https://www.erm.ms/
contact@erm-robotique.com

Blockly for NAO

User Guide

2023-12-10
en03

ERM Robotique&IOT
est une marque de
ERM Automatismes Industriels

ERM Automatismes
Industriels
561, allée Bellecour
84200 Carpentras, France

Presentation

Blockly for NAO is a tool to program the NAO robot easily with Blockly graphic language.

It is composed of two parts:

- a PC program, named "**Blockly for NAO**", which is the programming interface.
- a NAO program, named "Blockly for NAO Library", also called "**Module**", which is used for communication.

Blockly for NAO works with projects. These are files, with the extension ABLK, which contain the Blockly script, the animations and all the associated files *(image, music, etc. ...)*. The advantage of this mode of operation is that you can easily move or copy projects, because all the media will always be available.

The software suite is under a registered license and associated with a robot. Only one license file is needed to activate the bundle.

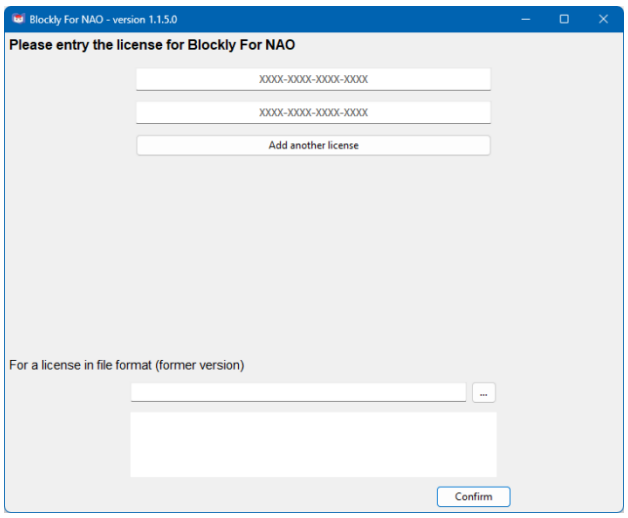
Installation

Installing the PC program

The installation is done via an installer. Let yourself be guided in this wizard.

Activation of the PC program license

At the first start, the PC program will automatically open the license entry window.



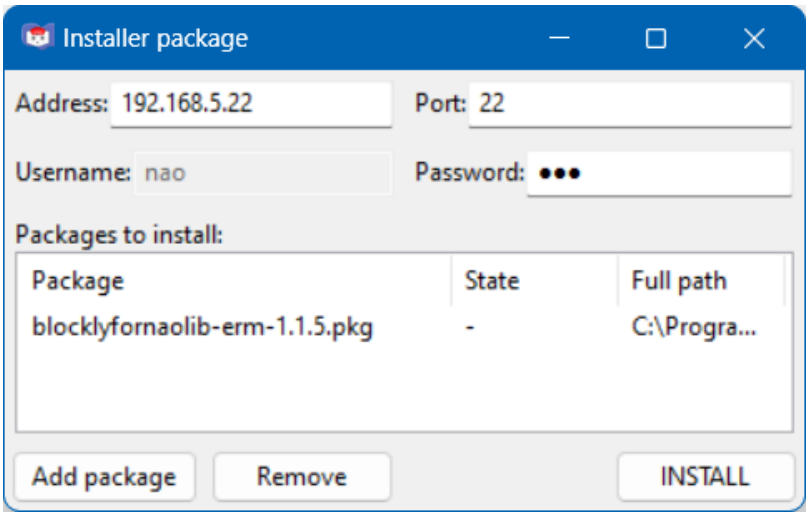
First, click on the **[...]** button to retrieve your license file, which was emailed to you.

The licenses are nominative, so check that the name that appears is yours.

Then press the **[confirm]** button. The application will restart and open the login screen. If the same window reappears, it means that the license is invalid.

Installing / updating the module in the robot

You must first have installed the PC program. Then connect your NAO robot to the same network as your computer. Open application, without connecting to robot (*leave the "IP address" field blank and validate*). Then, go to "**Tools**" tab and click on "**Install module on robot**" button.



In the **Address** field, enter the IP address of your robot.

In the **Password** field, enter the password of the robot *(default: nao)*.

Then if the list is empty, click on the **[Add pkg]** button and select the file *blocklyfornao-erm-x.x.x.pkg*, which is in the installation directory.

From now on, you must be as in the illustration image above.

Finally, click the **[Install]** button to start the installation.

Meaning of the error messages:

- "QI Timeout": The robot did not answer or the address is not correct. You should retry or check the settings and connections.
- "QI Cannot connect": Cannot connect to the robot.
- "QI Error": An error occurred while connecting to the robot.
- "SFTP Bad login / pwd": The user or password is incorrect.
- "SFTP Host not found": The robot's address is incorrect, or there is no file service.
- "SFTP Cannot connect": Unable to connect to the robot's file service.
- "SFTP Error in connection": An error occurred while connecting to the robot's file service.

Activation of the module license

This activation is almost automated. In fact, the first time you run a Blockly script, a dialog will ask you if you want to use the license installed on the computer to activate the robot module.

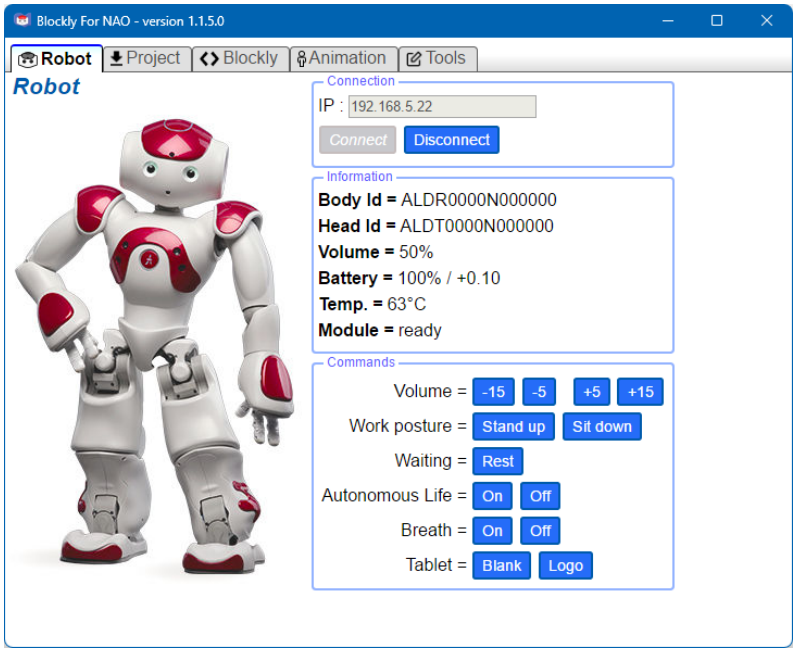
How to use

Connection

The connection to the robot can be done in two places.

Either by entering the IP address of the robot directly to the opening page of the PC program, or leaving this field blank and using the "*Robot*" screen, described in a following chapter.

Robot screen



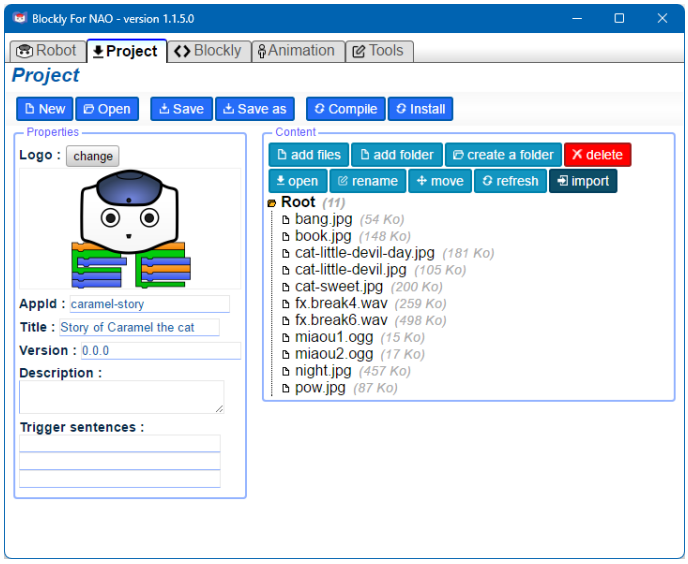
Body Id and **Head Id** are the serial numbers of the robot *(respectively the number of the body and head)*.

Battery indicates the level of charge *(it is 100% full and low above 30%)* and its use *(positive = the battery is filling, negative = it is discharging)*.

Temp. corresponds to the highest temperature of the engines *(correct above 85 ° C)*.

Module displays the status of the program installed in the robot.

Project Screen



The blue buttons are for managing the project itself: Create a **new** project, **open** a project, **save** a project and **upload** the project to the robot without running it.

The **Compile** button exports the script for use in third-party installation tools. The **Install** button compiles and installs the script in the robot, for use without the *Blockly for NAO* Windows interface (*but still requires the module*).

The "*Content*" group is for the management of files to be attached to the program, such as music, sound effects, images, etc ...

- **add files:** Add one or more files from the computer to the project.
- **add directory:** Allows you to add an entire directory and its contents to the project from the computer.
- **create directory:** Creates an empty directory in the project.

The added files and directories are placed in the selection, or where applicable, under the root.

- **delete:** Deletes the selected files and directories.
- **open:** Open the file by Windows, with the associated application by default.
- **rename:** Rename the selected files. If multiple files are selected, a numbering will be added at the end of each file.
- **move:** Move the selected items to another directory.
- **refresh:** Recalculate and redisplay the list of embedded files to the project. *The only usefulness of this button is in the case where the attached files are modified outside the Blockly for NAO program.*

The "*Properties*" group is only needed for compilation or installation.

Logo: script image

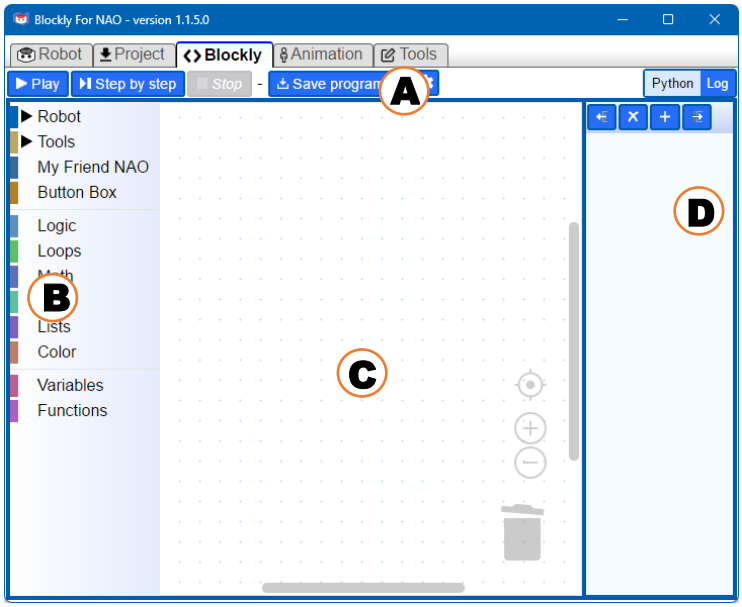
AppId: script identifier. It is used to find the program when it is installed. It's a unique name, made up of letters, numbers and dashes (no punctuation, no spaces, no accents).

Version: Script version number. It consists of one, two or three numbers separated by a dot.

Description: (*optional*) Text describing the script. It is displayed on programs listing installed programs, such as My Friend NAO.

Trigger sentences: (*optional*) Phrases that could be said to the robot to launch this script, in autonomous life, when the script is installed.

Blockly screen



The "*Blockly*" page is the Blockly programming interface.

Part **(A)** contains the execution control buttons: **[Play]**, **[Step]** and **[Stop]**.

The **[Save]** button is the same button as the "*Project*" button.

Part **(B)** is the toolbox that contains all the programming blocks. The blocks are arranged in a tree.

Part **(C)** is the work area. It's inside of it that you design your program, by dragging and dropping your blocks.

Part **(D)** is the log. In your Blockly script, you have the option to display information to the user.

The **[←]** and **[→]** buttons are for enlarging and narrowing the log area. The **[x]** buttons are for clearing all logs and the **[+]** button is for adding a separator line.

NAO blocks for Blockly

Blocs Robot / Audio

Say and **Animated Say:** To make the robot speak. The second block moves the arms of the robot to accompany the speech *(ideal for long texts)*.

Language: To change the spoken and recognized language of the robot. Attention, you must have previously installed the corresponding speech synthesizer.

Volume: To change the overall volume of the robot or to retrieve the value. The volume is between 0 *(quiet)* and 100 *(loud)*.

Play Audio: Play an audio file.

- If the action is "Play", the block is blocking. That is, the script will wait for the end of playback to resume.
- If the action is "Start", the block starts the audio in the background, and the script continues without delay.
- If the action is "Wait", the block waits for the end of the playback of an audio that has been launched in the background.
- If the action is "Stop", the block stops playing an audio that has been launched in the background.

The formats recognized by the robot are WAV, MP3 and OGG.

Stop all sounds: Stop all sound launched by the Blockly script.

Blocs Robot / LED

Leds: Applies the color, passed in parameter, on the selected group of LEDs. The group is selected with the drop-down list. Fade creates a transition in the color change. Some LED groups do not have all the colors. To extinguish, just apply the color black.

Blocs Robot / Program

Program: Run a subprogram installed in the robot. The subprogram is identified by a text consisting of an Application ID and the path to Behavior.

- If the action is "Run", the block is blocking. That is, the script will wait for the end of the subroutine to resume.
- If the action is "Start", the block starts the subprogram in the background, and the script continues without delay.
- If the action is "Wait", the block waits for the end of the subroutine that has been launched in the background.
- If the action is "Stop", the block stops the subroutine that has been launched in the background.

Robot / Memory

Memory: To read or modify a memory value. There are two parameters, the first is the address / key to access the memory and the second is the value to write.

Memory with raise: This block functions like the conventional memory block, except that it raises an event *(used in the case of an inter-program communication).*

Robot / Move

Position: Change the posture of the robot and activate the motors. The "*Rest*" position is special because it will put the robot in the closest rest position *(sitting or crouch)* and turn off the engines.

Animation: Play an animation that was created in the "*Animation*" page.

- If the action is "Play", the block is blocking. That is, the script will wait for the end of the animation to resume.
- If the action is "Start", the block starts the animation in the background, and the script continues without delay.
- If the action is "Wait", the block waits for the end of the playback of the animation that has been launched in the background.
- If the action is "Stop", the block stops the animation that was launched in the background. ATTENTION to the position at which the robot stops, it could unbalance it.

Stop all animations: Stop all animations launched by the script

Walk: Walk the robot. The parameters are relative coordinates, respectively the distance in meters in front of him, distance in meters to his left and the rotation in radian, in counterclockwise direction.

Walk with arms: like the "Walk" block, with the difference that the arms are left free

Robot / Sensor

System Sensor: Returns the value of the selected sensor:

- The level of the battery in percentage *(between 0 and 100)*.
- The position name in English *(Crouch, LyingBack, LyingBelly, Sit, SitRelax, Stand, StandInit, StandZero)*.
- The highest temperature in °C *(hot from 85°C)*.
- The state of the foot contact with ground that is a Boolean value *(True or False)*.

Hand sensor: Returns *True* if one of the selected sensors, of the selected hand, is currently stimulated.

Head Sensor: Returns *True* if any of the three checked sensors, head, is currently being stimulated.

Foot Sensor: Returns *True* if one of the two checked bumpers of feet, is currently pressed.

Read a NAO Mark: Wait for the robot to see a NAO Mark *(or LandMark)* and return its numerical value. If after ten seconds no tag has been seen, the block will return *Null*.

Read a QR-Code: Wait for the robot to see a QR Code and return its value in text format. If after ten seconds no tag has been seen, the block will return *Null*.

Speech reco.: Activates voice recognition, to detect words passed in parameter, separated by a semicolon. The block will return the recognized word.

Robot / Background

Autonomous Life: Enables or disables independent living, depending on the check mark.

Breath: Turns breath animation on or off in idle phrases, depending on the check mark.

Tools / File

Sound file: block to select an audio file *(WAV, MP3, OGG)* in the project files.

Animation: block to get an animation of the project.

Image file: a block used to select an image file *(PNG, JPG, BMP, GIF, SVG)* in the project files.

File: block allowing to select for any file in the files of the project.

Tools / Time

Wait: Pause the script with the value of the parameter, in seconds.

Timestamp: Returns a time stamp in seconds *(zero = 01/01/1970 00:00:00)*.

Tools / Log

Log detail, **Log info**, **Log warning** and **Log error**: Displays a message in the log panel. The color and the icon of the displayed text depends on the level.

Tools / Structure

Comment: Allows you to leave a comment in the script. Useful for rereading code. Does not execute any action.

My Friend NAO

The blocks in this section are for communication with the tablets of the My Friend NAO suite *(not included)*.

Display Text: Displays centered text, alone.

Display Image: Displays an image in full screen.

Clear: Clears the screen contents.

For each block, it is necessary to specify on which group of tablets the command must execute:

Group \ Tablets	Administrator	Leader	Player	Screen
All	✔	✔	✔	✔

Control	✔	✔	✖	✖
Without control	✖	✖	✔	✔
Public	✖	✔	✔	✔
Administrators	✔	✖	✖	✖
Leaders	✖	✔	✖	✖
Players	✖	✖	✔	✖
Screens	✖	✖	✖	✔
In game	✖	✔	✔	✖

Button Box

This section is dedicated to the interaction with the tool "Button box". This tool is a computer application, consisting of a defined number of buttons, images, input fields and text boxes. For more details, refer to the "Tools" chapter.

Display text: Shows text in the chosen position in the drop-down menu. Position # 1 is the title of the window. The value "Null" hides the line of text.

Display button: Displays a button, with the text in parameter, in the chosen position in the drop-down menu. The value "Null" hides the button.

Display input: Displays the input field, with the value in parameter. The value "Null" hides the field.

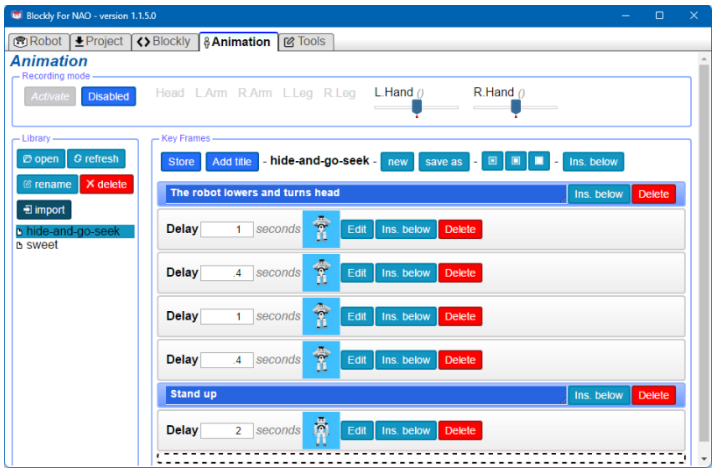
Display picture: Displays an image, in the chosen position in the drop-down menu. The value "Null" hides the image.

Clear: Hides all items in the button box.

Screen: Clears the screen and displays a title, an image and buttons, as the previous blocks would have done.

return screen: Waits for an interaction of the button box. If a button is clicked, the block returns the position of the button. If this is the validation of the input field, the block will return the entered text.

Animation screen



Recording mode part

In this part, you can enable and stop the recording features. Which includes:

- Control of touch engines, including "*Head*", "*Left Arm*", "*Right Arm*", "*Left Leg*" and "*Right Leg*" are the indicators *(green = you can move since motor are off)*.
- Finger motor control with "*Left Hand*" and "*Right Hand*" slides.
- Activation of the "*Key Frames*" part.

Upon activation of recording mode, the robot will become rigid. Its engines will activate to block the joints and maintain its position.

It is important to turn off autonomous life and breathing during recordings. You have the buttons on “Robot” page.

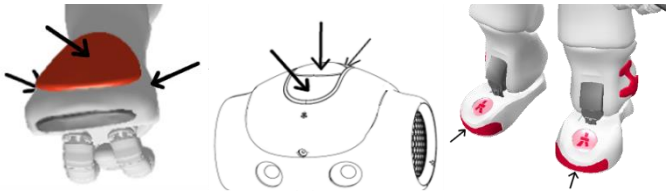
To move the arms *(wrist + elbow + shoulder)*, you have to hold your hand. In reality, you have to activate one of the two sensors on the side of the hands. The joints are flexible as long as these sensors are stimulated. The rear sensor of the hands is not considered. CAUTION: When releasing the hand sensors, continue to hold the arm as the reactivation of the motors is not instantaneous.

To move your fingers, you only need to use the "*Left Hand*" and "*Right Hand*" slides.

To move the head, keep your finger on the front sensor the head.

To move the leg, you just have to make a short push on the foot bumper. A first press to release the joints, a second press to block the engines. WARNING: The robot very quickly loses balance, so hold the robot by the torso.

Position of the sensors:



Library part

The list of animations saved in the project.
You can open, rename, and erase recorded animations.

CAUTION: If you rename an animation, you will have to rename the animation names in the Blockly script too.

To edit an animation, you must first select your animation, click **[open]**.
To copy an animation, simply open it and save it under a new name.

Key Frames part

This section consists of a button bar *(the first line)*, then the list of key positions that the robot will adopt. The latter is read from top to bottom and is itself composed of:

- Position lines with a gray background, a "*delay*" field and a thumbnail.
- Title lines, with a blue background. This element is purely visual. It allows you to leave notes or cut the animation in sections.
- A cursor, red and black blinking. Its role is to tell you where will be added the next posture of the robot. It is possible that the cursor is on an already existing line. In this case, this line will be overwritten the next time you add it.

The field "delay" indicates the time that must put the robot to reach the position, and thus create the movement.

The **[Store]** button copies the current position of the robot, and places it where the cursor is.
The **[Ins. Title]** adds a title to the cursor.
The **[new]** button resets the position list to zero, to create a new animation.

The **[save as]** is to save the position sequence to animation.

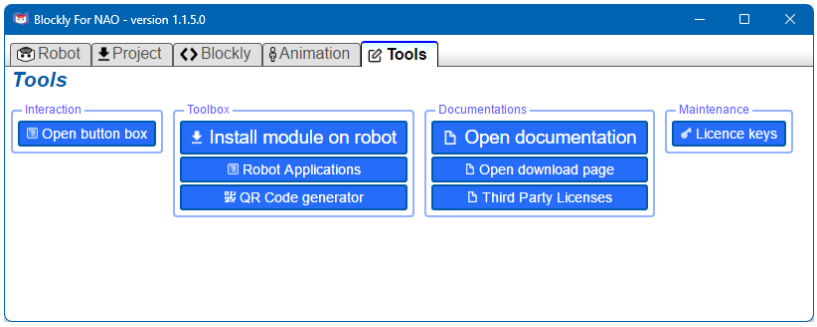
The **[Ins. below]** positions the cursor just below. If you click the button on the button bar, you will place the cursor at the very beginning of the list of postures.

The **[Edit]** button positions the cursor on the line. So, at the next "*Store*", this line will be replaced.

The **[Delete]** button removes the line.

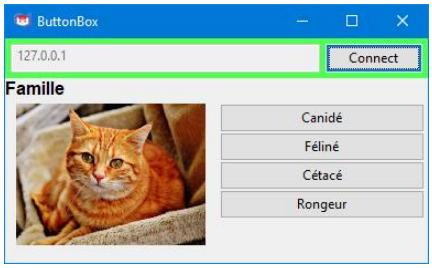
The cursor movement is automatic. As soon as you add a new position, the cursor will move below. If you are in modification, the cursor will remain in modification but for the following element *(except if there is a title or more nothing)*.

Tools screen



Button box

The button box is an application that allows a scheduled interaction with the Blockly script. It is the Blockly script that displays and hides the elements of the box. A block library is dedicated for this purpose.



The first line allows the connection to the robot. In the first field, you enter the IP address of the robot, then click on the **[Connect]** button. Click again on this button to disconnect. The color indicates the status of the connection: green = connected, red = disconnected.

The layout and the number of elements is predefined. There are:

- Ten text areas, from # 1 to # 10: The first is the title and is placed at the top. The others are located at the bottom.
- Two images, from # 1 to # 2, placed on the left, below the title and arranged vertically.
- Eight buttons, from # 1 to # 8, placed to the right of the buttons, below the title and arranged vertically.
- A text entry field, # 1, placed under the images and buttons

To display an element, you must give it some text that will be displayed *(and a path for the images)*. To make the element disappear, you pass it the value *Null*.

To validate a button, simply click on it.

To validate entered text, either press the Enter key on the keyboard, or click the **[>]** button on its right.

Texts and images are not interactive.

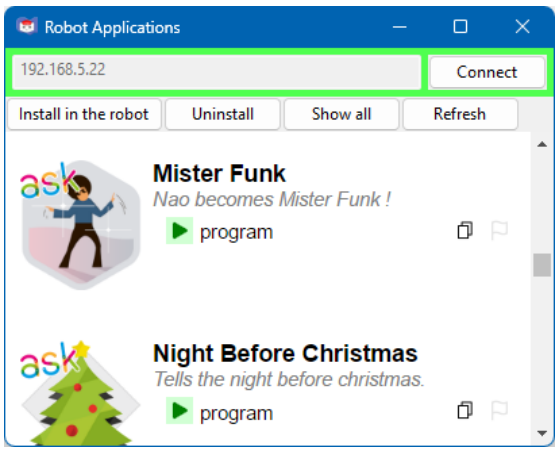
Install the module on the robot

See the dedicated chapter in the Installation section.

You can also use this utility to install other packages by clicking the **[add pkg]** button.

Robot Applications

Application manager installed in the robot.



The first line allows you to connect to the robot. In the first field, enter the robot's IP address, then click on the **[Connect]** button. Click this button again to disconnect. The color indicates the connection status: green=connected, red=disconnected.

By default, it displays only "Interactive" programs. To display other "library" or "service" programs, press the **[Show All]** button.

For each program, this tool lists all executables

The green **Play** ► button means the executable is off, and the button allows you to launch it.

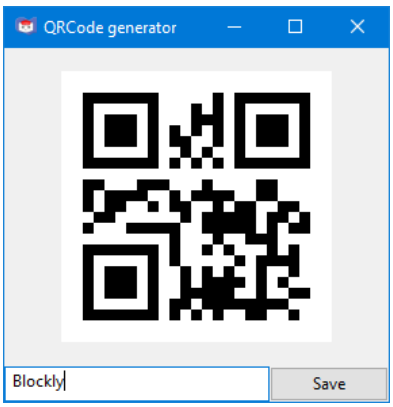
The red **Stop** ■ button means that the executable is started and the button turns it off.

The **Copy** button copies the full path of the executable to the clipboard.

The **Flag** button activates automatic launch of the executable when the robot is started. Caution: must be handled with care, by an informed person.

Creation of QR Code

Utility to create QR Code



As soon as you type the text in the dedicated field, the QR Code is drawn. By clicking the **[Save]** button, you save the QR Code in image format on the computer.

Open the documentation

Open this documentation.

Open the download page

Opens the download web page. Convenient to perform an update.

Third-party licenses

Displays the licenses for third-party applications / modules / libraries used in the design of this program.

Tips

Starting the robot:

The start of the NAO robot is not instantaneous. You must wait for the "ognak gnouk" and add a few extra seconds for all the services to be operational.

Know the IP address:

Once the robot is turned on, you briefly press the chest button. The robot will tell orally its state, as well as its address.

Launch of the program:

You can associate the ABLK files with the "Blockly for NAO.exe" program. This allows you to directly open the program by double-clicking ALBK project files.

Restart modules:

In case you modify the modules / services of the robot *(My Friend NAO, Blockly for NAO Lib, ...)* while using the Blockly program, you will be disconnected from the modules in question. By reconnecting, you will reconnect to all modules / services of the robot.

Technical information

Button box

For NAO developers: The button box uses ALMemory for communication. Here are the keys of the events used in the exchanges:

- Buttons display
  - `AskNAOTabletLib/btnboxEvent/btn1/label`
  - `AskNAOTabletLib/btnboxEvent/btn2/label`
  - `AskNAOTabletLib/btnboxEvent/btn3/label`
  - `AskNAOTabletLib/btnboxEvent/btn4/label`
  - `AskNAOTabletLib/btnboxEvent/btn5/label`
  - `AskNAOTabletLib/btnboxEvent/btn6/label`
  - `AskNAOTabletLib/btnboxEvent/btn7/label`
  - `AskNAOTabletLib/btnboxEvent/btn8/label`
- Display images
  - `AskNAOTabletLib/btnboxEvent/img1/data`
  - `AskNAOTabletLib/btnboxEvent/img2/data`
- Text posters
  - `AskNAOTabletLib/btnboxEvent/text1/label`
  - `AskNAOTabletLib/btnboxEvent/text2/label`
  - `AskNAOTabletLib/btnboxEvent/text3/label`
  - `AskNAOTabletLib/btnboxEvent/text4/label`
  - `AskNAOTabletLib/btnboxEvent/text5/label`
  - `AskNAOTabletLib/btnboxEvent/text6/label`
  - `AskNAOTabletLib/btnboxEvent/text7/label`
  - `AskNAOTabletLib/btnboxEvent/text8/label`
  - `AskNAOTabletLib/btnboxEvent/text9/label`
  - `AskNAOTabletLib/btnboxEvent/text10/label`
- Display of the input field
  - `AskNAOTabletLib/btnboxEvent/input1/force`
- Answer:
  - `AskNAOTabletLib/btnboxEvent/input1/entry` (*with the entered text in value*)
  - `AskNAOTabletLib/btnboxEvent/pushed` (*with the button number clicked in value*)

New name

Blockly for NAO software was previously named *AskNAO Blockly*.

To preserve a certain backward compatibility, this name may still be visible in addresses or technical data keys.

Changelog

Version 1.1.5 :

- Customizable execution speed, including turbo mode.
- Blockly for NAO and My Friend NAO name and logo changes.
- Button Box always visible when in use.
- Blocks added: - comment - text in number - mix list

Version 1.1.4

- Corrected operation without My Friend NAO (*AskNAO Tablet*) installed.

Version 1.1.3

- Improved readability of generated Python code
- Corrected management of sub-programs
- Improved script execution speed

Version 1.1.2

- Various corrections (*stability and display*)

Version 1.1.1

- Greek added to robot language selection

Version 1.1.0

- Code architectural redesign
- Python export added
- Added functionality to install script in robot
- Added tool for managing applications installed in the robot
- Japanese translation added