
Università Politecnica delle Marche
Ingegneria Informatica e dell'Automazione
Anno Accademico 2017-2018

# CONTROLLO IN TENSIONE DELLA VELOCITÀ DEI MOTORI

Studenti:
Brutti Marco
*marcobrutti3@gmail.com*

Ciancaglione Alessandro
*ale.cianca@outlook.it*

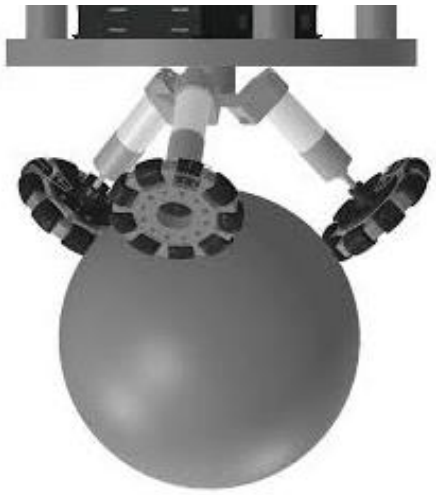
Di Muzio Daniele
*ddimuzio96@gmail.com*

Iezzi Christian
*christian.iezzi96@outlook.it*

Relatore:
Prof. Bonci Andrea

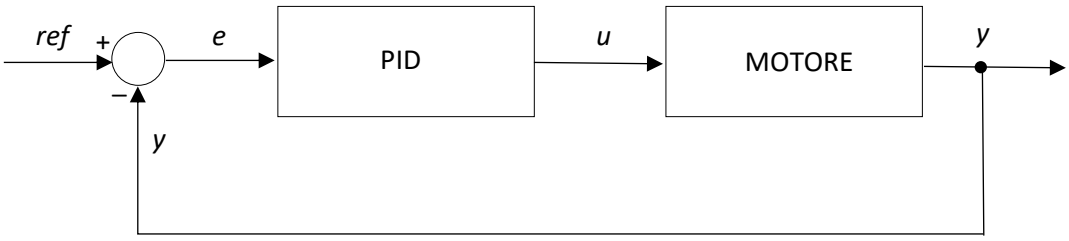
INDICE:

# Capitolo 1. INTRODUZIONE



Il Ballbot è un particolare tipo di robot in grado di bilanciarsi e muoversi autonomamente su di una palla poggiata su un piano orizzontale. L'obiettivo di questa tesina è quello di studiare, progettare e realizzare il controllo in tensione della velocità dei motori utilizzati per mantenere in equilibrio il robot. Essendo costituito da tre motori uguali disposti a 120° l'uno dall'altro, formando quindi un sistema equilibrato, si è deciso di studiarne il funzionamento prendendo in analisi un unico motore, replicando i risultati ottenuti sugli altri. Lo sviluppo del sistema di controllo è stato suddiviso in due sezioni: la parte hardware e la parte software. Si è quindi modellato il motore e simulato il controllo in ambiente MATLAB-Simulink. Successivamente si è testato, mediante l'utilizzo del software, il corretto funzionamento del sistema.

## 1.1. SCHEMA A BLOCCHI DEL SISTEMA DI CONTROLLO




Il seguente schema a blocchi rappresenta il sistema a catena chiusa nel quale la regolazione è automatica: l'uscita viene confrontata con l'ingresso di riferimento, in modo da produrre, ogni qual volta si verifichi una differenza tra il segnale di uscita e il segnale di riferimento, un'azione correttiva che riporti l'uscita al valore desiderato. Nello specifico la differenza tra l'ingresso, *ref*, dato dalla velocità di riferimento, e l'uscita, *y*, data dalla lettura della velocità elaborata dall'encoder, genera l'errore, *e*, necessario al sistema per applicare l'opportuna correzione, *u*, effettuata dal PID.

## 1.2. PID

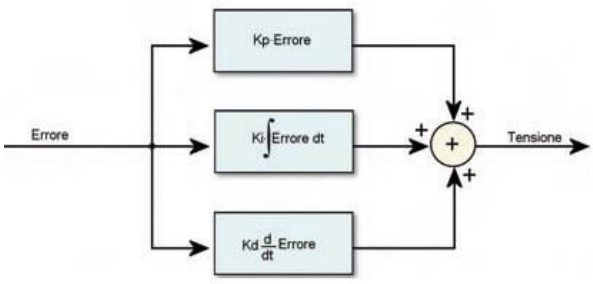
Il regolatore PID è un particolare tipo di controllore il quale genera l'uscita in base al contributo di tre termini:

- Proporzionale, azione proporzionale all'errore;
- Integrativo, azione proporzionale all'integrale dell'errore;
- Derivativo, azione proporzionale alla derivata dell'errore.

La legge di controllo di un generico PID è la seguente:

$m(t) = K_P e(t) + K_I \int_0^t e(\tau)d\tau + K_D \frac{de(t)}{dt}$,

dove $K_P$ è il guadagno proporzionale, $K_I$ è il guadagno integrale e $K_D$ è il guadagno derivativo.




Il PID è stato implementato su un dispositivo di controllo digitale e quindi si è discretizzato il PID con passo $T_s$, dove quest'ultimo è il periodo di campionamento; discretizzando abbiamo, quindi, ottenuto l'uscita del PID[1]:

$m[k] = m[k-1] + a \cdot e[k] + b \cdot e[k-1] + c \cdot e[k-2];$ (1)

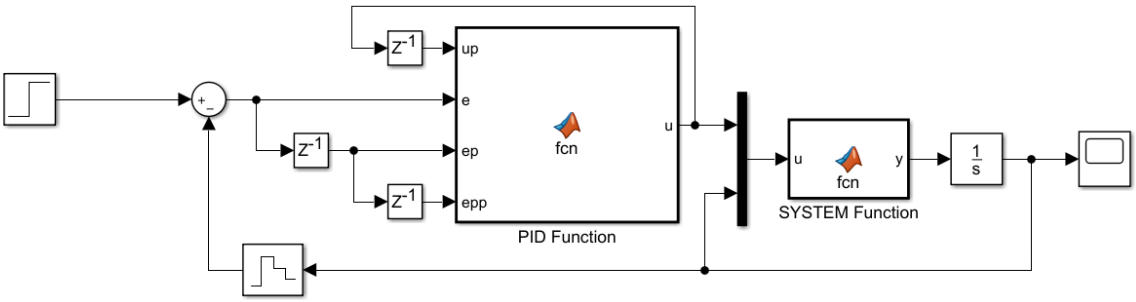
dove $a = K_P + K_I \cdot \frac{T_S}{2} + \frac{K_D}{T_S}$, $b = -K_P + K_I \cdot \frac{T_S}{2} - 2 \cdot \frac{K_D}{T_S}$ e $c = \frac{K_D}{T_S}$.

Per la taratura dei parametri del PID si è utilizzato Simulink, un software per modellare, simulare e analizzare sistemi dinamici; è strettamente integrato con Matlab.

Di seguito è riportata la modellazione dell'intero sistema di controllo in cui sono presenti:

- modello motore in tempo continuo ricavato da identificazione parametrica;
- controllore in tempo discreto (1);
- catena di controllo con blocco S&H di retroazione per discretizzare l'uscita del sistema;
- blocchi di ritardo unitario dei segnali di errore;
- ingresso di riferimento a gradino unitario.

[1] Gli approfondimenti riguardanti la discretizzazione del PID sono riportati in [1]




Tale sistema di controllo è applicato alla seguente funzione di trasferimento, SYSTEM Fuction, la quale modellizza il motore mediante i seguenti parametri:

```
function y = fcn(u)

%#codegen
Km=5.88;
Ke=0.07;
R=4.8;
Ir=0.006;

y=((-Km*Ke)/(R*Ir))*u(2)+(Km/(R*Ir))*u(1);
```

Dove *Km* $[\frac{N \cdot m}{A}]$ è la costante di coppia del motore, *Ke* $[\frac{V \cdot sec}{rad}]$ è la costante di tensione del motore, *R* $[\Omega]$ è la resistenza elettrica di armatura e *Ir* $[Kg \cdot m^2]$ è il momento di inerzia del motore; inoltre vi sono altri parametri che sono stati considerati ininfluenti nel calcolo della funzione di trasferimento per la modellazione del motore: il coefficiente di attrito viscoso del motore $[N \cdot m \cdot sec]$, il rendimento del motore pari ad 1, il rendimento della gear-box pari ad 1 e l'induttanza di armatura $[H]$.

L'accelerazione angolare y $[\frac{\text{rad}}{\text{sec}^2}]$ è pari a $y = \frac{\text{Km}}{\text{R}\cdot\text{Ir}} \cdot u(1) - \frac{\text{Km}\cdot\text{Ke}}{\text{R}\cdot\text{Ir}} \cdot u(2)$, dove u(1) [V] e u(2) $\left[\frac{\text{rad}}{\text{sec}}\right]$ sono rispettivamente la tensione di alimentazione e la velocità angolare.

Per quanto riguarda la taratura del PID si è eseguita una calibrazione manuale andando a modificare i tre parametri $K_P$, $K_I$ e $K_D$, verificando, tramite l'oscilloscopio virtuale di Simulink, l'uscita del sistema.

PID Fuction rappresenta la modellizzazione del PID, ovvero l'implementazione della funzione di trasferimento del PID vista in precedenza; quindi l'uscita del PID *u* è in relazione all'uscita del PID all'istante precedente *up*, all'errore *e*, agli errori agli istanti precedenti *ep* e *epp*, dei 3 parametri $K_P$, $K_I$ e $K_D$ e del tempo di campionamento *Ts*.

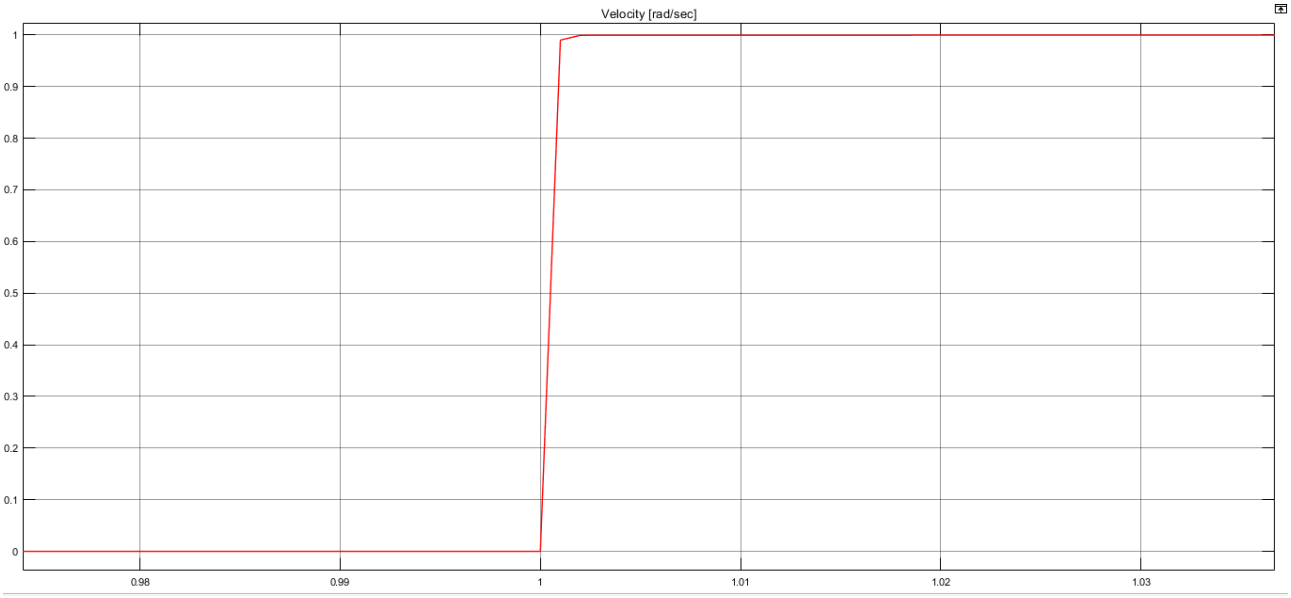
La funzione PID implementata in Simulink è la seguente:

```
function u = fcn(up,e,ep,epp)

Kp=4.85;
Ki=65;
Kd=0;
Ts=0.001;

u = up + (Kp+Ki*Ts/2+Kd/Ts)*e + (-Kp+Ki*Ts/2-2*Kd/Ts)*ep + (Kd/Ts)*epp;
```

Variando i parametri si è ottenuto il segnale di uscita, il più vicino possibile al segnale di uscita ideale, ovvero un segnale a gradino.

Di seguito è riportata l'uscita del sistema generata dai seguenti parametri, i quali sono quelli selezionati per il controllo implementato sull'hardware:

$K_P = 4.85, K_I = 65\ e\ K_D = 0.$




Dal grafico si può notare che dopo un tempo pari a 1s è stato fornito in ingresso al sistema un gradino unitario.

Si definisce tempo di assestamento del sistema il tempo necessario affinché l'uscita del sistema si porti definitivamente entro il ±Δ% del valore finale; considerando che il valore finale è pari ad 1 e si è scelto Δ=2% la fascia presa in esame è 0.98÷1.02.

Dalla simulazione su Simulink si è constatato che il tempo di assestamento è pari a 0.99ms poiché il sistema raggiunge il valore 0.98 e non esce più dalla fascia 0.98÷1.02.

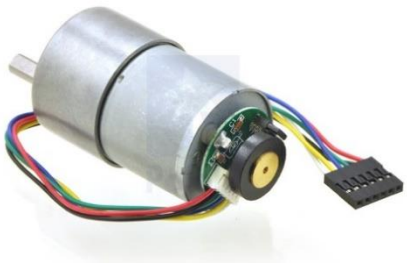
# Capitolo 2. HARDWARE

Di seguito sono descritti i vari componenti hardware utilizzati nella realizzazione del progetto.

## 2.1. RENESAS DEMOSTRATION KIT YRDKRX63N (100 PIN)


La scheda YRDKRX63N monta il µ-controller RX63N. La sua programmazione avviene mediante il programma e2studio, tramite il quale si eseguono anche codifica e debugging. L'interfacciamento tra PC e µ-controller avviene tramite porta USB.[2]

## 2.2. MOTORE POLOLU RB-POL-126 CON ENCODER E RUOTA OMNIDIREZIONALE



Il motore RB-POL-126 è di tipo brushed ed è alimentato con una tensione tra 0V e 12V; è dotato di un encoder rotativo incrementale di tipo magnetico ad effetto Hall.

[2] Gli approfondimenti della scheda YRDKRX63N e del µ-controller RX63N sono riportati in [2], [3] e [4].

Sul blocco motore-encoder sono collegati 6 cavi di colori differenti:

- Nero: alimentazione motore;
- Rosso: alimentazione motore;
- Blu: alimentazione encoder (Vcc: 3.5-20V);
- Verde: massa encoder;
- Giallo: output canale A encoder;
- Bianco: output canale B encoder.

Dai canali A e B dell'encoder escono rispettivamente due onde quadre identiche di valore compreso tra 0V e Vcc, sfasate tra loro di 90°. La frequenza delle due onde indica la velocità del motore mentre l'ordine di queste indica il suo verso di rotazione; nello specifico, se l'onda del canale A precede l'onda del canale B, allora il motore gira in senso orario, altrimenti in senso antiorario.[3]

## 2.3. MD10C – ENHANCED 10A DC MOTOR DRIVER


Il driver motori è una shield progettata per pilotare un motore in corrente continua, modulando la tensione mediante la tecnica PMW (Power Width Modulation). Dato un segnale PWM in ingresso restituisce in uscita una tensione continua pari al valore medio dello stesso.

[3] Gli approfondimenti del motore POLOLU RB-POL-126 sono riportati in [5]

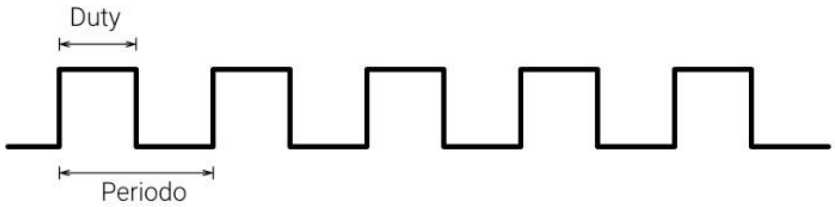
Nello specifico si è utilizzato il driver motori MD10C il quale ha in ingresso tre pin:

- GND: massa logica;
- PWM: PWM per la velocità di controllo;
- DIR: direzione di controllo;

Inoltre, ha una morsettiera a quattro pin:

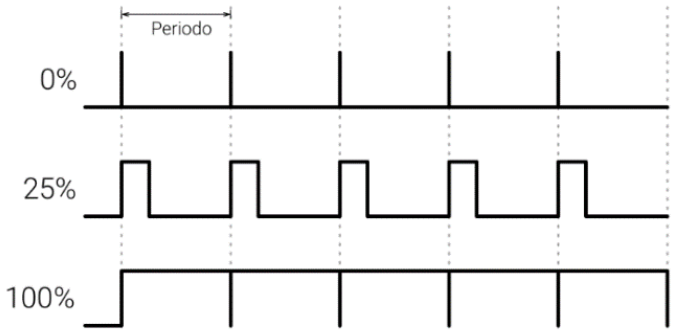
- POWER (+): alimentazione positiva;
- POWER (-): alimentazione negativa;
- Motor Output A: connesso al terminale A del motore;
- Motor Output B: connesso al terminale B del motore.[4]

La modulazione di larghezza di impulso o PWM (Pulse Width Modulation) è stata utilizzata per variare la velocità del motore. Infatti, questa tecnica, permette di ottenere una tensione media variabile dipendente dal duty cycle, ovvero il rapporto tra il periodo in cui il segnale è alto e il periodo totale considerato; calcolando il periodo T come: $T = \frac{1}{f} = T_{on} + T_{off}$ si ottiene il duty cycle $\delta = \frac{T_{on}}{T}$ .



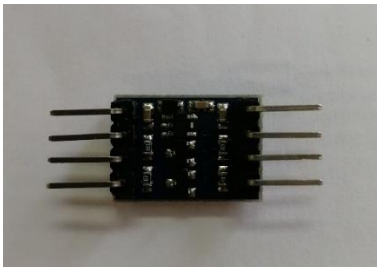

La tensione di uscita sarà pari a $V_{out} = V_{in} \cdot \delta = V_{in} \cdot \frac{T_{on}}{T}$.

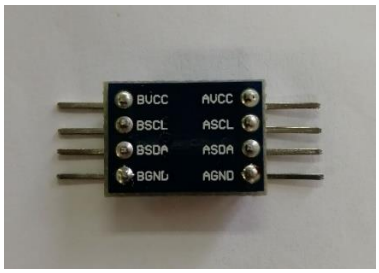
Nella seguente figura sono rappresentati alcuni esempi di duty cycle.




[4] Gli approfondimenti del driver motori MD10C sono riportati in [6]

## 2.4. DC/DC CONVERTER






Il convertitore DC/DC utilizzato converte una tensione continua in ingresso a 5V in una tensione continua in uscita a 3,3V. Si è scelto di utilizzarlo perché l'encoder genera un'onda quadra con tensione pari a 0V, per lo zero logico, e 5V, per l'uno logico, ma la scheda YRDKRX63N gestisce in ingresso segnali compresi tra 0V e 3,3V.

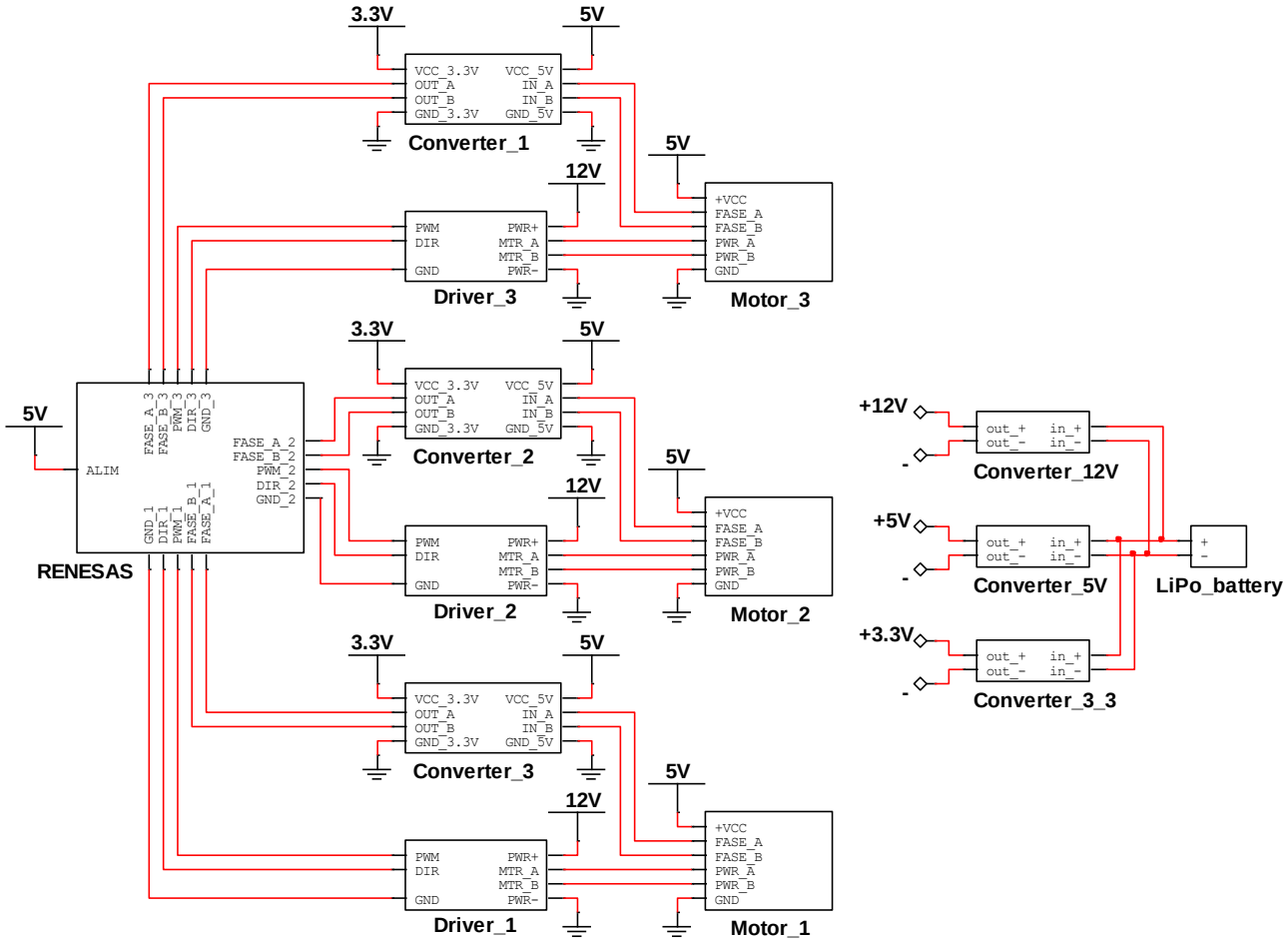
## 2.5. CABLAGGIO E COLLEGAMENTI

Si utilizzano tre tensioni di alimentazione differenti:

- 12V: alimentazione driver motori;
- 5V: alimentazione encoder;
- 3,3V: alimentazione convertitori DC/DC.

Il driver motori, in input, necessita di un segnale PWM e di un segnale, di livello logico alto o basso, per impostare il verso di rotazione del motore, oltre a richiedere una massa logica di riferimento; il tutto viene generato settando correttamente i pin della scheda di sviluppo. In uscita dal driver si ha la tensione da mandare in input al motore, il quale inizierà a ruotare. A questo punto l'encoder genererà due onde quadre in relazione con la velocità ed il verso di rotazione. Esse verranno convertite mediante l'utilizzo del convertitore DC/DC per poi essere mandate in input alla scheda di sviluppo.
Un altro tipo di convertitore DC/DC è stato utilizzato per partizionare la tensione di alimentazione della batteria; nello specifico ne sono stati utilizzati tre per fornire ai diversi componenti del circuito elettrico le specifiche tensioni riportate nell'elenco soprastante.

Di seguito è riportato il circuito elettrico dei collegamenti sopra descritti:




Dal circuito elettrico sopra riportato è possibile notare il partizionamento della tensione della batteria descritto in precedenza.

Nello specifico, nella seguente tabella, vengono indicati i collegamenti da/verso la scheda:

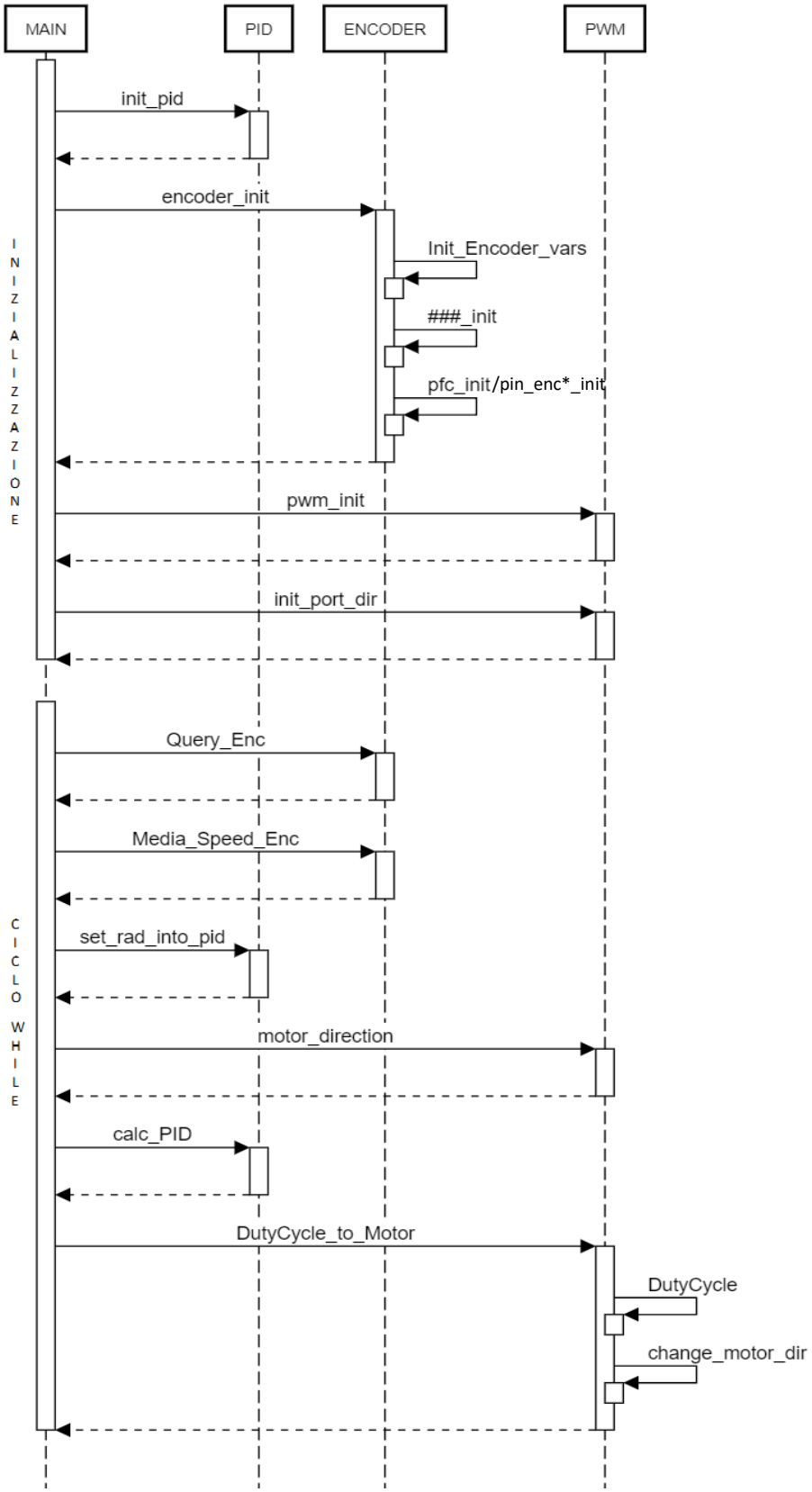
| | MOTORE 1 | MOTORE 2 | MOTORE 3 |
|---|---|---|---|
| ENCODER | Fase A: pin 25 JN2<br>Fase B: pin 6 JN2 | Fase A: R42<br>Fase B: pin 23 J8 | Fase A: R37/R36<br>Fase B: pin 9 JN2 |
| PWM | Pin 22 JN2 | Pin 15 J8 | Pin 23 JN2 |
| DIREZIONE | Pin 19 JN2 | Pin 16 JN2 | Pin 7 JN2 |

La scheda acquisisce in input le rispettive due fasi dell'encoder dei tre motori; in output invia un segnale PWM per ogni motore ed inoltre imposta il pin relativo al verso di rotazione al livello logico alto o basso.

# Capitolo 3. SOFTWARE

Di seguito si descrive il funzionamento del software sviluppato per la reazlizzione del progetto.

## 3.1. SCHEMA DELLE FUNZIONI



Per descrivere la funzionalità del software e come le funzioni interagiscono tra loro si è deciso di utilizzare un diagramma di sequenza basato sul linguaggio UML.
Con il seguente diagramma viene rappresentato l'andamento temporale delle funzioni utilizzate; successivamente verranno descritte nel dettaglio. Per motivi di limitazione hardware legati al numero di timer per gestire tre encoder e tre PWM è necessario usare per la gestione due periferiche di temporizzazione diverse, MTU e TPU. Questo comporta la necessità di due inizializzazioni diverse nella funzione *encoder_init*.
Si noti che nell'inizializzazione dell'encoder è presente una funzione *###_init*, dove ### sta ad indicare le unità MTU e TPU, poiché si è deciso di utilizzarle per gestire l'encoder. MTU e TPU, rispettivamente Multi-fuction Timer pulse Unit e Timer Pulse Unit, sono unità utilizzate per la gestione degli interrupt generati dagli encoder.

## 3.2. DESCRIZIONE DELLE FUNZIONI

Di seguito sono descritte le funzioni utilizzate nella stesura del software:

- *PID.h*

  Header file che contiene la dichiarazione di tutti i prototipi delle funzioni che si occupano della gestione del PID. Si ha un unico file per tutti e tre i PID che vengono controllati singolarmente mediante la dichiarazione di una struttura *PIDSt_Type_t* che verrà inizializzata poi tre volte, una per PID.

  - *init_pid*: Inizializza i valori del PID. Viene richiamata tre volte poiché ogni ruota viene gestita in maniera autonoma.
  - *calc_PID*: Calcola i valori che poi verranno modificati all'interno della struttura e che gestiranno la velocità del motore. Viene richiamata tre volte e prende come parametro in ingresso ogni volta una struttura diversa in modo tale che vengano modificati soltanto i valori del motore interessato.
  - *set_rad_into_pid*: Setta il valore della retroazione calcolato dalla funzione calc_PID all'interno della struttura. Viene richiamata tre volte andando a settare i valori per i tre motori.

- *PWM.h*

  Header file che contiene la dichiarazione di tutti i prototipi delle funzioni per la gestione del segnale PWM. Vengono definite delle costanti che verrano utilizzate per i settaggi e per i calcoli dell'onda PWM che andrà ad azionare il motore.

  - *PWM_init*: Inizializza i canali per il segnale PWM sulla scheda RENESAS. Viene richiamata tre volte perché ogni segnale PWM gestisce un motore diverso.
  - *Init_Port_Dir*: Inizializza le porte della scheda RENESAS come uscite e quindi setta i rispettivi pin nella maniera corretta.
  - *motor_direction*: Calcola la direzione del motore, senso orario se la velocità è positiva, senso antiorario se la velocità è negativa. Salva il risultato su una variabile che successivamente verrà utilizzata per settare la direzione del motore.
  - *DutyCycle_to_motor*: Calcola il valore del duty-cycle per generare il segnale PWM. Viene richiamata una sola volta ma esegue tre volte i calcoli utilizzando tre strutture diverse, una per ogni motore.
  - *DutyCycle*: Setta i valori dei registri della scheda RENESAS con il valore del duty-cycle calcolato.
  - *change_motor_dir*: Imposta sul registro della scheda RENESAS il bit che gestisce la direzione di rotazione del motore.

- *encoder_MOT*.h*[5]

  Header file che contiene la dichiarazione di tutti i prototipi delle funzioni per la gestione degli encoder. Si dichiara una struttura *encoder_data_t_mt_** contenente le variabili necessarie per la lettura degli encoder. Ci sono tre file header, uno per ogni encoder, ma ne viene descritto solo uno poiché alcune funzioni, pur avendo nomi diversi, sono implementate in egual modo; questo è dovuto al fatto che sulla scheda RENESAS la gestione di più encoder è assegnata

[5] Con * si indica il numero dell'encoder (1,2,3)

a unità diverse, MTU e TPU, che gestiscono in maniera differente i registri al proprio interno.

- *encoder_*_init*[5]: Inizializzazione dei parametri dell'encoder.
- *Init_Encoder_*_vars*[5]: Inizializzazione delle variabili utilizzate dall'encoder.
- *tpu_enc*_init*[5] (per encoder 1 *mtu2_init*): Inizializzazione della periferica sulla scheda RENESAS per l'utilizzo dell'encoder.
- *pin_enc*_init*[5] (per encoder 1 *pfc_init*): Inizializzazione dei pin sulla scheda RENESAS per l'utilizzo degli encoder.
- *Query_enc_**[5]: Calcolo delle variabili necessarie per la lettura degli encoder mediante l'utilizzo di interrupt opportunamente generati che permettono di ottenere una lettura della velocità istantanea della ruota.
- *Media_Speed_Enc_**[5]: Per ottenere un valore più preciso della velocità della ruota si utilizza una media tra gli ultimi 100 valori letti.

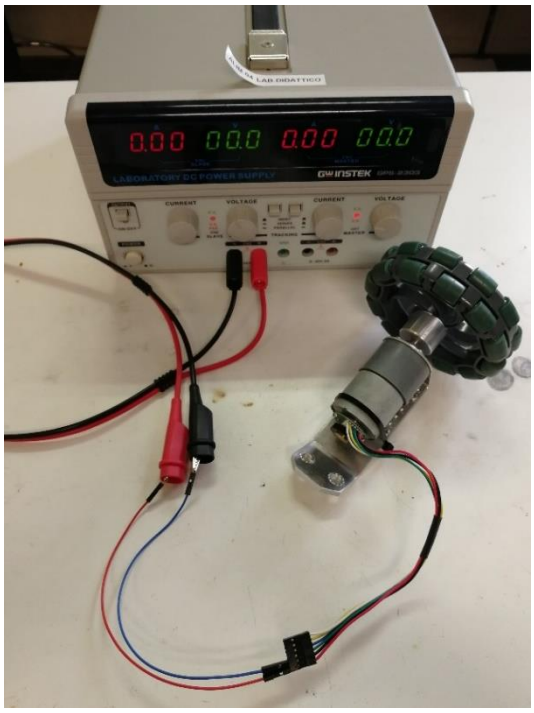
Si è cercato di realizzare un software che fosse il più ottimizzato e compatto possibile per facilitare eventuali future modifiche. Il codice sviluppato per la realizzazione del software è in linguaggio C.[6]

[5] Con * si indica il numero dell'encoder (1,2,3)
[6] Il codice è riportato in [7]

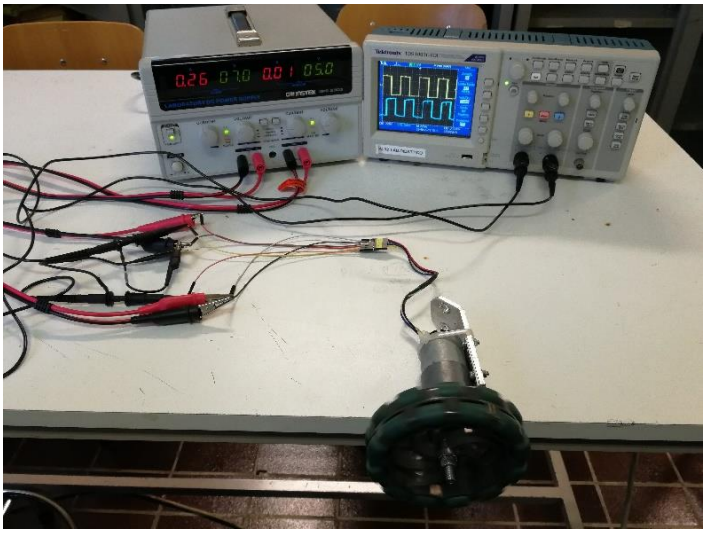
# Capitolo 4. TEST IN LABORATORIO

Si sono svolti diversi test, ognuno di essi per verificare il corretto funzionamento di ogni componente e la conseguente interazione con gli altri componenti.

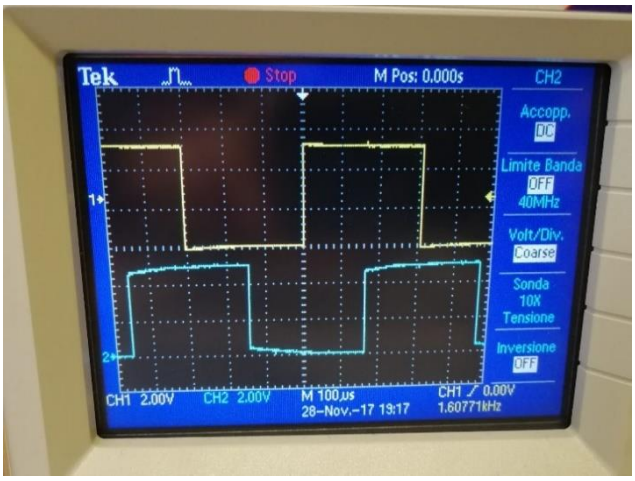
- La prima prova è stata svolta per verificare il corretto funzionamento del motore RB-POL-126 controllato con una tensione continua erogata da un alimentatore stabilizzato. Si è variata la tensione di alimentazione del motore nel suo range di funzionamento, ovvero tra 0V e 12V, al fine di osservare e verificare la corretta variazione di velocità per ogni valore di tensione.



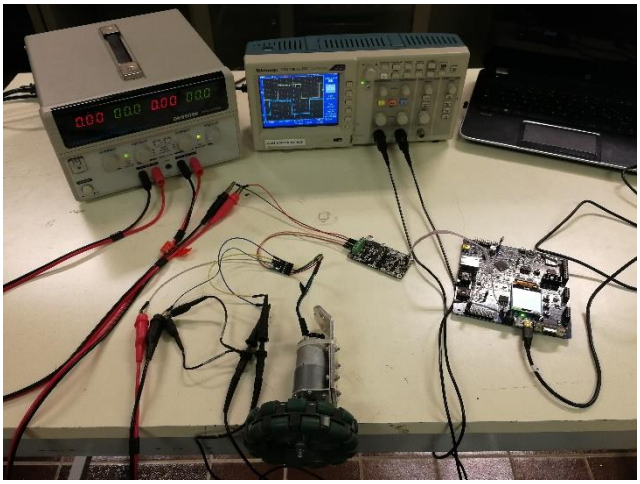
- Come seconda prova si è alimentato l'encoder e si sono analizzate, attraverso l'utilizzo di un oscilloscopio, le onde generate; si è verificata, quindi, la corretta corrispondenza ai valori della tensione di ingresso, per esempio:
  0V → duty cycle 0%;
  6V → duty cycle 50%;
  12V → duty cycle 100%.



Nella foto di seguito, notiamo un duty cycle del 50% il quale corrisponde ad una tensione di alimentazione di 6V.

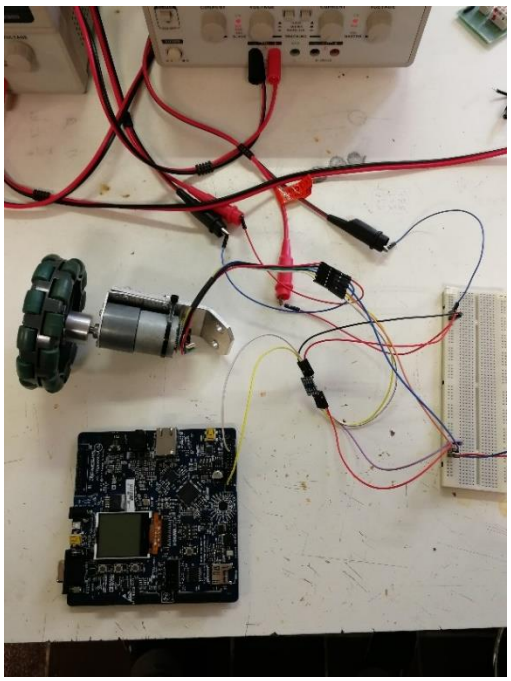



- Per questa prova si sono introdotti la scheda RENESAS e il driver motori: si è utlizzata la scheda RENESAS per generare un segnale PWM il quale viene convertito dal driver motori in una tensione continua per il motore.



Le onde generate dall'encoder sono state analizzate sempre con l'utilizzo dell'oscilloscopio per continuare a verificarne il corretto funzionamento.

- In questa prova l'onda generata dall'encoder è stata elaborata dalla scheda RENESAS, dopo essere stata convertita dal convertitore DC/DC da 5V a 3.3V, poiché la RENESAS accetta in ingresso tensioni fino a 3.3V.
  Si è quindi riuscito a leggere, dopo le dovute elaborazioni, la velocità angolare in rad/sec sullo schermo della RENESAS.

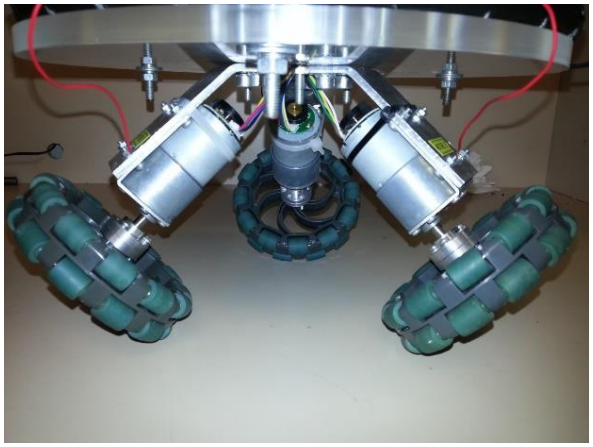


Di seguito alcuni valori di riferimento:

- 3V corrispondono circa 2.7rad/sec;
- 6V corrispondono circa 5.8rad/sec;
- 9V corrispondono circa 8.8rad/sec;
- 12V corrispondono circa 11.5rad/sec.

- Si è, quindi, introdotto il PID, implementato a livello software, come descritto in precedenza e si è verificato il suo corretto funzionanento: dato in ingresso un valore di riferimento, una velocità in rad/sec, il sistema deve riuscire a raggiungere e a mantenere costante quel valore.

- La prova finale è consistita nell'utilizzare tutti e 3 i motori contemporaneamente, leggendo le 3 velocità sullo schermo della RENESAS ed alimentando il tutto con una batteria ai polimeri di litio.




Con quest'ultima prova si è, quindi, conclusa la fase di progettazione e di test del sistema di controllo in tensione della velocità.

RIFERIMENTI:

[1] “Discrete_PID”
http://portal.ku.edu.tr/~cbasdogan/Courses/Robotics/projects/Discrete_PID.pdf
[2] “Hardware_manual_RX63N”
https://www.renesas.com/br/en/doc/products/mpumcu/doc/rx_family/r01uh0041ej0180_rx63n631.pdf?key=8465840296d9fc016aa3a9ea2f0069a4
[3] “User_manual_YRDKRX63N”
https://www.renesas.com/eu/en/doc/products/tools/r20ut2532eu0100_yrdkrx63n_um.pdf
[4] “Schematics_YRDKRX63N”
https://github.com/mattmunee/RenesasYRDKRX63N/blob/master/Schematics_YRDKRX63N_SCH_3.pdf
[5] “Datasheet_RB-POL-126”
https://www.scribd.com/document/155847469/Datasheet-motor-12V-With-Encoder
[6] “Driver_MD10C”
https://www.robotshop.com/media/files/PDF/user-manual-md10c-v2.pdf
[7] “CODICE”

Gli allegati sono contenuti all’interno del CD-ROM.