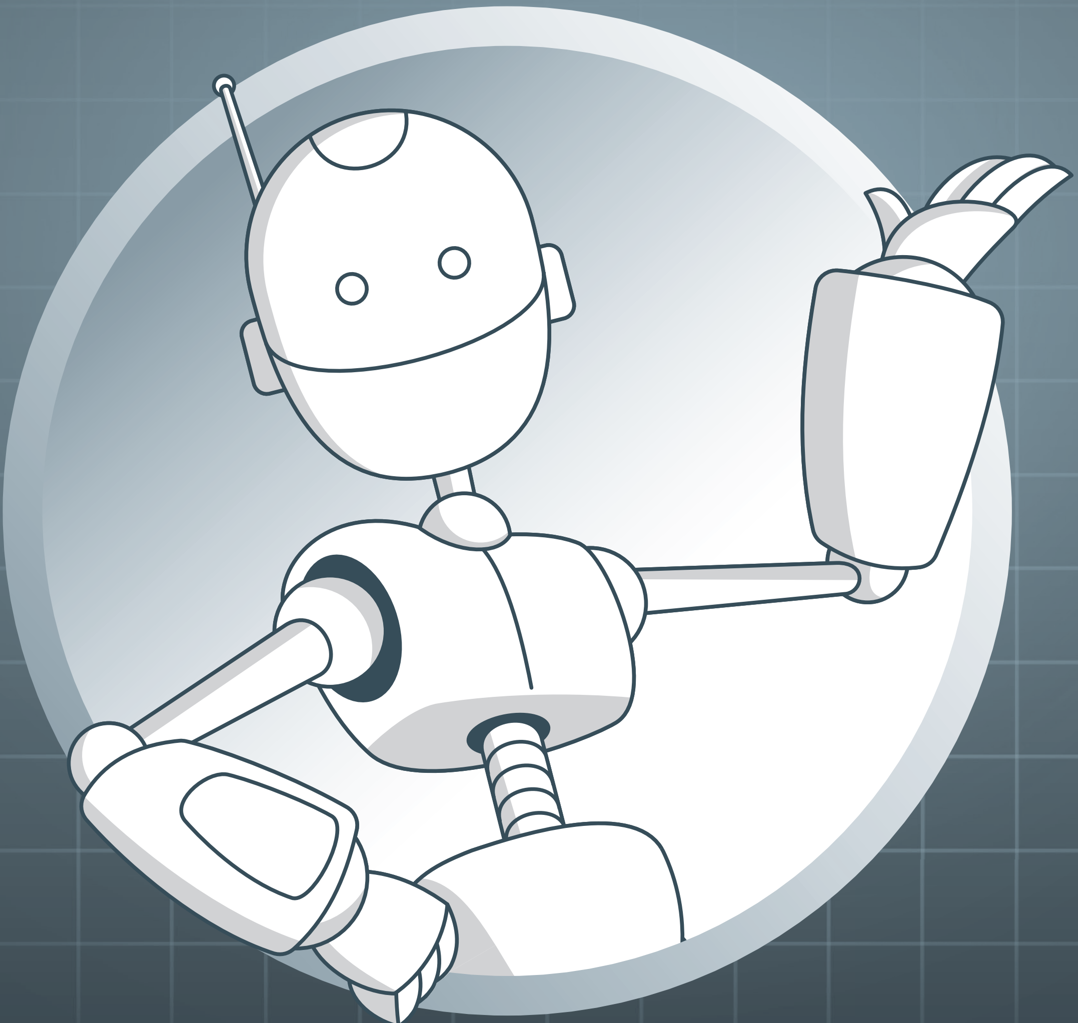
DIGITAL Teens ROBOTICS with Mindstorms EV3

2



binarylogic

TASK 1

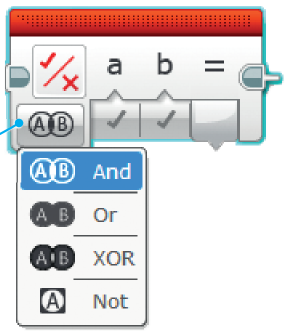
Logic operators

In a conditional statement, we use relational operators (for example > or =) to check if a condition is TRUE or FALSE. However, if we need to check for a condition that is more complex, we can use **logical operators**. They are also called boolean operators because there can only be one of two possible answers in any given case - TRUE or FALSE.

Logic Operations block

In Mindstorms EV3 programming, you can use the **Logic Operations** block if you want to perform a Logic operation. The Logic Operation takes inputs that are True or False, and produces a True/False output. The Logic operations available are AND, OR, XOR, and NOT and the block will calculate the output result based on the inputs.

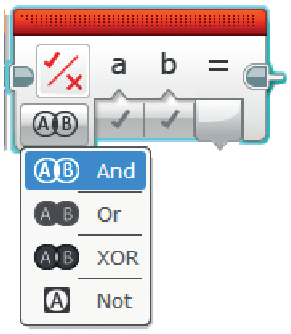
In the Mode selector of the Logic operation's block, you can select the mode you want.



Mode	Block's calculation Result based on the Inputs
And	True if both Input A and Input B are True, otherwise False
Or	True if either Input A or Input B (or both are) is True, False if both Input A and Input B are False
XOR	True if exactly one of Input A and Input B is True, False if both Input A and Input B are True, False if both Input A and Input B are False
Not	True if Input A is False, False if Input A is True

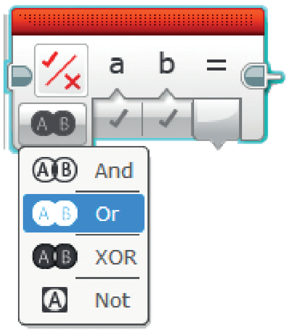
AND

The logical operator **AND** allows the user to connect two conditions. If the conditional expressions are both true, it returns true. If they are not both true, it returns false.



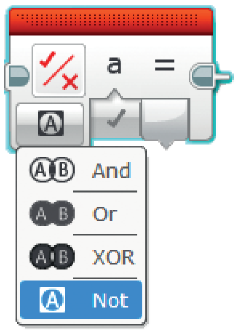
OR

The **OR** operator is used to connect two or more conditions. If at least one of them is true, it returns true. If none of them are true, it returns false.

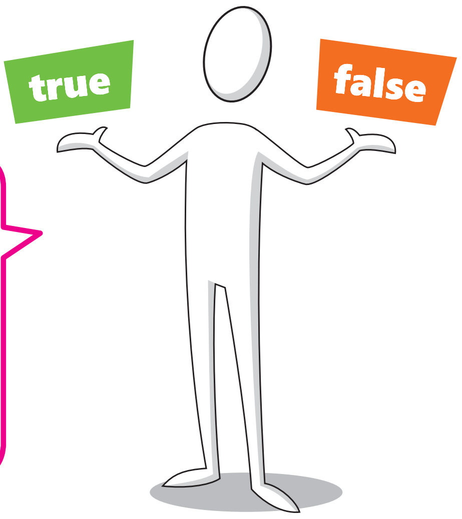


NOT

The logical operator **NOT** allows the user to express conditions that are best expressed in a negative way. If the conditional expression is false, only then does it return true. If the conditional expression is true, it returns false.

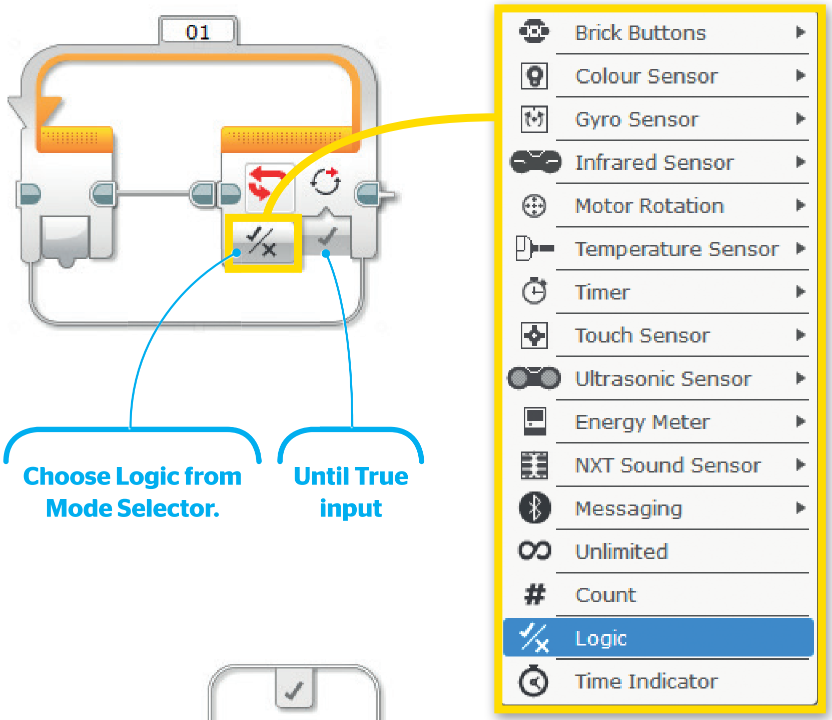


Robots can make decisions only when it is very clear which choices they have and under what circumstances. For this purpose, their decisions must be based on answers to questions that have two possible and simple answers: true or false (which means yes or no). Statements like these are called Boolean statements.



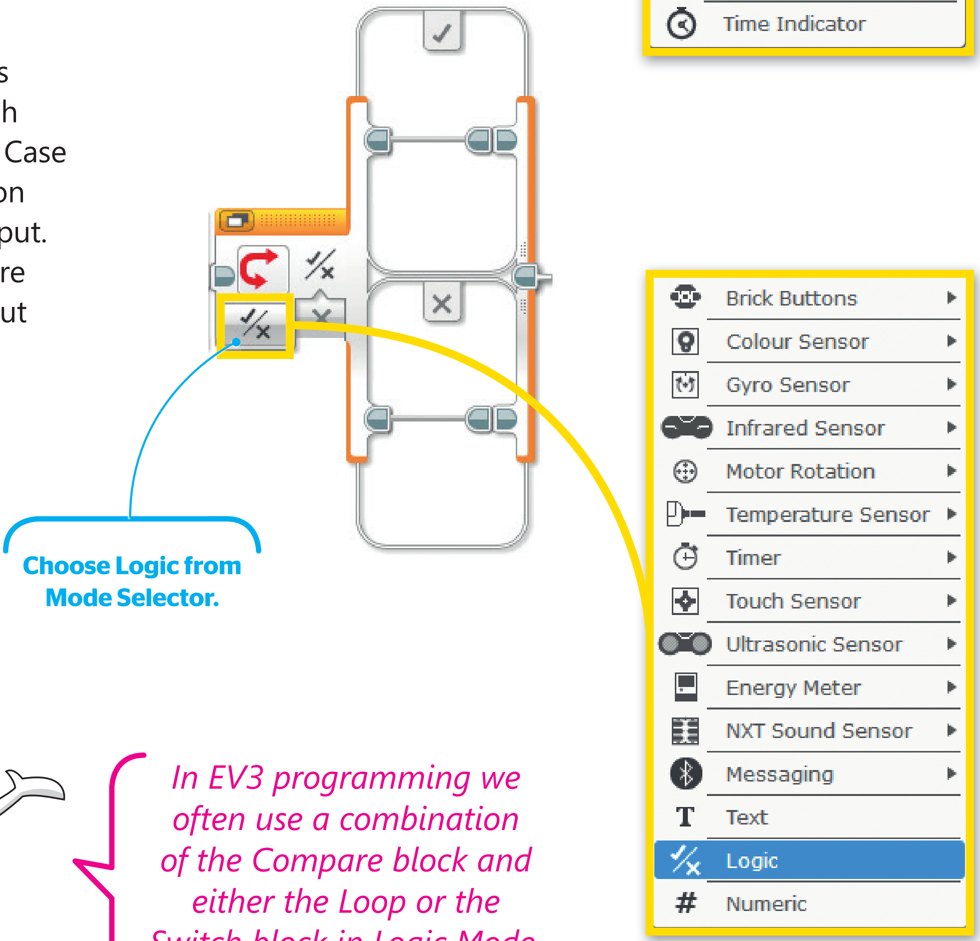
Loop block

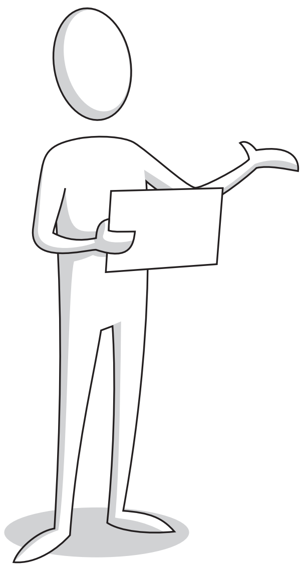
When the Loop block is in Logic mode, the loop will repeat until the Until True input is True at the end of the loop sequence. You can use the Data Wire to connect the Until True input to a Logic output of a programming block that is inside the loop.



Switch block

When the Switch block is in Logic mode, the Switch chooses between a True Case and a False Case based on the value of the Logic input. You can use the Data Wire to connect the Logic input to a Logic output of any programming block.

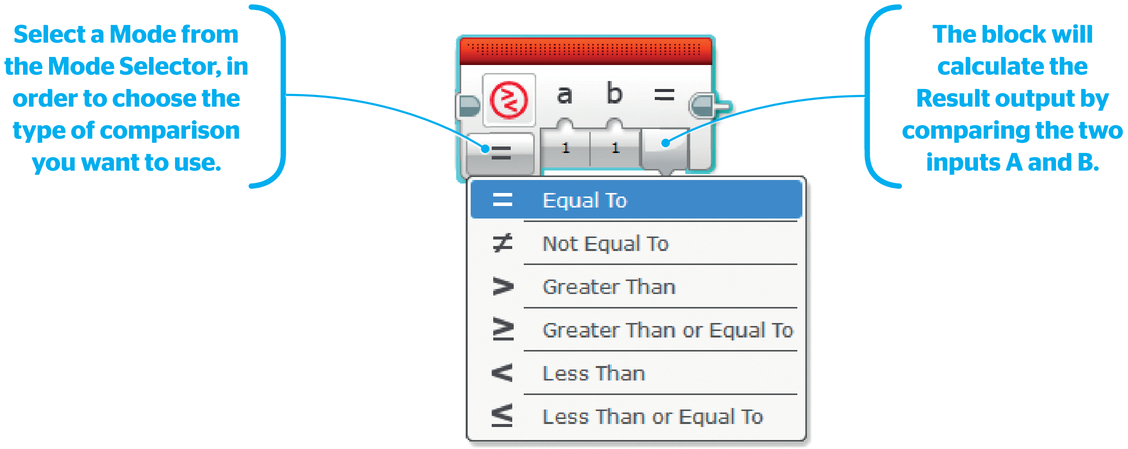




In EV3 programming we often use a combination of the Compare block and either the Loop or the Switch block in Logic Mode. The output value of the Compare block is used as an input value on the Loop or the Switch block.

Compare block

In Mindstorms EV3 programming, we often use comparisons between numbers in order to program our robot to do specific actions depending on whether the result of the comparison is True or False (Logic output). For this purpose, we can use the **Compare block**. The Compare block compares two numbers to find out whether they are equal/not equal, which number is greater than/less than or which number is greater than or equal to/less than or equal to. You can choose one of these six Modes and receive a comparison result of True or False.

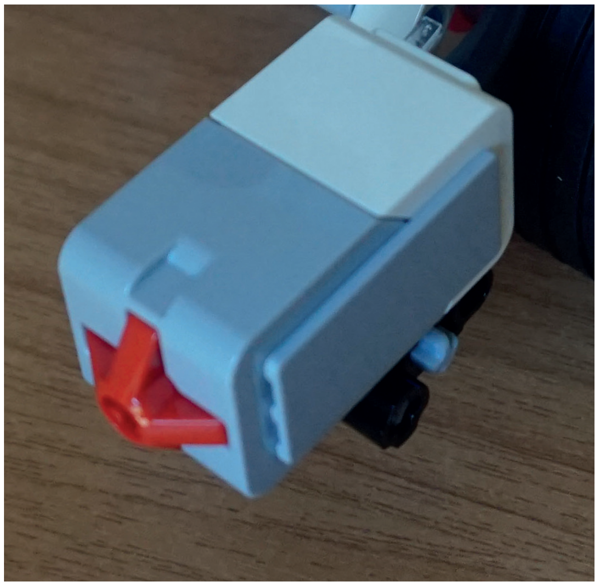

Select a Mode from the Mode Selector, in order to choose the type of comparison you want to use.
The block will calculate the Result output by comparing the two inputs A and B.
a
b
=
1
1
= Equal To
≠ Not Equal To
> Greater Than
≥ Greater Than or Equal To
< Less Than
≤ Less Than or Equal To

Compare block

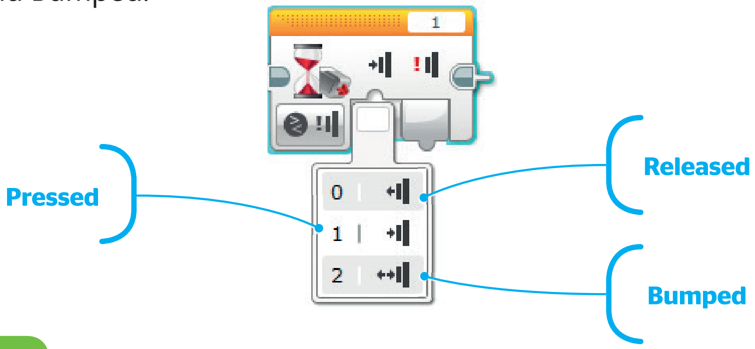
Mode		Inputs	Logic output
=	Equal To	A, B	True if A = B, otherwise False
≠	Not Equal To	A, B	True if A ≠ B, otherwise False
>	Greater Than	A, B	True if A > B, otherwise False
≥	Greater Than or Equal To	A, B	True if A ≥ B, otherwise False
<	Less Than	A, B	True if A < B, otherwise False
≤	Less Than or Equal To	A, B	True if A ≤ B, otherwise False

Programming with the Touch sensor

In Mindstorms EV3 programming, the Touch Sensor is an analog sensor that can detect whether or not its front button is pressed or released. Also, it is able to count single and multiple presses. It is used for the detection of obstacles or to trigger an action when the user presses it with their finger.



The Touch Sensor gives Logic data (True or False). The position of the Touch Sensor button is called its State, and is True when pressed in and False when not pressed in (released). If you change the mode selector from Measure to Compare that it has three more states: Released, Pressed and Bumped.



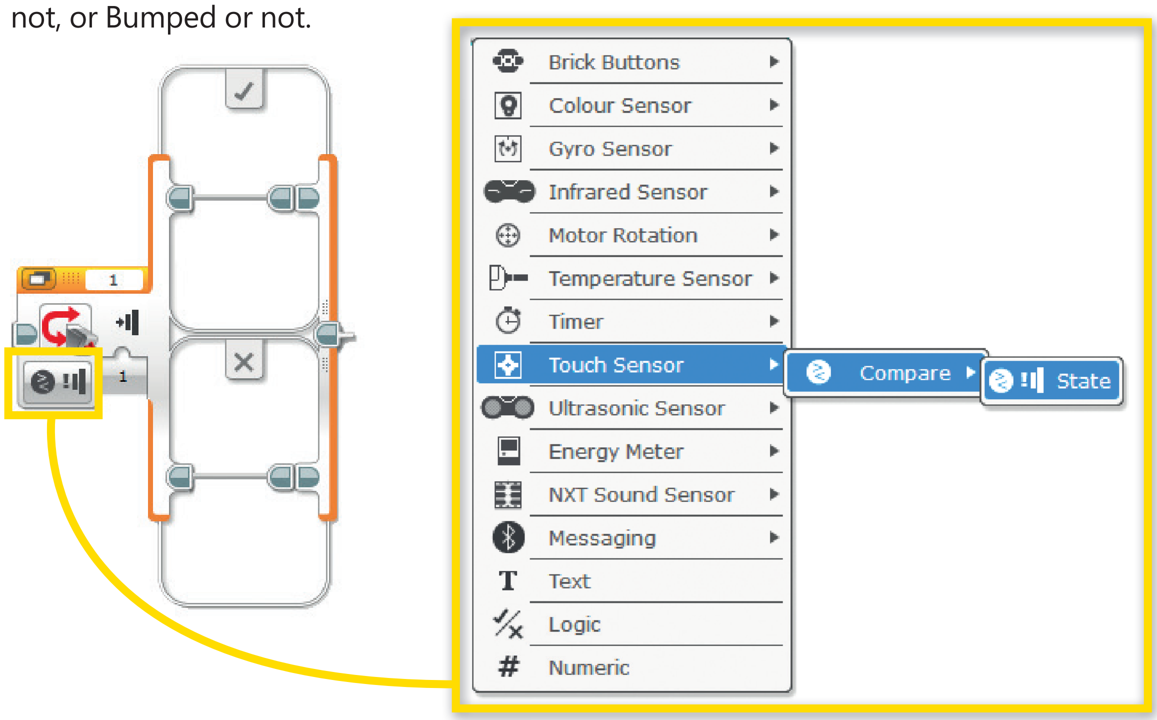
Touch sensor data:

Mode	State	Type	Logic output
Measure	Pressed in/not Pressed in	Logic	True if the button is pressed in, False if not.
Compare	Released	Logic	False if pressed in, True if not (opposite of Measure Mode).
	Pressed	Logic	True if pressed in, False if not (same as in Measure Mode).
	Bumped	Logic	True if the button has been pressed and released in the past. The next Bumped occurrence will then require a new press and release.

Depending on the situation, you can program your EV3 robot to respond to each state of the Touch sensor, by using the Wait block, the Loop block or the Switch block.

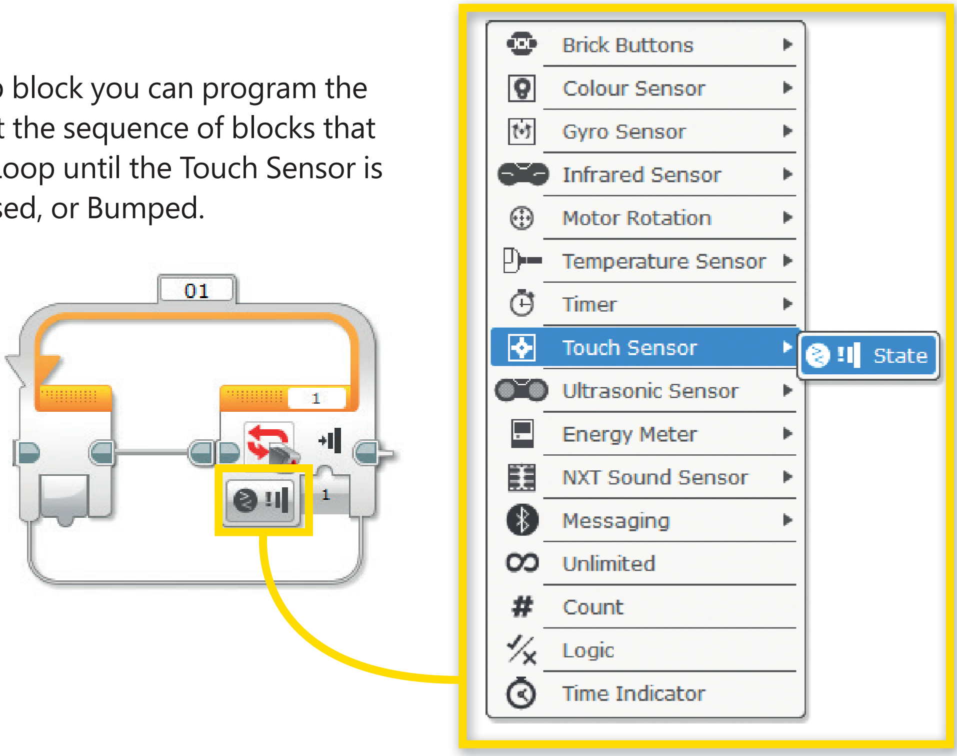
Switch block

Using the Switch block you can choose between two sequences (True case or False case) depending on whether the Touch Sensor is Pressed in or not, Released or not, or Bumped or not.



Loop block

Using the Loop block you can program the robot to repeat the sequence of blocks that are inside the Loop until the Touch Sensor is Pressed, Released, or Bumped.



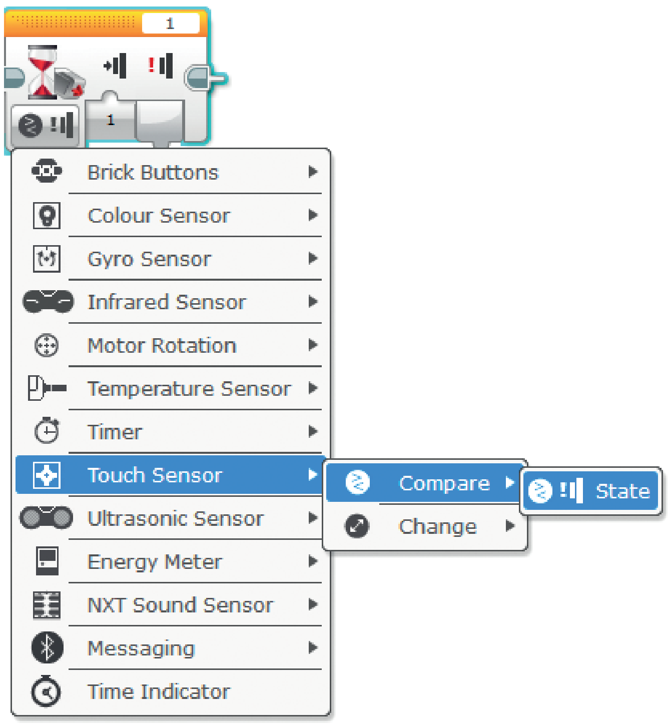
Wait block

Using the Wait block you can program the robot to wait for the Touch Sensor to be Pressed, Released, or Bumped or to simply wait to change its state.

Wait block modes:

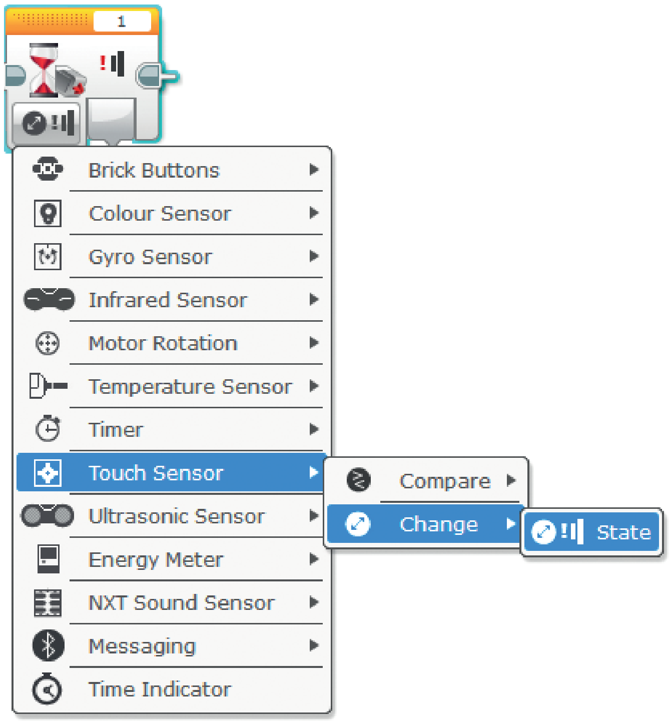
Compare - State Mode

Wait for the Touch Sensor to be Pressed, Released, or Bumped.



Change - State Mode

Wait for the Touch Sensor state to change.



hands on!

Given that k=5 *and* m=2 *fill in the tables below.*

CONDITION	Returns
k < 5 OR m ≥2	_ _ _ _ _ _ _ _
k ≥ 5 AND m >2	_ _ _ _ _ _ _ _
k+2 >5 OR m+1<4	_ _ _ _ _ _ _ _

CONDITION	Returns
NOT k < 6	_ _ _ _ _ _ _ _
NOT (k=5 or m<1)	_ _ _ _ _ _ _ _
NOT k >6	_ _ _ _ _ _ _ _

Make the appropriate combinations of k *and* m *in order that the following combinations make .sense.*

CONDITION	Returns
k......... AND m..........	*True*
k......... OR m..........	*False*
NOT (k=..... OR m=.....)	*False*