Aggiunta del controllore fuzzy su AIRBOARD

Stefano Bruschieri, PierPaolo Campari

Indice

Capitolo 1

Introduzione

1.1 Obiettivo del progetto

Implementazione di un controllo fuzzy su microprocessore (PIC18f252 - www.microchip.com) per il controllo di un motore elettrico per il movimento di un robot autonomo. Si tratta di scrivere un sistema di controllo a regole da sostituire allattuale controllo PID e fare gli opportuni test di confronto.

1.2 Fasi del progetto

Il progetto si é svolto in due fasi:

1. Scelta delle membership functions e di tutti parametri fuzzy attraverso la simulazione del controllore in ambiente Matlab. Confronto del controllore fuzzy con un controllore PID.

2. Implementazione vera e propria del controllore utilizzando il linguaggio di programmazione C e un compilatore compatibile con il PIC di riferimento. Il codice sorgente e i tool compatibili vengono ereditati dal progetto preesistente *AIRBoard* che realizza un controllore classico PID.
   Tools, entrambi reperibili sul sito www.microchip.com:

   MPLAB IDE ambiente di sviluppo integrato

   MPLAB C18 compilatore C

Capitolo 2

Progetto del controllore fuzzy

2.1 Progetto del controllore fuzzy

In fase di progetto sono stati individuati principalmente due tipi di controllori fuzzy: il primo considera come input l'errore e la variazione di velocitá del motore; il secondo invece segue il classico paradigma di realizzazione del controllore in funzione dell'errore e della variazione dell'errore stesso. Al fine di non distanziarci eccessivamente dal progetto originale di *AIRboard* si é scelto di implementare il secondo tipo di controllore. Come giá detto si é scelto di utilizzare come input l'errore e la variazione di errore, come output si ha l'incremento della variabile di controllo che nel nostro caso é il voltaggio applicato al motore.

Durante la fase di progetto non é stato possibile trascurare i vincoli derivanti dal fatto che il controllore doveva girare su un microprocessore con limitate capacitá sia di calcolo che di memoria. In particolare la fase di fuzzyficazione delle variabili di input non puó essere eseguita nel periodo di controllo perché troppo onerosa da un punto di vista computazionale; la fuzzificazione si traduce invece nella lettura di due tabelle (una per ciascun input).

Si é quindi deciso di alleggerire il carico computazionale del microprocessore a scapito di un maggiore utilizzo della EEPROM che, d'altra parte, ha una dimensione limitata a soli 256 byte. Sia il primo che il secondo input hanno valori che variano da -100 a +100; valori al di fuori di questo range vengono approssimati o al limite inferiore o a quello superiore. Sfruttando il fatto che le membership functions devono essere simmetriche rispetto allo 0 é possibile considerare solo il semiasse positivo; questo fatto, unito al vincolo che in corrispondenza di ciascuna ascissa la somma delle mf deve essere esattamente pari a 1 permette di fuzzyficare utilizzando un array di solo 101 valori per ogni input. Per il progetto del controllore fuzzy si é utilizzato

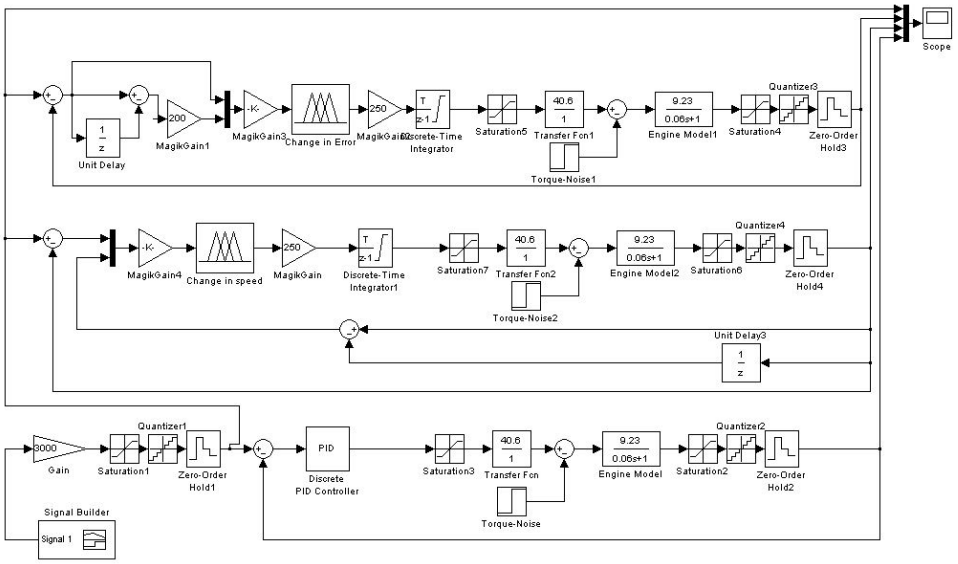

Figura 2.1: Schema simulink di confronto tra controllore fuzzy e PID

il *toolbox fuzzy* di Matlab[1] col quale peró non é possibile imporre i vincoli di simmetria e somma uguale a 1. Per garantire questi vincoli é necessario utilizzare forme di mf comode come ad esempio quelle triangolari o trapezioidali. Le mf gaussiane infatti renderebbero piú complessa l'imposizione dei due vincoli poiché il toolbox é grafico.

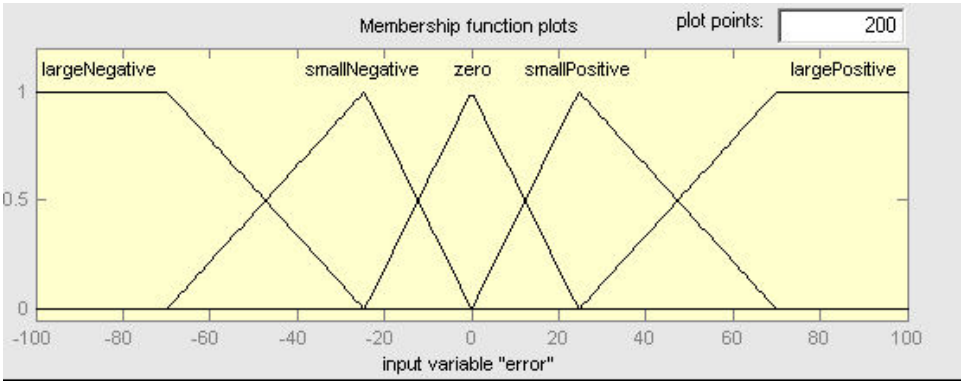

Figura 2.2: Membership functions riguardanti il primo input

Al solo scopo di appoggiare la fase preliminare del progetto abbiamo realizzato con Simulink[2] uno schema a blocchi in modo da valutare i risultati

[1] E' possibile richiamare il toolbox digitando il comnado *fuzzy* da workspace

[2] Tool, compreso in MatLab per la costruzione e l'analisi di modelli

dei controllori fuzzy (sia quello che considera come input la variazione di velocitá della ruota sia quello che ha come input la derivata dell'errore) in confronto al classico pid.

2.2 Strumenti di progetto

La difficoltá maggiore nel costruire una simulazione con Simulink sta nel modellizzare correttamente il motore. Il controllore fuzzy da noi ideato é pensato per controllare un insieme di sistemi il piú vasto possibile e quindi il modello del motore andrá modificato di volta in volta.

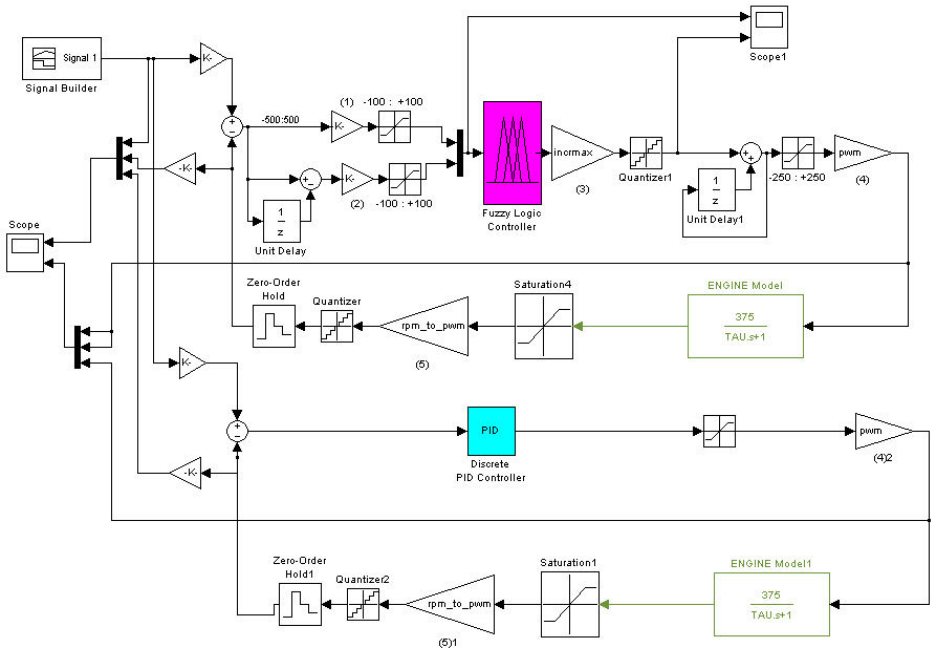

Figura 2.3: Simulazione del controllore fuzzy effetivamente implementato

Nonostante l'impossibilitá di realizzare un modello affidabile del sistema controllato, abbiamo realizzato uno schema Simulink (figura 2.3) allo scopo di studiarne l'andamento della risposta.

Questo schema, a differenza di quello preliminare (figura 2.1), é molto fedele al controllore implementato sul PIC[3].

[3]per dettagli implementativi leggere 3.3

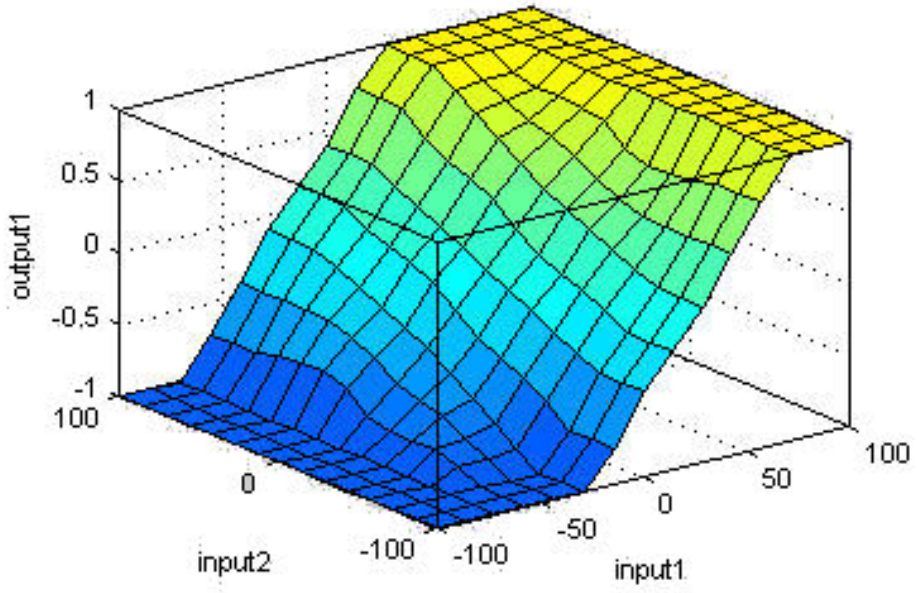

Figura 2.4: Superficie derivata dalle mf attualmente presenti nel PIC

Capitolo 3

Codifiche usate nel progetto

Una fase critica del progetto é stata la scelta delle codifica delle tabelle di fuzzyficazione e delle regole.

3.1 Codifica delle tabelle di fuzzyficazione

E' possibile caricare nella memoria del PIC due tabelle di fuzzyficazione contenenti esattamente 101 elementi ciascuna. La codifica delle due tabelle viene eseguita dallo stesso algoritmo contenuto nello script matlab *fuzzy_table*. Ciascun elemento della tabella é formato da un byte; di cui i 6 bit meno significativi indicano il livello di matching e i due bit piú significativi indicano quale membership function é abilitata (infatti, considerando che sul semiasse positivo si possono avere al massimo 4 mf, due bit sono sufficienti per indicare univocamente quella abilitata). Il primo bit indica che é attiva o la mf 0 (bit a 0) o la mf 2 (bit a 1); il secondo bit indica che che é attiva o la mf 1 (bit a 0) o la 3 (bit a 1).

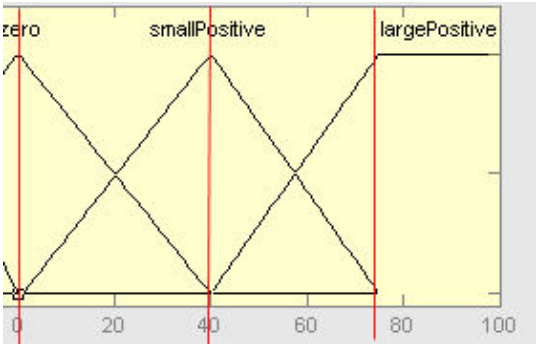

Figura 3.1: Esempio di matching per il primo input

Nella figura 3.1 si vedono 3 sezioni costruite nel semiasse positivo. Nella prima le uniche due mf che si sovrappongono sono *zero* e *smallPositive* cioé la mf 0 e la 1; quindi i primi 40 interi avranno i due bit piú significativi entrambi a 0. Nella seconda area la codifica é 10 e nellultima 11. E' chiaro che la codifica 01 é impossibile poiché non esistono fasce di ascisse in cui si intersecano la mf 0 e la 3. Da notare il fatto che ad ogni ascissa la somma delle mf é 1 e che in ogni punto si intersecano esattamente 2 mf.

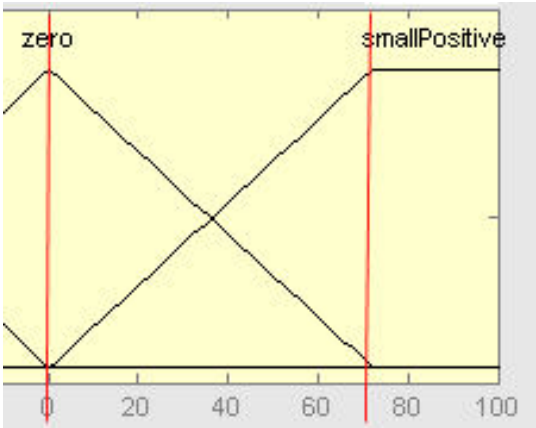

Figura 3.2: Esempio di matching per il secondo input

Per quanto riguarda il secondo input, esistono solo due fasce nel semiasse positivo. Nella prima fascia abbiamo le prime due mf (la 0 e la 1) e la codifica é quindi pari a 00. Nella seconda é presente solo la mf 1 e la codifica é 10.

Il classico paradigma fuzzy prevede che il grado di matching di ciascuna mf sia compresa tra 0 e 1. Potendo utilizzare solo 6 bit, il grado di matching varia da 0 a 63. In realtá i 6 bit indicano sempre il grado di matching della mf piú bassa. Ad esempio se si legge il valore corrispondente all'ascissa 40 troveremmo il seguente byte 10111111, infatti i primi 2 bit indicano che in quel punto troveremo le mf smallpositive e largepositive e il valore della mf piú bassa (smallpositive) é 63.

3.2 Codifica delle regole

AIRboard é in grado di leggere dalla eeprom le regole che costituiscono il controllore fuzzy. Ciascuna regola é codificata in un byte e, a causa delle limitazioni di memoria, é possibile caricare fino ad un massimo di 15 regole. Anche nella codifica delle regole abbiamo utilizzato il principio della simmetria, infatti, nella maggior parte dei controllori fuzzy, se esiste una regola del tipo:

Input 1	codifica
big_negative	—
small_negative	—
zero	00
small_positive	01
large_positive	10
none	11

Tabella 3.1: Codifica del primo input

> if in_1 is **small_positive** and in_2 is **big_positive** then out is a_value

esisterá anche la regola:

> if in_1 is **small_negative** and in_2 is **big_negative** then out is another_value

Per questa ragione se il sistema fuzzy da noi implementato contiene n regole, sará necessario caricare sul PIC solo

$$m = \frac{n}{2} + 1 < 15 \qquad \text{regole.} \tag{3.1}$$

La regola zero del tipo:

> If input_1 is **zero** and input_2 is **zero** then output is **zero**

non é sottintesa e deve essere inserita esplicitamente ogni qualvolta la si utilizzi.

I 2 bit piú significativi codificano le mf solo positive del primo input; i 3 bit meno significativi si riferiscono all'output codificando sia le mf positive che quelle negative e i 3 bit rimanenti codificano le membership function positive e negative del secondo input. In questo modo é possibile caricare regole miste del tipo:

> if input_1 is small_positive and input_2 is small_negative then output is zero

D'altra parte per simmetria il sistema dedurrá anche la regola ad essa simmetrica, cioé:

> if input_1 is small_negative and input_2 is small_positive then output is zero

Input 2	codifica
large_negative	001
small_negative	010
zero	011
small_positive	100
large_positive	101
none	111

Tabella 3.2: Codifica del secondo input

Output	codifica
supa_negative	000
large_negative	001
small_negative	010
zero	011
small_positive	100
large_positive	101
supa_positive	111

Tabella 3.3: Codifica dell'output

Decimale	Binario	Linguaggio naturale
027	00 011 011	if (error is ZERO) & (dError is ZERO) then (out is ZERO)
036	00 100 100	if (error is ZERO) & (dError is SMALL_POSITIVE) then (out is SMALL_POSITIVE)
083	01 010 011	if (error is SMALL_POSITIVE) & (dError is SMALL_NEGATIVE) then (out is ZERO)
092	01 011 100	if (error is SMALL_POSITIVE) & (dError is ZERO) then (out is SMALL_POSITIVE)
101	01 100 101	if (error is SMALL_POSITIVE) & (dError is SMALL_POSITIVE) then (out is LARGE_POSITIVE)
189	10 111 101	if (error is LARGE_POSITIVE) then (out is LARGE_POSITIVE)

Tabella 3.4: Esempi di codifica delle regole

3.3 Metodo di inferenza

Il metodo di inferenza utilizzato dal controllore fuzzy non si distacca dal canonico metodo *Sugeno*. In base al valore letto dagli encoder, il controllore ricava l'errore, tra varibile misurata e setpoint, e la variazione dell'errore nel tempo. Ricavati questi due termini il sistema carica dalla EEPROM i corrispondenti due bytes.

Dai due bytes vengono ricavati le membership function attivate dai due valori e il loro grado di matching. Degli antecedenti di ciascuna regola viene preso il minimo (and) e in questo modo viene calcolato il grado di attivazione di ciascuna regola. A questo punto, come si vede in figura 3.3, viene fatta una *media pesata* tra i pesi delle regole.

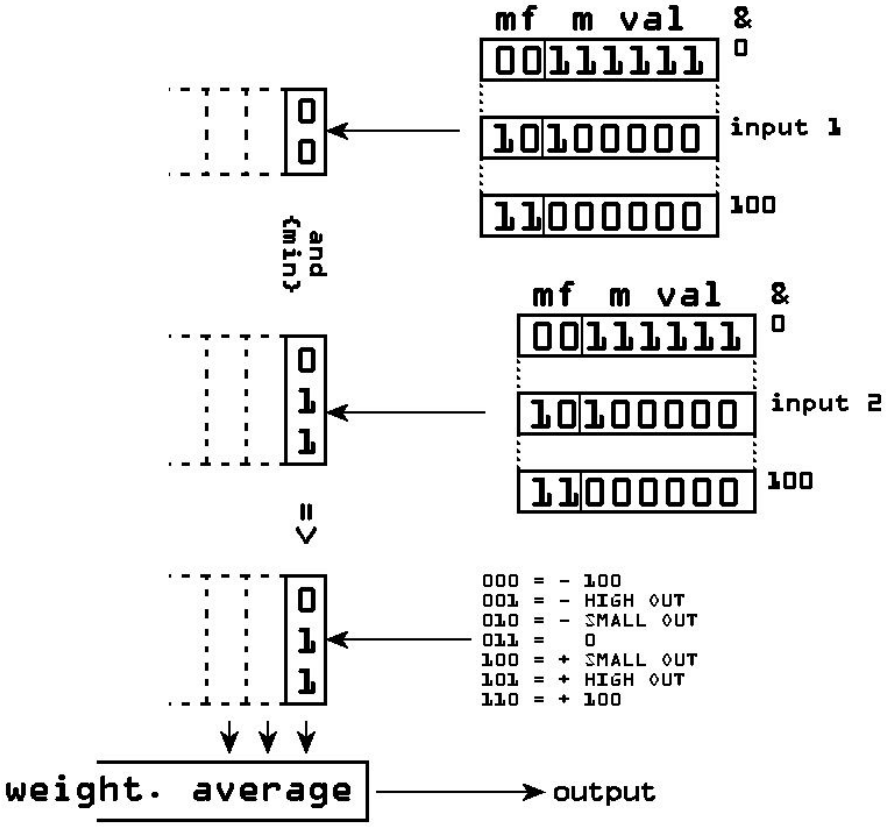

Figura 3.3: Schema del metodo di inferenza

Capitolo 4

Script di Matlab

In questo capitolo viene presentato come utilizzare gli script di matlab necessari per generare i file contenenti le codifiche delle regole e delle tabelle fuzzy. I file cosí generati verranno letti da *AIRboardUI*[1]. Oltre ai due script (fuzzy_table.m e rules.m) é necessario utilizzare il file di simulink *triangulars* e ovviamente il file changeError.fis nel quale é contenuto il progetto vero e proprio del controllore fuzzy ossia le forme delle membership function e le regole. Affinché funzioni triangulars in changeError gli input devono avere range da 0 a 1. Ricordarsi di impostare correttamente questo valore prima di esportare nella workspace di Matlab il progetto fuzzy. La prima cosa da fare prima di eseguire gli script é quella di avviare il file changeError.fis utilizzando il seguente comando da digitare nella workspace di Matlab

fuzzy changeError

A questo punto esportare il progetto nella workspace

file-> export -> To workspace...

o piú semplicemente *Ctrl+T*.

4.1 Tabelle fuzzy

Prima di attivare lo script é necessario generare un array provvisorio per ciascun input. Per generare i due array é necessario avviare la simulazione *triangulars* realizzata con Simulink. Lo schema a blocchi dipende dalla struttura del controllore nel senso che affinché funzioni changeError deve contenere un controllore con due input, di cui il primo ha 5 mfs triangolari e il secondo 3. D'altra parte é possibile modificare i parametri delle singole mfs o traslarle a piacere. Dopo la simulazione si hanno nel workspace due nuove variabili: *table_input_1* e *table_input_2*. A questo punto é necessario avviare per due volte lo script.

[1] Consultare l'apposito capitolo di *AIRBoard manual* per maggiori dettagli

fuzzy_table(table_input_1, 1)

genera il file first_input.dat

fuzzy_table(table_input_2, 2)

genera il file second_input.dat

4.1.1 Requisiti

Affinché lo script funzioni le tabelle di input devono presentare 3, 5 o 7 colonne, ciascuna corrispondente ad una membership function distinta ; le mfs devono essere simmetriche rispetto allo 0 e a somma pari a 1.

4.2 Regole

E' sufficiente richiamare lo script *rules* ed esso genera automaticamente il file rules.dat. Dalla workspace digitare semplicemente

rules

4.2.1 Requisiti

Affinché lo script funzioni deve esistere un input da 5 mf e uno da 3; le membership functions devono essere simmetriche e a somma pari a 1. Inoltre poichè le mf hanno un preciso identificativo nel file changeError.fis é necessario appurare che gli identificativi corrispondano esattamente alla mf corretta. I commenti nei due switch presenti nello script aiutano nell'identificazione delle mf.

Capitolo 5

Fase di testing del controllore

Durante la fase di testing abbiamo utilizzato il programma *AirboadUI*[1] collegato alla scheda[2] in figura 5.1 e collegato ad un robot sollevato da terra per motivi di sicurezza.

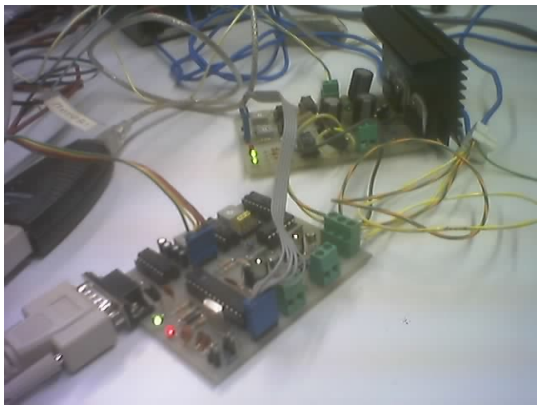

Figura 5.1: Scheda utilizzata per il testing

La fase iniziale prevede l'uploading dei parametri inerenti il controllore fuzzy ossia:

1. tabelle di fuzzyficazione
2. le regole di inferenza
3. i parametri:
   (a) incr max

[1]Per una descrizione completa di Airboard UI si rimanda al manuale utente di airboard
[2]Per una descrizione completa della schede di cotrollo e di potenza si rimanda alla tesina di Fedeli Francesco

(b) smallout

(c) highout

A questo punto abbiamo modificato il control time in modo da iniziare la fase di tuning. Infatti il valore MAX_TICK, derivante dal tuning, dipende dal parametro ct. In particolare nel nostro primo esperimento abbiamo impostato il ct pari a 100 (1/10 di secondo poiché il control time é espresso in millisecondi) col comando

ewct=100

A questo punto abbiamo avviato il tuning. Al termine di questa fase il controllore ha restituito come valore di MAX_TICK pari a 42. Questo parametro dipende anche da come é effettuato il collegamento dell'encoder al motore: se l'encoder é collegato alla ruota, come nel nostro caso, il parametro MAX_TICK é decisamente piú basso rispetto al caso di encoder collegato all'asse del rotore. In un secondo momento impostando il control time a 20, abbiamo osservato che il sistema impostava il MAX_TICK a 9.

Dopo la fase di tuning e dopo aver abilitato i motori abbiamo osservato come modificando il setpoint, la ruota raggiungesse effettivamente la velocitá impostata.