



Descrizione

Manuale operativo e progettuale della scheda di controllo “ScorBoard\_V2”.

Scheda sviluppata per il controllo di uno Scorbot a 6 motori DC Brushed.

Scorboard\_v2

MANUALE TECNICO

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.

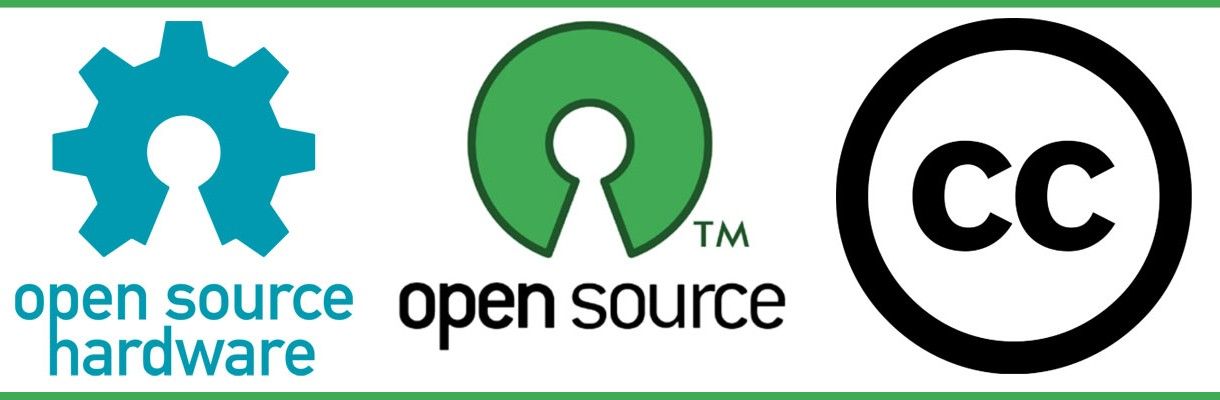
Il progetto è stato sviluppato da degli studenti nella logica **dell’Open Hardware e dell’Open Software**, qualunque modifica o sviluppo è ben accetta e sarebbe gradita una proposta di merge al progetto principale.

Questo manuale vuole si spiegare come usare la board, ma anche i passi che hanno portato alla sua realizzazione, le motivazioni tecniche, le caratteristiche pecuniari, sia hardware che software, al fine di semplificarne 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](#_Toc15988746)

[Global Board 1](#_Toc15988747)

[Motors 1](#_Toc15988748)

[Only one Motor Drive 1](#_Toc15988749)

[Logic 1](#_Toc15988750)

[Assemblaggio 2](#_Toc15988751)

[Componenti board 2](#_Toc15988752)

[Saldatura Componenti 3](#_Toc15988753)

[Collegamento board al sistema 4](#_Toc15988754)

[Alimentazione standard 4](#_Toc15988755)

[Alimentazione motori indipendente 4](#_Toc15988756)

[Ventola di raffreddamento 4](#_Toc15988757)

[Taratura 5](#_Toc15988758)

[Trimmer R-REF 5](#_Toc15988759)

[Trimmer R-OFF (facoltativo) 5](#_Toc15988760)

[Documentazione Elettronica 6](#_Toc15988761)

[Schema dell’Alimentazione 6](#_Toc15988762)

[Connettore Scorbot 7](#_Toc15988763)

[Driver Motori VNH5019 8](#_Toc15988764)

[Configurazioni del ponte 8](#_Toc15988765)

[Lettura corrente motori 9](#_Toc15988766)

[Misura sensibilità di uscita 11](#_Toc15988767)

[ADC ATmega 12](#_Toc15988768)

[Registri interni default 12](#_Toc15988769)

[Parametri programmabili 12](#_Toc15988770)

[Affidabilità tensioni di riferimento 12](#_Toc15988771)

[Traslatore di livello 13](#_Toc15988772)

[Documentazione Firmware 14](#_Toc15988773)

[Controllo motori 14](#_Toc15988774)

[Lettura delle correnti 15](#_Toc15988775)

[Lettura degli encoder & micro-switch 16](#_Toc15988776)

[Procedura di Discovery Home (Homming) 17](#_Toc15988777)

[Comunicazione SPI 18](#_Toc15988778)

[Gestione Interrupt 19](#_Toc15988779)

[Documentazione Libreria Comunicazione Raspberry-Py 20](#_Toc15988780)

[Tabella riassuntiva API 20](#_Toc15988781)

[How to Use 20](#_Toc15988782)

[Appendice A: Schema elettrico 21](#_Toc15988783)

# 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

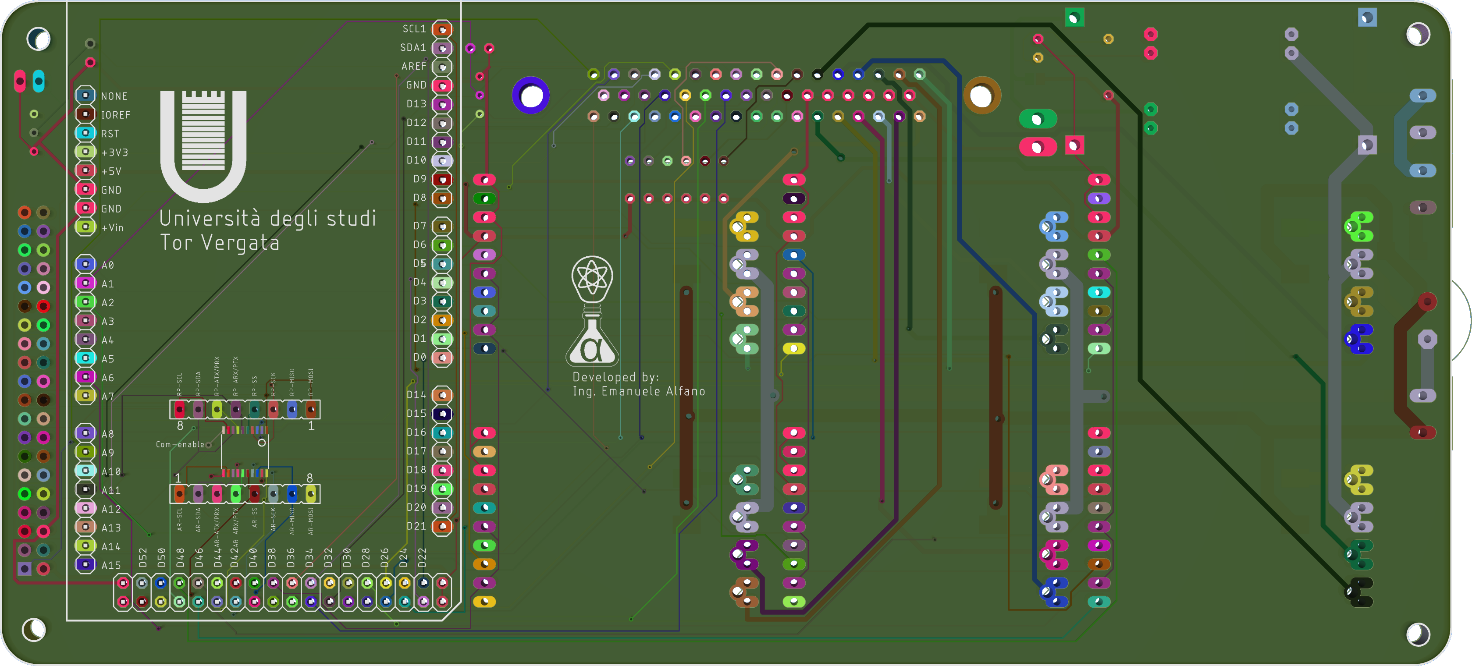
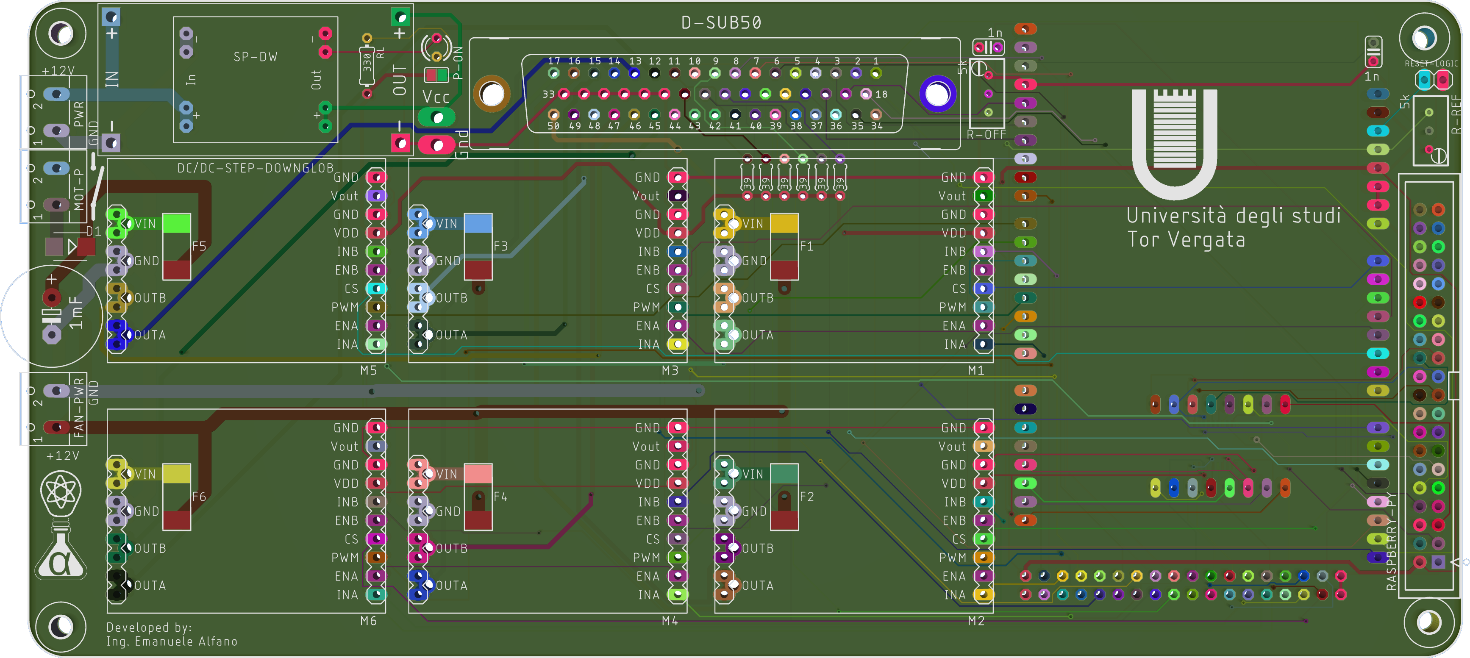
## Logic

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 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 Scorbot)
* 1x Resistore 330Ω
* 2x Trimmer di taratura  
  (**uno opzionale**, vedere sezione [Taratura](#_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 Scorbot)
* 6x Driver Motori VNH5019
* 1x Connettore RASPBERRY-PY B+
* 1x Traslatore di Livello smd TXB0108

## Saldatura Componenti

L’ordine di saldatura consigliato è:

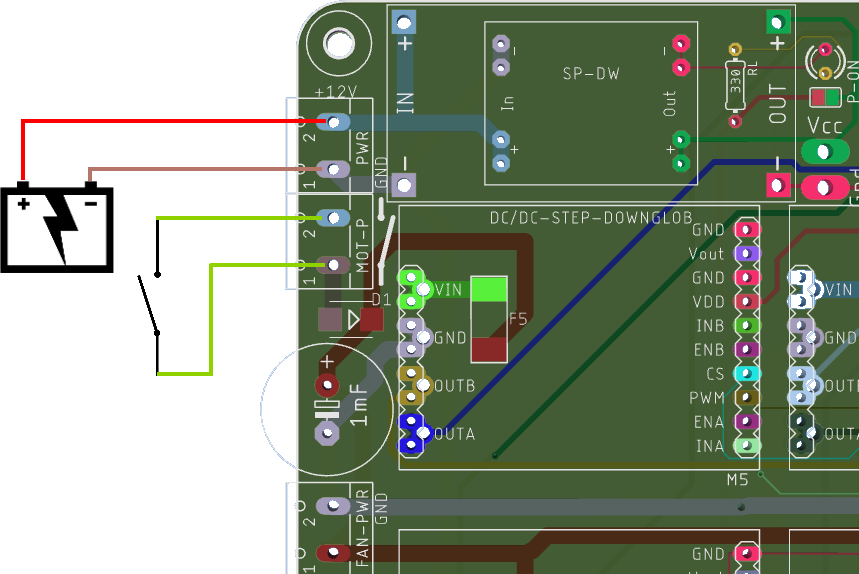
1. Saldare il traslatore di livello smd TXB0108 sul retro della board.
2. Saldare i componenti smd sul davanti della board (fusibili e diodo).
3. Saldare le Resistenze.
4. Saldare i **Pin Maschi** rivolti verso l’alto per l’Arduino Mega (per tenerli allineati si consiglia di farlo tenendo i pin maschi inseriti dentro i pin femmina nel mega, quindi levarlo se scomodo nei passaggi successivi).
5. Saldare i pin Maschi sui 6 Driver motori rivolgendo i maschi sotto il chip.
6. 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).
7. Saldare i Trimmer “R-REF” e *se necessario* il “V-OFF” e il loro condensatore da 1 nF. ([vedi qui](#_Taratura))
8. Saldare il connettore per il RASPBERRY-PY verso l’alto e con la tacca rivolta verso l’esterno.
9. Saldare il Led
10. 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)
11. Saldare i connettori a vite.
12. Saldare il connettore D-SUB50 rivolto verso l’alto.
13. Saldare condensatore da 1mF.
14. **Inserire nuovamente lo step-down quindi:**
    1. 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](#_Logic).
    2. Eseguita la taratura si saldi il Pad smd accanto al led connettendo i 2 estremi.
    3. Ora la parte logica della scheda riceve l’alimentazione.
15. Inserire, se tolti, l’Arduino mega e i driver motori.

Ad assemblaggio completo, il risultato dovrebbe essere:

[Foto scheda saldata]

## Collegamento board al sistema

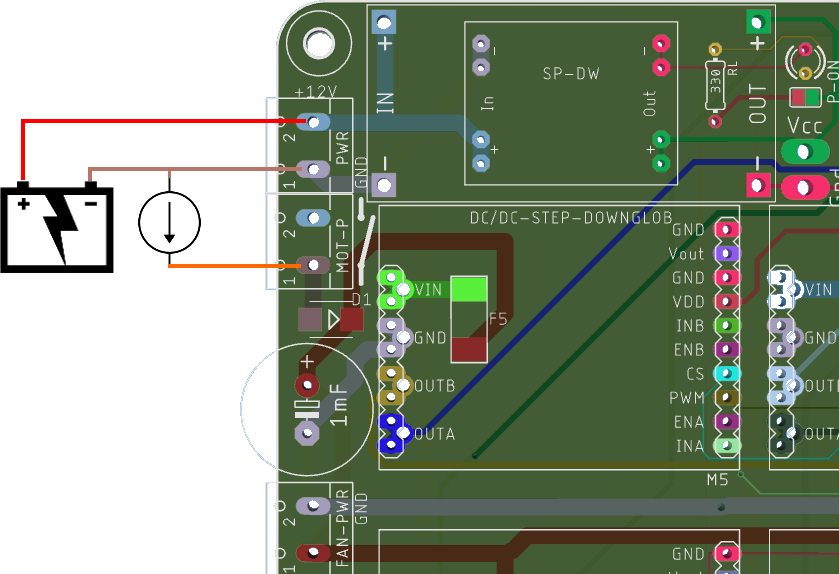
### Alimentazione standard

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.

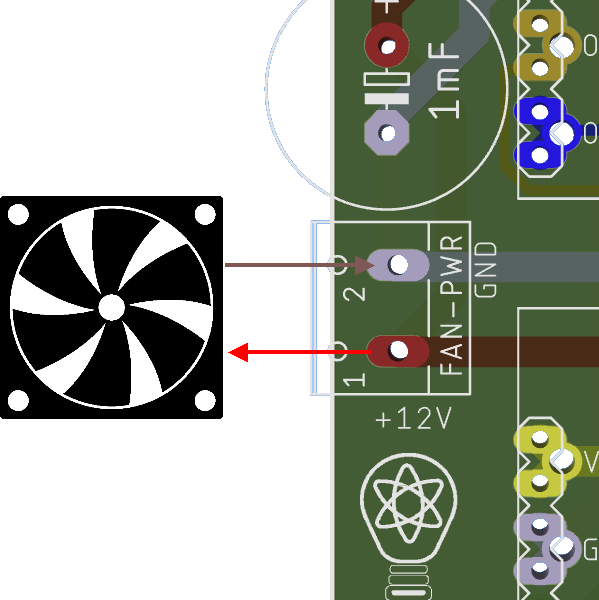
La batteria può essere una qualsiasi purché rispetti le caratteristiche presenti nella [Scheda tecnica](#_Global_Board)

### 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](#_Motors) nella sezione Motor.

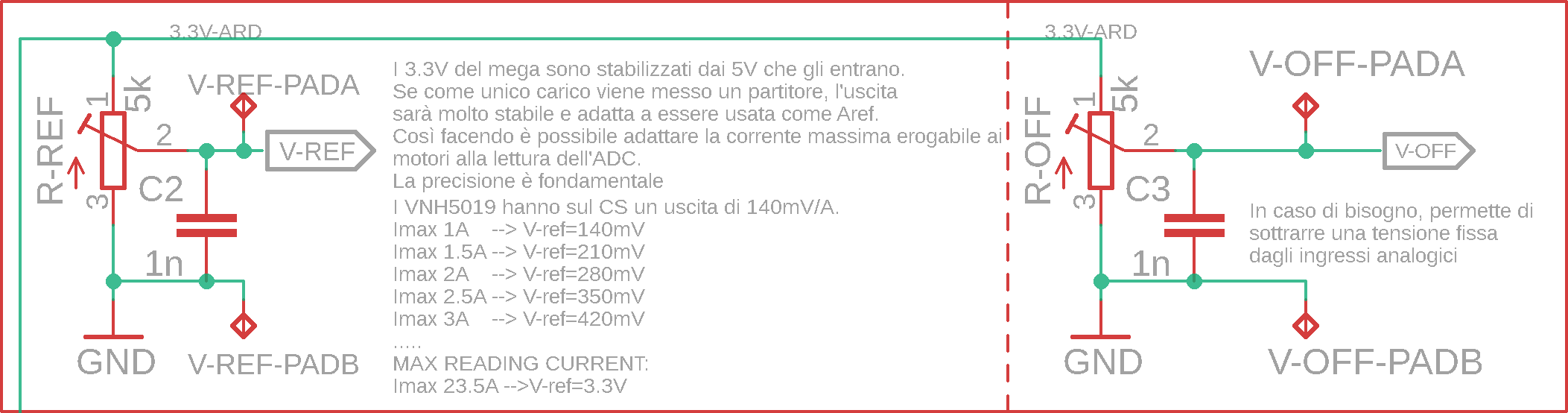
È importante collegare insieme le 2 GND!!!

### Ventola di raffreddamento

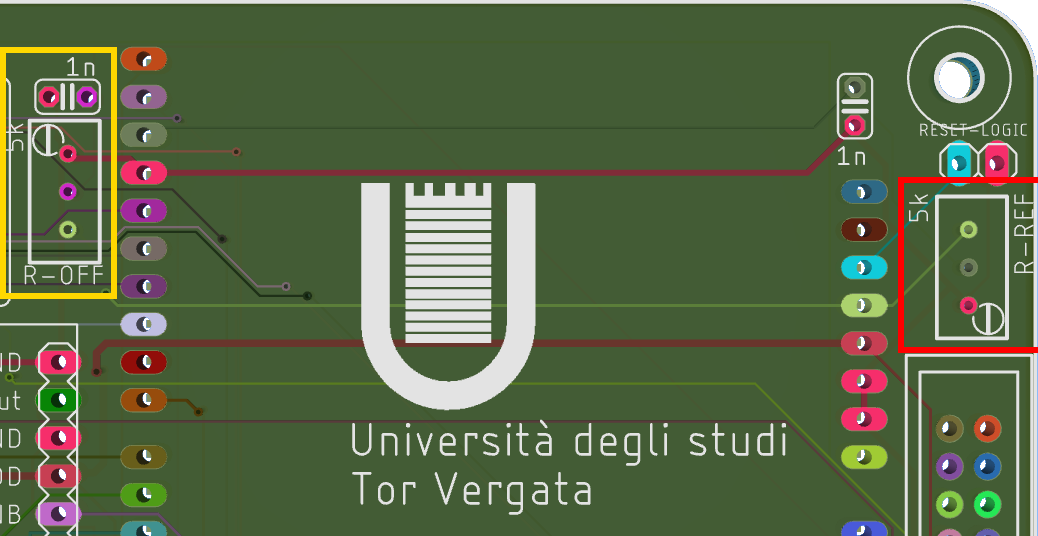
In caso risulti necessario collegare una ventola per raffreddare la scheda, è stata predisposto un collegamento alimentato con la tensione che alimenta i motori.

## 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

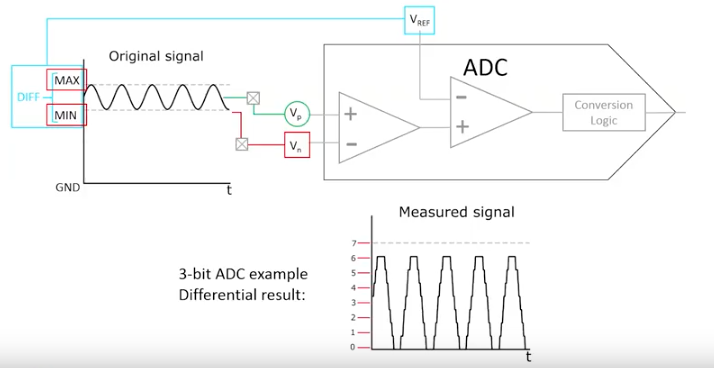
Il trimmer **(Rosso)** regola la tensione di riferimento dell’ADC presente dentro il mega. In base al voltaggio impostato nel trimmer, si imposta il fondo scala dell’ADC. Per stabilire la tensione di cui si ha bisogno bisogna stabilire la massima corrente che si vuole leggere, quindi con questa formula si ottiene il valore voluto del V-REF:

Dove:

* è la sensibilità di uscita del VNH5019 del pin CS che è pari a medi, per un calcolo esatto vedere la sezione del [Driver Motori](#_Misura_sensibilità_di).
* è la massima corrente che si vuole leggere
* è il numero da 0 ÷1023 letto dal mega
* 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.

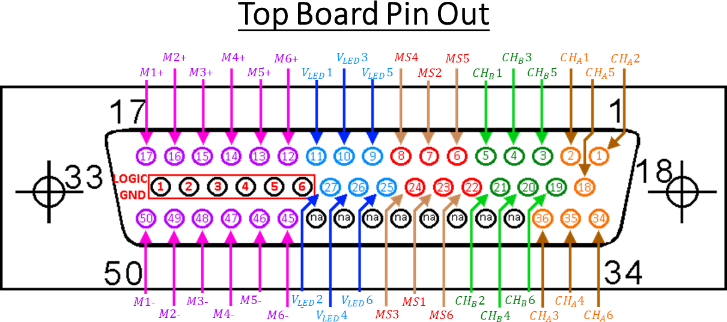
L’alimentazione ai motori avviene per mezzo di un connettore nel quale si può:

* 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](#_Alimentazione_standard))
* Collegare un secondo alimentatore esclusivo per i motori. ([schema](#_Alimentazione_motori_indipendente))

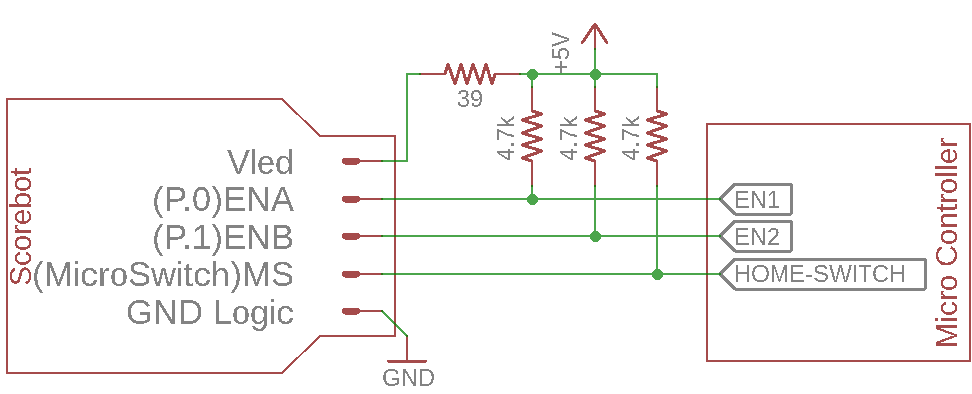
È predisposto un attacco comodo per una possibile ventola alimentata dall’alimentatore motori.

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](#_Specifiche_Tecniche).

## Connettore Scorbot



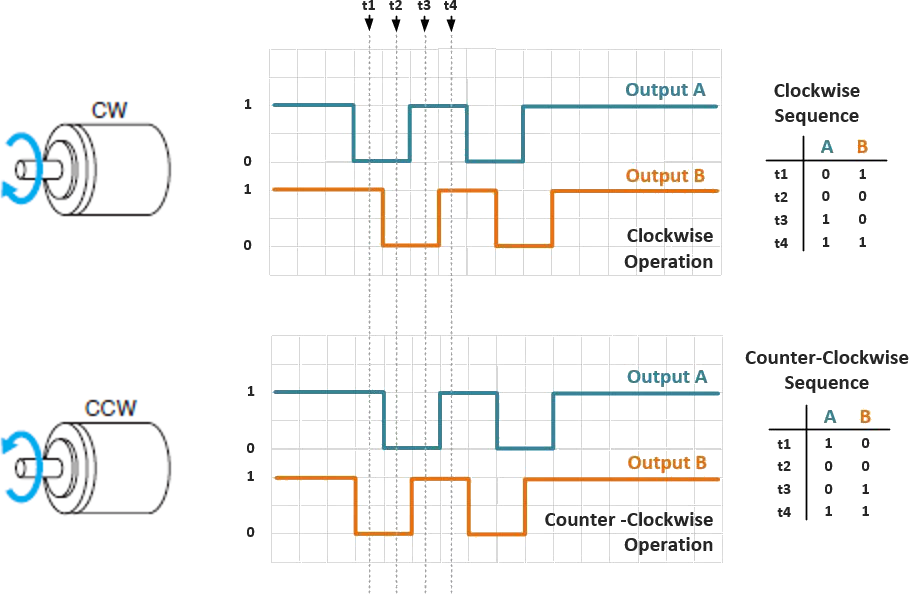
Un connettore abbastanza standard per gli Scorbot è il D-SUB-50, di cui qui è riportata la connessione dei pin sul connettore e sulla board.

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

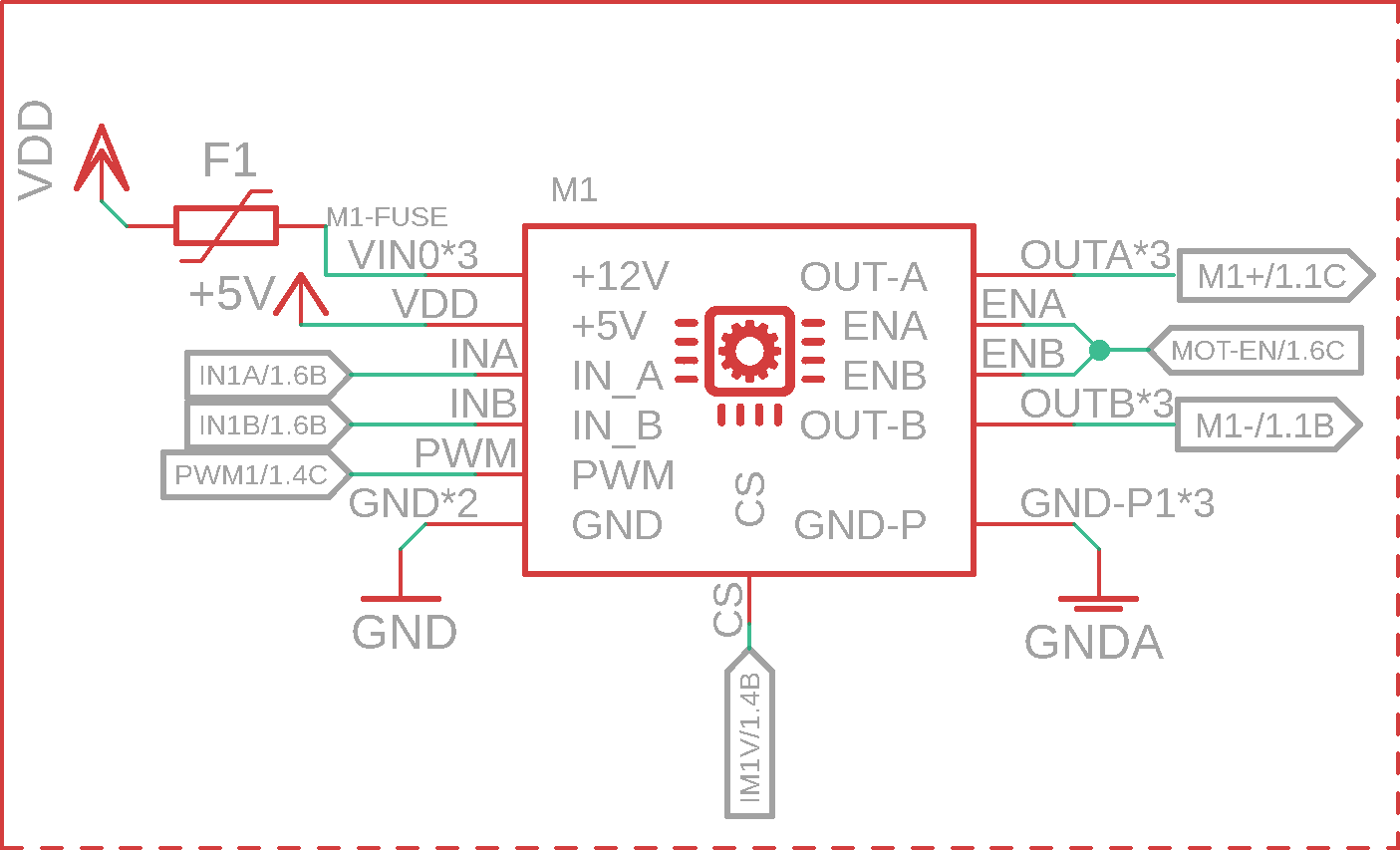
Nella scheda le resistenze di pull-up sono presenti dentro l’atmega. La resistenza da 39Ω invece è l’alimentazione e ne è presente una per ogni motore.

Maggiori informazioni si possono reperire dal datasheet dello Scorbot 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

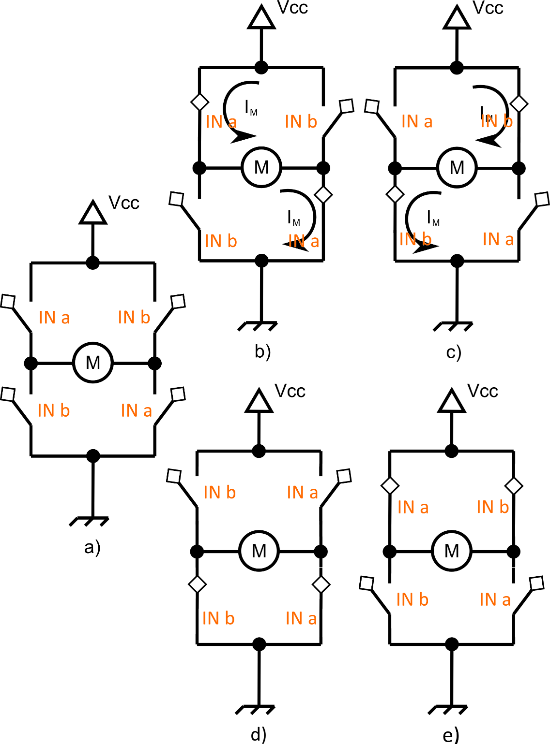


Per controllare ogni motore i usiamo i driver Pololu i “VNH5019”.

Questi drive sono dei ponti ad H molto potenti, sono infatti originariamente progettati per controllare i servo motori 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.

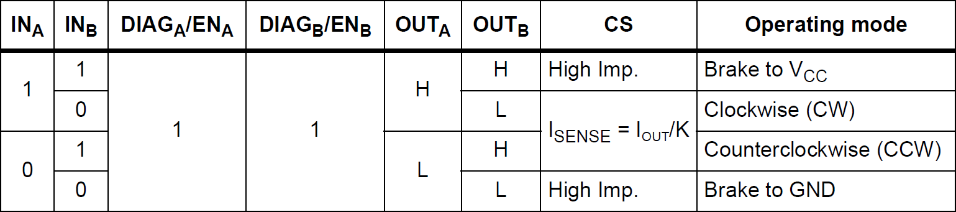
I driver hanno bisogno di 2 alimentazioni differenti, la prima è quella di controllo e va da 5.5V a 24V, la seconda è quella dei motori e può gestire costantemente fino a 12A e per alcuni periodi può arrivare a picchi di 30A

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)



### 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)

|  |  |
| --- | --- |
|  |  |
| 5% duty cycle 500hz | 10% duty cycle 500hz |
|  |  |
| 25% duty cycle 500hz | 50% duty cycle 500hz |
|  |  |
| 90% duty cycle 500hz | 100% duty cycle 500hz |

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](#_ADC_Atmega))

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)

|  |  |
| --- | --- |
|  |  |
| 5% duty cycle 4khz | 20% duty cycle 4khz |
|  |  |
| 40% duty cycle 4khz | 50% duty cycle 4khz |
|  |  |
| 75% duty cycle 4khz | 100% duty cycle 4khz |

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

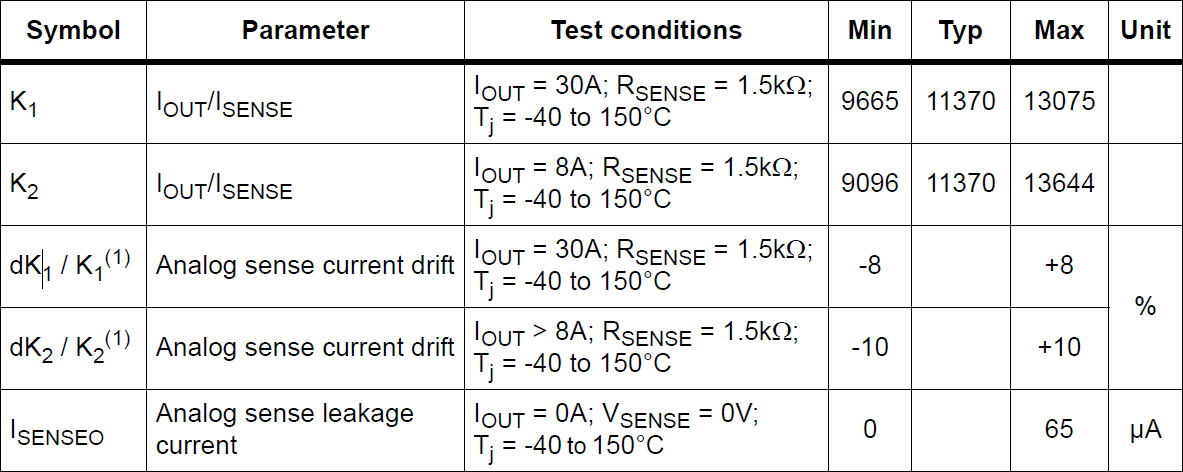
E contando che li deve leggere ciclicamente tutti e 6, il tempo di scansione è di 672us ~ 1,5Khz. ([dettagli](#_ADC_Atmega))

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).

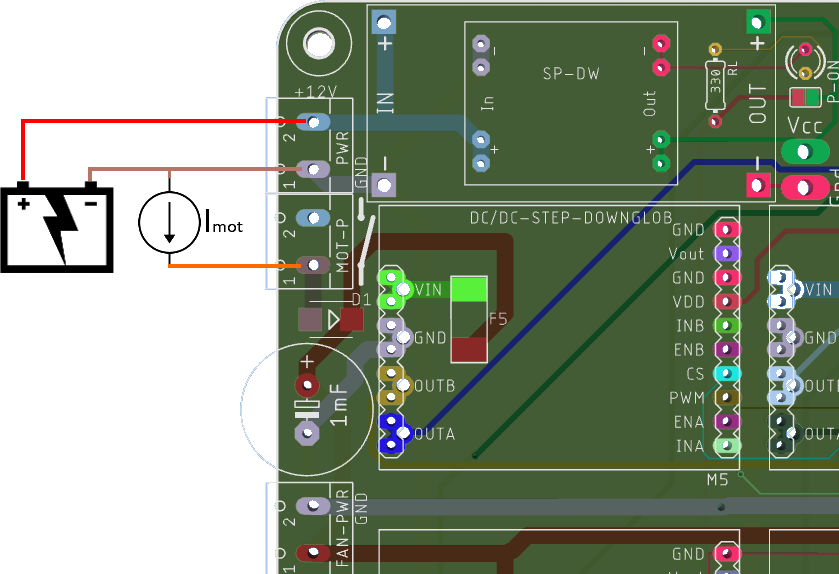
Vista la stabilità in lettura con questo PWM, questa è la frequenza impostata di default sulla scheda.

### 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.



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 , 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 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 Volmetro 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

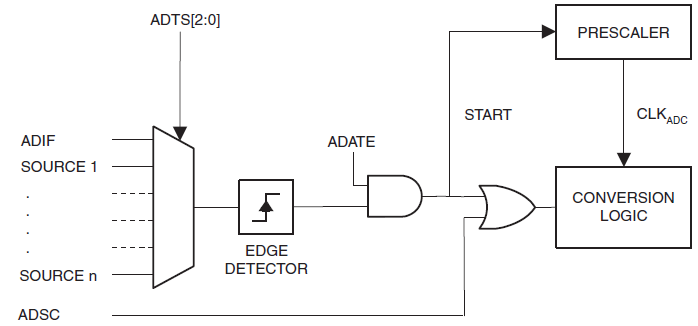
I registi interni sono impostati con le seguenti caratteristiche:

* 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](#_Trimmer_R-OFF_(facoltativo))
* **Vref 2.56V / 1.1V / Aref esterna**
  + Aref esterna: Riferimento selezionabile dal trimmer R-REF come visto in [Taratura](#_Trimmer_R-REF) (default)
  + Vref 2.56V e 1.1V: sono dei riferimenti interni al chip e sono fissi e molto stabili

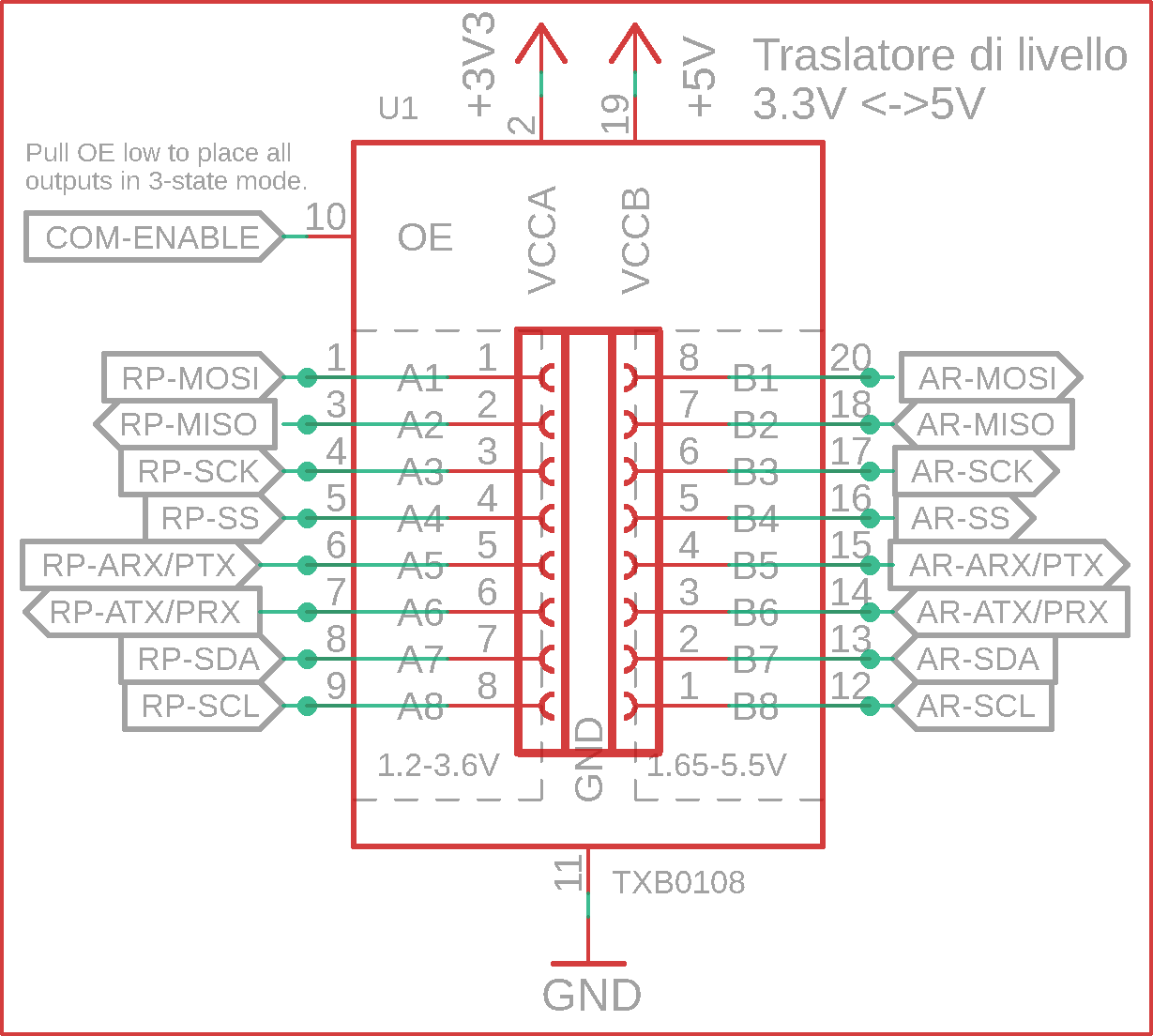


### Affidabilità tensioni di riferimento

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.

# Documentazione Firmware

Il firmware è stato scritto in C++, usando come editor il software open-source [SLOEBER](https://eclipse.baeyens.it/), ed i sorgenti scritti sono disponibili [qui](https://github.com/Alfystar/Scorbot-CA/tree/master/2_boardSystem/V2/SW).

Di seguito verranno spiegate le aree funzionali più importanti del codice, in particolare:

* Controllo motori
* Lettura delle correnti
* Lettura degli encoder & micro-switch
* Procedura di Discovery Home (Homming)
* Comunicazione SPI
* Gestione Interrupt

Per vedere più nel dettaglio gli aspetti e le interconnessioni si consiglia vivamente di andare a studiare nel dettaglio il codice.

## Controllo motori

Per controllare i motori si è progettata una classe apposita, per permettere di aggiungere comportamenti dinamici e avanzati. In particolare, i Task chiave di questa classe sono:

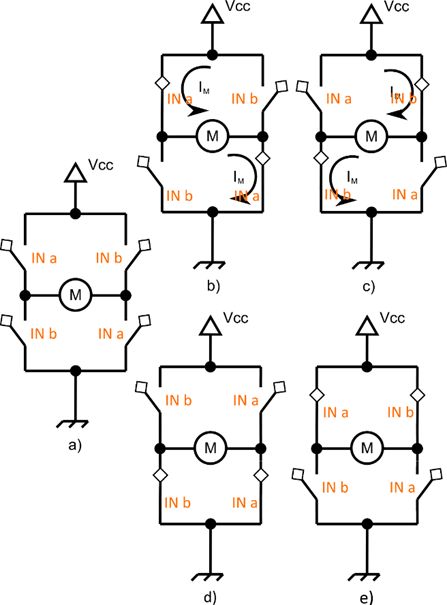
1. Mette a disposizione dei metodi di controllo dei motori temporizzati, in maniera trasparente al programmatore.
2. ~~Aggiungere funzionalità di sicurezza automatici (limitare range di movimento in una banda, così da evitare danni da urti).~~

### Classe motori

[Struttura della classe]

### 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
* reversDir();  
  (c & b) Stessa velocità, segno opposto
* soft\_stop();
* (e) hard\_stop(unsigned int delay\_time);
* (d) soft\_stop(unsigned int delay\_time);
* freeRun();

## Lettura delle correnti

## Lettura degli encoder & micro-switch

## Procedura di Discovery Home (Homming)

## Comunicazione SPI

## Gestione Interrupt

# Documentazione Libreria Comunicazione Raspberry-Py

Lato utilizzatore è stata scritta una libreria in C++, che si occupa di comunicare con la scheda arduino e inviare tutte le impostazioni e i comandi, una API quindi.

## Tabella riassuntiva API

|  |  |  |
| --- | --- | --- |
| Comando | Raspberry invia | Raspberry riceve |
| setPWM | Valori dei PWM di tutti i motori | Conteggio degli ENCODER |
| getCurrent |  | Ultime letture dei sensori di corrente |
| getSetting |  | Impostazioni salvate sulla EEPROM |
| setSetting | Nuovi impostazioni da salvate in EEPROM |  |
| goHome | Attiva la routine di ricerca della Home |  |

## How to Use

# Appendice A: Schema elettrico

