ALMA MATER STUDIORUM - UNIVERSITÀ DI BOLOGNA

SECONDA FACOLTA' DI INGEGNERIA
CON SEDE A CESENA

CORSO DI LAUREA
IN INGEGNERIA AEROSPAZIALE
Classe 10

Sede di Forlì

ELABORATO FINALE DI LAUREA

In MECCANICA DEL VOLO

SVILUPPO E IMPLEMENTAZIONE DI UNA LEGGE DI GUIDA PER VEICOLI TERRESTRI

CANDIDATO

Paolo Lombardi

RELATORE

Chiar.mo Prof. Fabrizio Giulietti

…….

Anno Accademico 2012/2013
Sessione III

Alla mia famiglia

Ringraziamenti

Desidero ringraziare sentitamente il Prof. Fabrizio Giulietti che mi ha seguito con sapienza e interesse durante le diverse fasi di questo percorso e ha permesso la realizzazione di questo progetto.

Un ringraziamento va anche al personale del Laboratorio di Meccanica del Volo, in modo particolare Matteo Turci, per i consigli e gli insegnamenti ricevuti durante questi mesi di lavoro. Inoltre, vorrei esprimere sincera gratitudine ai miei colleghi, Alberto Sodi e Encarnación Serrano Castillo , che mi hanno affiancato e supportato durante questa indimenticabile esperienza.

Paolo Lombardi

Indice

Elenco delle figure

Elenco delle tabelle

Introduzione

Negli ultimi anni, l'attenzione sulla possibilità di condurre una navigazione completamente o parzialmente autonoma da parte di velivoli e di veicoli è cresciuta notevolemente.

Basti pensare ai droni utilizzati in ambito bellico e di sorveglianza o all'utilizzo di UGV (Unmanned Ground Vehicle) in zone operative ad alto rischio.

Il seguente lavoro si prefigge l'obiettivo di validare una legge di guida che permetta la navigazione autonoma di un veicolo terrestre. La legge di guida studiata, nota come *beam rider*, storicamente utilizzata per il puntamento missilistico, si presta a una navigazione a waypoints.

Per garantire un effettiva autonomia si è dovuto dotare il nostro veicolo di sensori che fornissero una stima della posizione; la scelta è ricaduta su di un dispositivo GPS. Per avere un controllo più preciso del veicolo alle basse velocità, e per poter tarare l'autopilota di velocità (*Cruise Control*) si è scelto di utilizzare un sensore magnetico capace di rilevarne la velocità.

Per la validazione delle leggi di controllo e degli autopiloti è stata effettuata la simulazione in ambiente Matlab®/Simulink/Stateflow e successivamente per l'implementazione su schede di prototipazione rapida si è sfruttata la generazione automatica di codice C++ a partire dallo stesso schema Simulink.

L'elaborato consta di dieci capitoli. Nel primo capitolo si descrive la strumentazione utilizzata. Nei capitoli successivi si descrivono: la legge di guida, la parte di modellazione matematica del veicolo, i controllori automatici e la loro taratura, la logica di sicurezza e navigazione, lo schema Simulink impiegato nell'esperimento finale per la generazione del codice.

Nei capitoli finali si confrontano i dati simulati con i dati raccolti durante le prove sperimentali evidenziando la validità della legge di guida.

1 Descrizione della strumentazione

1.1 Veicolo terrestre

Come veicolo terrestre è stata impiegata un'auto radiocomandata tipicamente utilizzata nel campo dell'automodellismo.

E' stato possibile interfacciarne i servoattuatori e il comando del motore con delle schede di prototipazione rapida come la scheda *Arduino Duemilanove* in modo relativamente semplice.

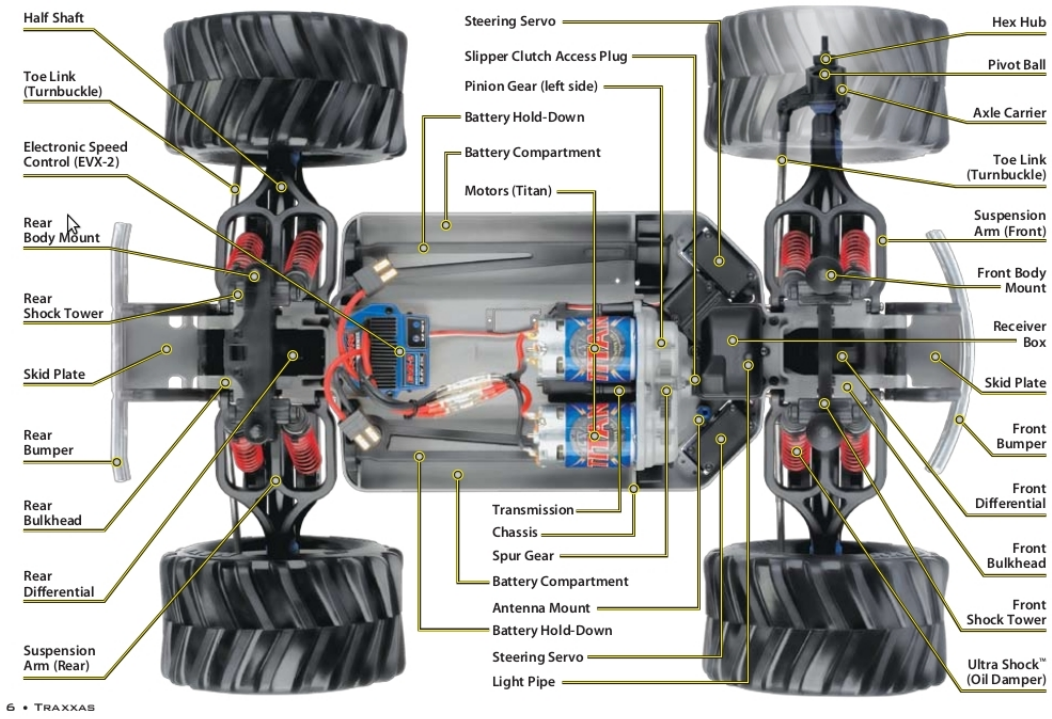

Figura 1: Auto E-Maxx

1.2 Scheda di prototipazione rapida : Arduino Duemilanove

Arduino Duemilanove è una scheda per la prototipazione rapida completamente open source, che negli ultimi anni ha avuto un notevole successo grazie alla facilità con la quale anche i neofiti possono realizzare dei progetti più o meno complessi, che toccano vari campi dell'elettronica applicata, dalla domotica alla robotica.

I motivi principali che hanno portato alla scelta di questa scheda sono i seguenti:

1. la facilità con la quale si può interfacciare con sensori/attuatori (quali GPS, moduli Xbee e servoattuatori), data l'esistenza di librerie reperibili in rete, che evitano di scrivere codice da zero e soprattuto non richiede conoscenze specifiche di programmazione per microcontrollori;

2. una documentazione molto valida, e numerosi progetti in rete;

3. il basso costo;

4. la possibilità di utilizzare anche codice in C e C++, generato a partire da schemi Simulink tramite il tool Code Generator, e di caricarlo direttamente tramite il suo IDE (*Integrated Development Environment*);

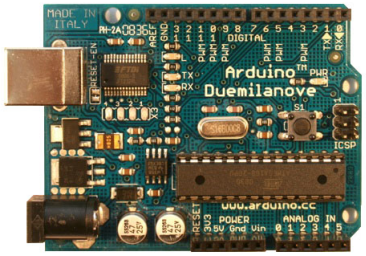

Figura 2: Arduino Duemilanove

Si riporta la tabella nella quale sono descritti i collegamenti dei pin di Arduino con gli altri moduli impiegati.

Numero Pin	funzione per arduino
0	RX seriale (TX GPS)
1	TX seriale (RX Xbee Pro)
2	1° canale ricevente
3	Giri motore
4	2° canale ricevente
5	3° canale ricevente
6	4° canale ricevente
7	5° canale ricevente
8	//
9	Servo1
10	ESC
11	Servo2
12	//
13	//

Tabella 1: Collegamenti di Arduino con gli altri moduli

1.3 Sensori

1.3.1 GPS

Per conoscere la posizione, la direzione e la velocità dell'auto si è utilizzato un *GPS Garmin* con frequenza di aggiornamento di 5Hz. La codifica delle informazioni avviene tramite delle frasi, le *sentenze NMEA*[1], in modo standardizzato. La sentenza utilizzata è la $GPRMC che contiene le informazioni essenziali descritte in seguito.

Per interfacciare il GPS con Arduino si è utilizzata una libreria disponibile in rete, la NMEA.h [1].

La sentenza $GPRMC ci fornisce informazioni di velocità, direzione, posizione e validità dell'informazione associata all'affidabilità ed integrità del segnale GPS.

[1] *NMEA* (o più specificatamente *NMEA 0183*) è uno standard di comunicazione di dati utilizzato soprattutto in nautica e nella comunicazione di dati satellitari GPS. L'ente che gestisce e sviluppa il protocollo è la *National Marine Electronics Association* . Questo protocollo si basa sul principio che la fonte, detta talker, può soltanto inviare i dati (sentences) e la ricevente, detta listener, può soltanto riceverli.

Figura 3: GPS Garmin

Si riporta un esempio di decodifica:
$GPRMC,225446,A,4916.45,N,12311.12,W,000.5,054.7,191194,020.3,E*68

NOME	ESEMPIO DATO	DESCRIZIONE
Identificativo sentenza	$GPRMC	Tipo di sentenza GPS
Tempo	225446	22ore 54 minuti 46 secondi UTC time
Validità segnale GPS	A	A=accettatoV=non valido
Latitudine	4916.45, N	49° 16.45" Nord
Longitudine	12311.12, W	123° 11.12' Owest
Velocità al suolo	000.5	0.5 knots
Direzione vera	054,7	54.7°
Data del fix	191194	19 novembre 1994
Magnetic variation	020.3 E	Magnetic variation 20.3° Est
Checksum	*68	Verifica dell'integrità dei dati GPS

Tabella 2: Esempio di decodifica sentenza $GPRMC

1.3.2 Sensore di velocità

Per la misurazione della velocità dell'auto si sono utilizzati due piccoli magneti al neodimio incollati solidalmente all'albero del motore elettrico in modo simmetrico. Misurando il tempo tra un passaggio e il successivo, si può facilmente ottenere una stima della velocità istantanea dell'auto.

Per rilevare il passaggio del magnete si è utilizzato un circuito molto comune in elettronica detto pick-up magnetico, il quale apre (o chiude) un circuito ogni qualvolta il magnete passi nelle immediate vicinanze del sensore. Tale variazione di livello logico, alto o basso, viene rilevata da Arduino tramite interrupt.

1.4 Attuatori

1.4.1 Servo

Per comandare lo sterzo dell'auto, azionato da servoattuatori elettrici, si sono adoperate le librerie *Servo*, già incluse di default nell'IDE di Arduino. Con questa libreria è possibile interfacciarsi in maniera immediata con servoattuatori, molto diffusi nel campo del modellismo, semplicemente comunicando il valore dell'angolo desiderato. Per il trimmaggio corretto dello sterzo, invece di intervenire sul trim dalla trasmittente, sono stati eseguiti dei test tramite Arduino per correggerne l'off set, eliminando così la necessità di ritarare i servo nell'eventualità in cui si cambi ricevente.

Figura 4: Servo

1.4.2 Motore ed ESC

I motori elettrici utilizzati nel campo del modellismo vengono solitamente controllati da un ESC[2] (*Electronic Speed Control*) .

Figura 5: ESC

[2]L'ESC è un circuito elettronico impiegato al fine di variare la velocità di un motore elettrico, la sua direzione, e può anche agire come freno dinamico. Sono spesso utilizzati su modelli radio controllati, in particolare per motori brushless ai quali forniscono una tensione trifase a bassa tensione.

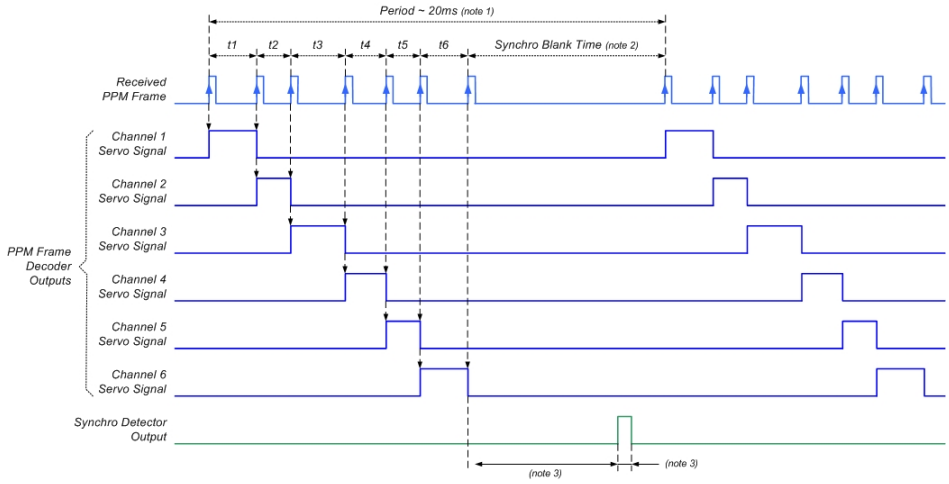

Figura 6: Esempio di segnale PPM

Nativamente l'ESC era collegata ad una ricevente da modellismo. Quest'ultima riceve un segnale PPM con un duty cicle di 50Hz, separa i segnali relativi ai diversi canali e li indirizza verso i servo o l'ESC.

Per comandare il motore si utilizza la stessa libreria *Servo* utilizzata in precedenza, giacchè i segnali sono della stessa natura ossia segnali di durata variabile da 1000 a 2000 microsecondi, centrati a 1500 microsecondi con frequenza di lavoro di 50 Hz.

L'informazione è contenuta quindi nella durata alta del valore. I segnali che sono utilizzati per comandare tali dispositivi, diffusissimi nel modellismo, sono gli stessi che vengono utilizzati per comandare i servo tramite radiotrasmittenti.

1.5 Comunicazione

1.5.1 Trasmittente e ricevente

Per la comunicazione in upload dalla stazione di terra, si sono utilizzati una trasmittente, una ricevente e 2 moduli Xbee Pro. Poichè ci si è accorti che le funzioni più semplici con le quali si potevano leggere i segnali dalla ricevente impegnavano per un tempo eccessivo il microcontrollore di Arduino, impedendo la contemporanea decodifica delle informazioni GPS, la ricevente a cinque canali è stata interfacciata utilizzando il progetto open-source MULTI-WII, opportunamente adattato e modificato per l'Arduino Duemilanove.

La funzione della trasmittente è quella di GCS (*Ground Control Station*) la cui logica verrà descritta in dettaglio in seguito (vedi Capitolo 5).

Figura 7: Radiocomando

Figura 8: Ricevente a 5 canali

1.5.2 Xbee Pro

Il modulo *Xbee Pro* permette la comunicazione in chiaro con altri moduli dello stesso tipo semplicemente alimentandoli. Si è collegato, sulla seriale di Arduino, il modulo che deve trasmettere i dati al modulo Xbee Pro (collegato a sua volta ad un PC portatile che funge da GCS per la telemetria).

É stato perciò possibile leggere dal PC i dati che la scheda Arduino scriveva sulla porta seriale, al fine di effettuare il rilevamento della posizione e di altre variabili fondamentali, sia durante le fasi di test e calibrazione, sia nella prove sperimentali di validazione. Data la portata del modulo Xbee Pro, per mantenere una telemetria affidabile si è scelto di eseguire l'esperimento in una zona di non oltre 100 metri di raggio.

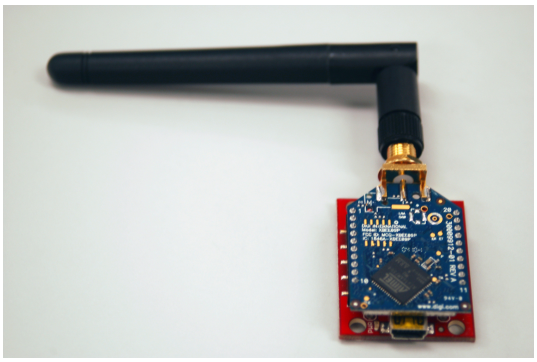

Figura 9: Modulo Xbee-Pro su Shield per PC

Si riporta di seguito una tabella che mostra i collegamenti dello Shield Xbee con Arduino.

Numero Pin Shield Xbee	Arduino 2009
1	//
2	Tx(Pin 1)
3	//
4	//
5	//
6	//
7	5 Volt
8	//
9	//
Gnd	Gnd
Vin	//

Tabella 3: Collegamenti Arduino Duemilanove - Xbee Shield

2 Legge di Guida

La legge di guida sviluppata e implementata per la navigazione è la *Beam Rider* [2]. Quest'ultima è solitamente usata dai missili per ingaggiare un obiettivo in movimento, cosicchè il raggio ('*beam*', in inglese) è nella pratica basato sul laser o sul radar. Questa tecnica di navigazione determina in base alla rotta da seguire e la distanza che si ha rispetto ad essa, la direzione di riferimento da inseguire per riportarsi in rotta. Nella figura seguente i punti O e B rappresentano due waypoints successivi:

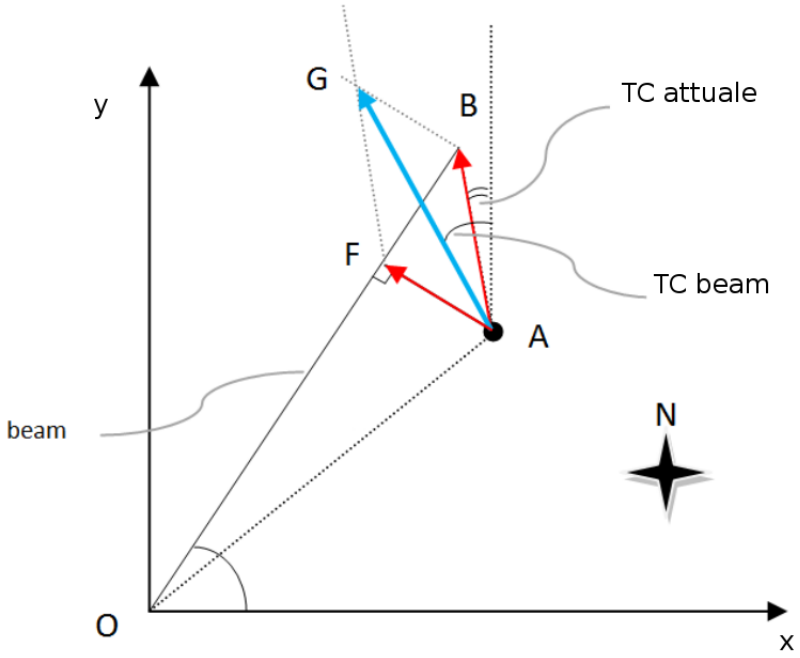

Figura 10: Sistema di guida Beam Rider

La formulazione matematica della legge di guida è la seguente:

$$\overline{AG} = \overline{AB} + K_2\overline{AF}$$

$$TC_{beam} = \arctan(\overline{AG})$$

Si può osservare che se ci si trovasse ad elevata distanza da un waypoint la correzione è praticamente trascurabile. Infatti si può facilmente intuire che $TC_{beam} \sim \arctan(\overline{AB})$ per elevati valori del rapporto $\overline{AB}/\ \overline{AF}$.

Per evitare questo si può considerare una correzione indipendente dalla distanza dal waypoint finale prendendone in considerazione il versore anzichè il vettore:

$$TC_{beam} = \arctan(\hat{AB} + K_2\overline{AF})$$

dove con $\hat{AB}$ si è indicato il versore tra il veicolo e il waypoint corrente.

In questo modo la correzione a parità di scostamento dalla rotta di riferimento è indipendente dalla distanza dal waypoint. Se vogliamo però che la correzione cresca avvicinandoci al waypoint e contemporaneamente che la correzione ad elevata distanza non sia trascurabile, si può suddividere la tratta in sottotratte di lunghezze minori ottenendo un comportamento che risulta essere preferibile data la possibilità di avere una correzione soddisfacente in entrambi i casi.

E' fondamentale la scelta di un raggio di tolleranza centrato nel waypoint: se la distanza dal waypoint corrente è inferiore a questa tolleranza, si passa al waypoint successivo. La definizione di questo raggio di tolleranza evita che l'auto entri in una traiettoria circolare attorno al waypoint: questo raggio di tolleranza deve essere perciò almeno maggiore del raggio minimo di sterzata dell'auto.

3 Modello Auto

3.1 Modello cinematico

Per simulare il comportamento dell'auto si è ricercato in letteratura un idoneo modello cinematico. Tra i vari modelli si è optato per il cosiddetto modello del *biciclo* o "*car-like*", il quale non considera le forze applicate all'auto, ma soltanto i vincoli di non scorrimento tra ruote e terreno e il vincolo di rigidità tra le ruote. Tale modello viene applicato anche a sistemi con più di due ruote e cinematicamente simili come è quello di un veicolo a quattro ruote. Le due ruote anteriori vengono fatte collassare in un unica ruota anteriore sita a metà dell'interasse anteriore e specularmente per quelle posteriori. I vincoli che determinano il moto del biciclo sulla ruota anteriore e posteriore sono quelli anolonomi del puro rotolamento.

$$\dot{x} = V\cos(\theta)$$

$$\dot{y} = V\sin(\theta)$$

$$\dot{\theta} = \frac{V}{l}\tan(\delta)$$

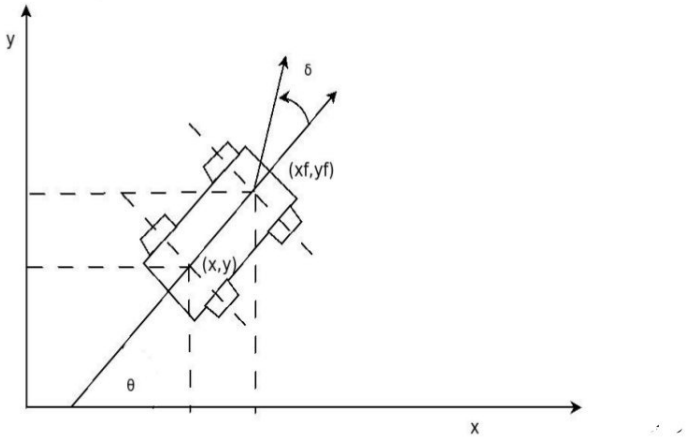

Figura 11: Modello Cinematico

Dove con x e y si intendono le coordinate del punto medio dell'interasse posteriore, con θ la direzione rispetto all'asse x e con l la distanza tra le ruote

anteriori e posteriori. Come si può osservare gli ingressi del nostro modello sono l'angolo di sterzo δ, che modifica quindi θ, e la velocità dell'auto V che corrisponde a quella della ruota posteriore.

Anche se le ipotesi fondamentali prevedono che i raggi di sterzata siano ampi si sono ottenuti comunque dei risultati validi sia in simulazione che nelle prove sperimentali.

3.2 Modellazione dinamica

La dinamica della velocità si è modellata come un sistema del primo ordine, dove per ingresso si ha il comando che viene inviato all'ESC e come uscita misurata il numero di giri al secondo all'albero.

Inizialmente si è operato un test su un banco prova, dando in ingresso un gradino di ampiezza nota; successivamente si sono stimati il guadagno statico e il polo, sapendo che a circa 3τ si ha il 95% del valore a regime (vedi [3]):

$$G(s) \doteq \frac{Y(s)}{U(s)} = \frac{Vel_{mis}(s)}{Cmd(s)} = \frac{Kp}{(1+s\tau)} = \frac{240}{(1+0.5s)}$$

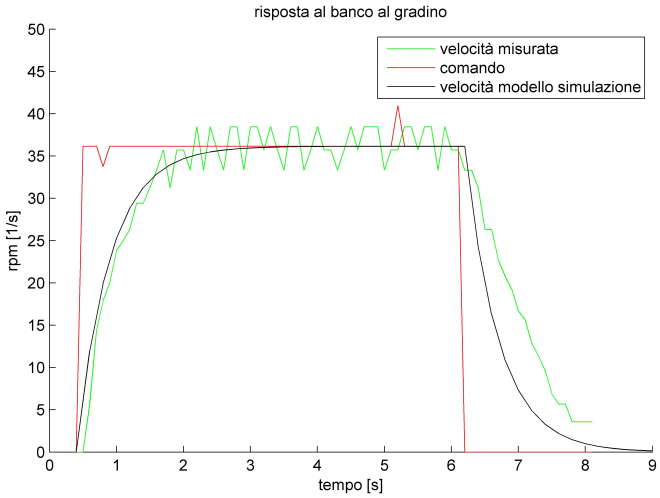

Figura 12: Confronto tra la risposta reale e simulata al banco

Per ottenere un miglior modello dell'auto, si è stimata la funzione di trasferimento come si era fatto per la prova al banco, sfruttando i dati raccolti durante una prova con l'auto su strada.

La funzione di trasferimento stimata è:

$$G(s) \doteq \frac{Y(s)}{U(s)} = \frac{Vel_{mis}(s)}{Cmd(s)} = \frac{Kp}{(1+s\tau)} = \frac{130}{(1+2.1s)}$$

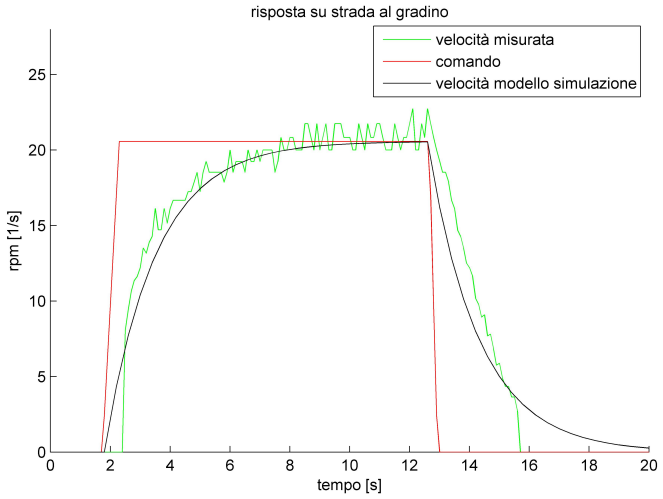

Figura 13: Confronto tra la risposta reale e simulata su strada

4 Autopiloti

4.1 Autopilota di direzione

Per la sintesi dell'autopilota di direzione si è scelto un controllo di tipo proporzionale. Si genera una correzione proporzionale, secondo una costante$K_{direzione}$ tarata sperimentalmente, all'errore tra rotta di riferimento TC_{beam} e quella misurata dal GPS $TC_{misurata}$. Il nostro angolo di sterzo comandato $\delta_{comandato}$sarà quindi:

$$\delta_{comandato} = K_{direzione}\,(TC_{beam} - TC_{misurata})$$

Per una più completa analisi si riporta di seguito lo schema Simulink che gestisce il controllo di direzione.

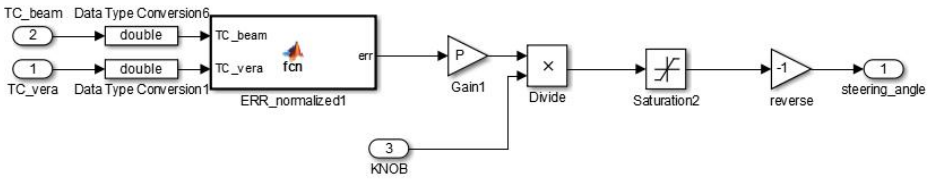

Figura 14: Schema Simulink del controllo di direzione

4.2 Autopilota di velocità

Per la sintesi dell'autopilota di velocità si è scelto un controllore di tipo PI (proporzionale e integrale). L'errore $e(t)$ che comanda il PI è la differenza tra il numero di giri all'albero comandati e numero di giri misurati dal sensore magnetico. Il numero di giri all'albero comandati, si calcola moltiplicando la velocità di riferimento per una costante di conversione. Si genera poi un comando per l'ESC tramite il PI, proporzionale all'errore di velocità in giri al secondo all'albero secondo la costante Kp, e proporzionale al suo integrale secondo la costante Ki. La presenza di un temine integrale ci garantisce che l'errore a regime sia nullo nel caso di ingressi costanti [1] mentre la presenza di un termine proporzionale ci assicura una maggiore rapidità nell'accelerazione:

$$Cmd(t) = Kp\,e(t) + Ki \int_{t_0}^{t} e(\tau)d\tau$$

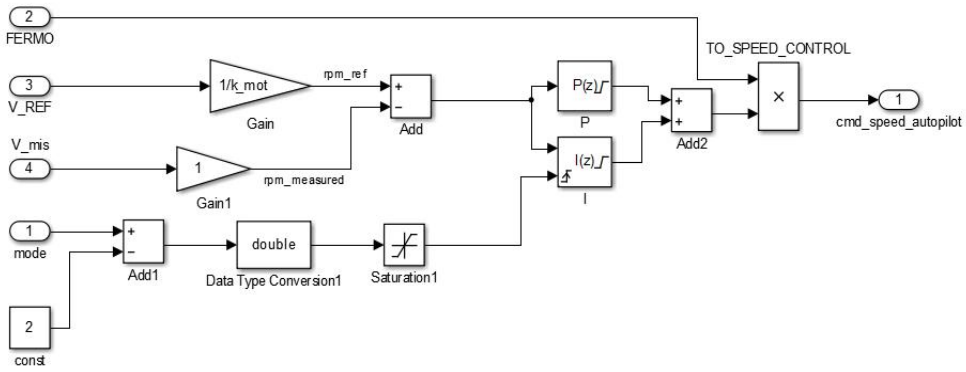

Figura 15: Schema Simulink del controllo di velocità

Per tenere conto del fenomeno della saturazione del comando (*windup*) e per resettare l'integratore quando la modalità autopilota di velocità è disabilitata, si sono utilizzati i blocchi Simulink.

Per completezza si riporta di seguito lo schema Simulink relativo al controllo di velocità.

Per la taratura delle variabili di controllo Kp e Ki si rimanda al Capitolo 7, paragrafo 1.

5 Logica della Ground Control Station

5.1 Logica della GCS per la taratura sperimentale dell'autopilota di velocità

Per la calibrazione dei parametri dell'autopilota di velocità si è settata la trasmittente in modo da avere uno switch a tre stati ai quali corrispondono tre modi di funzionamento:

1. modo1= tutto disabilitato;
2. modo2=radiocomando abilitato per il controllo completo del veicolo (direzione e velocità);
3. modo3=radiocomando abilitato per la sola direzione e controllo automatico per la velocità;

Dato il limitato numero di canali leggibili dalla ricevente cioè 5, si è pensato ad una gestione flessibile della stessa. Lo stick destro funge da comando di

manetta se in modo 2, da comando di una velocità di riferimento quando entriamo in modalità 3; in questo modo, grazie alla possibilità di generare codice in automatico da Simulink, è possibile utilizzare lo stesso hardware interpretandolo in modo intelligente.

Lo stick sinistro è stato invece settato come comando di sterzo.

I restanti canali sono stati utilizzati per la variazione dalla trasmittente dei guadagni per il proporzionale e l'integrale.

Tramite il modulo Xbee Pro venivano stampati in tempo reale su di un grafico tramite il software "SerialChart" la velocità di riferimento e quella reale.[3] In tal modo si è ottenuta una conoscenza immediata delle prestazioni del controllore PI, evitando di rientrare in laboratorio per l'analisi dei dati.

5.2 Logica della GCS per la validazione della legge di guida

Nella prova sperimentale finale si è utilizzato lo stesso numero di canali della ricevente.

La logica dello switch a due stati è:

1. modo 1: tutte le uscite sono disabilitate (Safety: Off).
2. modo 2: le uscite sono abilitate (Safety: Ok).

Si è sfruttato lo stesso switch a tre stati della fase di taratura, con una logica differente:

1. modo 1: solo ricevente abilitata;
2. modo 2: ricevente abilitata i cui comandi si sommano a quelli generati dal controllo automatico di direzione;
3. modo 3: ricevente abilitata i cui comandi si sommano a quelli generati dal controllo completamente autonomo del veicolo.

Per quanto riguarda i primi due canali si sono lasciati come nella logica per la calibrazione, essi comandano rispettivamente il gas e la sterzata.

Rimanendo un canale libero, lo si è scelto per controllare il guadagno del termine proporzionale alla correzione della legge di guida ossia K_2.

In questo modo è stato possibile variare il guadagno in tempo reale, non solo in simulazione ma anche nell'esperimento finale.

Si fa notare che l'informazione di status GPS viene utilizzata per interrompere la navigazione autonoma nel caso in cui il segnale non sia valido.

[3] Per ulteriori approfondimenti si rimanda alla pagina web :http://code.google.com/p/serialchart/

6 Simulazioni

6.1 Taratura dei guadagni della legge di guida

Per un calcolo preliminare del guadagno K_2 della legge di guida e per una stima di prima approssimazione si è simulato il percorso e si sono plottate le traiettorie in simulazione. Come guadagno minimo si è preso il valore zero, che corrisponde ad una guida waypoint di tipo *Pure Pursuit*. [4] Come guadagno massimo ci si è fermati a 10. Si riportano di seguito le figure più signficative delle simulazioni.

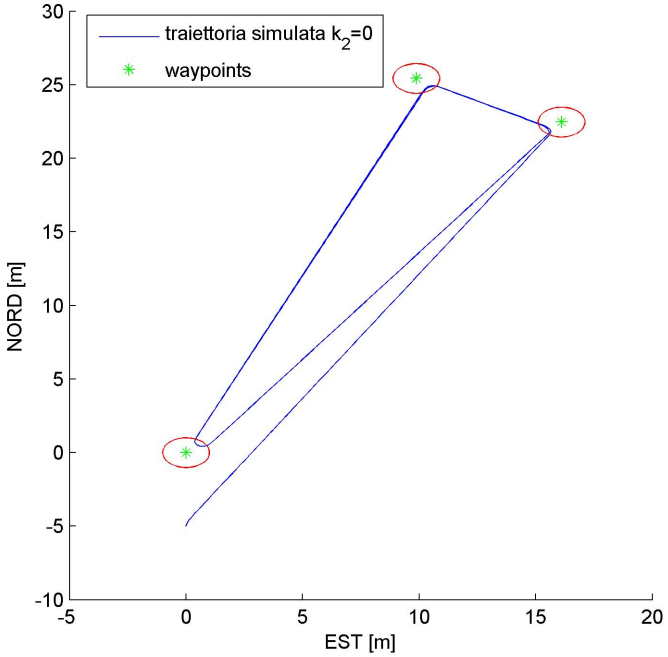

Figura 16: Simulazione con K_2=0

[4] Con il termine *Pure Pursuit* si fa riferimento ad una tecnica di inseguimento usata in combattimenti aerei in cui un aeromobile insegue un altro aeromobile puntando il proprio muso direttamente verso di esso.

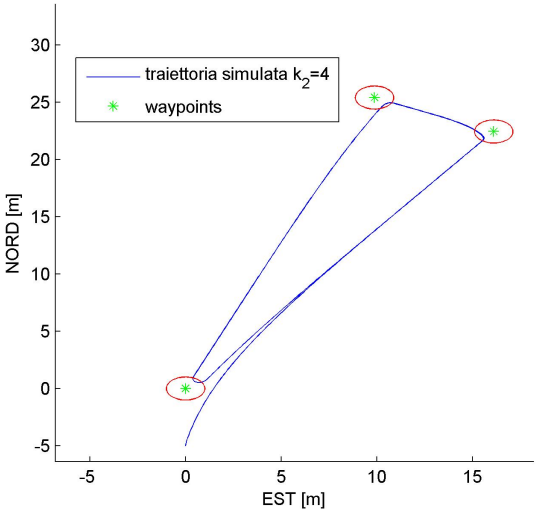

Figura 17: Simulazione con K_2=4

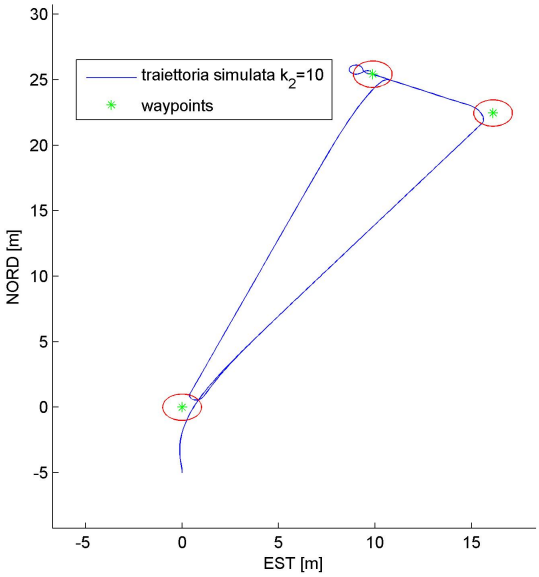

Figura 18: Simulazione con K_2=10

Come si può osservare perK_2=0 si ha che l'auto non si riporta in rotta, ma punta soltanto il waypoint, mentre col crescere del guadagno le prestazioni migliorano notevolmente, riuscendo a rientrare in rotta rapidamente.

6.2 Simulazione per la validazione della logica della ricevente

Per la validazione della logica della ricevente si sono dapprima simulati gli ingressi e gli switch della Trasmittente in Simulink e successivamente tramite il Tool generate code, si è esportato il codice in C++ e lo si è compilato dall'IDE di Arduino.

A un banco prova si è testata l'effettiva validità della logica della ricevente sia per la prova di taratura che per l'esperimento finale.

Nelle immagini sottostanti sono riportati gli schemi che simulano la ricevente.

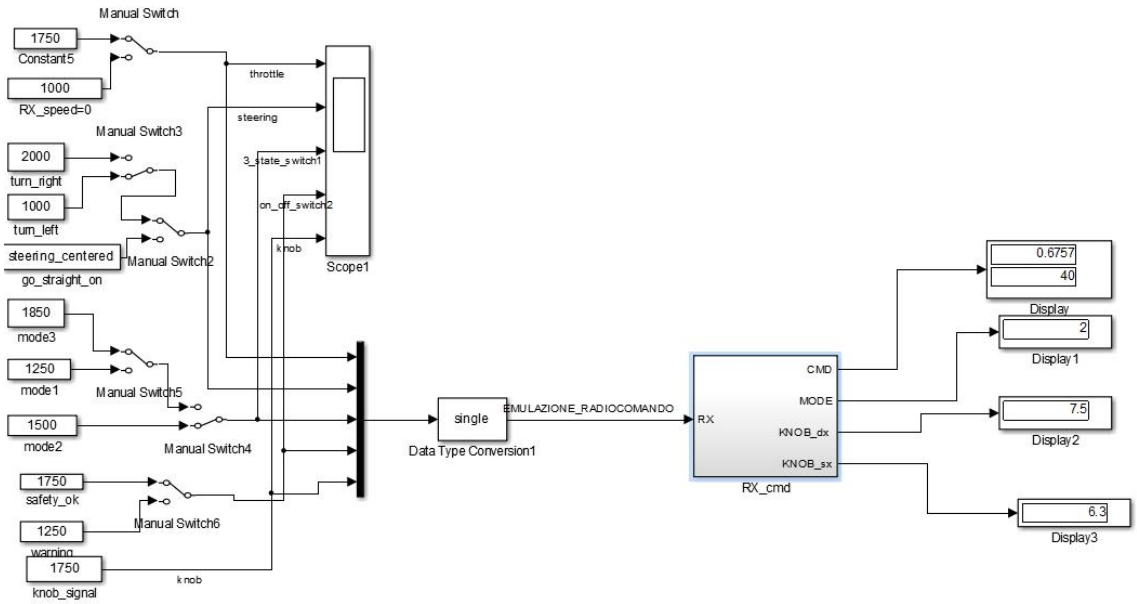

Figura 19: Schema Simulink della Ricevente - taratura sperimentale dei guadagni

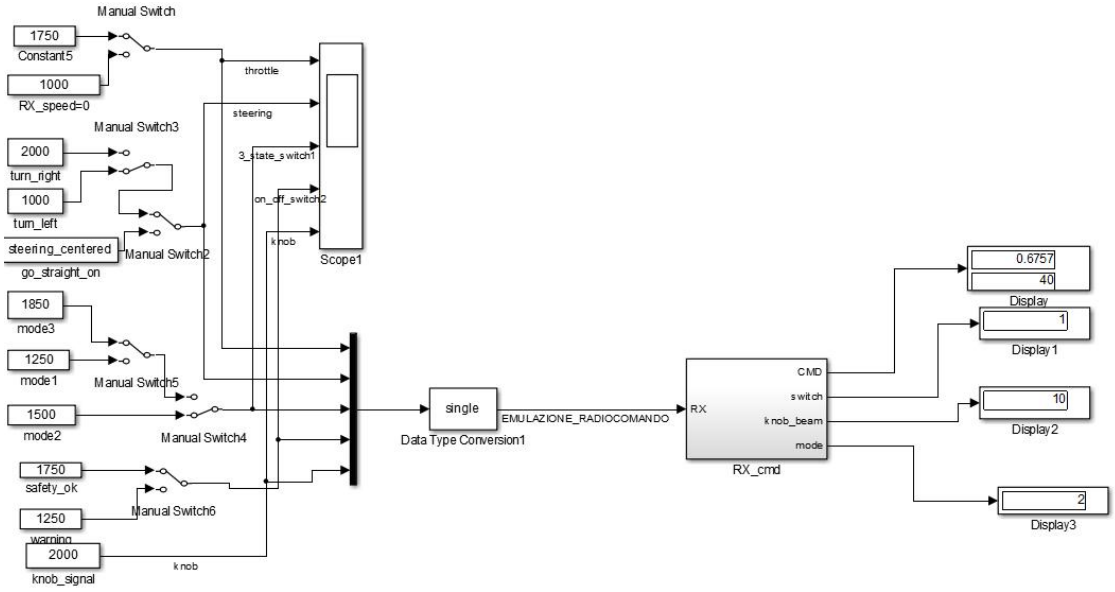

Figura 20: Schema Simulink della Ricevente - Esperimento Finale

6.3 Simulazione dell'esperimento finale

Si riporta lo schema Simulink utilizzato per la simulazione della prova finale.

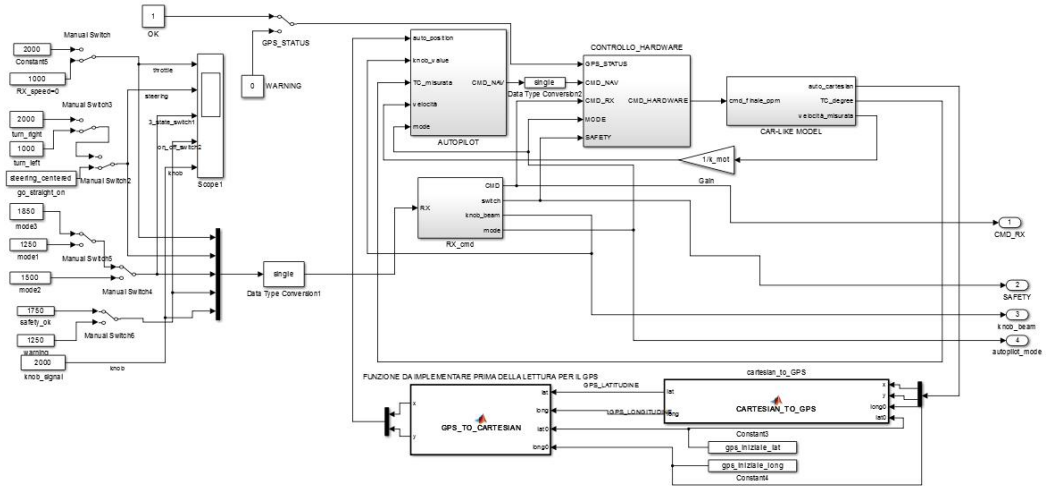

Figura 21: Schema Simulink per l'esperimento finale

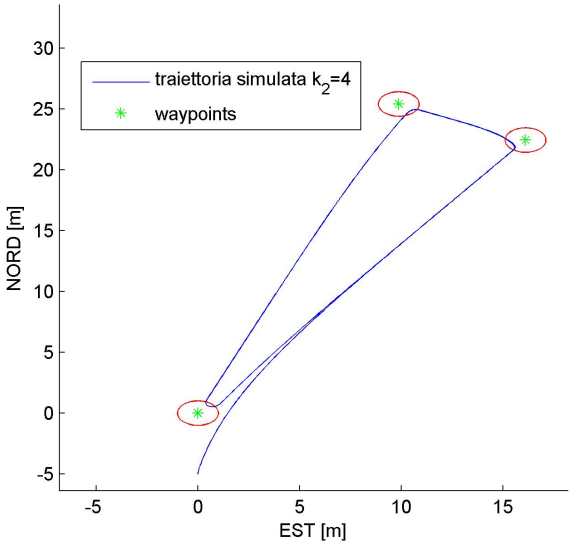

Figura 22: Traiettoria simulata- Esperimento finale

Per inizializzare tutti i parametri fondamentali del modello, quali le costanti dei controllori automatici, i waypoint, i parametri della trasmittente, delle velocità da mantenere nelle singole tratte, si utilizza un file Matlab *inizializzazione_parametri.m*. In questo modo cambiando waypoints e parametri, si ha un controllo più agevole della simulazione.

Si riportano le grandezze fondamentali nella simulazione finale con K_2=5.

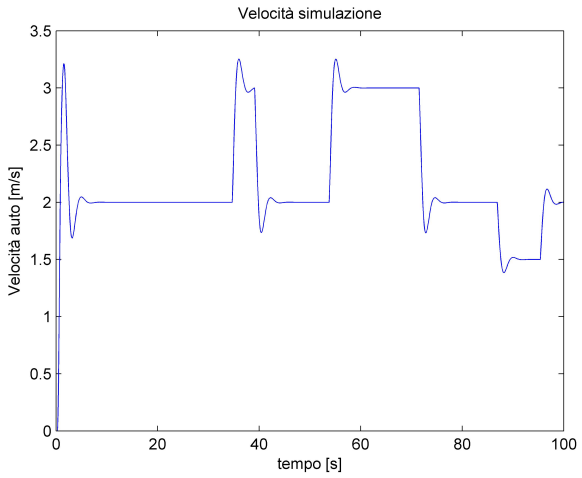

Figura 23: Velocità auto in simulazione - Esperimento finale

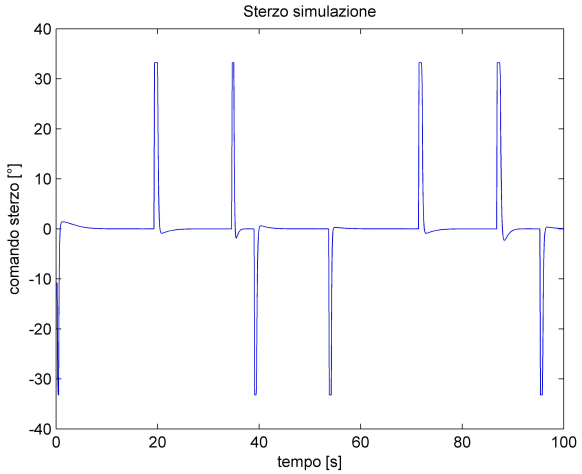

Figura 24: Comando sterzo in simulazione - Esperimento finale

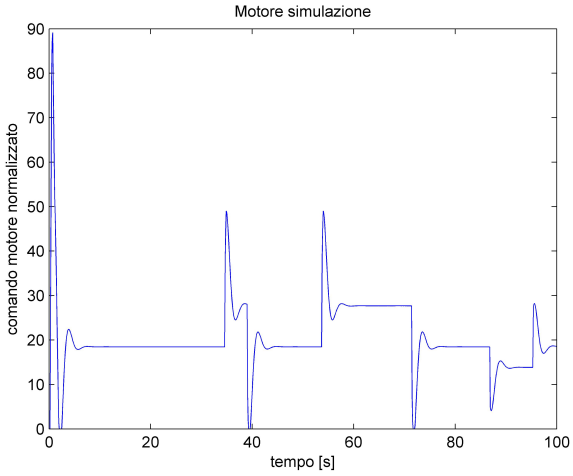

Figura 25: Comando motore in simulazione - Esperimento finale

7 Analisi dei dati sperimentali

7.1 Taratura dei guadagni dell'autopilota di velocità

Per la taratura dell'autopilota di velocità si è impiegata la logica della ricevente descritta nel paragrafo 5.1.

Per una rapida acquisizione e consapevolezza dei dati in telemetria e dei riferimenti comandati dalla ricevente, si è utilizzato il software *Serial Chart* configurato per la stampa di tre valori di velocità:

1. Velocità di riferimento in m/s;
2. Velocità misurata dal sensore magnetico convertita direttamente tramite una costante in m/s;
3. Velocità gps in m/s.

Dall'analisi in tempo reale della risposta alle velocità di riferimento si è cercato di tarare il PI in modo che il controllo risultasse robusto: si è posta l'attenzione sull'evitamento di sovraelongazioni, vibrazioni e comportamenti oscillatori, e contemporaneamente si è cercato di limitare i tempi di salita.

Questa logica è stata dapprima simulata e in seguito si è poi opportunamente adattata per la compilazione e il caricamento su Arduino. Si riporta l'immagine dello Schema simulink per la generazione del codice su Arduino.

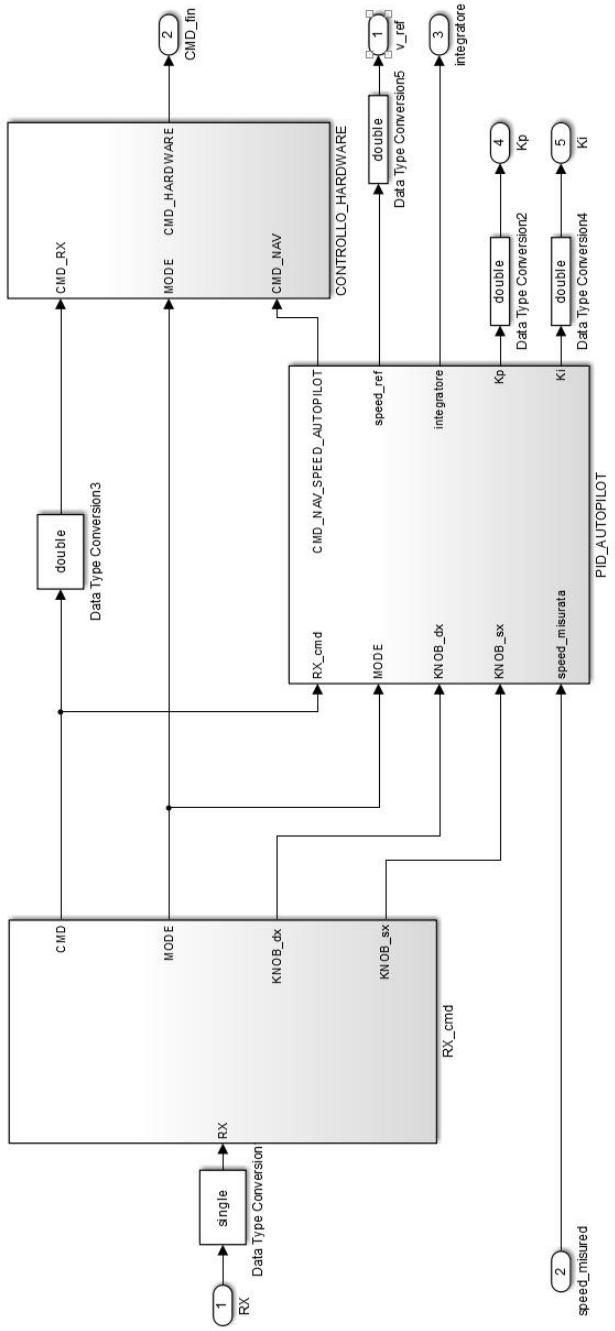

Figura 26: Schema Simulink - Taratura PI velocità

Non avendo ottenuto risultati soddisfacenti dalla prima prova sperimentale, si è provveduto alla taratura in ambiente Simulink del guadagno proporzionale e del guadagno integrale.

I valori trovati per il PI al banco sono:

P=0.006

I=0.0225

In seguito si è modificato l'autopilota e si è costatato che i risultati in simulazione corrispondevano con buona approssimazione a quelli ottenuti sperimentalmente.

Si riportano gli andamenti della risposta simulata e di quella reale con l'inserimento del PID al banco.

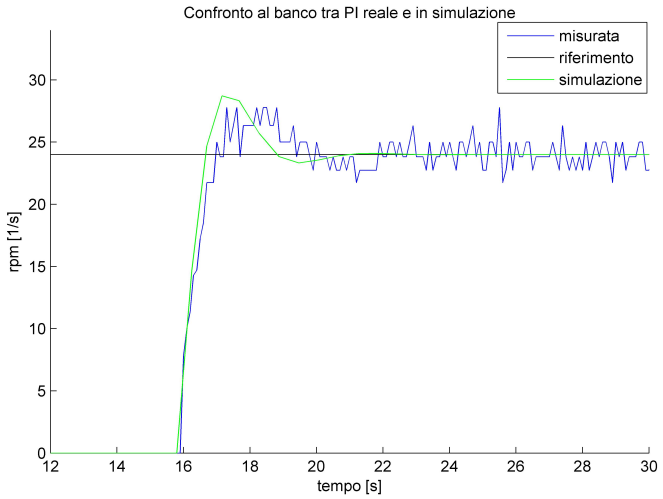

Figura 27: Confronto tra risposta del PI al banco e in simulazione

Successivamente si sono tarati i guadagni per la prova su strada seguendo lo stesso procedimento, e come si vede nel Capitolo 9, portano ad un buon inseguimento del riferimento di velocità.

In questo caso i valori per il PI sono:

P=0.02

I=0.045

7.2 Dati reali GPS

Si riportano le immagini relative alla posizione rilevata dal GPS ad auto ferma.

Si può osservare come l'errore di posizione si mantiene contenuto nel raggio di 1 metro per intervalli di tempo di circa 1 minuto.

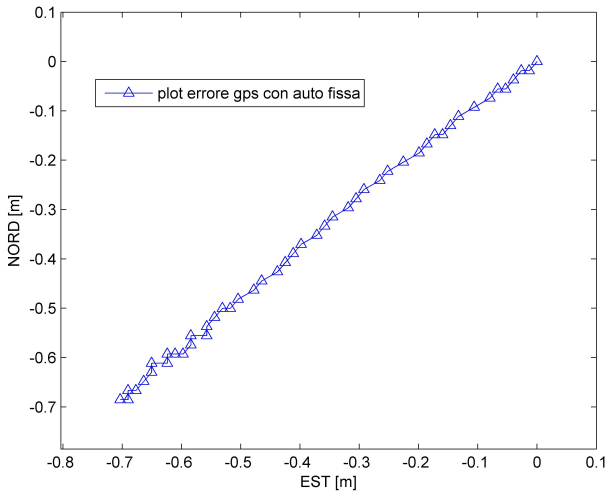

Figura 28: Errore GPS con auto ferma

La misura dell'incertezza nella posizione è stata fondamentale per definire anche i limiti dell'esperimento in termini di spazio tra un waypoint e il successivo, ed eventuali ostacoli.

Si riportano le immagini nelle quali si confrontano le velocità stimate dal GPS e dal sensore di velocità.

Il sensore misura una velocità in giri al secondo, convertiti in metri al secondo tramite una costante.

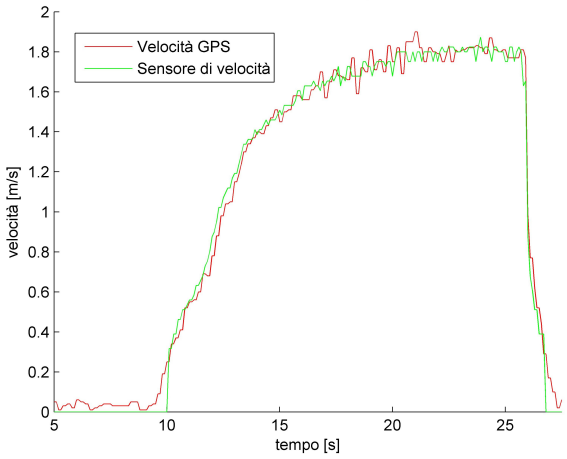

Figura 29: Confronto tra GPS e sensore di velocità

8 Schema Simulink per il test finale

Per il test finale si è utilizzato lo stesso schema Simulink utilizzato in simulazione, andando ad escludere il blocco Simulink "Car-Like model" sostituito dal nostro veicolo, ed andando a definire gli ingressi per quello che è il nostro schema Simulink, cioè un sistema di navigazione e controllori automatici, che implementa una logica di sicurezza ed è in grado di comunicare con una stazione di terra, interfacciandosi con i sensori e gli attuatori esterni.

Col Tool generate code di Simulink si è generato il codice finale da caricare su Arduino. Si riporta lo schema Simulink impiegato per la generazione automatica del codice.

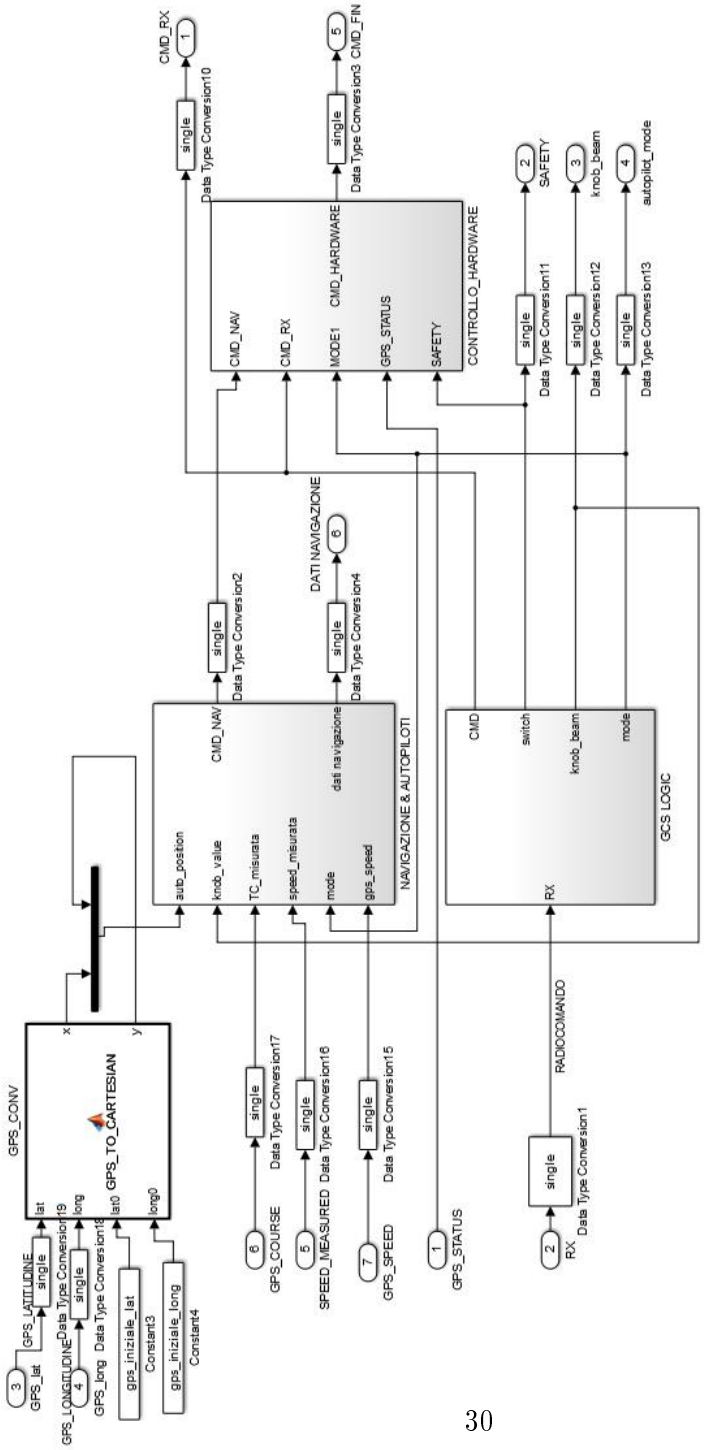

Figura 30: Schema Simulink per la generazione di codice C++

9 Telemetria e confronto con i dati simulati

Sono stati effettuati diversi test durante i quali si è proceduto alla raccolta dei dati in telemetria.

In questo capitolo si riportano i grafici dei dati più significativi.

Come si evince dalla figura 31, l'auto riesce effettivamente a raggiungere i waypoints ma in modo non certo ottimale. Le notevoli oscillazioni sono dovute a una errata taratura dell'autopilota di direzione, in altri termini la costante di guadagno dell'autopilota si presentava troppo elevata. Data l'instabilità dell'auto si è deciso quindi di terminare l'esperimento.

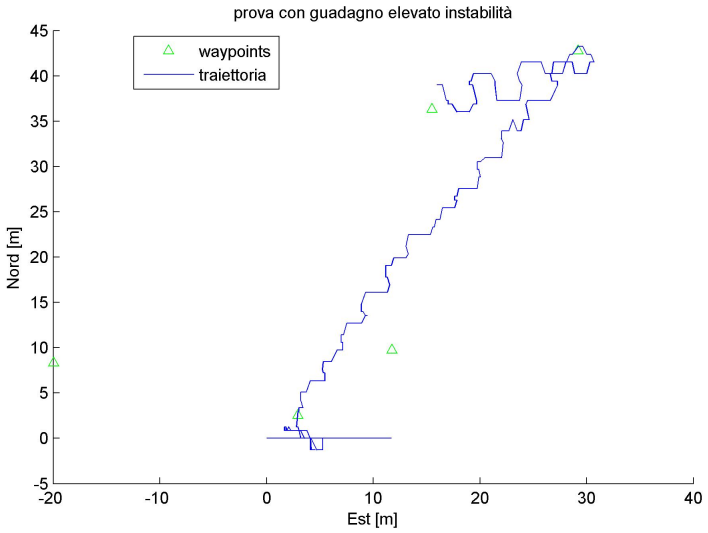

Figura 31: Esperimento 1

Si riporta per completezza un test dove la costante risultava ancora troppo elevata, anche se non come nel test precedente , con i waypoints disposti in maniera diversa.($K_{direzione}$=10)

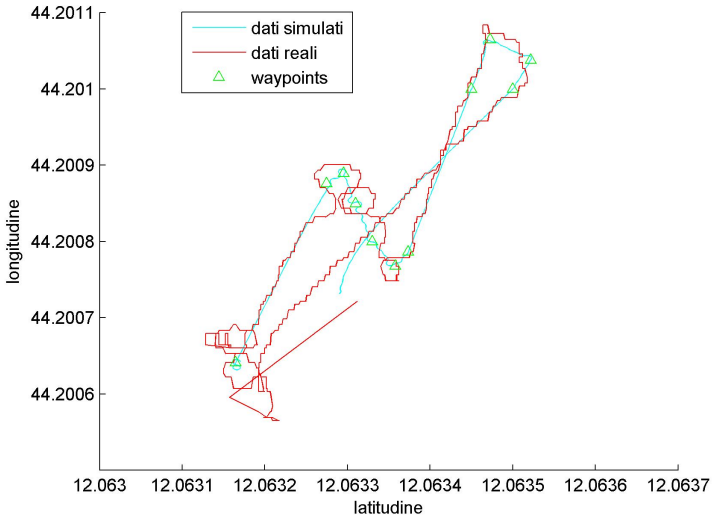

Figura 32: Esperimento 2

Dopo aver modificato l'autopilota, l'auto riesce a percorrere la traiettoria in modo soddisfacente. Si riportano i dati più significativi raccolti in telemetria durante l'esperimento finale($K_{direzione}$=0.6).

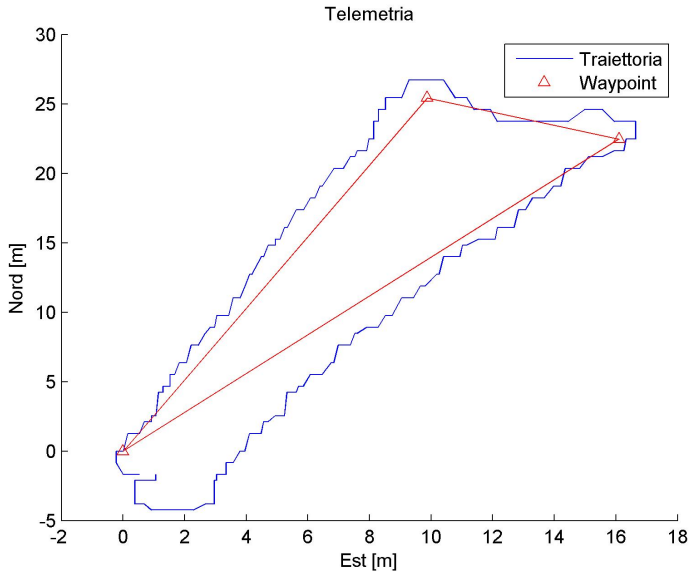

Figura 33: Esperimento finale

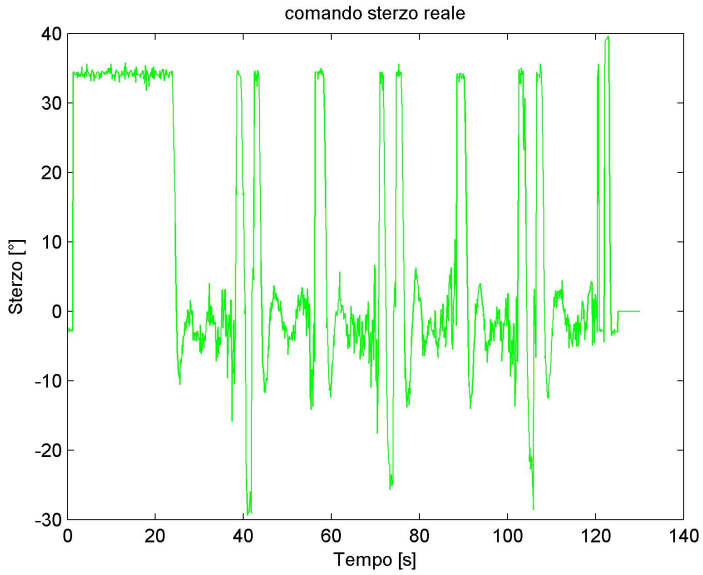

Figura 34: Comando di sterzo reale

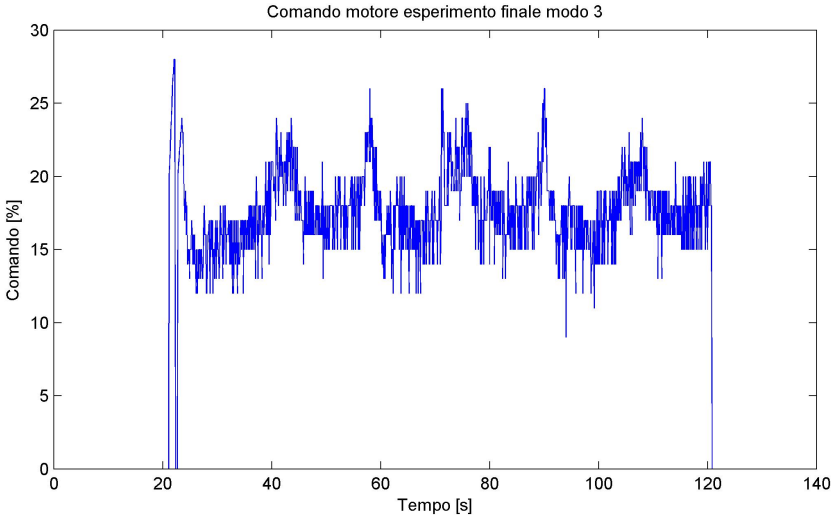

Figura 35: Comando Motore Reale

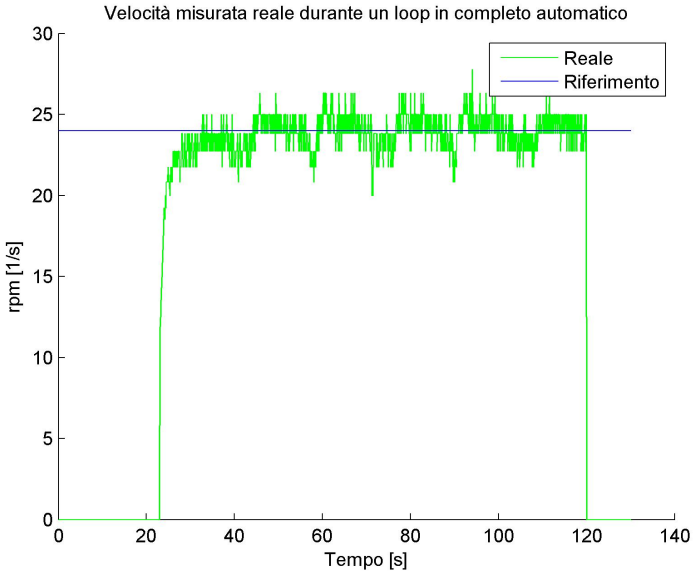

Figura 36: Velocità misurata dal sensore magnetico

Si riporta infine il confronto traiettoria reale-traiettoria simulata per lo stesso percorso. Si nota come qualitativamente i percorsi siano simili.

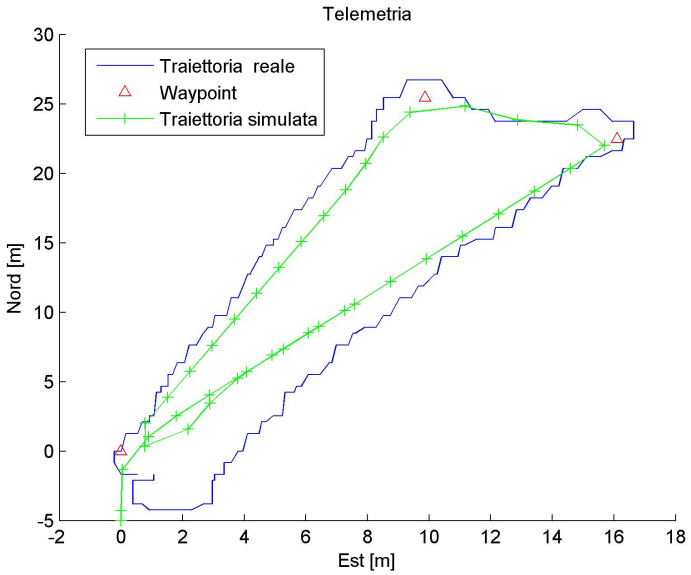

Figura 37: Confronto traiettoria reale e simulata

10 Conclusioni e Sviluppi futuri

Dal confronto tra le simulazioni e i dati sperimentali, si può affermare che lo sviluppo della legge di guida e l'implementazione delle logiche sono stati portati a termine con successo relativamente alla precisione dei sensori e che la stessa legge di guida può essere effettivamente utilizzata per la guida autonoma di un veicolo terrestre.

Data la modularità del modello realizzato in ambiente Simulink e la scelta di realizzare blocchi indipendenti, è possibile riciclarne la maggior parte limitando il lavoro per adattarlo ad altri veicoli o velivoli.

Si è infatti data molta importanza in fase preliminare e di progettazione dello schema Simulink alla modularità, riciclabilità e autonomia dei singoli blocchi. Quest'ultima ha permesso di verificare il funzionamento di ogni blocco singolarmente facendo emergere rapidamente eventuali bugs.

Dai dati raccolti in telemetria si può osservare come le prestazioni nell'inseguimento della traiettoria migliorino con l'affinarsi della taratura dell'autopilota di direzione, raggiungendo un comportamento sempre più simile a quello in simulazione. Per tale affinamento si rimanda a sviluppi futuri.

Riferimenti bibliografici

[1] http://www.maartenlamers.com/nmea/

[2] De Angelis E.L.,(2009). *Simulazioni software ed hardware-in-the-loop di un velivolo non abitato per monitoraggio ambientale*, Tesi di Laurea Specialistica, Università di Bologna, Forlì, 2009

[3] G. Marro, Controlli Automatici , 5a edizione, Zanichelli, Bologna, 2004.