



**UNIVERSITÀ DEGLI STUDI DI ROMA
TOR VERGATA**

FACOLTÀ DI INGEGNERIA

CORSO DI LAUREA IN INGEGNERIA DELL'AUTOMAZIONE

A.A. 2020/2021

Tesi di Laurea

Sviluppo di algoritmi di controllo delle correnti nelle bobine poloidali di macchine per la fusione Tokamak, con riguardo al design sistemico per la cooperazione tra sistemi embedded per l'attuazione, misurazione e centrali di controllo.

RELATORE

Daniele Carnevale

CANDIDATO

Emanuele Alfano

CORRELATORI

Marco Passeri

Dedico questa tesi ai miei cari nonni.

Grazie per aver sempre creduto in me.

Indice

Ringraziamenti	1
1 Introduzione	2
1.1 Fusione Termonucleare	2
1.2 Struttura di un Tokamak	2
1.3 Obiettivi della Tesi	2
1.4 Struttura della Tesi	2
2 Elementi Costitutivi	3
2.1 Trasformatore - Modello di Tokamak	4
2.1.1 Modellazione Fisica	5
2.1.2 Dal circuito alla dinamica	6
2.1.3 Funzione di Trasferimento	6
2.2 Trasduttore di Corrente	7
2.2.1 Sensore scelto	7
2.2.2 Criticità	8
2.2.3 Funzionamento Interno	8
2.2.4 Connessione elettrica	11
2.2.5 Misura	11
2.3 Driver di Corrente - IBT-2	13

2.3.1	Schema Elettrico	14
2.3.2	Connessione di Controllo	14
2.3.3	Benchmark Driver	16
3	Architettura di Sistema	18
3.1	Architettura ad alto livello	18
3.1	EMP - Libreria di Comunicazione Seriale	20
3.1	Protocollo - COBS	21
3.1.1	Metodo di codifica	21
3.1.2	Struttura del codice	21
3.1.3	Benchmark	21
3.2	Online Sampling	21
3.2.1	Interconnessione Arduino \Leftrightarrow Companion	21
3.2.2	Storage su file delle informazioni	21
3.3	Post Elaborazione con Matlab	21
3.3.1	Conversioni Dati	21
3.3.2	Creazione dei grafici e Filtraggio	21
4	Modello teorico di Controllo	23
4.1	Controllo a Errore Nullo	23
4.2	Simulazione Qualitativa su Simulink	23
5	Sviluppo Controllo reale	24
5.1	Codifica del controllore	24
5.2	Tuning delle costanti	24
5.3	Esperimenti	24

6 Conclusioni e sviluppi futuri	25
Appendice A - Codice Arduino	26
6.1 Set-up Registri	26
6.2 Generatore di Segnale	28
6.2.1 Segnali Base	28
6.2.2 Segnali Composti	29
6.3 Codici Controllore	30
Appendice B - Codice EMP	31
Appendice C - Matlab Post Elaboration	32
Elenco delle figure	33
Bibliografia	34
7 Software, Toolchain, Strumenti	35

Ringraziamenti

Corpo dei ringraziamenti

Capitolo 1

Introduzione

1.1 Fusione Termonucleare

1.2 Struttura di un Tokamak

1.3 Obiettivi della Tesi

1.4 Struttura della Tesi

Capitolo 2

Elementi Costitutivi

In questo capitolo si vogliono descrivere e caratterizzare i 3 elementi salienti dell'esperimento:

1. *Trasformatore - Modello di Tokamak*
2. *Trasduttore di Corrente*
3. *Driver di Corrente - IBT-2*

Verranno analizzate le loro caratteristiche chiave per mettere in luce il perché della scelta, e si evidenzieranno eventuali problemi che affliggono in componenti, problemi di cui si è tenuto conto nello sviluppo del progetto per poterli annullare.

2.1 Trasformatore - Modello di Tokamak

Come visto nell'introduzione, la tesi ha come obiettivo la prototipazione del sistema di controllo per le bobine poloidali presenti in impianti tokamak.

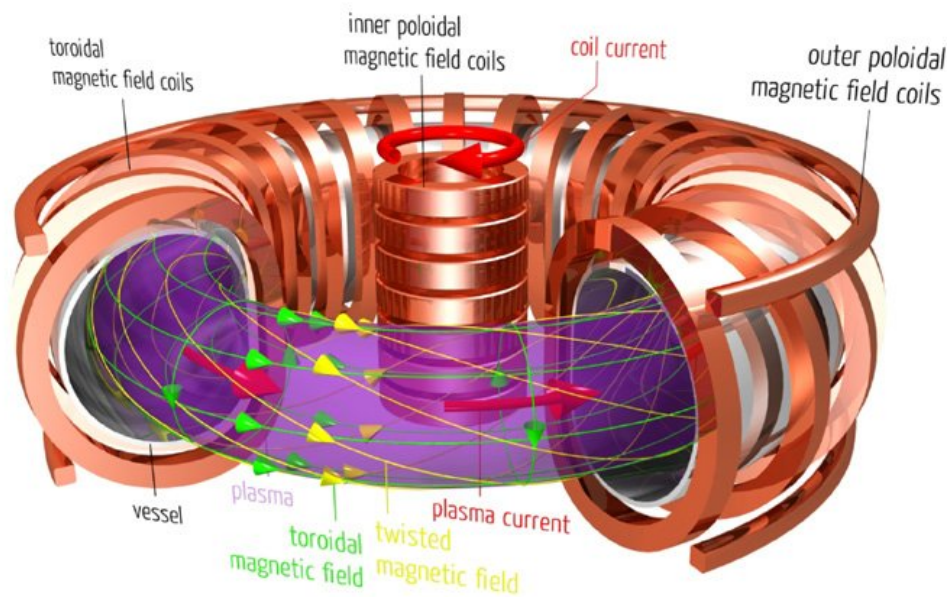


Figura 2.1: Interno Tokamak

Le bobine poloidali servono a controllare il plasma presente nel *Vessel* dell'impianto e confinare il Plasma all'interno della camera, cercando al contempo di comprimere il Plasma per realizzare eventi di **Fusione Nucleare**.

L'interazione tra le **Bobine e Plasma**, ha un modello matematico non dissimile da quello di **Trasformatore Elettrico** nella relazione Primario-Secondario (con i dovuti paragoni e le ovvie *Non-Linearità* presenti nel caso di un impianto reale).

Grazie a questa similitudine è stato possibile replicare in sicurezza la fisica presente all'interno di un Tokamak, nell'ambiente controllato del laboratorio.

2.1.1 Modellazione Fisica

Come descritto nell'articolo di Romero et al., «[Real time current profile control at JET](#)», è possibile modellare la dinamica tra il Plasma e la Bobina come un circuito la relazione tra un Primario e un Secondario di un Trasformatore.

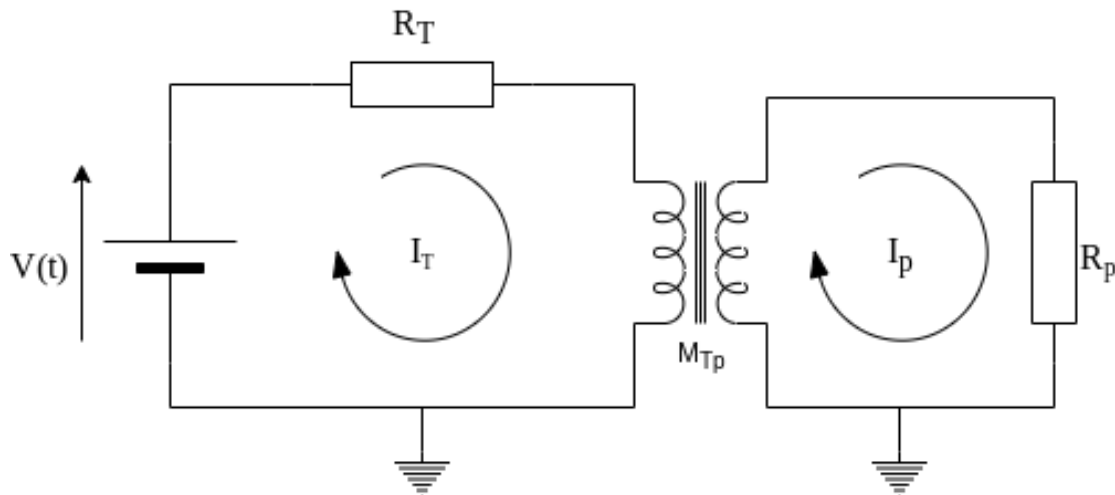


Figura 2.2: Circuito Equivalente Bobina/Plasma

Sperimentalmente si è in oltre visto che è possibile modellare questa relazione tra i 2 circuiti trascurando le forze indotte dal plasma dentro la bobina di controllo. Usando queste osservazioni si ottiene quindi il circuito equivalente:

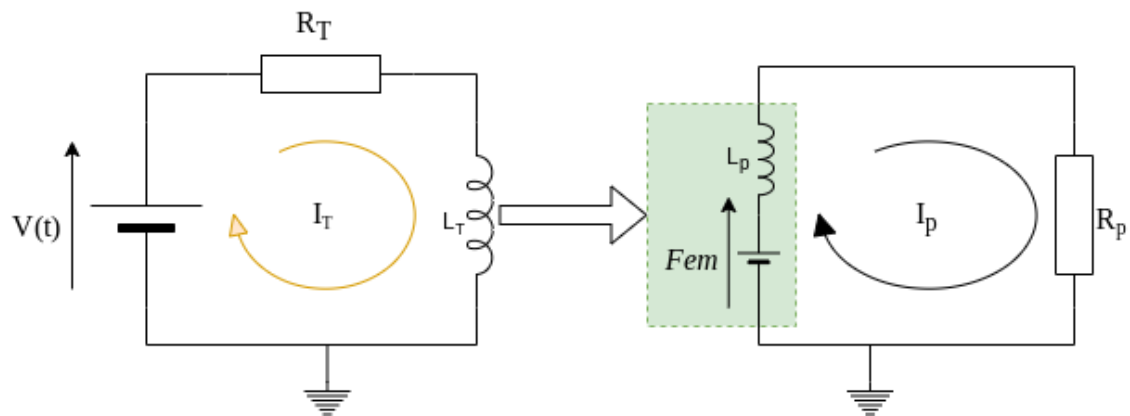


Figura 2.3: Circuito Equivalente Bobina/Plasma semplificato

2.1.2 Dal circuito alla dinamica

2.1.3 Funzione di Trasferimento

2.2 Trasduttore di Corrente

Come detto nell'introduzione, l'obiettivo della tesi è di controllare la corrente sul secondario usando il campo elettrico del secondario, la misura della corrente scorrente sul primario in tale ottica non sarebbe una misura di interesse. Essendo però un lavoro di ricerca, si è preferito poter misurare la corrente effettivamente circolante nel Primario del trasformatore, così da poter meglio interpretare i dati di misura senza ambiguità.

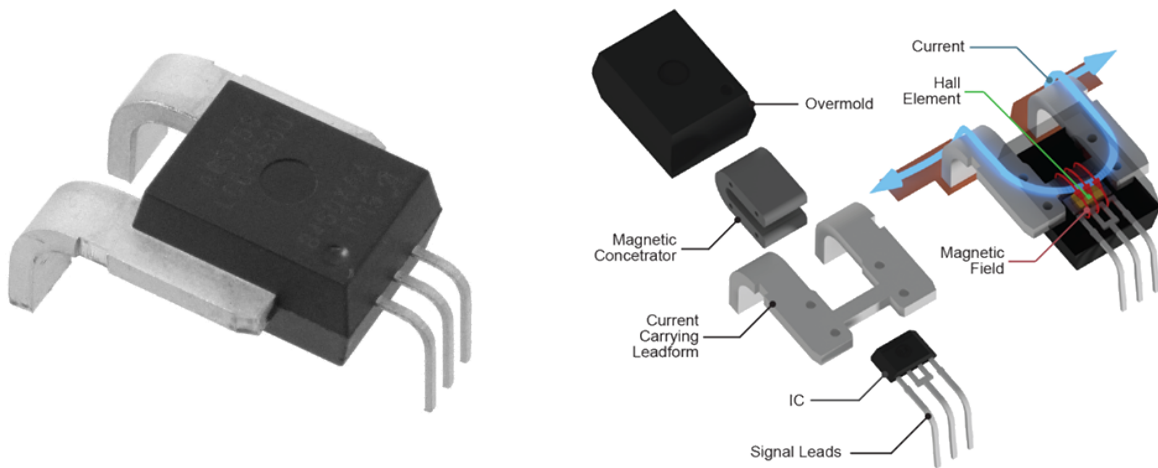


Figura 2.4: Sensore di Corrente

2.2.1 Sensore scelto

Per misurare la corrente del primario, si è posizionato in serie allo stesso il sensore di Corrente Allegro, *High-Precision Linear Current Sensor*.

La famiglia di sensori ha in generale le seguenti caratteristiche:

Bandwidth:	120 kHz
Output rise time :	4.1 μs
Ultralow power loss:	100 $\mu\Omega$ Resistenza Interna
Single supply operation	4.5 to 5.5 V
Extremely stable output offset voltage	

In oltre, delle tante varianti presenti, si è scelto di usare la ACS770LCB-100B-PFF-T, le cui caratteristiche chiave sono:

Primary Sampled Current:	± 100 A
Sensitivity Sens (Typ.)	20(mV/A)
Current Directionality	Bidirectional
T_{OP}	-40 to 150 ($^{\circ}$ C)

L'ampio margine di misura, e la robustezza alle variazioni di temperatura rendono il dispositivo perfetto per misurare i nostri esperimenti.

L'ampio margine di misura permette di comprendere tutti i possibili valori di corrente ottenibili in laboratorio, rendendo il prototipo adatto a scopi futuri.

2.2.2 Criticità

Unico punto dolente è il suo principio di funzionamento: essendo il sensore basato su un l'effetto Hall, ovvero una misura diretta del campo magnetico indotto dalla corrente nel conduttore, è importante tenere distante il sensore dal Trasformatore Centrale che nei suoi momenti di massimo flusso di corrente, genera ovviamente un campo magnetico non indifferente.

2.2.3 Funzionamento Interno

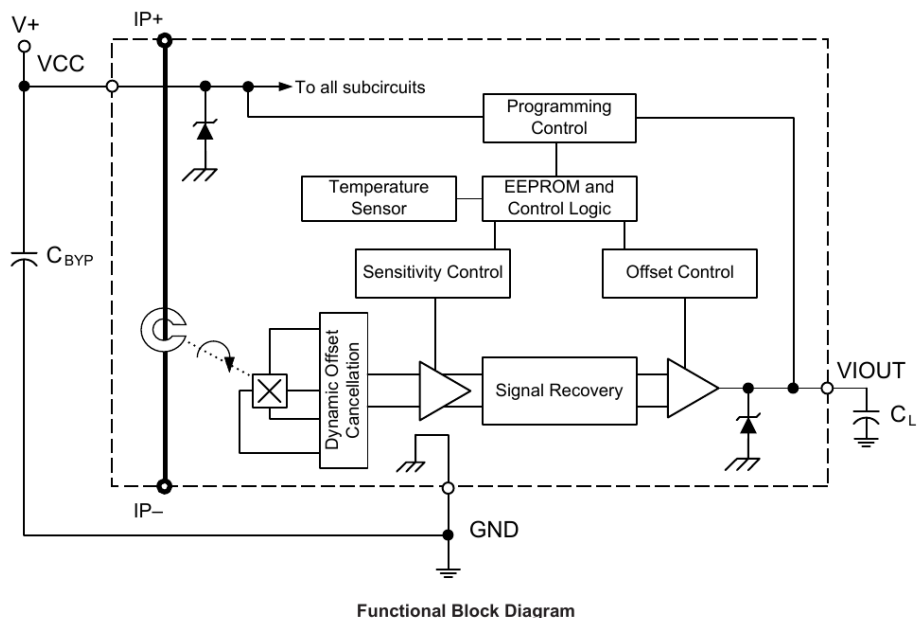


Figura 2.5: Schema a Blocchi

Tra le caratteristiche chiave dell'Allegro, *High-Precision Linear Current Sensor*, troviamo il disaccoppiamento fisico tra la corrente da misurare e il circuito di misura.

Questa caratteristica chiave, garantisce la salvaguardia del circuito logico a valle, dai possibili eventi catastrofici a monte.

Esso è in oltre fornito di sensori di temperatura e sistemi di *Signal Recovery* che permettono all'Hardware stesso di compensare parzialmente *Non-Linearity* termiche e nella misura dell'effetto Hall, ottenendo un output assimilabile a un segnale lineare:

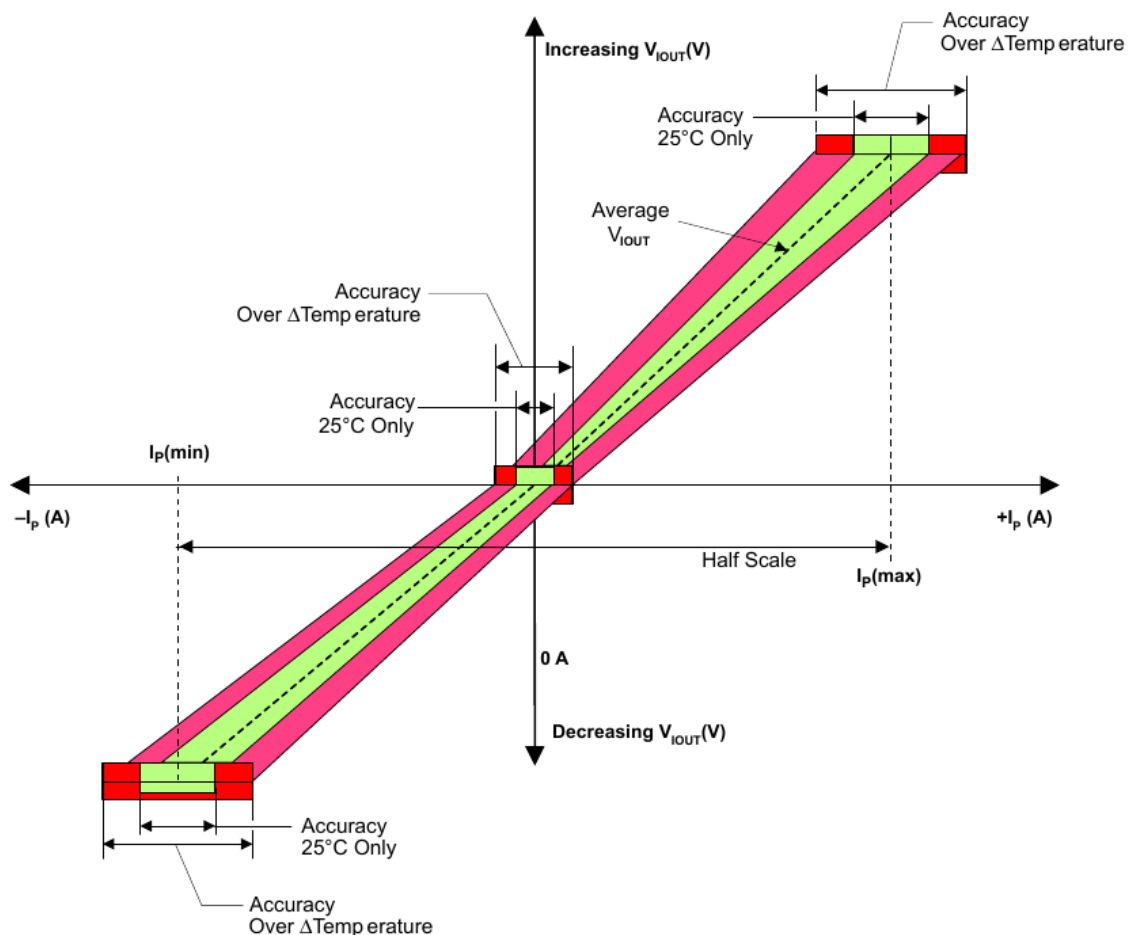


Figura 2.6: Sensibilità

Come si può vedere dal grafico, gli errori sono tanto più marcati quanto maggiore è la corrente da

misurare, ma leggendo dal datasheet abbiamo che questo errore, che dipende sì fortemente dalle temperature di esercizio dell'esperimento, non è mai, neanche negli esperimenti più sfortunati, superiore al $\pm 2\%$.

Anzi, alle temperature $\approx 25^\circ$, si mantiene contenuto tra $\pm 0.5\%$.

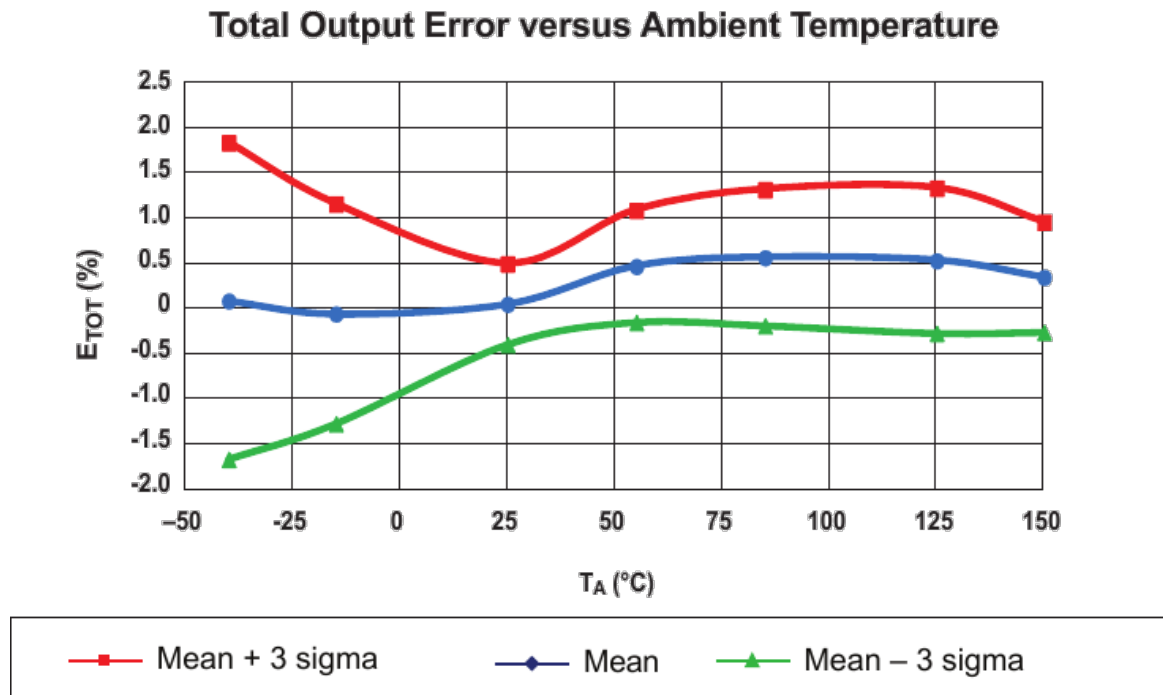


Figura 2.7: Temperatura/NonLinearità

2.2.4 Connessione elettrica

La connessione del sensore è particolarmente semplice, richiedendo esternamente solo un'alimentazione stabilizzata e portando subito in uscita la misura.

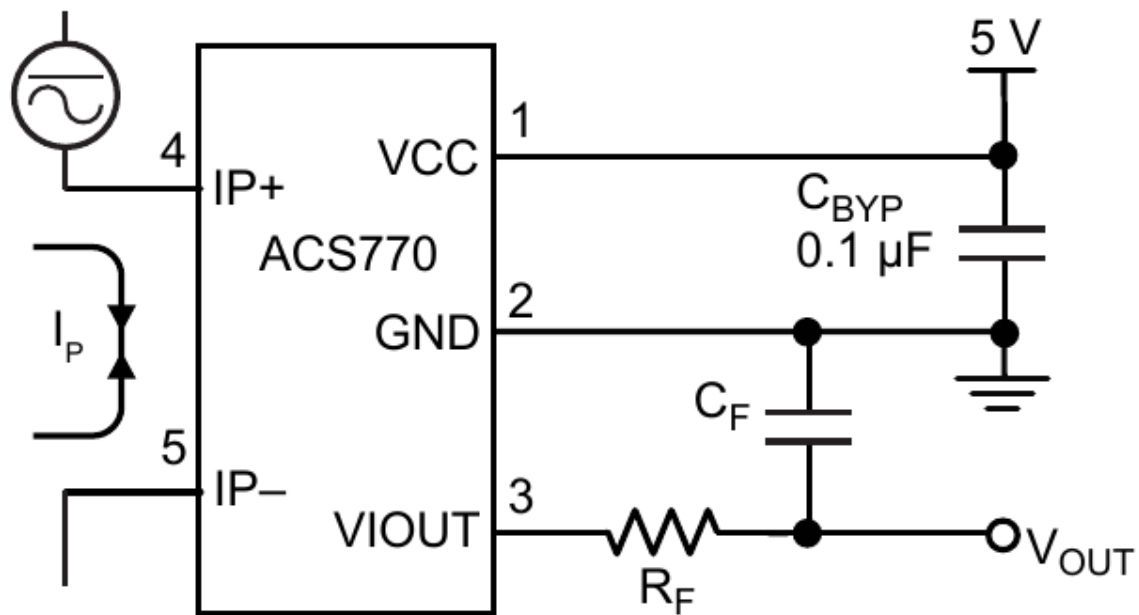


Figura 2.8: Collegamento dal Datasheet

Rispetto allo schema proposto dal datasheet, però, si è anche deciso di omettere il filtro passa-basso sulla **VIOUT**, questa scelta è stata presa per minimizzare il più possibile ritardi di misura della corrente istantanea, poiché le dinamiche del sistema sul secondario, come visto, sono di tipo derivativo, e quindi estremamente rapide.

2.2.5 Misura

Il trasduttore, misura della corrente sotto forma di tensione, la quale varia in base alla **Sensibilità** del modello in uso. Avendo noi il ACS770LCB-100B-PFF-T, il datasheet riporta:

Primary Sampled Current:	$\pm 100 \text{ A}$
Sensitivity Sens (Typ.)	$20(\text{mV/A})$
Current Directionality	Bidirectional

Ciò implica che la corrente misurata, è calcolabile come:

$$I_{read} = \frac{V_{Read}[\text{V}]}{V_{sense}[\text{V/A}]}$$

Rimozione Offset Essendo però il device ad alimentazione singola (0–5V), ma la corrente misurabile Bi-direzionale, sorge la necessità di spostare gli 0A a una tensione superiore agli 0V.

Il datasheet riporta che $V_{offset} = \frac{V_{cc}}{2} \approx 2.5\text{V}$. Da cui deriva che la vera misura di corrente è:

$$I_{read} = \frac{V_{Read} - V_{offset}}{V_{sense}} \frac{V}{[\text{V/A}]}$$

Al fine di poter misurare l'offset effettivo, durante il set-up viene eseguito a esperimento fermo una misura dell'offset attuale, usando la [Macro per calcolo Offset](#).

Il risultato della computazione, oltre ad essere usato nel controllo è inviato al computer per la post elaborazione dei dati nei grafici.

Analisi Sensibilità Usando nell'esperimento un ADC a 10Bit con tensione di riferimento a 5V, abbiamo che la massima sensibilità del μC , ovvero il suo bit meno significativo è pari a:

$$V_{step} = \frac{V_{cc}}{2^{10} - 1} = 4,887\text{mV}$$

Il che equivale a una **Sensibilità di Corrente** del $\mu\text{Controllore}$ pari a:

$$I_{step} = \frac{V_{step}}{V_{sense}} = 244,379\text{mA}$$

Il che rende la misura buona per osservare cosa stia accadendo, ma sicuramente non sufficientemente densa da poterla usare come parametro ingresso di controllo.

2.3 Driver di Corrente - IBT-2

Per l'attuazione del controllo di corrente nella bobina primaria del trasformatore, è stato usato il driver di corrente Handsontec, *BTS7960 High Current 43A H-Bridge Motor Driver*.

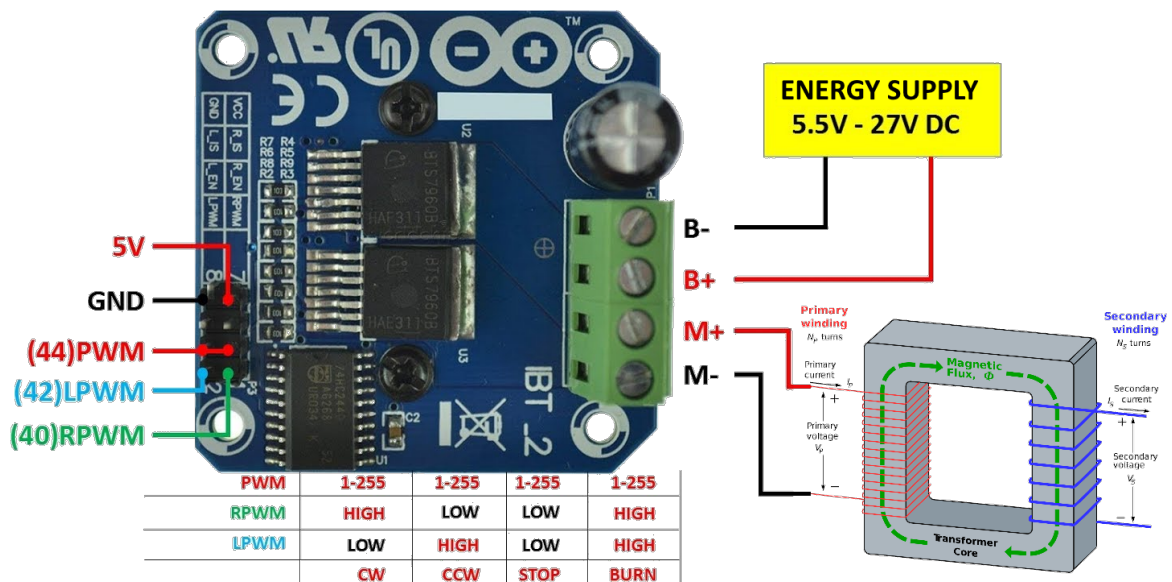


Figura 2.9: IBT-2 TopView.

Esso non è un comune Ponte-H integrato: per poter gestire potenze superiori è stato costruito usando 2 Half-Bridge (Infineon, *BTS7960B High Current PN Half Bridge NovalithIC T*) collegati assieme mediante una opportuna logica per ricreare un normale Ponte-H.

Questa scheda in particolare ha prestazioni interessanti per gli scopi di questa tesi, i principali sono elencati di seguito:

Power Input Voltage:	6 – 27 V
Peak current:	43 A
Massima Frequenza di PWM:	25 kHz
Protezione Sovra Tensioni	
Disaccoppiamento Ingresso di Potenza/Logica di controllo	

Di particolare interesse per l'esperimento è proprio la corrente di picco gestibile: avendo le dinamiche del sistema tempi inferiori ai 5 secondi, poter reggere correnti di picco così elevate rende la scheda perfetta per i nostri scopi.

2.3.1 Schema Elettrico

Al suo interno il driver è composto da 2 Half-Bridge Infineon, *BTS7960B High Current PN Half Bridge NovalithIC T*, connesse secondo lo schema:

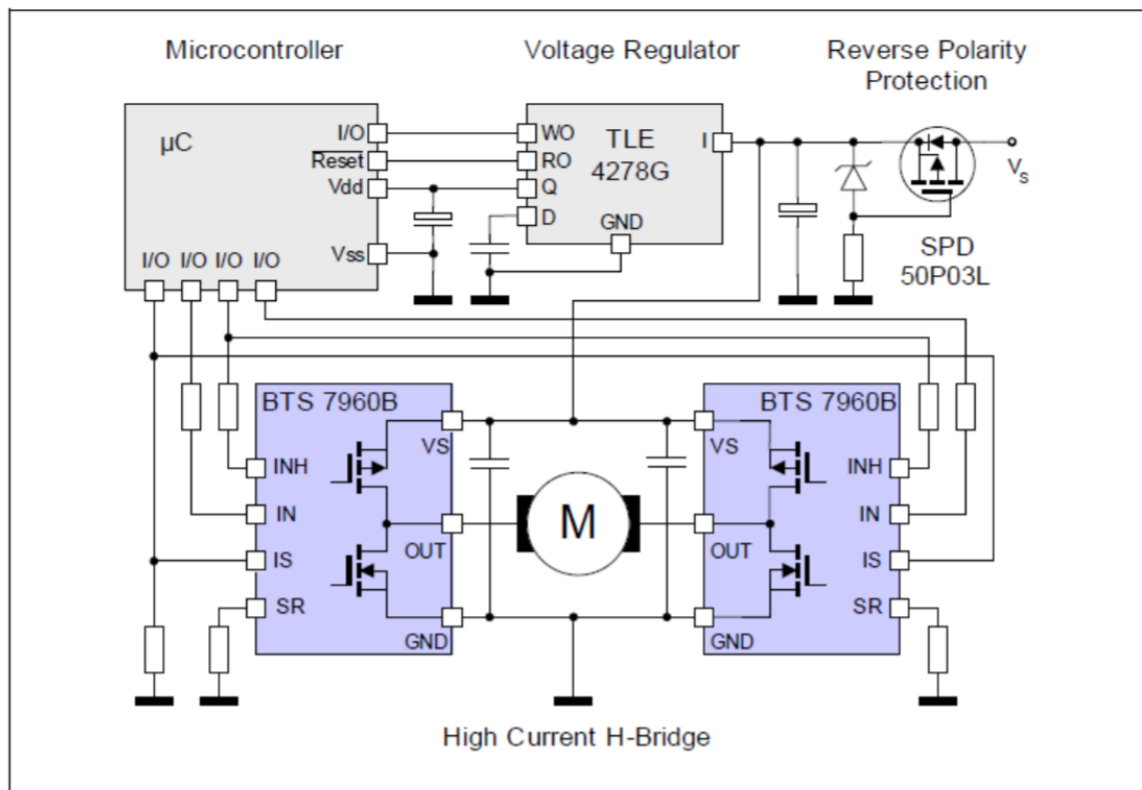


Figura 2.10: IBT-2 Schema Elettrico

Il μC è protetto dal carico connesso all'interno del BTS7960b, lasciando al μC solo il compito di controllare i segnali.

2.3.2 Connessione di Controllo

Il driver permette 2 modalità di funzionamento:

Doppio PWM Modalità operativa che richiede l'uso di 2 PWM

Ciascun PWM controlla uno dei 2 Half-Bridge, e per evitare di bruciare i driver devono essere controllati singolarmente, il vantaggio di questa configurazione è la possibilità di usare 2 frequenze di controllo diverse.

Singolo PWM Modalità operativa classica di un normale Ponte-H

In questa modalità, la porta nand presente sulla scheda attua la logica di controllo opportuna per governare i 2 Half-Bridge come fossero un normale Ponte-H.

Per il nostro esperimento si è scelto di usare il collegamento **Singolo PWM** così da evitare spiacevoli sorprese e avere il PWM di controllo sempre sincronizzato.

2.3.3 Benchmark Driver

Il driver sulla carta à buone prestazioni, ma non sono descritte le sue *Non-Linearità*, per farle risaltare si sono effettuati 2 esperimenti usando differenti input di controllo:

1. Onda Triangolare Periodica

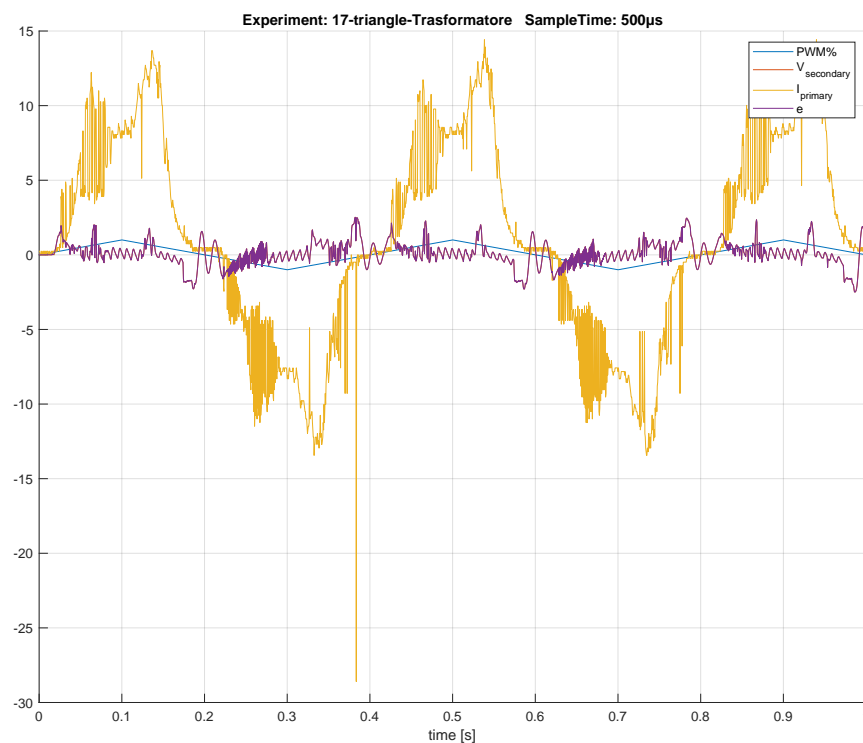


Figura 2.11: Onda Triangolare

Dead-Zone Inferiore L'onda triangolare si presta bene per far risaltare la problematica della Dead-Zone Inferiore, infatti in tutti gli intorno in cui il segnale passa per 0, è possibile vedere come la corrente non vari minimamente, è però possibile notare che i 2 lati non sono simmetrici tra di loro, questo è facilmente spiegabile dal fatto che il primo ha una condizione iniziale $\neq 0$ e di fatto stiamo ancora osservando l'esaurimento del transitorio, la soglia di Dead-Zone Inferiore è quindi calcolata vedendo il primo valore di PWM per cui il sistema risponde a destra degli 0.

2. Onda Trapezoidale Periodica

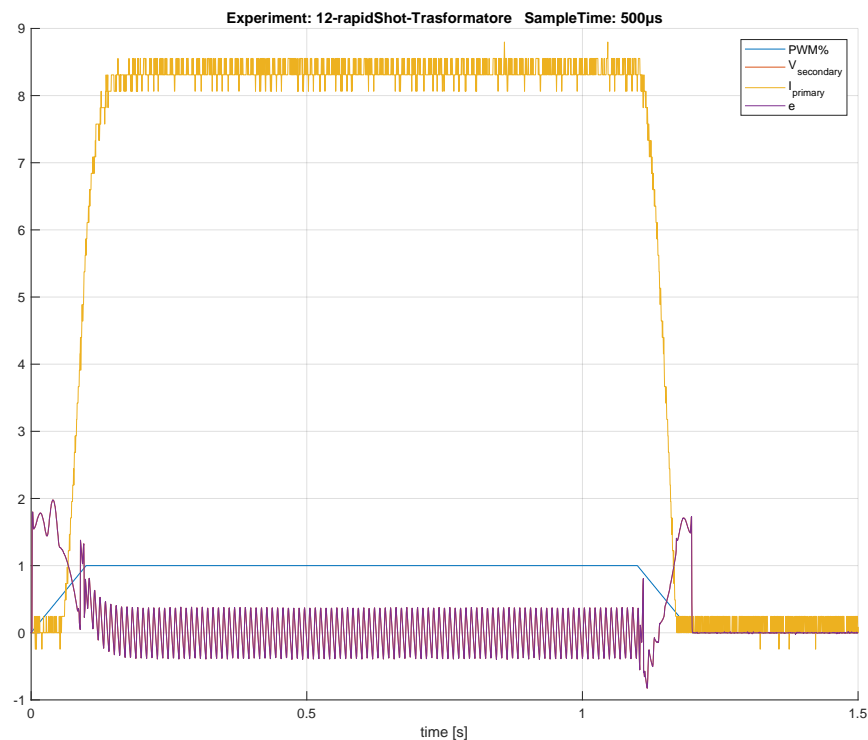


Figura 2.12: Onda Trapezoidale

Dead-Zone Superiore Con questo secondo segnale, si vuole mettere in evidenza il ritardo durante la discesa della rampa, pari a circa 20ms (*guarda 1.1s*), questo ritardo è in realtà dovuto da una seconda Dead-Zone presente però ai Duty-Cycle alti del PWM.

Disturbo 50Hz Essendo in oltre presente un segnale costante per un pò, quando i transistori terminano risulta evidente la presenza della 50hz nel segnale della corrente proveniente dall'alimentatore, questo disturbo è però dovuto alla fonte della corrente, ovvero la 220Vac del laboratorio, il medesimo esperimento realizzato con una batteria ovviamente non ha simili disturbi, ma in fase di test del driver, si è preferito usare un alimentatore da laboratorio.

Capitolo 3

Architettura di Sistema

3.1 Architettura ad alto livello

Il progetto finale ha come obiettivo la realizzazione di un architettura di controllo per le bobine poloidali presenti nei reattori tokamak.

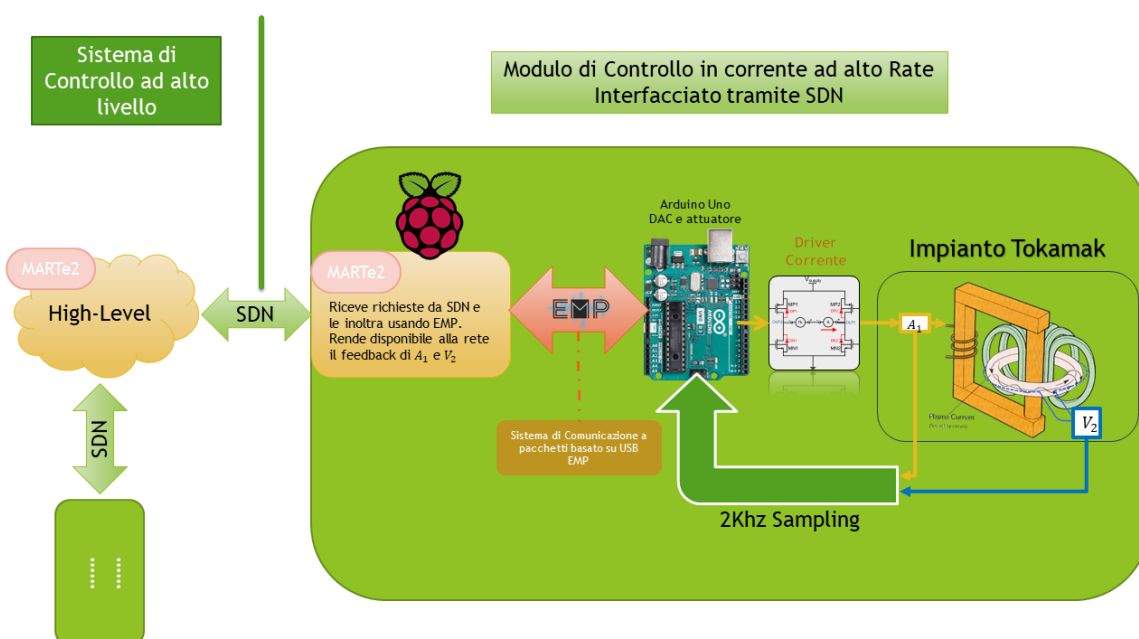


Figura 3.1: Architettura di controllo

Lo schema proposto realizza l'obiettivo è controllare una singola bobina, il progetto finale prevederà la ripetizione in serie del medesimo schema per il numero di bobine necessarie.

Dallo schema risulta evidente che tutti i componenti visti nel capitolo "???" si relazionano con lo stesso μ **Controllore**: l'**Arduino Uno**.

Per riportare i dati fuori e ricevere il riferimento da inseguire nella V_2 , è stata realizzato il [EMP - Libreria di Comunicazione Seriale](#)

[Embedded Message Pack](#), essa è stata scritta in C++ affinché possa essere Cross-Platform.

Il suo compito specifico, in questo progetto, è di mettere in comunicazione l'**Arduino Uno** con un nodo *MARTe2* installato su di una **Raspberry Pi**.

Quest'ultimo nodo ha il compito di mettere in rete il feedback dell'esperimento, e comunicare all'**Arduino Uno** eventuali cambio di riferimento. Questo ultimo tratto è realizzato mediante il protocollo **SDN**, che viaggia sopra Ethernet e dà garanzie Real-time.

Nella sua forma finale, il progetto prevede la riproduzione in serie di questo schema di controllo per arrivare a controllare tutte le bobine poloidali presenti in un tokamak.

EMP - Libreria di Comunicazione Seriale Embedded Message Pack



EMP (Embedded Message Pack) nasce con l'obiettivo di standardizzare un protocollo e creare una libreria C++ basata su classi Template, che permetta di automatizzare e standardizzare tutto il lavoro di programmazione necessario all'invio/ricezione di dei pacchetti dal formato Pre-Concordati tra 2 Device connessi Peer2Peer (Nessuna pretesa di network-ing).

Il raggiungimento dei suoi obiettivi, si sposa con la possibilità di supportare altre features interessanti:

Multiple-Package Il protocollo di comunicazione che si è deciso di usare per EMP ha permesso di estendere il suo funzionamento e permettere il trasporto, attraverso lo stesso mezzo, di ***pacchetti di tipologia e dimensione diversa*** all'interno della stessa libreria, evitando al contempo di inviare per ogni pacchetto più byte di quelli strettamente necessario. ⇒ **Alta Efficienza**

Zero Tempo di negoziazione Sempre grazie al protocollo di comunicazione, EMP è adatto ad un uso ‘Streaming’, questo perché non è necessario alcuna fase di sincronizzazione iniziale o durante la trasmissione in caso di perdita di dati, in aggiunta a ciò, EMP è in grado di scartare pacchetti errati in maniera trasparente all’utente. Tutto questo grazie al protocollo che **Auto-delimita i singoli pacchetti**. \Rightarrow **Trasparenza Totale**

Responsabilità Le uniche responsabilità a carico degli utilizzatori sono il riempimento dei pacchetti e la definizione degli stessi tra i 2 estremi della comunicazione.

Consistent Overhead Byte Stuffing (COBS)

Il protocollo di comunicazione che permette l’invio di **pacchetti diversi e senza fasi di negoziazione** alla base della libreria è **COBS**(IEEE, *Consistent Overhead Byte Stuffing (COBS)*).

Si tratta di un algoritmo per la codifica di byte, progettato per essere al tempo stesso efficiente e non ambiguo, che permette la definizione di *data-pack frame* **Auto-delimiti**.

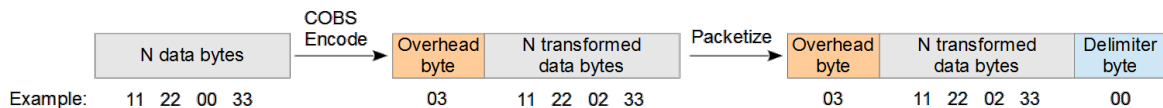


Figura 3.2: Esempio di COBS

title

3.1.1 Metodo di codifica

3.1.2 Struttura del codice

3.1.3 Benchmark

3.2 Online Sampling

3.2.1 Interconnessione Arduino \Leftrightarrow Companion

3.2.2 Storage su file delle informazioni

3.3 Post Elaborazione con Matlab

3.3.1 Conversioni Dati

3.3.2 Creazione dei grafici e Filtraggio

Capitolo 4

Modello teorico di Controllo

Se lo si desidera, utilizzare questo spazio per inserire un breve riassunto di ciò che verrà detto in questo capitolo. Inserire solo i punti salienti.

4.1 Controllo a Errore Nullo

4.2 Simulazione Qualitativa su Simulink

Capitolo 5

Sviluppo Controllo reale

Se lo si desidera, utilizzare questo spazio per inserire un breve riassunto di ciò che verrà detto in questo capitolo. Inserire solo i punti salienti.

5.1 Codifica del controllore

5.2 Tuning delle costanti

5.3 Esperimenti

Capitolo 6

Conclusioni e sviluppi futuri

Inserire qui le conclusioni trovate con la tesi, ed eventualmente eventuali idee per sviluppi futuri.

Appendice A

Arduino Code

6.1 Set-up Registri

Tic Timer

```
1 void periodicTask(int time) { // time in micro secondi
2   // PWM pin Disable, modalita CTC(pt1)
3   TCCR2A = (0x0 << COM2A0) | (0x0 << COM2B0) | (0x2 << WGM20);
4   // CTC(pt2), Prescalere 256
5   TCCR2B = (0 << WGM22) | (0x6 << CS20);
6   // T_cklock * Twant / Prescaler = valore Registro
7   OCR2A = (int)(16UL * time / 256);
8   TIMSK2 = (1 << OCIE2A); // attivo solo l'interrupt di OC2A
9 }
```

Listing 6.1: Tic Timer

Questa funzione imposta il TIMER2 in modalità Fast PWM, ovvero che si resetta quando arriva al conteggio finale, e calcola il valore da mettere nel registro affinché il conteggio sia il più vicino possibile a tempo desiderato

Frequenza PWM

```
1 enum pwmFreq: char {
2   hz30, hz120, hz490, hz4k, hz30k
3 };
4
5 void setMotFreq(pwmFreq freq) {
6   // TCCR0B is for Timer 0
7   #define myTimer TCCR0B
8   switch (freq) {
9     // set timer 3 divisor to 1024 for PWM frequency of 30.64 Hz
10    case hz30:
11      myTimer = (myTimer & B11111000) | B00000101;
12      break;
13    case hz120:
14      // set timer 3 divisor to 256 for PWM frequency of 122.55 Hz
15      myTimer = (myTimer & B11111000) | B00000100;
16      break;
17    case hz490:
18      // set timer 3 divisor to 64 for PWM frequency of 490.20 Hz
19      myTimer = (myTimer & B11111000) | B00000011;
```

```

20     break;
21     case hz4k:
22         // set timer 3 divisor to      8 for PWM frequency of  3921.16 Hz
23         myTimer = (myTimer & B11111000) | B00000010;
24         break;
25     case hz30k:
26         // set timer 3 divisor to      1 for PWM frequency of 31372.55 Hz
27         myTimer = (myTimer & B11111000) | B00000001;
28         break;
29     default:
30         setMotFreq(hz4k);
31         break;
32     }
33     #undef myTimer
34 }

```

Listing 6.2: Frequenza PWM

Mediante questa funzione si modifica il valore del Prescaler per il TIMER 0, modificando la velocità di conteggio si ottiene un PWM con una periodo, e quindi frequenza, che varia.

Offset Calculation

```

1  #define offsetCalc(pin,n)          \
2  ({                                  \
3      long read = 0;                 \
4      for (int i = 0; i < 1 << n; i++) { \
5          read += analogRead(pin);    \
6          delay(1);                   \
7      }                               \
8      read = read >> 5;               \
9      read;                           \
10 })

```

Listing 6.3: Macro per calcolo Offset

La macro misura con una distanza temporale di 1 ms 2^n -volte, la misura del Pin, e successivamente ne fa la media eseguendo un revers shift, questa tecnica di divisione semplicemente minimizza il tempo di calcolo, poiché ogni revers shift equivale a una divisione per 2, e si ha l'effetto di dividere per $\frac{read}{2^n}$ in $O(n)$ con un operazione per il processore molto semplice.

6.2 Generatore di Segnale

Per generare i segnali di controllo in Feed-Forward usati nel sistema, sono stati usati 2 diversi livelli di programmazione.

Un primo livello segnali di base, definiti su tutto \mathbb{R} , e usabili a piacere, e dei segnali composti e periodici da mandare durante l'esperimento. Tutti i segnali sono pensati per andare da -100% <-> 100%, è compito dell'attuazione eliminare le deadzone e traslare il controllo al valore più opportuno

6.2.1 Segnali Base

Rampa

```

1 int ramp(uint64_t t, int vStart, uint64_t tStart, int vEnd, uint64_t tEnd) {
2     // Saturazione
3     if (t < tStart)
4         return vStart;
5     else if (t > tEnd)
6         return vEnd;
7     // Retta
8     unsigned int dt = t - tStart;
9     return vStart + int((vEnd - vStart) / float(tEnd - tStart) * dt);
10 }

```

Listing 6.4: Rampa Saturata

La rampa è descritta come una retta nell'intervallo di interesse, saturata prima e dopo il tempo

desiderato

$$RampaSat(t) = \begin{cases} v_{start} + \frac{v_{end}-v_{start}}{t_{end}-t_{start}} * (t - t_{start}) & \forall t \in [t_{start}, t_{end}] \\ v_{start} & t < t_{start} \\ v_{end} & t > t_{start} \end{cases}$$

6.2.2 Segnali Composti

Onda Triangolare

```

1 int triangleSignal(uint64_t t, int msQuartPeriod) {
2     static uint64_t startTic = 0;
3     int dTic = t - startTic;
4     int pwm = 0;
5     if (dTic < ticConvert(msQuartPeriod))
6         pwm = ramp(dTic, 0, 0, 100, ticConvert(msQuartPeriod));
7     else if (dTic < (ticConvert(msQuartPeriod) * 3))
8         pwm = ramp(dTic, 100, ticConvert(msQuartPeriod), -100, ticConvert(msQuartPeriod)
9             * 3);
10    else if (dTic < (ticConvert(msQuartPeriod) * 4))
11        pwm = ramp(dTic, -100, ticConvert(msQuartPeriod) * 3, 0, ticConvert(msQuartPeriod)
12            * 4);
13    else {
14        pwm = 0;
15        startTic = t;
16    }
17    return pwm;
18 }
```

Listing 6.5: Onda Triangolare Periodica

Onda Trapezoidale

```

1 int rapidShot(uint64_t t) {
2     static uint64_t startTic = 0;
3     int pwmRapidShot;
4     long dTic = t - startTic;
5     if (dTic > t4) {
6         startTic = t;
7         pwmRapidShot = 0;
8         dTic = t - startTic;
9     }
10
11    if (dTic <= t1) {
12        pwmRapidShot = ramp(dTic, 0, 0, 100, t1);
13    } else if (dTic <= t2) {
14        pwmRapidShot = 100;
15    } else if (dTic <= t3) {
16        // falling ramp
17        pwmRapidShot = ramp(dTic, 100, t2, 0, t3);
18    } else if (dTic <= t4) {
19        pwmRapidShot = 0;
20    }
21    return pwmRapidShot;
22 }
```

Listing 6.6: Onda Trapezoidale Periodica

6.3 Codici Controllore

Appendice B

EMP Code

Appendice C

Matlab Post Elaborazione

Elenco delle figure

2.1	Schema interno di un Tokamak	4
2.2	Circuito Equivalente Bobina/Plasma all'interno di un Tokamak	5
2.3	Circuito Equivalente Bobina/Plasma all'interno di un Tokamak trascurando l'induzione del plasma verso la bobina	5
2.4	Sensore di Corrente ACS770LCB-100B-PFF-T	7
2.5	ACS770LCB-100B-PFF-T Schema a Blocchi	8
2.6	ACS770LCB-100B-PFF-T Sensibilità rispetto Temperatura	9
2.7	ACS770LCB-100B-PFF-T <i>Non-Linearità</i>	10
2.8	ACS770LCB-100B-PFF-T Schema di collegamento dal Datasheet	11
2.9	Driver Motori IBT-2 TopView & PinOut	13
2.10	IBT-2 Schema Elettrico	14
2.11	Esperimento con Onda Triangolare	16
2.12	Esperimento con Onda Trapezoidale	17
3.1	Schema finale dell'architettura di controllo	18
3.2	Esempio di COBS	21

Bibliografia

Allegro: High-Precision Linear Current Sensor**ACS770**

Allegro. *High-Precision Linear Current Sensor*. ACS770LCB-100B-PFF-T. Datasheet. 31 May 2019. URL: <https://www.allegromicro.com/~media/Files/Datasheets/ACS770-Datasheet.pdf>.

Handsontec: BTS7960 High Current 43A H-Bridge Motor Driver**IBT-2**

Handsontec. *BTS7960 High Current 43A H-Bridge Motor Driver*. Datasheet. 18 Sep 2019. URL: <https://www.handsontec.com/dataspecs/module/BTS7960%20Motor%20Driver.pdf>.

IEEE: Consistent Overhead Byte Stuffing (COBS)**COBS**

IEEE. *Consistent Overhead Byte Stuffing (COBS)*. Rapp. tecn. IEEE, 29 Jun 2002. URL: https://www.ieee802.org/3/efm/public/jul02/copper/oksman_copper_1_0702.pdf.

Infineon: BTS7960B High Current PN Half Bridge NovalithIC T**BTS7960b**

Infineon. *BTS7960B High Current PN Half Bridge NovalithIC T*. BTS7960b. Datasheet. 12 Jul 2016. URL: <https://www.infineon.com/dgdl/bts7960b-pb-final.pdf?folderId=db3a3043156fd5730116144c5d101c30&fileId=db3a30431ed1d7b2011efe782ebd6b60>.

Romero et al.: Real time current profile control at JET**TokamakCircuit**

Jesús Romero et al. «Real time current profile control at JET». In: *Fusion Engineering and Design* 43 (feb. 1998), pp. 37–58. DOI: [10.1016/S0920-3796\(98\)00261-0](https://doi.org/10.1016/S0920-3796(98)00261-0).

Capitolo 7

Software, Toolchain, Strumenti