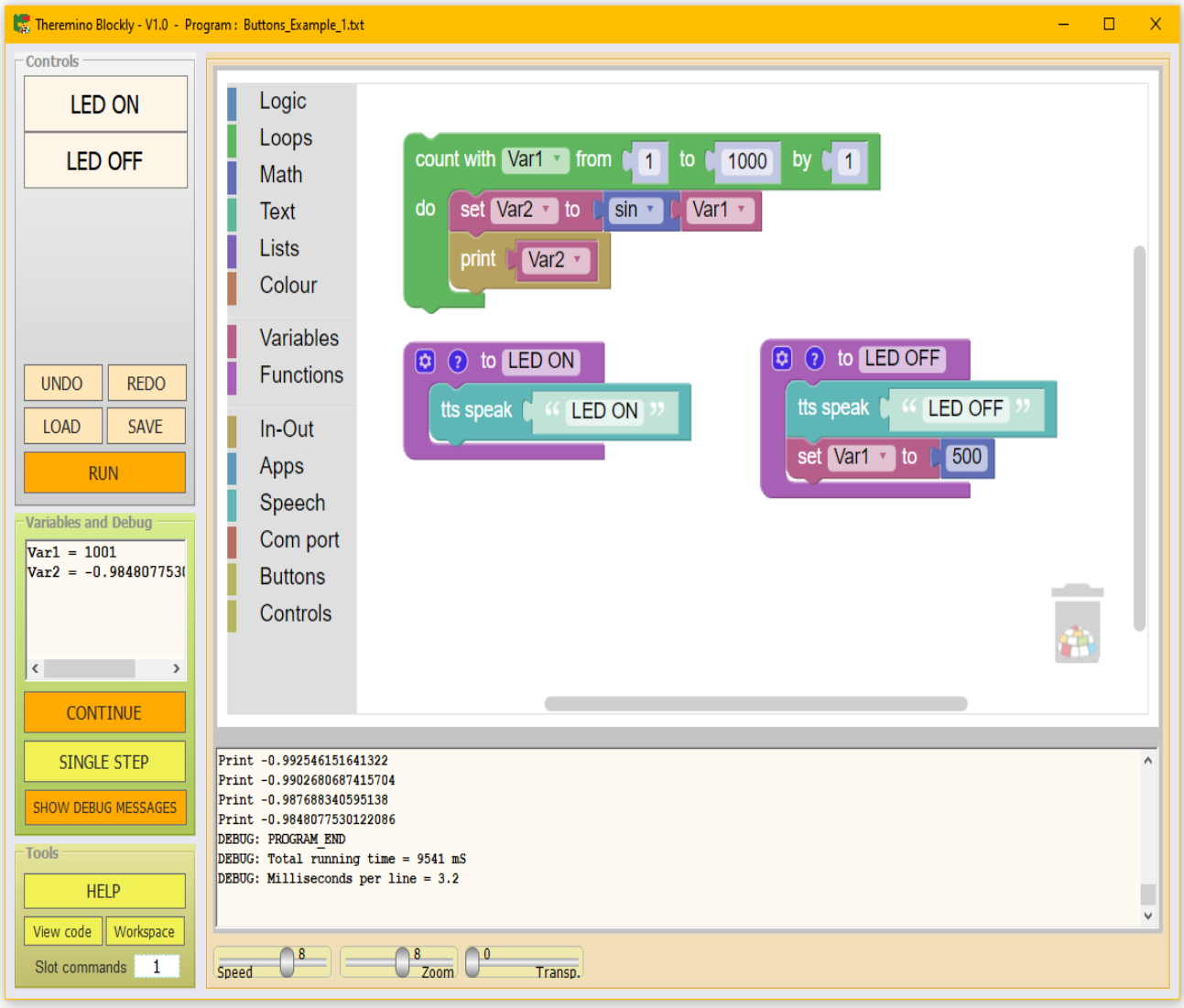
theremino **System**



Theremino Blockly V1.0

Page index

Premises

With **Blockly** you learn to program without having to study command syntax.

Connecting blocks is easy and intuitive ***even for the little ones*** and the program is created automatically.

Theremino_Blockly is the younger brother of ***Theremino_Automation***, it's simpler but still can access all the main functions of our system, from simple **input-output** to the **synthesis** and **vocal recognition,** to the control of **collaborative robots**, to the **ionizing radiations and Radon meters**, to the control of **cardiac arrhythmias**, to **chemicals analyses**, to measures with **oscilloscopes** and **spectrum analyzers**, etc...

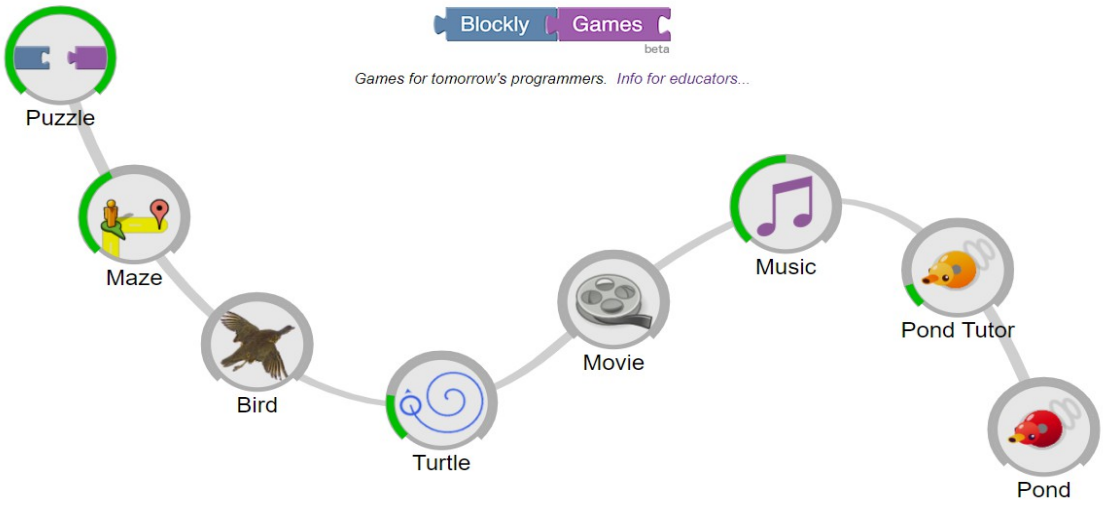
In ***This Page*** find an index of the (almost two hundred) applications of our system, all of them **free** and **Open Source**.

First steps with Blockly

Blockly was created by **Google** and teaching units, examples, and little games are available to help you learn its fundamentals.

Connecting blocks for beginners
https://www.ivana.it/jm/software-on-line/358-blockly

Games
https://blockly.games
https://www.brainpop.com/games/blocklymaze
https://blockly.games/music



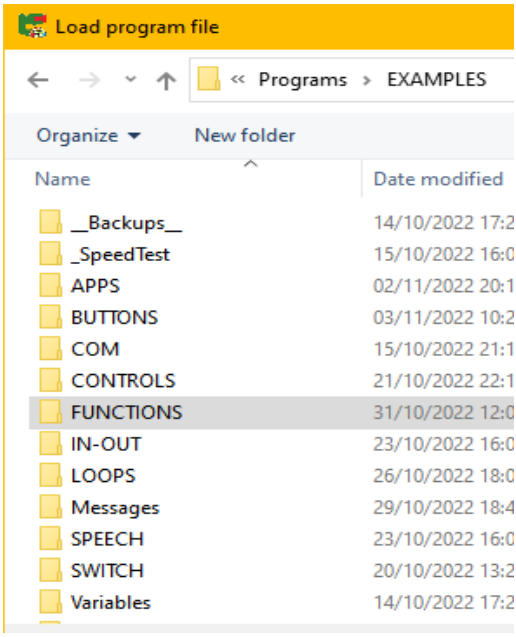
Load and run programs

The button **LOAD** opens a window with sample programs to choose from.

After loading a program you run it with the button **RUN**, or by starting a function with the buttons located in the upper area.

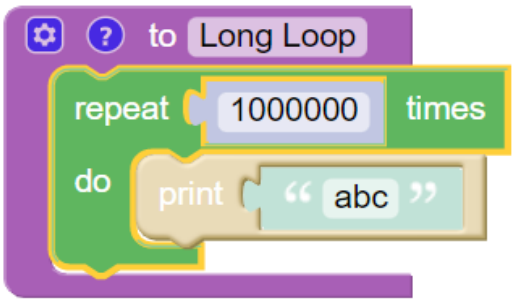
To stop the program, press the key **RUN** (which became **STOP**).

The **STOP** happens automatically when you click on the blocks area to edit them.



Simple programs to get started

We recommend starting with the simple examples found in the folder **Programs**. Run the programs to see what they do. Try moving the blocks with the mouse, adding others by taking them from the vertical menu on the left and finally restoring the original program with **UNDO** and the **recycle bin**.



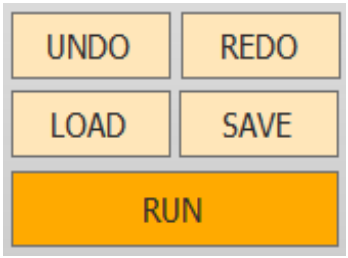
All the sample files are double, for example **FileName_1** and **FileName_2,** so you can make changes and tests, with less risk of losing the original examples.

However, remember tosave the program under a new name before starting to edit it.However if necessary you can restore all the original examples from files you download from the ***Blockly page***.

Edit and save programs

Every change you make to the program is automatically written to the file, so there's no need to remember to save your work before closing the application **Blockly**, or to load another program.

So to avoid modifying the examples, **before you start making changes** it is advisable to save the program with a new name, using the key **SAVE**.



- Pressing **SAVE** with the **left mouse button**, the program is saved with a new name.
- Pressing **LOAD** with the **left mouse button**, a window opens with many example programs to choose from.

- Pressing **SAVE** with the **right mouse button**, the program is quickly saved.
- Pressing **LOAD** with the **right mouse button**, you save the program and then immediately reload it.
- The button **UNDO** it is used to go back when changes have been made to the program and you want to eliminate them.
- The button **REDO** rebuilds discarded changes with **UNDO**.

Instead of **UNDO** and **REDO** you can also use key combinations **CTRL-Z** and **CTRL-Y** (press **CTRL** and holding it down press **Z** or **Y**).

- - - - -

When you reopen the program it will be exactly as you left it.
If you close the program in RUN it will restart with RUN active.

And if the program consists of only functions then
the first function runs when you start the program.

The block area

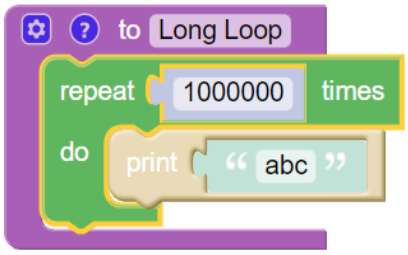
The operation of **Blockly** it's intuitive the best way to figure it out is to try it.

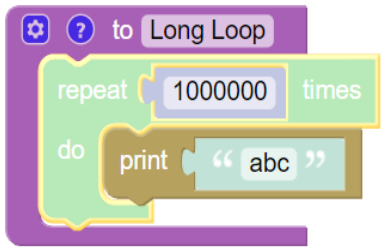
Load and try the examples that are in the folder **Programs**

See also the documentation, examples and games created by **Google**, which we indicated at the beginning of this document.

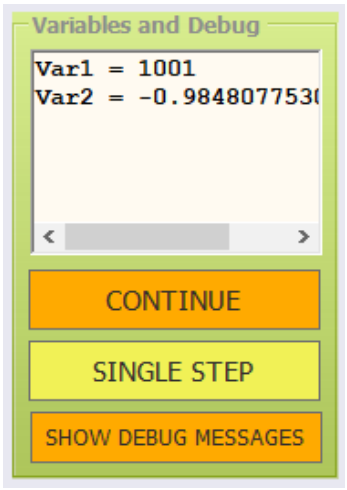
Program execution

During execution the blocks lighten, as seen in the green block of these two images. To see them better, lower the speed to 4 or even slower.

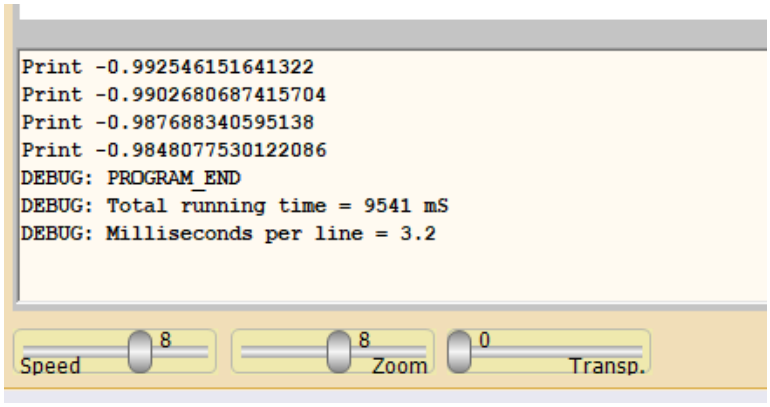




During rehearsals it is useful to keep the panel open **Variables and debugging** and enable the button **Show debug messages**.

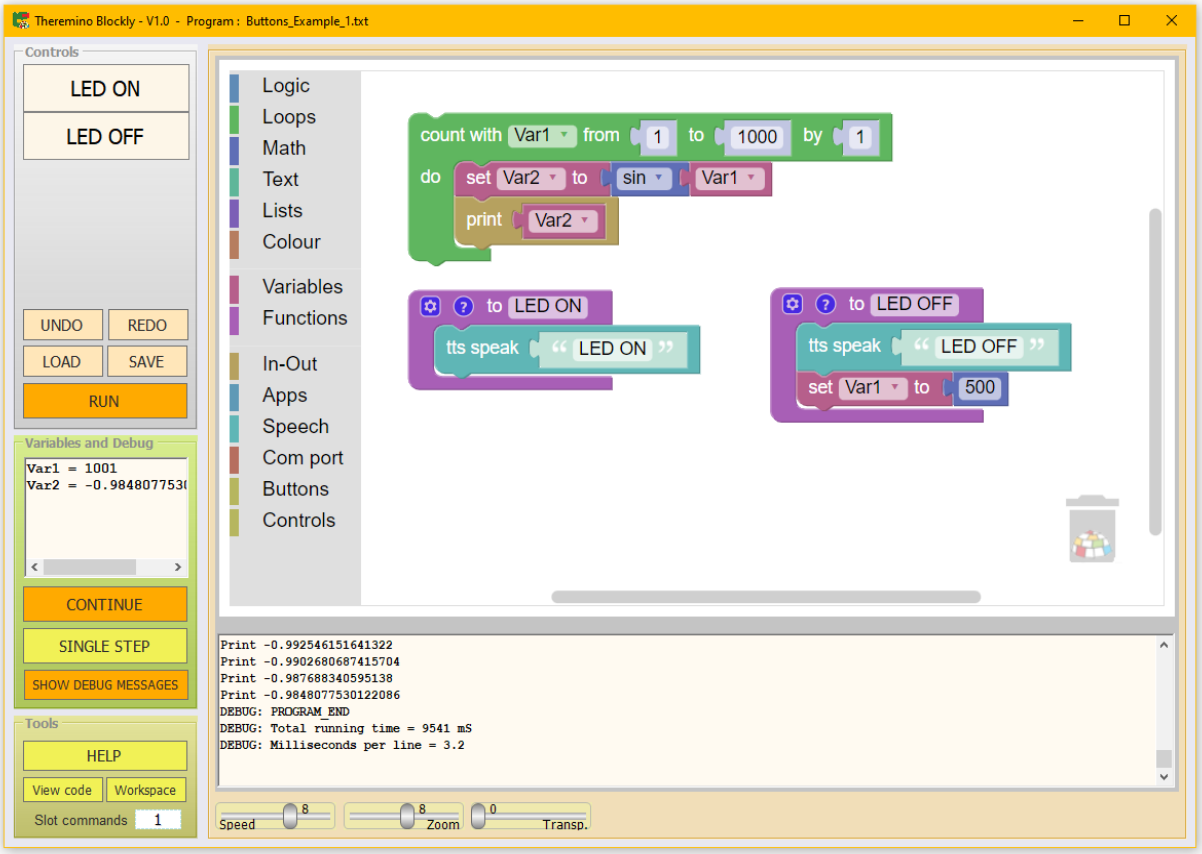


This way in the lower text area, in addition to the text sent with **PRINT**, you will also see service information preceded by the prefix **DEBUG**.



The details of the operation of these buttons are explained on the operation page [Variables and Debug Panel](#) .

Program structure



The programs are composed of functions (in purple color in this image) and each function brings up a button to execute it.

Some sections may also be outside of any function, such as the part enclosed in green in this image.

Program parts outside functions are executed (top to bottom) when RUN is pressed and also when functions are started.

To prevent functions from also executing external program zones, it is recommended to enclose these zones within a function. Then you will call these functions from the program, or with the buttons.

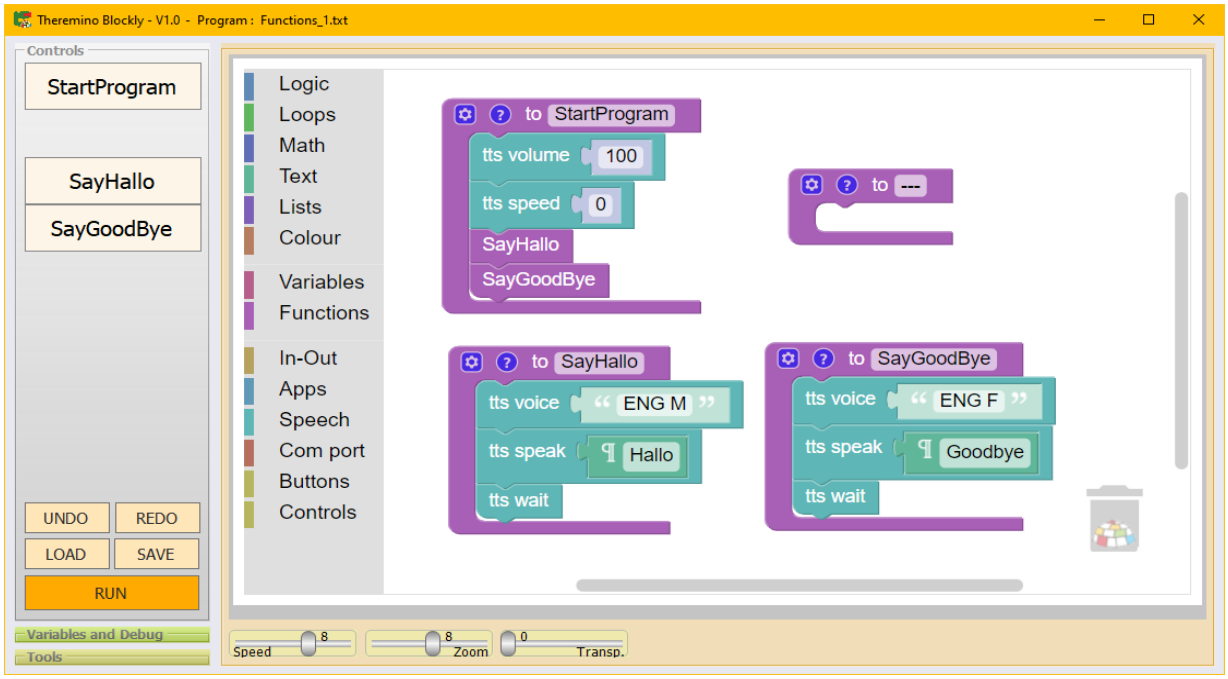
If you don't want functions to create a button, just add a non-alphabetic character to the beginning of their name. To see an example, open the program: **EXAMPLES\SPEECH\TTS_Example_1.txt**

If the program consists of only functions then **the first function is executed when you start the program with RUN.**

The function buttons

Function buttons appear in the Controls panel.

In this picture the buttons are **StartProgram**, **SayHello** and **SayGoodbye**.



Using the menu **Functions** which is in the middle of the **Blockly Tools-bar** you can create functions and give them a meaningful name.

When you create function blocks, buttons to execute them automatically appear in the Controls panel**.**

When you change the function block names, the control buttons are automatically updated.

To improve the visibility of the text in the buttons you can break the text into two lines. They add up **two consecutive spaces** at an appropriate point in the function name and the text breaks at that point.

All other spaces must be single, otherwise the text would become three lines and some words would disappear at the bottom.

If you don't want a function to create a button, add them non-alphabetic characters (eg **__** or **---**) at the beginning of its name.

If the program consists of only functions then **the first function is also executed when starting the program with RUN.**

Buttons and functions

The order of the buttons follows the order of the blocks (top to bottom and left to right) so if you want to swap buttons just move the blocks.

You can also leave blank spaces between buttons to group buttons that perform similar functions. To leave blanks, add non-alphabetic characters to the beginning of the function name, for example **__** or **---** , and then you move the blocks so as to bring the empty places to the right positions.

Function button numbering

Some blocks (**disable button, enable button, button colour, button slots** and **button text**) as well as locating buttons by their identifier (the function name of **Blockly**), can also identify them with a number.

Buttons are numbered from top to bottom. The first column of buttons goes from 1 to 16. If there are more than sixteen buttons, the second column goes from 16 to 32, the third from 33 to 48 and so on up to 128.

If the order of the blocks, and therefore of the buttons, is changed, the instructions that identify the buttons according to their number must also be checked and possibly modified. For this reason it is preferable to specify buttons with text and not with a number.

The only time it might be better to use a number is when you want to do the same thing on many buttons, for example use a loop that increments a numeric variable to disable or color them.

Execution of functions

By pressing the button relating to a function, all other operations are interrupted and the function is performed until the end. When the function ends, execution resumes where it left off.

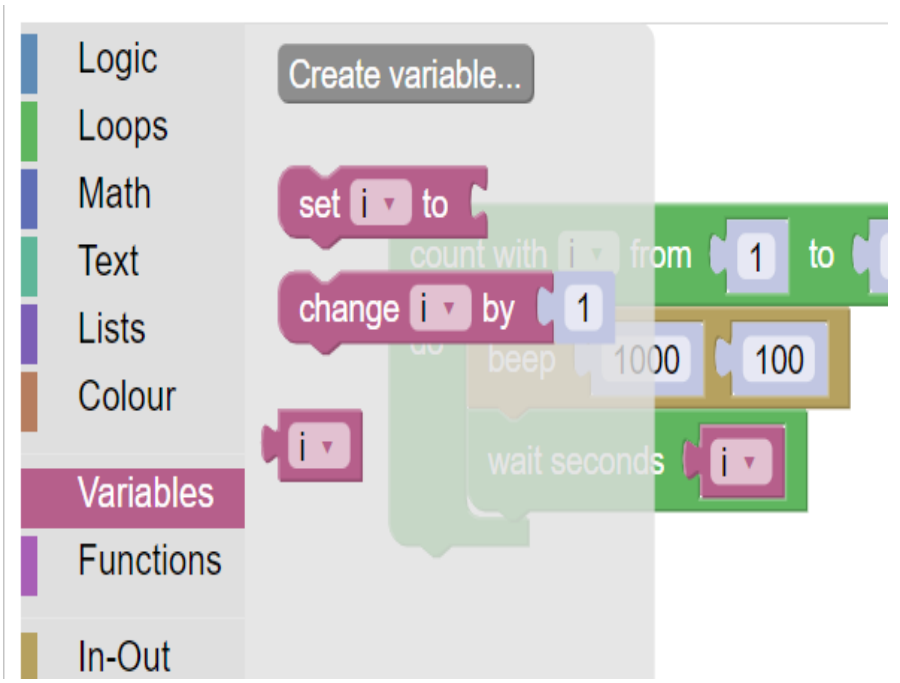
The functions can also be performed by the program.

Functions that run programmatically shouldn'tcreate a button, then add the characters (**__** or **---**) at the beginning of its name.

If the program consists of only functions then **the first function is also executed when starting the program with RUN.**

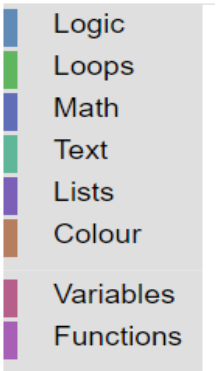
Block menu

By clicking on the menu on the left, you can choose from over one hundred different blocks, divided by topics and colours.



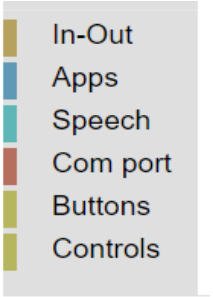
The first entries from **Logic** to **Functions** contain the building blocks of language.

The instructions of these blocks are similar to the instructions used in the most famous programming languages, such as: VisualBasic, CSharp, Java, C++, Pascal, Python, etc...

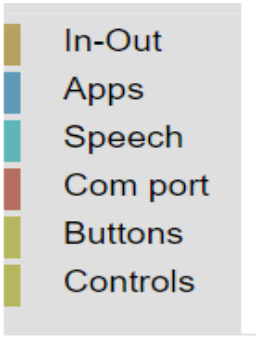


The six entries below from **In-Out** to **Controls** contain specific blocks, for communicate with the applications of the theremino system and therefore also with the outside of the PC.

Detailed explanations of these instructions are on the next pages.



Menu for external communications

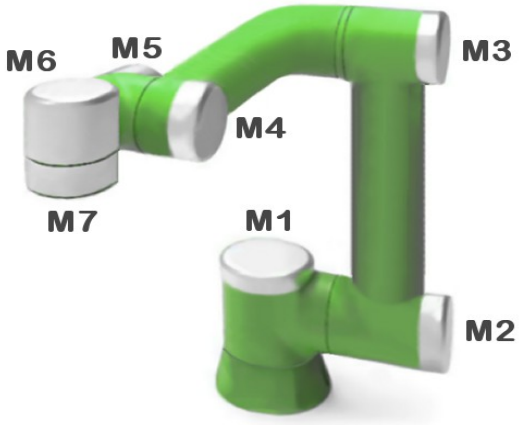


The last six items of the block menu contain specific instructions, to communicate with the applications of the theremino system and with the outside of the PC.

Communicating in real time is important, all living beings do it, but in general today's computers, despite being very fast, neglect these aspects.

The applications of our system work locally on a PC that we use as ***PLC***.

In our system the PC itself directly controls the sensors and motors, eliminating much of the otherwise costly and complex control hardware.



Here are some examples of what can be done with our free and Open Source software and with a few tens of dollars worth of engines.

Video1.mp4

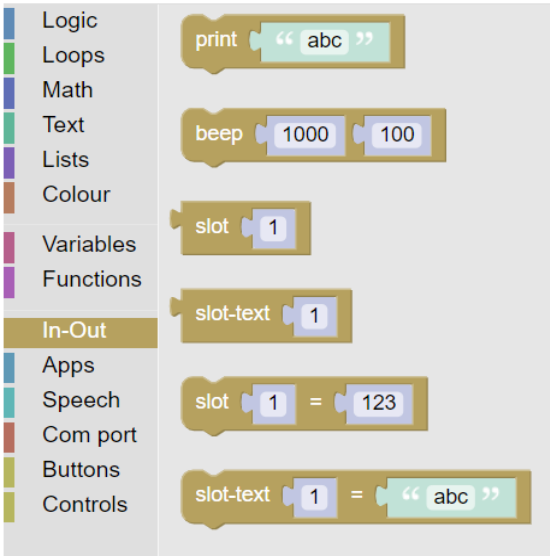
Video2.mp4

Video3.mp4

Video4.mp4

Web pages could not react with adequate response times for sensors and robotics (from hundredths to thousandths of a second)
therefore all the applications of our system work locally on a PC.

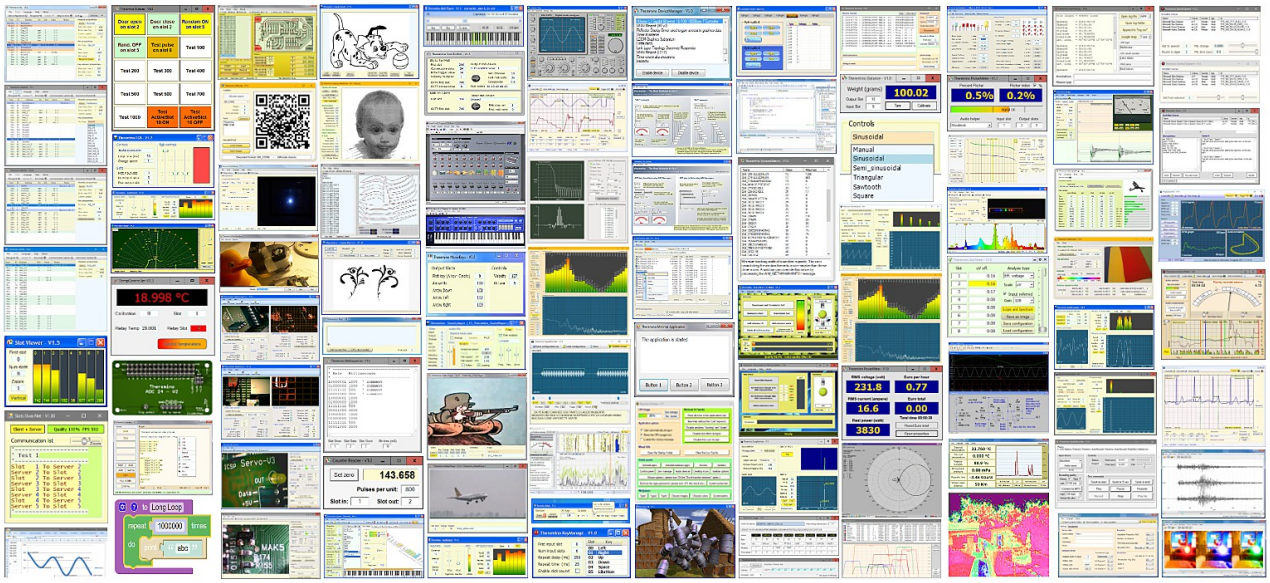
In-Out menu



The blocks in this section communicate with the applications of our system, but also with the sensors, actuators and humans outside the PC.

Communicating through the ***Numeric slots*** and the ***Text slots***, which are an exclusive invention of our system complex applications can be built even without being an expert in programming and electronics.

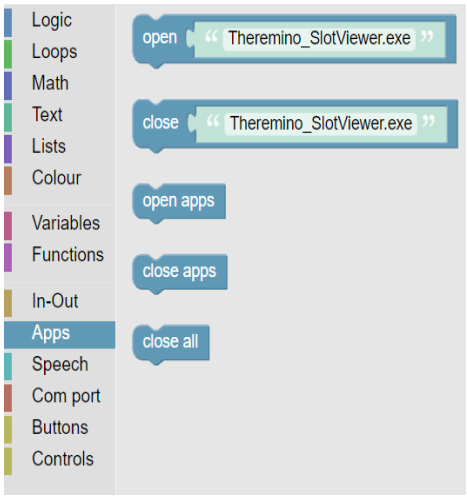
Through the ***Numeric Slots*** and the ***SlotText*** you communicate with numerous applications of our system written specifically to facilitate the operations of ***Input-Output***, this picture shows the most important ones.



Each of these applications takes care of a specific task. It would not be possible to do all these operations in Blockly and not even with the most advanced languages, because the complexity would become unmanageable. But by dividing the tasks over several applications, it is possible to build very complex systems that are robust and reliable at the same time.

All this **software,** completely **free** and **OpenSource**,
is the result of over ten years of continuous work by ***our team***.
To download them and for operating details open ***this page***.

Apps menu - Open files



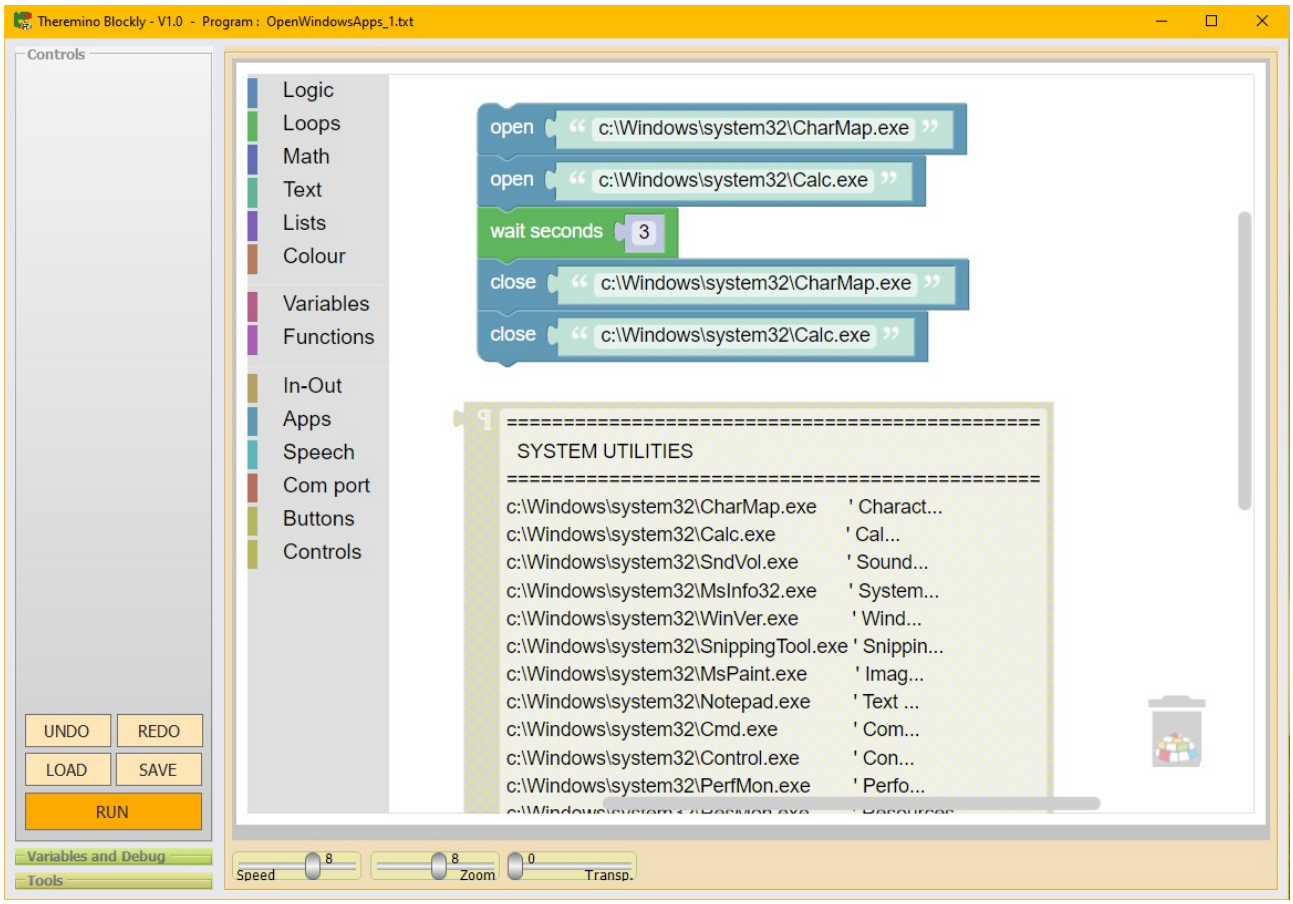
These examples open some applications of the Windows system.

```
Open "C:\Windows\Notepad.exe"

Open "C:\Windows\system32\Calc.exe"

Open "C:\Windows\system32\mspaint.exe"
```

The following image, taken from the "OpenWindowsApps" example, shows what the "Load" blocks look like.

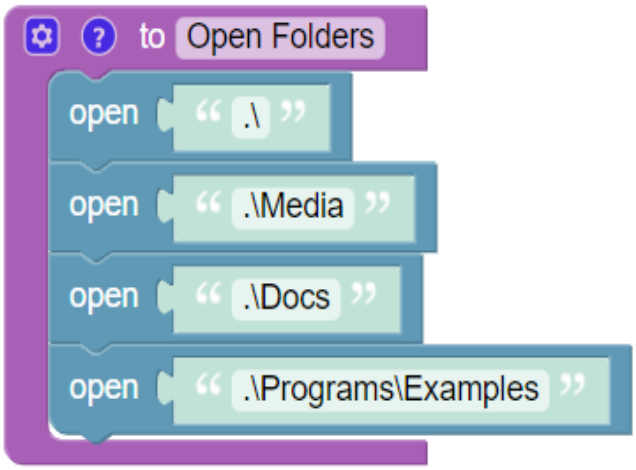


- - - - -

Experiment with the examples found in:
Programs\EXAMPLES\APPS

Apps menu - Open folders

These examples open some folders of the application **Theremino_Blockly**. Note that the dot plus a backslash indicates the root folder of the application **Blockly**, i.e. the folder of the file **Theremino_Blockly.exe**




```
Open ".\"
Open ".\Files"
Open ".\Media"
```

To indicate folders and subfolders of the application **Blockly** the dot and the slash can be omitted, so the previous three examples can simply become:

```
Open "" Open "Files" Open "Average"
```

The double point is used to indicate the parent folder. For example to open the folder that contains the file folder **Theremino_Blockly.exe** you can write:

```
Open ".."
```

The next examples open the C drive and some folders of the Windows system.

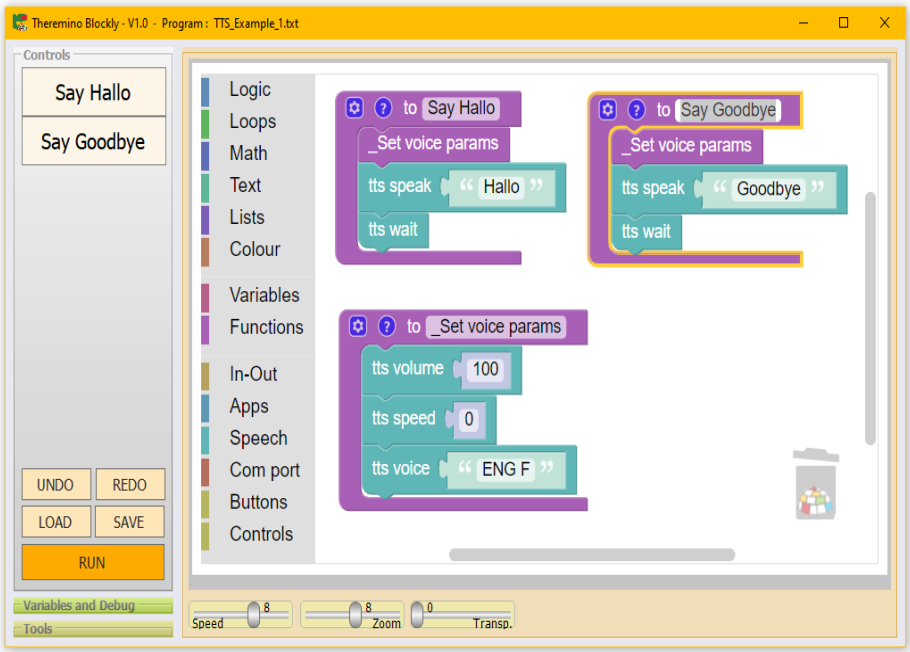
```
Open "C:\"
Open "C:\windows"
Open "C:\windows\system32"
```

- - - - -

Experiment with the examples found in:
Programs\EXAMPLES\APPS

Speech menu - Speech synthesis

The examples in the folder **Programs\EXAMPLES\SPEECH** show all the commands you use for speech synthesis **TTS extension** (**Text To Speech**).



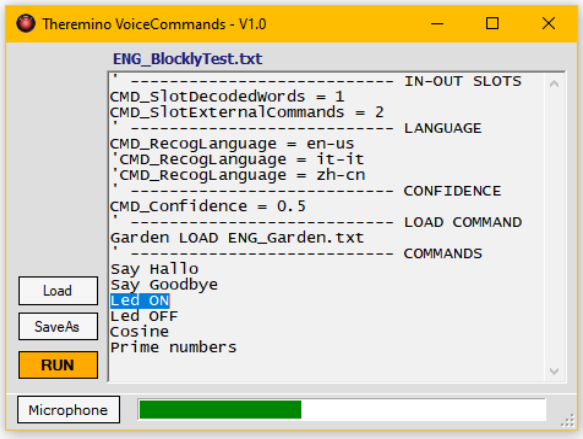
The available languages depend on Windows and you set them with its language settings panel.

Voice commands

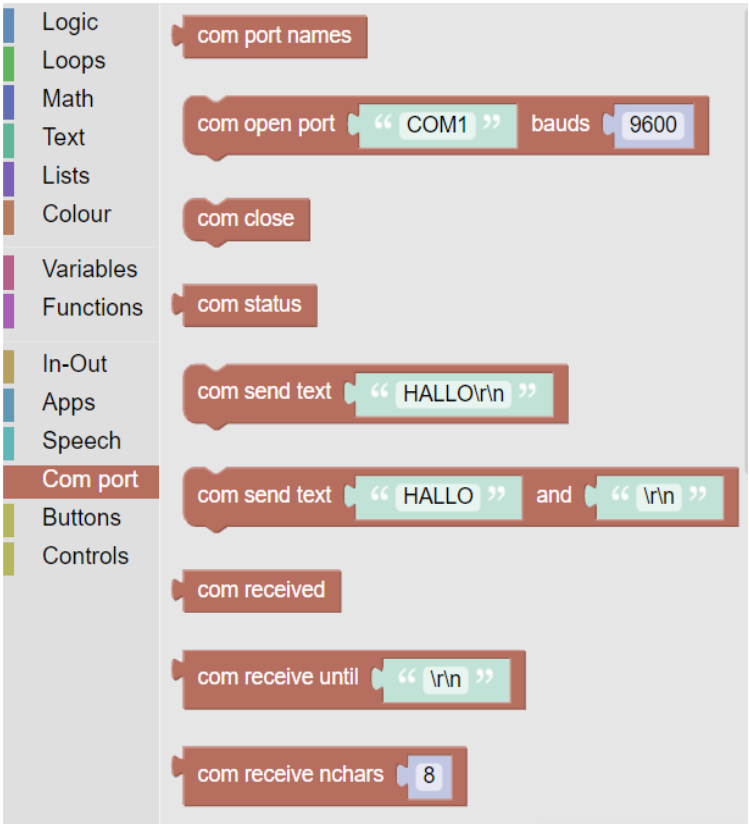
For voice commands you use the application **VoiceCommands** which you download from ***this page***.

The names of the ***external commands*** to be executed are received in the **Blockly SlotText** of commands, so you have to set the same **SlotText** both in **Blockly** that in **VoiceCommands**.

In turn **Blockly** can send to the app. **VoiceCommands** the following commands: **Run**, **Stop** and **Load FileName**, and also in this case the transmission and reception **SlotText** must match.



Com port menu - Serial ports



By means of the serial port many sensors can be controlled, for example scales. or GPS systems, or actuators, such as smart motors.

In some cases communicating via serial can be very complex, so before writing long and difficult programs, check in ***this page***.

In our system there are already many convenient applications for all the most common cases.

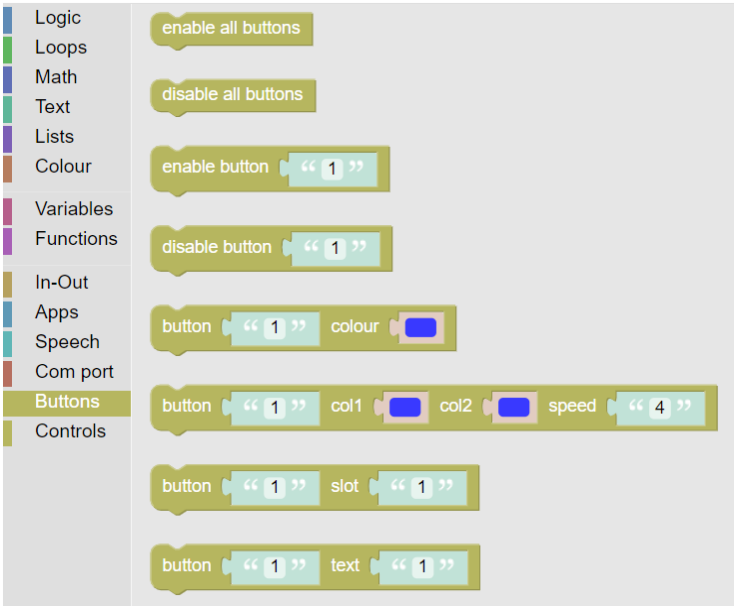
Special characters are used when sending text to wrap on a new line **CR** (carriage return) and **NL** (new line). But since they could be part of words like **CR**omium or o**NL**us, replace them with the sequences **\r** and **\n**.

Generally in the Windows system both are used and then written **\r\n** but some systems or sensors may use the only **CR**, or only **NL**.

Experiment with the examples found in the folder:
Programs\EXAMPLES\COM

It might be useful for communication tests
the application ***Theremino_Terminal***.

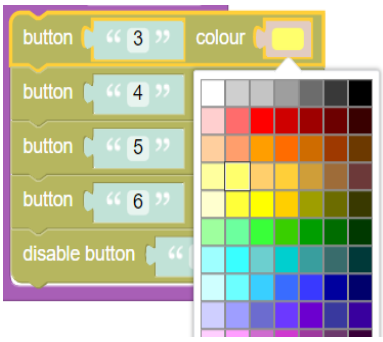
Menu Buttons - Control buttons



These blocks act on the control buttons to:

- Enable them
- Disable them
- Color them
- Make them flash
- Associate them with a Slot

Coloring the buttons can make using the program easier and more intuitive. You can also color the buttons differently running or make them flash.



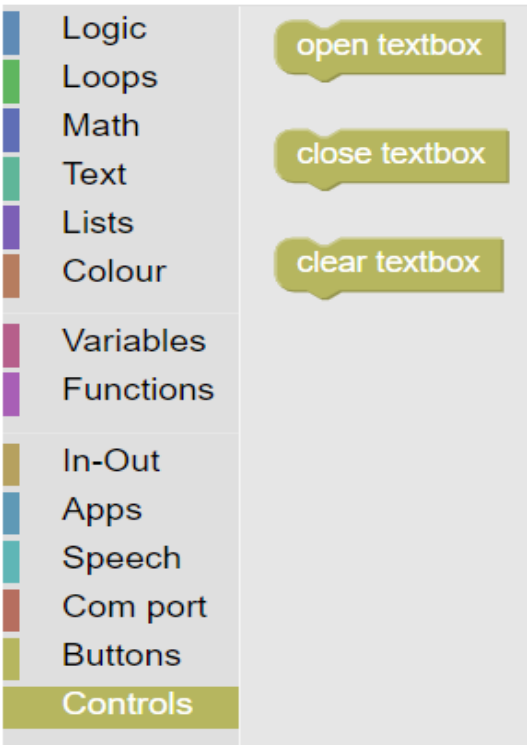
Disabling the running buttons can be useful to prevent them from being pressed twice by accident.

Binding Buttons to Slots opens up great possibilities. Functions can be performed based on external conditions, for example when a distance sensor detects the presence of a human in the work area. Or when a temperature exceeds a certain level.

The function associated with the button, and therefore with the Slot, is performed when the value of the Slot exceeds the value 500 (half of the range of values that in our system the values range from 0 to 1000) and is no longer performed until the value of the Slot returns below 500 and exceeds it again.

Experiment with the examples found in the folder:
Programs\EXAMPLES\BUTTONS

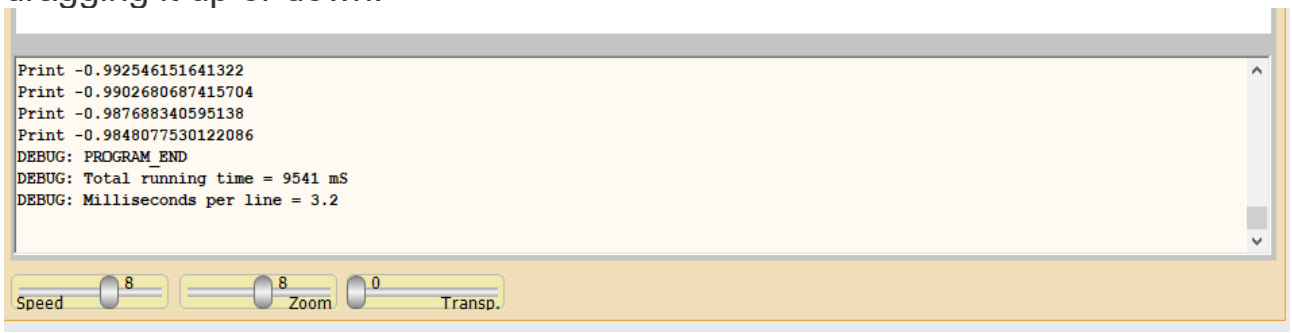
Controls menu - View controls



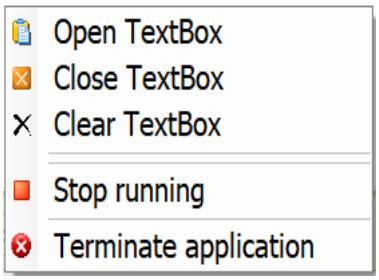
We will probably add more controls to this section in the future but currently the only controls are a large horizontal text box.

With these commands the program itselfcan open the text box (**open textbox**), close it (**close textbox**) and delete all text that contains (**clear text box**).

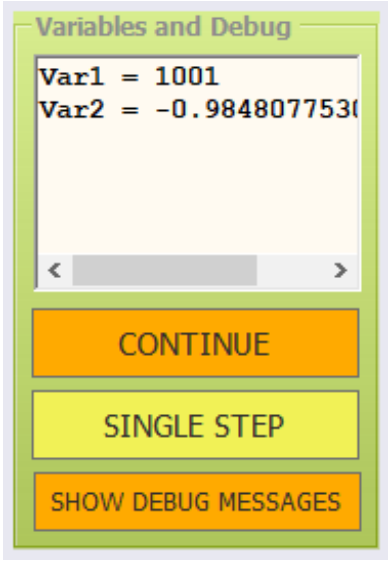
The text box can also be operated manually by pressing the left mouse button on its upper gray line and dragging it up or down.



Or you can right-click the text box and use the menu items.



The "Variables and Debug" panel



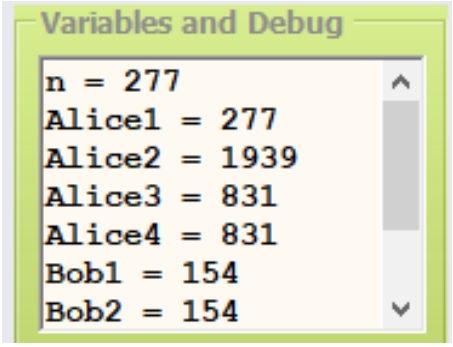
The controls on this panel make it easy to check and set up the software.

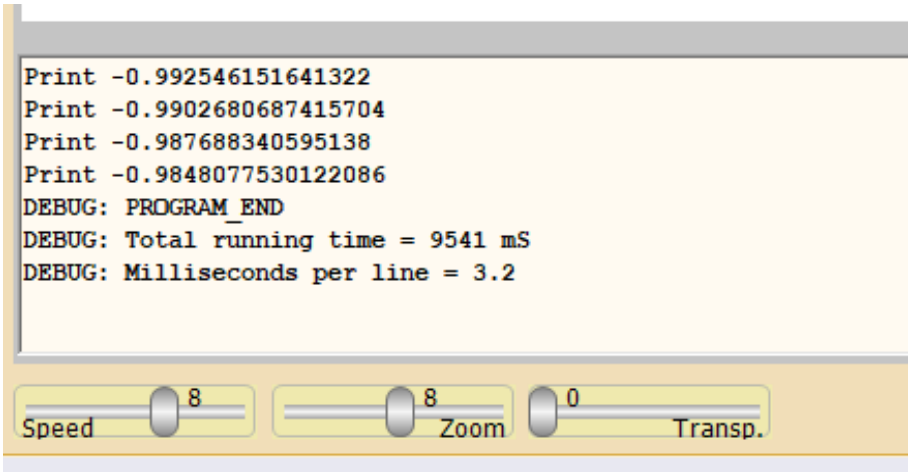
- **CONTINUE** When the program is paused, this button restarts it.
- **SINGLE-STEP** When the program is paused, it can be run one line at a time.
- **SHOW DEBUG** If enabled (orange) it shows the service information in the lower text area.

You can also use a ***external command*** to pause the application and then do it step by step.

Check the functioning of the programs

If the button **Show debug messages** is enabled (orange) and the panel **Variables and debugging** is visible, then during program execution the values of the variables and all commands of **debugging** they comecontinuously updated.

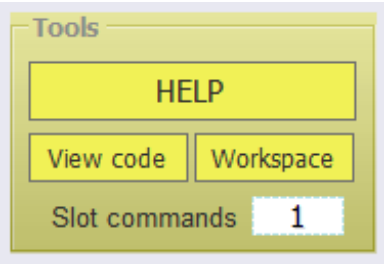




It is therefore possible to reduce the speed by lowering the slider **Speed**, or run the program one line at a time with **Single step**, to explore the details of the operation, improve the program and eliminate errors.

The "Tools" panel

This panel contains service commands, useful for learning and for sending commands from external applications.



- **HELP** opens the folder that contains the documentation in various languages.
- **View-code** displays program code in Java Script format.
- **Workspace** displays program blocks in XML format, which is the format chosen by Google to display **Blockly** on the web pages.
- **Slot commands** it is used to set the text slot to be used to send commands to the application **Blockly** from other applications of our system.To disable external commands, set to -1. Otherwise it is set with a number from 1 to 999.

Execution of external commands

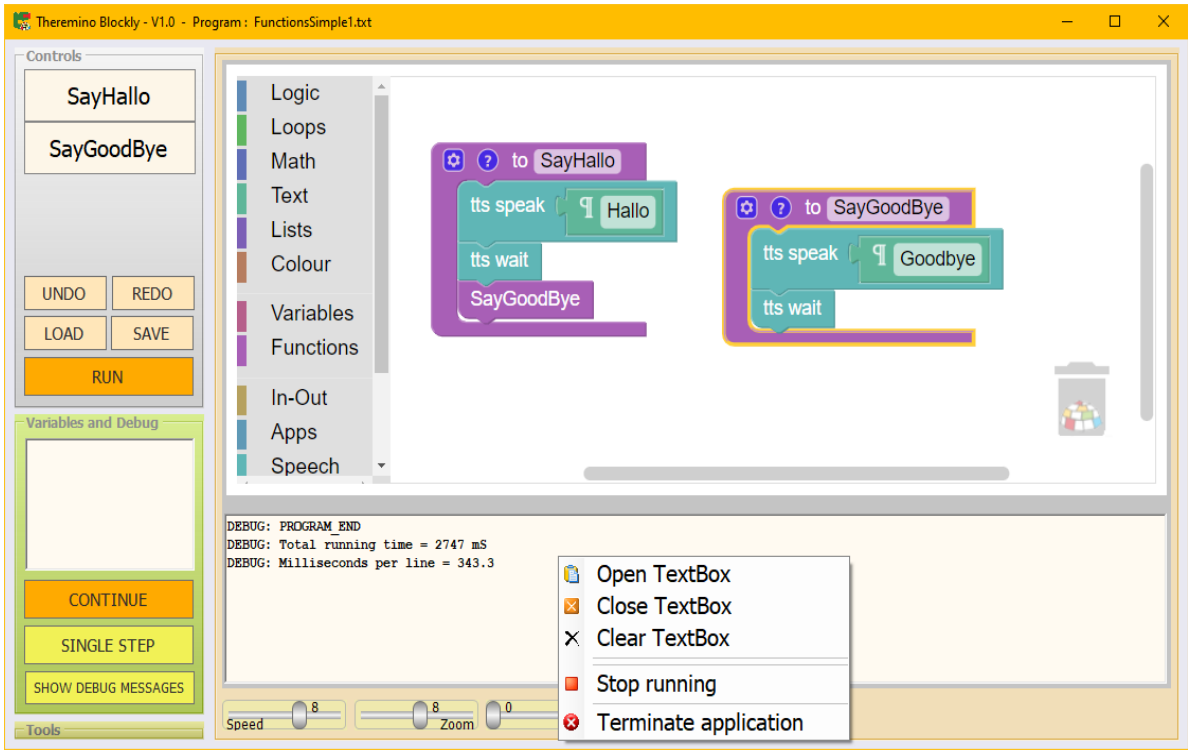
The text slot to be used for external commands is set as explained above.

Incoming commands from other applications can be **RUN**, **STOP**, **PAUSE**, **STEP**, **CONTINUE** and also all **function names by Blockly**.

The commands are not case sensitive.

The application menu

By clicking with the right mouse button (or by touching the touch screen without releasing your finger for two seconds), the menu shown below opens.



- **Open TextBox** opens the lower horizontal area, where you write with the PRINT instructions and where you send the service information (DEBUG) if the "Show debug messages" button is enabled (orange).

- **Close TextBox** closes the lower horizontal area.

- **Clear TextBox** delete all text from the lower horizontal area.

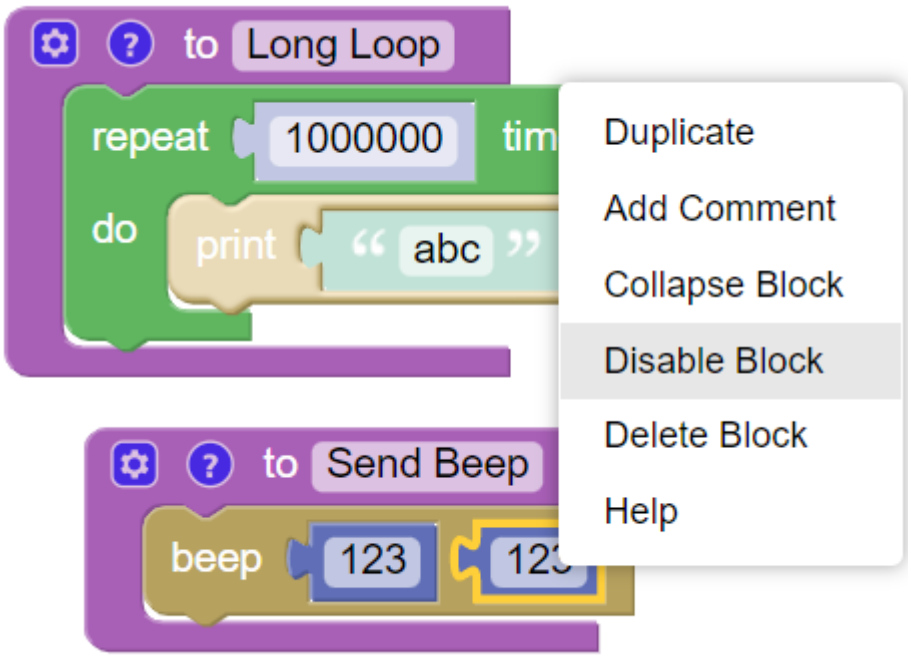
- **Stop running** and **Finish application** they stop the execution of the program and close the application **Blockly**.

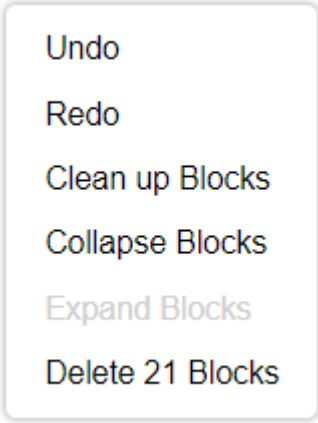
To change the height of the horizontal text area
use the left mouse button and drag up and down
the horizontal gray line that divides the text area from the block area.

There are also three blocks, in the **Controls** menu,
to open the text area, close it and delete the text it contains.

The menus of the blocks area

By clicking with the right mouse button on one of the blocks, a menu appears which allows you to duplicate the blocks, reduce them, disable them, enable them, delete them and also open web pages with information on their functioning.





By clicking instead in the white area between the blocks, another menu appears which acts on all the blocks, to rearrange them, reduce their size to a minimum, expand and eliminate them.

The trash

To eliminate the blocks, drag them with the mouse in the left vertical area, or use the menu items, or drag them over to the trash can.

In all cases it will be possible to recover the deleted blocks by clicking on the basket and moving them to the work area.

Alternatively you could recover the blocks using the button **UNDO** explained in the first pages of this document.

The bottom bar controls

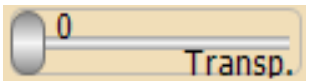
The cursor **SPEED** adjust the execution speed.

Speeds range from”1” (about three lines per second), to ”8” (five hundred lines per second), and up to ”12” (three thousand lines per second on fairly fast computers).

While writing the program it is good to use a medium speed. Speed usually”4” (20 lines per second), which is slow enough that you can visually follow the execution of the program.

Then, during the execution of the programs, it is advisable to use the speed "8", which uses little the CPU of the computer, and the higher speeds only if you have to execute particularly complex programs and which need to control hardware in real time, for example example for collaborative robots.

The cursor **ZOOM** sets the size of the text, both in the program window and in the service boxes.



This slider adjusts the transparency of the main window and allows you to see underneath it as well.

Run multiple instances of Blockly

If you try to launch a second instance of **Blockly** from the same file, it will not open. This behavior is intended to prevent opening two by mistake and thus prevent the options of the two instances from mixing. If necessary, you can force open a second instance by holding down **SHIFT** during startup.

If you want two or more copies of **Blockly** that work simultaneously,each with independent options, just copy the file **Theremino_Blockly.exe** with a different name.

You can keep the files if you prefer **Theremino_Blockly.exe** in separate folders or even all in the same folder.

Limits of Theremino Blockly

In **Blockly** the instructions are limited to the essentials, so for complex tasks one has to switch to more powerful programming environments, for example **Theremino_Automation** and, in some particularly difficult cases, also to languages **CSharp** and **VbNet** (with **Visual Studio**).

As a guideline you can count the blocks, if they become more than a hundred or two hundred it is good to switch to **Theremino Automation** and then, beyond a thousand or two thousand lines it is better to switch to **Visual Studio**.

Execution speed

In "interpreted" languages, unlike "compiled" ones, the instructions are read, character by character, and interpreted during execution itself. The execution of an interpreted language is therefore slower and moreover party **Blockly** was written by Google in **JavaScripts** which is slower than **dotnet**.

Therefore **Theremino-Blockly** is about ten times slower than **Theremino-Automation**, which itself is ten times slower than **Visual Studio** (**VbNet**, **CSharp**, **C++**).

However, speed is super-abundant for the tasks these languages are intended for. In fact we had to add the possibility of slowing them down, even thousands of times, to make it easier to understand what is happening.

Technical documentation

The application **Theremino_Blockly** is written in **DotNet** but it uses the code of **Blockly** he was born in **JS-Interpreter** which are written in **JavaScript.**

The blocks part runs in a control **WebView2** Supported by **Microsoft Edge.**

All necessary files are already present both in **Windows 10** that in **Windows 11**, without installation and without Internet.

- - - - -

Any developers who want to expand **Theremino_Blockly** with new blocks will have to add sections in the following files:

blockyToolbox.xml

- This file contains the definition part of the blocks to display them in the HTML page

theremino_custom_blocks.js

- The first half of this file contains block execution functions that generate calls to the DotNet application.
- The second half of the file contains block characteristic definitions.

Some special blocks may need additional functions or modifications in the file as well **theremino.js** and in some cases even in the **Module_Blockly.vb**

To learn more about these topics, consult the documentation that we have prepared in the file **References.rtf** located in the folder **Docs** which contains this same file of **HELP**.

- - - - -

Anyone wishing to update **Blockly** will have to replace **blockly_compressed.js**, **blocks_compressed.js** and **javascript_compressed.js** with the latest versions that can be downloaded from here: ***https://github.com/google/blockly/releases***

After replacing these files in the folder **Blockly\Files\JS** there is a good chance that our application is working badly or partially. In this case an expert programmer will have to use Visual Studio Community 2022, fix the problems and recompile the application.