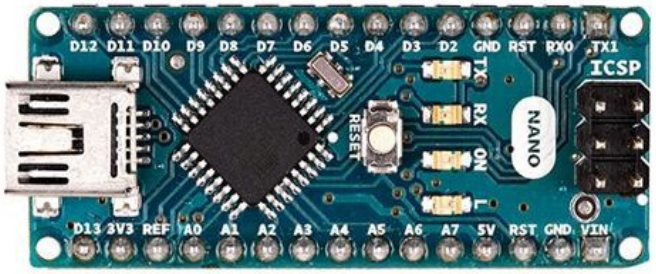
Theremino **System**

Theremino ArduHAL V1.1 Instructions

Principle of operation

You connect an Arduino Nano to the USB port.



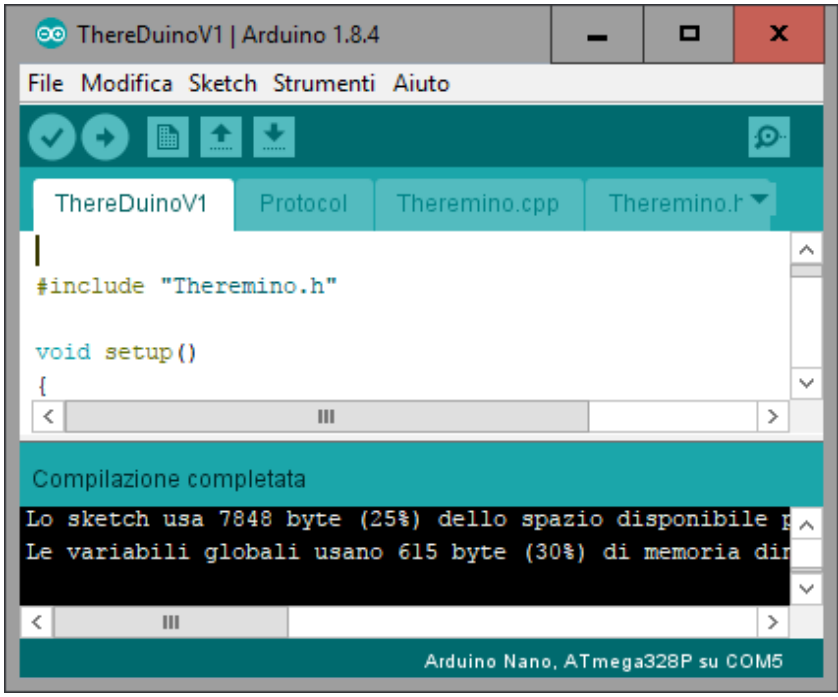
You could also use other Arduino models, but read ***This Page***.

You use the Arduino IDE to program the Arduino with our firmware (as explained in ***This Page***).

OR

You could buy a pre-programmed Arduino Nano, on the ThereminoStore site, or from eBay.

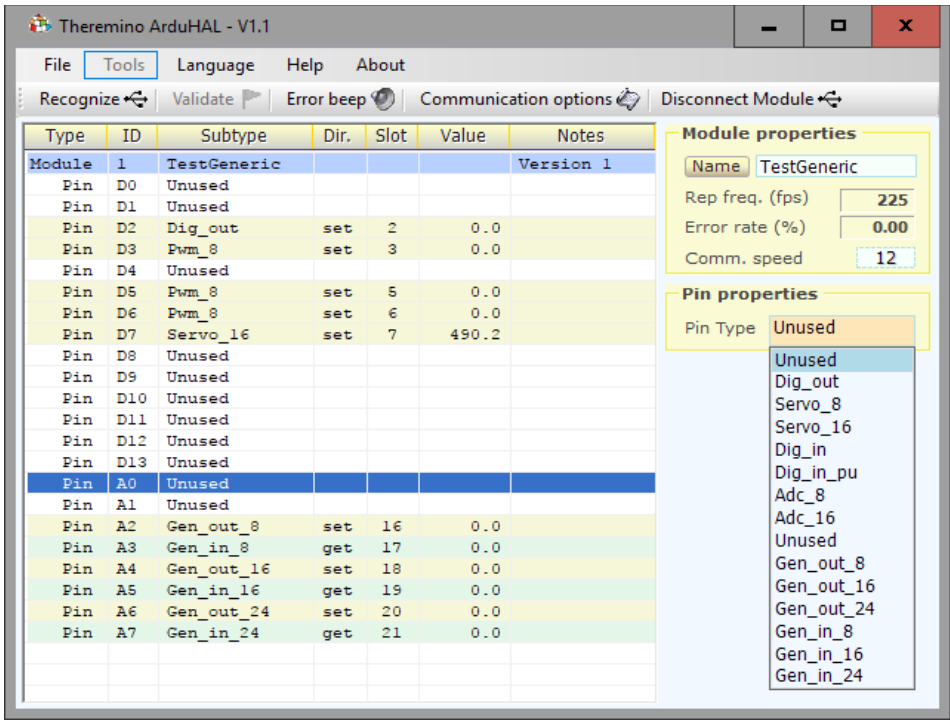
With the pre-programmed modules you just connect the USB cable, and avoid to install the Arduino IDE and program the Arduino.



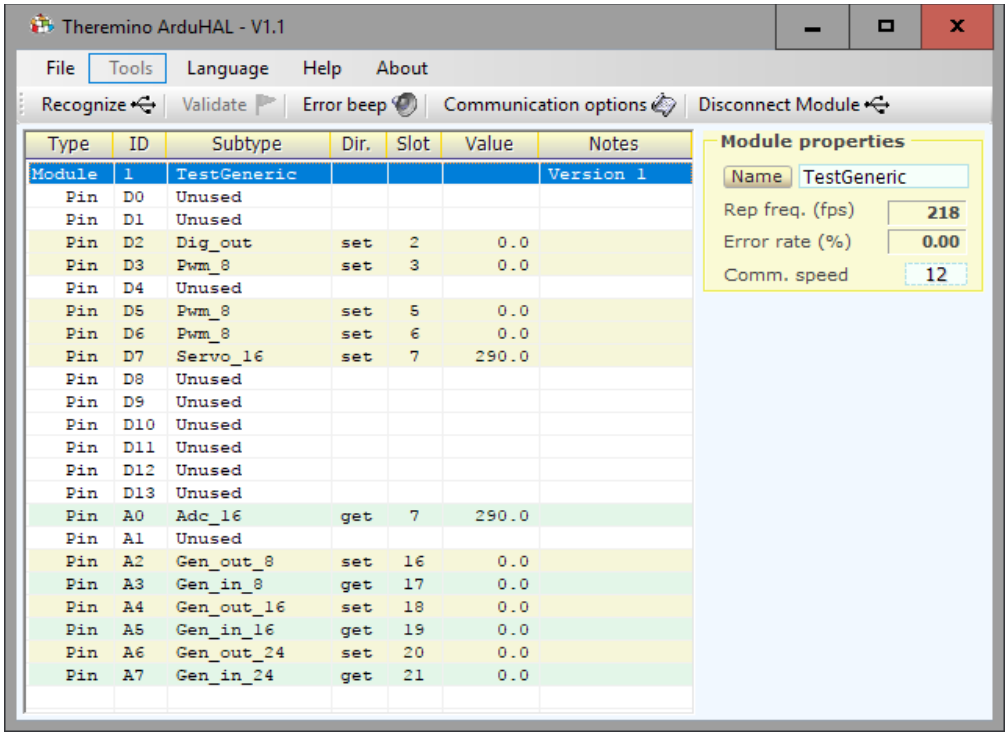
You launch the application ***ArduHAL*** and configure the IN-OUT Pins to read sensors, move motors, etc ..

You can use more than one hundred Theremino system applications, covering almost all fields, from scientific experiments to the music, to radioactivity, teaching... see ***This Page***.

All this works immediately without writing a single line of firmware or software.



The ArduHAL application



Theremino ArduHAL with a module Arduino Nano connected

The application ***ArduHAL*** (Hardware Abstraction Layer), which is discharged from ***This Page***, it appears with a very simple interface, but performs complex operations, with algorithms that benefit from many years of research and development with the Master modules.

Theremino ***ArduHAL*** is the heart of communication with the hardware, can communicate with many Arduino modules simultaneously, he knows the serial communication protocol, knows all the most common types of input-output and allows you to easily configure them.

Without the HAL, to communicate with the Arduino hardware would be difficult and would require much time and labor. For each type of InOut, such as moving a motor or even light a LED, you should write the appropriate firmware.

In Theremino system there are also two other HAL, the first is simply called HAL and communicates via USB with modules Master, the second is called NetHAL and communicates via WiFi, network and Internet with the NetModules. In this document sometimes they will be referred generically as HAL to indicate all three applications.

If you use hardware modules *then the HAL is indispensable and must remain on, you can minimize it, but must remain in operation.*

If you do not use hardware *then the HAL is not necessary, the theremino system applications of can communicate with each other, through the slot, even without HAL.*

When you add or subtract modules*, Some red lines warn that the configuration has changed. With the "Valid" button you choose to lose your old configuration and adapt existing hardware.*

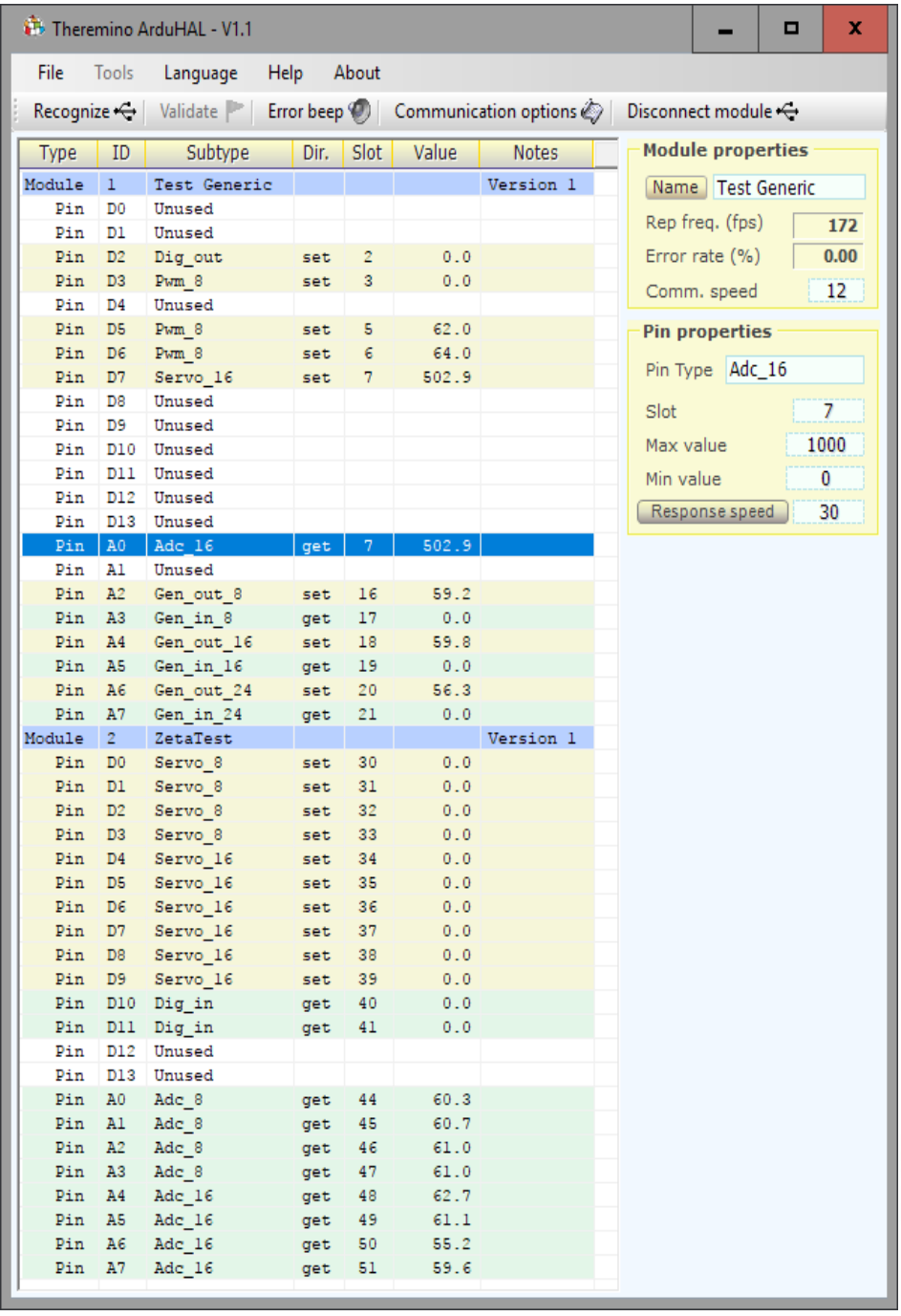
Connecting multiple modules Arduino

The application Theremino ArduHAL can communicate with any number of Arduino modules.

The Pin of all modules are read and written all at once and at the maximum speed from any Arduino. In other words, a slow Arduino does not cause a slowing of communication with others.

In the module list they will be presented in alphabetical order. So the list looks the same, even if you exchange the USB ports.

Theremino ArduHAL connected to two Arduino Nano modules (44 Pins in total)



Type	ID	Subtype	Dir.	Slot	Value	Notes
Module	1	Test Generic				Version 1
Pin	D0	Unused				
Pin	D1	Unused				
Pin	D2	Dig_out	set	2	0.0	
Pin	D3	Pwm_8	set	3	0.0	
Pin	D4	Unused				
Pin	D5	Pwm_8	set	5	62.0	
Pin	D6	Pwm_8	set	6	64.0	
Pin	D7	Servo_16	set	7	502.9	
Pin	D8	Unused				
Pin	D9	Unused				
Pin	D10	Unused				
Pin	D11	Unused				
Pin	D12	Unused				
Pin	D13	Unused				
Pin	A0	Adc_16	get	7	502.9	
Pin	A1	Unused				
Pin	A2	Gen_out_8	set	16	59.2	
Pin	A3	Gen_in_8	get	17	0.0	
Pin	A4	Gen_out_16	set	18	59.8	
Pin	A5	Gen_in_16	get	19	0.0	
Pin	A6	Gen_out_24	set	20	56.3	
Pin	A7	Gen_in_24	get	21	0.0	
Module	2	ZetaTest				Version 1
Pin	D0	Servo_8	set	30	0.0	
Pin	D1	Servo_8	set	31	0.0	
Pin	D2	Servo_8	set	32	0.0	
Pin	D3	Servo_8	set	33	0.0	
Pin	D4	Servo_16	set	34	0.0	
Pin	D5	Servo_16	set	35	0.0	
Pin	D6	Servo_16	set	36	0.0	
Pin	D7	Servo_16	set	37	0.0	
Pin	D8	Servo_16	set	38	0.0	
Pin	D9	Servo_16	set	39	0.0	
Pin	D10	Dig_in	get	40	0.0	
Pin	D11	Dig_in	get	41	0.0	
Pin	D12	Unused				
Pin	D13	Unused				
Pin	A0	Adc_8	get	44	60.3	
Pin	A1	Adc_8	get	45	60.7	
Pin	A2	Adc_8	get	46	61.0	
Pin	A3	Adc_8	get	47	61.0	
Pin	A4	Adc_16	get	48	62.7	
Pin	A5	Adc_16	get	49	61.1	
Pin	A6	Adc_16	get	50	55.2	
Pin	A7	Adc_16	get	51	59.6	

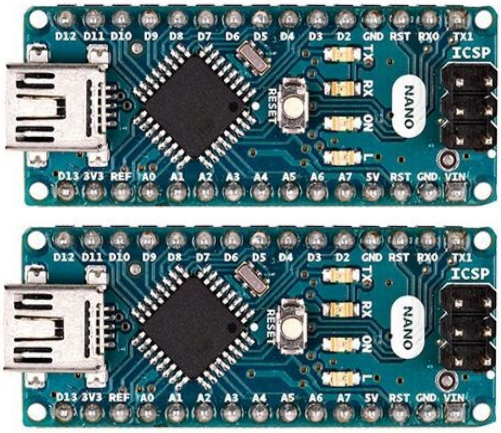
In this example we connected the two Arduino modules via a single USB cable. The cable goes to a USB HUB which provides four ports and the two Arduino were connected to two of these ports.

Any number of modules may be connected with any of the HUB configuration and USB ports.

You can use long USB cables up to a dozen meters, because we use a slow communication (with respect to the capacity of the USB).

The two modules were given the names "Test Generic" and "ZetaTest"

The names are stored in the Arduino themselves, to recognize them even if you exchange the USB ports.



Arduino and Master modules comparision

Advantages

- With Arduino is possible to add sensors connected in I2C, SPI or serial. We have prepared the "generic" types to communicate with the PC. So you just write a few lines and import a library to read even the strangest sensors.
- Even those with little programming knowledge can add some simple function. Instead modify the firmware of our Master is almost impossible. Not even an expert could do it, would take too long and it would not be worth.
- With Arduino you can program small operations to be performed quickly, for example, turn high an output and lower it after a time of a few microseconds. These same operations are not possible with a master module, because everything must be programmed on the PC and the minimum time allowed by the USB exchange is of a millisecond or two.
- An Arduino Nano costs less than half of a Master. It is also located less than three Euros on eBay.

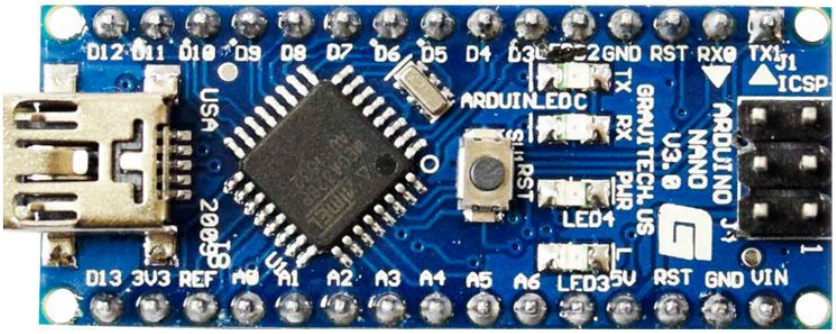
Disadvantages

- The simplified Arduino language (the Sketch) allows easy control of its firmware. But this leads to a great loss of performance. One example: the ADCs of our Master are read 16 times and is then averaged, while Arduino can read them only once, and is already a considerable load. To which the Adc Master give almost 14 bits (over-sampled), while those of Arduino only 10 bits.
- The communication speed with the PC is only a quarter of our Master even under the best conditions (about 200 exchanges per second against the normal 800 ... 900 of the Master).
- The pin can be configured with just a dozen kinds against almost thirty of the Master.
- Many types of Pin, such as capacitive keys, FastPwm and Stepper, are not feasible, the benefits are too scarce, better to use a Master.
- Nothing CapSensor and then nothing Theremin (musical instrument) with the features we're used to.
- The Adc24 is not connectable. We have not tried, but we know from experience that the SPI communication with the Adc24 module requires timings difficult to obtain. We did it on our Master, because we have all the possibilities given by a very powerful processor. And to do that we had to build a very complex set of Ping-Pong mechanism between two interrupts. Things like this are virtually impossible on Arduino (or at least we do not have the slightest idea of how to do it).

ATTENTION: You have to study well what you write in the loop Arduino, because any delay is reflected in the rate of exchange with the PC. Just one break of 100 mS to degrade the exchange rate from 200 per second to less than 10.

The Arduino Nano modules

We recommend using the Arduino Nano that is the most similar to the Theremino System Master modules.



The Arduino Nano is small and inexpensive, but has good performance and all our firmware has been developed for him. We tried other models, also more expensive, but they were worse. The Nano normally reaches above 200 exchanges per second, while other models (for example the Mega) do not even reached 20. And some models (for example Zero, Leonardo, Micro, Esplora, Yun and others), do not have the “serial event” function and will not work at all.

We gladly leave to Arduino experts the pleasure of doing experiments with Arduino and to modify our firmware to make them work. But be careful, to do a good job you will have to modify the ArduHAL application too.

We should also buy the base with the screw connectors. The Arduino modules do not have the +5V and GND placed comfortably on Pins as our Master, so it becomes difficult to connect the sensors and actuators.

With the base you can connect, for example, a Servo skinning their wires and screwing them. Not quite as comfortable with the Master but at least you can do it without welding.

The Arduino Nano has inputs and outputs that work between 0 and 5V. On the connectors there is a Pin with 5V and one with 3.3V to power sensors that need it.

The firmware for Arduino

Our system opens new possibilities for those who want to use the Arduino as Input Output for PC.

With the, over one hundred, Theremino System applications you can make measurements, automation, musical instruments, games and experiments of all kinds, both for teaching and for scientific laboratories.

Many have abandoned Arduino in a drawer after he realized that program it to communicate with the PC is not easy. Now these abandoned modules may be recovered.

For years we have waited for some Arduino expert prepare a communication protocol with our system. But in the end we had to learn the Arduino language and write it ourselves.

Firmware Features

Our firmware occupies only 25% of the program memory and 30% of the dynamic memory of an Arduino Nano. So more than two-thirds dell'Arduino remain free to add other libraries and user firmware.

In this 25% we did stay all the serial communication protocol, as well as the management of the most common types of InOut, including servo motors and the library "Servo" that implements them.

Firmware Structure

We have implemented the firmware for the communication with the ArduHAL in a library.

A library can easily be loaded by accessing the "Sketch" menu and using the "include library" command and finally choosing "Add library from ZIP file" (which you download from ***this page***).

Moreover, a library can be shared by many projects and therefore makes it easier to create new projects with parts of firmware written by the user.

Main File Structure "ThereDuinoV1.ino"

Our class is initialized and runs completely automatically, it just needs the following parts in the "ThereDuinoV1.ino" file:

- At the beginning of the file must be the line: `#include <Theremino.h>`
- In the "Setup" function the serial port is initialized. Normally the baud rate is 500,000, you can change it but it must be the same as you set in "CommOptions", accessible through the ArduHAL application “File” menu.
- The "Loop" function is completely empty and can be used by the user to write its firmware and possibly the functions "GenericRead" and "GenericWrite" to communicate numerical data with NetHAL (read the page of this document that explains Generic functions).
- Pay attention to what you write in the Arduino loop, because any delay is reflected in the exchange rate with the PC. A pause of 100 mS is enough to degrade the exchange rate from 200 per second to less than 10.

Write the firmware in the Arduino module

First, you must download the Arduino IDE from this page:
https://www.arduino.cc/en/main/software

Who has little experience can choose the "Windows Installer." Otherwise it is better to download the "Windows zip" version, which can then be unpacked in any folder and used on site without installing it. In addition, the zip version can easily be moved or copied from one PC to another. His only flaw is that you have to do everything manually and then knowing how to use ZIP, folders and shortcuts.

And here's the sequence to write the Arduino firmware

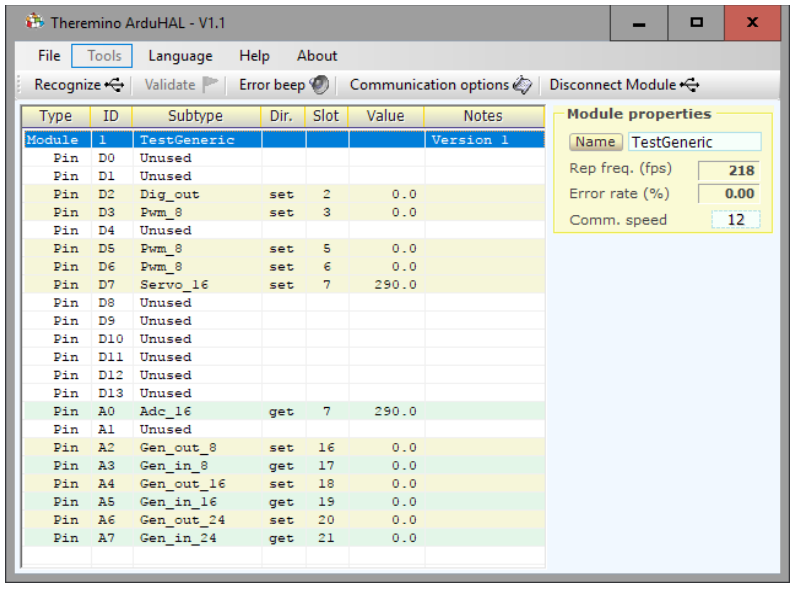
- Download the "ThereminoLibrary.zip" from ***This Page***.
- Start the Arduino IDE
- Open the “Sketch” menu, “#include library” and choose “Add library from ZIP file”
- In the dialog box that opens, locate the folder that contains “ThereminoLibrary.zip”, select it and press “Open”.
- Open the “File” menu and press "Open".
- Find the Theremino library (probably in “Documents/Arduino/Libraries).
- Open the "Tools" menu

  Select the module type (probably Arduino Nano),

  Select the processor (probably ATmega328P)

  Select the COM port. To figure out which is the right port, try unplugging and replugging the USB Arduino and each time open the "Tools / Port".
- Open the "Sketch" menu and click "Upload"
- If all went well, the green bar will grow up right and the Arduino is programmed.

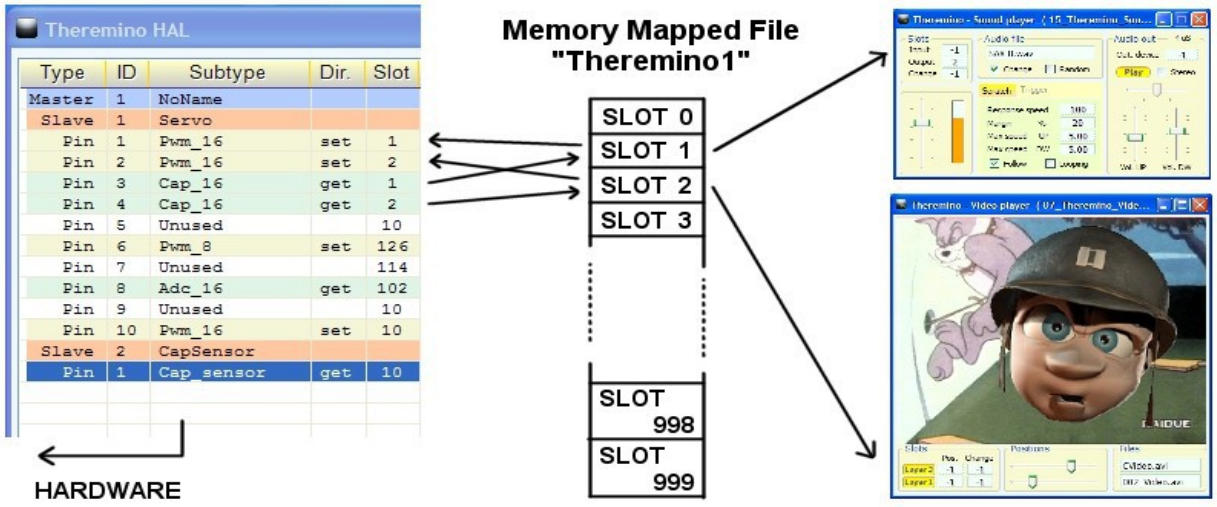
Finelly launch the ArduHAL application and must appear twenty-two “Unused” lines, as in this image.



Type	ID	Subtype	Dir.	Slot	Value	Notes
Module	1	TestGeneric				Version 1
Pin	D0	Unused				
Pin	D1	Unused				
Pin	D2	Dig_out	set	2	0.0	
Pin	D3	Pwm_8	set	3	0.0	
Pin	D4	Unused				
Pin	D5	Pwm_8	set	5	0.0	
Pin	D6	Pwm_8	set	6	0.0	
Pin	D7	Servo_16	set	7	290.0	
Pin	D8	Unused				
Pin	D9	Unused				
Pin	D10	Unused				
Pin	D11	Unused				
Pin	D12	Unused				
Pin	D13	Unused				
Pin	A0	Adc_16	get	7	290.0	
Pin	A1	Unused				
Pin	A2	Gen_out_8	set	16	0.0	
Pin	A3	Gen_in_8	get	17	0.0	
Pin	A4	Gen_out_16	set	18	0.0	
Pin	A5	Gen_in_16	get	19	0.0	
Pin	A6	Gen_out_24	set	20	0.0	
Pin	A7	Gen_in_24	get	21	0.0	

The "Slot"

The Theremino System Slots are identified by a number from 0 to 999 and are all part of a MemoryMappedFile called "Theremino1". Each Slot contains a "Float" number, which can be read or written by any module of the Theremino System.



In this picture, only HAL writes in the Slots, but in reality all the components of the system can both read and write in any of the Slots, even if already used by others.

Choosing the right Slot, you should be aware of two things:

- Ensure you do not use the same Slot by mistake, for two different functions.
- Avoid writing on the same Slot, with two or more components.

Input Pins, that are writing in the Slots, are indicated with “get”. If two of more input pins have the same Slot, then the HAL application warn with red lines and the text **SLOT CONFLICT**.

Many applications and Pins can read from the same Slot. But avoid configuring more than one Pin writing on the same Slot; doing so nothing is broken, but the results are unpredictable.

Sending multiple streams of data to the same Slot, all data are mixed and the last who writes wins. if you want to merge data in order, some rules are required.

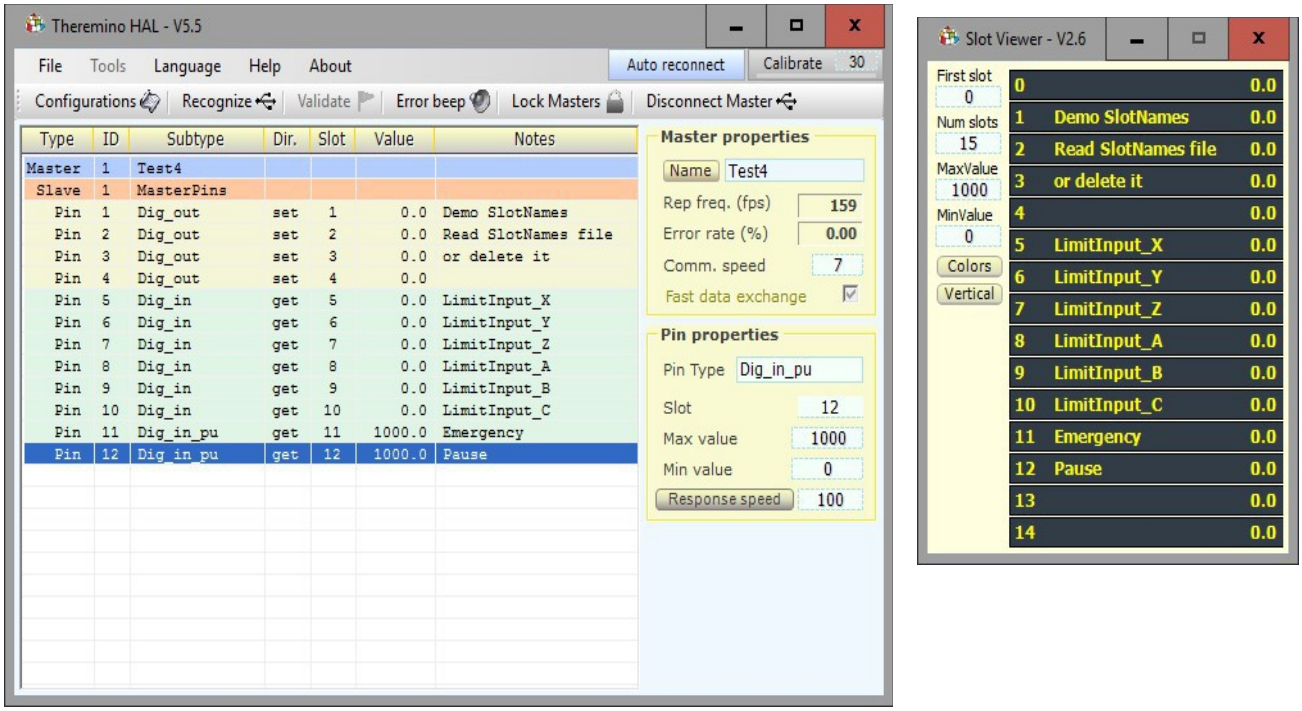
Type	ID	Subtype	Dir.	Slot	Value	Notes
Master	1	TestSlotCo...				
Slave	1	MasterPins				Firmware V5.0
Pin	1	Adc_16	get	1	105.3	
Pin	2	Adc_16	get	2	99.5	
Pin	3	Dig_in	get	4	0.0	SLOT CONFLICT
Pin	4	Dig_in	get	4	0.0	SLOT CONFLICT
Pin	5	Dig_in	get	5	0.0	
Pin	6	Dig_in	get	6	0.0	
Pin	7	Dig_in_pu	get	7	1000.0	
Pin	8	Unused				

To establish mathematics and logic rules between the Slots and to write complex behavioral algorithms, as well, we use Theremino Automation or Theremino_Script, or else any other programming language, such as C + +, CSharp, VBnet or VB6. Visual languages like MaxMSP, Processing, PureData, LabView and EyesWeb, can also be used. Plugins and examples for MaxMSP, are ready made here:
www.theremino.com/en/downloads/foundations

For more information about communication, please check this page:
www.theremino.com/en/technical/communications
www.theremino.com/en/technical/pin-types

The Slot names

All the HAL applications and even SlotViewer can view the names of the slot (or annotations or comments).



Important to note that the names are not related to physical Pins, but to the Slots.

The names are written in a file, that should be called “SlotNames.txt” and that must be in the same folder as "Theremino_ArduHAL.exe folder" and “Theremino_SlotViewer.exe”. If the file “SlotNames.txt”

If the file “SlotNames.txt” is not present then the comment field will remain empty.

To modify the Slot names you open the “File” menu, choose “Edit slot names file”, and edit it with the default system editor (normally NotePad). Finally you save the file and it will be reloaded automatically.

The rules are simple and are shown in the sample file, located in the latest versions of HAL and SlotViewer.

Each line of the file begins with the Slot number, followed by a space and the text to be displayed. The line can also continue with a comment, that does not appear, preceded by a single quote.

If you want to use the same file of comments, for both HAL and SlotViewer, you have to keep the files “SlotNames.txt”, “SlotViewer.exe” and "ArduHAL.exe", all in the same folder.

The "Slot Zero" commands

Other applications of Theremino system, or applications created by users, can send HAL commands.

Some applications may also change the parameters of all the Pin. To do this they should rewrite the configuration file and then send the "Recognize" command.

Currently two commands are defined:

- Command 1 = Recognize
- Command 2 = Calibrate (only used by modules with Pin CapSensor CapTouch or type)

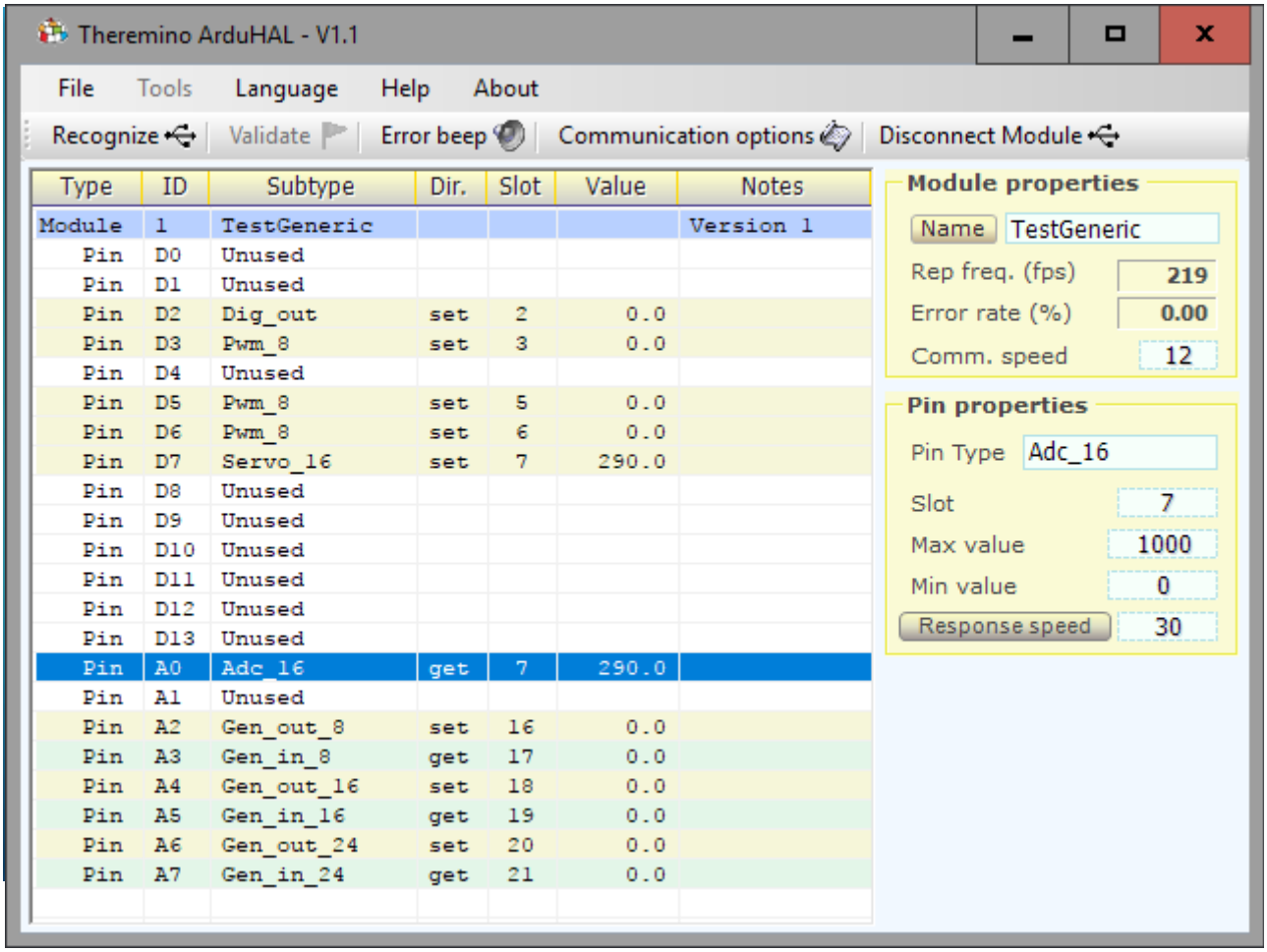
For safety, the commands must be preceded by a sequence. The sequence consists of two numbers (333 and 666) that correspond actually to the floating-point numbers, with seven digits of precision, 333.0000 and 666.0000. So it is virtually impossible for an ADC and other devices pose send this sequence casually. Anyway the Slot Zero should be reserved for the controls and therefore you should not assign it to the Pin and not use it to communicate numerical data between applications.

To send the "Recognize" command you must write:

```
--------------------------------------- VBNET
Slots.WriteSlot (0, 330)
System.Threading.Thread.Sleep (50)
Slots.WriteSlot (0, 660)
System.Threading.Thread.Sleep (50)
Slots.WriteSlot (0, 1)
--------------------------------------- CSharp
Slots.WriteSlot (0, 330);
System.Threading.Thread.Sleep (50);
Slots.WriteSlot (0, 660);
System.Threading.Thread.Sleep (50);
Slots.WriteSlot (0, 1);
--------------------------------------- Theremino Automation
Slot 0 = 333
Wait Seconds 00:05
Slot 0 = 666
Wait Seconds 00:05
Slot 0 = 1
--------------------------------------- Theremino Script
WriteSlot (0, 330)
Threading.Thread.Sleep (50)
WriteSlot (0, 660)
Threading.Thread.Sleep (50)
WriteSlot (0, 1)
```

The 50 milliseconds waiting instructions are used to give time to the HAL to read the Slot.

The HAL colors



The color scheme helps to recognize the components and their configuration

The Arduino (with TestGeneric name) provides twenty-Pin:

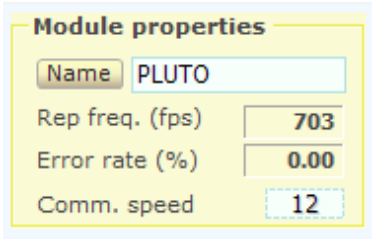
The Pins D0, D1, D4, D8 to D13 and A1 are "not used" and have a white background

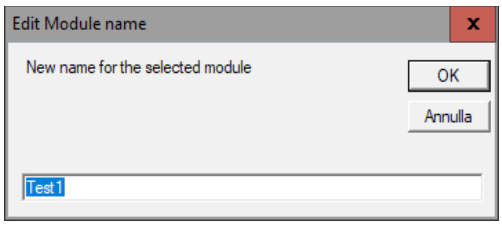
The light yellow background Pins are outputs (DigOut, PWM, and servo GenericOut)

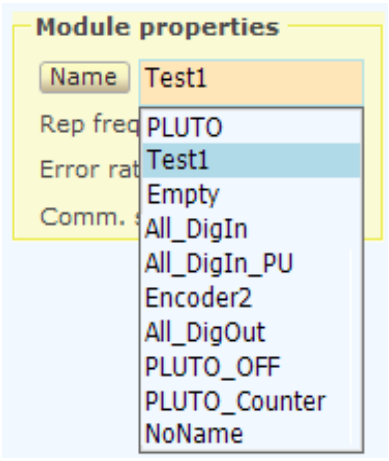
The light green background Pins are inputs (DigIn, ADC and GenericIn)

The blue line "Pin A0 Adc16" is the selected line and its properties are shown on the right

Module properties - The name







The selected module name can be changed in two ways:

- Pressing the "Name" button and changing its name.
- Clicking on the name box and choosing a different configuration from the drop down menu.

The module name It is written to the hardware module and is used to recognize when you reconnect it.

A newly attached module is called "NoName" or “YYYYYY”. We suggest you to rename the card differently, to distinguish it from all the others.

While dialing the name, the letter case (uppercase or lowercase) does not count.

If in the data base there are two modules with the same name, then the first configuration is used for both the Masters. It is therefore important to give different names to each module (unless you want to have a replacement module with the same name as the main one)

The modules are always listed in alphabetical order, so if you change the USB port then the module order does not change.

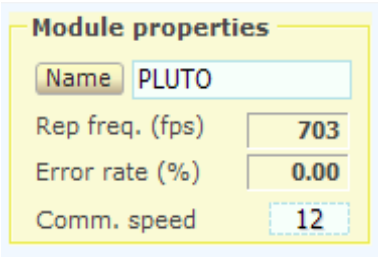
- - - - -

The HAL program almost always manages to use the right configuration when disconnecting, replacing and restoring components, but if you change the module names using a different computer or with another application (HAL located a separate folder - with separate parameters) or in other difficult and complicated cases, then the alignment between hardware and configurations could be lost.

If you lose the alignment you should restore the configuration manually, a Pin at a time, but experts can edit the configuration file and possibly copy this file entirely, or only a part of configurations, from a HAL application to another, on another computer or to another folder.

When the configuration is invalid to change the name of the module does not change the configuration file, but only the name written in hardware. You can then change the names of the modules to match them to the right ones in the configuration.

The module properties - Communication



- Number of communications per second
- Percentage of errors on the serial line (usually zero)
- Communication Speed Adjustment

The number of messages per second "Fps" should normally be above 150 and often above 200, by increasing the number of Pin used this frequency may decrease slightly.

For many applications, a 100 fps is more than enough, for some applications it is good to keep fps high as possible, at least 200.

Applications that require maximum bandwidth, such as WaveAnalyzer can not use Arduino modules but must necessarily use a Master or a NetModule.

The percentage of errors normally is zero. You should have errors only in case of severe electrical noise or if you select a baud rate too high.

Tips to increase the "Fps"

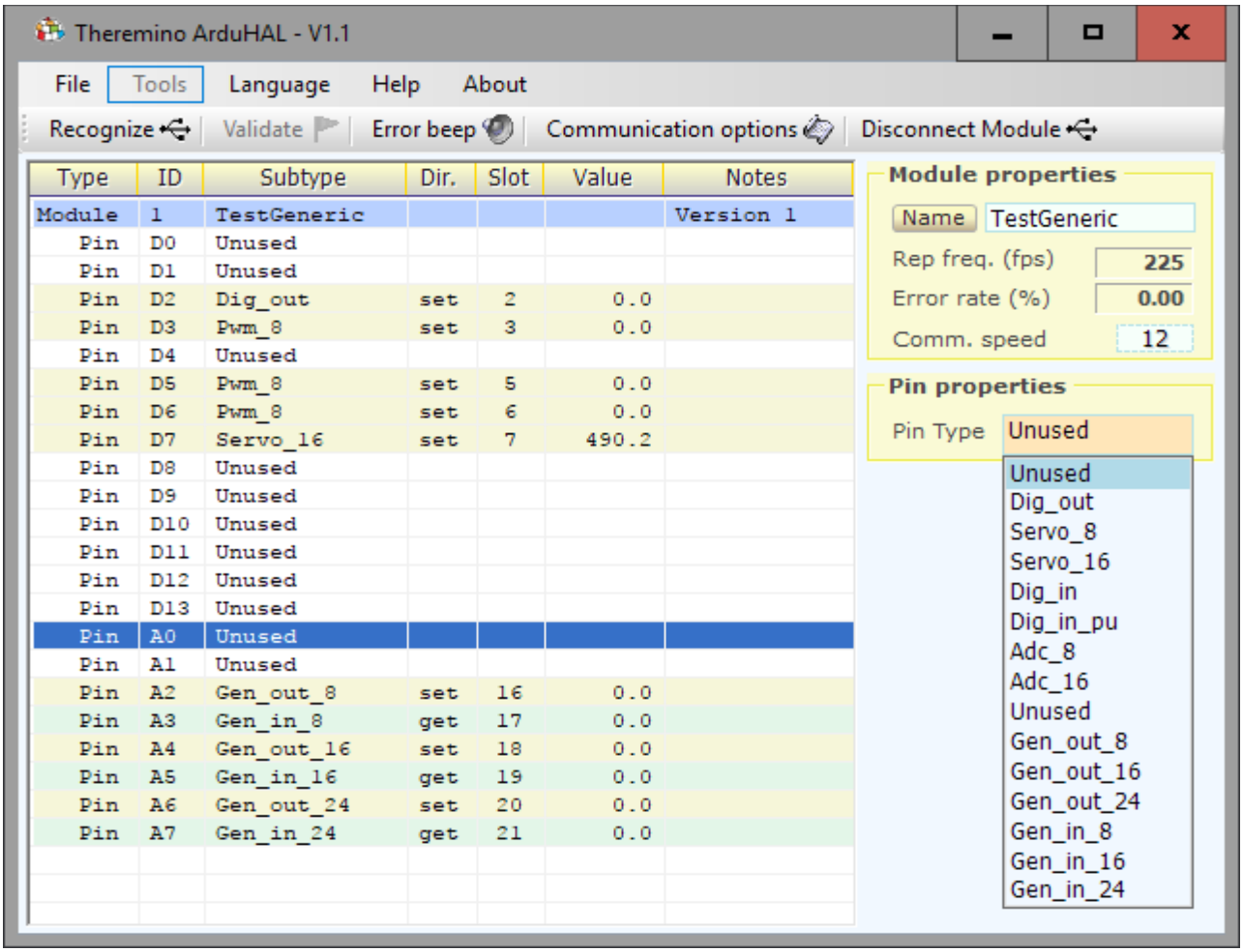
- Reduce the Arduino processor load (any instruction that wastes time in the Arduino "Loop" slows down the communication).
- Increase the "Comm Speed".
- Decrease the number of bytes used by configuring as "Unused" all possible Pins.
- Decrease the number of bytes used by configuring with 8-bit all the Pins that do not require greater resolution.

Adjust the "fps" Frequency:

With the "Comm speed" value you can adjust the refresh rate "fps".

To increase the response speed would be good to maximize the exchange rate, and set "Comm Speed" to "12". But for many applications exchanges percent per second it is more than enough, so you can usually lower "Comm Speed" and charge less the PC CPU.

The Arduino modules and their Pin



The Pin are almost all the same and can be configured in many different ways.

- D0 ... D13 can not be configured as ADCs.
- A0 ... A7 can be configured in any way, including ADC.
- Only Pin D3, D5, D6, D9, D10 and D11 can be configured as PWM.
- If using pin type Servo then D9 and D10 are no longer used as PWM.
  We have not blocked these Pin Arduino because some models have diefferent limitations.
  More information about the Servo library on ***This Page***

Our firmware was written and tested to the Arduino Nano.
On some models there may be some differences Arduino
and to get the maximum performance you should modify the firmware.

The types of Pin

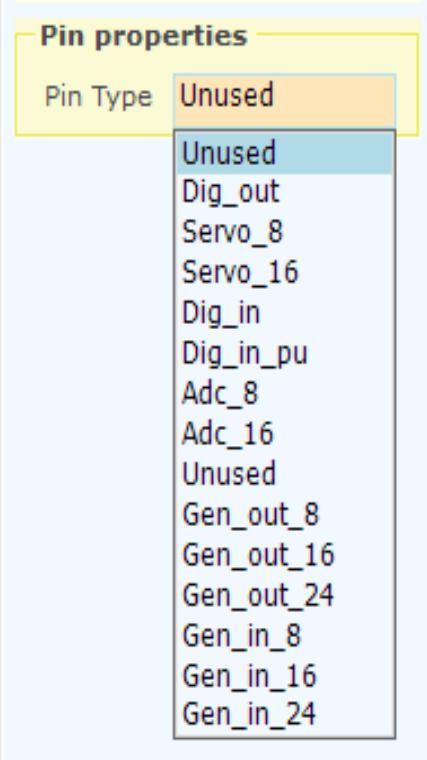
The Pins can be configured as:

- Not used

- Digital Output
- PWM output (490 or 960 Hz, depending on the pin)
- Output for servo-motors.

- Digital input
- ADC Input for potentiometers and transducers
- Counter, frequency and period inputs

- Generic inputs and outputs with 8, 16 or 24 bits



Special Pin:

- The D0 and D1 Pins accept Gen_in and Gen_out only
- The D2 to D13 Pins accept all types, except the ADC
- The A0 to A7 Pins accept all types, including ADC
- Only Pins D3, D5, D6, D9, D10 and D11 can be configured as PWM
- When using Servo Pins then D9 and D10 are no longer working as PWM

There are also the "Generic" Pins that can be used to communicate between the firmware and the PC. By means of these types it is possible to add some firmware and exchange numerical values through the application ArduHAL and finally with the slot of the Theremino system. You may for example read sensors connected in I2C and send values to your PC. More information on generic Pin in the page ***The Generic pins***.

- - - - - - - - -

All pin can also be configured as "Not used". This allows to decrease the number of bytes passing on serial and USB line and maximize the number of exchanges per second.

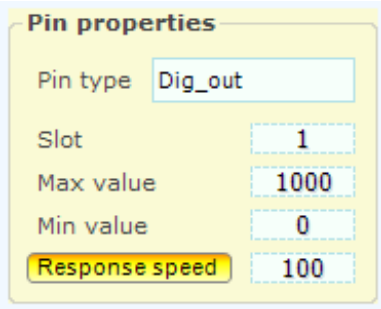
The choice between 8 and 16 bit, Available for some types of Pin, it allows to have the maximum resolution (16 bit) or a lower resolution (8 bits), but a higher bit-saving and thus a higher communication speed.

The types with pullup, Whose name ends in "_pu", allow to easily connect switches, buttons and open-collector devices, without having to add external resistors (pullup current typical = 250 uA).

These types of Pin are valid for **Arduino Nano.**

Other Arduino models have similar capabilities but we have not tested them.

The parameters common to all Pins



"Slot"indicates where to write or read data. The slots are a thousand, numbered from 0 to 999, and can be read or written by all Pins and all the Theremino System applications.

Please note: Many applications and Pins can read from the same Slot,
but avoid to configure more than one Pin, writing to the same Slot.
Doing so doesn't damage anything, but the results are undefined.

"Max value" normally set to 1000, indicates the value that the Pin must have, when at its maximum.

"Min value" usually set to zero, indicates the value that the Pin must have, when at its minimum.

By adjusting Max and Min, with values other than 0 and 1000, you can achieve any scale ratio and calibration. If you exchange the two values (min value larger than max), then the scale is reversed, this is useful to reverse the movement of the actuators or to turn the readings of sensors, that act reversed.

"Response speed" adjusts the filter IIR (Infinite Impulse Response) for the best compromise between noise and response speed. With a value 100, the filter is disabled and the maximum speed of response is obtained. The value 1 produces the maximum filtering, (elimination of any jitter) but a very slow response (approximately one second). Normally we use the value of 30, which provides a good filtering and a fast enough speed.

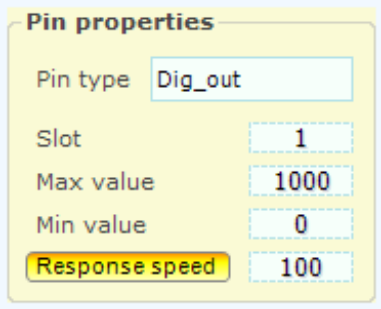
If the **"Response speed"** button is pressed, the IIR filter adapts to variations in order to obtain a higher reactivity, when there are wide variations and a greater damping, when the changes are minor. As a result you get a good stability of the digits, without too much sacrificing the settling time.

Some sensor signals may malfunction with **"Response speed"** pressed. This is specially true for sensors producing a signal with little variations around a high base value. In this case the signal never arrives to the final value or is very slow to arrive. If you experience this, disable **"Response speed".**

The IN-OUT range of 0V to 5V is valid for the Arduino Nano, other modules could be different.

The "Output" Pin types --> Dig / PWM / Servo

Dig_out

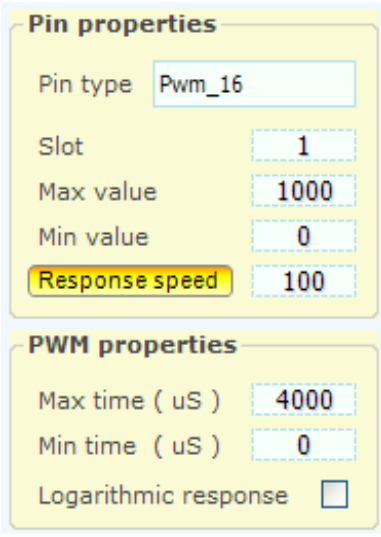


This type of pin provides a digital output.

The value coming from a Slot, limited between "Min value" and "Max value" and filtered by "Response speed", is compared with the value between "Min value" and "Max value". If exceeded, the Pin turns on, otherwise off.

The Pin can only assume voltages 0V (off) and 5V (on) and the output current is limited to approximately +/- 10 mA

Pwm_8 and Pwm_16



This type of Pins provide a PWM (pulse width modulation) output.

The value coming from a Slot, limited between "Min value" and "Max value" and filtered by "Response speed", is converted to pulses of width between "Min time (uS)" and "Max time (uS)"

The frequency of the pulses is 2000uS (500Hz) fast enough to turn on a LED with variable intensity. For users who require a real variable voltage a low pass filter is added, usually composed of a resistor and a capacitor.

The Pin provides pulses between the voltages 0V (off) and 5V (on) and the output current is limited to approximately +/- 10 mA

Servo_8 and Servo_16

This type of Pin controls servo motors directly.

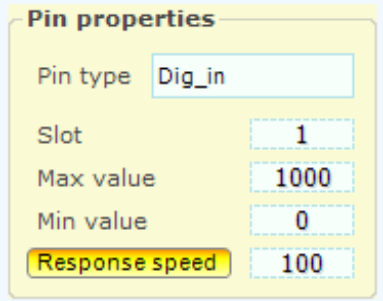
The value coming from a Slot, limited between "Min value" and "Max value" and filtered by "Response speed" is converted to pulses of width between "Min time (uS)" and "Max time (uS)"

The pulses repetition time is adjusted to normal aero-model servo, spinning around 180 degrees, between the min and max time.

The Pin provides 0V to 5V, adequate for normal 4 to 6 volts servos and a current sufficient to drive dozens of parallel servo.

The types of Pin in "Input" <-- Dig / ADC

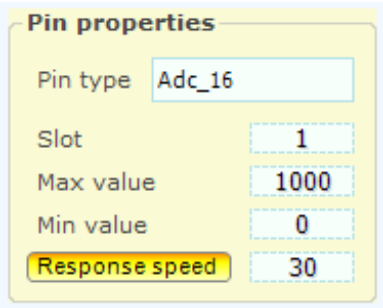
Dig_in and Dig_in_pu



This type of Pin provides a digital input.

The voltage value is read with a Schmitt Trigger, with low threshold = 1.5V and high threshold = 3V, and transformed into a On/Off information and finally to "Max value" or "Min value". The value is then filtered with "Response speed" and finally written into the Slot. The filtering produces intermediate values, roughly proportional to the ratio of time, between On and Off.

Adc_8 and Adc_16



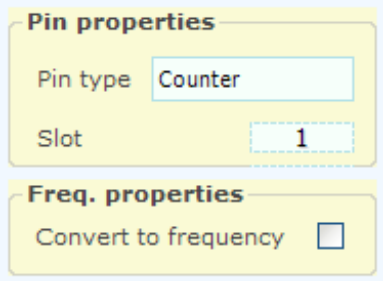
This type of Pin provides an analog input.

The voltage value from 0V to 5V is converted into a number between "Min value" and "Max value." The value is finally filtered with "Response speed" and then written into the slot. The filtering reduces the noise present in the input signal, but slows down response. The value 30 represents a good compromise between speed and noise.

The "Input" Pin types <-- Counter

This type is not currently implemented in the Arduino firmware.

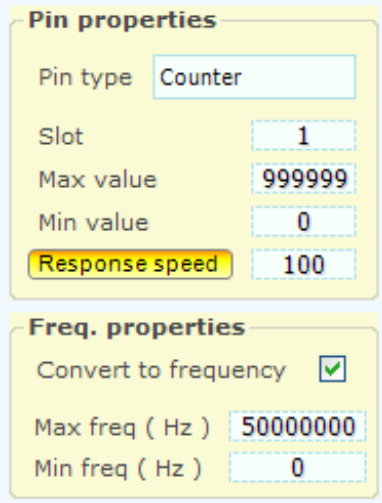
Counter and Counter_pu



All Pins can be programmed as Counter or Counter_pu but the maximum counting speed is quite limited, around a few KHz, depending on the load on the microcontroller and the duty-cycle of the signal.

If you need a higher speed you have to use a FastCounter on Theremino Master.

Counter and Counter_pu with the "Freq" option



Pins programmed as Counter or Counter_pu, can be transformed from counters to frequency-meters.

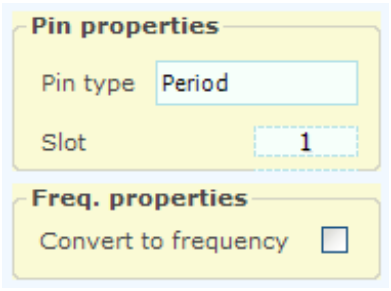
The frequency value, limited between "Min Freq" and "Max Freq", is then compared between "Min value" and "Max value" filtered with "Response speed" and finally sent to the Slot.

The "Counter" and "Counter_Pu" Pin-types use 16 bits for data transmission.

The "Input" Pin types <-- period

This type is not currently implemented in the Arduino firmware.

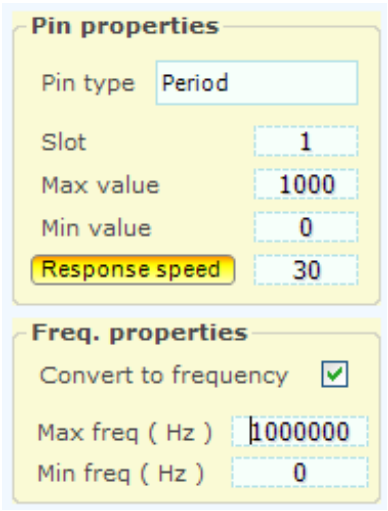
Period and Period_pu



This type of Pin, measures the period of a repetitive waveform, from peak to peak, up to a maximum period of about 260 seconds.

Resolution is half a microsecond and accuracy is +/-1%, over a range of temperature from 0C to 50C.

Period and Period_pu with the "Freq" option



Pins programmed as Period or Period_pu can be transformed from counters to frequency-meters.

This technique, allows to measure very low frequencies (up to about a tenth of a Hertz) with very high resolution.

The value of frequency, limited between "Min Freq" and "Max Freq", is compared between "Min true" and "Max value", filtered with "Response speed" and finally sent to the Slot.

The "Period" and "Period_pu" Pin-types use 32 bits for the data transmission.

The "Generic" Pin types

What we explain in this page is only for read or write special sensors or actuators, such as motors or sensors that communicate over I2C. Or to pre-condition the data with its own firmware before sending it to PC or from PC to the actuators.

To read and write the normal inputs and outputs is not necessary to use the functions explained in this page, just set up the Pins with the ArduHAL and they work immediately.

```
uint32_t n;

n = Theremino.genericRead8 (0);
Theremino.genericWrite8 (1, n);

n = Theremino.genericRead16 (A0);
Theremino.genericWrite16 (A1, n);

n = Theremino.genericRead24 (A2);
Theremino.genericWrite24 (A3, n);
```

In the GenericWrite and GenericRead functions you must indicate a Pin to use and that same Pin must be configured in the ArduHAL with the corresponding types Gen_in or Gen_out. If you make mistakes you will have unpredictable results.

The Pin are all valid to communicate "generic data". For the Pin name you can use a number from 0 to 21. For the last eight Pin you can also use the names A0 to A7. We recommend that you use D0 and D1 first, which can not be used as physical Pins.

When in the ArduHAL you set a Pin as generic, the correspondent physical Pin is no longer read (or written) by our firmware, but you will have to write the necessary firmware. So to use these functions you must know how to plan and organize without errors and it takes a good experience in programming.

Send data from the sensors to the ArduHAL, and finally to the PC applications

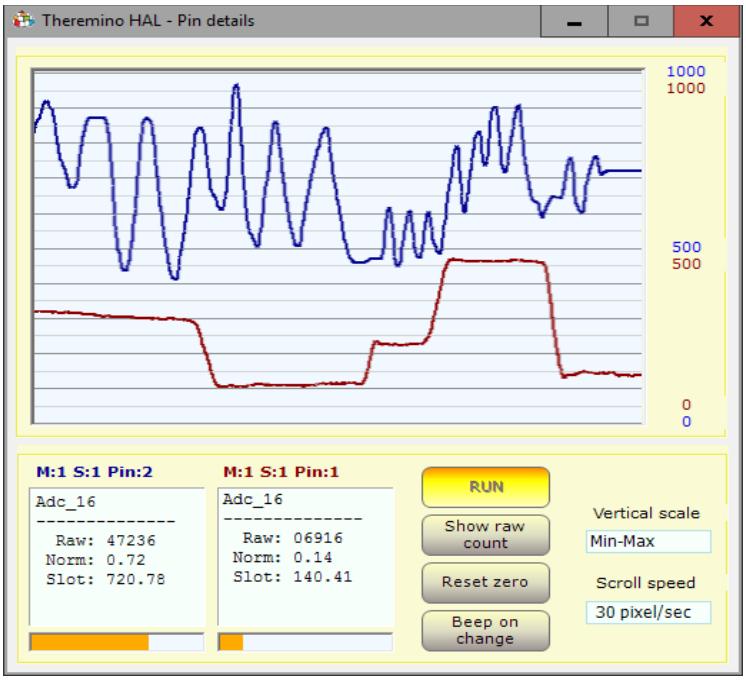
You choose a Pin
In the Arduino Loop you read the sensor and interpreting its data
In the Arduino Loop you send data with "GenericWrite" (8, 16 or 24)
In the ArduHal you get the data by configuring the Pin as Gen_in (8, 16 or 24)
In the ArduHal you set the Pin with a Slot through which the data of the sensor will go to the applications that use it.

Send data from PC applications to the ArduHAL PC, and then to the hardware

On the ArduHAL you choose a Pin
On the ArduHal you set the Pin with a Slot through which to get numeric data form PC apps.
On the ArduHal you send data to the Arduino by configuring the pin as Gen_out (8, 16 or 24)
In the Arduino Loop you read the data with GenericRead (8, 16 or 24)
In the Arduino Loop you interpret and translate the numeric data and send it to the hardware.

The "Read an I2C sensor" document that, you download from ***This Page***, is a good example of using Generic Pins.

The “Pin details” viewer



With a double-click on a active Pin line, this instrument is opened. To display two signals click on the first Pin then on the second Pin, with a single click.

The vertical scale can be set to Min-Max, that are the Pin settings MinValue and Max Value. Or it can be set to 24 levels from 0.01 to 50000 points for each vertical division (ten dark lines).

When the vertical scale is not set to Min-Max, the button "Set Raw zero" centers the traces.

In some cases it may be useful to check the raw values. For "Raw" values, use the "Show raw count".

The "Scroll speed" adjusts the graph scroll speed, from 0.1 pixel per second to 60 pixel per second.

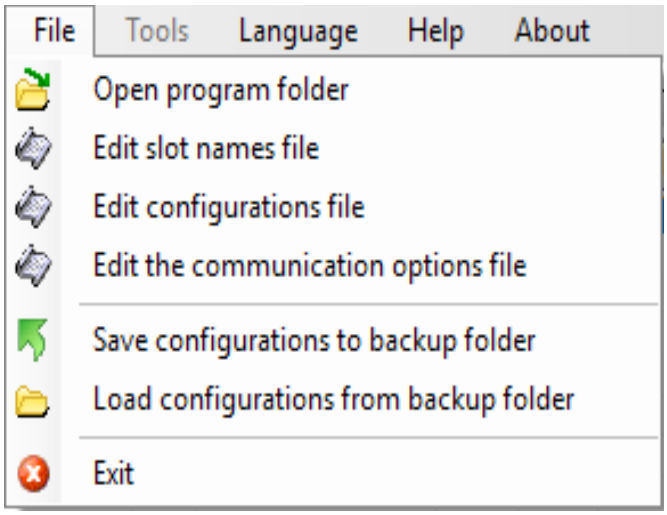
The two text boxes show the internal details of the Pins. In this image the text indicates what Module and Pin is. The text "M:1 Pin:2" means "Module 1, Pin 2".

The Pin details can help in the control and regulation of Output Input devices (sensors and actuators).

Some Pin types are more complex and have more intermediate values. Typically there is a "Raw" value with values very variable depending on the type of Pin, a "Normalized" value that always goes from 0 to 1, and a "Slot" value which normally ranges from 0 to 1000 and which is the simplified value available on Slots and easily usable by all the high-level software.

- **Raw** "Raw" value that can be a count, a time, a voltage or otherwise.
- **mS** Time in milliseconds
- **uSec** Time in microseconds
- **Smoot** Value that was passed in a FIR filter (only used in Cap8 and Cap16)
- **Mean** Average value (used in type Cap8 and Cap16 as zero calibration)
- **Norm** Normalized value between zero and one
- **Slots** Value written to or read from the slot associated with the Pin (normally 1 to 1000)
- **Out** Digitized value that can be processed only in "0" or "1" (used by DigOut only)

Menu commands



Open the program folder it may be useful to modify the documentation files and languages.

Edit the file: "SlotNames" Comments (or slot names) are explained on ***this page***.

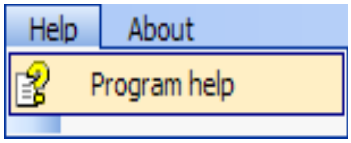
Edit configurations it may be convenient in some cases. For more info, please read "Questions and Answers" on the last page of this document.

Change communication options *the communication options file is opened to modify ports (Com1, Com2, etc ...), the speed (baud rate) and the names of modules to recognize. In the communication options file the choices are explained.*

Save the configuration as a backup *It allows to make the configurations of the security copies. If the configuration files are modified by mistake you can then load them from a previous version. Previous versions show the date and time they were saved.*

The language files are located in the "Docs" folder next to ThereminoHAL.exe application.

To make new language files just copy the file Language_ENG.txt, change "ENG" with "FRA", "ESP", "DEU" or "JPN" and edit text with Notepad.



This command opens the documentation files.

Commands Toolbar

Recognize
It recognizes the active modules.

Valid
When you add or subtract modules, you could be advised that the configuration has changed with red lines in the list. If you choose to lose your old configuration and adapt existing hardware, with this button you will make the new configuration effective.

Mistakes
When pressed communication errors are highlighted with a sound.

Communication Options
The communication options file is opened to modify ports (Com1, Com2, etc ...), the speed (baud rate) and the names of modules to recognize. In the communication options file the choices are explained.

Disconnect Module
Deletes the selected module from the list. This way you can delete unwanted modules without having to physically disconnect them from the USB.

Isolated Applications

Some Theremino system applications automatically launch its own HAL. This happens if there is a Theremino_HAL.exe in the folder ThereminoHAL located near to your application EXE file. You could also place Theremino_HAL.exe next to the exe file of the application, but it is better that the HAL has its own folder, with the “Docs” sub-folder containing documentation and language files.

These HAL use their own private configuration and if they have the "Master Lock" button, you can only connect to its Master, identifying them by name among those connected to the USB ports. An application composite in this way, will continue to operate even when copied to a different computer, and even if other Theremino System applications are connected with their Master, on other USB ports.

The applications that benefit most from these possibilities, are applications with a specific task, such as: Theremino Geiger, Theremino OilMeter, Theremino Weather, Theremino Theremin, Theremino Arm, Theremino Geo and Theremino EmotionMeter.

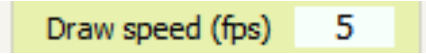
This does not mean that isolated applications can not communicate with each other. The modular communication is always possible and is done through the Slot, which are common to all applications.

To avoid using the same Slot for different tasks we have defined a broad pattern:

```
Experimental 100 slots 000-099
- - -
Theremino_Theremin 100-199
Theremino_SlotsToMidi 200-299
Theremino_MusicKeys 300-329
- - -
469 free slots 330-799
- - -
Theremino_OilMeter 800-809
Theremino_EEG 810-819
Theremino_Meteo 820-839
Theremino_Arm 840-849
10 free slots 850-859
10 free slots 860-869
10 free slots 870-879
Theremino_EmotionMeter 880-889
Theremino_Geiger 900-909
Theremino_Bridge 900-909
Theremino_GEO 910-919
Theremino_GeoPreampTester 920-929
Theremino_Radar 930-939
10 free slots 940-949
10 free slots 950-959
10 free slots 960-969
10 free slots 970-979
10 free slots 980-989
10 free slots 990-999
```

This scheme is only indicative. You can use the Slots as you like, providing that, the same Slots are not used for different tasks in the same PC. If you make a mistake does not break anything, but the data overlap with undefined results.

Adjusting the numerical boxes



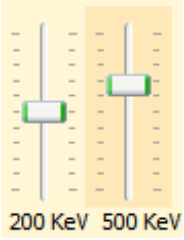
The HAL numerical boxes (and all other Theremino system applications) have been developed by us (note 1), to be more comfortable and flexible, than the original Microsoft TextBox.

The numerical values are adjustable in many ways
- By clicking and holding down the left mouse button and moving the mouse up or down
- With the mouse wheel
- By pressing the arrow-up and arrow-down keys
- With conventional methods used to write numbers with the keyboard
- With the usual selection and copy-paste methods

Moving the mouse up and down allows wide and fast adjustments
The mouse wheel allows a comfortable and immediate setting
The arrow keys allows fine adjustments without having to look away from what you are adjusting

*(1) Like all our software, their source files are available (Freeware and Open Source licensed under Creative Commons) and can be downloaded from here: **www.theremino.com/en/downloads/uncategorized** (See "Custom controls"). These controls can be used freely in any project without name a source. The "Open" sources also serve as a guarantee that we have not included malware.*

Adjusting the sliders



These are the original Microsoft cursors, they are pretty comfortable, so we just added the orange color and the possibility to reset them.

<<< Non-zero sliders are marked with an orange color, to reset them just click with the right mouse button (not all sliders have a zero, in this case they do not change color and cannot be reset with the mouse)

Sliders can be adjusted in the following ways
- Clicking the cursor with the right mouse button, to reset them
- Clicking the cursor with the left mouse button and moving the mouse up or down
- With the mouse wheel
- Using the left-arrow and right-arrow on your keyboard
- By pressing the up-arrow and down-arrow keys

The method of moving the mouse up and down, allows wide and fast adjustments.
The mouse wheel allows comfortable and immediate adjustments
The arrow keys allow fine adjustments without taking your eyes from what you are adjusting.
The arrow keys left/right or up/down have the same effect, it might be be more intuitive to use the first for horizontal cursors and the seconds for vertical sliders.

Questions and answers

Can I edit the text of the program panels in different languages?

Certainly, just edit the files: ".. \Docs\Language_Eng.txt" and ".. \Docs\Language_Ita.txt"
For languages German, French and Spanish just copy the English file three times with the following names: ".. \Docs\Language_Deu.txt", ".. \Docs\Language_Fra.txt", ".. \Docs\Language_Esp.txt" and edit them.

Can I edit the configuration file?

Normally the association between configurations and modules is kept aligned by the HAL, which uses the names of the modules to determine the proper configurations to be taken. Normally the HAL can use the right configuration even if you disconnect and replace modules.

In some cases, if you change your name to the modules with a HAL that is on a different computer, or to a different folder, then the you lose alignment between hardware configuration. In these cases, you can click on the drop down drop down the name of the form and restore the alignment by choosing the right configuration for each module.

To make more complex changes, you can open the "Theremino_HAL_ConfigDatabase.txt" file, with a text editor such as "Notepad", and manually edit the configurations that are quite simple.

How to reduce the CPU work?

- Close or minimize the "Component window details".
- Minimize the main window.
- Reduce the "Speed", as explained in the opening pages of this document.