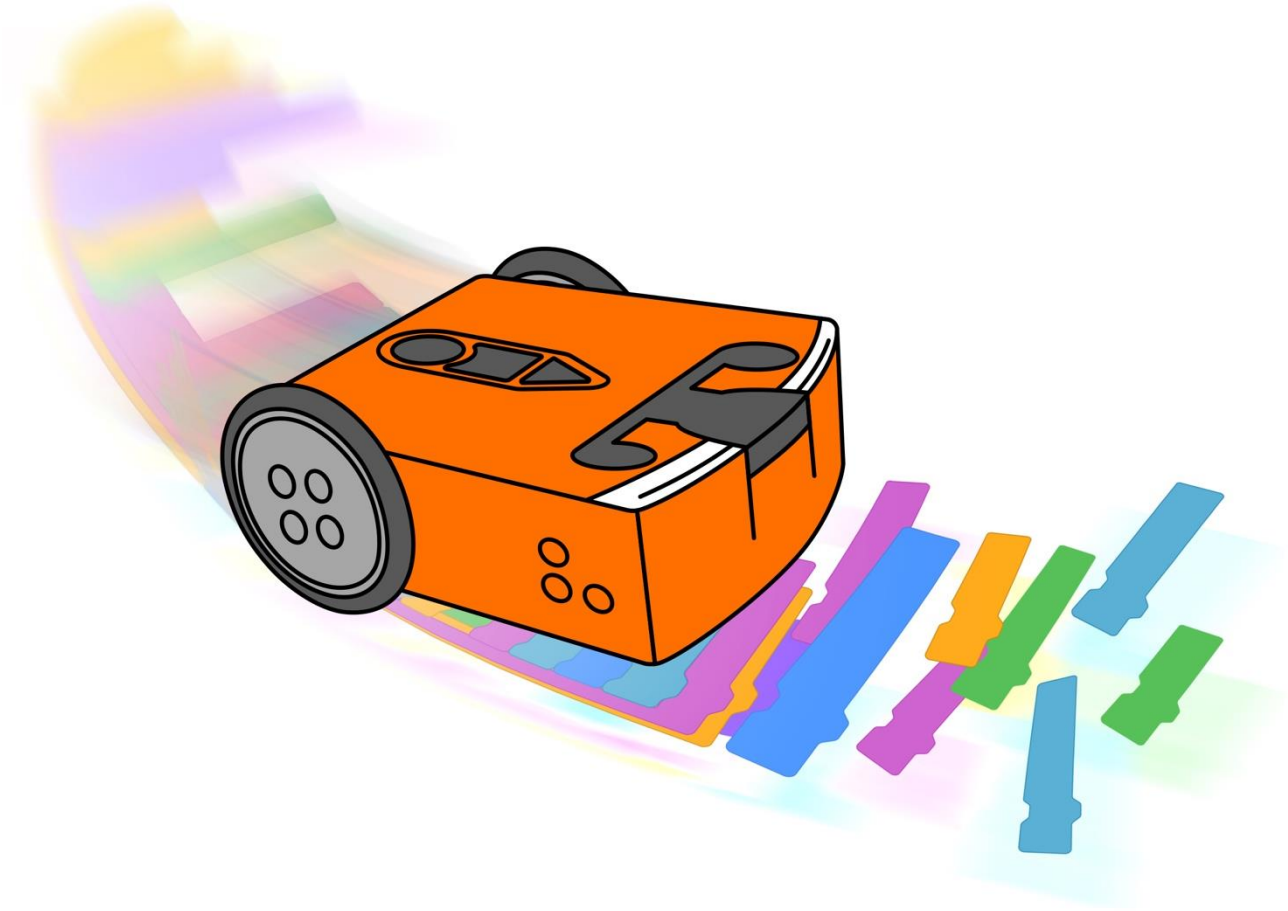
EdScratch lesson activities

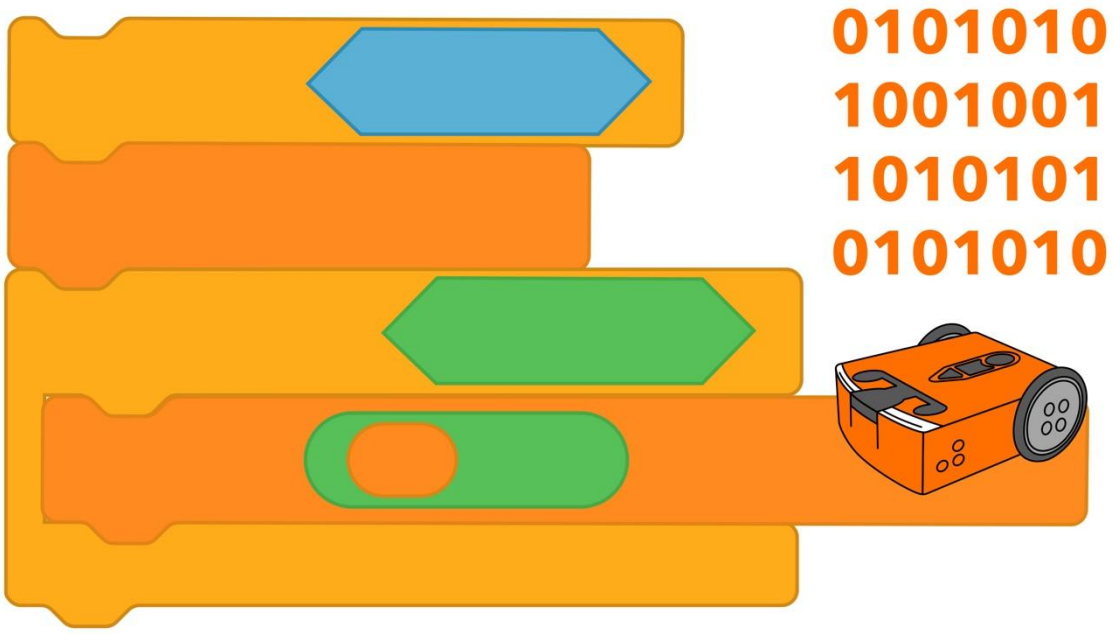
Student worksheets and activity sheets



Document number: 3.2.3.4.1 Rev 1.1

Contents

Unit 5: Versatile variables



Edison

Name_____

U5-1.1 Let's explore expressions

Believe it or not, computer languages are mostly for people, not computers.

Why is that?

Computers can actually only 'think' in numbers, not words or code blocks. For example, Edison cannot understand the blocks in EdScratch the way they look on your computer screen. The blocks need to be changed into a format that Edison can understand before the program can be downloaded. This process, which can take a bit of time, is the reason why it can take a little while for the **Program Edison** button in the pop-up window to appear when you are downloading a program from EdScratch to Edison.

Computer languages, like EdScratch, make it much easier for us to create programs. Without a computer language, you would need to write every single command using nothing but 1s and 0s!

The blocks in EdScratch make it much easier for us to give Edison the inputs we want and to write programs telling Edison what to do.

It is important to remember, however, that Edison sees all the code we program as numbers. Likewise, when Edison stores data in its memory from a sensor, this data is stored in the form of numbers.

Don't forget

Inputs are the information and instructions that you give a computer.

We can use numbers in different ways in the programs we write. Being able to use numbers, and a bit of mathematics, in computer programs allows us to get the robot to do many different things. One way we can use numbers and mathematics in EdScratch is in expressions.

Jargon buster

In code, an expression is a question that can be evaluated and resolved as being either 'true' or 'false'. Expressions can be used with other code, especially conditional code like until blocks or if blocks, to control the flow of a program.

Expressions let us compare two numbers or bits of data to each other. We can then tell Edison what to do depending on the result.

To be able to use expressions in an EdScratch program, you need to know how to read, evaluate and resolve expressions the way a computer does.

Name_____

Resolving expressions

In EdScratch, the Operators category contains the special blocks you need to write an expression. These blocks use mathematical notations (in other words, symbols) to compare the left side to the right side of the expression.

For example, 'Is A less than B?' is written in EdScratch as 'A < B' and 'Is A the same as B?' is written as 'A = B'.

Look at the list of the expressions that you can write in EdScratch:

Expression	Meaning
A = B	Is A the same as B?
A ≠ B	Is A not equal to B?
A > B	Is A greater than B?
A >= B	Is A greater than or equal to B?
A < B	Is A less than B?
A <= B	Is A less than or equal to B?

You can replace the 'A' and 'B' in the expressions with any value. You can also do computations to those values. For example, '(A + 2) > B' means 'Is A plus 2 greater than B?'

In code, expressions work in a specific order. When your expression includes computations, the expression will complete the computations first. It will then compare the left side of the expression to the right side and resolve to either 'true' or 'false.'

Because expressions have numbers and sometimes computations inside of them, it is tempting to think about them like mathematical problems which will have an answer that is a number. You need to think about expressions like a computer does, however. First, **evaluate** the expression by completing any computations on either side of the expression. Then, **resolve** the expression by comparing the left side of the notation to the right side. In other words, answer the question the expression is asking as either being 'true' or 'false'.

Why is that?

Even though expressions evaluate numbers and can contain mathematics, they always resolve to only one of two answers: either 'true' or 'false.' That's why they are so helpful when used with conditional code. Rather than having to worry about what the exact value is going to be, we can work inside sets of 'true' and 'false.'

For example, say you want a program to keep doing something as long as a value is greater than 10. Instead of writing a bunch of nested if blocks to check the exact value, you can just use one expression that says X > 10. The program will check that value, no matter what X is set to, and as long as X is greater than 10, the condition is true!

There is a special name in computer science for data which has only one of two possible values like this: we call this type of data Boolean data.

Name_____

Being able to understand what expressions mean and what they resolve to will help you follow what's going on in a program just by 'reading' it. This is an important skill in programming, known as **tracing**.

Jargon buster

Tracing code means working through a program line by line, recording important values. Being able to **trace** through a program and understand what is happening lets a programmer work out how their code should run, even when there are many different values inside that program. Tracing code can help find logical errors or bugs in the code, but it is also useful when you just need to understand what is happening in a program.

When you use programs that have values, and computations being done to those values, take a moment to trace through the program to make sure you understand what is going on. This can help you understand what should happen in the code and debug your program if things aren't working the way they should.

Try it out!

Try resolving the following expressions. First, write out what each expression means, then resolve it to either true or false.

For these questions, if A = 2 and B = 4, what does each of the following expressions mean (in other words, what question is it asking) and what does each resolve to (true or false)?

1. (A*2) = B

   Meaning: _____

   Resolves to: _____

2. A >= B

   Meaning: _____

   Resolves to: _____

3. (A+A) ≠ B

   Meaning: _____

   Resolves to: _____

4. (A-1) < (B-3)

   Meaning: _____

   Resolves to: _____

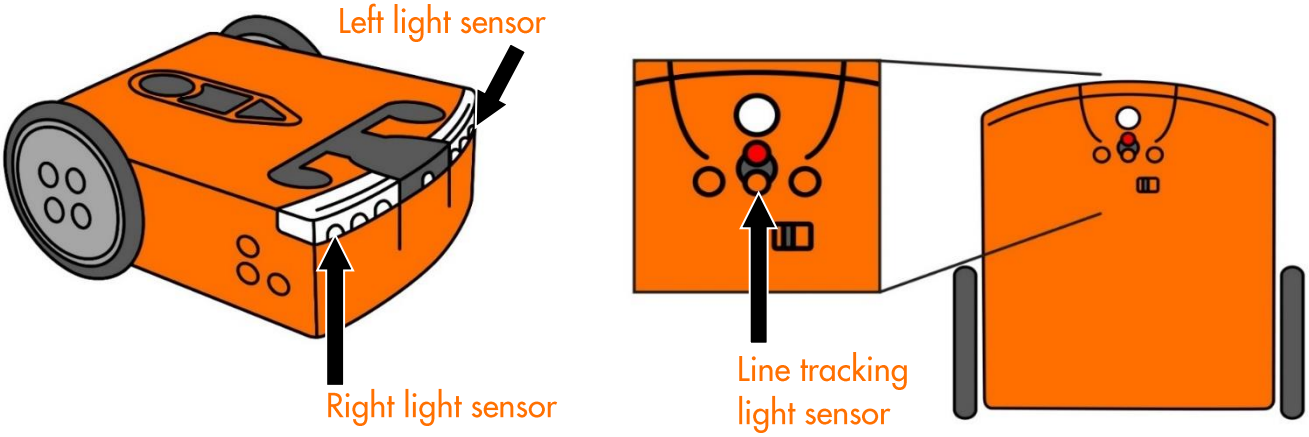
Name_____

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

Edison robots have different sensors that can detect different things. One type of sensor Edison has is the light sensors.

Task 1: Meet Edison's light sensors

Edison's light sensors are the sensors that let Edison detect and measure visible light. Edison's main light sensors are on the top of the robot, one on the right and one on the left of the robot. Edison also has a third light sensor, which is underneath the robot and works as part of the line tracking sensor. Look at your Edison robot. Do you see the different light sensors?



Much like Edison's infrared receiver can detect infrared light, the light sensors can be used to detect visible light, which is the portion of the light spectrum that people can see. All of Edison's visible light sensors work basically the same way as the light sensor in Edison's line tracker works when you use it to detect reflective and non-reflective surfaces under the robot.

Don't forget

The line tracking sensor works by shining light from the red LED in the line tracker onto the surface below the robot. The light sensor in the line tracker then measures how much of that light bounces up from the surface. Edison stores the value of the reflected light as a light reading. The more light that is reflected back to Edison, the higher the light reading.

Edison's light sensors don't need to rely only on reflected light from Edison's LEDs. We can also use these sensors to detect the visible light coming from any source near Edison. The sensors measure the detected light and store the value as a light reading. The more light that is detected, the higher the light reading.

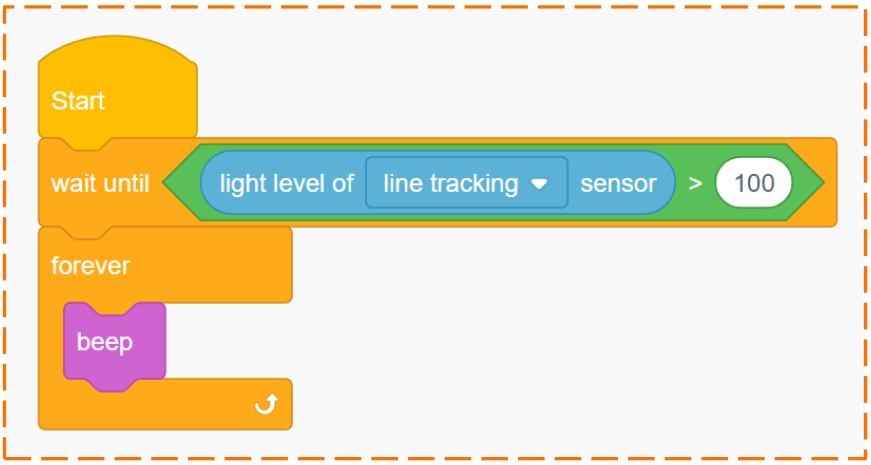
We can use the light reading from one of the light sensors as a value in an EdScratch program.

Name_____

Task 1: Light alarm

Let's use one of Edison's light sensors to make an alarm that will sound when Edison detects enough light.

Look at this EdScratch program:



This program tells Edison to wait until the light level of the line tracking sensor is greater than 100, then beep forever.

Why is that?

The light sensors measure any visible light detected and store the value as a light reading. Like all sensor data, Edison stores this value as a number. The expression in this program tells Edison to compare the value of the light reading from the line tracking sensor to the number 100.

Why 100? Remember that the more light that is detected, the higher the light reading. Edison's light sensors can give light readings with values ranging from 0 to just over 1000. That makes 100 a good starting point to test with this program.

Write the light alarm program and download it to your Edison robot. Before you press the play (triangle) button to run the program, cover up the line tracking sensor with something, like your thumb. Make sure the light sensor is completely covered up! Hold the robot so that the line tracking light sensor is pointing away from any bright lights, like the lights in the ceiling or sunshine coming in from a window.

Once you have the robot in position with the light sensor covered up, press play. When you are ready, move your thumb off of the light sensor and aim the robot towards a source of light. Once the robot detects enough light, the wait until block's condition will be met, and the code will move on to the next block, triggering the beep block 'alarm'.

Name_____

Task 2: Automatic street lamp

Have you ever seen a street lamp that comes on when it gets dark outside? Chances are that this is done using a light sensor! You can get Edison to behave this same way, turning on the robot's red LEDs when the light level gets too low and turning them off whenever there is plenty of light.

Write a program that gets your Edison to behave like a light-sensing street lamp. Your program should have Edison turn on the red LEDs whenever the robot detects very little visible light but turn the red LEDs off the rest of the time.

Hint!

- You only need to use one of Edison's top light sensors to give a light level reading for this program. Choose either the left or the right light sensor.
- Test different values to compare your light sensor reading to in order to see what works best.
- Feeling stuck? Look at the program from task 1. How can you modify it to make your automatic street lamp program?

Download and test your program in your robot. Put Edison into a spot with lots of light, then into a spot without much light. Does your program get Edison to work like an automatic street lamp?

1. What does your automatic street lamp program look like it? Write it here.

Name_____

U5-1.2a Change it up: Edison the moth

Have you ever noticed how some flying insects are attracted to bright lights? This type of behaviour is called positive phototropism. Moths exhibit positive phototropism, which is why they swarm around a bright light at night time. This kind of behaviour is also found in plants that grow towards the sun.

We can get Edison to mimic this behaviour, following the brightest light the robot detects.

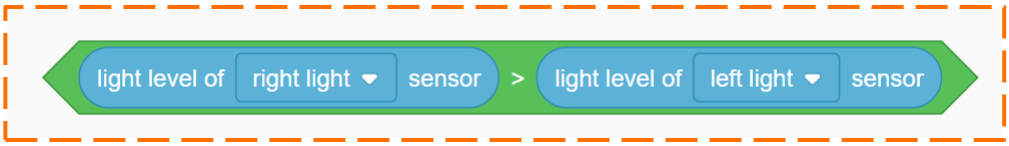
What to do

Let's program Edison to behave like a moth, following a bright light. Your program should get Edison to move towards the brightest light it detects. To write this program, you will need to use the light level readings from Edison's top two light sensors.

Don't forget

The light sensors measure any visible light detected and store the value as a light reading. Like all sensor data, Edison stores this value as a number. You can compare this value to another value in an EdScratch program and tell Edison how to react depending on the result.

You can compare the light level reading from one of Edison's light sensors to a fixed number. You can also compare the light level readings from one light sensor to the light level reading of a different light sensor:



You will need to use both of Edison's top light sensors in your program. Whenever the right light sensor is giving the higher reading, the robot should move towards the right. Whenever the left light sensor's light level reading is higher, however, the robot should move in that direction.

Write your program in EdScratch, then download and test it using your Edison robot. Use a bright light source, like a torch, and test to see if Edison follows the light around.

Hint!

If the room you are testing in is very bright, Edison won't be able to detect the light from your torch. If there is a bright source of light in the room, like a lot of sunshine coming in from a window, this may outshine your source of light and the robot might head for the other source of light instead!

Name_____

U5-1.2b Challenge up: Edison the cockroach

Some plants and animals exhibit a behaviour known as positive phototropism. These creatures are attracted to light, like moths which swarm around a bright light at night time. Other creatures, like some cockroaches, avoid light. This type of behaviour is known as negative phototropism.

Can you get Edison to mimic this behaviour, avoiding the light?

What to do

Program Edison to behave like a cockroach, moving to try to avoid any visible light. First, plan your program out using pseudocode.

1. Write down your pseudocode.

Hint!

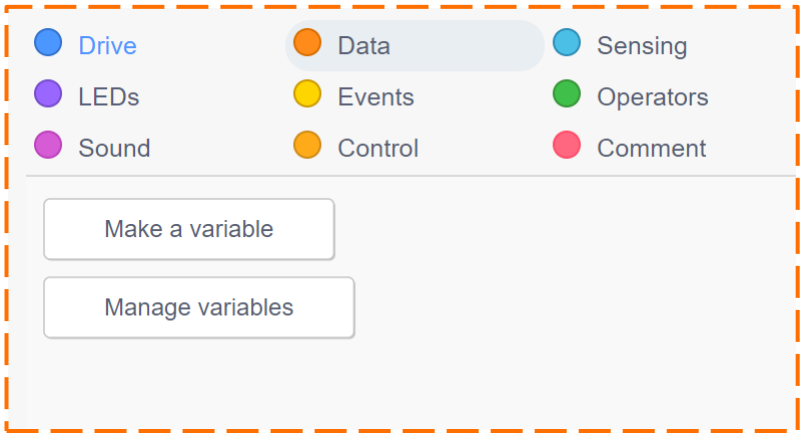
You can compare the light level reading from one of Edison's light sensors to a fixed number. You can also compare the light level readings from one light sensor to the light level reading of a different light sensor. Which approach will work best for what you are trying to do in this program?

Write your program in EdScratch using your pseudocode as a guide. Download and test your program in your Edison robot. Use a bright light source, like a torch, and test to see if Edison acts like a cockroach, avoiding the light.

Name_____

U5-1.3 Let's explore variables

In EdScratch, there is one category of blocks that looks very different to the other categories when you first open it: the Data category.



When you first click on the Data category, there are no blocks available for you to use in an EdScratch program. Instead, there are two buttons: the Make a variable button and the Manage variables button. By using the Data category, we can create **variables** to use in our EdScratch programs.

Jargon buster

A **variable** is a bit of memory that is used to store a value in a program. You can think of a variable like a container that you can use to store some bit of information in a way that will make sense to a computer.

In computer programming, we often use the same bit of information multiple times in a single program. Variables make it a lot easier to do this. Using variables in programs lets us tell the computer to store a specific bit of information inside a variable. We can then use that variable in different places in the program. Any time the computer sees the variable, it will recall whatever information is stored inside the variable.

In EdScratch, when you click the Make a variable button, a pop-up window will open up asking you to give your variable a name. Giving variables good names is important: you want the name of your variable to make sense to you and tell you what bit of information is stored inside.

Your EdScratch variable names can only contain lowercase English letters, uppercase English letters, numbers, and underscores (_). Other symbols, like exclamation marks (!) or spaces, are not allowed.

Why is that?

Variable names need to make sense to you and to the computer. Computer languages often have rules about what characters can be used in variable names. The computer can only understand variables with names that follow these rules.

Name_____

Give your variable a name that will help you identify what type of information is going to be stored inside the variable. If you want to change the name of a variable after you make it, you can do that using the Manage variables button.

Once you make and name a variable, it will appear in the Data category along with some other special blocks including the set block, the increment block and the decrement block. You can then use these blocks along with your variable in an EdScratch program.

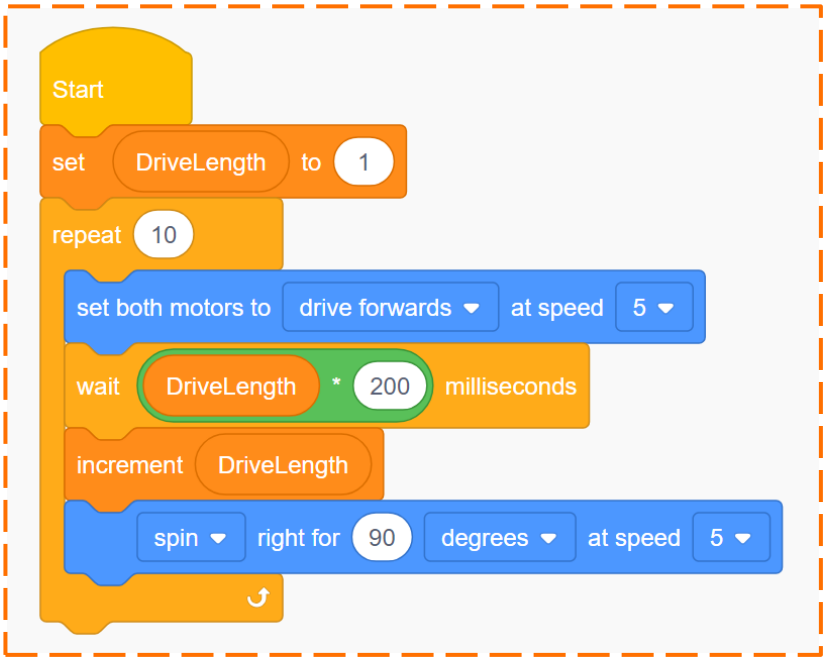
Jargon buster

In computer programming, **increment** means to increase by 1 and **decrement** means to decrease by 1.

Task 1: Trace the code

In EdScratch, we can use variables to store different values. Because these values are numbers, we can then do different things with these numbers, like compare them to other values or do computations with them.

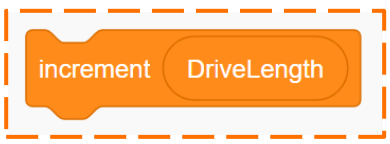
Look at the following program:



This program uses a variable named DriveLength.

It is important to note that a variable represents a value that is set somewhere in your program. That's why you always need to use code in your program to tell the computer what to set the variable to be.

This is what the set block in this program is doing. At the beginning of the program, the set block sets the variable DriveLength to be 1. The value of DriveLength isn't going to stay set to 1 forever, however. That's because this program uses another block from the Data category, the increment block:



Name_____

The increment block is inside the repeat loop and changes the variable DriveLength to a new value. What is this block changing the variable DriveLength to be?

That will depend on where in the program you are up to. In other words, it will depend on how many times the repeat loop has run.

Why is that?

One of the main reasons we use variables in programs is that a variable can hold a piece of information, even when the value of that information changes. This might sound really confusing but think about it like this:

Let's pretend you have a variable called **YearsOld** and the bit of information that this variable holds is your age. At your first birthday, **YearsOld** was set to 1. After that, each year on your birthday, **YearsOld** is incremented, which changes it to a new value: (**YearsOld** + 1).

A year after your first birthday, **YearsOld** is incremented. Because the value of **YearsOld** was 1, and since 1 + 1 = 2, the new value of **YearsOld** became 2. This repeats again on your next birthday, where the value of **YearsOld** is incremented and becomes 3.

The value of **YearsOld** changes each birthday, but the type of information doesn't change. **YearsOld** is your age, no matter how many birthdays you have!

When our program first starts, DriveLength will be set to 1, but because there is an increment block inside the looped code, the value will change each loop.

DriveLength is also being used in another place inside the looped code in this program:

The input value of this wait block will depend on the value of DriveLength and will also change each loop.

Be careful though!

While the value of the wait block will change depending on the value of DriveLength, this wait block will not change the value of DriveLength. Only a block from the Data category can change the value of a variable.

Name_____

You might also notice that this wait block is set to milliseconds, not seconds.

Why is that?

Whenever you want to use a variable or a block from the Operators category as an input in a wait block, you need to use this special millisecond wait block. Edison actually 'thinks' in milliseconds, not seconds, so using this block makes it possible to do computations which Edison will be able to understand.

Remember, 1 second = 1000 milliseconds.

Don't forget

Tracing code means working through a program line by line, recording important values.

Let's trace the program to work out what the value of the variable DriveLength and the input value of the wait block are going to be at different points when the code in this program runs.

1. Trace through the program to work out the values. Fill out the table with what the starting value of variable DriveLength will be at the beginning of each loop repetition, what the input value of the wait block will be in that loop, and what the new value of DriveLength will be after the increment block inside the loop runs. The first two rows have already been filled in for you.

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			
5			
7			
10			

Task 2: Write and run the program

What is the program going to make the robot do when you run it in Edison?

Write the program in EdScratch. Download it and run it in your Edison robot to see what it does. Then answer the questions.

Name_____

2. What does Edison do when you run this program? What 'shape' does the robot drive in?

_____

_____

3. Look at the code in the program. Explain why the robot drives in the pattern you see when you run the program in Edison. What in the code makes the robot move in that pattern?

_____

_____

_____

_____

Name_____

U5-1.3a Challenge up: Spiralling spider trap

Some spiders build webs which spiral into a central point in the middle. Can you program Edison to drive so that the robot spirals inward, like a spider laying a trap?

What to do

Write a program in EdScratch that will get Edison to drive in an inwards-spiral shape. Edison should drive, then turn, then repeat over and over, spiralling inwards.

Your program should use a variable to help you control how far Edison drives each time.

Hint!

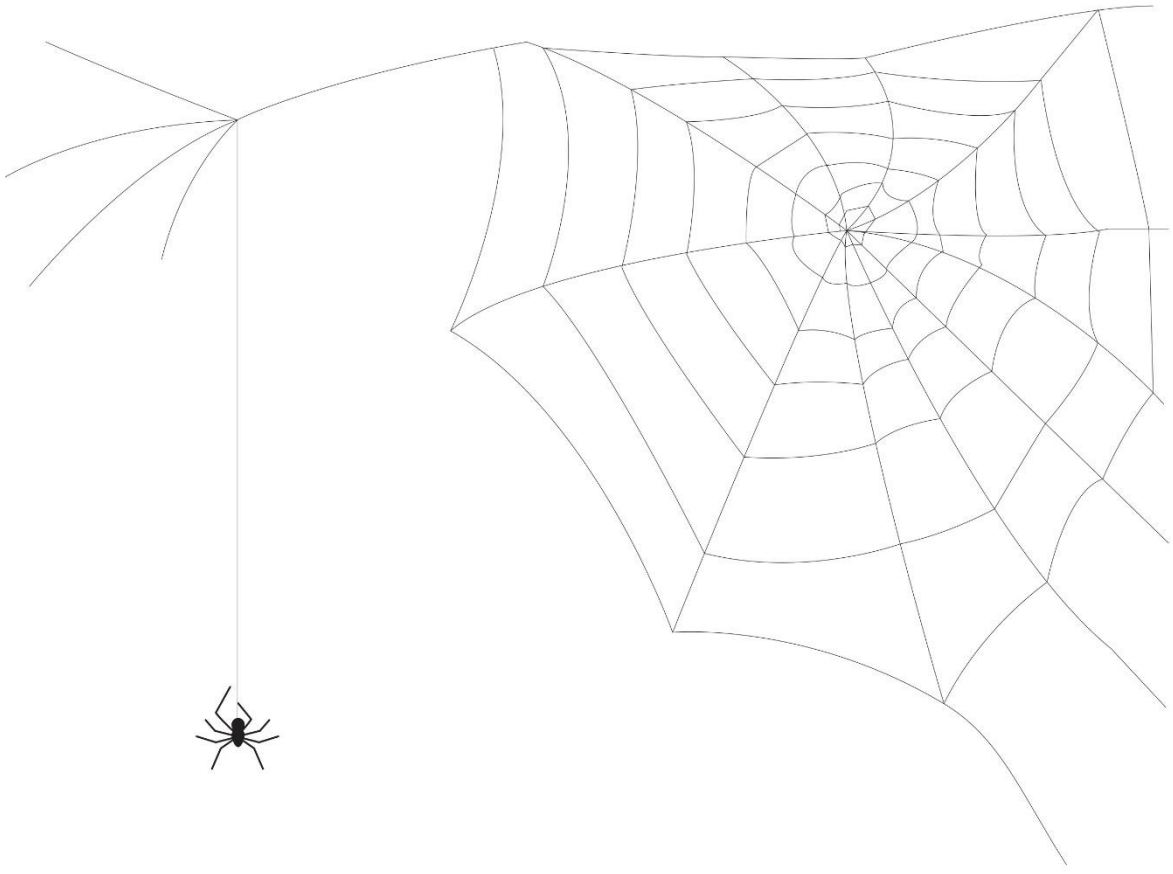
You might want to look at the program in activity U5-1.3 for some inspiration to help you write your program.

Think about how far you want Edison to drive each section compared to the previous section. You will also need to decide how far Edison should turn between each driving section.

Download your program and test it using your Edison robot. Experiment using different input values to see what works best.

Mini challenge!

Can you make Edison leave a 'sticky spider thread' trail behind the robot as it drives? Use something, like some string, to represent a spider's silk. See if you can engineer a solution so that Edison leaves a trail behind it as it drives.

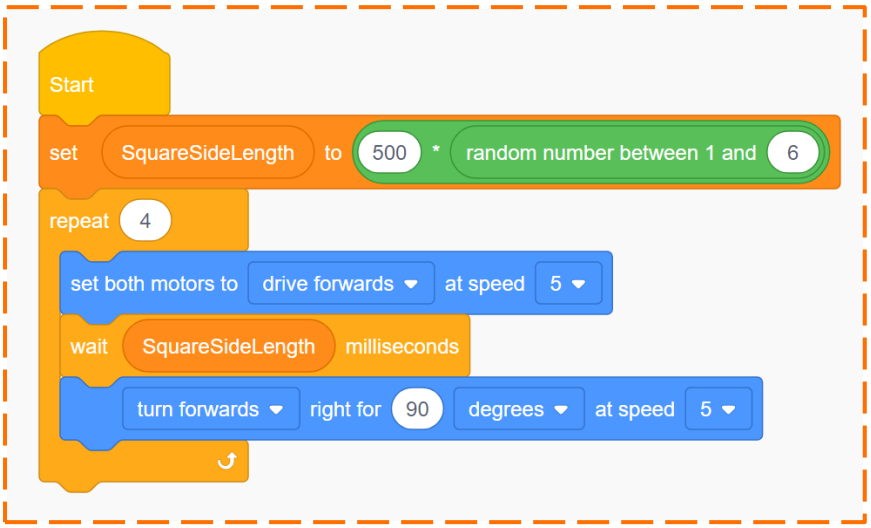


Name_____

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

Using variables lets us make programs that can do lots of interesting things.

Look at this program:



When Edison runs this program, the robot will drive in a square. But how big will that square be?

What to do

Write the program in EdScratch, then download it and run it with your Edison robot. Watch Edison drive the square. Now run the program again. Edison will again drive in a square, but chances are, this square will be a different size than the first!

Using the random number block inside the set block like this means that the value of the variable will change each time the program runs.

1. Each time this program runs, the variable SquareSideLength will be set to one of 6 different values. Fill out the table with the different values that SquareSideLength will be set to, depending on which random number is selected.

If the random number is:	SquareSideLength will be:
1	
2	
3	
4	
5	
6	

Mini challenge!

How can you tell which random number was used in the program? Think about how you can modify the random square program so that the robot will drive a random square, but also signal somehow to let you know which value was used in SquareSideLength. Modify the program and test it out in your Edison robot. Did it work?

Name_____

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

One of the most interesting ways we can use variables in EdScratch is to store and use data from Edison's sensors. Because all of Edison's sensors send back data to Edison as values, you can store a value from a sensor in a variable. You can also use sensor data to affect a variable, for example, by incrementing the variable every time a sensor detects something. By using variables and sensors together in a program, we can get Edison to react to sensor data in different ways.

Don't forget

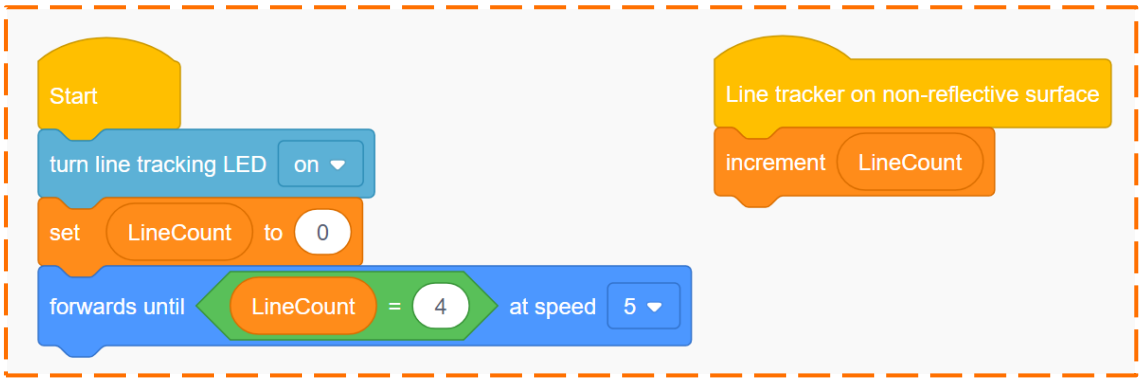
A variable is a bit of memory that is used to store some bit of information in a program.

Let's try using a variable with Edison's line tracking sensor to program Edison to drive until the robot detects four black lines.

What to do

Using Edison's line tracking sensor, you can make a program telling Edison to drive until it detects a black line. In this activity, you need to get the robot to drive over multiple black lines, only stopping once it detects the fourth black line.

Look at the following program:



1. What's going on in this program? When this program is downloaded to Edison, what will it tell the robot to do?

2. This program has a variable called LineCount in it. Why do you think the programmer chose to name the variable that?

Name_____

Try it out

Write the program in EdScratch, then download it to your Edison robot. Test the program using activity sheet U5-1 or make your own test space by marking out four parallel black lines on a white surface, such as a large poster, table or the floor. You can put the lines as far apart from each other as you like.

3. Using one of Edison's sensors with a variable lets us make all sorts of programs, using Edison in different ways. What else could you get Edison to do using variables and sensor data? Come up with at least one idea for a program that uses a variable in combination with sensor data. Describe your idea.

4. Can your idea be coded? Try to write your program in EdScratch. How did it go? Were you able to successfully write your program and test it using Edison? If so, did it work? Write about the process you took turning your idea into a program.

Name_____

U5-1.4a Challenge up: Edison the sprinter

The 400-metre dash is a sprinting event in many track and field competitions, including the Summer Olympics. This event is one of the longest 'sprint' events and that makes it different from other, shorter events. Most athletes who run the 400-metre dash know that they cannot sprint all-out with a 100% effort from start to finish. Instead, they break the race into different phases in order to balance endurance with speed.

We can program Edison to work like a 400-meter specialist, changing speed as the robot progresses through a course.

What to do

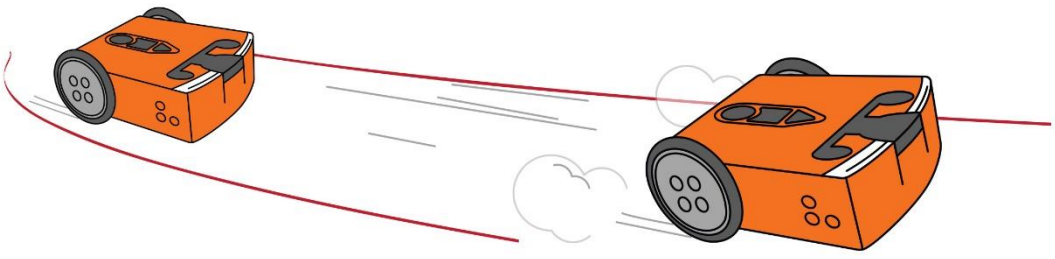
Write a program that gets Edison to behave like a 400-meter specialist, increasing speed as the robot passes markers (black lines) on a course. Your program should use Edison's line tracking sensor and a variable to help count how many lines the robot has encountered.

Test your program using activity sheet U5-1. For this activity, the fourth line is the finish line, so you need to be sure the robot crosses this line, rather than just stopping once it detects that line.

You can also make your own test space by marking out four parallel black lines on a white surface, such as a large poster, table or the floor. You can put the lines as far apart from each other as you like.

Hint!

You might want to look at the program in activity U5-1.4 for some inspiration to help you write your program.



Name_____

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

Edison's infrared (IR) light sensor is the sensor that lets Edison emit and detect infrared light. We can use the IR sensor to send messages to or receive messages from other Edison robots. One robot can send out an infrared message using its two IR LEDs which another robot's IR receiver can detect.

Don't forget

Infrared is sometimes abbreviated as IR. In EdScratch, IR message blocks are blocks that relate to Edison's infrared messaging using the infrared LEDs and infrared receiver.

To send an infrared message with your Edison robot in EdScratch, you need to use the send IR message block, which has an input parameter that allows you to send a specific message. You can change the message by changing the value of the input parameter in the send IR message block. The send IR message block has an input range of 0 to 255. In other words, Edison can send and receive 256 different 'messages'.

By using IR messages with different values, we can create a program telling Edison to react one way if the robot detects one IR message, and a different way if it detects another message. For this to work, we need to use a variable to store the data Edison receives with its IR sensor.

Let's try building and programming a flag machine using Edison's motor outputs which you can control using messaging from another Edison robot.

Don't forget

A variable is a bit of memory that is used to store some bit of information in a program. Because all of Edison's sensors send back data to Edison as values, you can store that value in a variable.

What to do

The goal of this activity is to build and program a flag machine that you can use to help make studying for a test more fun. You will need to work with a partner for this activity and use two Edison robots. One robot will be the flag machine which will have a two-sided flag with the word 'Yes' on one side of the flag and the word 'No' on the other side. This 'flag machine' robot will receive messages. The other robot will be the controller bot that will send out the messages.

There are three parts to this project: designing and building the flag machine, programming the flag machine, and programming the controller bot. When your flag machine is finished, and both robots are programmed, you will be able to work with your partner taking turns quizzing each other. You can use your controller bot to move the flag machine, signalling if your partner answered your question correctly or not!

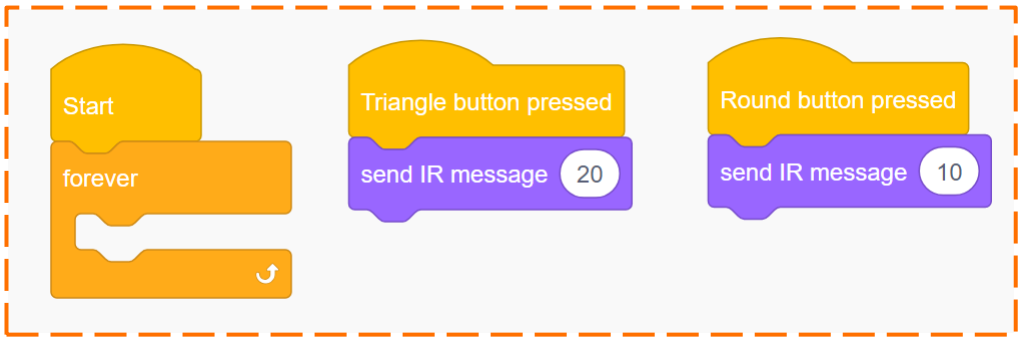
The first thing to do is program your controller bot.

Name_____

The controller bot program

The Edison robot that sends out the IR messages is the controller bot. This robot needs to be able to send out two different messages. One message will tell the flag machine to show the 'Yes' side of the flag. The other message will tell the flag machine to show the 'No' side.

Look at this program:



You can use this as your controller bot program. This program tells the Edison acting as the controller bot to send out an IR message whenever you press the round or triangle button on the controller bot. If you press the triangle button, the controller bot will send out IR message 20. If you press the round button, however, the controller bot will send out IR message 10 instead.

In your controller bot program, you can use whatever values you want between 0 and 255 for your two IR messages. The two messages need to have different values, however, so the receiving robot will be able to tell the messages apart.

Whatever values you send with your controller bot, you also need to use in your flag machine program.

1. Planning out how your two Edison robots will work together is the only way your creation will be able to function. Work with your partner to decide which button on the controller bot will do what. Write down your plan.

Round button:

- When you press the **round** button on the controller bot, it will send IR message _____ and that will tell the flag machine to show the _____ side of the flag.

Triangle button:

- When you press the **triangle** button on the controller bot, it will send IR message _____ and that will tell the flag machine to show the _____ side of the flag.

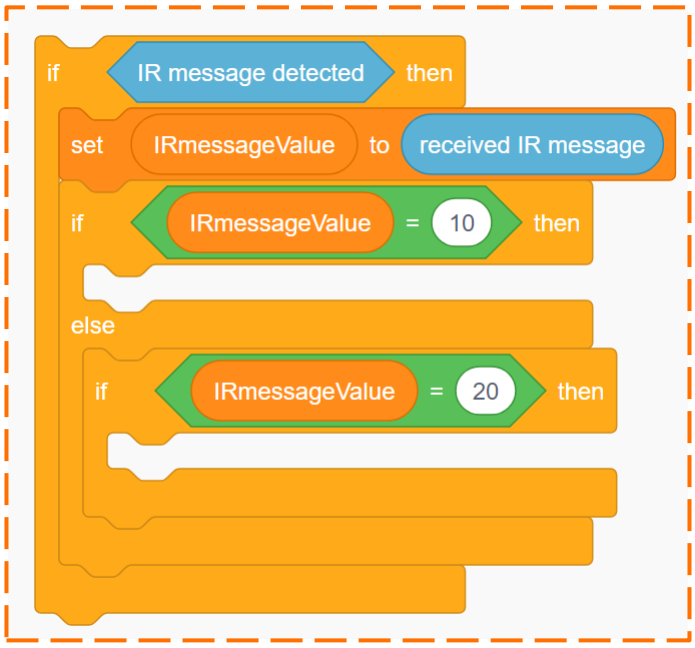
You will need to use this plan when you design your flag machine program.

Name_____

The flag machine and flag machine program

Design your flag machine to use Edison's outputs to move the two-sided flag when it receives the different IR messages. The flag machine should show the 'Yes' side of the flag when it receives one IR message and the 'No' side when it receives the other IR message. Use the plan you created with your controller bot to help you.

Look at the following code:



This isn't a full program, but you can use this code to help you build your flag machine program.

This section of code uses a variable called IRmessageValue. The value of this variable is set to be whatever the value of the received IR message is. When you work with sensor values, you should always store any values you want to use in your program in a variable. You can then have your program use that variable in whatever ways you need anywhere in the program.

Why is that?

Computers don't keep track of values from sensors unless you tell them to. When you tell a computer to store a value in a variable, you are telling the computer to remember that value.

Remember that Edison's IR detector is always on, checking for data over and over. The robot does not keep track of every value it detects forever. It only keeps the value in its working memory until it is called for in a program or replaced by a new reading. By storing the value of the IR message into a variable, you are telling Edison to hold on to that value, so you can do something with it.

The value will stay in the variable until the sensor detects a new IR message and the set block replaces the old value with the new one.

Program the flag machine using a variable to store any IR messages the robot detects. Once you have built and programmed your creation, try it out!

Name_____

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

Edison's infrared (IR) light sensor is the sensor that lets Edison emit and detect infrared light. We can use the IR sensor to send messages to or receive messages from other Edison robots. One robot can send out an infrared message using its two IR LEDs which another robot's IR receiver can detect.

Don't forget

Infrared is sometimes abbreviated as IR. In EdScratch, IR message blocks are blocks that relate to Edison's infrared messaging using the infrared LEDs and infrared receiver.

To send an infrared message with your Edison robot in EdScratch, you need to use the send IR message block, which has an input parameter that allows you to send a specific message. You can change the message by changing the value of the input parameter in the send IR message block. The send IR message block has an input range of 0 to 255. In other words, Edison can send and receive 256 different 'messages'.

By using IR messages with different values, we can create a program telling Edison to react in different ways depending on what IR message it detects. For this to work, we need to use a variable to store the data Edison receives with its IR sensor.

Let's try using two Edison robots, one driving and one navigating, to get through a treasure maze. The robots will need to use different IR messages to communicate in order to make it to the correct location.

Don't forget

A variable is a bit of memory that is used to store some bit of information in a program. Because all of Edison's sensors send back data to Edison as values, you can store that value in a variable.

What to do

You will need two Edison robots for this activity: one robot will be the navigator bot, and the other will be the driver bot. The driver bot will move through the treasure maze according to the messages it receives from the navigator bot.

There are three parts to this project: making the treasure maze, programming the driver bot, and programming the navigator bot.

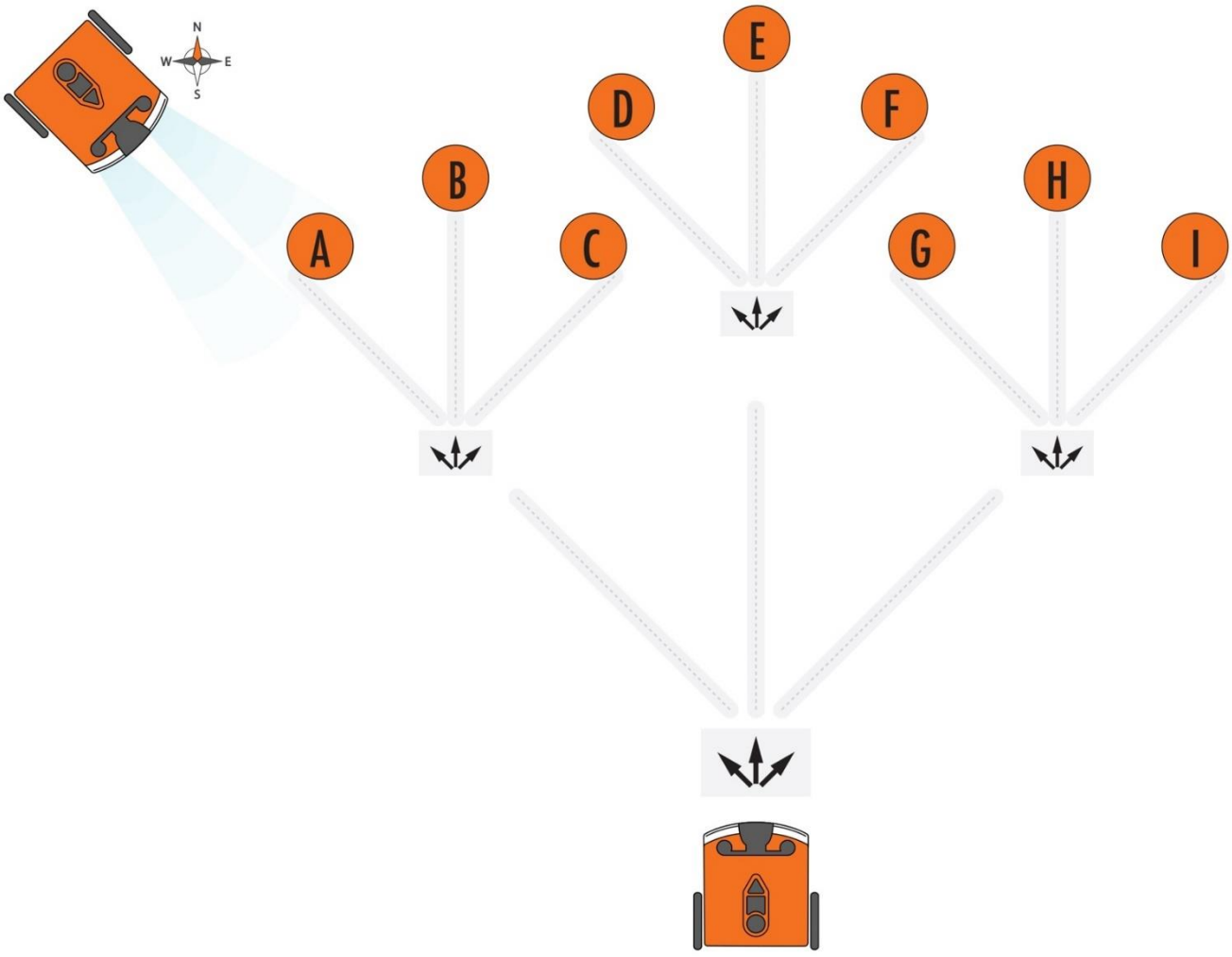
The first thing you need to do is make your treasure map maze which you will use as a test space.

Name_____

The treasure map maze

Make a branching treasure map which has many different possible end locations. Your driver bot will need to navigate this maze to get to the spot that holds the treasure. Your map needs to be big enough for Edison to drive, but not so big that the two robots cannot message back and forth to each other. You will also need a spot for the navigator bot to sit to send out commands.

The basic shape of your treasure map test space needs to look something like this:



All the paths in your map should be the same length as each other. Choose how long the paths will be, how many options come out of each junction, and how many junctions there are before the final treasure spots. Work together to design and decorate your treasure map test space.

Hint!

Use something like letters or numbers to mark the different possible treasure spots. You don't know where the real treasure will be yet!

Name_____

The navigator bot program

The Edison robot that is not going to move, but send out instructions to the driver bot is the navigator bot. You won't be able to write the navigator bot's final program until you know where on the treasure map you are sending the driver bot.

The navigator bot will send out 'commands' to the driver bot using a series of different IR messages. Each different message will tell the driver bot to do a different action. How many different messages your navigator bot needs to be able to send out will depend on your treasure map.

For example, if your map has three pathways coming out of each junction, you will need to be able to send three different IR messages: one to tell the driver bot to take the left path, one to tell the driver bot to take the middle path, and one to tell the driver bot to take the right path.

You will also need one more IR message. This message will be sent by the driver bot to the navigator bot when the driver bot is ready for the next command. This message will be the driver bot's way of saying, "I'm at the next junction. Now, where should I go?"

Each of the IR messages you send needs to have a different value so that the receiving robot will be able to tell them apart. You can use whatever values you want between 0 and 255 for your IR messages. Be sure to match up your messages across both your navigator bot and driver bot programs.

1. How many messages is your navigator bot going to need to be able to send to the driver bot? What value are you going to use for each message (including the driver bot's message to ask for the next command)? What will the driver bot need to do when it receives each separate message from the navigator bot? Work together and design your plan. Make notes to help guide you when you write your EdScratch programs.

_____

_____

_____

_____

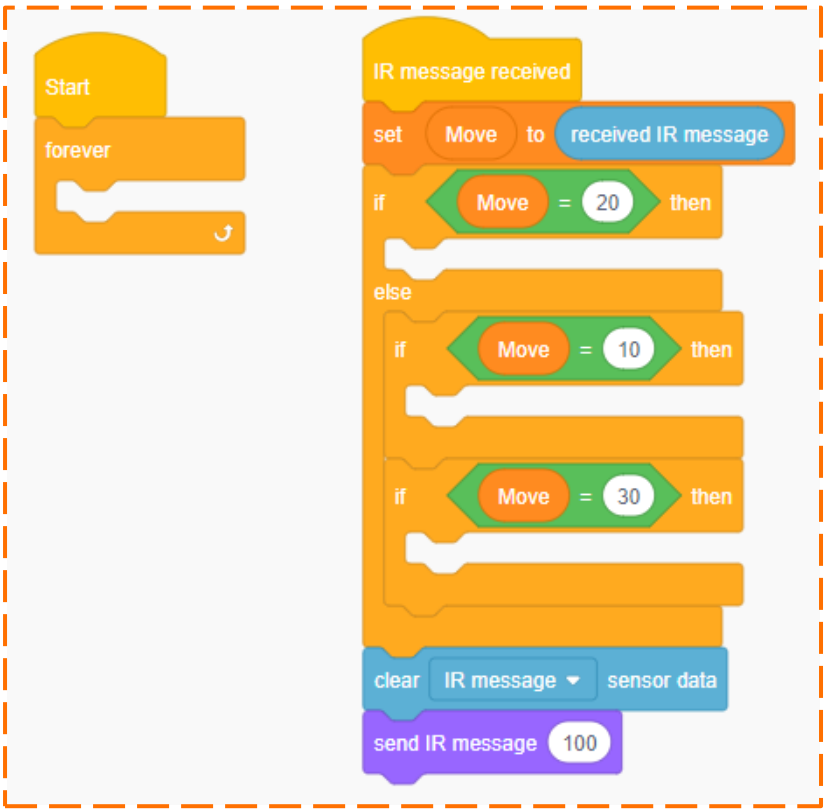
IR message value	Sending Edison (navigator or driver)	Receiving Edison will…

Name_____

The driver bot program

The Edison robot that is going to receive IR messages and drive according to those commands is the driver bot.

Look at the following code:



This isn't a full program, but you can use this code to help you build your driver bot program. This section of code uses a variable called Move. The value of this variable is set to be whatever the value of the received IR message is. When you work with sensor values, you should always store any values you want to use in your program in a variable. You can then have your program use that variable in whatever ways you need anywhere in the program.

Why is that?

Computers don't keep track of values from sensors unless you tell them to. When you tell a computer to store a value in a variable, you are telling it to remember that value.

Remember that Edison's IR detector is always on, checking for data over and over. The robot does not keep track of every value it detects forever. It only keeps the value in its working memory until it is called for in a program or replaced by a new reading. By storing the value of the IR message into a variable, you are telling Edison to hold on to that value, so you can do something with it.

The value will stay in the variable until the sensor detects a new IR message and the set block replaces the old value with the new one.

Name_____

Your driver bot program needs to check for an IR message, then store the value of that message as a variable. Depending on the value of that variable, the driver bot should take a different action to move down the corresponding path on your treasure map. Once the driver bot has arrived at the next junction, the robot needs to signal the navigator bot that it is ready for the next command.

Use your plan to help write the driver bot program. Once the driver bot is programmed correctly, it should be able to follow IR messages from the navigator bot and get to any treasure map location, no matter which one it is.

Try it out!

Once you have your treasure map ready and your driver bot programmed, choose a spot on your map to be the treasure location. Program your navigator bot to send out the IR messages in the correct sequence so that your driver bot will arrive at the treasure. Make sure you have the navigator bot wait for the driver bot's signal in between each command:

Run your programs in your robots to test them out. Did you arrive at the right treasure? If not, try to work out what went wrong. Adjust your programs and try again. Keep testing until you get your programs working.

Once you get to the treasure, choose a new spot on your map to the be the new treasure location and test again.

Don't forget

Once the driver bot is programmed correctly, it should be able to follow IR messages from the navigator bot and get to any treasure map location, no matter which one it is. You should only need to change the sequence of commands in your navigator bot program to send the driver bot from the start to any treasure location.

Name_____

U5-1.4d Change it up: The Edison chorus

Edison's infrared (IR) light sensor is the sensor that lets Edison emit and detect infrared light. We can use the IR sensor to send messages to or receive messages from other Edison robots. One robot can send out an infrared message using its two IR LEDs which another robot's IR receiver can detect.

Don't forget

Infrared is sometimes abbreviated as IR. In EdScratch, IR message blocks are blocks that relate to Edison's infrared messaging using the infrared LEDs and infrared receiver.

To send an infrared message with your Edison robot in EdScratch, you need to use the send IR message block, which has an input parameter that allows you to send a specific message. You can change the message by changing the value of the input parameter in the send IR message block. The send IR message block has an input range of 0 to 255. In other words, Edison can send and receive 256 different 'messages'.

By using IR messages with different values, we can create a program telling Edison to react in different ways depending on what IR message it detects. We can also get different Edison robots to only react to specific IR messages.

For this to work, we need to use a variable to store the data Edison receives with its IR sensor.

Let's try using IR messaging to help coordinate multiple Edison robots to play a song together in a round.

Don't forget

A variable is a bit of memory that is used to store some bit of information in a program. Because all of Edison's sensors send back data to Edison as values, you can store that value in a variable.

What to do

A round is a musical piece where two or more people (or robots!) sing or play the same melody, but each begins at a different time. While every participant is singing or playing a different part of the song, the melody still harmonises them together. In this activity, you will use multiple Edison robots to play a tune in a round.

You will need to work in a group for this activity with at least three Edison robots. One Edison will need to be the conductor bot, and all the other robots will be performer bots. The conductor bot will use IR messages to track the performer bots and tell each performer bot when to start playing music.

There are three parts to this project: choosing and arranging your song, planning your IR messages, and programming all of the robots. The first thing you need to do is choose a song for the robots to play.

Name_____

Choose and arrange your song

Together with your group, you need to choose which song you are going to perform. Many nursery rhyme songs work well in a round, such as *Row, Row, Row Your Boat*. You also need to decide at which points in the song you will have each new performer bot join in.

1. What song are your performer bots going to play and when will each new performer bot join in? Write down your plan to help you when you go to program each performer bot to play your song.

_____

_____

_____

_____

Optional: To program your song into EdScratch, you will need to write out each note. If you want, you can use this space to write down your music to make it easier to program into your performer bots.

Name_____

The IR message plan

One of your Edison robots will be your conductor bot. The conductor bot is not going to play the song, but instead manage all of the performer bots using IR messages. The conductor bot is going to need to both send and receive IR messages.

The conductor bot will need to send out messages to signal each performer bot to start playing one at a time. When a performer bot gets up to the place in the song where the next robot should join in, that performer bot needs to signal the conductor bot using a different IR message. The conductor bot will then know to get the next performer bot playing.

This will make a pattern in the IR messages that your conductor bot sends and receives. The conductor bot needs to send an IR message to start the first performer bot playing, wait for the signal from that performer bot, then send the next IR message to get the next performer playing. The conductor bot then needs to repeat this pattern of waiting for an IR message, checking which performer that message came from, then signalling the next performer over and over until all the performer bots have begun to play.

How many different messages your conductor bot needs to be able to send and receive will depend on how many performer bots are in your chorus. Each of the IR messages you send needs to have a different value so that the receiving robots will be able to tell them apart. You can use whatever values you want between 0 and 255 for your IR messages. Be sure to match up your messages across all of your robots' programs.

2. How many messages is your conductor bot going to need to be able to send and receive? What value are you going to use for each message? Which robot is each message for and what does the receiving robot need to do once it detects that IR message? Work together and design your plan. Make notes to help guide you when you write your EdScratch programs.

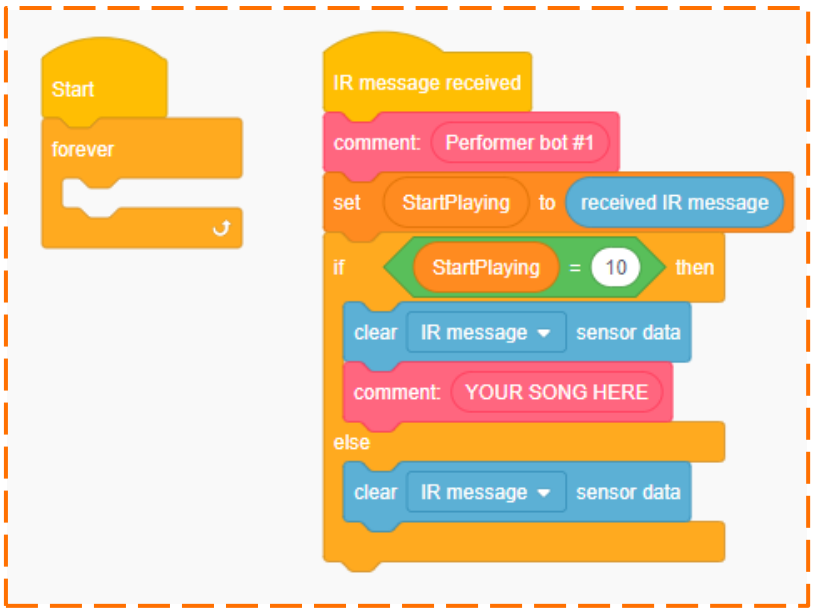
IR message value	Sending Edison (conductor or which performer)	Receiving Edison (conductor or which performer)	Receiving Edison's action

Name_____

Program all the robots

All of the Edison robots that are going to play the song are the performer bots. Each performer bot's program will tell that robot to wait to start playing until it receives an IR message from the conductor bot. Since we don't want all of the robots to start playing at the same time, each performer bot will need to start playing only if it detects its assigned IR message.

Look at the following code:



This isn't a full program, but you can use this code as an example to help you build your performer bots' programs. This section of code uses a variable called StartPlaying. The value of this variable is set to be whatever the value of the received IR message is. When you work with sensor values, you should always store any values you want to use in your program in a variable. You can then have your program use that variable in whatever ways you need anywhere in the program.

Why is that?

Computers don't keep track of values from sensors unless you tell them to. When you tell a computer to store a value in a variable, you are telling it to remember that value.

Remember that Edison's IR detector is always on, checking for data over and over. The robot does not keep track of every value it detects forever. It only keeps the value in its working memory until it is called for in a program or replaced by a new reading. By storing the value of the IR message into a variable, you are telling Edison to hold on to that value, so you can do something with it.

The value will stay in the variable until the sensor detects a new IR message and the set block replaces the old value with the new one.

Name_____

Each of your performer bots' programs needs to check for an IR message, then store the value of that message as a variable. Depending on the value of that variable, the performer bot should either start playing or just clear the data and go back to waiting for its IR message.

Use your plan to help write a program for each of your performer bots.

Hint!

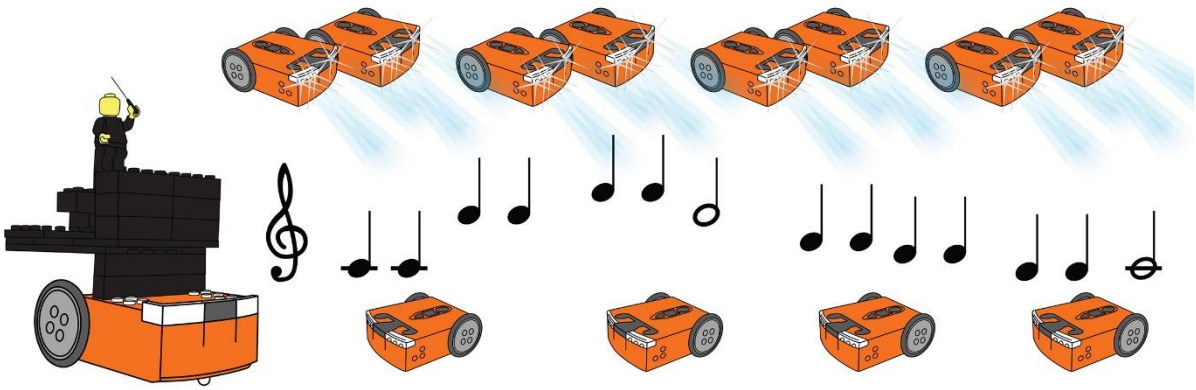
Be sure each performer bot has the right send IR message block in the correct spot in the song!

You also need to program the conductor bot using your plan as a guide. Remember, your conductor bot will also need to check for an IR message, then store the value of that message as a variable. Depending on the value of the variable, the conductor bot will need to send out a different IR message to get the next performer bot to start.

Hint!

Whenever you receive IR data and store that value as a variable, you should then clear the IR data using the clear IR message data block as the next action in your program.

Once all your bots are programmed, test out your programs with your robots. Did each robot play when it was supposed to? If not, try to work out what went wrong. Adjust your programs and try again. Keep testing until you get your programs working.



Activity sheet U5-1: Four lines