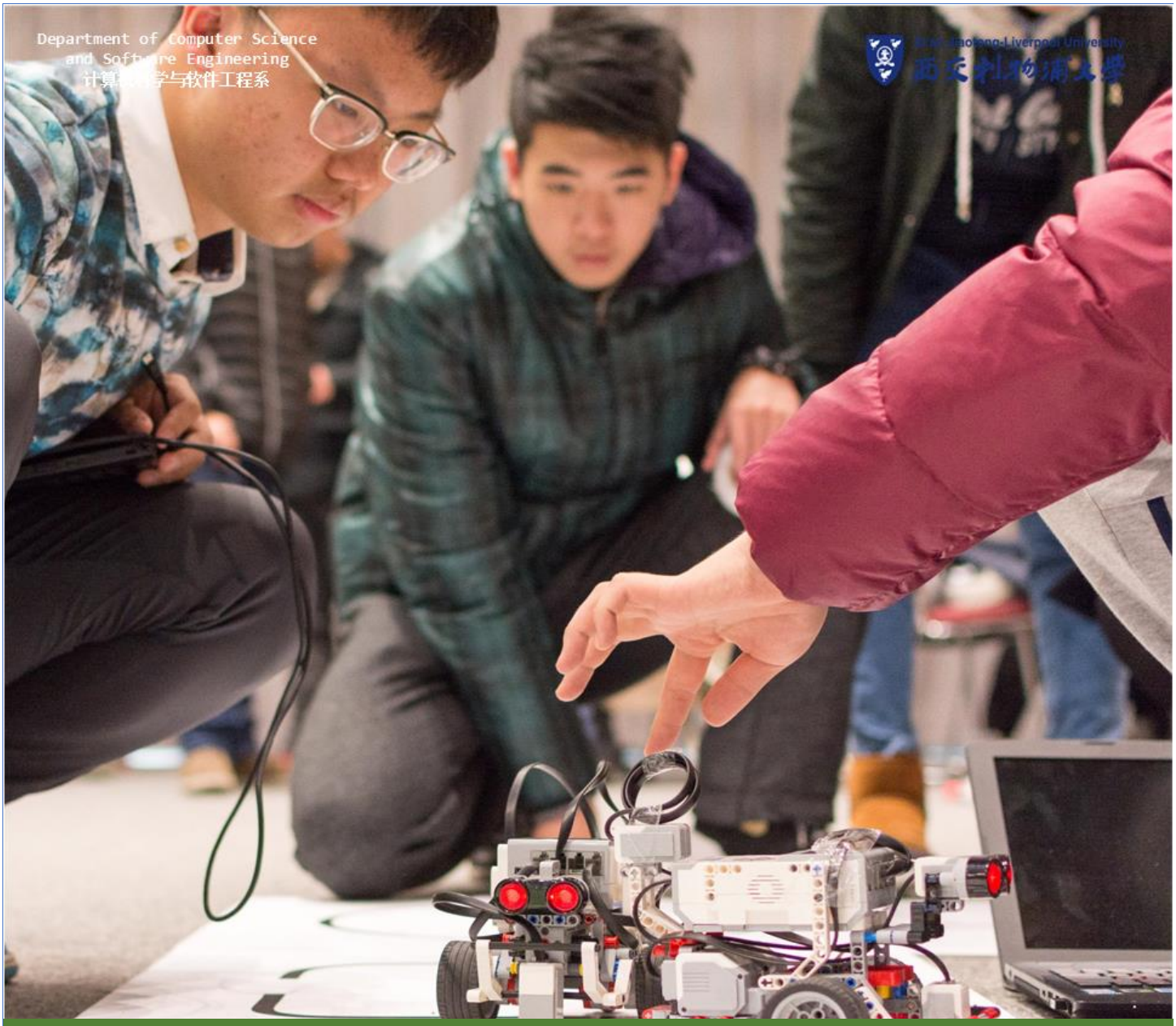


LEGO Mindstorms Robot Programming Tutorial Handbook 2019

CSE003 Fundamentals of Computer Programming

Lab Sessions: 4 to 7 **Weeks: 11, 12, and 13** **Semester 1, 2019-2020**

CSE003 – Fundamentals of Computer Programming
Lab 4 to 7: Introducing LEGO Mindstorms Robot Programming

Weeks 11 to 13, Semester 1, 2019-2020

This tutorial handbook introduces the concepts and techniques required to write a program using the LEGO Mindstorms editor. It also helps you understand the fundamentals of programming with a practice project, so you may develop your own Mindstorms program.

本教程手册介绍了使用 LEGO Mindstorms 编辑器编写简短的 Mindstorms 程序所需的概念和技术。这些教程将通过实践项目使您了解基础的编程特性并可以独立开发 Mindstorms 程序。

How this tutorial handbook is organised?

本教程手册是如何组织的?

This tutorial handbook has four chapters:

- Introduction to robots
- Introduction to programming
- Simple practice projects
- Introduction to LEGO Mindstorms connection to Scratch

教程手册包含了三个部分：

- 介绍机器人概念
- 介绍编程基础
- 一些简单的实践项目
- 介绍 LEGO Mindstorm 和 Scratch 的连接

How to use this tutorial handbook?

If this is your first time reading this tutorial, we recommend you skim both introduction chapters and then get a feel about the practice project (not assessed). Afterwards, you may come back to both introduction chapters for more details.

如果你第一次接触这种教程， 希望你们先粗略的浏览下 介绍章节 (不在评分标准中)，并且了解一些实践项目之后再仔细地阅读介绍章节的内容。

Why do you need to use this tutorial handbook?

为什么需要使用本教程手册？

This tutorial handbook covers labs from weeks 11 to 13. The group project, which weighs 35% of the CSE003 module mark and is based on the Scratch visual programming and the LEGO Mindstorm programming. Therefore, this tutorial handbook acts as a guide to successfully complete your group project of the CSE003 module.

本教程手册涵盖了 11 至 13 教学周的实验课内容。这次的小组作业占 CSE003 模块的 35%，其完全基于 Scratch 可视编程以及 LEGO Mindstorm 编程。因此，本教程手册是成功完成 CSE003 模块小组作业的指南。

Table of Contents

Below is a simple table of contents, you can navigate among chapters and sections:

This is the full table of contents and you can use it navigate among subsections as well.

Introduction to Mindstorms Robots

In this chapter you will get yourself familiar with LEGO Mindstorms Robot's hardware components. The details are covered by Dr Hai-Ning Liang in "CSE003 Fundamentals of Computer Programming". These tutorials will allow you to understand more about the hardware before starting the programming.

Note: your group either receive a set EV3 robot. You will only need the brick, servo motors, touch sensor, light sensor, ultrasonic sensor, cables and other non-electric components to finish the assignment.

The Brick

1. The brick (see Figure 1) is responsible for the control and for the power supply of a MINDSTORMS robot.
2. The LEGO brick connected with motors and sensors lets your agent perform different operations.

Motor Ports

1. The EV3 has four output ports for attaching motors – ports A, B, C and D.
2. Motor ports are output ports in the sense that through the motors the robot can do things. The motors are actuators in robotics terms.

Sensor Ports

1. EV3 has four input ports for attaching sensors – ports 1, 2, 3 and 4.
2. Sensor ports are input ports in the sense that the robot senses all the information around it through the sensor ports.
3. Be aware of the distinctions of both ports, they are not interchangeable.

USB Port

1. You can connect a USB cable to the USB port and download programs from your computer to the brick (or upload data from the brick to your computer).
2. You can alsouse the wireless Bluetooth connection (may not compatible to all Bluetooth devices though) for uploading and downloading.

Figure 1 An EV3 Brick

The Motors

The Servo Motors

The servo motor gives your robot the ability to move. With MINDSTORMS software the motors are even able to synchronize, so that a robot having one motor for each side could move in a straight line.

Built-in Rotation Sensor

The motors can be used either as motors and or as sensors, because each motor has a built-in rotation sensor. This lets you control your agent's movements more precisely. The rotation sensor measures motor rotations in degrees or full rotations (accuracy of ± one degree). This means that you could set a motor to turn for example 45°. The built-in rotation sensor in each motor also lets you set different speeds for your motors (by setting different power parameters in the software).

Figure 2 (Left) EV3 Servo motor and (right) EV3 medium motor.

The Medium Motors

The medium motors are only with EV3. Compared to a standard servo motor, it has all a servo motor's features but comes with reduced size and reduced power and torque.

The Sensors

The Touch Sensor

The touch sensor detects when it is being pressed by something and when it is released again.

Figure 3 Touch sensor for EV3 robots.

The Light Sensor

The light sensor enables your agent to read the intensity of reflected light and display intensity in percent.

Figure 4 Light sensor for EV3 robots.

The Ultrasonic Sensor

1. You can use the ultrasonic sensor to measure distance and detect movement. Therefore, you could use it for obstacle avoidance.
2. The distance is measured in centimetres and in inches.
3. The sensor is able to measure distances from 0 to 255 centimetres with a precision of ± 3 cm.
4. The sensor uses the same scientific principle as bats: it measures distance by calculating the time it takes for a sound wave to hit an object and return – just like an echo.
5. Large sized objects with hard surfaces return the best readings.
6. Objects made of soft fabric or that are curved (like a ball) or are very thin or small can be difficult for the sensor to detect.
7. Note that two or more ultrasonic sensors operating in the same room may interrupt each other's readings.

Figure 5 Ultrasonic sensor for EV3 robots.

The Gyro Sensor

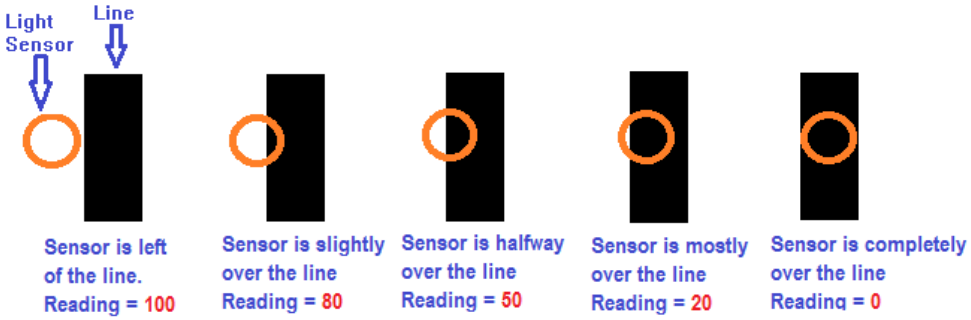
1. The EV3 Gyro Sensor measures the robot's rotational motion and changes in its orientation. You can measure the change of angles from its base position just like measuring with a gyroscope.
2. The accuracy is of +/- 3 degrees and maximum output is of 440 degrees/second.

Figure 6 Gyro sensor for MINDSTORM EV3

An Example: Line Following with the Light Sensor

We are going to use the *Light Sensor* to help our robots follow along a dark line. Specifically, we will follow the ***edge*** of the line.

Let's say we want our robot to follow a dark line on white paper. Remember, if we *calibrated* it, the light sensor will output a value from 0 to 100 depending on how much of the sensor is over the dark line. 0 means the sensor is totally over the line. 100 means the sensor is only over the white paper. Imagine what the sensor would see if it moved from left to right over a dark line (see picture below):



A line following program

We're going to tell our robot to follow the **left edge** of the line. In the picture above, we see that the sensor reading is **high** (> 50) if the sensor is mostly over the paper, and the sensor reading is **low** (< 50) if the sensor is mostly over the line. We can use this fact to keep the sensor lined up with the left edge of the line while the robot moves forward.

The following pseudocode does this:

- Get output from the sensor
- If the light sensor output is greater than (>) 50, we must be left of the line's edge
    - Move forward and turn slightly to the right
- If the output is less than (<) 50, we must be right of the line's edge
    - Move forward and turn slightly to the left
- Repeat from the beginning

Now you have finished your introduction, continue to read the next session to know about some details about programming and try the practice project to get yourself on board.

Introduction to LEGO Mindstorms Programming

This lab tutorial gives you a gentle introduction to programming your LEGO Mindstorms robot. You may also use it to look up for certain command or to check some of the examples.

We recommend you to quickly read this chapter and move on to the next chapter: practice project to get a feel about programming and then comes back here.

Useful Tips

Here are some useful tips you should know after getting your hands on the robots:

1. Always, always, always keep older versions of your program after making major changes. It is a common sense in programmer's world. Not without saying very occasionally the EV3 program will crash or keeps alerting you because of some not-so-obviously misplaced paths.
2. Know how a sensor works. For instance, the light sensor emits light and use the reflected light to determine light intensity and to estimate colour. Try to keep the distance of light sensor and surface to be measure around 1cm. Changes in ambient light also affects the sensor's reading.
3. Sensors have their limits, actuators too. In practice ultrasonic sensor would not be effectively measure objects that are 200cm away. Motors rotations will always gain and accumulate little errors so don't expect a robot to make several perfect 90 degree turns in a row.
4. If you are unsure about something, test it.

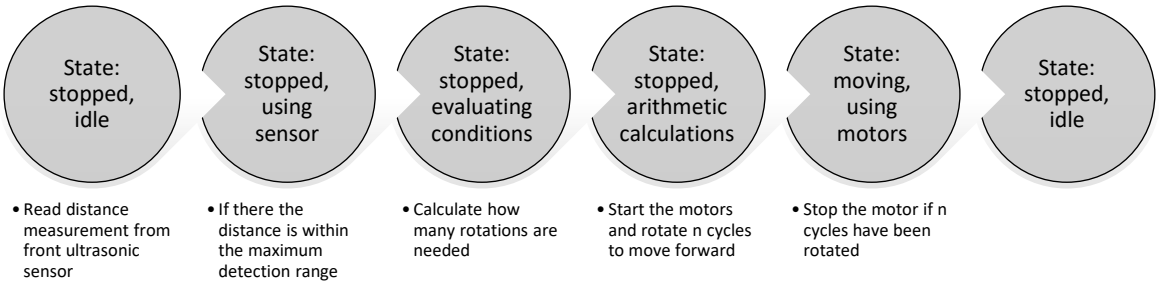
Have fun with your robots ☺

Programming Mindstorms: Concepts

This section introduces core concepts of Mindstorms programming; you will grasp some intuitions on programming. Don't panic if you do not recognise some terms, block names or block functionalities, move on and you will get the ideas when you start programming.

Commands and States

The paradigm of programming a Mindstorms is *imperative*: the program changes the *states* of the robot with each command. To illustrate, say if we wish a robot to detect an object in the front, to move forward and to stop by it, the program can look like this:



While programming, try to keep in mind how the states of the robots will be changed with each command in your program; this will get you out of some difficult situations.

Blocks and Commands

A block in LEGO Mindstorms software is a command and it changes the states of the robot. A state can be the status of certain action of a robot or the value of variable in the robot's internal memory.

The blocks have five types: actions, sensors, control flows (flows), data operations (data), advanced and customised blocks. In this tutorial you only need to know about **actions**, **sensors**, **flows** and **data**.

The blocks for **EV3** robots are different but the functionalities are mostly the same.

Action blocks

The Action blocks allow you **to control specific types of behaviour related to various output devices**. Like moving and rotating.

Medium Motor, Large Motor, Move Steering, Move Tank, Display, Sound, Brick Status Light (EV3)

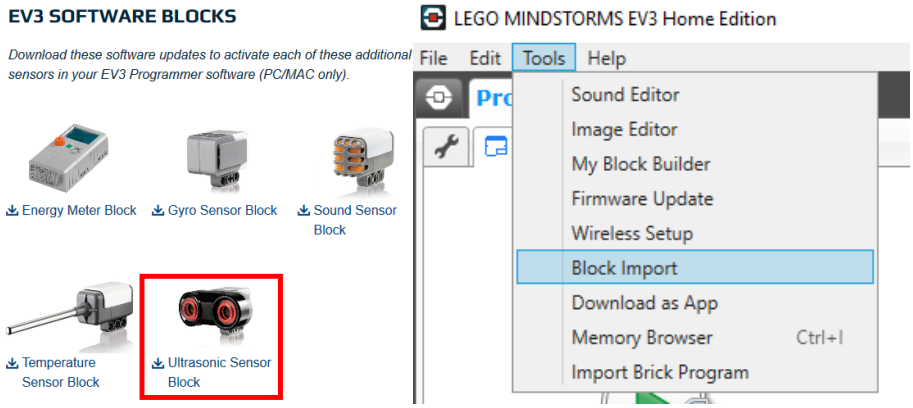
Sensor blocks

Sensor blocks mostly **take the measurements from sensors**. Use these blocks in combination with the sensors on your robot to control behaviour.

Brick Buttons, Colour Sensor, Infrared Sensor, Motor Rotation, Timer, Touch Sensor. (EV3)

Note, if the ultrasonice sensor block is unavailable in your EV3 software, please download and import the block from lego website and then import it into your EV3 software.

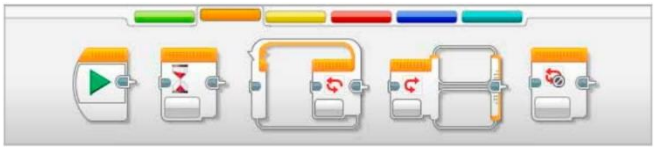
https://www.lego.com/en-us/mindstorms/downloads



Flow blocks

These blocks allow you to **combine blocks to create more complex behaviours**.

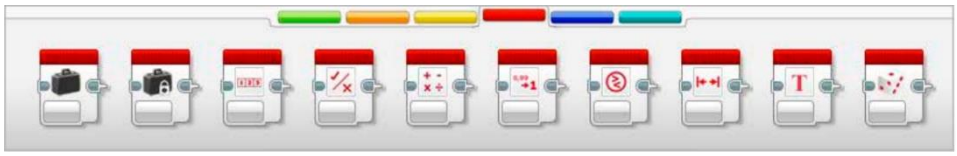
Start, Wait, Loop, Switch, Loop Interrupt. (EV3)



Data blocks

Data blocks defines the operations on numeric, Boolean and string variables. With these data blocks you can **read and write variables**, **perform arithmetic and logic calculations** and so on **to enable finer and more sophisticated manoeuvre** on your robots.

variable, constant, array operations, logic operations, math, round, compare, range, text, random (EV3)



Common blocks

Common blocks are a few frequently used blocks, most of them are duplicates from above mentioned blocks.

Programming Mindstorms: Techniques:

Connecting your robot

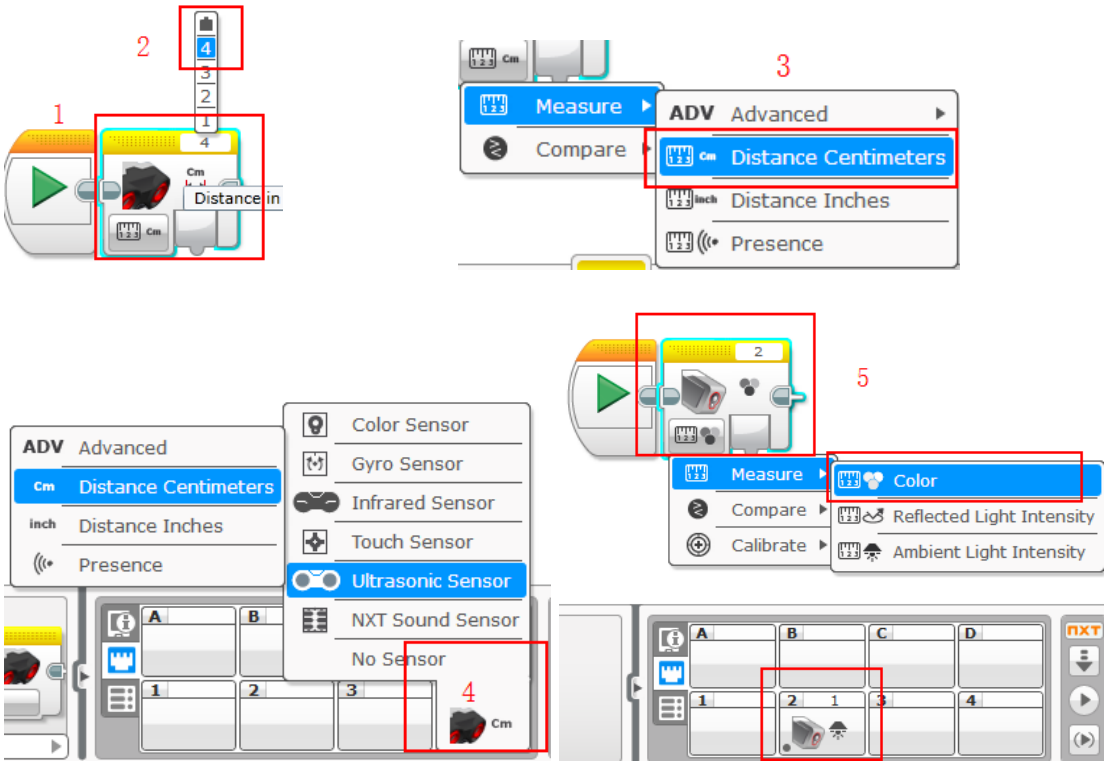
EV3:

Checking port view (Ultrasonic and light/Colour sensor)

Connect an ultrasonic sensor to port 4, connect the colour sensor (EV3) to port 2.

1. Drag and drop an ultrasonic block and click on it
2. Select port 4
3. Select metric unit
4. Play with the sensor and check the readings
5. Drag and drop a light/colour sensor block, now take your sensor around and measure objects in different colours, under different lightning conditions.

EV3:

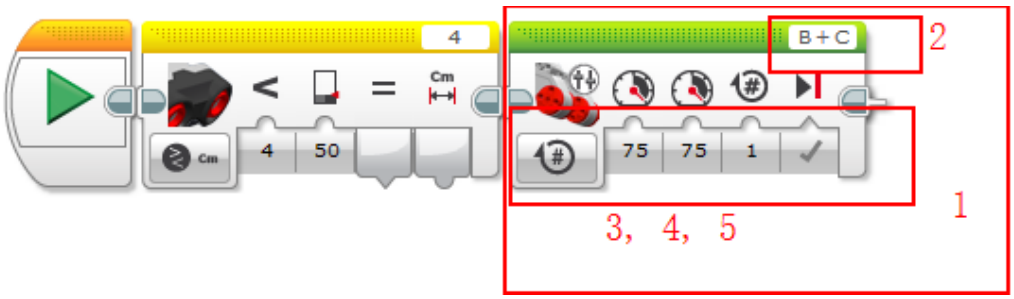


Moving in a straight line

1. Drag move block and place it after our ultrasonic sensor. Note for now the ultrasonic has no control over our robot's movements since we have not yet use the sensors output, you will know how to make use of the sensor later in this tutorial.
2. Select the ports for both motors
3. The direction of motor rotation on EV3 models are defined by positive or negative power. The actual movement direction depends on how and where the motors are installed on the robot.
4. Now define which motor is the left and which is on the right. The power balance can be adjusted so that the robot can turn or rotate.
5. Power, rotation duration and next action of both motors can be adjusted.
    a. Higher power allows the robot to move faster but induces more momentum, thus less manoeuvrability; lower power is slower, and robot may not be able to climb a slope.
    b. The working duration of the motors can be defined by cycles/degrees of rotations or time in seconds. Moving by rotations loosely controls the distance of the robot will travel, regardless of the power setting.
    c. Coasting allows the robot to travel a short distance after the motors stop.

(see illustrations on the next page)

EV3:

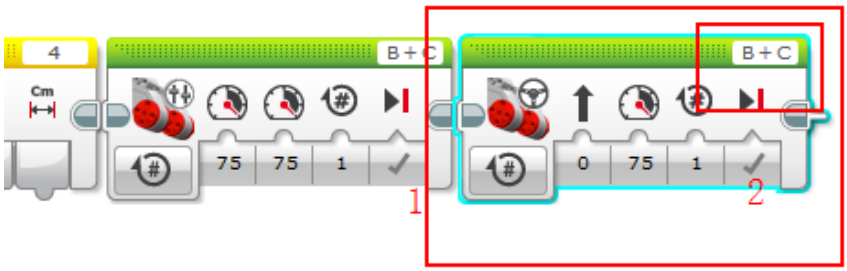


Rotation

Rotation is achieved by only rotating a single motor. If both motors are running while one runs faster than the other, then the robot will take a turn rather than rotate.

1. Move steering block (EV3) after the move block
2. Choose both ports (EV3) that connected to the motors

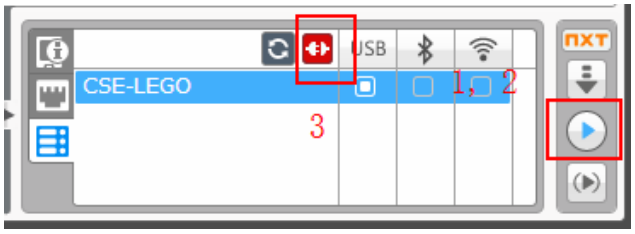
EV3:



Download to Mindstorms

1. Click to download the program to your robot
2. Wait for completion
3. Safely remove the robot. Then you can disconnect the robot. The software will be available on the brick in “my files” -> “software files”.

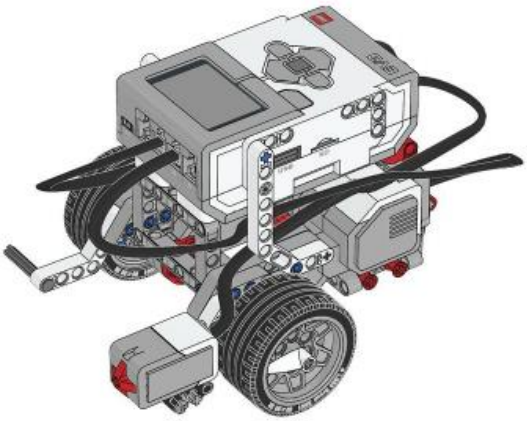
EV3:



Assembly

Build an EV3 bumper car. Your build does not have to be the same as the example below, just make sure:

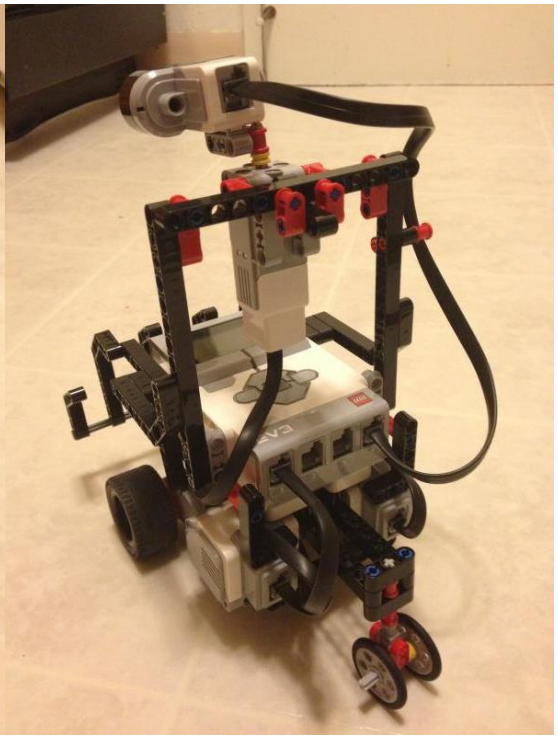
- there is a bumper on the front connected to port 1
- two motors are connected to port C and B.



Once completed, add an ultrasonic sensor on the top and make it face the front.

- Make sure the ultrasonic sensor is connected to port 4.
    - ultrasonic sensor can in fact be connected to any port, but the examples below assume it is connected to port 4.

You can even add another medium motor (EV3) under the ultrasonic sensor to make a simple radar.



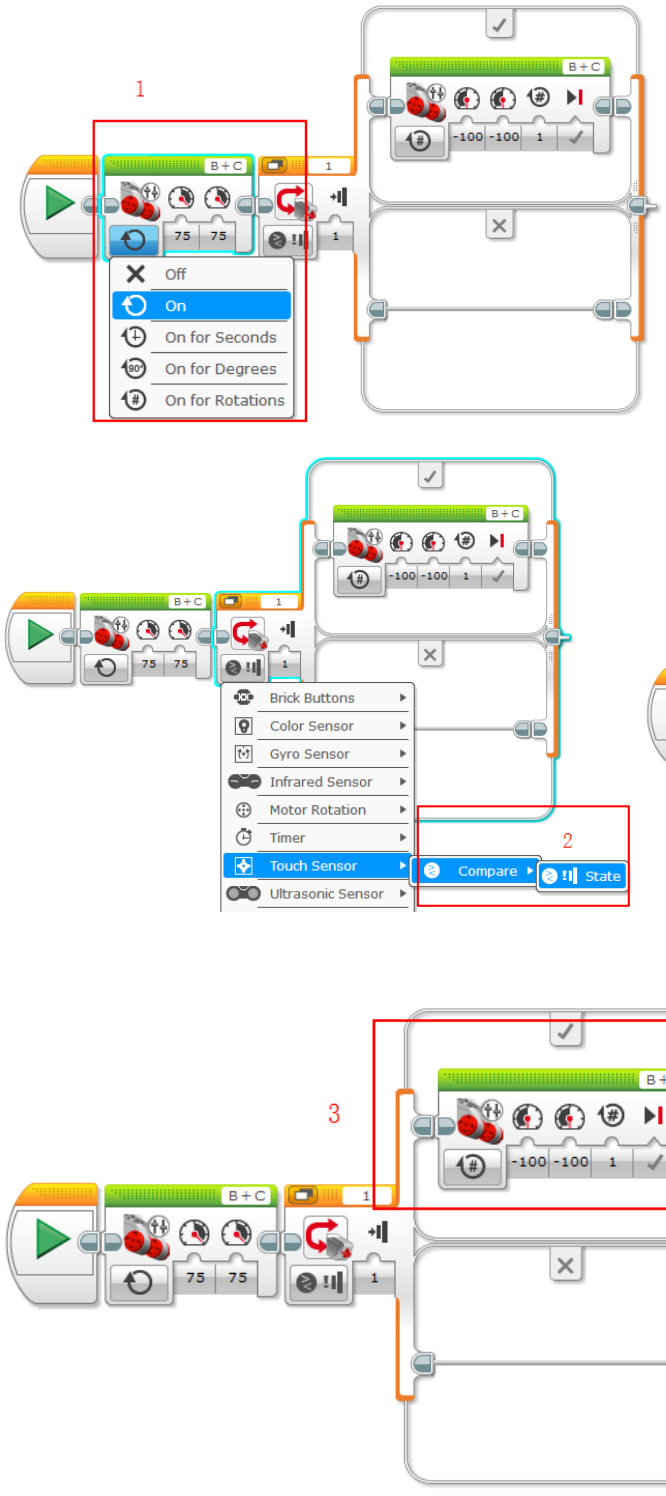
Use of Conditions

Conditional statements allow the robot to behave in response to conditions thus the robot can have more flexibility. Here we introduce the switch block, which is essentially an "if" statement.

This robot has a bumper in the front, we want our robot keep moving unless the bumper is being pressed down. As soon as the bumper is pressed the robot will go backwards.

1. Drop a move block with unlimited rotations
2. Drop a switch block thereafter, use "touch sensor being pressed" as the switch condition.
3. Drop a move block on the branch where the condition holds true, in other words, blocks on this branch will execute if "touch sensor being pressed" holds true.

EV3:

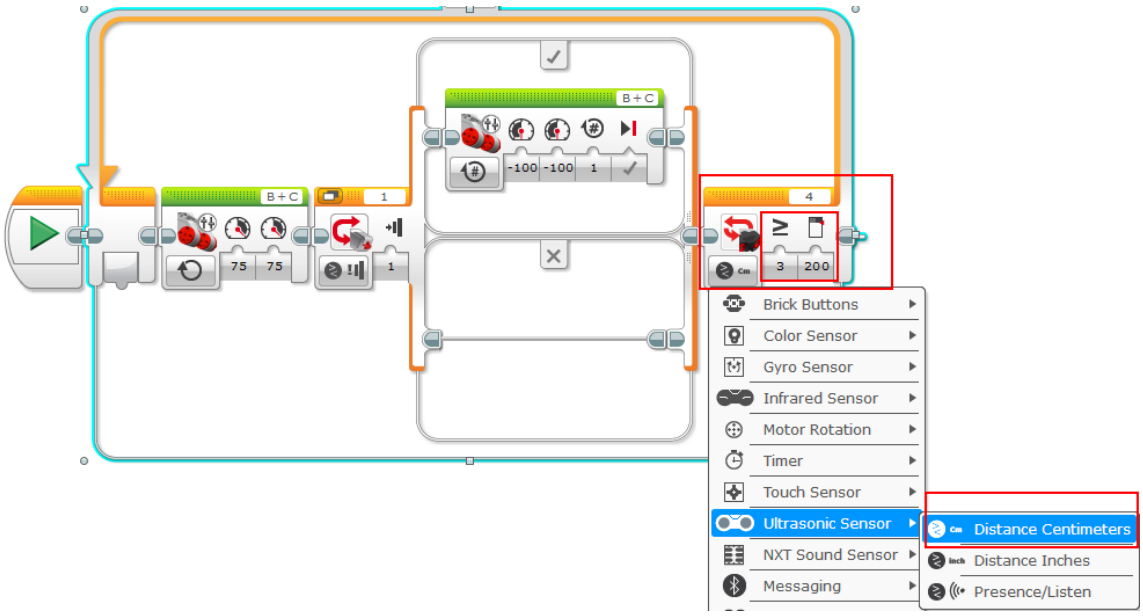

1
Off
On
On for Seconds
On for Degrees
On for Rotations
Brick Buttons
Color Sensor
Gyro Sensor
Infrared Sensor
Motor Rotation
Timer
Touch Sensor
Ultrasonic Sensor
Compare
State
2
Pressed
3

Use of Loops

In previous scenario, we use a switch block to control the behaviour of our robot, but that switch statement is only executed once, what if we wish the program keeps running on conditions and terminate once a certain condition holds true? We can use loop block.

Say we want our robot keep running the program as in Use of Conditions until the robot realise there is nothing ahead within 2 meters, here is how.

EV3:

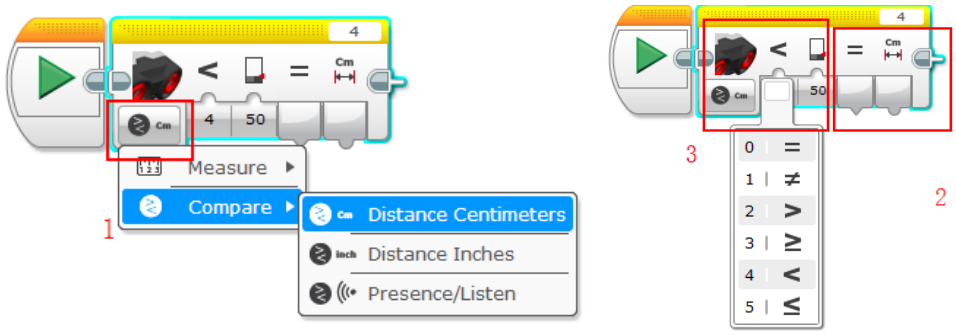


Reading from sensors (Ultrasonic Sensor)

Sensor readings are usually used in conjunction with switches. As shown in Use of Conditions, the program uses the touch sensor's states (pressed/not pressed) to determine whether to go backwards. In previous example the robot uses the readings from ultrasonic sensor to decide whether to continue the loop. Here you will go through the basics of reading from sensors.

1. Drag the ultrasonic sensor block to starting point, click on the the labelled button (EV3).
2. You will see there are two kinds of outputs, logical (Boolean) values and distance (numerical) values.
3. The Boolean output states whether or not the measured distances satisfy the comparison condition and the numerical outputs the measured distance.

EV3:



Use of Logical Operations

In the previous example you must noticed we have not yet use the outputs from the sensor. As mentioned usually two types of values are available from a sensor: logical values (Boolean) or numbers (numeric). You will learn how to use a logical operator in this example.

We want the robot to stop when it is close to something, hence we have two conditions as below, either of them being true will stop the robot:

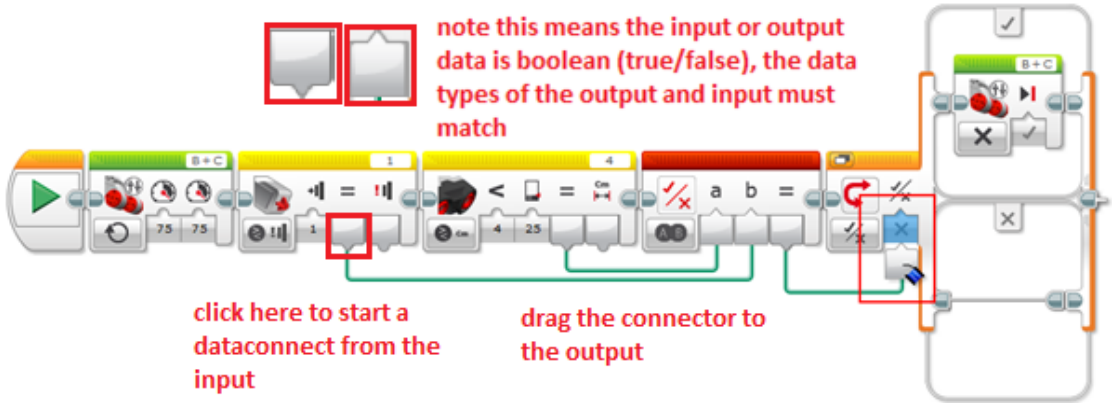
touching sensor being pressed

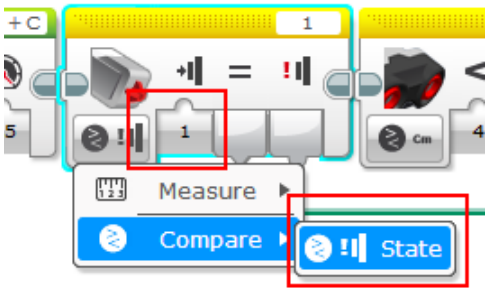
OR

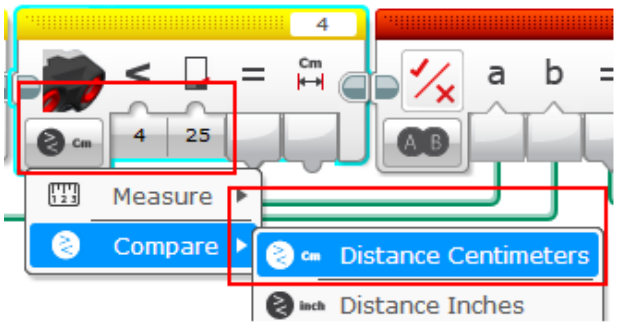
within 25 cm the ultrasonic sensor detected some objects ahead.

(see illustrations on the next page)

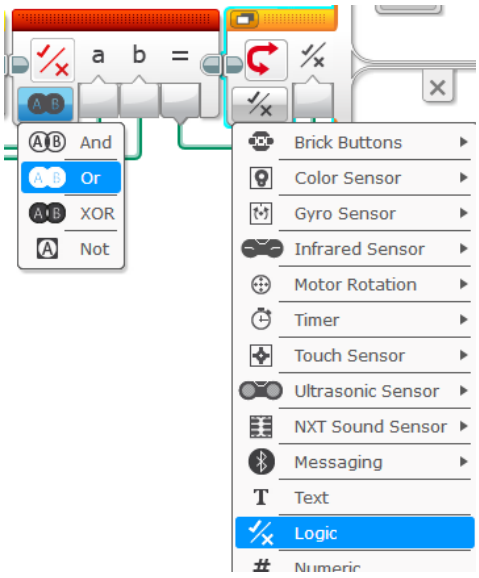
EV3:





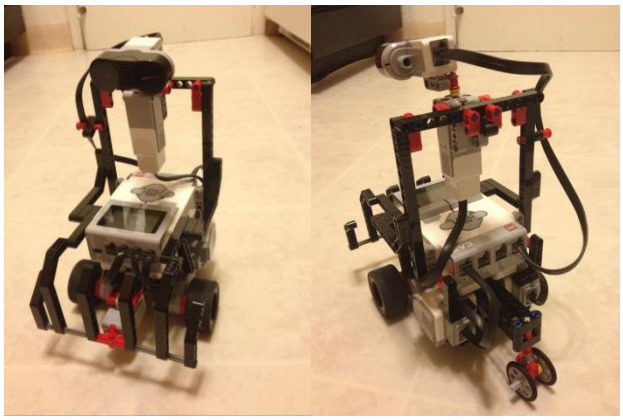


The sensor compares the measurement with 25; if the measurement is less than 25 the logical output will be true, otherwise the logical output will be false.



Use of Comparison operations (Radar)

In this example you will learn how to write a more sophisticated program: the core part of an ultrasonic radar.



First make sure you build a robot with a radar as mentioned in Assembly. Let's call the motor attached to the ultrasonic sensor "neck motor". We want this ultrasonic sensor to scan it's right and then left, determining which direction has more space, and then head to the direction with more space.

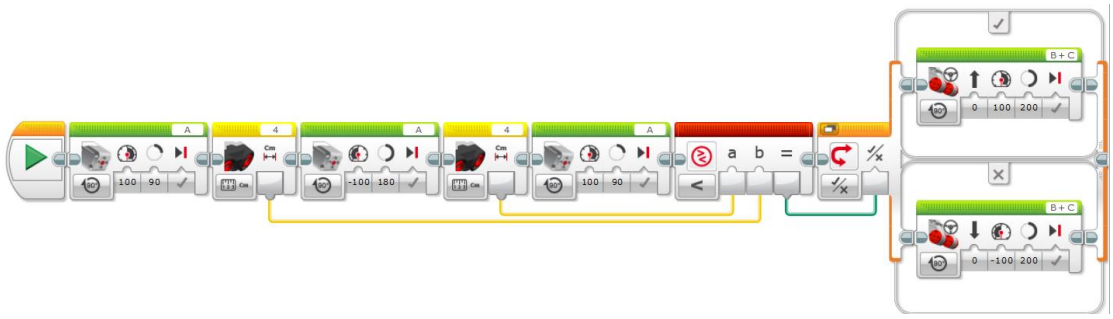
1. Turn the "neck" motor to the right to look right and take a reading from the ultrasonic sensor
2. Turn the "neck" motor to the left to look right and take another reading from the ultrasonic sensor
3. Return the neck to look straight ahead.
4. Use a Compare block to compare the two readings from the ultrasonic sensor to see which direction is farther.
5. Using the result of our Compare block on the two distances, if the right appears to have more space then turn right, otherwise turn left.

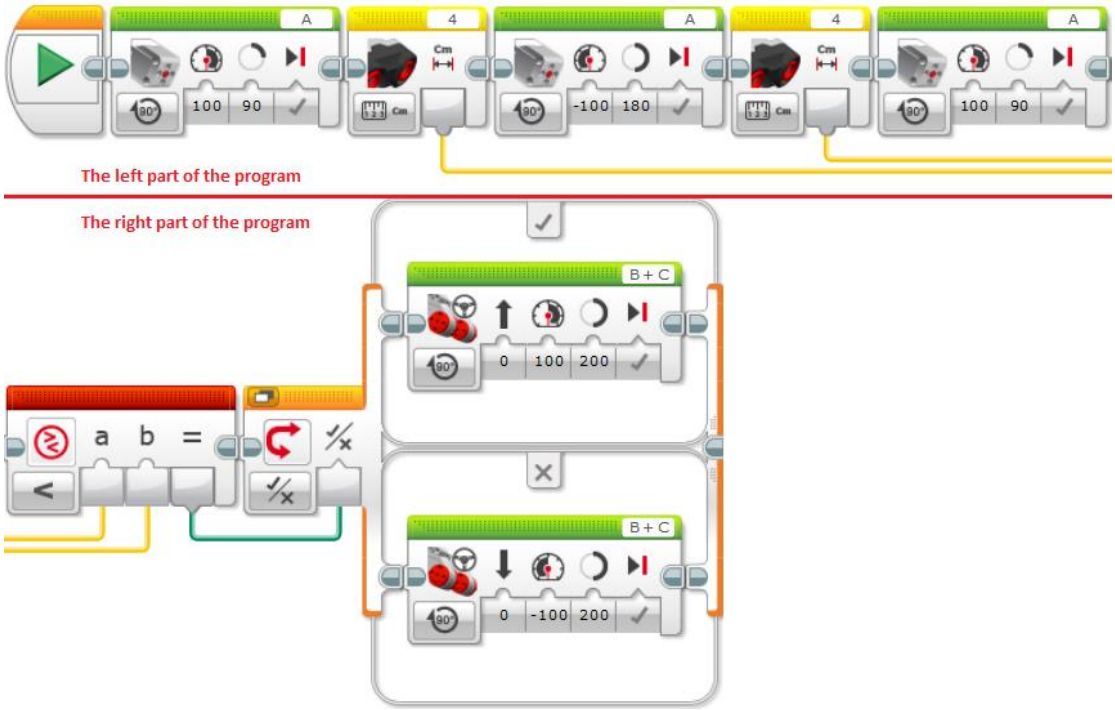
In this example we need to use the comparison operator to determine which measurement is greater.

(see illustrations on the next page)

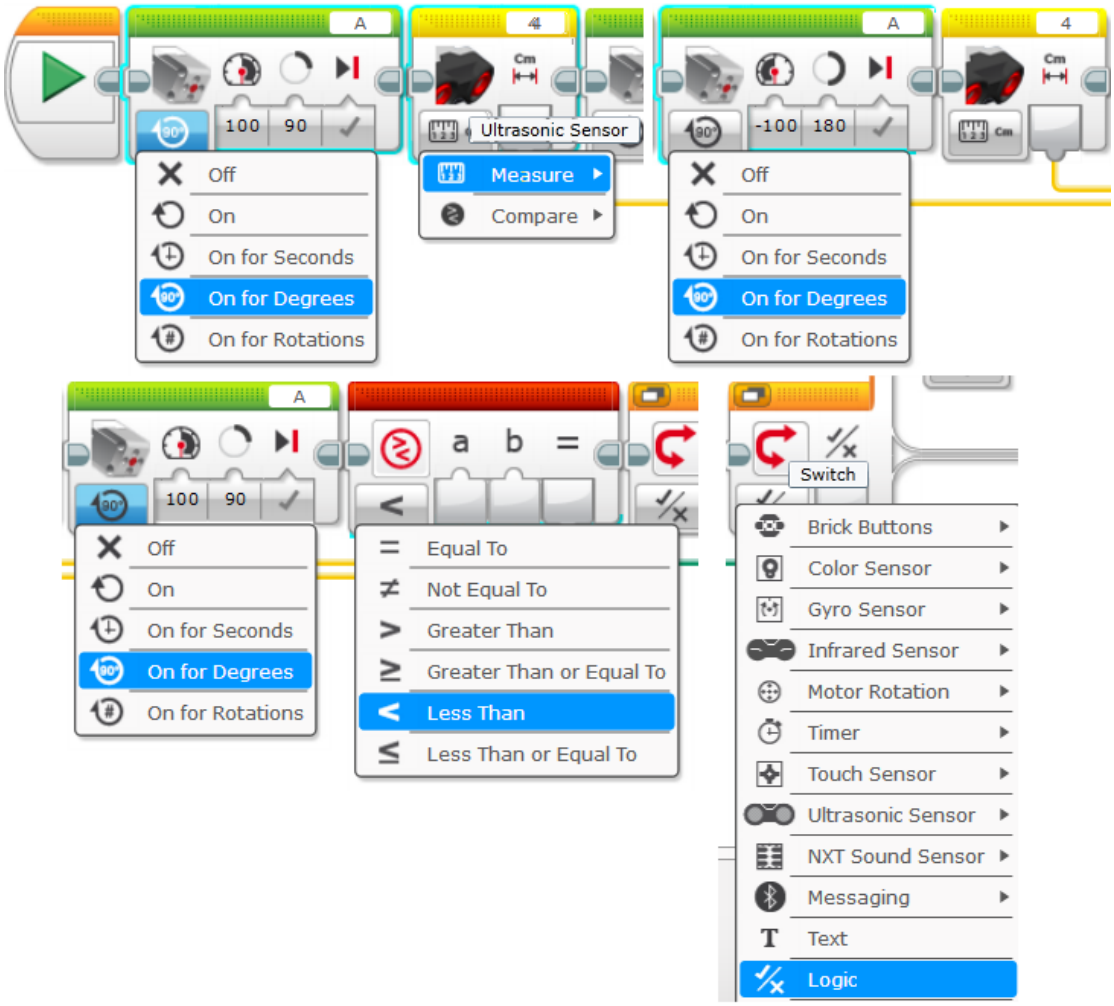
EV3:

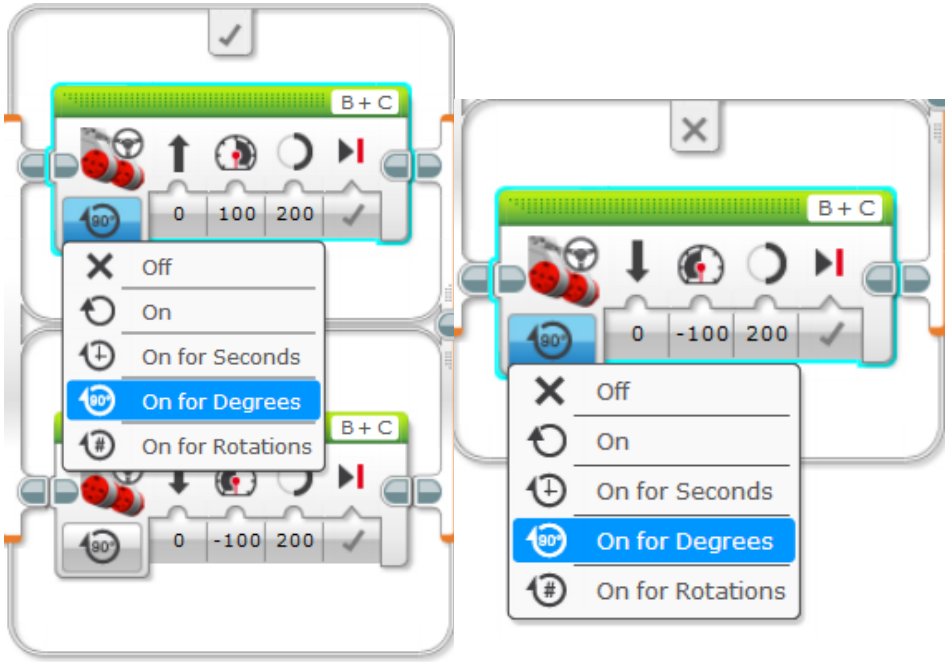
Note for EV3 you will need to use a medium motor as the neck motor.





(more on next page)


A
100
90
Off
On
On for Seconds
On for Degrees
On for Rotations
4
Ultrasonic Sensor
Measure
Compare
A
-100
180
Off
On
On for Seconds
On for Degrees
On for Rotations
4
A
100
90
Off
On
On for Seconds
On for Degrees
On for Rotations
a
b
=
Equal To
Not Equal To
Greater Than
Greater Than or Equal To
Less Than
Less Than or Equal To
Switch
Brick Buttons
Color Sensor
Gyro Sensor
Infrared Sensor
Motor Rotation
Timer
Touch Sensor
Ultrasonic Sensor
NXT Sound Sensor
Messaging
Text
Logic


B + C
0
100
200
Off
On
On for Seconds
On for Degrees
On for Rotations
B + C
0
-100
200
B + C
0
-100
200
Off
On
On for Seconds
On for Degrees
On for Rotations

Practice Project

Task Description

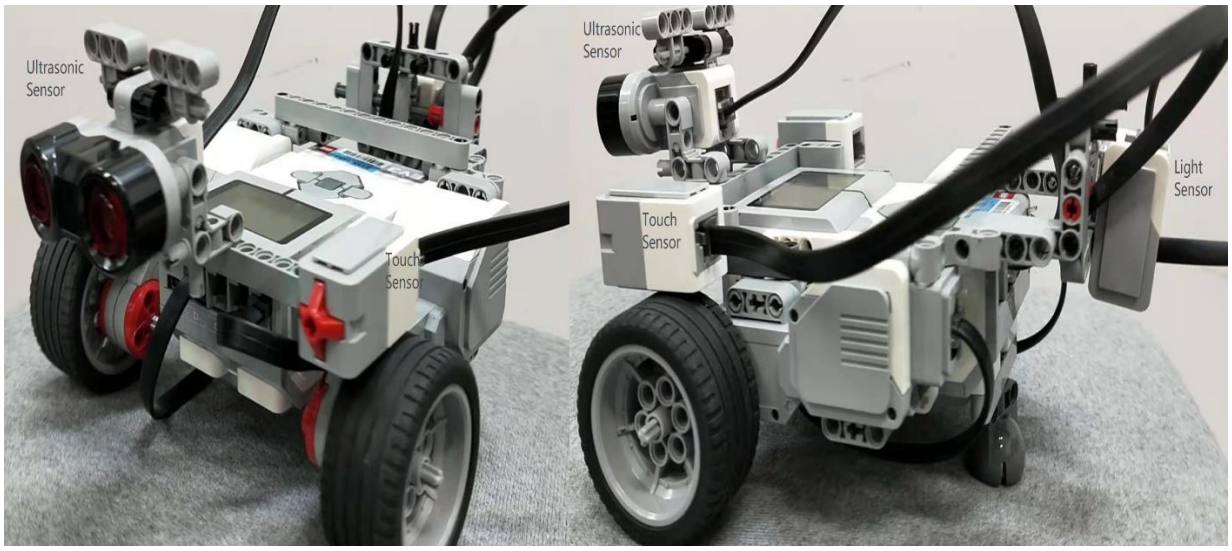
You are about to build a robot to finish the following task utilising ultrasonic, touch, light sensors, control flow statements and data links:

The robot moves forward to the obstacle in sight (<= 150cm) until it is 22cm away (twice as the length of a Mindstorms brick), at this point the robot should play a sound and then starts to move. If it bumps into anything, the robot makes a sound and then stop. Afterwards it takes a measurement of the light reflection readings on the ground and display it at the centre of the robot screen.

The above task seems to be a bit daunting at the first glance, but we can easily break it down as flowcharts and pseudocode, which you have learnt earlier.

You should have built a robot like below, but not necessarily the same, as long as it meets this specification: two motors, one on each side, an ultrasonic sensor mounted facing the front, a touch sensor with a bumper on the front, a colour or light sensor facing the ground (around 1cm above the ground).

Quickly build your robot and make sure it works; it does not have to be perfect.

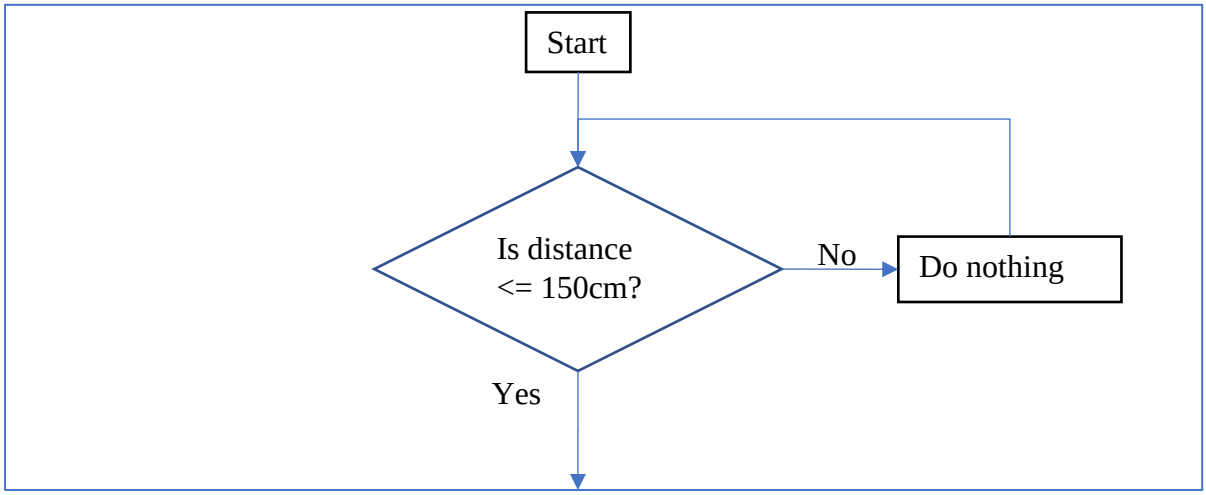


Problem breakdown: the first loop (distance <= 150cm)

The robot moves forward to the obstacle in sight (<= 150cm) until it is 22cm away (twice as the length of a Mindstorm brick), at this point the robot should make a sound…

As can be seen, the distance measurement needs to be checked twice (<=150cm and <=22cm). Since the robot goes to anything in sight (<=150cm), this condition should be checked first, if true it proceeds to the next check; if false the robot checks again.

Flowchart

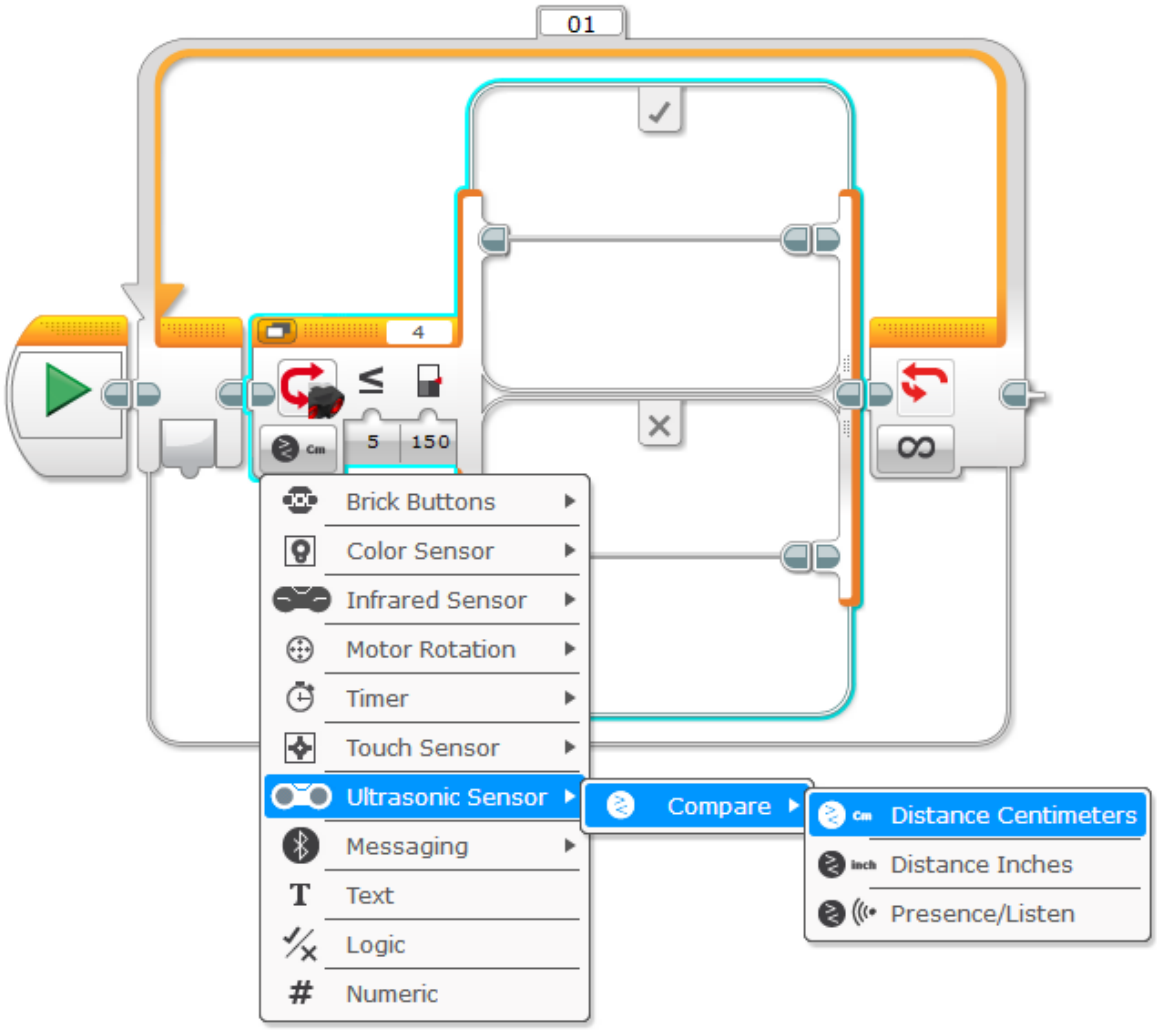


This check forms a loop. Therefore, you should use a loop block and a switch block to programme this behaviour.

Pseudocode

```
Loop forever {
	If distance <= 150cm {
		Do something
	} else {
		Do nothing
	}
}
```

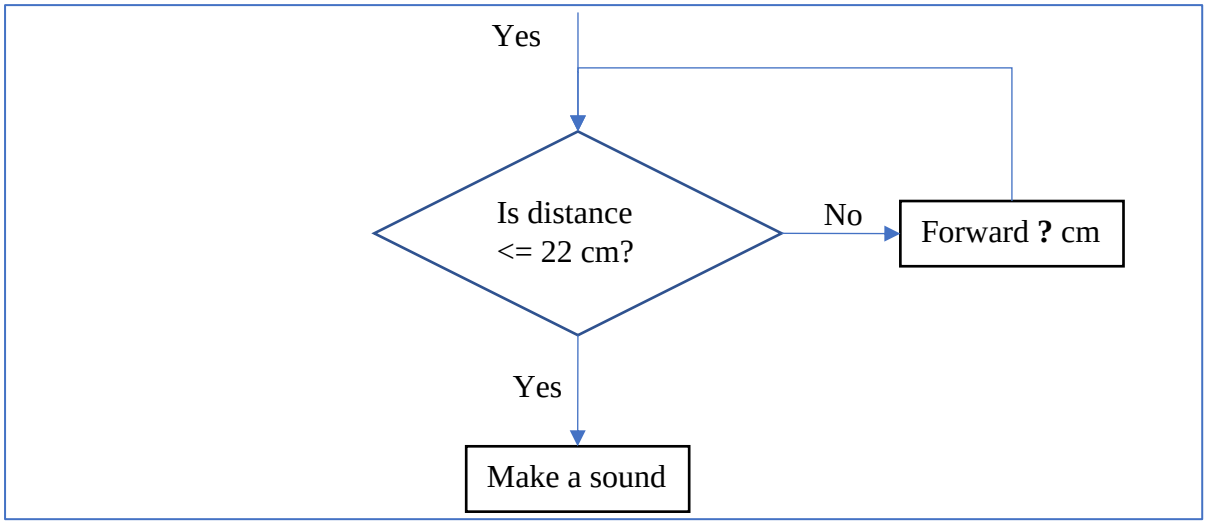
Code


01
4
5
150
Brick Buttons
Color Sensor
Infrared Sensor
Motor Rotation
Timer
Touch Sensor
Ultrasonic Sensor
Messaging
Text
Logic
Numeric
Compare
Distance Centimeters
Distance Inches
Presence/Listen

Problem breakdown: the second loop (distance <=22cm)

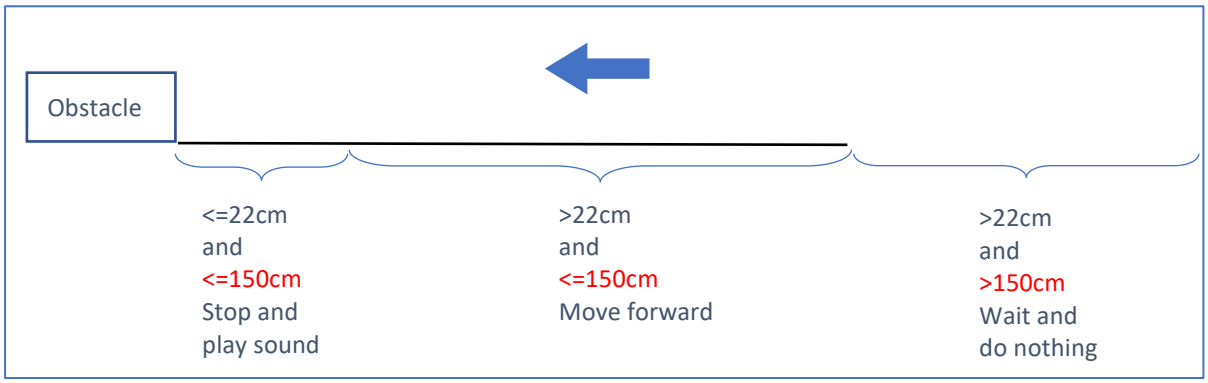
Think before programming

The second check (<=22cm) is similar. If the conditional is true, then the robot plays a sound; if not then the robot will move forward a bit and checks again. Therefore, it is another loop.



Note a question mark (**?**) is in the flowchart. It is the parameter you need to determine by yourself. You should do some experiments to find out the best value for the parameter.

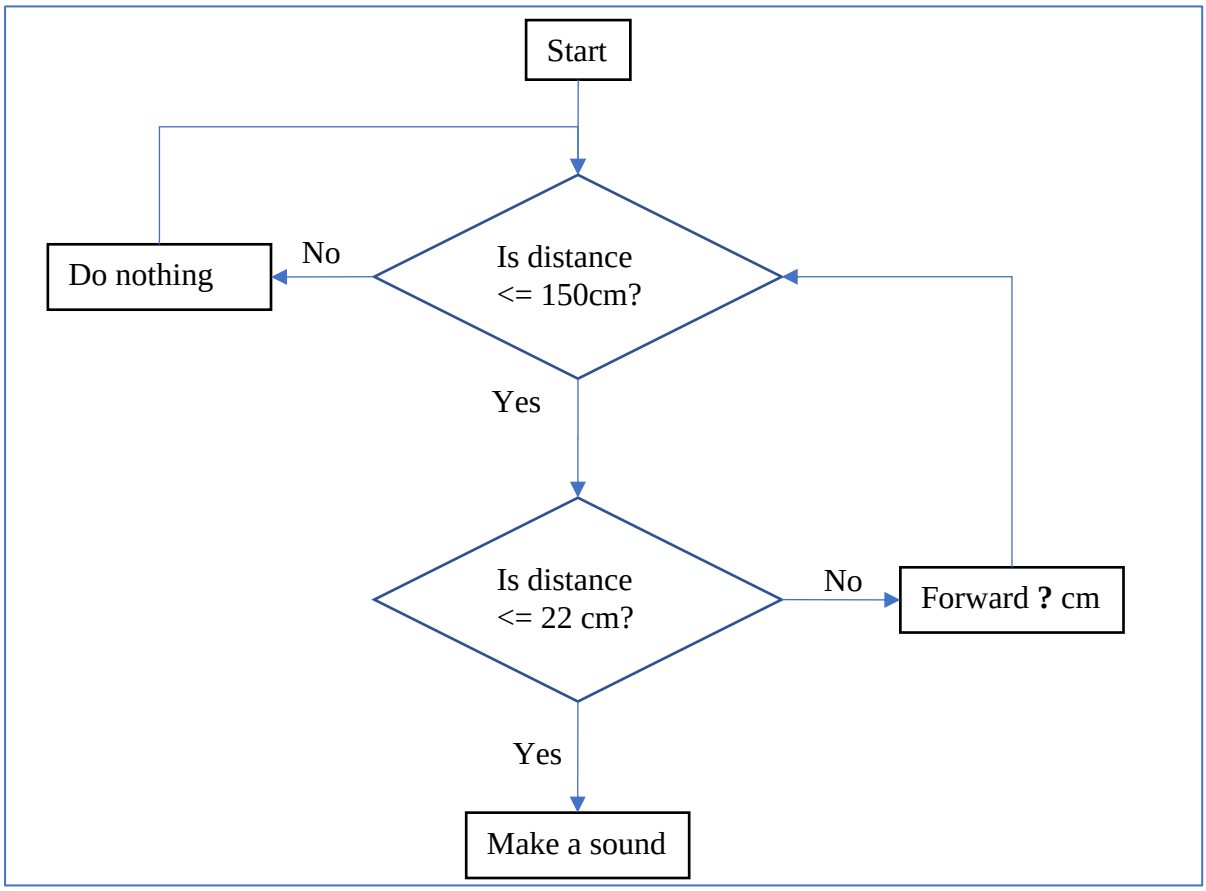
You can programme this loop straight away. But before doing programming, **give this flow chart a second thought**. Intuitively, below is what we want:



If programmed as in the above flowchart, we will not check “Is distance <= 150 cm” before moving forward. In this simple scenario it should work fine, but as your program becomes more complex you should know what exactly you are programming, otherwise these overlooked details will become a software bug that you may never catch.

Hence, we have a more well-thought flowchart.

Flowchart



Pseudocode

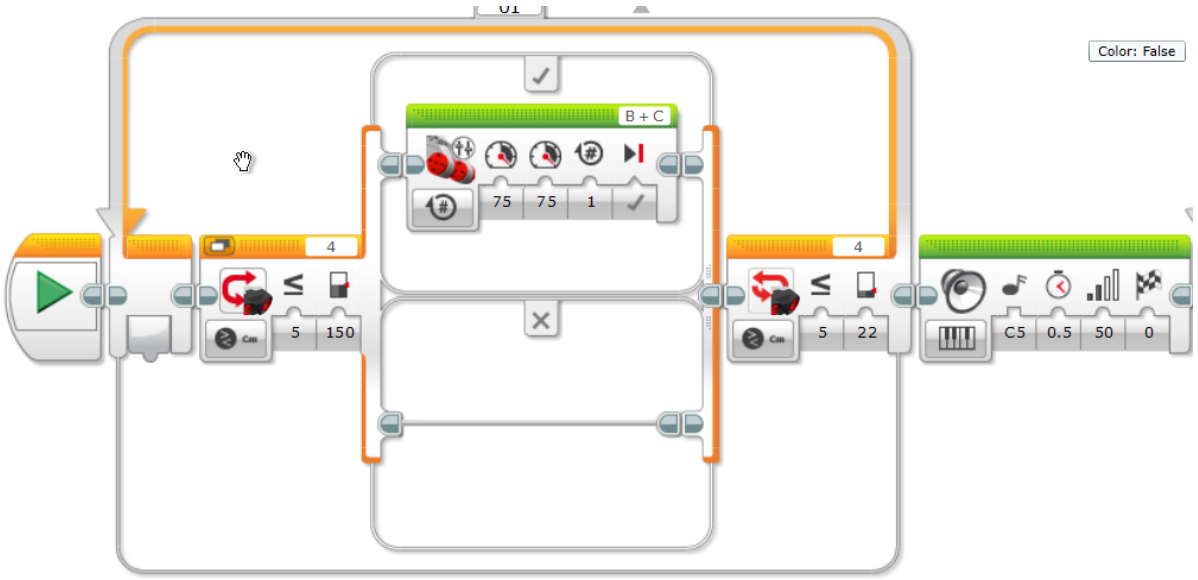
```
Loop until distance <= 22 {
	If distance <= 150cm {
		Forward ? cm
	} else {
		Do nothing
	}
}
Play a sound
```

Q: Notice any difference in the pseudocode? Think why it becomes this way?

A: There is "`Loop until distance <= 22`", this statement is supported as a single block in Mindstorms programming software.

Also, the two loops are merged as one, this simplification works since the measured distance cannot be <= 22 cm and > 150 cm at the same time. We don't have to consider the case where it holds true. Therefore, our current pseudocode covers all possible outcomes of distance measurement comparisons.

Code

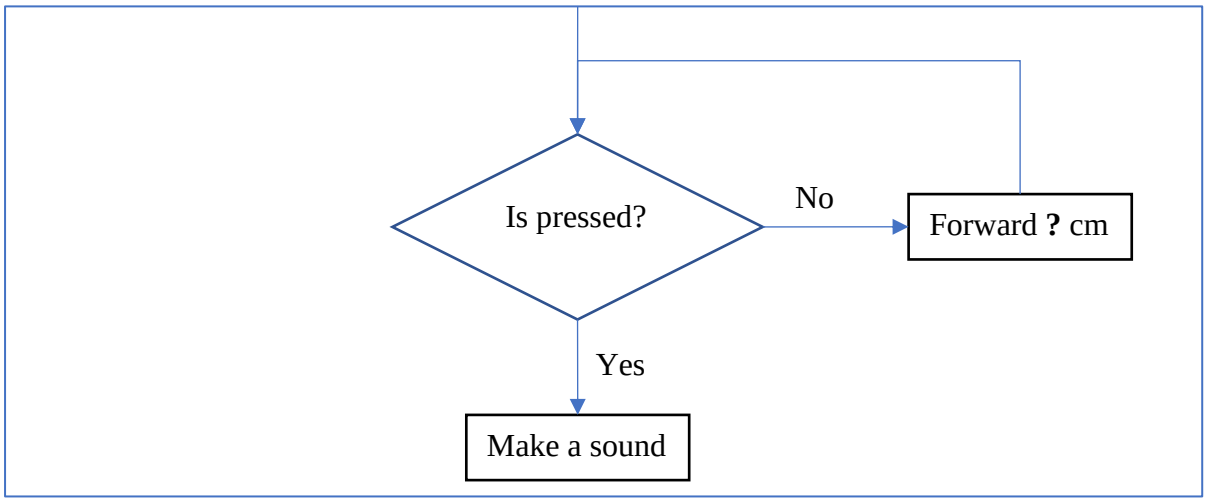


Problem breakdown: the third loop (touch sensor)

… and then starts to move. If it bumps into anything, the robot makes a sound and then stop.

Having completed the two loops earlier, this one is trivial. The switch block is now controlled by the touch sensor instead of the ultrasonic sensor.

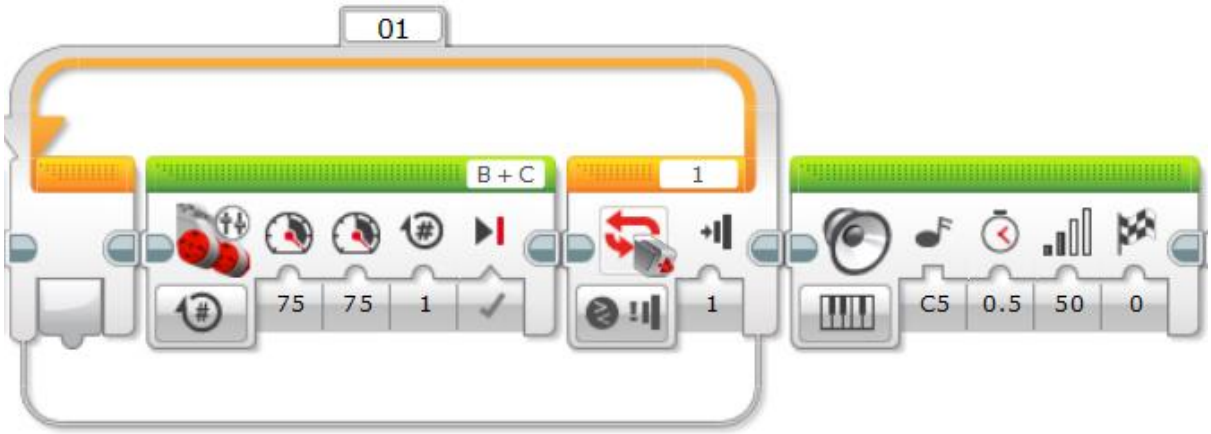
Flowchart



Pseudocode

```
Loop until touch sensor is pressed {
      Forward ? cm
}
Play a sound
```

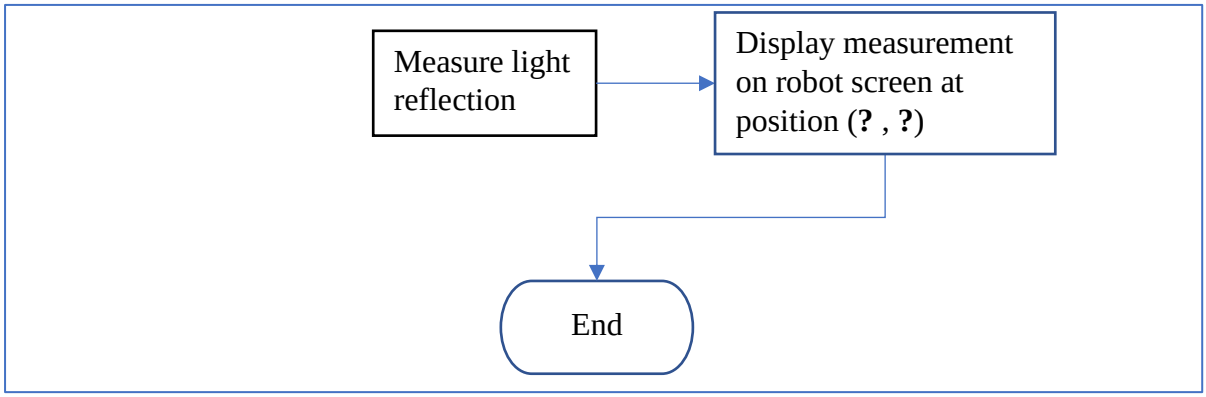
Code



Problem breakdown: display measurements

The last step is straightforward. The key here is to make use of the data link in Mindstorms programming software. If you are an NXT user, data conversion block is needed to convert numbers into text for displaying. Conversion is not required for EV3 models.

Flowchart



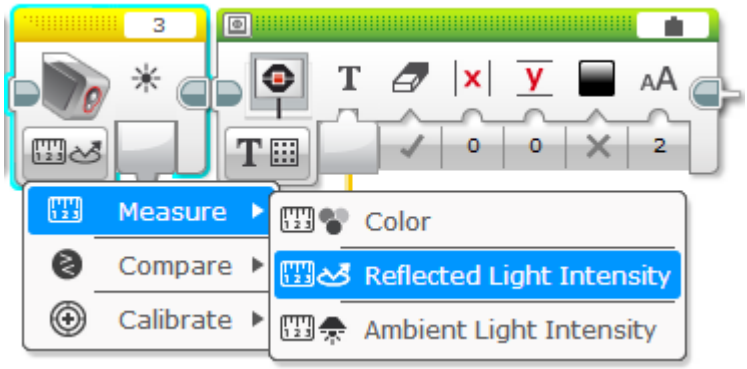
Pseudocode

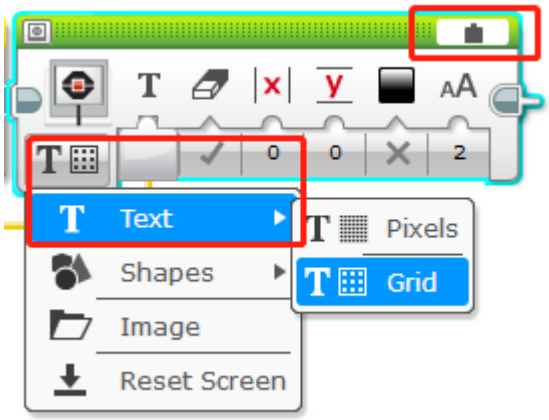
```
Measure light reflection

Display measurement on robot screen at position (? , ?)
```

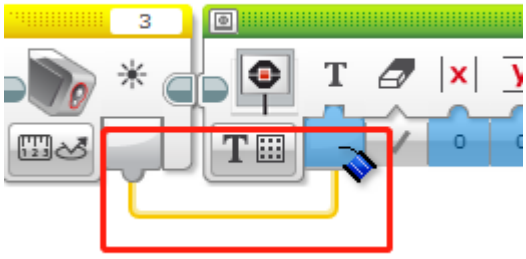
Q. When you run this program on your robot, can you read the numbers on the display? Do they stay there or just flashed out? What can you do to make it better?

A. Hint: try to use a loop

Code

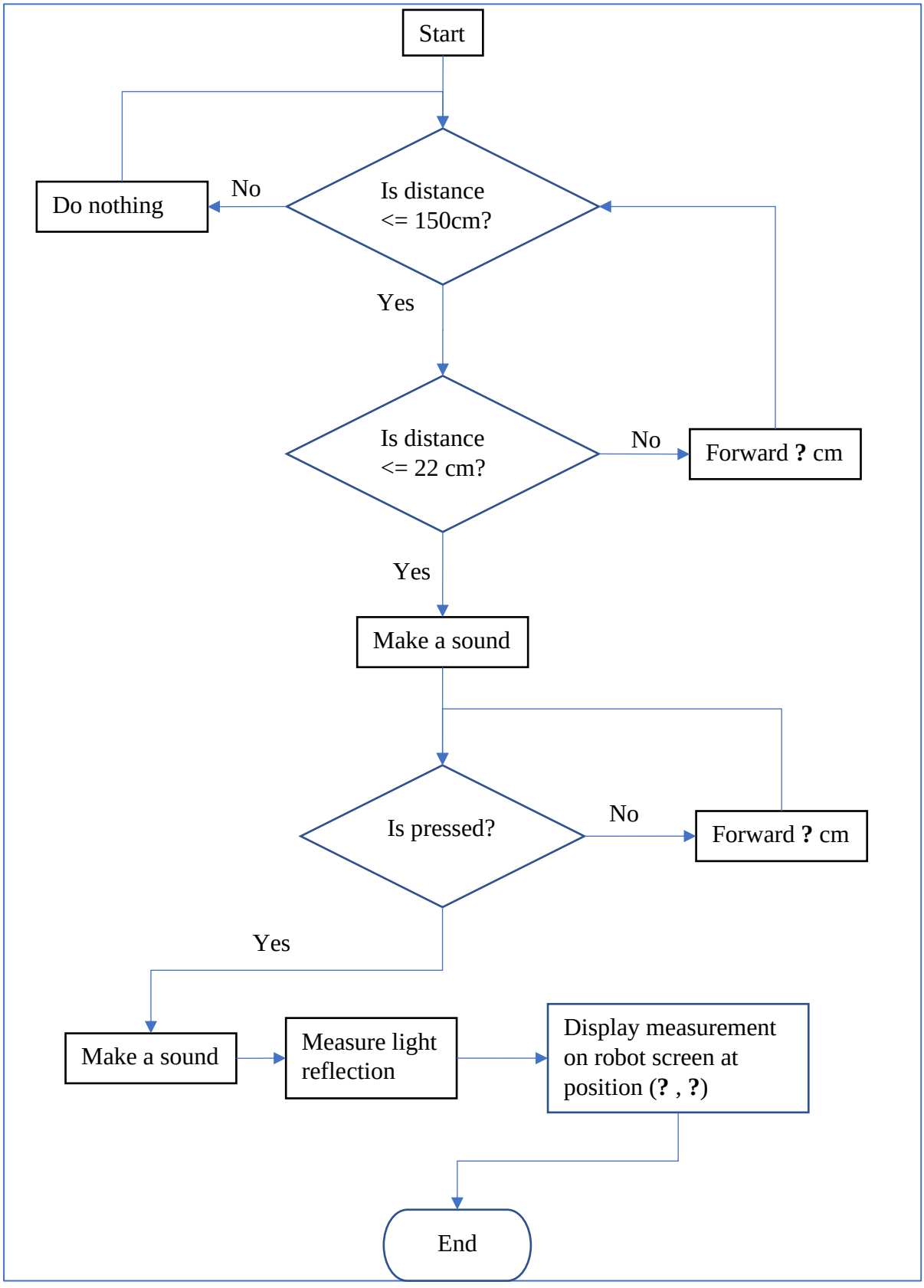

3
0
0
2
Measure
Compare
Calibrate
Color
Reflected Light Intensity
Ambient Light Intensity


0
0
2
Text
Shapes
Image
Reset Screen
Pixels
Grid


3
0

Summary

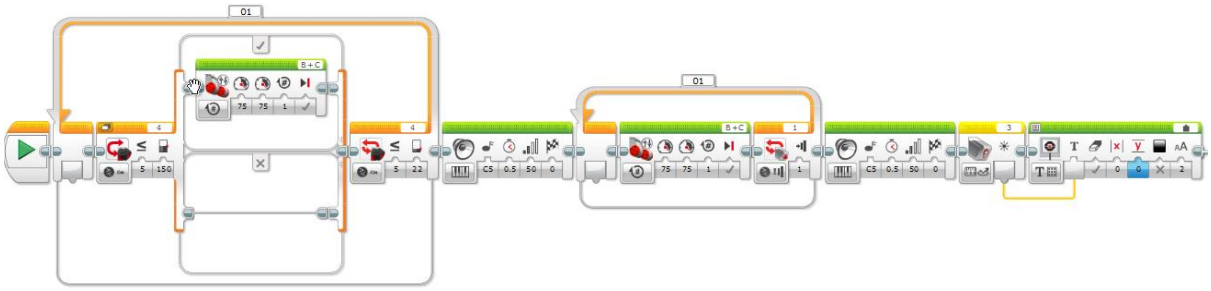
Flowchart



Pseudocode

```
Loop until distance <= 22 {
	If distance <= 150cm {
		Forward ? cm
	} else {
		Do nothing
	}
}
Play a sound
Loop until touch sensor is pressed {
	Forward ? cm
}
Play a sound
Measure light reflection
Convert numbers into text if necessary
Display measurement on robot screen at position (? , ?)
```

Code



Introduction to LEGO Mindstorms connection to Scratch

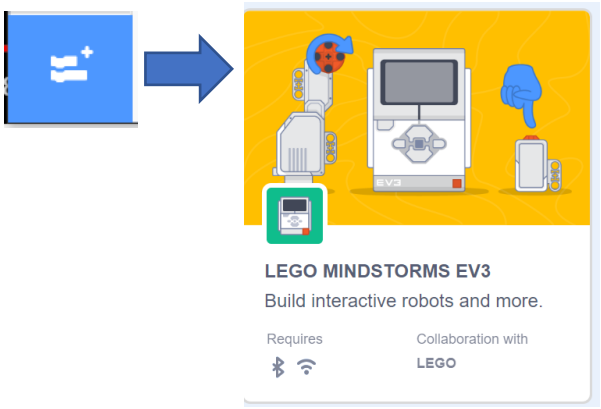
This section introduces how to connect the EV3 to the Scratch; you will learn how to connect some basic troubleshooting, and some basic functions that can be achieved, by associating both components.

Connection

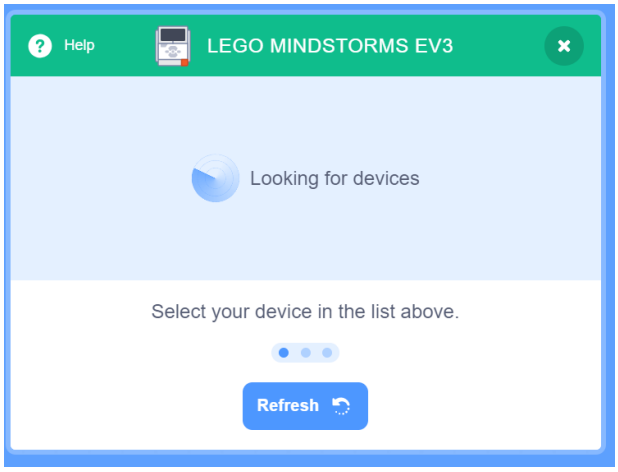
First turn on you brick and go to **Settings** and activate the **Bluetooth** so the brick can be discovered by Scratch.

Then verify if the **Scratch Link** is activated and if so turn on the **Scratch Desktop**.

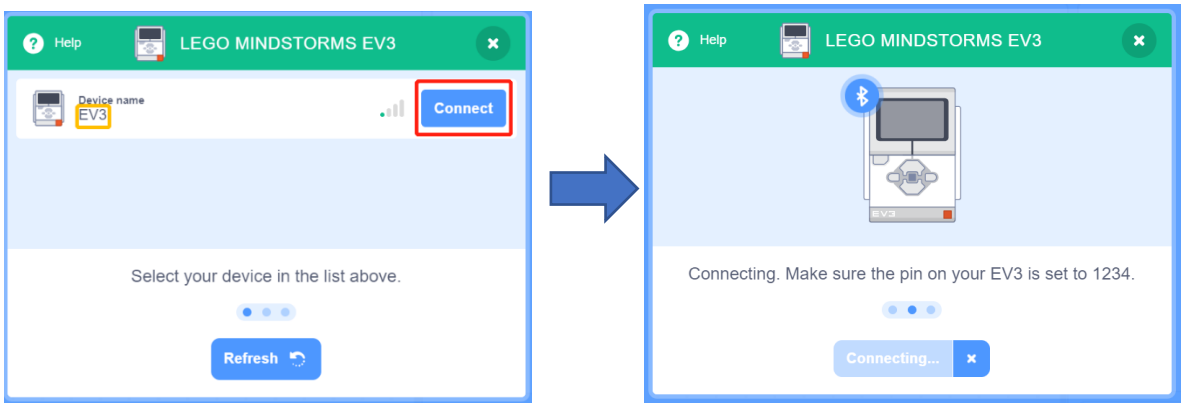
After **Scratch Desktop** starts click on **Add Extension** and select **LEGO MINDSTORMS EV3**



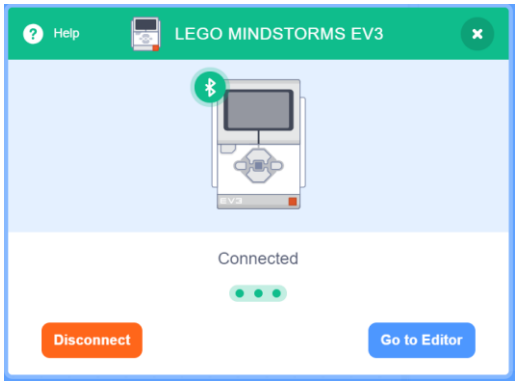
Scratch will then start looking for an EV3 to connect to.



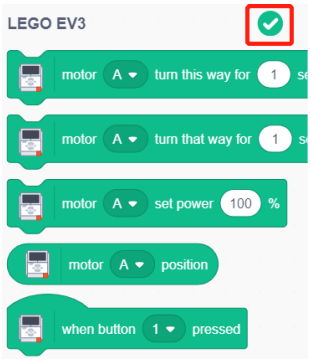
If everything is alright your Brick will be found. Then click on connect. Make sure that in the brick the EV3 bluetooth pin is set to 1234.



If there is no problem the connection will be confirmed.



On the upper right corner of the Scratch blocks area, we can see the green symbol indicating that there is an EV3 connected to this Scratch instance, only one EV3 can be connected to each Scratch instance at a time.

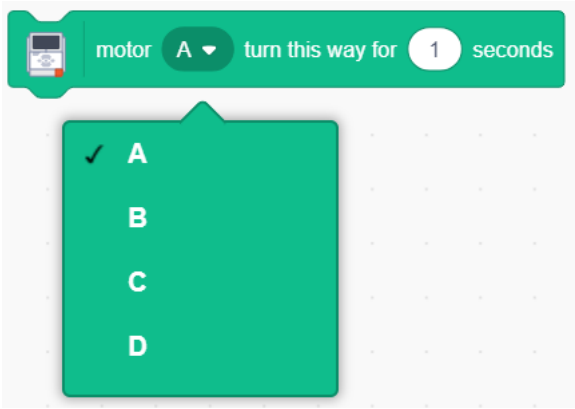


Blocks

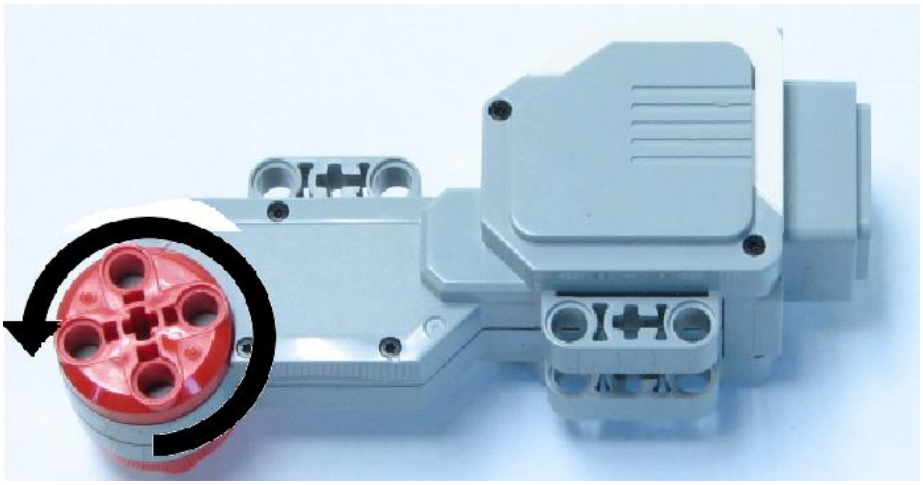
The extension contains 11 blocks. 4 motor blocks, 6 sensor blocks and 1 sound emitter block.

Motor Blocks

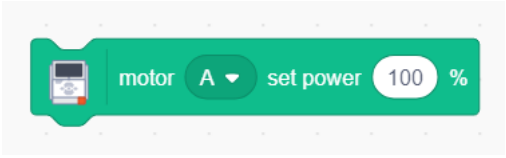
On the motor blocks, we can choose which motor is going to turn and which direction.



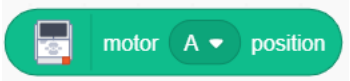
This way turns the motor counterclockwise on the position of the picture. **That way** turns the motor the other way. For a set number of seconds.



The motor **set power** allows changing the how powerful the motor spins when activated.

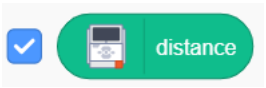


Motor position informs where the motor wheel is currently positioned.

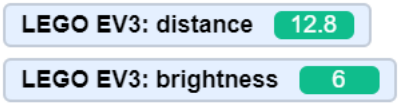


Input

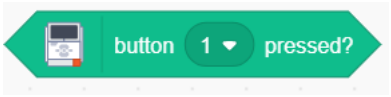
Distance shows the distance in inches calculated by the ultrasound. The current version can not communicate when the robot has the distance set for centimeters.



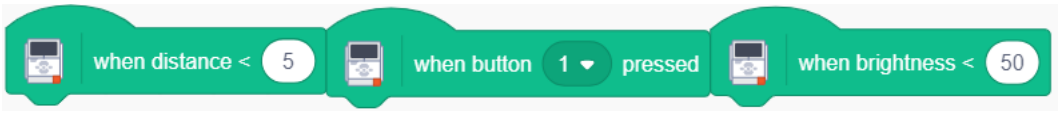
Brightness shows the brightness when the robot light sensor is set to **col-ambient.**



Button pressed detects if the touch sensor on a specific port has been pressed.

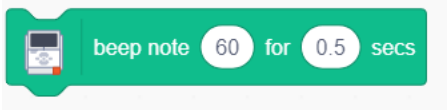


The **when** blocks detect the information shown on the above mentioned blocks, they work as an initial block.

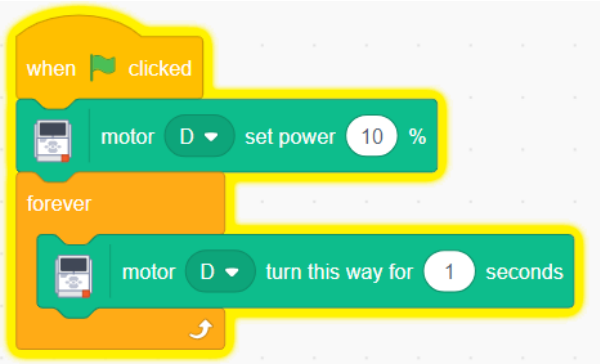


Sound Emitter

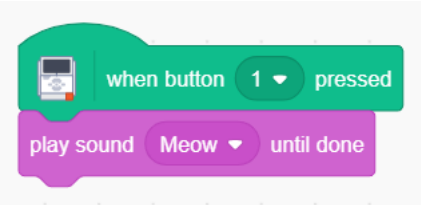
Beep node makes a beeping sound for a specific amount of time.



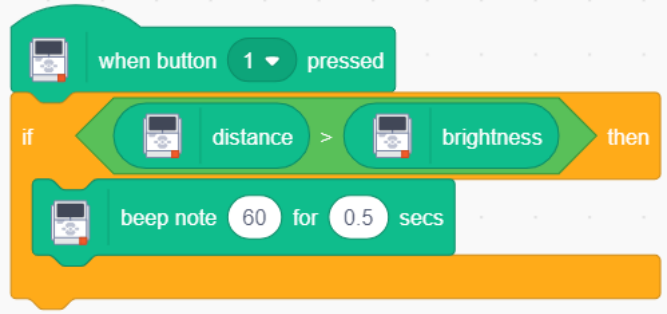
Code Sample



The first example shows the power of motor D being set to 10% and within an endless loop the motor keeps turning repeatedly for a second.



The second example makes the scratch character Meow when the touch sensor on port 1 is pressed.



The third example waits for the touch sensor on port 1 to be pressed. Then if the distance is greater than the brightness level perceived by the sensors a beep will be emitted for half a second.

The fourth example waits for the brightness to be lower than 50. Then it changes the size of the scratch character until the touch sensor is pressed.