommendations</b>	Complete activity U3-1.2 prior to this activity
<b>Resources needed</b>	- Basic supplies set, worksheet U3-1.3a - <i>Optional:</i> sheet music or access to look up music

#### Overview

This activity offers students another opportunity to use the 'forever' indefinite loop structure in a simple, but meaningful, way. Using the 'forever' loop to repeat musical note blocks is a good way to represent the common real-world phenomenon of getting a song stuck in our heads – i.e. an earworm. Practising using programming structures with real-world connections helps students connect the structure to its purpose, allowing them to identify uses for those structures in later activities when they tackle more complex programming challenges.

#### Tips and tricks

- If you want students to practice using the 'forever' loop but don't want to spend time on selecting music, you can ask students to open the demo program [Moving\\_with\\_music](#) and modify the program to create an earworm program using the tune in the demo program as their song.
- To emphasis where the loop ends, suggest students add a 'wait' block as the final block inside their earworm program so that Edison will pause before beginning the song again.

### Activity U3-1.4 Let's explore stacking and nesting loops

<b>Activity size</b>	Medium
<b>Delivery recommendations</b>	
<b>Resources needed</b>	Basic supplies set, worksheet U3-1.4, activity sheet U3-6

#### Overview

Students are introduced to the concept of using multiple loops in a program both by stacking them sequentially and by nesting one loop inside another. Nesting loops may seem like a new way of manipulating the flow of a program, but it really is just layering sequence. Working with programs with stacked and nested loops allows students to apply the computational thinking practices of decomposition (chunking problems into smaller parts) and pattern recognition as well as the application of sequential logic.

#### Tips and tricks

- If students are struggling to understand the loop concept, particularly nested loops, it can be helpful to draw out what is happening on paper or the whiteboard. You can do this by writing down each command one at a time in sequential order with your students or by writing down the program as it appears and then use arrows to indicate each looping action.
- The activity in task 1 (looking at code and working out what it does without running the program) is similar to a coding practice known as tracing. The formal practice of tracing involves stepping through a program line by line, recording important values. It is often done to help find errors or bugs in the code, but it is also useful when you just need to understand what is happening in a program. While the concept isn't explicitly introduced in this activity, and the need to record values in EdScratch is limited, getting students into the habit of tracing through programs will help them improve their ability to debug programs.
- If Edison's wheels catch the edge of the activity sheet's paper, this can throw the robot off slightly. You can fix this by taping the activity sheet down or replicating the activity sheet's pattern on a larger piece of paper.

#### Answer key

Question	Type	Answer/Sample answer	Marking notes
1	EA	3	
2	EA	<i>beeps</i>	
3	EA	15	
4	EA	<i>LED flashes</i>	
5	SE	<i>It's three squares in a row.</i>	
6	SE	<i>I think I can use a nested loop to get Edison to drive a square (which uses a loop to drive and turn 4 times) and then have that loop inside another loop that repeats 3 times.</i>	Ideally, students will see a correlation between the repeating pattern and the nested loop.

### Activity U3-1.4a Change it up: Edison the designer

<b>Activity size</b>	Large
<b>Delivery recommendations</b>	- Recommended to reinforce nested loops - Highly recommended if students are struggling with loop structures
<b>Resources needed</b>	Basic supplies set, worksheet U3-1.4a, activity sheet U3-7, craft supplies to make the test space

#### Overview

Give students the chance to put loops and sequence to the test by getting Edison to drive in some fabulous designs. This activity has students look at designs to identify repeating patterns, then construct a code solution as effectively as possible using loops and nested loops.

#### Tips and tricks

- Physically creating the design by marking it out or drawing it can help students recognise the repetitions that occur inside the pattern and better connect these to the loop structures they need inside their EdScratch code solution.
- Having students make their own patterns is a great way to have them think critically about what repetition looks like in practice. Creating a design which has a pattern inside a pattern is a great chance to apply computational thinking to design.

### Activity U3-1.4b Challenge up: Dance party!

<b>Activity size</b>	Large
<b>Delivery recommendations</b>	
<b>Resources needed</b>	- Basic supplies set, worksheet U3-1.4b - <i>Optional:</i> devices for playing music

#### Overview

Get the robots moving by putting Edison's outputs and the loop control structures into action. This mini project has students apply their understanding of sequence, repetition and problem-solving in a collaborative setting.

#### Tips and tricks

- This activity is designed to be used in groups to allow students to collaborate on a project. Students may work together across each step of the design, coding, and testing phases. Alternatively, students can divide up the work with certain team members in charge of specific tasks.
- The *Hokey Pokey* makes a great song choice if students are stuck for inspiration. The song lyrics lay out a plan for what students can program the robots to do!

## Lesson 2: Interrupts

Expand your understanding of computational logic by exploring interrupts in programming. This lesson introduces new blocks from the ‘Events’ category in EdScratch and uses these blocks to create subroutines. Students learn how interrupt-triggered subroutines work and apply their understanding of sequence to follow programs which include these types of subroutines. The computer programming practice of commenting is also introduced as a tool to help students track and explain their code and logic.

This lesson has a total of two base activities and five extension activities:

- [U3-2.1 Let’s explore interrupting the main program](#)
  - o [U3-2.1a Change it up: Try a clap instead](#)
  - o [U3-2.1b Challenge up: Cheater bot](#)
  - o [U3-2.1c Challenge up: Pick one](#)
- [U3-2.2 Let’s explore comments in coding](#)
  - o [U3-2.2a Challenge up: Create and comment](#)
  - o [U3-2.2b Challenge up: Share your comments](#)

### Activity U3-2.1 Let’s explore interrupting the main program

<b>Activity size</b>	Medium
<b>Delivery recommendations</b>	
<b>Resources needed</b>	Basic supplies set, worksheet U3-2.1

#### Overview

This activity introduces a set of new terms and concepts all related to one key idea: interrupting the main program. The fundamental computational thinking concept of sequence is given a new dimension with the introduction of interrupts and subroutines. These important programming concepts are critical for unlocking event-based programming with Edison, enabling the power of the robot’s sensors to be utilised.

This lesson activity reviews topics previously introduced to help students grasp how event-driven interrupts work. Students are introduced to the ‘Events’ block category in EdScratch and practice using a program which has an interrupt and subroutine to create a ‘decider bot’ with their Edison robot.

#### Tips and tricks

- While interrupts disrupt the sequential flow of a computer program, they are still bound by the same underlying logic. Subroutines will execute the commands contained within the subroutine in sequential order. Once complete, the program will return to the exact point of the main program

where it was prior to the interrupt, then continue to run the main program in sequential order.

- The decider bot can be used to answer any binary question. You may elect to have students create their own binary questions or create questions with two 'a or b' style options to answer with the decider bot.

### Answer key

Question	Type	Answer/Sample answer	Marking notes
1	EA	<i>The main program uses an indefinite 'forever' loop to make the robot turn on the right LED, wait .03 secs, turn the right LED off, turn the left LED on, wait .03 secs, and turn the left LED off. It will repeat that loop of actions forever.</i>	Students can word the answer differently but they should identify the main program's actions as described.
2	SE	<i>All the Event category blocks are yellow and shaped the same as the 'Start' block.</i>	Students should identify similarities between the Event blocks and the default Start block.

### Activity U3-2.1a Change it up: Try a clap instead

<b>Activity size</b>	Small
<b>Delivery recommendations</b>	Complete activity U3-2.1 prior to this activity
<b>Resources needed</b>	- Basic supplies set, worksheet U3-2.1a - <i>Optional:</i> worksheet U3-2.1 for reference

### Overview

This extension activity is designed to be used in conjunction with activity U3-2.1. Students once again program their Edison robots to be decider bots. This time, instead of the interrupt being caused by a button press, it is caused when the robot detects a clap.

While the primary objective of this activity is to reinforce the concepts of interrupts and subroutines, the activity also serves as a good soft introduction to using Edison's sensors. This can be used as a preparatory activity to help students become comfortable using sensors in programming before they begin to work with Edison's other sensors.

### Tips and tricks

- The robots may struggle to detect sounds when there is a high level of background noise. Having students tap a finger near the sound sensor on their Edison will create the same effect as clapping.

- Pressing the ‘play’ (triangle) button to run a program in Edison can create a small amount of noise which can trigger ‘Clap detected’ interrupts. If students have a ‘wait’ directly after the ‘Clap detected’ event (as in this example program <https://www.edscratchapp.com?share=pbz13M0a>) this can make it look like the program isn’t working correctly immediately after the ‘play’ (triangle) button is pressed. In actual fact, the event has just triggered, causing the ‘wait’. Have students wait until the set time has elapsed, then they will see the program will jump back to the main program, waiting for the next event trigger.

### Activity U3-2.1b Challenge up: Cheater bot

<b>Activity size</b>	Small
<b>Delivery recommendations</b>	Complete activity U3-2.1 prior to this activity
<b>Resources needed</b>	- Basic supplies set, worksheet U3-2.1b - <i>Optional:</i> worksheet U3-2.1 for reference

#### Overview

This extension activity is designed to be used in conjunction with activity U3-2.1. Students once again program their Edison robots to be decider bots. This time, in addition to the standard ‘pause’ subroutine running when triggered by an interrupt from one of the buttons (round or triangle) being pressed, they need to add a second subroutine to be a ‘cheat’ which gives a set answer whenever the other button is pressed.

In addition to reinforcing the concepts of subroutines and interrupts, students learn that they can have more than one subroutine per program. While not explicitly introduced, this activity demonstrates branching in programming and serves as a soft introduction to the concept of branching.

#### Tips and tricks

- An EdScratch program can only ever have one subroutine start with any given event. That’s why any event block, such as the ‘round button pressed’ event block, disappears from the block pallet when it is in the programming space.
- One possible answer to the cheater bot activity can be seen at <https://www.edscratchapp.com?share=K072a7Ym>

### Activity U3-2.1c Challenge up: Pick one

<b>Activity size</b>	Large
<b>Delivery recommendations</b>	Complete activity U3-2.1 prior to this activity
<b>Resources needed</b>	- Basic supplies set, worksheet U3-2.1c, maker-space materials to build the box - <i>Optional:</i> worksheet U3-2.1 for reference

#### Overview

This extension activity is designed to be used in conjunction with activity U3-2.1. Students once again program their Edison robots to be decider bots. This activity then asks students to create a physical box or another system to use with their decider bot. This physical design should allow Edison to ‘light up’ a choice. For example, students could design a box with two windows cut out of the front that words or pictures can be slid into. When the decider bot program runs, one of the two options will be lit up by the LEDs.

This physical design activity reinforces the concepts of subroutines and interrupts in programming and requires students to engage in problem-solving to create a tangible use for such a program.

#### Tips and tricks

- Students may find they need to increase the wait time in their subroutine depending on their design.
- The physical design needs to allow the robot’s buttons to be pressed in order to trigger the subroutine. Remind students of this as they create their designs.

### Activity U3-2.2 Let’s explore comments in coding

<b>Activity size</b>	Medium
<b>Delivery recommendations</b>	
<b>Resources needed</b>	Basic supplies set, worksheet U3-2.2

#### Overview

The practice of commenting in programming, and the ‘Comment’ block category in EdScratch, are introduced in this activity. Students see what comments look like in EdScratch and use an example program to help understand how comments are a helpful tool.

Some professional programs dislike comments, claiming that comments are a waste of time and arguing that ‘good code should be self-documenting’. Good comments help make great code, however, and from an educational perspective, commenting is a very helpful practice. Having students get into the habit of articulating what they are trying to achieve, especially when working with more complex code structures,

allows them to be far more independent problem-solvers when debugging. Commenting also allows for easier collaboration, as other students and facilitators can more quickly understand the programmer’s intentions. Moreover, commenting helps students practice explaining their ‘coder-thinking’ in human-readable terms. Being able to communicate about technological work effectively is a key skill in any digital literacy arsenal.

### Tips and tricks

- If students are struggling to understand what comments are, consider explaining comments using a printed document and some sticky-notes. Show how you can add sticky-notes to the document so that someone else can have some ‘hints’ about what is inside the document at specific points. The sticky-notes are a helpful way to leave notes about the ‘code’ (the document) but are not actually a part of the document.
- Remind students that Edison will ignore the comment blocks entirely. The robots cannot understand these blocks no matter what is put inside them.
- Adding a comment that says what you want the program to do won’t make the program do that thing: the actual code still needs to work.
- Simply adding comments doesn’t ‘debug’ a program. Comments can, however, be a great tool for students to use to find and fix logical errors, help them to work through their own thinking and to make changes to their code when debugging.
- How to use comments in coding isn’t something that is strictly defined and varies depending on the individual programmer. There are a few best practice rules, however, which you may want to introduce:
  - Comments should be easily understood – i.e. they should be ‘human readable’ (rather than being code snippets).
  - Comments should generally be at the top of the code they are explaining.
  - Comments only need to be added if the section of code is not self-evident.

### Answer key

Question	Type	Answer/Sample answer	Marking notes
1	SE	<i>I think that Edison will drive between 1 and 5 squares going forwards slowly, then drive between 1 and 5 triangles going backwards quickly. Edison will then repeat that whole set of actions 2 more times (a total of 3 times). If I clap at any time while the program is running, Edison will beep.</i>	

2	SE	<i>Yes, the comments do help because with so many loops it is a bit tricky to see what is going to happen. Having the comments made the code for driving the squares and triangle with the random number loop for each easier to understand.</i>	This is an opinion-based answer but should be supported by sound reasoning no matter the student's opinion.
3	SE	<i>The program mostly worked the way I expected but the robot beeped the whole time, and I didn't think that it would because I never even clapped.</i>	Any student experience is fine, but you may choose to see if the student's answer from question 1 aligns with the experience noted here.
4	EA	<i>The bug box has a warning that explains this: 'Driving the motors creates noise which may cause the 'clap' event to trigger. This may cause the 'clap event' blocks to trigger repeatedly while Edison is driving.'</i>	The cause noted in the sample answer is the reason this is occurring. Student answers must note this cause.

### Activity U3-2.2a Challenge up: Create and comment

<b>Activity size</b>	Large
<b>Delivery recommendations</b>	Recommended as a capstone activity following the completion of the 'Let's Explore' activities from unit 3
<b>Resources needed</b>	- Basic supplies set, worksheet 2.2a - <i>Optional:</i> maker-space materials

#### Overview

Students demonstrate their learnings from the unit and flex their creative thinking in this open-ended capstone project. This activity challenges students to create a program of their own design using the programming structures, coding practices and computational thinking skills they have learned. While a few parameters are given to set minimum requirements for what the programs need to include, students are free to design the program's purpose.

#### Tips and tricks

- Remind students that Edison has three types of outputs: sounds, LEDs and motor outputs.
- You may choose to encourage students to take a robotics build approach to this project, designing a program that uses their robot as a part of an engineered solution including some sort of physical build.
- Rather than having students copy out their programs manually, you can opt to have them use screen-shot software to capture and print an image. Alternatively, you can ask students to save their program and create a share

code of the program, providing the link as their answer. The share code link will be beneficial if you are having students complete activity U3-2.2b.

### Answer key

Question	Type	Sample answer	Marking notes
1	SE	<i>I want my program to turn Edison into a quiz guide bot. This will be a little like a decider bot, but instead of giving a random answer, I want to use it to be able to show someone if they answered a question right or wrong when I quiz them. It will have 2 subroutines, one to show that the answer was correct and one to show it was incorrect. When the program runs, Edison will flash the LEDs on and off. Once the person answers the question, I will be able to push the triangle button to play a tune and turn the lights off if the answer is wrong or push the round button to play a different tune while the robot spins with the lights on if the answer is correct.</i>	For reference, <a href="#">this link</a> shows the program initially devised by the programmer being described in the sample answer.
2	SE	<i>I used comments to label the subroutines, so I knew which was for the correct answer and which was for the wrong answer. One comment is at the top of the triangle button subroutine and says 'wrong answer'.</i>	You may want to confirm students are using good commenting practices in their choice of location and comment content.
3	SE	<i>One problem I had was that the music was still playing when the subroutine's control of the LEDs had ended. I fixed this in the triangle button subroutine by removing the 'play in background' block. Another problem I had was that the round button subroutine made the robot spin forever. I fixed this by changing the block to a drive block with a set time instead of using a wait block and the 'set motors' block.*</i>	Any experience is acceptable, but generally speaking, students will encounter issues. If a student has noted that they encountered no problems, you may want to check their program and encourage them to take risks and think creatively if the program they devised is very simple.

4	RC	Sample program available at <a href="https://www.edscratchapp.com/?share=GDLRNq0X">https://www.edscratchapp.com/?share=GDLRNq0X</a>	Having students provide a share code link instead of writing out their programs will enable them to reuse the program if you are having students complete activity U3-2.2b.
---	----	---	---

\* NB: the student also could have fixed the issue of the robot spinning forever by adding a ‘stop both motors’ block to the end of that subroutine. This is a good example of multiple coding solutions being equally viable to debug a problem.

### Activity U3-2.2b Challenge up: Share your comments

<b>Activity size</b>	Medium
<b>Delivery recommendations</b>	
<b>Resources needed</b>	Basic supplies set, worksheet U3-2.2b

#### Overview

Collaboration is at the heart of professional programming. Not only do teams of people within an organisation come together on programming projects, individuals from around the world swap and share code online. Good communication is key to enabling collaboration in coding to be successful. This activity explores just that, challenging students to use critical thinking to evaluate their own and a partner’s approach to commenting.

#### Tips and tricks

- This activity can also be run with small groups instead of pairs.
- If students completed and saved their programs from activity U3-2.2a, you can have them use the share code feature to swap programs and get into the second part of this activity more quickly.
- Students do not need to run their programs in the Edison robots for this activity, but it can be helpful if they want to see what a program does in order to decide about the comments included.
- You may want to provide students with some best-practice rules used in commenting professionally. Alternatively, you may ask students to compare their answers with these ‘best practices’ and discuss any differences:
  - Comments should be easily understood – i.e. they should be ‘human readable’ (rather than being code snippets).
  - Comments should generally be at the top of the code they are explaining.

- Comments only need to be added if the section of code is not self-evident.

### Answer key

Question	Type	Sample answer	Marking notes
1	SE	<p><i>1. Comments should be kept pretty short</i></p> <p><i>2. Comments should not repeat what's already clear in the code</i></p> <p><i>3. Comments should always be included in tricky or confusing sections</i></p> <p><i>4. If the comment is because you aren't sure about something (e.g. for debugging), include a question mark in the comment</i></p>	<p>The rules students come up with are far less important than the process – you may want to have students explain how they came up with their rules to understand how they collaborated and their logic.</p>

## Unit 4: What if...

Explore selection and branching in computer programming through the key computational concepts of conditionals and events. Key computer programming skills, such as developing pseudocode, are introduced in this unit to help students further their problem-solving abilities as they unlock the Edison robots' various sensor capabilities. Students learn about algorithms and use this understanding to create programs enabling more autonomous behaviour from the robots. Conditionals, sensing, interrupts and event-based programming are brought to life through the physical computing activities in this unit.

### Learning objectives

Students will:

- be introduced to the computational thinking concept of conditional selection (i.e. branching)
- explore how inputs work with Edison using new blocks in the 'Control', 'Sensing' and 'Events' categories in EdScratch
- experiment using Edison's sensors and input capabilities in working programs
- combine the concepts of sequence, loops and selection into working programs in EdScratch
- develop a practical understanding of how to use pseudocode and comments to plan, track and debug programs
- be introduced to the idea of algorithms in computer programming and learn how algorithms differ from programs
- expand their understanding of robotic applications through projects using sensing and inputs

**Key ideas:** conditionals (i.e. selection and branching), events, sensing and sensors, pseudocode, algorithms

### Lessons and activities in this unit

This unit includes two lessons with a total of nine base activities and 15 extension activities.

#### Lesson 1: Conditionals

- [U4-1.1 Let's explore using conditionals](#)
  - [U4-1.1a Change it up: Robot error or human error?](#)
- [U4-1.2 Let's explore if statements](#)
- [U4-1.3 Let's explore if statements and sequence](#)
- [U4-1.4 Let's explore stacking and nesting if statements](#)
  - [U4-1.4a Challenge up: Build a pulley](#)

#### Lesson 2: Sensing

- [U4-2.1 Let's explore pseudocode](#)
  - [U4-2.1a Change it up: Find the answer](#)

- [U4-2.2 Let's explore Edison's line tracker](#)
  - o [U4-2.2a Change it up: Drive inside a border](#)
- [U4-2.3 Let's explore algorithms](#)
  - o [U4-2.3a Challenge up: There's more than one way to follow a line](#)
- [U4-2.4 Let's explore Edison's obstacle detection](#)
  - o [U4-2.4a Change it up: Faster, faster, smash?](#)
  - o [U4-2.4b Challenge up: If line, go right. If obstacle, go left](#)
  - o [U4-2.4c Change it up: Where is the obstacle?](#)
  - o [U4-2.4d Challenge up: 3D maze](#)
- [U4-2.5 Let's explore messaging with Edison](#)
  - o [U4-2.5a Change it up: Remote-controlled flag machine](#)
  - o [U4-2.5b Challenge up: Build and control the EdCrane](#)
  - o [U4-2.5c Challenge up: Firefighting water cannon](#)
  - o [U4-2.5d Challenge up: Semi-automated digger](#)
  - o [U4-2.5e Challenge up: Hazardous material removal](#)
  - o [U4-2.5f Challenge up: Homing pigeons](#)

## Lesson 1: Conditionals

Along with loops, conditionals are some of the most important building blocks needed to create meaningful code in any computer programming language. This lesson introduces this fundamental computational thinking concept, known as ‘conditional selection’ or ‘branching’. Developing a firm understanding of conditionals is critical for success in the unit’s second lesson, which dives into Edison’s sensors.

Students develop an understanding of how sequential programming and programming logic work in programs which use conditional structures. Conditional structures make multiple options possible inside a program – different things will happen depending on whether or not the condition is met. Both ‘until’ conditionals and ‘if statement’ structures, as well as these structures’ corresponding EdScratch blocks, are introduced in this lesson. Students create programs in EdScratch using previously explored condition inputs (specifically, keypad button presses and claps) with both ‘until’ and ‘if’ conditional structures to explore how conditions function in practice.

This lesson has a total of four base activities and two extension activities:

- [U4-1.1 Let’s explore using conditionals](#)
  - o [U4-1.1a Change it up: Robot error or human error?](#)
- [U4-1.2 Let’s explore if statements](#)
- [U4-1.3 Let’s explore if statements and sequence](#)
- [U4-1.4 Let’s explore stacking and nesting if statements](#)
  - o [U4-1.4a Challenge up: Build a pulley](#)

### Activity U4-1.1 Let’s explore using conditionals

<b>Activity size</b>	Large
<b>Delivery recommendations</b>	
<b>Resources needed</b>	Basic supplies set, worksheet U4-1.1

#### Overview

Conditionals, one of the most basic building blocks needed to start writing meaningful programs in any computer language, are introduced in this activity. Conditionals in code are just like conditionals in life: one thing is conditional on another thing happening first.

Although ‘if’ statements are the type of conditional structure most frequently used in coding, this activity uses the ‘until’ blocks in EdScratch to introduce the concept of conditionals. The ‘until’ blocks are used because the idea of an action happening ‘until’ a certain condition is met can be easier for students to grasp initially.

The worksheet first walks students through a program using an indefinite ‘repeat until’ loop, which serves as the perfect starting point for understanding the overall

concept of conditionals. Students then use different sample programs to explore the concept further, gaining practice using the diamond-shaped event conditions in different ‘until’ blocks. The final question in the worksheet asks students to try to identify a possible use of an ‘until *condition*’ structure in the technology they see in their daily lives, helping them to begin to connect what they are learning with the programming that surrounds us in the real world.

### Tips and tricks

- If students are struggling with the concept of conditionals, work through daily life examples that use conditionals. For example, the chore of washing dishes can be thought about as ‘keep washing dishes until they are all clean’ and putting away laundry as ‘keep folding the washing until the laundry basket is empty’.
- Using programs that have button presses as events inside the program can become confusing. Remind students that to run a program, they need to press the ‘play’ (triangle) button once. This initial press **ONLY** starts the program – it will not be considered a button press *inside* of the program. In other words, once the program is running, the robot views the current condition as ‘triangle button pressed? – NO’.
- The syntax of EdScratch uses ‘until’ rather than ‘while’ (which is common in many general-purpose coding languages), but the code structure performs the same conditional function. If you are teaching EdScratch along with another language that uses ‘while’ conditionals, you may want to explain this connection to students, noting that there is no functional difference between ‘while *condition* = true’ and ‘until *condition* = false’.
  - Example: (while  $x < 4$ ) is the same as (until  $x = 4$ ). In both cases, the condition breaks when  $x = 4$ .

### Answer key

Question	Type	Answer/Sample answer	Marking notes
1	EA	<i>The ‘operators’ and ‘events’ categories have blocks that I think can be used as conditional input parameters. I think this because they have diamond-shaped blocks and the ‘until’ blocks have diamond-shaped holes.</i>	Student answers should note the diamond-shape correlation at a minimum.
2	EA	<i>You need to press the ‘round’ button while the program is running. This gives the condition for the ‘repeat until’ loop and ends the loop. The program then moves on to the next block in order, which is the ‘beep’ block.</i>	Student answers should identify that the round button push fulfils the conditional and that the program then continues in normal sequential order.

3	EA	4	
4	SE	<i>One real-life example of a program that would be good to start with a 'wait until condition' block is a burglar alarm. The event condition would be a motion detection event. The program would tell the alarm to wait until motion was detected. Then, if motion is detected, it would sound an alarm.</i>	This question pushes students to try to tie the programming concepts of event conditions and the 'until' conditional structure in with the world around them. It is not expected that they will explain existing technology accurately – only that they attempt to apply their understanding to technology.

### Activity U4-1.1a Change it up: Robot error or human error?

<b>Activity size</b>	Medium
<b>Delivery recommendations</b>	<ul style="list-style-type: none"> <li>- Complete activity U4-1.1 prior to this activity</li> <li>- Recommend to practice debugging with conditionals</li> <li>- Strongly recommend if students are struggling with conditionals</li> </ul>
<b>Resources needed</b>	Basic supplies set, worksheet U4-1.1a

#### Overview

Students work through a program which uses sequence, loops and conditionals to uncover why the robot is not behaving the way the programmer expects. While this may appear to be just a debugging activity, it is actually more about looking at computational thinking in action. The activity walks students through an exercise of decomposing a program and its 'problems' into smaller parts. Students then work through the underlying logic of each issue. In doing so, students reinforce their understanding of how conditionals work in combination with other elements of code.

The idea that there are limits to a computer's capabilities is also introduced. While the issues in this activity both turn out to be 'human', knowing that it is possible that an issue is a limit of the robot's capabilities is valuable for students to keep in mind when working with more complex projects and programs.

#### Tips and tricks

- Using programs that have button presses as events inside the program can become confusing. Remind students that to run a program, they need to press the 'play' (triangle) button once. This press ONLY starts the program – it will

not be considered a button press *inside* of the program. In other words, once the program is running, the robot views the current condition as ‘triangle button pressed? – NO’.

- Problem-solving is a major part of coding. This activity can help students practice problem-solving to fix a ‘broken’ program without feeling as frustrated as they might if they had written the program themselves.
- Collaboration can make finding solutions in coding a lot easier. Consider having students work on this activity in pairs or groups.
- You can set-up this activity in a different way to make it a more explicit lesson on decomposition. Have students work in groups or as a whole class, first looking at the programmer’s comments. Have students identify and break out the two separate issues without the aid of the worksheet. Work through each issue in turn, breaking each problem down into smaller pieces as needed. Have students collaborate to try to solve each small chunk, then see how that solves the initial comments from the programmer as a whole.

### Answer key

Question	Type	Answer/Sample answer	Marking notes
1	EA	<i>The robot is spinning left slowly.</i>	
2	SE	<i>Yes, because that is what the first block inside the loop tells the robot to do.</i>	If students did not expect this behaviour, you might want to review how loops work.
3	EA	<i>The robot starts spinning right because the condition of the ‘spin left’ block has been met, and so the robot moves on to the next block in the program which is ‘spin right’.</i>	If students do not understand why it is moving left again, you may want to review conditionals.
4	SE	<i>The logical error is that the programmer expected that as soon as they pushed the round button, the loop would break and the robot would move on to the next line of code (the beep). However, that’s not how loops work. The program needs to have each line of code inside the loop complete, then return to the top of the loop to check if the loop condition has been met. If the condition has been met, then the program will move on. When the programmer pushed the round button, but then didn’t push the triangle button, the program was stuck on the first line of code inside the loop.</i>	Student answers should correctly identify the logical error.

5	SE	<i>Dear future-me,            First off, don't give up. Coding takes practice and problem-solving. Here are some other things you can try: a) Walk through the code line by line, following what each block is telling the robot to do. b) Look at where the problem is and test another program with just that bit of code, to see what's happening. c) Ask a friend to look at my program and see if they can spot the issue.</i>	Resilience is key to coding! You may want to remind students to include 'don't give up' in their list of suggestions!
---	----	---	---

### Activity U4-1.2 Let's explore if statements

<b>Activity size</b>	Medium
<b>Delivery recommendations</b>	
<b>Resources needed</b>	Worksheet U4-1.2, activity sheet U4-1

#### Overview

This offline-activity introduces students to both 'if' and if-else conditionals. The idea that programs can branch is explored, and students practice the concept using an offline 'treasure map' with if-else instructions. The idea that conditionals allow a computer to decide something is more evident when using 'if' statements than it is when using 'until' conditional code blocks. Branching becomes especially clear when using if-else conditionals. Practising the concept of branching offline helps students become more familiar with the idea and see how branching works in combination with other computational thinking processes, such as sequence and repetition.

#### Tips and tricks

- Some students may note that the 'if-then' pattern used in 'if' statements appears to be the same as causality. In a way, this is an accurate comparison. Once the code is written, the 'THEN' statement will run if the 'IF' condition is met. However, in code, there are not locked if/then rules. You may want to explain that students set the 'causality' by determining the conditions (the IF) and the conditional code (the THEN).
- The instructions to find the treasures are all written in a format that could be considered rough pseudocode. The concept of pseudocode is introduced in activity U4-2.1. You may choose to introduce the concept formally with this activity or reference the two activities as you use each.

### Answer key

Question	Type	Answer/Sample answer	Marking notes
1	SE	<i>IF my brother goes into my room, THEN I tell on him.</i>	Any if-then situation which makes sense is acceptable.
2	EA	<i>E</i>	
3	EA	<i>B</i>	
4	EA	<i>D</i>	

### Activity U4-1.3 Let's explore if statements and sequence

<b>Activity size</b>	Small
<b>Delivery recommendations</b>	
<b>Resources needed</b>	Basic supplies set, worksheet U4-1.3

#### Overview

Building on students' basic understanding of both 'if' and if-else conditionals, this activity explores how 'if' statements work in EdScratch programs in relation to sequence. Like all code, 'if' and if-else blocks follow fundamental computer logic, including executing in sequential order. When the program encounters 'if' statements, the blocks are resolved in step-by-step sequential order. However, because the code inside an 'if' block is conditional on that block's condition being met, it is possible for the program to skip the conditional code. Understanding this behaviour is crucial for students to fully comprehend conditionals and be able to apply the logic of conditions in their computational thinking.

This activity also examines if-else blocks in sequential programs in EdScratch. Understanding that only the 'if' or 'else' section of code will run is critical for students to be able to write and follow programs using this structure.

#### Tips and tricks

- If students are struggling to follow what is happening in a program with 'if' or if-else blocks, you may want to encourage them to add comments into their programs. Writing out notes in their own words can help students understand a program's conditional flow.
- Remind students that they can always run a program in EdScratch with Edison to check their thinking and to see how a program works.

### Answer key

Question	Type	Answer/Sample answer	Marking notes
1	EA	<i>The round button needs to be pressed for the 'if' condition to be met.</i>	
2	SE	<i>I couldn't get the conditional code to run, even when I pressed the round button. When I ran the program, it beeped, then spun 60 degrees to the right. It skipped the code inside the 'if' block.</i>	Students are unlikely to be able to get the conditional code to run as there is not enough time between when they press the triangle button to start the program and when the program is checking for the round button press for them to have the button press completed. If they can complete the button press in time, their answers should note that all code runs in sequence.
3	SE	<i>I think that the conditional code (the stuff inside the 'if' block) gets skipped if the condition is not met and the program moves on to the next item of code after the 'if' block in sequential order.</i>	Student answers will ideally note that conditional code is skipped if the condition is not met.
4	EA	<i>No, the robot would not beep. This is because the beep is inside the 'if' part of the block which will only run if the robot detects a round button press.</i>	
5	EA	<i>There are 3 actions the robot will always take: 1) wait .5 secs, 2) check to see if a round button press has occurred and 3) turn the left LED on.</i>	If students only identify 1) and 3), you may want to explain that the if-statement condition check always occurs, regardless of the outcome.

### Activity U4-1.4 Let's explore stacking and nesting if statements

<b>Activity size</b>	Large
<b>Delivery recommendations</b>	
<b>Resources needed</b>	Basic supplies set, worksheet U4-1.4

#### Overview

Using conditionals effectively allows students to create meaningful programs and tap into Edison's powerful sensors. This means that students need to have a clear

understanding of how conditionals, especially ‘if’ and if-else structures, work in programs, including when these structures are stacked or nested.

Being able to follow the flow of a program which contains conditionals and loop control structures can seem like quite an imposing task at first. This activity challenges students to work through different programs, following the flow of the code to understand how the programs work. Putting their computational thinking skills to work, students will decompose programs to understand what will happen in different conditional situations. They then see the power of branching programs nested in loops by writing and testing a clap-controlled driving program.

### Tips and tricks

- Nesting sensor-based conditionals in loops is a very common way to use Edison’s sensors. You may want to spend extra time on this activity to ensure students are comfortable using nested and stacked ‘if’ statements.
- For the clap-controlled driving program: the robots may struggle to detect sounds when there is a high level of background noise. Having students tap a finger near the sound sensor on their Edison will create the same effect as clapping.
- Having students add comments to their clap-controlled driving program can help ensure that students understand what is happening inside the code.
- The clap-controlled driving program in this activity functions the same way as the clap-controlled driving program activated by the clap-controlled driving barcode program. You may choose to have students revisit that barcode to see this for themselves.

### Answer key

Question	Type	Answer/Sample answer	Marking notes
1	EA	<i>If you never press the round button, the condition will never be met, so the ‘else’ code will run each repetition of the loop. The robot will spin right 60 degrees, then beep, and repeat those actions forever.</i>	
2	EA	<i>If you never press the round button, the condition will never be met, so the ‘if’ code will be skipped each repetition of the loop. The robot will beep each loop forever.</i>	
3	EA	<i>To get the robot to drive backwards, first, you need to press the round button while the program is running. The robot will wait for .5 seconds then check if the triangle button has been pressed. Since it has not, it will run the ‘else’ code and drive backwards.</i>	

4	SE	<i>I think that the program uses the 'forever' block so that the program will keep running loop after loop as long as you want. If you didn't have the forever block, the program would tell the robot to wait for a clap, and then check for a second clap, but it would only do this one time. Once it detected either one or two claps and moved accordingly, the program would end.</i>	Students should identify that without the forever block, the program would only run once.
---	----	---	---

### Activity U4-1.4a Challenge up: Build a pulley

<b>Activity size</b>	Project
<b>Delivery recommendations</b>	Recommend as a capstone activity following the completion of the Let's Explore activities from unit 4, lesson 1
<b>Resources needed</b>	Basic supplies set, worksheet U4-1.4a, maker-space/crafting supplies, EdCreate kits/LEGO bricks

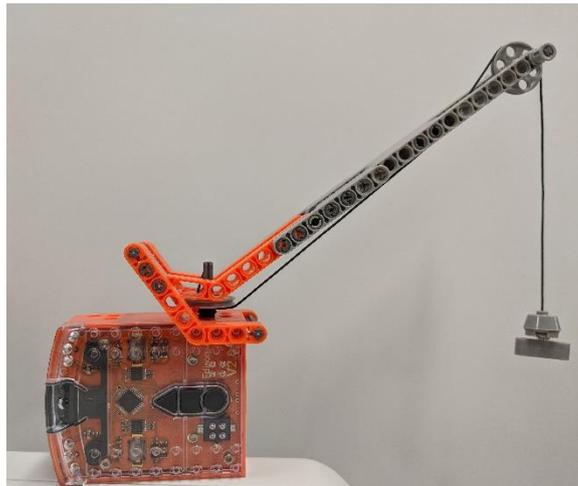
#### Overview

This open-ended engineering and programming challenge has students apply their understanding of 'if' statements to create a working pulley system using Edison.

#### Tips and tricks

- This activity works well in pairs or small groups.
- By design, this challenge offers minimal instructions and only a few hints. This is to encourage creativity and experimentation. You may choose to add more structure (for example, working through the physical design process with students) to the activity as best fits your students' needs.
- This challenge can be run as an engineering design challenge. You can read more about teaching engineering design at <https://meetedison.com/teach-engineering-design-with-edcreate/>
- There is no 'right' answer to what the pulley build and program should look like, but here is what one sample solution looks like:

- Physical design: (adapted from the [EdCrane EdBuild](#)):



- EdScratch pulley program: <https://www.edscratchapp.com?share=wBJRrQbj>

### Answer key

Question	Type	Sample answer	Marking notes
1	SE	<p><i>Regarding the program, my first version had the 'if' statements nested the wrong way. Both were in the 'if' part of the if-else block, so the only way to move the pulley backwards was to push the round button and THEN the triangle button. I got a classmate to look at my program, and she spotted this error. Once I moved the triangle button 'if' block to the 'else' section of the if-else block, it worked!</i></p> <p><i>The best part of this project was working with other people. When I got stuck and frustrated, I was able to get help from classmates, and then we fixed the problems together! That was a lot of fun.</i></p>	<p>Any student experience is acceptable. You may also choose to have students provide photos or diagrams of their pulley designs and submit their program solutions.</p>

## Lesson 2: Sensing

This relatively large lesson expands students' understanding of conditionals, loops, sequence, and interrupts by using Edison's various sensors in programs. Students further explore how inputs work in Edison using new blocks in the 'Control', 'Sensing' and 'Events' categories in EdScratch.

In addition to discovering how Edison's individual sensors work and can be programmed, this lesson uses Edison's sensor capabilities to explore key computer science concepts and skills in practical applications. This lesson introduces the computer science practice of pseudocode, the idea of algorithms, and the important but often misunderstood concept of how algorithms differ from programs. Students practice using pseudocode and algorithms both to plan out programs and to help in debugging their work. By experimenting with Edison's sensors and input capabilities in working programs, students expand their understanding of robotics applications and autonomous robotic behaviours in real-world contexts.

This lesson has a total of five base activities and 13 extension activities:

- [U4-2.1 Let's explore pseudocode](#)
  - o [U4-2.1a Change it up: Find the answer](#)
- [U4-2.2 Let's explore Edison's line tracker](#)
  - o [U4-2.2a Change it up: Drive inside a border](#)
- [U4-2.3 Let's explore algorithms](#)
  - o [U4-2.3a Challenge up: There's more than one way to follow a line](#)
- [U4-2.4 Let's explore Edison's obstacle detection](#)
  - o [U4-2.4a Change it up: Faster, faster, smash?](#)
  - o [U4-2.4b Challenge up: If line, go right. If obstacle, go left](#)
  - o [U4-2.4c Change it up: Where is the obstacle?](#)
  - o [U4-2.4d Challenge up: 3D maze](#)
- [U4-2.5 Let's explore messaging with Edison](#)
  - o [U4-2.5a Change it up: Remote-controlled flag machine](#)
  - o [U4-2.5b Challenge up: Build and control the EdCrane](#)
  - o [U4-2.5c Challenge up: Firefighting water cannon](#)
  - o [U4-2.5d Challenge up: Semi-automated digger](#)
  - o [U4-2.5e Challenge up: Hazardous material removal](#)
  - o [U4-2.5f Challenge up: Homing pigeons](#)

### Activity U4-2.1 Let's explore pseudocode

<b>Activity size</b>	Small
<b>Delivery recommendations</b>	
<b>Resources needed</b>	Worksheet U4-2.1, activity sheet U4-2

#### Overview

This offline activity introduces pseudocode, a helpful tool for tackling the important task of planning programs. Planning out programs effectively is an important skill for students to develop when mastering computational thinking. Being able to plan effectively requires students to apply the computational thinking concepts that they have learned (including sequence, repetition and conditionals) into a logical flow. Pseudocode is one of the most efficient tools for planning computer programs and is used by many professional programmers.

Developing a level of comfort with pseudocode allows students to plan out programs and algorithms with confidence. Working from a plan makes coding faster and debugging and problem-solving more effective. The ability to create program plans in pseudocode also allows students to concentrate on the logic their program needs without getting caught up in minor details or syntax. This will allow students far more clarity when planning programs using sensors, data, or complex control structures. Competency with pseudocode also improves students' abilities to apply their computational understanding to other programming languages, including text-based languages.

#### Tips and tricks

- The pseudocode in the worksheet questions uses indentation to indicate when actions are inside of control structures. Following the left-most alignment of the pseudocode in the worksheet questions will help show where a loop or 'if' statement ends and the next command after that control structure begins.
- Remind students that indented pseudocode is inside a loop or 'if' statement. In cases where there are nested conditionals or loops, the pseudocode will show this 'nesting' by being indented more than one level.
- The instructions to find the treasures in activity U4-1.2 are written in a format that could be considered rough pseudocode. You may choose to reference activity U4-1.2 as another example of pseudocode or use the two activities in conjunction.

#### Answer key

Question	Type	Answer	Marking notes
1	EA	<i>The letter E</i>	
2	EA	<i>The number 4</i>	
3	EA	<i>The number 4</i>	

### Activity U4-2.1a Change it up: Find the answer

<b>Activity size</b>	Small
<b>Delivery recommendations</b>	Complete activity U4-2.1 prior to this activity
<b>Resources needed</b>	- Worksheet U4-2.1a, activity sheet U4-2 - <i>Optional:</i> computers or tablets to look at EdScratch

#### Overview

Designed for use in conjunction with activity U4-2.1, this activity builds students' understanding and competency with pseudocode. Students practice writing pseudocode instructions, then reading and following pseudocode written by a classmate.

#### Tips and tricks

- While this activity requires students to work with at least one partner, you can also have students work in small groups divided into teams. Each team can work together to plan, write and test the pseudocode before exchanging with the opposite team.

#### Answer key

Question	Type	Sample answer	Marking notes
1	SE	<i>Start on 8 facing north repeat 3 times     forwards until letter     right 90 repeat 2 times     forwards 2     if animal         right 90 degrees     else         left 90 degrees forwards until letter  Lands on the answer of: C</i>	You may choose to have students also submit where their program is meant to end to check accuracy.

### Activity U4-2.2 Let's explore Edison's line tracker

<b>Activity size</b>	Medium
<b>Delivery recommendations</b>	
<b>Resources needed</b>	- Basic supplies set, worksheet U4-2.2, activity sheet U4-3 - <i>Optional:</i> EdMats, supplies for making your own lines

#### Overview

This two-part activity kick-starts students' exploration of using Edison's line tracker in EdScratch programs. Students first become familiar with the Edison robot's line tracking sensor technology, learning the fundamentals of how the sensor works. By

experimenting with the line tracker's red LED on different coloured surfaces, students learn how the robot can use the sensor to detect whether it is driving on a reflective or non-reflective surface.

Students then practice using the line tracking sensor in an EdScratch program to get the robot to drive until it detects a black (i.e. non-reflective) line.

### Tips and tricks

- If you want to explore more about how Edison's line tracking sensor works, consider having a breakout session to explore how light reflection and absorption works, including for coloured light.
- You can also use EdMats or make your own test space to run the 'drive until black' program.
- Remind students to develop the habit of checking the bug box in EdScratch. This can be especially helpful when working with sensors.

### Answer key

Question	Type	Answer	Marking notes
1	EA	<i>A white surface reflects more light back to Edison than a black surface reflects. I think this because the spot from the red LED is brighter on a white surface (compared to black) and that means more of the light is being reflected (rather than absorbed).</i>	The light will appear brighter on a white surface. If students note black, consider reviewing this with them.
2	EA	<b>Colour</b>	<b>Reflective or non-reflective?</b>
		Red	reflective
		Blue	non-reflective
		Green	non-reflective

### Activity U4-2.2a Change it up: Drive inside a border

<b>Activity size</b>	Medium
<b>Delivery recommendations</b>	Complete activity U4-2.2 prior to this activity
<b>Resources needed</b>	- Basic supplies set, worksheet U4-2.2a, activity sheet U4-4 - <i>Optional:</i> EdMats, supplies for making your own lines/borders

### Overview

Students put their understanding of pseudocode and Edison's line tracker to the test in order to create a program that gets Edison to drive inside a black border. By first planning out their program, students apply algorithmic thinking to how the line tracking sensor can be used to create a program where Edison will not cross a black

line. Students then translate their plan into EdScratch and test their program with the robot.

### Tips and tricks

- You can also have students use EdMats or make their own test space to run their ‘drive inside a black border’ programs.
- The barcode program ‘bounce in borders’ has Edison drive inside a black border. You may choose to have students run the barcode program and compare the robot’s behaviour when using the barcode program to the behaviour displayed when running their ‘bounce in borders’ style program.
- Encourage students to work out the source of any errors they encounter. Two common logical errors students may make when creating this program are: 1) forgetting to turn on the line detection sensor, or 2) forgetting to use an indefinite loop.
- This activity can be used as an extension activity following activity U4-2.3 *Let’s explore algorithms*, emphasising that students can create an algorithm for getting Edison to drive inside any border.

### Answer key

Question	Type	Sample answer	Marking notes
1	SE	<i>turn on line detection forever drive forwards until non-reflective drive backwards turn</i>	Student answers may not be completely correct compared to the final program – that’s okay.

### Activity U4-2.3 Let’s explore algorithms

<b>Activity size</b>	Medium
<b>Delivery recommendations</b>	
<b>Resources needed</b>	Basic supplies set, worksheet U4-2.3, activity sheet U4-4, supplies for making your own lines/borders and/or EdMats

### Overview

The concept of algorithms is introduced in this lesson along with the important but often misunderstood concept of how algorithms differ from programs. Students learn that algorithms are a tool for solving sets of problems and can guide the logic used in computer programs. They explore an algorithm for getting Edison to follow any black line using the robot’s line tracking sensor and see how the logic of the algorithm translates into an EdScratch program.

Learning that algorithms can inform computer programs, but that not all programs are algorithms, is an important concept to enable students to be able to practice so-

called 'algorithmic thinking'. Understanding that some problems need a specific, dedicated program to solve while others can be solved using an algorithm is an important skill for students to develop.

### Tips and tricks

- For the line tracking program to work, the difference between the dark and light surfaces needs to be easily understood by the robot, and there needs to be enough white space between any curves in the line. The lines that students make for Edison to follow need to be dark (e.g. black), approximately 1.5cm (0.6 inches) wide, and on a white background. Alternatively, they can use an EdMat.
- Make sure students start by placing Edison with the line tracking sensor on white. The robot can start near the black line but not on top of it.
- You may choose to have students run the barcode program 'follow a line' to see the same algorithm from the worksheet at work in that barcode program.
- Activity U4-2.2a can be used as an extension activity following this activity. You can demonstrate to students that they can create an algorithm for getting Edison to drive inside any black border.

### Answer key

Question	Type	Answer/Sample answer	Marking notes
1	EA	<i>The robot 'waddles' along the edge of the line. This is because the algorithm says that the robot should drive forwards to the left until the line tracking sensor is on a non-reflective (black) surface. That pushes the sensor along until it finds a black line to the left. Then the logic says that the robot should drive forwards to the right until the line tracking sensor is on a reflective (white) surface. That pushes the sensor off black, onto white. The robot alternates this behaviour, moving back and forth on the edge of the line. This makes the robot 'waddle' left and right as it goes along the line.</i>	Students should identify the robot's waddle is caused by the program logic going back and forth between the two 'until' blocks.
2	SE	<i>Edison was able to follow my line except in one spot where the line curved very sharply. The robot kept missing the curve and catching the other part of the line. I think this happened because the two parts of the line were really close together, so the robot would just find a non-reflective spot and start following it, even though it wasn't the part of the line I wanted it to follow.</i>	

### Activity U4-2.3a Challenge up: There's more than one way to follow a line

<b>Activity size</b>	Large
<b>Delivery recommendations</b>	Complete activity U4-2.3 prior to this activity
<b>Resources needed</b>	<ul style="list-style-type: none"> <li>- Basic supplies set, worksheet U4-2.3a, activity sheet U4-4</li> <li>- <i>Optional:</i> EdMats, supplies for making your own lines/borders</li> </ul>

#### Overview

The idea that there is no one 'right' way to program is brought to life in this activity. Students look at the algorithm for getting Edison to follow any black line using the robot's line tracking sensor and use its logic to come up with multiple programs. This semi open-ended programming challenge asks students to think about different ways the same underlying logic can be manifested in an EdScratch program. Having an appreciation that there are many ways to program a solution is an important skill for students to develop and will make them better able to collaborate effectively on coding projects. Thinking of multiple ways to apply the algorithm's logic also challenges students to apply their understanding of conditionals, interrupts and sensor inputs into working programs.

#### Tips and tricks

- Encourage students to work through the logic of the algorithm looking at different blocks in EdScratch. Testing things that do not end up working is fine! Explore why an attempt doesn't work and iterate to develop a program that does.
- For the line tracking program to work, the difference between the dark and light surfaces needs to be easily understood by the robot, and there needs to be enough white space between any curves in the line. The lines that students make for Edison to follow need to be dark (e.g. black), approximately 1.5cm (0.6 inches) wide, and on a white background. Alternatively, they can use an EdMat.
- Make sure students start by placing Edison with the line tracking sensor on white. The robot can start near the black line but not on top of it.
- You may want to have students compare answers with each other to see all the different variations of programs that are possible.

### Answer key

Question	Type	Sample answer	Marking notes
1	RC	<i>Program 1: This program uses an if-else statement to get Edison to follow a line</i> <a href="https://www.edscratchapp.com?share=7bx9x9DZ">https://www.edscratchapp.com?share=7bx9x9DZ</a>	Any program that gets Edison to follow any line is acceptable.
2	RC	<i>Program 2: This program uses events to get Edison to follow a line</i> <a href="https://www.edscratchapp.com?share=pDmX9Jbq">https://www.edscratchapp.com?share=pDmX9Jbq</a>	Any program that gets Edison to follow any line is acceptable.

### Activity U4-2.4 Let's explore Edison's obstacle detection

<b>Activity size</b>	Large
<b>Delivery recommendations</b>	
<b>Resources needed</b>	Basic supplies set, worksheet U4-2.4, objects for making obstacles

#### Overview

This three-part activity kicks off students' exploration of using obstacle detection with Edison in EdScratch programs. Students first become familiar with the Edison robot's infrared light sensor technology, learning the fundamentals of how the sensor works and how it can be used to detect obstacles. A basic program using obstacle detection is also introduced to explain how the sensor can be used to detect obstacles in EdScratch programs. Finally, students practice using obstacle detection by creating an algorithm to detect and avoid obstacles, then translate the logic of the algorithm into an EdScratch program.

#### Tips and tricks

- If any of the Edison robots are not detecting obstacles or are reacting to obstacles only when they are very close to the obstacle, you may need to calibrate the obstacle detection in that robot. Use the barcode and instructions in Appendix 2 of this guide to calibrate a robot's obstacle detection.
- Obstacles need to be opaque but not too dark (e.g. not black) and at least as tall as Edison for the robot to detect them.
- You may choose to have students run the barcode program 'avoid obstacles' to see the same algorithmic logic at work in that program.
- You may choose to have a breakout session on the electromagnetic spectrum and where IR and visible light fall within the spectrum in conjunction with this activity.

### Answer key

Question	Type	Answer/Sample answer	Marking notes
1	EA	turn obstacle detection beam on loop forever drive forwards if obstacle detected anywhere drive backwards turn	Student answers should have the basic logic of the sample answer but can be explained differently.
2	SE	<i>When I first went to write my algorithm, I had to think about what behaviour the robot needed to take to avoid any obstacle. This stumped me initially. I picked up my robot and pretended to drive it towards different obstacles on the desk, moving to avoid each one. By doing this a few times, I realised that if the robot backs up and then turns away, it will always avoid the obstacle.</i>	Having students articulate problems they encounter as well as the actions they took to resolve the issues helps bring explicit attention to their problem-solving skills.

### Activity U4-2.4a Change it up: Faster, faster, smash?

<b>Activity size</b>	Small
<b>Delivery recommendations</b>	Complete activity U4-2.4 prior to this activity
<b>Resources needed</b>	Basic supplies set, worksheet U4-2.4a, objects for making obstacles

#### Overview

Students practice using obstacle detection in working programs with their Edison robots while exploring the ideas of hardware limitations and trade-offs in programming. An Edison robot's obstacle detection relies on the robot being able to react to reflected IR light bouncing back to it from objects. When the robot travels at fast speeds, the time it takes for the light to reflect and be detected can outstrip the time it takes for the robot to smash into the obstacle. Testing different speeds in an obstacle detection program lets students explore this trade-off between speed and accuracy first-hand.

#### Tips and tricks

- If any of the Edison robots are not detecting obstacles or are reacting to obstacles only when they are very close even at low speeds, you may need to calibrate the obstacle detection of that robot. Use the barcode and instructions in Appendix 2 of this guide to calibrate a robot's obstacle detection.
- Obstacles need to be opaque but not too dark (e.g. not black) and at least as tall as Edison for the robot to detect them.

### Answer key

Question	Type	Sample answer	Marking notes
1	SE	<i>Edison cannot detect the obstacle in time and crashes into it.</i>	Students should find that the robot is unable to detect and react to obstacles if Edison is moving very fast.
1	SE	<i>The faster Edison drives, the less time there is for the robot's obstacle detection process to take place. The less time for the process, the worse the robot's ability to detect obstacles.</i>	Students should note the base reasoning in the sample answer in their answer.

### Activity U4-2.4b Challenge up: If line, go right. If obstacle, go left

<b>Activity size</b>	Medium
<b>Delivery recommendations</b>	- Complete activity U4-2.2 prior to this activity - Complete activity U4-2.4 prior to this activity
<b>Resources needed</b>	- Basic supplies set, worksheet U4-2.4b, activity sheet U4-5, objects for making obstacles - <i>Optional:</i> supplies for making your own sensor-driven 'mazes'

#### Overview

Designed to be used after students have explored both Edison's obstacle detection beam and line tracking sensor, this programming challenge gets students to apply computational thinking to use the sensors to solve a 'grid maze'. Students explore using multiple sensors in a single program. This task also requires that they combine multiple computational elements, including sequence, conditionals, input-outputs, and repetition.

The activity also lays out the option for students to design their own physical setup to be solved using the same basic idea of 'if line, do X. If obstacle, do Y.'

#### Tips and tricks

- If any of the Edison robots are not detecting obstacles or are reacting to obstacles only when they are very close even at low speeds, you may need to calibrate the obstacle detection of that robot. Use the barcode and instructions in Appendix 2 of this guide to calibrate a robot's obstacle detection.
- Obstacles need to be opaque but not too dark (e.g. not black) and at least as tall as Edison for the robot to detect them.
- An example solution using the activity sheet can be seen at <https://www.edscratchapp.com?share=8DQGkG0B>

### Activity U4-2.4c Change it up: Where is the obstacle?

<b>Activity size</b>	Small
<b>Delivery recommendations</b>	Complete activity U4-2.4 prior to this activity
<b>Resources needed</b>	Basic supplies set, worksheet U4-2.4c, objects for making obstacles

#### Overview

This activity reinforces students' understanding of how Edison's infrared sensor can be used to detect objects and further explores Edison's ability to detect the location of an obstacle relative to the robot. With the requirement to use interrupts from the 'events' category of blocks in EdScratch, students use their creativity to make three different subroutines for the robot to run, outputting different reactions depending on the location of the obstacle it detects.

#### Tips and tricks

- If any of the Edison robots are not detecting obstacles or are reacting to obstacles only when they are very close, you may need to calibrate the obstacle detection of that robot. Use the barcode and instructions in Appendix 2 of this guide to calibrate a robot's obstacle detection.
- One of the best obstacles for this activity is actually students' own hands! Have students face their Edison away from them, then test the program by putting a hand down in front of Edison to the right, then repeat on the left. Finally, have them put their hand right in front of the robot.
- An example solution to this activity can be seen at <https://www.edscratchapp.com?share=Eb2X7wYq>

### Activity U4-2.4d Challenge up: 3D maze

<b>Activity size</b>	Project
<b>Delivery recommendations</b>	Complete activity U4-2.4 prior to this activity
<b>Resources needed</b>	Basic supplies set, worksheet U4-2.4d, supplies for building the 3D maze

#### Overview

By creating and then solving their own three-dimensional mazes, students apply computational thinking and their understanding of Edison's infrared-driven obstacle detection to this 'real-world' project, getting Edison to operate like a driverless car. Just as driverless car technology is yet to be perfected, getting Edison to drive autonomously through a 3D maze using obstacle detection is not without pitfalls. This project is a great challenge for students to practice problem-solving, decomposition and collaboration.

### Tips and tricks

- A great example of a 3D maze that can be readily solved using obstacle detection can be seen in the video *Meet Edison - Autonomous Robotics Maze Challenge #1* available at [https://www.youtube.com/watch?v=caVNQYKr- 4](https://www.youtube.com/watch?v=caVNQYKr-4)
- Remind students of the trade-off between speed and accuracy with obstacle detection: the faster Edison drives, the less time there is for the robot's obstacle detection process to take place. The less time for the process, the worse the robot's ability to detect obstacles.
- Edison's obstacle detection uses infrared light which, when Edison is moving, can be far from perfect. As the robot moves, its position relative to objects changes. 'Old' infrared light bouncing off a wall can be detected from a new position, causing the robot to react in a way the programmer might not expect or want. Experimenting with using location-based obstacle detection events in different ways will let students create different options to get their robots autonomously through their maze.

### Activity U4-2.5 Let's explore messaging with Edison

<b>Activity size</b>	Medium
<b>Delivery recommendations</b>	It is recommended to complete activity U4-2.4 prior to this activity
<b>Resources needed</b>	Basic supplies set, worksheet U4-2.5

#### Overview

This activity introduces students to using the Edison robot's infrared sensor to send and receive IR messages. The robot's IR receiver can be used in multiple ways: to detect IR light reflected back to the robot from objects as part of Edison's obstacle detection beam, to detect IR messages from other Edison robots, and to detect IR messages from paired TV/DVD remote controls. Students learn the basics of how IR messages work in this activity, practising sending and receiving messages between multiple Edison robots.

### Tips and tricks

- This activity requires sets of at least two robots, so students should work in pairs or groups. The activity can also be done as a whole class activity, with one robot sending out a message for all the others to detect.
- Edison's messaging uses infrared which has a limited range, similar to that of a TV remote control. If students are struggling to get a 'receiving' robot to detect the message, have the students move the 'receiving' robots in closer to the 'sending' robot.
- If multiple groups are running this activity near each other, students may encounter 'cross-talk' with robots reacting to the messages being sent out by other groups.

- To learn more about TV/DVD remote controls and Edison, consider using activity U1-1.2c *Change it up: TV remote control barcodes* and/or activity U4-2.5a *Change it up: Remote-controlled flag machine*

#### Answer key

Question	Type	Answer	Marking notes
1	EA	<i>The 'send IR message' block is in the LEDs category in EdScratch because it is an output that uses Edison's infrared LEDs.</i>	

#### Activity U4-2.5a Change it up: Remote-controlled flag machine

<b>Activity size</b>	Project
<b>Delivery recommendations</b>	- Complete activity U4-2.5 prior to this activity - Activities U4-2.5a, U4-2.5b, U4-2.5c, U4-2.5d and U4-2.5e are all robotics build projects using IR messaging. Consider using at least one.
<b>Resources needed</b>	Basic supplies set, worksheet U4-2.5a, activity sheet U4-6, TV/DVD remote controls, maker-space supplies for creating the flags, EdCreate kits and/or other supplies for attaching the flags

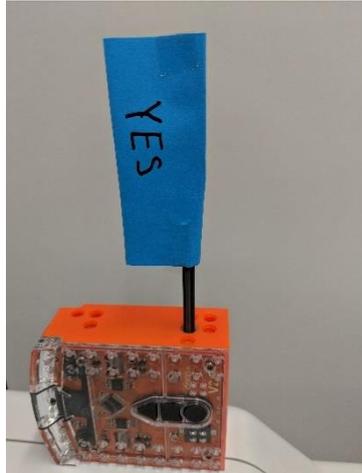
#### Overview

Students learn how to use Edison's programmable TV remote codes in this activity and then apply this knowledge into a robotics build project to create a remote-controlled flag machine. This semi open-ended project challenges students to create a physical computing solution with real-world application, applying their understanding of coding concepts, including conditionals and sensing.

#### Tips and tricks

- Students are welcome to build a flag machine in a different way to the project description in this activity or to come up with their own remote code detecting creation.
- You may want to familiarise students with using TV/DVD remote controls and Edison using activity U1-1.2c *Change it up: TV remote control barcodes* prior to this activity.
- A variation of this activity using IR messaging with variables is included in Unit 5, activity U5-1.4b *Challenge up: Edison-controlled flag machine*. You may choose to reference these two activities when using either or run them as two programming challenges for the one build design.
- There is no 'right' answer, but here is what one sample solution looks like:

- Physical design:



- Program: <https://www.edscratchapp.com?share=RDaeaaDV>

### Activity U4-2.5b Challenge up: Build and control the EdCrane

<b>Activity size</b>	Project
<b>Delivery recommendations</b>	- Complete activity U4-2.5 prior to this activity - Activities U4-2.5a, U4-2.5b, U4-2.5c, U4-2.5d and U4-2.5e are all robotics build projects using IR messaging. Consider using at least one.
<b>Resources needed</b>	Basic supplies set, worksheet U4-2.5b, activity sheet U4-6, TV/DVD remote controls, 1 EdCreate kit per EdCrane, EdCrane build instructions set (available at <a href="https://meetiedison.com/content/EdCreate/EdBuild-EdCrane-instructions.pdf">https://meetiedison.com/content/EdCreate/EdBuild-EdCrane-instructions.pdf</a> )

#### Overview

Using Edison to create programmable robotic builds is one of the most exciting things students can do with the robots. The EdBuild projects using the pre-set instructions and EdCreate kits are an excellent way to give students exposure to interactive engineering.

Like other EdCreate EdBuilds, the EdCrane responds to a TV or DVD remote control. However, the EdCrane uses the programmable TV remote codes, not the pre-set action barcodes. The EdCrane uses remote control commands in combination with a downloaded program from EdScratch, telling the EdCrane what action to perform when a specific remote code is detected.

#### Tips and tricks

- This project is ideal for pair or group work.
- Students may find it easiest to first lay out all of the EdCreate pieces onto their work surface and organise the parts into groups of the same piece type

and colour. This can help students identify and use the correct pieces as they work through the build.

- It is strongly recommended that the magnetic hooks be set up in advance of the build with adult supervision to ensure that these are set up correctly. Once the hook is set up the first time, it can be used as a single part over and over in the EdCrane build.
- The magnetic hook's spool can spin both clockwise and counter-clockwise. Whenever the string of the hook is fully extended out from the spool (i.e. the string is released all the way), whichever way the hook starts to wind will become the 'raise hook' direction. The opposite direction will be the 'lower hook' direction.
- Remind students that they must press the play (triangle) button on the Edison robot to run their downloaded program and override the default barcode behaviours.
- If multiple students are running programs using the remote-control codes in close physical proximity to each other, they may experience 'cross-talk' where one robot receives and reacts to the remote-control code from a different group. Try spacing students away from each other to minimise the issue.
- Students can download their EdScratch programs into the EdCrane before or after they build the crane, but they will need to pair the robot with a remote control using the barcodes before they build.
- The EdCrane build instructions also have programming instructions, including a link to a share-code. You may opt to have students use this code, modifying the program to match their plan and adjusting inputs as needed. Alternatively, you can give students a modified set of the EdCrane build instructions without the share-code and have students attempt to program the solution on their own. The worksheet assumes students do NOT have the share-code. The share-code can be seen at

<https://www.edscratchapp.com?share=5DMQ3XDw>

### Answer key

Question	Type	Sample answer			Marking notes
1	SE	<b>EdCrane action</b>	<b>Remote code</b>	<b>Remote control button</b>	
		Spin the magnetic spool clockwise	1	Volume up	
		Spin the magnetic spool counter-clockwise	2	Volume down	
		Spin the crane clockwise	3	Keypad right	
		Spin the crane counter-clockwise	4	Keypad left	

### Activity U4-2.5c Challenge up: Firefighting water cannon

<b>Activity size</b>	Project
<b>Delivery recommendations</b>	- Complete activity U4-2.5 prior to this activity - Activities U4-2.5a, U4-2.5b, U4-2.5c, U4-2.5d and U4-2.5e are all robotics build projects using IR messaging. Consider using at least one.
<b>Resources needed</b>	Basic supplies set, worksheet U4-2.5c, 1 EdCreate kit per EdTank, 2 Edison robots per EdTank, EdTank build instructions set (available at <a href="https://meetiedison.com/content/EdCreate/EdBuild-EdTank-instructions.pdf">https://meetiedison.com/content/EdCreate/EdBuild-EdTank-instructions.pdf</a> ), supplies for marking out the situation set-up

#### Overview

This programming challenge asks students to apply their understanding of coding concepts, including conditionals and sensing, to a real-world application. The EdBuild programming challenge projects use the pre-set instructions and EdCreate kits to create whichever EdBuild is needed for the scenario. This scenario uses the EdTank. By using the EdTank as the base for their firefighting robots, students do not need to design the engineering component and can concentrate on the programming component of their solutions.

Students must design two programs which will work in conjunction with each other in order to control the two robots in the EdTank and solve the firefighting scenario.

#### Tips and tricks

- This project is ideal for pair or group work.
- The building challenge in both this activity and activity U1-1.2e *Challenge up: Build and control the EdTank* is the same.
- Students may find it easiest to first lay out all of the EdCreate pieces onto their work surface and organise the parts into groups of the same piece type and colour. This can help students identify and use the correct pieces as they work through the build.
- You will need to reset the cannon and reload a rubber band each time you fire the cannon. This should be done manually to ensure the new band is loaded properly and that the firing 'pin' is pushed completely back into the starting location.
- For best results, use the orange bands that come with your EdCreate kit in the cannon.
- Just like many real tanks, the EdTank's design means it will be slow to turn left or right. When the EdCreate tracks are brand new, they may be extra grippy, which will make the EdTank turn even slower. You can reduce the grip by removing the tracks and lightly dusting them with talcum powder. Be sure to knock any excess powder off the tracks before putting them back onto the EdTank.

- Additional information about this programming project can be found in the EdCreate teaching guide at <https://meetiedison.com/content/EdCreate/EdCreate-teachers-guide.pdf>
- There is no 'right' answer, and solutions will vary depending on student-created set-ups. However, here is what one sample solution looks like <https://www.edscratchapp.com?share=80vBNQ0d>

### Activity U4-2.5d Challenge up: Semi-automated digger

<b>Activity size</b>	Project
<b>Delivery recommendations</b>	<ul style="list-style-type: none"> <li>- Complete activity U4-2.5 prior to this activity</li> <li>- Use activity U4-2.5a to introduce using remote control codes in programs prior to this activity</li> <li>- Activities U4-2.5a, U4-2.5b, U4-2.5c, U4-2.5d and U4-2.5e are all robotics build projects using IR messaging. Consider using at least one.</li> </ul>
<b>Resources needed</b>	Basic supplies set, worksheet U4-2.5d, activity sheet U4-6, TV/DVD remote controls, 1 EdCreate kit per EdDigger, 2 Edison robots per EdDigger, EdDigger build instructions set (available at <a href="https://meetiedison.com/content/EdCreate/EdBuild-EdDigger-instructions.pdf">https://meetiedison.com/content/EdCreate/EdBuild-EdDigger-instructions.pdf</a> ), supplies for creating the situation set-up

#### Overview

This programming challenge asks students to apply their understanding of coding concepts, including conditionals and sensing, to a real-world application. The EdBuild programming challenge projects use the pre-set instructions and EdCreate kits to create whichever EdBuild is needed for the scenario. This scenario uses the EdDigger. By using the EdDigger as the base for their semi-automated digger robots, students do not need to design the engineering component and can concentrate on the programming component of their solutions.

Students must design two programs which will work in conjunction with each other in order to control the two robots in the EdDigger and solve the scenario.

#### Tips and tricks

- This project is ideal for pair or group work.
- The building challenge in both this activity and activity U1-1.2f *Challenge up: Build and control the EdDigger* is the same.
- Students may find it easiest to first lay out all of the EdCreate pieces onto their work surface and organise the parts into groups of the same piece type and colour. This can help students identify and use the correct pieces as they work through the build.

- In this build, the top robot connects to the bottom robot on the third row of studs from the front of the bottom robot. Thus, the top robot overhangs off the back of the bottom robot by approximately 2 cm.
- Additional information about this programming project can be found in the EdCreate teaching guide at <https://meetedison.com/content/EdCreate/EdCreate-teachers-guide.pdf>
- There is no 'right' answer, and solutions will vary depending on student-created set-ups. However, here is what one sample solution looks like <https://www.edscratchapp.com?share=5DMe54Dw>

### Activity U4-2.5e Challenge up: Hazardous material removal

<b>Activity size</b>	Project
<b>Delivery recommendations</b>	- Complete activity U4-2.5 prior to this activity - Activities U4-2.5a, U4-2.5b, U4-2.5c, U4-2.5d and U4-2.5e are all robotics build projects using IR messaging. Consider using at least one.
<b>Resources needed</b>	Basic supplies set, worksheet U4-2.5e, 1 EdCreate kit per EdRoboClaw, 2 Edison robots per EdRoboClaw, EdRoboClaw build instructions set (available at <a href="https://meetedison.com/content/EdCreate/EdBuild-EdRoboClaw-instructions.pdf">https://meetedison.com/content/EdCreate/EdBuild-EdRoboClaw-instructions.pdf</a> ), supplies for creating the situation set-up

#### Overview

This programming challenge asks students to apply their understanding of coding concepts, including conditionals and sensing, to a real-world application. The EdBuild programming challenge projects use the pre-set instructions and EdCreate kits to create whichever EdBuild is needed for the scenario. This scenario uses the EdRoboClaw. By using the EdRoboClaw as the base for their hazardous material removal robots, students do not need to design the engineering component and can concentrate on the programming component of their solutions.

Students must design two programs which will work in conjunction with each other in order to control the two robots in the EdRoboClaw and solve the scenario.

#### Tips and tricks

- This project is ideal for pair or group work.
- The building challenge in both this activity and activity U1-1.2g *Challenge up: Build and control the EdRoboClaw* is the same.
- Students may find it easiest to first lay out all of the EdCreate pieces onto their work surface and organise the parts into groups of the same piece type and colour. This can help students identify and use the correct pieces as they work through the build.

- In this build, the top robot connects to the bottom robot on the second row of studs from the front of the bottom robot. Thus, the top robot overhangs off the back of the bottom robot by approximately 1 cm.
- The claw in the articulated arm of the EdRoboClaw is composed of 3 ‘fingers’ – two parallel fingers which are stationary (made from grey beams) and the frontmost finger which moves. The row of gears in the articulated arm controls this forward finger, including its positioning relative to the stationary fingers. The alignment of the forward most two gears can affect how the moving finger sits relative to the stationary fingers when the claw is fully open. When the claw is fully open, and the EdRoboClaw is sitting on a flat surface (such as a table or desk), the moving finger should be high enough that one of the EdCreate grey beams can slide under it between the finger and the table. If the moving finger is not this high, try gently separating the front of the arm and rotating the frontmost gear by one or two teeth clockwise independently of the next gear. You will be able to see this move the front finger. Reconnect the gears and the arm.
- The EdRoboClaw can pick up objects with some flat surfaces. Students may find that they are not as able to pick up and carry objects which are round, such as a pen. Try using the 7-hole long grey beam from the EdCreate kit.
- It will be easiest to start the robot with the claw already hovering open over the 7-hole long grey beam from the EdCreate kit representing the hazardous material.
- Additional information about this programming project can be found in the EdCreate teaching guide at <https://meetedison.com/content/EdCreate/EdCreate-teachers-guide.pdf>
- There is no ‘right’ answer, and solutions will vary depending on student-created set-ups. However, here is what one sample solution looks like <https://www.edscratchapp.com?share=rDGRoBba>

### Activity U4-2.5f Challenge up: Homing pigeons

<b>Activity size</b>	Project
<b>Delivery recommendations</b>	Recommended as a capstone activity following the completion of the Let’s Explore activities from unit 4
<b>Resources needed</b>	Basic supplies set, worksheet U4-2.5f, supplies for creating the situation set-up

#### Overview

This chiefly unstructured challenge asks students to create both the physical set-up and programming solution to get their robots to ‘act like homing pigeons’ and find their way ‘home’. The worksheet instructions are intentionally a bit vague. There are few set limitations, and only a minimal level of guidance is provided. This challenge

aims to simply give students a start-point from which to explore using their creativity, ingenuity and computational thinking skills.

#### Tips and tricks

- This activity works best in groups.
- This is NOT a 'fail-proof' activity. The physical limitations of the robot, the design of the set-up, the precision of the programming solutions, and other related factors all mean that even the best of solutions may not work 'perfectly'. Encourage students to take these limitations in stride and find the best solution they can. This can be used as an opportunity to discuss how in the real-world, situations are frequently 'messy', with factors that require compromise. Real-world applications often need us to engineer 'as-good-as-possible' solutions which may never be 100% perfect.
- Programming solutions will depend heavily on the physical set-up design that students create. Having students explain how their programming solutions and physical designs work together is a great exercise in thinking through real-world physical computing.

## Unit 5: Versatile variables

Dive into the key computational concepts of variables, data and expressions while applying prior learning from previous units. Students round out their exploration of EdScratch in this unit. Earlier concepts are revisited and expanded on using the additional flexibility afforded by including variables and operators to manage data within their programs.

### Learning objectives

Students will:

- be introduced to the computer science fundamentals of variables and data
- explore how data and variables can be used with Edison using new blocks in the 'Data' and 'Operators' categories in EdScratch
- further explore Edison's sensors and input capabilities using variables and operators to refine and modify behaviours in working programs
- apply computer science fundamentals such as tracing and debugging to work through projects using sensing and variables

**Key ideas:** variables and storing values, data, operators and computations in programs, tracing

### Lessons and activities in this unit

This unit includes one lesson with a total of four base activities and eight extension activities.

#### Lesson 1: Maths and data in EdScratch

- [U5-1.1 Let's explore expressions](#)
- [U5-1.2 Let's explore Edison's light sensors](#)
  - [U5-1.2a Change it up: Edison the moth](#)
  - [U5-1.2b Challenge up: Edison the cockroach](#)
- [U5-1.3 Let's explore variables](#)
  - [U5-1.3a Challenge up: Spiralling spider trap](#)
  - [U5-1.3b Change it up: Drive a random square](#)
- [U5-1.4 Let's explore using variables with sensor data](#)
  - [U5-1.4a Challenge up: Edison the sprinter](#)
  - [U5-1.4b Change it up: Edison-controlled flag machine](#)
  - [U5-1.4c Change it up: Hey Edison, where do I go?](#)
  - [U5-1.4d Change it up: The Edison chorus](#)

## Lesson 1: Maths and data in EdScratch

This lesson, which introduces the final unexplored sections of the EdScratch language, focuses on data. While block-based programming languages often remove the need for users to get into the details of data in programming, developing a basic working understanding of this fundamental computer science component is a critical part of developing students' digital literacy and in preparing them for text-based programming languages.

Variables, which can be created and utilised through the special 'Data' category in EdScratch, are introduced and then explored in this lesson. Students learn how variables are used to store values. Students experiment with using and manipulating data in working EdScratch programs, including using data gathered with Edison's inbuilt sensors. The concept of using maths in programming, including the important practice of using expressions, is introduced along with the related blocks from the 'Operators' category in EdScratch. The practice of tracing code is also explored and practised.

This lesson has a total of four base activities and eight extension activities:

- [U5-1.1 Let's explore expressions](#)
- [U5-1.2 Let's explore Edison's light sensors](#)
  - o [U5-1.2a Change it up: Edison the moth](#)
  - o [U5-1.2b Challenge up: Edison the cockroach](#)
- [U5-1.3 Let's explore variables](#)
  - o [U5-1.3a Challenge up: Spiralling spider trap](#)
  - o [U5-1.3b Change it up: Drive a random square](#)
- [U5-1.4 Let's explore using variables with sensor data](#)
  - o [U5-1.4a Challenge up: Edison the sprinter](#)
  - o [U5-1.4b Change it up: Edison-controlled flag machine](#)
  - o [U5-1.4c Change it up: Hey Edison, where do I go?](#)
  - o [U5-1.4d Change it up: The Edison chorus](#)

### 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 there is a bit of a stretch between prior units and the contents of this lesson, 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 activities. They will soon see the creative potential these new skills unlock inside programming!

### Activity U5-1.1 Let's explore expressions

<b>Activity size</b>	Small
<b>Delivery recommendations</b>	
<b>Resources needed</b>	- Worksheet U5-1.1 - <i>Optional:</i> Programming devices to look at 'operators' in EdScratch

#### Overview

This offline-activity introduces the concept of expressions in coding and demonstrates how computations can be used inside of expressions. The fact that all computer languages use numbers and mathematics is also discussed. Developing an understanding of how expressions and maths work offline builds students' capacity to engage in more complex computational thinking, enabling them to use these elements with confidence in EdScratch programs in later activities.

Many people have a fear that computer programming requires intense amounts of complicated mathematics. For many programming languages, including EdScratch, this is simply untrue. However, using expressions is required to do some things in EdScratch and to use some of Edison's sensors in meaningful ways. Likewise, being able to use basic maths, such as arithmetic, inside EdScratch programs allows students to do more interesting things with the robots and use Edison's sensors and data in more creative and exciting ways.

The idea of tracing is also introduced in this activity. The formal practice of tracing involves stepping through a program line by line, recording important values. Code traces are often done to help debug code but can also be useful when you just need to understand what is happening in a program, or what will happen when an input has a different value. While the need to record values in EdScratch is limited, this activity gets students familiar with working through computations, enabling them to develop the habit of tracing through programs in future activities.

#### Tips and tricks

- Binary is not taught in the EdScratch lessons. This activity does make mention of the fact that 'without a computer language, you would need to write every single command using nothing but 1s and 0s,' which is a nod to the fact that all computers operate using binary. If you are teaching binary, consider tying a separate lesson on binary into this activity.
- 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. If students put in computations which, for example, divide by zero, they may get errors, or the robot may behave in unexpected ways.
- While not required in the worksheet, you may choose to have students look at the 'operators' category in EdScratch to find the six blocks used in

expressions and begin to piece together how the blocks can be used in EdScratch programs.

### Answer key

Question	Type	Answer	Marking notes
1	EA	<i>Meaning: Is <math>2 * 2</math> the same as 4? Resolves to: True</i>	
2	EA	<i>Meaning: Is 2 greater than or equal to 4? Resolves to: False</i>	
3	EA	<i>Meaning: Is <math>2 + 2</math> not equal to 4? Resolves to: False</i>	
4	EA	<i>Meaning: Is <math>2 - 1</math> less than <math>4 - 1</math>? Resolves to: False</i>	

### Activity U5-1.2 Let's explore Edison's light sensors

<b>Activity size</b>	Large
<b>Delivery recommendations</b>	- Complete activity U4-2.2 prior to this activity - Complete activity U5-1.1 prior to this activity
<b>Resources needed</b>	Basic supplies set, worksheet U5-1.2

#### Overview

This activity introduces Edison's visible light sensors, explains how the sensors work, and then explores using the sensors with expressions to make two different EdScratch programs. In order to effectively use Edison's visible light sensors, which detect visible light and feed the result back to Edison as a light reading, students need to understand the concepts of events and conditionals as well as be able to use expressions in EdScratch programs.

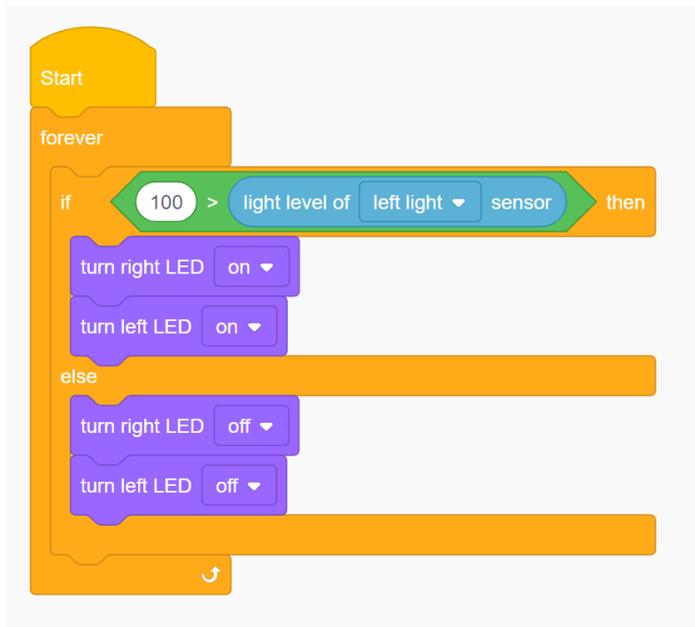
More than with any other sensor, using the light sensors exemplifies the idea that sensors create data and that it is this data which is used by the robot. Using the visible light sensors in programs helps students begin to see that there are layers of information 'behind the scenes' of block-based programming languages. Developing an appreciation for this fact helps prepare students to work with more creative and complex programs in EdScratch as well as setting the groundwork for moving into text-based languages, such as [EdPy](#).

#### Tips and tricks

- 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 the input parameter space in this case, which stands for 'not a number'.
- Edison will detect all ambient visible light and is unable to determine the source of the light. Take into consideration how bright or dark the room is when testing programs using Edison's light sensors.
- Encourage students to test different values in the expressions they are using with light sensor readings to find what works best with their robot in the environment where they are running their program.

### Answer key

Question	Type	Sample answer	Marking notes
1	RC		

### Activity U5-1.2a Change it up: Edison the moth

<b>Activity size</b>	Medium
<b>Delivery recommendations</b>	Complete activity U5-1.2 prior to this activity
<b>Resources needed</b>	Basic supplies set, worksheet U5-1.2a, torches (flashlights)

#### Overview

Students explore how to use Edison's light sensors together to create a program for Edison which makes the robot mimic phototrophic behaviour. The fact that the light sensors store light readings as numerical values, enabling those values to be compared to each other using operators in an expression, is demonstrated in this activity.

### Tips and tricks

- You will need a torch (flashlight) and a flat surface located away from any other sources of bright light, such as sunlight or overhead fluorescents.
- Once Edison ‘sees’ the bright source of light, the robot will drive towards it. By moving the torch, you can control where Edison drives.
- If the environment is brighter than the torch, Edison won’t be able to detect the torch, and will not respond as expected.
- The ‘moth’ program in this activity should function the same way as the program activated by the ‘follow a torch’ barcode program. You may choose to have students revisit that barcode to see how their program compares to the barcode program.
- One programming solution can be seen at <https://www.edscratchapp.com?share=80vEnJYd>

### Activity U5-1.2b Challenge up: Edison the cockroach

<b>Activity size</b>	Medium
<b>Delivery recommendations</b>	- Complete activity U5-1.2 prior to this activity - This activity builds on concepts demonstrated with more explanation in U5-1.2a. Consider using this as an extension to that activity.
<b>Resources needed</b>	Basic supplies set, worksheet U5-1.2b, torches (flashlights)

### Overview

Students explore how to use Edison’s light sensors together to create a program for Edison making the robot mimic negative phototropic behaviour. The fact that the light sensors store light readings as numerical values, enabling those values to be compared to each other using operators in an expression, is demonstrated in this activity.

### Tips and tricks

- You will need a torch or flashlight and a flat surface located away from any other sources of bright light, such as sunlight or overhead fluorescents.
- Once Edison ‘sees’ the bright source of light, the robot should drive in the opposite direction. By moving the torch, you can control where Edison drives.
- If the environment is brighter than the torch, Edison won’t be able to detect the torch, and will not respond as expected.
- The ‘cockroach’ program is effectively the same as the ‘moth’ program from activity U5-1.2b, but the logic of how the robot should respond to light is inverse.

### Answer key

Question	Type	Sample answer	Marking notes
1	SE	<pre> Forever   if right light sensor &gt; left light sensor     go left   else     go right </pre>	

### Activity U5-1.3 Let's explore variables

<b>Activity size</b>	Large
<b>Delivery recommendations</b>	
<b>Resources needed</b>	Basic supplies set, worksheet U5-1.3

#### Overview

The concept of variables, one of the fundamental components of code, is introduced in this activity. The worksheet explains how variables work, then has students 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.

Even though EdScratch enables students to write many programs without using variables, the full scope of the language can only be tapped into by using variables. Created and managed using the 'Data' category in EdScratch, variables function differently than other blocks in the language. Spending the time to work through this somewhat explanation-heavy activity will help students gain mastery over creating and using variables in meaningful programs.

#### 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 refresher on using Edison's 'set motors' blocks, refer to activity U2-2.5 *Let's explore Edison's motors*.
- Binary is not taught in these lessons and, therefore, the 'bit shift' blocks are not included in the student worksheets. If you are teaching binary, you might choose to use the 'bit shift' blocks in an extension lesson on variables and binary.
- 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, 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>

### Answer key

Question	Type	Answer/Sample answer	Marking notes																												
1	EA	<table border="1"> <thead> <tr> <th>In loop #</th> <th>Starting value of DriveLength</th> <th>Wait block input value (in milliseconds)</th> <th>New value of DriveLength</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>1</td> <td>200</td> <td>2</td> </tr> <tr> <td>2</td> <td>2</td> <td>400</td> <td>3</td> </tr> <tr> <td>3</td> <td>3</td> <td>600</td> <td>4</td> </tr> <tr> <td>5</td> <td>5</td> <td>1000</td> <td>6</td> </tr> <tr> <td>7</td> <td>7</td> <td>1400</td> <td>8</td> </tr> <tr> <td>10</td> <td>10</td> <td>2000</td> <td>11</td> </tr> </tbody> </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	
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																												
2	SE	<i>This program makes Edison drive in a right-angled spiral that grows out from the middle.</i>	Students should note that the robot drives in an outward spiralling pattern.																												
3	SE	<i>The code uses a variable which increases by 1 in each loop repetition. Because the variable is multiplied by 200 and that becomes the value of the 'wait' block, the 'wait' time grows longer each loop. This means the robot ends up driving longer each loop, causing the robot to drive in an outwards spiral.</i>	Students should identify that the increasing value of the variable increases the time the motors 'wait' while 'set to drive forwards' which increases the distance of each leg driven.																												

### Activity U5-1.3a Challenge up: Spiralling spider trap

<b>Activity size</b>	Medium (main activity) Large (main activity + bonus engineering challenge)
<b>Delivery recommendations</b>	Complete activity U5-1.3 prior to this activity
<b>Resources needed</b>	- Basic supplies set, worksheet U5-1.3a - <i>Optional:</i> Supplies for making the 'spider silk', worksheet U5-1.3 (for reference)

#### Overview

Building on the programming concepts explored in activity U5-1.3, this extension activity asks students to re-imagine the 'spiral-out' program from activity U5-1.3 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.

An extra ‘mini 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.

### Tips and tricks

- You may want to review the ‘spiral-out’ program from activity U5-1.3 to help students begin to work out their ‘spiral-in’ programs.
- Remind students to use good variable names and avoid illegal characters in their variable’s name.
- 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.
- An example programming solution can be seen at <https://www.edscratchapp.com?share=rDBEwzbR>

### Activity U5-1.3b Change it up: Drive a random square

<b>Activity size</b>	Small
<b>Delivery recommendations</b>	Complete activity U5-1.3 prior to this activity
<b>Resources needed</b>	Basic supplies set, worksheet U5-1.3b

### Overview

This activity uses a variable with the ‘random’ number block so that the value of the variable is randomly set each time the program runs. This activity demonstrates another way of using variables in programs in EdScratch using just set values (as opposed to sensor data) and provides students with another opportunity to practice using computation inside a program.

The extra ‘mini challenge’ pushes students to think about how data is being stored in the program and challenges students to create a secondary way of using this data in the program by getting Edison to ‘signal’ the value.

### Tips and tricks

- If students are completing the mini challenge, you may want to remind them that a variable can be used multiple times in a program.

- An example programming solution for the mini challenge can be seen at <https://www.edscratchapp.com?share=7bxNWWYZ>

### Answer key

Question	Type	Answer	Marking notes
1	EA	<b>If the random number is:</b>	<b>SquareSideLength will be:</b>
		1	500
		2	1000
		3	1500
		4	2000
		5	2500
		6	3000

### Activity U5-1.4 Let's explore using variables with sensor data

<b>Activity size</b>	Medium
<b>Delivery recommendations</b>	
<b>Resources needed</b>	- Basic supplies set, worksheet U5-1.4, activity sheet U5-1 - <i>Optional:</i> Supplies for marking out a test area

#### Overview

This activity introduces how variables and sensors can be used in combination. There are two main ways sensor data can be used with variables: the data from sensors can be stored in a variable, and sensor events can be used to affect variables. The programming example in this activity focuses on the second way of using variables and sensors together. The program increments a variable each time an event is detected. Students see this capability in action by reading, then running a program which gets Edison to drive, counting the black lines it detects.

While an explicit example is not shown in this activity, the idea that the value from a sensor can be stored in a variable is important. Variables can be set to specific values by the programmer, but can also be set by other inputs, such as a sensor reading.

#### Tips and tricks

- Activity U5-1.4a is an extension to the example program in this activity. You may choose to have students try this extra programming challenge while they have their test area for this activity set up.
- Questions 3 and 4 on the worksheet ask students to come up with their own idea for using sensor data in a variable in a working EdScratch program. If you want to keep this activity shorter, consider using these questions as a class discussion, rather than having students work on their own ideas and program design.

**Answer key**

Question	Type	Answer/Sample answer	Marking notes
1	EA	<i>This program first tells Edison to turn on the line tracker and set the variable LineCount to zero. It then tells the robot to drive forward until LineCount is equal to four. There is an interrupt 'event' in this program with a subroutine that says that any time Edison detects a non-reflective (black) line, the robot should increment (add 1 to) the value of LineCount. Once that event has triggered 4 times, the 'until' condition will be met and the robot will stop driving.</i>	Student answers should identify the key ideas noted in the sample answer.
2	SE	<i>I think that the variable is called 'LineCount' because the information that this variable is storing is related to how many black lines the robot has detected, or 'counted'.</i>	Student answers should identify that the lines detected by the robot are being 'counted' using this variable to store the ongoing count.
3	SE	<i>My idea is to use Edison's obstacle detection to track how many people walk into my room when I'm not home. I can store the number of obstacles detected (i.e. people who walk past Edison) in a variable, increasing the value by 1 for each obstacle detected. Then I can get the program to beep a number of times equal to that variable when I push the round button, telling me how many people came in.</i>	
4	SE	<i>I was able to write my program, but it took a few tries. <a href="https://www.edscratchapp.com?share=RDaGLe0V">https://www.edscratchapp.com?share=RDaGLe0V</a> My first version didn't have the 'forever' loop with the main program, so it ended really quickly and didn't work. I also realised I needed to reset the variable once I had checked the detector so that it wouldn't just keep adding all of the new obstacles to old ones. In testing, I did notice that sometimes just one person walking in front of Edison would get stored as multiple obstacles detected, so this system isn't perfect, but it does work!</i>	It's possible that students will have come up with an idea that cannot be brought to fruition.

### Activity U5-1.4a Challenge up: Edison the sprinter

<b>Activity size</b>	Medium
<b>Delivery recommendations</b>	Complete activity U5-1.4 prior to this activity
<b>Resources needed</b>	- Basic supplies set, worksheet U5-1.4a, activity sheet U5-1 - <i>Optional:</i> Supplies for marking out a test area

#### Overview

Designed as an extension activity for the program example in activity U5-1.4, this programming challenge has students apply their understanding of sensor events and variables to get the robots to track events and modify outputs accordingly. By changing the robot's output each time a variable's value changes, students can see that the variable's value is actually changing, helping to make the idea of data in variables more tangible.

#### Tips and tricks

- To see the speed change clearly, it is best to use a larger test area. You may also want to advise students to change speed by more than one each time. For example, start at speed 1, then increase to speed 3 from the first line.
- This activity is an extension to the example program in activity U5-1.4. You may choose to have students try both programming challenges while they have their test area set up.
- An example programming solution can be seen at <https://www.edscratchapp.com?share=Eb1j5aDm>

### Activity U5-1.4b Change it up: Edison-controlled flag machine

<b>Activity size</b>	Project
<b>Delivery recommendations</b>	- Complete activity U5-1.4 prior to this activity - Activities U5-1.4b, U5-1.4c and U5-1.4d all use variables to store and use IR messaging values. Consider using at least one.
<b>Resources needed</b>	Basic supplies set, worksheet U5-1.4b, maker space supplies for creating the flags, EdCreate kits and/or other supplies for attaching the flags

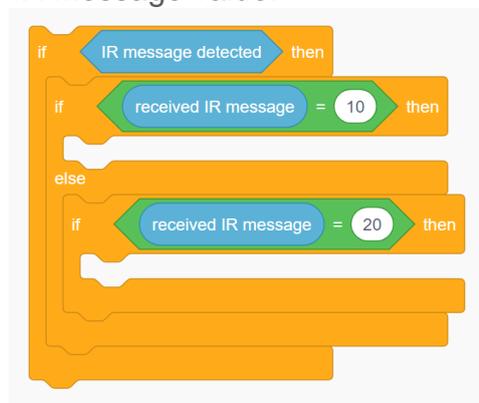
#### Overview

This activity dives into storing data from one of Edison's sensors as a value in a variable. Edison's IR messaging allows multiple Edison's to send and receive messages. The robots can send and detect 256 different messages. These unique messages can be used to trigger different behaviours in the receiving robot, depending on what message it detects.

This semi open-ended project challenges students to create a physical computing solution with real-world applications, applying their understanding of coding concepts, including conditionals, sensing, and using variables to store data. Students collaborate using two Edison robots to create a controller bot (message sender) and flag machine (message receiver) which work together. The sending robot needs to be able to send out one or two separate messages, depending on the button that is pressed. The receiving robot needs to react differently to each of those messages, generating a different output depending on which message it receives.

### A special note on variables and IR message data

When using IR messages in programs using specific messages to trigger different events, students might set up their programs without a variable, using just an expression to check the IR message value:



As far as programming procedure goes, this isn't ideal.

The convention in computer programming is to save any sensor data into a variable. Then, if your program uses multiple 'if' checks in the code, you have the program check the variable multiple times, rather than read the sensor multiple times. This is because the sensor could change while the checks are occurring.

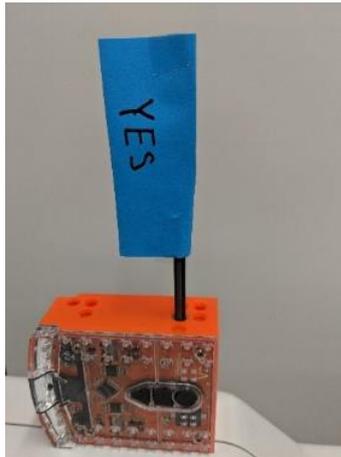
In EdScratch in particular, students will discover that the 'IR message = 20' code in the above program will never run. This is due to how Edison deals with IR message data in EdScratch. Once the robot processes the 'if' statement in the 'if-else' code block, the value of the message has been read, and so the value is cleared from the register. If that message did not equal 10, the 'if' code will not run, but the IR register is still reset to '0'. Therefore, the second 'if' (inside the 'else') only ever sees a value of '0' and never runs.

Computers which use sensors often clear the sensor registry in similar ways, making this an excellent example of why it's much better to get into the habit of using variables to store data, rather than rely on a sensor read directly.

### Tips and tricks

- You may want to review how IR messaging works with Edison prior to this activity using activity U4-2.5 *Let's explore messaging with Edison*.

- Students are welcome to build a flag machine in a different way to the project description in this activity or to come up with their own IR message detecting creation.
- A variation of this activity using TV/DVD remote controls is included in Unit 4, activity U4-2.5a *Change it up: Remote-controlled flag machine*. You may choose to reference these two activities when using either or run them as two programming challenges for the one build design.
- There is no 'right' answer, but here is what one sample solution looks like:
  - Physical design:



- Program set: <https://www.edscratchapp.com?share=V0eg4KDa>

#### Answer key

Question	Type	Sample answer	Marking notes
1	SE	<ul style="list-style-type: none"> <li>•When you press the round button on the controller bot, it will send IR message <code>_10_</code>, and that will tell the flag machine to show the <code>'no'</code> side of the flag.</li> <li>•When you press the triangle button on the controller bot, it will send IR message <code>_20_</code>, and that will tell the flag machine to show the <code>'yes'</code> side of the flag.</li> </ul>	Students should use this to help them in their design process. Values and movement should be reflected in their 'flag machine' program.

#### Activity U5-1.4c Change it up: Hey Edison, where do I go?

<b>Activity size</b>	Project
<b>Delivery recommendations</b>	<ul style="list-style-type: none"> <li>- Complete activity U5-1.4 prior to this activity</li> <li>- Activities U5-1.4b, U5-1.4c and U5-1.4d all use variables to store and use IR messaging values. Consider using at least one.</li> </ul>
<b>Resources needed</b>	Basic supplies set, worksheet U5-1.4c, maker-space materials for creating the test space

### Overview

This activity dives into storing data from one of Edison's sensors as a value in a variable. Edison's IR messaging allows multiple Edison's to send and receive messages. The robots can send and detect 256 different messages. These unique messages can be used to trigger different behaviours in the receiving robot, depending on what message it detects.

This semi open-ended project challenges students to design a test space and create multiple programs getting two Edison robots to use IR messaging to complete a branching maze. Designed to be done as a collaborative project, students need to work together to design their treasure maze, program their two robots in line with their maze, and be able to change the sending commands to get the driver bot to any end location in the maze. To create a successful outcome in this project, students need to combine the various concepts they have learned including sequence, conditionals, sensing, and using variables with data.

### Tips and tricks

- This project can be done in pairs but is ideal for groups.
- If this is your first activity using variables and IR message data, you may want to read the *Special note on variables and IR data* in the overview of activity U5-1.4b of this guide.
- A great way to test students when they have completed this project is to assign the students the location of the treasure. Students should be able to then demonstrate a successful outcome by changing just the navigator bot's program to send the correct sequence of messages to the driver bot. When both bots run their respective programs, the driver bot should get to the assigned finish spot.
- Remind students to get the receiving robot running its programs before the sending robot starts its program.
- Students' programs will depend heavily on their treasure map design. A set of example programs using the 'base design' from the worksheet as the treasure map are included for reference:
  - Driver bot: <https://www.edscratchapp.com?share=RDgmg30B>
  - Navigator bot: <https://www.edscratchapp.com?share=RDaGLj0V>

### Answer key

Question	Type	Sample answer	Marking notes															
1	SE	<p><i>Our map has three paths coming out of each junction, so we will need a total of 4 IR messages:</i></p> <table border="1"> <thead> <tr> <th>IR message value</th> <th>Sending Edison</th> <th>Receiving Edison will...</th> </tr> </thead> <tbody> <tr> <td>10</td> <td>Navigator</td> <td>Go left</td> </tr> <tr> <td>20</td> <td>Navigator</td> <td>Go straight</td> </tr> <tr> <td>30</td> <td>Navigator</td> <td>Go right</td> </tr> <tr> <td>100</td> <td>Driver</td> <td>Send out next command</td> </tr> </tbody> </table>	IR message value	Sending Edison	Receiving Edison will...	10	Navigator	Go left	20	Navigator	Go straight	30	Navigator	Go right	100	Driver	Send out next command	
IR message value	Sending Edison	Receiving Edison will...																
10	Navigator	Go left																
20	Navigator	Go straight																
30	Navigator	Go right																
100	Driver	Send out next command																

### Activity U5-1.4d Change it up: The Edison chorus

<b>Activity size</b>	Project
<b>Delivery recommendations</b>	<ul style="list-style-type: none"> <li>- Complete activity U5-1.4 prior to this activity</li> <li>- Activities U5-1.4b, U5-1.4c and U5-1.4d all use variables to store and use IR messaging values. Consider using at least one.</li> </ul>
<b>Resources needed</b>	Basic supplies set, worksheet U5-1.4d, sheet music or access for students to look up songs

#### Overview

This activity dives into storing data from one of Edison's sensors as a value in a variable. Edison's IR messaging allows multiple Edison's to send and receive messages. The robots can send and detect 256 different messages. These unique messages can be used to trigger different behaviours in different receiving robots.

This semi open-ended project challenges students to collaborate to get multiple Edison robots to use IR messaging to trigger different actions. This project highlights what is possible when programming using multiple different IR messages by using the various messages to not only trigger different outputs but also to have multiple robots check and react to messages. To create a successful outcome in this project, students need to combine the various concepts they have learned including sequence, conditionals, sensing, and using variables with data.

#### Tips and tricks

- This project needs a minimum of 3 Edison robots, but works great with 4+ robots, making it ideal for groups.
- If this is your first activity using variables and IR message data, you may want to read the *Special note on variables and IR data* in the overview of activity U5-1.4b of this guide.

- Consider giving your students some song options or sheet music to use in this activity. Good options for songs in a round include:
  - *Row, Row, Row Your Boat*
  - *Three Blind Mice*
  - *Frère Jacques*
  - *Farmer in the Dell*
- To help minimise the chance of robots missing IR messages, suggest students run this program with the performer bots in an arch facing the conductor bot.
- If multiple groups are running programs near each other using the same message values, they may experience cross-talk between groups.
- Remind students to get the receiving robots running their programs before the sending robot starts its program.
- A similar programming challenge to this activity is included in Unit 2 in activity U2-2.3a *Change it up: Play a song in a round*. That activity has students use wait blocks to time each robot's start. You may choose to use activity U2-2.3a *Change it up: Play a song in a round* and this activity together, highlighting the advantages and disadvantages of each approach.
- Clearing IR data with the 'clear IR message data' block after a variable has been set with the IR message value helps minimise issues when using multiple IR messages in a program by clearing old data from the robot's memory. Encourage students to clear the IR data as a next action after setting the variable's value with that IR data.
- Students' programs will depend heavily on their song choice. A set of example programs using four robots playing *Row, Row, Row Your Boat* are included for reference:
  - Conductor bot: <https://www.edscratchapp.com?share=ZDRedPby>
  - Performer bots: <https://www.edscratchapp.com?share=QbOQ3OYk>

### Answer key

Question	Type	Sample answer	Marking notes
1	SE	<i>Our performer bots will play 'Row, Row, Row your boat', and each new bot will join in when its predecessor finishes the first verse.</i>	

2	SE	<i>We have three performer bots, so we will need a total of 5 IR messages:</i>			
		<b>IR message value</b>	<b>Sending Edison</b>	<b>Receiving Edison</b>	<b>Receiving Edison's action</b>
		10	Conductor	Performer #1	play
		1	Performer #1	Conductor	send play command to performer # 2
		20	Conductor	Performer #2	play
		2	Performer #2	Conductor	send play command to performer # 3
		30	Conductor	Performer #3	play

## Unit 6: Inventor's time!

Put all of your Edison and EdScratch knowledge into action! By designing and developing projects of their own using iterative cycles of planning, making and testing, students put the key computational thinking, problem-solving, programming, and physical computing concepts they have learned to work in this culmination unit.

### Learning objectives

Students will:

- learn about the design-build-test cycle and strategies, such as decomposition and iterative testing, for physical computing problem-solving
- demonstrate their understanding of key computational thinking and computer science principals through open-ended projects

**Key ideas:** the design-build-test cycle, decomposition and problem solving, iterative testing

### Lessons and activities in this unit

This unit includes one lesson with a total of two base activities and five extension activities.

#### Lesson 1: Design, build, test, repeat

- [U6-1.1 Let's explore the design-build-test cycle](#)
  - [U6-1.1a Challenge up: Invent an imaginary creature](#)
  - [U6-1.1b Challenge up: Invent a cotton ball launcher](#)
  - [U6-1.1c Challenge up: Invent a burglar alarm](#)
  - [U6-1.1d Challenge up: Invent a mousetrap](#)
  - [U6-1.1e Challenge up: Invent a combination safe](#)
- [U6-1.2 Let's explore a haunted house](#)

## Lesson 1: Design, build, test, repeat

With the whole of the EdScratch language explored, this lesson focuses on creative problem-solving across both programming and physical design challenges. Students learn about the design-build-test cycle and formal strategies for physical computing problem-solving, including decomposition and iterative testing. These skills are then put to the test as students demonstrate the knowledge they have gained around key computational thinking, programming, and computer science principals through open-ended projects.

This lesson has a total of two base activities and five extension activities:

- [U6-1.1 Let's explore the design-build-test cycle](#)
  - o [U6-1.1a Challenge up: Invent an imaginary creature](#)
  - o [U6-1.1b Challenge up: Invent a cotton ball launcher](#)
  - o [U6-1.1c Challenge up: Invent a burglar alarm](#)
  - o [U6-1.1d Challenge up: Invent a mousetrap](#)
  - o [U6-1.1e Challenge up: Invent a combination safe](#)
- [U6-1.2 Let's explore a haunted house](#)

### Activity U6-1.1 Let's explore the design-build-test cycle

<b>Activity size</b>	Large (base tasks only) Project (with the build)
<b>Delivery recommendations</b>	
<b>Resources needed</b>	- Basic supplies set, worksheet U6-1.1, activity sheet U6-1 - <i>If doing build:</i> maker-space supplies

#### Overview

This lesson formally introduces a structured process for creating robotic inventions using Edison robots. While many of the activities throughout the EdScratch lessons afford students chances to think creatively and create using Edison, this activity provides an explicit framework for approaching physical computing projects. The primary purpose of this activity is to introduce the creation process in a structured way and demonstrate the value of taking the time to plan (design) – which is a major part of programming and engineering practice. Students brainstorm possible ideas for ‘inventions’ using Edison, then work through the process of designing both the physical engineering component and the programming component of their chosen idea.

Learning the design-build-test cycle and practising the design component will enable students to achieve greater success in open-ended engineering and programming projects, including the other activities in this lesson. Having a framework for approaching open-ended projects helps students to focus, plan, and apply learnings

as they go. For students with lots of ideas but unfocused energy, using a framework enables them to channel their creativity into the iteration process. For students who lack confidence with open-ended challenges, having a structure to refer to can help embolden them to try new things and unlock their creative potential.

### Tips and tricks

- If you are running this activity as a group or whole class, consider managing the timer for the brainstorming session. Forcing students to move onto their next idea when the time is up will keep them on task and help get them into the flow. It also allows you to spot any students who may have given up and encourage them to keep going. No bad ideas in brainstorming!
- The initial brainstorming task is best done as individuals, but the review of ideas and design portions of this activity can easily be done in partners or groups.
- If students have decomposed their project into more than two parts, have them use extra paper or make separate sections to design and plan each part.
- While students do not need to actually build and program their creations to complete the base tasks, having access to Edison and EdScratch may prove helpful for their design process.
- The core of this activity only goes through the ‘design’ stage. The ‘mini challenge’ is not mini – actually building and testing the ideas students have come up with will turn this activity into a major project.
- Student ideas may or may not be possible. If you do have students attempt to build and program their designs, remind them that even if their design is not achievable, that doesn’t make it a failure. Look at why it is not achievable. For example, is there a limit to how the robot works or the materials that they used that caused it to be unsuccessful?
  - Unsuccessful designs can teach just as much, if not more than, successful designs.

### Answer key

Question	Type	Sample answer	Marking notes
1	SE	<i>Project name: EggBeater Bot. My idea is to attach whisks to Edison’s powered sockets and turn Edison into a hand mixer.</i>	
2	SE	<i>There are two main parts: the physical design and the program. Each of these has two parts: Physical design – 1) making the whisks and 2) attaching the whisks to Edison. Program design – 1) getting the program to be able to start and stop and 2) getting the program to move the whisks at different speeds.</i>	The project should have a physical engineering and software programming component.

3	SE	<p><i>Whisks: I think I will use a pencil for the staff for the main shaft of the whisk, then make the blades using strips of cut plastic from recycled bottles. I will glue these to the pencil. Attaching the whisks: I think I will use mostly EdCreate parts to attach the whisks to the wheels of Edison. That way I won't need to have a separate way of attaching them into the powered socket too.</i></p>	<p>Any combination of drawings, diagrams, written plans, etc is acceptable so long as the students' design plan is detailed.</p> <p>The sample answer has been deliberately kept short – it is intended to serve as a possible partial example only.</p>
4	SE	<p><i>The program should use button press events. If the triangle button is pressed, the motors should start turning forwards. If the round button is pressed, the motors should move faster one speed. The motors should move faster each round button press up to a top speed of 5. Anytime the triangle button is pressed while the motors are moving, they should stop.</i></p>	<p>Pseudocode or any combination of drawings, diagrams, written plans, etc is acceptable so long as the students' design plan is detailed.</p> <p>The sample answer has been deliberately kept short – it is intended to serve as a possible partial example only.</p>

### Activity U6-1.1a Challenge up: Invent an imaginary creature

<b>Activity size</b>	Project
<b>Delivery recommendations</b>	<ul style="list-style-type: none"> <li>- Complete activity U6-1.1 prior to this activity</li> <li>- Activities U6-1.1a, U6-1.1b, U6-1.1c, U6-1.1d, and U6-1.1e all offer an open-ended project idea that requires students to apply the design-build-test cycle. Consider using at least one.</li> </ul>
<b>Resources needed</b>	Basic supplies set, worksheet U6-1.1a, maker-space/crafting supplies, EdCreate kits/LEGO bricks

#### Overview

Open-ended projects, including this challenge, promote creative problem solving and give students opportunities to find STEM applications to real-world scenarios. One of five project options, the objective of this challenge is for students to build an imaginary creature that can move and react to the world using at least one of Edison's sensors.

Students apply the design-build-test cycle to this open-ended challenge. With a set end-goal and only a few established success criteria, this activity is designed to encourage creative problem-solving through the meaningful application of learned skills. Students must apply their programming and computational thinking skills to this challenge, but also need to experiment with the physical creation, which adds an engineering component to this project.

#### Tips and tricks

- This project works well in pairs or small groups.
- Remind students that unsuccessful designs can teach just as much, if not more than, successful designs. Encourage students to look at what isn't working in their creation through the lens of the design-build-test cycle. Decompose what's going on into smaller parts and problem-solve each component.
- There is no single 'best way' of approaching the problem or a set solution. Student creations can vary wildly as will the programming component of their solutions.

#### Activity U6-1.1b Challenge up: Invent a cotton ball launcher

<b>Activity size</b>	Project
<b>Delivery recommendations</b>	- Complete activity U6-1.1 prior to this activity - Activities U6-1.1a, U6-1.1b, U6-1.1c, U6-1.1d, and U6-1.1e all offer an open-ended project idea that requires students to apply the design-build-test cycle. Consider using at least one.
<b>Resources needed</b>	Basic supplies set, worksheet U6-1.1b, maker-space/crafting supplies, EdCreate kits/LEGO bricks, cotton balls or craft pom-poms (for launching)

#### Overview

Open-ended projects, including this challenge, promote creative problem solving and give students opportunities to find STEM applications to real-world scenarios. One of five project options, the objective of this challenge is for students to build a cotton ball launcher that can shoot as high as possible, as far as possible, or as accurately as possible.

Students apply the design-build-test cycle to this open-ended challenge. With a set end-goal and choice of the objective as the only established success criteria, this activity is designed to encourage creative problem-solving through the meaningful application of learned skills. Students must apply their programming and computational thinking skills to this challenge, but also need to experiment with the physical creation, which adds an engineering component to this project.

### Tips and tricks

- This project works well in pairs or small groups.
- Remind students that unsuccessful designs can teach just as much, if not more than, successful designs. Encourage students to look at what isn't working in their creation through the lens of the design-build-test cycle. Decompose what's going on into smaller parts and problem-solve each component.
- There is no single 'best way' of approaching the problem or a set solution. Student creations can vary wildly as will the programming component of their solutions.

### Activity U6-1.1c Challenge up: Invent a burglar alarm

<b>Activity size</b>	Project
<b>Delivery recommendations</b>	- Complete activity U6-1.1 prior to this activity - Activities U6-1.1a, U6-1.1b, U6-1.1c, U6-1.1d, and U6-1.1e all offer an open-ended project idea that requires students to apply the design-build-test cycle. Consider using at least one.
<b>Resources needed</b>	Basic supplies set, worksheet U6-1.1c, maker-space/crafting supplies, EdCreate kits/LEGO bricks

### Overview

Open-ended projects, including this challenge, promote creative problem solving and give students opportunities to find STEM applications to real-world scenarios. One of five project options, the objective of this challenge is for students to build a burglar alarm that uses at least one of Edison's sensors to detect a 'threat' and then produce some output to frighten off the would-be thief.

Students apply the design-build-test cycle to this open-ended challenge. With a set end-goal and only a few established success criteria, this activity is designed to encourage creative problem-solving through the meaningful application of learned skills. Students must apply their programming and computational thinking skills to this challenge, but also need to experiment with the physical creation, which adds an engineering component to this project.

### Tips and tricks

- This project works well in pairs or small groups.
- Remind students that unsuccessful designs can teach just as much, if not more than, successful designs. Encourage students to look at what isn't working in their creation through the lens of the design-build-test cycle. Decompose what's going on into smaller parts and problem-solve each component.

- There is no single ‘best way’ of approaching the problem or a set solution. Student creations can vary wildly as will the programming component of their solutions.

### Activity U6-1.1d Challenge up: Invent a mousetrap

<b>Activity size</b>	Project
<b>Delivery recommendations</b>	- Complete activity U6-1.1 prior to this activity - Activities U6-1.1a, U6-1.1b, U6-1.1c, U6-1.1d, and U6-1.1e all offer an open-ended project idea that requires students to apply the design-build-test cycle. Consider using at least one.
<b>Resources needed</b>	Basic supplies set, worksheet U6-1.1d, maker-space/crafting supplies, EdCreate kits/LEGO bricks

#### Overview

Open-ended projects, including this challenge, promote creative problem solving and give students opportunities to find STEM applications to real-world scenarios. One of five project options, the objective of this challenge is for students to build a mousetrap that uses at least one of Edison’s sensors to trigger the trap and then produce some output to alert the creator that the trap has been sprung.

Students apply the design-build-test cycle to this open-ended challenge. With a set end-goal and only a few established success criteria, this activity is designed to encourage creative problem-solving through the meaningful application of learned skills. Students must apply their programming and computational thinking skills to this challenge, but also need to experiment with the physical creation, which adds an engineering component to this project.

#### Tips and tricks

- This project works well in pairs or small groups.
- Remind students that unsuccessful designs can teach just as much, if not more than, successful designs. Encourage students to look at what isn’t working in their creation through the lens of the design-build-test cycle. Decompose what’s going on into smaller parts and problem-solve each component.
- There is no single ‘best way’ of approaching the problem or a set solution. Student creations can vary wildly as will the programming component of their solutions.

### Activity U6-1.1e Challenge up: Invent a combination safe

<b>Activity size</b>	Project
<b>Delivery recommendations</b>	- Complete activity U6-1.1 prior to this activity - Activities U6-1.1a, U6-1.1b, U6-1.1c, U6-1.1d, and U6-1.1e all offer an open-ended project idea that requires students to apply the design-build-test cycle. Consider using at least one.
<b>Resources needed</b>	Basic supplies set, worksheet U6-1.1e, maker-space/crafting supplies, EdCreate kits/LEGO bricks, TV/DVD remotes (as an option for the safe code)

#### Overview

Open-ended projects, including this challenge, promote creative problem solving and give students opportunities to find STEM applications to real-world scenarios. One of five project options, the objective of this challenge is for students to build a combination safe that only opens when the correct sequence (code) is entered. The code that opens the safe can be a sequence of triangle and round button presses, a sequence of TV/DVD remote code signals, a sequence of IR messages from a different Edison robot, or some combination of all these options.

Students apply the design-build-test cycle to this open-ended challenge. With a set end-goal and choice of the objective as the only established success criteria, this activity is designed to encourage creative problem-solving through the meaningful application of learned skills. Students must apply their programming and computational thinking skills to this challenge, but also need to experiment with the physical creation, which adds an engineering component to this project.

#### Tips and tricks

- This project works well in pairs or small groups.
- Remind students that unsuccessful designs can teach just as much, if not more than, successful designs. Encourage students to look at what isn't working in their creation through the lens of the design-build-test cycle. Decompose what's going on into smaller parts and problem-solve each component.
- There is no single 'best way' of approaching the problem or a set solution. Student creations can vary wildly as will the programming component of their solutions.

### Activity U6-1.2 Let's explore a haunted house

<b>Activity size</b>	Project
<b>Delivery recommendations</b>	Recommend as a capstone project for both unit 6 and the EdScratch lesson set
<b>Resources needed</b>	Basic supplies set, worksheet U6-1.2, maker-space/crafting supplies, EdCreate kits/LEGO bricks, a black work surface, white masking tape or alternative

#### Overview

More reminiscent of a 'Change it up' or 'Challenge up' activity, this project differs from all the other 'Let's explore' activities in the lesson plans. Designed as a capstone project for both unit 6 and the whole of the EdScratch lessons, this activity asks students to let their imaginations flow. By creating their own scenarios, test environments, inventions, and programs, students have the opportunity to apply creative problem solving and their programming and computational thinking skills.

Part design challenge, part programming challenge and part exercise in collaboration, this project is intended to be done in groups with very few limitations or parameters for success being pre-set. To help students get into the experimental mindset, the first task in this project (the ghost hunter challenge) presents a more structured scenario. What the room (test space) needs to be is established already, and there are hints as to what needs to be included in the programming solution. Moreover, the need to build a physical creation is removed. Once students have solved this room, however, the boundaries disappear.

Programming is inherently creative, and much of technology exists outside of the computer screen. This project has students finish their EdScratch experience with a challenge that allows them to be imaginative, resourceful, and have fun. In so doing, students solidify what they have learned, strengthen their resilience and problem-solving capacities, and are emboldened to take the next steps in their coding and robotics adventures.

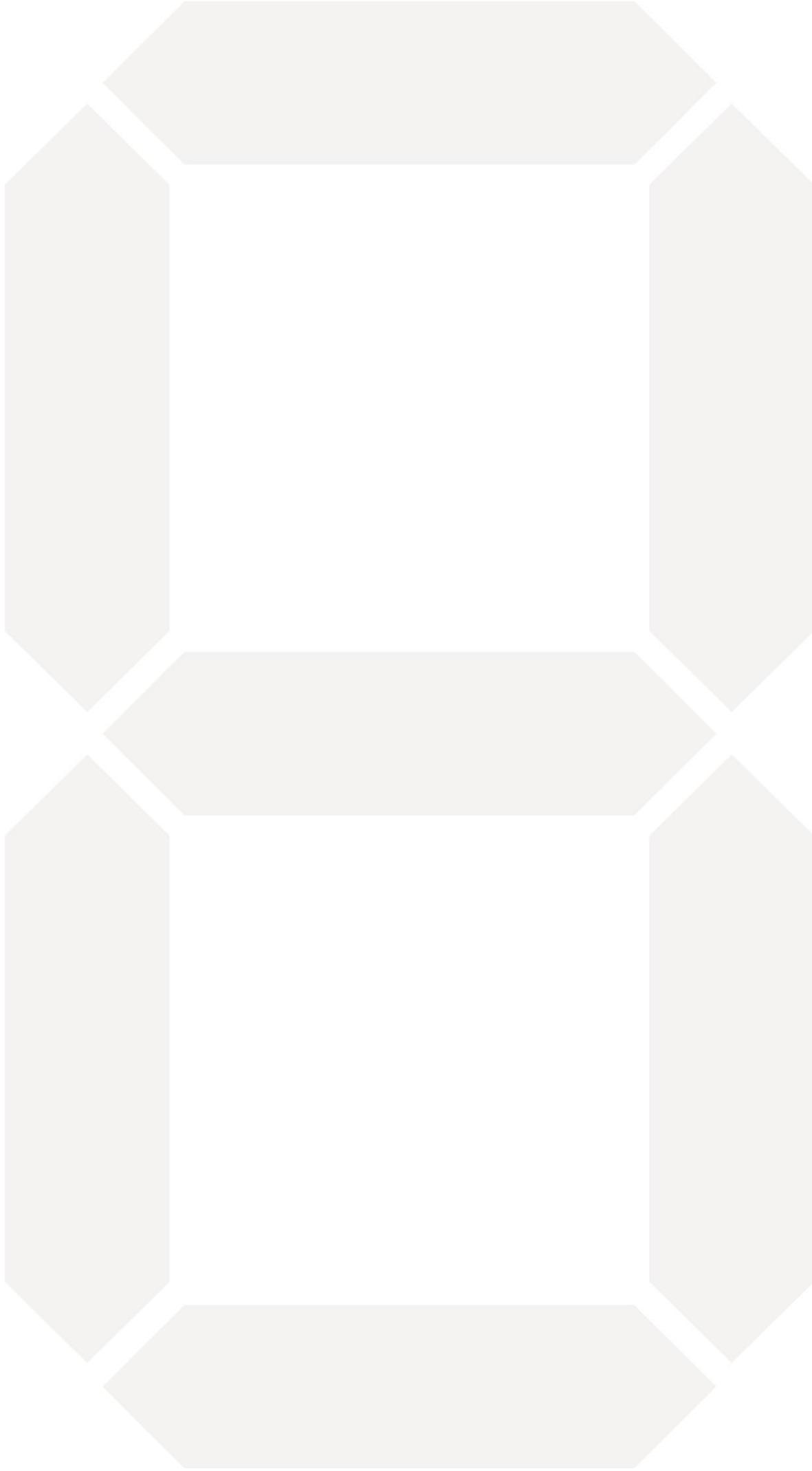
#### Tips and tricks

- This project is intended to be done in groups.
- A possible solution to the ghost hunting room can be seen at <https://www.edscratchapp.com?share=B05j61bZ>
- Remind students that unsuccessful designs can teach just as much, if not more than, successful designs. Encourage students to look at what isn't working in their creation through the lens of the design-build-test cycle. Decompose what's going on into smaller parts and problem-solve each component.

#### Answer key

The worksheet does have spaces for 'answers', but these areas do not require marking. Have students use these areas as a loose framework and working space to capture their designs. There are no 'right' answers in this project. Encourage students to be creative, take risks, apply what they've learned, and have fun!

**Appendix 1: Blank digital display number**



## Appendix 2: Calibrate obstacle detection

You can regulate the sensitivity of Edison's obstacle detection system. By making the obstacle detections system more sensitive, Edison can detect obstacles further away. By making the system less sensitive, Edison will only detect very close obstacles. Follow the instructions on this sheet to adjust your Edison's obstacle detection system.

### Read the barcode

1. Place Edison on the right side of the barcode, facing the barcode.
2. Press the record (round) button three times.
3. Wait while Edison drives forward, scanning the barcode.



Barcode – Calibrate obstacle detection

### Set maximum sensitivity

After scanning the barcode, set Edison down on a table or desk and remove any obstacles in front of Edison. Then press the play (triangle) button. Edison is now in calibration mode.

The left sensitivity is calibrated first.

1. Repeatedly press the play (triangle) button, which increases sensitivity, until the red LED on the left is flickering.
2. Repeatedly press the record (round) button, which decreases the sensitivity, until the LED completely stops flickering.
3. Press the stop (square) button to switch over to calibrate the right side.
4. Repeatedly press the play (triangle) button until the right red LED is flickering.
5. Repeatedly press the record (round) button until the LED completely stops flickering.
6. Press the stop button to complete the calibration.

### Special note: custom sensitivity

It is possible to set the distance that obstacles are detected. To do this, scan the 'calibrate obstacle detection' barcode, place an obstacle in front of Edison at the distance you want Edison to detect obstacles, press the play button and then repeat steps 1 through 6 to set the sensitivity.