Sistema theremino

Theremino IotHAL
Leggere i sensori I2C

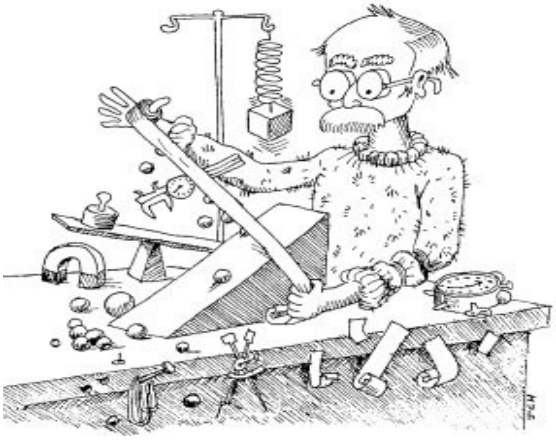
Leggere sensori collegati in I2C

Per utilizzare moduli ESP32 con il nostro sistema, non è necessario saper programmare.

Tutti i sensori semplici, come pulsanti, pulsanti capacitivi, encoders, servo motori,
e sensori analogici di ogni genere, sono collegabili direttamente allo ESP32,
senza nessun firmware aggiuntivo.

Tutti i tipi di InOut più comuni sono già pronti e basta sceglierli con la applicazione IotHal.

Quello che spiegheremo qui di seguito serve solo
per collegare sensori digitali e per effettuare elaborazioni speciali.



In questo documento mostreremo come leggere sensori connessi con il protocollo I2C, e come effettuare elaborazioni nel modulo stesso.

Questi esempi illustrano le tecniche fondamentali per comunicare con i sensori speciali, e per inviare i loro dati al sistema theremino. Queste tecniche possono servire anche come traccia per connettere altri tipi di sensori, ad esempio i sensori SPI.

- - - - - -

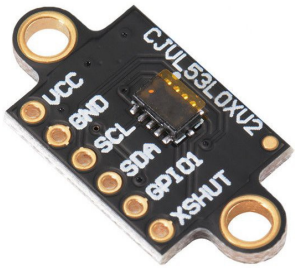
In ***questa pagina*** troverete la applicazione IotHAL e il file "IotModule_I2C_Sensors.zip", che contiene i progetti di Arduino completi, per tutti i sensori descritti in questo documento.

Per programmare lo ESP32, ricordatevi di impostare il nome e la password della vostra rete e di preparare l'IDE di Arduino, come spiegato nella documentazione della applicazione IotHAL.

- - - - - -

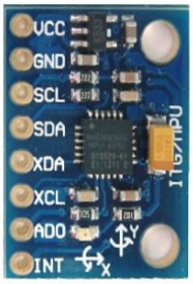
In tutti gli esempi, i dati vengono inviati dal modulo ESP32 alla ***applicazione IotHAL*** per mezzo dei tipi di Pin “Generic” e poi inviati agli Slot, tramite i quali qualunque altra applicazione del sistema theremino potrà utilizzarli. Ad esempio la ***applicazione ECG*** per la ricerca delle aritmie.

Sommario degli esempi

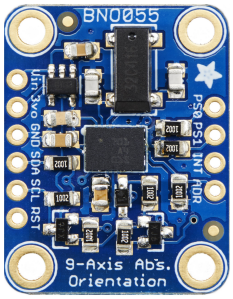


ESEMPIO 1 - Un sensore Laser che misura la distanza fino a due metri, con risoluzione di un millimetro e precisione migliore di cinque millimetri.

Questo esempio è molto semplice. Il firmware preparato da noi è facile da capire e funziona subito, senza bisogno di regolazioni.

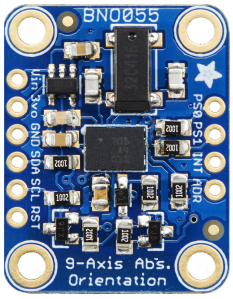


ESEMPIO 2 - Un accelerometro a tre assi (x, y, z) che può essere usato per la rilevazione dei terremoti. Questo sensore è economico (2 o 3 Euro) e facile da usare, ma abbastanza rumoroso, per cui si potranno rilevare solo terremoti vicini e abbastanza forti.



ESEMPIO 3 - Un accelerometro a tre assi (x, y, z), che può essere usato per la rilevazione dei terremoti. In questo esempio usiamo un sensore di precisione il BNO055 (accelerometro, giroscopio e bussola) e lo programmiamo per essere un semplice accelerometro. Questo sensore costa dieci volte di più, ma è meno rumoroso del precedente.

Con questo sensore si possono rilevare anche terremoti di media intensità.



ESEMPIO 4 - Il BNO055 contiene un accelerometro, un giroscopio e una bussola, ognuno con tre assi (x, y, z). Il firmware unisce i nove assi e produce un orientamento assoluto su tre assi (che viene anche chiamato "piattaforma inerziale").

Questo sensore è abbastanza complesso, dovrete leggere molto per imparare a calibrarlo e usarlo. Le istruzioni sono nelle prime righe del firmware stesso (nel file .ino).

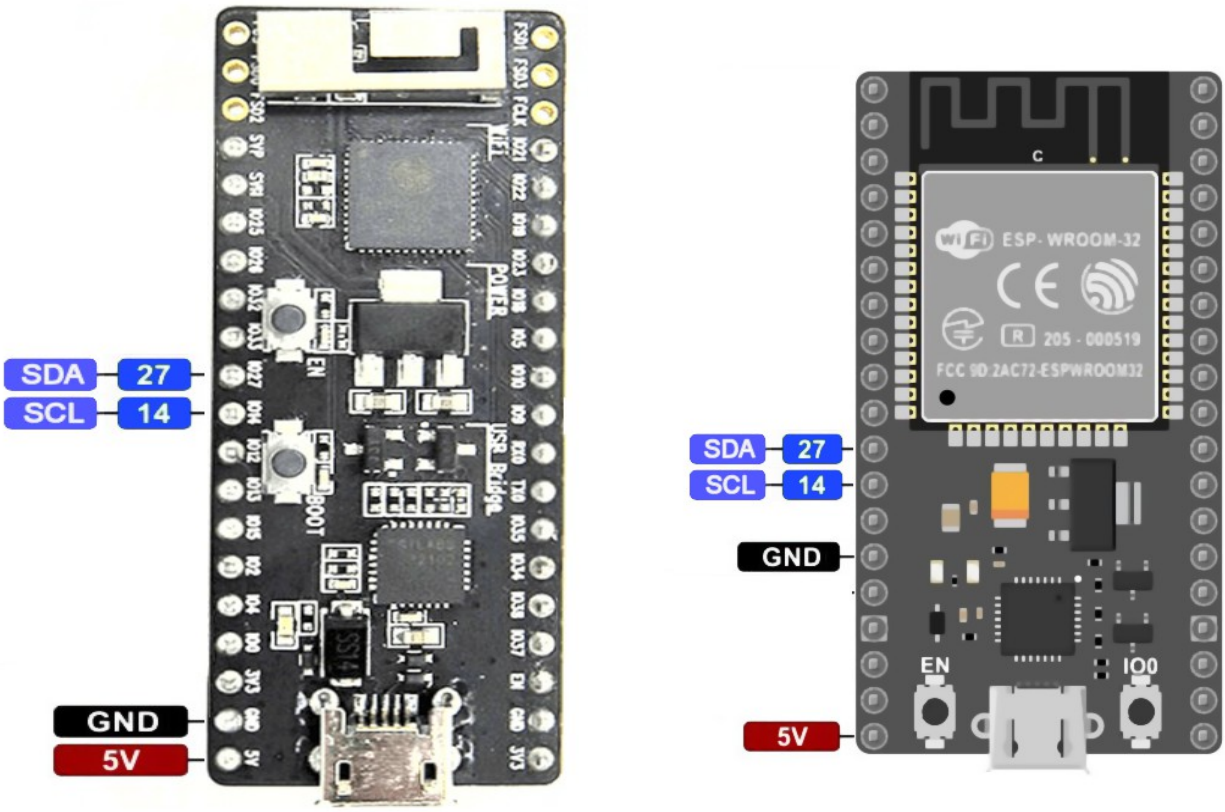
ESEMPIO 5 - Un sensore per misurare la frequenza cardiaca.

Questo esempio spiega anche le tecniche usate nel firmware e può servire per imparare a scrivere progetti simili.

Connessioni ai moduli ESP32

Per connettere tutti i sensori di questi esempi, usiamo sempre gli stessi quattro pin.

Ecco la loro posizione sulle schede più comunemente usate.



Le immagini complete, con i nomi di tutti i Pin e le annotazioni per i Pin speciali, sono nella documentazione della applicazione IotHAL, che si scarica da ***questa pagina***

- - - - -

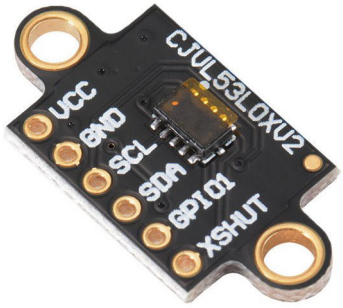
Una volta letti i dati dai sensori, per inviarli dal firmware verso la applicazione IotHAL, utilizziamo i Pin 36, 39 e 25 (oppure 34 che sul modulo wroom è più comodo). Questi "Pin" non corrispondono ai Pin fisici del modulo, ma sono solo degli identificatori che utilizziamo per le funzioni GenericWrite e GenericRead, che comunicano i dati tra il firmware e la applicazione IotHAL.

Al posto di 36, 39, 25 (o 34), avremmo potuto usare qualunque altro Pin inutilizzato, oppure potremmo programmare nuovi Pin "Generici", a partire dall'80 in poi, come spiegato nelle ultime pagine della documentazione della applicazione IotHAL.

Esempio 1 - Un sensore che misura la distanza

Questo sensore misura la distanza per mezzo di un raggio Laser ***(Nota 1)***.

Questo sensore può riconoscere con precisione la posizione di una mano ed ha una risposta veloce. Per cui può anche sostituire i CapSensors, negli strumenti musicali di tipo Theremin.



Caratteristiche:

- Distanza di misura fino a oltre un metro.
- Misura tramite il “Tempo di volo” della luce Laser.
- Risoluzione di un millimetro.
- Precisione di circa 5 mm
- Consumo 20 mA

(Nota 1) *I laser sono pericolosi solo perché il loro fascio è concentrato. Questo sensore pur utilizzando luce Laser ha una apertura di 35 gradi, quindi molto simile a quella di un normale LED. Ed ha anche una potenza di emissione molto simile a quella di un LED. Per cui la luce emessa è la stessa che verrebbe emessa da un LED infrarosso come quelli dei telecomandi dei televisori. L'unica differenza è che si tratta di luce coerente, cioè con un unica frequenza (o quasi). Ma la coerenza non genera pericolosità per cui possiamo considerare questo dispositivo non più pericoloso di un LED a infrarossi.*

Evitate comunque di avvicinare un occhio entro pochi centimetri dal punto di uscita, non perché si tratti di un Laser ma perché è luce infrarossa e quindi non visibile. Se lo si punta in un occhio da vicino e per lungo tempo anche un LED infrarosso può essere dannoso per la vista. Il sole è infinitamente più pericoloso, ma la sua luce è visibile e difficilmente lo si guarderebbe fisso a lungo.

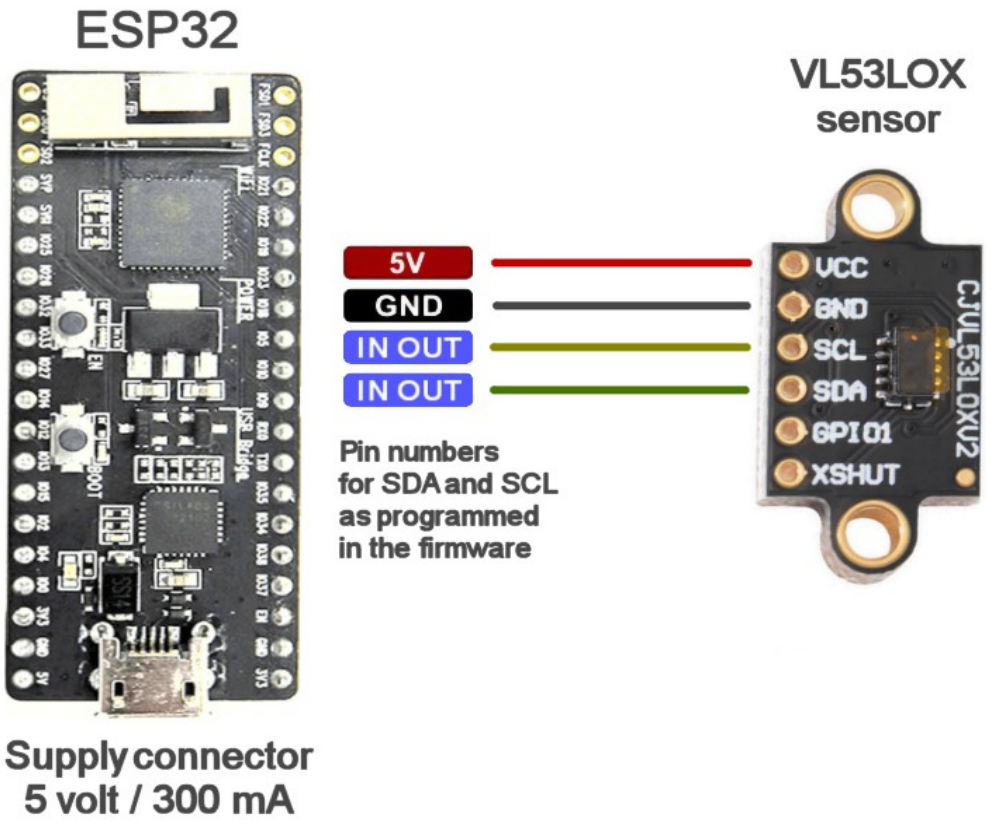
Nel datasheet questo Laser è definito come “Classe 1”. Ed è anche specificato che è studiato per rimanere in Classe 1, in tutte le condizioni, includendo i guasti.

Secondo Wikipedia: “Un laser di Classe 1 è sicuro in tutte le condizioni di utilizzo normale. Ciò significa che l'esposizione massima consentita (MPE) non può essere superata quando si guarda il laser ad occhio nudo o con l'ausilio di ottiche di ingrandimento tipiche (ad es. Telescopio o microscopio).”

Esempio 1 - Collegamenti

I fili da collegare allo ESP32 sono quattro VCC (5V), GND, SCL e SDA.

Quindi prendiamo quattro fili (piccoli e flessibili) e li colleghiamo facendo molta attenzione. Più di tutto si deve stare attenti a non invertire VCC e GND.



Per tutti gli esempi abbiamo scelto i Pin:

- 27 per il segnale SDA
- 14 per il segnale SCL

Per individuare la posizione dei Pin sui moduli ESP32,
consultare le immagini all'inizio di questo documento (pagina 4)

Esempio 1 - Leggere i dati con la applicazione IotHAL

Per leggere il sensore VL53LOX:

- Si prepara l'IDE di Arduino, come spiegato nella documentazione della applicazione IotHAL.
- Si programma lo ESP32 con lo sketch "IotModule_DistanceMeter.ino".
- Si lancia la Applicazione “IotHal”

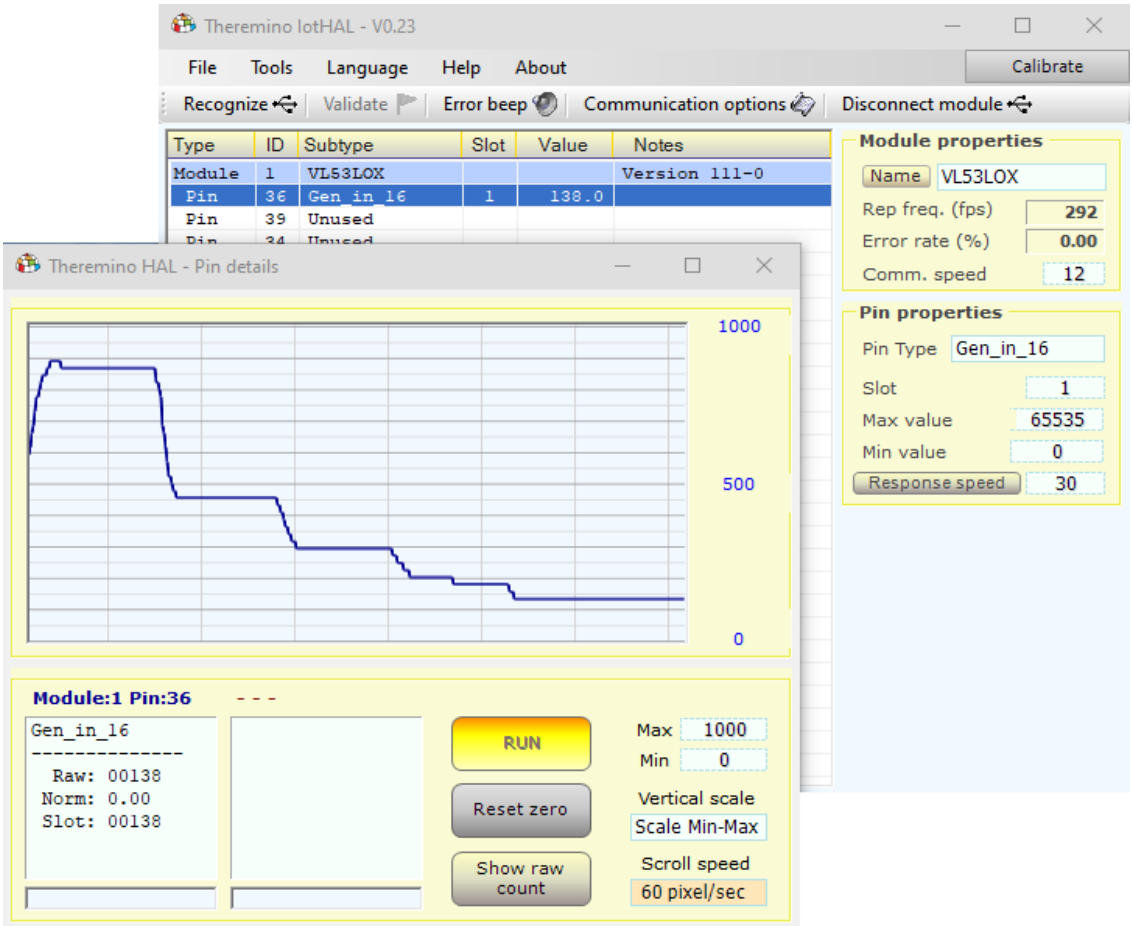
Siccome nella riga “genericWrite” abbiamo impostato il Pin “36” anche nella applicazione IotHAL dobbiamo configurare questo Pin come ingresso generico, per cui: PinType del Pin 36 = Gen_in_16

Ricordarsi anche di impostare “Velocità risposta” a 30 su questo Pin, in modo da filtrare il rumore e ottenere una misura più stabile.

Con questo sensore conviene ottenere un valore in millimetri e non da 0 a 1000 che vengono messi di default quando si configurano i Pin. Per cui imposteremo Max value = 65535 e otterremo il numero in millimetri.

Il 65536 deriva dal fatto che utilizziamo un ingresso Gen_in_16, cioè 16 bit, che valgono 65536. Quindi impostando un Min = 0 e un Max = 65535 riotteniamo esattamente il valore grezzo che ci viene inviato dal sensore.

E in questo caso il valore grezzo del sensore sono proprio i millimetri da 0 a 2000.



Esempio 2 - Un accelerometro a tre assi

In questo esempio utilizziamo un accelerometro a tre assi (x, y, z) che può essere usato per la rilevazione dei terremoti.

Caratteristiche:

- Questo sensore è economico (2 o 3 Euro)
- E' un sensore facile da programmare, il firmware per leggere i dati è molto semplice, una decina di righe in tutto.
- Misurazione su tre assi con fondo scala di +/- 2G
- La risoluzione di misura è di 16 bit ma i dati contengono molto rumore, per cui si possono rilevare solo terremoti vicini e abbastanza forti.

- - - - - -

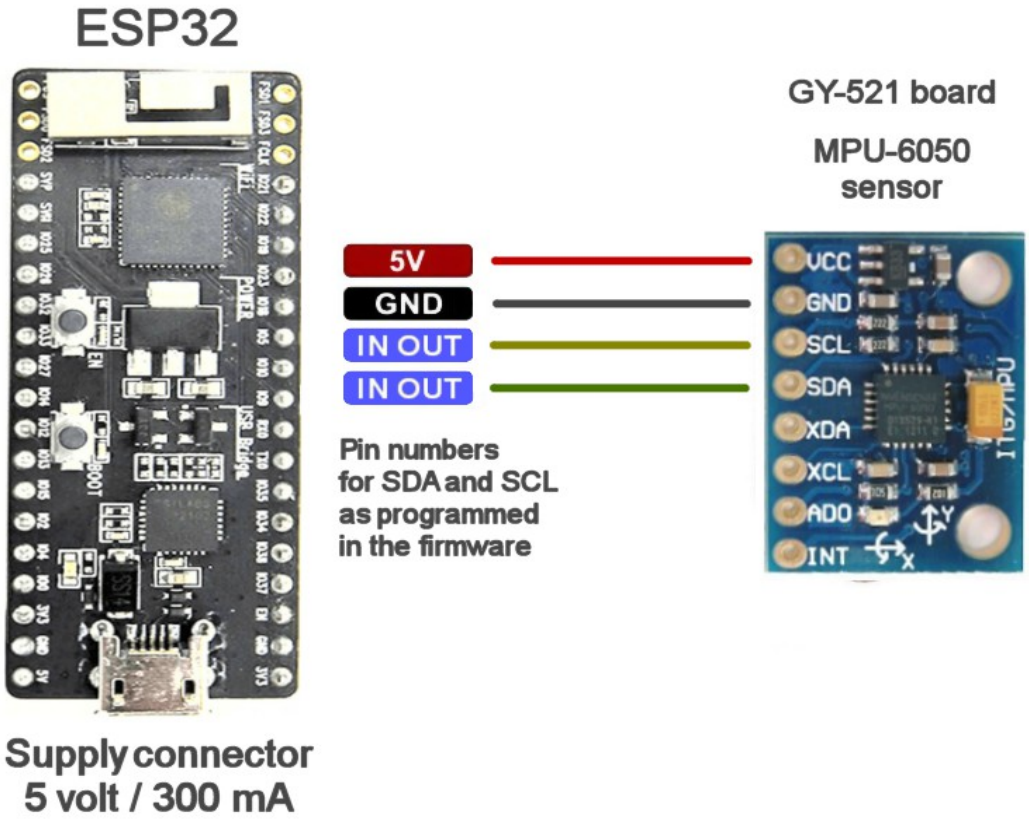
Per i terremoti si consiglia di usare il sensore "Esempio 3",

che è notevolmente meno rumoroso.

Esempio 2 - Collegamenti

I fili da collegare allo ESP32 sono quattro VCC (5V), GND, SCL e SDA.

Quindi prendiamo quattro fili (piccoli e flessibili) e li colleghiamo facendo molta attenzione. Più di tutto si deve stare attenti a non invertire VCC e GND.



Per tutti gli esempi abbiamo scelto i Pin:

- 27 per il segnale SDA
- 14 per il segnale SCL

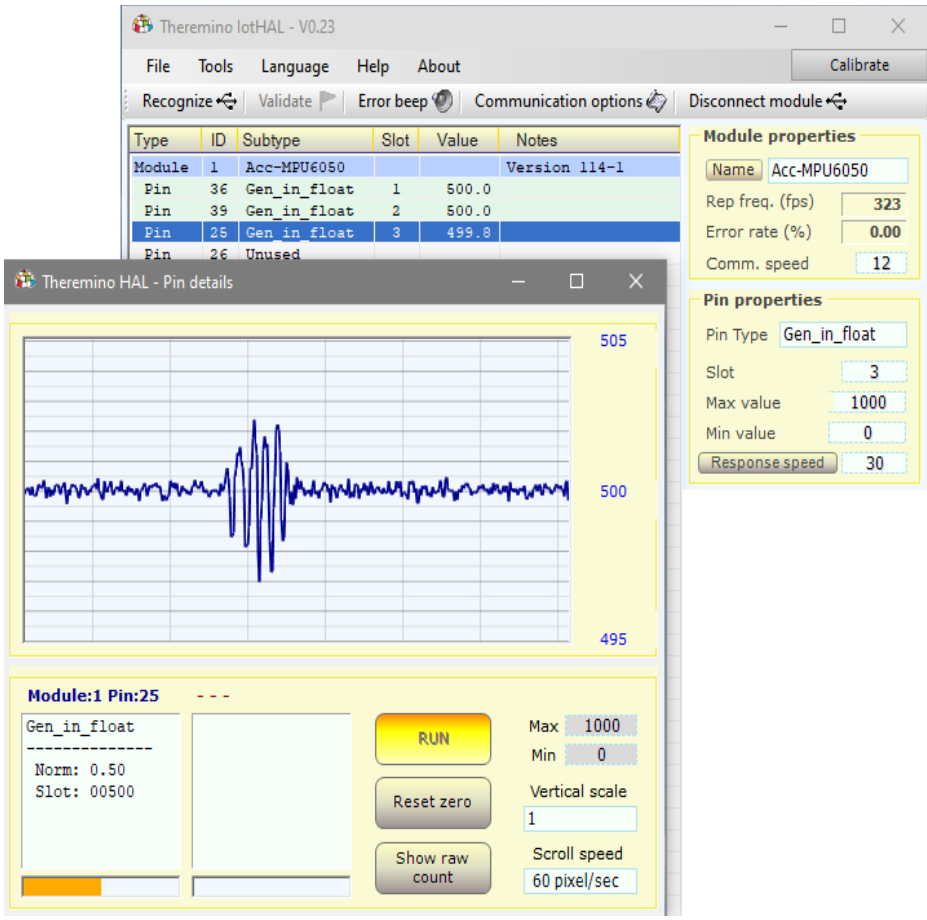
Per individuare la posizione dei Pin sui moduli ESP32,
consultare le immagini all'inizio di questo documento (pagina 4)

Esempio 2 - Leggere i dati con la applicazione IotHAL

Per leggere il sensore MPU-6050:

- Si prepara l'IDE di Arduino, come spiegato nella documentazione della applicazione IotHAL.
- Si programma lo ESP32 con lo sketch "IotModule_AccMPU6050.ino".
- Si lancia la Applicazione “IotHal”

Si configurano i primi tre Pin come: "Gen_in_float" (con i float i valori non sono limitati nell'intervallo da 0 a 1000, ma possono anche essere minori di zero o maggiori di 1000, con movimenti di grande intensità)



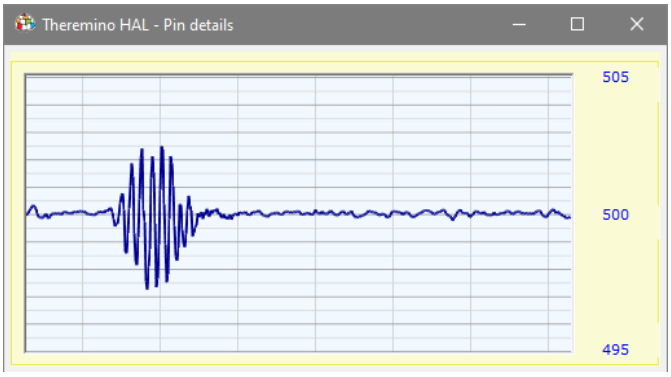
Il firmware invia valori da 0 a 1000 (centrati su 500), ma è possibile modificare Min e Max nell'HAL e ottenere inviare valori da -1 a +1.

Regolando nell'HAL Min=-1 e Max=1 si ottengono misure tarate in G.

```
+2.0 = +2G
+1.0 = +1G
 0.0 =  0G
-1.0 = -1G
-2.0 = -2G
```

La prima immagine è un esempio di come potrebbe apparire il grafico con un terremoto abbastanza forte e vicino.

Nella seconda immagine si vede una oscillazione di intensità simile alla precedente, ma misurata con il sensore della prossima pagina.



Come si può vedere la sensibilità è maggiore e il rumore è minore.

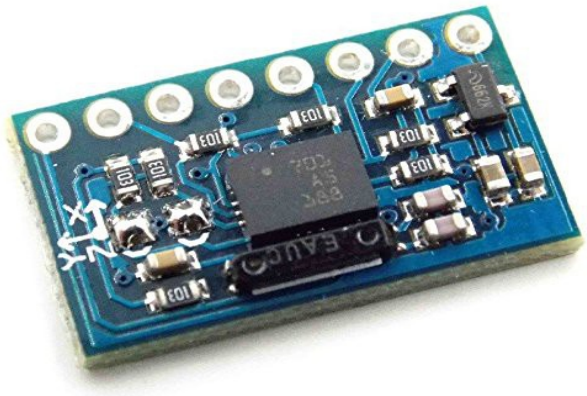
Il firmware trasforma i dati di questo sensore da "accelerometro" a "velocimetro", in modo da renderlo similie a un classico geofono da 4.5 Hz. Vedere i particolari nell'ultima pagina di questo documento.

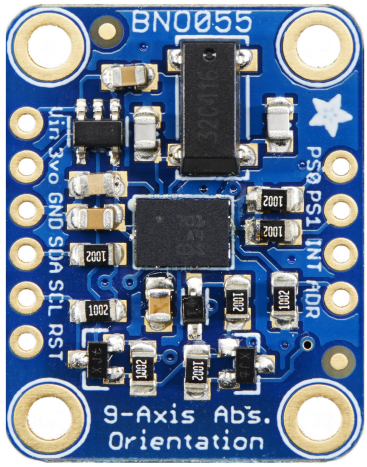
Esempio 3 - Accelerometro a tre assi (migliore)

In questo esempio utilizziamo il BNO055, lo stesso modulo che illustreremo con maggiori dettagli nel prossimo esempio.

Per usare questo modulo come accelerometro lo abbiamo programmato in modo speciale, sprecando parte delle sue caratteristiche, ma ottenendo un accelerometro notevolmente migliore di quello dell'esempio precedente.

Con questo sensore si possono rilevare terremoti più deboli e più lontani. Il BNO055 costa un po' di più dello MPU6050 (da dieci a trenta Euro al posto che due o tre Euro) ma il miglioramento che si ottiene vale la spesa aggiuntiva e anche la attesa, nel caso lo si facesse arrivare dalla Cina.





Per usare il sensore BNO055 come accelerometro, basta programmarlo con il firmware "IotModule_AccBNO055". L'unica differenza è che i dati ricevuti non sono rotazioni nello spazio, ma sono le accelerazioni sui tre assi X/Y/Z (con fondo scala +/-2G)

- - - - -

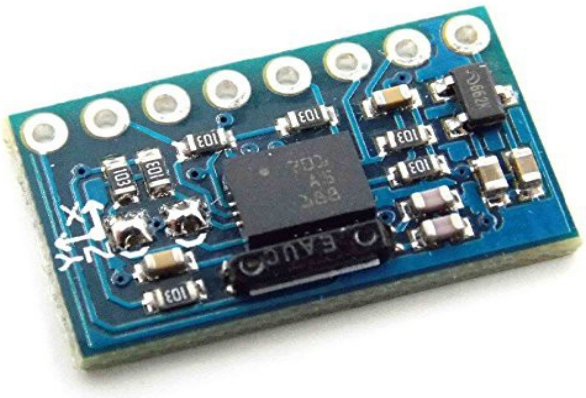
Per collegare il sensore, consultare le istruzioni dell'esempio seguente (Esempio 4)

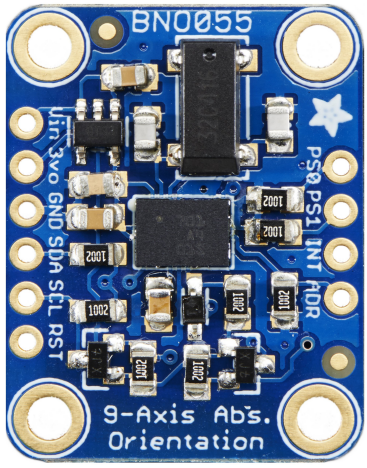
Per ricevere i dati con IotHAL consultare le istruzioni dell'esempio precedente (Esempio 2)

Il firmware trasforma i dati di questo sensore da "accelerometro" a "velocimetro",
in modo da renderlo simile a un classico geofono da 4.5 Hz.
Vedere i particolari nell'ultima pagina di questo documento.

Esempio 4 - Accelerometro, Giroscopio e Bussola

In questo secondo esempio utilizziamo il BNO055, un sensore intelligente che fa la fusione dei nove assi direttamente nel chip. Con altri sensori trasformare i dati provenienti da accelerometro, giroscopio e magnetometro, in un vero e proprio "Orientamento spaziale 3D", richiederebbe di scrivere algoritmi complessi, difficili da testare e da regolare.





Esistono vari modelli di questo modulo. Quello a sinistra si trova su eBay, spedito dalla Cina, per meno di dieci Euro, spedizione compresa. Quello a destra, prodotto e commercializzato da SparkFun, costa più di trenta Euro.

Caratteristiche:

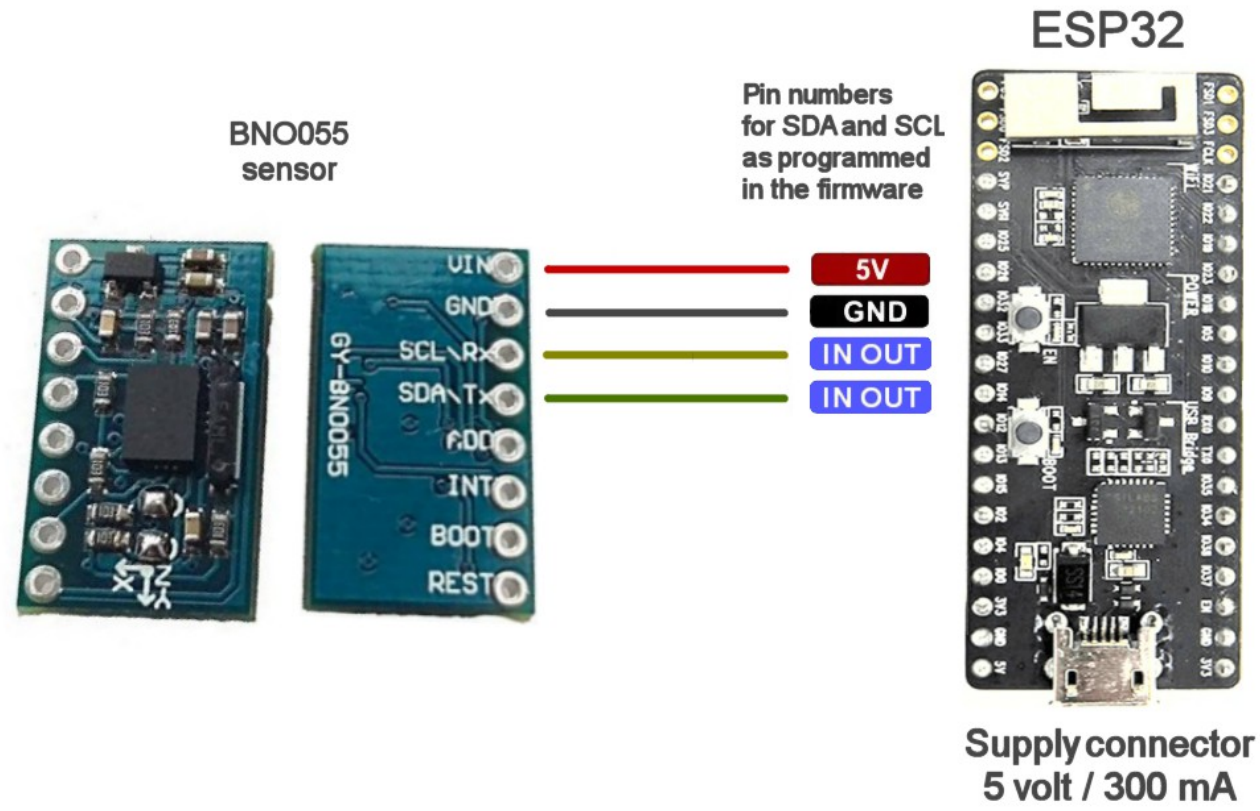
- Tensione di alimentazione (BNO055 chip) : da 2.4 a 3.6 volt
- Tensione di alimentazione (modulo completo) : 5 volt
- Corrente di alimentazione : Meno di 15 mA
- Risoluzione : Circa 12 bit
- Frequenza di campionamento : 100 Hz

Esempio 4 - Collegamenti (modulo cinese)

I fili da collegare allo ESP32 sono cinque VCC (5V), GND, SCL, SDA e 3.3V

Quindi prendiamo quattro fili (piccoli e flessibili) e li colleghiamo facendo molta attenzione.

Più di tutto si deve stare attenti a non invertire VCC e GND.

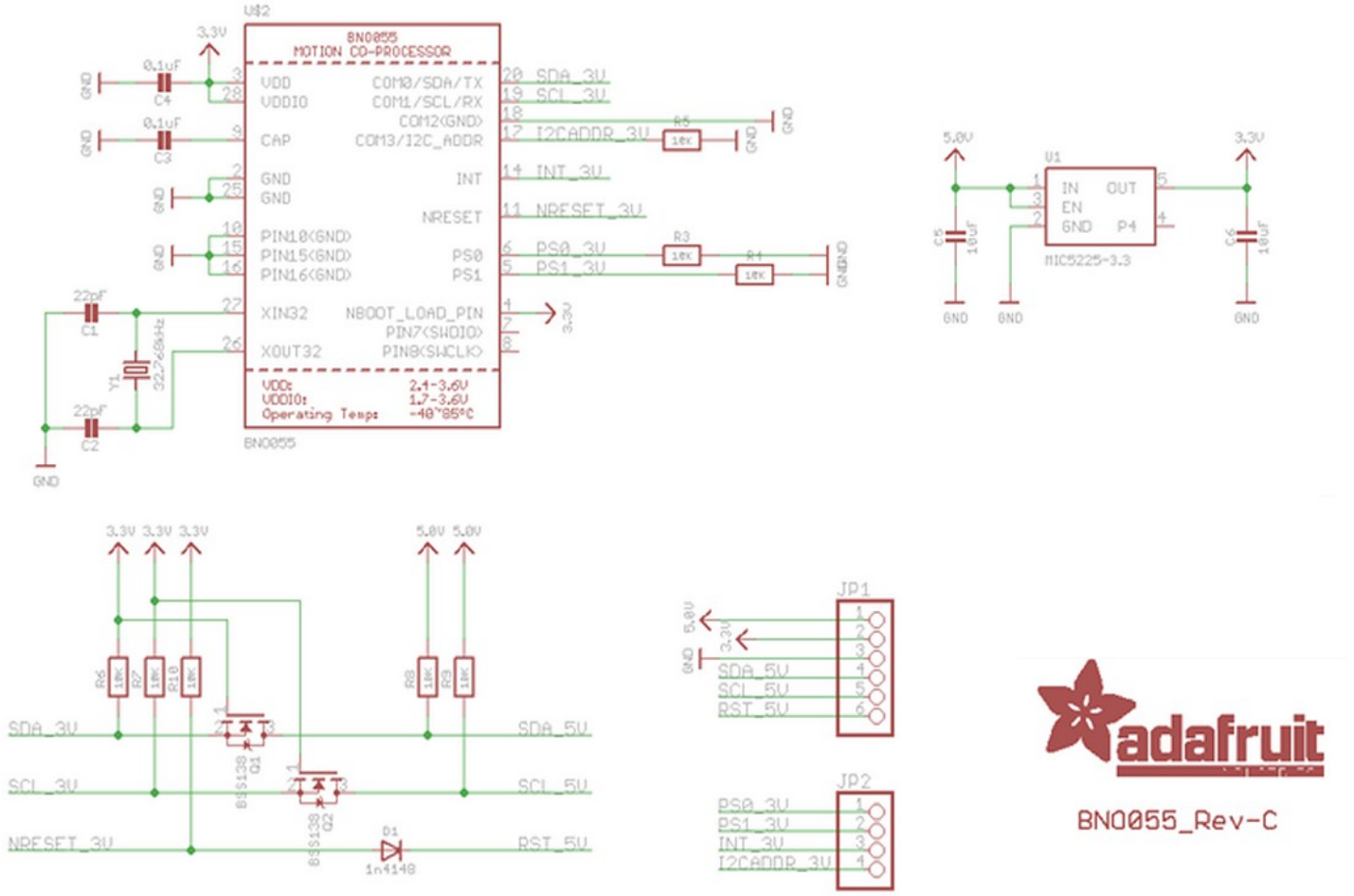


Per tutti gli esempi abbiamo scelto i Pin:

- 27 per il segnale SDA
- 14 per il segnale SCL

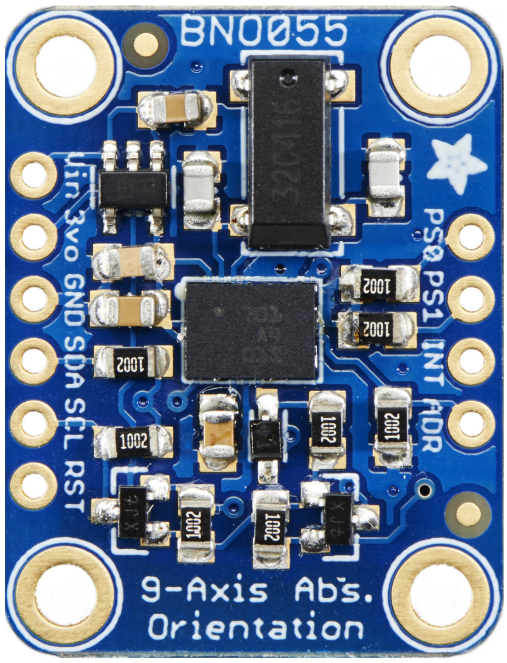
Per individuare la posizione dei Pin sui moduli ESP32,
consultare le immagini all'inizio di questo documento (pagina 4)

Esempio 4 - Collegamenti (modulo Adafruit)



Questo modulo, prodotto a Adafruit, è più costoso.

Questo modulo contiene un regolatore che genera i 3.3 volt e i traslatori di livello per i segnali SDA e SCL.



Quindi dovremo collegare:

- GND con il GND dello ESP32
- Vin con il 5 volt dello ESP32
- SDA con il Pin 27
- SCL con il Pin 14

Esempio 4 - Leggere i dati con la applicazione IotHAL

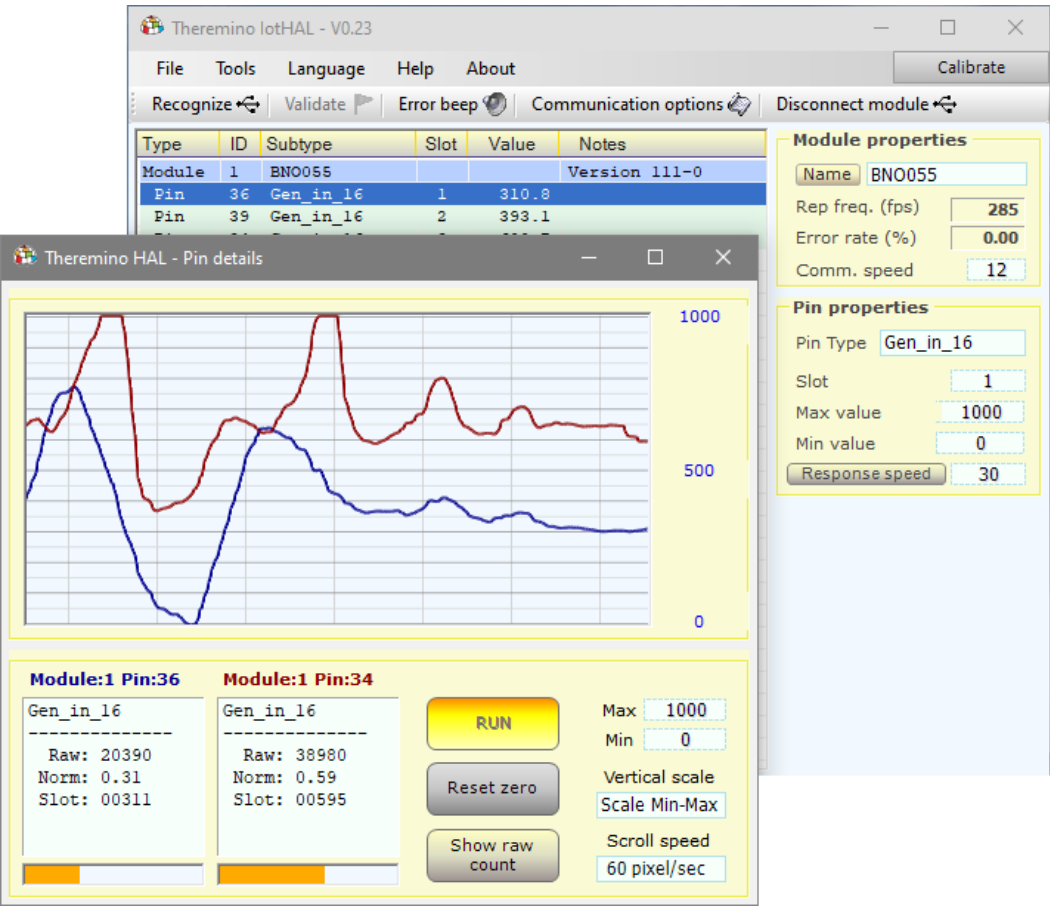
Per leggere lo "orientamento assoluto" del sensore BNO055:

- Si prepara l'IDE di Arduino, come spiegato nella documentazione della applicazione IotHAL.
- Si programma lo ESP32 con lo sketch "IotModule_AbsoluteOrientation.ino".
- Si lancia la Applicazione “IotHal”

Siccome nelle righe “genericWrite” del firmware abbiamo impostato i Pin 36/39/25, anche nella applicazione IotHAL dobbiamo configurare questo Pin come Ingressi Generici, per cui:

- PinType del Pin 36 = Gen_in_16
- PinType del Pin 39 = Gen_in_16
- PinType del Pin 25 = Gen_in_16

Se si utilizza il modulo WROOM, per ottenere tre Pin consecutivi, si potrebbe usare il Pin 34 al posto del 25. In questo caso si dovrebbe cambiare il numero da 25 a 34, sia nell'HAL che nel firmware e poi riprogrammare il modulo.



Ricordarsi anche di impostare “Velocità risposta” a 30 su questi Pin, in modo da filtrare il rumore e ottenere una misura più stabile.

Con questo sensore si ottiene un valore da 0 a 1000, che sono impostati come valore di default dalla applicazione HAL quando si configurano i Pin.

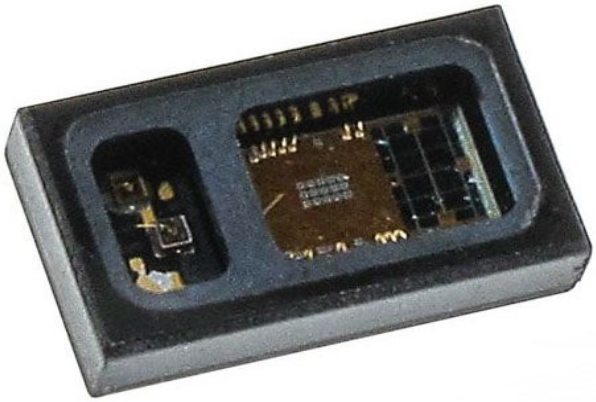
Volendo ottenere una misura in gradi, si modificherebbero i valori di Min e Max con -90 e +90.

Esempio 5 - Sensore per la frequenza cardiaca

Questo esempio, oltre a fornire il firmware già pronto come i precedenti, spiega anche i particolari delle funzioni che abbiamo scritto.

Si possono utilizzare indifferentemente i modelli MAX30102 e MAX30105 che sono quasi identici tra loro. Evitare il precedente MAX30100 che ha prestazioni minori e che non funzionerebbe perché i suoi registri interni sono diversi.

Questi chip hanno i collegamenti sotto. Per cui è molto difficile saldarli e di solito li si compra già saldati su piccole piastrine chiamate “breakout board”.



Ecco un esempio di “breakout board”.

Il MAX30102, comunica attraverso una interfaccia I2C e non sarebbe possibile leggerlo con un modulo Master.

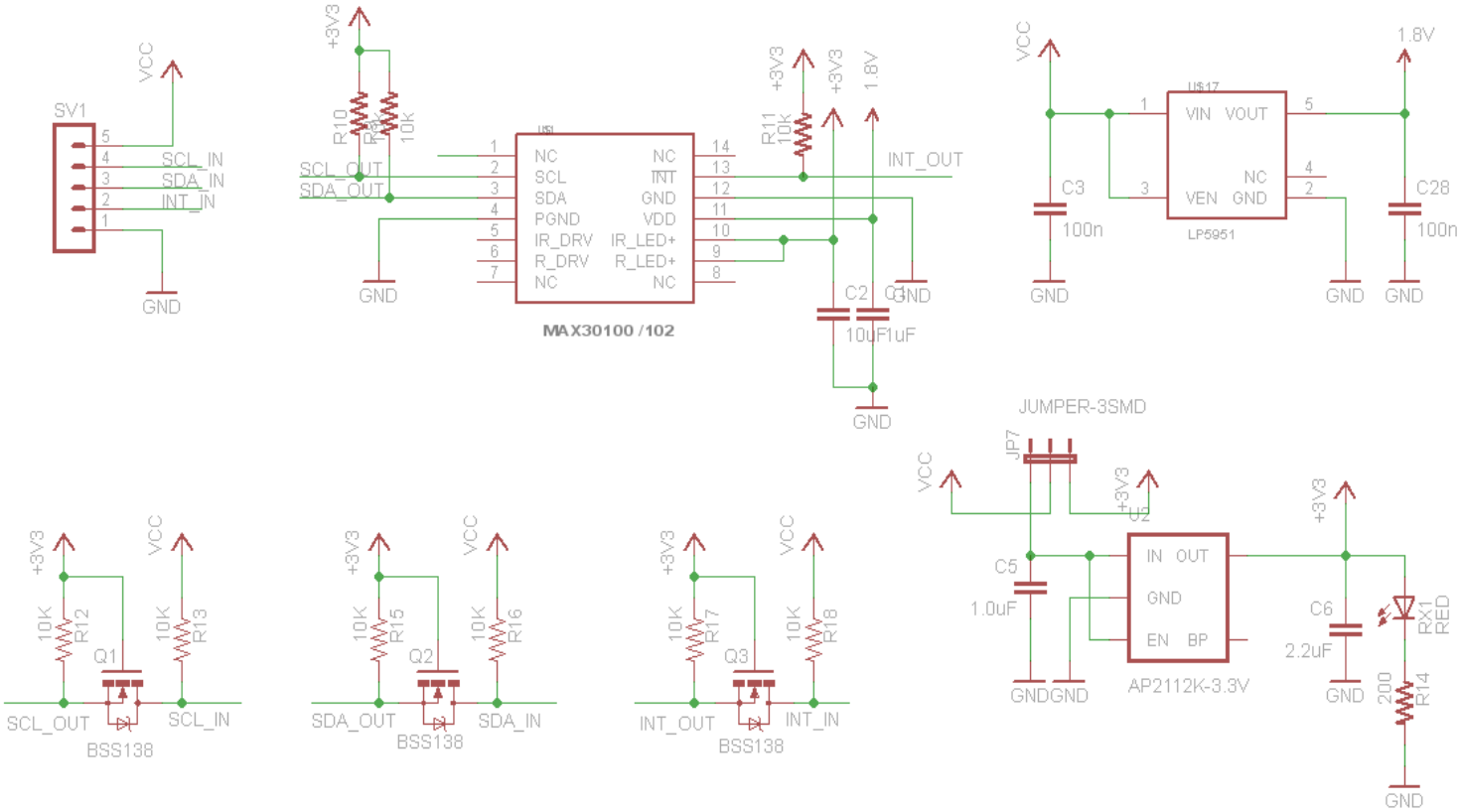
Si tratta quindi di un buon esempio dove conviene usare un Arduino o un ESP32 al posto di un Master. Ma attenzione a non prenderci gusto, in gran parte dei casi utilizzare un Arduino è più scomodo e le prestazioni sono minori.

In questo caso particolare una banda passante di 50 Hz è più che sufficiente, per cui la lentezza di comunicazione dei moduli Arduino non creerebbe problemi.

Se si utilizza un ESP32 si ottiene una buona velocità di comunicazione e anche il collegamento WiFi tra il modulo e il PC che ospita la applicazione HAL e le altre applicazioni del sistema theremino.

Esempio 5 - Modelli di sensori - MAX30102 Protocentral

Questo modello prodotto da Protocentral, una ditta indiana di Bangalore, è tra i migliori attualmente sul mercato. Ha una forma ben studiata e due comodi tagli dove far passare un tessuto elastico per tenere fermo il dito. Le sue caratteristiche sono ***ben specificate*** e si trova anche lo schema elettrico. L'unico suo difetto è di costare 25 Euro.



Questo schema può essere utilizzato come riferimento. Altri schemi più semplici non hanno i Jumper di selezione e alcuni non hanno nemmeno i tre Mosfet. I Mosfet traslano il livello dei segnali I2C tra la tensione del chip MAX3010x (1.8 volt) e la tensione del processore a cui lo si collega (3.3 volt o 5 volt).

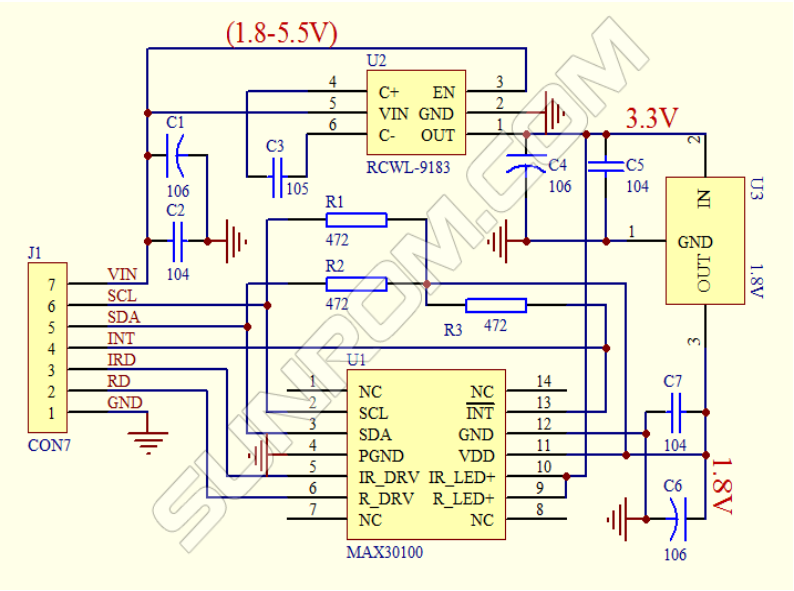
Probabilmente si riesce a comunicare anche senza questi transistor, ma non abbiamo provato tutti i tipi di moduli, per cui non possiamo assicurarlo. E' però importante, e consigliamo di controllarlo, che i resistori di pullup dei segnali I2C siano collegati alla tensione del processore (3.3 volt o 5 volt).

Esempio 5 - Modelli di sensori - MAX30102 Cinese verde

Questo modello lo si trova su eBay per pochi Euro. Di solito solo dai cinesi, per cui ci vuole un mese per averlo.

A quanto pare i resistori di pullup sono collegati alla tensione di 1.8 volt, ma non lo abbiamo provato, quindi non possiamo assicurare che funzioni.

Quelli che producono il “Pulse - Protocentral” della pagina precedente dicono che non può funzionare ma forse non lo hanno provato nemmeno loro e lo dicono per vendere il loro.



Ecco il suo schema. Non fate caso al chip 30100 al posto di 30102, il circuito stampato e i componenti sono gli stessi per i due chip.

I resistori di pullup R1, R2 e R3 sono effettivamente collegati alla tensione di 1.8 volt per cui potrebbe non funzionare.

Probabilmente si potrebbe tagliare la pista che va alla tensione di 1.8 volt e collegarla a VIN.

C'è anche la possibilità che questo sia uno schema vecchio e che i moduli in vendita su eBay siano stati corretti.

Oppure la I2C può funzionare anche così. Bisognerebbe provarne uno.

Modelli di sensori - MAX30102 Cinese magenta

Anche questo modello è su eBay. Costa pochi Euro, ma lo si trova solo dai cinesi, per cui ci vuole un mese per averlo.

Non si riesce a trovare lo schema elettrico per capire come sono collegati i traslatori di livello, ma lo abbiamo provato e Il numero “21” che identifica il chip come MAX30102 arriva correttamente.

Quindi siamo certi che la comunicazione I2C funziona.

Purtroppo ne abbiamo acquistato un solo esemplare e ha i LED che non si accendono. Ora ne abbiamo ordinati altri tre e appena arriveranno pubblicheremo ulteriori notizie.

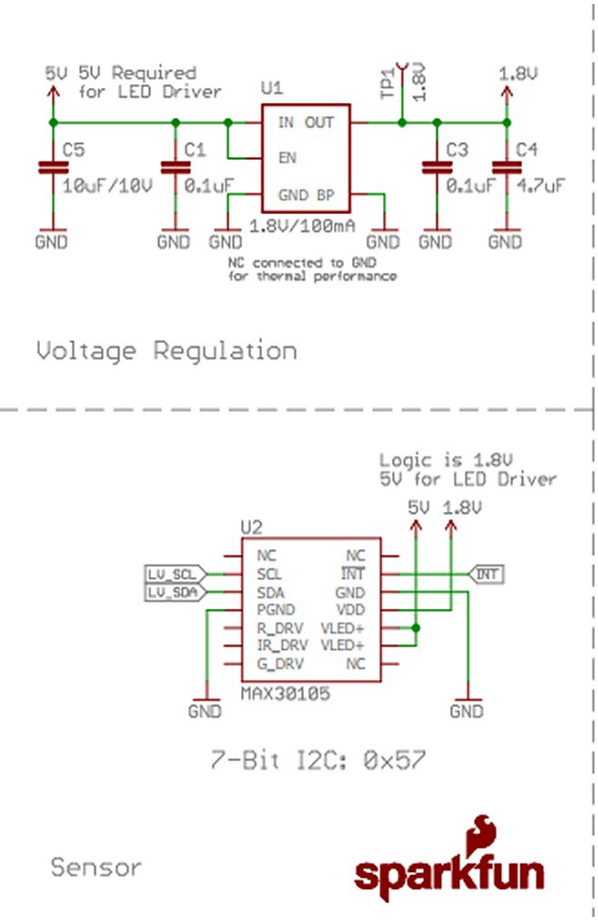
Esempio 5 - Modelli di sensori - MAX30105 Sparkfun

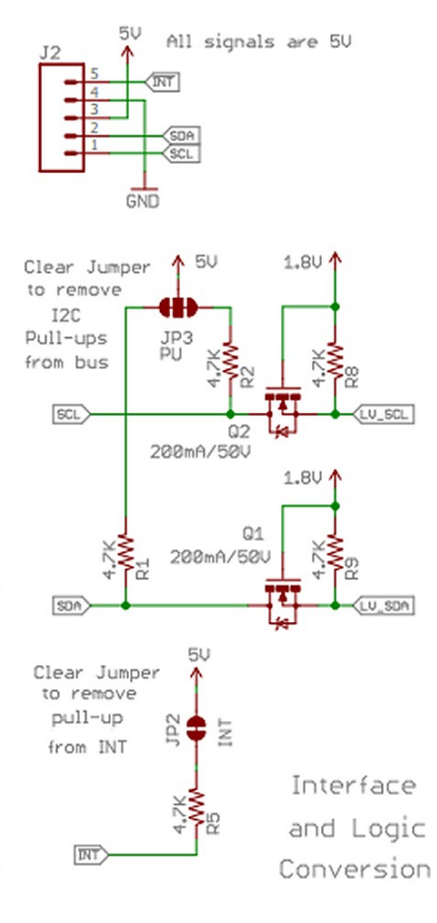
Questo MAX30105 è migliore dei precedenti, costa più di quelli cinesi (circa 19 euro spedizione compresa), ma ha tre led (IR, RED e GREEN) al posto di due (IR e RED).

Per cui può fare da cardio-frequenzimetro e pulsi-ossimetro come il MAX30102, ma anche da sensore per le polveri e per i fumi.

Inoltre le informazioni fornite da Sparkfun sono complete e precise e il PCB è ben studiato.

La tensione di alimentazione di 5 volt è ben specificata sulla serigrafia. Mentre nei modelli delle pagine precedenti non è specificata e questo poteva facilmente causare dubbi ed errori. Gli altri modelli scrivono solo Vin, per poter utilizzare sia il MAX30100 che il MAX30102 che vanno rispettivamente a 3.3 e 5 volt.





Lo schema non lascia dubbi, i convertitori di livello ci sono e i resistori di pullup sono collegati correttamente al 5 volt.

Ci sono anche dei jumper per eliminare i pullups nel caso i resistori fossero già presenti all'esterno. Nel nostro caso i pullup servono e quindi non si dovrà modificare niente.

Un altro vantaggio è che ci sono distributori in tutto il mondo e quindi lo si può avere in due giorni. Ad esempio gli Italiani possono trovarlo da ***Robot Italy***.

Esempio 5 - Quale modello di sensore acquistare

Per chi vuole spendere poco consigliamo di utilizzare questo modello:

- Costa poco, ma si consiglia di acquistarne due, perché non tutti funzionano.
- Le ultime versioni montano il MAX30102 che è migliore del 30100. Il circuito stampato è lo stesso ma i costruttori fanno un segno con la penna nera sull'ultima cifra. Comunque fate attenzione che nella pagina di vendita sia ben specificato il 102.

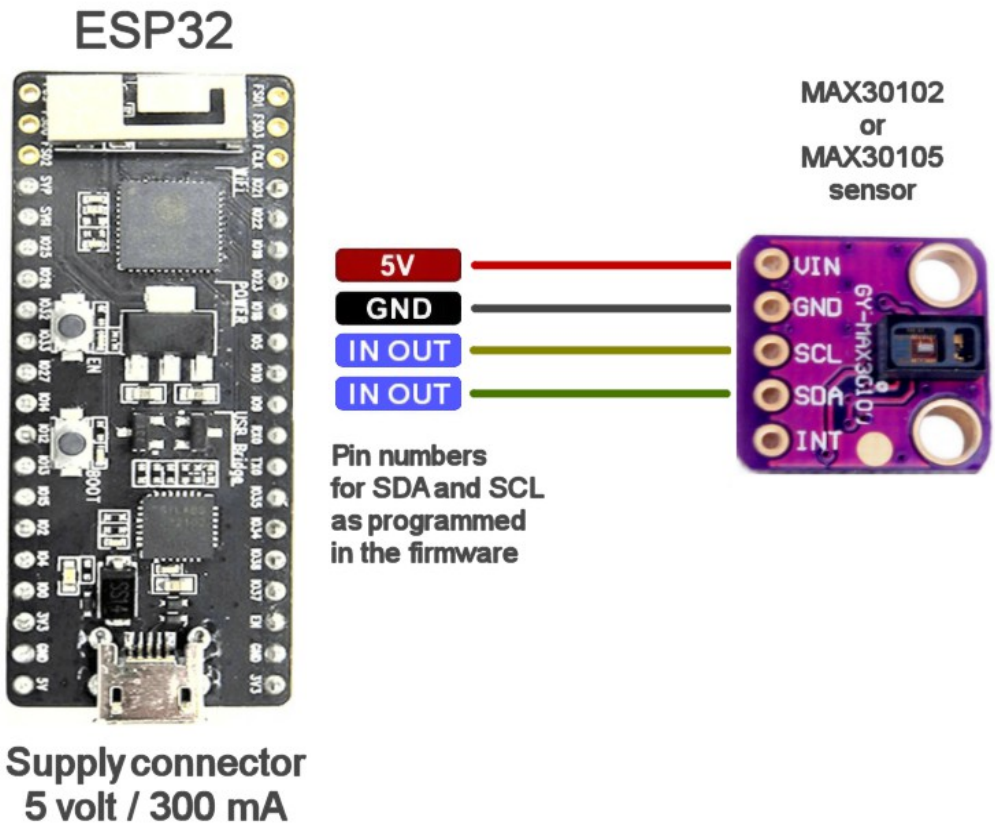
Spendendo un po' di più, si acquista lo Sparkfun:

- La qualità costruttiva è migliore e ha una buona documentazione.
- Monta il MAX30105 per cui può misurare anche le polveri e i fumi.
- Arriva in due o tre giorni.

Esempio 5 - Collegare il sensore allo ESP32

I fili da collegare allo ESP32 sono quattro VCC (5V), GND, SCL e SDA.

Quindi prendiamo quattro fili (piccoli e flessibili) e li colleghiamo facendo molta attenzione. Più di tutto si deve stare attenti a non invertire VCC e GND.



Per tutti gli esempi abbiamo scelto i Pin:

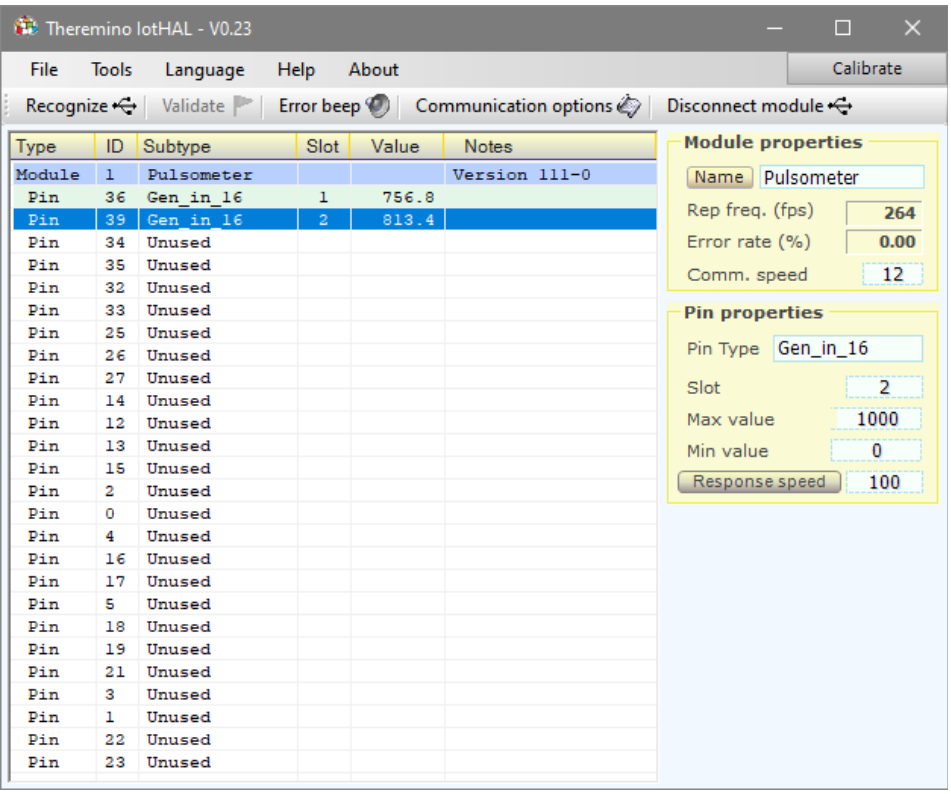
- 27 per il segnale SDA
- 14 per il segnale SCL

Per individuare la posizione dei Pin sui moduli ESP32,
consultare le immagini all'inizio di questo documento (pagina 4)

Esempio 5 - Leggere i dati con la applicazione IotHAL

Per leggere il sensore della frequenza cardiaca:

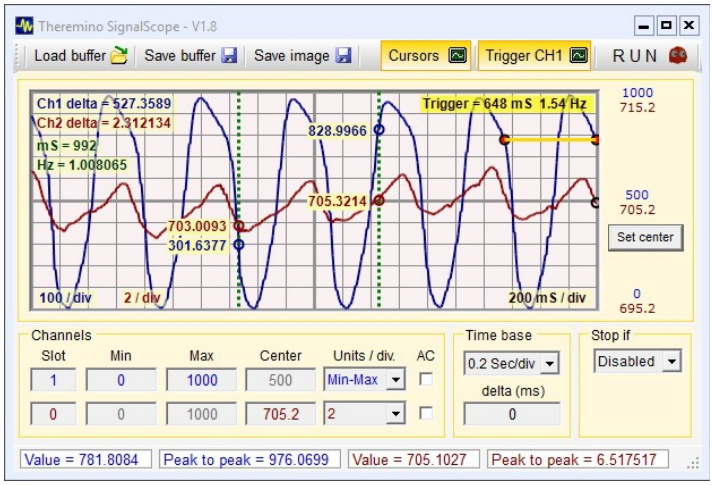
- Si prepara l'IDE di Arduino, come spiegato nella documentazione della applicazione IotHAL.
- Si programma lo ESP32 con lo sketch "IotModule_Pulsometer.ino".
- Si lancia la Applicazione “IotHal”

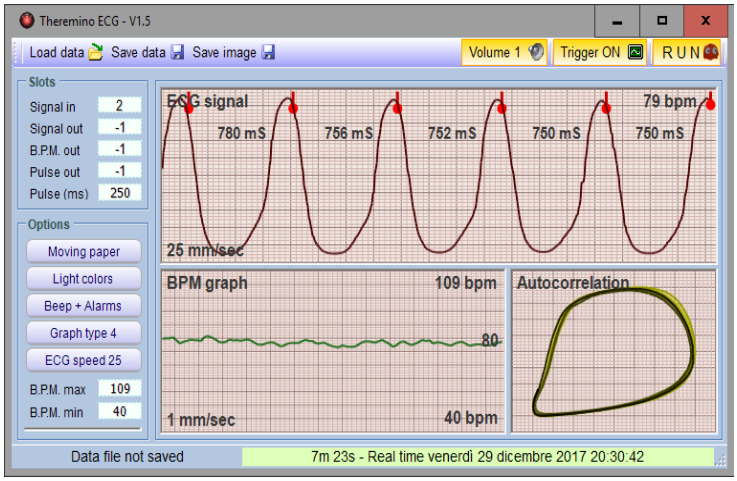


Siccome nel firmware, nelle due righe “genericWrite_16” abbiamo impostato i Pin “36” e “39”, anche nella applicazione IotHAL dovremo configurare il "PinType" di questi due Pin come "Gen_in_16".

Ricordarsi anche di impostare “Velocità risposta” a 30 su tutti e due i Pin, in modo da filtrare ulteriormente i dati.

Il dato numerico ricevuto sul Pin 39 viene inviato dallo IotHAL allo Slot 1, dal quale tutte le applicazioni del sistema possono leggerlo, ad esempio ***Theremino_SignalScope*** o ***Theremino_ECG***, che si vedono nelle due immagini seguenti.

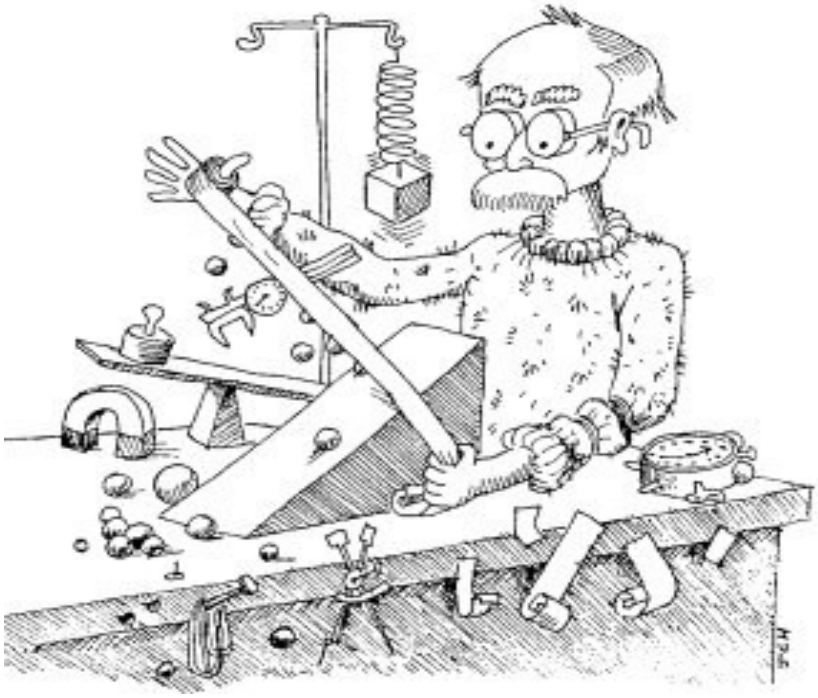




Tecniche usate nel firmware dei sensori

Per usare i sensori descritti in questa documentazione
non è necessario proseguire nella lettura.

Proseguite solo se volete imparare a scrivere firmware per sensori diversi da questi,
o per studiare le nostre tecniche per i filtri e per il controllo automatico di sensibilità



Nella prossima pagina spieghiamo come partire dal file "IotModule.ino" di base
e aggiungere le righe necessarie per leggere il sensore cardio tachimetrico.

Nelle pagine seguenti spiegheremo le tecniche da noi usate
per filtrare il segnale del sensore cardio tachimetrico
e per regolare il guadagno automaticamente.

Modificare il firmware partendo da: IotModule.ino

I passi seguenti sono solo una traccia
per chi volesse sviluppare progetti simili a questi.

Per leggere i sensori descritti in questo documento **non seguite queste istruzioni**,
ma tornate indietro di un po' di pagine e utilizzate il firmware già pronto.

Esempio di lettura di un sensore I2C (versione semplice senza i filtri)

- Si carica il file "IotModule.ino" nell'ArduinoIDE e lo si salva con un nome diverso, ad esempio “Pulsometer.ino”, in modo da non modificare il nostro firmware originale e poterlo riutilizzare per altri progetti.
- Si aggiungono i file specifici per il sensore (ad esempio “MAX30102.cpp” e “MAX30102.h”)
- Si aggiungono le righe seguenti all'inizio del file “Pulsometer.ino”

```
#include <Wire.h>
#include "MAX30102.h"
MAX30102 sensor;
```

- Sempre nel file “Pulsometer.ino”, nella funzione “void setup()”, si aggiungono le righe per inizializzare la libreria Wire (comunicazione I2C) e il sensore.

```
Wire.begin(pinSDA, pinSCL);          // Initialize I2C pins
sensor.begin();                      // Initialize the sensor
sensor.setLEDs(60, 60, 0);           // RED, IR and GRN LEDs - Using IR only
sensor.setPulseWidth(pw118);         // PulseWidth = 118 uS (adc 16 bit)
sensor.setSampleRate(sr1000);        // SampleRate = 1000 samples per second
sensor.setWorkingMode(wmHeartRate); // We implemented HeartRate only
sensor.setAdcRange(rge16384);        // adc range = max (16 uA)
```

- Sempre nel file “Pulsometer .ino”, nella funzione “void loop()”, si aggiungono le righe per leggere il sensore e inviare i suoi dati all'IotHAL

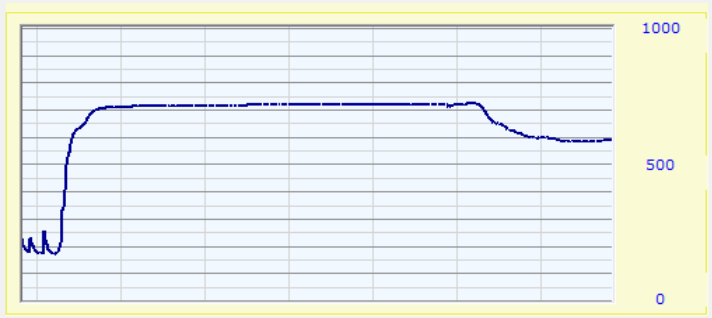
```
// ---------------------------------------- Read the sensor
sensor.readSensor();
// ---------------------------------------- Send RED and IR values to IotHAL
Theremino.genericWrite16(36, sensor.IR);
Theremino.genericWrite16(39, sensor.RED);
```

- Si scrive il firmware sullo ESP32 e si lancia lo IotHAL
- Nello IotHAL si configurano i Pin 36 e 39 come Gen_in_16
- Se il sensore è collegato si dovrebbe iniziare subito a ricevere i dati.

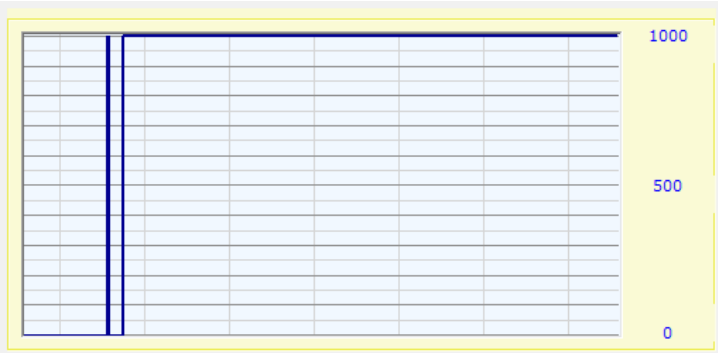
Filtrare i dati del sensore per la frequenza cardiaca

I dati in arrivo dal sensore sono di piccola ampiezza e le pulsazioni sono quasi invisibili.

La scala di misura che qui vediamo è "normalizzata" da 0 a 1000 e le pulsazioni non raggiungono nemmeno un millesimo di questa scala. Inoltre le pulsazioni sono mascherate da rumore e da dondolii dovuti ai movimenti della mano. In pratica si vede solo una linea che va in alto mettendo il dito e in basso togliendolo.

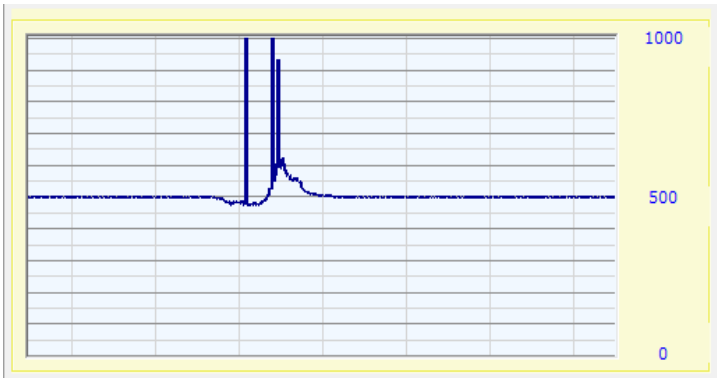


In questa immagine si vede la salita (a sinistra) quando il dito viene infilato, una zona piatta di circa 4 secondi con il dito fermo e una discesa provocata dall'aver mosso un po' il dito. Come si vede le pulsazioni sono totalmente invisibili



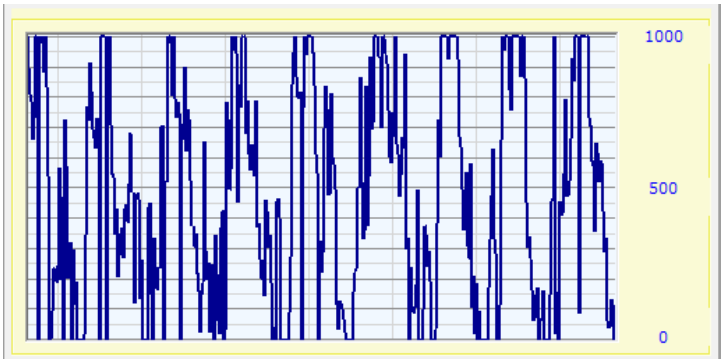
Dobbiamo quindi amplificare il segnale ma amplificando tutto ciò che è sotto alla metà va sotto zero e le parti oltre la metà vanno fuori scala in alto.

Per cui mettendo il dito si vede qualcosa come in questa immagine. Quindi prima di amplificare si devono eliminare i dondolii con un filtro passa alto.



Ecco l'effetto del filtro passa alto sul segnale non amplificato.

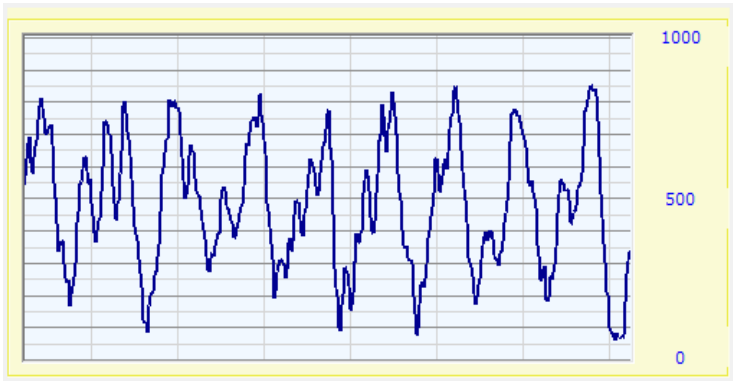
Nella parte sinistra il dito non c'era. Nel momento in cui viene inserito si vede un forte disturbo e dopo circa mezzo secondo il filtro passa alto riporta il segnale a metà scala.



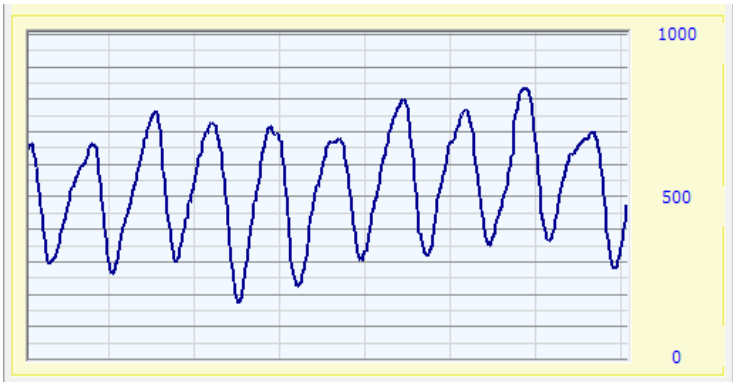
Avendo un segnale ben centrato sul valore centrale possiamo amplificarlo molto e iniziare a vedere qualcosa.

Ecco l'effetto di una amplificazione di 2000 volte.

Si cominciano a intravedere le pulsazioni ma sono sommerse da molto rumore.



Con un filtro passa basso le pulsazioni diventano ben riconoscibili.

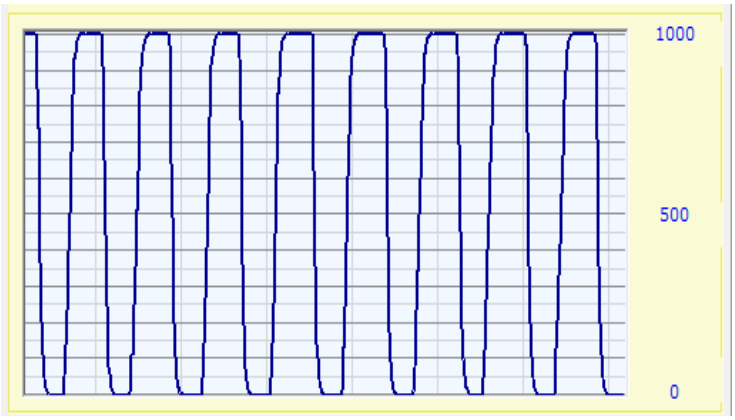


Aggiungendo un secondo stadio al filtro passa basso e regolando a 30 il filtro IIR nel NetHAL il rumore sparisce completamente.

La stabilità non è eccezionale ma è già migliore delle immagini che vengono pubblicate in rete.

Ecco un tipico ***esempio di pulsazioni*** scaricato dal sito di Sparkfun che costruisce la break-board utilizzata in queste prove.

A questo punto è importante notare che tutti i test precedenti sono stati fatti sul segnale peggiore possibile. Uno di noi (che scrive qui) ha quasi sempre le mani gelate e una circolazione periferica quasi inesistente, per cui è un ottimo soggetto per mettere alla prova i sensori.



Se si dispone di un segnale forte tutto diventa più facile. Ecco il segnale di un paziente con la pressione alta e una circolazione periferica normale.

In questo caso si potrebbe anche amplificare meno per non tosare il segnale in alto e in basso.

Comunque per misurare la frequenza e le aritmie la forma del segnale non conta. Per cui un segnale forte come questo, anche se squadrato, potrebbe fornire misure molto stabili.

Implementare i filtri nel firmware

Un filtro passa basso può essere costruito con una sola riga di software.

```
LowPass += (InputData - LowPass) * 0.02;
```

Per implementare un filtro passa alto si aggiunge una seconda riga che fa la differenza tra il segnale e l'uscita del passa basso. Quindi togliendo dal segnale le frequenze basse restano solo quelle alte, e si ottiene il passa alto.

```
LowPass += (InputData - LowPass) * 0.02;
HiPass = sensor.IR - LowPass;
```

Questi semplici filtri sono l'esatto equivalente dei filtri hardware composti da un resistore e un condensatore e sono anche regolabili. Se si aumenta il coefficiente (che qui è 0.02) la frequenza di taglio si alza.

E' necessario regolare questi filtri sperimentalmente perché la frequenza di taglio dipende dal tempo di ripetizione con cui queste righe vengono chiamate. E questo tempo dipende a sua volta da quanta elaborazione si aggiunge nel Loop di Arduino.

Per evitare questa laboriosa taratura, nella libreria "ThereminoFilters" abbiamo misurato il tempo di ripetizione del loop e corretto i filtri ad ogni passo.

Si può quindi impostare una frequenza di taglio in Hz (e frazioni di Hz), e questa verrà rispettata sempre (a patto che la frequenza di ripetizione del loop sia almeno il doppio della frequenza più alta di nostro interesse). Questa non è una richiesta difficile da rispettare perché solitamente la frequenza di ripetizione è almeno dieci volte maggiore delle frequenze più alte del segnale. Un'alta frequenza di ripetizione si chiama "sovra-campionamento" e serve per evitare i fenomeni di aliasing, cioè il ribaltamento nella banda del segnale, dei segnali indesiderati (rumore) che hanno frequenze maggiori della frequenza di campionamento.

Esempio di utilizzo dei filtri

Questo esempio mostra come si usano i filtri della libreria "ThereminoFilters".

Prima di tutto tra le prime righe del file ".ino" si deve aggiungere la riga:

```
#include "Utility/ThereminoFilters.h"
```

Poi si devono dichiarare tutti i filtri che useremo, e le loro frequenze di taglio, nella zona che si trova subito prima della funzione "void Loop()"

```
// ------------------------------- Filters declarations - HiPass 0.7 Hz
Filter hipass1(0.7, true);
Filter hipass2(0.7, true);
Filter hipass3(0.7, true);
Filter hipass4(0.7, true);
// ------------------------------- Filters declarations - LoPass 3 Hz
Filter lopass1(3, false);
Filter lopass2(3, false);
Filter lopass3(3, false);
Filter lopass4(3, false);
// ------------------------------- Filters declarations - LoPass 2 Hz for AutoGain
Filter lopass5(2, false);
```

Infine si utilizzano i filtri uno dopo l'altro. In questo caso ne abbiamo utilizzati otto per ottenere esattamente lo stessa curva di risposta del sensore “Theremino Pulsometer” che si vede in ***questa pagina***.

Attenzione: Ogni filtro dichiarato deve essere usato una volta sola. Se si ripetesse due volte la riga di un filtro l'effetto sarebbe quello di una riga sola, e si sprecherebbe tempo di calcolo.

```
void loop()
{
    sensor.readSensor();
    float filtered = sensor.IR;

    // ------------------------------------------- Hi Pass - 4 stages
    filtered = hipass1.run(filtered);
    filtered = hipass2.run(filtered);
    filtered = hipass3.run(filtered);
    filtered = hipass4.run(filtered);
    // ------------------------------------------- Low Pass - 4 stages
    filtered = lopass1.run(filtered);
    filtered = lopass2.run(filtered);
    filtered = lopass3.run(filtered);
    filtered = lopass4.run(filtered);

    ....
    ....
```

Nelle righe seguenti del loop il segnale filtrato viene amplificato e inviato allo IotHAL, come vedremo nella prossima pagina.

Amplificare il segnale e inviarlo allo IotHAL

Il blocco seguente regola il guadagno (amplificazione) per ottenere un segnale di uscita di ampiezza costante con tutti i pazienti.

```
    // ------------------------------------------- Auto gain
    float v = abs(filtered);
    v = lopass5.run(v);
    float gain = 24000 / v;
    if (gain > 5000) gain = 5000;
    filtered *= gain;
```

Il segnale filtrato viene “rettificato” con la funzione abs. Cioè la parte negativa del segnale viene ribaltata in positivo. Poi il valore rettificato viene passato in un filtro passa basso e si ottiene una stima della ampiezza del segnale. Poi si calcola il guadagno che si dovrà applicare al segnale. Il numero 24000 è stato trovato sperimentalmente per ottenere la massima ampiezza ma lasciando un piccolo margine sopra e sotto. Poi si limita il guadagno a 5000 per evita che cresca troppo quando manca il segnale. Se crescesse troppo il rumore verrebbe amplificato fino a sembrare un segnale utile. E infine si effettua la amplificazione con la riga “filtered *= gain”

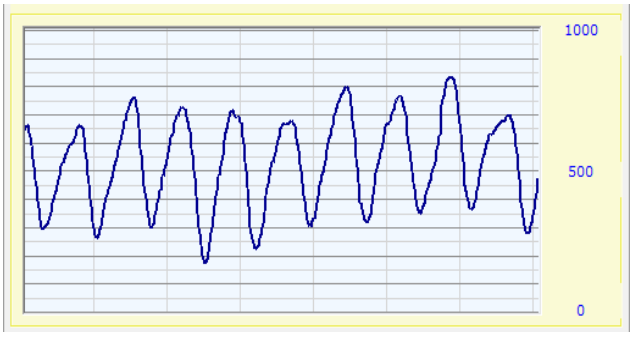
```
    // ---------------------------------------- Limit amplitude to unsigned 16 bits
    filtered += 32768;
    if (filtered > 65535) filtered = 65535;
    if (filtered < 0) filtered = 0;
```

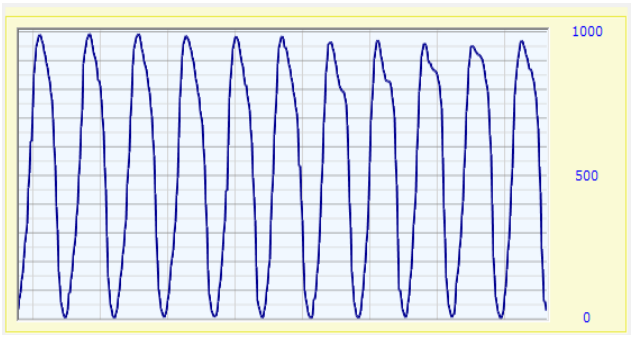
Nella prima riga il segnale viene traslato in alto di 32768 (metà di un 16 bit) così non è più un numero centrato sullo zero ma centrato in un numero intero senza segno da 16 bit. Nelle due righe seguenti viene limitato per farlo stare in un numero da 16 bit, cioè tra 0 e 65535.

```
    // ---------------------------------------- Send raw IR and filtered IR to IotHAL
    Theremino.genericWrite16(36, sensor.IR);
    Theremino.genericWrite16(39, filtered);
}
```

Infine si invia allo IotHAL il segnale “sensor.IR” non filtrato e non amplificato.
E con una seconda riga si invia anche il valore filtrato.

Nelle prossime due immagini si vede il miglioramento che si ottiene con il guadagno automatico

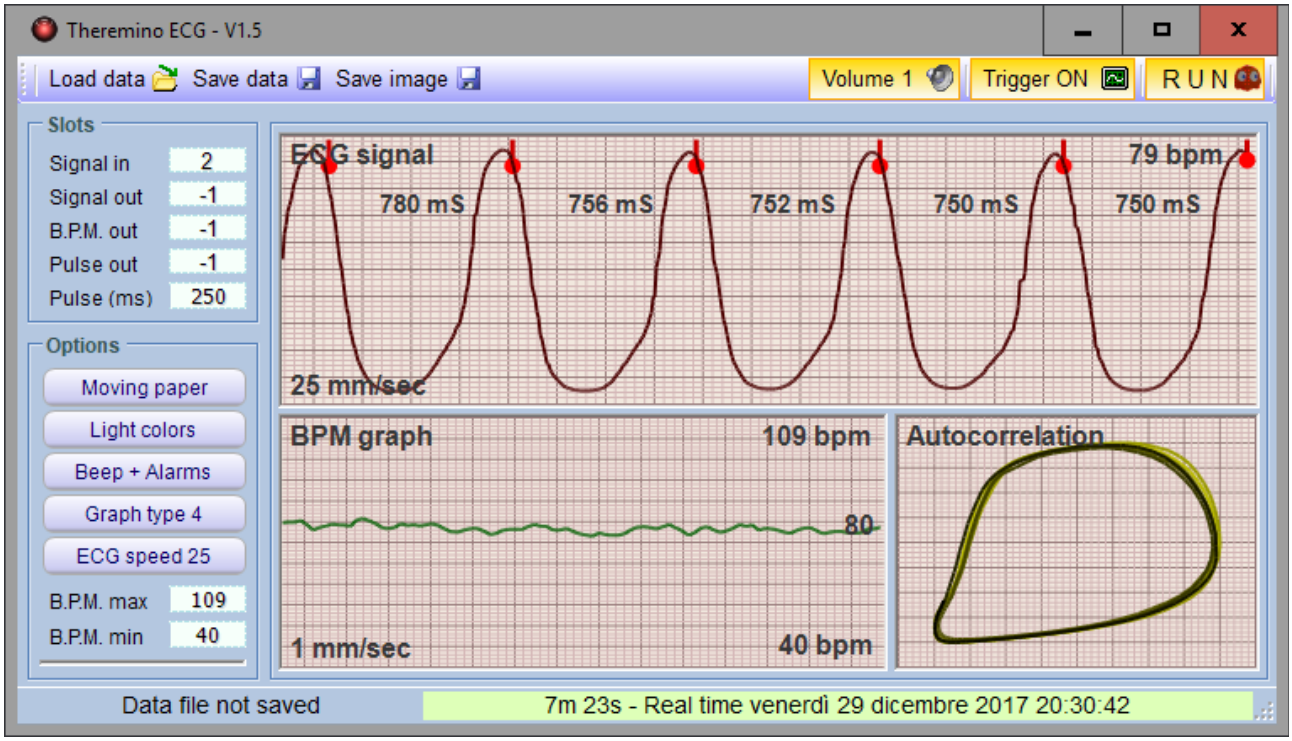




Confronto con il PulsoSensor

Chi ha la pressione bassa e le mani sempre fredde produce un segnale debole perché ha una circolazione periferica scarsa. In alcune circostanze, ad esempio durante la digestione, la circolazione periferica si riduce ulteriormente. In questi momenti alcune persone potrebbero diventare soggetti difficilissimi da misurare.

Quindi, sia nel ***PulsoSensor*** (collegato al modulo Master), che in questo sensore MAX30102 (collegato ad uno ESP32), abbiamo ottimizzato la curva di risposta e la amplificazione per massimizzare la affidabilità nella ***misura della frequenza e nella ricerca delle aritmie***.



In questa immagine si vede la applicazione ECG che si scarica da ***questa pagina***.

Utilizzando quattro filtri passa alto e quattro passa basso si ottengono quasi gli stessi risultati che abbiamo ottenuto con i filtri a resistori e condensatori del ***PulsoSensor***.

Il PulsoSensor va comunque leggermente meglio perché la luce attraversa il dito e non viene riflessa dai primi strati della pelle. Questo è ben spiegato nella sua ***documentazione***.

- - - - - -

Tutte le misure qui mostrate sono state fatte su un soggetto notoriamente difficile (l'autore di queste pagine) che ha la pressione bassa e che in certi momenti della giornata ha la circolazione periferica quasi inesistente (mani gelate).

Con altri soggetti il segnale può essere notevolmente migliore. In alcuni casi il segnale può essere così forte da saturare e diventare quasi un'onda quadra. Questa deformazione del segnale non crea problemi dato che per la ricerca delle aritmie siamo interessati solo alla frequenza e non alla forma d'onda.

Calcolare la saturazione di ossigeno

Il segnale dei sensori MAX3010x è appena sufficiente per misurare la frequenza cardiaca. E anche nelle migliori condizioni il segnale non è molto stabile e si deve stare fermi durante la misura della frequenza.

Per ottenere un minimo di affidabilità abbiamo filtrato pesantemente il segnale e modificato continuamente la amplificazione. Ma queste tecniche sono incompatibili con la misura della saturazione perché l'algoritmo che la calcola ha bisogno dei segnali “RED” e “IR” non filtrati.

Misurare la saturazione di ossigeno richiederebbe un segnale molto più ampio e privo di rumore e anche nelle condizioni migliori la precisione di misura sarebbe scarsa, ***vedere questa pagina***.

Dalle nostre prove nella gran parte dei casi reali si possono ottenere solo numeri a caso e del tutto inutili. Solo con alcuni pazienti che danno un segnale molto forte, e stando perfettamente fermi, si potrebbe ottenere un minimo di precisione.

Secondo noi un apparecchio così inaffidabile non serve a molto e non vale la pena di perderci del tempo, per cui lasciamo volentieri ad altri il divertimento di provarci.

- - - - - - -

Le librerie di Maxim per calcolare la saturazione si scaricano da qui:
https://github.com/sparkfun/SparkFun_MAX3010x_Sensor_Library/tree/master/src

Attenzione che non basta aggiungere i file “c++” e “h” al nostro progetto. Tutte le nostre realizzazioni lavorano in modo continuo su un dato per volta, mentre gli algoritmi di Maxim lavorano solo su un lungo buffer di campioni memorizzati. Bisognerebbe dunque scomporli e ricomporli in modo diverso, ed è un lavoro per cui ci vuole molta esperienza nella programmazione.

Prima di intraprendere un lavoro del genere consigliamo di provare il progetto completo di Maxim, senza IotHAL, ma in seriale come lo hanno concepito loro. Potrete constatare che nella quasi totalità dei casi non fornisce nessun risultato, cioè va in errore o da numeri totalmente sbagliati.

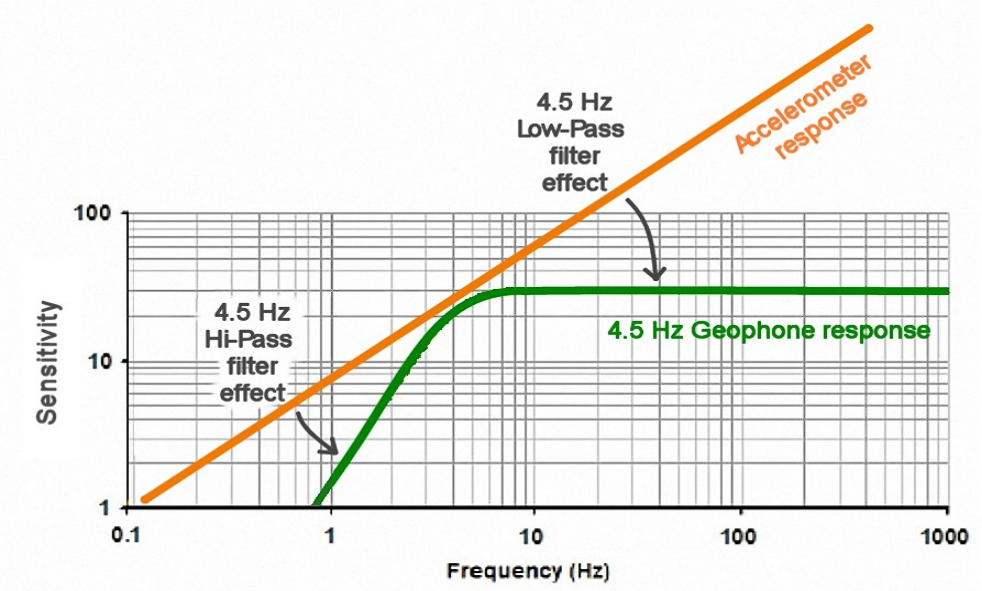
Solo riuscendo a farlo funzionare in modo affidabile si potrebbe pensare di perderci del tempo e collegarlo allo IotHAL.

Trasformare gli accelerometri in velocimetri

Il firmware degli esempi 2 e 3 (accelerometri a tre assi) contiene filtri passa basso (uno per asse) che trasformano il sensore da "accelerometro" a "velocimetro".

Il filtro passa basso integra i dati della accelerazione nel tempo e l'integrale della accelerazione è la velocità.

Con questo filtro si ottengono dati simili a quelli che si otterrebbero dai geofoni elettromeccanici, come quelli della immagine qui a destra.



I geofoni più usati hanno una risposta che inizia a 4.5 Hz per cui nel firmware abbiamo regolato la frequenza di taglio dei filtri a 4.5 Hz.

I filtri passa alto correggono la pendenza per le frequenze da 4.5 Hz in giù, mentre quelli passa alto la correggono dai 4.5 Hz in su.

I filtri passa alto, oltre a ottenere un risposta in frequenza simile a quella dei geofoni, eliminano anche la componente continua dei segnali. Quindi il valore medio del segnale è sempre esattamente a metà tra il minimo e il massimo (il valore 500 nel nostro sistema).

Nel firmware troverete la linea:

```
#define USE_FILTERS
```

Commentando questa linea si eliminano tutti i filtri. Fare un test senza filtri potrebbe essere utile per calibrare la sensibilità con il valore di accelerazione di gravità, che è circa uno.

Poi troverete due linee che determinano le frequenze dei filtri:

```
const float HiPassFreq  = 4.5;
const float LowPassFreq = 4.5;
```

Cambiando la frequenza di taglio, si potrebbero simulare geofoni diversi. Ad esempio i geofoni da 10 Hz, o quelli costosissimi da 2Hz.

Normalmente si dovrebbero mantenere le due frequenze (passa alto e passa basso), uguali tra loro, ma abbiamo lasciato la possibilità di fare esperimenti e di regolarle indipendentemente.