UNIVERSITÀ POLITECNICA DELLE MARCHE
FACOLTÀ DI INGEGNERIA
Corso di Laurea in Ingegneria Informatica e dell'Automazione
Anno Accademico 2018-2019

SELF BALANCING BICYCLE: CONTROLLO IN COPPIA CON LETTURA DELLA VELOCITÀ DEL MOTORE DI TRAZIONE

Studenti **:**
Daniele Polucci
daniele.polucci@gmail.com;

Emanuele Fares
ema9.7@hotmail.it;

Michele Brocchini
-

Professore :
Andrea Bonci

INDICE:.

Sommario

INTRODUZIONE

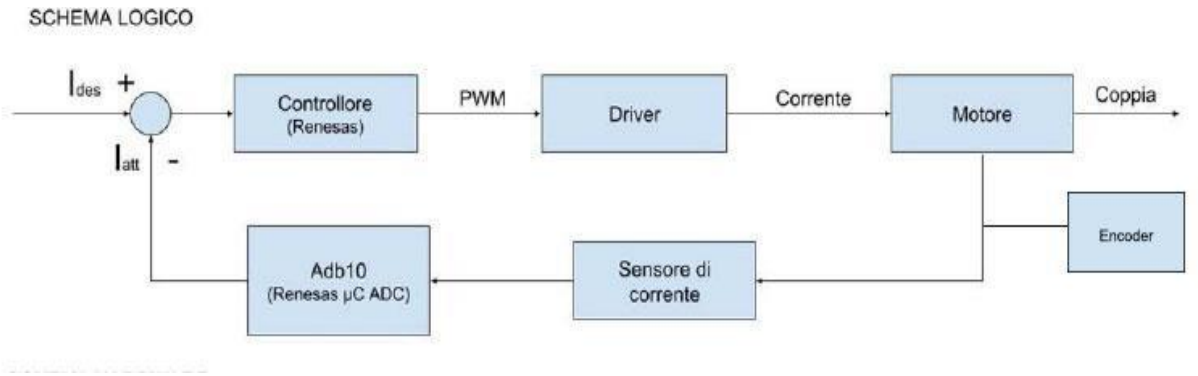
Questa relazione presenta lo studio e l'implementazione, hardware e software, del controllo in coppia del motore di trazione di una bicicletta automatizzata, la cui guida avviene senza l'ausilio del ciclista.

Il nostro obiettivo è quello di controllare la coppia del motore di trazione affinché la velocità tangenziale sulla ruota della bicicletta sia tale da permettere la stabilità della stessa.

Secondo il lavoro condotto da studenti precedenti, la velocità minima per raggiungere tale obiettivo è di 1.8 m/s, che considereremo come la velocità desiderata sulla quale basare i nostri studi.

Il motore installato precedentemente permetteva una velocità massima di 1 m/s e la bicicletta non rimaneva perfettamente in equilibrio, pertanto lo abbiamo sostituito con il motore presentato all'interno di questa relazione.

STRATEGIA DI CONTROLLO

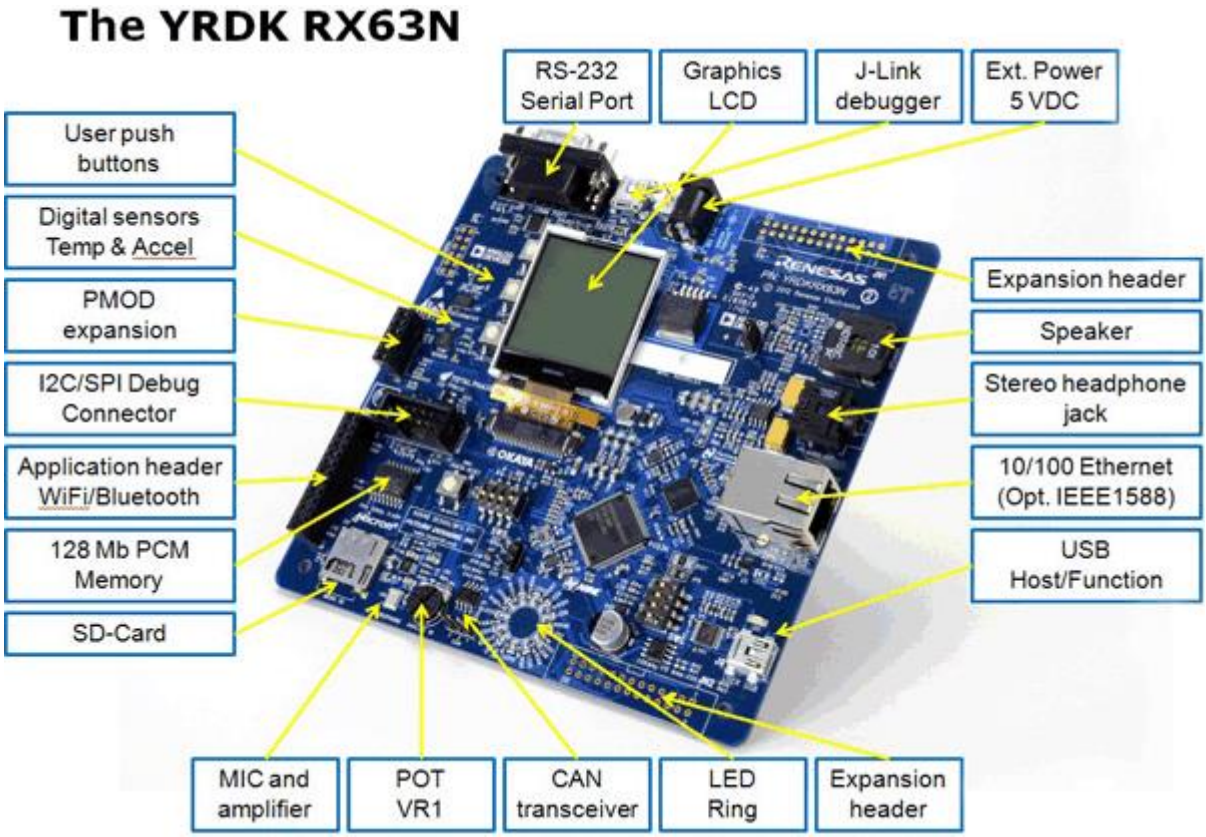


Lo schema a blocchi presentato rappresenta il sistema a catena chiusa di regolazione automatica del motore: nella retroazione un sensore di corrente permette di leggere la corrente assorbita dal motore per ottenere una certa coppia. Tale valore viene poi confrontato con l'ingresso di riferimento, una corrente che si vuole venga assorbita dal motore, di modo che, ogni volta che si verifica una differenza tra il segnale di uscita e il segnale di riferimento, si produce un'azione correttiva che riporta l'uscita al valore desiderato (eseguita dal PID).

Nel frattempo l'encoder del motore permette di effettuare una lettura della velocità angolare del motore, affinché si possa avere una verifica dell'ottenimento dei risultati desiderati in velocità.

STRUMENTI UTILIZZATI:

A. RENESAS:

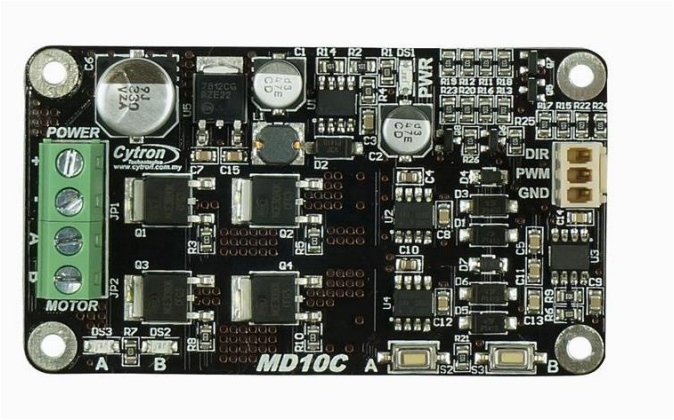


Per quanto riguarda il controllo e l'elaborazione dei dati, è stata utilizzata un'interfaccia prodotta dalla Renesas, la demonstration Kit YRDKRX63N.

YRDKRX63N è la scheda di sviluppo su cui è montato il controller RX63N ad alte prestazioni; core della famiglia RX631 che include:

- 32-bit MCU capace di operare oltre i 100 MHz;
- FPU (Floating-PointUnit) per i calcoli aritmetici;
- Oltre 21 canali per ADC a 12-bit e oltre 2 canali per DAC- unità Timers;
- MTU2 [Multi Timer Unit Function], con funzioni di: input capture / output compare / counter clearing per generazione di segnali PWM, e controllo motori;
- watchdog timer Independent e funzione CRC per lo standard di applicazioni domestiche (IEC 60730);
- molte funzioni di comunicazione: Ethernet, SCI, RSPI, CAN, I2C;

B. MD10C – ENHANCED 10° DC MOTOR DRIVER



Il driver motori è una scheda progettata per pilotare un motore in corrente continua , usando come input per la regolazione della velocità un segnale PWM proveniente dal microcontrollore. La scheda restituisce in uscita una tensione continua pari al valore medio del segnale PWM.
La MD10C ha le seguenti caratteristiche:

- Controllo bidirezionale per il motore DC;
- Tensione di alimentazione che va da 3V a 25V;
- Sopporta fino a 10A continui e 15A di picco massimo di corrente(10 secondi);
- 3,3 V e 5V ingresso livello logico;
- I componenti a stato solido forniscono tempi di risposta più veloce;
- Frequenza di controllo PWM fino a 10kHz;
- Gestisce modulo e segno PWM;
- Dimensioni: 75mm x 43mm;

Essa ha per ingresso cinque pin:

- GND: massa logica;
- PWM: PWM per la velocità di controllo;
- DIR: direzione di controllo;
- NC: non connesso. Questo pin non è usato;
- VIN: pin di alimentazione scheda;

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;

Nel resto della scheda troviamo:

- LED A(rosso): si accende quando l'output di A è alto e quello di B è basso. La corrente circola da A a B;
- LED B(rosso): si accende quando l'output di A è basso e quello di B alto. La corrente circola da B ad A;

- POWER LED (verde): si accende quando la scheda è alimentata;
- TEST BUTTON A: quando viene premuto la corrente circola da A a B e il motore girerà in senso orario(o antiorario in base alla connessione);
- TEST BUTTON B: quando viene premuto la corrente circola da B a A e il motore girerà in senso antiorario(o orario in base alla connessione);
- JUMPER: selettore di alimentazione scheda
    - PWR: La scheda è alimentata dall'alimentazione del motore. Si può utilizzare solo quando il motore alimentato con un ingresso maggiore di 14V;
    - VIN: La scheda è alimentata da VIN. Bisogna avere 12V di ingresso su VIN per far si che l'alimentazione di ingresso del motore possa essere compresa tra i 3V ai 25V.

C. MOTORE 12V DC RB-Dfr-439 146 rpm con Encoder

Il motore a corrente continua RB-Dfr-439 con codificatore è uno dei motori DC di alta qualità personalizzati da DFRobot. È un motore silenzioso con uscita a coppia elevata con encoder incorporato. L'encoder Hall fornisce 663 impulsi per rotazione ed è in grado di rilevare una rotazione di 0,54 gradi dall'albero. Questa soluzione soddisfa i requisiti generali di controllo rendendola una scelta ideale per la robotica.
Il motore ha le seguenti specifiche:

Rapporto di trasmissione: **51**: 1
Tensione nominale: 12V
No load RPM(dopo riduttore) @12V: 146 rpm ≃ **15,29 rad/s**
No load RPM (prima riduttore) @12V: 8000 rpm
Corrente di carico assente @12V: 0,23 A
Coppia nominale @12V: 139 oz-in ≃10kg.cm≃ **980mNm**
Velocità nominale (dopo riduttore) @12V: 73 rpm ≃ **7,6 rad/s**
Corrente di stallo: **3,6 A**
Coppia di stallo: 556 oz-in ≃ 3,92 Nm
Encoder Resolution: **13 PPR** (663 PPR per albero di cambio)

Encoder Hall a due fasi.
Dimensioni: 123x36x36 mm
Peso: 270g
Tipo di albero: a forma di D

Il datasheet originale è riportato in [8];

C.1. Encoder

Insieme al nostro motore RB-Dfr-439 vi è incluso un encoder ad effetto hall.
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.
Le due onde vengono lette dalla Renesas grazie all'uso di registri MTU o TPU che sono gli stessi per l'uso del PWM.

C.2. Encoder Adapter FIT0324

Questo Encoder Adapter è progettato appositamente per i motori DC DFRobot 12V. Esso comprende due resistori di pull-up esterno che sollevano la tensione di uscita quando il transistor è spento.

C.3. DC/DC Converter

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

C.4. ALCUNI CALCOLI UTILI : La velocità angolare desiderata.

Osserviamo di seguito alcuni calcoli teorici necessari ricavare la velocità angolare desiderata dal nostro motore affinché la bicicletta si muova ad 1.8 m/s.

$$\boldsymbol{V_c = w_c * R_c}$$

Velocità tangenziale sulla corona.

$$V_p = w_p * R_p$$

Velocità tangenziale sul pignone.

$$V_c = V_p \text{ e}$$

$$\frac{R_c}{R_p} = \frac{18}{28} \quad => \quad \frac{w_c}{w_p} = \frac{18}{28}$$

$$V_r = w_p * R$$

Se $Vr = 1.8\, m/s$ allora deve essere

$$w_p = 6{,}254\, rad/s.$$

L'attuale motore genera una coppia massima di *0.98 Nm* con una velocità di *146 rpm* (con carico): la velocità angolare dell'albero motore sarà dunque $w_c = 15{,}29\, rad/s$.

Essa sarà anche la velocità angolare della corona che ha un raggio(*Rc*) di 0,0575 m.

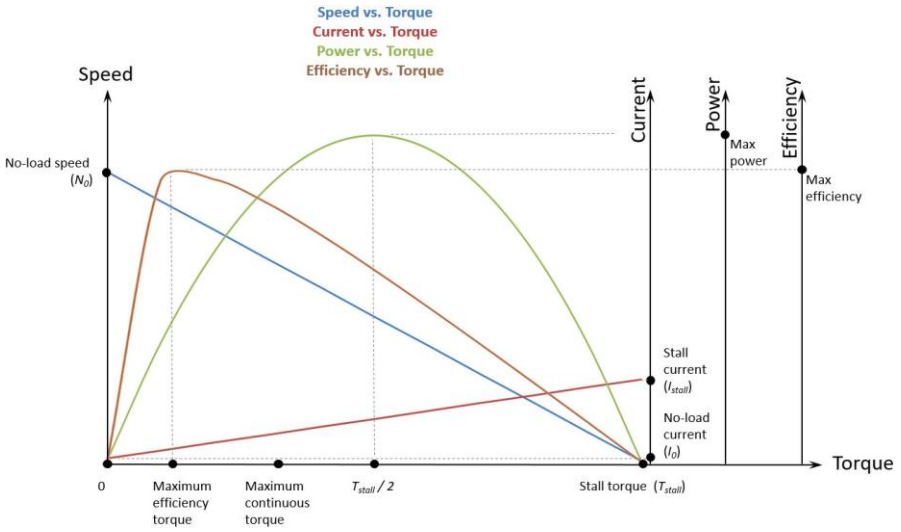
Il rapporto dei raggi corona/pignone è di *18/28* e dalla formula della velocità tangenziale, poiché la velocità tangenziale sul pignole è uguale a quella sulla corona, anche il rapporto $w_c/w_p = 18/28$.

Il raggio della ruota della bicicletta *R* è 0,185 m.

Dalla velocità desiderata è quindi possibile dedurre la velocità angolare alla quale vogliamo far girare il nostro motore.

Per la velocità desiderata di *1.8 m/s* abbiamo bisogno di una velocità angolare sulla corona di circa *6,254 rad/s.*

C.5. ALCUNI CALCOLI UTILI: La costante di coppia del motore RB-Dfr-439



Per il controllo della coppia del motore in corrente è inoltre utile trovare alcuni parametri dello stesso per poterlo così modellare via software.
Dal datasheet è stata ricavata la costante di coppia del motore che, come vedremo nel capitolo successivo, è indispensabile per la scelta dei parametri del PID attraverso cui controllarlo.
Osservando le curve delle relazioni dei parametri dei motori DC, riportate nel grafico qui sopra, abbiamo applicato la relazione

$$\boldsymbol{T = K_T * (I - I_o)}$$

Dove T è la coppia del motore, K_Tè la costante di coppia, I è la corrente di armatura a quella coppia del motore, I_o è la corrente di armatura a carico assente.
Sono stati usati, come dati, la coppia di stallo (3,92 Nm) e la corrente di stallo (3,6 A).Abbiamo così ottenuto una costante di coppia di 1,08 (N*m)/A.

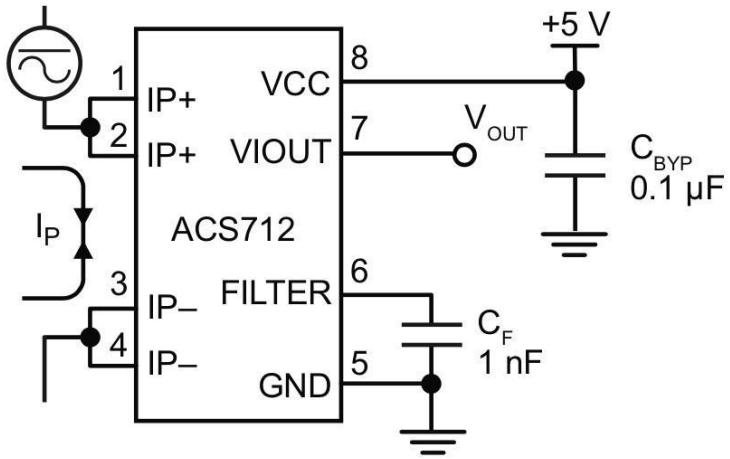
D. SENSORE DI CORRENTE:

Nel corso della progettazione sono stati utilizzati due diversi sensori di corrente e confrontati tra loro: l'ACS712 e il MAX471 GY-471.

D.1. ACS712

Il primo sensore utilizzato per la realizzazione della Self Balancing Bicyle, è un ACS712 prodotto dalla Allegro MicroSystems. Di tale integrato esistono 3 versioni a seconda della massima corrente misurabile (5-20-30A); per la realizzazione della Self Balancing Bicycle ed anche per le prove in laboratorio è stato utilizzato il modello da 5A.

Sul *datasheet* è riportato lo schema tipico:



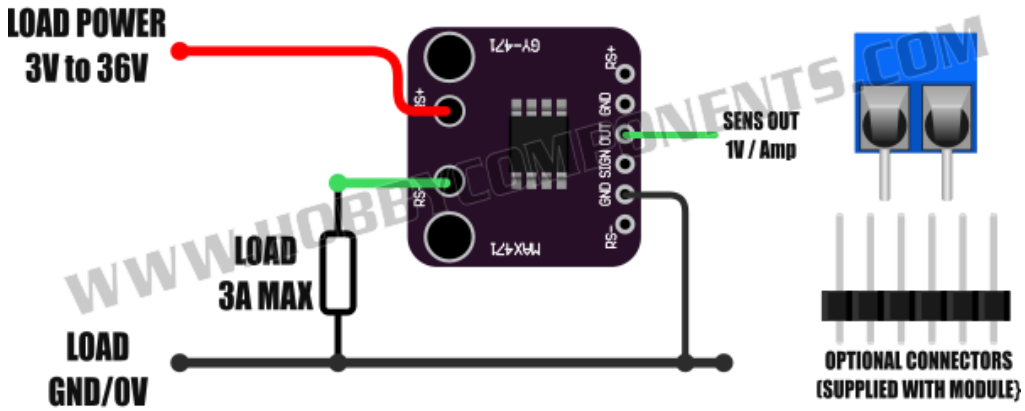
Il sensore necessita di un'alimentazione a 5V, e viene collegato in serie tra il driver del motore ed il motore stesso.
Dal connettore OUT sulla schedina **esce una tensione di valore compreso tra 1,575V e 3,425V** direttamente proporzionale all'intensità della corrente che attraversa il sensore. L'ACS712 è un sensore bidirezionale, ciò significa che è in grado di misurare correnti in ingresso (AIN) che circolano in entrambi i versi entro un range che va da -5 A a +5 A.

La costante di proporzionalità che lega la corrente AIN alla tensione in uscita dal pin OUT del sensore, viene definita *Sensitivity (S).* Essa assume valore pari a 185 mV/A da costruttore. È possibile verificare quanto detto, moltiplicando la S per la variazione massima della corrente AIN (10A, -5 A ÷ +5 A), ottenendo così 1,85 V, che è esattamente lo scostamento massimo (1,575 V ÷ 3,425 V) della tensione in uscita dal pin OUT del nostro sensore. Ci denota che per un valore di corrente AIN uguale a 0, avremo una tensione di 2,5 V, anche denominata come tensione di Offset (Vo).

Le relazioni tra tensione d'uscita e corrente d'ingresso sono quindi date dalle seguenti espressioni:
1) $V" out = Vo + S*Ain$
2) $A" in = (Vout – Vo) / S$

Il nostro motore assorbe una corrente inferiore ai 3 A. Quindi abbiamo sviluppato il progetto una seconda volta utilizzando i sensori MAX471 GY-471 come suggerito e analizzato in [6].

D.2. MAX471 GY-471
Il secondo sensore utilizzato è il MAX471 montato sulla scheda GY-471.
In confronto al precedente non ha bisogno di una alimentazione esterna, ma si alimenta attraverso la corrente che riceve nel pin R+.
Si noti che vi sono due ingressi R+ e due uscite R-, rispettivamente a sinistra e a destra del sensore. Come è mostrato in [12], il datasheet del MAX471, i due R+ sono in corto tra di loro e anche i due R-, in tal modo è possibile "leggere" la corrente da un lato o dall'altro della GY-471.
La differenza principale con i sensori precedenti è una sensitività di 1 V/A per una capacità di lettura tra -3 A a + 3 A,(con ampio errore, come indicato in [12] pag. 4-5, tra + e - 100 mA) e quindi un Offset di 0 V.



L'uscita con un rapporto 1:1 degli ampere in ingresso ha reso molto più facile le verifiche e l'implementazione del controllo.
L'ipotesi di utilizzo di questo sensore per gli studenti che hanno approcciato lo stesso tipo di controllo sul ballbot (riferimento [6]) ci ha messo di fronte al seguente problema: considerando che la variazione è di 6 Ampere, la nostra scala raggiungerà minimo i 6 Volt, non supportati da Renesas. Perciò potrebbe occorrere l'implementazione di un circuito che

adatti la tensione ad un valore massimo di 3,3 Volt attraverso l'uso di un amplificatore operazionale esterno.

> Per la nostra progettazione, poiché non abbiamo bisogno di considerare una direzionalità doppia (avanti e indietro) della ruota posteriore, abbiamo deciso di considerare solo i valori di corrente tra 0 e 3 A, quindi un ingresso tra 0 e 3V.

I sensori danno in uscita una tensione analogica, mentre la Renesas esige input in formato digitale quindi è stato utilizzato il convertitore analogico-digitale a 10 bit(ADb10) già presente in essa.

E. ADb10

È un convertitore analogico/digitale a 10bit integrato nella Renesas. Questo significa che è possibile lavorare su 1024 livelli (da 0 a 1023) di quantizzazione del segnale. Nel nostro caso il convertitore viene usato per la conversione del segnale analogico letto dal sensore di corrente. Esso viene così trasformato in digitale per essere elaborato dal microcontrollore.
La seguenti tabelle schematizza la correlazione tra il valore di tensione elaborato dal sensore di corrente e il livello logico del convertitore.

ADb10 per ACS712

1024 Livelli Logici

Sensore di Corrente		***ADB10 (Renesas)***
Corrente in ingresso	***Valore di tensione al µC***	0 - 469 (inutilizzati)
-5 A	1,575 V	470
0 A	2,5 V	747
+5 A	3,425 V	1023

ADb10 per MAX471 GY-471

1024 Livelli Logici

Sensore di Corrente		***ADB10 (Renesas)***
Corrente in ingresso	***Valore di tensione al µC***	
0 A	0 V	
3 A	3 V	931
Non letta	3 V ++	932-1023(non utilizzati)

I valori ricevuti dal sensore di corrente sono elaborati da un filtraggio FIR (finite impulse response - risposta infinita all'impulso). Esso è implementato a livello di codice ed effettua una media mobile ponderata sostituendo ad ogni nuovo valore letto la media di tutti i valori nella finestra di campionamento, pesati in ordine di lettura: quindi, il più recente avrà un peso maggiore rispetto ai valori meno recenti.
Per maggiori dettagli si veda nel capitolo **Software: ADb10.**

F. IL 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:

$$\begin{cases} u(t) = K_P e(t) + K_I \int_{t_0}^{t} e(\tau)\, d\tau + K_D \frac{de(t)}{dt} \\ u(s) = \underbrace{K_P e(s)}_{u_P(s)} + \underbrace{K_I \frac{e(s)}{s}}_{u_I(s)} + \underbrace{K_D s e(s)}_{u_D(s)} \end{cases}$$

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 successivamente lo si è discretizzato con passo Ts, dove quest'ultimo è il periodo di campionamento.

Sono state effettuate due tarature PID rispettivamente per il sensore **ACS712** e per il sensore **MAX471.**

F.1. PID per ACS712

Per la taratura dei parametri del primo PID si è utilizzata una simulazione Simulink (un software per modellare, simulare e analizzare sistemi dinamici che è strettamente integrato con Matlab) assieme ad alcuni test per la taratura reale e manuale in laboratorio.
La simulazione è la riproposizione per il nostro motore della simulazione utilizzata dagli studenti Brutti,Ciancaglione,Di Muzio,Iezzi in [7].

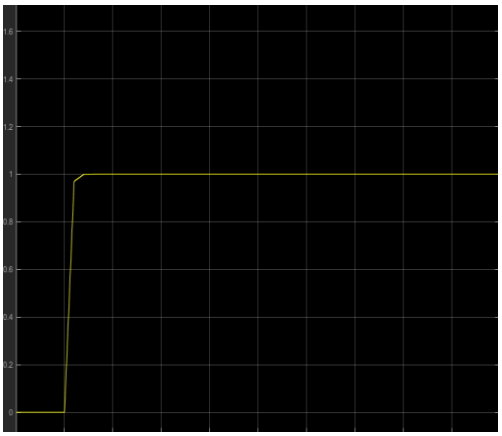
Non avendo potuto effettuare una identificazione parametrica per ottenere la funzione di trasferimento del motore, si è provato a modellare il motore nel modo seguente

```
1   function y = motore(u)
2 -   Km = 1,08;
3 -   y = Km*u;
4
```

Dove *Km* [Nm/A] è la costante di coppia del motore ottenuta nei calcoli del capitolo **C.5**, *y* è la coppia e *u* è l'uscita controllata dal PID.

Nella simulazione si è eseguito il primo metodo di Ziegler-Nichols, verificando, tramite l'oscilloscopio virtuale di Simulink, l'uscita del sistema.

Variando i parametri si è ottenuto il segnale di uscita, il più vicino possibile al segnale di uscita ideale, ovvero un segnale a gradino. Qui a lato 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=0.32 e K_D=0.

Senza una funzione di trasferimento corretta del motore, già dai primi test, risulta evidente che il PID ottenuto dalla simulazione non è del tutto corretto.
Dopo aver effettuato diversi test inserendo i valori manualmente nel software si sono ottenuti dei parametri efficienti per il controllo reale:

Parametri (utilizzati) PID per sensore ACS712

KP	**5**
KI	**0.3**
KD	**1**

Si noti come nella simulazione si abbia utilizzato un $K_D = 0$, mentre nella situazione reale si necessita di un $K_D = 1$.
Pertanto in mancanza di una identificazione dei parametri del motore sconsigliamo l'uso della simulazione, sia per il tempo che si impiegherebbe nella modellazione e utilizzo che per una maggiore precisione del risultato cercato.

F.2. PID per sensore MAX471 GY-471

Per tale motivo per il sensore MAX471 la taratura è stata effettuata direttamente in modo manuale modificando il software, aiutati inoltre dalla migliore sensitività del motore che, come detto in **D.2** restituisce 1 V per ogni Ampere.

Parametri (utilizzati) PID per sensore MAX471 GY-471

KP	**1.9**
KI	**0.001**
KD	**1**

F.3. Osservazioni

Nel caso di un futuro utilizzo del motore RB-Dfr-439, effettuare l'identificazione parametrica potrebbe far risparmiare tempo nella taratura del PID.

G. CABLAGGIO E COLLEGAMENTI

Per il nostro obiettivo sono necessarie diverse tensioni di alimentazione:

- 12 V per alimentare il motore
- 5 V per alimentare l'encoder
- (5 V per alimentare il sensore di corrente ACS712)
- 5 V per alimentare la scheda Renesas

Si possono utilizzare diverse soluzioni per alimentare i vari sensori, **l'importante è che il ground sia comune a quello della scheda**. Nello schema successivo al paragrafo è stato riportato il collegamento utilizzato per i nostri test da banco; osserviamo che abbiamo alimentato il sensore di corrente attraverso il PIN 1-JN1 della Renesas che da in output una tensione di 5V, mentre gli altri componenti sono stati alimentati esternamente con un alimentatore o alternativamente con una batteria e il loro ground collegato alla scheda. Il convertitore DC/DC non è strettamente necessario al funzionamento del sistema; infatti la Renesas, che all'ingresso può gestire segnali fino a 3,3V, tronca autonomamente il segnale alto proveniente dall'encoder del valore di 5V. L'uso del convertitore rimane comunque più corretto in quanto evita lo "stress" dei pin d'ingresso a causa di tensioni più alte.

Analizziamo come sono collegati i componenti con la scheda Renesas:

Sensore di corrente ACS712

- **JN1-PIN 1 :** alimentazione dalla Renesas al sensore di corrente (5V).
- **JN1-PIN 4 :** ground per il sensore di corrente.
- **JN1-PIN 16:** pin di ingresso alla renesas per l'uscita analogica del sensore di corrente.

Sensore di corrente MAX471 GY-471

- **J16-PIN 11:** ground per il sensore di corrente.
- **JN1-PIN18:** pin di ingresso alla renesas per l'uscita analogica del sensore di corrente.

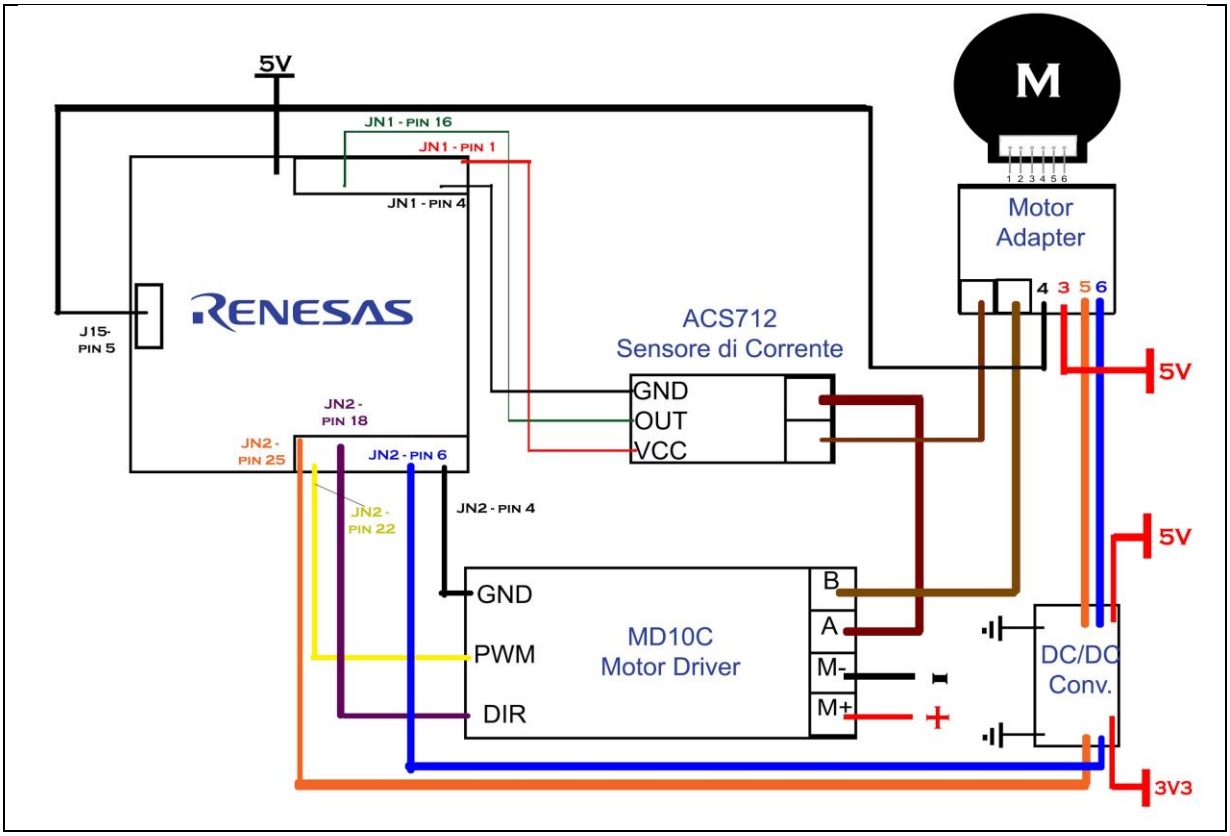
Driver Motore

- **J15-PIN 11:** ground per il driver motore.
- **JN2-PIN 23:** pin di uscita del PWM dalla Renesas e in ingresso al driver motore

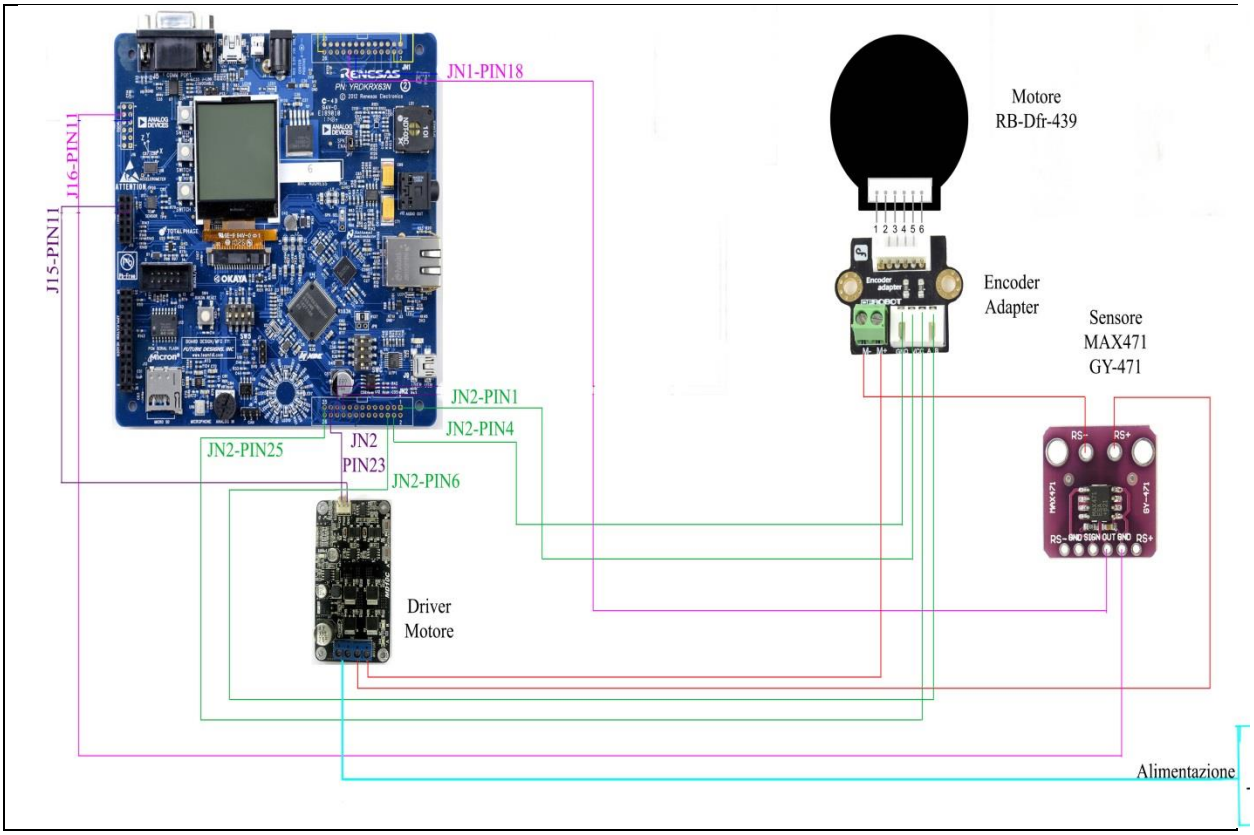
Encoder

- **JN2-PIN 25:** ingresso del segnale della fase A dell'encoder verso la Renesas (cavo 5)
- **JN2-PIN 6:** ingresso del segnale della fase B dell'encoder verso la Renesas (cavo 6)
  Attenzione: scambiare le due fasi cambia il verso di lettura dei segnali.
- **JN2-PIN 4:** ground per l'encoder
- **JN2-PIN 1:** alimentazione 5V per encoder

G.1. Cablaggio per i test da banco con il sensore ACS712

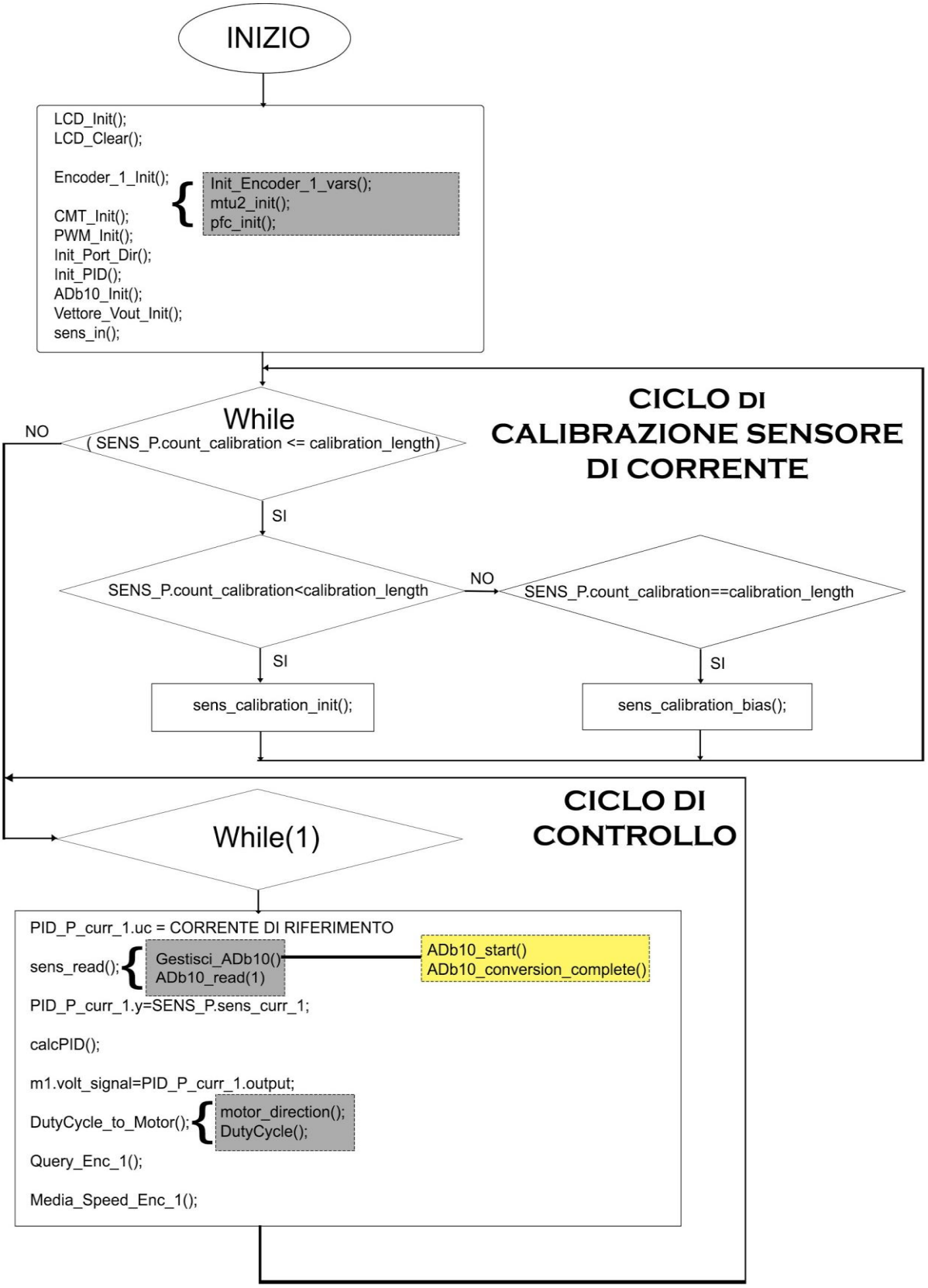


G.2. Cablaggio con sensore MAX471 GY-471 (Cablaggio finale sulla bici)



SOFTWARE

Di seguito è riportato lo schema del programma principale e le sue varie funzioni.



La prima implementazione del programma è stata effettuata insieme al TASK 2 (controllo dello sterzo) poiché si voleva un codice già adatto all'utilizzo di due motori con due funzioni differenti. Per tale motivo, oltre a contenere diverse possibilità di uso per diversi pin da cui controllare il motore, nella libreria PWM.h è stata implementata una struct "motor" che ci permette di dividere i dati dei segnali da inviare al motore di trazione o al motore dello sterzo (infatti tali motori sono di tipo e potenza differente).
Il programma è stato implementato prendendo spunto da un demo per il controllo in coppia dei motori del Ballbot evitando l'uso delle variabili globali extern e creando strutture apposite per passare tramite puntatore gli argomenti alle funzioni.
Di seguito sono descritte in breve le funzioni utilizzate di ogni libreria:

1) CMT

Grazie alla libreria CMT abbiamo potuto contare il tempo del ciclo di controllo che risulta essere di circa 1 millisecond0.

- CMT_init(): inizializzazione del CMT;
- get_ms(): funzione di tipo int32_t che restituisce il valore di una variabile contatore millis.
- CMT_isr(): interrupt usato per l'aumento del contatore millis; esso avviene ogni millisecondo.

2) ADb10

- *ADb10_init():* inizializzazione dell'ADb10;
- *Adb10_start()*: inizia la conversione dell'ADb10;
- *filtraggio*: filtro FIR, un primo **if** distingue la fase di riempimento del vettore *vout* da quella di "scorrimento" della finestra mobile grazie al confronto tra variabile *contatore* e *length.*

  Quando *contatore* è minore di *length*, tramite *read* si legge il valore contenuto nel registro e poi si esegue la somma pesata contenuta nel **for** e quindi verrà assegnata alla variabile *vout* il risultato della media calcolata ponderata.

  Nel secondo caso si effettua lo shift del valore nella posizione i-esima con quello nella posizione precedente (i-1) in modo da inserire il valore più recente nella posizione length-1 e si scarta il meno recente.

  Osservazione : *lenght* nella libreria ADb10.h è settato a 1000; un valore inferiore non renderebbe efficace il filtraggio. Tale risultato è stato ottenuto tramite una simulazione Matlab effettuata dagli studenti che si sono occupati del controllo in coppia per il Ballbot, per maggiore dettagli si veda pag.22 di [6];
- *ADb10_read():* lettura dei valori convertiti;
- *ADb10_conversion_complete():* funzione che avvisa del completamento della conversione;
- *Gestisci_ADb10():* funzione che gestisce l'uso dell'ADb10 e termina al completamento della conversione;
- *vettore_vout_init():* inizializzazione vettore vuoto dove verranno inseriti i valori convertiti dall'ADb10;

3) PWM

- *PWM_Init() :* inizializza i canali per il segnale PWM attraverso i registri MTU2.
- *Init_Port_Dir():* Inizializzazione dei PIN per l'uscita del PWM.
- *DutyCycle():* Settaggio valori registri per il duty cycle del PWM
- *motor_direction():* Settaggio della tensione da inviare al motore per cambiare la direzione. Essa è calcolata dal segnale dato in argomento, se esso è positivo la direzione sarà in senso orario, se invece il segnale è di segno negativo, il senso sarà antiorario;
  NOTA: Per il nostro obiettivo non si necessita di un cambio di direzione, ma poiché, come già detto, alcune funzioni del codice sono condivise con il TASK 2 abbiamo deciso di mantenere anche nel nostro programma dimostrativo le funzioni del cambio di direzione.
- *change_motor_dir():*Settaggio dei registri per la direzione del motore
- DutyCycle_to_Motor(): calcola il valore del duty-cycle per generare il PWM da inviare al motore.

4) SENSORE DI CORRENTE

- *sens_calibration_init():* Inizializza sia il sensore che le variabili per la calibrazione e inizia la calibrazione stessa con 1000 valori di riferimento.
- *sens_calibration_bias():* calcolo dei valori ottenuti attraverso fattori di correzione del sensore (appunto il bias).
- *sens_read():* lettura della corrente assorbita dal motore in Ampere.

5) PID

- *init_pid():* Inizializza il PID.
- *init calcPID():* calcolo per la correzione del segnale per il controllo.

6) LETTURA ENCODER

Importante, nell'utilizzo dell'encoder, è la dichiarazione nel suo header file di una struttura che abbiamo chiamato *encoder_data* che contiene le variabili necessarie per la lettura.

- *encoder_init()*: inizializzazione dei parametri dell'encoder
- *Init_Encoder_vars*: inizializzazione delle variabili utilizzate dall'encoder
- *mtu2_init()*: inizializzazione della periferica MTU2 sulla scheda Renesas per l'utilizzo dell'encoder
- *pin_enc_init():* inizializzazione dei pin sulla scheda Renesas per l'utilizzo degli encoder.

- *Query_enc()*: calcolo delle variabili necessarie per la lettura degli encoder mediante l'utilizzo degli interrupt opportunamente generati che permettono di ottenere una lettura della velocità istantanea della ruota.

6.1) Registri per la priorità degli interrupt.

All'interno della libreria per l'uso dell'encoder sono stati utilizzati molti registri necessari al controllo delle priorità degli interrupt. Qui di seguito sono stati elencati i registri e in breve le loro funzioni e la pagina del manuale hardware della Renesas in cui poter approfondire il loro uso.

- **Interrupt Request Register (IR) pag 353 :** da settare per generare una richiesta di interrupt
- **Interrupt Request Enable Register (IEN) pag 354 :** settati affinchè la richiesta di interrupt sia output alla destinazione selezionata dalla richiesta
- **Interrupt Source Priority Register (IPR) pag 355:** i loro bit specificano il livello di priorità della sorgente di interrupt corrispondente.
- **DTC Activation Enable Register (DTCE) pag 357:** attivano il DTC

  DTC e DMAC sono periferiche del microcontrollore. Entrambi servono allo scopo di trasferire dati da e verso la memoria e le periferiche, indipendentemente dal controllo della CPU.
  -chiarimento in [11]

- **Group m Interrupt Source Register (IS) pag 370:** contengono bit che sono flag di stato della richiesta di interrupt del modulo periferico assegnata al bit j-esimo nel gruppo m.
- **Group m Interrupt Enable Register (EN) pag 373:** settati per abilitare le richieste di interrupt al gruppo m corrispondente.
- **Group m Interrupt Clear Register (CLR) pag 375:** scrivendo 1 su questi bit si cancella il bit della flag di stato dell'interrupt (GRPm.ISj)
- **Unit Selecting Register (CN) pag 376:** permettono di selezionare MTU o TPU.

TEST IN LABORATORIO

Si sono svolti diversi test, ognuno di essi per verificare il corretto funzionamento di ogni componente in risposta al codice.
I primi test sono stati effettuati con la collaborazione del gruppo di lavoro al TASK 2 (controllo dello sterzo) poiché si desiderava preparare già un codice pronto ad ospitare l'utilizzo di due motori differenti.
Tali test sono stati effettuati sul motore RB-POL-126 utilizzato dal gruppo del TASK 2 per i loro obiettivi. Sono stati quindi ri-effettuati sul nostro motore destinato alla trazione.

01. La prima prova è stata eseguita per verificare il corretto funzionamento del motore controllato semplicemente con una tensione continua erogata da un alimentatore. Si è osservata la corretta variazione di velocità per ogni valore di tensione tra 0 V e 12 V.
02. Successivamente si è introdotta la scheda Renesas e il driver testando il cambio di direzione del motore con i tasti A e B, aggiunto poi il sensore di corrente è stato testato il codice con una lettura dei valori tramite LCD.
    Grazie al programma è stato possibile leggere :
    - La corrente assorbita (in Ampere)
    - Il valore di riferimento dato al PID (in Ampere)
    - Il segnale di uscita calcolato dal PID per il PWM (in volt)

    Come segnale di riferimento è stato passato sia un vettore con valori di corrente di vario segno che un unico valore costante. **Per il nostro obiettivo ci interessa un valore di riferimento costante della corrente**, ma grazie ai valori di diverso segno nel vettore è stato rilevato un problema nel codice per il cambio di direzione e correggerlo: vedi in **Appendice: Problematiche riscontrate e soluzioni** il problema (a).
    Con l'uso dell'oscilloscopio si sono osservati i segnali in ingresso e in uscita dei pin della Renesas per verificare il corretto funzionamento del tutto.

I test precedenti, come premesso, sono stati ri-effettuati sul nostro motore destinato alla trazione:

04. È stata quindi necessaria una taratura del PID, come descritto in precedenza, ed è stato nuovamente testato il funzionamento del motore.
05. Il test seguente è stato effettuato solo sulla lettura dei dati dall'encoder. Utilizzando solo la libreria e le funzioni per la lettura dell'encoder, si è potuto così osservare che il motore senza carico può raggiungere una velocità angolare reale di circa 16 rad/s.
06. Nel test successivo, unendo quindi il controllo in corrente con la lettura dell'encoder, si è riscontrato che per valori superiori a 9.9 V per il segnale del PWM gli interrupt dell'encoder smettevano di azionarsi portando una lettura fissa a 0 rad/s (impossibile con il motore evidentemente in movimento): si veda problema e soluzione (b).

Tabella corrispondenze corrente velocità- sensore ACS712 – motore senza carico.

Valore di corrente in riferimento	Segnale in tensione ottenuto dal controllo del PID inviato con modulazione PWM	Velocità angolare del motore ottenuta.
0 A	0.097 V	0 rad/s
0.5 A	2.3 V	2.8 rad/s
1 A	4.89 V	6.1 rad/s
1.5 A	7.55 V	9.8 rad/s
2 A	8.32 V	11.2 rad/s
2.5 A	9.89 V	13 rad/s
3 A	12 V	15.5 rad/s

07. Questo test è stato eseguito sul motore montato sulla corona della bicicletta. Sono state effettuate diverse prove di controllo con diversi valori di corrente in riferimento costruendoci una tabella dei valori ottenuti come sopra: si osserva una diminuzione di circa 2 rad/s data dai componenti meccanici della bicicletta.

Tabella corrispondenze corrente velocità- sensore ACS712

Valore di corrente in riferimento	Segnale in tensione ottenuto dal controllo del PID inviato con modulazione PWM	Velocità angolare del motore ottenuta.
0 A	0.097 V	0 rad/s
0.5 A	2.5 V	2.5rad/s
1 A	5.2 V	5.2 rad/s
1.5 A	7.6 V	8.8 rad/s
2 A	10.2 V	12 rad/s
2,5 A	11.4V	13.5 rad/s
3 A	12 V	14.9 rad/s

08. In questo test sono stati cambiati i sensori ACS712 con i MAX471 GY-471 e son stati cambiati i relativi parametri nel programma nelle librerie *sensore.h* e *ADb10.h*.
09. Si sono uniti i programmi con il programma del TASK 1 e del TASK 2.

10. Sono statiti effettuati alcuni test per la verifica delle calibrazioni e delle letture dei sensori riscontrando il problema ©.
11. Infine si è tarato un nuovo PID per il nostro motore e sono stati effettuati i test finali.

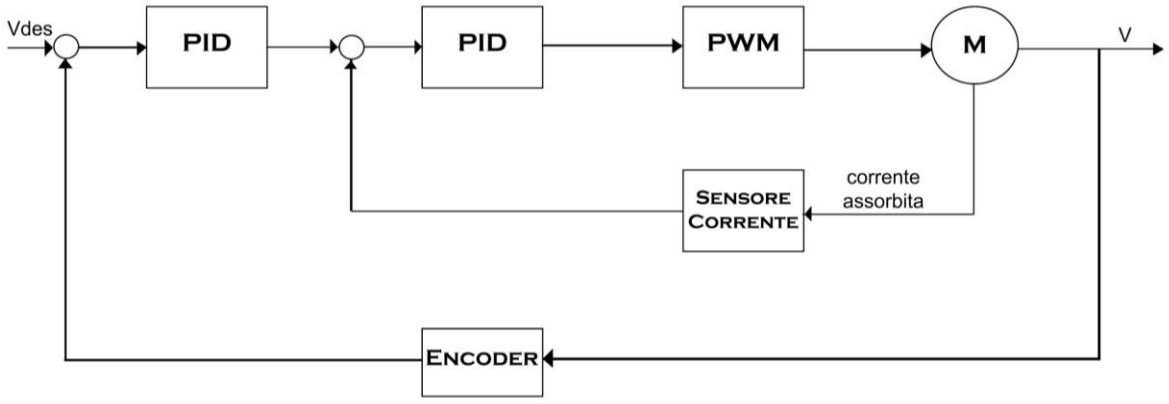
Tabella corrispondenze corrente velocità- sensore MAX471 GY-471

Valore di corrente in riferimento	**Segnale in tensione ottenuto dal controllo del PID inviato con modulazione PWM**	**Velocità angolare del motore ottenuta.**
Tra 0A e 0.5 A il MAX471 GY-471 risente di vari errori (si veda schematico) che rendono i dati non coerenti. La lettura inizia ad essere precisa circa a 0.5 A con i risultati ottenuti sperimentalmente di seguito.		
0.5 A	2.5 V	2.5rad/s
1 A	5.4 V	5.2 rad/s
1.5 A	6 V	6.9 rad/s
2 A	9 V	12 rad/s
2,5 A	11 V	13. rad/s
3 A	12 V	14 rad/s

CONCLUSIONI

Per la velocità desiderata di *1.8 m/s* abbiamo quindi bisogno sulla corona di una velocità angolare di *6,254 rad/s.* Dalla tabella ottenuta nel test finale osserviamo che per un valore in riferimento di 1.4 A dovremmo ottenere la velocità desiderata.

Per un migliore controllo automatico, si ipotizza l'uso di un controllo in velocità esterno a quello in corrente e quindi, con l'utilizzo di due PID come mostrato in figura, implementabile anch'esso conoscendo i parametri del motore specificati nel capitolo del PID.



APPENDICE : Problematiche riscontrate e soluzioni.

Lista dei problemi riscontrati:

a. **Problema:** il software non permette il cambio della direzione.
   Soluzione: la variabile "stato" che era stata dichiarata dentro la funzione motor_direction() non cambiava valore, pertanto è stata quindi dichiarata all'inizio della libreria.

b. **Problema:** il programma del controllo in corrente prevedeva l'utilizzo dell'LCD per leggere e confrontare i valori di corrente assorbita, i valori in uscita dal PWM e il valore di riferimento. Tali valori erano settati come lettura di un float completo (fino a fine riga). Dal momento che la lettura richiedeva per l'uscita del PID un numero superiore a 9.9 si generava un overflow nei registri dell'LCD andando in contrasto con i registri utilizzati per l'interrupt dell'encoder.
   Soluzione: per correggere l'errore bastava chiedere all'LCD una lettura dei valori fino alla terza cifra dopo la virgola. In tal modo il programma e la lettura funzionano correttamente.

c. **Problema:** all'unione dei programmi l'ADb10 convertiva solo il primo valore di campionamento ricevuto.
   Soluzione: in un **if – else** della funzione *GestisciADb10()* la variabile "stato" compariva sempre settata ad 1(conversione ultimata), non potendo così ricominciare una nuova conversione. Settando nell'**else** stato = 0 si permette al convertitore di continuare a convertire.

RIFERIMENTI

[1] “Hardware_manual_RX63N”
https://www.renesas.com/eu/en/doc/products/mpumcu/doc/rx_family/r01uh0041ej0180_rx63n631.pdf

[2] “User_manual_YRDKRX63N”
https://www.renesas.com/eu/en/doc/products/tools/r20ut2532eu0100_yrdkrx63n_um.pdf

[3]”Schematics_YRDKRX63N”
https://github.com/mattmunee/RenesasYRDKRX63N/blob/master/Schematics_YRDKRX63N_SCH_3.pdf

[4]”Driver_MD10C”
https://www.robotshop.com/media/files/PDF/user-manual-md10c-v2.pdf

[5]Relazione progetto per Corso Laboratorio di Automazione ”SELF BALANCING BICYLCE”, Bonci & Camerlengo,Di Girolamo,Simoni,Tesei; UNIVPM, 2015-2016
https://drive.google.com/file/d/1-CijEeeapKsV44OU55fokKwo_R_xCvo6/view?usp=sharing

[6]Relazione progetto per Corso Laboratorio di Automazione ”Controllo in coppia Motori Ballbot”, Bonci & Bottoni,Buccolini,Elisei,Luzi; UNIVPM

[7]Relazione progetto per Corso Laboratorio di Automazione ”Controllo in tensione della velocità dei motori”, Bonci & Brutti,Ciancaglione,Di Muzio,Iezzi; UNIVPM, 2017-2018

[8]Schematics_Motore12V DC RB-Dfr-439 146 rpm con Encoder
https://image.dfrobot.com/image/data/FIT0277/mechanical.jpg

[9]”How to read a dc motors datasheet”
https://medium.com/luosrobotics/how-to-read-a-dc-motors-datasheet-f70fa440452b

[10]SIMULATION OF ELECTRIC MACHINE AND DRIVE SYSTEMS USING MATLAB AND SIMULINK by Mahmoud Riaz, Sc.D. Professor of Electrical Engineering, Department of Electrical and Computer Engineering, University of Minnesota.→
http://people.ece.umn.edu/users/riaz/

[11] “Renesas:**KNOWLEDGE BASE** - What is the difference between a DTC and DMAC?”
https://en-support.renesas.com/knowledgeBase/16978598

[12] “MAXIM: precision, high-side, Current-Sense Amplifiers”
https://pdfserv.maximintegrated.com/en/ds/MAX471-MAX472.pdf