Corso di Laurea Magistrale in Ingegneria Meccanica
A.a. 2026/2027
Sessione di Laurea: Settembre 2026



Sviluppo di logiche di controllo e modelli per veicoli in scala

Relatori:

Alessandro Vigliani

Angelo Domenico Vella

Candidato:

Alex d'Alfonso

Indice

Abstract **3**

1. Introduzione **4**

1.1 Stato dell'arte e motivazioni 4

1.2 Scopo e struttura della tesi 6

2. Architettura Hardware **7**

2.1 Sistema di Propulsione: motore elettrico e batteria 8

2.2 Attuazione e Controllo: servosterzo e controller motore 10

2.3 Sistema di Localizzazione: Unità GPS-IMU 13

2.4 Architettura Speedgoat 15

3. Tecnica di controllo tramite Input manuale **18**

3.1 Controller: Logitech F710 Wireless 18

3.2 Protocollo di comunicazione Wireless UDP 21

3.3 Integrazione interfaccia I/O con l'Hardware 25

4. Tecnica di controllo ad anello chiuso **34**

4.1 Localizzazione e stima di stato tramite Filtro di Kalman Esteso 35

4.2 Ottimizzazione della matrice di covarianza 37

4.3 Determinazione della funzione di costo dell'ottimizzazione 42

4.4 Determinazione punto di riferimento e calcolo errori 43

5. Conclusioni e applicazioni future **45**

5.1 Analisi dei risultati 45

5.2 Considerazioni conclusive 45

5.3 Possibili Sviluppi e applicazioni future 46

Lista delle figure **47**

Lista delle tabelle **47**

Bibliografia **48**

Appendice A **50**

Abstract

Questa tesi affronta lo sviluppo di modelli e logiche di controllo per veicoli in scala, ossia piattaforme sperimentali che riproducono un veicolo reale mantenendone le proporzioni e le principali caratteristiche dinamiche, ma in dimensioni ridotte. Tale categoria di veicoli ha assunto un ruolo sempre più rilevante in ambito accademico e industriale, poiché consente di effettuare test in modo più agevole, riducendo sia gli spazi necessari sia i costi di gestione e manutenzione senza rinunciare alla rappresentatività del comportamento del veicolo reale.

Nel lavoro vengono presentate due differenti modalità di controllo. La prima prevede la guida manuale tramite joystick, sfruttando le due levette analogiche per comandare velocità e angolo di sterzo in tempo reale. La seconda introduce invece una strategia di controllo autonomo ad anello chiuso: grazie alla presenza a bordo di un'unità IMU-GPS e all'implementazione di un filtro di Kalman opportunamente configurato, il veicolo è in grado di seguire autonomamente una traiettoria di riferimento. In particolare, a ogni istante vengono calcolate la distanza del veicolo dal percorso desiderato e l'orientamento rispetto al punto di riferimento più vicino; tali informazioni permettono di intervenire sugli errori di cross-track e di heading al fine di minimizzare lo scostamento dalla traiettoria pianificata. Per fare ciò è stato svolto un processo di ottimizzazione delle matrici di covarianza del filtro di Kalman, indispensabile per garantire stime accurate dello stato e quindi migliorare le prestazioni complessive del controllo autonomo. L'obiettivo del lavoro è la progettazione e l'implementazione di un'architettura di controllo per veicoli in scala, facendo uso di modelli, sensori di bordo e algoritmi di stima per migliorare precisione, stabilità e riproducibilità dei test sperimentali.

1. Introduzione

1.1 Stato dell'arte e motivazioni

Negli ultimi decenni, lo sviluppo di veicoli a guida autonoma e dei sistemi avanzati di assistenza alla guida ha assunto un ruolo di primissimo piano sia nella ricerca accademica che nell'ambito dell'industria automobilistica [1], [2]. Infatti, per implementare sistemi di controllo per la guida autonoma è necessario integrare algoritmi complessi, una stima precisa della posizione e una buona conoscenza della dinamica del veicolo [3], con l'obiettivo di garantire elevati standard di sicurezza, efficienza e stabilità.

In tale contesto, i veicoli in scala hanno acquisito un ruolo sempre più rilevante, in quanto rappresentano una soluzione estremamente efficace per studiare sistemi complessi senza ricorrere a veicoli a grandezza naturale [4], [5]. I vantaggi offerti da queste piattaforme sono notevoli:

- Maggiore ripetibilità delle prove: le dimensioni ridotte facilitano il controllo dell'ambiente circostante, che viene configurato in base alle simulazioni che si intendono svolgere, riducendone i tempi. Tutto questo aumenta la velocità delle simulazioni e dell'intero iter che le precede, garantendo rapidità e ripetibilità. [6]
- Minore rischio: la possibilità di operare in spazi contenuti contribuisce a minimizzare i rischi associati a malfunzionamenti o errori umani
- Guadagno economico: la minore complessità delle prove sperimentali consente di diminuire i costi relativi all'attività di ricerca. Inoltre, anche eventuali danni ai componenti del veicolo risultano meno onerosi rispetto a quelli che si verificherebbero su un veicolo reale. [7]

A questi aspetti si aggiunge il crescente interesse verso le architetture di controllo per la guida autonoma. Un veicolo autonomo condivide infatti numerose caratteristiche con i veicoli in scala, non soltanto dal punto di vista geometrico e cinematico, ma anche relativamente alla dinamica del sistema, all'interazione con la sensoristica installata a bordo, come per esempio IMU, GPS o RADAR [8], e all'integrazione con sistemi di calcolo reale per la determinazione di una traiettoria da seguire [9].

Proprio su questo tema, quello dell'inseguimento di una traiettoria, la letteratura propone numerosi approcci differenti: dai più classici, come l'algoritmo di Pure Pursuit proposto da

Coulter e MacAdams [10], ad approcci più innovativi basati sul controllo predittivo, come quelli discussi da Falcone [11] e da Ji [3].

L'efficacia di queste tecniche di controllo dipende in larga misura da una determinazione accurata della posizione della piattaforma. A questo scopo, il filtro di Kalman esteso rappresenta uno strumento particolarmente efficace, poiché consente di ottenere stime affidabili anche in presenza di rumore, come dimostrato da Kia [12]. Tuttavia, la definizione delle matrici di covarianza risulta spesso complessa; per questo motivo, si ricorre frequentemente ad algoritmi genetici per ottimizzarne i parametri e ridurre al minimo gli errori di stima, secondo l'approccio proposto da Loebis [13].
Alla luce di quanto esposto, i veicoli in scala si confermano dunque una risorsa preziosa per la ricerca e l'innovazione nel campo della guida autonoma: piattaforme sperimentali flessibili, economiche e sicure, capaci di replicare in modo fedele il comportamento dei veicoli reali [4].

1.2 Scopo e struttura della tesi

Il presente lavoro di tesi si inserisce in questo ambito di ricerca, e riguarda lo sviluppo e l'implementazione di logiche di controllo su una piattaforma in scala 1:5 (Losi 5ive-T 2.0).
La tesi è organizzata in capitoli secondo la seguente struttura:

2. **Architettura hardware**: in questo capitolo vengono descritte in dettaglio tutte le componenti hardware del Losi 5ive-T 2.0, illustrandone le caratteristiche principali e il ruolo di ciascuna all'interno delle sperimentazioni e dei test.
3. **Logica di controllo tramite Joystick**: in questo capitolo viene illustrato l'intero processo necessario per ottenere un controllo completo e preciso del veicolo tramite joystick. Viene spiegato come Simulink acquisisce i segnali provenienti dal joystick e come essi sono elaborati per determinare la velocità del veicolo e l'angolo di sterzo. Vengono descritte, inoltre, le varie scelte progettuali e le strategie implementate, per garantire una risposta stabile e precisa agli input.
4. **Logica di controllo tramite Closed-Loop**: in questo capitolo viene fornita un'introduzione al controllo ad anello chiuso (Closed-Loop) e viene spiegato il funzionamento del filtro di Kalman esteso. Successivamente, viene descritta nel dettaglio la procedura di ottimizzazione della matrice di covarianza del filtro di Kalman, finalizzata a minimizzare l'errore di posizione tra la posizione stimata del veicolo e un punto di riferimento lungo la traiettoria desiderata. Infine, viene illustrato come viene scelto il punto di riferimento lungo la traiettoria, punto necessario alla determinazione dei comandi di sterzo e velocità da applicare al Losi per permettergli di seguire il percorso.
5. **Conclusione:** vengono forniti i risultati di questa attività di tesi, oltre a considerazioni conclusive sulle due logiche di controllo e a possibili spunti, integrazioni e applicazioni future

2. Architettura hardware

Il veicolo selezionato per lo svolgimento di questa tesi è il Losi 5ive-T 2.0 (*Figura 1*), un modello in scala 1:5 che rappresenta una piattaforma particolarmente adatta per attività sperimentali avanzate. Il Losi combina una struttura robusta con una geometria fedele a quella di un fuoristrada reale, permettendo di replicare con buona accuratezza fenomeni dinamici. Per soddisfare le esigenze specifiche del lavoro svolto il veicolo è stato opportunamente modificato rispetto alla sua configurazione originale, così da renderlo più adatto alle esigenze di sperimentazione.

Inizialmente il veicolo era equipaggiato con un motore termico, accompagnato da un serbatoio per la benzina e da un sistema di trasmissione meccanica. Questa configurazione non risultava idonea agli obiettivi della tesi, che richiedevano un livello di controllo più elevato e una maggiore affidabilità. Per questo, tutte le componenti legate alla propulsione sono state rimosse e sostituite da un motore elettrico e una batteria, soluzioni che garantiscono una gestione più semplice dei parametri di funzionamento e una riduzione significativa delle variabili. Nelle pagine seguenti verranno presentate tutte le componenti hardware presenti sul veicolo oltre alle specifiche tecniche, fornendo anche le motivazioni che hanno guidato la scelta.



Figura 1: Configurazione iniziale del Losi 5ive-T 2.0

2.1 Sistema di propulsione: motore elettrico e batteria

Il motore elettrico selezionato per la nuova configurazione del veicolo è un RS PRO Brushless, motore trifase progettato per garantire prestazioni affidabili e compatibili con le esigenze delle prove sperimentali previste in questa attività. Si tratta di un'unità caratterizzata da una velocità nominale di 4000 RPM, tensione di alimentazione di 24 V. Dal punto di vista meccanico, il motore presenta coppia nominale di 0.25 Nm, con una coppia di picco che può raggiungere 0.75 Nm, che permette al sistema di gestire anche condizioni che richiedono più potenza erogata. Ulteriori informazioni tecniche e specifiche più dettagliate riguardo il motore sono riportate nella documentazione tecnica dedicata [14].

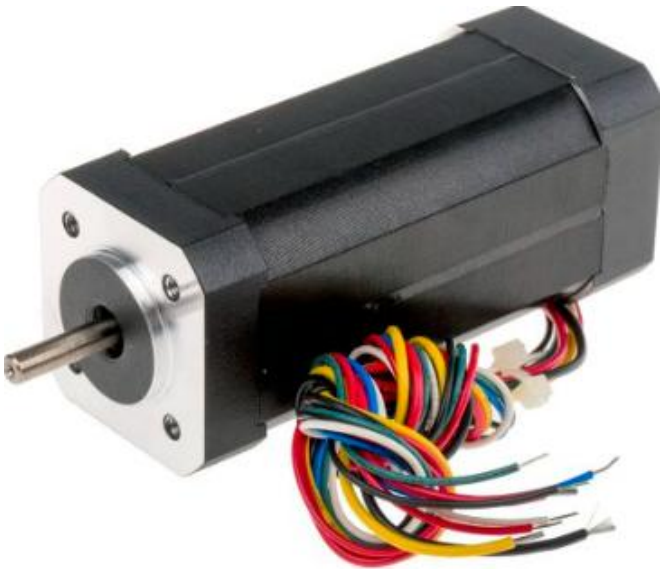


Figura 2: Motore elettrico RS PRO Brushless

Il cambio originale è stato rimosso e la linea di trasmissione è stata riprogettata collegando direttamente il motore elettrico all'albero tramite un pignone a 20 denti. Questa scelta progettuale ha permesso di semplificare significativamente la struttura meccanica del veicolo, riducendo il numero di componenti meccaniche in movimento e di conseguenza il numero di fonti di guasto, aumentando l'affidabilità del sistema. La scelta del pignone selezionato non è stata casuale, ma risponde alla necessità di limitare l'accelerazione laterale ottenibile dal veicolo, responsabile di sollecitazioni laterali e quindi instabilità dinamica. Il pignone da 20 denti è la scelta migliore per i motivi appena spiegati, rispetto ad esempio all'utilizzo di un pignone a 25 o 30 denti [15].

Per l’alimentazione del veicolo è stata adottata una batteria a litio da 24 V, scegliendo nello specifico una configurazione 7S4P con una capacità complessiva di 30000 mAh. Questa soluzione è stata individuata come la più adatta per soddisfare le esigenze energetiche del sistema. La selezione di questo modello è stata influenzata dallo spazio disponibile all’interno del telaio, reso utilizzabile dopo la rimozione del motore a benzina e delle componenti associate. È stato necessario inoltre considerare il peso massimo sostenibile dalla struttura del veicolo, per non compromettere dinamica e stabilità, oltre alla compatibilità elettrica e prestazionale con il motore brushless installato, che richiede una tensione di alimentazione adeguata e una capacità sufficiente per garantire un’autonomia idonea alle sessioni di test. La batteria scelta rappresenta dunque un equilibrio ottimale tra disponibilità di energia, ingombro fisico del sistema e compatibilità con l’architettura del veicolo.



Figura 3: Batteria a litio 7S4P

2.2 Attuazione e Controllo: servosterzo e controller motore

Sul veicolo è inoltre istallato un servomotore dedicato allo sterzo, componente essenziale per garantire un controllo preciso della direzione durante le manovre. Il dispositivo utilizzato è il S900S 1/5, servo in grado di fornire la coppia necessaria a generare un'azione di sterzata reattiva anche in condizioni dinamiche complesse. Il suo funzionamento si basa sulla rotazione del braccio del servo, che trasmette il movimento a un sistema di aste e leve di sterzo collegate alle ruote anteriori. Con questo meccanismo si trasforma il comando elettrico proveniente dal sistema di controllo in un movimento meccanico accurato, permettendo al veicolo di effettuare cambi di direzione con elevata precisione. È importante sottolineare che questa componente non ha subito modifiche durante il processo di elettrificazione, era presente anche nella versione originale del Losi ed è stata mantenuta in quanto pienamente compatibile con il nuovo sistema di trazione.



Figura 4: Servomotore di sterzo S900S

Per garantire il corretto funzionamento del motore, il veicolo è stato dotato anche di un controller dedicato alla gestione della trazione, dispositivo che regola in maniera accurata la potenza erogata, la velocità di rotazione e la direzione del motore. L'adozione di un controllore specifico è fondamentale, soprattutto in presenza di algoritmi di controllo eseguiti in tempo reale con Speedgoat.

Il modello selezionato per questa attività è il Maxon ESCON 50/5 (*Figura 5*), controllore servo PWM a quattro quadranti, caratterizzato da un'elevata potenza e da dimensioni estremamente compatte, che lo rendono adatto all'utilizzo per un veicolo in scala. La scelta di questo specifico controller è stata guidata dall'ottima compatibilità elettrica e funzionale con il motore brushless installato, oltre che per la sua capacità di garantire un controllo molto preciso della velocità ad anello chiuso, modalità di controllo particolarmente indicata per il sistema di trazione adottato grazie alla presenza di un sensore di Hall integrato nel motore.
Il controllore è dotato di diversi canali I/O analogici e digitali che permettono una gestione flessibile dei segnali di comando e delle misure provenienti dal sistema. La configurazione del dispositivo avviene tramite l'interfaccia grafica "ESCON Studio", software disponibile per i sistemi Windows che consente di impostare in maniera corretta tutti i parametri operativi del controller.

Una delle modalità principali attraverso cui il controllore riceve i comandi è rappresentata dai segnali PWM (Pulse Width Modulation), ovvero segnali digitali che alternano rapidamente stati di ON e OFF di una grandezza fisica, come tensione o corrente. In questo tipo di segnale non varia la frequenza di un ciclo ON-OFF ma la durata relativa della fase ON rispetto alla fase OFF, parametro definito duty cycle. Per esempio, un duty cycle del 100% corrisponde a un segnale costantemente in stato ON, mentre un duty cycle del 50% indica che il segnale resta attivo per metà tempo e inattivo per l'altra metà. Variando questo rapporto si possono ottenere tutti i valori intermedi fino allo stato completamente OFF, ovvero duty cycle pari a 0%. In pratica, un segnale PWM permette di modulare un segnale digitale in modo da ottenere un comportamento equivalente a quello di segnale analogico.
Nel caso specifico, L'ESCON utilizza il PWM per regolare la quantità di potenza fornita al motore, adattandone la velocità di rotazione con elevata precisione in funzione dei comandi provenienti dal Joystick.
Anche il servomotore di sterzo impiega un segnale PWM per far determinare l'angolo di sterzata, traducendo il duty cycle in un movimento meccanico del braccio del servo. La presenza di questa tipologia di segnale sia nel sistema di trazione che in quello di sterzo rende i segnali PWM fondamentali per il funzionamento coordinato dell'intero sistema di attuazione del veicolo.

Il controller è stato correttamente collegato a tutto il resto dell'hardware. L'installazione ha richiesto una particolare attenzione alla disposizione dei cablaggi, alla gestione delle alimentazioni e alla compatibilità dei segnali, elementi fondamentali per garantire affidabilità e sicurezza durante le prove. Infatti, una configurazione non ottimale dei collegamenti potrebbe causare interferenze, cadute di tensione compromettendo la qualità del test.
Per una descrizione più approfondita delle modalità con cui sono stati organizzati i collegamenti elettrici si rimanda al capitolo 1.2 di [16], dove sono spiegate in dettaglio tutte le scelte effettuate.



Figura 5: Controller ESCON 50/5

2.3 Sistema di Localizzazione: unità GPS-IMU

Il veicolo è inoltre dotato del modulo canmod.gps (*Figura 6*), dispositivo che integra in un'unica unità i segnali provenienti da un ricevitore GPS/GNSS e da un'unità di misura inerziale (IMU). Grazie alla fusione di questi dati, il modulo è in grado di fornire informazioni indispensabili per la localizzazione e per l'analisi della dinamica del sistema, combinando misure di posizione, velocità e accelerazione. Canmod.gps appartiene alla più ampia piattaforma CANmod, sviluppata per la decodifica e la gestione di dati connessi tramite rete CAN, standard ampiamente utilizzato in ambito automobilistico per la sua affidabilità.

L'adozione di questo modulo riveste un'importanza cruciale esclusivamente per l'implementazione della modalità di controllo ad anello chiuso, in quanto il funzionamento del sistema di guida autonoma con una traiettoria richiede una stima della posizione del veicolo il più precisa possibile.

Durante la fase di tracking della traiettoria, il calcolatore determina istante per istante gli errori di track error e di heading error, calcolati confrontando la posizione stimata dal veicolo con un punto della traiettoria preso come riferimento, di cui si parlerà poi nei capitoli successivi. Senza una misura accurata della posizione, questi errori risulterebbero affetti da rumori, compromettendo la capacità del veicolo di seguire la traiettoria.

Nel controllo tramite joystick, invece, non si è utilizzato questo strumento, in quanto la guida manuale non richiede informazioni di localizzazione, basandosi esclusivamente sugli input forniti dall'operatore.



Figura 6: Modulo CANmod.gps

Il modulo CANmod.gps fornisce le coordinate geodetiche del sistema, ovvero latitudine, longitudine e altitudine. Oltre ai dati di localizzazione, il dispositivo misura anche le accelerazioni lungo gli assi X e Y, informazioni fondamentali per descrivere il comportamento dinamico sul piano orizzontale, e la velocità angolare attorno all'asse Z, ovvero l'angolo di imbardata. Complessivamente, quindi il modulo restituisce 6 grandezze.

Considerata la complessità nel lavorare con coordinate geodetiche, che non sono lineari come le coordinate cartesiane e per questo sono molto difficili da utilizzare nei modelli matematici, si è resa necessaria una procedura di conversione. Inoltre, l'utilizzo delle coordinate cartesiane si adatta perfettamente al Filtro di Kalman impiegato nel sistema di controllo (di cui si parlerà nel capitolo successivo).

Per la conversione è stata adottata una procedura di conversione tramite la funzione *latlon2local* di MATLAB. Essa consente appunto di trasformare le coordinate geodetiche di un punto in coordinate cartesiane. La funzione chiede in input le 3 coordinate geodetiche del punto che si desidera convertire e le 3 coordinate cartesiane del punto scelto come origine del sistema di riferimento cartesiano e restituisce in output le coordinate cartesiane.

2.4 Architettura Speedgoat

Il veicolo è stato ulteriormente dotato della macchina-bersaglio in tempo reale Speedgoat, (*Figura 7*) elemento cruciale per lo svolgimento di prove e simulazioni in tempo reale. L'integrazione di questo sistema è fondamentale, in quanto consente di eseguire direttamente su veicolo gli script sviluppati in Simulink/Matlab, garantendo un'interazione continua tra hardware e software e assicurando un comportamento del sistema coerente a quello previsto in fase di progetto.

In particolare, Speedgoat non si limita a comandare gli attuatori elettronici e meccanici del veicolo, come il servosterzo e il controller del motore elettrico, ma svolge anche un ruolo essenziale nella gestione dei dati, il tutto in tempo reale. Infatti, è Speedgoat che si occupa della raccolta, dell'elaborazione e della sincronizzazione delle informazioni provenienti dai sensori. Questo consente di monitorare in modo accurato lo stato del sistema, valutando parametri e prestazioni con elevata precisione. La possibilità di osservare tutto in tempo reale permette di individuare rapidamente eventuali anomalie e migliorare la qualità complessiva del controllo.



Figura 7: Macchina bersaglio in tempo reale Speedgoat

Grazie all'integrazione con l'Add-on Simulink Real-Time di Mathworks e con i blockset di Input-Output forniti da Speedgoat, è possibile trasferire i modelli sviluppati in ambiente di simulazione direttamente su hardware, consentendo un'esecuzione immediata e sincronizzata con i segnali provenienti dal computer. Per poter utilizzare correttamente Speedgoat è

necessario stabilire una connessione tra il PC su cui sono presenti gli script e la macchina bersaglio installata sul veicolo (*Figura 8*). Tale collegamento richiede l'inserimento dell'indirizzo IP della macchina bersaglio all'interno della finestra di configurazione apposita in simulink, accessibile tramite la barra degli strumenti sotto la sezione 'Real-Time', come mostrato nella figura sottostante. Una volta impostato l'indirizzo Simulink, Real Time è in grado di comunicare con Speedgoat, trasferire il modello compilato ed eseguirlo in tempo reale.

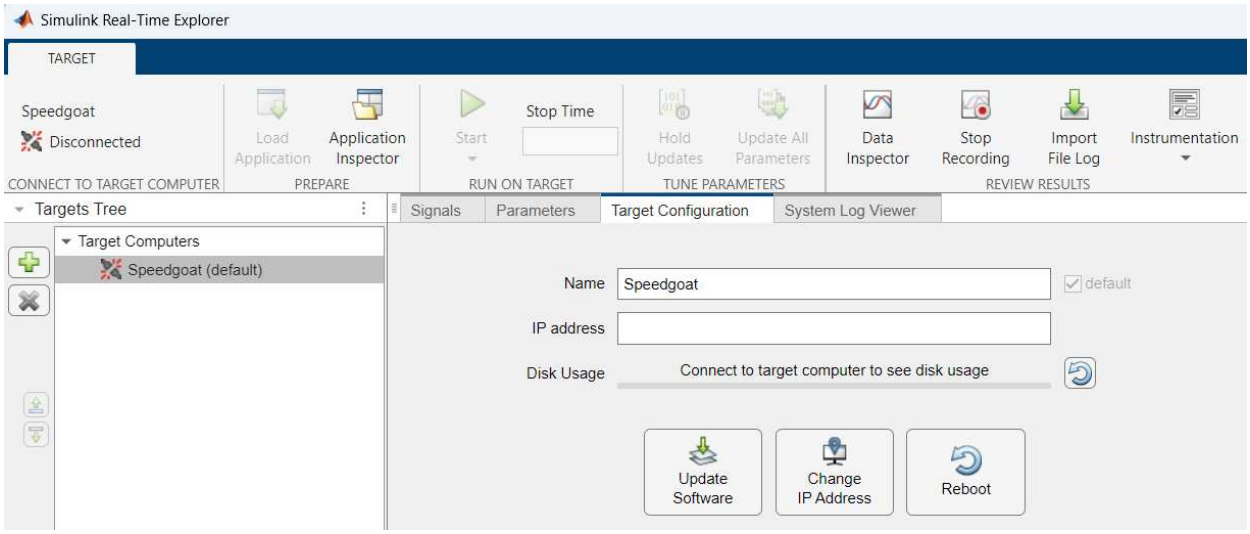


Figura 8: Barra degli strumenti Speedgoat

Il collegamento tra PC e macchina Speedgoat può avvenire sia tramite router sia attraverso cavo LAN diretto. Per garantire la massima stabilità, ridurre eventuali ritardi nella comunicazione e aumentare la velocità della simulazione, nel corso di questa tesi si è sempre adottato il collegamento diretto tramite LAN.

Speedgoat si avvale di un'architettura avanzata di interfacce di input-output che consente di collegare la macchina real-time ai diversi dispositivi installati sul veicolo, tra i quali sensori, attuatori e controllori. Infatti, ogni modulo hardware presente è completamente integrato su Simulink Real-Time attraverso l'utilizzo dei blockset di input-output forniti da Speedgoat. Questi blocchi forniscono una libreria di blocchi specifici che possono essere configurati all'interno dei modelli Simulink per eseguire operazioni di lettura e scrittura.

Nel corso di questa tesi è stata utilizzata la libreria I/O 397, che include blocchi dedicati alla gestione dei segnali analogici, digitali, PWM e i protocolli necessari per interfacciarsi con i dispositivi installati su veicolo. Nel capitolo successivo verranno descritti nel dettaglio i blocchi utilizzati negli script di controllo, specificando il ruolo di ciascun blocco e illustrando come ogni elemento sia stato impiegato per eseguire azioni dirette sul veicolo, come la lettura dei dati dei sensori o il comando degli attuatori. Questa analisi permetterà di comprendere in maniera chiara come Simulink, grazie a Speedgoat, interagisce con l'hardware come gestisce l'esecuzione real-time del sistema.

3. Tecnica di controllo tramite Input manuale

In questo capitolo vengono descritti tutti gli step necessari per l'implementazione del controllo del veicolo tramite input manuale, in particolare tramite un Joystick. L'obiettivo è quello di illustrare nel dettaglio il percorso seguito, partendo dalle modalità con cui Simulink acquisisce e interpreta i segnali in input. La fase di acquisizione rappresenta il primo elemento critico, in quanto il segnale deve essere letto, elaborato e convertito in grandezze compatibili con il modello. Successivamente viene spiegato come tali segnali vengono opportunamente elaborati e utilizzati per generare i comandi di controllo del veicolo fino a giungere alla realizzazione di un sistema di controllo completo.

3.1 Controller: Logitech F710 Wireless

Per questa applicazione è stato scelto il Logitech F710 Wireless Gamepad, un controller wireless progettato per l'uso su PC. Questo joystick offre una combinazione di precisione e flessibilità, caratteristiche che lo rendono ideale per il comando del Losi. La scelta di un dispositivo wireless ha permesso di ridurre l'ingombro di cavi e di facilitare i movimenti del programmatore. Inoltre, la connessione wireless del F710 assicura una comunicazione stabile e con un ritardo ridotto.



***Figura 9**: Logitech F710 Wireless Gamepad*

Per acquisire i segnali provenienti da Joystick, nel modello Simulink si è utilizzato il blocco 'Joystick input', progettato specificamente per la gestione dei controller manuali.
Questo blocco (*figura 10*) rappresenta il punto di ingresso dei comandi dell'operatore all'interno del modello di controllo. Il blocco è dotato di due uscite:

- Axes, che rappresenta i segnali analogici generati dalle levette del Joystick. I segnali possono assumere qualsiasi valore compreso nell'intervallo [-1; +1], dove i valori estremi corrispondono alle posizioni massime delle levette e lo zero rappresenta la posizione neutra.
- Buttons, che corrisponde ai segnali digitali associati ai tasti presenti sul controller. Ogni segnale assume il valore 1 quando il tasto corrispondente è premuto, 0 in caso contrario.



Figura 10: Blocco 'Joystick Input' di Simulink

Il modello F710 di Logitech è dotato di 10 segnali digitali e 5 segnali analogici, due per ogni levetta analogica (movimento destra-sinistra e avanti-indietro) più un ulteriore segnale generato dai tasti RT e LT, posizionati nella parte superiore del Joystick. Per questa applicazione si sono utilizzati soltanto due segnali analogici: il movimento destra-sinistra della levetta destra, impegnato per controllare l'angolo di sterzo, e il movimento avanti-indietro della levetta sinistra, utilizzato per regolare la velocità e il verso di avanzamento.

Come spiegato in precedenza, Simulink interpreta i segnali analogici fornendo loro un valore compreso tra -1 e +1. Per il controllo dello sterzo, si è stabilito che il valore +1 corrisponde al massimo angolo di sterzo del veicolo verso destra, -1 corrisponde al massimo angolo di sterzo del veicolo verso sinistra, mentre un valore pari a 0 indica una posizione neutra delle ruote, ovvero sterzo nullo. I valori intermedi sono proporzionali alla posizione della levetta

del joystick e al relativo angolo di sterzo. In entrambe le direzioni il massimo angolo di sterzo è pari a 32.5°.
Allo stesso modo per la velocità, un segnale analogico pari a +1 rappresenta la massima velocità in avanti, -1 rappresenta la massima velocità in retromarcia e il valore 0 rappresenta velocità nulla, ovvero veicolo fermo. Il massimo valore di velocità, in entrambi i sensi di marcia, è pari a 4km/h.

Le levette del Joystick sono affette da un piccolo drift: utilizzando uno scope su Simulink, blocco che permette di vedere in real time il valore di un segnale, si è notato come anche con le levette in posizione di riposo i due valori analogici non sono pari a 0. Questo comportamento, tipico dei controller, potrebbe però portare ad errori indesiderati nel sistema di controllo; per esempio, verrebbe applicato un angolo di sterzo anche quando la levetta relativa è in posizione di riposo. Per correggere questo effetto indesiderato si è utilizzata la funzione tangente iperbolica, opportunamente tarata come mostrato nella formula seguente:

$$\delta = |\delta_0| \cdot \tanh\big(3000 \cdot (\delta_0 - 0.003906)\big) \quad \textbf{(1)}$$

Dove δ rappresenta l'angolo corretto senza drift, mentre δ_0 rappresenta il valore letto dal joystick mentre le costanti sono state opportunamente calibrate per ottenere l'effetto desiderato. Gli stessi valori sono risultati idonei anche al segnale che rappresenta la velocità.
Questa funzione ha il compito di mantenere il segnale pari a 0 quando le levette non sono azionate, eliminando così il drift, ma allo stesso tempo garantisce che il comportamento delle levette sia lineare e proporzionale quando vengono effettivamente utilizzate. Tanh quindi agisce proprio come un filtro, che fa comparire a 0 il segnale analogico se la levetta non azionata, mentre lascia invariato il segnale se è maggiore o minore di 0. In questo modo, si ottiene un segnale corretto, che riflette fedelmente l'input dell'operatore senza introdurre errori di offset.

3.2 Protocollo di comunicazione Wireless UDP

Il blocco 'Joystick input' utilizza una S-function per comunicare con Simulink, ovvero una funzione scritta con un linguaggio di programmazione (solitamente MATLAB, C o C++) che permette di estendere le funzionalità dell'ambiente di simulazione, integrando nuovi comportamenti e driver che non sono disponibili nella libreria standard di Simulink. Le S-function sono fondamentali per collegare Simulink ai dispositivi esterni, perché definiscono con precisione come il blocco deve leggere e interpretare i segnali provenienti dall'hardware. Nel caso specifico, la S-function permette al blocco di dialogare direttamente con il sistema e il driver del controllore, garantendo la comunicazione con il joystick

Durante le prime prove realizzate con il veicolo e il Joystick, si è notato che la S-function utilizzata dal blocco "Joystick Input" è incompatibile con Simulink real-time, add-on di Simulink indispensabile per connettersi alla macchina bersaglio Speedgoat e per realizzare simulazioni in tempo reale. Non esistono attualmente Joystick che presentano una S-function certificata per l'uso in real-time, perché la comunicazione con dispositivi USB richiede driver che non possono essere eseguiti in real-time. L'unico modo di utilizzare il Joystick, quindi, è quello di eseguire il modello Simulink non in tempo reale, ma questa modalità impedisce qualsiasi comunicazione diretta con il veicolo: Simulink riuscirebbe a dialogare con il Joystick, ma non con Speedgoat e quindi con il veicolo.
Questo limite ha reso necessario elaborare una soluzione alternativa per sviluppare il controllo manuale del veicolo, che consiste nel dividere il modello Simulink in due parti distinte, ciascuna con un ruolo specifico:

- la prima parte non viene eseguita in modalità real-time e ha come unico scopo quello di leggere il segnale del Joystick attraverso il blocco 'Joystick input' e la relativa S-function, potendo sfruttare tutte le funzionalità che essa apporta senza i vincoli imposti dal real-time;
- la seconda parte, invece, è sviluppata utilizzando Simulink Real-Time e viene eseguita on target connettendosi a Speedgoat. È questo lo script responsabile della comunicazione diretta con il veicolo, dell'elaborazione dei comandi e dell'attivazione del servosterzo e del controller del motore.

Le due parti sono in comunicazione tra loro mediante il protocollo UDP (User Datagram Protocol), integrato su Simulink. Esso permette di inviare pacchetti di dati a un indirizzo IP specificato senza stabilire una connessione rigida tra i dispositivi. Questa caratteristica rende il protocollo molto flessibile, in quanto è sufficiente conoscere l'indirizzo IP della macchina

destinataria per trasmettere i dati. Inoltre, la comunicazione tramite UDP risulta istantanea, caratteristica essenziale per garantire una risposta tempestiva ai comandi impartiti tramite joystick.

Nella prima parte dello script, quella eseguita non in modalità real-time, è stato dunque inserito il blocco 'UDP send', con lo scopo di inviare un pacchetto di dati a una porta specifica di un determinato indirizzo IP. In questa applicazione, il pacchetto di dati inviato contiene i due segnali analogici acquisiti da Joystick, quello della velocità e quello dell'angolo di sterzo. Come si vede nell'immagine di riferimento, il blocco 'UDP send' presenta due ingressi:

- *data*, al cui vanno collegati i segnali da trasmettere;
- *length,* che indica il numero di byte effettivamente validi da includere nel pacchetto. Questo parametro permette di specificare la dimensione reale dei dati trasmessi, evitando l'invio di byte non significativi.

Per stabilire la comunicazione è necessario inserire nel blocco UDP Send l'indirizzo IP della macchina destinataria e il numero della porta. L'indirizzo IP identifica univocamente il target all'interno della rete; in quest'applicazione si è inserito l'indirizzo IP di Speedgoat. Il numero della porta specifica, invece, può essere scelto liberamente dal programmatore, purchè sia la stessa inserita nel blocco di ricezione dei dati.

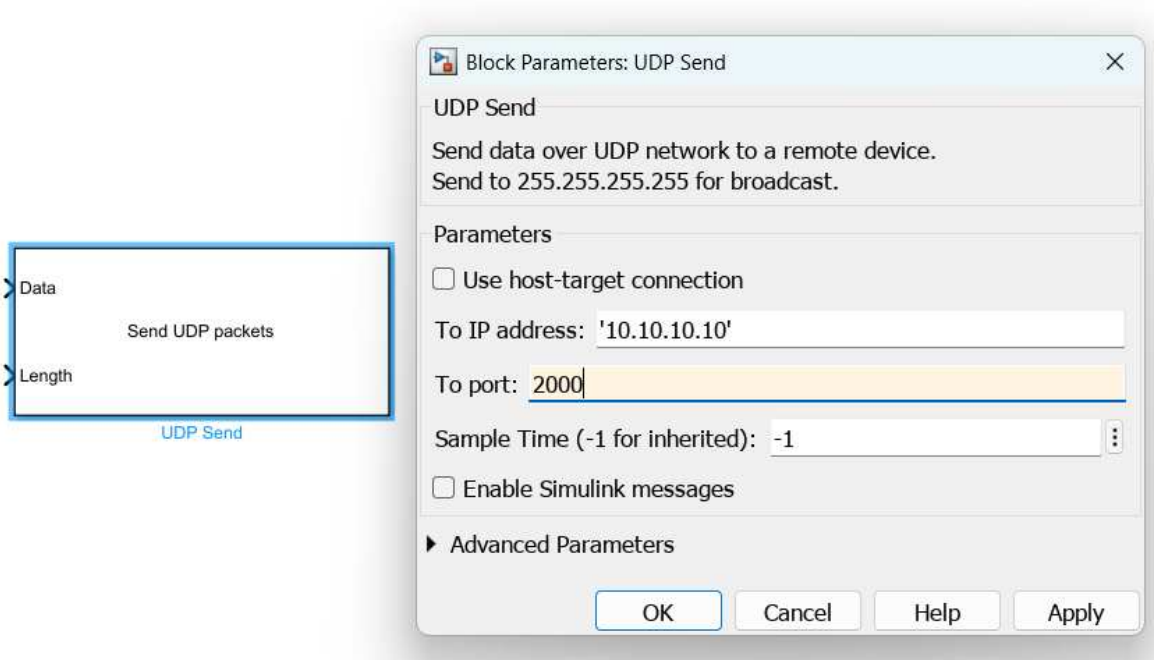


Figura 11: Blocco 'UDP send' di Simulink

Nel secondo script, quello eseguito in tempo reale sulla macchina bersaglio, è stato inserito il blocco 'UDP receive' (*Figura 12*), necessario per acquisire il pacchetto di dati inviato dall'altra parte del modello. Il blocco presenta le due seguenti uscite:

- *data*, che contiene il segnale ricevuto;
- *length,* che esattamente come il blocco visto in precedenza indica il numero di byte effettivamente ricevuti nel pacchetto.

Il blocco include anche il parametro 'Receive width', il quale definisce il numero massimo di elementi che il vettore può contenere. Per semplificare la gestione dei segnali e rendere tutto il modello più chiaro si è scelto di inviare i due segnali analogici dal Joystick separatamente, utilizzando quindi due blocchi 'UDP send' e di conseguenza due blocchi 'UDP receive'. Questa scelta semplifica notevolmente sia l'invio, in quanto si ha a che fare con vettori meno pesanti, sia la decodifica dei comandi.

Per quanto appena spiegato, in 'Receive width' si è inserito il valore 1, indicando che ogni blocco deve ricevere un unico elemento alla volta

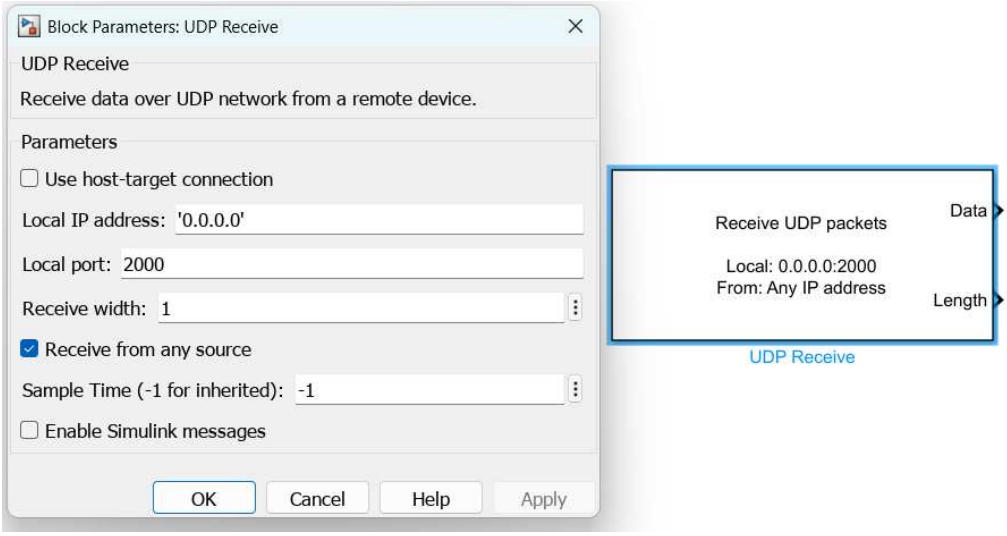


Figura 12: Blocco 'UDP receive' di Simulink

Poiché il protocollo UDP trasmette i pacchetti di dati in formato byte, è necessario effettuare una conversione del segnale prima dell'invio. In particolare, i valori letti da Joystick sono di tipo double, mentre il blocco 'UDP send' richiede dati di tipo uint8, ovvero numeri interi senza segno (solo positivi) a 8 bit. Questa discrepanza rende indispensabile una fase di adattamento del formato dei dati. Per questo, nel primo script il segnale viene convertito da double a uint8. Nella seconda parte, quella da eseguire in tempo reale, i due segnali inviati

andranno poi riconvertiti in double, in quanto è questo il formato adatto per la corretta attuazione di servomotore e controller.

Inoltre, come già specificato, è fondamentale assicurarsi che il numero della porta specifica inserito in due blocchi UDP Send/Receive coincida (si vedano le Figure 11 e 12, in cui i valori inseriti in 'To port' e 'Local Port' sono identici). In caso contrario, il trasferimento del pacchetto di dati non avviene in maniera corretta, poiché Speedgoat non sarebbe in grado di ricevere il flusso associato alla porta prevista. Il protocollo UDP, infatti, instrada i pacchetti esclusivamente sulla porta indicata: se il numero non coincide, il pacchetto viene scartato. Nel modello sviluppato, poiché i due segnali sono stati gestiti separatamente, si sono utilizzate porte differenti, una per ogni segnale. In questo modo ogni segnale viene instradato sulla propria porta dedicata, semplificando la gestione e prevenendo eventuali interferenze.

È altrettanto fondamentale assicurarsi che i due script siano eseguiti in contemporanea durante l'attività di controllo tramite Joystick. Entrambi, infatti, svolgono un ruolo indispensabile all'interno dell'architettura complessiva: il primo è responsabile della lettura dei segnali provenienti dal Joystick, il secondo gestisce la comunicazione con la macchina bersaglio e quindi con il veicolo. L'assenza o il malfunzionamento di uno dei due comprometterebbe il funzionamento dell'intero sistema di controllo.

3.3 Integrazione interfaccia I/O con l'hardware

Nel secondo script, quello da eseguire in tempo reale collegandosi alla macchina bersaglio, sono poi stati inseriti i blocchi di libreria di Speedgoat, necessari affinché la macchina real-time possa agire correttamente sul servosterzo e sul controllore del motore.

Come prima operazione è stato inserito il blocco 'Setup' (figura 13), indispensabile per stabilire la comunicazione: senza di esso, il sistema non sarebbe in grado di interfacciarsi con l'hardware. Ogni modello destinato all'esecuzione in tempo reale deve contenere solo un blocco 'Setup', che deve essere scelto in base ai moduli di I/O utilizzati. Nel nostro caso, essendo impiegata la scheda IO397, è stato selezionato il corrispondente Setup fornito dalla libreria Speedgoat.

Il file 'Speedgoat_IO397_RCP.mat' rappresenta il configuration file del modulo I/O397 e contiene tutte le informazioni necessarie per l'utilizzo corretto del modulo, comprese le funzioni abilitate e le impostazioni per l'interfacciamento con Simulink Real-Time. E' fondamentale inserire correttamente questo file di configurazione, poiché garantisce che il modello riconosca il modulo, configuri correttamente le porte di I/O e permetta il funzionamento di tutti i dispositivi presenti sul veicolo.

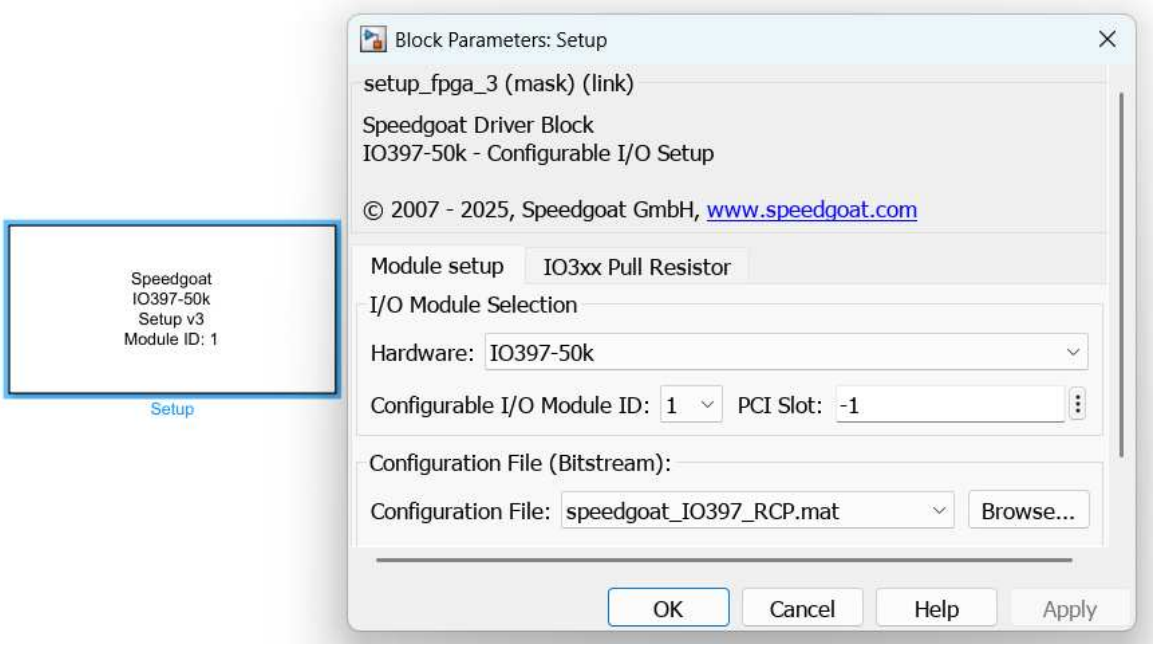


Figura 13: Blocco 'Setup' per IO397 di Simulink

Successivamente sono stati inseriti i blocchi PWM generation (Figura 14), responsabili della generazione dei segnali PWM necessari per comandare sia il controllore del motore che il servomotore dello sterzo. Il blocco presenta un unico ingresso, al quale va collegato il valore di duty cycle del segnale PWM che determina la quantità di potenza trasmessa al dispositivo. È importante sottolineare che il segnale in uscita dal joystick non corrisponde al valore di duty cycle richiesto dal blocco. Come già visto in precedenza, in uscita dal blocco 'Joystick Input' il segnale assume un valore compreso tra -1 e +1. Prima di entrare in questo blocco, quindi, il segnale deve essere opportunamente scalato e adattato, in modo da mappare l'intero intervallo del segnale in uscita da Joystick sul corrispondente valore di duty cycle. In questo modo, il segnale diventa compatibile al blocco PWM Generation.

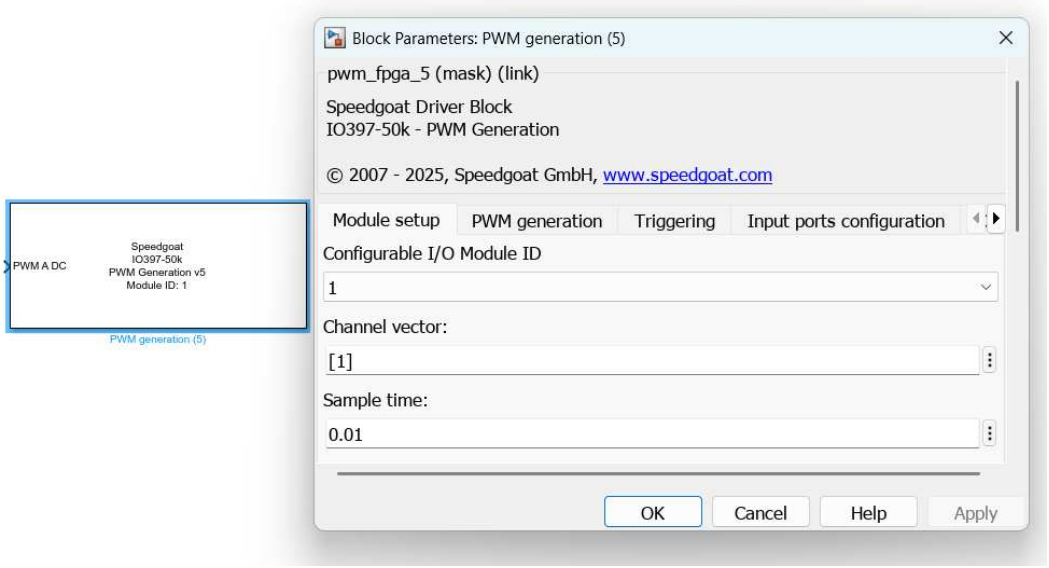


Figura 14: Blocco 'PWM generation (5)' di Simulink

È importante prestare attenzione a un dettaglio: il blocco PWM generation rimane lo stesso indipendentemente dalla grandezza che si desidera controllare, sia essa l'angolo di sterzo o la velocità. Ciò che varia è il valore impostato in 'Channel vector', all'interno del blocco, che determina quale canale fisico di IO397 verrà utilizzato per la generazione. Ad esempio, se nel campo viene inserito il numero 1, come mostrato in figura 14, il segnale PWM generato agisce sul servosterzo modificando l'angolo delle ruote. Se invece viene inserito il numero 2, il blocco controllerà il controllore del motore elettrico regolando la velocità di avanzamento del Losi.

L’ultimo aspetto da considerare è il senso di marcia del veicolo. Il segnale PWM generato dai blocchi è in grado di controllare solamente l’intensità della potenza erogata, ossia il duty cycle, ma non può determinare il verso di rotazione del motore, che non varia con il variare del PWM.

Per gestire correttamente la direzione di avanzamento o retromarcia si utilizzano i blocchi ‘Digital Output’, mostrati in figura 15. Questi blocchi permettono di inviare segnali digitali al controllore del motore, stabilendo se il veicolo deve muoversi in avanti o indietro. In combinazione con i segnali PWM, i blocchi ‘Digital Output’ consentono di controllare sia la velocità che il senso di marcia, garantendo un funzionamento completo e coerente del sistema di trazione.

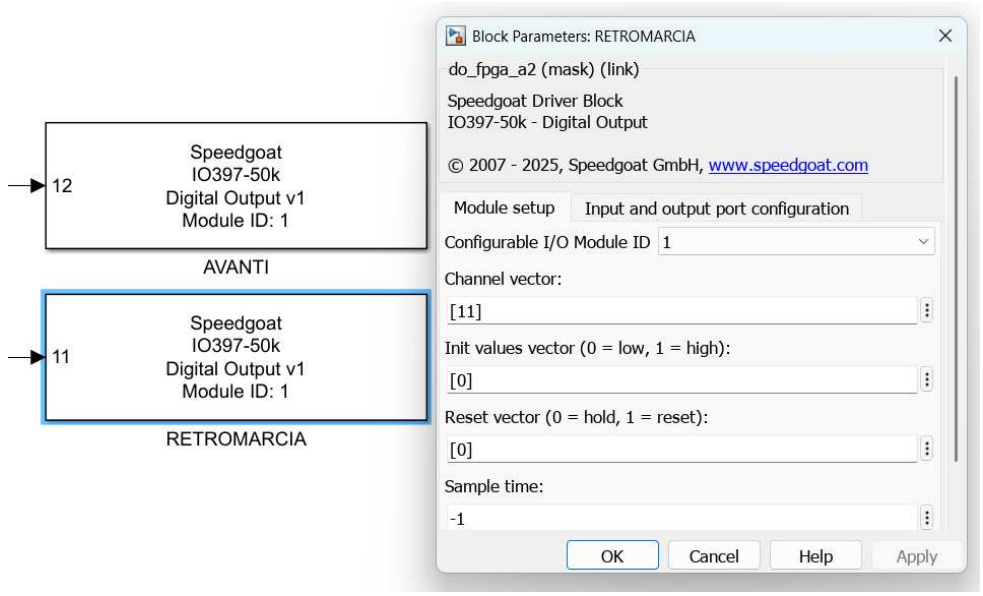


Figura 15: Blocco ‘Digital Output’ di Simulink

Anche in questo caso, come per i blocchi ‘PWM generation’, il blocco ‘Digital Output’ è unico; la struttura del blocco non cambia, così come il tipo di segnale richiesto in input. Ciò che cambia è il valore inserito nel parametro ‘Channel vector’. Ad esempio, inserendo il numero 12 tra parentesi quadre, Speedgoat interpreta il comando come istruzione per far muovere il veicolo in avanti. Se invece viene inserito il numero 11 (come nel caso mostrato in figura 15), il blocco imposta il veicolo in retromarcia.

Si precisa che il blocco ‘digital output’ deve essere presente nel modello con entrambe le direzioni di avanzamento, esattamente come mostrato nella figura. Infatti, si è osservato che

per un corretto funzionamento dello script i due blocchi devono necessariamente essere presenti entrambi, anche in caso di applicazioni che prevedono un solo senso di marcia.
Il funzionamento dei blocchi è molto semplice: ciascun blocco si attiva quando riceve in ingresso il valore 1, mentre si disattiva se riceve il valore -1. E' inoltre essenziale che i due blocchi non vengano mai attivati contemporaneamente, poiché ciò potrebbe generare conflitti nel controllore e compromettere il corretto senso di marcia del veicolo. La logica di controllo deve quindi garantire che l'attivazione di un blocco comporti in automatico la disattivazione dell'altro. Bisogna però analizzare con attenzione la logica di attivazione di questi due blocchi.
Basarsi esclusivamente sul segno del segnale di velocità letto dal Joystick nel primo script e trasmesso tramite UPD al secondo può infatti comportare rischi per i componenti hardware del veicolo. In particolare, se l'operatore agisce rapidamente sulla levetta, il segnale passa istantaneamente da un valore positivo, corrispondente alla marcia avanti, a uno negativo, che indica il cambio di senso di rotazione, o viceversa. Poichè il duty cycle del segnale è unico e valido per entrambi i sensi di marcia, un cambio improvviso potrebbe causare un'inversione immediata del verso di rotazione del motore. Questo fenomeno risulta particolarmente critico a elevati RPM: un'inversione istantanea del moto ad alti regimi genera sollecitazioni meccaniche molto intense sull'albero motore e sulla trasmissione che potrebbero compromettere l'integrità dei componenti.

Per gestire questa criticità è stata sviluppata una funzione specifica per la gestione dei due 'Digital Output', scritta per garantire che non vengano mai attivati simultaneamente e per prevenire inversioni improvvise del motore. La funzione, divisa in due parti e riportata per intero *nell'Appendice A*, ha lo scopo di consentire l'inversione del verso di rotazione del motore solo quando il segnale proveniente dal Joystick cambia di segno e quando gli RPM del motore sono inferiori a un valore soglia prestabilito. Questo valore soglia rappresenta il limite oltre il quale l'inversione del moto sarebbe meccanicamente rischiosa. Se il motore si trova a una velocità di rotazione superiore a questo valore soglia, la funzione impone una riduzione graduale dell'intensità di rotazione, e soltanto dopo aver raggiunto il valore soglia, la funzione impone l'inversione del senso di marcia. La funzione è stata scritta per funzionare per entrambi i versi di marcia.
Questo approccio garantisce un controllo sicuro del senso di rotazione, proteggendo le componenti e assicurando un comportamento stabile e affidabile del veicolo durante la guida manuale. La funzione riceve in input il segnale della velocità, proveniente da Joystick e

ricevuto tramite UDP, e il valore dei giri del motore elettrico in tempo reale, acquisito tramite i moduli Speedgoat. In base a questi dati la funzione elabora la logica e restituisce in output una coppia di valori che possono assumere +1 o -1. Questi valori vengono quindi utilizzati per attivare o disattivare i blocchi 'Digital Output', stabilendo in modo sicuro il senso di marcia del veicolo e garantendo che soltanto uno dei due possa essere attivato in un determinato istante.

Il valore istantaneo dei giri del motore viene acquisito tramite un blocco 'Analog input', fornito da Speedgoat e Simulink Real-Time per leggere segnali analogici provenienti dai sensori installati sul veicolo.

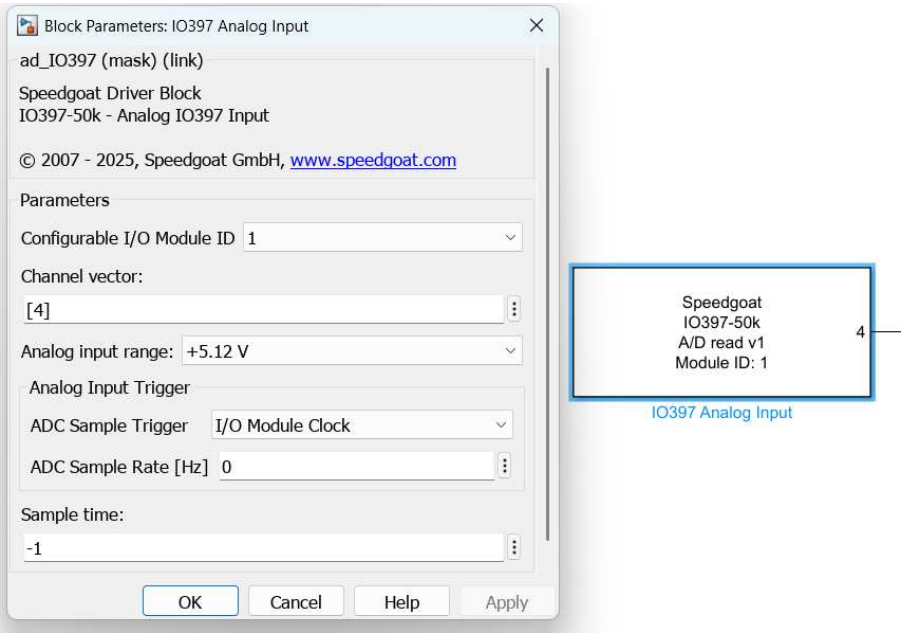


Figura 16: Blocco 'Analog Input' di Simulink

Come per i blocchi analizzati in precedenza, anche il blocco 'Analog input' è unico, ma il segnale che legge può variare in base al valore inserito nel parametro 'Channel Vector'. Questo parametro permette di selezionare quale ingresso analogico del modulo IO397 deve essere utilizzato, rendendo possibile l'acquisizione di diverse grandezze fisiche con lo stesso blocco. In particolare, le grandezze misurate possono essere:

- [1]: misura il segnale PWM relativo all'angolo di sterzo;
- [2]: misura il segnale PWM relativo alla velocità;
- [3]: misura la corrente;
- [4]: misura la velocità di rotazione del motore elettrico (caso in figura 16).

E’ importante chiarire un’ulteriore questione relativa ai ‘channel vector’ dei diversi blocchi utilizzati in questo capitolo. Il fatto che, a seconda del numero inserito, il blocco restituisce in output o legge in input una specifica grandezza non è una caratteristica intrinseca del modulo IO397, ma dipende esclusivamente da come sono stati cablati i vari componenti e i loro sensori sul veicolo. Il modulo IO397, infatti, si limita a leggere le tensioni presenti sui canali senza interpretarne il significato fisico, senza sapere a quale segnale è riferita la grandezza che si sta misurando. Di conseguenza, la corrispondenza tra canale e grandezza è determinata dalla configurazione dei sensori e dai collegamenti effettuati.
Per esempio, con una diversa disposizione dei sensori, in un blocco ‘Analog Input’ il valore [1] in ‘Channel vector’ poteva restituire la corrente, mentre il valore [3] poteva restituire il segnale PWM relativo all’angolo di sterzo. Per questo motivo, le associazioni tra canale e grandezza descritte in questo capitolo sono valide solo per l’applicazione specifica descritta in questa tesi, e non devono essere considerate come uno standard generale del modulo IO397. Ogni utilizzo di Speedgoat richiede uno studio accurato sulla configurazione dei cablaggi e dei sensori

Per concludere, si può osservare come i blocchi utilizzati fino a questo punto lavorino con segnali di duty cycle o tensioni associate alle diverse grandezze fisiche. Per ottenere il valore effettivo della grandezza misurata è necessario effettuare una mappatura dei dati, traducendo i segnali elettrici in valori fisici interpretabili.
Nel caso specifico della velocità del motore elettrico, fornita in input alla funzione per la gestione dei sensi di marcia, si è utilizzato per la mappatura un blocco ‘Lookup Table n-D’, visibile nella figura sottostante. Questo blocco consente di convertire i segnali analogici letti in tempo reale in valori di velocità effettiva del motore, permettendo alla funzione di controllo di operare in modo accurato e sicuro, adattando il comportamento dei blocchi ‘Digital Output’ e ‘PWM Generation alle condizioni reali del veicolo.

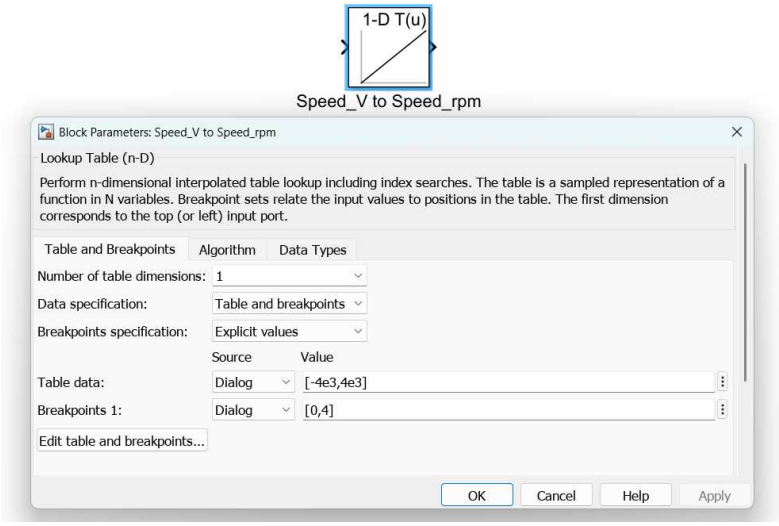


Figura 17: Blocco 'Lookup Table (n-D)' di Simulink

All'interno del blocco 'Lookup Table n-D', i campi 'Table Data' e 'Breakpoint 1' vengono utilizzati per definire rispettivamente gli intervalli dei valori di output e di input.

Di seguito vengono riportati i due modelli Simulink appena descritti, insieme a un breve riepilogo di ciò che viene fatto al loro interno. Per migliorarne la leggibilità all'interno del documento, anziché inserire gli screenshot dei modelli, verranno riportati diagrammi a blocchi che ne descrivono fedelmente la logica, risultando però più facilmente interpretabili.

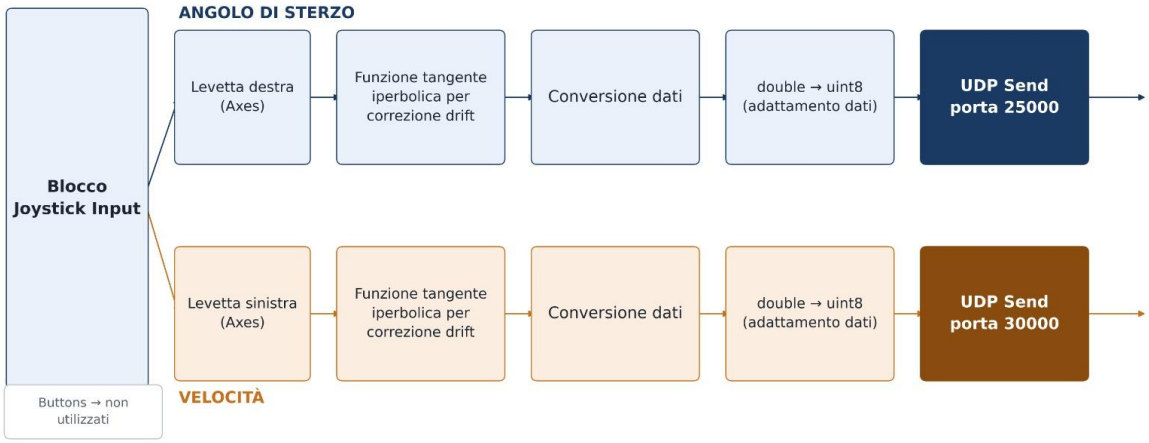


Figura 18: Schema logico del modello Simulink eseguito in locale

In figura 18 è mostrato il modello simulink eseguito in locale, responsabile della lettura dei due segnali provenienti dal Joystick. Dopo la lettura, i due segnali vengono corretti dal drift che gli affligge, utilizzando la tangente iperbolica come mostrato nella formula (1).

Successivamente, i due segnali vengono convertiti in uint8, formato adatto al protocollo UDP, per poi essere inviati all'altro modello, presentato in figura 19.
Nel blocco "conversione dati" il segnale viene trasformato in un intero compreso tra 0 e 255, processo necessario per la conversione in uint8 in quanto questa tipologia di segnale può assumere solo valori compresi in questo intervallo. La conversione avviene utilizzando la seguente formula:

$$Output_{blocco} = (V_{att} + V_{max}) \cdot \frac{255}{2 \cdot V_{max}} \quad \textbf{(2)}$$

Dove V_{att} rappresenta la velocità del Losi depurata dal drift mentre V_{max} è la massima velocità che può assumere il veicolo. Lo stesso procedimento viene applicato in maniera del tutto analoga anche all'angolo di sterzo.

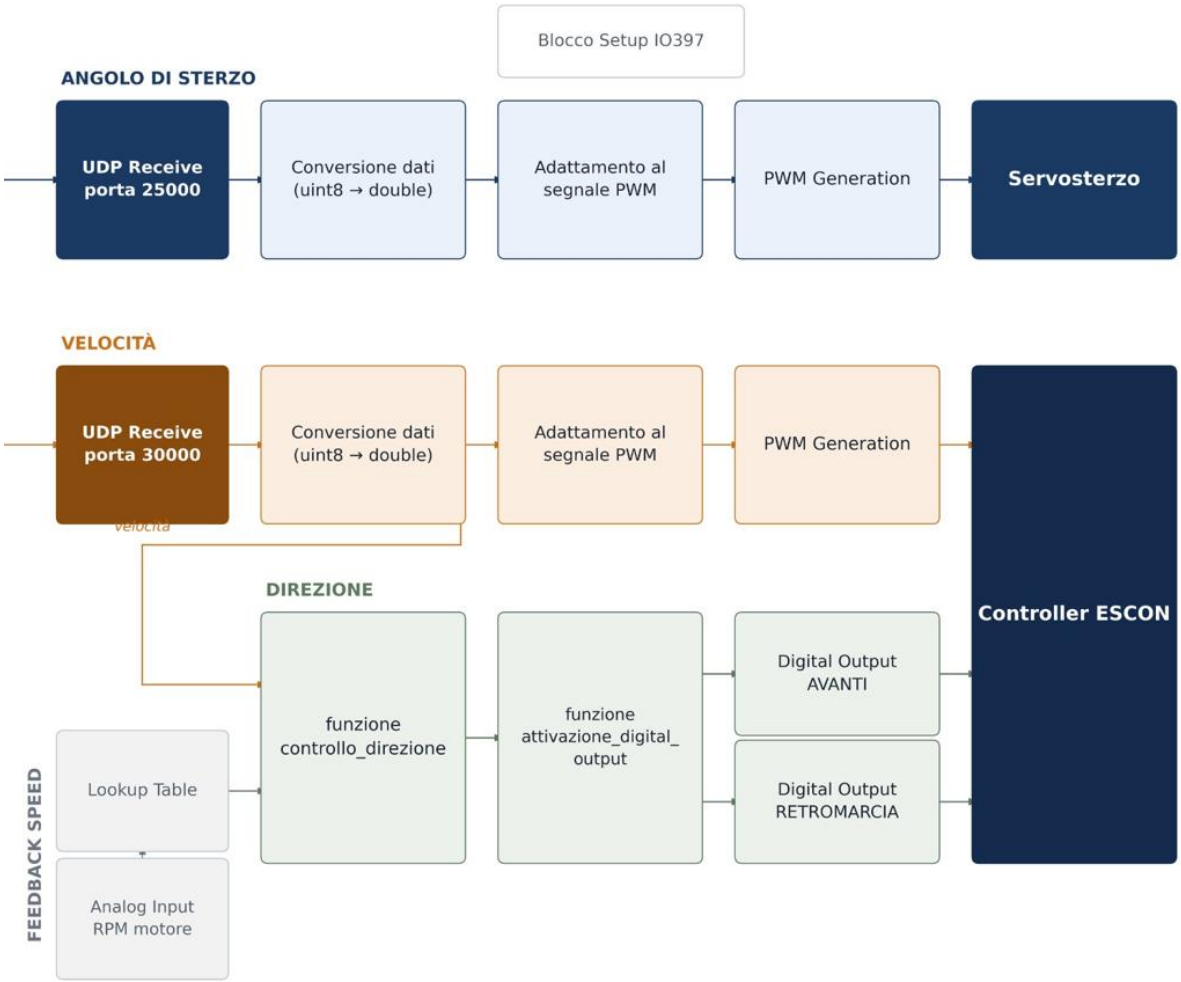


Figura 19: Schema logico del modello Simulink eseguito in real-time

In figura 19 è mostrato lo schema logico del modello eseguito in real-time, responsabile della comunicazione con Speedgoat. I due segnali di velocità e angolo di sterzo vengono ricevuti dai blocchi *UDP receive*, riconvertiti in double (prima nel formato [0 255], poi nel loro valore reale) e infine inviati al servosterzo e al controller utilizzando i blocchi PWM della libreria IO397. In questo modello sono inoltre presenti le due funzioni "*controllo_direzione*" e "*attivazione_digital_output*" entrambe consultabili nell'Appendice A, indispensabili per determinare il corretto verso di rotazione del motore elettrico. Infine, è presente anche il blocco di setup IO397, fondamentale per il corretto funzionamento di tutti i blocchi della libreria.

Questa suddivisione in due modelli distinti rappresenta una soluzione efficace al problema di compatibilità tra la S-function del Joystick e Simulink Real-Time, garantendo una comunicazione stabile e un controllo affidabile del veicolo.

4. Tecnica di controllo ad anello chiuso

Nella seconda parte della tesi viene approfondita la modalità di controllo ad anello chiuso (Closed-Loop). In questa configurazione il veicolo assume un comportamento sostanzialmente paragonabile a quello di un veicolo a guida autonoma, poiché il controllo non avviene più tramite un input manuale in tempo reale ma è finalizzato a far seguire al Losi una traiettoria predefinita. La traiettoria viene sviluppata in ambiente Matlab e successivamente fornita al veicolo attraverso Speedgoat, che ne consente l'esecuzione in tempo reale.

Il controllo avviene nelle seguenti fasi, che vengono eseguite ciclicamente durante il moto e permettono al sistema di correggere in tempo reale la traiettoria seguita dal veicolo:

1) la prima fase consiste nell'individuazione della posizione del veicolo nello spazio della traiettoria. In questa fase, grazie ai dati forniti dall'unità GPS-IMU installata a bordo, vengono stimati la posizione del veicolo e il suo orientamento, Questa informazione rappresenta lo stato corrente del sistema ed è fondamentale per confrontare la posizione reale del veicolo con il percorso di riferimento che deve essere seguito. In questa fase è fondamentale l'utilizzo del filtro di Kalman esteso, necessario per ridurre l'influenza del rumore sui dati acquisiti dall'unità GPS-IMU. Al fine di garantire un funzionamento corretto ed efficace del filtro è stata effettuata un'ottimizzazione delle matrici di covarianza, il cui procedimento e i risultati saranno illustrati nelle pagine successive.
2) Ricerca del punto della traiettoria più vicino al veicolo: una volta stimata la posizione del veicolo, questa viene confrontata con l'insieme dei punti che definiscono la traiettoria. Tra questi punti viene individuato quello che minimizza la distanza dal veicolo, il quale viene assunto come punto di riferimento per il controllo. In seguito verrà inoltre presentata una modalità alternativa per la selezione del punto di riferimento, basata sulla tecnica del Lookahead.
3) La terza fase riguarda il calcolo dell'errore di orientamento (heading error) e dell'errore laterale (cross-track error), al fine di guidare il veicolo lungo la traiettoria desiderata. A partire dal confronto tra la posizione e l'orientamento del veicolo e quelli del punto di riferimento individuato, vengono calcolati i due errori, che rappresentano rispettivamente la differenza di orientamento e la distanza laterale dal percorso. Il controllo utilizza tali errori per generare i comandi di sterzo e velocità,

con l'obiettivo di ridurli progressivamente e di mantenere il veicolo il più possibile aderente alla traiettoria assegnata.

4.1 Localizzazione e stima di stato tramite filtro di Kalman Esteso

Un filtro di Kalman esteso (Extended Kalman Filter, EKF) è un algoritmo di stima utilizzato per determinare lo stato di un sistema dinamico che evolve nel tempo. Esso rappresenta un'estensione del classico filtro di Kalman, il quale risulta applicabile esclusivamente a sistemi lineari. Grazie all'introduzione di opportune approssimazioni, l'EKF esteso consente invece di affrontare anche applicazioni non lineari, rendendolo particolarmente adatto a numerose applicazioni ingegneristiche.

La stima dello stato avviene a partire dalle misurazioni provenienti dai sensori, affette da rumori e incertezze. Il principio di funzionamento dell'EKF si basa sull'aggiornamento continuo di un insieme di variabili di stato che descrivono la condizione istantanea del sistema. In particolare, l'algoritmo è in grado di prevedere l'evoluzione dello stato nel tempo mediante un modello matematico del sistema, integrando al contempo le informazioni fornite dai sensori in tempo reale.

Un ruolo fondamentale è svolto dalla matrice di covarianza, la quale rappresenta il livello d'incertezza associato sia alla fase di previsione del modello sia alla fase di aggiornamento basata sulle misure. Attraverso questa matrice, l'EKF è in grado di bilanciare in modo ottimale il contributo del modello e quello dei sensori, determinando quanto peso attribuire a ciascuna informazione al fine di ottenere una stima dello stato quanta più accurata e affidabile possibile.

Il filtro opera in due fasi principali: nella prima fase, quella di predizione, si calcola lo stato successivo del sistema e l'errore associato ad esso, andando a modificare la matrice di covarianza, mentre nell'ultima fase, quella di correzione, il filtro utilizza nuove misurazioni per migliorare la sua predizione.

L'EKF è ampiamente utilizzato in numerose applicazioni di diversi ambiti industriali, grazie alla sua capacità di fornire stime accurate dello stato di un sistema anche in presenza di rumore e incertezza delle misurazioni. In ambito navale, per esempio, è fondamentale per la stima di posizione, della velocità e dell'orientamento dei veicoli, garantendo una navigazione più precisa e affidabile. È inoltre largamente utilizzato anche per sistemi autonomi, dove viene integrato con sensori e telecamere tecnologicamente avanzati per consentire una

corretta percezione dell'ambiente e una stima accurata dello stato del sistema. Inoltre, un ulteriore settore di applicazione è quello aerospaziale e della difesa, nel quale l'EKF viene utilizzato principalmente per sistemi di tracciamento e navigazione, permettendo a missili e veicoli spaziali di mantenere una traiettoria precisa e di correggere eventuali deviazioni durante il movimento.

Per impiegare questo strumento è utilizzato il blocco "Extended Kalman Filter" presente su Simulink. Per la configurazione nel blocco, per scelta delle matrici di covarianza delle misurazioni e per l'elaborazione di tutti i segnali in ingresso al blocco si rimanda a [16], mentre nelle pagine seguenti è spiegato dettagliatamente tutto il processo di ottimizzazione della matrice di covarianza di stato del sistema (cov_{st} presente in Figura 20).

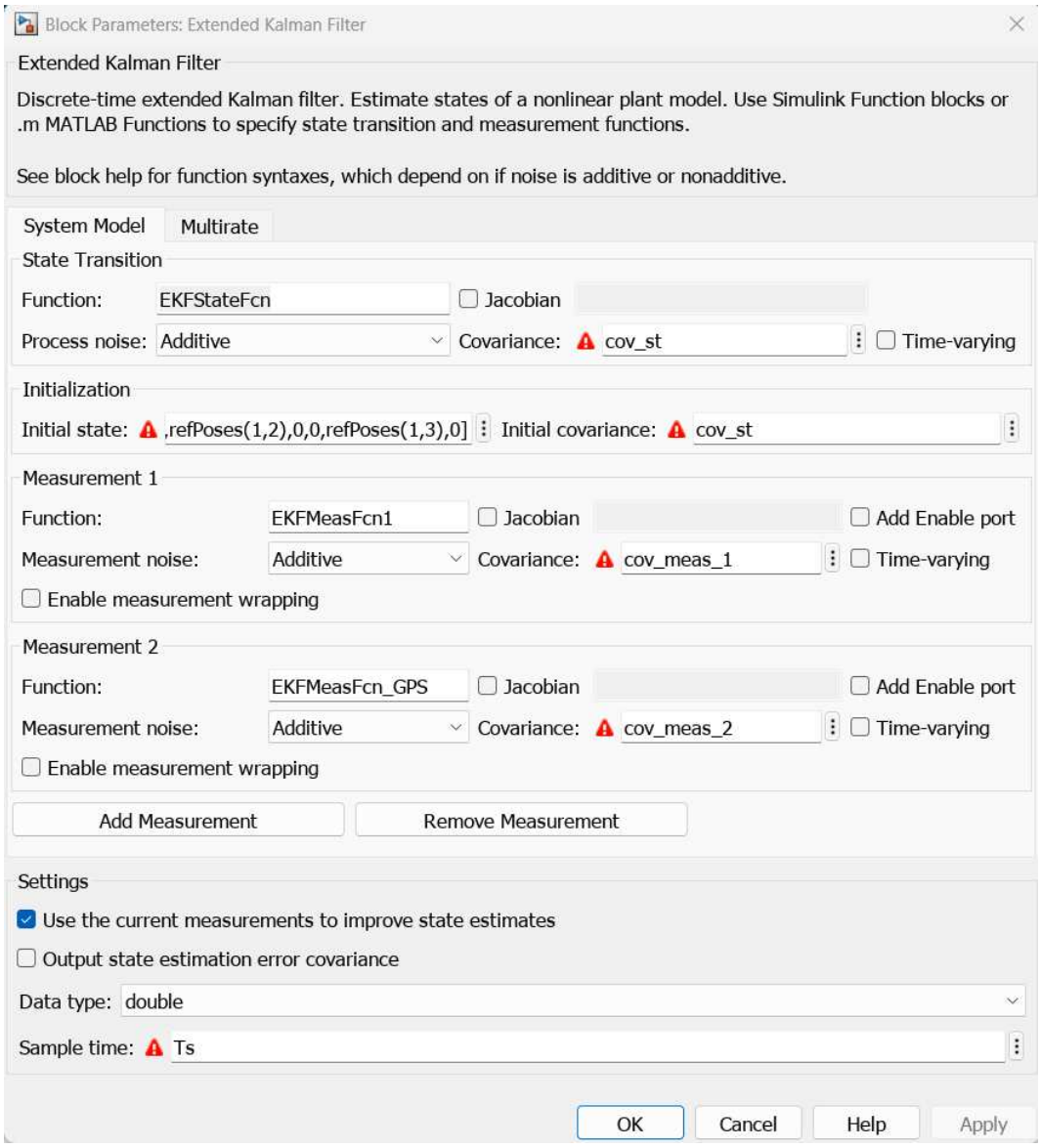


Figura 20: Blocco 'Extended Kalman Filter' di Simulink

4.2 Ottimizzazione della matrice di covarianza

Per diminuire l'errore dovuto alla stima del filtro è stato necessario ottimizzare la matrice di covarianza del filtro di Kalman. Per fare ciò è stato implementato uno script utilizzando un Algoritmo Genetico (GA), tipo di algoritmo che segue i principi della selezione naturale e dell'evoluzione biologica per ottimizzare grandezze. In particolare, viene creata una popolazione generica di soluzioni, si valuta ogni soluzione selezionando quelle migliori per la funzione di costo e le si fanno combinare tra loro per generare ulteriori soluzioni; si ripete questo processo finché non ci si avvicina alla soluzione ottimale.

Ogni probabile soluzione, ovvero ogni individuo della popolazione, è composto da 6 geni, ovvero 6 variabili da ottimizzare: x, y, v_x, v_y, Ψ, Ψ̇. che in questa applicazione rappresentano rispettivamente le 2 posizioni x e y, le velocità v_x e v_y, l'angolo di imbardata e la velocità angolare.
In questa tesi la funzione obiettivo (o funzione di costo) da ottimizzare è rappresentata dalla distanza tra la posizione stimata del veicolo e un punto della traiettoria preso come riferimento. L'ottimizzazione agisce sui valori di covarianza associati ai sei geni che costituiscono ogni individuo della popolazione. Il valore assegnato a ogni covarianza determina il peso relativo attribuito al modello rispetto alle misure provenienti dai sensori: una covarianza di valore ridotto esprime una maggiore fiducia nel modello a danno di quanto osservano i sensori. Al contrario, un valore di covarianza elevato indica minore fiducia nel modello e porta un maggiore utilizzo delle informazioni raccolte dai sensori, nonostante esse siano affette inevitabilmente da errori e incertezze.
L'obiettivo del processo di ottimizzazione è quindi quello di trovare la configurazione ottimale dei sei valori di covarianza associati ai sei geni, in modo da ottenere il migliore compromesso tra affidabilità del modello e le misure provenienti dai sensori. La soluzione ottimale è quella che minimizza la funzione di costo, garantendo una stima dello stato del veicolo il più possibile accurata e robusta.

Il processo di ottimizzazione è affidato a un algoritmo genetico, che opera su una popolazione di soluzioni candidate e la fa evolvere per generazioni successive attraverso operatori di selezione, ricombinazione e mutazione. Ciascun individuo è rappresentato da un vettore di $n_v = 6$ variabili di ottimizzazione $x = (x_1, \dots, x_6)$, vincolate a un intervallo di ammissibilità:

$$x_{lb} \leq x_i \leq x_{ub}, \quad i = 1, \dots, 6 \qquad \textbf{(3)}$$

con $x_{lb} = 8 \cdot 10^{-4}$ e $x_{ub} = 100$ uguali per tutte le variabili. Il punto di partenza della ricerca è ottenuto perturbando casualmente il valore unitario di ciascuna variabile:

$$x_i(0) = 0.5 + u_i, \quad u_i \sim U(0,1) \tag{4}$$

dove u_i è una realizzazione di una variabile aleatoria uniforme in $(0, 1)$, per cui ogni variabile iniziale risulta compresa tra 0.5 e 1.5.

I parametri che governano l'evoluzione della popolazione sono riassunti in Tabella 1. La popolazione conta $N_{pop} = 60$ individui: una dimensione maggiore favorisce l'esplorazione dello spazio di ricerca a scapito del costo computazionale, poiché a ogni generazione occorre valutare la funzione obiettivo per ogni individuo. Il tasso di mutazione $p_m = 0.05$ è la probabilità con cui ciascun gene subisce una modifica casuale nel passaggio tra generazioni e mantiene la diversità genetica evitando la convergenza prematura verso minimi locali. Infine, con $n_e = 2$ si applica una strategia di elitismo: i due migliori individui di ogni generazione sono copiati invariati nella successiva, garantendo che il miglior valore della funzione obiettivo non peggiori nel corso delle iterazioni.

Tabella 1: Parametri di configurazione dell'algoritmo genetico

Parametro	**Simbolo**	**Valore**	**Descrizione**
Dimensione popolazione	N_{pop}	60	Numero di individui (soluzioni candidate) per generazione
Tasso di mutazione	p_m	0.05	Probabilità che un singolo gene subisca una modifica casuale
Individui elitari	n_e	2	Migliori individui trasferiti invariati alla generazione successiva
Numero di variabili	n_v	6	Dimensione del vettore delle variabili di ottimizzazione
Limite inferiore	x_{lb}	$8 \cdot 10^{-4}$	Valore minimo ammesso per ciascuna variabile
Limite superiore	x_{ub}	100	Valore massimo ammesso per ciascuna variabile
Punto iniziale	x_i	$0.5 + u_i$	Estrazione uniforme in [0.5, 1.5] per ogni variabile

Il codice Matlab relativo all'ottimizzazione è riportato in appendice A.

E’ importante specificare che i parametri presentati in Tabella 1 costituiscono l’impostazione finale del processo di ottimizzazione. Tali valori sono stati definiti a seguito di una fase di taratura iterativa, con l’obiettivo di identificare la configurazione ottimale per minimizzare l’errore tra la posizione stimata dal veicolo e il punto di riferimento.

Le opzioni dell’algoritmo genetico sono definite mediante la funzione *optimoptions* di MATLAB (Listing A.3) e successivamente fornite alla funzione ga, che esegue l’ottimizzazione. Le impostazioni adottate sono riassunte in Tabella 2; tutte le altre opzioni mantengono il valore predefinito.

Tabella 2: Opzioni dell’algoritmo genetico

Opzione	Valore	Ruolo
CreationFcn	@gacreationuniform	Generazione della popolazione iniziale
SelectionFcn	@selectionroulette	Selezione dei genitori (roulette)
MutationFcn	{@mutationadaptfeasible, mutationRate}	Mutazione adattiva nel rispetto dei limiti
PopulationSize	populationSize (60)	Numero di individui per generazione
EliteCount	eliteCount (2)	Individui migliori conservati
PlotFcn	@gaplotscorediversity, @gaplotbestf, @gaplotstopping	Monitoraggio in tempo reale
FunctionTolerance	1e-5	Criterio di arresto per convergenza
MaxGenerations	60	Numero massimo di generazioni
InitialPopulationMatrix	factor0	Individuo di partenza personalizzato

La popolazione iniziale è generata da *gacreationuniform*, che estrae gli individui con distribuzione uniforme all’interno dei limiti delle variabili. Poiché l’opzione

InitialPopulationMatrix contiene un solo individuo (factor0), questo costituisce il punto di partenza personalizzato descritto in precedenza, mentre i restanti individui sono creati dalla funzione di creazione.

La selezione dei genitori avviene con il metodo della roulette (*selectionroulette*): a ogni individuo è associata una probabilità di essere scelto proporzionale al proprio valore atteso scalato s_i, in modo da favorire le soluzioni migliori senza escludere le peggiori. Trattandosi di un problema di minimizzazione, la scalatura predefinita per rango assegna $s_i = 1/\sqrt{r_i}$, con r_i posizione in classifica dell'individuo ($r_i = 1$ per il migliore). La probabilità di selezione risulta quindi:

$$p_i = \frac{s_i}{\sum_{j=1}^{N_{pop}} s_j}, \quad s_i = \frac{1}{\sqrt{r_i}} \tag{5}$$

dove N_{pop} è la dimensione della popolazione. Per la mutazione si usa mutationadaptfeasible, che genera perturbazioni adattive, la cui direzione e ampiezza dipendono dall'esito delle generazioni precedenti, e garantisce il rispetto dei limiti. Il parametro mutationRate fissa la probabilità di mutazione di ciascun gene ed è stato impostato a un valore basso (0.05). Dimensione della popolazione ed elitismo hanno i valori di Tabella 1: 60 individui, di cui i 2 migliori conservati a ogni generazione.

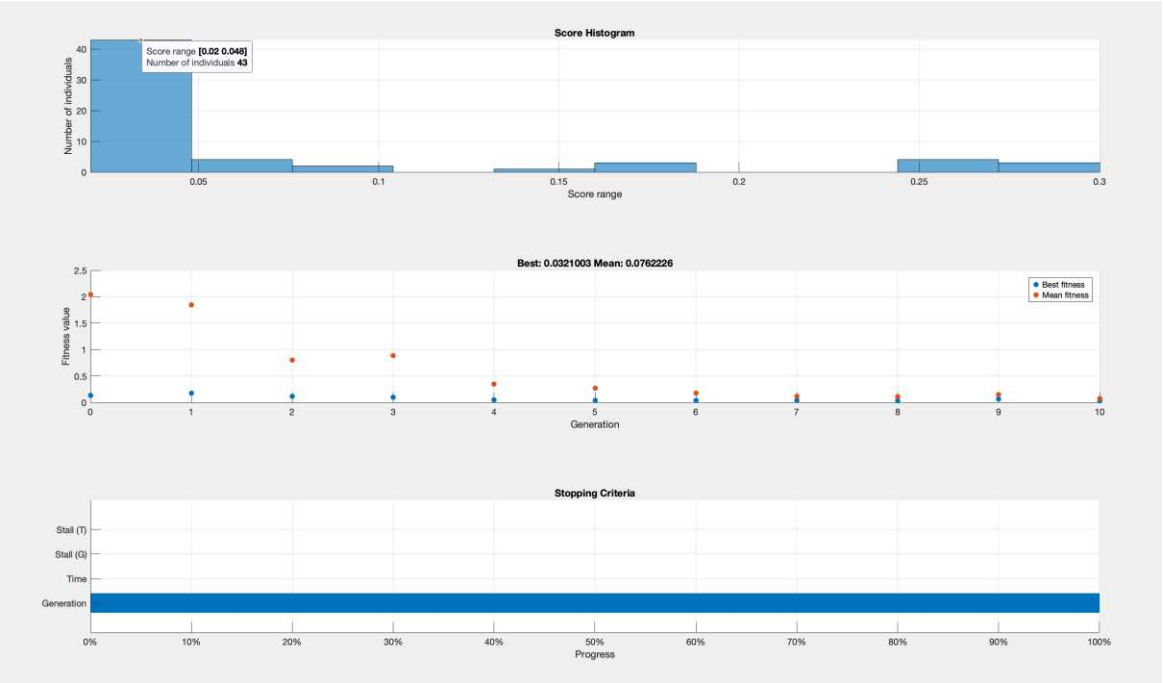


Figura 21: *Grafici che rappresentano l'andamento dell'algoritmo genetico*

Come illustrato nel secondo grafico di figura 23, l'utilizzo dell'algoritmo genetico consente di ridurre progressivamente l'errore tra la posizione del veicolo e il punto di riferimento della traiettoria (che costituisce la funzione obiettivo della simulazione). Con il susseguirsi delle generazioni si osserva una costante diminuzione sia dell'errore minimo sia del valore medio, calcolato per ciascuna generazione.
Si specifica, per una migliore visualizzazione dei grafici, che la figura 23 fa riferimento a una simulazione effettuata con soli 10 generazioni, a differenza di quelle eseguite per quest'attività di tesi, che invece prevedono 60 generazioni per ogni simulazione.

Al fine di validare i risultati ottenuti dal processo di ottimizzazione è stato realizzato un ulteriore script dedicato alla verifica e all'analisi dell'errore. Il codice completo è presente per intero nell'Appendice A con il nome 'Verifica parametri ottimizzazione. Nello specifico il codice confronta il valore dell'errore minimo teorico stimato in fase di ottimizzazione con un valore di errore calcolato a posteriori applicando direttamente le sei covarianze appena ottimizzate. La bontà dello script di ottimizzazione è confermata se lo scarto tra le due grandezze è inferiore al valore di tolleranza pari a 10^{-2}.
Si precisa che questo script di controllo è stato realizzato unicamente con funzione di conferma, concepito quindi come test di robustezza finale. Di conseguenza, in presenza di un'ottimizzazione eseguita correttamente, il superamento della verifica rappresenta quanto atteso.

4.3 Determinazione della funzione di costo dell'ottimizzazione

È necessario soffermarsi sulla funzione di costo del processo di ottimizzazione, ovvero sulla distanza tra il punto in cui si trova il veicolo e una traiettoria di riferimento. Questa distanza è stata calcolata nel seguente modo:

(6)

$\alpha = 0.4;$
$distance = \sqrt{((xhat.Data(:,1) - Var_trajectory(:,1))^2 + ((xhat.Data(:,2) - Var_trajectory(:,2))^2};$
$distance_{rms} = rms\ (distance(500{:}end);$
$distance_{max} = max(distance(500);$
$err = \alpha \cdot distance_{rms} + (1-\alpha) \cdot distance_{max};$

L'algoritmo calcola inizialmente il vettore *distance,* i cui elementi rappresentano gli scostamenti spaziali (distanza euclidea) tra la posizione stimata e la traiettoria di riferimento per ciascuna campione. Un algoritmo di ottimizzazione, però, non richiede come funzione di costo un vettore ma un unico valore scalare. Per questo, è necessario combinare due indicatori: l'errore a regime e l'errore massimo. In particolare:

- errore a regime ($distance_{rms}$): rappresenta la radice media quadratica dell'errore ed è impiegato per valutare l'accuratezza globale e la stabilità nel complesso. Nel calcolo sono stati volutamente esclusi i primi 500 campioni della simulazione, scelta necessaria per non includere nel calcolo le dinamiche del transitorio iniziale, fase in cui l'errore è fisiologicamente elevato a causa delle notevoli tempistiche richieste dall'algoritmo per convergere dalle condizioni iniziali. L'inclusione di questi campioni altererebbe la media, facendola dominare dai picchi del transitorio.
- Errore massimo ($distance_{max}$): rappresenta il valore di picco assoluto dell'errore lungo tutta la simulazione (compresi i primi 500 campioni esclusi in precedenza). Esso ha il compito di monitorare le deviazioni istantanee più critiche, legate a scostamenti puntuali che potrebbero compromettere la stabilità del sistema.

La funzione di costo complessiva viene definita da una combinazione lineare convessa dei due contributi mediante il parametro di peso alpha. La scelta di combinare questi due contenuti è dovuta dall'esigenza di evitare i limiti di ciascuno fattore; un'ottimizzazione basata unicamente sull'errore RMS ignorerebbe la presenza di picchi di errore istantanei che, anche se brevi, potrebbero essere importanti in un contesto reale. Al contrario, utilizzare soltanto l'errore massimo renderebbe la funzione di costo molto sensibile a un singolo picco, anche presente nel transitorio iniziale, trascurando la qualità complessiva del processo a regime.

Assegnare un peso del 40% all'errore di regime e un 60% all'errore massimo rappresenta il compromesso ottimale: garantisce infatti un'adeguata penalizzazione dei picchi ma senza sacrificare la precisione media.

4.4 Determinazione punto di riferimento e calcolo errori

Una volta completata la fase di ottimizzazione delle matrici di covarianza, si procede alla determinazione del punto di riferimento sulla traiettoria desiderata. Si ricorda, infatti, che tale ottimizzazione è stata condotta con l'obiettivo di far seguire al Losi un percorso predefinito. Per fare ciò, risulta quindi necessario quantificare lo scostamento spaziale e orientativo, calcolando nello specifico l'errore laterale (cross-track error) e l'errore di orientamento (heading error), errori necessari alla determinazione dei comandi di sterzo e velocità da applicare al veicolo. Per poter calcolare questi errori è necessario individuare un punto di riferimento sulla traiettoria.

Come punto di riferimento è stato selezionato il punto a distanza minima rispetto alla posizione stimata del veicolo. La procedura prevede il calcolo della distanza euclidea tra la posizione attuale del veicolo e ciascun punto appartenente alla traiettoria di riferimento, individuando successivamente quello che minimizza tale distanza.

Una valida alternativa a quanto appena descritto è rappresentata dall'introduzione della tecnica del lookahead (o punto di mira avanzato), strategia di guida e di controllo molto utilizzata e consolidata nell'ambito della guida autonoma e dei sistemi di navigazione assistita.

Questa tecnica prevede una prima fase analoga a quella precedentemente descritta, con lo scopo di individuare il punto della traiettoria che minimizza la distanza euclidea rispetto alla posizione stimata del veicolo. Tuttavia, questo punto non viene utilizzato come riferimento per il calcolo degli errori; il riferimento, infatti, viene traslato più avanti lungo il percorso di una determinata quantità. Essendo infatti la traiettoria rappresentata da una sequenza discreta di punti identificati da un indice numerico progressivo, l'implementazione pratica di questa tecnica si traduce nella selezione del punto di riferimento caratterizzato da un indice incrementato di n posizioni rispetto al punto a distanza minima. Il parametro n viene opportunamente dimensionato in funzione della grandezza della traiettoria e della cinematica del veicolo.

I principali benefici di questo approccio si vedono nei tratti a curvatura variabile, in particolare nei punti in cui si susseguono un tratto rettilineo e uno curvo o in presenza di inversioni di angolo di sterzo. Puntare a un riferimento geometrico avanzato consente infatti al sistema di anticipare la variazione di curvatura avviando l'azione di sterzatura prima dell'effettivo ingresso in curva. Ciò permette di ridurre l'inerzia di risposta del controllore e di minimizzare l'errore di cross-track evitando il ritardo di inseguimento.

Una volta determinato il punto di riferimento, il sistema passa all'ultima fase dell'architettura di controllo in closed-loop: il calcolo del cross track error (e_{ct}) e dell'heading error (e_{Ψ}), calcolati utilizzando le seguenti formule:

$$e_{ct} = \big((Y_r - Y_{Losi}) \cdot \cos\Psi_r\big) - (X_r - X_{Losi}) \cdot \sin\Psi_r \quad \textbf{(7)}$$

$$e_{\Psi} = \Psi_R - \Psi_{Losi} \quad \textbf{(8)}$$

Dove le grandezze con i pedici R rappresentano le coordinate e l'angolo di imbardata del punto di riferimento sulla traiettoria, mentre quelle con il pedice Losi rappresentano coordinate e angolo di imbardata del punto in cui è posizionato il veicolo (si veda [16] per maggiori dettagli).

Tali grandezze sono fondamentali per la generazione dinamica dei comandi di sterzo e di velocità necessari a correggere la traiettoria del veicolo e garantire il corretto inseguimento del percorso.

Per quanto concerne l'attuazione di questi comandi, la modalità di comunicazione verso il veicolo si mantiene del tutto analoga a quanto illustrato nel capitolo 2; vengono infatti impiegati i medesimi blocchi funzionali offerti dalla libreria I/O 397. La differenza risiede ovviamente nell'origine dei segnali di pilotaggio: mentre in precedenza i comandi venivano forniti in modalità manuale tramite Joystick, in questo caso essi derivano direttamente dall'elaborazione in tempo reale del controllo ad anello chiuso.

Nell'Appendice A viene riportato uno script dimostrativo che illustra l'applicazione pratica della tecnica del lookahead, impostata nello specifico con un indice pari a 10. Sempre all'interno dell'Appendice A è inoltre riportato lo script utilizzato per la generazione di una traiettoria di esempio, con lo scopo di mostrare la struttura di una traiettoria definita come un insieme discreto di punti descritti da un indice progressivo. Per facilitare la comprensione della geometria del percorso ottenuto, la traiettoria realizzata è mostrata in Figura 24.

5. Conclusioni e applicazioni future

5.1 Analisi dei risultati

Il lavoro svolto ha permesso di caratterizzare e implementare due modalità di controllo indipendenti sulla piattaforma sperimentale Losi, ciascuna con caratteristiche specifiche e un proprio percorso di validazione.

In particolare, la prima modalità, incentrata sul controllo manuale tramite joystick, ha dimostrato la validità del protocollo di comunicazione UDP, resosi necessario per superare i limiti di compatibilità tra Simulink real-time e i dispositivi di input manuale. L'attività ha inoltre confermato l'efficacia dei blocchi della libreria I/O per l'attuazione dei segnali in tempo reale, nonché la robustezza della logica di gestione del verso di marcia nel prevenire sollecitazioni meccaniche indesiderate sul motore.

Per quanto riguarda la modalità di controllo ad anello chiuso, l'analisi si è concentrata sul processo di calibrazione e ottimizzazione. In particolare, il lavoro sull'ottimizzazione della matrice di covarianza all'interno del filtro di Kalman Esteso (EKF) ha consentito di mitigare significativamente l'impatto del rumore introdotto dall'unità GPS-IMU, garantendo una stima di stato solida anche in presenza di misure non ideali. Parallelamente, la realizzazione della funzione di costo ha permesso di bilanciare in modo rigoroso la risposta del veicolo nel calcolo dell'angolo di sterzo e della velocità, partendo dalla corretta determinazione del punto di riferimento e dalla valutazione del cross-track error e dell'heading error.

Complessivamente, i risultati ottenuti evidenziano la solidità delle scelte progettuali effettuate, sia per quanto riguarda l'acquisizione e l'elaborazione dei segnali, sia per la sintesi degli algoritmi di stima e di controllo.

5.2 Considerazioni conclusive

L'obiettivo principale della tesi, ovvero la progettazione e la presentazione delle due differenti modalità di controllo sulla piattaforma Speedgoat è stato pienamente conseguito. La struttura modulare sviluppata mostra come le stesse componentistiche hardware possano servire due logiche di controllo completamente diverse tra loro. Da un lato, il sistema è in grado di rispondere direttamente ai comandi impartiti da un operatore tramite joystick;

dall'altro, la stessa infrastruttura può essere utilizzata per un sistema di controllo ad anello chiuso completamente autonomo, in cui l'azione di guida è determinata da una traiettoria decisa in precedenza, dall'elaborazione degli errori di inseguimento e da algoritmi ottimizzati.
Più in generale, il lavoro svolto conferma quanto affermato in apertura di tesi, ovvero come i veicoli in scala rappresentino uno strumento efficace, economico e sicuro per lo sviluppo e la validazione di logiche di controllo, ponendosi come un valido banco di prova anche in ottica di applicazioni sulla guida autonoma di veicoli a grandezza reale.

5.3 Possibili sviluppi e applicazioni future

A partire dall'hardware e dalle architetture di controllo presentate in questo lavoro di tesi si individuano interessanti spunti per sviluppi e per future attività di ricerca:

- relativamente al controllo manuale, un naturale sviluppo consiste nell'affiancare all'unità GPS-IMU ulteriori sensori di percezione, come per esempio radar o moduli fotocamera per lo streaming video. Tale upgrade permetterebbe di estendere l'impiego del controllo manuale ad ambienti remoti, non mappati e ad alto rischio per l'incolumità umana: in tali contesti, la piattaforma Losi, pur di dimensioni non trascurabili, risulterebbe comunque meno ingombrante e più sicura da impiegare rispetto a un operatore umano, costituendo un importante vantaggio operativo.
- Per quanto concerne la guida autonoma, invece, le attività future potranno concentrarsi sull'evoluzione della tecnica di ricerca del punto di riferimento, approfondendo così la strategia di lookahead presentata in questa tesi. Un esempio potrebbe essere la progettazione di un lookahead dinamico e adattativo, capace di modificare la distanza di anticipo in funzione della velocità istantanea del veicolo o della curvatura locale del tracciato. Questo permetterebbe di ottimizzare la stabilità e la precisione di guida anche su tratte complesse a geometria variabile, come curve a S o cambi improvvisi di traiettoria.

Lista delle figure

Figura 1: Configurazione iniziale del Losi 5ive T 2.0 7
Figura 2: Motore elettrico RS PRO Brushless 8
Figura 3: Batteria a litio 7S7P 9
Figura 4: Servomotore di sterzo S900S 10
Figura 5: Controller ESCON 50/5 12
Figura 6: Modulo CANmod.gps 13
Figura 7: Macchina bersaglio in tempo reale Speedgoat 15
Figura 8: Barra degli strumenti Speedgoat 16
Figura 9: Logitech F710 Wireless Gamepad 18
Figura 10: Blocco 'Joystick Input' di Simulink 19
Figura 11: Blocco 'UDP send' di Simulink 22
Figura 12: Blocco 'UDP receive' di Simulink 23
Figura 13: Blocco 'Setup' per IO397 di Simulink 25
Figura 14: Blocco 'PWM generation (5) di Simulink 26
Figura 15: Blocco 'Digital Output' di Simulink 27
Figura 16: Blocco 'Analog Input' di Simulink 29
Figura 17: Blocco 'Lookup Table (n-D) di Simulink 30
Figura 18: Schema logico del modello Simulink eseguito in locale 31
Figura 19: Schema logico del modello Simulink eseguito in real-time 32
Figura 20: Blocco 'Extended Filter Kalman' di Simulink 36
Figura 21: Grafici che rappresentano l'andamento dell'algoritmo genetico 40
Figura 22: Traiettoria sviluppata su MATLAB 59

Lista delle tabelle

Tabella 1: Parametri di configurazione dell'algoritmo genetico 38
Tabella 2: Opzioni dell'algoritmo genetico 39

Bibliografia

[1] AtechItalia, “Guida autonoma: La rivoluzione tecnologica del futuro”, https://atechitalia.com/impatto-della-guida-autonoma-sul-futuro-delle-auto/

[2] Ford, “Guida Autonoma – cos’è e come funziona”, https://www.ford.it/tecnologia/autonomous-driving

[3] J. Ji, A. Khajepour, W. Melek, and Y. Huang, "Path planning and tracking for vehicle collision avoidance based on model predictive control with multiconstraints," *IEEE Transactions on Intelligent Transportation Systems*, vol. 18, no. 6, pp. 1320–1332, June 2017.

[4] C. A. Micro, D. Brennan, and A. A. Malikopoulos, "A scaled smart city for experimental validation of connected and automated vehicles," *IEEE Transactions on Intelligent Vehicles*, vol. 4, no. 3, pp. 320–331, Sept. 2019.

[5] M.K. O’Kelly, H. Abbas, and R. Mangharam, "F1TENTH: An open-source evaluation platform for autonomous racing," in *Proc. IEEE International Conference on Robotics and Automation (ICRA)*, Paris, France, 2020, pp. 8920–8926.

[6] H. Okuda, Y. Tsurumine, and T. Suzuki, "Development of 1/10 scaled autonomous vehicle testbed for trajectory control and validation," *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 12450–12455, 2017.

[7] W.T. Zheng, R. K. He, and Y. L. Chen, "Scaled vehicle platforms for autonomous driving research: A comprehensive review," *IEEE Access*, vol. 9, pp. 11245–11258, 2021.

[8] B. Barshan and H. F. Durrant-Whyte, "Inertial navigation system for mobile robots," *IEEE Transactions on Robotics and Automation*, vol. 11, no. 3, pp. 328–342, June 1995.

[9] R. Rajamani, *Vehicle Dynamics and Control*, 2nd ed. New York, NY, USA: Springer, 2012.

[10] G.M. Coulter and C. A. MacAdam, "Implementation and testing of a look-ahead driver model for autonomous vehicle guidance," *Vehicle System Dynamics*, vol. 31, no. 5, pp. 385–402, 1999

[11] P. Falcone, F. Borrelli, J. Asgari, H. E. Tseng, and D. Hrovat, "Predictive active steering control for autonomous vehicle path following," *IEEE Transactions on Control Systems Technology*, vol. 15, no. 3, pp. 566–580, May 2007.

[12] S. S. Kia, S. F. Roundy, and F. E. Martinez, "Optimal state estimation for autonomous vehicle localization using Extended Kalman Filter," *IEEE Control Systems Letters*, vol. 5, no. 4, pp. 1180–1185, Oct. 2021.

[13] D. Loebis, R. Sutton, and J. Chudley, "Adaptive tuning of a Kalman filter using genetic algorithms for autonomous underwater vehicle navigation," *Control Engineering Practice*, vol. 12, no. 8, pp. 931–939, 2004.

[14] Motore elettrico RS PRO Brushless, "Data Sheet RS PRO Brushless" https://docs.rs-online.com/5a4a/0900766b81580e4b.pdf

[15] Shuning, Liu *"Control Development and Experimental Validation of a 1:5 Scaled Vehicle"* [Tesi di laurea Magistrale, Politecnico di Torino, 2024]

[16] Andrea, Pinter *"State estimation and path following control of a 1:5 scaled vehicle"* [Tesi di laurea Magistrale, Politecnico di Torino, 2024]

[17] Losi, "Manuale Losi 5ive T 2.0" https://www.losi.com/product/1-5-5ive-t-2.0-v2-4x4-bnd-gas-short-course-truck-blue-/LOS05014V2T1.html#

[18] SBG System, "EKF – filtro di Kalman Esteso" https://www.sbg-systems.com/it/glossary/extended-kalman-filter-ekf/

[19] Cordis Europa, "Sviluppare solidi veicoli a guida autonoma in grado di percorrere lunghe distanze", https://cordis.europa.eu/article/id/457148-developing-robust-self-driving-vehicles-that-can-go-the-distance/it

[20] L.E Aguilar, C. A. Micro, D. Brennan, and A. A. Malikopoulos, "A scaled electric vehicle platform for testing autonomous driving algorithms," *IEEE Transactions on Vehicular Technology*, vol. 72, no. 4, pp. 4310–4322, 2023.

APPENDICE A

- SCRIPT 1: '*Controllo direzione*'

Prima parte della funzione per il controllo del verso della velocità di rotazione nel controllo tramite Joystick.

```
function Sel  = controllo_direzione(velocita, RPM)
%
% INPUT:
%   Velocita  - Velocità  desiderata, da Joystick[km/h]
%   RPM       - Velocità angolare misurata in tempo reale [rpm]
%
% OUTPUT:
%   Sel       - Direzione: 0 = fermo, 1 = avanti, 2 = indietro

% Necessario inizializzare questa variabile per andare avanti
persistent currentSel
if isempty(currentSel)
    currentSel = 0; % Veicolo inizialmente fermo
end

speed_tol = 10; %[RPM]

% Inversione sicura: aspetta che la velocità si annulli prima di invertire
if abs(RPM) < speed_tol   % Ora è sicuro cambiare direzione
    if velocita>0
        currentSel = 1;  % avanti
    elseif velocita<0
        currentSel = 2;  % indietro
    else
        currentSel = 0;  % fermo
    end
else
    % Se direzione desiderata è opposta a quella attuale → rallenta
    if (currentSel == 1&&velocita<0) || (currentSel == 2&&velocita>0)
        currentSel = 0;  % imposta fermo per consentire rallentamento
    end
end

% Output della direzione
Sel = currentSel;
end
```

- SCRIPT 2: '*Attivazione digital output'*

Seconda parte della funzione per il controllo del verso della velocità di rotazione nel controllo tramite Joystick.

```
function [marcia_avanti, retromarcia] = attivo_digital_output(sel)

  persistent avanti indietro
  if isempty(avanti)
     avanti=2;
  end
  if isempty (indietro)
     indietro=2;
  end

  marcia_avanti = avanti;
  retromarcia = indietro;

  switch sel
      case 1
          marcia_avanti = 1;
          retromarcia = -1;
      case 2
          marcia_avanti = -1;
          retromarcia = 1;
      case 0
          marcia_avanti = -1;
          retromarcia = -1;
  end
end
```

- SCRIPT 3: '*Ottimizzazione matrice di covarianza'*

Script per l'ottimizzazione della matrice di covarianza dell'EKF

```
% Dati del veicolo
eval(['run Data_txt' filesep 'data_vehicle'])
Ts = 0.005; %sample time
% ----------------------- Plot ------------------------
F_Size = 14;
plotcol = {'-k','--r','-.b',':g','.m','-k'};

% Funzioni ottimizzazione
populationSize = 60;
mutationRate = 0.05;
eliteCount = 2;

factor0 = [0.5+rand,0.5+rand,0.5+rand,0.5+rand,0.5+rand,0.5+rand];
lowb = [0.8e-3, 0.8e-3, 0.8e-3, 0.8e-3, 0.8e-3, 0.8e-3];
upb = [100, 100, 100, 100, 100, 100];

options = optimoptions('ga', ...
    'CreationFcn', @gacreationuniform, ...
    'SelectionFcn', @selectionroulette, ...
    'MutationFcn', {@mutationadaptfeasible, mutationRate}, ...
    'PopulationSize', populationSize, ...
    'EliteCount', eliteCount, ...
    "PlotFcn",{@gaplotscorediversity,@gaplotbestf,@gaplotstopping}, ...
    "FunctionTolerance",1e-5, ...
    "MaxGenerations",60, ...
    "InitialPopulationMatrix",factor0);
counter = 1;
factor = ga(@error,6,[],[],[],[],lowb,upb,[],options);

function err = error(x)
    clear xhat Var_trajectory distance
    model_name = 'Kin_Filter_model.slx';

    count = evalin("base","counter");
    assignin("base","counter",count)
    if count == 1
        randomNumber = randi([1, 30000]);
        assignin("base","seed",randomNumber)
    elseif count == 60
        count = 0;
    else
    end
    seed = evalin("base","seed");
    assignin("base","seed",seed);
    covariance0 = [1e-5, 1e-5, 1e-7, 1e-7, 1e-5, 1e-5];
```

```
    cov_st_x = x(1)*covariance0(1);
    cov_st_y = x(2)*covariance0(2);
    cov_st_vx = x(3)*covariance0(3);
    cov_st_vy = x(4)*covariance0(4);
    cov_st_psi = x(5)*covariance0(5);
    cov_st_psi_dot = x(6)*covariance0(6);
    cov_meas_1_vx = 0.1;
    cov_meas_1_psi_dot = 1e-2;
    cov_meas_1_psi = 1;
    cov_meas_2_x = 4;
    cov_meas_2_y = 4;

    cov_st =
diag([cov_st_x,cov_st_y,cov_st_vx,cov_st_vy,cov_st_psi,cov_st_psi_dot]);
    cov_meas_1 = diag([cov_meas_1_vx,cov_meas_1_psi_dot,cov_meas_1_psi]);
    cov_meas_2 = [cov_meas_2_x,0;0,cov_meas_2_y];

    assignin("base","cov_st",cov_st)
    assignin("base","cov_meas_1",cov_meas_1)
    assignin("base","cov_meas_2",cov_meas_2)

    % Run
    disp('working ...')
    disp(seed)
    disp(count)
    string_sim = strcat({'sim '},{model_name});
    eval(string_sim{1});
    count = count+1;
    assignin("base","counter",count)

    % Calcolo errore
    alpha = 0.4;
    distance = sqrt((xhat.Data(:,1)-Var_trajectory(:,1)).^2 +
(xhat.Data(:,2)-Var_trajectory(:,2)).^2);
    distance_rms = rms(distance(500:end));
    distance_max = max(distance);
    err = alpha*distance_rms + (1-alpha)*distance_max;

    % Salva x e err in un file o variabile
    results_path = 'results_log_ottimizzazione11.mat';

    if isfile(results_path)
        load(results_path, 'log_x', 'log_err');
    else
        log_x = [];
        log_err = [];
    end
    log_x = [log_x; x];
```

```
    log_err = [log_err; err];

    [min_err, idx_min] = min(log_err);
    x_best = log_x(idx_min, :);
    cov_st_best = diag(x_best .* covariance0);
    cov_st_last = cov_st;
    mean_err = mean(log_err);

    save(results_path, 'log_x', 'log_err', 'min_err', 'x_best',
'cov_st_best','cov_st_last','mean_err');
    save('workspace_finale.mat');
 end
```

- SCRIPT 4: '*Verifica parametri ottimizzazione*'

Script che permette di verificare se i risultati ottenuti nell'ottimizzazione sono coerenti con l'errore.

```
%  Script di verifica dei parametri ottimali di GA, Alex d'Alfonso
clear; clc; close all
disp('=== Verifica parametri ottimali della GA ===');

% 1. Caricamento dei dati di simulazione
eval(['run Data_txt' filesep 'data_vehicle'])

% 2. Caricamento risultati ottimizzazione
results_path = 'results_ottimizzazione.mat';
if ~isfile(results_path)
    error('File dei risultati non trovato: %s', results_path);
end

load(results_path, 'x_best', 'min_err', 'cov_st_best');

disp('Parametri ottimali caricati:');
disp(x_best);
fprintf('Errore minimo registrato dalla GA: %.6f\n', min_err);

% 3. Impostazioni di base
model_name = 'Kin_Filter_model.slx';
Ts = 0.005;

covariance0 = [1, 1, 1, 1, 1, 1];
scale = 10.^x_best;
cov_st_x       = scale(1) * covariance0(1);
cov_st_y       = scale(2) * covariance0(2);
cov_st_vx      = scale(3) * covariance0(3);
cov_st_vy      = scale(4) * covariance0(4);
cov_st_psi     = scale(5) * covariance0(5);
cov_st_psi_dot = scale(6) * covariance0(6);

cov_meas_1_vx = 0.1;
cov_meas_1_psi_dot = 1e-2;
cov_meas_1_psi = 1;
cov_meas_2_x = 4;
cov_meas_2_y = 4;

cov_st     =
diag([cov_st_x,cov_st_y,cov_st_vx,cov_st_vy,cov_st_psi,cov_st_psi_dot]);
cov_meas_1 = diag([cov_meas_1_vx,cov_meas_1_psi_dot,cov_meas_1_psi]);
cov_meas_2 = [cov_meas_2_x, 0; 0, cov_meas_2_y];

assignin('base','cov_st',cov_st);
```

```
assignin('base','cov_meas_1',cov_meas_1);
assignin('base','cov_meas_2',cov_meas_2);

% 4. Gestione del seed per il rumore
% Se il seed non era già presente nei dati, ne imposto uno di default
if exist('seed','var') ~= 1
    seed = 12345;
end
assignin('base','seed',seed);

% 5. Esecuzione simulazione
disp('Simulazione con parametri ottimali in corso...');
tic
simOut = sim(model_name, 'SrcWorkspace', 'base');
toc
disp('Simulazione completata.');

% 6. Estrazione risultati
% Recupera i segnali (devono avere gli stessi nomi del tuo modello)
try
    xhat = simOut.get('xhat');
    Var_trajectory = simOut.get('Var_trajectory');
catch
    error('Impossibile trovare "xhat" o "Var_trajectory" tra le uscite della
simulazione.');
end

% 7. Calcolo dell'errore come nella GA
alpha = 0.4;
distance = sqrt((xhat.Data(:,1) - Var_trajectory(:,1)).^2 + ...
                (xhat.Data(:,2) - Var_trajectory(:,2)).^2);

if length(distance)>500
    distance_rms = rms(distance(500:end));
else
    distance_rms = rms(distance);
end
distance_max = max(distance);

err_verifica = alpha*distance_rms + (1-alpha)*distance_max;

% 8. Confronto con l'errore minimo della GA
fprintf('\n=== RISULTATI CONFRONTO ===\n');
fprintf('Errore calcolato con x_best : %.6f\n', err_verifica);
fprintf('Errore minimo GA registrato : %.6f\n', min_err);
diff_abs = abs(err_verifica - min_err);
fprintf('Differenza assoluta          : %.6e\n', diff_abs);

if abs(err_verifica - min_err)<1e-2
```

```
    disp('☑ OK Errore coerente con quello calcolato dalla GA.');
else
    disp('⚠ Attenzione: errore differente da quello calcolato con la GA.');
end

% 9. (Opzionale) Salva workspace di verifica
save('workspace_verifica.mat');
disp('Workspace di verifica salvato in "workspace_verifica.mat".');
```

- SCRIPT 5: '*Realizzazione traiettoria'*

Script per la creazione della traiettoria da far seguire al Losi.

```
n_points = 10000; % punti per ogni curva

% Semicerchio verticale sinistro
theta1 = linspace(-pi/2, pi/2, n_points)';
r1 = 6;
X1 = -r1*cos(theta1);
Y1 = r1*sin(theta1);

% Segmento orizzontale lungo 3
X2 = linspace(0,3,n_points)';
Y2 = 6*ones(n_points,1);

% Curva a S raggio 3
theta3 = linspace(pi/2,0,n_points)';
X3 = 3 + 3*cos(theta3);
Y3 = 3 + 3*sin(theta3);
theta4 = linspace(pi,pi/2,n_points)';
X4 = 9 + 3*cos(theta4);
Y4 = 3 - 3*sin(theta4);

% Segmento orizzontale lungo 2
X5 = linspace(9,11,n_points)';
Y5 = zeros(n_points,1);

% Semicerchio verticale
theta6 = linspace(pi/2,-pi/2,n_points)';
X6 = 11 + 3*cos(theta6);
Y6 = -3 + 3*sin(theta6);

% Segmento orizzontale
X7 = linspace(11,0,n_points)';
Y7 = -6*ones(n_points,1);
X_tot = [X1; X2; X3; X4; X5; X6; X7];
Y_tot = [Y1; Y2; Y3; Y4; Y5; Y6; Y7];

% Orientamento della traiettoria
psi_tot = [atan2(diff(Y_tot), diff(X_tot)); 0];

% Salvataggio
refPoses = [X_tot, Y_tot, psi_tot];

% Plot
figure;
plot(X_tot,Y_tot,'LineWidth',2);
hold on;
```

```
plot(0, 6, 'ro', 'MarkerSize', 6, 'MarkerFaceColor', 'r');
axis equal;
grid on;
xlabel('X');
ylabel('Y');
title('Circuito');
```

Si allega l'immagine della traiettoria realizzata con il seguente script.

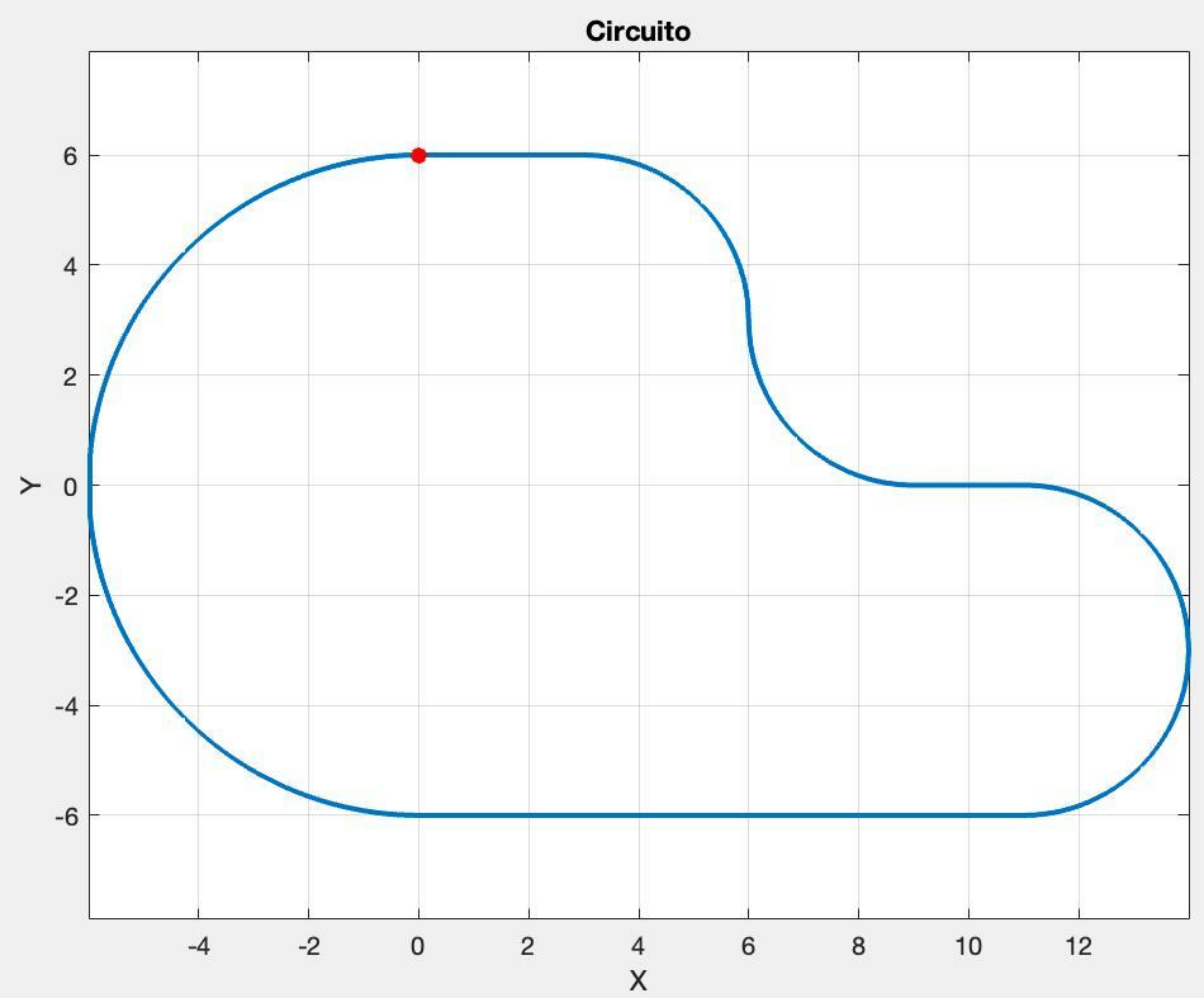


Figura 22: *Traiettoria sviluppata su MATLAB*

- SCRIPT 6: '*Tecnica del lookhahead'*

Funzione realizzata per mostrare l'utilizza della tecnica del lookahead, con indice di lookahead pari a 10.

```
 function [ref_pose,im] =ref_gen(ref_tr, im_prev)
 %INPUT
 %   ref_tr: traiettoria di riferimento [Nx3, le colonne sono x,y,theta]
 %   pose: vettore che contiene la posizione attuale del veicolo [x,y,theta]
 %   im_prev: indice dell'ultimo punto usato come riferimento nell'ultima
iterazione
 %OUTPUT
 %   ref_pose: punto della traiettoria usato come riferimento
 %   im: indice del punto di riferimento

 N=2000;
 lookahead_offset = 10;

 ref_low = max(im_prev-N,1);
 ref_high = min(size(ref_tr,1),im_prev+N);
 ref = ref_tr(ref_low:ref_high,1:2);  %creo una finestra di riferimento,
simmetrica con centro nel punto usato come riferimento
 % point of the reference trajectory closest to the vehicle
 dis=vecnorm(ref'-Posizione_veicolo(1:2)*ones(1,size(ref,1)));  %distanza
euclidea tra posizione attuale del veicolo e ogni punto della finestra

 [~,im]=min(dis);   %punto più vicino nella finestra alla posizione del
veicolo
 ref_mom = ref(im,:);
 index = find(abs(ref_tr(:,1)-ref_mom(1,1))<0.00001 & abs(ref_tr(:,2)-
ref_mom(1,2))<0.00001); %cerco tutti gli indicic nella traiettoria dove x e y
sono quelli del punto più vicino
 index_mom = find(abs(index - im_prev)<1000,1); %tra tutti gli indici, prendo
quello più vicino all'indice del punto di riferimento usato in precedenza
 index_real = index(index_mom,1);

 lookahead_index = min(index_real + lookahead_offset, size(ref_tr,1));

 ref_pose=ref_tr(lookahead_index,1:3)'; %prendo posizione completa
 im = lookahead_index; %aggiorno indice

 end
```