Blockdescriptions mBot

Content

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

»Start«

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

Setting options:

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

If global variables have been created:

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

Action

The category »Action« contains all blocks with which your mBot directly performs an action.

The following blocks belong to the category »Action«:

- Blocks to control the motors
- Blocks for output control
- Speaker control blocks
- Blocks for controlling the status lights

In expert mode, the »Action« category is subdivided into the following subcategories:

- »Driving«
- »Move«
- »Display«
- »Sound«
- »Status light«

»Drive ... speed % ... time ms ... «

With this block you can move your mBot. The mBot then moves at the entered speed for the specified time.

Settings and input values:

- Option, direction in which your mBot should move, either forward or backward.
- Number, tempo with which your mBot should move. Between 1 and 100, where 1 means »very slow« and 100 »very fast«.
• Number, time your mBot should move for, in milliseconds.

»Drive ... speed % ... «

With this block you let your mBot drive in the desired direction with the given speed without setting a time. So the robot will keep going if you don’t stop it.

Settings and input values:
• Option, direction in which your robot should move.
• Number, speed at which your robot should move, between 1 and 100, where 1 means »very slow« and 100 »very fast«.

»Turn ... speed % ... time ms ... «

With this block you can rotate your robot in the desired direction. The robot rotates at the specified speed for the specified time.

Settings and input value:
• Option, direction in which your robot should rotate.
• Number, speed at which your robot should rotate, between 1 and 100, where 1 means »very slow« and 100 »very fast«.
• Number, time your robot should spin for, in milliseconds.

»Turn ... speed % ... «

With this block you can rotate your robot in the desired direction. The robot rotates at the given speed until you stop it again.

Settings and input value:
• Option, direction in which your robot should rotate.
• Number, speed at which your robot should rotate, between 1 and 100, where 1 means »very slow« and 100 »very fast«.

»Steer ... speed % left ... speed % right ... time ... «

With this block you can let your robot take a curve by setting the speed of the two motors independently of each other. You can also control how long you want your robot to take this turn.

Settings and input value:
• Option, direction in which your robot should move.
• Number, speed of the left motor, between 1 and 100, where 1 means »very slow« and 100 »very fast«.
• Number, speed of the right engine, between 1 and 100, where 1 means »very slow« and 100 »very fast«.
• Number, time in milliseconds your robot should take the curve for.

»Steer ... speed % left ... speed % right ... «

With this block you can let your robot take a curve by setting the speed of the two motors independently of each other. Your robot will keep driving this curve until you stop it.

Settings and input value:
• Option, direction in which your robot should move.
• Number, speed of the left motor, between 1 and 100, where 1 means »very slow« and 100 »very fast«.
• Number, speed of the right engine, between 1 and 100, where 1 means »very slow« and 100 »very fast«.
»Stop«
This block stops your robot immediately.

»Motor ... on speed % ... time ms ... « [Expert-Block]
With this block you can control a motor connected to your robot. You can also control the speed of the motor and the time the motor should run for.

Settings and input value:
- Option, motor you want to control.
- Number, speed of the motor, between -100 and 100. The motor runs backwards if the speed is a negative number.
- Number, time for which the motor should run.

»Motor ... on speed % « [Expert-Block]
With this block you can control a motor of your robot by controlling its speed. The motor will continue to run at the specified speed until you stop it or change its speed.

Settings and input value:
- Option, motor you want to control.
- Number, speed of the motor, between -100 and 100. The motor runs backwards if the speed is a negative number.

»Stop motor port ... « [Expert-Block]
With this block you can stop an engine of your robot.

Settings and input value:
- Option, motor you want to stop.

»Show on Serial Monitor «
With this block your robot can output a string via the serial bus. To receive these messages, you need to connect your robot to your computer via USB.

Settings and input value:
- String, message to be displayed on the Serial Monitor.

»show text LED matrix Port ... «
With this block you can display a text of your choice on the LED Matrix. The text is displayed as scrolling text on the LED Matrix and cycles once.

Options and input values:
- option, choose the port your LED matrix is connected to.
- string, the string you want to display.

»show ... LED matrix Port ... «
With this block you can display either a single image or a sequence of images on your LED Matrix.

Options and input values:
• option, choose between displaying a single image or a sequence of images as an animation.
  option, choose the port the LED matrix is connected to.

• image or list of images, the image or animation you want to display.

»clear LED matrix Port ... «
With this block you can clear the screen of your LED matrix, which will turn off all LEDs.
Options:
  • option, choose the port the LED matrix is connected to.

»set brightness LED matrix Port ... « [Expert block]
With this block you can control the brightness of the LED Matrix.
options and input values:
  • option, choose the port the LED matrix is connected to.
  • Number, brightness of the LED matrix, number between 0 and 9, where 0 means off and 9 is the maximum brightness

»Play frequency Hz ... duration ms ... « [Expert-Block]
With this block you can get your robot to play a sound from its speaker. You can control the frequency and duration of the sound.
Settings and input value:
  • Number, frequency of the tone, tones are only audible between 20 and 20000 Hz.
  • Number, duration for which the sound is to be played.

»Play ... note ... «
With this block you let your robot play a note for a certain time.
Settings and input value:
  • Option, playing time of the note.
  • Option, note to be played.

»Turn LED on ... colour ... «
With this block you can control one of the RGB-LEDs of your robot and let the LED light up in one color.
Settings and input value:
  • Option, position of the LED you want to turn on.
  • Colour, the color in which the LED should light.

»Turn LED off ... «
With this block you can switch off one of the LEDs of your robot.
Settings and input value:
  • Option, position of the LED you want to turn off.

Sensors
The category »Sensors« contains blocks with which you can control and read the sensors of your mBot. The following sensors can be controlled:

- Buttons
- Ultrasonic sensor
- Infrared sensor
- Light sensor
- Timer

A special feature of the sensor blocks in contrast to the action blocks is that many sensor blocks return a »value«. You can recognize the value type returned by a sensor block by the colour of the connector. The connector is the connector with which NEPO blocks can be inserted into other blocks. A detailed description can be found in the NEPO section.

»Button pressed? «

With this block you can check the status of the button of your mBot.

Return value:

- Logical value, »true« if the key is pressed, otherwise »false«.

»Get distance cm ultrasonic sensor Port ... «

With this block you can check your distance sensor. It measures the distance to the next obstacle in front of the robot by emitting ultrasonic waves and measures how long they take to return.

Settings and input value:

- Option, port to which the ultrasonic sensor is connected.

Return value:

- Number, distance to the next obstacle, in centimeters.

»Get Line infrared sensor Port ... «

With this block you can query the state of the infrared sensor. This sensor measures the shading of the background by infrared light.

Settings and input value:

- Option, port to which the sensor is connected.
- Option, position of the sensor to be checked.

Return value:

- Logical value, "true" if the background is dark, otherwise "false".

»Get light % light sensor Port ... «

With this block you can check the state of the light sensor. This sensor can measure the brightness of the environment.

Settings and input value:

- Option, port to which the light sensor is connected.

Return value:

- Number, brightness of the ambient light, between 1 and 100, where 1 means »dark« and 100 »bright«.
Get value ms timer ... «

With this block you can display the value of a timer.

Settings and input value:

- Option, number of the timekeeper you want to query

Return value:

- Number, time since last reset or program start, in milliseconds.

Reset timer ... «

With this block you can reset a timer or start a new one if the selected number has not measured any time before.

Settings and input value:

- Option, number of the timekeeper to be reset or started.

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 else« 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 indefinitely«

With the block »repeat indefinitely« you can endlessly run the blocks on your robot. All blocks which are within the »repeat indefinitely« block will be executed endlessly. 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«.

Input:
- Blocks to be repeated indefinitely.

»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:
- 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«.
»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 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.

»for each item in list« [Expert-block]

With the block »for each item in list« all list items will successively be bound to a variable. The variable can be used within the loop. With each loop cycle the next item of the list will be bound until all list elements have been processed.

Settings and input value:
- Type of list elements, »Number«, »String«, »Boolean«, »Colour«, »Connection«
- Variable, name of free choice; list elements will be delivered one after the other to the variable.
- List that contains elements of the desired type. If the list elements are not of the correct type then the list will not fit to the input slot.
- Blocks which will be executed as often as there are elements in the list.

»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.

**»test« [Expert-block]**

Using the »test« block will perform a test and returns a value which depends on the test result.

Input values:

- Boolean value for the test; if no input value is given "true" will be assumed.
- Arbitrary value, evaluated by a block.
- Arbitrary value, evaluated by a block.

Return value:

- Value of arbitrary type.

**»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:
»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 Radian 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 /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 »square root«, »absolute«, »invert« (multiply by -1), »ln« (natural logarithm), »log10« (decadic logarithm), »e^« (exponential function), »10^« (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 value:
- Option, trigonometric function to choose from.
- Number, in radian measure.
»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.

»list evaluation« [Expert-block]
With the block »list evaluation« you may analyse a list.

Modes for list evaluation:
• sum - addition of all list values
• min - smallest value in the list
• max - largest value in the list
• average - average of all list values
• median - median of all list values
• standard deviation - standard deviation of all list values
• random item - one list value randomly selected

Settings and input value:

• Mode of list evaluation, choose one mode.
• List of numbers

Return value:

• Number, calculated according to the selected list evaluation 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:
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.

»append Text« [Expert-block]

The »append text« block will append some string to a given string, for instance to extend a message with a signature.

Input values:

- String, to which some other text will be appended.
- String, that shall be appended.
Lists [Expert-blocks]

A list is a sequence of data (of identical parameter types) with elements in a fixed succession. Create a »list« type variable to make use of a »List« block. NEPO provides five different parameter types for lists: number, string, boolean, colour and connection.

NEPO provides the following »List« blocks:

- create list
- repeat element in list
- length of
- is empty?
- find in list
- get list element
- set list element
- get sublist

»create list«

The block »empty list« creates a list with no content. The block »list« generates a list with some predefined values.

This block may only be used in the context of a »set variable« block.

Using »+« or »-« enables you to extend or reduce the list at its end.

Settings and input values:
- List type, »Number«, »String«, »Boolean«, »Colour«, »Connection«.
- Create further list element, append to the end of the list.
- Delete list element at the end of the list.
- Values, according to the list type. Here the input of initial values is possible.

Return value:
- Empty list or list, containing elements of the specified list type.

»repeat element in list«

The »repeat element in list« block generates a list of equal elements.

Settings and input values:
- List type, »Number«, »Boolean«, »String«, »Colour« or »Connection«.
- List element, according to the selected list type. This value will be repeated in the list.
- Number, defining how often the list element will be repeated.

Return value:
- List of the specified list type, containing the specified number of equal elements.

»length of«

The »length of« block returns a value which is the length of the list given as parameter. An empty list has a length of 0.

Input value:
- List, evaluated by an appropriate block.

Return value:
- Number, number of list elements.
»is empty?«
A list given as parameter will be checked whether it is empty.
Input value:
  • List, evaluated by an appropriate block.

Return value:
  • Boolean, either »true« or »false«.

»find in list«
A list is searched for an item. If the item is in the list, the list position will be returned. If the item is not in the list the result is -1.
Settings and input values:
  • List, that will be examined.
  • Position of the occurrence, either »first« or »last«.
  • Value, list item to be found.

Return value:
  • Number, indicating the position where the element was found in the list.
  
  Note: Counting list positions will start with 0.

»get list element«
This block accesses an item of a list. Depending on the settings this item may be altered.
A list is given as parameter. Then a drop-down-list specifies what will happen to the list item under consideration.
Settings:
  • List, that is under consideration.
  • Option, action for the element found: »get« reads the element and leaves it unchanged, »get and remove« reads the element and removes it from the list, »remove« just removes the element from the list.
  • Option, position of the element found: »#«, »# from end«, »first«, »last« or »random«.
  • Number, indicating the list position. This input value is not required if »first«, »last« or »random« was selected as position.
  
  Note: Counting of list positions starts with 0.

Return value:
  • List element that has been found at the specified list position; »undefined«, if the list position does not exist.
  
  With selection of »remove« there is no return value; instead the list will be shortened by this list element.

»set list element«
In a list given as input parameter one specified element will be replaced by a new value.
Settings and input values:
  • List, that will be changed.
- Option, action for the element found; «set» changes the element, «insert at» inserts a new element into the list.
- Option, position for the element to be changed: »#«, »# from end«, »first«, »last« or »random«.
- Number, indicating the list position. This input value is not required if »first«, »last« or »random« was selected.
- List element, that will be set or inserted at the selected list position.

»get sublist«

From a list given as parameter a sublist will be created. The sublist contains all those elements that match the further specifications of the block.

Settings and input values:
- List to be examined.
- Position, start of the sublist: »#«, »# from end« or »first«.
- Number, indicating the list position. This input value is not required if »first« has been selected.
  Note: Counting of list positions starts with 0.
- Position, end of the sublist: »#«, »# from end« or »last«.
- Number, indicating the list position. This input value is not required if »last« has been selected.
  Note: Counting of list positions starts with 0.

Return value:
- List, a sublist of the same type as the given list.

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.

»Colour with red ... green ... blue ... « [Expert Block]
With this block you can create your own colour by mixing red, green and blue.

input values:

- number, the red part in your color. Value between 0 and 255.
- number, the green part in your color. Value between 0 and 255.
- number, the blue part in your color. Value between 0 and 255.

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« oder »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.
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 it’s 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.

- The name of a function block has to start with a lower case letter. Special characters are not allowed in a function block name.
- If the function block defines input parameters (local variables), all the parameter slots have to be occupied when calling the function.

»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:

- 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.
»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.

Messages [Expert-Blocks]
This category contains blocks with which your mBot can communicate with other mBots. This is done by infrared signal, so the mBots should be able to see each other to communicate with each other.

This category contains the following blocks:
- »Send message«
- »Receive message«

»Send message «
With this block your mBot can send a message to all other mBots nearby.

Input option:
- String, message you want to send.

»Receive message «
With this block your mBot can receive messages from other mBots in the area.

Return value:
- String, received message from other mBots in the environment.