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

Corpo dei ringraziamenti

Capitolo 1

Elementi Costitutivi

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.

1.1 Trasformatore - Modello di Tokamak

La tesi va scritta usando la terza persona, per quanto possibile, tranne casi veramente eccezionale.

In inglese è piuttosto standard usare la prima persona (plurale) in testi tecnici. In italiano no.

1.1.1 Modellazione Fisica

1.1.2 Funzione di Trasferimento

1.2 Sensore di Corrente

Benché per gli obiettivi di controllo la lettura della corrente sul primario non è indispensabile, si è però preferito poter misurare cosa stia succedendo all'interno del sistema.

1.2.1 Regime di funzionamento

Per allo scopo di misurare la corrente è stato messo in serie al primario il sensore di Corrente Allegro, *High-Precision Linear Current Sensor* Le scelte che hanno portato alla sua scelta sono:

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	

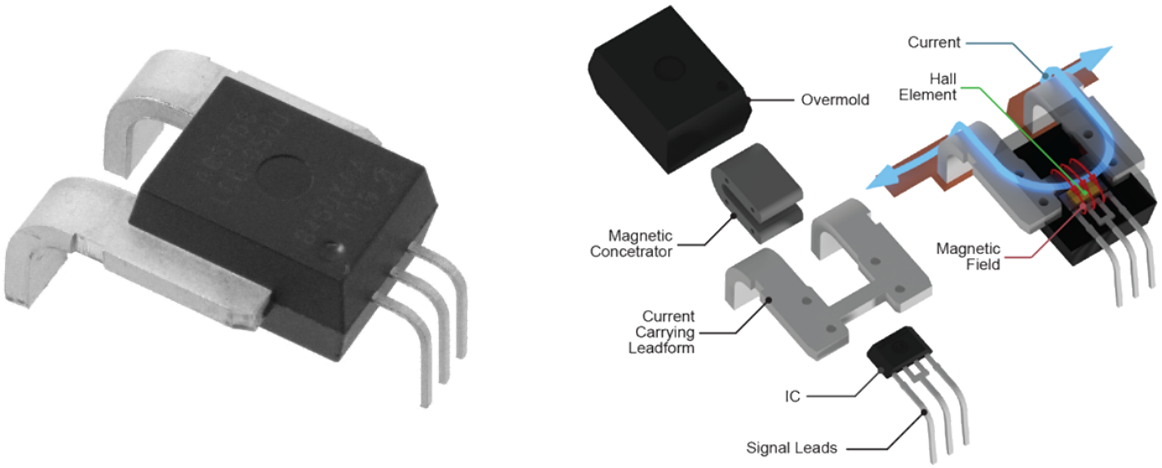

Figura 1.1: Sensore di Corrente

Delle tante varianti presenti, si è scelto di usare la ACS770LCB-100B-PFF-T, le cui caratteristiche chiave di questa variante sono:

Primary Sampled Current:	$\pm$ 100 A
Sensitivity Sens (Typ.)	20(mV/A)
Current Directionality	Bidirectional
T_{OP}	–40 to 150 (°C)

Queste caratteristiche lo rendono adatto per misurare i nostri esperimenti comprendo tutti i possibili valori di corrente misurabili.

Essendo però il principio di funzionamento basato su un sensore a effetto Hall, ovvero una misura diretta del campo magnetico indotto, è importante tenere distante il sensore dal trasformatore che nei suoi momenti di massimo flusso, genera ovviamente un campo magnetico non indifferente.

1.2.2 Funzionamento Interno

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, come è possibile vedere nel suo schema a blocchi:

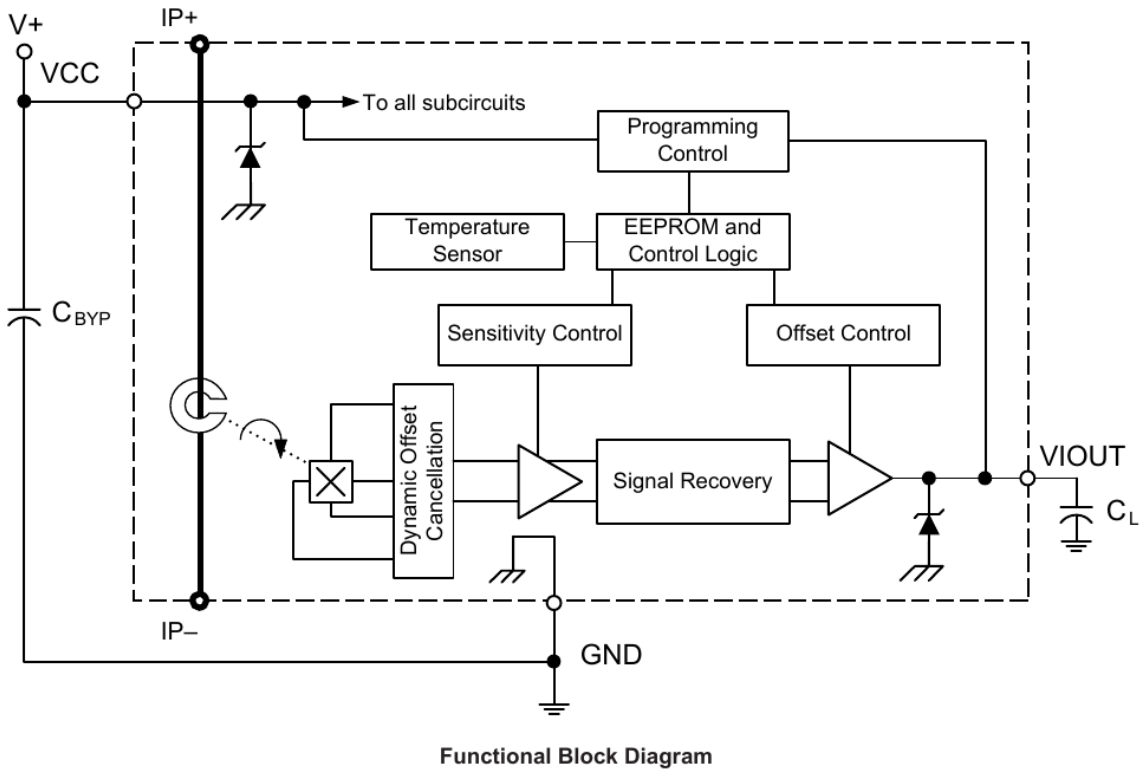

Figura 1.2: Schema a Blocchi

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 in

Hardware di compensare derive termiche e *Non-Linearità*, ottenendo un output assimilabile a un segnale lineare:

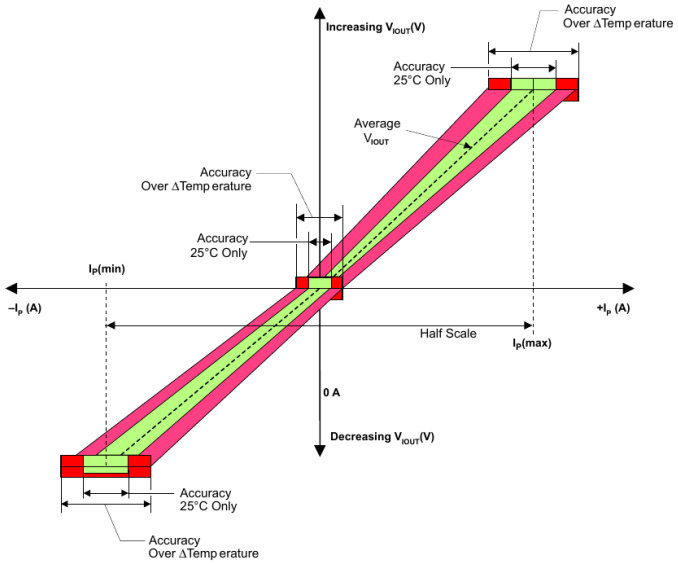

Figura 1.3: Sensibilità

Esso ovviamente varia in base alla temperatura, e l'errore è tanto più marcato tanto maggiore è la corrente da misurare, ma leggendo dal datasheet abbiamo che questo errore, che dipende si dalle temperature di esercizio dell'esperimento, non è mai, neanche negli esperimenti più sfortunati, superiore al $\pm 2\%$.

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

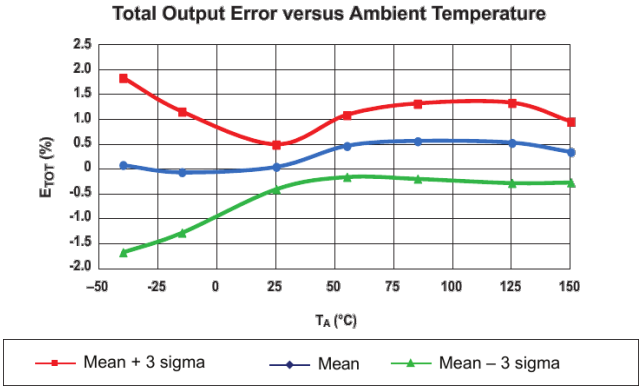

Figura 1.4: Temperatura/NonLinearità

1.2.3 Connessione elettrica

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

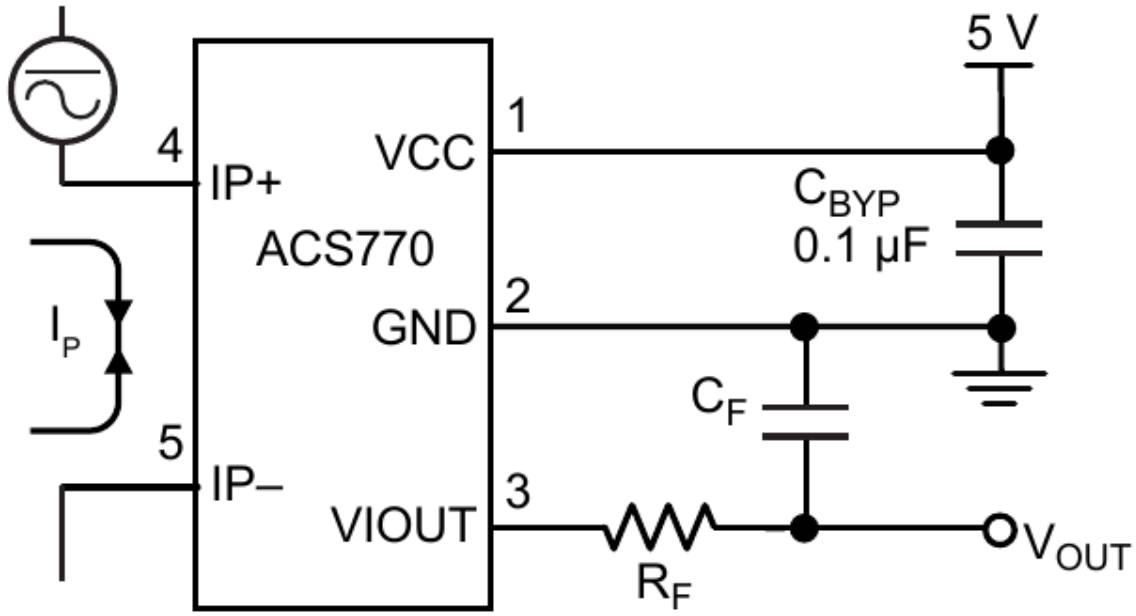

Figura 1.5: Schema di Collegamento

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.

1.2.4 Misura

La misura della corrente viene riportata 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:	± 100 A
Sensitivity Sens (Typ.)	20(mV/A)
Current Directionality	Bidirectional

Ciò implica che la corrente misurata, è calcolabile come:

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

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

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

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

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.

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

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

Portando questo valore nella corrente otteniamo che la Sensibilità in corrente del μControllore è pari a:

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

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

1.3 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* .

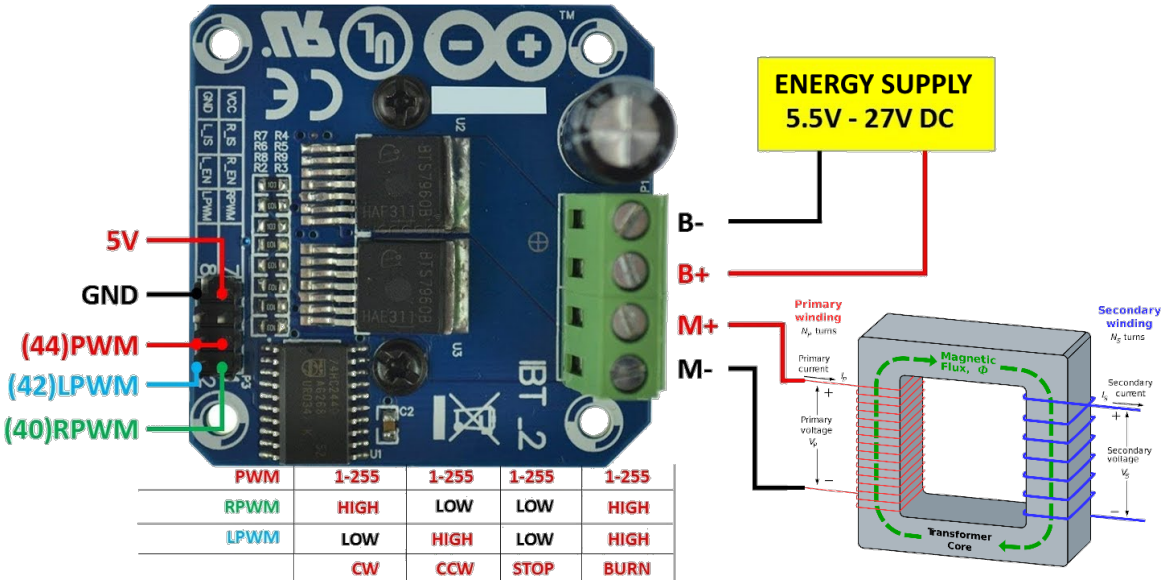

Figura 1.6: IBT-2 TopView.

Esso non è un comune Ponte-H integrato, per poter gestire potenze superiori è stato costruito usando 2 Half-Bridge collegati insieme mediante una opportuna logica per ricreare un normale Ponte-H.

Questa scheda in particola 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.

1.3.1 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-Brige 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.

1.3.2 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

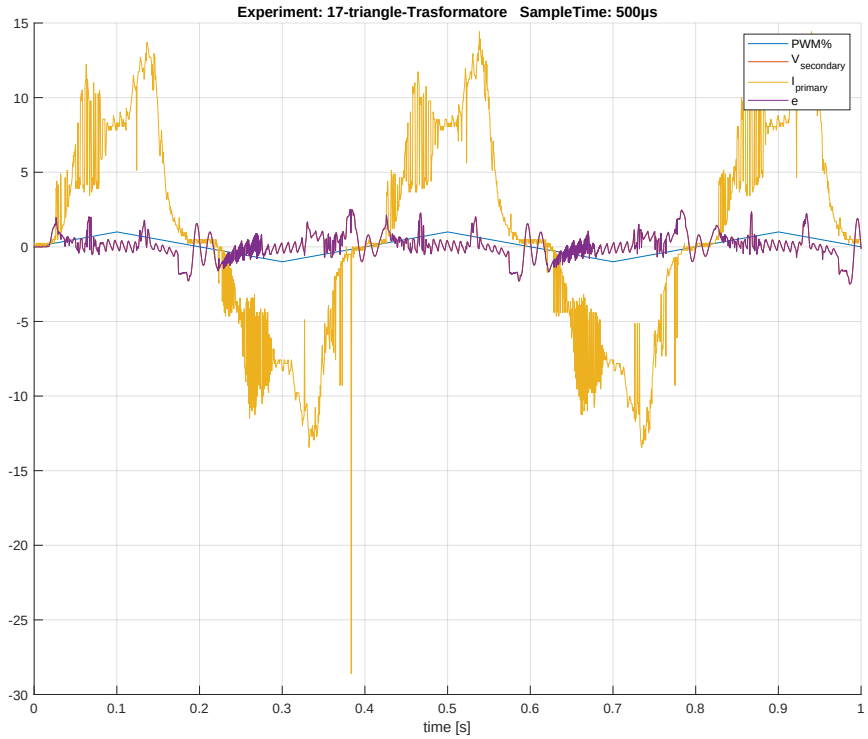

Figura 1.7: Onda Triangolare

Dead-Zone Inferiore L'onda triangolare si presta bene per far risaltare la problematica della Dead-Zone Inferiore, infatti in tutti gli intorni 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

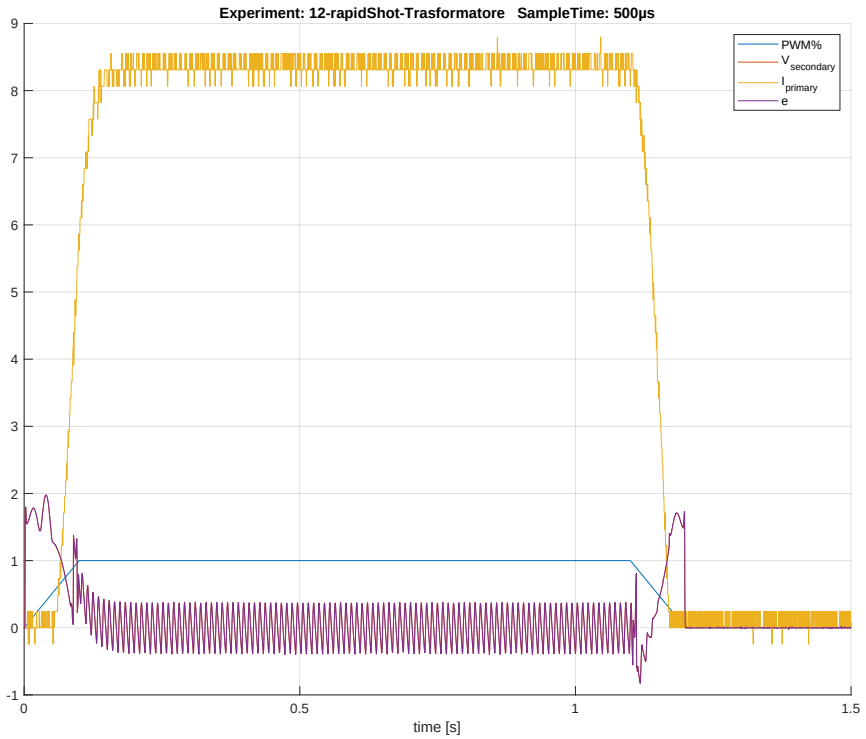

Figura 1.8: Onda Trapezoidale

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 transitori 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 non ha disturbi così marcati.

Capitolo 2

Architettura di Sistema

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.

In cosa consiste la catena di acquisizione

2.1 Architettura ad alto livello

2.2 Protocollo di Comunicazione - EMP

2.2.1 Metodo di codifica

2.2.2 Struttura del codice

2.2.3 Benchmark

2.3 Online Sampling

2.3.1 Interconnessione Arduino ⇔ Companion

2.3.2 Storage su file delle informazioni

2.4 Post Elaborazione con Matlab

2.4.1 Conversioni Dati

2.4.2 Creazione dei grafici e Filtraggio

Capitolo 3

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.

3.1 Controllo a Errore Nullo

3.2 Simulazione Qualitativa su Simulink

Capitolo 4

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.

4.1 Codifica del controllore

4.2 Tuning delle costanti

4.3 Esperimenti

Capitolo 5

Conclusioni e sviluppi futuri

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

Appendice A
Arduino Code

5.1 Set-up Registri

Tic Timer

```
1   void periodicTask(int time) {      // time in micro secondi
2     // PWM pin Disable, motalita 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 5.1: 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 affinchè 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 5.2: 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 5.3: 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.

5.2 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 compositi 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

5.2.1 Segnali Base

Rampa

```
int ramp(uint64_t t, int vStart, uint64_t tStart, int vEnd, uint64_t tEnd) {
  // Saturazione
  if (t < tStart)
    return vStart;
  else if (t > tEnd)
    return vEnd;
  // Retta
  unsigned int dt = t - tStart;
  return vStart + int((vEnd - vStart) / float(tEnd - tStart) * dt);
}
```

Listing 5.4: Rampa Saturata

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

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

5.2.2 Segnali Composti

Onda Triangloare

```
int triangleSignal(uint64_t t, int msQuartPeriod) {
  static uint64_t startTic = 0;
  int dTic = t - startTic;
  int pwm = 0;
  if (dTic < ticConvert(msQuartPeriod))
    pwm = ramp(dTic, 0, 0, 100, ticConvert(msQuartPeriod));
  else if (dTic < (ticConvert(msQuartPeriod) * 3))
    pwm = ramp(dTic, 100, ticConvert(msQuartPeriod), -100, ticConvert(msQuartPeriod)
    * 3);
  else if (dTic < (ticConvert(msQuartPeriod) * 4))
    pwm = ramp(dTic, -100, ticConvert(msQuartPeriod) * 3, 0, ticConvert(msQuartPeriod
    ) * 4);
  else {
    pwm = 0;
    startTic = t;
  }
  return pwm;
}
```

Listing 5.5: Onda Triangolare Periodica

Onda Trapezoidale

```
int rapidShot(uint64_t t) {
  static uint64_t startTic = 0;
  int pwmRapidShot;
  long dTic = t - startTic;
  if (dTic > t4) {
    startTic = t;
    pwmRapidShot = 0;
    dTic = t - startTic;
  }

  if (dTic <= t1) {
    pwmRapidShot = ramp(dTic, 0, 0, 100, t1);
  } else if (dTic <= t2) {
    pwmRapidShot = 100;
  } else if (dTic <= t3) {
    // falling ramp
    pwmRapidShot = ramp(dTic, 100, t2, 0, t3);
  } else if (dTic <= t4) {
    pwmRapidShot = 0;
  }
  return pwmRapidShot;
}
```

Listing 5.6: Onda Trapezoidale Periodica

5.3 Codici Controllore

Appendice B
EMP Code

Appendice C
Matlab Post Elaborazione

Elenco delle figure

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.