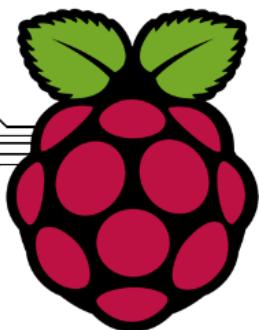
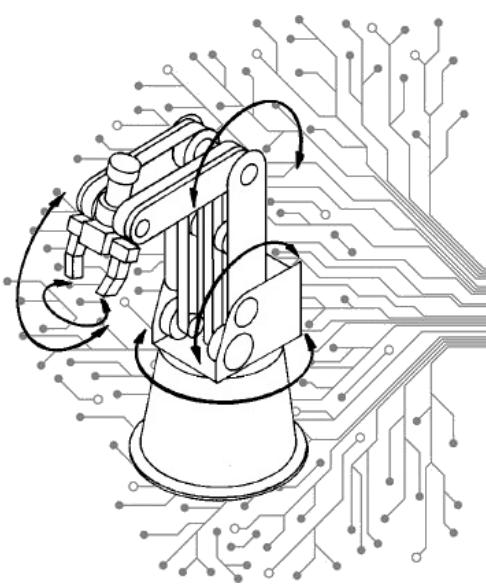




SCORBOARD-V2

MANUALE TECNICO *Descrizione*

Manuale operativo e progettuale della scheda di controllo "ScorBoard-V2".
Scheda sviluppata per il controllo di uno Scorbot



Ingegnere Emanuele Alfano
Università degli studi di Tor Vergata, 2019

Introduzione

Il seguente manuale è pensato per spiegare come usare la board "ScorBoard_V2", ma fa parte del progetto più grande di controlli automatici dell'università di Tor Vergata.

Il progetto completo è disponibile su GitHub all'indirizzo: <https://github.com/Alfystar/Scorbot-CA>

Tutto il materiale usato e descritto sotto è trovabile a questo indirizzo, e tanto altro.

La board è stata disegnata usando EAGLE 9.4.2 Educational.

Il codice firmware è stato sviluppato usando l'editor avanzato **SLOEBER**, un editor pensato appositamente per programmare gli Arduini con un editor avanzato e una struttura professionale, lavorando direttamente sul chip ATmega, senza perdere l'uso delle librerie Arduino.

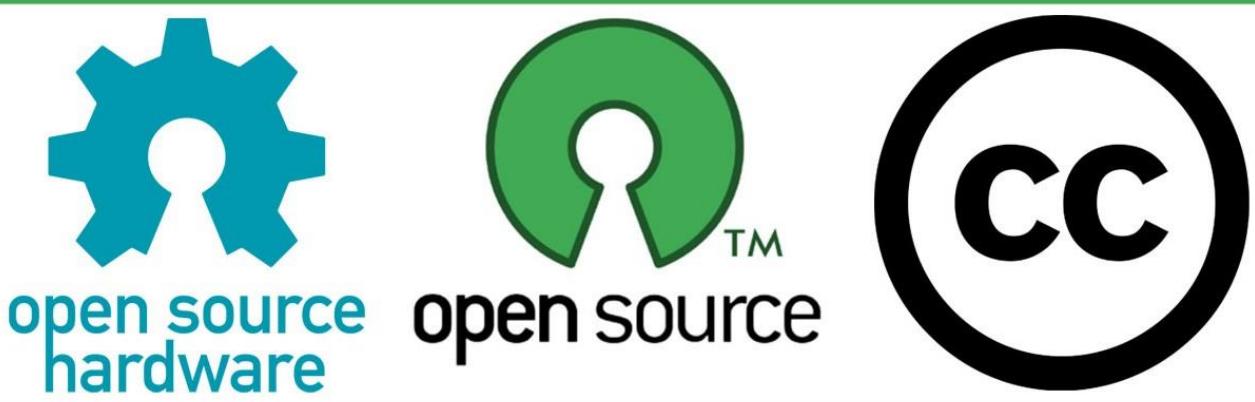
Il progetto è stato sviluppato da degli studenti nella logica **dell'Open Hardware e dell'Open Software**, qualunque modifica o sviluppo è ben accetta e sarebbe gradita una proposta di merge al progetto principale.

Questo manuale vuole spiegare come usare la board, ma anche i passi che hanno portato alla sua realizzazione, le motivazioni tecniche, le caratteristiche pecuniarie, sia hardware che software, al fine di semplificare l'uso e accelerare il loro sviluppo da parte di altri ricercatori.

Di seguito è presente un indice con tutti gli argomenti trattati, pregherei di leggere la tabella tecnica con i massimi valori in corrente e tensione applicabili alla scheda, così da evitare spiacevoli danni.

Grazie per l'attenzione

Ingegnere Emanuele Alfano



Indice

SPECIFICHE TECNICHE	1	ADC ATMEGA	11
Global Board	1	Registri interni default	11
Motors	1	Parametri programmabili	11
Only one Motor Drive	1	Affidabilità tensioni di riferimento	11
Logic	1	Traslatore di livello	12
ASSEMBLAGGIO	2	FIRMWARE (ARDUINO)	13
Componenti board	2	Architettura di Sistema	13
Saldatura Componenti	2	Principio di funzionamento Software	13
Collegamento board al sistema	3	Controllo motori	14
Alimentazione standard	3	Modalità di controllo supportate	14
Alimentazione motori indipendente	3	Macchina a stadi automatica	14
Ventola di raffreddamento	3	Sistem Motor	15
Taratura	4	Frequenza PWM	15
Trimmer R-REF	4	Lettura delle correnti	17
Trimmer R-OFF (facoltativo)	4	Free-running & Interrupt	17
DOCUMENTAZIONE ELETTRONICA	5	Impostazioni per il V-REF	17
Schema dell'Alimentazione	5	Lettura Diretta/Differenziale	17
Connettore Scrbot	6	Calcoli e conversioni	17
Driver Motori VNH5019	7	Class AdcDevice	18
Configurazioni del ponte	7	Lettura degli encoder & micro-switch	19
Lettura corrente motori	8	Il buffer Circolare	19
<i>Frequenza 500hz (default arduino)</i>	8	Elaborazione encoder	20
<i>Frequenza 4khz (default board)</i>	9	Classe ScrbFeed	21
Misura sensibilità di uscita	10	Procedura di Discovery Home (Homming)	22
		APPENDICE A: SCHEMA ELETTRICO	23

Specifiche Tecniche

Caratteristiche elettriche della scheda per un uso a REGIME:

Global Board

	Min	Typical	Max	Unit
Board Consumption Ampere	2	Base on motor	14	A
Board Voltage Power	7	12	28	V

Motors

	Min	Typical	Max	Unit
Motors Current	Base on motor	Base on motor	10	A
Motors Voltage	7	Base on motor	41	V

N.B.

È possibile alimentare i motori con un secondo alimentatore rispecchiante le proprie necessità, purché la corrente erogata (continuativamente) non superi i 10A (massimo consentito dalle piste), con possibili picchi fino a 30A non continuativi

Only one Motor Drive

	Min	Typical	Max	Unit
Motor Current	Base on motor	12	30	A
Motors Voltage	7	Base on motor	41	V

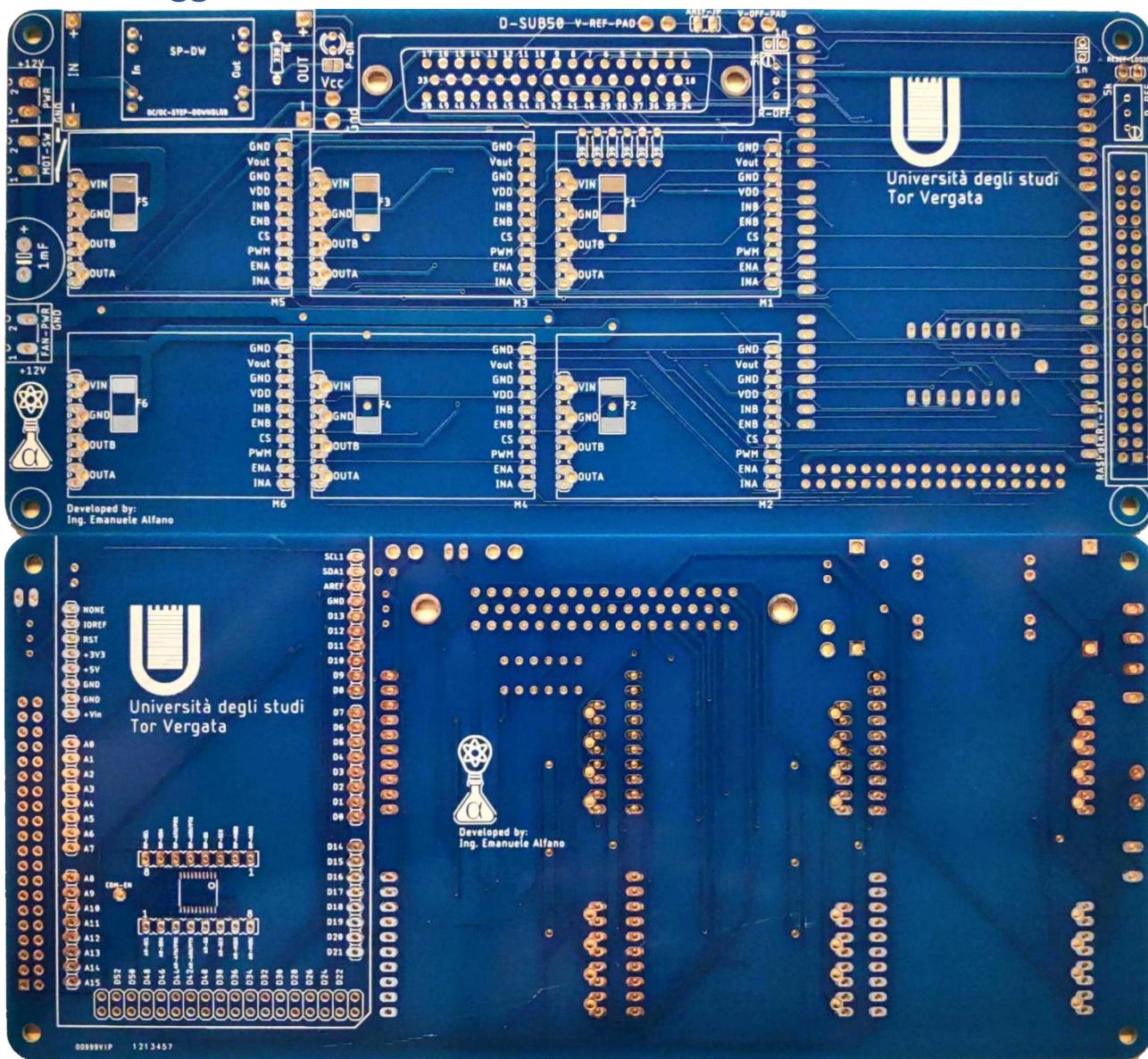
N.B.

Le specifiche riguardano il driver motori, il fusibile che si posiziona (di default 3A) limita la massima corrente erogabile dal driver

Logic

	Min	Typical	Max	Unit
Logic Current consumption	2	3	4	A
Logic Voltage	4.5	5	5.5	V

Assemblaggio



Componenti board

La scheda è principalmente una Motherboard che connette l'Arduino Mega ai driver motori e ai sensori dello Scorbot. Il kit fornisce un PCB e i componenti necessari:

- 6x Resistori 39Ω (PWR encoder sensori Scorbott)
- 1x Resistore 330Ω
- 2x Trimmer di taratura
(uno opzionale, vedere sezione [Taratura](#))
- 1x Condensatore 1mF
- 2x Condensatore 1n
- 1x Diodo protezione alimentazione
- 1x Led indicatore di alimentazione
- 6x Fusibili smd motori
- 3x Connettori a Vite passo 5mm
- 1x Step-Down del tipo LM2596 o MP1584
- 1x Connettore D-SUB50 (attacco Scorbott)
- 6x Driver Motori VNH5019
- 1x Connettore RASPBERRY-PY B+
- 1x Traslatore di Livello smd TXB0108

Saldatura Componenti

L'ordine di saldatura consigliato è:

1. Saldare il traslatore di livello smd TXB0108 sul retro della board.
 2. Saldare i componenti smd sul davanti della board (fusibili).
 3. Saldare le Resistenze e condensatori da 1nF.
 4. Saldare i **Pin Maschi** rivolti verso l'alto per l'Arduino Mega (per tenerli allineati si consiglia di farlo tenendo i pin maschi inseriti dentro i pin femmina nel mega, quindi levarlo se scomodo nei passaggi successivi).
 5. Saldare i pin Maschi sul jumper “AREF-JP” rivolgendoli verso l'alto.
 6. Saldare i pin Maschi sui 6 Driver motori rivolgendo i maschi opposti al chip.
 7. Saldare i **Pin femmine** rivolte verso l'alto nelle piazzole dei 6 Driver Motori (per tenerli allineati si consiglia di farlo tenendo i pin femmina inseriti dentro i pin maschi dei driver, quindi levarli se scomodo nei passaggi successivi).
 8. Saldare i Trimmer “R-REF” e *se necessario* il “V-OFF” e il loro condensatore da 1 nF. ([vedi qui](#))
 9. Saldare il connettore per il RASPBERRY-PY verso l'alto e con la tacca rivolta verso l'esterno.
 10. Saldare il Led (GND in alto, problemi con la serigrafia)
 11. Saldare lo step-down in uso mettendo **femmine sulla board** rivolte verso l'alto e i maschi sullo step-down (per tenerli allineati si consiglia di farlo tenendo i pin maschi inseriti dentro i pin femmina, quindi rimuovere per semplificare i passaggi successivi)
 12. Saldare i connettori a vite.
 13. Saldare il connettore D-SUB50 rivolto verso l'alto.
 14. Saldare condensatore da 1mF.
- 15. Inserire nuovamente lo step-down quindi:**
- a. Posizionando i puntali di un voltmetro sopra i Pad di test accanto tarare l'uscita dello step-down a esattamente 5V, o comunque nei margini dichiarati nella [Scheda tecnica](#).
 - b. Eseguita la taratura si saldi il Pad smd accanto al led connettendo i 2 estremi.
 - c. Ora la parte logica della scheda riceve l'alimentazione.
16. Inserire, se tolti, l'Arduino mega e i driver motori.

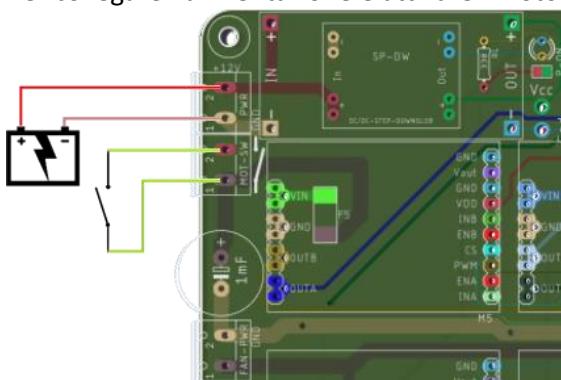
Ad assemblaggio completo, il risultato dovrebbe essere:

[Foto scheda saldata]

Collegamento board al sistema

Alimentazione standard

Per collegare l'alimentazione e attivare i motori è necessario collegare ai connettori indicati l'alimentazione e un interruttore.

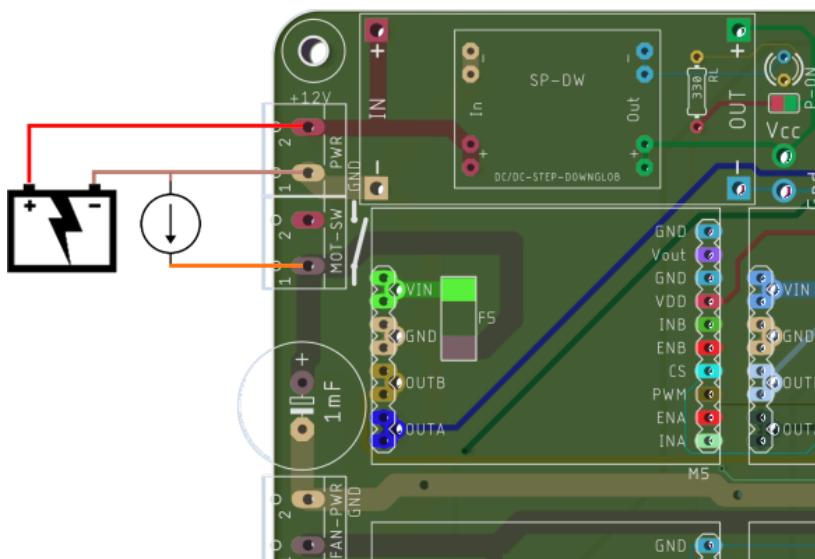


L'interruttore permette di attivare o disattivare manualmente l'alimentazione ai motori, da parte dell'operatore.

La batteria può essere una qualsiasi purché rispetti le caratteristiche presenti nella [Scheda tecnica](#)

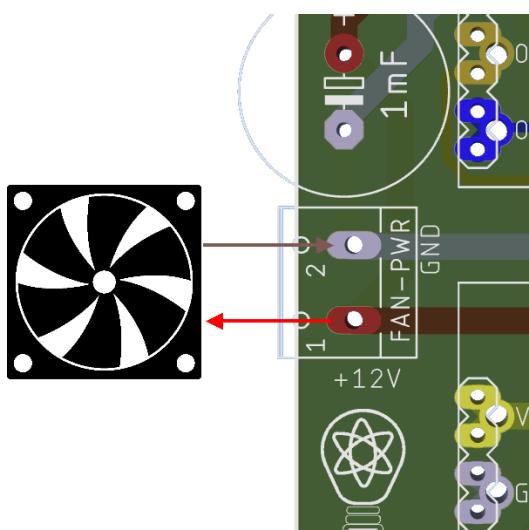
Alimentazione motori indipendente

È altresì possibile collegare al posto dello switch, che mette in contatto i 12V della batteria con i motori, un secondo alimentatore parallelo e con diverse caratteristiche elettriche, stando attenti a non superare i parametri presenti nella [Scheda tecnica](#) nella sezione Motor.



È importante collegare insieme le 2 GND!!!

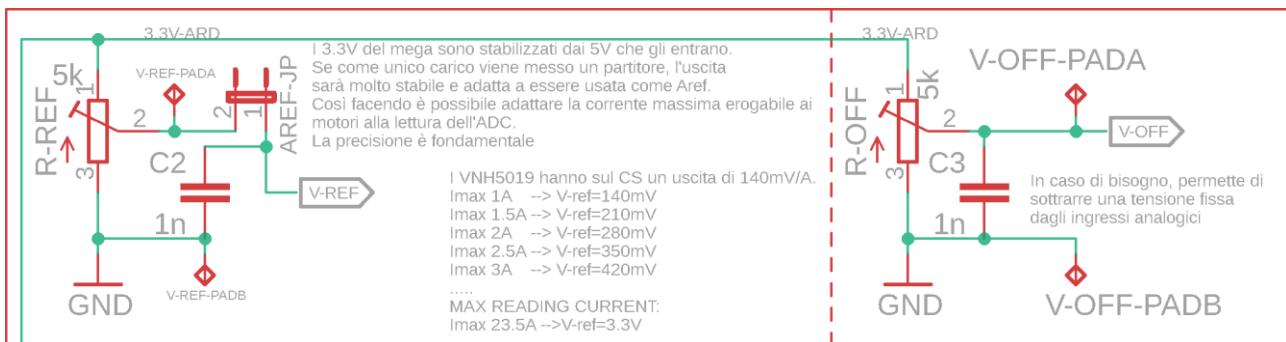
Ventola di raffreddamento



In caso risulti necessario collegare una ventola per raffreddare la scheda, è stata predisposto un collegamento alimentato con la tensione che alimenta i motori.

Taratura

La scheda è praticamente plug-and-play, unico aspetto che necessita di una taratura sono i riferimenti per la lettura della corrente dei motori.



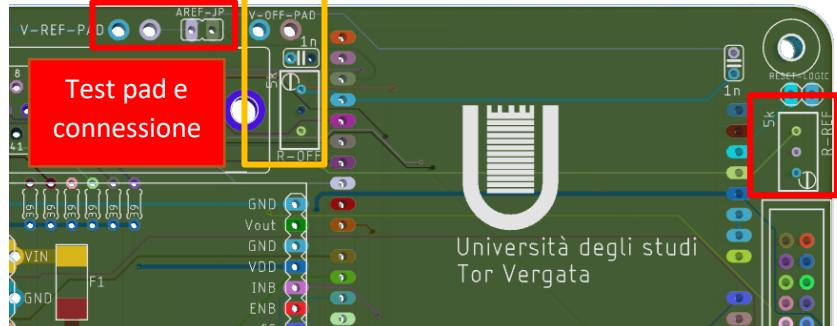
Trimmer R-REF

Il trimmer (**Rosso**) regola la tensione di riferimento dell'ADC se si usa il riferimento esterno ([maggiori info](#)). In base al voltaggio impostato nel trimmer, si imposta la sensibilità dell'ADC. Per stabilire la tensione di cui si ha bisogno bisogna stabilire la massima corrente che si vuole leggere, quindi con questa formula si ottiene il valore voluto del V-REF:

$$V_{ref\ need} = I_{max} * V_{cs}$$

$$I_{read} = \frac{V_{ref} \cdot ADC}{1024 \cdot V_{cs}}$$

$$ADC = \frac{I_{read} \cdot 1024 \cdot V_{cs}}{V_{ref}}$$



Dove:

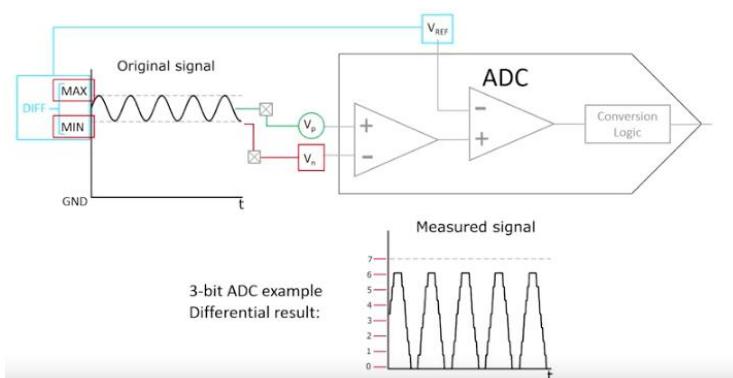
- V_{cs} è la sensibilità di uscita del VNH5019 del pin CS che è pari a **0,140 $\frac{V}{A}$** medi, per un calcolo esatto vedere la sezione del [Driver Motori](#).
- I_{max} è la massima corrente che si vuole leggere
- ADC è il numero da $0 \div 1023$ letto dal mega
- 1024 è la sensibilità dell'ADC che è da 10bit

N.B. La massima corrente rilevabile è di 23.5A, la quale però è ben oltre i limiti progettuali della scheda.

Trimmer R-OFF (facoltativo)

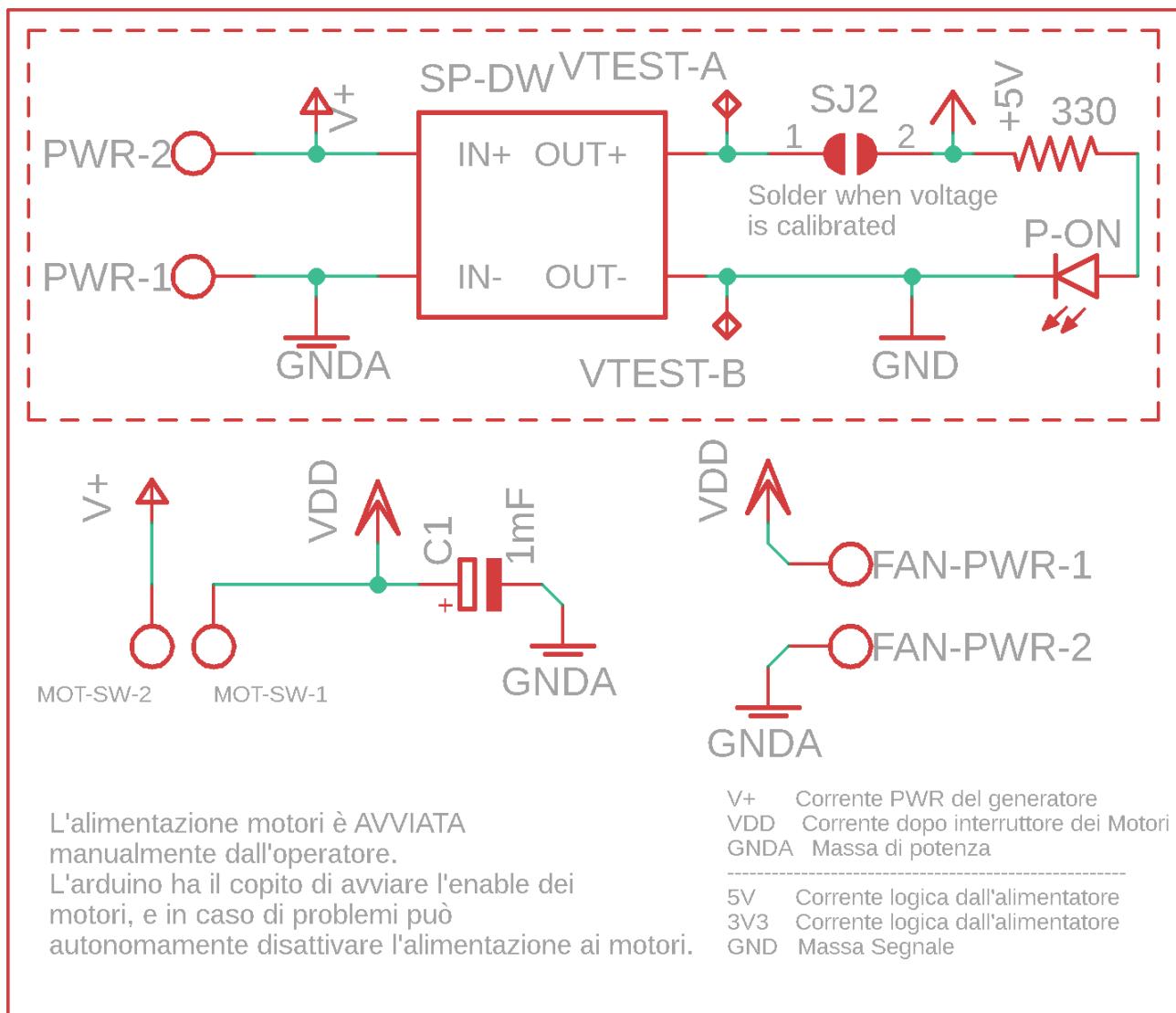
Il trimmer (**Giallo**) permette di sottrarre un errore di tensione analogicamente, preservando così la sensibilità dell'ADC. L'opzione **DEVE** essere attivato via software, e qualora non sia necessario si può non saldare.

Il trimmer permette di scegliere la soglia di sottrazione, così da riportare in maniera «Virtuale» a 0 la tensione base, annullando così questo errore costante di lettura e non facendo perdere scala di misura.



Documentazione Elettronica

Schema dell'Alimentazione



La scheda viene alimentata alla tensione operativa dei motori (che deve comunque essere superiore a 7V), si consigliano 12V.

L'alimentazione arriva allo step-down il quale, dopo la taratura, ha saldato il Pad SJ2, da questo momento la parte logica risulta essere alimentata e per evidenziare ciò, un led si accenderà.

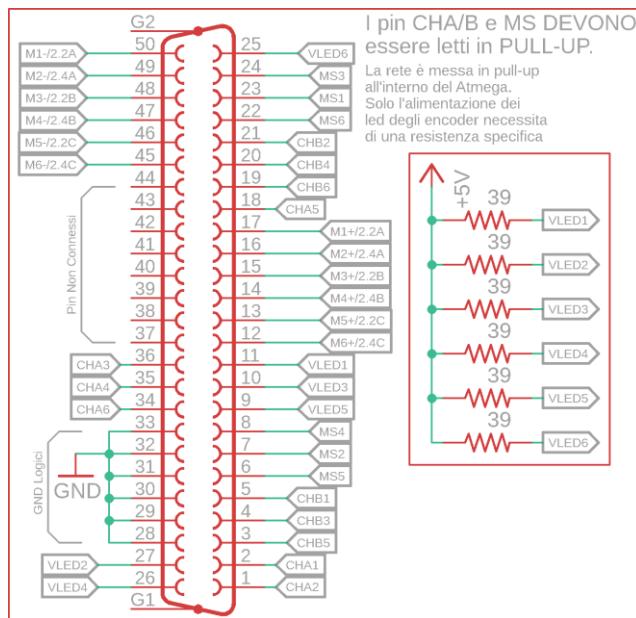
L'alimentazione ai motori avviene per mezzo di un connettore nel quale si può:

- Inserire uno switch e alimentare i motori con lo stesso alimentatore della logica, dando all'operatore la possibilità di scollegare la corrente qualora ce ne fosse necessità. ([schema](#))
- Collegare un secondo alimentatore esclusivo per i motori. ([schema](#))

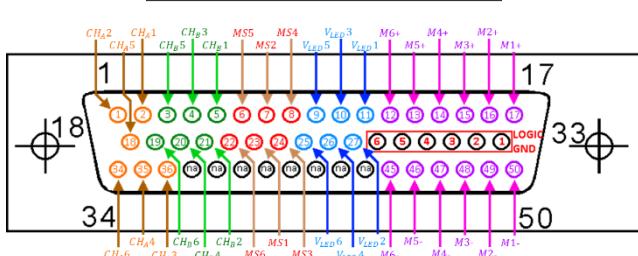
È predisposto un attacco comodo per una possibile ventola alimentata dall'alimentatore motori.

La scheda è progettata per poter erogare contemporaneamente ai vari motori fino a 10A senza rotture e in sicurezza, per maggiori dettagli sulle specifiche elettriche consultare la [Scheda tecnica](#).

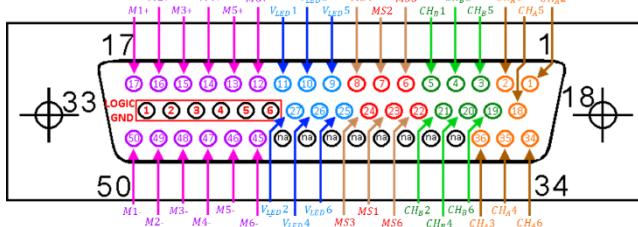
Connettore Scrbot



Frontale Connettore Maschio

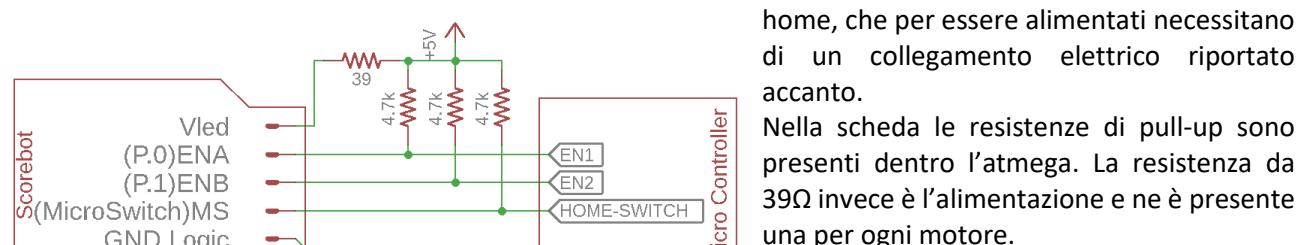


Top Board Pin Out



Un connettore abbastanza standard per gli Scrbot è il D-SUB-50, di cui qui è riportata la connessione dei pin sul connettore e sulla board.

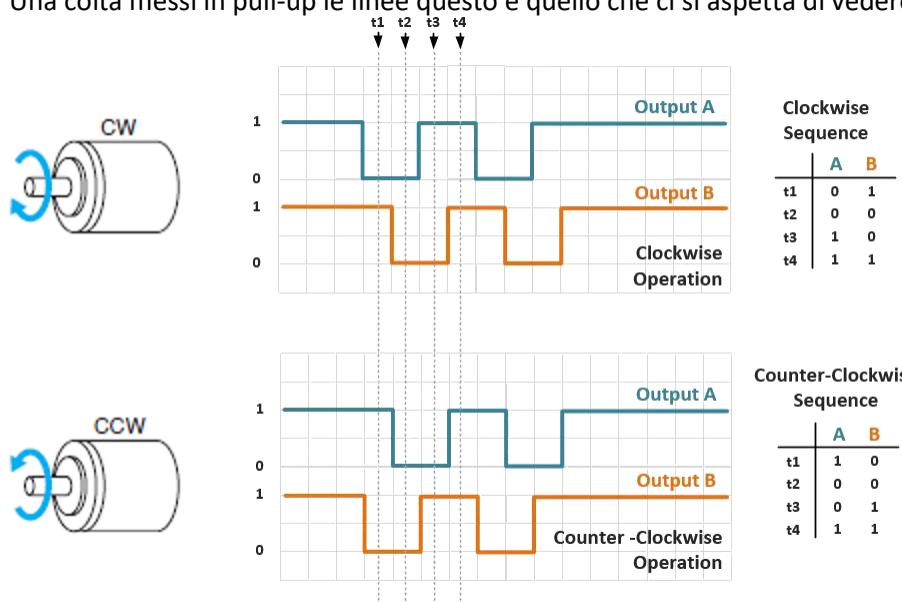
I sensori montati sopra ogni motore dello Scrbot sono encoder ottici e micro-switch per la ricerca della home, che per essere alimentati necessitano di un collegamento elettrico riportato accanto.



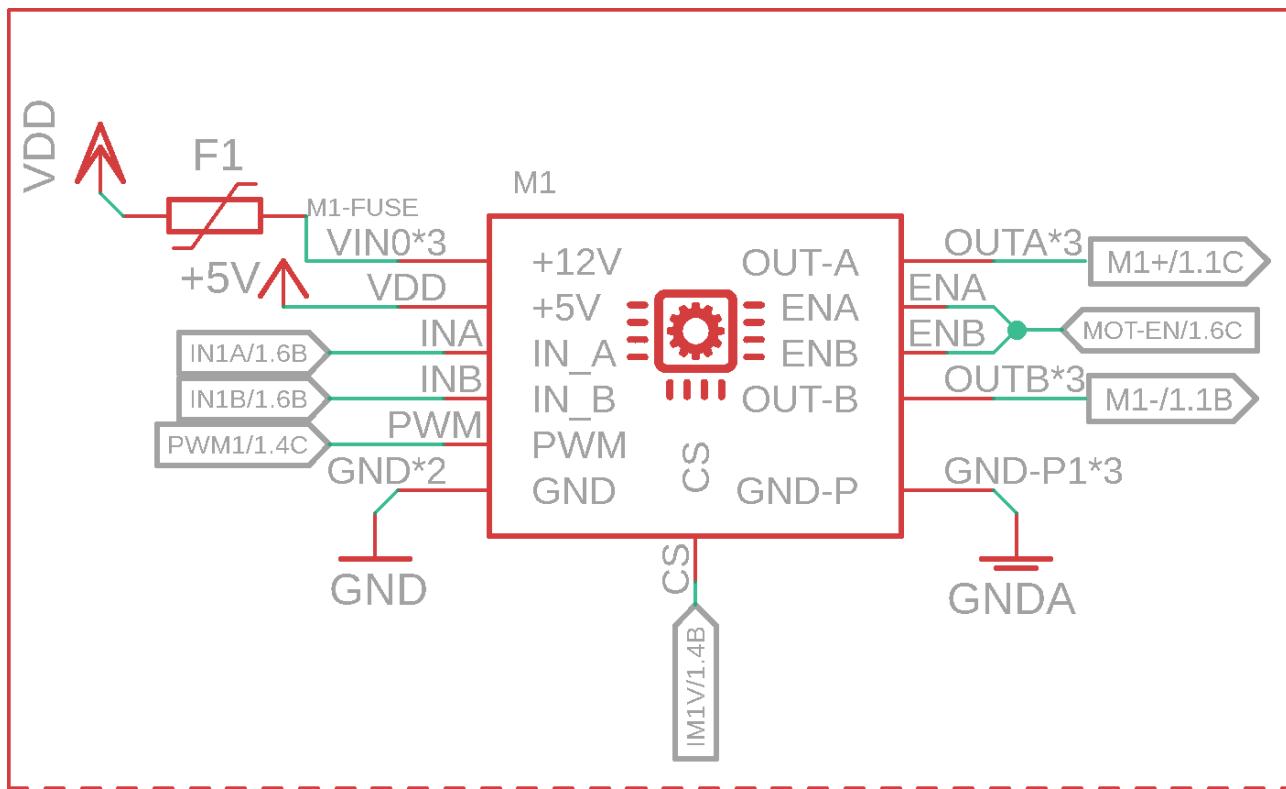
Nella scheda le resistenze di pull-up sono presenti dentro l'atmega. La resistenza da 39Ω invece è l'alimentazione e ne è presente una per ogni motore.

Maggiori informazioni si possono reperire dal datasheet dello Scrbot in uso nel proprio caso.

Una colta messi in pull-up le linee questo è quello che ci si aspetta di vedere:



Driver Motori VNH5019



Per controllare ogni motore i usiamo i driver Pololu i "VNH5019".

Questi drive sono dei ponti ad H molto potenti, sono infatti originariamente progettati per controllare i servomotori dentro le automobili, infatti ciascuno dei driver è in grado di controllare motori fino a 41V e 30A. Ogni driver motori è però protetto da un fusibile da 3A che ne limita la massima corrente per mantenerlo in sicurezza.

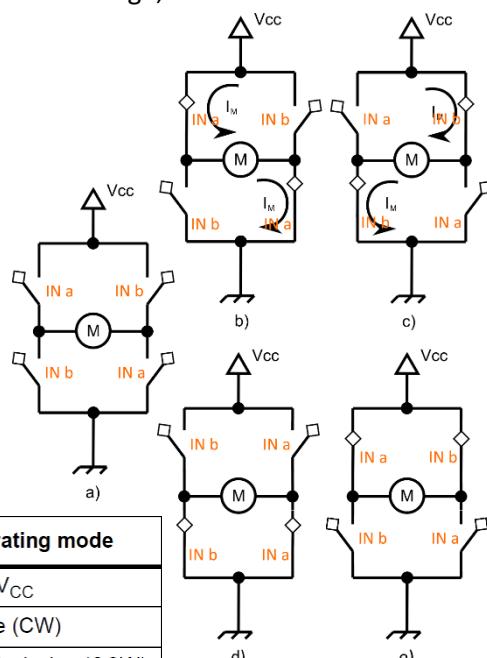
I driver hanno bisogno di 2 alimentazioni differenti, la prima è quella di controllo e va da 5.5V a 24V, la seconda è quella dei motori e può gestire costantemente fino a 12A e per alcuni periodi può arrivare a picchi di 30A.

Sono presenti 2 pin di enable per disattivare selettivamente i 2 half-bridge, che nel nostro caso sono comandati insieme per dare il controllo al Mega di disattivare la board, qualora fosse rilevato un malfunzionamento.

Configurazioni del ponte

Le possibili configurazioni del ponte sono:

- freeRun (a)
- drive_motor (b & c)
- soft_stop (d)
- hard_stop (e)



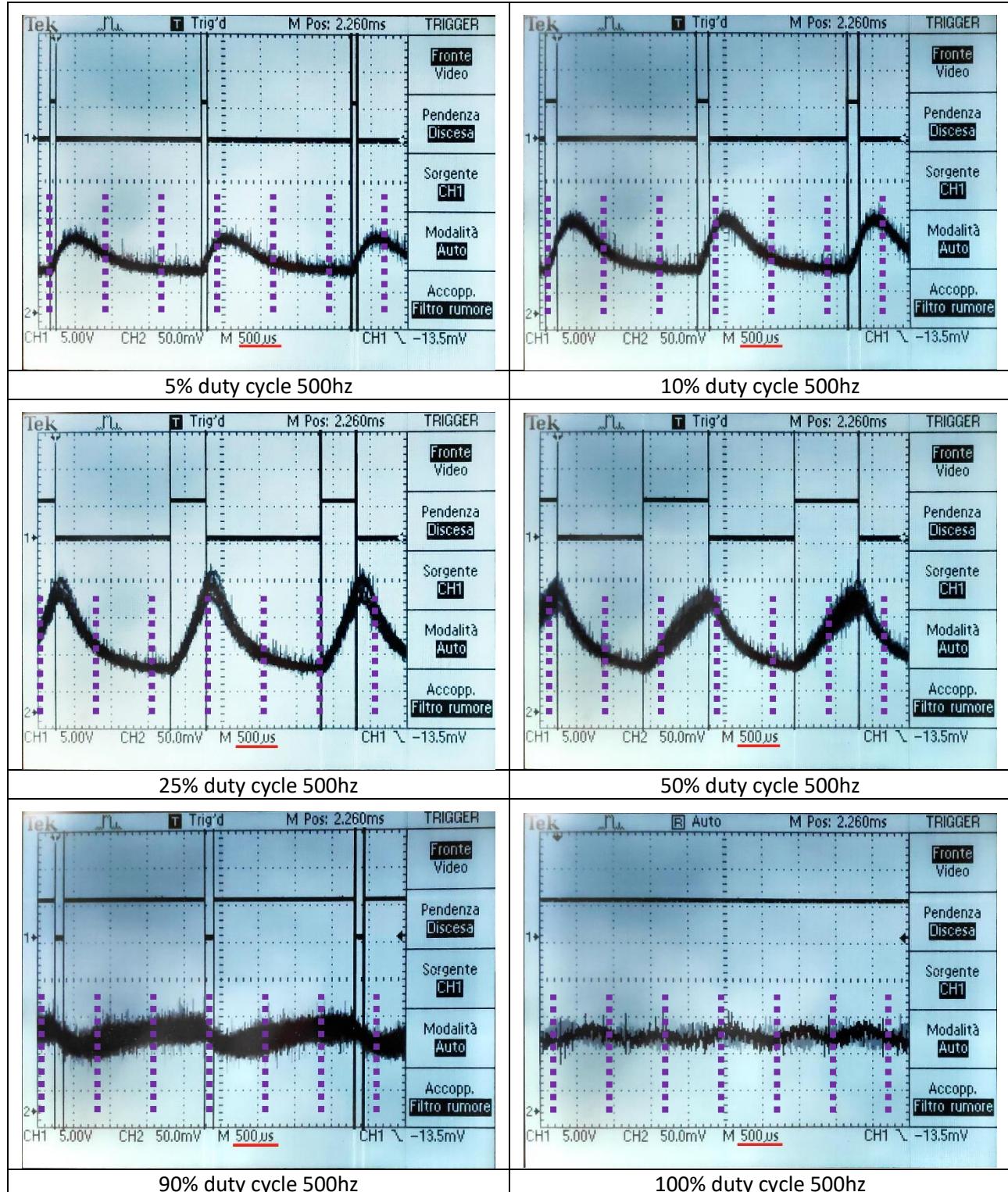
IN _A	IN _B	DIAG _A /EN _A	DIAG _B /EN _B	OUT _A	OUT _B	CS	Operating mode
1	1	1	1	H	H	High Imp.	Brake to V _{CC}
	0				L	I _{SENSE} = I _{OUT} /K	Clockwise (CW) Counterclockwise (CCW)
0	1			L	H		
	0				L	High Imp.	Brake to GND

Lettura corrente motori

Qui di seguito non riportate le risposte di un motore con PWM variato in Duty-Cycle e anche in frequenza. Sul **canale 1** si può vedere il segnale di controllo, sul **canale 2** la tensione letta dal pin CS.

In **Viola** esempi di distribuzione di campionamento.

Frequenza 500hz (default arduino)

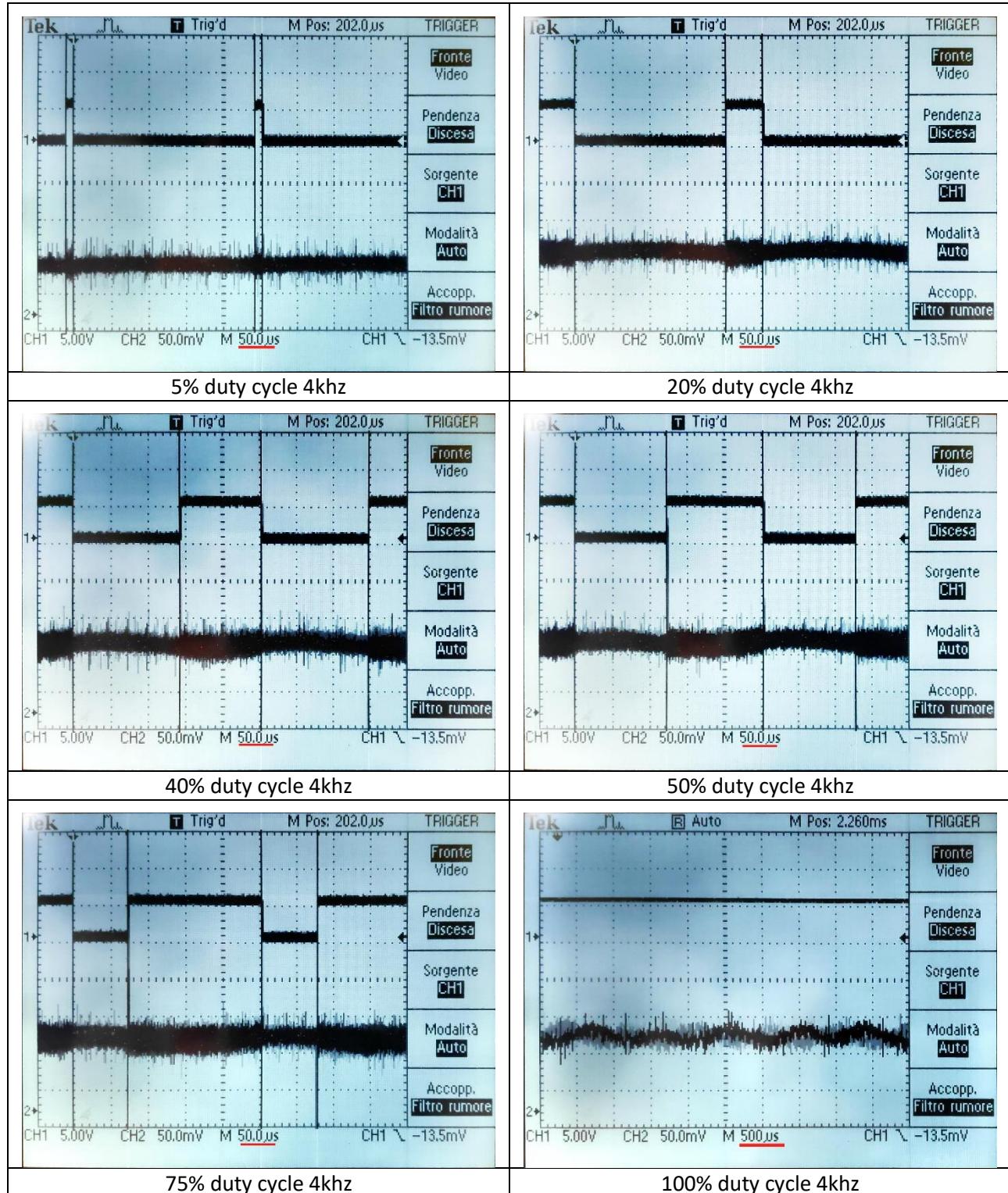


Essendo la durata di lettura di un motore da parte dell'Arduino di 112us ~ 8,9Khz.

E contando che li deve leggere ciclicamente tutti e 6, il tempo di scansione è di 672us ~ 1,5Khz. ([dettagli](#))

È evidente che le letture possono avvenire un po' ovunque, e in questa configurazione la lettura della corrente può non essere molto affidabile, quindi sconsigliata.

Frequenza 4khz (default board)



Essendo la durata di lettura di un motore da parte dell'Arduino di 112us ~ 8,9Khz.

E contando che li deve leggere ciclicamente tutti e 6, il tempo di scansione è di 672us ~ 1,5Khz. ([dettagli](#))

A questa frequenza tra una campionatura e l'altra il PWM ha già esegue almeno 3 Cicli, a questa frequenza la corrente si avvicina a essere una linea retta rendendo ampiamente trascurabile l'effetto carica-scarica dell'induttore, a questa frequenza di lavoro la lettura è afflitta molto più dagli errori di misura dello strumento, che comunque si mantengono in una banda di 50 mV (~±300mA di sensibilità in lettura).

Vista la stabilità in lettura con questo PWM, questa è la frequenza impostata di default sulla scheda.

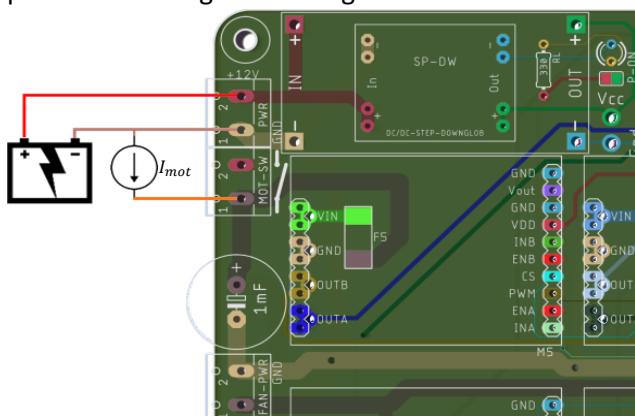
Misura sensibilità di uscita

Il chip VNH2SP30 genera in uscita un flusso di corrente proporzionale a quello che scorre dentro i motori, di seguito le caratteristiche descritte nel datasheet.

Symbol	Parameter	Test conditions	Min	Typ	Max	Unit
K ₁	I _{OUT} /I _{SENSE}	I _{OUT} = 30A; R _{SENSE} = 1.5kΩ; T _j = -40 to 150°C	9665	11370	13075	
K ₂	I _{OUT} /I _{SENSE}	I _{OUT} = 8A; R _{SENSE} = 1.5kΩ; T _j = -40 to 150°C	9096	11370	13644	
dK ₁ / K ₁ ⁽¹⁾	Analog sense current drift	I _{OUT} = 30A; R _{SENSE} = 1.5kΩ; T _j = -40 to 150°C	-8		+8	%
dK ₂ / K ₂ ⁽¹⁾	Analog sense current drift	I _{OUT} > 8A; R _{SENSE} = 1.5kΩ; T _j = -40 to 150°C	-10		+10	
I _{SENSEO}	Analog sense leakage current	I _{OUT} = 0A; V _{SENSE} = 0V; T _j = -40 to 150°C	0		65	µA

All'aumentare della corrente migliora la sensibilità che in generale non è lineare, ma ci si avvicina molto.

La board del VHN5019 in uscita mette una resistenza di caduta e un filtro passa basso da 10khz, e loro dichiarano una uscita pari a $0,140 \frac{V}{A}$, se però si è intenzionati a misurare con esattezza questo valore la procedura da seguire è la seguente:

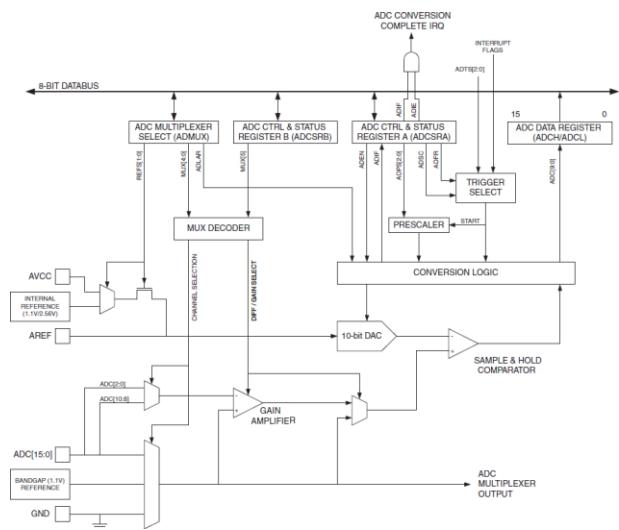


1. Collegare l'alimentazione dei motori a un generatore di corrente controllato come in figura (se collegata scollegare la ventola)

2. Da programma impostare Fisso il PWM al 100% del driver motore che si vuole misurare e allo 0% il PWM dei restanti, infatti con il circuito descritto sopra, il 100% della corrente erogata dal dall'alimentatore del motore verrà riversata dentro al driver motori (meno qualche milli-ampere dei led del VNH5019), mentre ad alimentare la logica ci penserà la batteria principale.
3. Iniziare a far aumentare a step la corrente I_{mot} e leggere la tensione di uscita, o con l'ADC interno dell'Arduino (si consiglia di impostare come riferimento la tensione a 1.1V interni, essendo molto più stabile e certa), o con un Voltmetro posizionando i puntali tra CS e GND.
Se anche il motore fosse bloccato non sarebbe un problema poiché si usa un generatore di corrente.
4. Far variare la corrente lentamente e segnare i vari valori di tensione letti e le corrispettive correnti. Attenzione a non oltrepassare il limite del fusibile (3A di default)
5. Al termine di ciò si può fare un grafico con sulle **X le Correnti** e le **Y i Voltaggi**, ed ecco la risposta di uno specifico Driver Motori

N.B. La risposta letta è specifica del driver motore utilizzato, ogni chip VNH2SP30 ha una risposta **UNICA**

ADC ATMEGA



Caratteristiche Principali:

- **10-bit Resolution**
- **13µs - 260µs Conversion Time**
- **16 Multiplexed Single Ended Input Channels**
- **14 Differential input channels**
- **0V - VCC ADC Input Voltage Range**
- **2.7V - VCC Differential ADC Voltage Range**
- **Selectable 2.56V or 1.1V ADC Reference Voltage**
- **Free Running or Single Conversion Mode**
- **Interrupt on ADC Conversion Complete**
- **Sleep Mode Noise Canceler**

Per rendere la scheda il più flessibile possibile sono stati predisposti i 2 trimmer di taratura.

Registri interni default

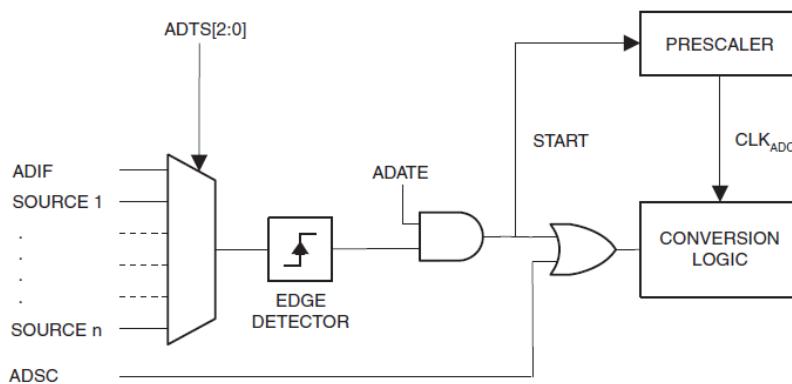
I registri interni sono impostati con le seguenti caratteristiche:

- Lettura dai Pin da A0:A6 per input analogico
- Modalità di free running
Ovvero, finita una conversione inizia immediatamente la successiva, minimizzando così i tempi
- Prescaler impostato a 1/128 Clock
Al fine di dare 125Khz di clock all'ADC, da datasheet deve essere tra 50khz e 200khz
- Interrupt catch
Per leggere i valori letti solo a lettura terminata e lasciando il tempo di CPU ad altre Routine.

Parametri programmabili

È inoltre possibile selezionare come modalità di funzionamento:

- **Lettura diretta / Lettura differenziale**
 - ▶ **Diretta:** legge il valore mettendo lo 0 a GND (default)
 - ▶ **Differenziale:** legge il valore sottraendo la tensione V-OFF del trimmer R-OFF come visto nella [Taratura](#)
- **Vref 2.56V / 1.1V / Aref esterna**
 - ▶ Aref esterna: Riferimento selezionabile dal trimmer R-REF come visto in [Taratura](#) (default)
 - ▶ Vref 2.56V e 1.1V: sono dei riferimenti interni al chip e sono fissi e molto stabili

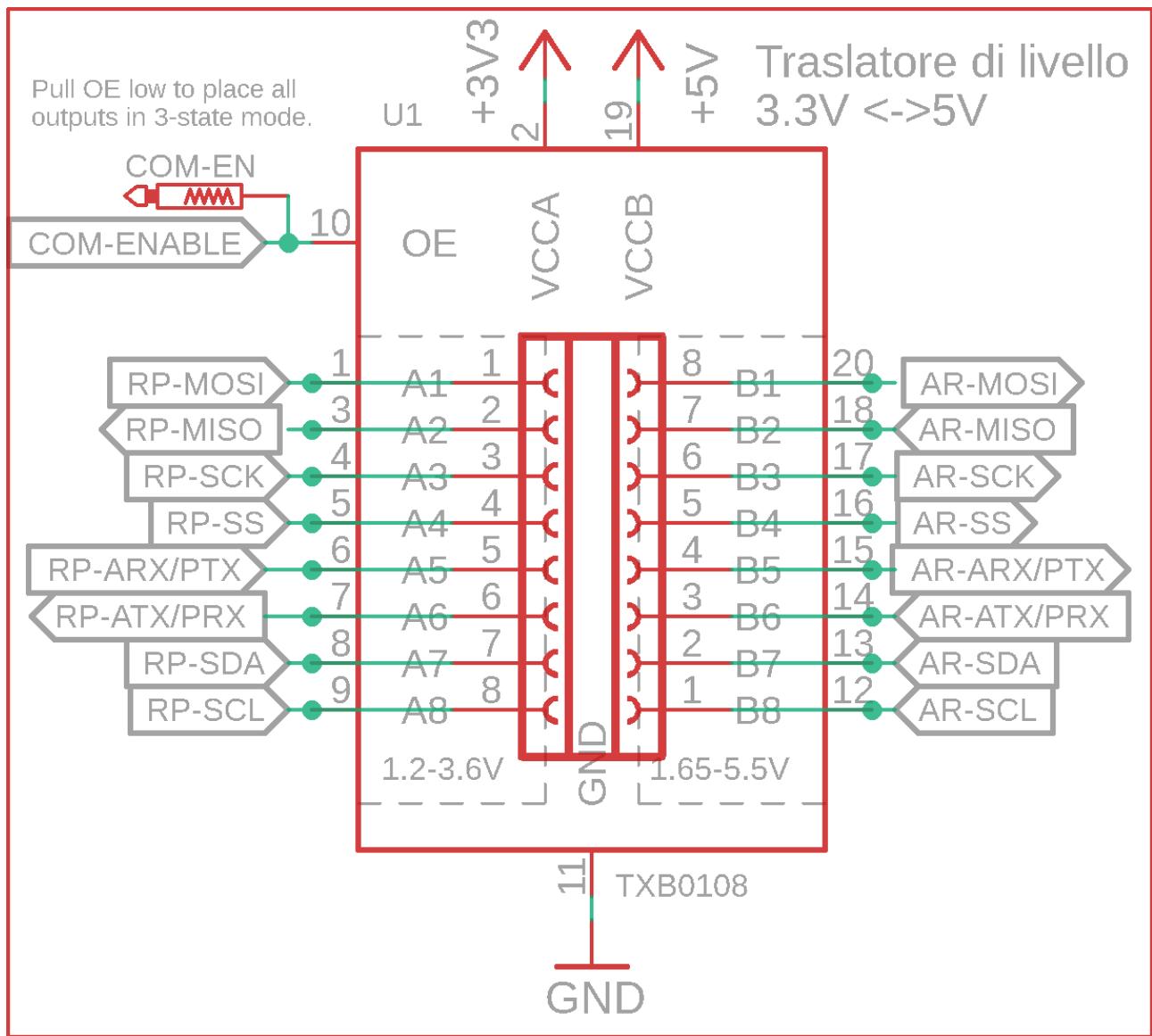


Affidabilità tensioni di riferimento

La tensione di riferimento AREF ha la **Necessità** di essere molto stabile, a questo scopo entrambi i trimmer sono alimentati ESCLUSIVAMENTE dai 3.3V generati internamente da uno stabilizzatore lineare, il quale elimina ogni possibile fruscio di fondo dato dallo step-down e rende il valore selezionato stabile e robusto.

Traslatore di livello

La scheda è stata pensata per essere l'attuatore del sistema, mentre i calcoli e i controlli vengono gestiti dal RASPBERRY-PY, ma i 2 lavorano a 2 diverse tensioni logiche, in generale a queste distanze non dovrebbero essere presenti problemi di disturbo, ma per garantire una sicura comunicazione si è deciso di aggiungere un traslatore di livello bidirezionale.



Il traslatore in questione è il “TXB0180”, per semplificare il debug o un eventuale bypass sono stati disposti dei pin a passo normale per entrambe le file.

L'attivazione del chip è controllata dal RASPBERRY-PY, per garantire che se i segnali arrivano è perché sono voluti.

I 3.3V sono presi direttamente dall'uscita del Raspberry-Py, così da garantire la massima compatibilità e sicurezza nelle tensioni.

L'iniziale RP sta per RASPBERRY-PY, il lato A è collegato al connettore del RASPBERRY-PY infatti.

L'iniziale AR sta per ARDUINO, il lato B è collegato ai pin dell'Arduino infatti.

Di default è in uso la comunicazione SPI (MISO, MOSI, SCK, SS), dove il RASPBERRY-PY è Master e l'Arduino Slave, ma sono state predisposte anche altre comunicazione in caso di sviluppi futuri.

Firmware (Arduino)

Architettura di Sistema

Il firmware è stato scritto in C++, usando come editor il software open-source [SLOEBER](#), ed i sorgenti scritti sono disponibili sul [repository online](#).

Il programma è strutturato con:

- Logica procedurale per l'analisi dati acquisiti.
- Gestione ad eventi (interrupt) per l'acquisizione dei dati dalle periferiche.
- Gestione a eventi per la comunicazione e attuazione dei pacchetti.

Tenendo bene a mente che l'Arduino è mono-core, una simile struttura permette di essere tempestivi e di non perdere nulla dell'evoluzione del sistema, rendendolo un rudimentale sistema "real-time" a eventi.

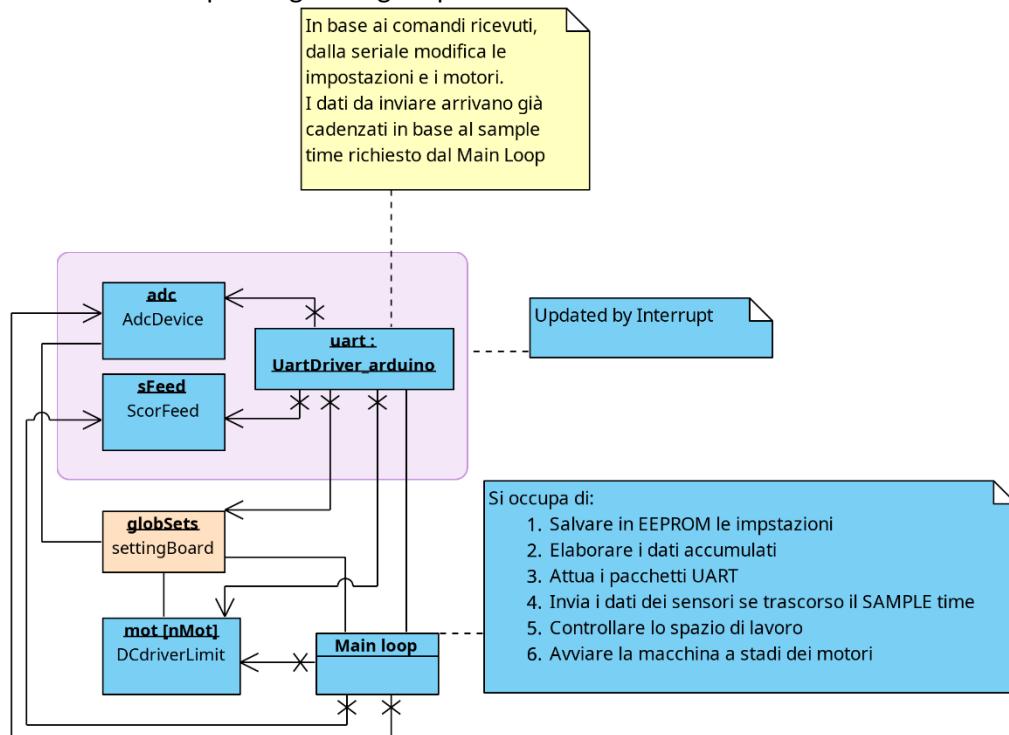
Principio di funzionamento Software

Il Main principale viene seguito sempre e a loop legge di dati che si sono accumulati, li elabora e salva i risultati quindi procede al sistema successivo.

Tramite Interrupt vengono gestite le periferiche di acquisizione dati quali:

- 1) UART (Seriale)
- 2) ADC
- 3) Encoder

Gli oggetti che hanno il compito di gestire gli aspetti del sistema sono:



Il Main loop è una istanza fittizia, in realtà è il loop dell'Arduino, ovvero il while(true) infinito di un Main.

La descrizione delle classi e di come svolgono il loro compito è descritta nei seguenti capitoli e approfondisce gli aspetti inerenti a:

- [Controllo motori](#)
- [Lettura delle correnti](#)
- [Lettura degli encoder & micro-switch](#)
- [Procedura di Discovery Home](#) (Homing)

Per vedere più nel dettaglio l'implementazione di queste specifiche si consiglia vivamente di andare a studiare nel dettaglio il codice.

Controllo motori

Per controllare i motori si è progettata un sistema apposito, per permettere di aggiungere comportamenti dinamici e avanzati. In particolare, i Task chiave di questo Sistema sono:

1. Mette a disposizione dei metodi di controllo dei motori temporizzati
2. Limitare il range di movimento leggendo gli encoder (così da evitare danni da urti).

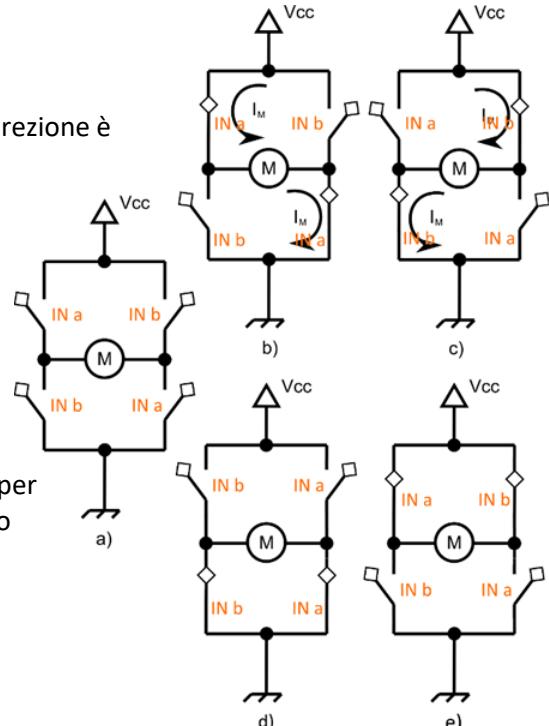
Modalità di controllo supportate

Grazie alla temporizzazione è possibile garantire che in caso di disconnessioni dal controllo i motori non restino sempre in tensione causando urti e possibili danni.

Al contrario, se non viene ricevuto un comando entro 2 secondi (di default), in automatico vengono spenti i motori.

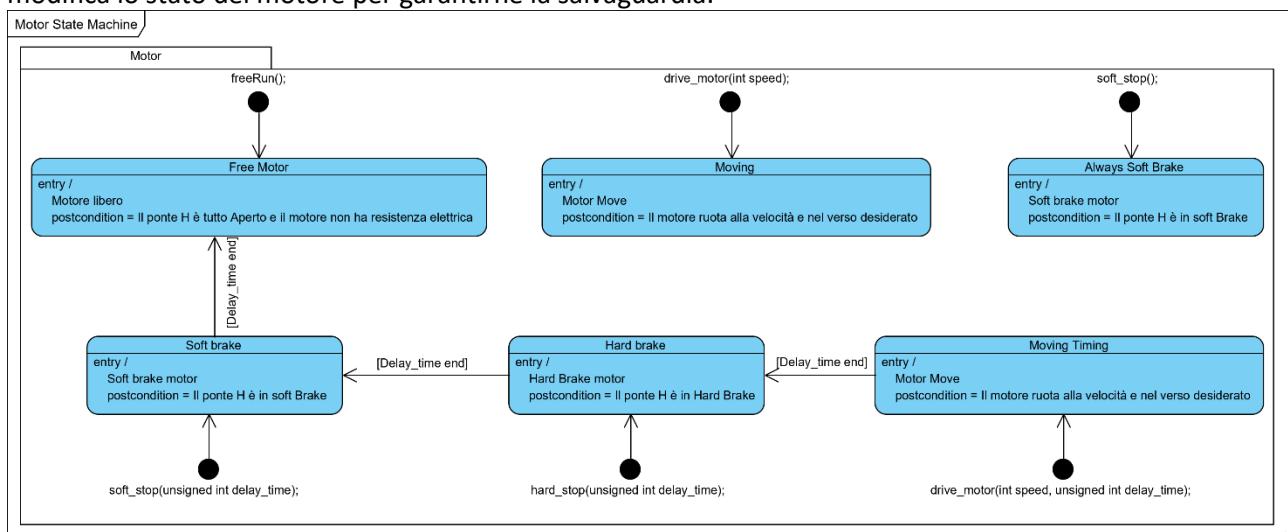
Modalità di controllo dei motori sono:

- **drive_motor (int speed);**
[b & c] Controllo normale del motore Destra-Sinistra, la direzione è data dal segno della velocità
- **soft_stop ();**
[d] Cortocircuita il motore su se stesso a Massa, si ottiene un effetto frenante finché il motore è in movimento (causato dalle correnti indotte del motore)
- **hard_stop (unsigned int delay_time);**
[e] Cortocircuita il motore su se stesso passando per VCC, ciò causa un effetto frenante molto più forte dovuto al contributo in corrente dell'alimentazione
Una volta fermo rimangono delle forti correnti parassite, per evitare danni quindi si deve specificare un tempo massimo oltre il quale, in caso di scollegamento dal controllo, in automatico il ponte si muova verso il soft_stop.
- **freeRun();**
Il motore è completamente sconnesso e libero.



Macchina a stadi automatica

Come accennato prima, se non vengono ricevuti nuovi comandi entro il time-out, in automatico la classe modifica lo stato del motore per garantirne la salvaguardia.

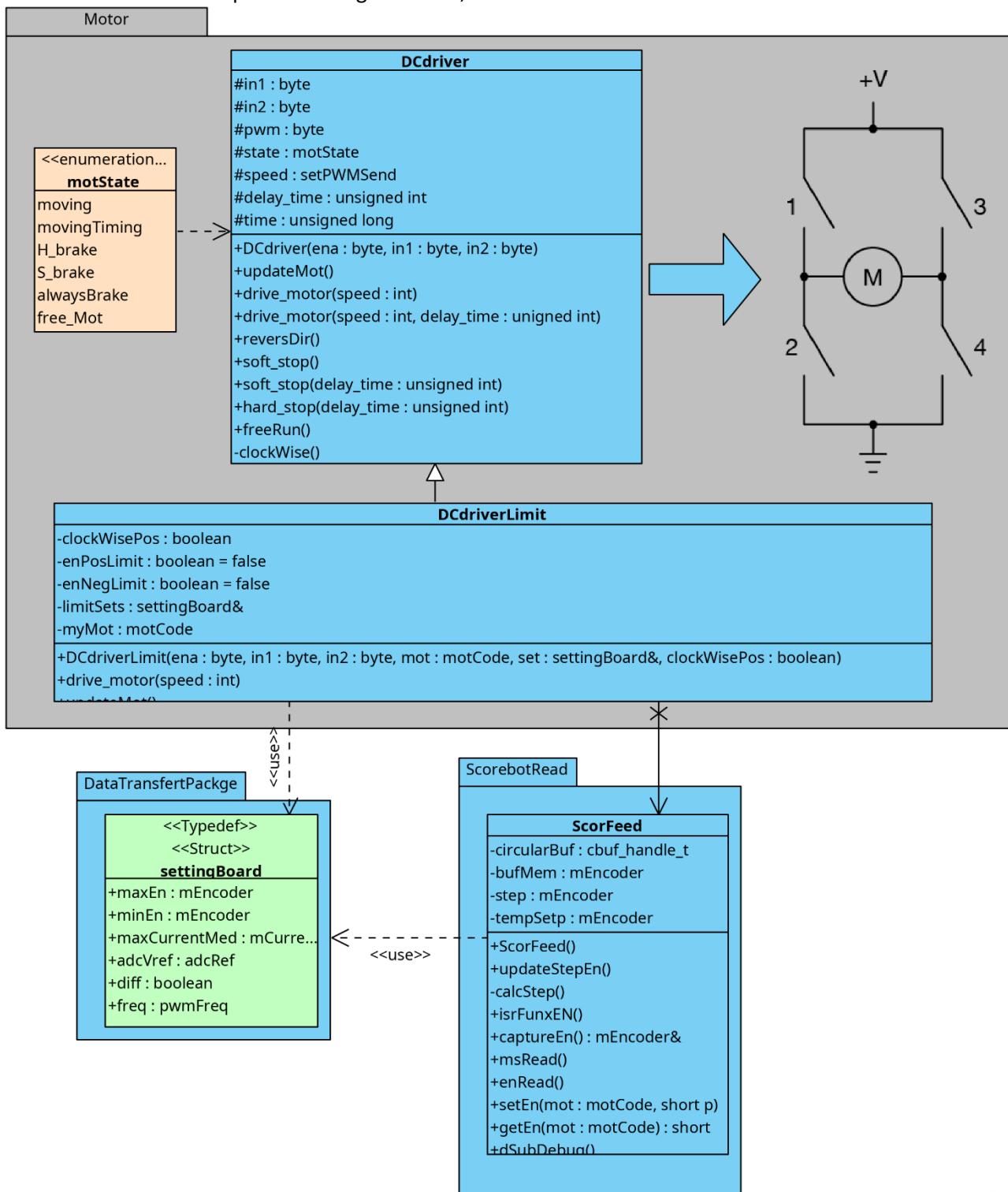


NOTA BENE

Da tenere ben a mente che il countdown è il tempo passato dall'ultimo ingresso in uno stato, quindi la macchina a stati entra in funzione solo se il time-out impostato arriva prima dell'arrivo di un nuovo comando. L'update della macchina a stati deve essere chiamato nel Main loop nei momenti liberi. Di conseguenza si garantisce il delay_time, ma non che al suo esatto termine la macchina evolva lo stato.

Sistem Motor

Quanto descritto sopra è Codificato all'interno della classe “**DCdriver**”, la classe “**DCdriverLimit**” aggiunge le funzionalità di sicurezza per i limiti degli encoder, di fatto è la classe istanziata:

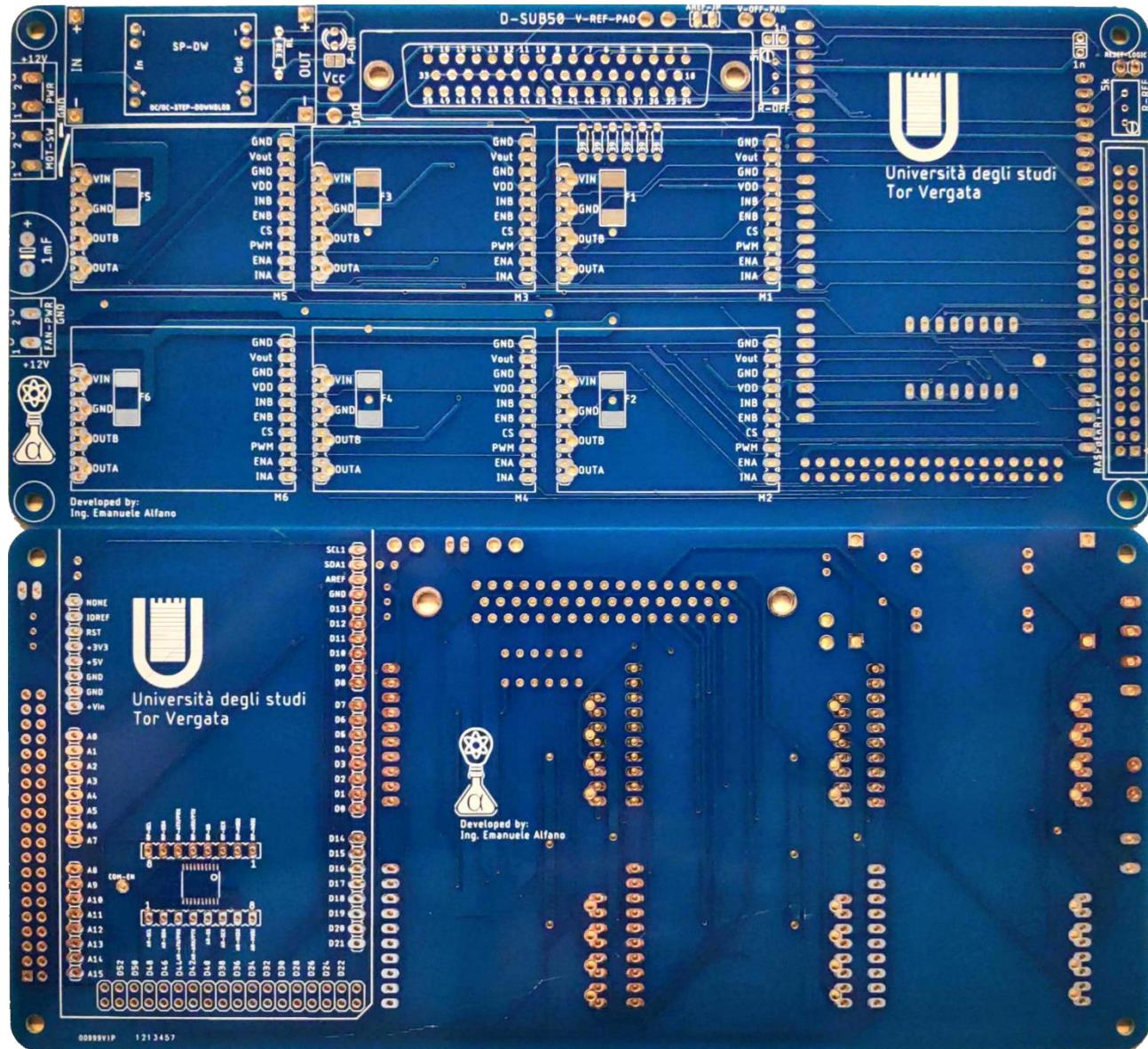


Ogni motore è un oggetto con un proprio stato, il metodo “`updateMot`” esegue l’update della macchina a stati.

Frequenza PWM

Esiste il metodo “`setMotFreq(pwmFreq freq)`” che imposta la frequenza di di conto dei Timer/Counter che muovono i motori. Per vedere come la frequenza influisce sulla corrente vedere sezione: [Lettura corrente](#)

motori.

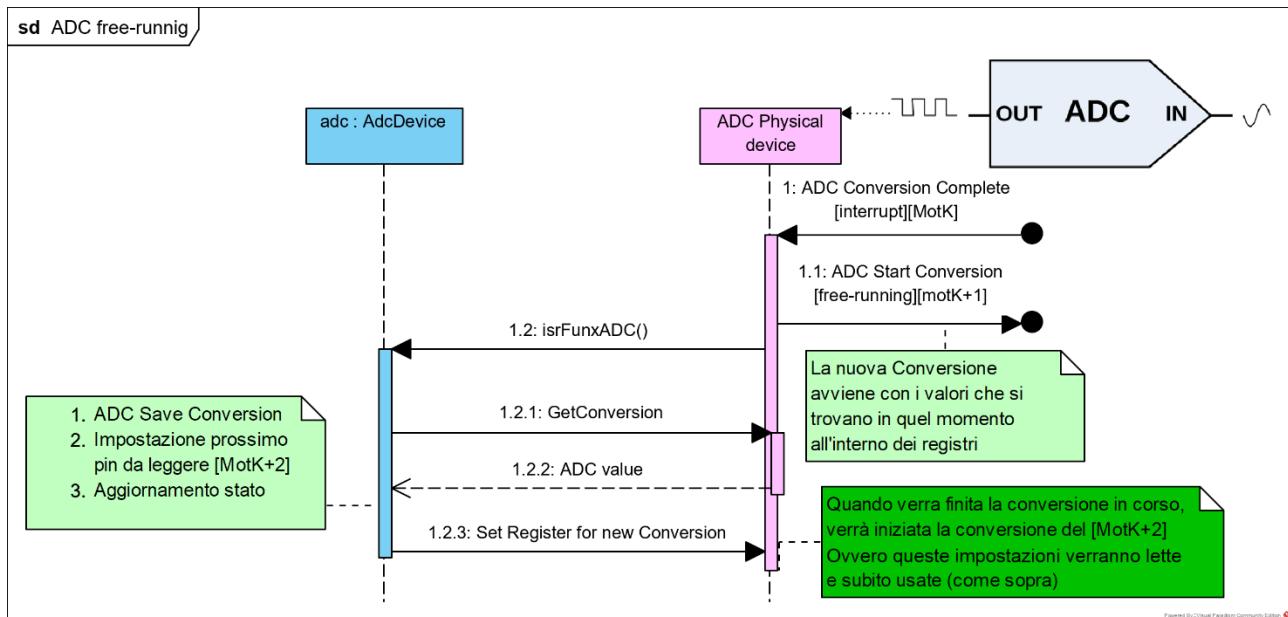


Lettura delle correnti

Come visto nella [sezione dedicata all'ADC](#), la periferica è in grado di generare degli interrupt ogni volta che viene terminata una campionatura. È selezionata la modalità il **free-running** per permettere che al termine di ogni conversione inizi immediatamente la successiva. La lettura e le impostazioni vengono gestite in interrupt. Tutti i dati raccolti vengono poi salvati e messi a disposizione del programma principale.

Free-running & Interrupt

RICORDA la modalità free-running inizia immediatamente una nuova scansione con l'ADC usando le impostazioni che si trovano all'interno dei registri quando viene generato l'interrupt.



Impostazioni per il V-REF

In base a come si impostano i registri è possibile scegliere una diversa sorgente per il V-REF dell'ADC:

Table 26-3. Voltage Reference Selections for ADC

REFS1	REFS0	Voltage Reference Selection ⁽¹⁾
0	0	AREF, Internal V _{REF} turned off
0	1	AVCC with external capacitor at AREF pin (Non supportata)
1	0	Internal 1.1V Voltage Reference with external capacitor at AREF pin
1	1	Internal 2.56V Voltage Reference with external capacitor at AREF pin

Tenendo presente la [procedura di taratura](#), Se si utilizza il trimmer si deve connettere il jumper "AREF-JP", se invece si intende usare il riferimento esterno il jumper deve essere rimosso.

- L'AREF esterno è stato progettato per essere il più stabile e privo di rumori possibile, maggiori dettagli nella [sezione dell'ADC](#).
- I riferimenti interni invece sono garantiti stabili e protetti da aumenti di temperatura, oscillazioni nell'alimentazione ecc... dalla Atmel.

Lettura Diretta/Differenziale

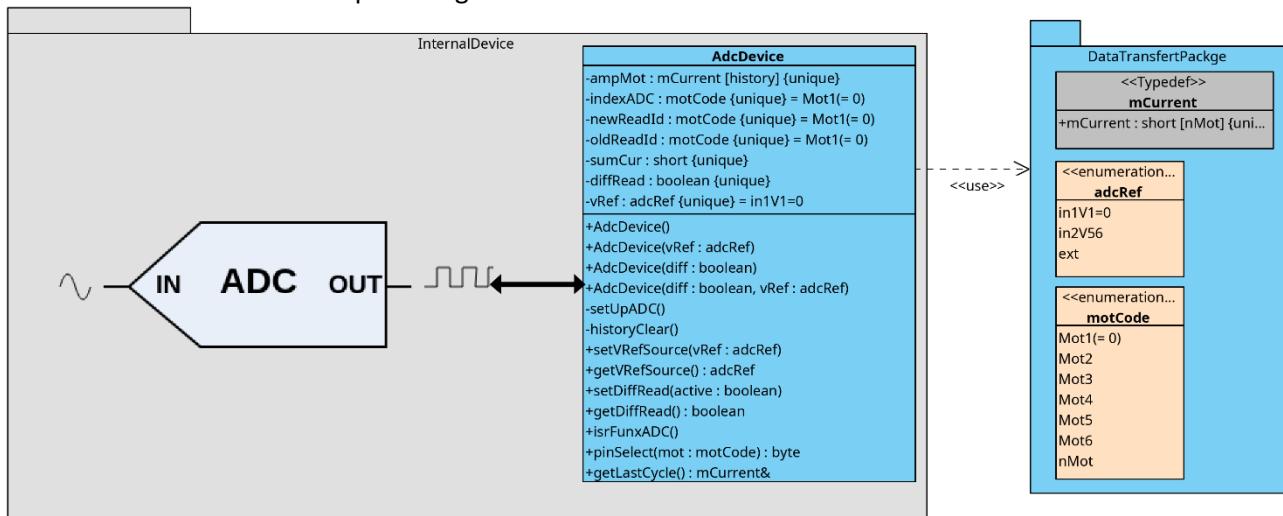
Di default la lettura avviene direttamente, è tuttavia possibile modificare l'impostazione e far eseguire una misura differenziale all'ADC. Il principio di funzionamento della lettura differenziale è spiegato nella [sezione taratura](#).

Calcoli e conversioni

A tal proposito si rimanda il lettore nuovamente alla sezione [taratura](#).

Class AdcDevice

La classe “**AdcDevice**” si occupa della gestione dell’ADC:



Come visto sopra la lettura avviene in interrupt, e chiedendo alla classe si può ottenere l’ultimo ciclo di lettura dei motori, la corrente media o la corrente di un motore specifico.

Viene tenuto uno storico degli ultimi 8 cicli di lettura per permettere un invio stabile e calcolare la corrente media.

Powered by Yed (version 3.15.2, build 3592)

Lettura degli encoder & micro-switch

Tutti i pin degli encoder raggiungono dei pin PCINT: **Pin Interrupt Change**, questi pin sono tutti di interrupt, e si attivano ogni volta che è rilevato un cambio di stato su di un pin, non essendo specifici sono raggruppati in porte da 8 bit nei quali se uno dei pin chiama interrupt viene chiamata sempre la stessa funzione.

Poiché però potrebbero avvenire spiacevoli coincidenze e far chiamare interrupt in maniera molto frequente senza però aggiornare realmente lo stato dei pin la lettura degli encoder è stata temporizzata avvenendo con una frequenza costante di 4Khz, ovvero ogni overflow del Timer/Counter 5.

Al momento dell'interrupt andiamo a leggere lo stato di tutti i pin degli encoder, e salviamo la configurazione dei pin in un *BUFFER CIRCOLARE* di short, dove ogni bit è in codifica “1 hot” con i pin degli encoder.

NOTA BENE, i valori non vengono elaborati immediatamente, bensì ricordati (in ordine) per essere analizzati in seguito dal Main loop (così da lasciare disponibile il tempo di CPU in caso di altri interrupt).

Il buffer Circolare

perché l'uso di un Buffer circolare? Perché non leggere e analizzare immediatamente il dato così da capire se l'encoder è cresciuto o decresciuto? Per rispondere a questa domanda è sufficiente ricordare che il programma ha altri compiti oltre la lettura degli encoder, compiti che richiedono dei timing molto pressanti, il sistema degli encoder invece necessita di tempestività solo quando viene rilevato uno step (per non perdere il passo), mentre per l'analisi dei dati è possibile analizzare in un secondo momento, quando la CPU non è occupata con carichi real-time.

A questo scopo si è previsto una memorizzazione a buffer circolare, la quale mantiene in memoria la configurazione delle porte, che possono essere lette e salvate in circa di 375ns (a 16Mhz Clock, 2 porte da leggere e salvare, 3 operazioni per ogni porta e refresh nella testa del buffer).

Successivamente i dati accumulati, vengono analizzati nel Main loop, regolarmente, mantenendo quindi il buffer scarico.

Volendo andare a fare i conti, vediamo il sistema come fosse una catena di produzione:

- Il servente (la CPU) per elaborare ogni cella impiega ~186 colpi di Clock (~80KHz).([algoritmo](#))
- Il setup è trascurabile (caricamento dei dati in memoria)
- Lo stesso servente ha altri compiti da svolgere che occupano approssimativamente il 10% del tempo.
- La dimensione del buffer è di 128 interi (2 byte)

L'elaborazione di ogni cella aggiorna lo stato di tutti e 6 gli Encoder.

Con questi dati possiamo dire che la massima frequenza di arrivo è lievemente inferiore a 70Khz (dovendo gestire le altre routine, approssimiamo a circa a 65Khz la massima frequenza di arrivi, per garantire il funzionamento sicuro).

Sperimentalmente si osserva che la frequenza massima di un encoder dei motori è non supera i 2 KHz massimi.

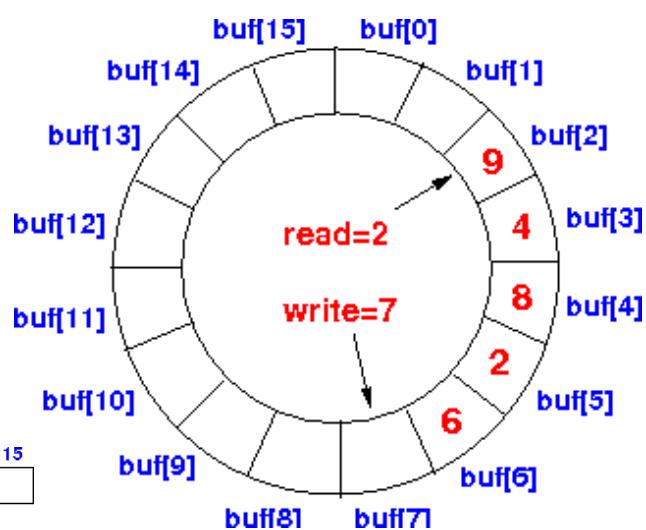
La dimensione del buffer garantisce un tempo di stoccaggio senza perdita di dati (con arrivi ogni 2Khz che è comunque ben oltre il normale) pari a:

$$\frac{1}{2} \text{ ms} \times 128 = 64 \text{ ms}$$

buf:	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	9	4	8	2	6											

↑ ↑

read=2 write=7



Oltre questo tempo non si garantisce più la presenza di tutti i dati e potrebbe iniziare la sovrascrittura della coda. In tal caso il software mantiene coerente il buffer e non ci sarebbero problemi software, ma si perderebbe il conteggio degli encoder.

Elaborazione encoder

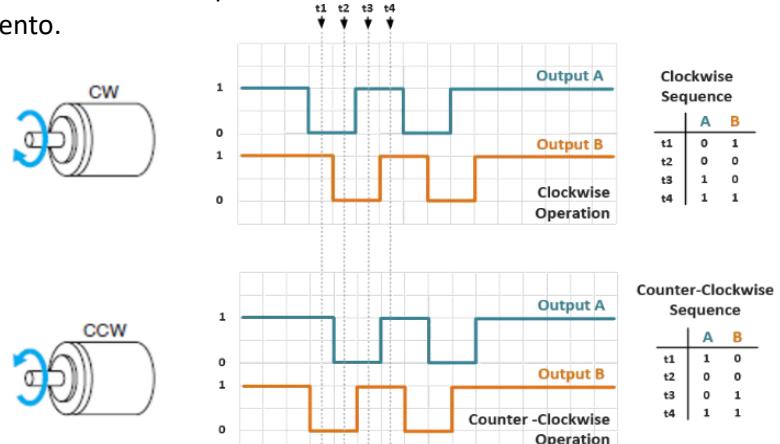
Ogni cella viene analizzata da questo algoritmo, che conoscendo lo stato precedente e l'attuale è in grado di capire se il motore è restato fermo, è aumentato o diminuito.

```
//VARIABILI PRIVATE DI calcStep
#define im 0 //impossibile
//          0   1   2   3   4   5   6   7   8   9   10  11  12  13  14  15
int8_t enc_states[] = { 0, -1, 1, im, 1, 0, im, -1, -1, im, 0, 1, im, 1, -1, 0 }; /*[old]BA-BA[new]*/
byte chAold, chBold, chAnew, chBnew, code;
void calcStep(int oldEn, int newEn) {
    /* Monto i bit BA (A LSB) */
    chAold = oldEn & 0x00FF;
    chBold = (oldEn & 0xFF00) >> 7; //disallineo di 1 per semplificare l'or binario
    chAnew = newEn & 0x00FF;
    chBnew = (newEn & 0xFF00) >> 7;

    //Calcolo primo encoder
    code = ((chAold & 1) | (chBold & 0b10)) << 2;
    code |= ((chAnew & 1) | (chBnew & 0b10));
    passi[0] += (enc_states[code]);
    //Calcolo Restanti 6
    for (byte i = 1; i < nMot; i++) {
        chAold >>= 1;
        chBold >>= 1;
        code = ((chAold & 1) | (chBold & 0b10)) << 2;
        chAnew >>= 1;
        chBnew >>= 1;
        code |= ((chAnew & 1) | (chBnew & 0b10));
        passi[i] += (enc_states[code]);
    }
}
```

L'algoritmo consiste nel mettere in quadratura i bit attuali e precedenti e in base al numero che viene fuori da questo OR binario si determina l'incremento.

Old		New		Value	
B	A	B	A	Dec	Encoder-step
0	0	0	0	0	0
0	0	0	1	1	-1
0	0	1	0	2	+1
0	0	1	1	3	Impossibile
0	1	0	0	4	+1
0	1	0	1	5	0
0	1	1	0	6	Impossibile
0	1	1	1	7	-1
1	0	0	0	8	-1
1	0	0	1	9	Impossibile
1	0	1	0	10	0
1	0	1	1	11	+1
1	1	0	0	12	Impossibile
1	1	0	1	13	+1
1	1	1	0	14	-1
1	1	1	1	15	0



Per capire la logica degli incrementi basta leggere la tabella di verità accanto ricordando che il tempo scorre in una sola direzione.

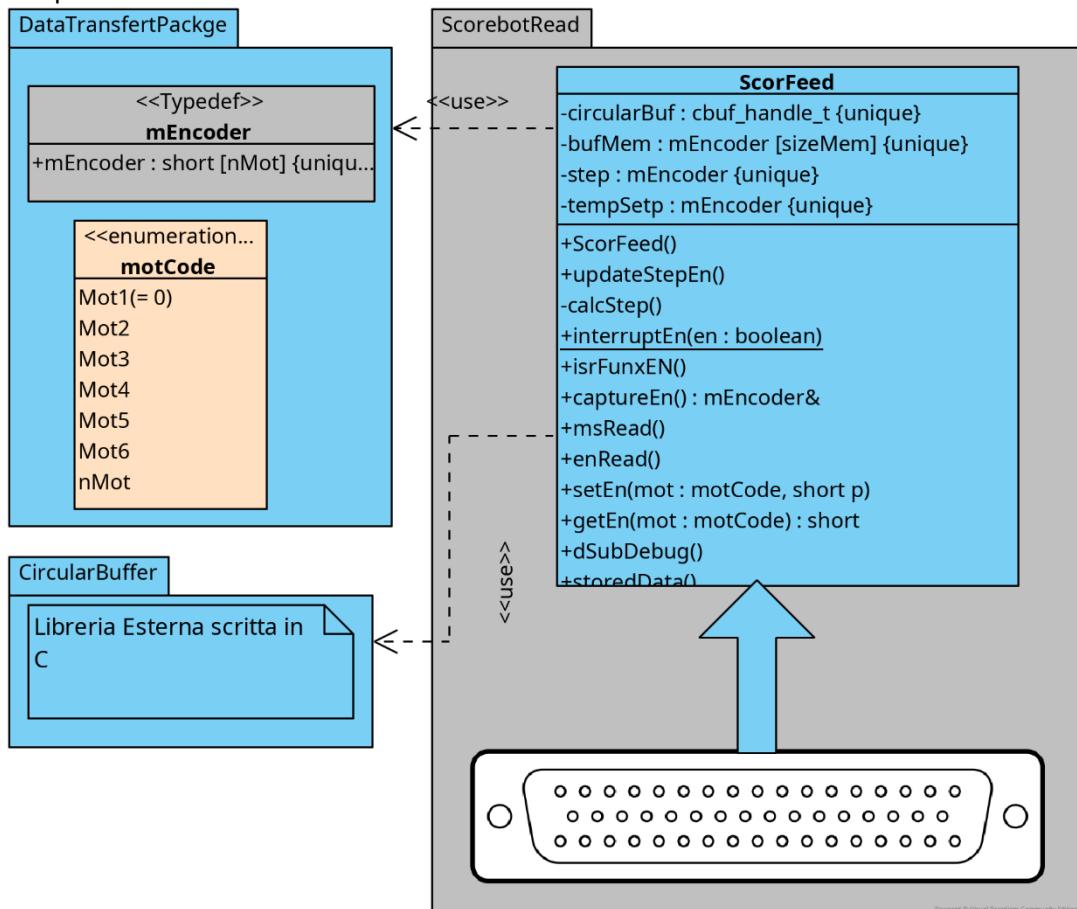
Per esempio, vediamo il **T2** che è uguale in entrambe le tabelle di verità, se la combinazione successiva è:

- “10” l’encoder farebbe +1 poiché il giro è orario.
- “01” l’encoder farebbe -1 poiché il giro è antiorario.

L’aver salvato in una locazione di memoria l’incremento accelera la decodifica al tempo di un accesso in memoria, mentre se fosse stato usato un metodo “if-else” il tempo di computazione sarebbe aumentato da un minimo di 4 Clock fino a 16 Clock.

Classe ScorFeed

La classe responsabile della lettura dei sensori dello Scorbott è la “ScorFeed”:



La classe attua sia la lettura dei micro-switch in polling, che la gestione dell'interrupt e il setup dei pin e del timer per gli encoder.

La libreria di buffer circolare è un efficiente implementazione in C scaricata online

Procedura di Discovery Home (Homming)

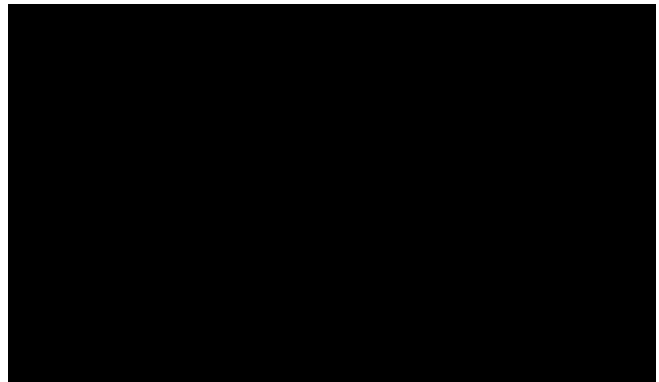
La strategia di Homming consiste nel muovere singolarmente i motori fino a trovare i Micro-Switch della home.

1. Si inizia con l'avambraccio che sale cercando la home
2. Successivamente inizia la ricerca il gomito
3. Polso e pinza hanno i motori a **controllo differenziale**, vanno per tanto cercati insieme:
 - Si cerca la home del ROLL
 - Si cerca la home del PITCH
4. Trovata questa PSEUDO HOME, viene ri-eseguito il ciclo, per garantire che la posizione raggiunta sia vera e non influenzata dagli spostamenti degli altri link
5. Viene cercata la home della base

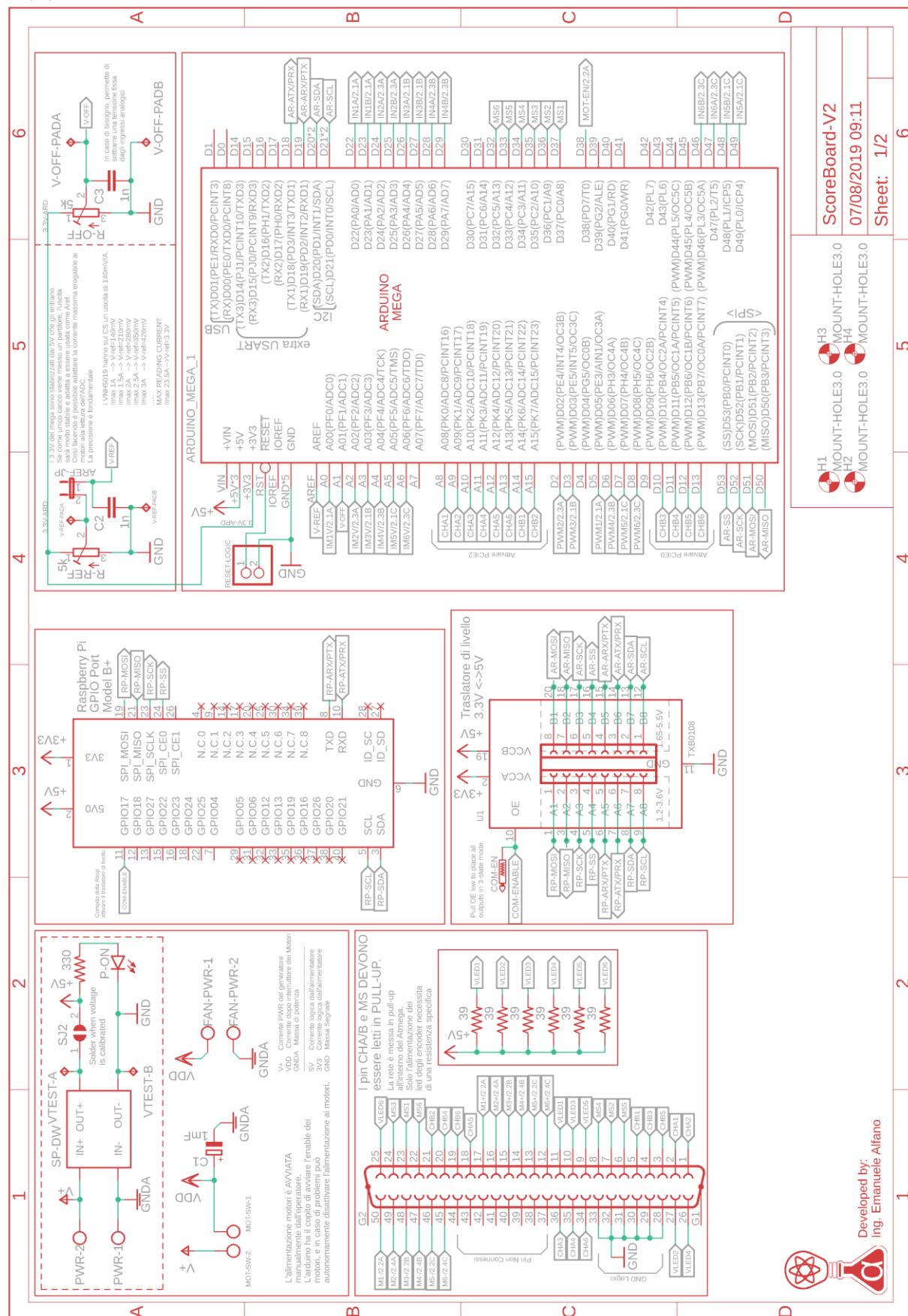
Al termine tutti gli encoder vengono messi a 0 e la procedura è terminata.

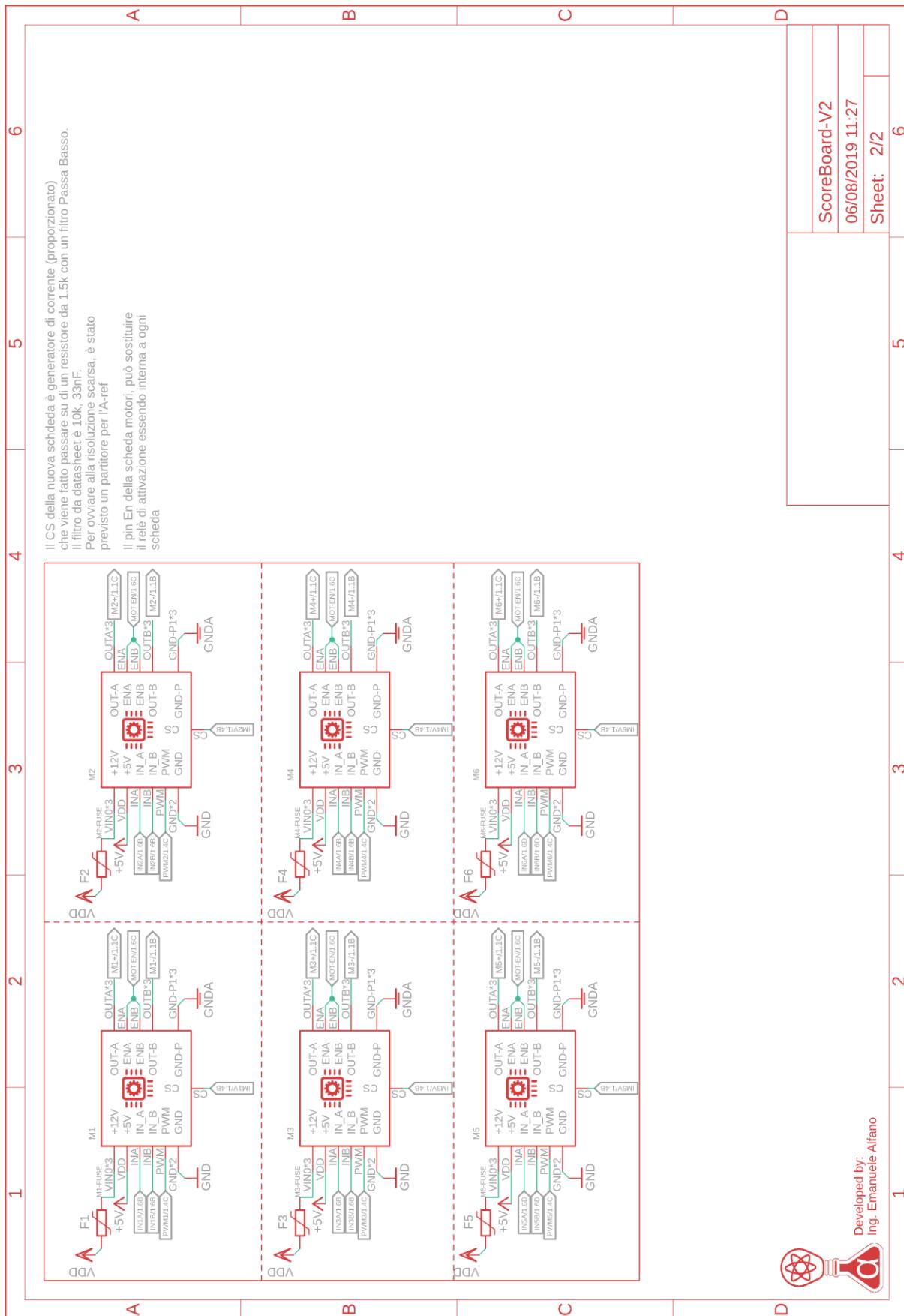
È possibile vedere un video esplicativo all'indirizzo:

https://youtu.be/lcqI7zQ_A4U



Appendice A: Schema elettrico





Developed by:
Ing. Emanuele Alfano

