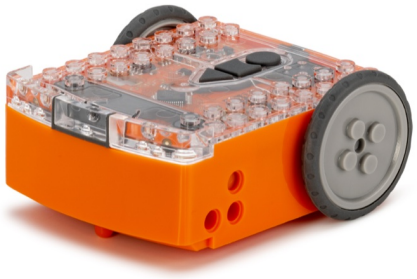


Contents

- Drive . . . 1
- LEDs . . . 22
- Sound . . . 27
- Variables . . . 32
- Events . . . 44
- Control . . . 61
- Sensing . . . 76
- Operators . . . 94
- Comment . . . 114
- Advanced . . . 116
- Functions . . . 121

EdScratch V3 Block Help

Edison V3 Robot

Drive

The Forwards for block

The Backwards for block

The Left for block

The Right for block

The Forwards until block

The Backwards until block

The Left until block

The Right until block

The Set both motors to block

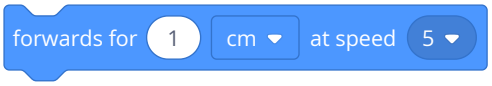
The Set right motor to block

The Set left motor to block

The Stop block

The Forwards for block

The forwards for block drives the wheels to move the robot in the forward direction.



The block has three parameters:

Value - Can be a constant (typed into the input and does not vary), a variable, a sensing or an operator block.

Units - There are three options that can be selected: cm, inch and seconds.

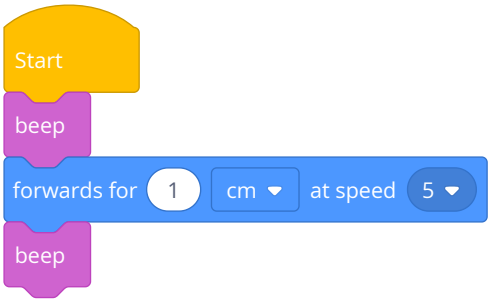
Speed - The speed can be either selected from 1 to 10 (as a constant value), a variable, a sensing or an operator block.

Note:

The forwards for block must complete before the next block can start. For example, using the forwards for block does not allow other blocks to execute simultaneously. To do this see the set both motors to block.

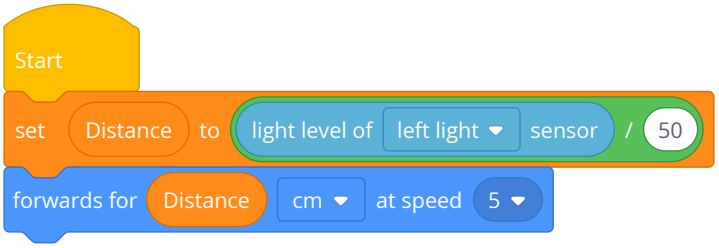
Example uses:

Example 1: Beep, then drive the Edison robot forward 15 cm at speed 1, then beep again.



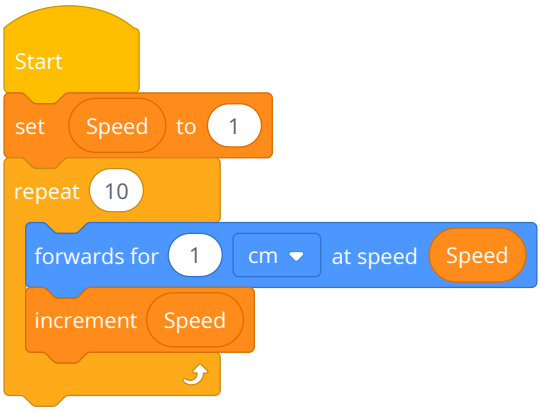
In this example program, Edison beeps, but does not start driving until the beep has finished playing. Also, the second beep does not start until Edison has stopped driving.

Example 2: Setting the drive distance using a variable and data from a sensor



In this example program Edison sets the Distance variable to the light level from the left light sensor (a value from 1 to 1,000) divided by 50. The Edison then drives forward for the value in the (Distance) variable at speed 5. The result is that Edison drives further under brighter light and drives shorter under dimmer light.

Example 3: Changing the drive speed using a variable.

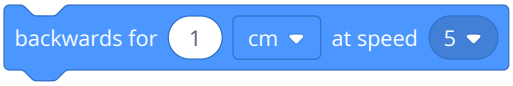


In this example program, the variable Speed is set to 1. The program then enters a repeat loop for 10 times. Through each loop the Edison drives forward 5 cm at the speed in the variable. Initially, the value in the Speed variable is 1, so the Edison drives forward for 5cm at speed 1. The next block in the repeat increments (+1) the Speed variable. In the second repeat of the forwards block the value in the Speed variable is 2, so the Edison drives forward for 5cm at speed 2.

This process repeats the full 10 times of the repeat loop, causing Edison to drive faster and faster until stopping when the program ends.

The Backwards for block

The backwards for block drives the wheels to move the robot in the backward direction.



The block has three parameters:

Value - Can be a constant (typed into the input and does not vary), a variable, a sensing or an operator block.

Units - There are three options that can be selected: cm, inch and seconds.

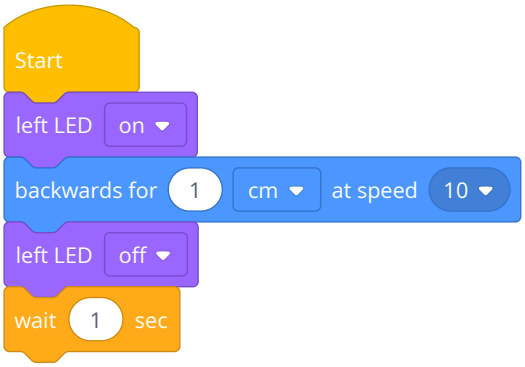
Speed - The speed can be either selected from 1 to 10 (as a constant value), a variable, a sensing or an operator block.

Note:

The backwards for block must complete before the next block can start. For example, using the backwards for block does not execute other blocks simultaneously. To do this see the set both motors to block.

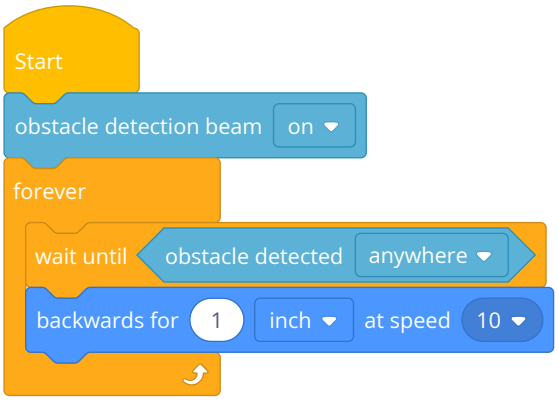
Example uses:

Example 1: LED lights on while driving



In this example program, the left LED is set to on and stays on while the Edison drives backwards for one second. The left LED is set to off after the Edison has stopped driving. The program then waits for one second before ending.

Example 2: When there is an obstacle drive backwards



In this example program, the obstacle detection beam is first activated. The program then enters a forever loop. Inside the loop the wait until block stops the program from progressing until an obstacle is detected anywhere, when this happens the Edison drives backwards one inch at speed 10. This behaviour repeats forever in a loop. This results in Edison moving backwards from objects placed in front of it.

The Left for block

The left for block drives the wheels to move the robot in the left direction.

The block has four parameters:

Movement - There are three options that can be selected: spin, turn forwards and turn backwards.

Value - Can be a constant (typed into the input and does not vary), a variable, a sensing or an operator block.

Units - There are two options that can be selected: degrees and seconds.

Speed - The speed can be either selected from 1 to 10 (as a constant value), a variable, a sensing or an operator block.

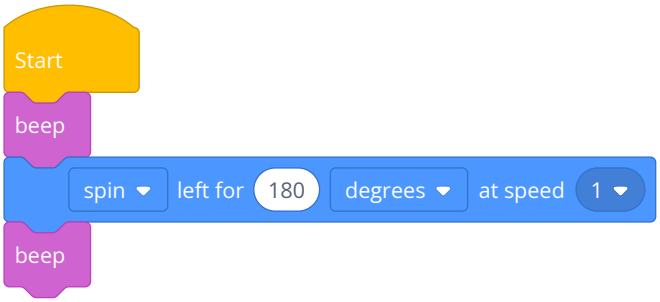
Note:

Turning and spinning is more accurate at lower speeds. This is because the wheels can lose traction and slip at high speeds. Speed 1 is best when accurate turning is needed.

The left for block must complete before the next can block start. For example, using the left for block does not execute other blocks simultaneously. To do this see the set both motors to block.

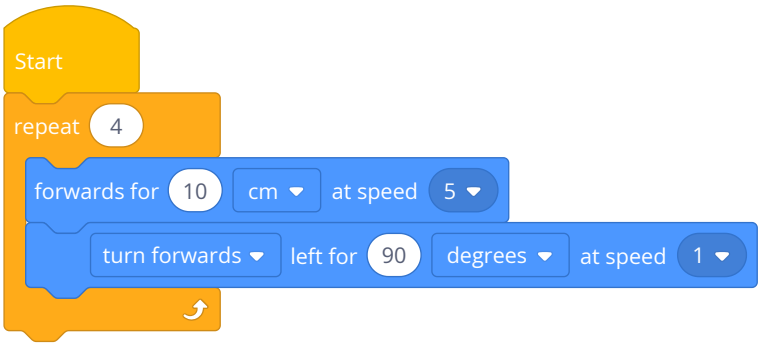
Example uses:

Example program 1: beep, spin, beep



In this example program, Edison beeps, but does not start spinning until the beep has finished playing. Also, the second beep does not start until Edison has stopped spinning.

Example program 2: Drive in a square



In this example program, the repeat loop repeats driving forwards for 10cm at speed 1 and turning forward left for 90 degrees four times. This results in Edison driving in a square pattern.

The Right for block

The right for block drives the wheels to move the robot in the right direction.

The block has four parameters:

Movement - There are three options that can be selected: spin, turn forwards and turn backwards.

Value - Can be a constant (typed into the input and does not vary), a variable, a sensing or an operator block.

Units - There are two options that can be selected: degrees and seconds.

Speed - The speed can be either selected from 1 to 10 (as a constant value), a variable, a sensing or an operator block.

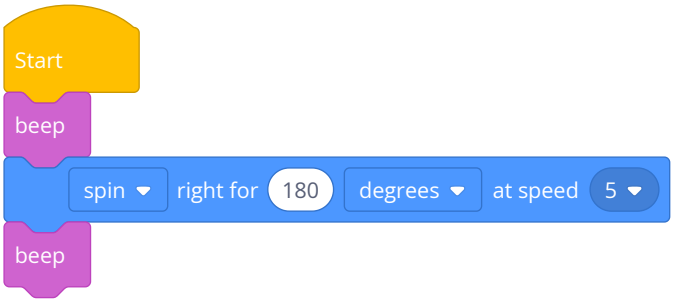
Note:

Turning and spinning are more accurate at lower speeds. This is because the wheels can lose traction and slip at high speeds. Speed 1 is best when accurate turning is needed.

The right for block must complete before the next block can start. For example, using the right for block does not allow other blocks to simultaneously execute. To do this see the set both motors to block.

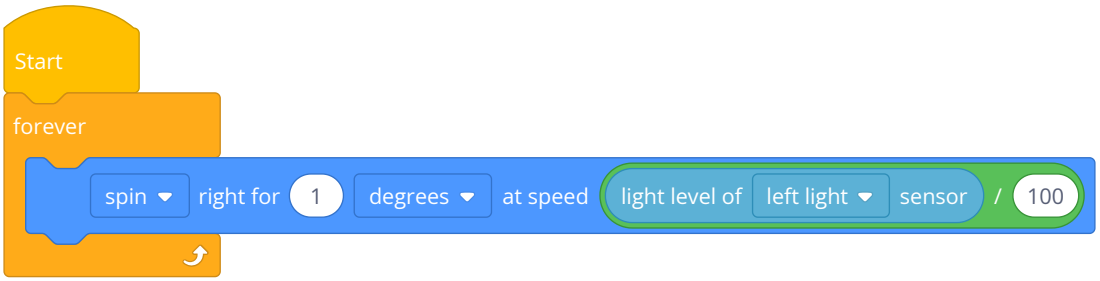
Example uses:

Example program 1: Beep, spin, beep



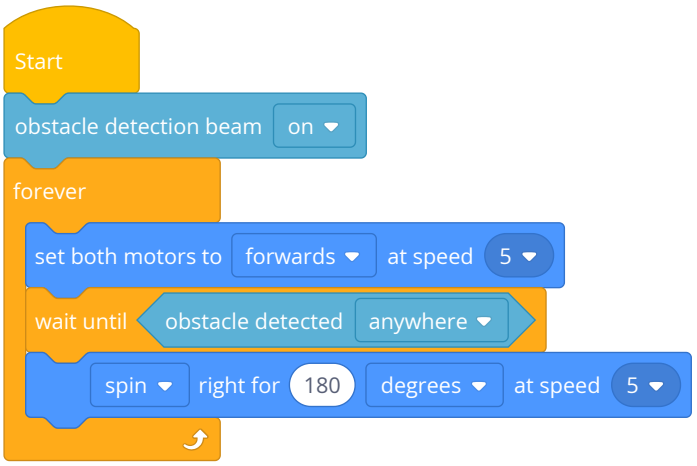
In this example program Edison beeps, but does not start spinning until the beep has finished playing. Also, the second beep does not start until Edison has stopped spinning.

Example 2: Spin right with the speed controlled by the light level



In this example program Edison spins right inside a forever loop. Each time the right for block executes, Edison spins 1 degree. The speed is set by the light level reading from the left light sensor (1 to 1,000) and is divided by 100 (result 0 to 10). The result is that Edison spins faster under brighter light and slower under dimmer light.

Example program 3: Avoid obstacles



In this example program, the obstacle detection beam is activated. The program then enters a forever loop where both motors are set to drive forwards. The program then waits until an obstacle is detected anywhere. When this happens, the program progresses to the next block and the Edison robot spins right for 180 degrees at speed 1. This results in Edison driving away from obstacles in its path.

The Forwards until block

The forwards until block drives the wheels to move the robot in the forwards direction until the specified condition is true. Once the specified condition is true the wheels stop and the program progresses to the next block.

The block has two parameters:

Conditional input - The conditional input accepts a diamond shaped block that can be either a sensing or operator block.

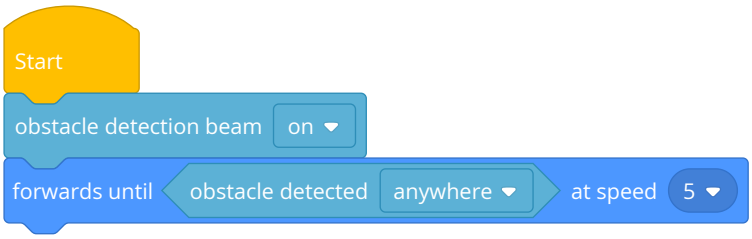
Speed - The speed can be either selected from 1 to 10 (as a constant value) a variable, a sensing or an operator block.

Note:

The forwards until block must complete before the next block can start. For example, using the forwards until block does not execute other blocks simultaneously. To do this see the set both motors to block.

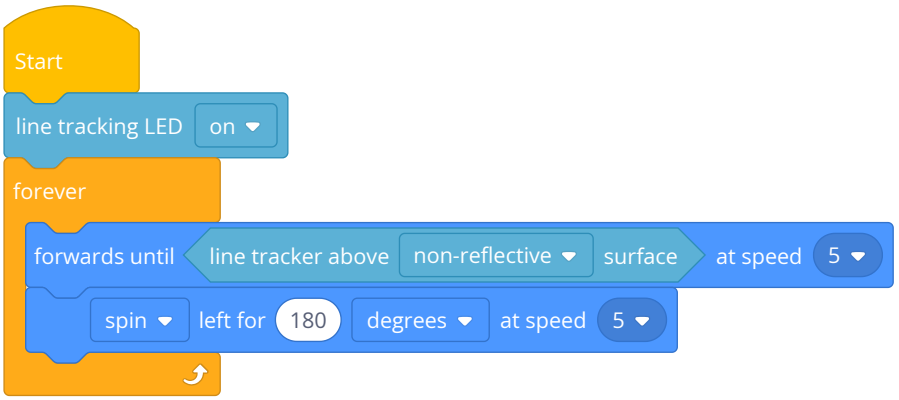
Example uses:

Example program 1: Stop for an obstacle



In this example program, the obstacle detection beam is activated, then the robot drives forward at speed 5 until an obstacle is detected. When this happens, the robot stops moving and the program ends.

Example program 2: Bounce in borders



In this example program, the line tracker LED is activated, then the program enters a forever loop. Inside the loop the forwards until block drives the robot forward at speed 1 until the line tracker is on a non-reflective surface (i.e. black). When this happens, the next block starts which spins the robot left for 180 degrees at speed 5. The program then loops back to the forwards until block. This results in Edison driving away from detected black lines.

The Backwards until block

The backwards until block drives the wheels to move the robot in the backwards direction until the specified condition is true. Once the specified condition is true the wheels stop and the program progresses to the next block.

The block has two parameters:

Conditional input - The conditional input accepts a diamond shaped block that can be either a sensing or operator block.

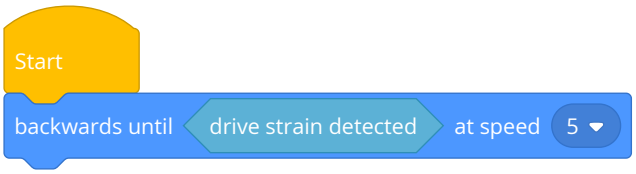
Speed - The speed can be either selected from 1 to 10 (as a constant value), a variable, a sensing or an operator block.

Note:

The backwards until block must complete before the next block can start. For example, using the backwards until block does not execute other blocks simultaneously. To do this see the set both motors to block.

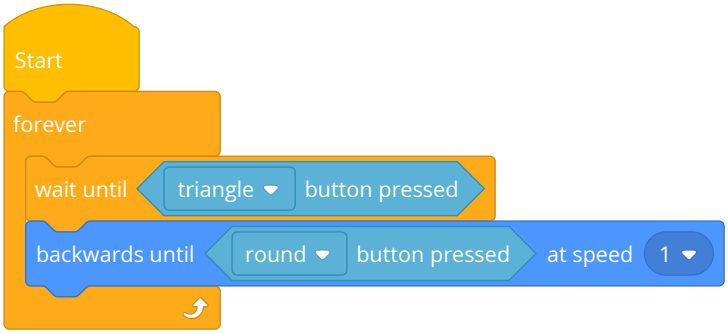
Example uses:

Example program 1: Stop when drive is strained



In this example program, the robot drives backwards at speed 10 until the wheels are strained (are unable to rotate or become stuck). When this happens, the motors stop and the program ends.

Example program 2: Drive control with buttons



In this example, the program starts with a forever loop. The first block in the loop waits until the triangle button is pressed. When this happens, the robot drives backwards at speed 1 until the round button is pressed. When this happened the robot's wheels stop and the program loops back to waiting for the triangle button to be pressed.

The Left until block

The left until block drives the wheels to move the robot in the left direction until the specified condition is true. Once the specified condition is true the wheels stop and the program progresses to the next block.

The block has three parameters:

Movement - There are three options that can be selected: spin, turn forwards and turn backwards.

Conditional input - The conditional input accepts a diamond shaped block that can be either a sensing or operator block.

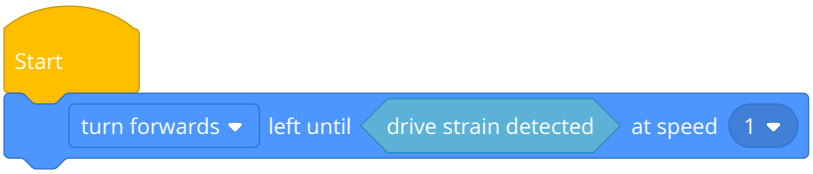
Speed - The speed can be either selected from 1 to 10 (as a constant value), a variable, a sensing or an operator block.

Note:

The spin left until block must complete before the next block can start. For example, using the spin left until block does not execute other blocks simultaneously. To do this see the set both motors to block.

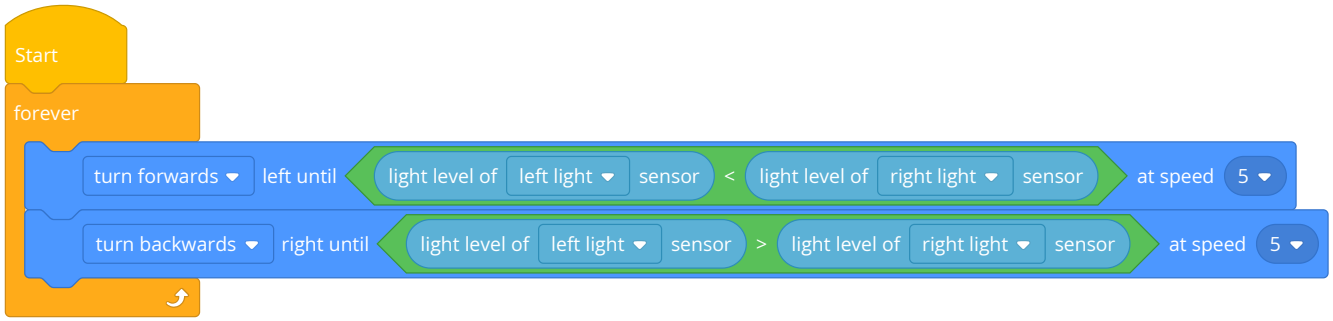
Example uses:

Example program 1: Stop when drive is strained



In this example program, the robot drives forward left (right wheel moving) at speed 1 until the wheel becomes strained (unable to rotate or becomes stuck). When this happens, the motor stops and the program ends.

Example program 2: Follow light



In this example, the program starts with a forever loop. The first block in the loop turns the robot left at speed 5 until the left light level is less than the right light level. When this happens the motor stops and the next block starts. The next block turns the robot right at speed 5 until the left light level is greater than the right light level. The result of this is that the Edison robot drives towards light.

The Right until block

The right until block drives the wheels to move the robot in the right direction until the specified condition is true. Once the specified condition is true the wheels stop and the program progresses to the next block.

The block has three parameters:

Movement - There are three options that can be selected: spin, turn forwards and turn backwards.

Conditional input - The conditional input accepts a diamond shaped block that can be either a sensing or operator block.

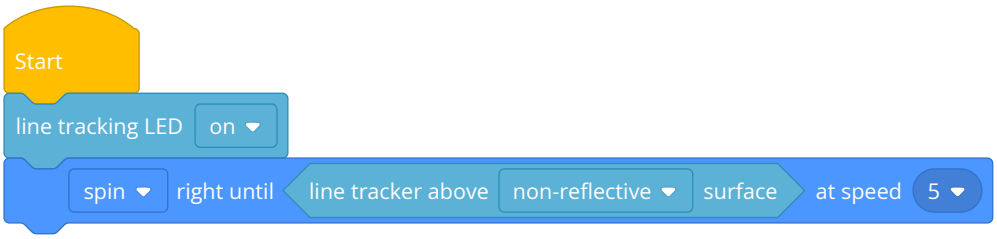
Speed - The speed can be either selected from 1 to 10 (as a constant value), a variable, a sensing or an operator block.

Note:

The right until block must complete before the next block can start. For example, using the spin right until block does not execute other blocks simultaneously. To do this see the set both motors to block.

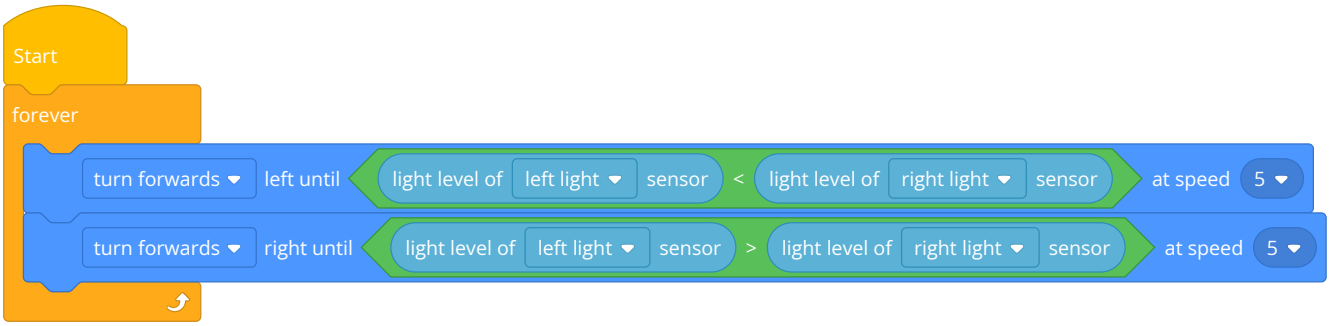
Example uses:

Example program 1: Stop on black surface



In this example, program the robot spins right at speed 5 until the line tracker detects a non-reflective surface (black). When this happens that motor stops and the program ends.

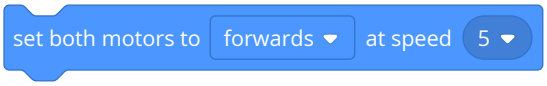
Example program 2: Follow light



In this example, the program starts with a forever loop. The first block in the loop turns the robot left at speed 5 until the left light level is less than the right light level. When this happens the motor stops and the next block starts. The next block turns the robot right at speed 5 until the left light level is greater than the right light level. The result of this is that the Edison robot drives towards light.

The Set both motors to block

The set both motors to block drives the wheels to move the robot in the selected direction and speed. As this drive block sets the motors, the program can continue straight to the next block. This is different to the drive blocks that have a distance, time or angle to complete before the program continues to the next block.



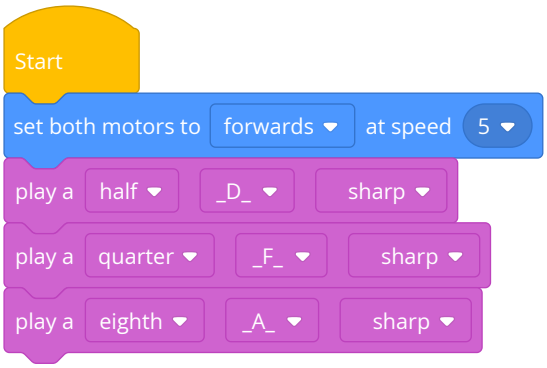
The block has two parameters:

Movement - There are six options that can be selected: drive forwards, drive backwards, spin robot left, spin robot right, turn robot left and turn robot right.

Speed - The speed can be either selected from 1 to 10 (as a constant value), a variable, a sensing or an operator block.

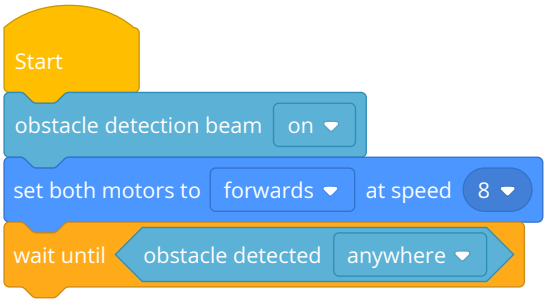
Example uses:

Example program 1: Music while driving



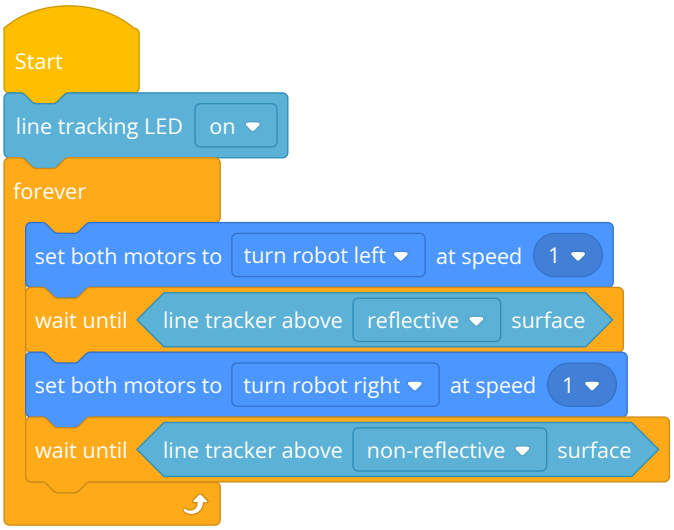
In this example program the Edison robot is set to drive forward, then three notes are played while the robot is still driving. After the last note finishes playing the program ends and the Edison robot stops driving.

Example program 2: Drive until an obstacle



In this example program the obstacle detection beam is activated, then both motors are set to drive forwards at speed 8. The next block waits until any obstacle is detected. When this happens, the program ends and the robot stops driving.

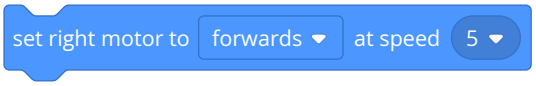
Example program 3: Follow a line



In this example program, the line tracker LED is activated and the program enters a forever loop. Inside the loop both motors are set to turn robot left at speed 1, then the wait until block holds the program there until the line tracker detects a reflective surface (white). When that happens, both motors are then set to turn robot right at speed 1 and the wait until block hold the program there until the line tracker detects a non-reflective surface (black). When that happens the program loops back to the start of the forever loop and repeats. This results in Edison following a non-reflective line.

The Set right motor to block

The set right motor to block drives the wheels to move the robot in the selected direction and speed. As this drive block sets the motors the program can continue straight to the next block. This is different to the drive blocks that have a distance, time or angle to complete before the program moves to the next block.



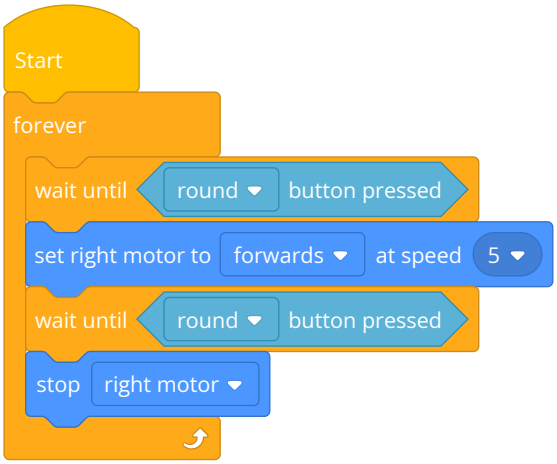
The block has two parameters:

Movement - There are two options that can be selected: forwards and backwards.

Speed - The speed can be either selected from 1 to 10 (as a constant value), a variable, a sensing or an operator block.

Example uses:

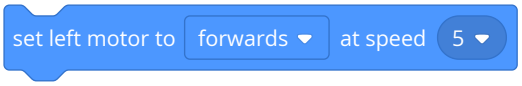
Example program 1: Motor on and off with a button press



In this example, the program enters a forever loop. The first block in the loop waits until the round button is pressed. When this happens, the right motor is set to forwards at speed 5, the program then progresses to another wait until round button pressed block. When this happens, the right motor is stopped and the program loops back to the start of the forever loop.

The Set left motor to block

The set left motor to block drives the wheels to move the robot in the selected direction and speed. As this drive block sets the motors the program can continue straight to the next block. This is different to the drive blocks that have a distance, time or angle to complete before the program moves to the next block.



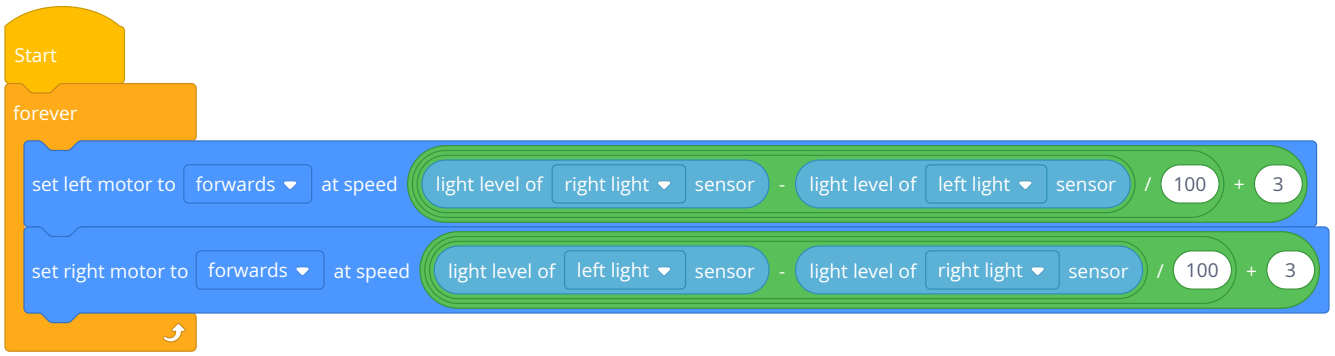
The block has two parameters:

Movement - There are two options that can be selected: forwards and backwards.

Speed - The speed can be either selected from 1 to 10 (as a constant value), a variable, a sensing or an operator block.

Example uses:

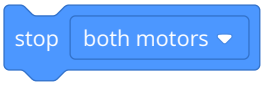
Example program 1: Light following



In this example, the program enters a forever loop. The first block in the loop sets the left motor forwards at a calculated speed. The calculated speed is the light level from the right sensor minus the light level from the left sensor divided by 100, 3 is also added to the result to set a minimum speed. The next block sets the right motor to forwards at a calculated speed. The calculated speed is the light level from the left sensor minus the light level from the right sensor divided by 100, 3 is also added to the result to set a minimum speed. The program then loops back to the start of the forever loop. The calculated results control the speed so that the side that has the least amount of light drives the fastest. This results in the Edison robot always trying to face and drive towards the brightest light.

The Stop block

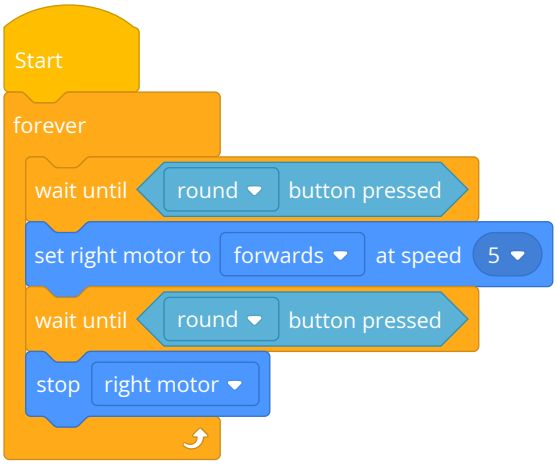
The stop block stops the motors from driving the wheels.



The stop block has only one parameter for selecting which motor/s are stopped. The options are both motors, left motor and right motor.

Example uses:

Example program 1: Motor on and off with a button press



In this example, the program enters a forever loop. The first block in the loop waits until the round button is pressed. When this happens, the right motor is set to forwards at speed 5, the program then progresses to another wait until round button pressed block. When this happens, the right motor is stopped and the program loops back to the start of the forever loop.

LEDs

The Set left LED block

The Set right LED block

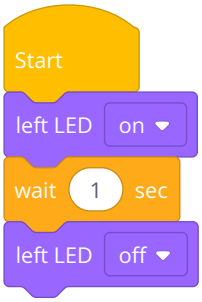
The Send IR message block

The Set left LED block

The set left LED block controls the left LED. There are two options on or off.

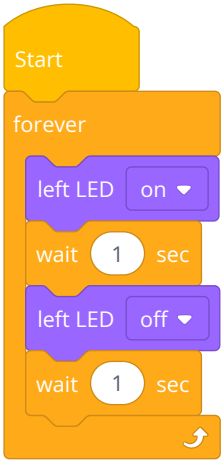
Example uses:

Example program 1: Blink left LED once



In this example the program begins by setting the left LED on, then waits for one second, then sets the left LED off.

Example program 2: Blink left LED forever



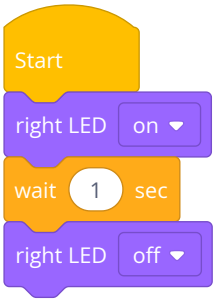
In this example, the program begins by entering a forever loop. Inside the loop the left LED is set to on, then the program waits for one second, then the left LED is set to off, then the program waits for one second, then the program repeats the loop from the beginning.

The Set right LED block

The set right LED block controls the right LED. There are two options on or off.

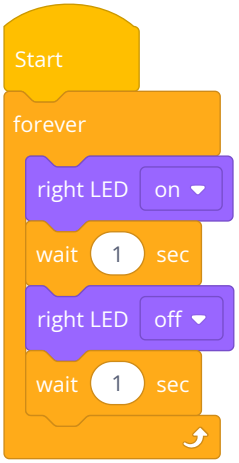
Example uses:

Example program 1: Blink right LED once



In this example, the program begins by setting the right LED on, then waits for one second, then sets the right LED off.

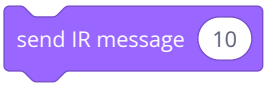
Example program 2: Blink right LED forever



In this example, the program begins by entering a forever loop. Inside the loop the right LED is set to on, then the program waits for one second, then the right LED is set to off, then the program waits for one second, then the program repeats the loop from the beginning.

The Send IR message block

The send IR message block sends data via infrared (IR) light. The data that can be sent is a value from 0 to 255. The value can be a constant (typed into the input and does not vary), a variable, a sensing or an operator block.



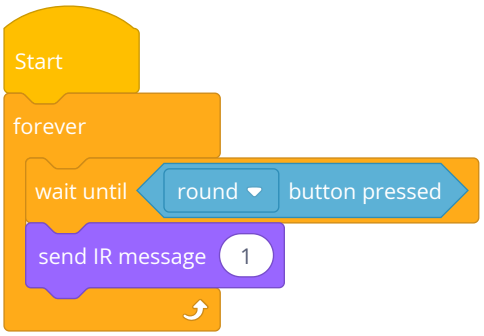
The sent data can be received by any other Edison robot that is in range of the IR light signal. The maximum range can be up to 10 meter/30 feet under ideal conditions (i.e. indoors, no direct sunlight, white ceiling and walls for the IR light to bounce off).

Note:

If you use a variable, the number in the variable could exceed the 255 limit (8-bits). When this happens, the number sent via IR will 'wrap around' (i.e. 256 sends 0, 257 sends 1, 258 sends 2...).

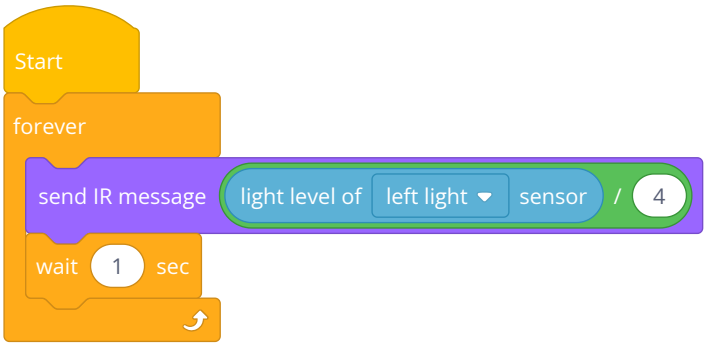
Example uses:

Example program 1: Send IR message 1 when round button pressed



In this example, the program begins by entering a forever loop. Inside the loop the program waits for the round button to be pressed. When this happens, the IR message 1 is sent. The program then loops back to the beginning of the forever loop.

Example program 2: Send light level every one second



In this example, the program begins by entering a forever loop. Inside the loop the program sends the light level reading from the left light sensor divided by four. The value is divided by four as the output from the light sensor is 1 to 1,000, but the highest number that can be sent with IR is 255 (1,000/4 = 250). This additional calculation allows the full range of the light sensor to be sent. The program then waits for one second and then loops back to the start of the forever loop.

Sound

The Beep block

The Play a note block

The Set music tempo to block

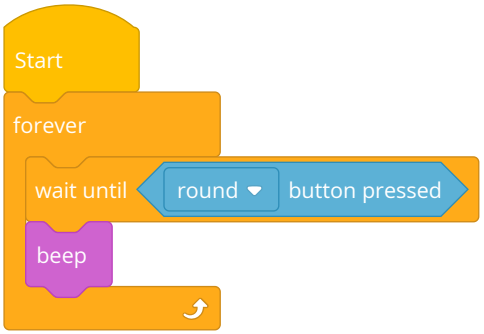
The Play music in background block

The Beep block

The beep block plays a 3.5kHz beep sound for 125mS (0.125 seconds).

Example uses:

Example program 1: Beep when round button is pressed



In this example, the program begins by entering a forever loop. Inside the loop the program waits for the round button to be pressed. When this happens, the beep is played. The program then loops back to the start of the forever loop and repeats.

The Play a note block

The play a note block plays a musical note. The block parameters allow the length, note and type of note to be selected.

The block has three parameters:

Length - There are four options: whole, half, quarter and eighth.

Note - There are ten options: C, B, A, G, F, E, D, middle C, low B and rest.

In English, notes are named with letters: 'A B C D E F G

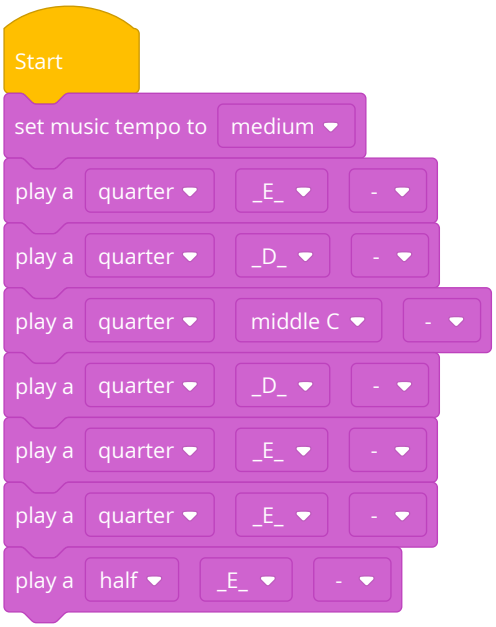
In most Romance languages (like Spanish), notes use: 'Do Re Mi Fa Sol La Si

Note modifier - There are three options: - (not modified), sharp and flat.

Note:

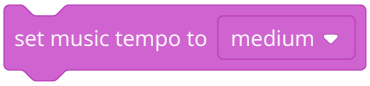
The play a note block must complete before the next block can start. For example, using the play a note block does not execute other blocks simultaneously. To do this see the play music in background block.

Example uses:

Example program 1: Mary had a little lamb

In this example, the program begins by setting the tempo for the music notes, then plays the notes for the nursery rhyme Mary had a little lamb in order, playing each note fully before moving on.

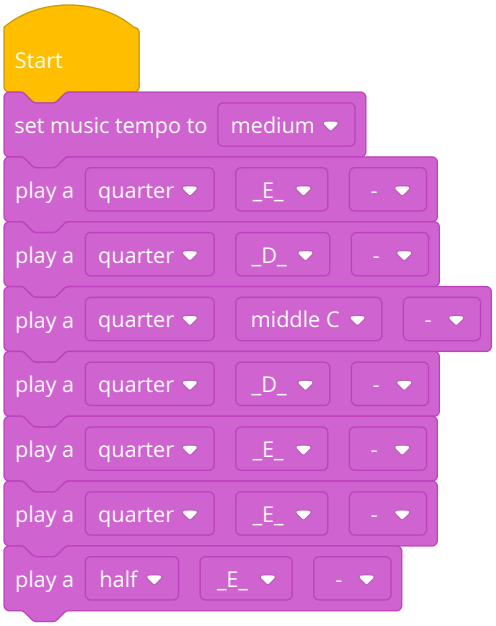
The Set music tempo to block

The set music tempo to block sets the tempo that following notes are played at.



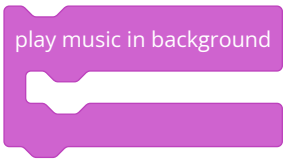
There are five options: very slow, slow, medium, fast and very fast.

Example uses:

Example program 1: Mary had a little lamb

In this example, the program begins by setting the tempo for the music notes, then plays the notes for the nursery rhyme Mary had a little lamb at that speed, playing each note fully before moving on.

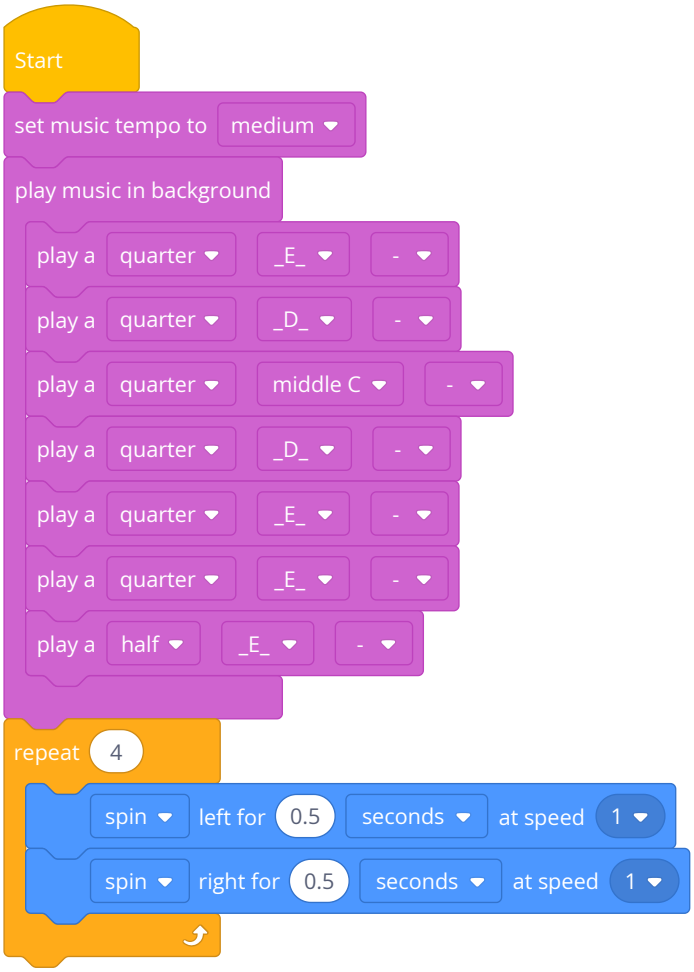
The Play music in background block

The play music in background block allows musical notes placed inside the block to play while the Edison robot executes other parts of the program.



Example uses:

Example program 1: Mary had a little lamb while dancing



In this example program, the program begins by setting the tempo for the music notes, then the notes inside the play music in background block start to play, the program enters the repeat loop and the spin left and spin right blocks run four times with the tune still playing in the background. This results in Edison dancing while playing music.

Variables

The Variable block

The Set to block

The Increment block

The Decrement block

The Bit shift right block

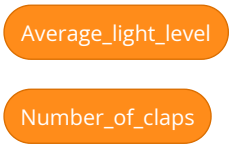
The Bit shift left block

The Variable block

A variable block is a place where data can be stored and used by a program. The data in a variable can vary and change while the program executes. This is why it is called a variable.

The data in a variable is a number that can range from -1,073,741,824 to 1,073,741,823 (signed 32-bit).

Variables should be customised by naming them something that makes sense to the program. For example:

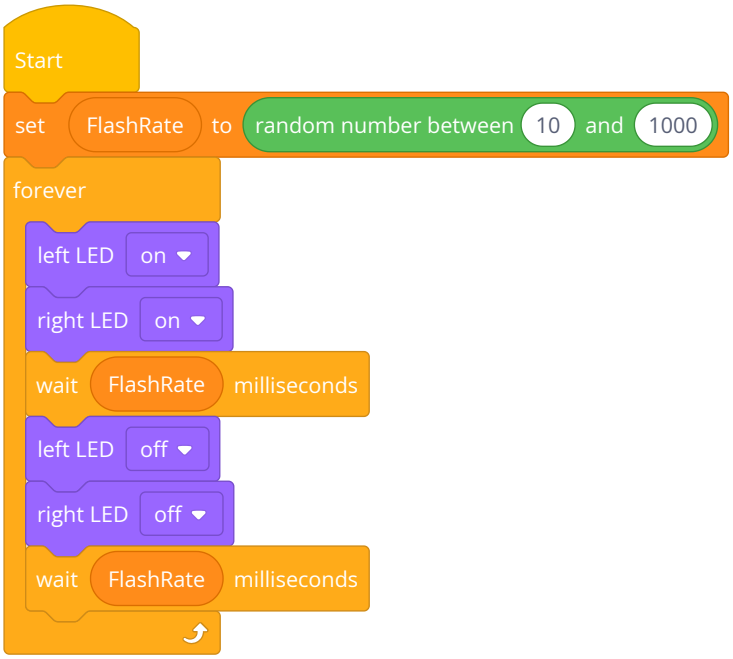


Note:

If the number in a variable exceeds the range of -1,073,741,824 to 1,073,741,823, then the program will halt and Edison will play a failure tone.

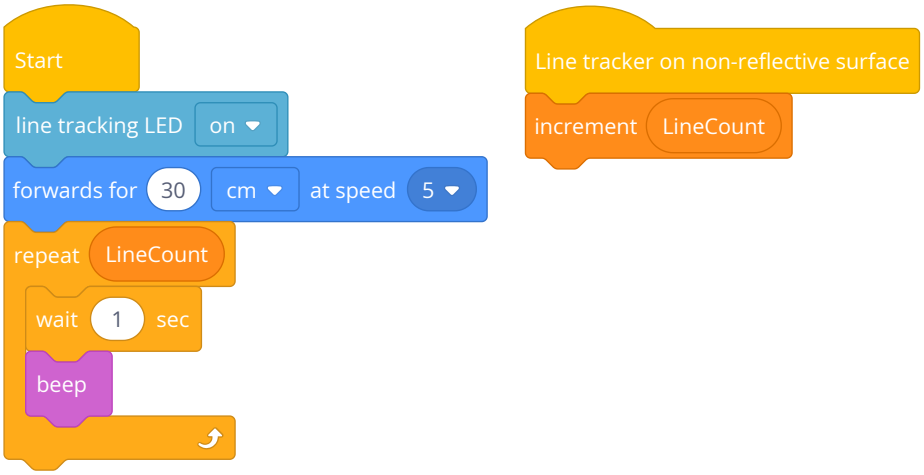
Example uses:

Example program 1: A random flash rate set at the program start



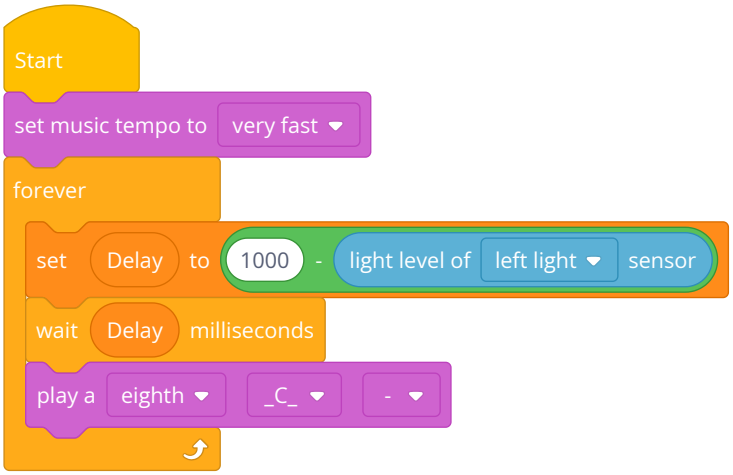
In this example program, the variable 'FlashRate' is set to a random number between 10 and 100. The program then enters a forever loop. Inside the loop the left and right LEDs are set to on, then the program waits for the millisecond value in 'FlashRate', then the left and right LEDs are set to off, then the program again waits for the millisecond value in 'FlashRate'. The program then loops back to the start of the forever loop and repeats. This results in Edison flashing its LEDs at different speeds each time the program executes

Example program 2: Count black lines driven over



In this example program, the line tracker LED is first set to on, then the Edison robot drives forwards for 30cm at speed 5. While driving forward any detected non-reflective (black) surfaces by the line tracker will trigger the program to go to the event block Line tracker on non-reflective surface. When this happens, the variable 'LineCount' is incremented (+1). When driving for 30 cm has completed the program repeats the wait for 1 second and beep blocks for the number of times in the variable 'LineCount'.

Example program 3: Light radiation counter



In this example program, the music tempo is first set to very fast. The program then enters a forever loop. Inside the loop the program sets the variable 'Delay' to 1000 minus the left light level (inverting the light level reading). The next block waits for the value in the 'Delay' variable. The program then plays the note C and loops back to the start of the forever loop and repeats. The result of this program is that Edison will beep slow when in darkness and faster when the light level is brighter.

The Set to block

The set to block is used to set a value in a variable.

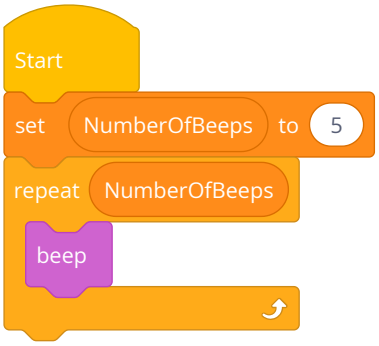
There are two inputs:

Variable - This is where the variable block is placed.

Value - Can be a constant (typed into the input and does not vary), a variable, a sensing or an operator block.

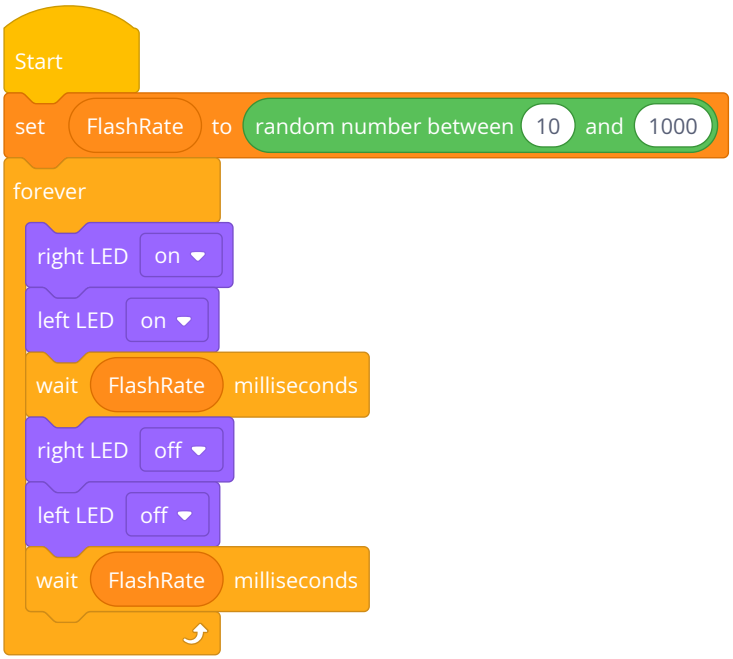
Example uses:

Example program 1: Set the number of beeps



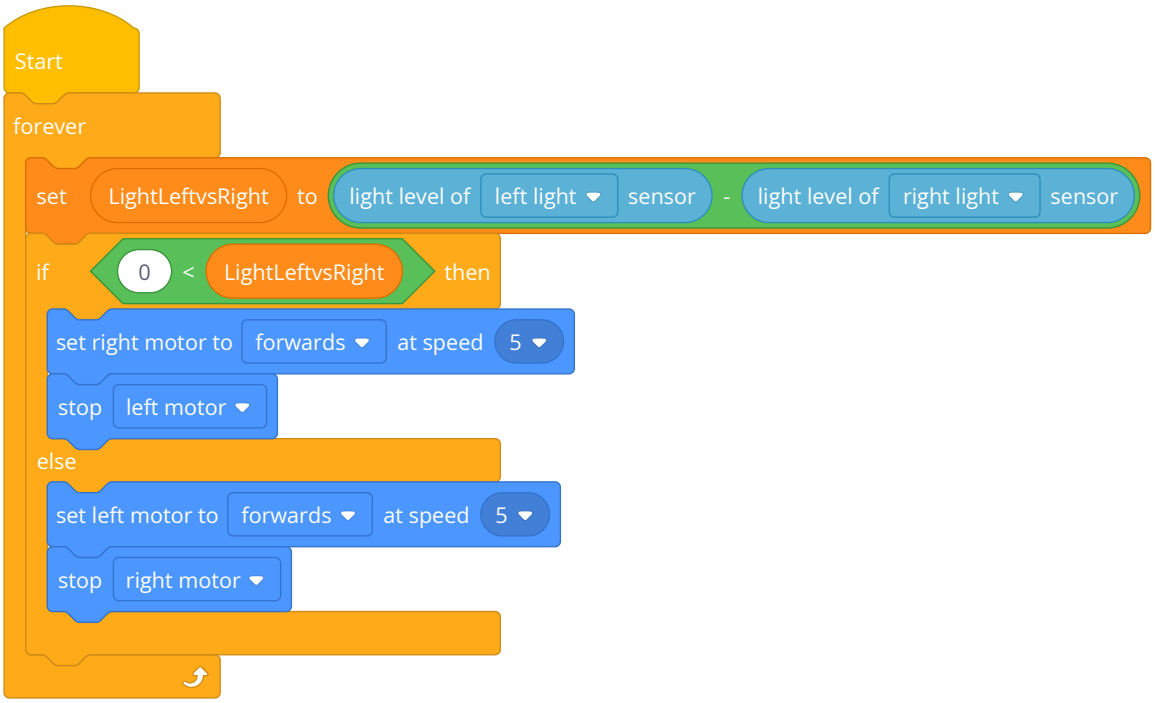
In this example program, the variable 'NumberOfBeeps' is set to 5, then the repeat block repeats the beeping for the value 5 held in the variable 'NumberOfBeeps'.

Example program 2: A random flash rate set at the program start



In this example program, the variable 'FlashRate' is set to a random number between 10 and 1000. The program then enters a forever loop. Inside the loop the left and right LEDs are set to on, then the program waits for the millisecond value in 'FlashRate', then the left and right LEDs are set to off, then the program again waits for the millisecond value in 'FlashRate'. The program then loops back to the start of the forever loop and repeats. This results in Edison flashing its LEDs at different speeds each time the program executes

Example program 3: Follow the light



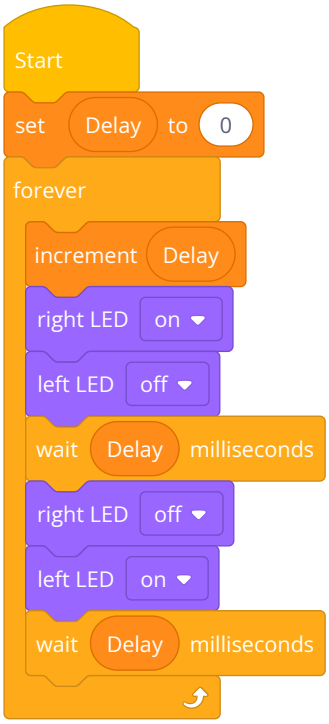
In this example, the program begins by entering a forever loop. Inside the loop the program sets the variable 'LightLeftvsRight' to the left light level minus the right light level. The if else block then tests whether the value in the variable 'LightLeftvsRight' is greater than 0. If the result is true, then the right motor is set to drive forwards and the left motor is stopped. If the result is false, then the left motor is set to drive forward and the right motor is stopped. The program then loops back to the start of the forever loop and repeats. This results in Edison driving towards the brightest light source.

The Increment block

The increment block increments (adds 1) to the variable placed in the block.

Example uses:

Example program 1: Decelerating flashing



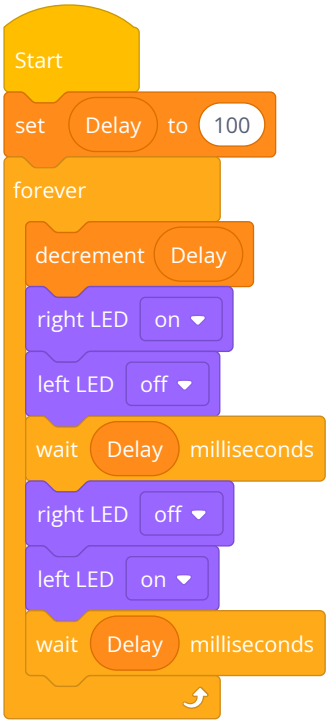
In this example program, the variable 'Delay' is set to 0 the program then enters a forever loop. Inside the loop the variable 'Delay' is incremented (+1), then the right LED is set to on and the left LED is set to off, then the program waits for the value in the variable 'Delay' in milliseconds, then the right LED is set to off and the left LED is set to on and then again, the program waits for the value in the variable 'Delay' in milliseconds. The program then loops back to the start of the forever loop and repeats. As the program repeats through the forever loop, the variable 'Delay' is incremented each time increasing the duration that the LEDs are on and off. After 1,000 loops the time the LEDs are on is one second and the time the LEDs are off is one second.

The Decrement block

The decrement block decrements (minus 1) to the variable placed in the block.

Example uses:

Example program 1: Accelerating flashing



In this example program, the variable 'Delay' is set to 100 the program then enters a forever loop. Inside the loop the variable 'Delay' is decremented (-1), then the right LED is set to on and the left LED is set to off, then the program waits for the value in the variable 'Delay' in milliseconds, then the right LED is set to off and the left LED is set to on and then again, the program waits for the value in the variable 'Delay' in milliseconds. The program then loops back to the start of the forever loop and repeats. As the program repeats through the forever loop, the variable 'Delay' is decremented each time reducing the duration that the LEDs are on and off. After 100 loops the time the LEDs are on is one millisecond and the time the LEDs are off is one millisecond.

The Bit shift right block

The bit shift right by block mathematically adjusts the connected variable block in its binary form by shifting the bits right by the set value.

The number format that computers use is called binary. It is a number system that is only comprised of the numbers 0 and 1. Each number inside a computer is a string of 1s and 0s. Here are some examples of binary numbers that are 8-bits long:

Human/decimal	Computer/8-bit Binary
0	00000000
1	00000001
12	00001100
255	11111111

Here are some examples of binary numbers that are 32-bits long:

Human/decimal	Computer/8-bit Binary
0	00000000000000000000000000000000
1	00000000000000000000000000000001
1000	00000000000000000000001111101000
1,073,741,823	00111111111111111111111111111111

Edison V3 uses both 8-bit and 32-bit binary numbers. For example, variable blocks are able to store large 32-bit numbers (actually 31-bit signed), but only 8-bit numbers can be sent via the send IR message block. Further, some sensors, such as the light sensors have a range of 1 to 1000. As the highest possible 8-bit number is 255, then the full range of the light sensor reading cannot be sent via the send IR message block. This is where the bit shift right block can help!

The light sensor reading is 10-bit binary. Here are the minimum and maximum values:

Human/decimal	Computer/8-bit Binary
1	0000000001
1,000	1111101000

If the number 1000 was sent with the send IR message block the received number would only be the last 8-bits. This would be 11111010 (binary) or 232 (human/decimal). This is a very different number to the actual light reading of 1000.

To send the light level data via the send IR message block, the bit shift right block can be used to shorten the number into an 8-bit number (shift right by 2) without losing too much of the data. Then, on the receiving end the bit shift left block can be used to restore the light level reading back to 10-bits (shift left by 2).

Step in process	Human/decimal	Computer/8-bit Binary
Light reading	951	1110110111 (10-bit)

Bit shift right (2)	237	0011101101 (10-bit)
IR data send/receive	237	11101101 (8-bit)
Bit shift left (2)	948	1110110100 (10-bit)

In this scenario the resulting number is not perfectly the same, as some information has been lost. However, the most significant bits containing more than 99% of the original data has been retained through this process.

The Bit shift left block

The bit shift left by block mathematically adjusts the connected variable block in its binary form by shifting the bits left by the set value.

The number format that computers use is called binary. It is a number system that is only comprised of the numbers 0 and 1. Each number inside a computer is a string of 1s and 0s. Here are some examples of binary numbers that are 8-bits long:

Human/decimal	Computer/8-bit Binary
0	00000000
1	00000001
12	00001100
255	11111111

Here are some examples of binary numbers that are 32-bits long:

Human/decimal	Computer/8-bit Binary
0	00000000000000000000000000000000
1	00000000000000000000000000000001
1000	00000000000000000000001111101000
1,073,741,823	00111111111111111111111111111111

Edison V3 uses both 8-bit and 32-bit binary numbers. For example, variable blocks are able to store large 32-bit numbers (actually 31-bit signed), but only 8-bit numbers can be sent via the send IR message block. Further, some sensors, such as the light sensors have a range of 1 to 1000. As the highest possible 8-bit number is 255, then the full range of the light sensor reading cannot be sent via the send IR message block. This is where the bit shift right block can help!

The light sensor reading is 10-bit binary. Here are the minimum and maximum values:

Human/decimal	Computer/8-bit Binary
1	0000000001
1,000	1111101000

If the number 1000 was sent with the send IR message block the received number would only be the last 8-bits. This would be 11111010 (binary) or 232 (human/decimal). This is a very different number to the actual light reading of 1000.

To send the light level data via the send IR message block, the bit shift right block can be used to shorten the number into an 8-bit number (shift right by 2) without losing too much of the data. Then, on the receiving end the bit shift left block can be used to restore the light level reading back to 10-bits (shift left by 2).

Step in process	Human/decimal	Computer/8-bit Binary
Light reading	951	1110110111 (10-bit)

Bit shift right (2)	237	0011101101 (10-bit)
IR data send/receive	237	11101101 (8-bit)
Bit shift left (2)	948	1110110100 (10-bit)

In this scenario the resulting number is not perfectly the same, as some information has been lost. However, the most significant bits containing more than 99% of the original data has been retained through this process.

Events

The Any obstacle detected event block

The Obstacle detected left event block

The Obstacle detected right event block

The Obstacle detected ahead event block

The Clap detected event block

The Triangle button pressed event block

The Round button pressed event block

The Line tracker on reflective surface event block

The Line tracker on non-reflective surface event block

The Line tracker changes surface event block

The IR message received event block

The Drive strained event block

The Remote code received event block

The Any obstacle detected event block

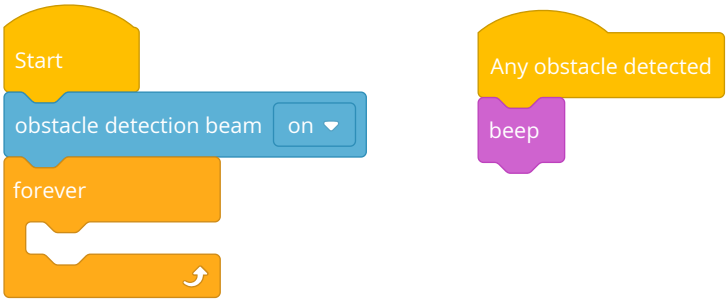
The any obstacle detected event block causes the program to go to this block when an obstacle is detected ahead by the obstacle detector

Note:

Always ensure that the obstacle detector beam is set to on in the main program.

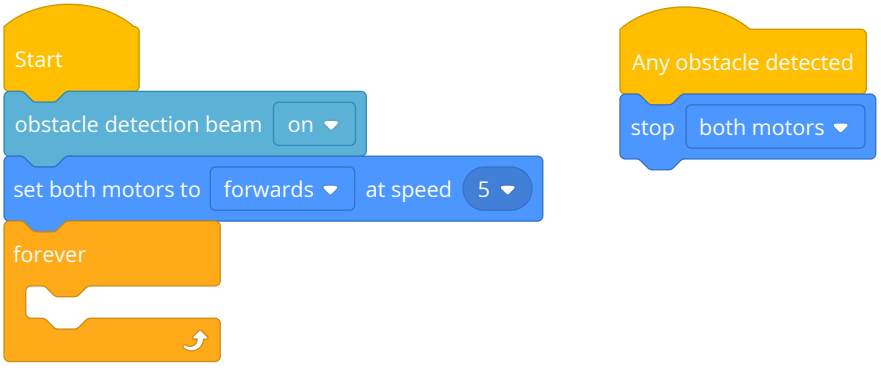
Example uses:

Example program 1: Beep when any obstacle is detected.



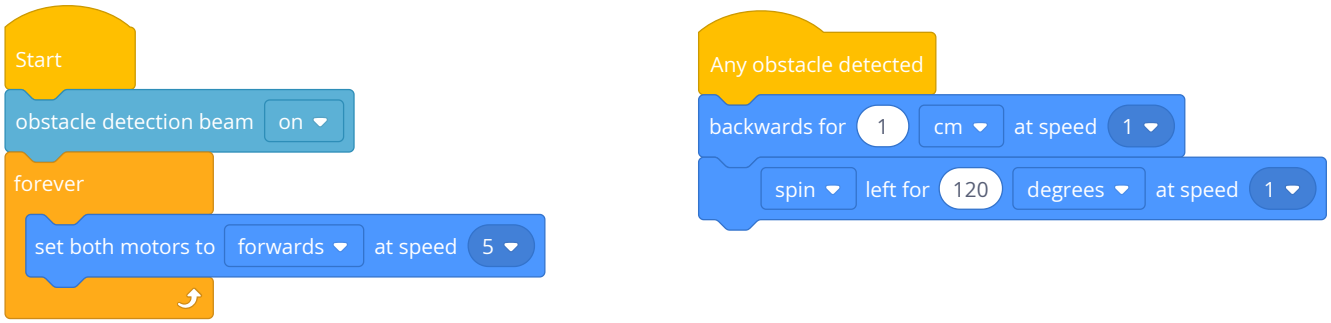
In this example program, the obstacle detector is first set to on, the program then enters the forever loop block. When any obstacle is detected, the program exits the forever loop and goes to the Any obstacle detected event block. This causes the beep block to execute and the Edison robot beeps. The program then returns to where it left from in the main program.

Example program 2: Stop driving when there is an obstacle



In this example program, the obstacle detector is first set to on, then both motors are set to drive forwards at speed 5, the program then enters the forever loop block. When any obstacle is detected, the program exits the forever loop and goes to the Any obstacle detected event block. This causes the stop both motors block to execute and stops both motors from running. The program then returns to where it left from in the main program.

Example program 3: Avoid obstacles



In this example program, the obstacle detector is first set to on, then the program enters a forever loop, inside the loop both motors are set to drive forwards at speed 5. When any obstacle is detected, the program exits the forever loop and goes to the Any obstacle detected event block. The blocks under the Any obstacle detected event block drive the Edison backwards for 1 cm at speed 1, then spin left for 120 degrees at speed 1. The program then returns to where it left in the forever loop and both motors are set to drive forward at speed 5 again. This results in Edison driving away from obstacles in its path.

The Obstacle detected left event block

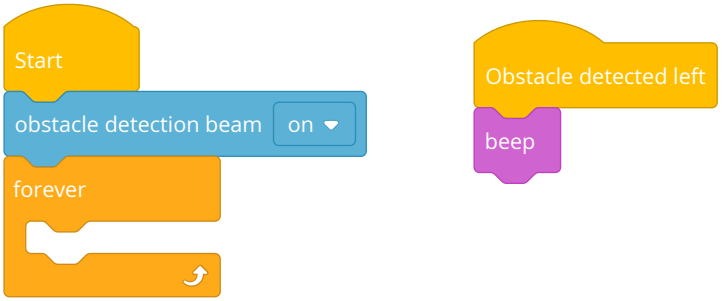
The obstacle detected left event block causes the program to go to this block when an obstacle is detected on the left by the obstacle detector.

Note:

Always ensure that the obstacle detector beam is set to on in the main program.

Example uses:

Example program 1: Beep when an obstacle is detected on the left



In this example program, the obstacle detector is first set to on, the program then enters the forever loop block. When an obstacle is detected to the left, the program exits the forever loop and goes to the Obstacle detected left event block. This causes the beep block to execute and the Edison robot beeps. The program then returns to where it left in the forever loop block.

The Obstacle detected right event block

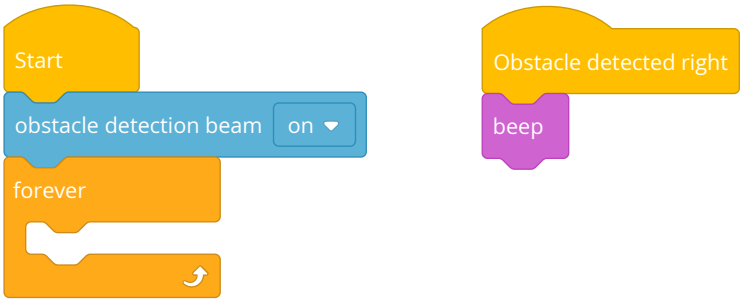
The obstacle detected right event block causes the program to go to this block when an obstacle is detected on the right by the obstacle detector.

Note:

Always ensure that the obstacle detector beam is set to on in the main program.

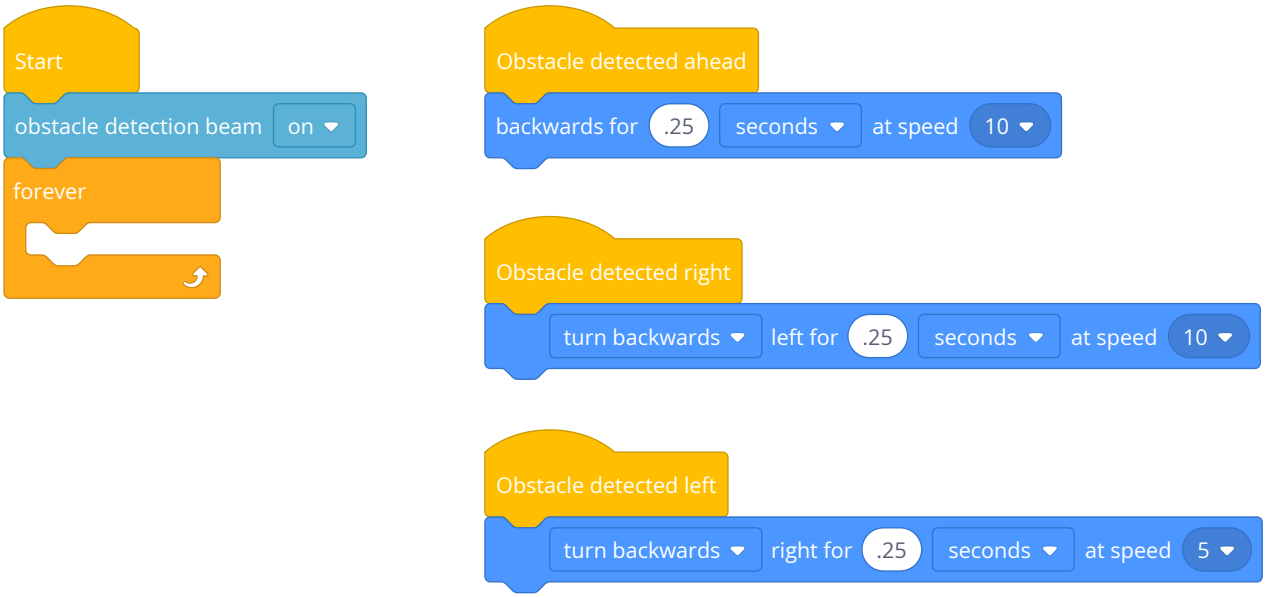
Example uses:

Example program 1: Beep when an obstacle is detected on the right



In this example program, the obstacle detector is first set to on, the program then enters the forever loop block. When an obstacle is detected to the right, the program exits the forever loop and goes to the Obstacle detected right event block. This causes the beep block to run and the Edison robot beeps. The program then returns to where it left in the forever loop block.

Example program 2: Obstacle evader



In this example program, the obstacle detector is first set to on, the program then enters the forever loop block. When...

...an obstacle is detected ahead the program goes to the Obstacle detected ahead event block and the Edison robot drives backwards for 0.25 seconds at speed 10 and then returns to where it left in the forever loop block.

...an obstacle is detected to the right the program goes to the Obstacle detected right event block and the Edison robot turns left backwards for 0.25 seconds at speed 10 and then returns to where it left in the forever loop block.

...an obstacle is detected to the left the program goes to the Obstacle detected left event block and the Edison robot turns right backwards for 0.25 seconds at speed 10 and then returns to where it left in the forever loop block.

The Obstacle detected ahead event block

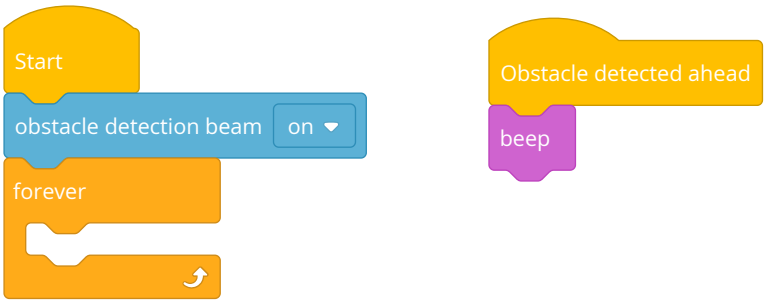
The obstacle detected ahead event block causes the program to go to this block when an obstacle is detected ahead by the obstacle detector. This means that, at the same time, both the left and right obstacle detectors have detected an obstacle.

Note:

Always ensure that the obstacle detector beam is set to on in the main program.

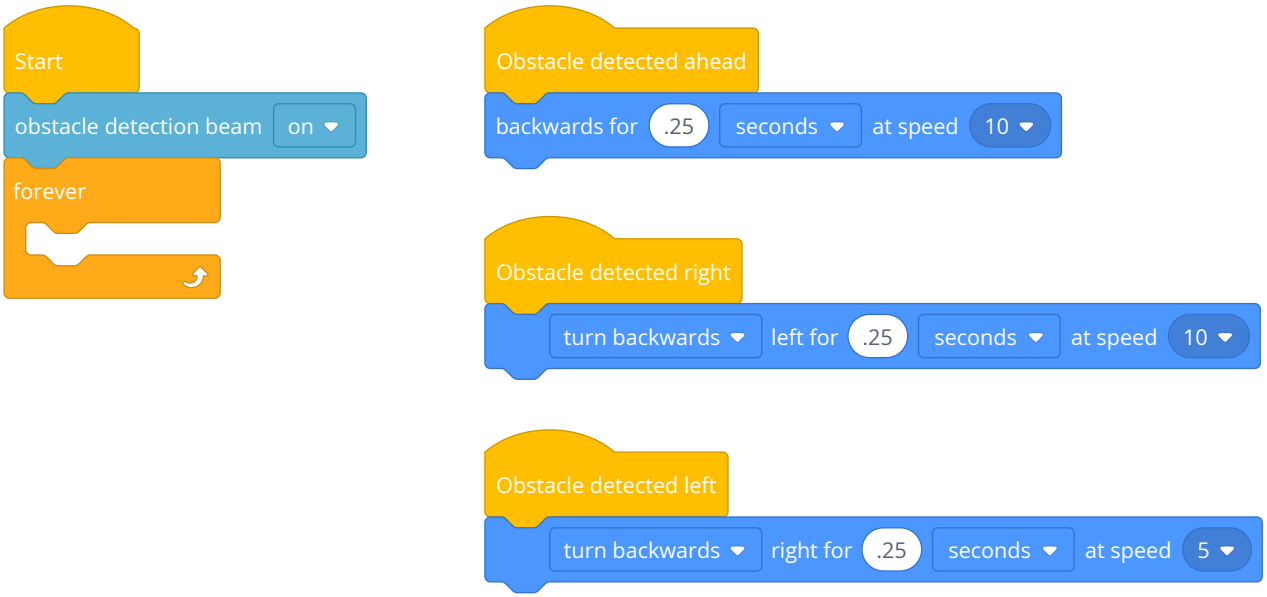
Example uses:

Example program 1: Beep when an obstacle is detected ahead



In this example program, the obstacle detector is first set to on, the program then enters the forever loop block. When an obstacle is detected ahead, the program exits the forever loop and goes to the Obstacle detected ahead event block. This causes the beep block to execute and the Edison robot beeps. The program then returns to where it left in the forever loop block.

Example program 2: The obstacle evader



In this example program, the obstacle detector is first set to on, the program then enters the forever loop block. When...

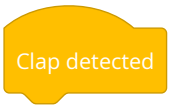
...an obstacle is detected ahead the program goes to the Obstacle detected ahead event block and the Edison robot drives backwards for 0.25 seconds at speed 10 and then returns to where it left in the forever loop block.

...an obstacle is detected to the right the program goes to the Obstacle detected right event block and the Edison robot turns left backwards for 0.25 seconds at speed 10 and then returns to where it left in the forever loop block.

...an obstacle is detected to the left the program goes to the Obstacle detected left event block and the Edison robot turns right backwards for 0.25 seconds at speed 10 and then returns to where it left in the forever loop block.

The Clap detected event block

The clap detected event block causes the program to go to this block when a clap is detected.

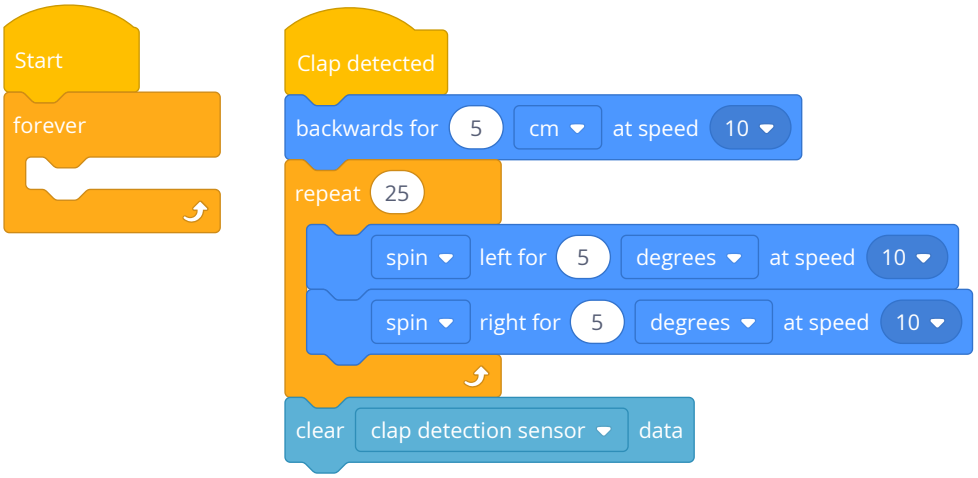


Note:

When Edison is driving the motor noise and bumping cause the clap sensor to trigger, so it's best not to use the clap sensor when driving. Also, if there are likely to be false or unwanted noise detections, then the clap sensor data can be cleared with the clear sensor data block set to clap detector (see example program).

Example uses:

Example program: Edison is scared of loud noises



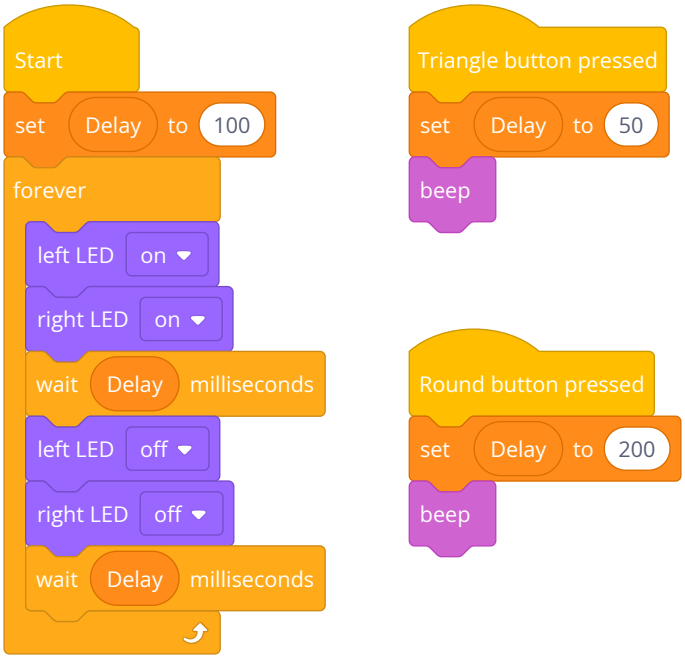
In this example, the program begins by entering an empty forever loop and nothing happens. When a clap (or other loud scary sound) is detected, the program exits the forever loop and goes to the Clap detected event block. The blocks below the Clap detected event block then execute, driving backwards 5 cm at speed 10 (jump in fright), then entering a repeat loop set to 25 times where the spin left and spin right blocks repeatedly execute causing the Edison to shake. Finally, (when Edison recovers from its fright) the clear sensor data block clears out any false clap detections that may have occurred while driving. The program now returns to the forever loop.

The Triangle button pressed event block

The triangle button pressed event block causes the program to go to this block when the triangle button is pressed.

Example uses:

Example program: Control LED flash speed setting with buttons



In this example, the program begins by setting the variable called 'Delay' to 100, the program then enters a forever loop. Inside the forever loop, the left and right LEDs are set to on and off with a wait for the value in the 'Delay' variable in milliseconds. This results in the LEDs being on for 100 milliseconds (0.1 seconds) and off for 100 milliseconds (0.1 seconds).

When the triangle button is pressed, the program goes to the Triangle button pressed event block. The blocks below the Triangle button pressed event block then executes, setting the variable called 'Delay' to 50, then playing a beep. The program then returns to the forever loop, but now the value in the variable called 'Delay' is 50, so the time between the LEDs being on and off is now 50 milliseconds (0.05 seconds). This makes them flash on and off faster.

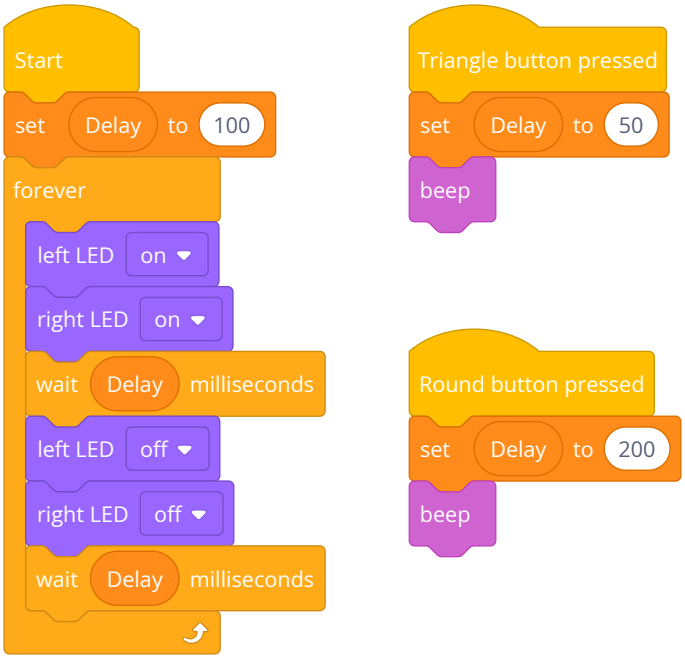
When the round button is pressed, the program goes to the Round button pressed event block. The blocks below the Round button pressed event block then executes, setting the variable called 'Delay' to 200, then playing a beep. The program then returns to the forever loop, but now the value in the variable called 'Delay' is 200, so the time between the LEDs being on and off is now 200 milliseconds (0.2 seconds). This makes them flash on and off slower.

The Round button pressed event block

The round button pressed event block causes the program to go to this block when the round button is pressed.

Example uses:

Example program: Control LED flash speed setting with buttons



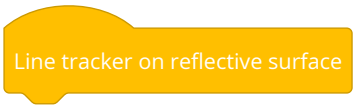
In this example, the program begins by setting the variable called 'Delay' to 100, the program then enters a forever loop. Inside the forever loop, the left and right LEDs are set to on and off with a wait for the value in the 'Delay' variable in milliseconds. This results in the LEDs being on for 100 milliseconds (0.1 seconds) and off for 100 milliseconds (0.1 seconds).

When the triangle button is pressed, the program goes to the Triangle button pressed event block. The blocks below the Triangle button pressed event block then executes, setting the variable called 'Delay' to 50, then playing a beep. The program then returns to the forever loop, but now the value in the variable called 'Delay' is 50, so the time between the LEDs being on and off is now 50 milliseconds (0.05 seconds). This makes them flash on and off faster.

When the round button is pressed, the program goes to the Round button pressed event block. The blocks below the Round button pressed event block then executes, setting the variable called 'Delay' to 200, then playing a beep. The program then returns to the forever loop, but now the value in the variable called 'Delay' is 200, so the time between the LEDs being on and off is now 200 milliseconds (0.2 seconds). This makes them flash on and off slower.

The Line tracker on reflective surface event block

The line tracker on reflective surface event block causes the program to go to this block when the line tracker sensor detects a change from a non-reflective surface (i.e. black) to a reflective surface (i.e. white).

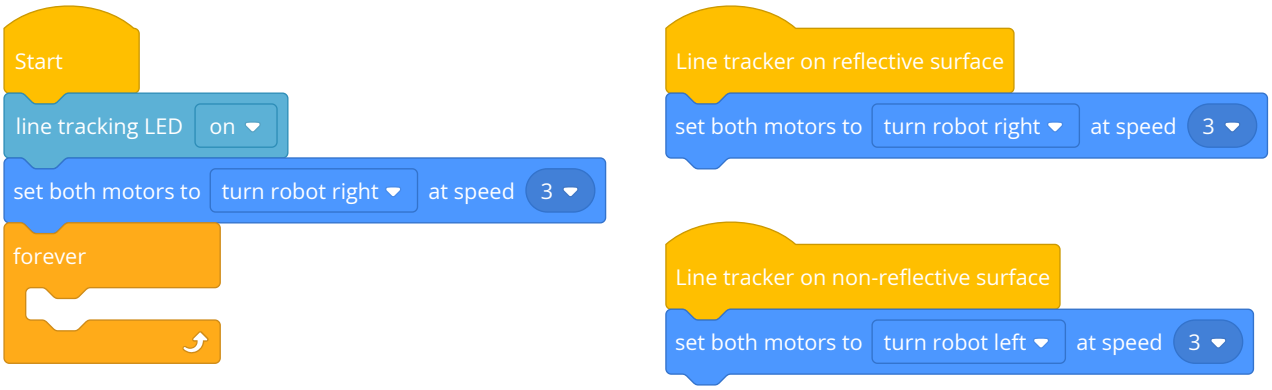


Note:

The line tracker LED must be set to on for the Line tracker on reflective surface event block to work.

Example uses:

Example program: Follow the edge of a line



In this example, the program begins by setting the line tracking LED to on, the program then causes the robot to turn right (looking for the edge of a line).

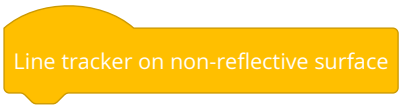
When the line tracker detects a change from a reflective (white) to non-reflective (black) surface, the program goes to the Line tracker on non-reflective surface block and executes the drive block to set the robot to turn left (turning back towards the reflective (white) surface).

When the line tracker detects a change from a non-reflective to reflective surface, the program goes to the Line tracker on reflective surface event block and executes the drive block to set the robot to turn right (turning back towards the non-reflective (black) surface).

Each time the robot turns either left or right it moves forward following the edge of the line.

The Line tracker on non-reflective surface event block

The line tracker on non-reflective surface event block causes the program to go to this block when the line tracker sensor detects a change from a reflective surface (i.e. white) to a non-reflective surface (i.e. black).

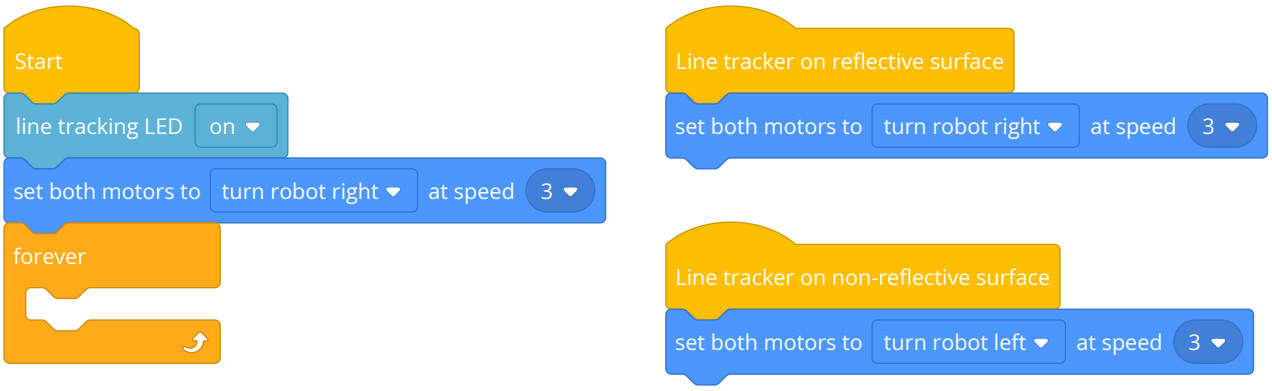


Note:

The line tracker LED must be set to on for the Line tracker on non-reflective surface event block to work.

Example uses:

Example program: Follow the edge of a line



In this example, the program begins by setting the line tracking LED to on, the program then causes the robot to turn right (looking for the edge of a line).

When the line tracker detects a change from a reflective (white) to non-reflective (black) surface, the program goes to the Line tracker on non-reflective surface block and executes the drive block to set the robot to turn left (turning back towards the reflective (white) surface).

When the line tracker detects a change from a non-reflective to reflective surface, the program goes to the Line tracker on reflective surface block and executes the drive block to set the robot to turn right (turning back towards the non-reflective (black) surface).

Each time the robot turns either left or right it moves forward following the edge of the line.

The Line tracker changes surface event block

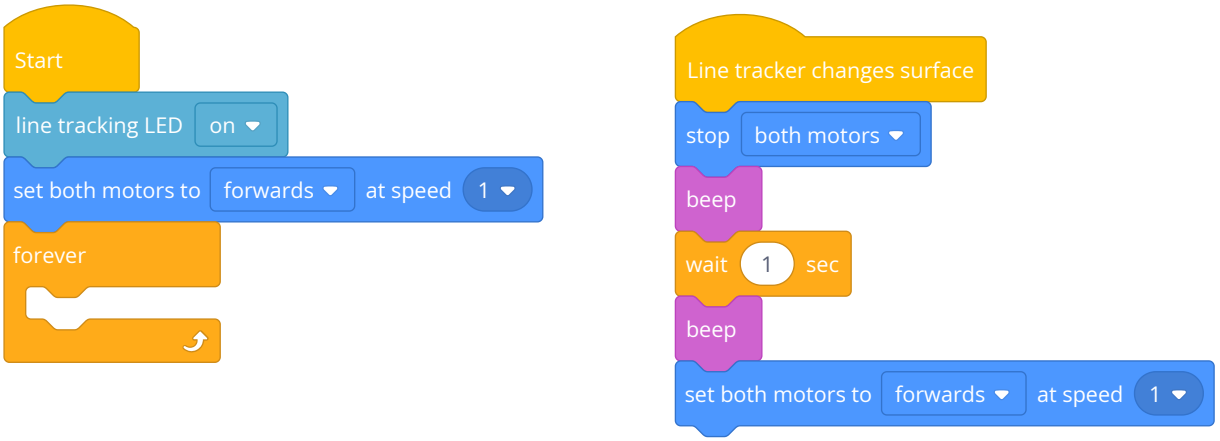
The line tracker changes surface event block causes the program to go to this block when the line tracker sensor detects either a change from a non-reflective (i.e. black) surface to reflective (i.e. white) surface or a from a reflective (i.e. white) surface to a non-reflective (i.e. black) surface.

Note:

The line tracker LED must be set to on for the Line tracker changes surface event block to work.

Example uses:

Example program: Stopping for change



In this example, the program begins by setting the line tracking LED to on, the robot then drives forward at speed 1 and enters a forever loop. The robot will then continue to drive forwards until the line tracker sensor detects a change in the reflectivity of the surface (white to black or black to white). When this happens, the program goes to the Line tracker changes surface block. The code below the Line tracker changes surface then executes, stopping both motors, beeping, waiting 1 second, beeping again, then setting both motors to again drive forward.

The IR message received event block

The IR message received event block causes the program to go to this block when an Infrared (IR) message from another Edison robot is received.

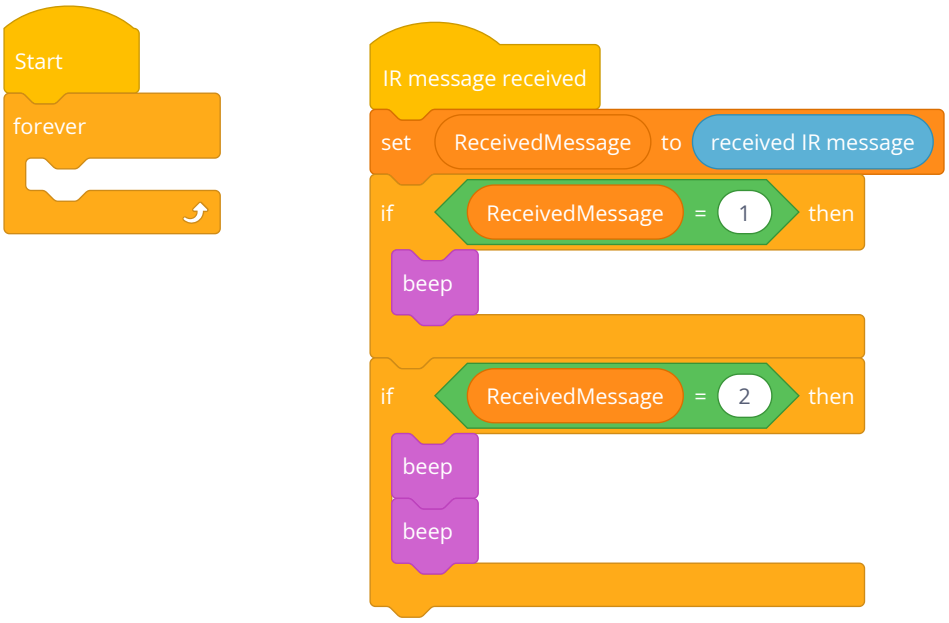
Note:

The IR message received event block will not work when the IR receiver is being used to detect obstacles.

The program will go to the IR message received event block when any IR message is received from another Edison robot. The IR message received event block does not decode the data in the IR message. Decoding the message requires that the message data be put into a variable and then decoded using an if then block.

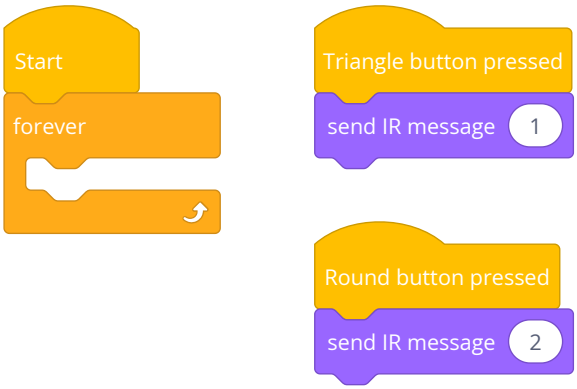
Example uses:

Example program: Receive IR message to beep once or twice



In this example, the program begins by entering an empty forever loop and nothing happens. When an IR message is received the program goes to the IR message received event block. The code below the IR message received event block then runs. First the received message data is stored in the variable 'ReceivedMessage'. This is required as each time the data from the received IR message is read it is also cleared. Now that the message data is in a variable the variable is tested if it is equal to 1, if true, then the beep block is played once. Next, the variable is tested if it is equal to 2, if true, then the beep block plays twice. The program now returns to the forever loop.

IR message sending program:



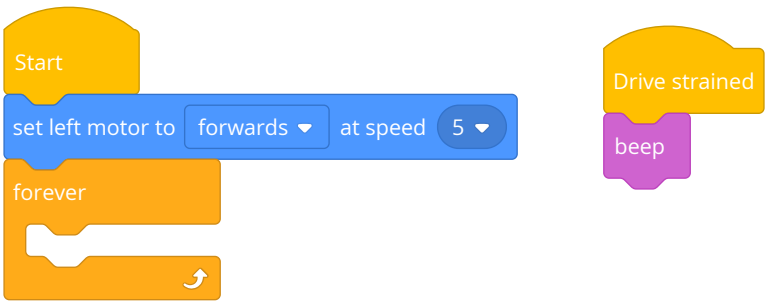
The above program is used in the Edison robot that is sending the IR message. The program first enters an empty forever loop. When the triangle button is pressed, the program goes to the Triangle button pressed event block and an IR message is sent with the number 1. When the round button is pressed, the program goes to the Round button pressed event block and an IR message is sent with the number 2.

The Drive strained event block

The drive strained event block causes the program to go to this block when one or both motors are being driven, but are unable to rotate (under strain).

Example uses:

Example program: Alarm when the left wheel cannot rotate



In this example, the program begins by setting the left motor to drive forwards, then enters a forever loop. When the wheel cannot rotate due to being under strain, the program goes to the Drive strained event block and executes the code below. The code below the Drive strained event block is the beep block which plays a beep. The program then returns to the forever loop.

The Remote code received event block

The remote code received event block causes the program to go to this block when a remote-control code is received.

Note:

Make sure you press the triangle button to start your program. This is easy to forget as the Edison robot will respond as a remote-control robot and not do what you have programmed it to.

To use the remote control blocks the Edison robot must first be taught which buttons of your remote control correspond to which remote code. Learn more about this here:

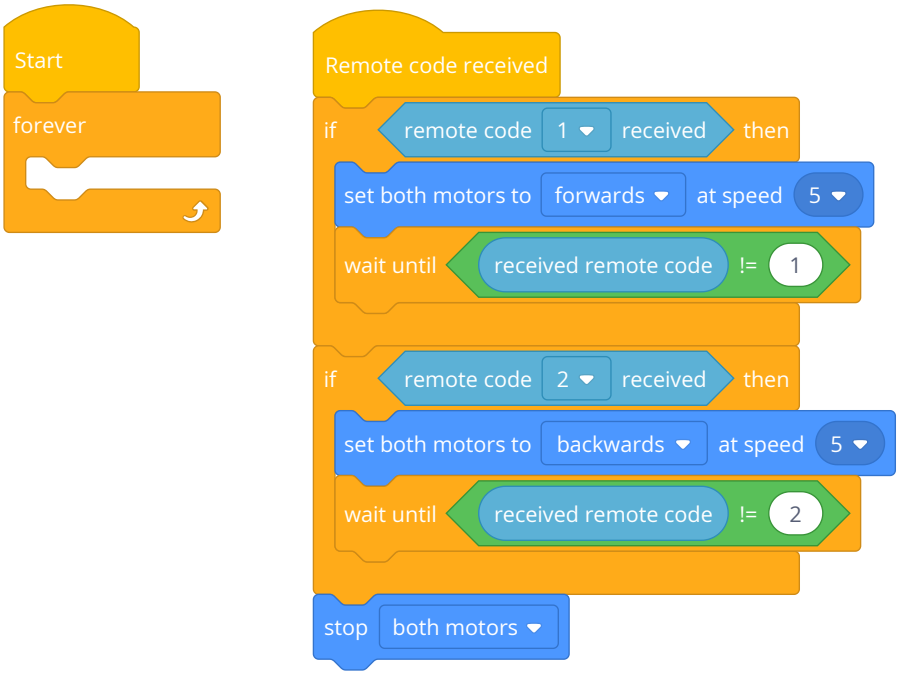
https://meetedison.com/content/Edison-robot-barcodes.pdf

If the program behaves unexpectantly, you may need to use the clear sensor data block to clear the contents of the remote code received block. See example program 1 below.

Receiving infrared remote codes will work with approximately 90% of standard TV remote controls, however each brand uses slightly different data formats. When decoding data, this can cause some unexpected behaviour from one remote control brand to another. Advanced remote-control decoding works best with the EdRemote (available now).

Example uses:

Example program: Remote control drive forwards or backwards



In this example, the program begins by entering an empty forever loop. When a remote-control code is received the program goes to the Remote code received event block. The code below the Remote code received event block executes, the remote code is tested to see if it is equal to 1, if true, then both motors are set to drive forwards, this continues until the code being received no longer equals 1. Next, the received code is tested to see if it is equal to 2, if true, then both motors are set to drive backwards. This continues until the code being received no longer equals 2. Finally, after the remote codes have been tested and are no longer being received the stop block stops both motors. The program returns to the empty forever loop.

Control

The Wait sec block

The Repeat block

The Forever block

The If then block

The If then else block

The Wait until block

The Repeat until block

The Wait milliseconds block

The Wait sec block

The wait sec block pauses the program for an amount of time in seconds. The amount of time can be from 0.001 seconds (1 millisecond) to 1,073,741 seconds (12.4 days).

The wait block has just one parameter:

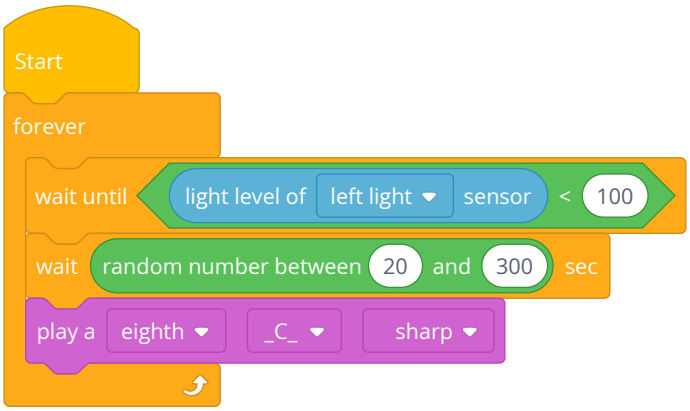
Value - Can be a constant (typed into the input and does not vary), a variable, a sensing or an operator block.

Example uses:

Example program 1: Blink the left LED

In this example, the program begins by entering a forever loop. Inside the loop the left LED is set to on, then the program pauses and waits for 0.5 seconds, then the left LED is set to off, then the program again pauses and waits for 0.5 seconds and then loops to the start.

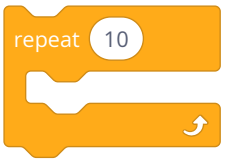
Example program 2: Nighttime robot cricket



In this example, the program begins by entering a forever loop. Inside the loop there is a wait until block that pauses the progression of the program until the level of light on the left light sensor is less than 100. When this happens, the program progresses to wait for a random number of seconds between 20 and 300. When this wait ends the speaker plays a note. The program then loops back to the beginning of the forever loop. This results in behaviour similar to a cricket at nighttime, so it can't be found.

The Repeat block

The repeat block is a type of loop that only loops a set number of times. All the blocks inside the repeat block execute only for this set number of times.



The repeat block has just one parameter:

Value - Can be a constant (typed into the input and does not vary), a variable, a sensing or an operator block.

Example uses:

Example program 1: Drive in a square

In this example, the program begins by entering into a repeat block that is set to repeat four times. Inside the repeat block the robot drives forward 10cm, then spins right 90 degrees. These two blocks inside the repeat block execute four times. The result is that the Edison robot drives in the shape of a square.

The Forever block

The forever block is a type of loop block that loops unconditionally forever (unless the program is stopped). The blocks placed inside the forever block execute repeatedly

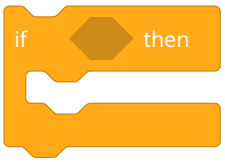
Example uses:

Example program: Blink a LED

In this example, the program begins by entering a forever loop. Inside the forever loop the left LED is set to on, then the wait block pauses the program for 0.25 seconds, then the left LED is set to off, then the wait block pauses the program for 0.25 seconds. The program then loops back to the set left LED on block and the sequence repeats forever.

The If then block

The if then block executes the blocks if the input condition is true. If the condition is not true, then the program continues to the next block in the program.



This block has one input parameter:

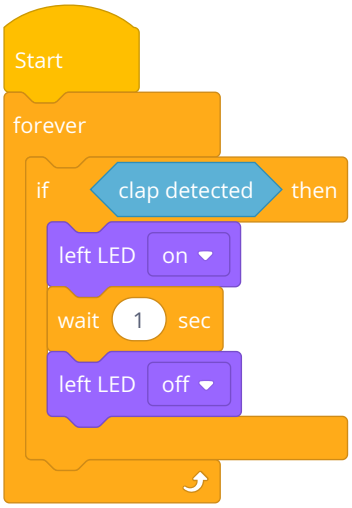
Conditional input - The conditional input accepts diamond shaped blocks that can be either an operator block or sensing block.

Note:

The if then block is usually used inside a forever loop, so that the if then condition can be tested multiple times. If a program has just an if then block, then the test will be executed once and the program will exit.

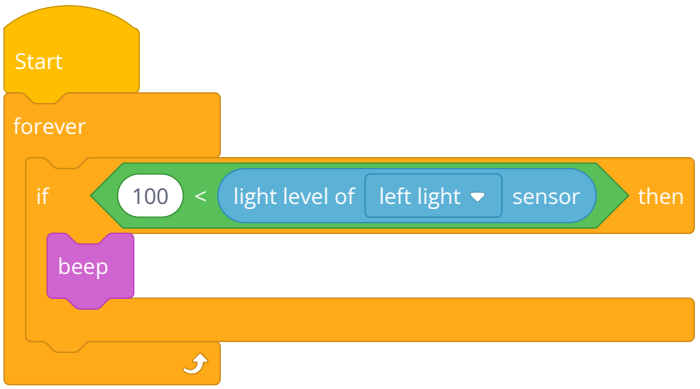
Example uses:

Example program 1: Blink LED if there is a clap



n this example, the program begins by entering a forever loop. The first block in the forever loop is the if then block that tests if there has been a clap. If false, then the contents of the if then block is skipped and the program loops back to start of the forever loop. If a clap has been detected (true), then the blocks inside the if then block execute. First the left LED is set to on, then the program pauses for one second, then the left LED is set to off. The program now loops back to the start of the forever loop. The result is that the Edison robot blinks its left LED for one second whenever it detects a clap.

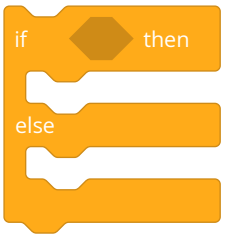
Example program 2: Draw open alarm



In this example, the program begins by entering a forever loop. The first block in the forever loop is the if then block that tests if the level of light on the left light sensor is greater than 100. If this is true (light level is greater than 100) then the beep block executes and the Edison robot beeps like an alarm. If it is not true (light level is less than 100), then the beep block is skipped and the program loops back to the start of the forever loop. The result is that the Edison beeps rapidly whenever it detects a light level above 100.

The If then else block

The if then else block executes the first set of blocks if the input condition is true. If the condition is false, then the second set of blocks execute.



This block has one input parameter:

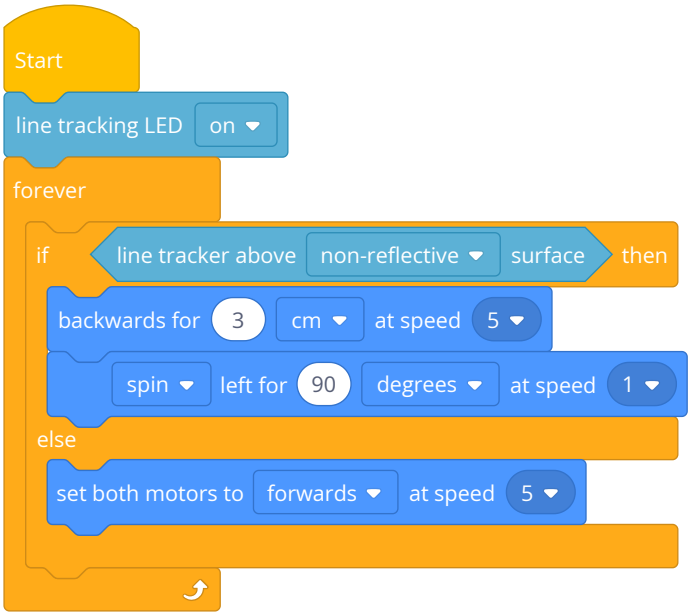
Conditional input - The conditional input accepts diamond shaped blocks that can be either an operator block or sensing block.

Note:

The if then else block is usually used inside a forever loop, so that the if then else condition can be tested multiple times. If a program has just an if then else block, then the test will be executed once and the program will exit.

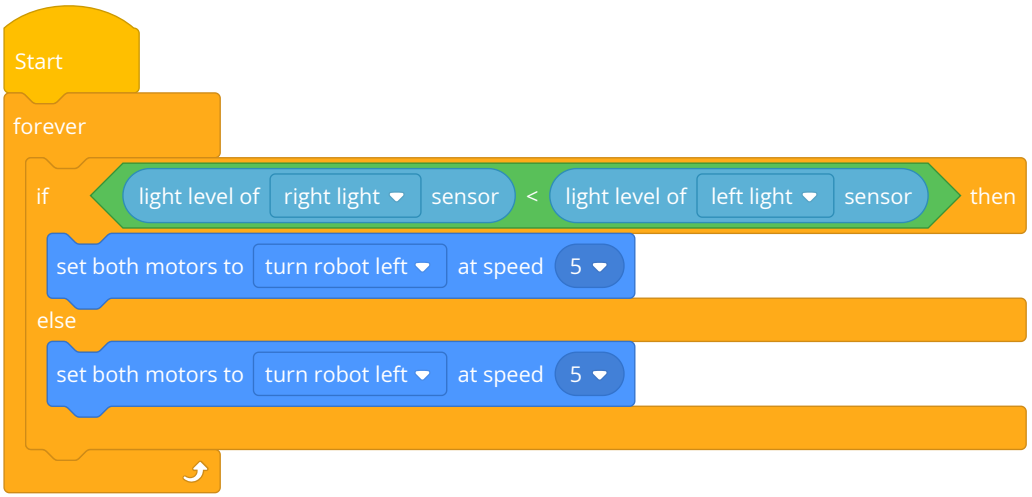
Example uses:

Example program 1: Bounce in borders



In this example, the program begins by setting the line tracker sensor LED to on, then enters a forever loop. The first block in the forever loop is the if then else block that tests if the line tracker sensor is on a non-reflective surface (black). If this is true then the first set of blocks execute driving the robot backwards and then spinning 90 degrees. The program skips over the else block and then loops back to once again test if the line tracker sensor is on a non-reflective surface (black), if this is false (on white), then the else block executes, setting both motors to drive forward. This results in the Edison robot 'bouncing' off a black line and turning around. This results in Edison driving away from detected black lines.

Example program 2: Follow the light



In this example, the program begins by entering a forever loop. The first block in the forever loop is the if then else block that tests if the light level on the right sensor is less than the light level on the left sensor. If this is true (light is brighter on the left) then the next block to execute is the set both motors to turn robot left (the robot turns left towards the light). Alternatively, if the result is false (light is brighter on the right), then the next blocks to execute are the else blocks. Now the block set both motors to turn robot right execute (the robot turns towards the light). The result is that the Edison robot drives towards the brightest light source (try using a flashlight to control where Edison drives).

The Wait until block

The wait until block pauses the program until the input condition is true.

This block has one input parameter:

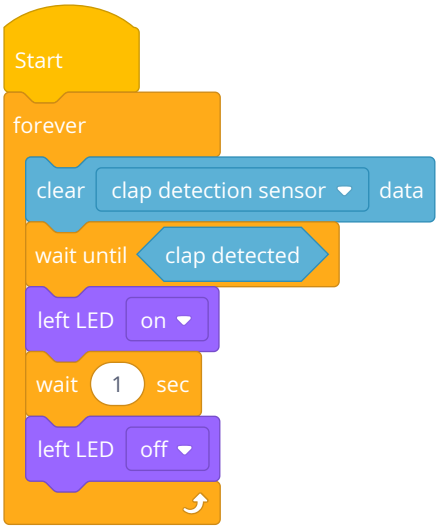
Conditional input - The conditional input accepts diamond shaped blocks that can be either an operator block or sensing block.

Note:

The wait until block is usually used inside a forever loop, so that the wait until condition can be tested multiple times. If a program has just a wait until block, then the test will be executed once and the program will exit.

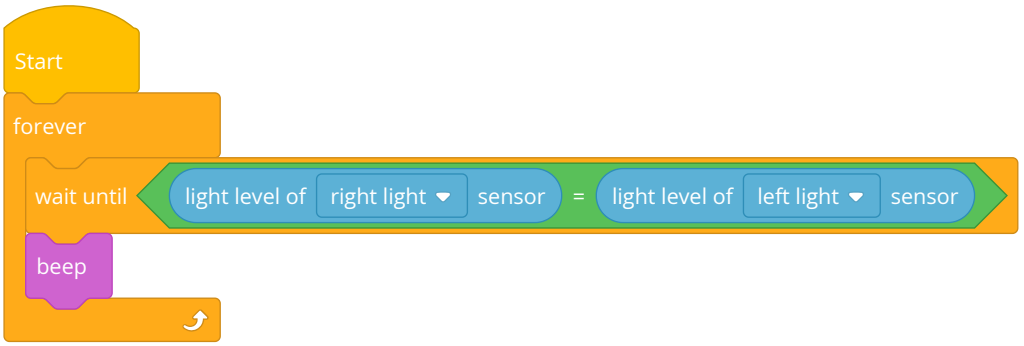
Example uses:

Example program 1: Blink LED if there is a clap



In this example, the program begins by entering a forever loop. The first block in the forever loop is the clear sensor data block that clears any false clap data. Next is the wait until block with a conditional input clap detected block. The program will pause here and wait until a clap is detected. When a clap is detected, the program progresses to set the left LED on, then wait for one second, then set the left LED off. The program now loops back to the wait until block. Each time there is a clap the left LED is set to on for one second. The result is that the Edison robot blinks its left LED for one second whenever it detects a clap.

Example program 2: Source of brightest light detector



In this example, the program begins by entering a forever loop. The first block in the forever loop is the wait until block with a conditional input using an operator requiring the light level on the right light sensor to equal the light level on the left light sensor. If the condition is not true (the light levels are not equal), then the program pauses here. When the light level on both light sensors is equal, then the program progresses to the beep block. The program now loops back to the wait until block and the program repeats. This results in the Edison robot beeping when it is pointing towards the brightest source of light

The Repeat until block

The repeat until block executes the blocks inside it, until the input condition is true.

This block has one input parameter:

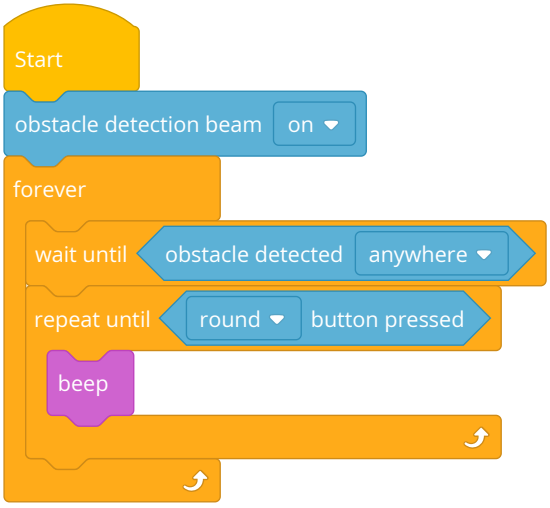
Conditional input - The conditional input accepts diamond shaped blocks that can be either an operator block or sensing block.

Note:

The repeat until block is often used inside a forever loop, so that the repeat until condition can be tested multiple times. If a program has just a repeat until block, then the test will be executed once and the program will exit.

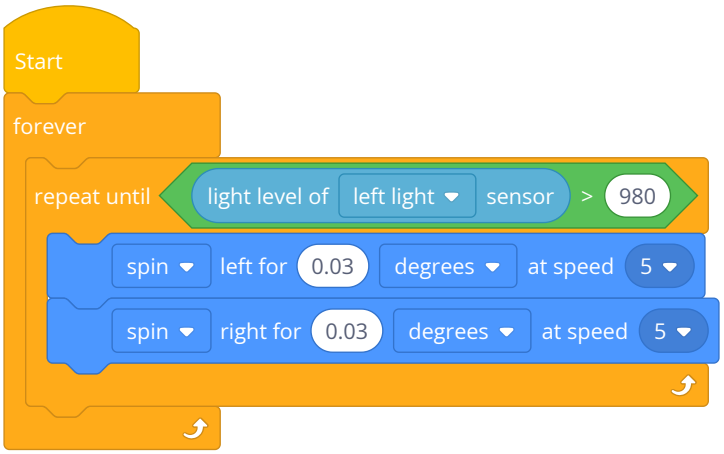
Example uses:

Example program 1: Stop the alarm from sounding



In this example, the program begins by setting the obstacle detector to on (setting the alarm) and then enters a forever loop. The first block in the forever loop is a wait until block with a conditional input of obstacle detected anywhere (the intruder detector). The program will pause and wait here until an obstacle (intruder) is detected. When an obstacle is detected, the program progresses to the repeat until block, this causes the beep block to be executed and repeat continuously (the alarm sound), 'until' the round button is pressed (stopping and resetting the alarm).

Example program 2: Edison shivers cold without warm light



In this example, the program begins by entering a forever loop. The first block in the forever loop is the repeat until block with a conditional input using an operator requiring the light level of the left light sensor to be greater than 980 (very bright/sunlight). If this is not true, then the blocks below execute, quickly spinning the Edison robot left and right (shivering). If the light level of the left light sensor increases above 980 (Edison moves into very bright light), then the condition becomes true and the program pauses (The Edison robot is warm and happy).

The Wait milliseconds block

The wait milliseconds block pauses the program for an amount of time in milliseconds (ms). A millisecond is one one-thousandths (1/1000) of a second or 0.001 seconds.

1500 milliseconds = 1.5 seconds

100 milliseconds = 0.1 seconds

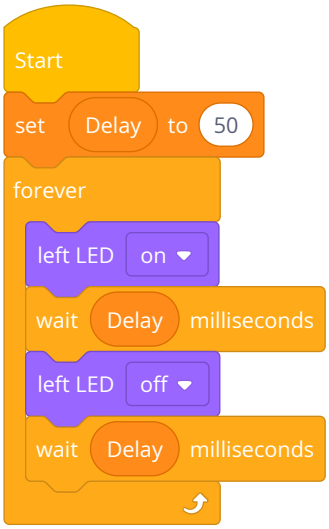
25 milliseconds = 0.025 seconds

The wait milliseconds block has just one parameter:

Value - Can be a variable, a sensing or an operator block

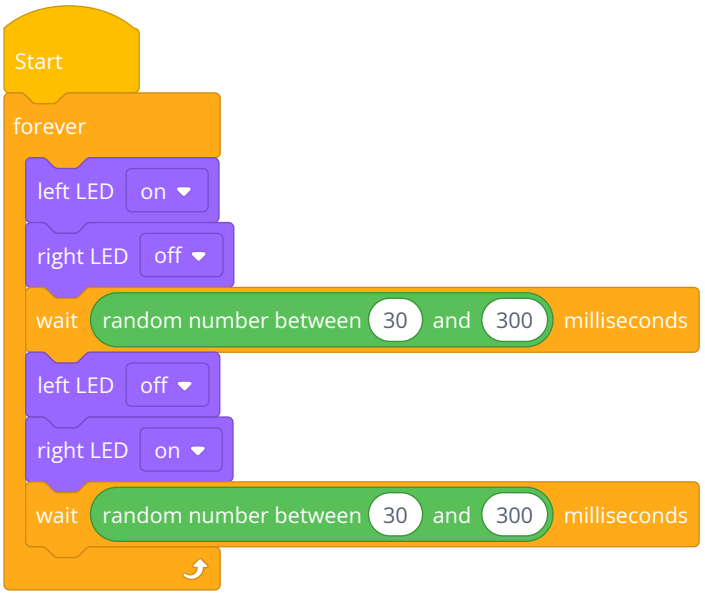
Example uses:

Example program 1: Fast blinking LED



In this example, the program begins by setting the variable 'Delay' to 50, then enters a forever loop. Inside the forever loop the left LED is set to on, then the wait milliseconds block pauses for the value in 'Delay' milliseconds, in this case 50 milliseconds. Then the left LED is set to off, then the wait milliseconds block pauses for 50 milliseconds. The program now loops back to the start of the forever loop.

Example program 2: Crazy lights



In this example, the program begins by entering a forever loop. Inside the forever loop, the left and right LEDs are set to on and off. The wait times between setting on and off are controlled by the wait milliseconds blocks. The time in milliseconds is set by the random number blocks inside them. These are set to generate random numbers between 30 and 300 milliseconds. The result is a randomised flashing of the Edison robot's LED lights.

Sensing

The Line tracker LED on/off block

The Obstacle detection beam on/off block

The Clear sensor data block

The Clap detected block

The Button Pressed block

The Obstacle detected block

The Line tracker on surface block

The Remote code received block

The IR message detected block

The Drive strain detected block

The Received remote code block

The Received IR message block

The Light level of sensor block

The Line tracker LED on/off block

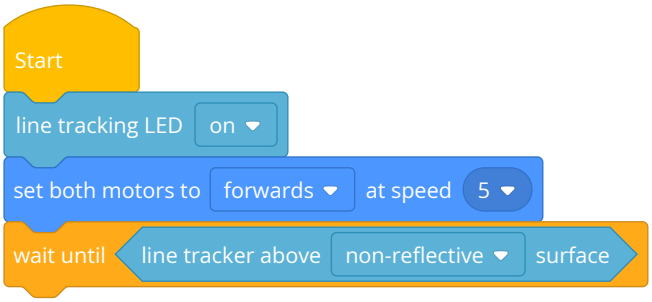
The set line tracker LED on/off block sets the line tracker LED to either on or off.

Note:

The line tracker LED must be on for the line tracking sensor to work.

Example uses:

Example program: Stop on black surface



In this example, the program begins by setting the line tracking LED on, then both motors are set to drive forwards. The program then waits until the line tracker sensor detects a non-reflective surface (black), when this happens the program ends and the motors stop.

The Obstacle detection beam on/off block

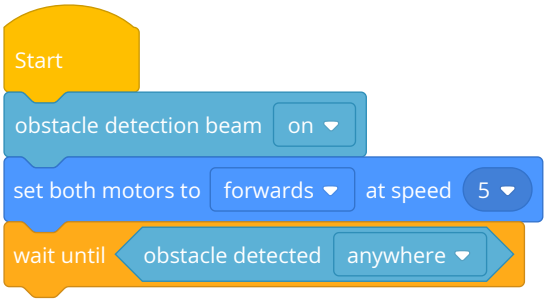
The set obstacle detection beam on/off block sets the obstacle detection beam to either on or off.

Note:

The obstacle detection beam must be on for obstacle detection to work.

Example uses:

Example program: Stop for an obstacle



In this example, the program begins by setting the obstacle detection beam on, then both motors are set to drive forwards. The program then waits until an obstacle is detected anywhere, when this happens the program ends and the motors stop.

The Clear sensor data block

The clear data block clears any sensor data (detections) from the selected sensor. This block is used when there might be false or unwanted detections that stop the program from working in the way that you want. An example is when a program drives the Edison robot as a result of detecting a clap. Generally, driving is quite noisy and can re-trigger an additional detection of a clap. This causes the program to detect a second clap that didn't happen. The clear sensor data block is used to clear the sensor data of any unwanted detections like this.

The clear data block has five input options:

clap detector

keypad

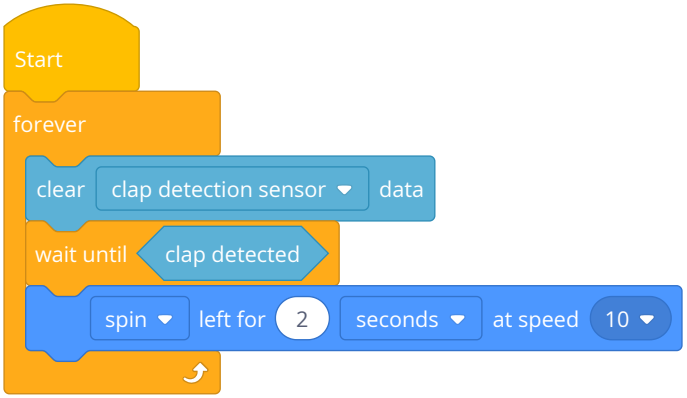
obstacle detector

remote code

IR message

Example uses:

Example program 1: Clap and spin



In this example, the program begins by entering a forever loop. Inside the forever loop, the clear sensor data block is set to clear the clap detector, next the program waits until a clap is detected, when this happens the program progresses and the robot spins left for two seconds. During the spinning the motors, wheels and skid make noise, this noise re-triggers the clap detector, so now the clap detector thinks that there was a second clap. The program loops back to the start of the forever loop (still holding onto the clap detection), but then the clear sensor data block clears the detection and the wait until block will not be triggered until there actually is a clap.

The Clap detected block

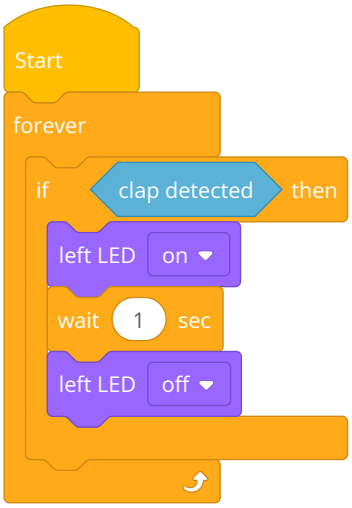
The clap detected block is a sensor input block that indicates whether or not a clap has been detected. It is a diamond shaped block so the data in the block can only be either true (clap has been detected) or false (clap has not been detected).

Note:

Other noises such as the motors driving can cause the clap detected block to false trigger. Use the clear sensor data block to clear false or unwanted clap detections.

Example uses:

Example program 1: Blink LED if there is a clap



In this example, the program begins by entering a forever loop. Inside the forever loop there is an if then block that tests if a clap has been detected. When a clap is detected, the left LED is set to on, then the program waits for one second, then the left LED is set to off. The program then loops back to the start of the forever loop. The result is that the left LED flashes for one second whenever a clap is detected

The Button Pressed block

The button pressed block is a sensor input that indicates whether or not a button has been pressed. There are two button options:

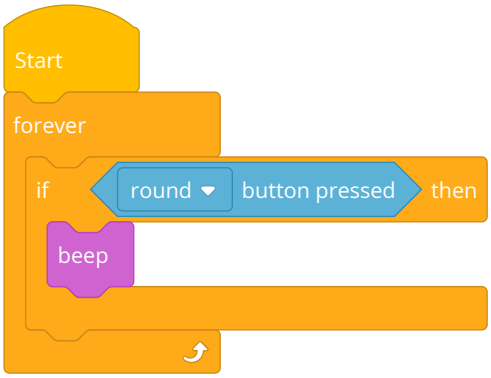
round

triangle

The button pressed block is a diamond shaped block so the data in the block can only be either true (button has been pressed) or false (button has not been pressed).

Example uses:

Example program: Beep on button press



In this example, the program begins by entering a forever loop. Inside the forever loop there is an if then block that tests if the round button has been pressed. When the round button is pressed, the program executes the beep block and then loops back to the start of the forever loop.

The Obstacle detected block

The obstacle detected block is a sensor input block that indicates whether or not an obstacle has been detected. There are four detection options:

anywhere

ahead

left

right

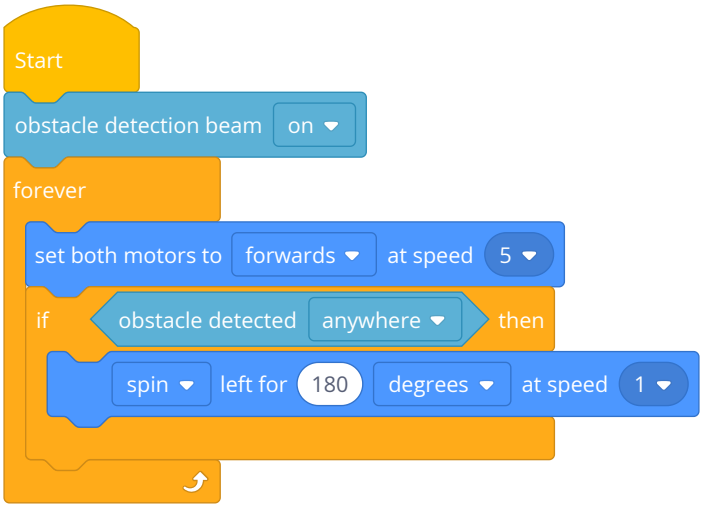
The obstacle detected block is a diamond shaped block so the data in the block can only be either true (obstacle has been detected) or false (obstacle has not been detected).

Note:

To use the obstacle detected block the set obstacle detection beam on/off block must be used to set the obstacle detection beam on.

Example uses:

Example program: Avoid obstacles



In this example, the program begins by setting the obstacle detection beam to on, then enters a forever loop. Inside the forever loop the motors are set to both drive forwards, then there is an if then block that tests if an obstacle has been detected anywhere. When this happens, the robot spins 180 degrees and loops back to the start of the forever loop. At the beginning of the forever loop both motors are again set to drive forwards. This results in Edison driving away from obstacles in its path.

The Line tracker on surface block

The line tracker on surface block is a sensor input block that indicates whether or not the surface is reflective. A reflective surface is a surface that reflects the line tracker LED's light back to the light sensor. This is usually a white surface, but a red surface will also reflect the red light from the line tracker's red LED. A non-reflective surface is a surface that does not reflect the line tracker LED's light back to the light sensor. This is usually black, but a blue or green surface will also not reflect the red light from the line tracker's red LED.

There are two detection options:

reflective (white)

non-reflective (black)

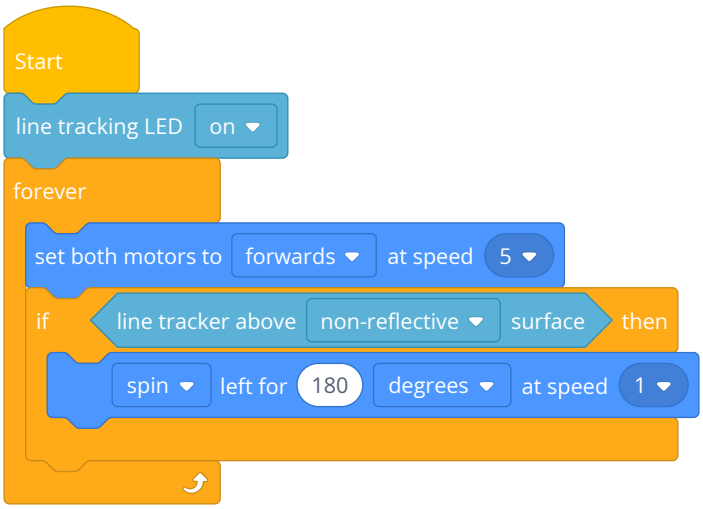
The line tracker on surface block is a diamond shaped block so the data in the block can only be either true (surface type has been detected) or false (surface type has not been detected).

Note:

To use the line tracker on surface block the set line tracking LED on/off block must be used to set the line tracking LED on.

Example uses:

Example program: Bounce in borders



In this example, the program begins by switching the line tracking LED on, then enters a forever loop. Inside the forever loop the motors are set to both drive forwards, then there is an if then block that tests if the surface is non-reflective. When this is true, the robot spins 180 degrees and loops back to the start of the forever loop. At the beginning of the forever loop both motors are again set to drive forwards. This results in Edison driving away from detected black lines.

The Remote code received block

The remote code received block is a sensor input block that indicates whether or not a particular remote-control code has been received.

There are eight remote code options from 0 to 7.

The remote code received block is a diamond shaped block so the data in the block can only be either true (code has been received) or false (code has not been received).

Note:

Make sure you press the triangle button to start your program. This is easy to forget as the Edison robot will respond as a remote-control robot and not do what you have programmed it to.

To use the remote code received block the Edison robot must first be taught which buttons of your remote control correspond to which remote code. Learn more about this here:

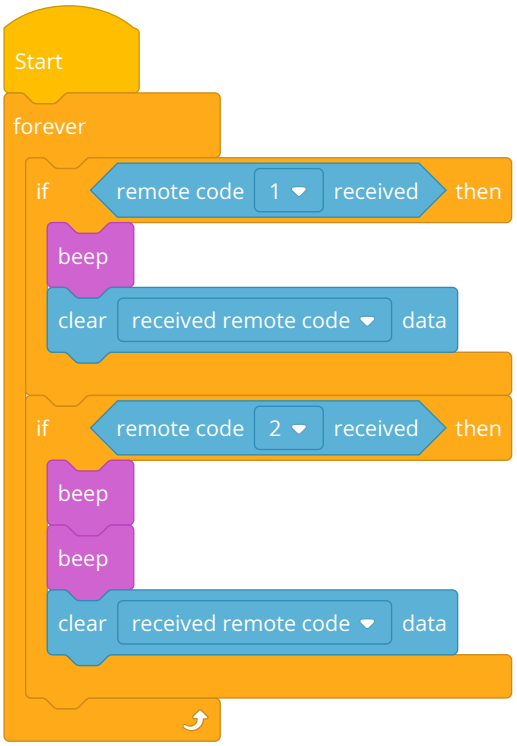
https://meetedison.com/content/Edison-robot-barcodes.pdf

If the program behaves unexpectedly, you may need to use the clear sensor data block to clear the contents of the remote code received block. See example program 1 below.

The remote code received block will work with approximately 90% of standard TV infrared (IR) remote controls, however each TV brand uses slightly different IR data formats. When Edison is decoding IR data, these differences can cause some unexpected behaviour from one remote control brand to another. Advanced remote-control decoding works best with the EdRemote (available mid 2025).

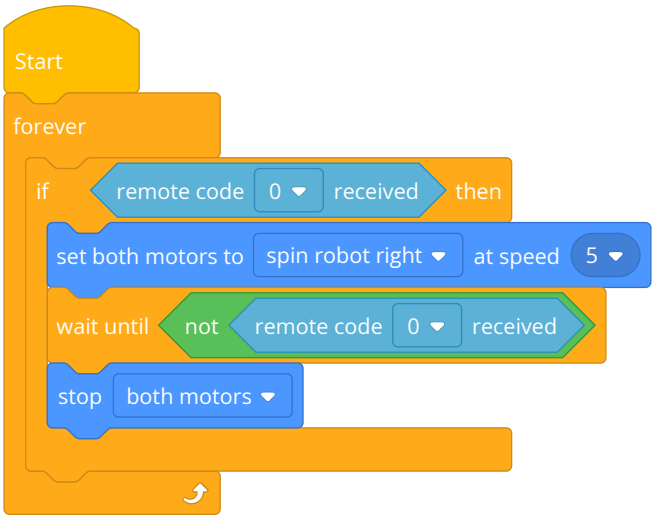
Example uses:

Example program 1: One beep or two?



In this example, the program begins by entering a forever loop. Inside the forever loop there are two if then blocks that test the received remote code. The first if then tests if the received remote code is remote code 1. If this is true then the beep block plays once and the sensor data is then cleared. If it is false, the program progresses to the next if then block and tests if the received remote code is remote code 2. If this is true then the beep block plays twice and the sensor data is then cleared. The program then loops back to the beginning of the forever loop.

Example program 2: Spin on remote control command (works best with the EdRemote)



In this example, the program begins by entering a forever loop. Inside the forever loop the if then block tests if remote code 0 has been received. If this is true, then both motors are set to spin the robot right. This continues until the wait until block condition passes. To pass the received remote code must not be remote code 0 (the remote-control button is no longer being pressed). When this happens, the program progresses to stop both motors and then loop to the start of the forever loop. This results in Edison spinning while the remote button is held down.

The IR message detected block

The IR message detected block is a sensor input that indicates whether or not an IR message has been detected (received) from another Edison robot. The term 'IR' is short for 'infrared'. Infrared refers to a band of light that is not visible to humans. This makes is useful as a means to communicate data short distances.

The IR message detected block is a diamond shaped block so the data in the block can only be either true (IR message has been received) or false (IR message has not been received).

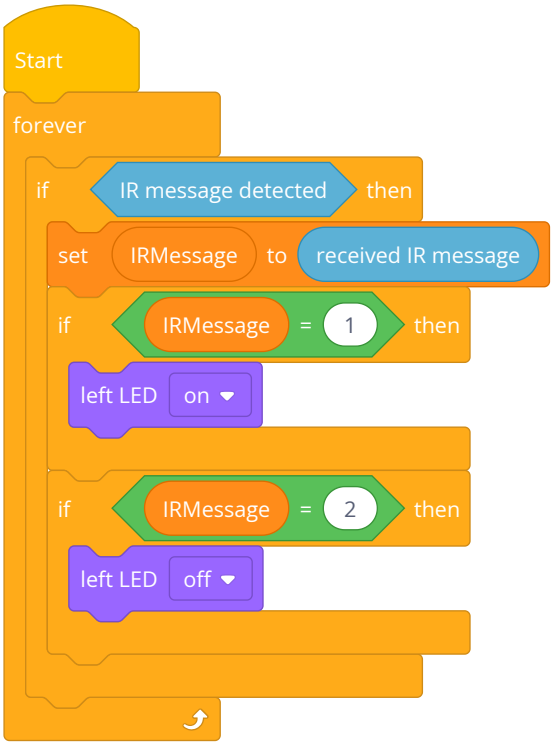
Note:

The IR message detected block does not know the contents of the IR message that was received. It only tells the program if an IR message has been received. To read the data in the IR message use the received IR message block. See the example program.

Infrared messaging between Edison robots works best indoors away from direct sunlight. The typical range can be up to 10 meters (30 feet).

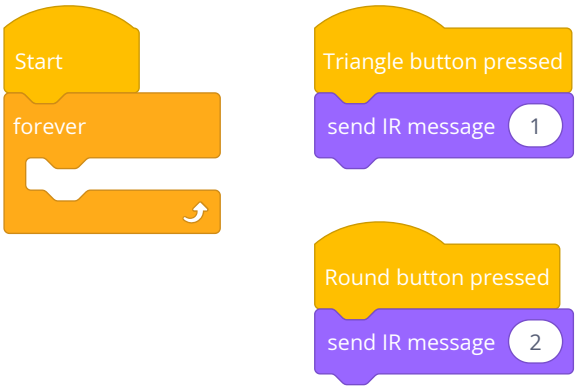
Example uses:

Example program: Edison remote light on and off



In this example, the program begins by entering a forever loop. Inside the forever loop the first if then block tests if an IR message has been received, if this is true then the variable 'IRMessage' is set to the value from the received IR message block. Doing this allows the received message to be read/tested multiple times without being cleared or lost. The next if then statement tests if the data in 'IRMessage' is equal to 1, if this is true, then the left LED is set to on. If this is not true, then the program moves on to the next if then block. This if then block tests if the data in 'IR Message' is equal to 2, if this is true, then the left LED is set to off. The program then loops back to the start of the forever loop.

The IR message sending Edison robot executes this program.



Pressing the triangle button sends an IR message with the number 1 (set the LED on) and pressing the round button sends an IR message with the number 2 (set the LED off)

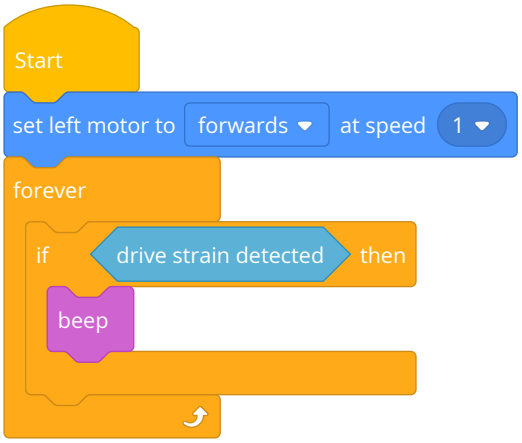
The Drive strain detected block

The drive strain detected block is a sensor input block that indicates whether or not one or both of the wheels have stopped rotating due to being under strain (physically prevented from rotating when trying to drive).

The drive strain detected block is a diamond shaped block so the data in the block can only be either true (wheel is under strain) or false (wheel is not under strain).

Example uses:

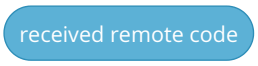
Example program: Wheel strained alarm



In this example, the program begins by setting the left wheel to drive forwards, next the program enters a forever loop. Inside the loop there is an if then block that tests if drive strain is detected. When this is true the beep block executes. The beep block will continue to execute until the strain is removed and the wheel can rotate.

The Received remote code block

The received remote code block is a sensor input block that contains data received from a remote control. The data is a number from 0 to 7.

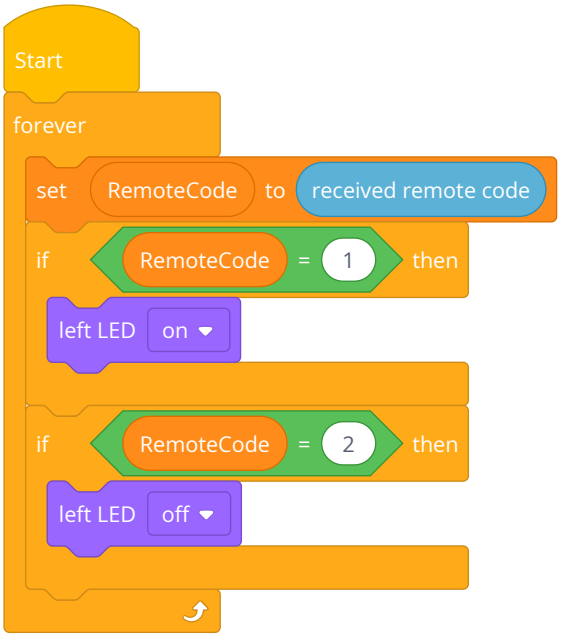


Note:

When a program executes this block, its contents are cleared. It is always best to put the data from the received remote code block into a variable before testing it with a conditional (if then) block. See example program.

Example uses:

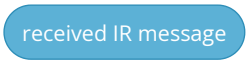
Example program: Remote control light on and off



In this example, the program begins by entering a forever loop. The first block inside the forever loop puts the data from the received remote code block into the variable 'RemoteCode'. The variable 'RemoteCode' can now be tested. The first if then block tests if contents of 'RemoteCode' is equal to 1. If this is true then the left LED is set to on. If this is false, then the next if then block tests if contents of 'RemoteCode' is equal to 2. If this is true, then the left LED is set to off. The program then loops back to the start of the forever loop.

The Received IR message block

The received IR message block is a sensor input block that contains data received from another Edison robot. The term 'IR' is short for 'infrared'. Infrared refers to a band of light that is not visible to humans. This makes it useful as a means to communicate data short distances. The data is based on an 8-bit binary number and has a range of 0 to 255.



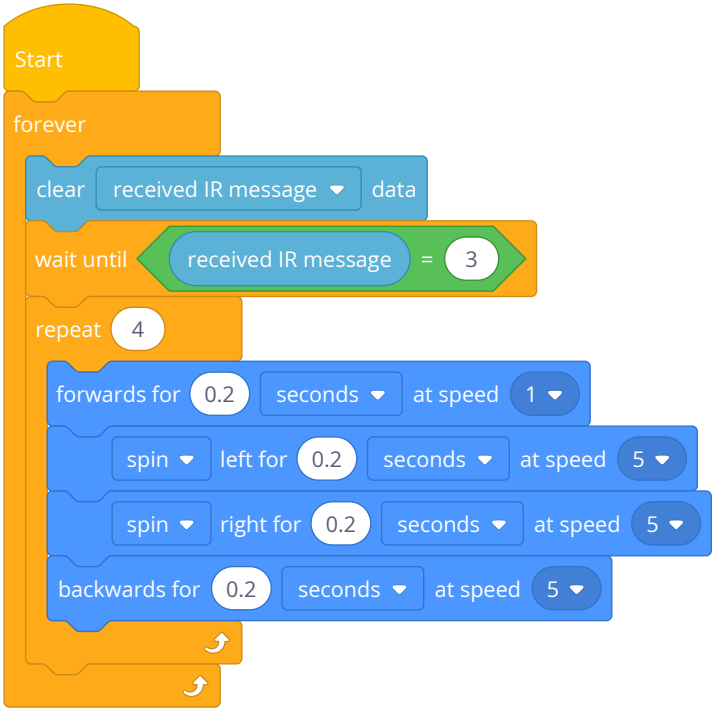
Note:

When a program executes this block its contents is cleared. It is always best to put the data from the received remote code into a variable before testing it with a conditional (if then block) block.

Infrared messaging between Edison robots works best indoors away from direct sunlight. The typical range can be up to 10 meters (30 feet).

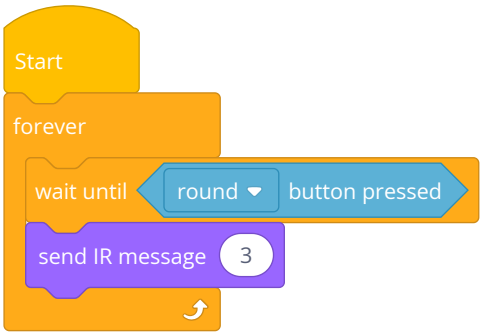
Example uses:

Example program: Start a dance revolution



In this example, the program begins by entering a forever loop. The first block inside the forever loop clears the IR message sensor data, the program then waits until an IR message is received that is equal to 3. When this happens, the program enters a repeat loop for four times. Inside the repeat loop are the Edison robot's dance manoeuvres. Each of these moves is based on time not distance. This is really important if there is a whole class of Edison robots dancing in sync to music. The dance manoeuvres repeat four times, the program then loops back to the start of the forever loop.

Example program: Send IR message 3



In this example, the program begins by entering a forever loop. The first block inside the forever loop waits until the round button is pressed, when this happens the message number 3 is transmitted via infrared light. The program then loops back to the start of the forever loop.

The Light level of sensor block

The light level of sensor block is a sensor input block that contains data from the light sensors. The light level is a number between 1 and 1000. The light level of sensor block data is constantly updated.

There are three light sensor inputs to select from:

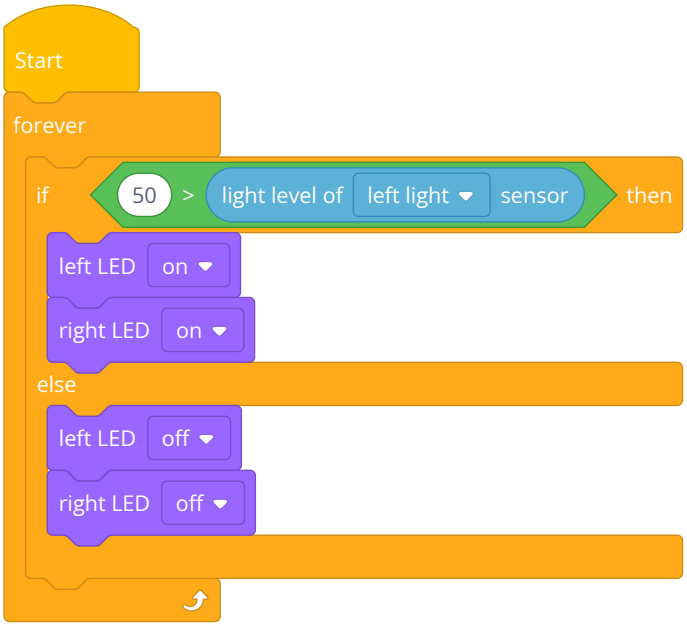
- left light (sensor at the front left side)

- right light (sensor at the front right side)

- line tracking (sensor used to detect lines under the robot)

Example uses:

Example program: Night light



In this example, the program begins by entering a forever loop. The first block inside the forever loop tests if the light level reading on the left light sensor is less than 50. If this is true, then the left and right LEDs are set to on. If this is false, then the program goes to else and the left and right LEDs are set to off. The program then loops to the start of the forever loop and repeats. This results in Edison's LEDs being on while it's in a dim light.

Operators

The Addition block

The Subtraction block

The Multiplication block

The Division block

The Random number block

The Absolute value block

The Less than block

The Less than or equal to block

The Equal to block

The Greater than or equal to block

The Greater than block

The Not equal to block

The Is between block

The And block

The Or block

The Not block

The Addition block

The addition operator block adds two values together. The values can be a constant (typed into the input and does not vary), a variable, sensing or an operator block.

Example uses:

The constant value 2 is added to the constant value 2.

The variable 'Counter' is added to the constant value 2.

The sensor input light level of sensor is added to the constant value 10.

The Subtraction block

The subtraction operator block subtracts one value from another. The values can be a constant (typed into the input and does not vary), a variable, sensing or an operator block.

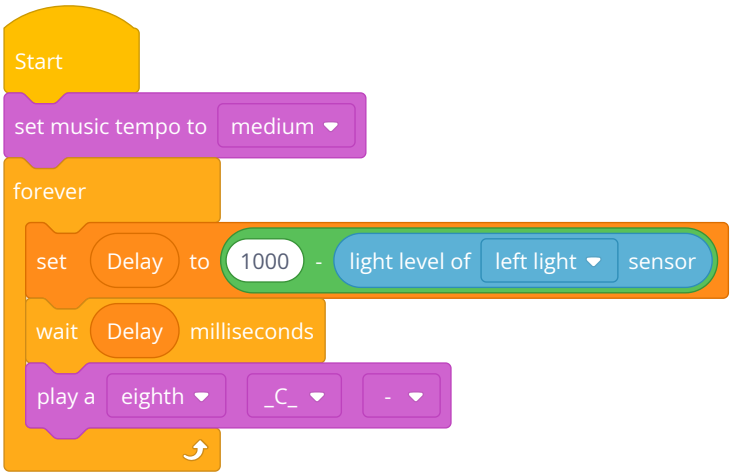
Example uses:

The constant value 10 is subtracted from the constant value 200.

The constant value 100 is subtracted from the variable 'Counter'.

The constant value 10 is subtracted from the sensor input light level of sensor.

Example program: Light radiation counter



In this example program, the music tempo is first set to very fast. The program then enters a forever loop. Inside the loop the program sets the variable (Delay) to 1000 minus the left light level (inverting the light level reading). The next block waits for the value in the (Delay) variable. The program then plays the note C and loops back to the start of the forever loop and repeats. This results in the Edison beeping slow when in darkness and faster when the light level is brighter.

The Multiplication block

The multiplication operator block multiplies two values. The values can be a constant (typed into the input and does not vary), a variable, sensing or an operator block.

Example uses:

The constant value 2 is multiplied by the constant value 4.

The variable 'Counter' is multiplied by the constant value 2.

The sensor input light level of sensor is multiplied by the constant value 4.

The Division block

The division operator block divides one value by another value. The values can be a constant (typed into the input and does not vary), a variable, sensing or an operator block.

Note:

it is not possible to divide by zero, Edison will experience an error if the right most input is zero.

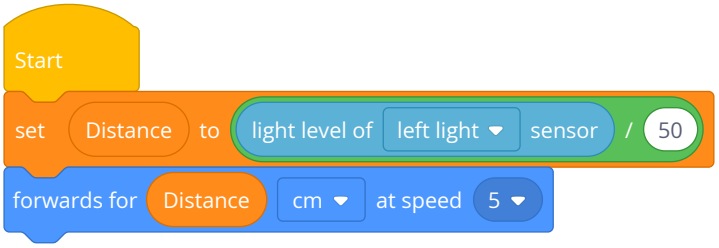
Example uses:

The constant value 10 is divided by the constant value 2.

The variable 'Counter' is divided by the constant value 2.

The sensor input light level of sensor is divided by 10.

Example program: Setting the drive distance using a variable and data from a sensor



In this example program, Edison sets the (Distance) variable to the light level from the left light sensor (a value from 1 to 1,000) divided by 50. The Edison then drives forward for the value in the (Distance) variable at speed 5. This results in Edison driving further in bright light, to a maximum distance of 20cm.

The Random number block

The random number between block generates a random number between two values. The values can be a constant (typed into the input and does not vary), a variable, sensing or an operator block.

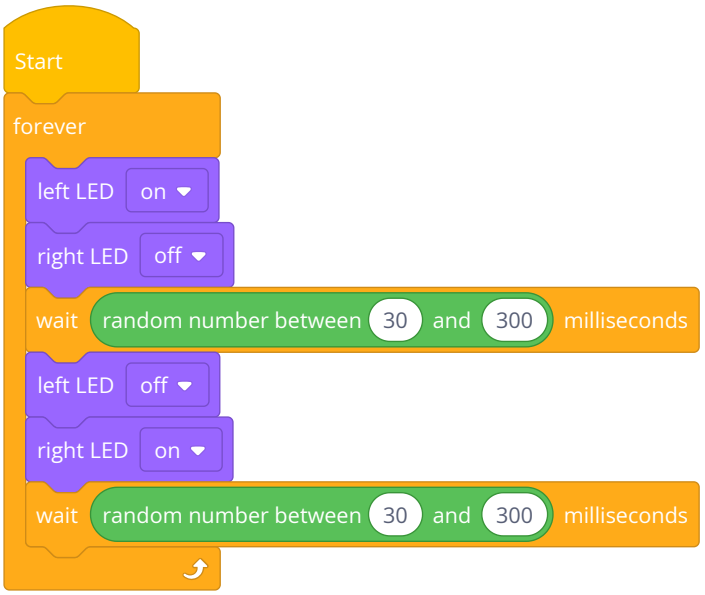
Example uses:

Generate a random number between the constant value 10 and the constant value 100.

Generate a random number between the variable 'Counter' and the constant value 5000.

Generate a random number between the constant value 10 and the sensor input light level of sensor.

Example program: Crazy lights



In this example, the program begins by entering a forever loop. Inside the forever loop, the left and right LEDs are set to on and off. The wait times between setting on and off are controlled by the wait milliseconds blocks. The time in milliseconds is set by the random number blocks inside them. These are set to generate random numbers between 30 and 300 milliseconds. The result is a crazy and inconsistent flashing of the Edison robot's LED lights.

The Absolute value block

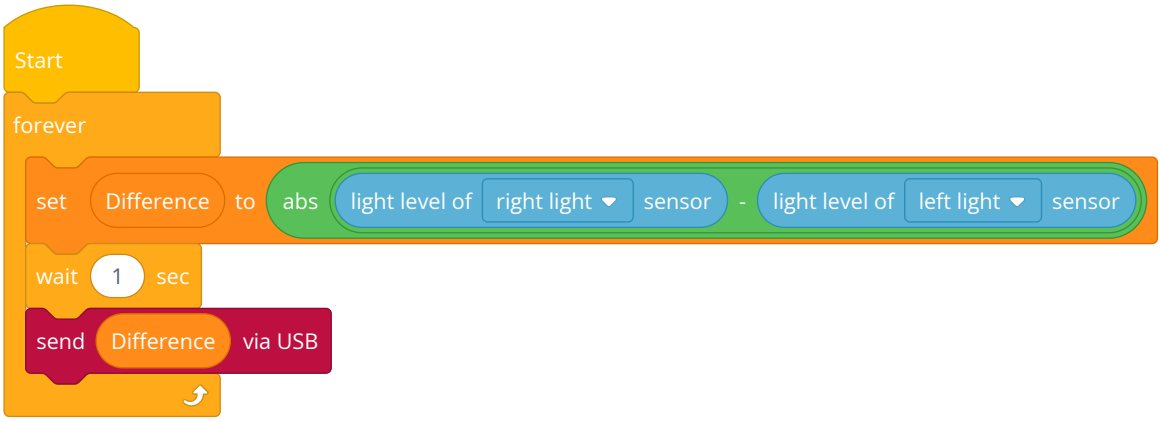
The abs (absolute value) block converts a value to an absolute value. An absolute value is a value that is always positive. The value can be a constant (typed into the input and does not vary), a variable, sensing or an operator block.

Example uses:

Has a result of the value 3.

Has a result of the value 3.

Example program: report the difference between light levels



In this example, the program begins by entering a forever loop. The first block in the forever loop sets the variable 'Difference' to the absolute value of the light level from the right light sensor minus the light level from the left light sensor. The abs block ensures that the result is always positive (no negative numbers). The program then waits for one second and then sends the value in the variable 'Difference' via USB. This value is always a positive (absolute) value.

The Less than block

The less than block compares two input values to determine if the first value is less than the second value and returns a result that is either true or false. The values can be a constant (typed into the input and does not vary), a variable, sensing or an operator block.

The less than block is a diamond shaped block so the result of the block can only be either true or false.

Example uses:

Result: true

Result: false

Is the variable 'Counter' less than the constant value 120.

Is the light level from the left sensor less than the constant value 500.

The Less than or equal to block

The less than or equal to block compares two input values to determine if the first value is less than or equal to the second value and returns a result that is either true or false. The values can be a constant (typed into the input and does not vary), a variable, sensing or an operator block.

The less than or equal to block is a diamond shaped block so the result of the block can only be either true or false.

Example uses:

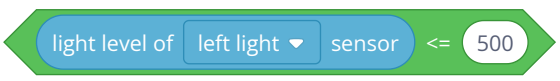
Result: true

Result: false

Result: true

Is the variable 'Counter' less than or equal to the constant value 120.

Is the light level from the left sensor less than or equal to the constant value 500.



The Equal to block

The equal to block compares two input values to determine if the first value is equal to the second value and returns a result that is either true or false. The values can be a constant (typed into the input and does not vary), a variable, sensing or an operator block.

The equal to block is a diamond shaped block so the result of the block can only be either true or false.

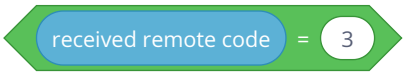
Example uses:

Result: true

Result: false

Is the variable 'Counter' equal to the constant value 60.

Is the received remote code equal to the constant value 3.



The Greater than or equal to block

The greater than or equal to block compares two input values to determine if the first value is greater than or equal to the second value and returns a result that is either true or false. The values can be a constant (typed into the input and does not vary), a variable, sensing or an operator block.

The greater than or equal to block is a diamond shaped block so the result of the block can only be either true or false.

Example uses:

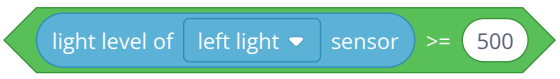
Result: true

Result: false

Result: true

Is the variable 'Counter' greater than or equal to the constant value 120.

Is the light level from the left sensor greater than or equal to the constant value 500.



The Greater than block

The greater than block compares two input values to determine if the first value is greater than the second value and returns a result that is either true or false. The values can be a constant (typed into the input and does not vary), a variable, sensing or an operator block.

The greater than block is a diamond shaped block so the result of the block can only be either true or false

Example uses:

Result: true

Result: false

Is the variable 'Counter' greater than the constant value 120.

Is the light level from the left sensor greater than the constant value 500.

The Not equal to block

The not equal to block compares two input values to determine if the first value is not equal to the second value and returns a result that is either true or false. The values can be a constant (typed into the input and does not vary), a variable, sensing or an operator block.

The not equal to block is a diamond shaped block so the result of the block can only be either true or false.

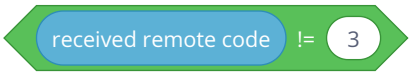
Example uses:

Result: true

Result: false

Is the variable 'Counter' not equal to the constant value 120.

Is the received remote code not equal to the constant value 3.



The Is between block

The is between block determines if a value is within a range of two other values. The values can be a constant (typed into the input and does not vary), a variable, sensing or an operator block.

The is between block is a diamond shaped block so the result of the block can only be either true or false.

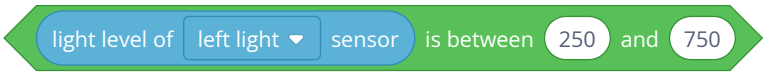
Example uses:

Result: true

Result: false

Is the variable 'Counter' between the constant value 60 and the constant value 120.

Is the sensor light level of left light sensor between the constant value 250 and the constant value 750.



The And block

The and block determines if two inputs are both true. If both inputs are true, then the result will also be true. If one or both inputs are false, then the result is false. The inputs accept diamond shaped blocks that can be either an operator block or sensing block.

The and block is a diamond shaped block so the result of the block can only be either true or false.

Example uses:

Result: true

Result: false

Is both the line tracker on reflective surface “and” drive strain detected.

Is the variable 'Counter' greater than 50 “and” received remote code is equal to the constant value 3.

The Or block

The or block determines if the first input “or” the second input are true. If one or both inputs are true, then the result will be true. If both inputs are false, then the result will be false. The inputs accept diamond shaped blocks that can be either an operator block or sensing block.

The or block is a diamond shaped block so the result of the block can only be either true or false.

Example uses:

Result: true

Result: true

Result: false

Is either the line tracker on reflective surface “or” drive strain detected.

Is either the variable 'Counter' less than the constant value 50 “or” received remote code is equal to the constant value 3.

The Not block

The not block determines if the input is “not” true. If the input is false, then the output will be true. If the input is true, then then the output will be false. The input accepts diamond shaped blocks that can be either an operator block or sensing block.

The not block is a diamond shaped block so the result of the block can only be either true or false.

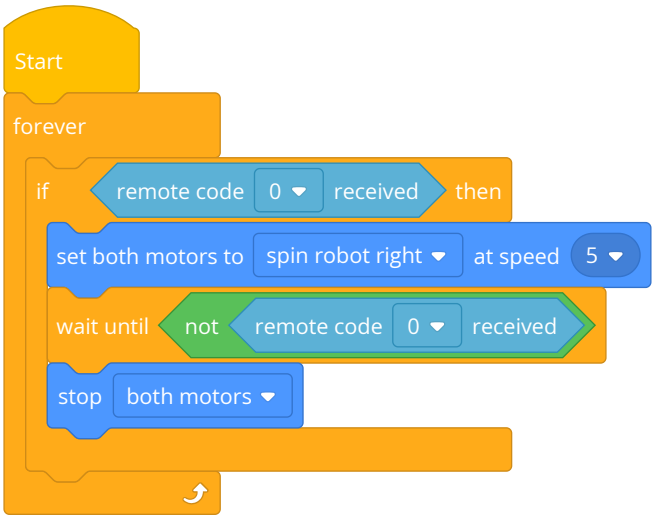
Example uses:

Result: false

Result: true

Is remote code 0 not received.

Example program: Spin on remote control command (works best with the EdRemote)



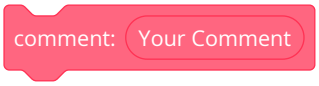
In this example, the program begins by entering a forever loop. Inside the forever loop the if then block tests if remote code 0 has been received. If this is true, then both motors are set to spin the robot right. This continues until the wait until block condition passes. To pass, the received remote code must not be remote code 0 (the remote-control button is no longer being pressed). When this happens, the program progresses to stop both motors and then loop to the start of the forever loop. This results in Edison spinning while the remote button is held down.

Comment

The Comment block

The Comment block

The comment block is used to add notes to a program. The notes describe in simple terms how the program is intended to work. Making comments in a program helps when someone else needs to edit your program or you need to edit your program a long time after you wrote it.

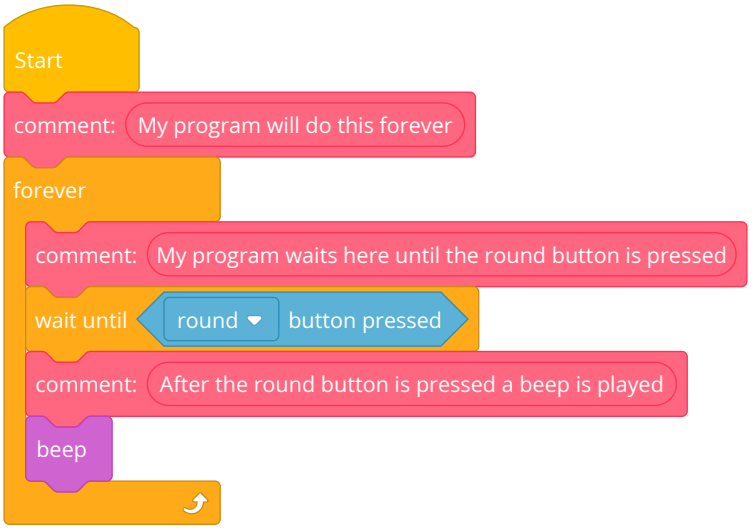


Note:

Comments are not programmed into Edison so do not change the behaviour of your program.

Example uses:

Example program: Comments explain how the program works in simple terms



Advanced

The Send via USB block

The Wait for USB data and save to block

The store value in the data log block

The store multiple values in the data log block

The Send via USB block

The send via USB block transmits a value to a computer via the USB connection. The value can either be a constant (typed into the input and does not vary), a variable, a sensing or an operator block.

To see the number on the computer, click the USB button in EdScratch and a pop-up window will appear. Click Start USB to connect to the Edison V3 robot. The data received from the Edison V3 will appear in the Data from USB window.

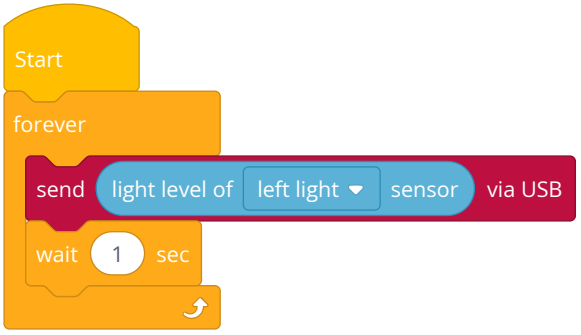
The send to USB block can also be used to help debug a program.

Note:

The Edison V3 wheels will not drive while connected to USB. This is due to international safety standards that do not allow products intended for children to be able to drive a motor while connected to a USB port.

Example uses:

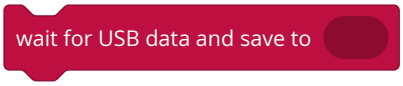
Example program: Measure light level



In this example, the program starts by entering a forever loop. The first block inside the loop sends the light level of the left light sensor via USB. The program then waits one second and repeats.

The Wait for USB data and save to block

The wait for USB data and save to block pauses the program until data is received via USB and then places the data into a variable.

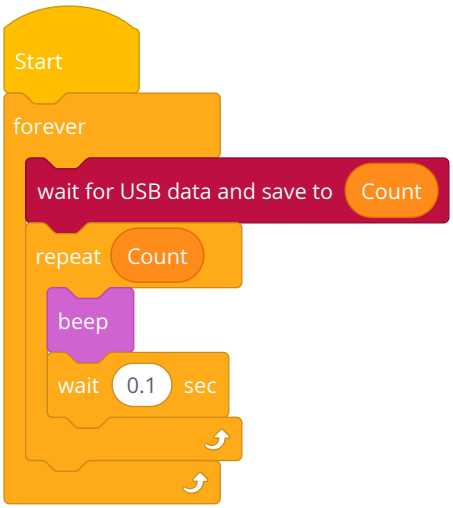


Note:

The Edison V3 wheels will not drive while connected to USB. This is due to international safety standards that do not allow products intended for children to be able to drive a motor while connected to a USB port.

Example uses:

Example program: Number of beeps from USB



In this example, the program starts by entering a forever loop. The first block inside the loop waits until data is received from USB. When data is received, the data is saved to the variable 'Count'. The program then progresses to the repeat block which repeats for the value of the 'Count' variable. Inside the repeat block the beep block and wait sec blocks execute. This results in Edison beeping the number of times sent via USB.

To test this program, click the USB button in the EdScratch app and a pop-up window will appear. Click Start USB to connect to the Edison V3 robot. To send data to the Edison V3, type a number into the Data to USB box and click the Send to USB button.

The store value in the data log block

The store value in the data log block stores a single value into Edison V3's internal data log. The log entry also contains a time stamp.

The block has 1 parameter:

Value – Can be a constant (typed into the input and does not vary), a variable, a sensing or an operator block.

Note:

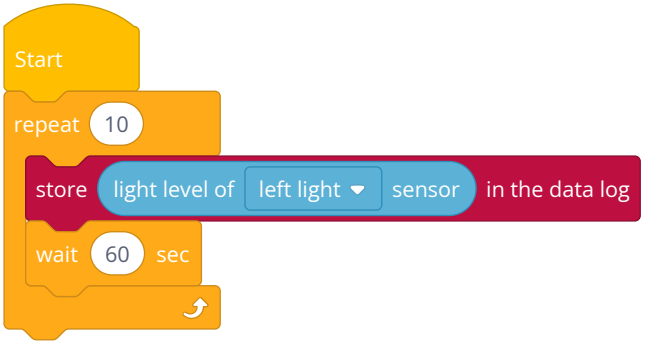
Edison V3 will keep track of the current time to save with the data in the log. Edison V3's current time can be updated using the "update time" button in the USB box.

The data log stores data between power cycles. If the data log is not cleared it could contain data from multiple program executions. The data log can be cleared using the "clear data log" button in the USB box.

Edison V3's data log can hold 4,500 records

Example uses:

Example: Log the light level of the left sensor once a minute for 10 minutes.



In this example program, the repeat loop repeats storing the light level of the left light sensor into the log and then waiting for a minute ten times. This results in Edison's log containing 10 records of light level data spaced a minute apart.

The store multiple values in the data log block

The store multiple values in the data log block stores one or more values into Edison V3's internal data log. The log entry also contains a time stamp.

The block has 3 parameters:

Value #1 – Can be a constant (typed into the input and does not vary), a variable, a sensing or an operator block.

Value #2 – Optional, can be a variable, a sensing or an operator block.

Value #3 – Optional, can be a variable, a sensing or an operator block.

Note:

Value #2 and value #3 can be left without an input. Any value left without an input will be ignored by Edison.

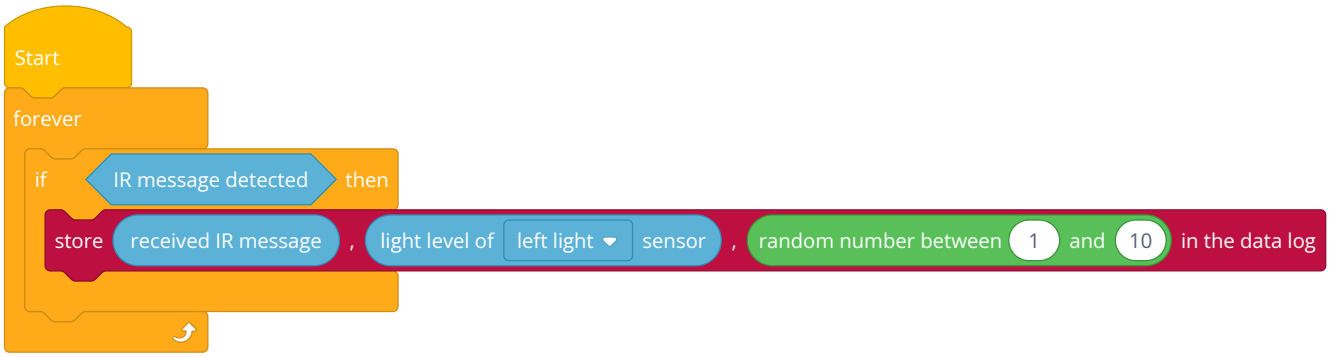
Edison V3 will keep track of the current time to save with the data in the log. Edison V3's current time can be updated using the "update time" button in the USB box.

The data log stores data between power cycles. If the data log is not cleared it could contain data from multiple program executions. The data log can be cleared using the "clear data log" button in the USB box.

Edison V3's data log can hold 4,500 records

Example uses:

Example: Log received remote codes, the current light level and a random value.



In this example, the program begins by entering a forever loop. Inside the forever loop the first if then block tests if an IR message has been received, if this is true then the value of the IR message, the light level of the left light sensor and a random number between 1 and 10 are stored in the log. The program then loops back to the start of the forever loop.

Functions

The Function start block

The Function call block

The Function start block

The function start block is where your function begins. Connect blocks to the function start block to create a custom function.

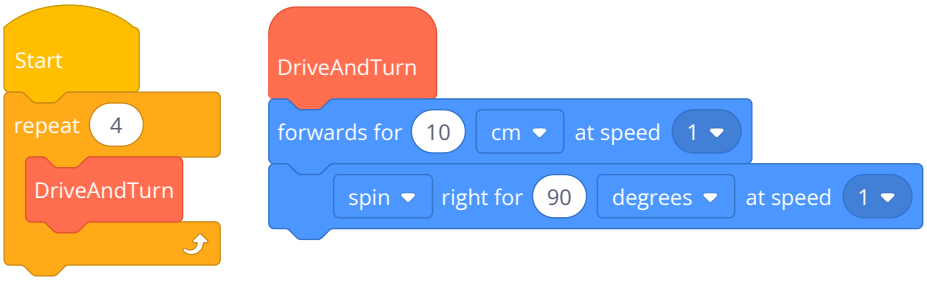
Function blocks help to simplify a program as the code can be grouped making it easier to read. Functions also help to reduce the size of a program as the repetitive parts of a program only need to be written once.

Note:

Function call blocks cannot be placed under function start blocks.

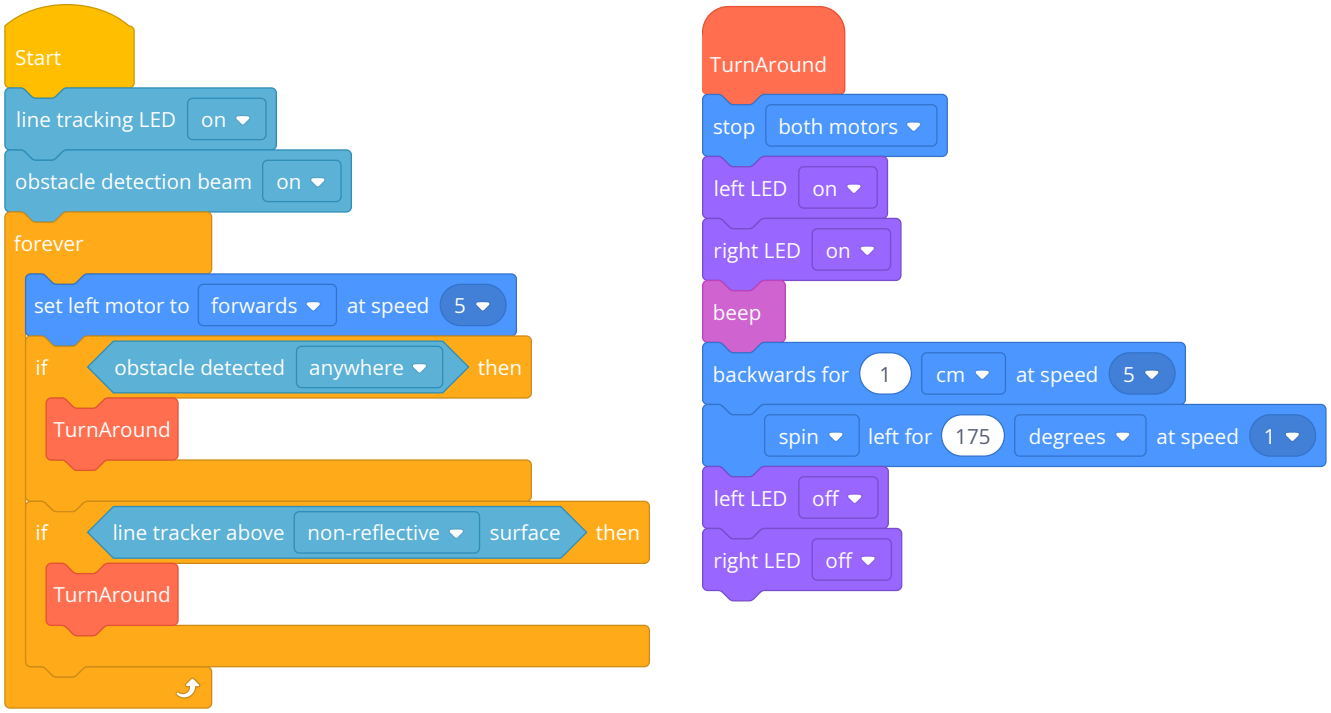
Example uses:

Example program 1: Drive in the shape of a square



In this example, the program starts by entering a repeat for four times loop. Inside the repeat block is a function call named 'DriveAndTurn'. When the program gets to this block it goes to the corresponding function start block. Under this block the robot is instructed to drive forwards 10cm, then spin right 90 degrees. This function is called four times inside the repeat loop. This results in Edison driving in a square.

Example program 2: Avoid lines and obstacles



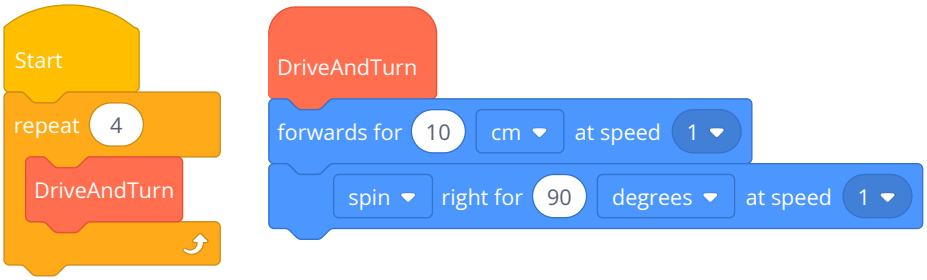
In this example, the program starts by setting the obstacle detector and line tracker LED to on and then enters a forever loop. Inside the forever loop both motors are set to drive forward. Next, the if then block tests for any obstacle detected, if true, then the function call block 'TurnAround' causes the program to the function start block 'TurnAround'. The program then stops the motors, sets the LED light to on, beeps, drives backwards, spins 175 degrees and sets the LEDs to off and exits back to the main program where it left off. The next if then block tests if the surface is non-reflective (black), if true, then the function call block 'TurnAround' causes the program to go to the function start block 'TurnAround'. The program then stops the motors, sets the LED light to on, beeps, drives backwards, spins 175 degrees and sets the LEDs off and exits back to the main program where it left off. This results in Edison driving away from lines and objects it detects.

The Function call block

The function call block tells the program to go to the function start block.

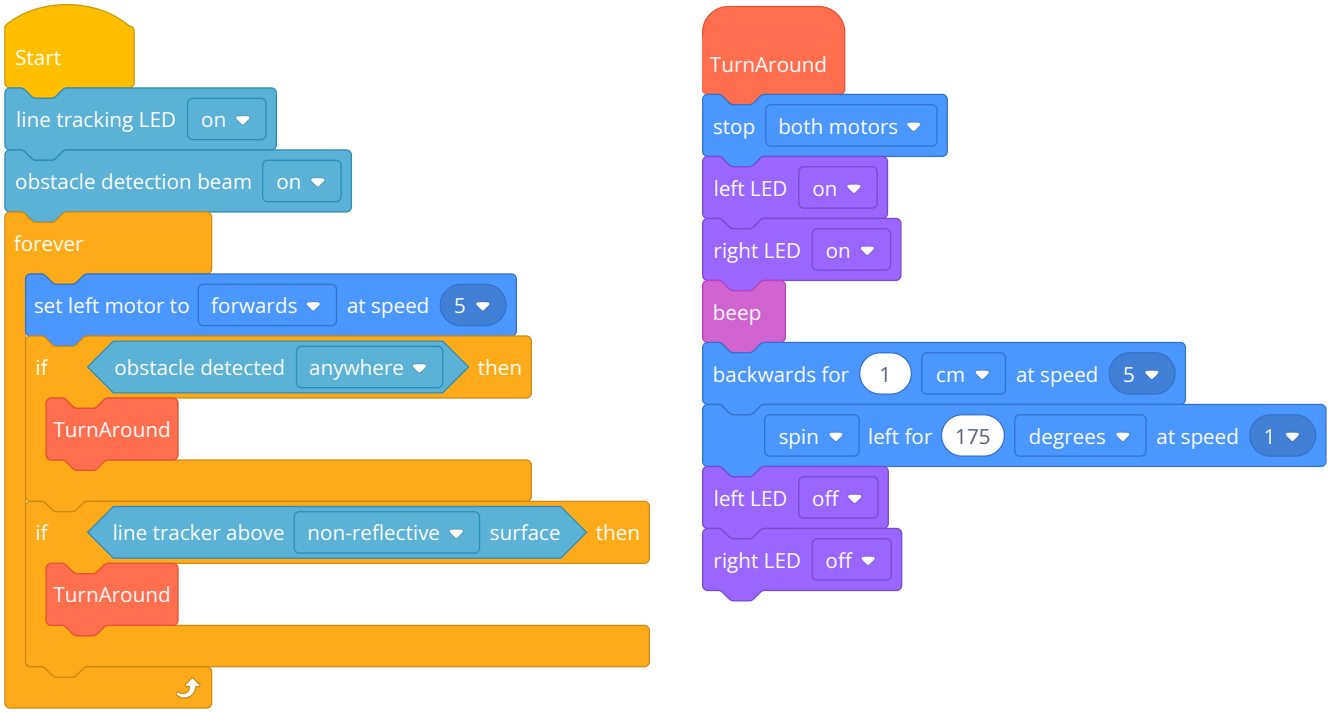
Example uses:

Example program 1: Drive in the shape of a square



In this example, the program starts by entering a repeat for four times loop. Inside the repeat block is a function call named 'DriveAndTurn'. When the program gets to this block it goes to the corresponding function start block. Under this block the robot is instructed to drive forwards 10cm, then spin right 90 degrees. This function is called four times inside the repeat loop. This results in Edison driving in a square.

Example program 2: Avoid lines and obstacles



In this example, the program starts by setting the obstacle detector and line tracker LED to on and then enters a forever loop. Inside the forever loop both motors are set to drive forward. Next, the if then block tests for any obstacle detected, if true, then the function call block 'TurnAround' causes the program to the function start block 'TurnAround'. The program then stops the motors, sets the LED light to on, beeps, drives backwards, spins 175 degrees and sets the LEDs to off and exits back to the main program where it left off. The next if then block tests if the surface is non-reflective (black), if true, then the function call block 'TurnAround' causes the program to go to the function start block 'TurnAround'. The program then stops the motors, sets the LED light to on, beeps, drives backwards, spins 175 degrees and sets the LEDs off and exits back to the main program where it left off. This results in Edison driving away from lines and objects it detects.