Programming blocks WeDo

»Home«

- Action
- Sensors
- Control
- Logic
- Math
- Text
- Lists [Expert-blocks]
- Colours
- Variables
- Functions [Expert-blocks]

»Start«

Each program starts with the »Program start block«, which cannot be deleted. The first block to be used is connected directly to the »Sequence connection« on the program start block.

Options:

- Create a new global variable.
- Remove the global variable.

If global variables were created:

- Text, name of the variable.
- Type of the variable.
- Value, initial value of the variable.

After you have created the global variable, the »start«-Block looks like this:

Action

The category »action« contains all blocks that control the WeDos movement or any other action the roboter can perform. It contains the following blocks:

- Blocks that control the robots movement
- Blocks that control the robots display
- Blocks that control the speaker
- Blocks that control the robots LEDs

When in Expert-mode the »action«-category is further divided into

- Movement
- Display
- Sound
- LED

»Motor ... Speed ... time ... «

This blocks sets the speed of the connected motor to the given speed for the designated time.

Input:

- Number, between -100 and 100 to control the speed and direction of the motor
- Number, time in milliseconds. 1000 milliseconds equal 1 second

» Motor ... Speed ... « [Expert-Block]

This blocks controls the speed of the connected motor.
Input:

- Number, between -100 and 100, controls the speed and direction of the motor.

»Stop Motor «  [Expert-Block]
This block stops the connected motor instantaneously.

»Show text «
With this block you can show any text in the WeDo-App. Each use of this block will print the text into a new line, so you don't have to worry about spacing.

Input:

- String, text that you want to show in the app.

»Clear Screen «
This block clears the text-window of your WeDo-App. Once it is empty you can close it by clicking "Ok".

»Play ... note ... «
With this block you can make your WeDo play a certain note for a period of a time.

Option:

- Length of the note, from whole to a sixteenth.
- Note, that you want your WeDo to play.

»Play frequency ... time ... «  [Expert-Block]
This block makes your WeDo play a certain frequency for a time.

The human ear can only register frequencies between 30 and 15000 Hz (Hertz). This may vary from person to person and from age to age.

Input:

- Number, Frequency of the note that you want your WeDo to play. Between 20 and 15000
- Number, Time you want the note to be played for.

»Turn LED on «
With this block you can control the LED of your WeDo and change it's colour.

Input:

- Colour, that you want the LED to be in (Choice of 10).

»Turn LED off «
This block turns the LED of your WeDo off.

Sensors
The category »sensors« contains all blocks that can control the WeDo's built-in sensors:
The main difference between blocks from the action-category is, that most of the sensor-blocks return a value to the next block, for example the infrared sensor returns a distance. You can see the type of value a block returns by the colour of its connector.

»Button pressed ?«
This block checks whether the WeDo’s button is currently being pressed. It then returns a logical value to the next block.

Return value:
- Logical value, "true" if the button is pressed, "false" otherwise.

»Gyroscope tilted «
This block uses the build-in gyroscope of the WeDo to check the robot's orientation. It then returns "true" or "false", depending on whether the robot is tilted into the specified direction.

Options:
- Direction, specify the direction you want the robot to check for, so e.g. left or right.

Return value:
- Logical value, "true" if the robot is tilted in that direction, "false" otherwise.

»Get distance Infrared «
This sensor uses infrared-light to check the distance to the nearest object. Be aware that the robot can only measure the distance of the direction the sensor is facing.

Return value:
- Number, between 0 and 10, 0 for very close, 10 for far away

»get value timer «
With this block you can get the value of the timer, which gets started automatically at program start.

Return value:
- Number, current value of the chosen timer in milli-seconds since start of the program or the last reset.

»Reset timer 1 «
With this block you can reset the value of the built-in timer to 0.

Control
The category "control" includes blocks with which the program sequence can be controlled. The category includes the following blocks:
- if do
- if do else
- repeat indefinitely
• repeat n times
• wait
• wait until
• repeat while/until [Expert-Block]
• for each item in list [Expert-Block]
• count with from to [Expert-Block]
• break out/continue with next iteration of loop [Expert-Block]

»if do«

With the block »if do« you can selectively trigger actions to be executed by your robot. The »if do« block therefore requires a logical value as an input parameter, the condition. Only if the condition of the »if« statement is true, the inserted block will be executed. In a nested »if do« block, if a further distinction was added, the first »if« condition is queried. If it is not fulfilled (condition = false), the second »else if« condition will be checked. Also this second condition requires a logical value as an input parameter.

The conditions may arbitrarily be expanded by clicking the "+" plus symbol. The "+" minus symbol reduces the block.

Settings and input values:

• Insert an additional condition.
• Delete the last condition.
• Boolean value, »true« or »false«.
• Blocks that will be executed

»if do else«

With the block »if do else« you can selectively trigger actions which are executed by your robot. The »if do then« block therefore requires a logical value as an input parameter. If the condition in »if« is true the inserted block will be executed, otherwise (condition is not fulfilled = false) the block connected to the »else« statement will be executed. In a nested »if do else« block with further distinctions added, the first »if« condition is queried. If it is not true, the second condition »else if« is checked. Also the second condition requires a logical value as an input parameter. Only when both conditions are not true, the block which is inserted at the »else« statement will be executed.

The conditions may arbitrarily be expanded by clicking the "+" plus symbol. The "+" minus symbol reduces the block.

Settings:

• Insert an additional if-do-else condition.
• Delete the last if-do-else condition.
• Boolean value, »true« or »false«.
• Do blocks that will be executed if the according condition evaluates to »true«.
• Else blocks that will be executed if the according condition evaluates to »false«.

»repeat n times«

With the block »repeat« you can run other block as many times as you like. All blocks in the »repeat« block will be executed as often as defined in the entry field. The blocks are applied sequentially from top to bottom. Once the last block has been executed, the program repeats with the first block again. Therefore, this block is also called »loop«.

Settings and input:

Only integer values can be entered.
• Number that indicates how often the contained blocks will be repeated.
• Blocks to be repeated as often as defined.

»wait«
With the block »wait« you can "pause" your program at the point where you inserted the »wait« block. Your program will then remain for the specified duration at this point. After the specified time the next block will be executed. For example you can display text in the screen of your robot for exactly the time you specified in the »wait« block.

Settings and input values:
• Number, waiting time in milliseconds.

»wait until ...«
With the block »wait until« you can "stop" your program at the point you insert the »wait until« block. Your program then waits until the condition is true. The »wait until« block you can extended by click on "+" symbol. Your program then waits until (at least) one of the condition of your »wait until« block is true.

Settings:
• Add a new condition.
• Delete the last condition.
• Boolean value, »true« or »false«.

»break out/continue with next iteration of loop« [Expert-block]
With the block »break out/continue with next iteration of loop« a loop can be terminated ahead of schedule. As soon as the block is entered within a sequence of blocks, all further blocks up to the end of the loop will be ignored.

Settings:
• Type of breaking behavior, »break out« or »continue with next iteration«.

»count with from to« [Expert-block]
With the block »count with from to« you can run other block as many times as you like. All blocks in the »count with from to« block will be executed as long as the counting is in progress. The blocks are applied sequentially from top to bottom. Once the last block has been executed, the program repeats with the first block again if the counting is in progress. The last parameter declares the step width for counting.

Settings:
• Variable, name of free choice; numbers of the counter will be delivered one after the other to the variable.
• Number, initial value of the counter.
• Number, final value of the counter. As the counter exceeds this value the loop ends.
• Number, defining the increment. The variable value will increase by this amount after every loop cycle.
• Blocks that will be executed in every loop cycle.
**»repeat while/until« [Expert-block]**

With the block »repeat while/until« you can run other block as many times as you like. All blocks in the »repeat while/until« block will be executed as long as the condition in the entry field is true. The blocks are applied sequentially from top to bottom. Once the last block has been executed, the program repeats with the first block again if the condition is still true. Therefore, this block is also called »conditional loop«.

Settings and input values:
- Option, »while« or »until«, defining the type of the conditional repetition.
- Boolean value, »true« or »false«.
- Blocks that will be repeated while/until the condition evaluates to »true«.

---

**Logic**

With the »logic« blocks you can "create" conditions. With this condition you can interrelate states, values, and events with each other.

The following »logic« blocks are available at NEPO :
- comparison
- and/or
- not [Expert-block]
- true/false
- null [Expert-block]
- test [Expert-block]

**»comparison«**

With the block »comparison« you can compare different parameters of the same type (number, color, logical value, text). This block can only be used in conjunction with another block which requires a logical value as an input parameter.

Settings and input values:
- Value for the left hand side.
- Comparison, select one of =, , <, >.
- Value for the right hand side.

Return value:
- Boolean value, »true« or »false«.

**»and/or«**

With the block »and/or« you can interrelate logical values with each other. The »and/or« block with the setting "and" will be true only if both logical parameters are "true": If the block has the setting »or « it is sufficient if one of the two parameters is "true", so that the »and/or« block will return true.

Settings:
- Boolean value on the left hand side.
- Boolean function, »and« or »or«.
- Boolean value on the right hand side.

Return value:
- Boolean value »true« or»false«.
**true/false**

With the block «true/false» you can return either the logical value «true» or «false» to another block.

Settings:
- Boolean value, make your choice.

Return value:
- Boolean value, «true» or «false».

**not** *[Expert-block]*

Using the «not» lets you invert a logical value and pass this value to another block.

Input value:
- Boolean value, to be inverted.

Return value:
- Boolean value, «true» or «false»; result of inverting.

**null** *[Expert-block]*

The block «null» is a place holder for an input value that is not yet specified. If for instance a new connection variable has been created and is not yet bound, the initial value of this connection will be set to «null».

Return value:
- Boolean value, «null».

---

**Math**

Using «Math» blocks offers calculations and parameter settings.

The following «Math» blocks are available at NEPO:
- parameter
- calculating
- mathematical function [Expert-Block]
- trigonometric function [Expert-Block]
- constant [Expert-Block]
- number property [Expert-Block]
- change by ... [Expert-Block]
- round [Expert-Block]
- list evaluation [Expert-Block]
- remainder of [Expert-Block]
- constrain [Expert-Block]
- random integer [Expert-Block]
- random fraction [Expert-Block]

**parameter**
With the block «parameter» you can send numbers to another block.

Settings and input values:
- Number

Return value:
- Number

»calculating«

With the block «calculating» you can sum up, subtract, multiply, and divide numbers. This block can only be used in conjunction with another block which requires a number as an input parameter.

Settings and input values:
- Number, first number on the left you want to calculate with.
- Option, mathematical operator, choose one of +, -, ×, ÷, ^.
- Number, second number on the right you want to calculate with.

Return value:
- Number, result of the calculation.

Degree and Radiant Measurements

In standard blocks of Open Roberta, especially along with the actions «turn… by x degree» and in the sensor block «get angle gyrosensor …» the angles will be represented in degree (0° ... 360°). The unit is called degree and will be represented by the Symbol °. 90° represents a right angle, 180° a half circle or a turn around, and 360° represents a full circle.

The trigonometric functions in the expert mode, like sin, cos, tan, ... the input parameters will be expected to be given in radian measure. The unit is called RAD. A right angle is represented by π/2, the half circle or turn around is π, and the full circle is 2π.

To convert degree (°) into radian measure (RAD) or vice versa you may use the following calculations:
- degree to radian measure: (x degree / 180) * π
- radian measure to degree: (x RAD * 180) / π

»mathematical function« [Expert-block]

With the block «mathematical function» some elementary mathematical functions may be calculated. Available functions are «sqrt», «abs», «invert» (multiply by -1), «ln» (natural logarithm), »log10« (decadic logarithm), »e^x« (exponential function), »10^x« (base 10 exponent)

Settings and input value:
- Option, mathematical function, choose one.
- Number to apply the function to.

Return value:
- Number, result of the function application.

»trigonometric functions« [Expert-block]

With the block «trigonometric function» sine, cosine, tangent and their respective revers functions can be calculated. Input values are expected in radian measure (see hint above).

Settings and input:
-
• Option, trigonometric function to choose from.
• Number, in radian measure.

Return value:
• Number, result of the trigonometric calculation.

»constant« [Expert-block]
With the »constant« block some mathematical constant values are available: »π« (3,1415...), »e« (2,718...), »ϕ« (1,618...), »sqrt(2)« (1,414...), »sqrt(½)« (0,7071...), »«

Settings:
• Option, mathematical constants, choose one.

Return value:
• Number, value of the selected mathematical constant. Infinity will return »infinity«.

»number property« [Expert-block]
With the block »number property« you check whether a given number has a specific property: »even«, »odd«, »prime«, »whole«, »positive«, »negative«, »divisible by«.

Settings and input values:
• Number, the property of this input value will be checked.
• Option, number property, choose one.
• Number, evaluated from a block. The second input value is only required for the property »divisible by«.

Return value:
• Boolean value, »true« or »false«, depending on the selected property.

»change by ... « [Expert-block]
The block »change by ... « increments a numerical variable by a defined value.

Settings and input values:
• Variable name, which value is to be changed. Choose the variable to be changed.
• Number, given by a block.

»round« [Expert-block]
With the block »round« values may be rounded. Rounding will set the decimal places to 0. It depends on the value of the decimal places whether the block rounds up or down. You may also decide by settings to always round up or down.

Settings and input value:
• Type of rounding, choose a round mode.
• Number you want to round.

Return value:
• Number, rounded whole number, according to the round mode.
»remainder of« [Expert-block]

The block »remainder of« calculates a division and returns the remainder of the division.

Input values:
- Number to be divided (dividend).
- Number, divisor.

Return value:
- Number, rest of the division.

»constrain« [Expert-block]

The block »constrain« ensures that given boundaries will not be exceeded.

Input values:
- Number, that will be constrained.
- Number, lower bound.
- Number, upper bound.

Return value:
- Number, constrained by the lower and the upper bound.

»random integer« [Expert-block]

With the block »random integer« you may generate random integer numbers within defined limits.

Input values:
- Number, lower bound
- Number, upper bound

Return value:
- Number, a whole random number from within the upper and lower bounds.

»random fraction« [Expert-block]

With the block »random fraction« a random value between 0.0 and 1.0 is calculated.

Return value:
- Number, positive random value between 0.0 and 1.0.

Text

»Text« blocks perform simple operations on text strings.

Available »Text« blocks in NEPO are:
- Text
- create text [Expert-block]
- append text [Expert-block]
»Text«
The simple »Text« block creates a little text.
Input value:

• String, containing arbitrary characters.

Return value:

• String, containing arbitrary characters.

»comment«
Document your program with this block, so that you and others will find it easier to understand your program later. This comment will also be visible in the generated source code.
Input value:

• String, containing arbitrary characters.

»create Text« [Expert-block]
The »create text« block compiles a text from different input parameters. Using the + sign will insert further input slots. All input parameters will be connected one after the other. Essentially the »create text« block converts an arbitrary input value into a text string.
Input values:

• Arbitrary values (numbers, text, logical values, colours)

Return value:

• String, containing arbitrary characters, compiled sequentially from all input values.

Colours
In the category »Colours« you will find all the blocks you can use to create and select colours.
»Colour picker «

With this block you can choose from a pool of predefined colours for your robot.

options:

- option, choose the colour you want to use.

return value:

- colour, your selected colour.

Variables

The »Variables« blocks are used to create global variables of six different parameter types. The global variables store values which may be used at arbitrary positions in a program.

Each variable has to be created prior to its use. Create a variable by clicking the »+« sign in the »start« block. Each variable has to be assigned a name and a parameter type. Further clicks on the »+« sign will create additional variables.

The name of a variable has to be unique in a program.

Settings and input values:

- Insert another variable.
- Delete this variable.
- Variable name, special characters and space sign are not allowed.
- Variable type, `number`, `boolean`, `string`, `colour`, `connection`, `list number`, `list boolean`, `list string` or `list connection`.
- Initial values, according to the variable type.

These «Variables» blocks are available in NEPO:

- set variable
- get variable

**»set variable«**

Using the block «set variable» will assign a value to a variable. Depending on the variable type the value may be assigned by an input connector.

Settings and input value:

- Variable, which value is to be changed.
- Value, new value for the variable, evaluated from a suitable block.

**»get variable«**

Using the block «get variable» returns the value of a variable to another block. The type of the output parameter is equal to the type that has been assigned to the variable in the «start» block.

Settings:

- Variable, that will be read. The value will not be changed by reading.

Return value:

- Value, stored in the variable.

---

**Functions [Expert-blocks]**

With «Functions» blocks (in Java also called «methods» or «subprograms») you may create more readable programs. NEPO offers some function blocks:

- Function blocks with/without input parameters and no return parameter
- Function blocks with/without input parameters and one return parameter
- If block to be used within a function

With elementary function blocks you may program simple and complex functions which may be used by function calls. Each programmed function is available immediately as an additional block in the «Functions» category. Those are

- Function calls with/without input parameters and without return parameter
- Function calls with/without input parameters and also one return parameter

Depending from the number of input and return parameters the selection of function blocks varies.

A unique name is assigned by NEPO to each new function definition (doSomething, doSomething2, ...). This name may be altered. Overwriting of a defined function is not possible.

**»Function blocks with/without input parameters and no return statement«**

In a function block with/without input parameter and without return parameter a sequence of program statements will be condensed. The input parameters will be determined by the local variables of the function block. Local variables may be generated by using the «++» sign of the function block. Local variables will not be initialized.

If a function block contains an if-block the function may be terminated before reaching its end.
The function block is available in the »Functions« category immediately after its definition. Using function blocks improves the readability of complex programs.

Settings:

- Function name, no special characters and white spaces allowed.
- Generate new local variables, that will be assigned with the function call.
- Delete the associated local variable.

Example function with two local variables and no return parameter:

```
+ sayHello with:
+ variable x : Number ->
+ variable y : Number ->
show text: "Hallo Roberta"
```

Using the function in a program:

```
+ doSomething2
sayHello
x 13
y 43
```

»Function blocks with/without input parameters with return statements«

In a function block with/without input parameter and with return parameter a sequence of program statements will be condensed. The input parameters will be determined by the local variables of the function block. Local variables may be generated by using the «+» sign of the function block. Local variables will not be initialized. After running through all the blocks of the function a value will be returned.

If a function block contains an if-block the function may be terminated before reaching it's end. An alternative value may be returned by the if-block.

The function block is available in the »Functions« category immediately after its definition. Using function blocks improves the readability of complex programs.

Settings:

- Function name, no special characters and white spaces allowed.
- Generate new local variables, that will be assigned with the function call.
- Delete the associated local variable.
- Data type for the return value; choose one data type.
- Block, that returns a value of the defined data type.

Return value:

- Value of the defined data type, evaluated in an appropriate block.

»if block to be used within a function«

The if statement within a function is of special importance. As the function sequence meets an if statement the validity of the condition will be checked.

If statement within a function with return value:

Example: If x > 5 the function terminates and returns "Winner" and there will be no movement of the
If the condition evaluates to »true« the rest of the function will be ignored and the function terminates. The second input value of the if statement will be returned. A possibly defined return value of the function will be ignored. The return value data type is determined by the return data type of the function definition.

If the condition evaluates to »false« the sequence of the function blocks will be continued. The return value of the function will be returned after reaching the end of the function.

If statement within a function with no return value:

- An if statement within a function without return value has just the if statement as input parameter.
- If the condition evaluates to »true« the rest of the function will be ignored and the function terminates.
- If the condition evaluates to »false« the sequence of the function blocks will be continued.

Settings and input values:

- Boolean value, possibly evaluated by a condition.
- Value of the specified data type.

Return value:

- Value of the specified data type.

»function call without return value «

With this block you can call a previous defined function in your program.

input values:

- element, depending on the required input value of the function.

»function call with return value «

With this block you can call a previous defined function in your program.

input values:

- element, depending on the required input value of the function.

return value:

- element, depending on the return value of the function.