

theremino
•the•real•modular•in-out•

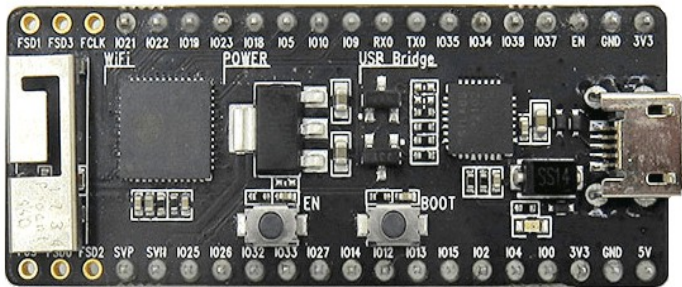
Theremino **System**

Theremino IoT HAL

V1.x

Instructions

Principle of operation

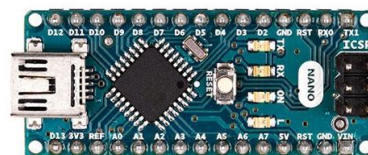


The lotModule (ESP32) works with 5 volts and communicate via WiFi with an Access Point, or directly with the PC equipped with a WiFi module that supports Mobile HotSpot function. For example, a notebook or a recent Tablet with Windows 10.

Using an Access Point as a bridge, any device with Windows can connect to lotModule.

You can then read the data on your PC and even transfer them from one to another lotModule. You could, for example, connect a potentiometer to one lotModule and with this rotating a servo motor that is located on another.

You can also transfer data to and from the Master modules, NetModules and Arduino modules, in any number and combination.



With the [lotHAL](#) application you configure the input-output Pins, to read sensors, move engines, etc...

Depending on the model, you can connect up to 26 or 28 sensors and/or actuators.

You can use over a hundred of Theremino System apps. , covering almost all fields, from scientific experiments to the music, to radioactivity, teaching methods... see [This Page](#).

Everything works quickly, without writing a single line of firmware or software, and without installations.

Type	ID	Subtype	Slot	Value	Notes
Module	1	Test_ADC			Version 60-1
Pin	36	Adc_16	0	0.0	
Pin	39	Adc_16	1	0.0	
Pin	25	Unused			
Pin	26	Unused			
Pin	32	Adc_16	4	195.4	
Pin	33	Adc_16	5	61.6	
Pin	27	Unused			
Pin	14	Unused			
Pin	12	Unused			
Pin	13	Unused			
Pin	15	Unused			
Pin	2	Unused			
Pin	4	Unused			
Pin	0	Unused			
Pin	37	Adc_16	14	0.0	
Pin	38	Adc_16	15	0.0	
Pin	34	Adc_16	16	0.0	
Pin	35	Adc_16	17	0.0	
Pin	1	Unused			
Pin	3	Unused			
Pin	9	Unused			
Pin	10	Unused			
Pin	5	Dig_in	22	0.0	
Pin	18	Dig_in	23	0.0	
Pin	23	Dig_in	24	0.0	
Pin	19	Dig_in	25	0.0	
Pin	22	Dig_in	26	0.0	
Pin	21	Dig_in	27	0.0	

Module properties
Name: Test_ADC
Rep freq. (fps): 812
Error rate (%): 0.00
Comm. speed: 12

First use of the modules

The first time you have to prepare the Arduino IDE, with this procedure

1) Install the Arduino IDE and start it.

2) Follow these instructions:

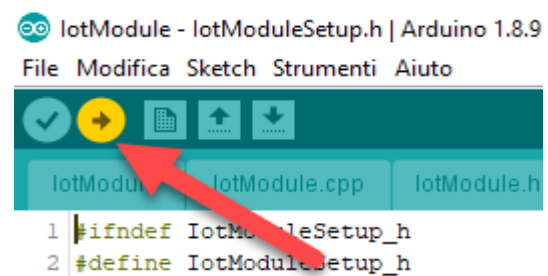
- Open "File" / "Preferences"
 - In line "Additional Board Manager URLs ..." copy the following line:
`https://dl.espressif.com/dl/package_esp32_index.json`
 - Go to "Tools" / "Board: xxx" / "Board Manager..."
 - Type "ESP32" in the upper text box, select it and install it.
 - Select "Tools" / "Board: xxx" / "ESP32 Arduino" / "ESP32 Pico Kit"
 - Select "Tools" / "Partition scheme" / "Minimal SPIFFS ..."
- (Instructions from: [Installing the board in the Arduino IDE ESP32](#))

To program the module, follow this procedure

- ◆ Start the Arduino IDE.
- ◆ Select "File" / "Open" and load the "IotModule.ino" file
- ◆ Select the file "IotModuleSetup.h" (in the upper bar)
- ◆ Set the name and the password of your WiFi network.
- ◆ Modify the IOTMODULE_PINOUT_TYPE line to follow the used module type (set 0/1/2 for WROOM / PICO-KIT / TTGO-T7)
- ◆ The other "Setup" options, are explained in [this page](#).
- ◆ Check the list of ports in the "Tools" / "Port", then connect the module, locate the new COM port, which has been added to the list, and select it.

If the COM port does not appears [read here](#).

- ◆ Program the module and wait a few seconds.
- ◆ Launch the "Theremino_IotHAL.exe" application and check that the module to appear on his list, as explained on the next page.



Connect to the module with the IoT HAL

When the module is powered and connected to the network, opening the IoT HAL application, you should see his Pin list, as in the following image.

The screenshot shows the 'Theremino IoT HAL - V0.5' application. The main table lists pins with columns: Type, ID, Subtype, Slot, Value, and Notes. The first row is a module entry: Type 'Module', ID '1', Subtype 'Test_ADC', Slot empty, Value 'Version 60-1'. Subsequent rows list individual pins (e.g., Pin 36, 39, 25, 26, 32, 33, 27, 14, 12, 13, 15, 2, 4, 0, 37, 38, 34, 35, 1, 3, 9, 10, 5, 18, 23, 19, 22, 21) with their respective InOut types (e.g., 'Adc_16', 'Unused', 'Dig_in') and current values. The 'Module properties' panel on the right shows fields for Name (Test_ADC), Rep freq. (fps) (812), Error rate (%) (0.00), and Comm. speed (12).

Type	ID	Subtype	Slot	Value	Notes
Module	1	Test_ADC		Version 60-1	
Pin	36	Adc_16	0	0.0	
Pin	39	Adc_16	1	0.0	
Pin	25	Unused			
Pin	26	Unused			
Pin	32	Adc_16	4	195.4	
Pin	33	Adc_16	5	61.6	
Pin	27	Unused			
Pin	14	Unused			
Pin	12	Unused			
Pin	13	Unused			
Pin	15	Unused			
Pin	2	Unused			
Pin	4	Unused			
Pin	0	Unused			
Pin	37	Adc_16	14	0.0	
Pin	38	Adc_16	15	0.0	
Pin	34	Adc_16	16	0.0	
Pin	35	Adc_16	17	0.0	
Pin	1	Unused			
Pin	3	Unused			
Pin	9	Unused			
Pin	10	Unused			
Pin	5	Dig_in	22	0.0	
Pin	18	Dig_in	23	0.0	
Pin	23	Dig_in	24	0.0	
Pin	19	Dig_in	25	0.0	
Pin	22	Dig_in	26	0.0	
Pin	21	Dig_in	27	0.0	

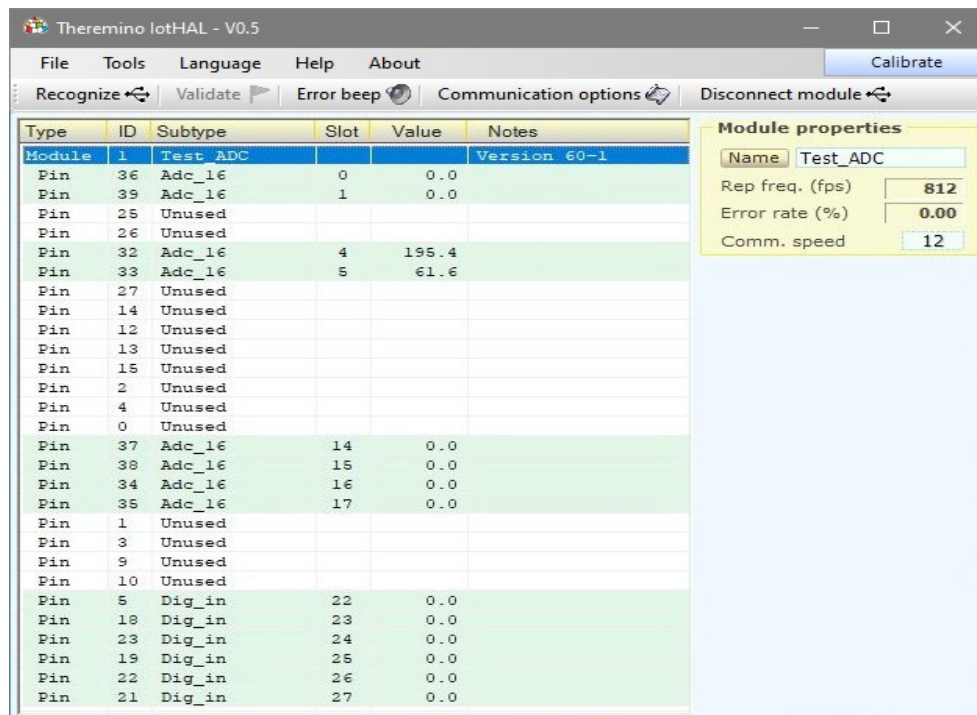
The first line is relative to the module, with his name and version. The first number "60" is the firmware version, and the second "1" is the module type (WROOM or PICO etc...)

The following lines are relative to individual Pins, with the Pin name, the Pin InOut type, the Slot at which it is associated, and the current value.

Selecting the first row you can change the properties of the module, and selecting the following lines you can modify the properties of the individual Pins.

If the form does not appear (list completely white) please retry with the button "Recognize". Try also to repeat the procedure on the previous page, checking in "IoTModuleSetup.h" file, that the WiFi network name, password and other properties are right. And possibly read also the advice in the last pages of this document.

Theremino lotHAL



Theremino lotHAL connected to a lotModule via WiFi

The Theremino lotHAL (Hardware Abstraction Layer), which you download from [This Page](#), It appears with a simple interface, but carries out complex operations with optimized algorithms.

lotHAL is the heart of communication with the hardware, can communicate with many lotModules simultaneously, knows the USB protocol and knows all the most common types of input-output.

Without the lotHAL communicate with the hardware would be difficult (as with the Arduino), would require much time and effort (as with the Arduino) and finally, for all types of InOut, such as move a motor or even light a LED, you should write the appropriate firmware (like Arduino).

In Theremino system, there are also three other HAL applications, the first is simply called HAL and communicates via USB with the Master modules, the second is called NetHAL and communicates via WiFi, network and Internet with the NetModule modules, the third is called ArduHAL and communicates with the Arduino modules.

Sometimes we will use the generic name HAL, to indicate all the four 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 applications of the system can communicate with each other, through the Slots, even without any 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

The application Theremino IoT HAL can communicate with any number of modules.

All Pin are read and written all at once and the maximum speed permitted by each module. In other words, a slow module does not cause a slowing of communication with others.

Example of IoT HAL connected to two IoTModule via WiFi

In this picture you see two IoT HAL modules named as "Test_ADC" and "Test_Pwm_8ch".

You may also notice that the two firmware versions are different, this is not always possible, but in this case were compatible.

If you connect other modules, they are recognized and listed below. If the number of lines exceeds the height of the window displays a vertical scroll bar that allows you to scroll through them.

The modules are recognized at the start of IoT HAL application, or by pressing "Recognize", if the application is already started.

The names are stored in the modules themselves. So even if you change networks, access points and IP addresses, the modules are recognized by name, and each module is configured with its configuration.

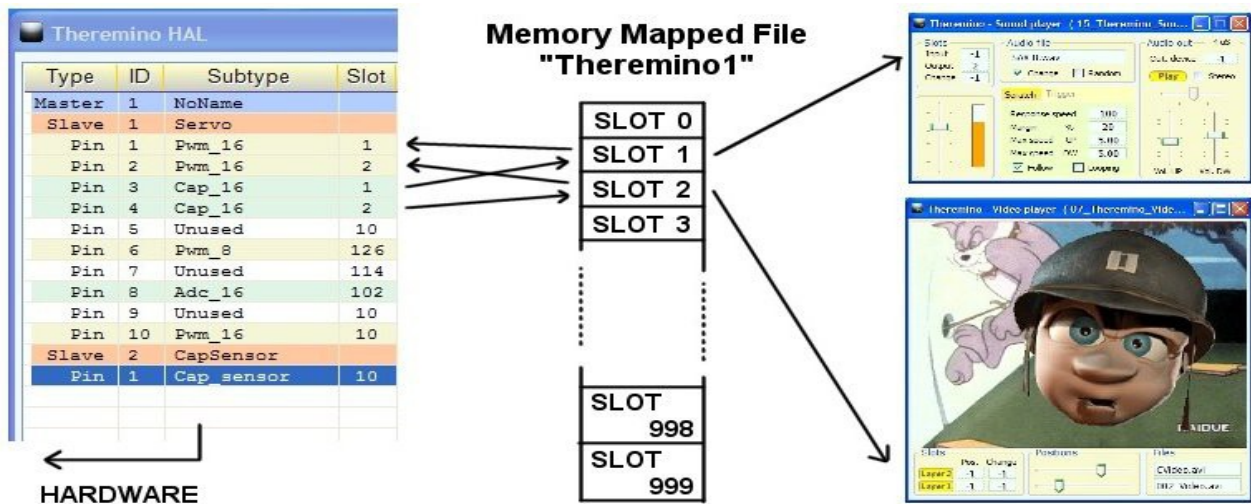
The module list is presented in alphabetical order, so the list will always appear in the same order, even if you swap IP addresses.

The screenshot shows the Theremino IoT HAL - V0.23 application window. The main window has a menu bar (File, Tools, Language, Help, About) and a toolbar with buttons: Recognize, Validate, Error beep, Communication options, and Disconnect module. The main area displays a table of connected modules and their pin configurations. The table has columns: Type, ID, Subtype, Slot, Value, and Notes. The first module is "Test_ADC" (Module 1, Version 107-0) and the second is "Test_Pwm_8ch" (Module 2, Version 109-0). The right sidebar shows the "Module properties" for the selected module, "Test_ADC", with fields for Name, Rep freq. (fps) (202), Error rate (%) (0.00), and Comm. speed (9).

Type	ID	Subtype	Slot	Value	Notes
Module	1	Test_ADC			Version 107-0
Pin	36	Dig_in	1	0.0	
Pin	39	Adc_16	2	0.0	
Pin	34	Dig_in	3	0.0	
Pin	35	Dig_in	4	0.0	
Pin	32	Adc_16	5	0.0	
Pin	33	Adc_16	6	0.0	
Pin	25	Dig_in	7	0.0	
Pin	26	Dig_in	8	0.0	
Pin	27	Dig_in	9	0.0	
Pin	14	Dig_in	10	0.0	
Pin	12	Dig_in	11	0.0	
Pin	13	Dig_in	12	0.0	
Pin	15	Dig_in	13	0.0	
Pin	2	Dig_in	14	0.0	
Pin	0	Adc_16	15	0.0	
Pin	4	Adc_16	16	0.0	
Pin	16	Adc_16	17	0.0	
Pin	17	Adc_16	18	0.0	
Pin	5	Unused			
Pin	18	Unused			
Pin	19	Dig_in	21	0.0	
Pin	21	Dig_in	22	0.0	
Pin	3	Dig_in	23	0.0	
Pin	1	Dig_in	24	0.0	
Pin	22	Dig_in	25	0.0	
Pin	23	Dig_in	26	0.0	
Module	2	Test_Pwm_8ch			Version 109-0
Pin	36	Dig_in	50	0.0	
Pin	39	Dig_in	51	0.0	
Pin	34	Pwm	52	0.0	
Pin	35	Pwm	53	0.0	
Pin	32	Pwm	54	0.0	
Pin	33	Pwm	55	0.0	
Pin	25	Pwm	56	0.0	
Pin	26	Pwm	57	0.0	
Pin	27	Dig_in_pu	58	0.0	
Pin	14	Pwm	59	0.0	
Pin	12	Pwm	60	0.0	
Pin	13	Pwm	61	0.0	
Pin	15	Pwm	62	0.0	
Pin	2	Pwm	63	0.0	
Pin	0	Dig_in	64	0.0	
Pin	4	Dig_in	65	0.0	
Pin	16	Dig_in	66	0.0	
Pin	17	Dig_in	67	0.0	
Pin	5	Dig_out	68	0.0	
Pin	18	Dig_out	69	0.0	
Pin	19	Dig_out	70	0.0	
Pin	21	Dig_out	71	0.0	
Pin	3	Dig_out	72	0.0	
Pin	1	Dig_out	73	0.0	
Pin	22	Dig_out	74	0.0	
Pin	23	Dig_out	75	0.0	

The "Slots"

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 only the 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 highlighted in light green. If two of more input pins have the same Slot, then the HAL application warn with red lines and the text **SLOT CONFLICT**.

Molte applications and many Pin can read the same slot. But avoid configuring more than one Pin writing on the same slot, doing so does not break anything, but you get undefined results.

If you send multiple data streams to the same Slot, all the data are mixed and wins the last who write. To merge the data in an orderly way some rules are required.

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

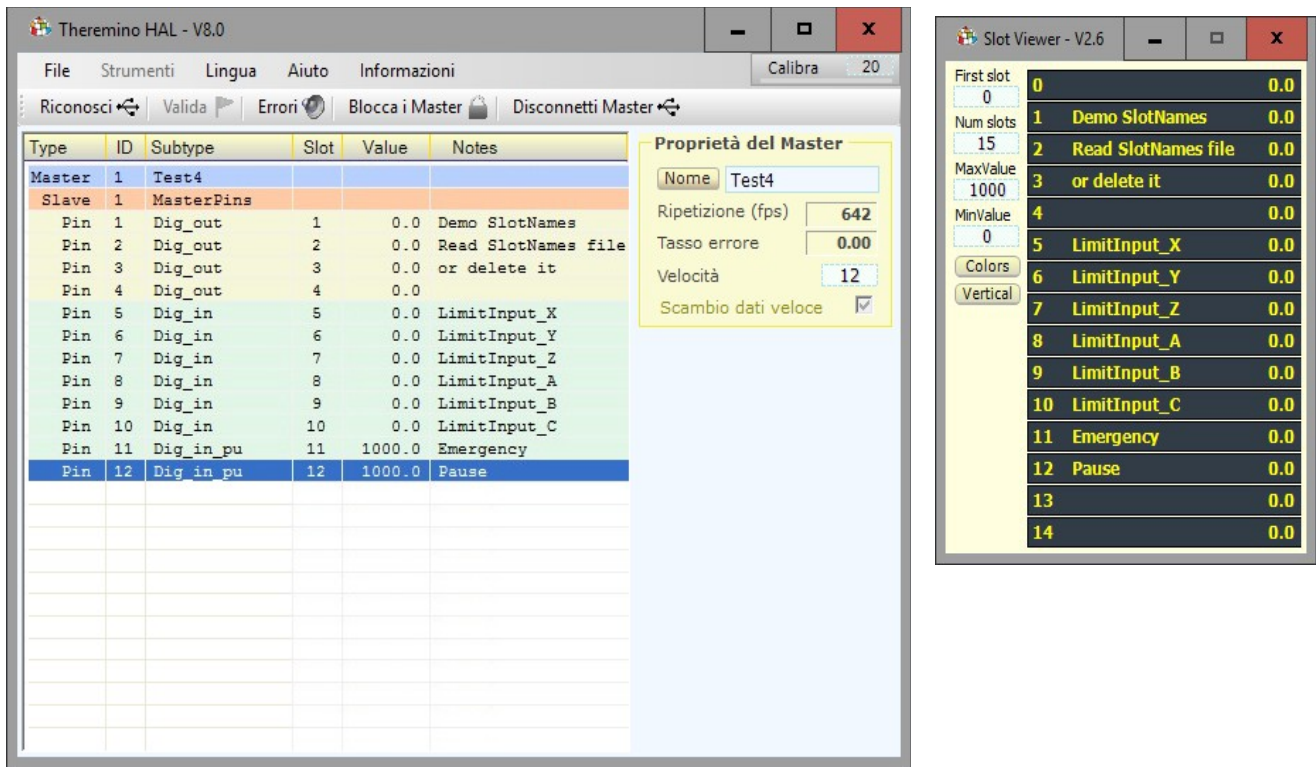
More information about the communication in these pages:

www.theremino.com/en/technical/communications

www.theremino.com/en/technical/pin-types

The Slot names

All the HAL applications, and even the 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.

Slots for text commands

From early 2024, Theremino System applications and user-created applications can send commands in text form and receive responses from the HAL, using two **TextSlot** to communicate.

In the next few pages we will see that one also exists **Slots** to send numeric commands, but with text commands you can do many more things, practically everything you could do with the mouse and keyboard on the HAL interface.

When to use Number Slots instead of TextSlots

There is only one case in which it is preferable to use the Numeric Command Slot.

When you want to know the number of Masters (i.e. hardware modules) that have been recognized and are active, with numeric Slots you just need to read the value of the Slot and this happens in a few microseconds, whereas if we did it with TextSlots it would be necessary to send a command to ask for the value and then wait for the response. And this would take tens of milliseconds.

For this reason and also to make it easier for those who are already using them in their applications, we have also kept the numeric slot commands active.

Numbers to use for TextSlots

The **TextSlot** they have numbers from 0 to 999 like number slots but they are two different things. Even if the same number is used **TextSlot** and **numeric Slots** they do not interfere with each other.

If not set differently, the **TextSlot zero** for the commands and the **TextSlot one** for the answers.

Use other Slots instead of TextSlots zero and one

Usually the command TextSlots are zero and one, but it may happen that you want to use multiple independent applications on the same PC. In these cases each application would reside in separate folders along with its HAL and access its Master modules using the "Lock Master" command. In these cases, a different command slot can be assigned to each HAL. For the commands you can use any TextSlot (from 0 to 999) but be careful not to use it for anything else.

To assign numbers to the command slots, manually modify the last lines of the "Theremino_HAL_INI.txt" file. So to use, for example, TextSlots 300 and 301, you would write: **CommandTextSlot=300** and **ResponseTextSlot= 301**.

Be careful not to delete the "=" signs. In case of errors, slots zero and one are used and the HAL rewrites the correct line in the INI file.

How to send commands

After sending a command, before sending another it is best to wait at least 50 or 100 ms to give applications time to perform writing and reading even in the worst cases.

To minimize this wait, you check the text in the CommandTextSlot and continue as soon as it is deleted from the HAL.

See the Automation example: **Demo Programs\SlotText Commands\Commands_to_HAL.txt**

Commands with TextSlots

In the next pages we will list all the usable commands but **only the HAL application recognizes them all.**

ArduHAL, IoTHAL, and NetHAL use only a portion of the commands, and many of the commands are only valid when the Pin types that use them are selected.

Commands ending in ON/OFF are enablements. Any value other than a zero or the word OFF is considered ON. Normally 1/0 or ON/OFF are used.

Both commands and their additional parameters can be written indifferently with uppercase or lowercase characters in any combination.

The command parameters, such as the names of the Masters, the types of Pins, the types of Filters and the types of Adc24 channels, are recognized only on the basis of their alphanumeric characters. For parameters, spaces and other signs do not matter. So for example "Dig_in_pu" could be written as "diginpu" or even "DIG-IN___PU" and would still be recognized and considered valid.

Top bar commands

- | | |
|--------------------------------------|---|
| ● Recognize | It has the same effect as the Recognize button. |
| ● Validate | Has the same effect as the Validate button. |
| ● ErrorBeep ON/OFF | Enable or disable the Error Beep button. |
| ● LockMasters ON/OFF | Enable or disable the Lock Masters button. |
| ● Disconnect | Disconnects the selected Master (module). |
| ● NoEdits ON/OFF | Enable or disable the NoEdits button. |
| ● Calibrated | Enable or disable the Calibrate button. |
| ● CalibrationValue N (1..100) | Numeric value for the "calibrate" box. |

Master Commands (or modules)

- | | |
|----------------------------------|---|
| ● SelectMaster N (note1) | Select the Master (if there are more than one). |
| ● LoadConfig XXX | Load the XXX configuration for the selected Master. |
| ● Speed N (1..12) | Communication speed for the selected Master. |
| ● FastDataExchange ON/OFF | Enable or disable FastExchange (only for Slave modules) |

(Note1)

The SelectMaster command is also recognized if you write SelectModule which may be more appropriate for Arduino and ESP32 modules.

Commands with TextSlots (part 2)

Select PINs

To select the Pin a text string is used although in many cases this string will simply contain numbers from 1 to 12.

But when using the Arduino and Esp32 modules, the Pins are not numbered in sequence, for example 22, 35, 28, 60 etc. and in some cases they are alphanumeric, for example A0, A1, A3, D0, D1, D3 etc. .

So in all cases instead of a number the name of the Pin is indicated as a text string.

Command to select PINs

● **SelectPin XXX** A text string is used for the Pin name

Numbering for the ADC24 and CAP-SENSOR pins

When using the Masters with the Theremino_HAL app. and connecting an Adc24 and/or one or more CapSensors, the pin numbering is different from what you might imagine.

The Master Pins range from 1 to 12.

Then the numbering continues indicating the Adc24 with 13, 14 and following.

(in this case they range from 13 to 28)

And also the CapSensors continue with the numbering.

(in this case are indicated with 29 and 30 but without Adc24 would be 13 and 14)

Type	ID	Subtype	Slot	Value	Notes
Master	1	Adc24AndCa...			
Slave	1	MasterPins			Firmware V5.0
Pin	1	Unused	← 1		
Pin	2	Unused			
Pin	3	Unused			
Pin	4	Unused			
Pin	5	Unused			
Pin	6	Unused			
Pin	7	Adc_24			
Pin	8	Adc_24_din			
Pin	9	Adc_24_dout	109	917.6	
Pin	10	Unused			
Pin	11	Unused			
Pin	12	Unused	← 12		
Adc24	1	Unused	← 13		
Adc24	2	Unused			
Adc24	3	Unused			
Adc24	4	Unused			
Adc24	5	Unused			
Adc24	6	Unused			
Adc24	7	Unused			
Adc24	8	Unused			
Adc24	9	Unused			
Adc24	10	Unused			
Adc24	11	Unused			
Adc24	12	Unused			
Adc24	13	Unused			
Adc24	14	Unused			
Adc24	15	Unused			
Adc24	16	Unused	← 28		
Slave	2	CapSensor			
Pin	1	Unused	← 29		
Slave	3	CapSensor			
Pin	1	Unused	← 30		

Commands with TextSlots (part 3)

Pin properties (Pin properties panel)

- **SetPinType XXX** Set the type for the currently selected Pin.
- **SetSlotNo(0..999)** Set the Slot for the selected Pin.
- **SetMaxValue N** Set MaxValue for the selected Pin.
- **SetMinValue N** Set MinValue for the selected Pin.
- **SetPinType XXX** Set the type for the selected Pin.
- **SetResponseSpeedButton ON/OFF** Enable or disable the ResponseSpeed button.
- **SetResponseSpeed N (1..100)** Sets the ResponseSpeed value.

Properties for PWM and SERVO Pins (PWM properties and SERVO properties panels)

- **MaxTime N** Set Max time for the selected Pin.
- **MinTime N** Set Min time for the selected Pin.
- **LogarithmicResponse ON/OFF** Enable or disable the Logarithmic response button.

Properties for STEPPER type Pins (Stepper properties panel)

- **MaxSpeed N** Set Max sp. (mm/min) for the selected Pin.
- **MaxAcc No** Set Max acc. (mm/sec/s) for the selected Pin.
- **StepsPerMm N** Set Min time for the selected Pin.
- **LinkedtoPrevious ON/OFF** Enable or disable the Linked to previous button.

Properties for PWM FAST type Pins (Pwm_Fast properties panel)

- **PwmFastFrequency N** Set the PWM frequency.
- **PwmFastDutyCycle N(0..1000)** Set the Duty Cycle.
- **FrequencyFromSlot ON/OFF** Enable or disable frequency control from the Slot.
- **DutyCycleFromSlot ON/OFF** Enable or disable duty cycle control from the Slot.

Properties for CAP type Pins(Touch properties panel)

- **MinVariation N (-1000..+500)** Set the minimum variation.
- **ProportionalArea N (-500..+1000)** Set the proportional area.

Properties for COUNTER and PERIOD type Pins(Freq properties panel)

- **ConvertToFrequency ON/OFF** Enable or disable frequency conversion.
- **MaxFrequency N** Set the maximum frequency.
- **MinFrequency N** Set the minimum frequency.

Commands with TextSlots (part 4)

Properties of CapSensor and Usound type Pins (CapSensor and Usound properties panels)

- **MaxDist N** (50..9999) Set the maximum distance in mm.
- **MinDist N** (0..5000) Set the minimum distance in mm.
- **Area N** (0..9999) Set the area in square mm (only for CapSensors).
- **RemoveErrors ON/OFF** Enable or disable error checking for the Usound type

Properties of Adc24 type Pins (Adc24 properties panel)

- **NumberOfPins N** (0..16) Set the PIN number (only for Masters, not ESP32).
- **SamplesSec N** (10..19200) Set the sampling rate.
- **Filter XXX** Set the filter type (MaxSpeed, Fast, Medium etc..)

Properties of Adc24_ch and Adc24_cn_b type Pins (Adc24_channel properties panel)

- **Type XXX** Set the type (Differential, Pseudo Diff, Single ended).
- **Gain N** Set the gain (0, 2, 4, 8, 16, 32, 64, 128)
- **Biased ON/OFF** Enable or disable 1.65 volt bias.

VALUE REQUESTS

- **GetMasterName (note 1)** Request for the name of the selected Master (module).
- **GetFPS** Request for the current communication frequency.
- **GetPinType** Type request for the selected Pin.

ANSWERS

- **OK** The command was executed successfully.
- **Error text** It also includes the command that generated the error.
- **Required parameter** The value that was requested with one of the **GET** commands.

(Note 1)

The GetMasterName command is recognized even if you write GetModuleName which might be more appropriate for Arduino and ESP32 modules.

Commands with the numerical Slots

The Theremino System applications, or other applications created by users, can communicate with the HAL, sending commands or receiving data, using a numerical Slot to communicate.

For example an applications may change the parameters of all the Pins, rewriting the configuration file, and then sending the command "Recognize". Or an application could verify how many Masters are really connected, sending the "Recognize" command, and then reading their number from the Command Slot. Or a musical application could calibrate the CapSensor modules or the CapKeys, sending the "Calibrate" command (actually unimplemented in ESP32 modules).

Using other Slots in place of the Slot zero.

Normally the Command Slot is zero, but may happen that you want to use multiple independent applications on the same PC. In these cases, each application would reside in separate folders together with its lotHAL and accesses its modules. In these cases, a different Command Slot can be assigned to each lotHAL Slot. For the commands you can use any Slot (from 0 to 999) but be careful not to assign it to any Pin.

To assign a non-zero number to the Command Slot, manually edit the last line of the file "Theremino_lotHAL_INI.txt". So to assign, for example, the Slot 300, you would write: **CommandSlot= 300**. Be careful not to delete the "=" sign. If something is wrong, the Slot zero is used and the HAL rewrites the corrected line in the INI file.

How to send commands

Currently two commands are defined:

- ◆ Recognize You send "NAN_Recognize," or the number "1"
- ◆ Calibrate (*) You send "NAN_Calibrate," or the number "2"

Applications that are not able to send special numbers NaN (Not a Number), they can use the numbers "1" and "2" instead of the "NAN_Recognize" and "NAN_Calibrate".

For safety, the commands "1" and "2" 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's virtually impossible that an ADC or other devices can send this sequence by mistake.

Response messages

Responses, and error messages, are communicated with numbers in the command slot.

- ◆ -1 The "Recognize" command is still executing.
- ◆ 0 No lotModule have been found. The list is completely white.
- ◆ 1 and up The number of Modules that has been recognized.
- ◆ NAN_MasterError One of the connected Mdules has stopped communicating.

The Command Slot - Examples

To send the "Recognize" command you write:

```
----- VBNET
Slots.WriteSlot (0, NAN_Recognize)

----- CSharp
Slots.WriteSlot (0, NAN_Recognize);

----- Theremino Script
WriteSlot (0, NAN_Recognize)
```

As explained on the previous page, some applications (Theremino Automation for example) are not capable to use the NAN special numbers. Not using NANs the previous examples would become:

```
----- VbNet
Slots.WriteSlot (0, 333)
System.Threading.Thread.Sleep(50)
Slots.WriteSlot (0, 666)
System.Threading.Thread.Sleep(50)
Slots.WriteSlot (0, 1)

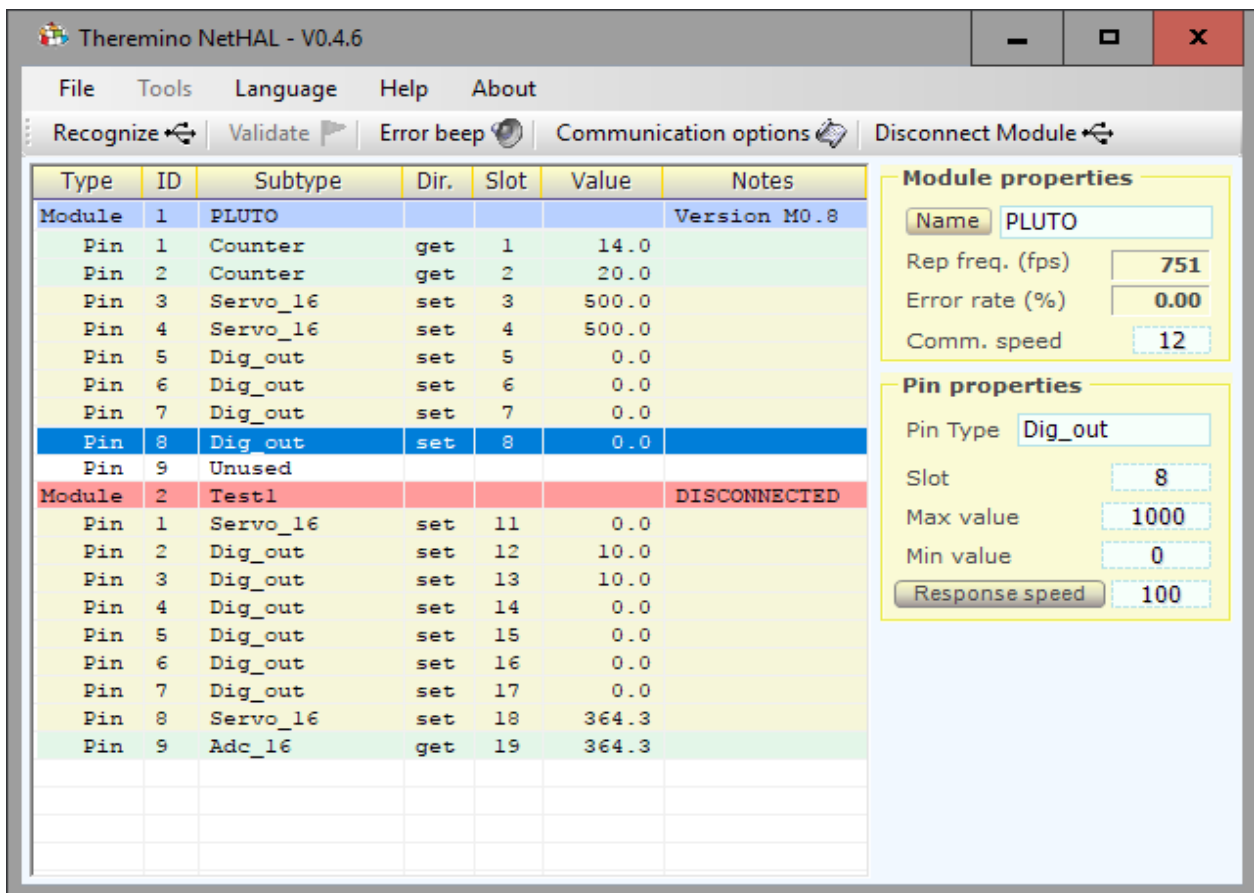
----- CSharp
Slots.WriteSlot(0, 333);
System.Threading.Thread.Sleep(50);
Slots.WriteSlot(0, 666);
System.Threading.Thread.Sleep(50);
Slots.WriteSlot(0, 1);

----- Theremino Automation
Slot 0 = 333
Wait Seconds 0.05
Slot 0 = 666
Wait Seconds 0.05
Slot 0 = 1

----- Theremino Script
WriteSlot (0, 333)
Threading.Thread.Sleep(50)
WriteSlot (0, 666)
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 first module (named PLUTO) It provides:

Two "Pins" configured as "Counter" (the light green color indicates an Input)

Two "Pins" configured as "Servo_16" (the light yellow color indicates an Output)

Four "Pins" configured as "Dig_out" (of which the pin 8 is selected)

A "Pin" configured as "Unused" (the white color indicates "not used")

The second form (with the name TEST1) provides:

A "Pin" configured as "Servo_16" (the light yellow color indicates an Output)

Six "Pins" configured as "Dig_out" (the light yellow color indicates an Output)

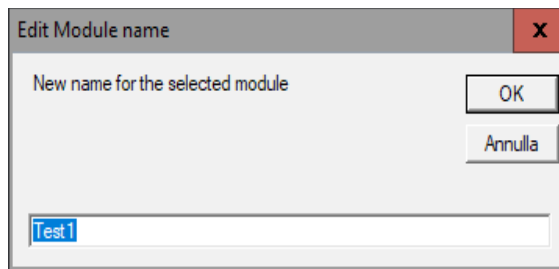
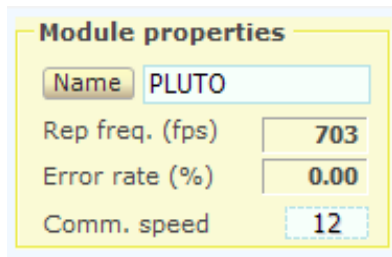
A "Pin" configured as "Servo_16" (the light yellow color indicates an Output)

A "Pin" configured as "Adc_16" (the light green color indicates that it is an Input)

The blue line "Pin 8 - DigOut" is the selected row and its properties are shown on the right

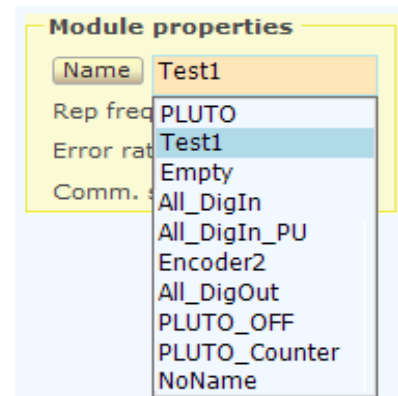
The line "Module 2 - Test1" has a red background because the "Test1" module is disconnected (off)

The properties of the module - The name



The name of the selected module can be modified 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 new newly attached module is called "NoName". We suggest you to rename the module, to distinguish it from all the others.

In the names of the modules the "case" of the letters (uppercase or lowercase) does not matter.

If in the data base there are two modules with the same name, then the first configuration is used for both the modules. 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).

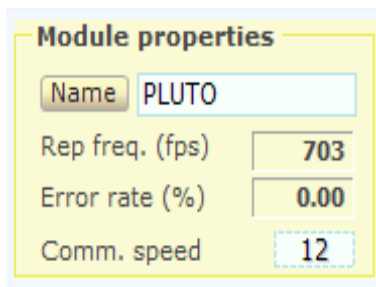
The modules are always listed in alphabetical order, so if the IP address changes, then the module order does not change.

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



Module properties	
Name	PLUTO
Rep freq. (fps)	703
Error rate (%)	0.00
Comm. speed	12

- Number of communications per second
- Percentage of transmission errors (normally zero)
- Communication Speed Adjustment

The number of messages per second "Fps" should normally be above 500 and often above 800, increasing the used Pins this frequency may drop 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 400 or 500.

The percentage of errors normally is zero. You should have errors only in case of another network on the same channel, or if the distance from the access point exceeds 20..50 meters.

If the signal is very weak or disturbed, the percentage of errors increases and the connection may be lost. **But as soon as the signal improves the connection is automatically restored.**

Tips to increase the "Fps" frequency

- Move closer to the access point
- Change the WiFi channel on the access point in order to avoid disturbances from neighbors.
- Use the application [ThereminoWiFi](#) to see which channels are less used.
- Use a better Access Point.
- Increase the "Comm Speed".
- Decrease the number of bytes used by configuring as "Unused" all the unnecessary Pins.

Adjust the "Fps" frequency

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

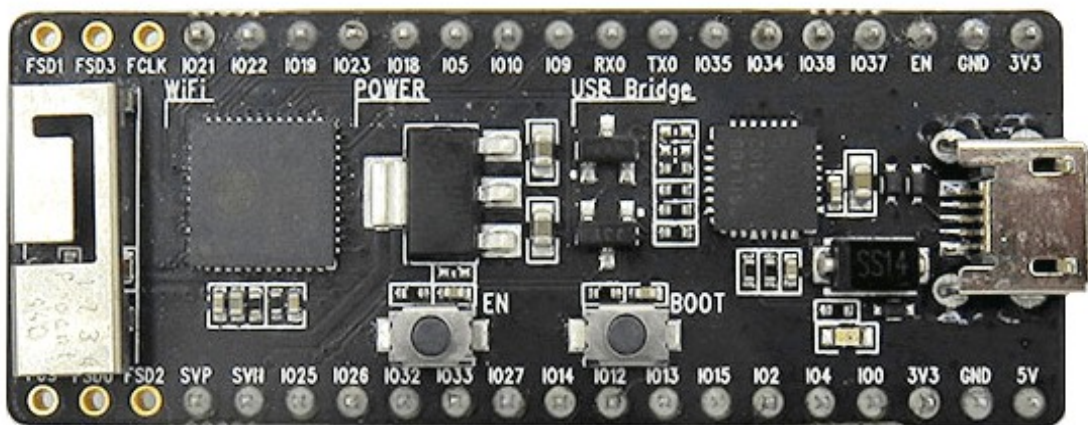
To increase the response speed would be good to maximize the exchange rate, and set "Comm Speed" to "12". But for many applications 100 exchanges per second it is more than enough, so normally you can adjust "Comm Speed" from 8 to 10 and charge less the CPU.

Number of connectable modules

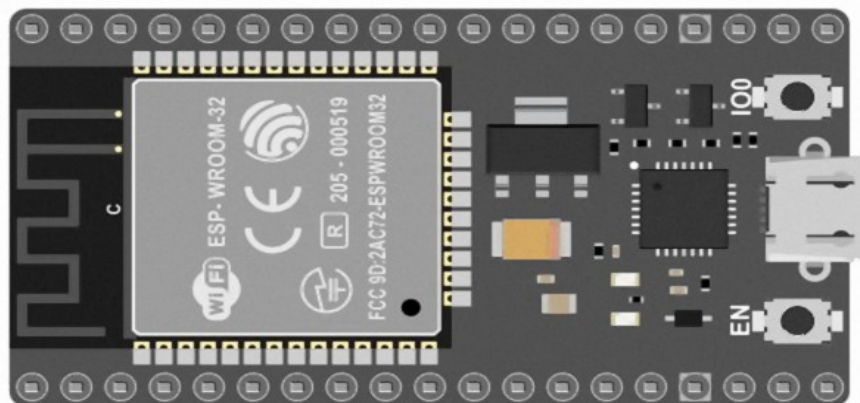
The modules all communicate on the same WiFi channel and necessarily share the band. So the communication speed with the Access Point decreases increasing the number of connected modules. Different behaviors can occur depending on the Access Point model.

In an experiment (with a LinkSys and DD-WRT firmware) the speed was 600 Fps with a single module, and it went down to 320 Fps with six modules connected. By increasing the modules to 8 and beyond, we have experienced further slowdowns and also significant communication delays.

The different types of ESP32 modules



This is the model that we prefer ESP32, it communicate faster than the others, has a larger number of Pin and the pin layout is better.

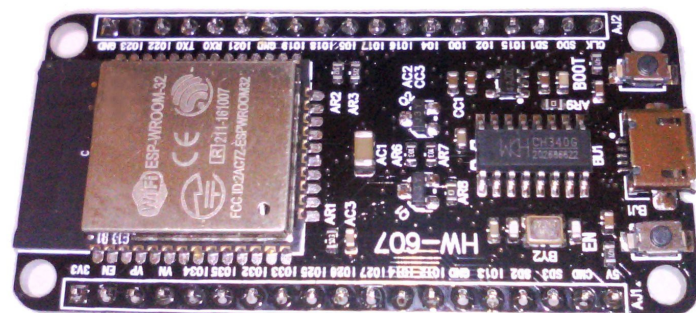


To use these modules you must enable the line `"#define IOTMODULE_PINOUT_TYPE 1"` in the `"IotModuleSetup.h"` file This ESP32 model is to be used as second choice.

To use these modules you must enable the line `"#define IOTMODULE_PINOUT_TYPE 0"` in the `"IotModuleSetup.h"` file

Most of ESP32 are similar to the previous two. But there are also different (which we did not try), and even apparently similar, but non-functional or that require special care.

For example, the module of this image, is programmed without errors, but then it not connects to the Access Point. Furthermore, its LED will never light up, either by programming or by pressing the reset button.



We have found that to make operating this model of modules, you must connect the Pin "0" to 3.3 volts through a 1k resistor (if you connect it directly no longer able to program). Moreover this model has no LEDs and has the WiFi antenna works very badly, it works only near to the Access Point.

Consumption and voltages of the ESP32

Supply Current	Conditions of measurement
10 mA	Disconnect
200 mA	In some moments during the connection
140 mA	WiFi On
150 mA	HAL connected
160 mA	All DigIn
160 mA	All DigIn_pu
160 mA	All DigOut
280 mA	Some PWM to 5 Mhz
320 mA	In some cases, with PWM and long wires

When the consumption exceeds 200 mA may occur communication errors and in some cases even the loss of communication. It happens more easily under particular conditions, with long wires and with PWM set at frequencies from 1 MHz up.

*Using high-frequency PWM it is good to use short wires. Or to add a resistor in series with the signal, positioned close to the ESP32 module. **See the next page.***

Setting Pins 1 and 3 as DigIn, DigOut or Pwm, interrupts the serial communication (the serial debug via USB), but does not increase the consumption.

Bringng DigIn inputs to half voltage, or disturbing the open DigIn inputs, with high-frequency signals, does not increase the consumption in a measurable way.

The inputs do not tolerate voltages higher then approximately 3.5 volts. If you connect them to the 5 volts to large electricity consumers and may break. The 5-volt signals can be connected to the inputs through a resistor of at least 1 k ohms.

An electrolytic capacitor 1000 uF or even from 2200 uF (better if low-ESR) between 5V and GND, greatly improves the waveforms, when you look at the oscilloscope. It also increases the stability, and you can connect averagely powerful servo to the 5 volts line, without causing power supply holes. An electrolytic capacitor between 3.3V and GND, has no visible effects on the waveforms, but it could increase the stability in some cases.

WARNING: The large electrolytic can be dangerous for the components of the module, download them before you connect them and connect them to shut down. Espressif says that if you add capacitors large capacity you might have to reset partial problems, during brief power failures.

If connecting motors or other devices, the USB connector should only be used for programming, and not to supply power to the connected devices. Otherwise, the WiFi communication could be interrupted, as soon as you move the Servo or other users who consume a lot of current.

Connecting the output signals

Pay attention to the following cases:

- ◆ When capacitive loads higher than a few tens of pF are piloted with the outputs of the Esp32, for example a shielded cable of one meter that already makes from 50 to 300 pF, or even only connecting wires a few tens of centimeters long.
- ◆ And even worse, when signals that change very frequently are sent to these outputs, such as PWM or STEPPER.

In these cases defects of all kinds can occur, errors on the signals and even loss of communication.

These defects occur because in switching the ESP32 does not limit the output current to reasonable values (for example 10 mA) but does everything it can to load the capacity with the highest current it can give. From the simulations it would be said that it reaches 3 amps, surely it will not be true but in any case it "shoots" an exaggerated current creating two types of problems:

- (1) The switching harmonics interfere with the WiFi signal.
- (2) The power supply fails during switching and errors occur, or even continuous module reset and disconnections of the HAL application.

Fortunately, there is a simple solution: add a resistor in series to each Exp32 output that must drive capacitive loads in excess of a few tens of pF.

A 330 ohm resistor can be fine in most cases. With this value the current is limited to about 10 mA and it is still possible to drive loads of many hundreds of pF (for example a shielded cable of a few meters), with square wave signals up to 300 KHz, and even a little well.

If you want to drive higher capacities, or increase the transmissible frequency, you could decrease the resistor up to 100 ohms. Better not go further, in order to avoid the problems mentioned at the beginning.

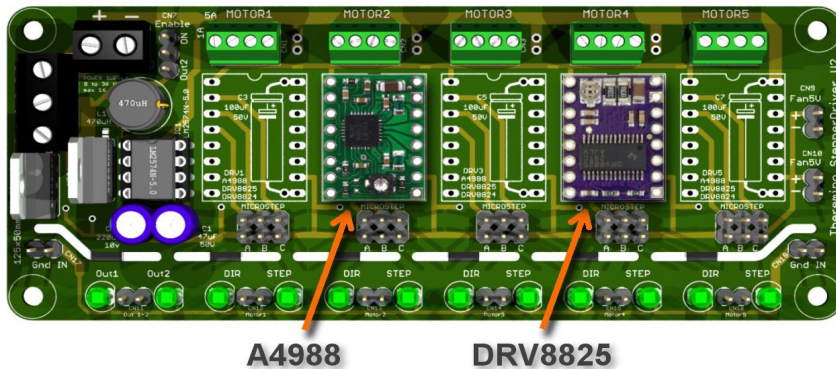
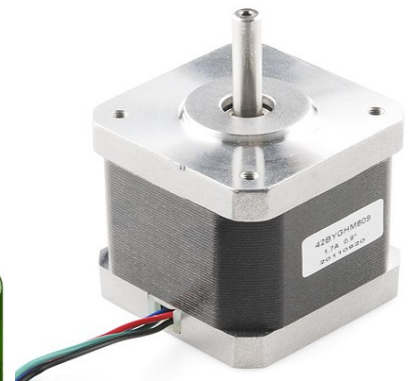
So with 100 ohm and 300 pF capacitance it could go up to 1 or 2 MHz. While with 100 ohm and 30 pF capacitance it could go up to 10 or 20 MHz.

These calculations are to ensure a square wave with fairly steep edges, i.e. switching times of less than a tenth of the total time of the period.

Connecting the output signals for Stepper Motors

The stepper motors are connected through specific driver modules.

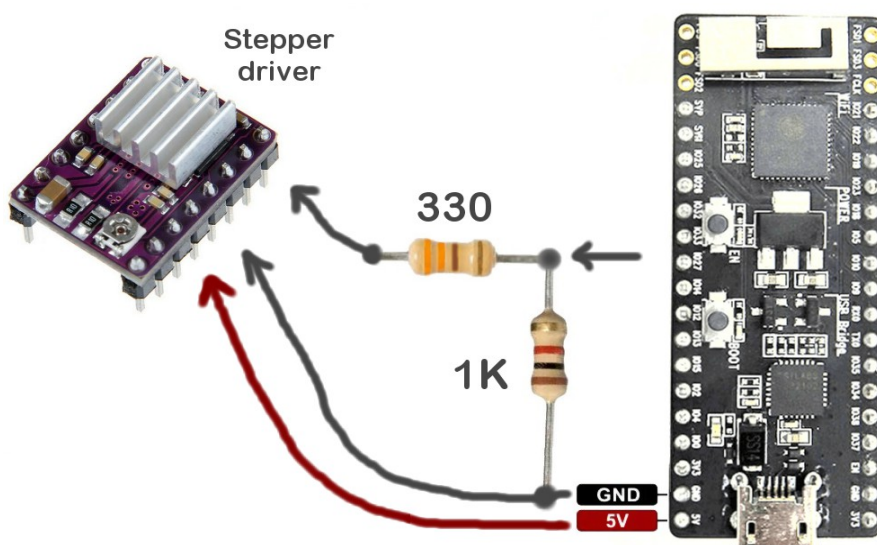
For information on drivers see [this page](#).



To ensure that the Stepper motors stay perfectly still when powering on, or restarting the module, a pull-down resistor should be added to the output signals that go to the motor drivers.

The value of the pull-down resistor must be between 1k and 4.7k and must be connected between the output Pin and the ground (GND).

The eventual 330 ohm resistor (see the previous page) must be connected after, along the signal wire that goes to the motor driver, but not too far from the module (a few centimeters maximum).



The two resistors of this image must be repeated for each STEP and DIR wire that goes from the IOT module to the Stepper Motor drivers.

The Pin types

The pin can be configured as:

- ◆ Not used
- ◆ Digital Output
- ◆ PWM output (0.02 Hz to 40 MHz)
- ◆ DAC Output (adjustable output voltage)
- ◆ Output for servo-controls
- ◆ Output for Stepper Motors
- ◆ Digital input
- ◆ Counter, frequency and period
- ◆ Encoders input for two/four phases
- ◆ ADC Input for potentiometers and transducers
- ◆ Input for capacitive buttons
- ◆ Adc24

Pin properties	
Pin Type	Unused
	Unused
	Dig_out
	Pwm
	Servo
	Dac_8
	Dig_in
	Dig_in_pu
	Cap_16
	Counter
	Counter_pu
	Period
	Period_pu
	Encoder_a
	Encoder_a_pu

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 (resistor pullup typical = 45k, which produces a current of 70 uA when the input is low).

All Pin Output can provide approximately +/- 20 mA current. If you use simultaneously many output Pins, the current available for each descends progressively up to +/- 10 mA.

All the Pins can be configured as "Unused" this allows to decrease the number of bytes that transit and maximize the number of exchanges per second.

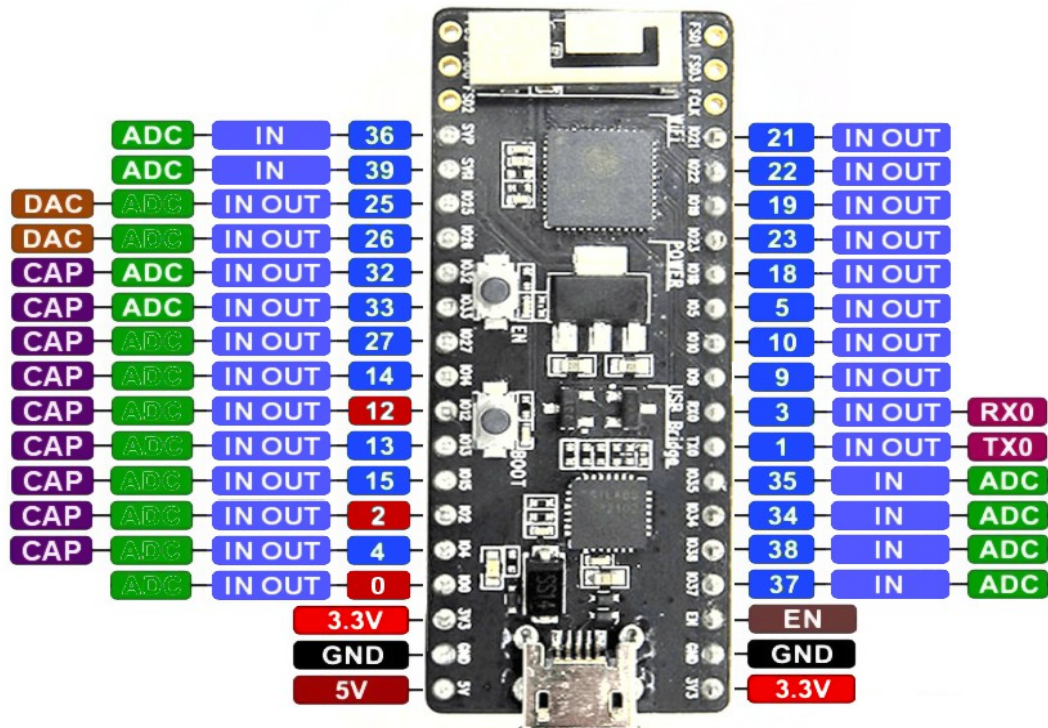
Not every pin can be configured in all possible ways. The images of the next pages show the available features for each Pin, for the most common lotModule models.

The Pin types for the ESP32 cards - Pico V4

This is the model that we prefer ESP32, it communicate faster than the others, has a larger number of Pin and the pin layout is better.

To use these modules must enable the line

"#define IOTMODULE_PINOUT_TYPE 1" in the "IotModuleSetup.h" file



IN = Inputs only, and no pullups

ADC = Adc when WiFi is not active

IN OUT = DigIn / Counter / Period / Encoder
DigOut / Pwm / Servo

RX0 **TX0** = Used by serial debugger

0 = Pin "0" must be high at startup and low while programming

2 = Pin "2" must be low while programming

12 = Pin "12" must be low at startup and while programming

Leave Pins 0, 2 and 12 open, or use them as output only, to control high impedance devices
Pins 2 and 12 could be input-pull-up, connected to GND with a button or an open collector

The Pins with the **IN** label they can be programmed as: DigIn, Counter, Period and Encoder.

The Pins labeled **IN OUT** they are also programmable as: DigOut, Pwm and Servo.

The Pins labeled **DAC** send in output a voltage from 0 to 3.3 volts.

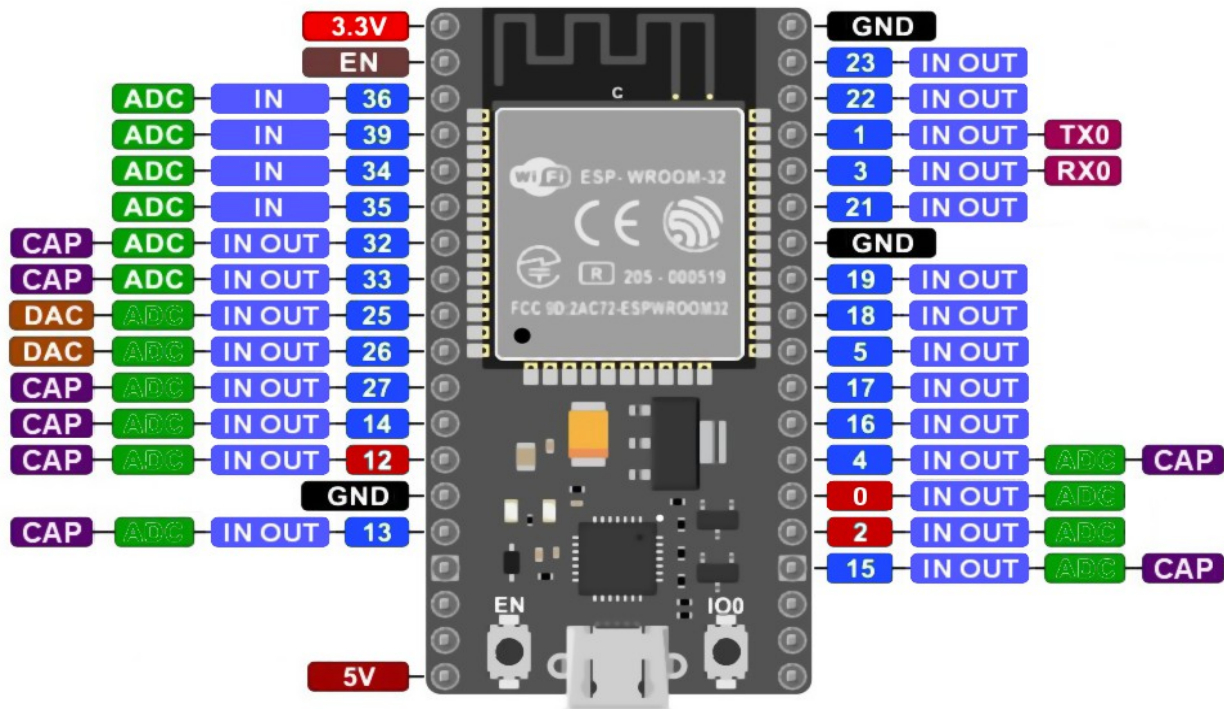
The Pins with the word **ADC** canceled they could be used as ADC, but only communicating via USB and disabling the WiFi communication (this can not be done in the current version).

The Pins "0", "1", "2", "3" e "12" can be used only with special attentions, otherwise the module will work badly. See the next page: [Connecting the special Pins](#)

The Pin types for the ESP32 cards - WROOM

This ESP32 model is to be used as second choice.

To use these modules must enable the line
`"#define IOTMODULE_PINOUT_TYPE 0"` in `lotModuleSetup.h` file



IN = Inputs only, and no pullups

ADC = Adc when WiFi is not active

IN OUT = DigIn / Counter / Period / Encoder
 DigOut / Pwm / Servo

RX0 TX0 = Used by serial debugger

0 = Pin "0" must be high at startup and low while programming

2 = Pin "2" must be low while programming

12 = Pin "12" must be low at startup and while programming

Leave Pins 0, 2 and 12 open, or use them as output only, to control high impedance devices

Pins 2 and 12 could be input-pull-up, connected to GND with a button or an open collector

The Pins with the **IN** label they can be programmed as: DigIn, Counter, Period and Encoder.

The Pins labeled **IN OUT** they are also programmable as: DigOut, Pwm and Servo.

The Pins labeled **DAC** send in output a voltage from 0 to 3.3 volts.

The Pins with the word **ADC** canceled they could be used as ADC, but only communicating via USB and disabling the WiFi communication (this can not be done in the current version).

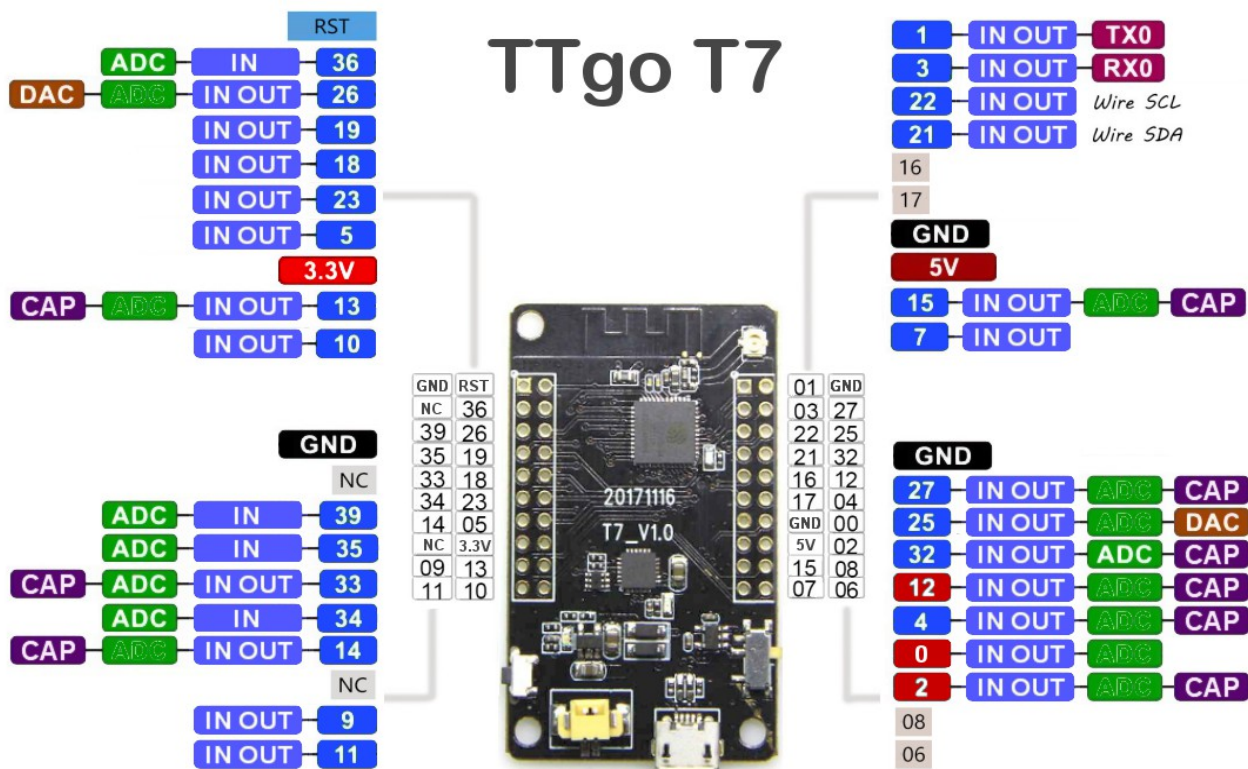
The Pins "0", "1", "2", "3" e "12" can be used only with special attentions, otherwise the module will work badly. See the next page: [Connecting the special Pins](#)

The Pin types for the ESP32 cards - WROOM

This model also has the connector for an external antenna and a UPS function (with 3.7 volt lithium battery) which is recharged by the 5 volt coming from the USB connector.

To use these modules must enable the line

"#define IOTMODULE_PINOUT_TYPE 2" in lotModuleSetup.h file



Leave Pins 0, 2 and 12 open, or use them as output only, to control high impedance devices
Pins 2 and 12 could be input-pull-up, connected to GND with a button or an open collector

The Pins with the **IN** label they can be programmed as: DigIn, Counter, Period and Encoder.

The Pins labeled **IN OUT** they are also programmable as: DigOut, Pwm and Servo.

The Pins labeled **DAC** send in output a voltage from 0 to 3.3 volts.

The Pins with the word **ADC** canceled they could be used as ADC, but only communicating via USB and disabling the WiFi communication (this can not be done in the current version).

The Pins "0", "1", "2", "3" e "12" can be used only with special attentions, otherwise the module will work badly. See the next page: [Connecting the special Pins](#)

Connecting the special Pins

Pins 1 and 3

These pins are used to Debug the firmware, by using a serial port connected through the USB cable.

Setting them as DigIn, DigOut or PWM, or by connecting them to low impedance devices, the serial communication stops working.

If not using the serial connection, these two pins can be used both in input and output, configuring them as: DigIn, Counter, Period, Encoder, DigOut, Pwm and Servo.

You can also use them as Generic IN/OUT, 8/16/24 and Float. Using them as Generic the serial connection continues to work.

Pin 0

This Pin must be kept **high** during the **startup**

This Pin must be kept **low** during the **programming**

Keep this Pin disconnected or use it in output to drive high impedance devices.

Pin 2

This Pin must be kept **low** during the **programming**

Keep this Pin disconnected or use it in output to drive high impedance devices.

Since it does not cause problems when you force it down, you could also use it for a normally open push-button connected to GND, or connected to the collector of an open-collector transistor.

Pin 12

This Pin must be kept **low** during the **startup**

This Pin must be kept **low** during the **programming**

Keep this Pin disconnected or use it in output to drive high impedance devices.

Since it does not cause problems when you force it down, you could also use it for a normally open push-button connected to GND, or connected to the collector of an open-collector transistor.

Adc24 connections

To connect the Adc24 we used pins 0, 2, 4, which are special and difficult to use. In this way we do not occupy any of the pins that are normally used.

Using the special pins as "Generic"

All the special pins could be used also as "Generic", to communicate user data with the HAL, and in this case they do not produce any collateral effect.

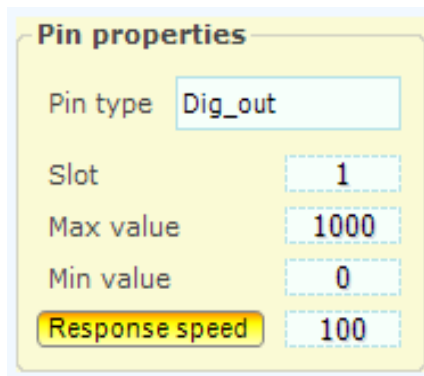
Stato dei Pin dopo il reset del modulo

When the module is restarted, or because of a power failure, or because someone has pressed the reset button, the pins are in an initial state that is not yet under the control of the IoT HAL application. This table indicates the initial status of each Pin.

GPIO	Input mode after reset
0	InPullUp
1	InPullUp
2	InFloat
3	InPullUp
4	InPullDown
5	InPullUp
6	InPullUp
7	InPullUp
8	InPullUp
9	InPullUp
10	InPullUp
11	InPullUp
12	InPullDown
13	InFloat
14	InFloat
15	InPullUp
16	InFloat
17	InFloat
18	InFloat
19	InFloat
21	InFloat
22	InFloat
23	InFloat
25	Float
26	Float
27	InFloat
32	Float
33	Float
34	Float
35	Float
36	Float
37	Float
38	Float
39	Float

To avoid turning on dangerous or annoying devices, it is recommended to add a resistor between their Pin and GND. A resistor of about 10k ohms is suitable for all the "float" Pins, while for Pins with "pullup", or that are connected to resistors on the printed circuit, a lower value should be used. 1k resistors should be fine, but it is better to check each case.

The common parameters of all the Pins



Pin properties	
Pin type	Dig_out
Slot	1
Max value	1000
Min value	0
Response speed	100

"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 the Pin and all the Theremino system applications.

Warning: many apps and many pin can be read from the same slot, but you must avoid configuring more than one Pin writing on the same slot, doing it 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.

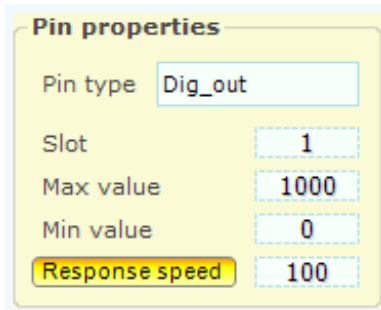
"Response speed" adjusts the filter IIR (Infinite Impulse Response) for the best compromise between noise and response speed. With a value 100, the filter is disabled and the maximum speed of response is obtained. The value 1 produces the maximum filtering, (elimination of any jitter) but a very slow 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 Pin types in "Output" -> Dig_out / Dac_8

◆ Dig_out



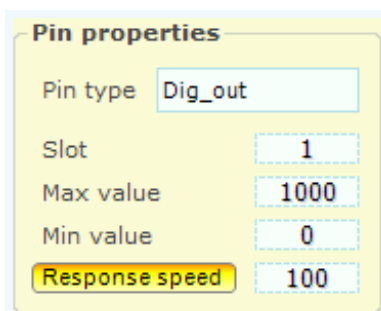
Pin properties	
Pin type	Dig_out
Slot	1
Max value	1000
Min value	0
Response speed	100

This type of pin provides a digital output.

The incoming value from one slot, limited between "Min Value" and "Maximum value" and filtered by "Response speed" is compared with the intermediate value between "Min value" and "Max value", if it exceeds the Pin turns on, otherwise off.

The Pin can only assume voltages 0 volts (off) and 3.3 Volt (on) and the output current is limited to approximately +/- 10 mA

◆ Dac_8



Pin properties	
Pin type	Dig_out
Slot	1
Max value	1000
Min value	0
Response speed	100

This type of Pin provides an output analog voltage adjustable from 0 to 3.3 volts.

The incoming value from one slot, is scaled with the values "Min Value" and "Max value" and transformed into a variable output voltage.

The output voltage on the pin is adjustable from 0 Volt and 3.3 Volt and the output current is limited to approximately +/- 20 mA

The resolution of the DAC has only 8 bits, then you can only get 256 voltage levels (with steps of approximately 13 mV between one level and the next).

To obtain a higher resolution (up to 20 bits), you can use the PWM outputs and a simple adapter consists of a resistor and a capacitor, as explained in the following pages.

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

◆ Servo



The screenshot shows a configuration window for a Servo pin. It is divided into two sections: 'Pin properties' and 'Servo properties'. In the 'Pin properties' section, 'Pin type' is set to 'Servo_16', 'Slot' is 1, 'Max value' is 1000, 'Min value' is 0, and 'Response speed' is 100. In the 'Servo properties' section, 'Max time (uS)' is 2500 and 'Min time (uS)' is 500.

Pin properties	
Pin type	Servo_16
Slot	1
Max value	1000
Min value	0
Response speed	100

Servo properties	
Max time (uS)	2500
Min time (uS)	500

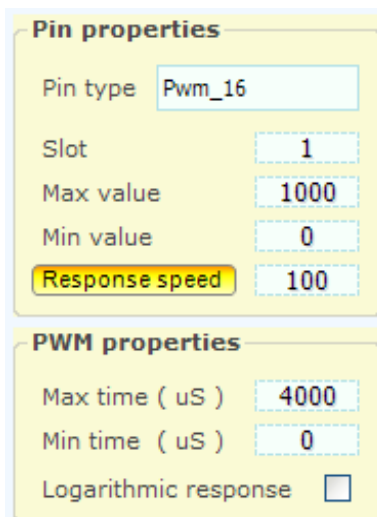
This type of Pin can control directly the servo controls.

The incoming value from one slot, limited between "Min Value" and "Maximum value" and filtered by "Response speed" is transformed into a pulse width between "Min time (uS)" and "Max time (uS)"

The repetition time is adequate to normal servos for modeling that, between min and max, revolve about 180 degrees.

The Pin provides voltages of 0 and 3.3 volts, adequate in all normal servo powered from 3 to 6 volts and a current sufficient to drive dozens of paralleled servo.

◆ PWM



The screenshot shows a configuration window for a PWM pin. It is divided into two sections: 'Pin properties' and 'PWM properties'. In the 'Pin properties' section, 'Pin type' is set to 'Pwm_16', 'Slot' is 1, 'Max value' is 1000, 'Min value' is 0, and 'Response speed' is 100. In the 'PWM properties' section, 'Max time (uS)' is 4000, 'Min time (uS)' is 0, and 'Logarithmic response' is unchecked.

Pin properties	
Pin type	Pwm_16
Slot	1
Max value	1000
Min value	0
Response speed	100

PWM properties	
Max time (uS)	4000
Min time (uS)	0
Logarithmic response	<input type="checkbox"/>

This type of Pin provides a PWM output (pulse width modulation)

The value coming from one slot, limited between "Min Value" and "Max value" and filtered by "Response speed", is transformed into a pulse width between "Min time (uS)" and "Max time (uS)"

The Pin provides pulses between the voltages 0 V (off) and 3.3 Volt (on) and the output current is limited to approximately +/- 20 mA

The frequencies settable range 0.02 Hz to 40 MHz, but the accuracy drops as they frequency increases, as explained in the following pages.

The Servo and PWM Pin types have limitations, read the following two pages.

The Pin types in "Output" -> Servo and PWM

The processor used in ESP32 IoTModule has limited resources, so you can program the Stepper, Servo and PWM pins with the following rules:

- ◆ There are sixteen timers in all and these timers are used for Steppers, Servos and PWMs.
- ◆ Each Stepper uses two timers, so there are a maximum of eight Steppers available.
- ◆ Steppers are allocated first, so two Servos or PWMs are lost for each Stepper set.
- ◆ The remaining channels can be set all as Servo, or all as Pwm, or in any combination of the two types. For example, you could set up to eight Servos and eight Pwm, or twelve Servos and four Pwm, and so on.
- ◆ If the Servos are not even in number, a channel is lost. So if for example you set seven Servos, then you will not be able to set nine PWM, but only eight. And if you set fifteen Servos, the sixteenth cannot be set as Pwm.
- ◆ When the timers are no longer enough to keep the PWM frequencies separate, the last PWMs are connected in pairs and changing the frequency of one changes the frequency of both PWMs of the pair.

This image gives an idea of how PWM channels are coupled and share the same frequency (in red).

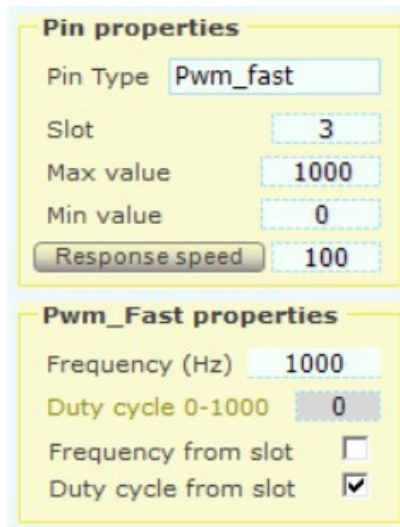
If Steppers were also used, each Stepper would occupy two lines. So the number of Pwm (and Servo) available would decrease by two and the Pwm would couple two lines earlier.

		ledc channel															
		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
numero pwm	0																
	1	Y															
	2	Y		Y													
	3	Y		Y		Y											
	4	Y		Y		Y		Y									
	5	Y		Y		Y		Y		Y							
	6	Y		Y		Y		Y		Y		Y					
	7	Y		Y		Y		Y		Y		Y		Y			
	8	Y		Y		Y		Y		Y		Y		Y		Y	
	9	Y		Y		Y		Y		Y		Y		Y		Y	
	10	Y		Y		Y		Y		Y		Y		Y		Y	
	11	Y		Y		Y		Y		Y		Y		Y		Y	
	12	Y		Y		Y		Y		Y		Y		Y		Y	
	13	Y		Y		Y		Y		Y		Y		Y		Y	
	14	Y		Y		Y		Y		Y		Y		Y		Y	
	15	Y		Y		Y		Y		Y		Y		Y		Y	
	16	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y

Some Pin can be used as input only and can not be set as Servo or PWM.

See the Pin maps in [this Page](#) and in the [next page](#).

The "Output" Pin types -> PWM



The image shows two configuration windows. The top window, titled "Pin properties", has a "Pin Type" dropdown set to "Pwm_fast". Below it are input fields for "Slot" (3), "Max value" (1000), "Min value" (0), and "Response speed" (100). The bottom window, titled "Pwm_Fast properties", has a "Frequency (Hz)" field set to 1000, a "Duty cycle 0-1000" field set to 0, and two checkboxes: "Frequency from slot" (unchecked) and "Duty cycle from slot" (checked).

The minimum frequency that can be generated is 0.02 Hz and the maximum is about 40 MHz. The Duty Cycle goes from zero (always low output signal) up to 100% (always high output signal).

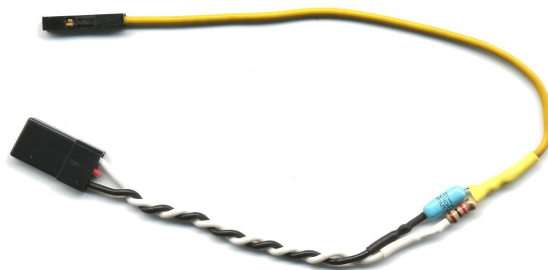
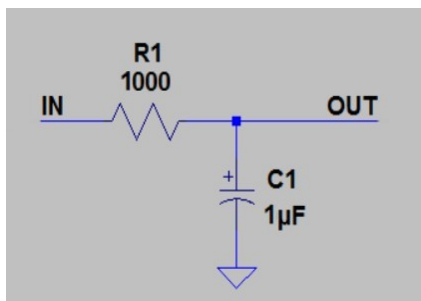
The value coming from the Slot sets the signal time ratio between high and low.

The slot value, usually between 0 and 1000, is first filtered then scaled with "Min value" and "Max value." Normally you set Min = 0 / Max = 1000, and adjusts the PWM, providing values from 0 to 1000.

The granularity of the adjustment depends on the frequency.

- ◆ At 1000 Hz the accuracy of the Duty Cycle is 16 bits (error: 0.0015%) and that of the frequency is 14 bits (errors: 0.006%)
- ◆ A 16 KHz precision of the Duty Cycle is 12 bits (errors: 0.024%) and that of the frequency is 10 bits (error: 0.1%)
- ◆ At 1 MHz, the accuracy of the Duty Cycle drops to only 6 bits (error: 1.5%) and that of the frequency is only 4-bit (errors: 6%)

With the PWM type pins and a simple adapter you can obtain an analog output voltage, adjustable with accuracy from 0 to 3.3 volts. Set the frequency to 15 kHz.



For more information on the adapters, read this page:

<https://www.theremino.com/en/hardware/adapters#pwm>

Frequency and resolution of the PWM signals

Frequency	Resolut.	High-Speed Channels	Low-Speed Channels	Min. step
40 Mhz	1 bit	Fixed 50%	Fixed 50%	
20 Mhz	2bit	0% 50% 100%	50% 100%	
10 Mhz	3bit	0% 25% 50% 75% 100%	25% 50% 75% 100%	
5 Mhz	4bit			
2500 kHz	5bit			
1250 kHz	6bit	64 values	64 values	51 mV
625 kHz	7bit			
312 kHz	8bit	256 values	256 values	13 mV
156 kHz	9bit			
78 kHz	10bit	1024 values	1024 values	3.2 mV
39 kHz	11bit			
19 kHz	12bit	4096 values	4096 values	800 uV
9 kHz	13bit			
4 kHz	14bit	16384 values	16384 values	200 uV
2 kHz	15bit			
1 kHz	16bit	65536 values	65536 values	50 uV
610 Hz	17bit			
305 Hz	18bit			
152 Hz	19bit			
76 Hz	20bit	over one million values	over one million values	3 uV
38 Hz	20bit			
19 Hz	20bit			
9 Hz	20bit			
4 Hz	20bit			
2 Hz	20bit			
1 Hz	20bit			

When using low frequencies, the PWM signals have a resolution **(Note 1)** very high, even 16 to 20 bits when using frequencies below the KHz.

At higher frequency the resolution **(Note 1)** progressively decreases, until reaching zero for frequencies above 20 MHz

(Note 1) The resolution represents the precision with which it is possible to adjust, both the time and the frequency of the PWM signal.

Peculiarities of the PWM signals

High frequency problems

By adjusting the PWM output channels with high frequencies (from 100 KHz and up), they can take place communication errors. These errors are due, both to voltage drops caused by the high consumption of the PWM signals, both to the harmonics emitted from the wires connected to the PWM outputs. The harmonics of the signal disturb the sensitive input circuits connected to the antenna.

Increasing the frequency of the PWM and long connecting wires to the PWM output, the problem is aggravated. And by using special PWM frequencies, the module may also disconnect.

In the case of long wires it is not even possible to use shielded cable because of its high capacity. In some cases, having to necessarily use of high PWM frequencies, one might think of adding a resistor in series with the PWM signal (close to ESP32 module) and then use a shielded cable. The resistor will limit the output current, and in collaboration with the cable capacitance, form a low pass that will eliminate the radio frequency disturbances. The resistor should be chosen to highest possible value, consistent with the controlled circuit needs and the bandwidth required.

These problems are eliminated completely by using low frequencies for the PWM signals, and maintaining the wires short and away from the antenna.

See the page: [Connect the output signals](#)

With high frequency PWM, LEDs do not light up anymore.

When you raise the PWM frequency, just little electrical capacity (and the LEDs have much) to overload the output circuits. Over the maximum current the PWM outputs produce a DC voltage, which is the average of the PWM signal. Under these conditions, the LEDs begin to turn on only when it exceeds 2.5 or 3 volts, i.e., when the PWM signal is at the maximum.

Instead, at low frequencies, the LEDs light up with flashes, with a duration equal to the top part of the PWM signal. Under these conditions, the average light output is proportional to the PWM signal timing, and the LED will light up even with the PWM trimmed to 20% or less.

The "Output" Pin types --> Stepper

Each stepper motor needs two physical Pins, one for the STEPS and one for the DIRECTION. The micro-controller would place the Pins to pleasure, but we decided to limit confusion, by specifying preset positions, for Stepper and Stepper_Dir Pins (positions are: 1-2, 3-4, 5-6, 7-8, 9-10).

The Pin type Stepper reading from a value, which is simply the destination in mm. Simple applications, can specify a destination far away, and let it do all the firmware. Most demanding applications, can calculate their own path and send frequent intermediate destinations. With this technique, an application can check the working speed (feed), and determine precisely the path, even in multiple dimensions. To get smooth motion just 20 destinations per second (up to 50 for the most demanding applications).

To reverse the direction of movement of an axis, they swap the values "1000" and "0", of boxes "1000 means mm" and "0 means mm".

Specific parameters of the Stepper Pins

Stepper properties	
Max sp. (mm/min)	900
Max acc. (mm/s/s)	100
Steps per mm	200

Max Speed - This is the fastest speed, in millimeters per minute. The firmware continuously checks the destinations sent by software. If the software is asking too much for the engine, the firmware restricts his speed, to avoid losing steps. Raise this value until you see that the motor will lose steps (It makes a high pitched noise and stops) and then decrease it by a 20..50%, to return to a safe area. Repeat the tests under load, or by braking the motor manually, so make sure you have some room.

Max Acc - This is the maximum acceleration (and deceleration), in millimeters per minute. The firmware continuously checks the destinations sent by software. If the software is asking too much for the engine, the firmware restricts its acceleration, to avoid losing steps. Raise this value until you see that the motor will lose steps during changes of direction (It makes a high pitched noise and stops) and then decrease it by a 20..50%, to return to a safe area. Repeat the tests under load, or by braking the motor manually, so make sure you have some room.

Steps for mm - Here you have to set the step, the engine is in a spin, multiplied by the microstep, set in the controller, and divided into millimeters, produced by a rotation of motor. If each spin, produces a millimeter of movement, and the engine is a 200 steps per revolution, and don't use the microstep, then you set the value: $200 \text{ (steps per revolution)} \times 1 \text{ (microstep)} / 1 \text{ (mm per revolution)} = 200$. If using sixteen microstep then you set the value: $200 \text{ (steps per revolution)} \times 16 \text{ (microstep)} / 1 \text{ (mm per revolution)} = 3200$.

The Stepper_Dir Pin type have no parameters to adjust. I'm just a placeholder for the physical output Pin, establishing the direction of the motor. It is not necessary to use the value, that these Pins are writing into the Slot, but some applications may find it useful. The value that is written into the Slot, is the distance to the destination, in millimeters (and up to fractions of a thousandth of a millimeter). This information can be used for diagnostic purposes, or for algorithms that must meet a specified tolerance. With this information the software can work with closed loop and always at maximum speed. Continually checking distance of each engine by the target, the software can slow down exactly when you need it, without doing complex calculations of speed, trajectories and accelerations.

The "Output" Pin types --> Stepper and Stepper_Dir

The Stepper values

The value read from the Slot, is related (with "1000 means mm" and "0 means mm") and transformed into a value between zero and one. If you set "1000 means mm" = 1000 and "0 means mm" = 0, then do not run conversions of scale and the value that comes out of the Slot is considered "mm".

From here on, the value is always in millimeters. "Zero" indicates zero millimeters and "one" indicates 1000 mm. This value is not limited to between zero and one, but between two billion positive step, and two billion negative step. If you are using "Steps for mm = 200" the limits are: +10 Km and -10 Km.

The value is then filtered with an IIR filter (linear or growth), with adjustable "Response Speed". The output value of the filter is called "Filtered".

The final value that is sent to the hardware is a STEP number (pre-multiplied by the value "Steps for mm") and represents the "destination".

The special value NAN_Reset, has the special meaning of resetting the axis. When you write a Reset, a Pin Stepper Slot, the motor stops immediately. Subsequently, the first value that will be written into the Slot, will be the value "zero reference". The NAN_Reset is available in Theremino Automation as "Reset", or in the new class "ThereminoSlots", available with the source code of Theremino Automation.

The Stepper_Dir values

Each Stepper_Dir Pin is always associated with a Stepper Pin.

It is a particular Pin, either exit or entry. It provides the motor the direction of the electrical signal (so an output) but at the same time acts as an input and provides information to the software.

The raw value that is read by the hardware, is the number of steps (positive or negative), missing to reach the "destination" specified. The HAL application calculates mm (and fractions), by dividing the raw value, for the value "Steps for mm". Finally this value in millimeters, is written into the Slot, and can be read by other applications, normally a CNC application.

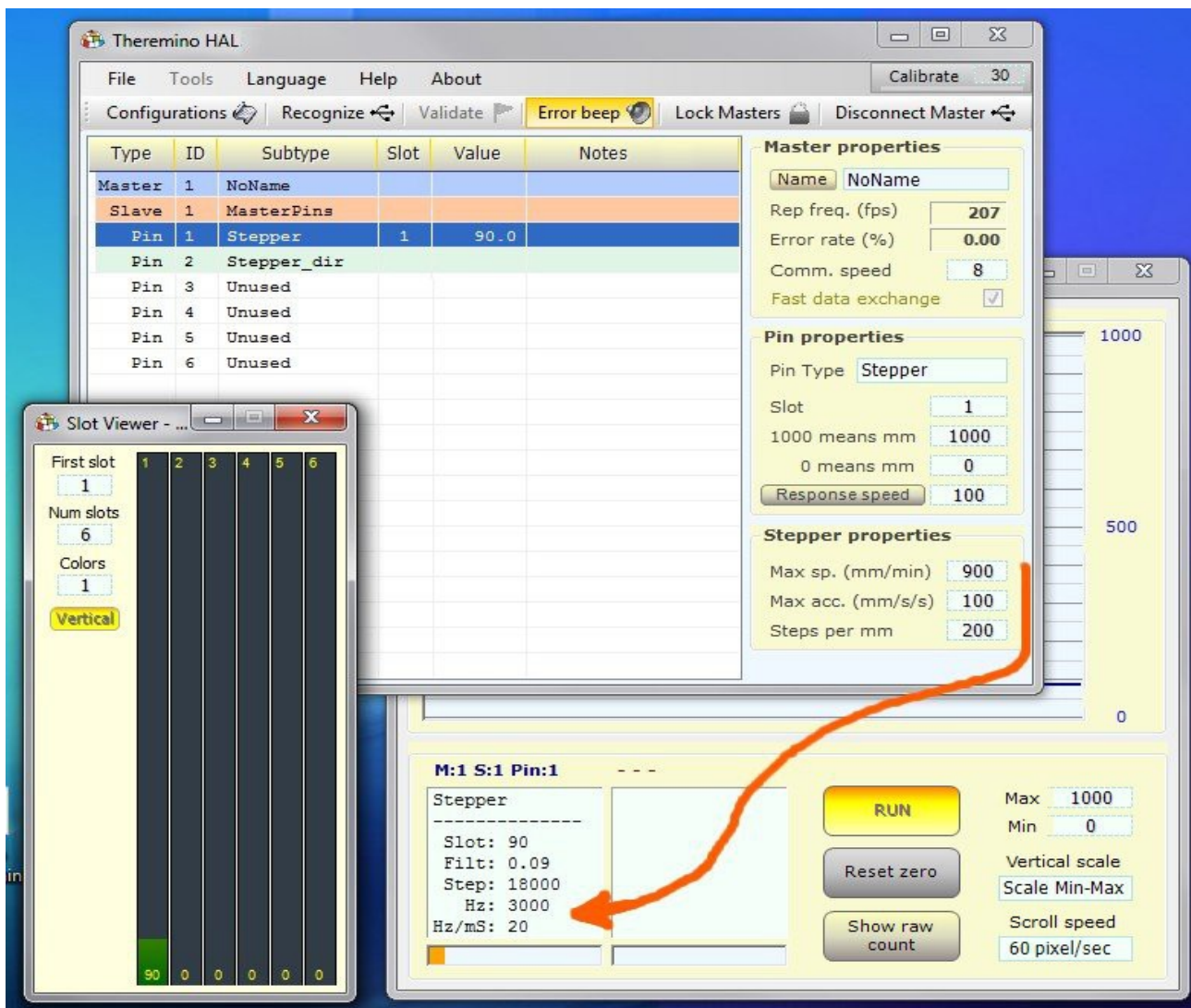
The CNC application, knowing the remaining distance and destination (specified by herself), can calculate, with a simple subtraction, the actual location of the engine. Knowing the location of each engine, in every moment, control algorithms are simplified and their operation is more accurate.

More detailed info about Stepper Pins and about stepper motor drivers, here:

<http://www.theremino.com/en/hardware/outputs/motors>

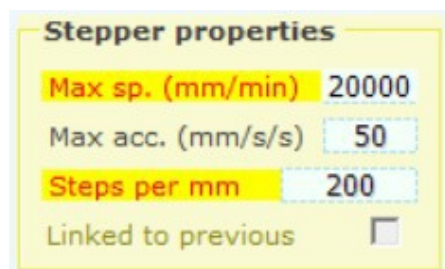
<http://www.theremino.com/en/technical/pin-types>

The "Output" Pin types --> Stepper details



To show Pin details double click on the line of the stepper Pin. In the second window, at the bottom, you read the details of the selected Pin.

Pin details show the product of the "Max Speed" and "Steps per mm" in Hz (steps per second). These values are useful during testing and to determine how many micro-step to use. In some cases, It can be useful to know the rough target (in steps), in place of destination in mm.



The maximum step frequency is defined in the "IotModuleSetup.h" file. Depending on the selected range, the maximum obtainable frequency ranges from 1200 Hz to 307200 Hz. If the maximum frequency is exceeded with the set parameters, you are warned by the boxes "MaxSpeed" and "Steps per mm", which are colored yellow and Orange.

If the boxes are colored, then you must reduce "MaxSpeed", or reduce "Steps per mm", also decreasing the microstep setting on the driver.

The "Input" Pin types <-- Dig / ADC / Cap

◆ Dig_in and Dig_in_pu

Pin properties	
Pin type	Dig_in
Slot	1
Max value	1000
Min value	0
Response speed	100

This pin type provides a digital input.

The voltage value is read with a digital input (low if less than 0.8 volts and higher if greater than 2.5 volts) and transformed into an ON-OFF information that ultimately become "Max value" and "Min value." Then the value is filtered with "Response speed" and written into the slot. The filtering produces intermediate values approximately proportional to the time relationship between On and Off

◆ Adc_16

Pin properties	
Pin type	Adc_16
Slot	1
Max value	1000
Min value	0
Response speed	30

This Pin type provides an analog input.

The voltage value from 0 Volt to 3.3 Volt is converted into a number between "Min value" and "Max value." The value is 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.

◆ Cap_16

Pin properties	
Pin type	Cap_16
Slot	1
Max value	1000
Min value	0
Response speed	30

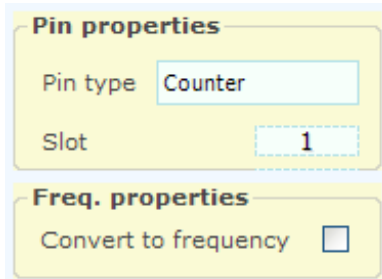
Touch properties	
Min variation	10
Proportional area	0

This type of Pin allows you to read simple buttons like with a Makey Makey (<http://vimeo.com/60307041#>) but with superior performance. (the keys are not resistive but capacitive and therefore can be adjusted to work even just by touching them, without contact, through an insulator and without an earth connection additional wire)

In addition to the classic MakeyMakey operation, it is also possible to obtain a gradual control such as with the "slider" sliders, the control of "expression" determined by the pressure speed of the keys or reading rough capacitive values, such as humidity sensors. Further informations on capacitive buttons on [these pages](#).

The "Input" Pin types <-- Counter

◆ Counter and Counter_pu

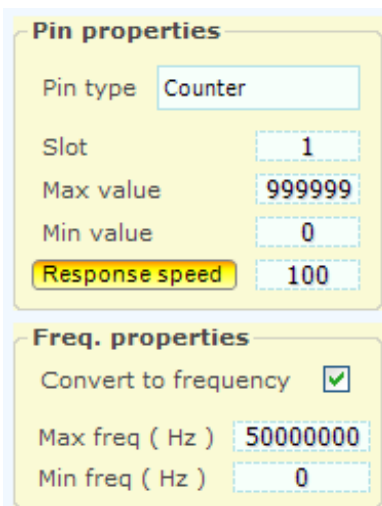


The screenshot shows the 'Pin properties' dialog box. Under the 'Pin properties' section, 'Pin type' is set to 'Counter' and 'Slot' is set to '1'. Under the 'Freq. properties' section, the 'Convert to frequency' checkbox is unchecked.

All pins can be programmed as Counter or Counter_pu.

The counting is done with a hardware counter that works up to frequencies of several tens of MHz.

◆ Counter and Counter_pu with the "Freq" option



The screenshot shows the 'Pin properties' dialog box with additional frequency-related settings. Under 'Pin properties', 'Pin type' is 'Counter', 'Slot' is '1', 'Max value' is '999999', 'Min value' is '0', and 'Response speed' is '100'. Under 'Freq. properties', the 'Convert to frequency' checkbox is checked, 'Max freq (Hz)' is '50000000', and 'Min freq (Hz)' is '0'.

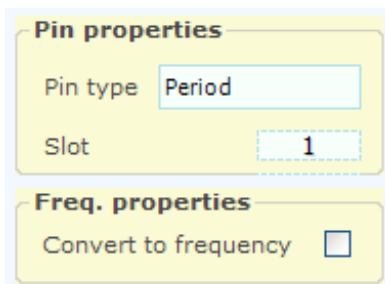
The Pins programmed as Counter or Counter_pu can be transformed from counters to frequency meters.

The frequency value to be compared "freq Min" and "Max freq", is then compared between "Min value" and "Max value," filtered with "Response speed", and finally sent to the slot.

The frequency counter with the option are suitable for fairly high frequencies extent (from 10 KHz up). For low frequencies (from a few kHz up to 0.02 Hz) it is preferable to use the "Period" explained on the next page, which provide a measure more stable and less noisy.

The "Input" Pin types <-- Period

◆ Period and Period_pu



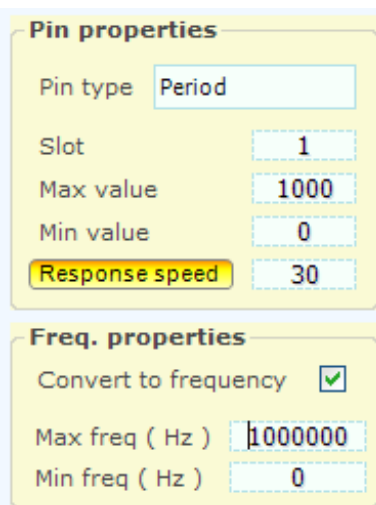
The screenshot shows two configuration panels. The top panel, titled "Pin properties", has "Pin type" set to "Period" and "Slot" set to "1". The bottom panel, titled "Freq. properties", has "Convert to frequency" set to an unchecked checkbox.

This type of Pin measure the period of a repetitive wave form, from rise to rise, from a minimum period of 50 uS, up to a maximum of about 260 seconds.

The resolution is three microseconds and accuracy is +/- 1% in a range of ambient temperatures from 0C to 50C

The value read is in microseconds and fractions of a microsecond. If the value is less than 50 uS it is discarded. In this way possible errors, due to electrical disturbances during the signal fronts, are eliminated.

◆ Period and Period_pu with the "Freq" option



The screenshot shows two configuration panels. The top panel, titled "Pin properties", has "Pin type" set to "Period", "Slot" set to "1", "Max value" set to "1000", "Min value" set to "0", and "Response speed" set to "30". The bottom panel, titled "Freq. properties", has "Convert to frequency" checked, "Max freq (Hz)" set to "1000000", and "Min freq (Hz)" set to "0".

The Pin programmed as Period or Period_pu can be transformed from counter to frequency meter.

This technique allows to measure very low frequencies (up to less than a tenth of a Hertz) with very high resolution. The max measurable frequency is 20 KHz.

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.

If the frequency value exceeds 20 KHz it is discarded. In this way possible errors, due to electrical disturbances during the signal fronts, are eliminated.

The measures carried out with the Period Pins are affected by few microseconds of noise. So at the frequency of 1000 Hz we could obtain approximately ten-bit of signal to noise ratio. These characteristics are slightly inferior to those obtainable with the Master modules, because the Master are based on a Micro-Controller, while the ESP32 contains an operating system (Free-RTOS).

The "Input" Pin types <-- Encoder

Pin properties
Pin Type
Slot
Max value
Min value
Response speed

Pin properties
Pin Type

All pins can be programmed as Encoder.

Each encoder requires two input Pins: Encoder_A and Encoder_B or Encoder_A_Pu, and Encoder_B_Pu. This pair of inputs reads the two "quadrature" phases of the Encoder.

The encoder count is written in the slot associated with the "Encoder_A" Pin. Each "Encoder_A" pin uses 16 bits for data transmission, while the "Encoder_B" is just a placeholder and will not send data.

The encoder reads the angular position of a pin, such as potentiometers, but the number of rounds is unlimited. The generated count goes from 0 to 65535 (16 bits). When the count exceeds 65535 the number starts from zero. This system allows many applications, asynchronous to each other, to read the sequence number without losing counts.

For more information on encoders, read this page:

<http://www.theremino.com/en/hardware/inputs/sensors#encoders>



There are Encoders like small pots (the best known are the KY-040 of this image). These models are mechanical and provide 18, 20 or 24 pulses per revolution depending on the manufacturer. The firmware obtains from these pulses 72, 80 or 96 angular positions for each revolution, if adjusted in the software mode (x2), or 36, 40 or 48, if adjusted in hardware mode (x4).

Professional (optical or magnetic encoder) work without problems, but those with mechanical contacts, such as that of this image, probably will malfunction, because the lotModule inputs are not Schmitt Trigger type. To make them work without error, you should add the Schmitt Trigger module located on the Theremino site.

Set the module encoders as "Software" or "Hardware"

By changing "IOTMODULE_ENCODERS_TYPE" in the "lotModuleSetup.h" file (and reprogramming the module), you can change the behavior of the encoder. With hardware encoder there are no frequency limits. But with software encoders, the maximum counting speed is limited to a few tens of KHz, and is dependent on the load on the micro-controller. If necessary, you can limit the frequency using encoders with fewer steps per revolution.

In compensation software encoders have a greater immunity to disturbance and give a double number of counts (x4), compared to those hardware (x2), then you will get a higher resolution.

The "Input" Pin types <-- Adc24

To enable the module the firmware must be programmed with IOTMODULE_ADC24_PINS from 1 to 16, then you select the Pin 60 line and set its Pin Type as "Adc24".

When the Adc24 works, the value of the Pin 60 (Adc_24) grows in proportion to the sampling rate (with 100 sps and MaxSpeed filter, the value increases by 100 values per second).

Pin properties
Pin Type: Adc_24

Adc24 properties
Samples/sec.: 3200
Filter: Max Speed

Properties of the Pin 60 configured as Adc24

Samples per second: Sampling rate which is divided between all active inputs (for example with 600 sps and three inputs, each input is sampled 200 times per second).

Filter: Eight filters are available to choose from, for the best compromise between noise and speed of response.

Pin properties
Pin Type: Adc_24_ch
Slot: 13
Max value: 1000
Min value: 0
Response speed: 100

Adc24_channel props
Type: Differential
Gain: 1
Biased to Vmax / 2: ☐

Properties of the Pins "Adc_24_ch"

Important to note that the 16 input Pin are in pairs (1-2, 3-4 ... 15-16) and then their type (Differential, Pseudo and Single) and the gain (from 1 to 128), are valid for both the inputs of the pair.

The "Biased to Vmax / 2" can instead be activated separately on each of the sixteen inputs.

Pin properties
Pin Type: Adc_24_ch_b

Adc24_channel props
Type: Differential
Gain: 1
Biased to Vmax / 2: ☐

Properties of the Pins "Adc_24_ch_b"

If the pair of inputs is set as Differential, the second couple input becomes "Adc_24_ch_b".

This is a special type that does not send data to the slot, and it is only a placeholder for the sensor reference connection.

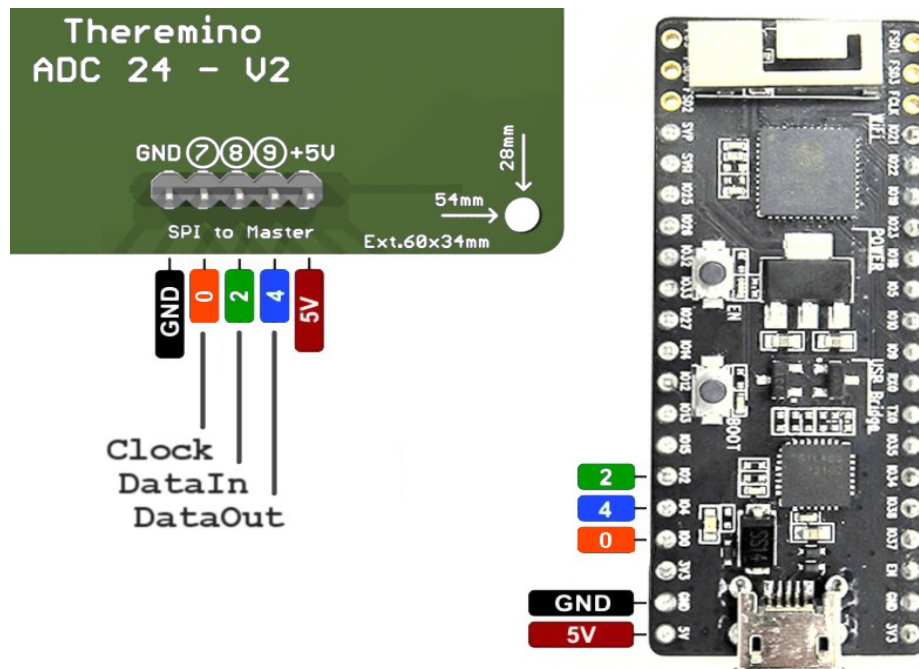
Information on the Adc24 module and usage manual with detailed explanations and examples:
<http://www.theremino.com/en/hardware/adapters#adc24>

The Adc24 module is connected to five pins of the lotModule as shown on the next page.

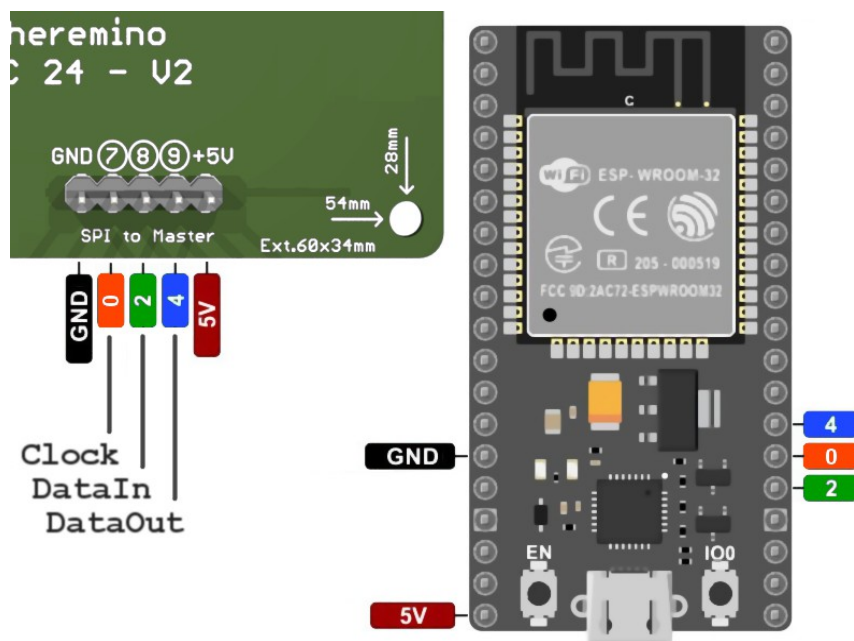
Adjacent wires with strong signals could disturb this connection.

If the wires are long, use shielded cable and decrease the communication frequency as explained on [this page](#).

Adc24 module connections



Connections between the Adc24 and the PICO KIT V4



Connections between the Adc24 and the ESP WROOM 32

The "Generic" Pin types

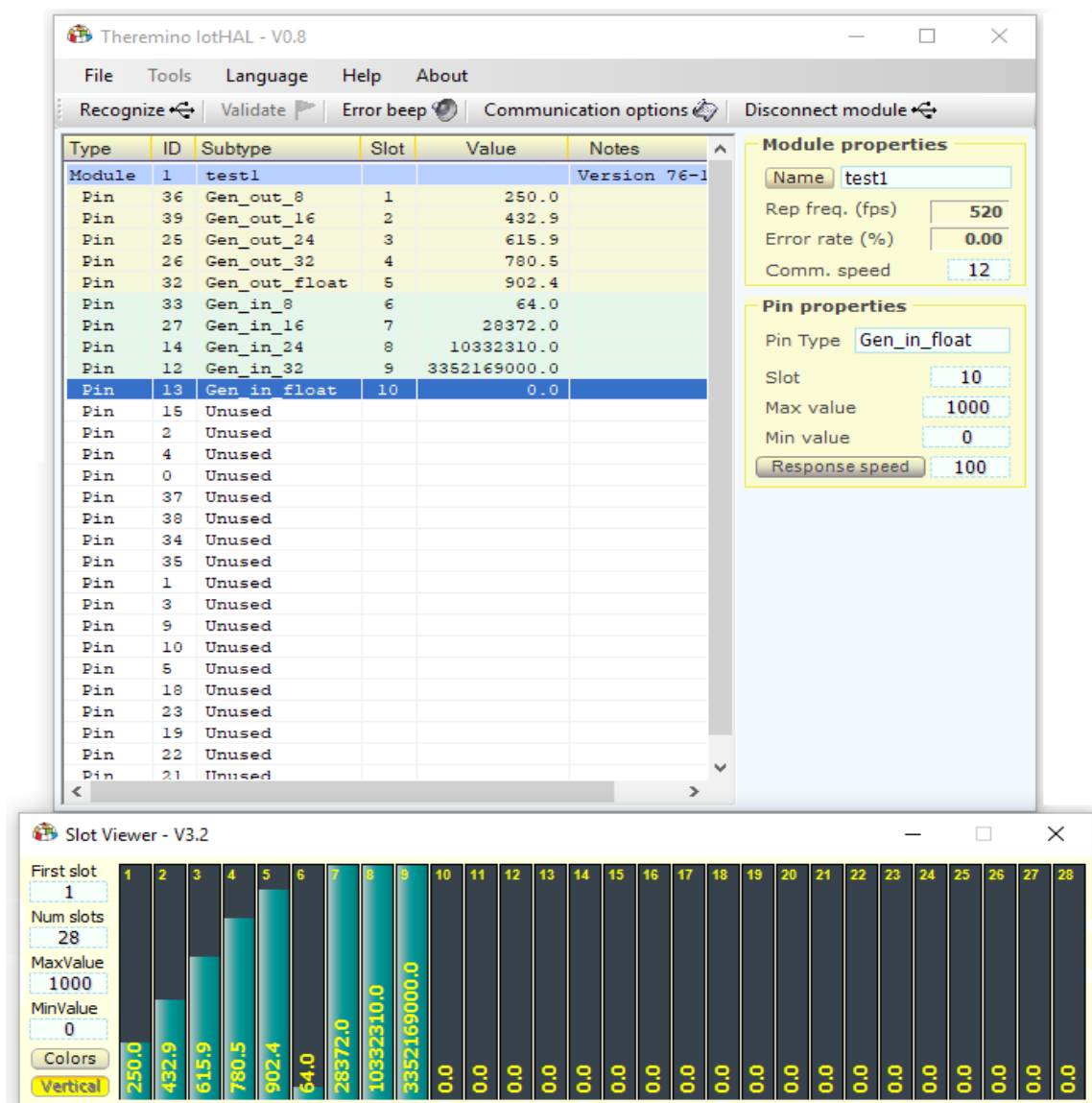
To read and write the normal lotModule Input-Output Pins, it is not necessary to use the functions explained in this page, just set up the Pin with the lotHAL and everything works immediately.

Instead for generic Pin where you also write the firmware. So you have to know how to plan and organize without error, and it takes a good experience in programming.

What explained 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 your own firmware, before sending them to PC, or from PC to the actuators.

The lotHAL Generic Pins, (or Virtual Pins), have been introduced to allow bidirectional data exchange of numeric variables, between the internal data to the firmware of the lotModule, and the Slots managed by the lotHAL. These are additional variables unrelated to the functions of the physical Pin and its value.

In the GenericWrite and GenericRead you indicate a Pin and the same Pin must be configured in the lotHAL with the corresponding type Gen_in or Gen_out. If you make mistakes you will have unpredictable results.



Programming the "Generic" pins

The number of necessary generic Pin depends on the characteristics of the hardware device, and the number of values that you want to transfer. Any physical Pin of the module can be used as a generic Pin. If all the physical pins are used or you have to use a large number of generic pins, you can add a certain number of virtual ones by recompiling the firmware. This number is set in the constant `IOTMODULE_GENERIC_PINS` defined in the `lotModuleSetup.h` file and can vary from 0 up to a maximum of 20, regardless of the type of generic Pin used (float, 24, 16 or 8 bits). The names of the generic "virtual" Pins start from Pin 80 and end at Pin 99.

Any modification to the constant `IOTMODULE_GENERIC_PINS` must be followed by recompiling the firmware in the Arduino IDE, followed by the `lotModule` programming and finally the validation of the new configuration by pressing the "Valid" button on `lotHAL` application.

Send data from hardware sensors to the applications on the PC

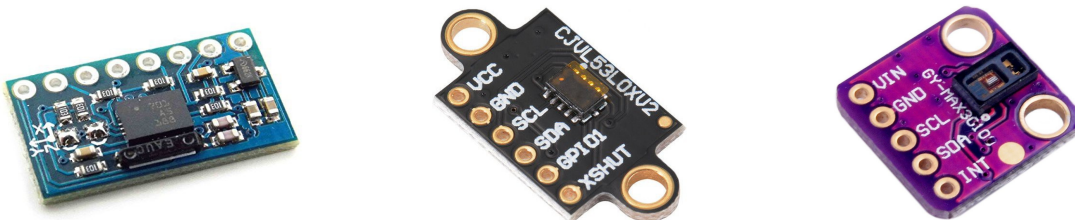
- You choose a Pin to use
- In the Loop which is in the ".ino" file, you read the sensor and interpret its data
- In the Loop which is in the ".ino" file, data is sent with "GenericWrite" (8, 16, 24 or Float)
- On the `lotHal` you read the data by configuring the Pin as `Gen_in` (8, 16, 24 or Float)
- On `lotHal` you set the chosen Pin with a slot through which the data of the sensor will go to the applications that use it

Send data from the PC applications to the hardware actuators

- On the `lotHal` you choose a Pin to use
- On the `lotHal` you set the chosen Pin with a slot through which to send numeric data
- On the `lotHal` you send data to module configuring the pin as `Gen_out` (8, 16, 24 or Float)
- In the ".ino" file "Loop", you read the data with `GenericRead` (8, 16, 24 or Float)
- In the ".ino" file "Loop", you interpret the numeric data and send it to the hardware.

Reading special sensors with the Generic Pins

The "Read an I2C sensor" document, which you download from [This Page](#), is a good example of using the "Generic Pins".



Examples of use of the "Generic" Pins

In the following lines you can see an example of code to be inserted in the Loop Arduino for the transfer of the four basic data types of generic Pin.

In the first two rows the 8-bit value of the slot associated with the Pin Generic 32 (slot and input Pin in the IotHAL, genericRead function in the IotModule) is transferred back to the associated slot at the generic Pin 33 (genericWrite function in the IotModule, Pin and slot output in the IotHAL).

The other two pairs are doing the same, manipulating 16 and 24-bit values and using the Pin 27, 14, 80 and 81. The last pair does the same with "Float" numbers and using the Pin 82 and 83.

```
uint32_t n;  
float f;  
  
n = IotModule.genericRead8(32); // IotHAL Pin 32 configured as gen_out_8  
IotModule.genericWrite8(33, n); // IotHAL Pin 33 configured as gen_in_8  
  
n = IotModule.genericRead16(27); // IotHAL Pin 27 configured as gen_out_16  
IotModule.genericWrite16(14, n); // IotHAL Pin 14 configured as gen_in_16  
  
n = IotModule.genericRead24(80); // IotHAL Pin 80 configured as gen_out_24  
IotModule.genericWrite24(81, n); // IotHAL Pin 81 configured as gen_out_24  
  
f = IotModule.genericReadFloat(82); // IotHAL Pin 82 configured as gen_out_float  
IotModule.genericWriteFloat(83, f); // IotHAL Pin 83 configured as gen_in_float
```

Once these lines are written in Arduino Loop() function, you try them with the IotHal, by changing the value of the Value column with the mouse, or with the aid of the SlotViewer.

Transfer integers from the IotModule to the PC Slots, and vice versa

Often you use integers in 16-bit (0 to 65535). But in some cases you may have small values, and then stay in 8-bit (0 to 255), or very large values, and then you need 24 bits (0 to 16777215).

The standard of Theremino system is to relate all the values to a "normalized" excursion from 0 to 1000. But in some cases, for example, a sensor that provides an integer value, it can be useful to transfer the product value without changing it. To achieve this behavior edit box "Max value" with an appropriate value to the number of bits set (255 for 8 bits / 65535 for 16-bit / 16777215 for 24 bits).

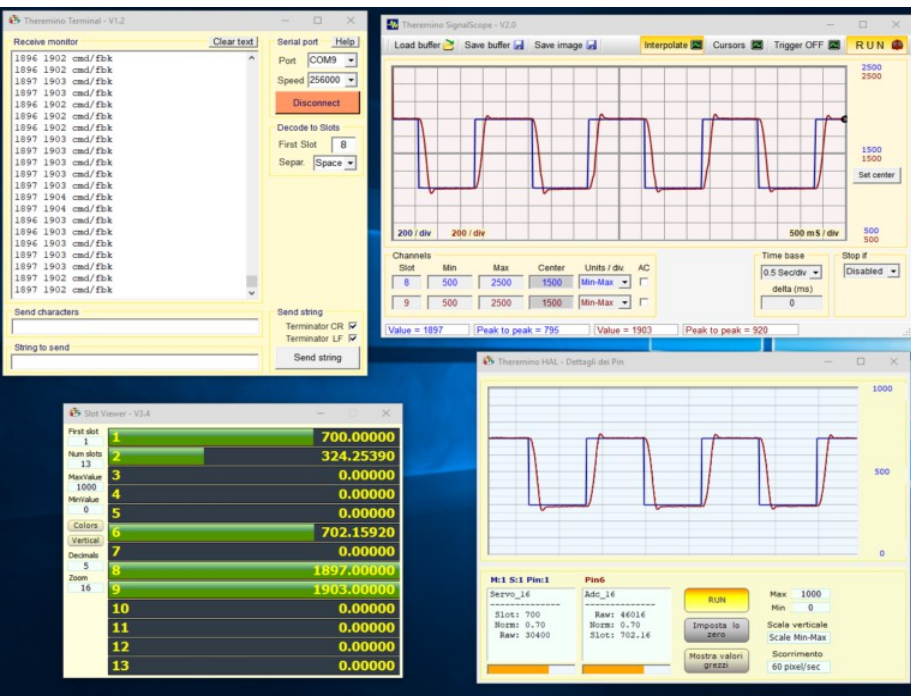
Transfer Float numbers from the IotModule to the PC Slots, and vice versa

The Float numbers, called "Float" or "Double" in the Arduino IDE, or "Single" in the PC applications, can be huge or very small and always have the same precision. These numbers are transferred as they are, if you set "Min value = 0" and "Maximum value = 1000". Otherwise you can use "Min value" and "Maximum value" to scale them at will.

Debugging with the "Generic" Pins

When writing the module firmware (both the one that users can write in the xxxx.ino file, and the one written by us, which is in the "IoTModule" folder), a common need is to know the values of the internal variables.

Historically, hardware debuggers were used for this task, which were expensive and difficult to connect. A second method (which also uses Arduino) is to send the values of the variables through the serial link.



In the firmware, the values of the variables to be followed are sent to the serial port and the [Terminal application](#) interprets the values and sends them to the Slots.

In practice, you can see the variations of the internal firmware variables, as if you connected an oscilloscope to them.

You can then use the [Signal Scope](#) (or other applications of the theremino system), to see the variation over time of their values.

It seems to connect an oscilloscope inside the firmware.

For those who use micro, and especially for those who write servomechanism control firmware, this is an exceptionally useful possibility!

By means of the IoTHAL application it is possible to use the Generic Pin and obtain the same performance also via WiFi (without connecting the USB cable for the serial).

The values of the internal firmware variables are transferred directly to the desired slots using one of the Generic instructions, for example the following:

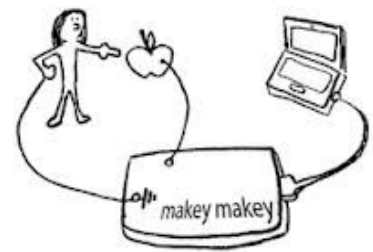
```
IoTModule.genericWriteFloat (Slot number, Value to write);
```

Resistive or capacitive buttons

To see what you can do with simple buttons, watch this great video of the Makey Makey: [#http://vimeo.com/60307041 #](http://vimeo.com/60307041)

The buttons on the Makey Makey are resistive and not capacitive, it will only work if the resistance is less than about 4 Mega Ohm, it needs an additional wire as ground reference and doesn't work through insulating materials such as plastic. Furthermore, the buttons on the Makey Makey are only six (not expandable), each Makey Makey can provide up to 20 keys, you can connect more Makey Makey in chain, but at the end, all keys are sent to the keyboard that manages just a maximum of six: www.makeymakey.com/faq Finally, the Makey Makey keys have only on/off operation, without intermediate adjustments and do not feel keys pressing speed (Velocity).

Makey Makey!



Theremino System capacitive keys, can do much more. They can be expanded at will, by adding Master modules (6 buttons each) and lotModules (9 buttons each) in an unlimited number, as shown here: www.youtube.com/watch?v=NbC5kIRS_6s and here: www.youtube.com/watch?v=2RzwUfXhFZY

Moreover, Theremino System keys, can provide a gradual control as well, like if they were sliders, and control the "expression", determined by the keystrokes speed.

The three types of capacitive keys

Touch properties	
Min variation	20
Proportional area	0

Touch properties	
Min variation	20
Proportional area	150

Touch properties	
Min variation	40
Proportional area	-30

- **On/off keys**
"Min variation" from 10 to 50
"Proportional area" should be ZERO
- **Proportional keys**
"Min variation" from 10 to 100
"Proportional area" from 100 to 200 (for a maximum of about 1000)
- **Keys with velocity**
"Min variation" from 25 to 50 (adjusted for maximum output)
"Proportional area" -30 (adjust to a maximum of about 1000)

Generic capacitive measuring

Touch properties	
Min variation	40
Proportional area	-30

- **Capacitive sensors (humidity sensors, variable caps. etc.)**
"Min variation" from -1 to -1000 (minimum value setting)
"Proportional area" from 1 to 1000 (maximum value setting)

Caution: with this type of Pin is not obtained a measure of the electrical capacity, but only the value of a sensor or a mechanical position. Many factors contribute to get a non-linear measurement, first of all the capacity of the connection cable. The cable must be very short, and after calibration, you should not move it. In all cases it will have to make adjustments of scale and appropriate linearizations in the software.

"Min Variation" and "Proportional Area" parameters

Min variation

Eliminates small variations and prevents electrical noise from triggering keys, without having touched them. Raising this parameter, keys become less sensitive. It should keep as low as possible, just enough to eliminate all noises.

For keys with velocity, the best setting for this parameter is obtained by pressing the button quickly and repeatedly and adjusting "Min variation" with the mouse wheel, in order to get the maximum output signal. To make this adjustment easier, temporarily set "Proportional area" with a negative number large enough, for example -50.

Proportional Area

Is set at about 1000, when the finger is in the maximum position of the slider, or when you press buttons, as fast as possible.

This value should normally be higher for Pin 1 and 2 (less sensitive), or in case of long wires and large objects.

Zero calibration of the capacitive buttons

If you change the mechanical arrangement of keys or their position, if you move the wires that connect them or if you approach metal objects, while the HAL program is working, it is possible to lose the zero calibration of the keys. If zero is not well calibrated, capacitive keys can become less sensitive or even not work at all.

If you remove capacity from the keys (shortening the wires or take them away from metal objects) calibration is automatically made immediately, but it is impossible to distinguish an increase in capacity due to a finger or a shift in the wires.

We have tried many methods of automatic recognition, with slow drift or timed calibration, but none worked well and all compromise the accuracy of the normal keys operation.

You should therefore, try not to move the wires of the keys, the keys themselves and conductive objects within a radius of about ten centimeters, during operation.

To check if a key is calibrated, release your hands from the button and verify in the details of its Pin, that the values "Smoot" and "Mean" are equal to each other, or are very close (not more than one point of difference)

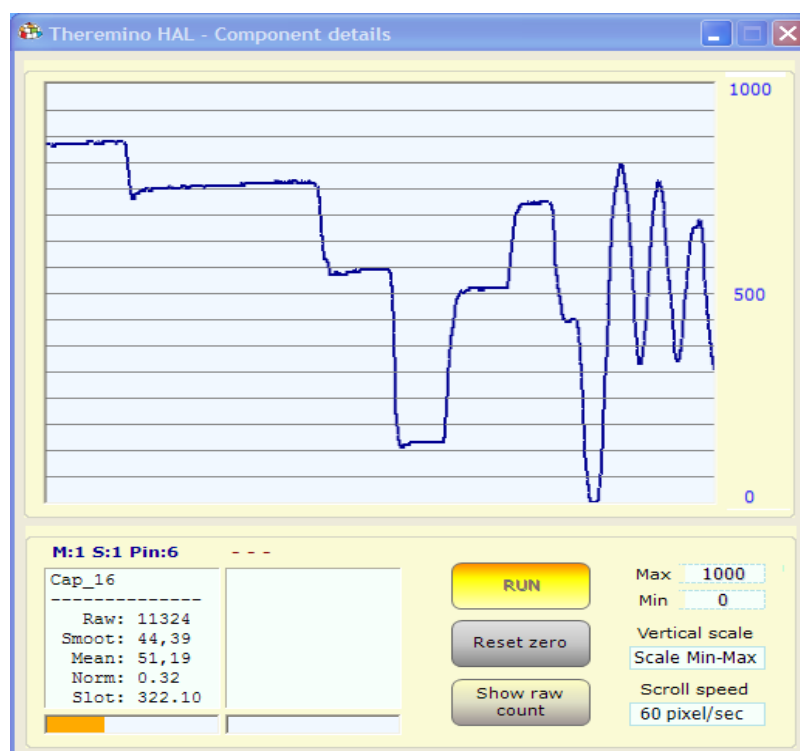
When in doubt, press Calibrate (keep your hands away from the keys while performing the zero calibration)

Reading the capacitive sensors

Setting MinVariation with a negative value completely changes the operating mode. For example, humidity sensors (capacitive models and without control circuit) could be connected. It could also improvise sensors to read the rotation of a Pin or linear displacements. Sensors of this type may be simple, but also very reliable.

By Setting Min Variation with a negative value, the meaning of Min Variation and Proportional Area changes:

- ◆ Min Variation sets the minimum and Proportional area the maximum, of measurable capacity.
- ◆ The Calibration button is disabled. The calibration is fixed and is the same value of Min Variation.
- ◆ The range of usable capacities is from a few pF to a few nanoFarad.



The "Slider" type capacitive keys



"Proportional area" must be a positive number, this determines the "Proportional" operation.

With a capacitive button of this shape, a continuous adjustment similar to a cursor "slider", can be obtained.

The control is carried out with a finger, all top=1000, all bottom=0

These keys are suitable for volume control and can act as a "panic button" (when you unplug your finger from the button, the volume is zeroed)

These are the standard setting or thr "Slider" buttons (note 1)

Pin properties	
Pin type	Cap_16
Slot	2
Max value	1000
Min value	0
Response speed	30

Touch properties	
Min variation	20
Proportional area	150

"Max value" normally set at 1000 (Note 2)

"Min value" normally held at zero (Note 2)

"Response speed" is normally set to 30 (light filtering)

"Min variation" is set normally from 10 to 100 (better to raise it slightly to obtain the maximum sensitivity in the lower part)

"Proportional area" is normally set to 200 (about 100 for less sensitive keys or with long wires)

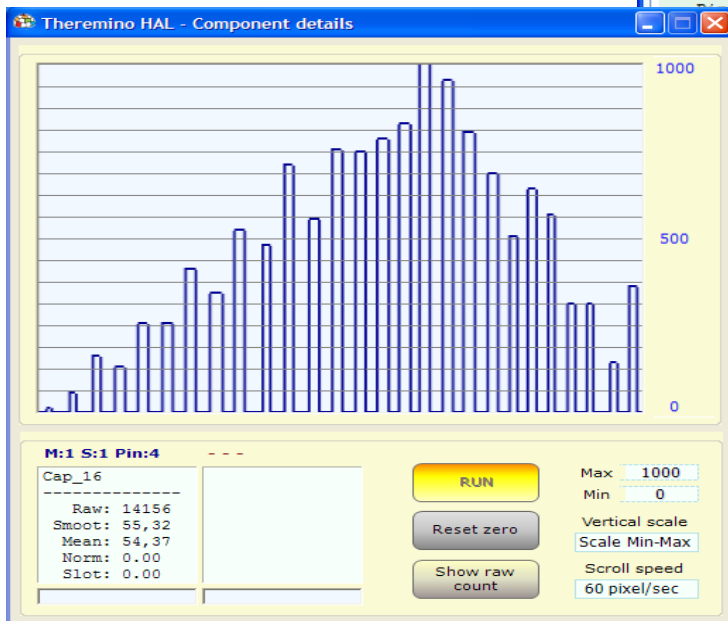
(Note 1) For keys of the "Slider" kind, it is always better to use "Cap_16"

(Note 2) To reverse the output signal, it can be exchanged Min with Max (Max = 0 and Min = 1000)

The capacitive keys with "Velocity"

"Proportional area" should be a negative number.

This determines the "Velocity type operation"



Theremino HAL - V3.0						
File Tools Help About						
Edit configurations Recognize Validate Beep on errors Calibrate 20						
Type	ID	Subtype	Dir.	Slot	Value	Notes
Master	1	Theremin				
Slave	1	MasterPins				
Pin	1	Unused				
Pin	2	Cap_16	get	2	0.0	
Pin	3	Cap_16	get	3	0.0	
Pin	4	Cap_16	get	4	0.0	
Pin	5	Cap_16	get	5	0.0	
Pin	6	Cap_16	get	6	0.0	
2	CapSensor					
1	Cap_sensor	get	0	12.8		
2	Theremin2					
1	MasterPins					
1	Dig_out	set	1	Sleep		
2	Unused					
3	Cap_16	get	89	0.0		
4	Cap_16	get	57	0.0		
5	Cap_16	get	55	0.0		
6	Adc_8	get	66	1000.0		

Keyboards that allow you to play notes loud or soft, depending on how you press the keys, are very popular for musical applications.

The capacitive buttons can be adjusted to measure the pressure speed of the key and turn it into a value from 0 to 1000 (approximately). For a good operation of the "Velocity" the communication speed must be high (from 200 to 500 fps), and must adjust buttons, one by one, so as to obtain a maximum value slightly greater than 1000

Pin properties

Pin type: Cap_16

Slot: 2

Max value: 1000

Min value: 0

Response speed: 30

Touch properties

Min variation: 40

Proportional area: -30

These are the keys for the adjustments with "Velocity"

"Max value" is normally held at 1000 (Note 1)

"Min value" is normally held at "0" (Note 1)

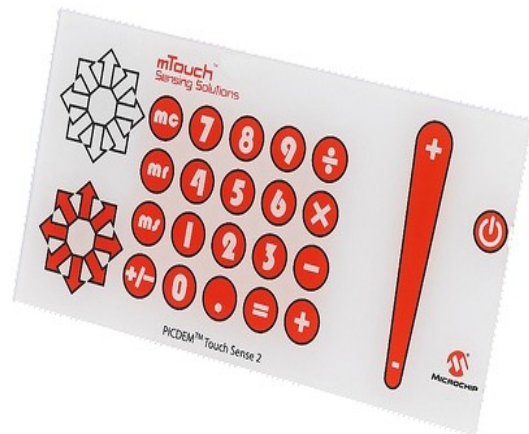
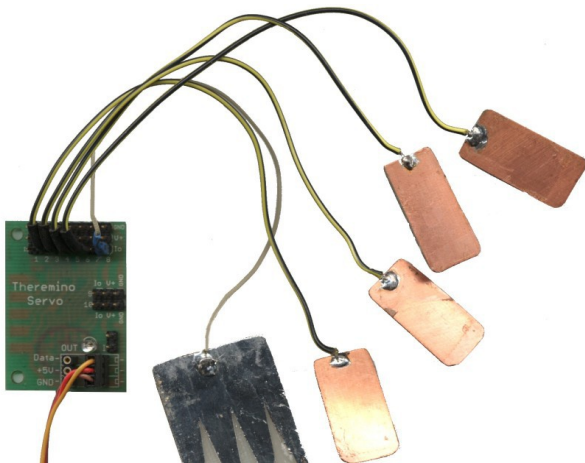
"Response speed" is normally held to 30 (best not to edit)

"Min variation" It is normally set to 50 (and about 25 for the Pins 1 and 2 that are less sensitive, or for keys with long wires)

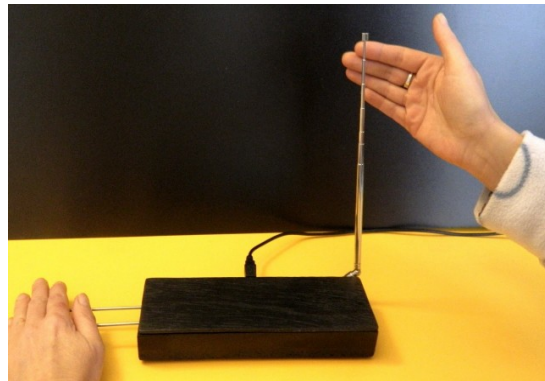
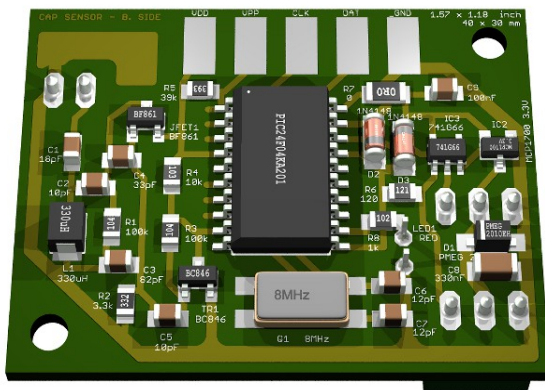
"Proportional area" is set normally to -40 (or about -20 for 1 and 2 keys that are less sensitive or wires with long keys)

(Note 1) To invert the output signal you can exchange Max and Min (Max and Min = 0 = 1000)

Differences between CapacitiveKeys and CapSensors



Capacitive keys cannot replace CapSensor modules, the first work only at short distances (from a few millimeters to a few centimeters), while CapSensors work up to distances of several meters, and can be adjusted for an almost perfectly linear response. The capacitive keys on the other hand are much cheaper and are better suited to arrange keyboards with many keys.



Mechanical construction of the capacitive keys

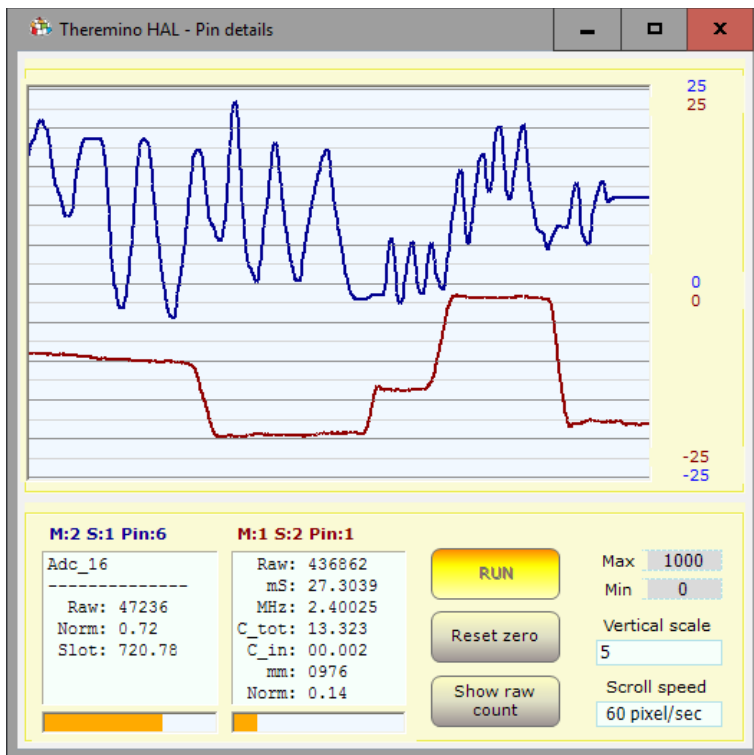


Make sure that capacitive keys are well isolated, otherwise it may be enough a small spark of static electricity, to produce communication errors. Nothing is broken, but communication is interrupted and you have to press the button "Recognize".

Face them with copper below and vetronite (thin) above, or even better, copper held above, but a thin sheet of insulating plastic is added, possibly printed in color with the shape of the keys, as in the image at the beginning of this page.

The wires going from the keys to the Pins, must be as short as possible and there must be at least 5 or 10 millimeters, between them. The key operation and the noise reduction, improves by decreasing the capacitance. Experiments were conducted in "impossible" situations, with long wires and any kind of keys, from potted flowers to various fruits and, with individual adjustment, always was running good.

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 "Scale Min-Max", which corresponds to the Min and Max textbox values.

Or it can be set in 24 levels from 0.01 to 50000 points per vertical division (ten dark lines). When the vertical scale is set to those values, to center the traces you press "Reset zero."

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/second to 60 pixel/second.

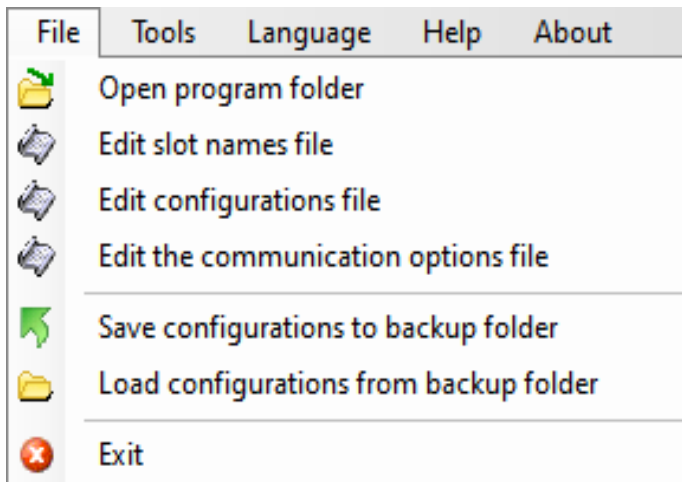
The two text boxes, show the internal details of the Pins. The title indicates which Pin is analyzed, in this image the text "M:1 Pin:2" means "Module 1, Pin 2" ("S" is not used in the IoTHAL)

Details of the Pin, may help in the control and regulations of Input Output devices (sensors and actuators).

Some types of Pin are more complex and have more intermediate values. In general, a "Raw" value exists, with very variable values, depending on the type of Pin, a "Normalized" value which always goes from 0 to 1 and a "Slot" value which normally ranges from 0 to 1000 and that can be considered as the "Simplified" value available on Slots, 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 processed by a smoothing 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 only "0" or "1" (used by DigOut only)

Menu commands



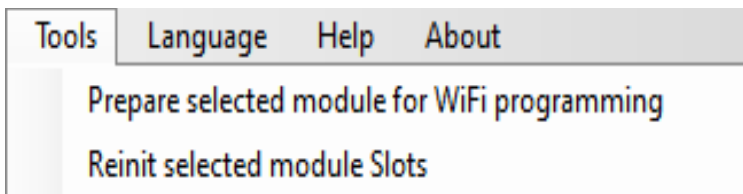
Open the work folder it may be useful to modify the documentation files and languages.

Edit the file: "SlotNames" Comments (slot or names) are explained on page 8.

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

Change communication options the communication options file is opened to change the valid IP addresses, the communication port and the names of valid lotModule. The instructions and examples of options, are in the same communication options file, called "CommOptions.txt".

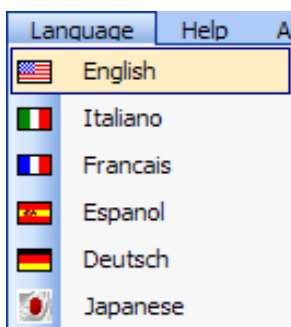
Save the configuration as a backup It allows to make the configurations of the security copies. If the configuration files are modified by mistake you can then load them from a previous version. Previous versions show the date and time they were saved.



Prepare for WiFi programming

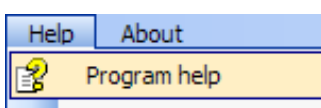
see the details on the last pages of this document.

Re-initialize module Slots re-number the Slots of the selected module.



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

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



This command opens the documentation file (this file itself).

Commands Toolbar



Recognize

Is used to recognize the active modules.

Validate

When editing the modules, you are warned 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 it will make the new configuration effective.

Error beep

When pressed the communication errors are highlighted with a sound.

Communication Options

The communication options file is opened to change the valid IP addresses, the communication port and the names of the valid lotModules. The instructions and examples of use are in the same communication options file, which is called "CommOptions.txt".

Disconnect Module

Deletes the selected module from the list. This way you can delete unwanted modules without having to physically turn off them.

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 connects only to Modules with the name in the CommOptions file. 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 Modules, 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 spin boxes

Draw speed (fps) 5

The numerical boxes HAL (and all other applications of Theremino system) have been developed by us (Note 1) to be more comfortable and flexible than Microsoft's original 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
-
- By pressing SHIFT the variation speed is multiplied by one hundred
 - By pressing CTRL the variation speed is multiplied by ten
 - By pressing ALT the rate of variation is divided by ten

Moving the mouse up and down allows wide and fast adjustments

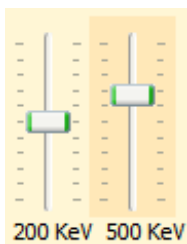
The mouse wheel allows a comfortable and immediate setting

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 a 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 naming the source. The "Open" source, serves 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 more intuitive to use the first for horizontal cursors and the second for vertical sliders.

Questions and answers

Can I change the text of the panels, to different languages?

Of course, just edit the file: "..\Docs\Language_Eng.txt" and "..\ Docs\Language_Ita.txt"

For German, French and Spanish, just copy the file English three times with the following names: "..\Docs\Language_Deu.txt", "..\Docs\Language_Fra.txt", "..\Docs\Language_Esp.txt"

Can I edit the configuration file?

Normally the association between configurations and modules is kept aligned by ThereminoHAL, 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. But 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 alignment between hardware configuration and you lose. 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 details" window and minimize the main window.
- Reduce the "Comm speed", as explained in the opening pages of this document.

What to do if the module does not work

The module connects to the network but does not appear in the IoT HAL

Sometimes the network may be in an abnormal state, some say it is due to the Proxy, others to be due to configuration errors. We recommend that you perform the following sequence:

- ◆ Click on the search box (bottom left)
- ◆ Write CMD and wait for it to appear "Command Prompt"
- ◆ Click the right mouse button and choose "Run as administrator"
- ◆ Copy the following two commands, one by one, and give ENTER to everyone.
`netsh int ip reset`
`netsh winsock reset`
- ◆ Try again if the modules appear on IoT HAL (or restarting it, or by pressing "Recognize")
- ◆ If you are even trying to make a restart.
- ◆ If they do not yet appear also try the following command (but completely reset the firewall and therefore may block some programs).
`netsh advfirewall reset`

Check the starting procedure of IoTModule

- ◆ Connect the IoTModule to a USB port
- ◆ Reprogram the module "ESP_LOG_VERBOSE " in the "IoTModuleSetup.h" file
- ◆ Start the app. [Theremino Terminal](#) (or the Arduino IDE "Serial Monitor")
- ◆ Connect COM port which is connected with the IoTModule Speed = 115200
- ◆ Press the reset button and read the lines coming into the terminal (or monitor).
- ◆ Check if you connect to the network (if not, check the network name or password)
- ◆ If necessary, copy everything that comes in and it analyzed by an expert.

Check the type of module

Follow the advice of [This Page](#) about the module types. In some cases, you must connect the Pin 0 with 3.3 volts through a 10k resistor.

The USB connection does not work and does not appear a COM port

Some computers may be missing the driver for the CH340 chip or for the chip CP2102. Read the instructions to install them on the next page.

If you can not communicate with the module

On some computers, it may either not be able to communicate with the lotModule via USB and serial (COM).

Symptoms

1. The operating system signal an error when you connect the USB cable.
2. By connecting the USB cable does not appear a new COM port.

The communication with the lotModule passes through a CH340 or a CP2102 chip. These chips convert the data from USB to serial and need a special driver, which in some computers may be missing. In these cases you can solve the problem by installing the driver manually.

Download and install the drivers for the chip CH340

This is the official download page of the manufacturer "Quin Heng Technology" for the driver for all types of CH340. It is valid for Win10/8.1/8/7/Vista/XP and Server 2016..2000/ME/98, both 32 and 64 bit, and certified by Microsoft Digital Signature:
http://www.wch.cn/download/CH341SER_EXE.html

- ◆ Open the page by clicking on the link above
- ◆ Press the blue button "DOWNLOAD"
- ◆ Wait for the download to complete
- ◆ Click the file CH341SER.EXE
- ◆ Agree to the changes
- ◆ Press "Install"

Download and install the driver for the chip CP2102

The driver for this chip is always pre-installed on Windows. However in the rare cases that the COM port does not appear, just go to this page and install it:

<https://www.silabs.com/products/development-tools/software/usb-to-uart-bridge-vcp-drivers>

- ◆ Open the page with a click on the link above
- ◆ Download "Download Windows 10 Universal" (by pressing "Download VCP")
- ◆ Unzip the zip in a folder of your choice
- ◆ Launch "CP010xVCPiInstaller_x86.exe" (or "_x64.exe" on 32bit systems)

The "IotModuleSetup.h" file (main options)

IOTMODULE_LOG_LEVEL ESP_LOG_NONE

Writing VERBOSE instead of NONE the module sends a lot of useful information to the serial. In some cases they can help you understand why the module does not connect to the WiFi network. To see the messages you can use a "Serial Terminal" type application.

TODO

IOTMODULE_PINOUT_TYPE 1

Set this option with "0" if you are using an ESP32-WROOM-32 module.

Set this option with "1" if you are using an ESP32-PICO-KIT V4 module.

Set this option with "2" if you are using an ESP32-TTGO-T7 v1.0 module.

Set this option with "3" if you are using an ESP32-TTGO-T7 v 1.3 module.

The images of the modules are in the first pages of this document.

IOTMODULE_INOUT_PINS 99

With this option you can set the number of pins that will be displayed in the IoT HAL application. To see all the Pin you set 28 for the PICO and TTGO modules, or 26 for the WROOM module. By setting a number larger than 28 (for example 99) you are sure to enable all the pins.

IOTMODULE_ADC24_PINS 0

With this option (from 0 to 16), you adjust the Pin number of the Adc24 module that will be displayed in the IoT HAL application. With "0" the Adc24 module is completely disabled.

IOTMODULE_GENERIC_PINS 0

With this option (from 0 to 20), you set the number of "generic" pins that will be displayed in the IoT HAL application. Normally this option is set to "0", because usually the unused physical Pins are enough to use them as placeholders for the "generic" Pins.

TODO

The "lotModuleSetup.h" file (Network config)

TODO ->

The main change introduced in version v.191 is the multi SSID configuration in lotModule.h.

This feature is very useful if you know in advance the access credential (SSID, password) of some Access Point you are going to connect, together with the type of IP connection (DHCP, static IP); you can prepare the configuration in advance, programming the firmware only once.

At every power up the module will search for a match between recognized and configured SSIDs, and it will continuously refresh the list of recognized SSIDs looking for a match. If a match is found and the connection is successful, the matched SSID will then be used as a reference: if the module temporarily loses wifi connectivity it will try to reconnect to the same SSID.

In order for the module to be able to connect to another network, its input power must be cycled off and on, or its reset button must be pressed.

This new release introduces some new parameters and some incompatibilities with parameters used before.

Multi word string parameters allow you to specify multiple items on one line; each item is separated from the other by one or more spaces.

As you will see in the examples, the multiword parameters form a sort of configuration organized in columns, where each column identifies a possible configuration.

See appendix **Network configuration** for a comprehensive parameters description

- > TODO

The "IotModuleSetup.h" file (secondary options)

Special configurations

TODO

IOTMODULE_ENCODERS_TYPE 1

Select "1" to use a "software" method to read the Encoders. With software encoders, the number of encoder steps is multiplied by four, whereas with hardware encoders the steps are multiplied by two. So software encoders are more accurate and are even more resistant to electrical noise. On the other hand the hardware ones support higher frequencies.

IOTMODULE_FUOTA_ENABLE 1

Selecting "1" enables the possibility of updating the firmware of the module via WiFi. Usually you leave it enabled, but those who fear that extraterrestrials will reprogram the module for their evil purposes, can set this parameter to zero. In this way no one will be able to reprogram the module, and you will have to go to the place each time to connect the module via USB.

IOTMODULE_ADC24_COMMUNICATION_SPEED 16

With this option you can set the connection speed with the Adc24 module from 1 MHz to 16 Mhz. Usually the maximum speed is used (16) and this guarantees the maximum performances for all the used Pin (it minimizes the noise on the "Period" type Pin and slightly improves some other function). Lowering the communication speed may be necessary if the connection wires between the ESP32 and the Adc24 module were long. With 1 MHz (and shielded cables) it could even go up to ten meters or more.

IOTMODULE_OUTPUT_WATCHDOG_ENABLE = 1

This line sets up a safety device which, in case of loss of communication, brings to zero all the pins configured as output (DigOut, DAC, PWM, SERVO). With ENABLE = 1 the zeroing occurs after one second, with ENABLE = 0 the zeroing never takes place.

The "IotModuleSetup.h" file (WiFi Mode Pin option)

IOTMODULE_WIFI_MODE_PIN = n

This feature can be useful in the test phase as it allows you to connect the IotModule to an access point, or to a PC without having to reprogram the firmware.

- ◆ If WIFI_MODE_PIN is -1 or is not defined, then the WiFi mode of operation of the IotModule, Station or SoftAP, it depends exclusively on the value of the variable IOTMODULE_WIFI_MODE described previously.
- ◆ If a **valid value** is assigned to the IOTMODULE_WIFI_MODE_PIN variable that identifies a Pin of the IotModule (36 for example), then the operating mode is decided exclusively by the state of the relevant physical Pin, when the IotModule is started or reset.

Valid pins are all physical pins (not Generic or Adc24 from 60 upwards). Furthermore, to be valid, the Pin must be enabled, i.e. visible in the IotHAL list, but it does not need to be configured.

If the Pin level is low, WiFi is initialized in Station mode, if the Pin level is high, WiFi is initialized in SoftAP mode.

Everything works easily and automatically in the DHCP context, but also with static IP configuration if the network configuration of the access point is compatible.

Warning: if the IotModule is started in SoftAP mode and the access point is also nearby, the PC will display two identical and hardly distinguishable SSIDs: one is that of the AP and the other of the IotModule.

The "IotModuleSetup.h" file (Stepper options)

IOTMODULE_STEPPER_TIMING 0

Step frequency intervals for the Stepper motors (mm/min are relative to 200 step/mm)

IOTMODULE_STEPPER_TIMING = 8 :	320...307200 Hz -	92160 mm/min
IOTMODULE_STEPPER_TIMING = 7 :	160...153600 Hz -	46080 mm/min
IOTMODULE_STEPPER_TIMING = 6 :	80...76800 Hz -	23040 mm/min
IOTMODULE_STEPPER_TIMING = 5 :	40...38400 Hz -	11520 mm/min
IOTMODULE_STEPPER_TIMING = 4 :	20...19200 Hz -	5760 mm/min
IOTMODULE_STEPPER_TIMING = 3 :	10...9600 Hz -	2880 mm/min
IOTMODULE_STEPPER_TIMING = 2 :	5...4800 Hz -	1440 mm/min
IOTMODULE_STEPPER_TIMING = 1 :	3...2400 Hz -	720 mm/min
IOTMODULE_STEPPER_TIMING = 0 :	2...1200 Hz -	360 mm/min

FUOTA - Firmware Update On The Air

Using a module that can be connected without wires (apart from the power supply that can also be locally generated near the module, for example, by means of a photovoltaic panel), it is possible that the final location is very difficult to reach physically.

If this fact can be an advantage from a side, the other one can reveal a big problem, if you were to update the firmware.

The FUOTA mode is to meet this problem, and is available on the Arduino IDE, as an alternative to USB programming.

To take advantage of this mode, the lotModule must already be programmed once (via USB), with the lotModule firmware, and should already be able to successfully connect to the WiFi network. That is the lotHAL Application must be able to show it in the connected modules list.

As far as possible the simultaneous recognition of several modules, the application Arduino IDE allows the programming of a single module at a time.

Caution

The module accepts the update via WiFi only if the constant "IOTMODULE_FUOTA_ENABLE" is set to 1 in the file "lotModuleSetup.h"

If you want to be sure to prevent the WiFi upgrade, set the constant at 0

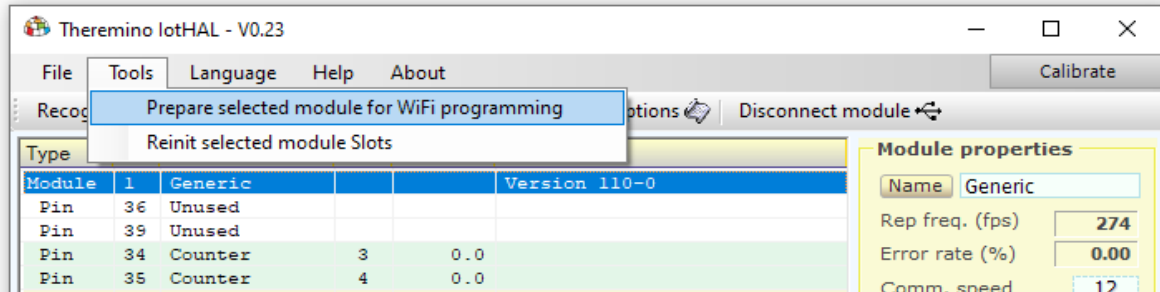
Before launching the FUOTA programming you must be sure that the new version of firmware will work properly.

If there are errors that compromise the access to the WiFi network, (for example the wrong net name or password) you can no longer communicate with the lotModule radio.

If you do something wrong you will be forced to go where is located the module, and connect it to a USB port.

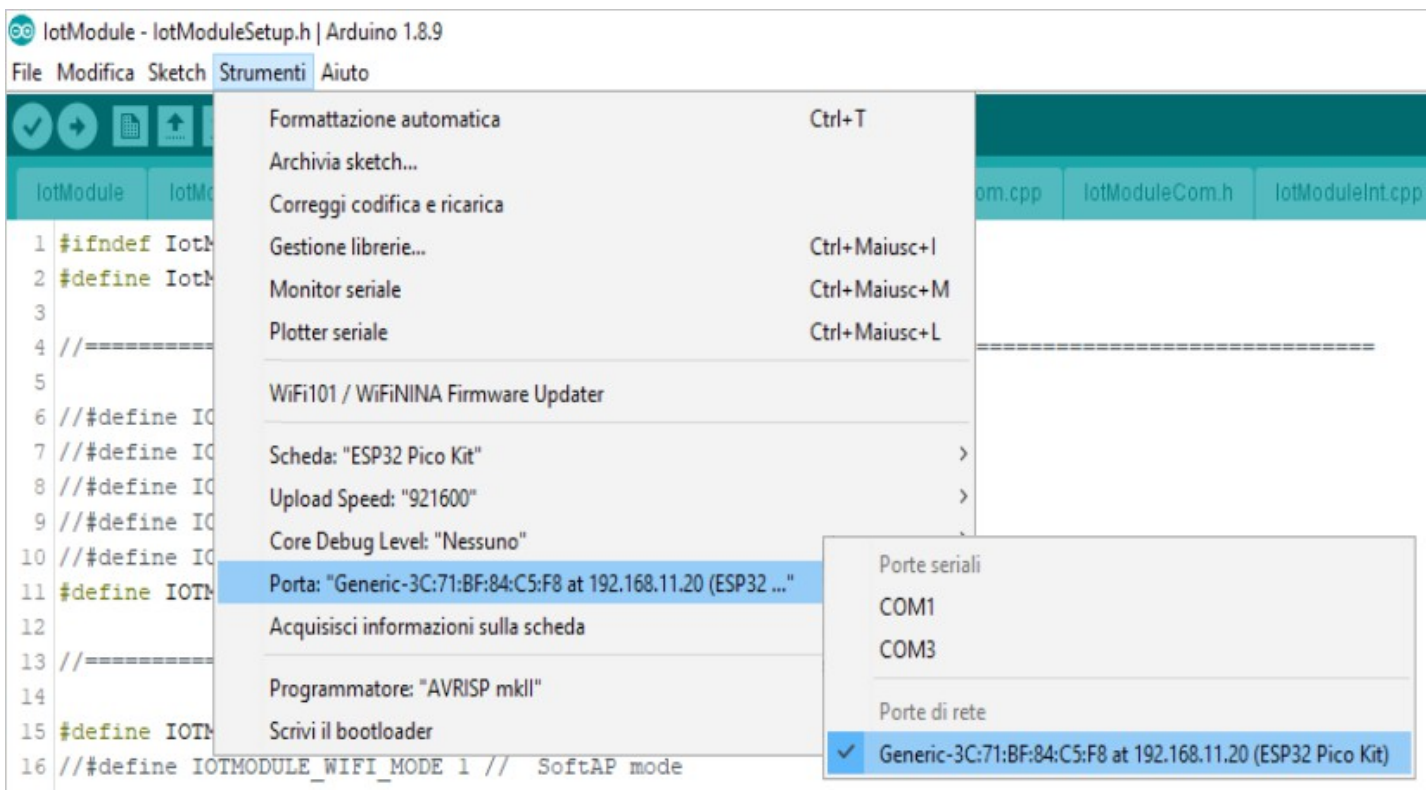
FUOTA - The update procedure

- ◆ Select, on the IoT HAL application, the IoTModule to update.
- ◆ Select the menu "Tools" / "Prepare selected module for WiFi programming"



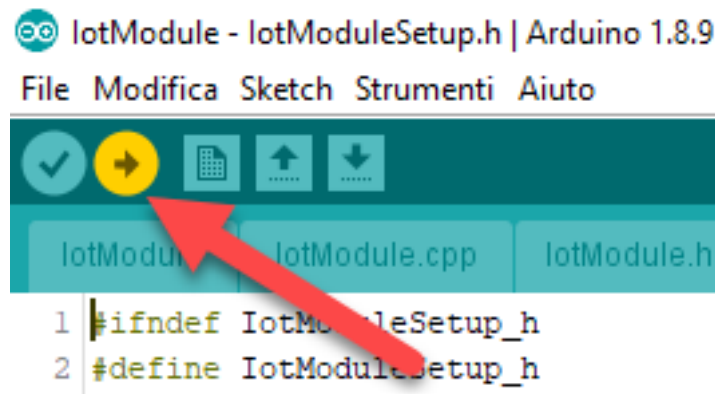
- ◆ The module at this point will disconnects from the IoT HAL and will remain visible to the Arduino IDE, between the **network ports**, waiting for the programming.
- ◆ After 120 seconds, the module will automatically reconnect with the IoT HAL, so that it can recover without going on site to reset it, in the case that the FUOTA request was made in error.
- ◆ Select the module, identified by the network port:

<Name> - <MAC> at <IP> (<card type>)

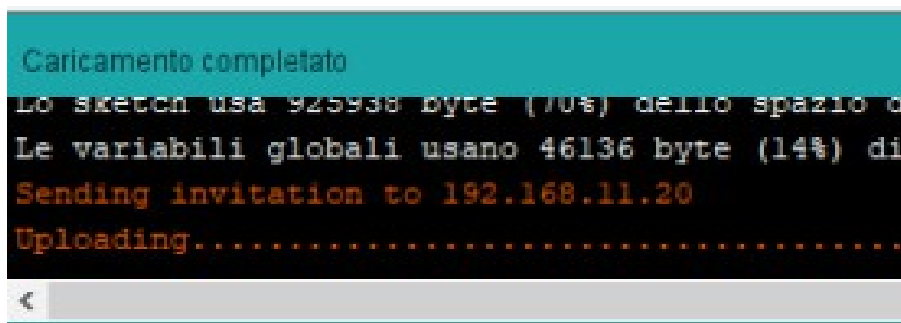


FUOTA - Send the update via WiFi

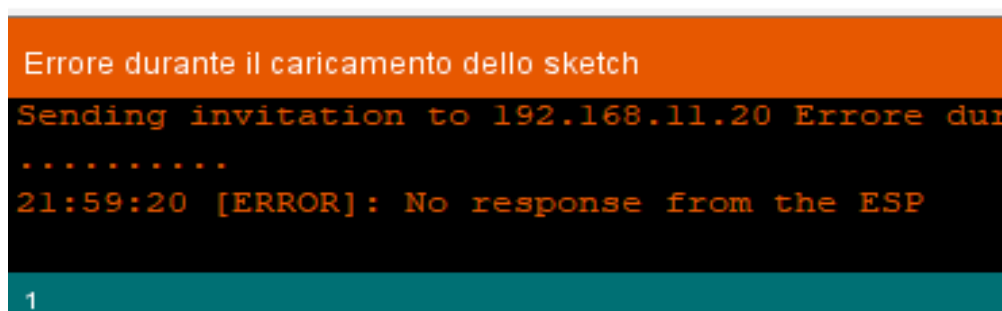
Press the standard execution key of the Arduino IDE, it will recompile the sketch and program the lotModule via Wifi



If successful you will read "Charging completed"



In case of error (lotModule not predisposed to FUOTA or other communication errors), you will read "An error occurred during the loading of the sketch"

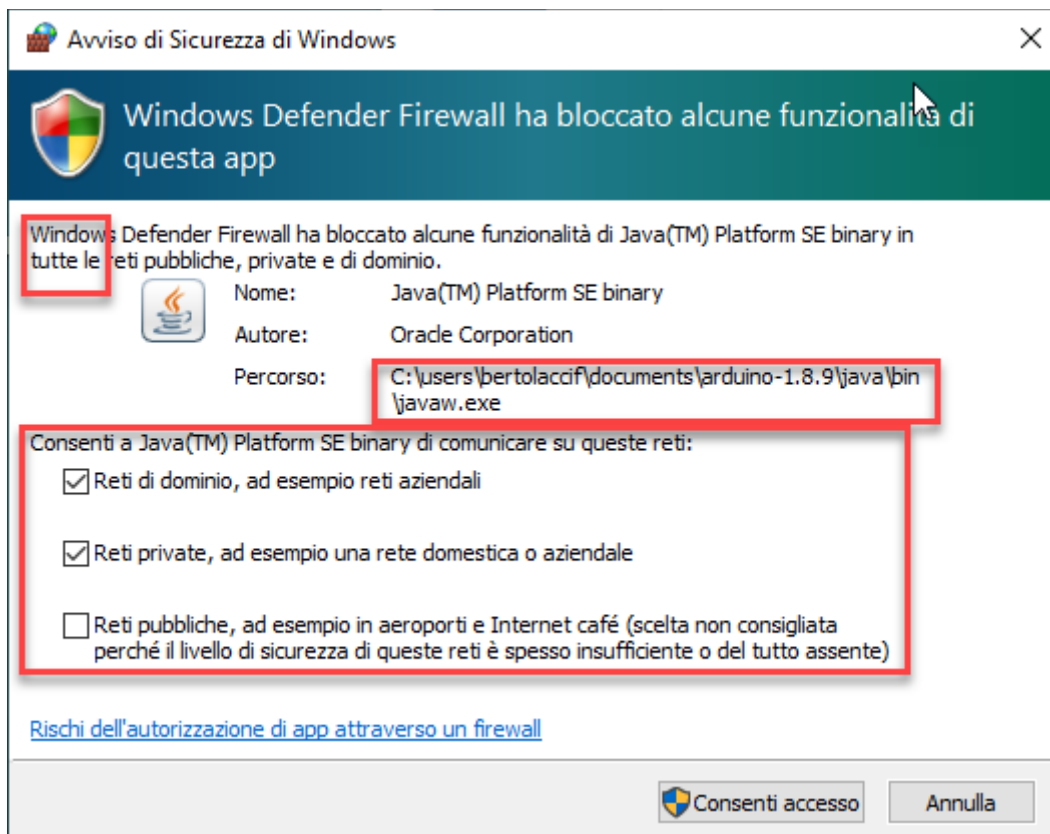


FUOTA - Network and Firewall

The Arduino IDE is able to connect to the IoTModule to execute the FUOTA procedure, only if the PC where the IDE is running, and the IoTModule, are on the same network.

That is, they must be on the same domain as Broadcast (it does not matter if the PC is connected with an Ethernet cable or via WiFi).

Furthermore, the Windows firewall rules relating to the Arduino IDE must have been activated. They are proposed at the first launch of the IDE in this way:



Connect lotHAL and lotModule on separate networks

The lotHAL application automatically recognizes all lotModules only if the PC where it is run and the lotModules are configured and connected to the same network, for example if their IP belongs to the 192.168.1.0/24 network (mask 255.255.255.0), and therefore 192.168.1.5, 192.168.1.6, 192.168.1.7 and so on.

With little effort it is also possible to connect lotHAL and lotModule on different networks. The only constraint is that the configuration of the lotModules, that do not belong to the same network as the PC, have been configured with static IP or with the equivalent reservation DHCP.

Let's assume that the PC where the lotHAL application is run is part of the "goofy" 192.168.1.0/24 network, and part of the lotModules are on the same network, and others on the "pluto" network 192.168.99.0/24, and the two networks are reachable at each other by routing.

The lotModules on the "goofy" network will be automatically recognized, while those placed on the "pluto" network, to be recognized, must be listed punctually in the CommOptions.txt file (See the examples in the file), using the WanIpInfo directive:

WanIpInfo <IP address lotModule> <UDP port lotModule> <timeout ms>

Valuing the parameters in this way:

WanIpInfo 192.168.99.5 49153 1000

WanIpInfo 192.168.99.6 49153 1000

WanIpInfo 192.168.99.7 49153 1000

Taking advantage of the described directive it is also possible to connect the lotModules through a VPN or, in an extreme case, directly via the internet; in the latter case the individual lotModules are exposed to risks of malfunctions due to the direct exposure of their UDP port on the internet.

Furthermore, competence in configuring the NAT and firewall rules of Internet-connected routers and participants in the connection is also required, this is beyond the scope of this documentation.

Here is an example of a configuration that allows you to connect three lotModules via the Internet, without VPN, connected directly to the network of a router and reached through point NATs on three different UDP ports:

WanIpInfo <public static ip of the router> 49153 1000

WanIpInfo <public static ip of the router> 49154 1000

WanIpInfo <public static ip of the router> 49155 1000

Replace <public static IP of the router> with the actual public IP assigned by the Internet provider.

Local network or Internet

Internet is more vulnerable than the WiFi and wired local network. To increase security on the Internet you could use a VPN. In any case, the greatest dangers are not from hackers, but gaps in speech, human error and hardware and software defects. So, as already written, never use our system to control dangerous or essential equipment.

Network configuration

Multi SSID configuration

IOTMODULE_STATIC_IP : deprecated, it must be removed from the configuration

IOTMODULE_WIFI_MODE : already used as a numeric value, now is a single word string

It defines the main connection mode of the module. May be STATION or SOFTAP

IOTMODULE_WIFI_ASSOC : new single word string

It defines the wifi mode associated with a single configuration "column". May be STATION or SOFTAP

IOTMODULE_IP_MODE : new multi word string

It defines the IP level configuration mode. May be DHCP or STATIC

IOTMODULE_IP_IP : new multi word string

It defines the IPv4 module address. See examples

IOTMODULE_IP_NETWORK : new multi word string

It defines the IPv4 network mask. See examples

IOTMODULE_IP_GATEWAY : new multi word string

It defines the IPv4 network gateway address. See examples

IOTMODULE_IP_DNS : new multi word string

It defines the IPv4 address of the DNS server. See examples

IOTMODULE_WIFI_SSID : already used as single word string, now is a multi word string

It defines the wireless local area networking name

IOTMODULE_WIFI_PASSWORD : already used as single word string, now is a multi word string

It defines the wireless local area networking password

IOTMODULE_SPACE_CHAR : new single character

It defines one single character for space substitution in SSID and password words

IOTMODULE_UDP_PORT_NUMBER 49153

The default UDP port number for the IoT HAL communication is 49153. To use a different port number un-comment and adapt the following line. And set the same port in the IoT HAL too.

IOTMODULE_WIFI_CHANNEL 1

If you set the module as SoftAP, then with this option you can choose the WiFi channel (from 1 to 13).

How to convert configuration parameters in lotModule.h file from v.182 to v.191

Example 1 (station, dhcp)

v.182:

```
#define IOTMODULE_WIFI_MODE 0
#define IOTMODULE_WIFI_SSID "ssid1"
#define IOTMODULE_WIFI_PASSWORD "pwd1"
```

v.191:

```
#define IOTMODULE_WIFI_MODE "STATION"
#define IOTMODULE_WIFI_ASSOC " STATION"
#define IOTMODULE_WIFI_SSID " ssid1 "
#define IOTMODULE_WIFI_PASSWORD " pwd1 "
#define IOTMODULE_IP_MODE " DHCP "
#define IOTMODULE_IP_IP " - "
#define IOTMODULE_IP_NETWORK " - "
#define IOTMODULE_IP_GATEWAY " - "
#define IOTMODULE_IP_DNS " - "
```

Example 2 (station, static)

v.182:

```
#define IOTMODULE_WIFI_MODE 0
#define IOTMODULE_WIFI_SSID "ssid1"
#define IOTMODULE_WIFI_PASSWORD "pwd1"
#define IOTMODULE_STATIC_IP "192.168.11.1 192.168.11.254 255.255.255.0 192.168.11.254"
                                     (module IP) (gateway IP) (net mask) (DNS IP)
```

v.191:

```
#define IOTMODULE_WIFI_MODE "STATION"
#define IOTMODULE_WIFI_ASSOC " STATION "
#define IOTMODULE_WIFI_SSID " ssid1 "
#define IOTMODULE_WIFI_PASSWORD " pwd1 "
#define IOTMODULE_IP_MODE " STATIC "
#define IOTMODULE_IP_IP " 192.168.11.1 "
#define IOTMODULE_IP_NETWORK "255.255.255.0 "
#define IOTMODULE_IP_GATEWAY "192.168.11.254 "
#define IOTMODULE_IP_DNS "192.168.11.254 "
```

Example 3 (softap)

v.182:

```
#define IOTMODULE_WIFI_MODE 1
#define IOTMODULE_WIFI_CHANNEL 1
#define IOTMODULE_WIFI_MODE_PIN -1
#define IOTMODULE_WIFI_SSID "ssid1"
#define IOTMODULE_WIFI_PASSWORD "pwd1"
```

v.191:

```
#define IOTMODULE_WIFI_MODE "SOFTAP"
#define IOTMODULE_WIFI_CHANNEL 1
```

```

#define IOTMODULE_WIFI_MODE_PIN -1
#define IOTMODULE_WIFI_ASSOC " SOFTAP"
#define IOTMODULE_WIFI_SSID " ssid1 "
#define IOTMODULE_WIFI_PASSWORD " pwd1 "
#define IOTMODULE_IP_MODE " DHCP "
#define IOTMODULE_IP_IP " - "
#define IOTMODULE_IP_NETWORK " - "
#define IOTMODULE_IP_GATEWAY " - "
#define IOTMODULE_IP_DNS " - "

```

Example 4 (station with optional softap, dhcp)

v.182:

```

#define IOTMODULE_WIFI_MODE 0
#define IOTMODULE_WIFI_CHANNEL 1
#define IOTMODULE_WIFI_MODE_PIN 33
#define IOTMODULE_WIFI_SSID "ssid1"
#define IOTMODULE_WIFI_PASSWORD "pwd1"

```

v.191:

Is now possible to have two distinct pairs of SSID/password.

```

#define IOTMODULE_WIFI_MODE "STATION"
#define IOTMODULE_WIFI_CHANNEL 1
#define IOTMODULE_WIFI_MODE_PIN 33
#define IOTMODULE_WIFI_ASSOC " STATION SOFTAP"
#define IOTMODULE_WIFI_SSID " ssid1 ssid2 "
#define IOTMODULE_WIFI_PASSWORD " pwd1 pwd2 "
#define IOTMODULE_IP_MODE " DHCP DHCP "
#define IOTMODULE_IP_IP " - - "
#define IOTMODULE_IP_NETWORK " - - "
#define IOTMODULE_IP_GATEWAY " - - "
#define IOTMODULE_IP_DNS " - - "

```

New possible configurations available with v.191

Example 5 (multi station, dhcp)

```
#define IOTMODULE_WIFI_MODE "STATION"
#define IOTMODULE_WIFI_ASSOC " STATION STATION STATION STATION STATION STATION"
#define IOTMODULE_WIFI_SSID " ssid1 ssid2 ssid3 ssid4 ssid5 ssid6 "
#define IOTMODULE_WIFI_PASSWORD " pwd1 pwd2 pwd3 pwd4 pwd5 pwd6 "
#define IOTMODULE_IP_MODE " DHCP DHCP DHCP DHCP DHCP DHCP "
#define IOTMODULE_IP_IP " - - - - - "
#define IOTMODULE_IP_NETWORK " - - - - - "
#define IOTMODULE_IP_GATEWAY " - - - - - "
#define IOTMODULE_IP_DNS " - - - - - "
```

Example 6 (multi station, mixed dhcp + static)

```
#define IOTMODULE_WIFI_MODE "STATION"
#define IOTMODULE_WIFI_ASSOC " STATION STATION STATION STATION STATION STATION STATION STATION STATION "
#define IOTMODULE_WIFI_SSID " ssid1 ssid2 ssid3 ssid4 ssid5 ssid6 ssid7 ssid8 "
#define IOTMODULE_WIFI_PASSWORD " pwd1 pwd2 pwd3 pwd4 pwd5 pwd6 pwd7 pwd8 "
#define IOTMODULE_IP_MODE " DHCP DHCP DHCP DHCP DHCP DHCP STATIC STATIC "
#define IOTMODULE_IP_IP " - - - - - 192.168.11.1 172.20.1.1 "
#define IOTMODULE_IP_NETWORK " - - - - - 255.255.255.0 255.255.0.0 "
#define IOTMODULE_IP_GATEWAY " - - - - - 192.168.11.254 172.20.1.254 "
#define IOTMODULE_IP_DNS " - - - - - 192.168.11.254 172.20.1.254 "
```

Example 7 (multi station, mixed dhcp + static, optional softap)

A low level on pin 33 at boot selects the station modes, a high level selects the SofAP mode.

```
#define IOTMODULE_WIFI_MODE "STATION"
#define IOTMODULE_WIFI_CHANNEL 1
#define IOTMODULE_WIFI_MODE_PIN 33
#define IOTMODULE_WIFI_ASSOC " STATION STATION STATION STATION STATION STATION STATION STATION STATION SOFTAP"
#define IOTMODULE_WIFI_SSID " ssid1 ssid2 ssid3 ssid4 ssid5 ssid6 ssid7 ssid8 ssid9 "
#define IOTMODULE_WIFI_PASSWORD " pwd1 pwd2 pwd3 pwd4 pwd5 pwd6 pwd7 pwd8 pwd9 "
#define IOTMODULE_IP_MODE " DHCP DHCP DHCP DHCP DHCP DHCP STATIC STATIC DHCP "
#define IOTMODULE_IP_IP " - - - - - 192.168.11.1 172.20.1.1 - "
#define IOTMODULE_IP_NETWORK " - - - - - 255.255.255.0 255.255.0.0 - "
#define IOTMODULE_IP_GATEWAY " - - - - - 192.168.11.254 172.20.1.254 - "
#define IOTMODULE_IP_DNS " - - - - - 192.168.11.254 172.20.1.254 - "
```

Example 8 (space substitution)

Spaces characters are already used as separators between words in multi word strings; if you need to insert one or more spaces in the SSID and/or password, you must identify one character not used yet in the credential pair.

This particular character must be specified in the dedicated directive and used in place of spaces in configuration words; these characters will be automatically replaced by spaces by the firmware during the configuration analysis, as you can see in this example.

```
#define IOTMODULE_WIFI_MODE "STATION"
#define IOTMODULE_SPACE_CHAR '&'
#define IOTMODULE_WIFI_ASSOC " STATION "
#define IOTMODULE_WIFI_SSID " SSID&WITH&spaces "
#define IOTMODULE_WIFI_PASSWORD " password&with&SPACES "
#define IOTMODULE_IP_MODE " DHCP "
#define IOTMODULE_IP_IP " - "
#define IOTMODULE_IP_NETWORK " - "
#define IOTMODULE_IP_GATEWAY " - "
#define IOTMODULE_IP_DNS " - "
```

Please note the use of single quotes in character definition.

In this case the effective SSID and password words will be internally converted from:

"SSID&WITH&spaces" and "password&with&SPACES"

to:

"SSID WITH spaces" and "password with SPACES"

Example 9 (how to avoid wrong configurations)

In order to avoid wrong configurations, in the columns where the DHCP mode is used, it is mandatory to put a placeholder (any character or word you like) as a fake value of unneeded configuration values (IP, NETWORK, GATEWAY and DNS); in the following example the four placeholders needed (red area, last 5 columns) have been omitted. This is wrong, after the configuration analysis phase during boot, only the first column with STATIC IP configuration will be taken in charge by the firmware.

```
#define IOTMODULE_WIFI_MODE "STATION"
#define IOTMODULE_WIFI_ASSOC " STATION STATION STATION STATION STATION"
#define IOTMODULE_WIFI_SSID " ssid1 ssid2 ssid3 ssid4 ssid5 ssid6 "
#define IOTMODULE_WIFI_PASSWORD " pwd1 pwd2 pwd3 pwd4 pwd5 pwd6 "
#define IOTMODULE_IP_MODE " STATIC DHCP DHCP DHCP DHCP "
#define IOTMODULE_IP_IP " 192.168.11.1 "
#define IOTMODULE_IP_NETWORK " 255.255.255.0 "
#define IOTMODULE_IP_GATEWAY " 192.168.11.254 "
#define IOTMODULE_IP_DNS " 192.168.11.254 "
```

This is an example of right configuration, the character "-" has been used as the value placeholder:

```
#define IOTMODULE_WIFI_MODE "STATION"
#define IOTMODULE_WIFI_ASSOC " STATION STATION STATION STATION STATION"
#define IOTMODULE_WIFI_SSID " ssid1 ssid2 ssid3 ssid4 ssid5 ssid6 "
#define IOTMODULE_WIFI_PASSWORD " pwd1 pwd2 pwd3 pwd4 pwd5 pwd6 "
#define IOTMODULE_IP_MODE " STATIC DHCP DHCP DHCP DHCP "
#define IOTMODULE_IP_IP " 192.168.11.1 - - - - "
#define IOTMODULE_IP_NETWORK " 255.255.255.0 - - - - "
#define IOTMODULE_IP_GATEWAY " 192.168.11.254 - - - - "
#define IOTMODULE_IP_DNS " 192.168.11.254 - - - - "
```

Other improvements

- The keywords STATION, SOFTAP, DHCP, STATIC are case insensitive, may be used all uppercase, lowercase or in a mix.
- SoftAP IP configuration mode is available exclusively in DHCP; STATIC mode has been removed.
- Debug log messages, optionally configurable and available in the serial monitor window in Arduino IDE, have been improved to reflect available and configured SSID
- New pinout for [ESP32-TTGO T7 v1.0](#) and [ESP32-TTGO T7 v1.3](#) boards.
 - The pinout type 3 has been introduced and it must be used with ESP32-TTGO T7 v1.3 module because pin 18 and 19 are crossed from ESP32-TTGO T7 v1.0 module.
- Correction on GPIO 36, 37 and 39
 - GPIO 36 and 39 were not handled correctly and they changed status together.
 - Corrected the possible pin types of GPIO 37
- Compatibility with Arduino ESP32 library v 1.0.5
 - Arduino ESP32 library v.1.0.5, based on ESP-IDF v3.3, available from 2021/02/23, introduces a change to a data type definition that prevents the sketch to compile
- PINOUT revision of ESP32-TTGO T7 v1.3
 - GPIO7 and GPIO11 have been removed, not usable since already connected to external flash chip, replaced by GPIO16 and GPIO17
- New parameter for TX wifi power : IOTMODULE_WIFI_TX_POWER_INDEX
 - Possible values are:
 - 10 = 19.5 dBm (default, used by previous firmware revision)
 - 9 = 19 dBm
 - 8 = 18.5 dBm
 - 7 = 17 dBm
 - 6 = 15 dBm
 - 5 = 13 dBm
 - 4 = 11 dBm
 - 3 = 8.5 dBm
 - 2 = 7 dBm
 - 1 = 5 dBm
 - 0 = 2 dBm
 - For example, to set the max TX power to 13 dBm:
`#define IOTMODULE_WIFI_TX_POWER_INDEX 5`
- Optimization of the wifi connection speed if only one configuration column is defined
 - The wifi network scan operation in the case of multi SSID configuration introduces a small delay in the connection. If the multi SSID configuration is not required, i.e. if there is only one column for the SSID parameters in the configuration, the change introduced cancels this delay speeding up the connection to the limit.

Compatibility with Arduino IDE

Version v.191 firmware compilation and FUOTA feature have been tested successfully with Arduino IDE v.1.8.10 and v.1.8.19; they have not been tested yet with Arduino IDE v.2.0.