

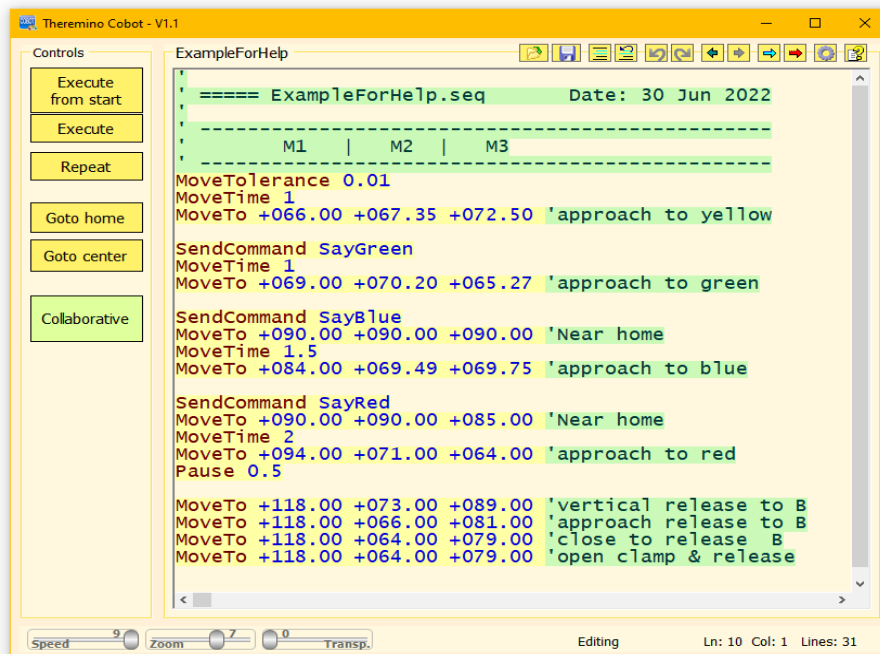
theremino
•the•real•modular•in-out•

theremino **System**

Theremino Cobot

V1.2

The Theremino Cobot application



This application contains the necessary to move a Cobot, i.e. a robotic arm or any other mechanism designed to collaborate with humans.

Devices with one or more motors, arranged in any configuration, can be controlled and it is not necessary to specify the dimensional characteristics of the mechanisms.

We have voluntarily eliminated the three-dimensional description with all the complications that it entails and we directly specify the position for each motor, in degrees, millimeters or in any other unit of measurement you prefer.

Three-dimensional calculations are always inaccurate, as you would have to set all the parameters that contribute to the errors. But they are parameters that are difficult to measure and vary according to the rotations and weights transported.

We then delegate the calculations to an analog computer, which is the mechanics itself, which performs them with absolute precision. Taking into account every possible factor, from the forces involved to gravity, to failures, to structural inaccuracies and even inaccuracies caused by gears, motors and their feedback loops.

These simplifications greatly facilitate the use of the application but in some cases it would be desirable to obtain more linear movements or greater control of the operations. In such cases read the [advanced techniques](#) at the end of this document.

How a Cobot is made

Cobots are designed to work together without endangering humans, so they have a different structure from industrial robots.

The Theremino Cobot application can control any number of motors in any mechanical configuration they are arranged. Separate motors could be controlled, such as a treadmill and a guillotine to cut the flowing product. Or you could control an arm in SCARA configuration or a DELTA type robot, but normally you control an anthropomorphic type Cobot which is the most flexible and therefore usable for many different tasks in medium and small industries.

Here is an example of an anthropomorphic Cobot arm.

This image is just a scheme, the shape could also be different but the main components will always be the same and the proportions will also be more or less these.

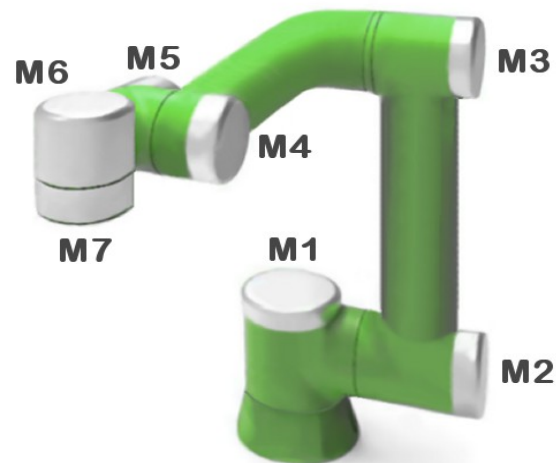
Each silver disc represents a rotation joint and therefore also a motor.

Let's give each part a meaningful name:

- **M1 BASIC** Rotation on the horizontal plane. It could also rotate 360 degrees but usually only works forward and has side stops. It doesn't need much torque but it needs to be precise (little backlash) because its errors are multiplied by the whole length of the arm.
- **M2 SHOULDER** Raise your whole arm. This is the motor that must have the most torque and must also be very precise.
- **M3 ELBOW** It completes the work of the shoulder and allows you to reach every position. This motor must be averagely accurate.

The joints **M4,M5,M6** and **M7** are the **HAND**

- **M4 and M5 WRIST** These joints rotate up-down and right-left, they don't have to be precise or have a lot of torque, but they need to be small and light.
- **M6 ROTATION** This joint could provide continuous rotation, for example to control a screwdriver. In this case M7 will not be there.
- **M7 CLAMP** This is not a rotary joint but a tool for gripping, screwing, welding, painting etc...



Design safe cobots

A cobot must be able to work side by side with a human being without posing a danger. It must have rounded surfaces and in all possible rotations the parts of him must not get too close to each other, otherwise they could pinch the skin or fingers like a nutcracker.



The safety of a Cobot must be intrinsic. The mechanics itself does not have to dispose of strength and speed capable of doing harm.

This is a simple principle, easy to understand, no software or electronic protections.

No rules or protocols that can go wrong and no mechanisms that can break.

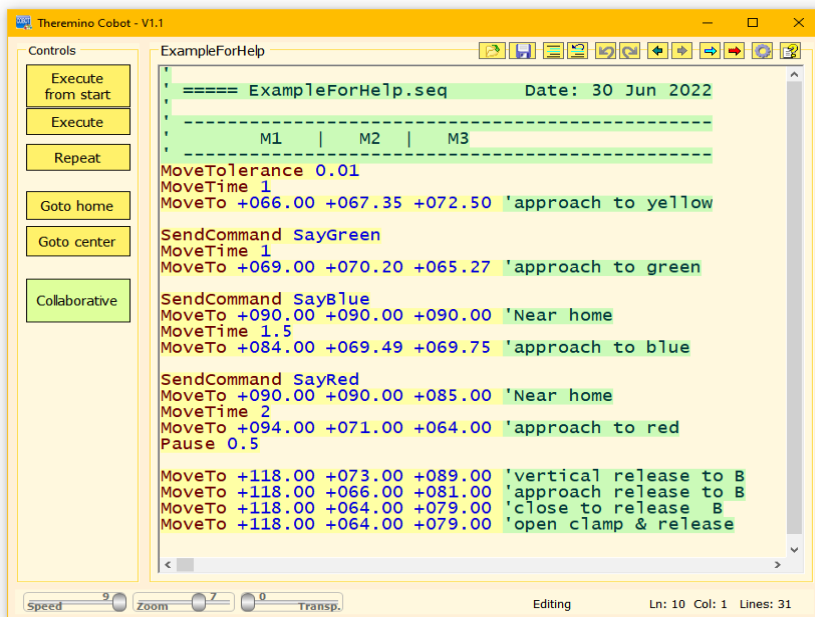
The safety of a Cobot must be intrinsic and not based on certifications, electronic systems or software, because the software can go wrong, the electronics can break and the certifications could be incomplete or misinterpreted.

The security mechanisms based on software, protocols and certifications, in addition to not being sufficient, can also increase the danger of the system because they generate a false trust in humans, who are therefore led to trust blindly and take risks.

Watch [this video](#) about a robot who broke a child's finger.

And read our documentation **Theremino Cobot Security** published in English, Italian and Chinese.

Use the Cobot application



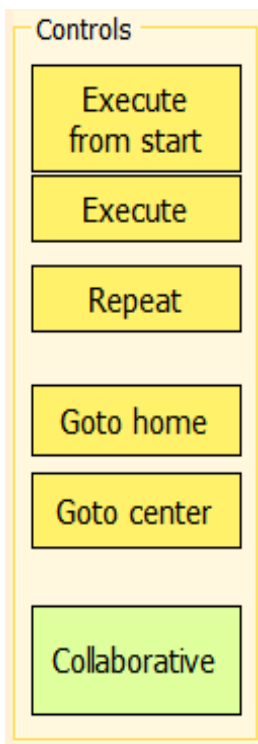
This application is controllable in many ways, as we will see in the following pages, but its basic operation is simple.

You write the command lines in the list and execute them with the commands on this page.

You change the values in the list by watching the movements of the Cobot.

Or are you sending commands from another application (which is usually Theremino_Automation).

Here are the main commands:



Pressing **Execute from start** starts execution from the beginning.

With **Execute** execution starts from the line with the cursor, i.e. the line selected with the mouse or keyboard.

If you enable the button **Repeat** then after executing the last line the execution restarts from the first line.

With **Goto home** you make the Cobot move to the pre-selected position in the mode **Collaborative**.

With **Goto center** makes the Cobot move to the pre-selected position in the mode **Collaborative**.

With **Collaborative** you access the collaborative mode that we will see on the next page.

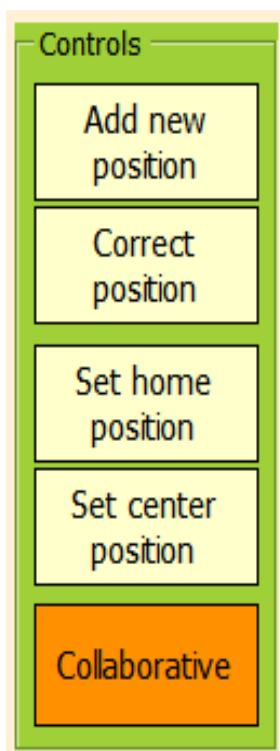
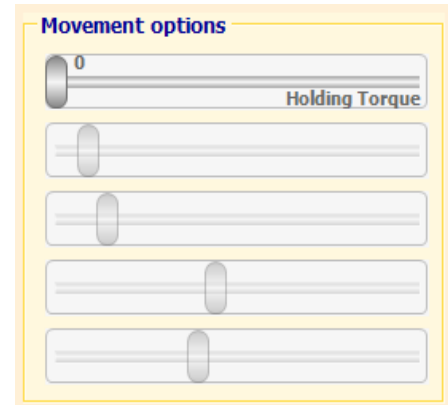
To stop execution, press again **Execute** or click on a line in the command list.

The "Collaborative" way

When you enter the collaborative mode, the cursor appears in the options window **Holding torque** which serves to limit the torque.

Moving **Holding torque** to the left the torque is greatly limited and it becomes possible to manually rotate the joints of the Cobot.

The details of this cursor and the other four are explained in the following pages.



Pressing **Add new position** a line is added to the sequence **MoveTo** with the current position of all motors.

With **Correct position** the line is changed **MoveTo** selected with the current position of the motors.

With **Set home position** position the **Home position** is set with the current position of the motors **Note 1**

With **Set center position** the **Center position** is set with the current position of the motors.

Pressing the button **Collaborative** you exit collaborative mode and return to normal operating mode.

Collaborative mode is only for making changes after making them press the "Collaborative" button and return to normal operating mode.

Note 1- If the motors are used over 360 degrees, the Home position must be in the first turn (from 0 to 4095 for FeeTech) and before turning off the arm, it should be brought back to the Home position.

Edit command sequences

During normal operation, and without entering collaborative mode, you can edit commands of the sequence with the keyboard and with the Mouse.

If the sequence is running, it stops automatically as soon as you place the mouse cursor on the list of commands or use the UP and DOWN arrows.

Right-clicking on the commands opens a menu that we will see in detail in the last pages of this document.

Valid commands, which we will see in the following pages, are highlighted with a yellow background color, while errors with a red color.

Add locations

You can add locations by manually typing commands MoveTo or by copying and pasting other ready-made lines.

You can also add positions using the Collaborative mode, possibly even with an external button while manually moving the Cobot to the desired positions.

Run the commands line by line

You can test individual commands by running them with a double click of the mouse on the first word of the line.

If there is a line selected (with a blue background) then you can use the UP and DOWN arrows on your keyboard to execute the previous and following lines.

Change positions with the mouse

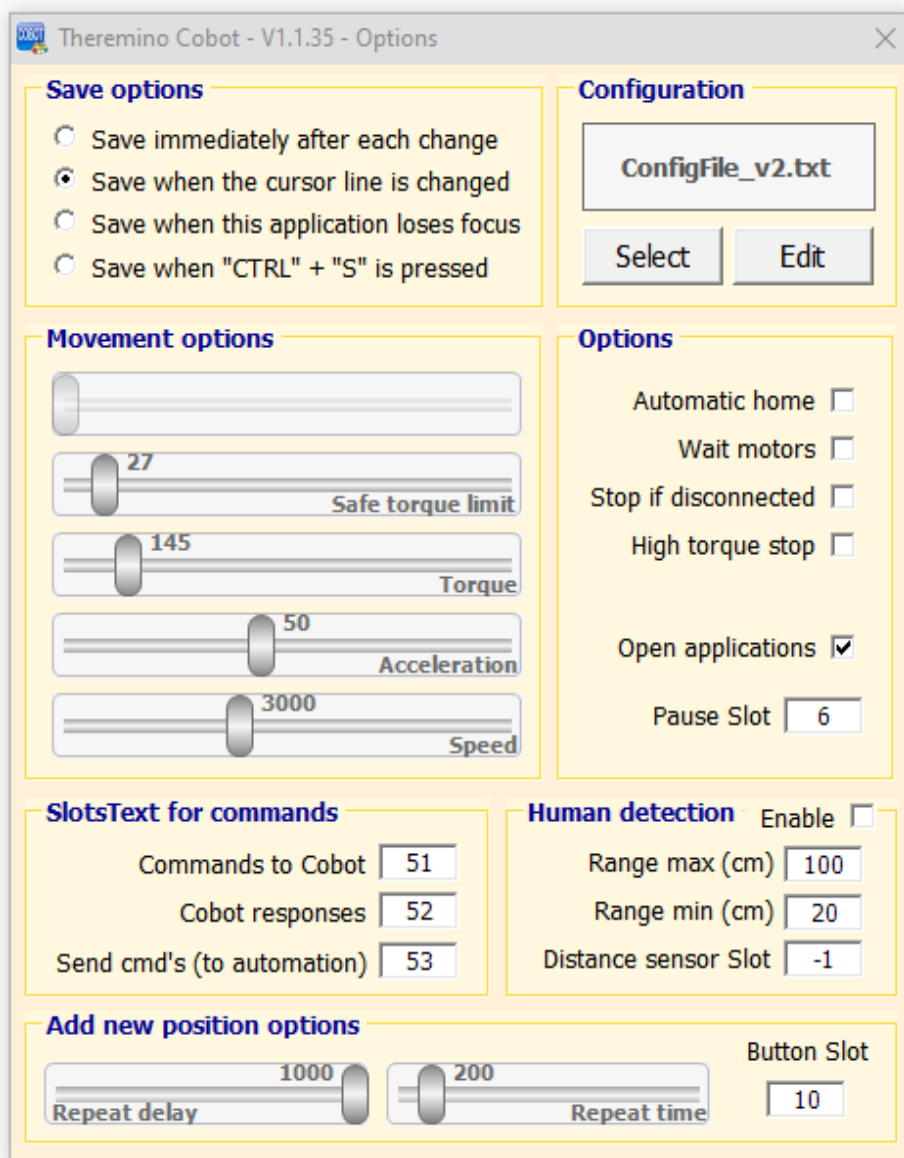
Row rotation values can be changed MoveTo and see the actual movement in the Cobot while making the change.

- With a double click select a numeric value of a MoveTo row.
- As long as the numerical value is selected the mouse wheel modifies it and you can see the movement immediately by looking at the Cobot and without needing to pay attention to the position of the mouse cursor.

The speed of variation of the numerical values can be modified by pressing the keys SHIFT, CTRL and ALT while using the mouse wheel.

Normally the speed when no keys are pressed is a whole number for each click of the mouse wheel, while the speeds with SHIFT, CTRL and ALT are 10, 0.1 and 0.01. These values can be modified in the configuration which is explained in the following pages.

The options panel

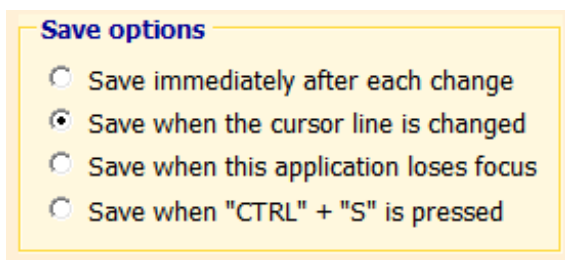


This panel is opened by pressing the gear button located at the top right of the main window.



In the following pages we will see the meaning and use of the various areas of this panel.

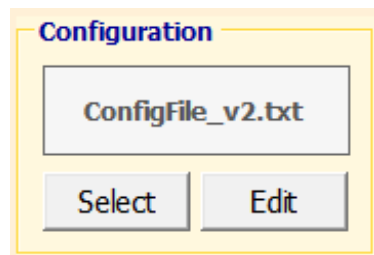
Options panel- Save options



These options determine when the command list is saved to disk.

It can be saved automatically in various ways or when you press CTRL-S. In all cases, the list is also saved before loading another from file or when closing the application.

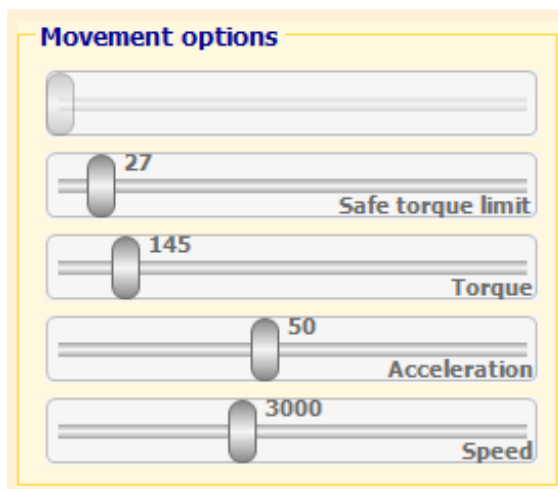
Options panel-Configuration



With these buttons you choose the configuration file to use and open it in a text editor (usually NotePad) to edit it.

The details of the configuration are explained on the next pages.

Options panel - Movement options



The first slider at the top, which is disabled here, is for limiting the pair in "Collaborative" mode.

If you right click the sliders go to the default value. The maximum and default values of these cursors are in the configuration.

Values torque, acceleration and Speed they are used as limits inside the motor itself.

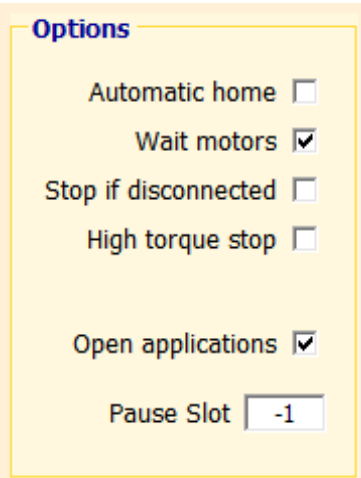
In some motors (FeeTech) the torque limit is not controlled well in the firmware and when it is lowered the acceleration no longer works well so it is good to keep it at maximum **Note 1**

Note 1 - If the torque must be kept at maximum, torque limitation can be delegated to the Cobot application, which will lower the motor torque when the torque rises above the limit set in **Safe torque limit**

Not all motors implement these settings in the firmware, the only ones that use all the features of the Cobot application they are the FeeTech and the TMOT (Theremino Motors) (under planning in 2022).

Options panel - Options

These options control some general behavior of the application.



The screenshot shows a yellow-bordered panel titled "Options". It contains the following settings:

- Automatic home
- Wait motors
- Stop if disconnected
- High torque stop
- Open applications
- Pause Slot

- **Automatic home** - Enable automatic movement to Home positions each time the application starts and closes [Note 1](#)
- **Wait motors** - Normally you keep it enabled and disable it only to try command sequences without motors.
- **Stop if disconnected** - If you enable it, it stops the execution of the sequence if the motor voltage drops below 2 Volts [Note 2](#)
- **High torque stop** - If you enable it, it stops the execution of the sequence if the torque becomes excessive.

- **Open applications** - If you enable this option then all the applications with the name `Theremino_xx.exe` will be started and closed when starting and closing. The applications to be started must be located in the same folder and in the sub-folders of the `Theremino_Cobot.exe` file.

Usually it is good to create an "Apps" folder to contain all the applications to be launched.

If you use the `Theremino_Automation` application then it is better to leave this task to it too [Note 1](#) and disable this option.

- **Pause Slot** - Used to control pause with an external button. By setting a valid Slot number, execution is paused if the value of that Slot exceeds 500. Execution restarts when the value drops below 500. By setting -1, this option is inactive.

[Note_1](#)

If you use the `Theremino_Automation` application then you get a more robust operation driving everything from Automation and disabling the options **Automatic home** and **Open applications**. Furthermore, with Automation, disconnection events and sending a command could also be better managed **StopExecution** when necessary. In this case it is advisable to also disable the option **Stop if disconnected**.

[Note_2](#)

The motor voltage is read from the Slots relating to each motor. Slot values are in the configuration.

Options Panel - SlotsText for Commands

SlotsText for commands

Commands to Cobot

Cobot responses

Send cmd's (to automation)

Here the Slot Texts are set **Note 1** to receive commands from external applications, to give answers and also to command external applications (usually Theremino_Automation).

If you don't use them, disable them setting them with the value -1

Note_1 - Text Slots are used to communicate text messages between applications.

Options panel - Human detection

Human detection Enable

Range max (cm)

Range min (cm)

Distance sensor Slot

These options adjust the start and end of the progressive safety zone. Below Range min the speed is limited to a minimum. Above Range max it is not limited. Between the two extremes there is a progressive limitation of the speed.

To make this mechanism work, the Distance sensor Slot is set with a valid Slot value and then a distance sensor is connected to this Slot.

For maximum reliability we recommend using multiple sensors [as explained in this section of our site](#).

By increasing the number of sensors (even up to 17 with little expense), and directing them in various directions and you get confidence that at least one sensor will detect the presence of a human in the safe zone.

Options panel - Add new position / Correct selected position

Add new position / Correct selected position

Repeat delay

Repeat time

Button Slot

These options adjust the initial delay and repeat time (in milliseconds) and the Slot to be used for an external button. Then the HAL application will be set so that by pressing this button the value in the Slot exceeds 500.

When the button is pressed, a new line will be added to the command list with the current position of all the motors, and if it is held down for a long time, many lines will be added with delay and cadence adjusted by the two cursors.

If there is a row selected (highlighted with a blue background) the external button modifies it instead of adding a new row.

To disable this option, set the Slot with the value -1

Understand and change the configuration

Many operations that we will explain in the following pages depend on the values written in the configuration. In the next pages we will explain the sections of the configuration, one by one.

Motor type and parameters for each of the motors

Motor - The number of lines starting with "Motor" determines the number of motors

Name - The type of motor determines some characteristics for the movements

Slots - This motor's slots start from this one **basic number**

Min - Limit motor movement (must be less than Max)

Max - Limits motor movement (must be greater than Min)

Type	Name	Slots	Min	Max
Motor	STS3215	100	-30000	+30000
Motor	STS3215	200	-30000	+30000
Motor	STS3215	300	-30000	+30000
Motor	STEPPER	400	+0	+360
Motor	SERVO	500	+50	+950
'Motor	SERVO	600	+50	+950
'Motor	Sente150	700	-860000	860000

Slots for motor parameters

These are the increments in the motor register map to add to the **basic number** of the motor.

MOTOR SLOT INCREMENTS
SlotDestination1
SlotPosition2
Slot Velocity3
Slot Torque4
SlotVoltage5
SlotTemperature6
SlotMoving7
SlotCurrent8
SlotTorqueLimit20
SlotAccLimit21
SlotSpeedLimit22
SlotOffOnCenter23

Number of integer and decimal digits for MoveTo instruction numbers

Example with 3 and 2 : `MoveTo +123.45 -123.45 ----- +000.00`

=====

= DIGITS (3 to 9) and DECIMALS (0 to 9) for edit numbers

=====

digits3

decimals 2

Speed of adjusting positions with the mouse wheel

By changing these multipliers you can achieve your preferred speeds.

=====

= EDIT SPEED MULTIPLIER

= -----

= Normally 1 (with Feetech or Theremino motors)

= Set to 10, 100 or more to increase the mouse wheel effect

=====

EditSpeedMultiplierSHIFT 10

EditSpeedMultiplierCTRL 0.1

EditSpeedMultiplierALT 0.01

EditSpeedMultiplier 1

Tolerance for GotoCenter and Home commands

This is the tolerance that will be used after the GotoCenter and GotoHome commands

=====

= END MOVING TOLERANCE

= -----

= Before to update Acc Or Speed and after GotoCenter and Home

=====

EndMovingTolerance10

Default values for Torque, Acceleration and Speed sliders

When you click with the right mouse button the cursors go to the value established by these lines.

=====

= SLIDERS

=====

SafeTorqueMax500'usually 200 to 400

Safe Torque Default150'usually 150

Torque Default500' min=1 max=1000

AccDefault50' min=1 max=255

SpeedDefault3000' min=1 max=65535

Command sequences

Commands to move the motors

MoveTo n n n n.. The numbers n are the destinations in steps, mm or degrees

MoveTolerance n The number n indicates the tolerance in steps, mm or degrees

MoveSpeed n Interpolate with step speed, mm or degrees per second

MoveTime n Interpolate with fixed time in seconds

The command **MoveTolerance** causes a wait, at the end of each **MoveTo**, until all the motors have arrived at the destination within the specified tolerance.

The last command **MoveTolerance**, **MoveSpeed** or **MoveTime** which is encountered, it is used for all subsequent lines until another is encountered.

The commands **MoveSpeed** and **MoveTime** they cause a **motion interpolation**. The route to the destination is sent a hundred times per second dividing the total distance into many small sections. This way all motors arrive at their destination together.

With some motors that communicate slowly and don't have acceleration (for example the Sentels) the interpolation works badly. To disable it, write **MoveTime 0**

Torque, acceleration and speed regulation

Safe Torque n Only for smart motors (not steppers and servos)

Torque n Only for smart motors (not steppers and servos)

Acceleration n Only for smart motors (not steppers and servos)

Speed n Only for smart motors (not steppers and servos)

Checking the

Breaks n Pause for "n" seconds

Restart Restart the program from the first line

Stop Stop the execution of the program

Communication commands with other applications

Slot x = n Write the value n in the slot x

SendCommand string Send a command to Automation or other applications

All numbers denoted by "n" can also be fractional and for decimals you can use either the point or the comma.

Times in seconds can also indicate fractions, down to thousandths of a second.

The commands explained one by one

MoveTo

Moves motors to indicated destinations.

Here are some examples:

```
MoveTo +000 +000 +000 ' All three motors at 0 degrees
```

```
MoveTo +360 ---- ---- ' Motor 1 to 360 deg and motors 2 and 3 stay where they are
```

```
MoveTo +89.9 +11.1 +3.00 ' Motors 1, 2 and 3 with one integer and two decimals
```

```
MoveTo -89.9 -11.1 -3.00 ' The same values as the previous line but in negative
```

In the configuration (explained in the previous pages) you can indicate how many decimals and how many whole digits to display.

With a good setting of the number of digits and decimals it is possible to maintain a good alignment of the columns for all numbers, even in the case of positive, negative numbers and with many decimals.

The dashes indicate that the destination of the motor is the same as it already had previously. Therefore dashes are set for motors that must not move.

MoveTolerance

The tolerance is used by the instructions `MoveTo` and is used to wait for the completion of the movements of all the motors.

The program continues to the next line only if the difference between the destination and the current position is smaller than the specified tolerance.

Here are some examples:

```
MoveTolerance 0.5 ' Wait for all motors to be within half a degree of destination
```

```
MoveTolerance 15 ' Proceed quickly approximating the path
```

MoveSpeed

The program runs every instruction `MoveTo` with pre-set speeds.

Every instruction `MoveTo` will send to motors numerous intermediate destinations, one every ten milliseconds, until time runs out.

Time is calculated with the formula: `Path` (in units, i.e. steps / degrees or millimeters) divided `Speed` (in units per second), using the motor path that has the furthest to go.

If all motors are fast enough to follow these directions then all motors will arrive at their destination together.

Here are some examples:

```
MoveSpeed 50 ' Send many intermediate positions to respect the given speed
```

```
MoveSpeed 999 ' Proceed very quickly
```

Probably in the latter case the motors will lag behind and therefore the program will wait until all the motors have arrived within the `MoveTolerance`, before moving on to the next line.

MoveTime

The program runs every instruction `MoveTo` in a pre-set time.

Every instruction `MoveTo` will send to motors numerous intermediate destinations, one every ten milliseconds, until time runs out.

If all motors are fast enough to follow these directions then all motors will arrive at their destination together.

Here are some examples:

```
MoveTime 2 ' Interpolate with many positions in between for two seconds
```

```
MoveTime 0.01 ' Proceed very quickly.
```

Probably in the latter case the motors will lag behind and therefore the program will wait until all the motors have arrived within the `MoveTolerance`, before moving on to the next line.

With some motors that communicate slowly and have no acceleration (for example the Sentel) the interpolation works badly and the motor goes jerky.

To disable it, write `MoveTime 0`

Safe Torque

This instruction sets the SafeTorque slider which adjusts the safe torque. Command valid only for Smart Motors (not stepper and servo).

Currently this command is under construction and depends a lot on the type of motors used. Not in all cases it will work as expected.

Here are some examples:

```
Safe Torque 300 ' All motors with safety torque 300
```

torque

This instruction sets the Torque slider which sends maximum torque to all motors. Command valid only for Smart Motors (not stepper and servo).

Here are some examples:

```
Torque 300 ' All motors with maximum torque 300
```

acceleration

This instruction sets the Acceleration slider which sends acceleration to all motors. Command valid only for Smart Motors (not stepper and servo).

Here are some examples:

```
Acceleration 20 ' All motors with acceleration 20
```

Speed

This instruction sets the Speed fader which sends speed to all motors. Command valid only for Smart Motors (not stepper and servo).

Here are some examples:

```
Speed 1000 ' All motors at 1000 speed
```

Breaks

This instruction halts execution for the time indicated in seconds and fractions.

Here are some examples:

```
Pause 12 ' Wait for 12 seconds
```

```
Pause 0.5 ' Wait for half a second
```

```
Pause 1.53 ' Wait for 1 second and 53 cents
```

Restart

This instruction restarts the program

Here is an example:

```
Restart          ' Restart from the first line of the program
```

Stop

This statement stops the program

Here is an example:

```
stop            ' Stop the program execution
```

Slots

This instruction writes an immediate number into a Slot

Here are some examples:

```
slots 3 = 1000 ' Write the number 1000 in Slot 3
```

```
slots 9 = 0.33 ' Write the number 0.33 in Slot 9
```

```
slots 999 = 10 ' Write the number 10 in Slot 999
```

SendCommand

This instruction sends a text string in the SlotText indicated in the options panel with: `Send cmd's (to automation)`

After sending the command, the SendCommand instruction waits for the command to be received (ie for the SlotText to be empty again).

Here are some examples:

```
SendCommand Beep ' Send "Beep" command
```

```
SendCommand Multiple beeps ' Send "Multiple beeps" command
```

Slots and SendCommand

With this instruction the numeric slots are written.

The Slots are the communication center of the theremino system and those who read these instructions **should** already know what they are.

Otherwise we recommend [read this section](#) and the following on SlotText, and maybe even the whole [communications page](#), from start to finish.

Here are some examples that write in a Slot

`slots 1 = 12`

The value 12 is written to Slot 1

`slots 2 = 1000`

In Slot 2 the numerical value 1000 is written

And this is an instruction that writes to SlotText using the SendCommand function which is explained in the last pages of this document.

`SendCommandBeep`

Please note that these instructions have a simplified syntax compared to the Slot instructions of the Automation application.

- Only immediate values are written, no mathematical formulas or functions.
- Parentheses are not used.
- No values are read from the Slots, only immediate numbers are written.
- The SlotText are not written but the SendCommand function explained in the last pages of this document is used

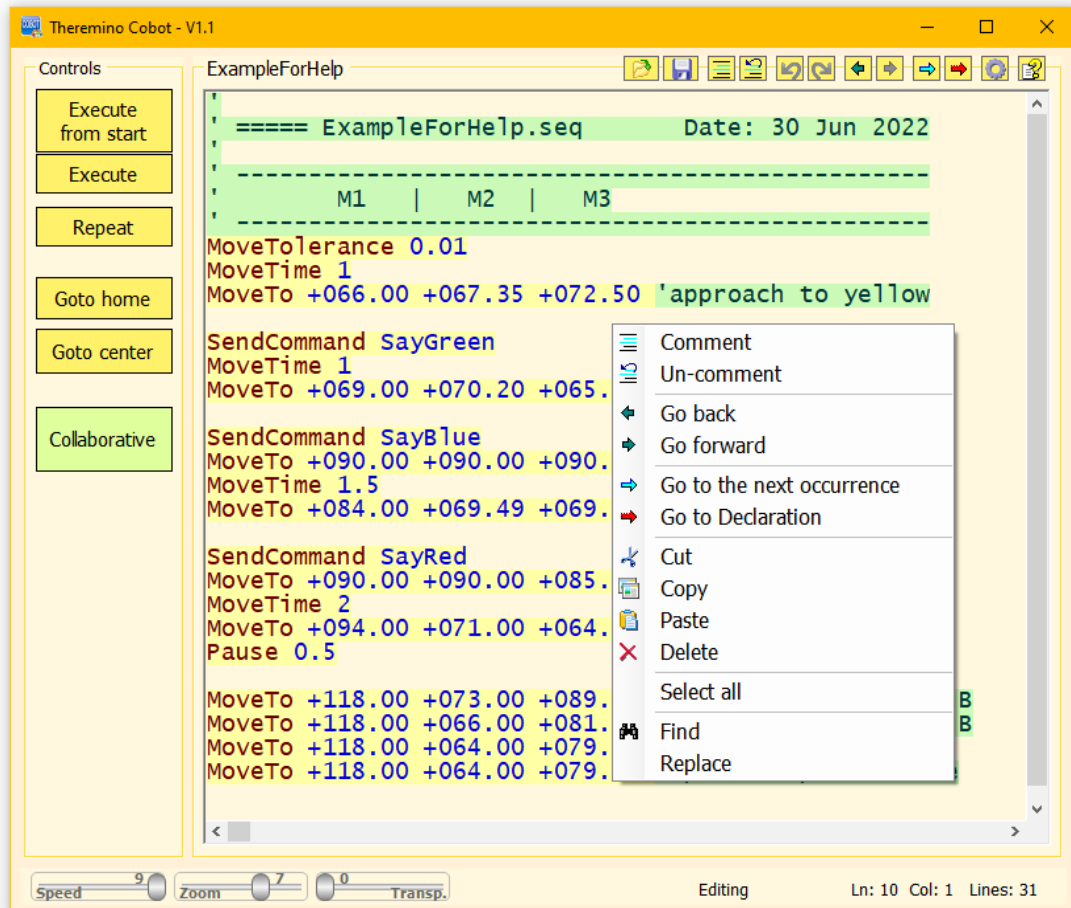
Be careful not to confuse Slots with SlotTexts, they have similar addresses (from 0 to 999),but they write and read in different memory zones.

Furthermore, Slots contain numbers (integer or floating point), while SlotTexts contain character strings (up to 100,000 characters).

And finally, SlotTexts can only be used to communicate between applications and not to communicate with the HALs and Master or Arduino modules.

The program menu

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



"Comments" and **"Uncomment"** they are used to comment (add the initial quote) to entire areas of the program. Or to delete comments.

"Go Back" and **"Go Forward"** they move the cursor, and also the visible page, to the previously visited program sections.

"Go to the next occurrence" searches for other occurrences of the selected word.

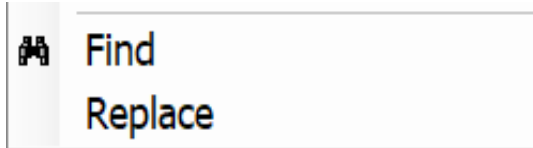
"Go to declaration" searches for the selected word only in the declaration lines.

The commands **"Cut"**, **"Copy"**, **"Pastes"**, **"Delete"** And **"Select All"**, copy, paste, delete and select parts of the program. Instead, you could also use the CTRL-X, CTRL-C, CTRL-V, DEL, and CTRL-A keys.

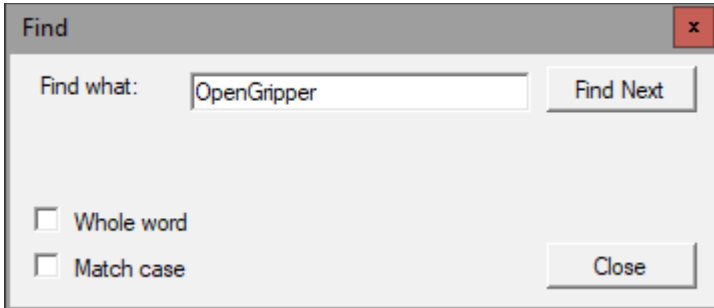
"Find"(or CTRL-F) and **"Replace"**, open the window for finding and replacing words and phrases.

Find and Replace

The last items at the bottom of the application menu open two similar windows.



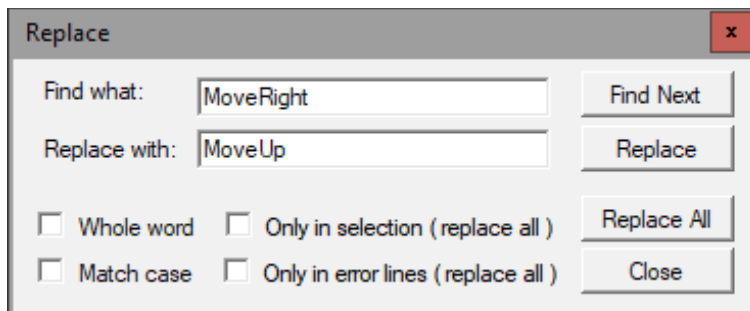
The “FIND” window



With this window you can search for words or phrases in the program text.

- ◆ If “Whole word” is enabled, the word must be complete.
- ◆ If you enable "Match case", the words must also match in upper and lower case.
- ◆ With "Find next" (or with F3), you go to the next occurrence of the word you are looking for. If the end of the program is reached, the search restarts from the beginning.

The “REPLACE” window



This window has the same options as the previous one, but it also allows you to replace the word (or phrase) with another one.

If "Replace" is pressed, only one replacement is made. Instead with "Replace All" all occurrences are replaced.

The substitution can take place in the whole program or only in the selected area, or only in the lines that contain errors (or "warnings").

The top bar controls



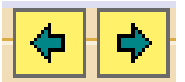
The first two buttons load and save command sequences.



These two buttons comment and un-comment the selected lines.



The two blue arrows they are used to roll back program changes and to rebuild deleted changes.



The two dark ARROWS move the cursor, and also the visible page, to the previously visited program sections.

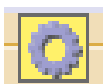


The blue ARROW looks for all occurrences of functions, variables or even simple words.



The red ARROW searches the declarations only (Button, Key, Label and Variable) (inherited from Automation and currently hardly usable).

The search functions are convenient, just select a word or even just place the cursor on it and then press the arrow repeatedly.



The gear opens the options window.

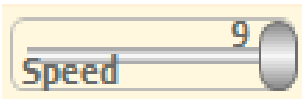


The question mark opens the instruction file (Help) in the chosen language. For this command to work you need to copy the Help files of your preferred languages into the "Docs" folder.

If the Help file is not found then a message appears suggesting that you open the Docs folder and copy the file into it.

Or you can choose to select a Help file in your preferred language located in the "Docs" folder or any other folder. *To change the selected file, click the button with the right mouse button.*

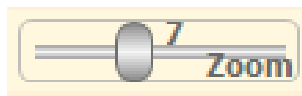
The bottom bar controls



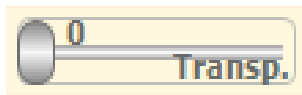
This slider adjusts the playing speed.

Speeds range from "1" (one instruction per second), as far as "8" (ten thousand instructions per second), it's at "9" (the maximum speed allowed by the system).

While writing the program it is good to use a medium speed. Speed usually "5" (20 instructions per second), which is slow enough that you can visually follow the execution of the program.



The ZOOM slider establishes the size of the text, both in the program window and in the Debug window.



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

Executing line 15

The middle of the bottom bar shows the current status which can be "Executing line nn", "Ready", "Editing" and other commands that indicate special conditions. These same messages are also sent in the Text Slot **CobotResponses** and can be read by Theremino Automation or other applications. See the pages about [communications with Automation](#).

Ln: 17 Col: 1 Lines: 18

The right part of the bottom bar shows information about the program:

- The total number of lines
- The line where the cursor is (starting from line 1)
- The column where the cursor is (starting from column 1)

Advanced techniques

As we have already seen, the Cobot application is deliberately simple and this greatly facilitates its use in most cases, but there are some cases which cannot be managed simply and which require special techniques.

In the following pages we will explain some techniques that facilitate the following operations:

- Make linear movements
- Move with orthogonal coordinates X, Y and Z
- Send commands to the Theremino Automation application
- Receive commands from the Theremino Automation application or from other applications of the theremino system.

Linear movements

Using MoveSpeed or MoveTime the movement is broken into many small segments, all motors are driven one hundred times per second (if the communication line used allows it) and all will arrive together at the destination.

MoveSpeed and MoveTime improve path control but if the destination is far away you still get visibly curved movements. If it is necessary to minimize this problem, the following techniques can be used:

- Decrease MoveSpeed (or increase MoveTime) to check if unwanted cornering is produced by too much speed. For some motors, which have to make a greater movement than others, the speed could be higher than the limit set in the motors (in the HAL for the Steppers or in the cursors of the Cobot application for the programmable motors). In these cases some motors lag behind and the path follows even greater curves than the inevitable ones, caused by the geometry of the mechanics.
- If decreasing the speed is not enough then it is advisable to increase it again then break up the motions into multiple lines to create more closely spaced waypoints. Short stages must therefore be set up where precise movements and positions are required and vice versa stages far from each other in areas where the precision of the route does not matter.
- Place the waypoints of the route in suitable places. One point must be at the beginning of the area where you want to have maximum precision and the others, close to each other, along the entire area that must be precise.

3D extension

In some cases the proposed methods may not be enough, for example to make a long and perfectly straight weld.

We are therefore also preparing the possibility of carrying out three-dimensional calculations and therefore specifying the position of the tip of the arm in the three spatial coordinates X, Y and Z, but bear in mind that:

- Setting up features will become significantly more complex and arm handling will also not be as straightforward as it is now.
- If the requests are numerous, this option will probably be available in 2023.

Commands from Cobot to Automation

In Automation the `Label Event_CommandsFromCobot` is used to receive commands from the `Theremino_COBOT` application, but it could also receive commands from any other application capable of writing to the "Text Slots".

When an external application writes a string with a command in the `Slot_CommandsFromCobot` the event is notified by calling this `Label`.

The Automation program is aborted, whatever it was doing, and the instructions after the `Label` are executed until `Return`.

The command text is read with the "CommandText" function and then decoded with a "Select" structure, as in the following example:

```
Variable Numeric Slot_CommandsFromCobot = 53
Stop

Label Event_CommandsFromCobot
  Select CommandText
    "Beep" houses
      ExecBeep
  '
    "Multiple Beep" Houses
      ClearCommand
      ExecBeepMultiple
  '
  CaseElse
    Print "Unrecognized command: " + CommandText
  EndSelect
  ClearCommand
Return

Label ExecBeep
  Beeps 440 300
  Wait Seconds 0.5
Return

Label ExecBeepMultiple
  For v1 = 1 To 3
    Beeps 880 400
    Wait Seconds 0.5
  Next
Return

Label ClearCommand
  SlotText(Slot_CommandsFromCobot) = ""
Return
```

Note that in the `Case "Multiple Beep"` a `ClearCommand` instruction has been added before executing the command. This instruction tells the COBOT application to continue immediately, without waiting for the command to finish executing.

On the next page is an example of a sequence that sends these commands.

The commands from COBOT to Automation

This chapter explains how to send commands from the COBOT application to the Automation application. The following example is a sequence that moves three motors and sends the "Beep" and "Multiple Beep" commands between one movement and the next.

```
Move Tolerance 1
MoveTime 0.5
MoveTo +00137.000 +00300.000 +00022.000
SendCommandBeep
MoveTo +00153.000 +00303.000 +00025.000
SendCommandMultiple beeps
```

The file containing this sequence is run by the Theremino_COBOT application and must be in its "Sequences" folder.

The commands sent to the Automation application can be any text string, even separated by spaces, and are decoded without taking into account upper or lower case.

Any multiple TAB or Space characters are turned into single spaces by the COBOT application before sending, and leading and trailing TABs and spaces are stripped.

The user himself can write his own commands, remembering that:

- The commands must be written in the sequence of the COBOT application.
- Instructions to decode them in Automation's CommandsFromCobot event.

In the Cobot application you have to set the text slots to be used for the commands. Open the options panel (with the cog at the top right) and locate the "Slots for Text-Commands" section located at the bottom.

The text slots that are set in the Cobot application must be the same ones that are declared in the Automation variables with the names: **Slot_CommandsToCobot**, **Slot_CobotResponses** and **Slot_CommandsFromCobot**.

You can also send commands that press buttons, sending their name, these methods are better explained in the Automation documentation.

See also the examples in the folder
"Demo Programs\SlotText Commands"

On the next page we will see that commands can also be sent from the Automation application to the Cobot application.

The commands from Automation to COBOT

To send commands to the Cobot application, the variables are set **Slot_CommandsToCobot** and **Slot_CobotResponses** and then use them in Automation (or other applications) with commands that write and read text slots.

Commands from Automation to COBOT (text strings in the **Slot_CommandsToCobot**)

- **ExitFromEdit** (exit "edit" status)
- **StartExecution** (automatic exit from the "edit" state)
- **StopExecution** (automatic exit from the "edit" state)
- **GotoHome** (automatic exit from the "edit" state)
- **Goto Center** (automatic exit from the "edit" state)
- **EnableRepeat**
- **DisableRepeat**
- **SelectLine nnn**
- **ExecuteSelectedLine**
- **ArrowUP**
- **ArrowDOWN**
- **EnableCollaborative**
- **DisableCollaborative**
- **AddNewPosition**
- **CorrectSelectedPosition**
- **DeleteSelectedLine**
- **SetHomePosition**
- **SetCenterPosition**
- **SetHoldingTorque nnn**
- **SetSafeTorqueLimit nnn**
- **SetTorque nos**
- **SetAcceleration nnn**
- **SetSpeed nnn**
- **LoadSequence SeqName.seq** (the name "xxxxx.seq" must not contain spaces)

The Cobot application deletes the text from the Slot after receiving and interpreting it. The application that sent the command must wait for the command to be received before sending others.

Answers from the Cobot (text strings in the **Slot_ResponsesFromCobot**)

- **Editing**
- **Ready**
- **Execution running at line nnn**
- **Motors disconnected**
- **Human detected - Speed reduced**
- **Human too near - Execution stopped**
- **Too much torque - Execution stopped**
- **Motors disconnected - Execution stopped**
- **Execution paused**