



VAB - Veicolo auto bilanciato

Relazione di progetto

Laboratorio di Sistemi Meccatronici II
Università degli Studi di Bergamo

Kilometro rosso

A.A. 2019/2020

CALEGARI ANDREA - 1041183
PIFFARI MICHELE - 1040658

September 10, 2020

Contents

1	Introduzione	1
2	Dinamica	3
2.1	Scomposizione del VAB	3
2.1.1	Considerazioni iniziali	3
2.1.2	Grandezze di supporto	4
2.1.3	Calcolo componenti dinamiche e potenziali per ogni corpo rigido del sistema . .	4
2.2	Equazioni del moto	8
2.2.1	Lagrangiana	8
2.2.2	Rapporto di trasmissione	10
2.2.3	Sistema non lineare	10
2.2.4	Linearizzazione	11
3	Controllo	15
3.1	Introduzione	15
3.2	Analisi in anello aperto	15
3.3	Controllore	16
3.3.1	Posizionamento dei poli continui	17
3.3.2	Posizionamento dei poli discreti	18
3.3.3	Setpoint di velocità	18
3.4	OPC - UA	19
3.4.1	Idea base OPC-UA	20
3.5	OPC-UA e V.A.B.	20
3.6	Rumore	21
4	Simulink	23
4.1	Introduzione	23
4.2	Simulazione del sistema non lineare	24
4.3	Simulazione del sistema lineare	25
4.4	Motore	26
4.5	Modello complessivo (al momento)	28
4.6	Discretizzazione	29
4.7	Sensori	30
4.8	RTOS - XENOMAI	32
I	Allegati	35
A	Appendice A	37

List of Figures

2.1	Baricentri dei singoli corpi rigidi	4
2.2	Grandezze di supporto	5
2.3	Ragionamento per il calcolo dei contributi dinamici dello chassis	8
3.1	Luogo delle radici del V.A.B. modellizzato in anello aperto	16
3.2	Schema concettuale del controllo ([1])	16
3.3	Root locus del sistema in anello chiuso	18
3.4	chiusura dell'anello di controllo di $\dot{\phi}$ con un controllore I [1]	19
3.5	Taratura del controllore	19
3.6	Schema di massima dell'utilizzo di Raspberry Pi 3	20
3.7	Parametri impostabili lato client	21
3.8	Diagramma rappresentante le funzionalità del server	22
4.1	Risposta del sistema non lineare in anello chiuso	23
4.2	Risposta del sistema lineare in anello chiuso	24
4.3	Implementazione simulink delle equazioni differenziali	24
4.4	Risposta in anello aperto del sistema reale	25
4.5	Implementazione simulink del sistema linearizzato	25
4.6	Risposta in anello aperto del sistema lineare	26
4.7	Schema a blocchi del motore	26
4.8	Transitorio del motore	27
4.9	Zoom del grafico in figura Fig.4.8	27
4.10	Clamp della coppia erogata da parte del motore	28
4.11	Simulink	29
4.12	Modello Simulink del controllore	30
4.13	Modello Simulink dei sensori <i>encoder</i> e <i>I.M.U.</i>	31
4.14	Modello Simulink discreto del motore	32
4.15	Per specificare quale file <i>.c</i> compilare	33
4.16	Definizione e temporizzazione del <i>WHILE</i>	34
4.17	Esecuzione a tempo fissato delle funzioni di lettura e calcolo	34

1

Introduzione

L'approccio seguito per la stesura del modello dinamico del veicolo autobilanciato ha da subito preso una via meno *tradizionale* rispetto al classico metodo risolutivo: abbiamo infatti preferito, dato il nostro *background* informatico, approcciare il problema direttamente in ambiente Matlab, sfruttando sin da subito le potenzialità di calcolo offerte dal software di *Mathworks*.

Nello specifico, per la parte di stesura e definizione della dinamica, abbiamo inizialmente seguito una via risolutiva duale, portando avanti sia un'analisi letterale, sfruttando le potenzialità del **calcolo simbolico** messe a disposizione delle funzionalità di **live scripting**, sia uno studio numerico (considerando quindi le varie grandezze fisiche con i valori definiti delle specifiche di progetto).

In linea di massima lo sviluppo del progetto ha seguito un andamento a step gradualmente, cadenzati da incontri settimanali in cui poter confrontare e consolidare lo *stato di avanzamento dei lavori*: nello specifico, il lavoro ha seguito uno sviluppo in questa direzione step by step, rappresentabile in linea di massima da queste *pietre miliari*:

- **Dinamica di ogni singolo corpo rigido:** abbiamo impostato il problema della dinamica andando a considerare il veicolo auto bilanciato come un insieme di corpi rigidi di cui poterne studiare la dinamica in maniera separata;
- **Dinamica completa del VAB:** siamo andati poi a considerare il sistema nella sua completezza, unendo i contributi dei corpi rigidi considerati in prima battuta singolarmente. Questo ci ha permesso di ottenere le equazioni del moto, in forma non lineare, le quali hanno permesso una completa descrizione del sistema che abbiamo modellizzato;
- **Linearizzazione:** siamo poi andati a linearizzare queste equazioni dinamiche (appunto non lineari) nell'intorno dell'equilibrio;
- **Definizione del controllo:** tramite la tecnica di *pole placement*, siamo andati a definire il controllore più adatto per questo sistema;
- **Modellizzazione del motore:** introduzione del modello del motore sulla base delle caratteristiche reali contenute nella specifica di progetto;
- **Discretizzazione:** passaggio a tempo discreto per i segnali derivanti dal mondo analogico, ovvero per tutto ciò che concerne la parte di sensoristica;
- **Non idealità:** siamo andati a modellizzare anche la presenza di disturbi, di natura stocastica e legati alla quantizzazione;
- **Trasformazioni di blocchi in *interpreted function*:** conversione delle funzionalità di controllo del motore e filtraggio dei segnali provenienti dai sensori in blocchi di codice *Matlab* per favorirne poi la successiva conversione ed implementazione a bordo del controllore;

2

Dinamica

2.1 Scomposizione del VAB

2.1.1 Considerazioni iniziali

Per il calcolo delle equazioni dinamiche del sistema siamo andati a considerare ogni singolo corpo rigido componente il sistema, calcolandone le grandezze fisiche di posizione e velocità, seguendo un approccio cartesiano. Nello specifico abbiamo considerato il sistema composto da:

- Asta
- Utente a bordo dello chassis
- Chassis (nel corso della trattazione sarà chiamata talvolta anche base)
- Ruota (che poi sarà considerata con un contributo doppio, essendo il VAB composto da due ruote)

Ognuno di questi corpi rigidi separati è individuato da un punto, che ne rappresenta il centro di massa (o baricentro del corpo stesso): avremo quindi il seguente insieme di punti caratterizzanti il sistema (figura 2.1)

- P_a
- P_b
- P_c
- P_r

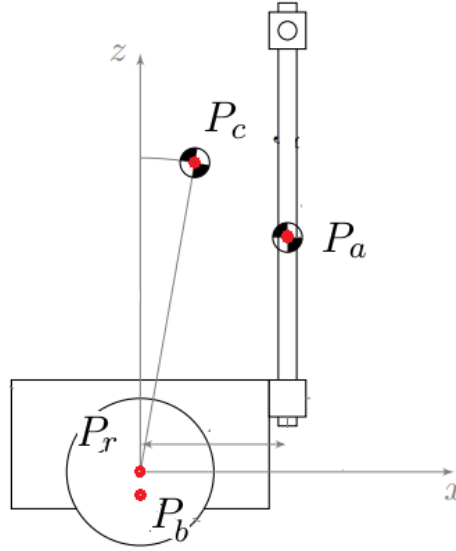


Figure 2.1: Baricentri dei singoli corpi rigidi

2.1.2 Grandezze di supporto

Prima di andare a definire le componenti di energia potenziale e cinetica di ogni singolo corpo, siamo andati ad introdurre alcune grandezze geometriche di supporto che definiremo qui di seguito.

Nello specifico abbiamo introdotto i seguenti parametri, specificati anche in figura 2.2:

- l_a : rappresenta la congiungente tra il centro del sistema di riferimento XZ e il centro dell'asta, utilizzata appunto come manubrio, individuato come

$$l_a = \sqrt{\left(\frac{h_a}{2} + \frac{h_b}{2}\right)^2 + \left(\frac{w_b}{2}\right)^2}$$

- l_c : questa grandezza rappresenta per noi l'altezza del baricentro del corpo dell'utente, la quale ovviamente andrà a dipendere dal valore di inclinazione del corpo stesso. Considerando il corpo inizialmente in posizione verticale, avremo che questa lunghezza corrisponde alla congiungente dal centro del sistema di riferimento al punto P_c , che equivale a dire:

$$l_c = 0.55h_c + \frac{h_b}{2}$$

- l_b : rappresenta lo spostamento verso il basso, lungo l'asse z, del baricentro dello chassis. Da specifiche del progetto sappiamo che questa grandezza ha valore (con segno negativo) di

$$l_b = 0.1m$$

- β : angolo formato con la verticale dalla congiungente tra il centro del sistema di riferimento e il punto P_a . Si ricava, con un semplice approccio trigonometrico che, l'angolo in questione, ha questa forma:

$$\arctan\left(\frac{\frac{w_b}{2}}{\frac{h_a}{2} + \frac{h_b}{2}}\right)$$

2.1.3 Calcolo componenti dinamiche e potenziali per ogni corpo rigido del sistema

Per ognuno dei corpi rigidi definiti in precedenza siamo andati appunto a calcolare:

- **Coordinate spaziali \mathbf{P}** espresse nel sistema di riferimento XZ. A queste due coordinate cartesiane ne va aggiunta una terza, relativa alle coordinate angolari (per poter tener così conto dei contributi inerziali dei corpi);

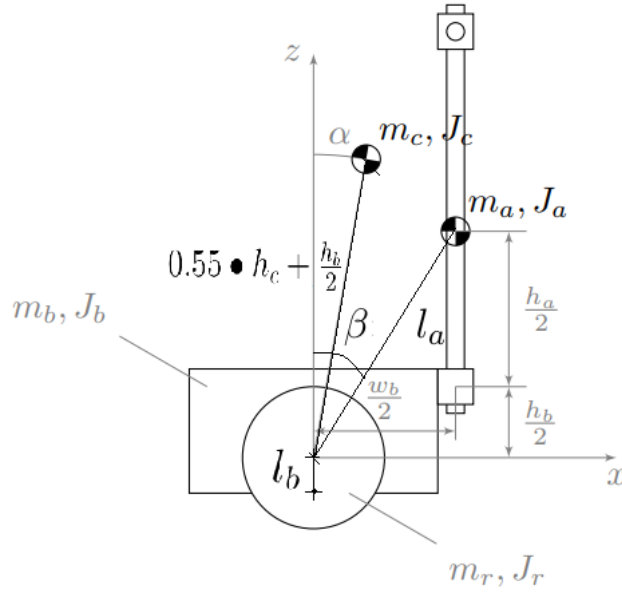


Figure 2.2: Grandezze di supporto

- **Vettore delle velocità** $\mathbf{V} \rightarrow$ vettore 3×1
- **Matrice delle masse** $\mathbf{M} \rightarrow$ matrice 3×3
- **Energia cinetica** $\mathbf{T} \rightarrow \frac{1}{2} \mathbf{V}^T \mathbf{M} \mathbf{V}$
- **Energia potenziale** \mathbf{U}
- **Lagrangiana parziale** \mathbf{L} del singolo corpo rigido

Asta

- $$P_a = \begin{pmatrix} r \phi(t) + l_a \sin(\beta + \theta(t)) \\ l_a \cos(\beta + \theta(t)) \\ \theta(t) \end{pmatrix}$$
- $$V_a = \begin{pmatrix} r \dot{\phi}(t) + l_a \cos(\beta + \theta(t)) \dot{\theta}(t) \\ -l_a \sin(\beta + \theta(t)) \dot{\theta}(t) \\ \dot{\theta}(t) \end{pmatrix}$$
- $$M_a = \begin{pmatrix} m_a & 0 & 0 \\ 0 & m_a & 0 \\ 0 & 0 & m_a l_a^2 + J_a \end{pmatrix}$$
- $$T_a = m_a l_a^2 (\dot{\theta}(t))^2 + \frac{m_a r^2 (\dot{\phi}(t))^2}{2} + \frac{J_a (\dot{\theta}(t))^2}{2} + m_a \cos(\beta + \theta(t)) l_a r \dot{\theta}(t) \dot{\phi}(t)$$
- $$U_a = g l_a m_a \cos(\beta + \theta(t))$$
- $$L_a = m_a l_a^2 (\dot{\theta}(t))^2 + \frac{m_a r^2 (\dot{\phi}(t))^2}{2} + \frac{J_a (\dot{\theta}(t))^2}{2} + m_a \cos(\beta + \theta(t)) l_a r \dot{\theta}(t) \dot{\phi}(t) - g m_a \cos(\beta + \theta(t)) l_a$$

Chassis (base)

- $P_b = \begin{pmatrix} r \phi(t) - l_b \sin(\theta(t)) \\ -l_b \cos(\theta(t)) \\ \theta(t) \end{pmatrix}$
- $V_b = \begin{pmatrix} r \dot{\phi}(t) - l_b \cos(\theta(t)) \dot{\theta}(t) \\ l_b \sin(\theta(t)) \dot{\theta}(t) \\ \dot{\theta}(t) \end{pmatrix}$
- $M_b = \begin{pmatrix} m_b & 0 & 0 \\ 0 & m_b & 0 \\ 0 & 0 & m_b l_b^2 + J_b \end{pmatrix}$
- $T_b = m_b l_b^2 (\dot{\theta}(t))^2 + \frac{m_b r^2 (\dot{\phi}(t))^2}{2} + \frac{J_b (\dot{\theta}(t))^2}{2} - m_b \cos(\theta(t)) l_b r \dot{\theta}(t) \dot{\phi}(t)$
- $U_b = -g l_b m_b \cos(\theta(t))$
- $L_b = m_b l_b^2 (\dot{\theta}(t))^2 + \frac{m_b r^2 (\dot{\phi}(t))^2}{2} + \frac{J_b (\dot{\theta}(t))^2}{2} - m_b \cos(\theta(t)) l_b r \dot{\theta}(t) \dot{\phi}(t) + g m_b \cos(\theta(t)) l_b$

Utente

- $P_c = \begin{pmatrix} r \phi(t) + l_c \sin(\alpha + \theta(t)) \\ l_c \cos(\alpha + \theta(t)) \\ \alpha + \theta(t) \end{pmatrix}$
- $V_c = \begin{pmatrix} r \dot{\phi}(t) + l_c \cos(\alpha + \theta(t)) \dot{\theta}(t) \\ -l_c \sin(\alpha + \theta(t)) \dot{\theta}(t) \\ \dot{\theta}(t) \end{pmatrix}$
- $M_c = \begin{pmatrix} m_c & 0 & 0 \\ 0 & m_c & 0 \\ 0 & 0 & m_c l_c^2 + J_c \end{pmatrix}$
- $T_c = m_c l_c^2 (\dot{\theta}(t))^2 + \frac{m_c r^2 (\dot{\phi}(t))^2}{2} + \frac{J_c (\dot{\theta}(t))^2}{2} + m_c \cos(\alpha + \theta(t)) l_c r \dot{\theta}(t) \dot{\phi}(t)$
- $U_c = g l_c m_c \cos(\alpha + \theta(t))$
- $L_c = m_c l_c^2 (\dot{\theta}(t))^2 + \frac{m_c r^2 (\dot{\phi}(t))^2}{2} + \frac{J_c (\dot{\theta}(t))^2}{2} + m_c \cos(\alpha + \theta(t)) l_c r \dot{\theta}(t) \dot{\phi}(t) - g m_c \cos(\alpha + \theta(t)) l_c$

Ruota

$$\bullet P_r = \begin{pmatrix} r \phi(t) \\ 0 \\ \phi(t) \end{pmatrix}$$

$$\bullet V_r = \begin{pmatrix} r \dot{\phi}(t) \\ 0 \\ \dot{\phi}(t) \end{pmatrix}$$

$$\bullet M_r = \begin{pmatrix} m_r & 0 & 0 \\ 0 & m_r & 0 \\ 0 & 0 & J_r \end{pmatrix}$$

$$\bullet T_r = \frac{(m_r r^2 + J_r) (\dot{\phi}(t))^2}{2}$$

$$\bullet U_r = 0$$

$$\bullet L_r = \frac{(m_r r^2 + J_r) (\dot{\phi}(t))^2}{2}$$

Veicolo completo

Una volta trovati le componenti dinamiche dei singoli corpi rigidi, possiamo andare a definire l'energia cinetica e potenziale totale del sistema, per poter poi calcolare l'equazione di Lagrange per l'intero sistema. In sostanza quindi avremo:

$$L = L_a + L_b + L_c + 2L_r$$

Quello che ne deriva è quindi la seguente equazione Lagrangiana, in grado di descrivere la dinamica dell'intero sistema (riportiamo il risultato letterale, ovvero non legato alla sostituzione di alcun valore numerico che descrive il sistema):

$$L = \left(m_a l_a^2 + m_b l_b^2 + m_c l_c^2 + \frac{J_a}{2} + \frac{J_b}{2} + \frac{J_c}{2} \right) (\dot{\theta}(t))^2 + \left(m_r r^2 + J_r + \frac{m_a r^2}{2} + \frac{m_b r^2}{2} + \frac{m_c r^2}{2} \right) (\dot{\phi}(t))^2 + \\ \left(m_c \sigma_4 l_c r + m_a \sigma_3 l_a r - m_b \cos(\theta(t)) l_b r \right) \dot{\theta}(t) \dot{\phi}(t) + g m_b \cos(\theta(t)) l_b - g m_c \sigma_4 l_c - g m_a \sigma_3 l_a \\ \sigma_3 = \cos(\beta + \theta(t)) \sigma_4 = \cos(\alpha + \theta(t))$$

Note sul calcolo delle componenti dinamiche

- Nel calcolo della matrice di massa abbiamo considerato tre componenti:
 - Componente di massa lungo x;
 - Componente di massa lungo z;
 - Componente di massa rotazionale: dalla meccanica è noto che un corpo, con una certa massa che si trova in uno stato di rotazione, avrà un contributo inerziale che dipende dal braccio rispetto all'asse intorno al quale avviene la rotazione.
- Nello specifico abbiamo considerato il teorema di *Huygens-Steiner* per ogni singolo corpo rigido che è stato preso in esame.

$$I_a = I_{cm} + m d^2$$

Esso permette di definire l'inerzia di un corpo come la somma di due diverse componenti:

- * Momento d'inerzia definito rispetto all'asse passante per il centro di massa: questo parametro rappresenta il valore che è fornito dalle specifiche del progetto per gli specifici corpi;
- * Prodotto tra la massa m del corpo preso in considerazione e la distanza tra l'asse rispetto a cui si riferisce la rotazione e quello passante per il centro di massa;

Questa componente inerziale è visibile nelle matrici di massa in posizione (3,1): si sottolinea come invece, per la matrice di massa relativa alle ruote (matrice M_r 2.1.3), non sia presente la componente esplicitata dal teorema di *Huygens-Steiner* per il fatto che il centro di massa della ruota coincide con quello del sistema di riferimento XZ e quindi di conseguenza non è necessaria alcuna traslazione di asse;

- Nel calcolo simbolico in ambiente *Matlab* siamo andati ad utilizzare le funzionalità di *collect* e *simplify*, per permettere di ridurre e semplificare le equazioni. Nello specifico, con il comando *simplify*, tramite l'opzione *Steps*, siamo andati a settare il numero di step che l'algoritmo di calcolo simbolico andrà ad eseguire per poter ridurre e semplificare il maggior numero di termini possibili ($\uparrow Steps, \uparrow Compattezza eq symb$);
- Le ruote, nel calcolo della dinamica completa del VAB, sono state considerate con un contributo doppio;
- Nello studio dello chassis (base), abbiamo seguito questo approccio: il baricentro della base stessa sappiamo essere posizionato ad una quota differente rispetto al centro del sistem di coordinate XZ preso come riferimento. Per questo motivo il suo contributo in termini cinetici e potenziali dipende dal valore dell'inclinazione dello chassis, ovvero dal valore dell'angolo *caratterizzante* il sistema θ : questo concetto è evidenziato in figura 2.3. L'aggiunta di π al valore di θ è necessaria per poter rendere sensibile i valori di energia cinetica e potenziale al *quadrante* in cui si trova ad essere posizionato il centro di massa della base stessa (P_b).

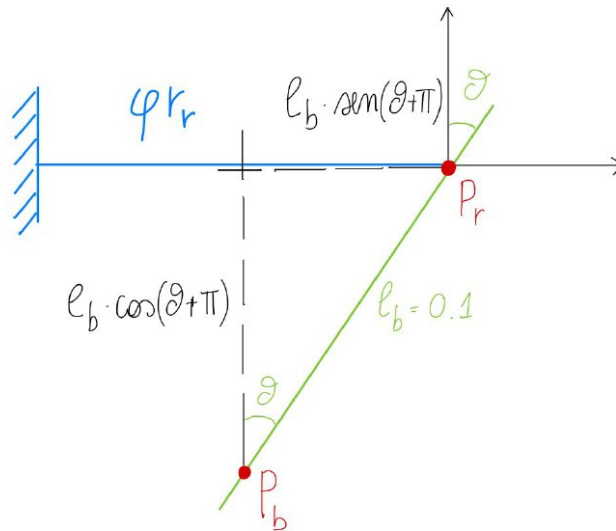


Figure 2.3: Ragionamento per il calcolo dei contributi dinamici dello chassis

2.2 Equazioni del moto

2.2.1 Lagrangiana

Una volta definita la dinamica del sistema ed ottenuta quindi l'equazione di Lagrange che ne caratterizza il comportamento, andiamo a ricavare le equazioni del moto, le quali consentono di definire

l'andamento delle *coordinate libere* (scelte in fase iniziale di progetto) che sappiamo essere

- $\phi = q_1 = \text{angolo di rotazione delle ruote}$
- $\theta = q_2 = \text{angolo di inclinazione dello chassis}$

In particolare possiamo definire le cosiddette *equazioni di Eulero-Lagrange*, ovvero un insieme di n equazioni differenziali (con n pari al numero di coordinate libere del sistema), la cui risoluzione fornisce le equazioni del moto del sistema.

$$\frac{d}{dt} \left(\frac{\partial}{\partial \dot{q}_i} L \right) - \frac{\partial}{\partial q_i} L = Q_i \rightarrow i = 1, \dots, n$$

Sono da effettuare alcune annotazioni sugli addendi presenti nell'equazione precedente (eq 2.2.1):

- q rappresenta la coordinata libera (nel nostro caso saranno θ e ϕ);
- al secondo membro della lagrangiana troviamo le componenti generalizzate, relative alle forze attive esterne. Nel nostro sistema andiamo a modellizzare, come forze attive, quelle che sono le forze motrici immesse dai motori che gestiscono il movimento del V.A.B.: in particolare, nel nostro caso, per il legame matematico tra le componenti generalizzate e il concetto di lavoro virtuale, avremo che l'equazione lagrangiana sarà equiparata al valore di torque immessa dal motore nel sistema;
- molto importante, ai fini della correttezza delle equazioni del moto, è il valore da assegnare al secondo membro *torque*; infatti esso varia a seconda che il sistema agisca sulla coordinata libera q a valle o a monte della trasmissione.

Nello specifico, si osserva che il motore è composto da due parti: lo statore e il rotore. Esse trasmettono una coppia uguale e inversa ai corpi a cui sono solidalmente collegati: lo statore, posizionato nella base del Segway, influenza la coordinata libera θ ed esercita una coppia sullo chassis pari ad una generica coppia $-2C_m$. Il fattore moltiplicativo 2 è dovuto al fatto che nella base sono presenti due motori.

Nello stesso modo, il rotore, trasmette all'albero motore una coppia C_m : a valle della trasmissione si misura dunque una coppia $\frac{C_m}{\tau}$ dovuta alla presenza della trasmissione. Questa coppia ridotta all'albero dell'utilizzatore è la coppia che in definitiva va ad agire sulle ruote e quindi sulla coordinata ϕ .

Si ottiene dunque il seguente sistema di equazioni, la cui risoluzione permette di ottenere due equazioni differenziali del secondo ordine nelle variabili θ e ϕ :

$$\begin{cases} \frac{d}{dt} \left(\frac{\partial}{\partial \dot{\theta}} L \right) - \frac{\partial}{\partial \theta} L = -2C_m \\ \frac{d}{dt} \left(\frac{\partial}{\partial \dot{\phi}} L \right) - \frac{\partial}{\partial \phi} L = \frac{C_m}{\tau} \end{cases}$$

dove τ è il rapporto di trasmissione il cui calcolo è riportato nella sezione 2.2.2. La soluzione di questo sistema di equazioni verrà ad essere utilizzata per rappresentare il comportamento del sistema reale, basandosi su una coppia di equazioni non lineare: questa tematica verrà poi approfondita direttamente nella sezione 4.2.

2.2.2 Rapporto di trasmissione

Come da specifiche del progetto, ognuno dei due motori è collegato alle ruote mediante una doppia sequenza di riduttori:

- Un primo riduttore epicicloidale con rapporto di trasmissione $\tau_1 = 0.1$;
- Un secondo riduttore, una cinghia dentata che presenta un rapporto di trasmissione facilmente calcolabile come rapporto tra il numero dei denti dei due alberi, essendo il passo uguale in entrambi gli alberi.

$$\tau_2 = \frac{\text{Numero_denti_puleggia_ingresso}}{\text{Numero_denti_puleggia_uscita}} = \frac{Z_{in}}{Z_{out}} = \frac{22}{26}$$

Il rapporto di trasmissione completo τ è quindi definito come:

$$\tau = \tau_1 \cdot \tau_2 = 0.085$$

2.2.3 Sistema non lineare

La risoluzione del sistema di equazioni sopra riportato permette di ottenere due equazioni differenziali del secondo ordine: queste soluzioni sono state ottenute sfruttando la potenzialità del calcolo simbolico messo a disposizione da Matlab, come si vede nel listato seguente.

```
theta2_diff = subs(theta2,{phi,vel,theta,vel_ang,C_m},{q_1 q_1_p q_2 q_2_p u})
phi2_diff = subs(phi2,{phi,vel,theta,vel_ang,C_m},{q_1 q_1_p q_2 q_2_p u })
```

Listing 2.1: Risoluzione del sistema di equazioni

Le soluzioni così ottenute rappresentano e governano la dinamica del sistema: esse verranno quindi utilizzate per andare ad analizzare il comportamento del sistema considerato reale, utilizzando il controllore che è stato progettato sul sistema linearizzato.

Un passaggio importante per la simulazione del sistema non lineare, è quello rappresentato dal salvataggio di queste due equazioni in esame: infatti esse sono, per forza di cose, richiamate all'interno della simulazione *Simulink*: per fare ciò abbiamo utilizzato il comando *matlabFunction()*, il quale va a scrivere in una funzione di matlab le due equazioni in esame.

```
matlabFunction(theta2_diff,'File','theta_secondo');
matlabFunction(phi2_diff,'File','phi_secondo');
```

Listing 2.2: Salvataggio in funzioni Matlab

Nello specifico queste funzioni avranno come input i valori da cui dipendono le equazioni differenziali e, componendoli algebricamente tra di loro, fornirà come output i valori di $\ddot{\theta}$ e $\ddot{\phi}$.

TODO (mettere le dipendende di theta pp e phi pp)

2.2.4 Linearizzazione

TODO linearizzazione attorno alla massa 70 kg e rileggere paragrafo

Nell'approccio alla modellistica dei sistemi dinamici, uno step molto importante è quello che concerne la linearizzazione del sistema, ovvero il passaggio da un insieme di equazioni non lineari ad un set di equazioni lineari definite nell'intorno di un punto specifico dello stato del sistema: questo nuovo sistema di equazioni andrà a definire un sistema lineare in grado di approssimare il comportamento dinamico del sistema non lineare vicino all'equilibrio.

Il punto in questione è quello in cui la variazione dello stato del sistema è nullo: detto quindi $\mathbf{x}(t)$ il vettore di stato, l'equilibrio sarà dato da $\dot{\mathbf{x}}(t)$.

Perciò, il passo successivo alla risoluzione del sistema di equazioni di Lagrange in θ e ϕ che è stato svolto riguarda la linearizzazione: nel contesto del controllo, linearizzare è importante poiché permette di progettare il controllore stesso sul sistema tangente a quello reale (ovvero il sistema lineare), potendo poi testarlo direttamente sul sistema reale (ovvero quello descritto da equazioni non lineari).

Calcolo della posizione di equilibrio

Per poter quindi procedere con la linearizzazione, è necessario innanzitutto calcolare l'equilibrio del sistema, cioè il punto in cui $\ddot{\theta} = 0$: questa condizione corrisponde ad una situazione in cui lo stato del sistema risulta essere in una condizione di equilibrio dinamico, ovvero quando la sua variazione nel tempo è nulla (nessuna accelerazione \rightarrow velocità costante \rightarrow posizione lineare):

$$\begin{pmatrix} -0.01174364 - 7.105427e-15i \\ 3.129849 - 7.105427e-15i \end{pmatrix}$$

Queste posizioni di equilibrio le abbiamo ottenute per mezzo della istruzione Matlab `??`, nella quale si vede come, partendo dall'equazione differenziale del sistema non lineare espressa in termini di $\ddot{\theta}$ si ricava la posizione di equilibrio del sistema.

```
equilibri = solve(subs(theta2_differenziabile,{q_1 q_1_p q_2_p u},{0,0,0,0})==0,q_2)
```

Listing 2.3: Calcolo posizione di equilibrio

Come si vede, siamo interessati solamente al valore di equilibrio relativo di θ : per questo motivo che poniamo a 0 (valore arbitrario) i valori di ϕ e $\dot{\phi}$, poiché non risultano essere di interesse per il calcolo dell'equilibrio.

I risultati ottenuti sono, con buona approssimazione, considerabili numeri naturali e rispondono a quanto ci aspettavamo: essendo presente un offset (w_b) tra quello che è il baricentro del sistema di riferimento e quello relativo al manubrio, non ci aspettavamo di ottenere due equilibri perfettamente di 0° (posizione verticale a "testa in giù") e di 180° (posizione verticale a "testa in giù").

Per completezza e per avere comunque un feedback più concreto, abbiamo provato ad azzerare il valore di w_b , andando a ricalcolare la posizione di equilibrio, ottenendo effettivamente i valori di

$$\begin{pmatrix} 0 \text{ deg} \\ 180 \text{ deg} \end{pmatrix}$$

Dal risultato ottenuto, considerando nulla la componente immaginaria, possiamo notare che (osservazioni comuni per i sistemi assimilabili al pendolo):

- il sistema ha due equilibri; il primo ($\theta = -0.01174364 \text{ rad} = -0.67 \text{ deg}$) ci si aspetta che sia instabile in quanto il baricentro del sistema V.A.B. (compreso di utente a bordo) risulta essere sopra all'asse delle ruote;
- il secondo ($\theta = 3.129849 \text{ rad} = 179.32 \text{ deg}$) è un equilibrio stabile, poiché il baricentro sta sotto l'asse delle ruote e, un eventuale disturbo dopo un certo tempo di transitorio, risulterebbe avere effetto nullo sullo stato del sistema, che ritornerebbe alla medesima posizione;

Calcolo del sistema lineare

Ovviamente, ai fini del controllo, si è linearizzato attorno al primo equilibrio; non avrebbe infatti senso controllare il sistema quando questo risulta essere capovolto (cosa che per di più risulta essere fisicamente impossibile, se il veicolo autobilanciato viene fatto muovere su una superficie).

I vettori che definiscono le **variabili di stato del sistema** e le **uscite del sistema** sono i seguenti:

$$\begin{aligned} x = \text{variabili di stato} &= \begin{bmatrix} x_1 \rightarrow \phi \\ x_2 \rightarrow \dot{\phi} \\ x_3 \rightarrow \theta \\ x_4 \rightarrow \dot{\theta} \end{bmatrix} \\ y = \text{output del sistema} &= \begin{bmatrix} y_1 \rightarrow \ddot{\theta} \\ y_2 \rightarrow \ddot{\phi} \end{bmatrix} \end{aligned}$$

Per individuare il risultato finale della linearizzazione, abbiamo seguito un approccio matriciale: nello specifico abbiamo utilizzato la notazione matriciale nel caso di sistemi SIMO (*Single Input Multiple Output*), la cui stesura ha utilizzato le seguenti funzioni di supporto:

$$\left\{ \begin{array}{l} \mathbf{f}_1 \rightarrow \dot{x}_1(t) = \overline{x_2} = 0 \text{ (poichè all'equilibrio)} \\ \mathbf{f}_2 \rightarrow \dot{x}_2(t) = \ddot{\phi} \\ \mathbf{f}_3 \rightarrow \dot{x}_3(t) = \overline{x_4} = 0 \text{ (poichè all'equilibrio)} \\ \mathbf{f}_4 \rightarrow \dot{x}_4(t) = \ddot{\theta} \\ \mathbf{g}_1 \rightarrow y_1(t) = \phi = x_1 \\ \mathbf{g}_2 \rightarrow y_2(t) = \theta = x_2 \end{array} \right.$$

Sfruttando le caratteristiche *matematiche* del sistema non lineare, ottenibili tramite lo sviluppo in serie di Taylor nell'intorno dell'equilibrio, possiamo definire i termini matriciali che permettono di definire il sistema, secondo le seguenti equazioni:

$$\left\{ \begin{array}{l} \dot{x}(t) = Ax(t) + Bu(t) \\ y(t) = Cx(t) + Du(t) \end{array} \right.$$

$$A = \begin{bmatrix} \frac{\partial}{\partial x_1} f_1 & \frac{\partial}{\partial x_2} f_1 & \frac{\partial}{\partial x_3} f_1 & \frac{\partial}{\partial x_4} f_1 \\ \frac{\partial}{\partial x_1} f_2 & \frac{\partial}{\partial x_2} f_2 & \frac{\partial}{\partial x_3} f_2 & \frac{\partial}{\partial x_4} f_2 \\ \frac{\partial}{\partial x_1} f_3 & \frac{\partial}{\partial x_2} f_3 & \frac{\partial}{\partial x_3} f_3 & \frac{\partial}{\partial x_4} f_3 \\ \frac{\partial}{\partial x_1} f_4 & \frac{\partial}{\partial x_2} f_4 & \frac{\partial}{\partial x_3} f_4 & \frac{\partial}{\partial x_4} f_4 \end{bmatrix} \quad B = \begin{bmatrix} \frac{\partial}{\partial u} f_1 \\ \frac{\partial}{\partial u} f_2 \\ \frac{\partial}{\partial u} f_3 \\ \frac{\partial}{\partial u} f_4 \end{bmatrix} \quad C = \begin{bmatrix} \frac{\partial}{\partial x_1} g_1 & \frac{\partial}{\partial x_2} g_1 & \frac{\partial}{\partial x_3} g_1 & \frac{\partial}{\partial x_4} g_1 \\ \frac{\partial}{\partial x_1} g_2 & \frac{\partial}{\partial x_2} g_2 & \frac{\partial}{\partial x_3} g_2 & \frac{\partial}{\partial x_4} g_2 \end{bmatrix}$$

$$D = \begin{bmatrix} \frac{\partial}{\partial u} g_1 & \frac{\partial}{\partial u} g_2 \end{bmatrix}$$

Nel specifico del veicolo autobilanciato, nel caso con utente con peso di 70 kg e altezza 1.77 m, siamo andati ad ottenere i seguenti valori numerici per le matrici in esame:

$$A = \begin{bmatrix} 0 & 1.0 & 0 & 0 \\ 0 & 0 & -15.2286 & 0 \\ 0 & 0 & 0 & 1.0 \\ 0 & 0 & 5.43924 & 0 \end{bmatrix} \quad B = \begin{bmatrix} 0 \\ 2.7498 \\ 0 \\ -0.2612 \end{bmatrix} \quad C = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

$$D = \begin{bmatrix} 0 & 0 \end{bmatrix}$$

Ricordando infine che un generico sistema dinamico può essere scritto come:

$$G(s) = C(sI - A)^{-1}B + D$$

sostituendo i valori numerici matriciali ottenuti al passo precedente, possiamo ottenere il sistema di equazioni, le quali rappresentano l'insieme delle funzioni di trasferimento caratterizzanti il sistema:

$$\begin{pmatrix} \frac{2.75}{s^2} - \frac{1.749e+13}{2.392e+13 s^2 - 4.398e+12 s^4} \\ \frac{2.75}{s} - \frac{1.749e+13}{2.392e+13 s - 4.398e+12 s^3} \\ - \frac{1.149e+12}{4.398e+12 s^2 - 2.392e+13} \\ - \frac{1.149e+12 s}{4.398e+12 s^2 - 2.392e+13} \end{pmatrix}$$

Si può notare come la prima e la seconda riga della matrice differiscano solo per un fattore derivativo, così come la terza e quarta riga; questo è ovvio ed atteso in quanto x_2 è a derivata di x_1 e x_4 la derivata di x_3 .

Nello specifico, queste funzionalità di linearizzazione, sono state racchiuse all'interno del *live script* Matlab nominato *VAB_simulazioni_discrete.mlx*: in questo modo siamo stati in grado di rendere più efficace ed efficiente la prosecuzione della simulazione, non dovendo eseguire ogni volta anche questa parte di calcolo matematico relativo alla linearizzazione. Siamo andati infatti a sfruttare le funzionalità di salvataggio offerte da Matlab, per salvare in un file *.mat* l'intera *workspace* (file che abbiamo nominato *WS_VAB_wb_0_5.mat*), in maniera tale da permettere, agli script che ne avessero bisogno, di aprire il workspace e leggere tutte le variabili di interesse.

3

Controllo

3.1 Introduzione

Per rendere maggiormente leggibile il codice Matlab e, allo stesso tempo, ridurre i tempi necessari per l'esecuzione, abbiamo deciso di dividere le varie funzionalità in diversi *livescript*: in particolar modo la parte relativa allo studio ed alla definizione del controllore, siamo andati ad inserirla all'interno del file *GainCalculator.mlx* dove si può vedere che, come prima istruzione, si va a caricare dal workspace, risultante dalla linearizzazione, le matrici A-B-C-D che definiscono il sistema lineare modellizzato.

3.2 Analisi in anello aperto

Prima di procedere con l'individuazione del controllore più adatto per stabilizzare il V.A.B., siamo andati ad effettuare una breve analisi sul sistema ad anello aperto, per ottenere così alcuni spunti sulla correttezza del modello che era stato steso fino a quel punto.

Come noto dalla teoria, il sistema risulta essere asintoticamente stabile se gli autovalori della matrice A risultano avere tutti parte reale negativa; è instabile invece se è presente almeno un autovalore con parte reale strettamente positiva.

Abbiamo quindi calcolato gli autovalori con il comando seguente:

```
eig(A_real)
```

Listing 3.1: Calcolo autovalori matrice A (non simbolica)

La presenza di autovalori con parte reale strettamente negativa è stata confermata anche dalla visualizzazione del luogo delle radici (*rlocus*) del sistema in anello aperto: come si vede in figura 3.1, il polo che si trova nel semi-piano reale positivo non potrà, in alcun modo, essere stabilizzato, ovvero portato nel semi-piano sinistro.

Questa breve analisi ci ha permesso di avere una conferma numerica del fatto ovvio che, senza un controllo, il V.A.B. in esame non riesce a mantenere da solo la posizione verticale.

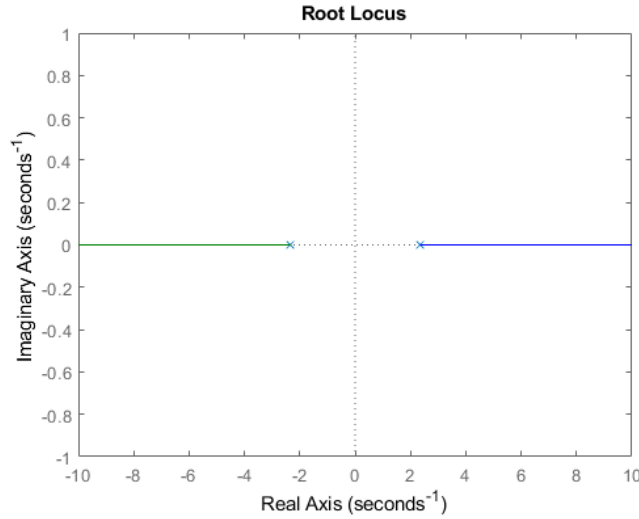


Figure 3.1: Luogo delle radici del V.A.B. modellizzato in anello aperto

3.3 Controllore

Per quanto riguarda il sistema del veicolo autobilanciato, siamo andati a far riferimento a quella classe di problemi definiti come *problemi di regolazione*, in cui il sistema si presenta inizialmente con una condizione iniziale dello stato non nulla, condizione che si intende di riportare a zero con una velocità di convergenza assegnata.

Nello specifico, questo tipo di problema relativo alla scelta del controllore, è stato risolto mediante l'utilizzo dell'assegnazione degli autovalori ottenuti tramite retroazione dello stato, in cui il moto del sistema è composto completamente dal moto libero che si vuole controllare e annullare in un tempo a piacere.

Il posizionamento dei poli, e quindi la scelta del guadagno del regolatore, va eseguita sul solo sistema lineare: questo a conferma sia di quanto riportato in figura 3.2 (nel blocco di colore blu infatti è riportato il sistema lineare), sia di quanto detto in precedenza in merito al fatto che il controllore venga ad essere definito lavorando sul sistema linearizzato.

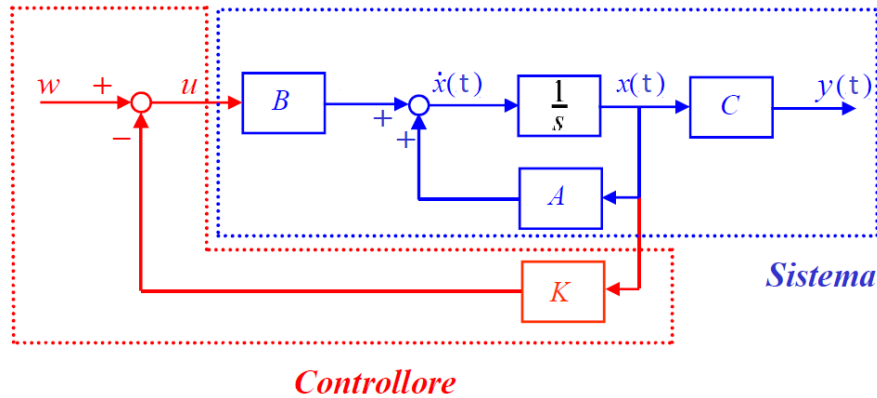


Figure 3.2: Schema concettuale del controllo ([1])

Ciò che è riportato graficamente in figura 3.2, può essere riscritto matematicamente come segue:

$$\begin{cases} \dot{x}(t) = Ax(t) + Bu(t) \\ y(t) = Cx(t) + Du(t) \end{cases}$$

$$u(t) = -Kx(t) + w(t) = \text{legge di controllo}$$

Dunque, sostituendo la legge di controllo nell'equazione di stato del sistema, possiamo ottenere le seguenti equazioni che ci permettono di definire il legame del sistema in anello chiuso con la matrice K :

$$\begin{aligned}x(t) &= Ax(t) + Bu(t) = Ax(t) + B(-Kx(t) + w(t)) \\&= Ax(t) - BKx(t) + Bw(t) = (A - BK)x(t) + Bw(t) \\A_{cl} &= A - BK = \text{matrice di stato in anello chiuso (cl} \rightarrow \text{closed loop)}\end{aligned}$$

3.3.1 Posizionamento dei poli continui

Per scegliere il valore da assegnare a K , e quindi definire il guadagno del controllore, è necessario scegliere la posizione desiderata dei poli, in base a quelle che sono le specifiche di velocità desiderate.

$$G(s) = \frac{\omega_n^2}{s^2 + 2\xi\omega_n s + \omega_n^2}$$

$$\xi = 0.7$$

$$\omega_n = 2\pi f_{\text{propria}}$$

Dovendo posizionare due coppie di poli complessi coniugati si sono scelte due frequenze ad una decade di distanza, in maniera tale da poter considerare le variabili che sono controllate disaccoppiate in frequenza: essendo il limite di banda (ovvero entro quali limiti il segnale passa senza essere attenuato e/o modificato) del sistema interno molto più alto del limite di banda del sistema esterno, possiamo considerarli disaccoppiati in frequenza.

$$f_{\text{propria}\theta} = 0.2 \text{ Hz}$$

$$f_{\text{propria}\phi} = 0.02 \text{ Hz}$$

TODO Nello specifico, una prima osservazione, è il fatto che il polo più veloce (ovvero quello con frequenza pari a 0.2 Hz) è stato assegnato alla parte relativa al controllo dell'angolo θ , essendo che è auspicabile un controllo maggiormente reattivo per quanto riguarda la stabilizzazione della base, piuttosto che il rapido raggiungimento del set point di posizione spaziale (e quindi angolare ϕ delle ruote).

Una seconda osservazione è che, motivi esterni, il motore risulta avere una coppia massima molto limitata (per via delle limitazioni di corrente): è stato dunque necessario posizionare i poli ad una frequenza tale che permettessero di non saturare per molto tempo la coppia fornita dal motore. Questa scelta ha inevitabilmente rallentato la risposta del sistema.

Definita quindi la frequenza a cui posizionare i poli, siamo andati a risolvere l'equazione al denominatore della forma generica della f.d.t ($G(s)$) con due poli complessi: la risoluzione di questa equazione, come è noto dalla teoria, permette di ottenere i valori della variabile di Laplace s che azzerano il denominatore stesso, che corrisponde a risolvere questa equazione (in forma generica):

$$s^2 + 2\xi\omega_n s + \omega_n^2 = 0$$

Nella nostra simulazione i valori numerici ottenuti per i poli sono stati i seguenti:

$$\begin{aligned}polo_{\theta} &= \begin{pmatrix} -\frac{7\pi}{250} + \frac{\pi\sqrt{51}i}{250} \\ -\frac{7\pi}{250} - \frac{\pi\sqrt{51}i}{250} \end{pmatrix} \\polo_{\phi} &= \begin{pmatrix} -\frac{7\pi}{25} + \frac{\pi\sqrt{51}i}{25} \\ -\frac{7\pi}{25} - \frac{\pi\sqrt{51}i}{25} \end{pmatrix}\end{aligned}$$

Con il comando *place* di Matlab si ottiene dunque:

$$K = [-0.0023, -0.0278, -28.1397, -7.7022]$$

Ricordando a questo punto quanto detto nella prima parte di questo capitolo in merito alla definizione della matrice A in anello chiuso, siamo andati a definirla appunto come

$$A_{cl} = A - BK$$

utilizzando il valore di K appena trovato: una volta trovata la nuova matrice A , siamo andati a ri-effettuare le stesse analisi portate avanti anche per il sistema in anello aperto. Come si vede in figura 3.3, grazie al posizionamento degli autovalori, siamo stati in grado di spostare i nuovi poli nel semi-piano sinistro: questo significa quindi che siamo riusciti a stabilizzare il nostro sistema dinamico.

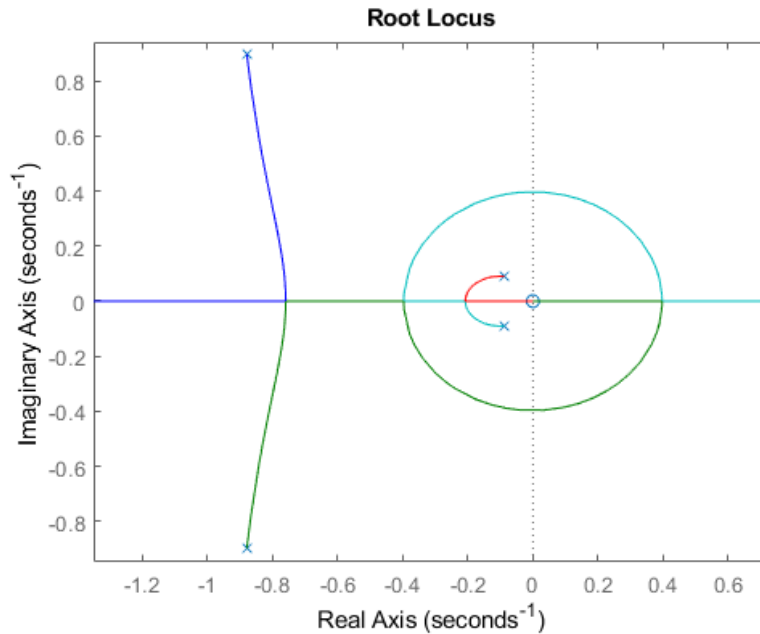


Figure 3.3: Root locus del sistema in anello chiuso

3.3.2 Posizionamento dei poli discreti

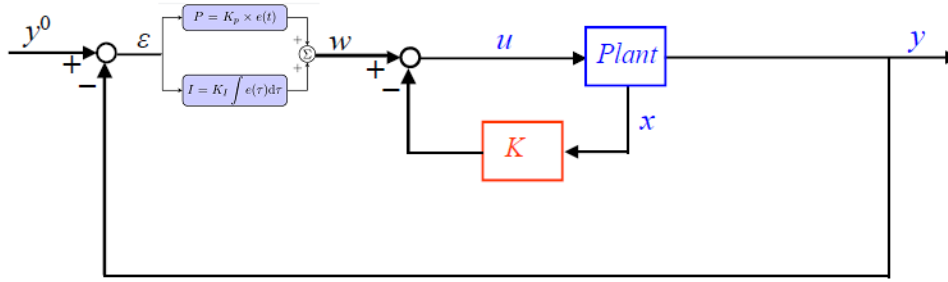
TODO

Nello sviluppo del sistema, come presenteremo poi nella sezione di simulazioni (sezione TODO), siamo andati anche ad inserire il passaggio a tempo discreto, a seconda di quelle che sono le specifiche temporali di campionamento del controllore.

Nel definire quindi un controllo a tempo discreto è stato necessario anche andare a riportare il posizionamento dei poli in ambito discreto.

3.3.3 Setpoint di velocità

Oltre al controllo del moto libero del sistema è stata implementata anche la possibilità di inserire un setpoint sulla velocità a fine transitorio del Segway. In questo modo si apre la possibilità per l'utente finale di impostare la velocità desiderata e di mantenerla nel tempo nonostante i vari disturbi che in un sistema reale influiranno sulla macchina (vento, salita, discesa..)

Figure 3.4: chiusura dell'anello di controllo di $\dot{\phi}$ con un controllore I [1]

In Fig.3.4 si nota come l'anello di $\dot{\phi}$ sia esterno rispetto all'anello di retroazione dello stato; solitamente si procede dunque nel progettare il controllore avendo cura di lasciare una decade di spazio tra la frequenza del polo dell'integratore e la coppia di poli più lenta del controllore della retroazione dello stato. In questo caso, per ragioni puramente pratiche, non è stato possibile: il sistema presenta una risposta particolarmente lenta agli input viste le limitazioni di coppia di cui si approfondirà successivamente; sarebbe quindi stato necessario quindi troppo tempo per raggiungere il setpoint di velocità se si fosse proceduto a lasciare una decade di distanza. Si è invece proceduto a posizionare il polo dell'integratore tramite la funzione di Tuning offerta da Matlab Simulink stesso:

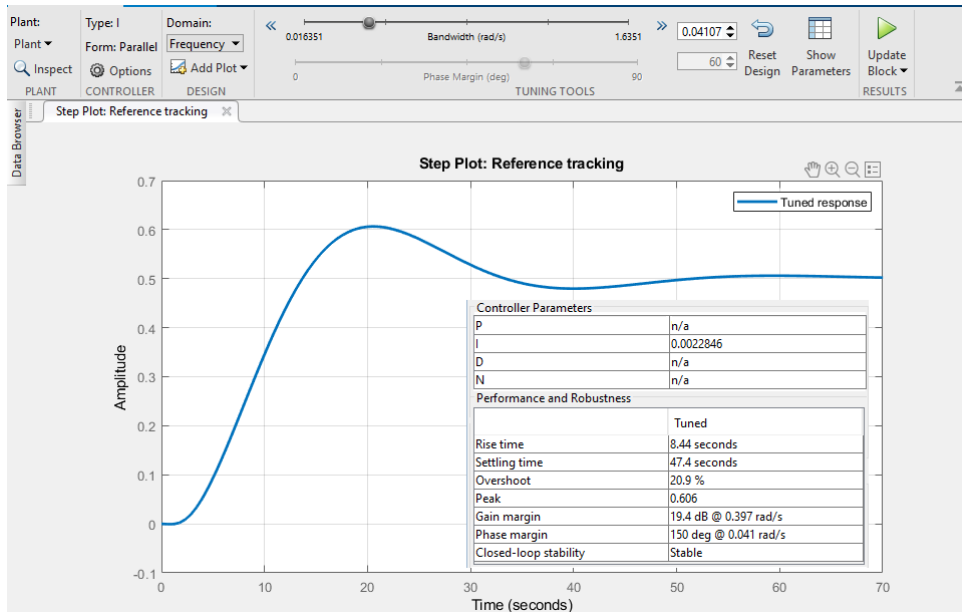


Figure 3.5: Taratura del controllore

Il tuning è stato effettuato sul sistema lineare in quanto Matlab richiede un sistema lineare per questo tipo di tecniche. Da notare inoltre, in Fig.3.5 che la frequenza $f_3 = \frac{0.041 \text{ rad/s}}{2\pi} = 0.0065 \text{ Hz}$ che è circa $\frac{1}{3} f_{\text{propria } \phi}$. Sempre in Fig.3.5 si osserva come comunque, non avendo rispettato la decade di distanza, il sistema sia comunque molto lento e arrivi a regime in circa 70 s. TODO: perchè arriva a 0.5? per la retroazione interna?

3.4 OPC - UA

Per quanto riguarda la parte di controllo, il sistema presenta un articolato insieme di controllori inter-operanti tra di loro: nello specifico ad ogni singolo controllore sono affidate delle mansioni ben specifiche, tutte ovviamente volte al controllo e alla stabilizzazione del *veicolo auto bilanciato*.

In questa fase dello sviluppo del progetto, siamo andati ad implementare parte del codice che verrà installato, in un secondo momento, a bordo del raspberry: esso infatti svolge, all'interno del sistema

(come si vede in figura 3.6) una comunicazione a due direzioni, che ne determinano due comportamenti differenti:

- Come **server** per la parte di comunicazione *OPC-UA* (per il settaggio dei guadagni);
- Come **master** nella comunicazione seriale verso Arduino (per quanto riguarda invece la gestione dell'algoritmo di controllo);

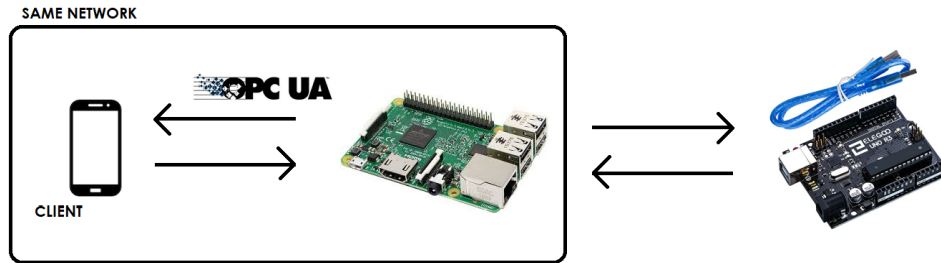


Figure 3.6: Schema di massima dell'utilizzo di Raspberry Pi 3

In questa fase abbiamo quindi sviluppato la parte relativa all'utilizzo di *Raspberry Pi 3* come *server OPCUA*.

3.4.1 Idea base OPC-UA

L'*Open Platform Communications Unified Architecture* (OPC UA) è un protocollo di comunicazione automatico per l'automazione industriale.

OPC UA sostituisce il protocollo *OPC Classic*, conservando tutte le funzionalità del predecessore. Poiché OPC Classic era stato costruito su una tecnologia Microsoft detta modello a oggetti per componenti distribuiti, era vincolato a Microsoft, ma questa caratteristica è diventata sempre più limitante.

OPC UA risulta completamente interoperabile tra i diversi sistemi operativi usati, aggiungendosi a Windows e alle tecnologie industriali come i PLC, inoltre comprende Linux, iOS e anche sistemi operativi per dispositivi mobili come Android. Consentire al maggior numero di dispositivi possibile di comunicare contribuisce al progresso dell'IoT.

Queste caratteristiche di **interoperabilità** sono state sfruttate al massimo in questo contesto, potendo così creare, in maniera semplice e veloce, una comunicazione tra differenti tipologie e famiglie di dispositivi.

3.5 OPC-UA e V.A.B.

Nel contesto del progetto del *veicolo auto bilanciato*, siamo andati ad utilizzare il protocollo di comunicazione *OPC-UA* come supporto per il tuning dei parametri relativi al **gain** del controllore, ovvero ai parametri del vettore K , che abbiamo chiamato (all'interno dello script di Python):

- K_{ϕ}
- K_{ϕ_p}
- K_{θ}
- K_{θ_p}

Nello specifico, lo scambio di parametri tra server e controllore avviene tramite unfile *.txt*, che permette, in maniera semplice e immediata, di implementare uno scambio di informazioni tra il server *OPC-UA* strutturato in Python e l'ambiente *real-time* introdotto a bordo del controllore *Raspberry Pi 3*.

In particolare abbiamo due files:

- **Un file temporaneo** ("*GainParametersToController.txt*") utilizzato come pipeline per il passaggio dei parametri tra server e controllore. Questo risulta essere un file temporaneo che viene ad essere cancellato e ricreato ogni qualvolta che il server viene spento e successivamente riaccessso. Nello specifico, ad ogni riaccensione, i valori iniziali di questo file, vengono settati con gli stessi valori presenti nel file *definitivo* (qualora quest'ultimo esista già: in caso contrario si procede con un'inizializzazione dei parametri a 0);
- **Un file definitivo** ("*GainParametersConfirmed.txt*") il quale invece viene creato una e una sola volta e sul quale poi vengono salvati i parametri che saranno poi letti all'accensione successiva del server ed utilizzati come parametri iniziali per il controllore.

Questi file possono essere settati con i parametri presenti nel server che sono visibili in figura 3.7, nello specifico:

- **Submit change to controller** permette di scrivere i valori dei gains sul file "*GainParametersToController.txt*";
- **Store definitively in file** permette di scrivere i valori dei gains sul file "*GainParametersConfirmed.txt*";
- **SHUT DOWN SERVER** permette invece di spegnere il server e di cancellare successivamente il file temporaneo.

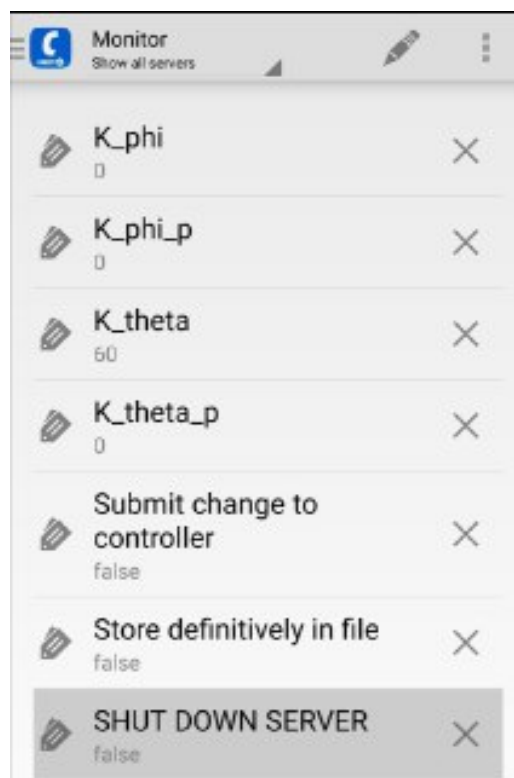


Figure 3.7: Parametri impostabili lato client

Abbiamo racchiuso in figura 3.8 il funzionamento di massima del codice lato server: codice che siamo andati a testare utilizzando un'apposita app per smartphone Android (OPC-UA Android client); per quanto riguarda invece l'effettiva implementazione, abbiamo riportato nell'Appendice A l'intero codice prodotto, per la gestione della comunicazione OPC-UA.

3.6 Rumore

Piattaforma inerziale → PSD Questione anti windup → coppia già limitata quindi nessun vincolo di saturazione

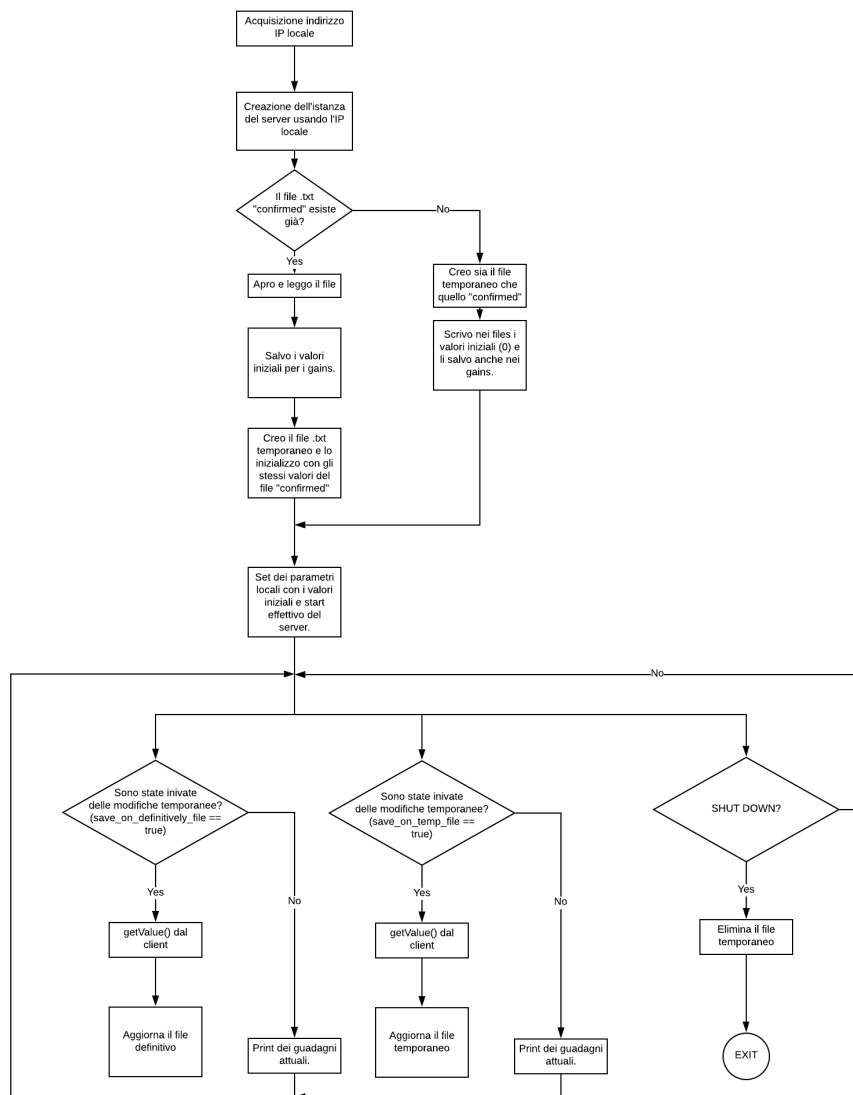


Figure 3.8: Diagramma rappresentante le funzionalità del server

4

Simulink

4.1 Introduzione

TODO: rivdere questa introduzione (io la rivedrei)

Data la straordinarietà degli eventi che erano in corso dal punto di vista sanitario nel nostro paese si è reso necessario svolgere gran parte del lavoro in modalità a distanza e quindi senza la possibilità di testare il modello matematico appena ottenuto e i successivi risultati dovuti all'azione di controllo sul V.A.B., il lavoro quindi è stato svolto per la maggior parte sfruttando il tool *Simulink* di Matlab che è, in poche parole, un risolutore di equazioni differenziali.

Abbiamo così adottato una metodologia di lavoro basata su prototipi sempre più simili a quello che dovrebbe essere il sistema reale.

Si va ora ad analizzare la risposta del sistema al controllo ottenuto nei punti precedenti, per assicurarci, che quanto scritto sopra valga oltre che nella teoria anche nella pratica:

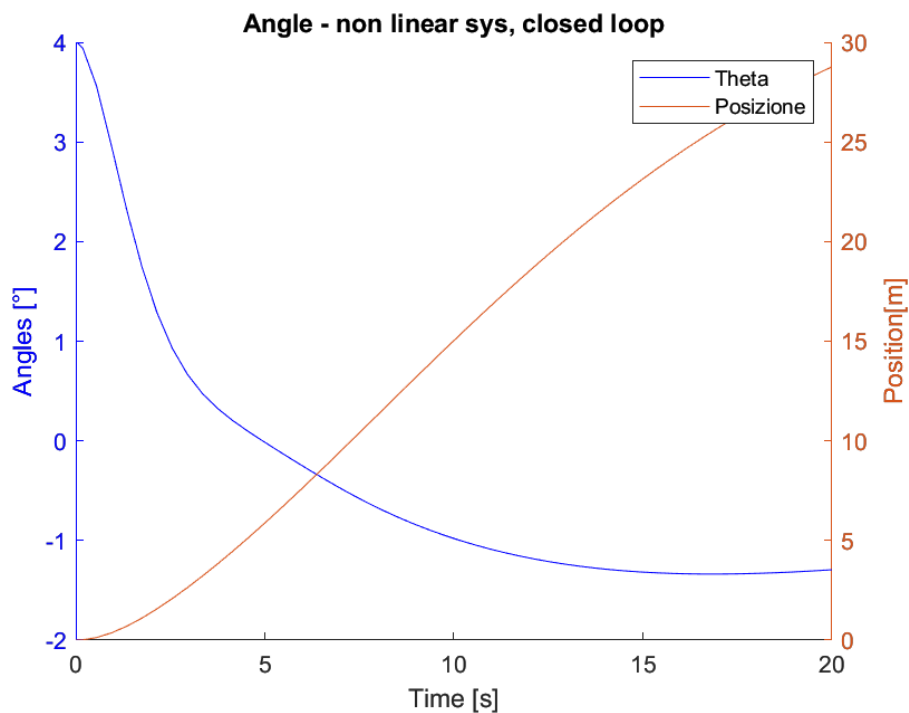


Figure 4.1: Risposta del sistema non lineare in anello chiuso

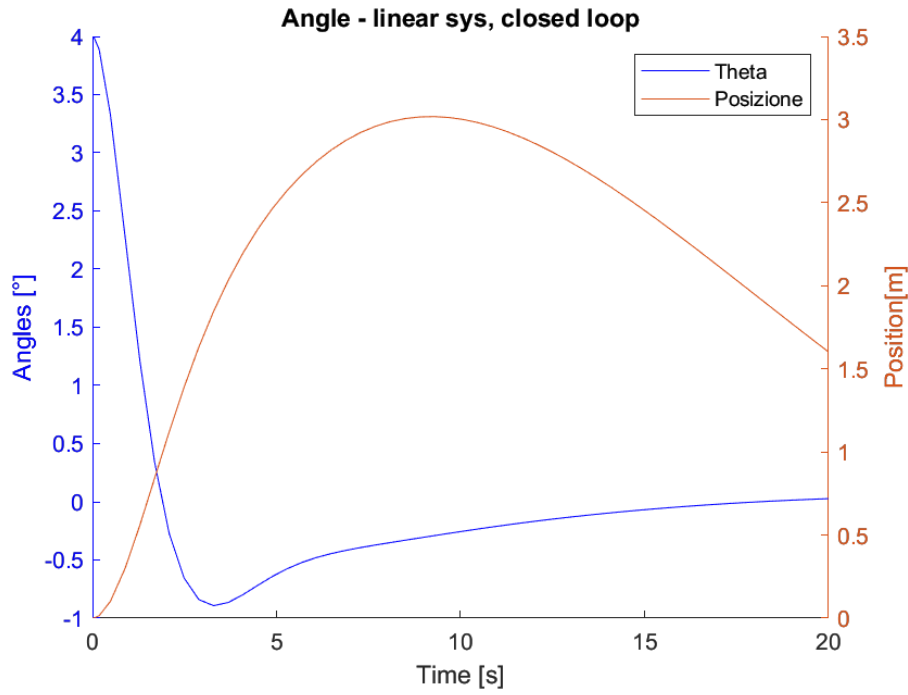


Figure 4.2: Risposta del sistema lineare in anello chiuso

TODO: come spieghiamo la differenza? rifare la simulazione?

4.2 Simulazione del sistema non lineare

Il primo compito che abbiamo risolto è stato quello di implementare le equazioni differenziali ottenute nel capitolo precedente:

- $\ddot{\phi} = f_{\ddot{\phi}}(M_c, \theta, \dot{\theta}, C_m)$
- $\ddot{\theta} = f_{\ddot{\theta}}(M_c, \theta, \dot{\theta}, C_m)$

Dove M_c sarebbe la massa del passeggero, θ e $\dot{\theta}$ lo stato del sistema e C_m la coppia erogata dal motore. Si può notare come entrambe le equazioni differenziali siano indipendenti dalla coordinata libera ϕ .

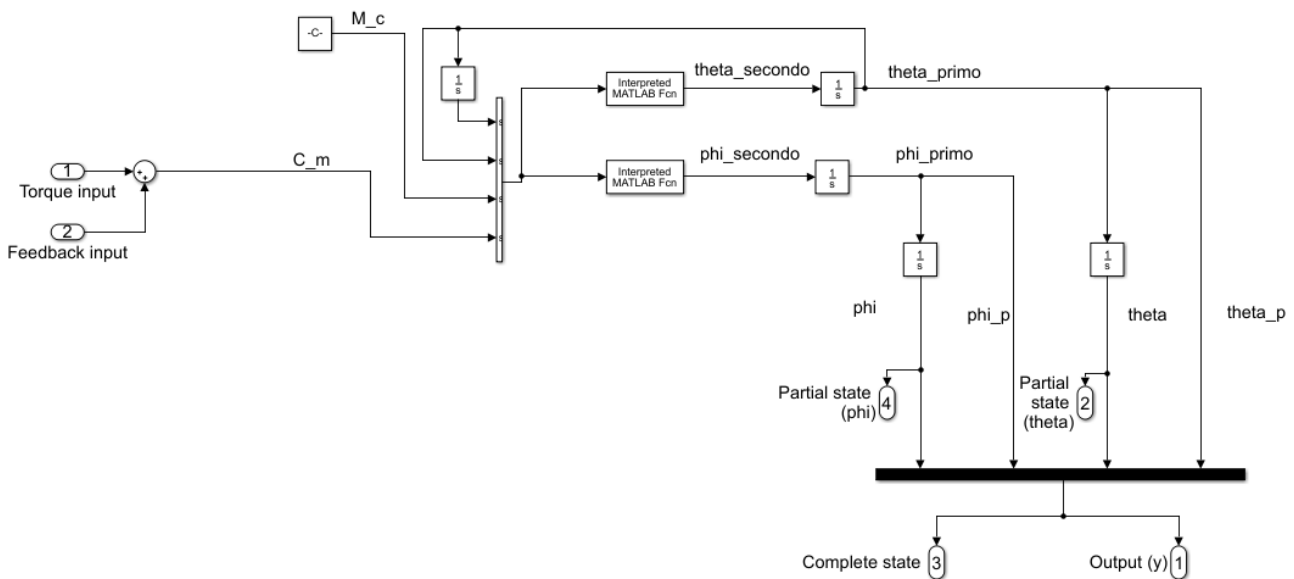


Figure 4.3: Implementazione simulink delle equazioni differenziali

In Fig.4.3 le *interpreted function* altro non sono che $f_{\ddot{\phi}}$ e $f_{\ddot{\theta}}$. A valle di esse sono presenti degli integratori che permettono di ottenere lo stato x completo del sistema. Si può facilmente notare come $\dot{\theta}$ e θ siano collegate direttamente all'input delle *interpreted function*. Si è dunque proceduto a simulare il sistema per verificare la bontà di quanto ottenuto; in particolare, il sistema in anello aperto, dovrebbe oscillare all'infinito vista la mancanza di attriti nel modello.

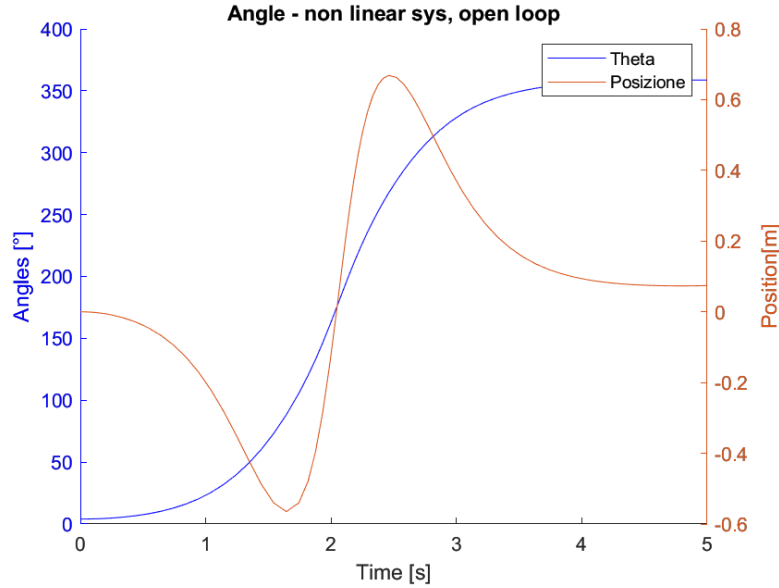


Figure 4.4: Risposta in anello aperto del sistema reale

La simulazione, il cui risultato è riportato in Fig.4.4 è stata svolta per 5 secondi e con un angolo iniziale di 4° ; il grafico mostra dunque l'andamento di θ nel tempo e della posizione che in termini matematici si esprime come $posizione = \phi \cdot r_{ruota}$

4.3 Simulazione del sistema lineare

Si è inoltre creato un altro modello sfruttando il sistema lineare ottenuto prima con l'obbiettivo di semplificare il problema e di velocizzare le simulazioni con lo scopo di testare rapidamente nuove tecniche di controllo che se avessero dato esito positivo sul modello lineare sarebbero poi state testate sul simulink che imita il comportamento reale del sistema.

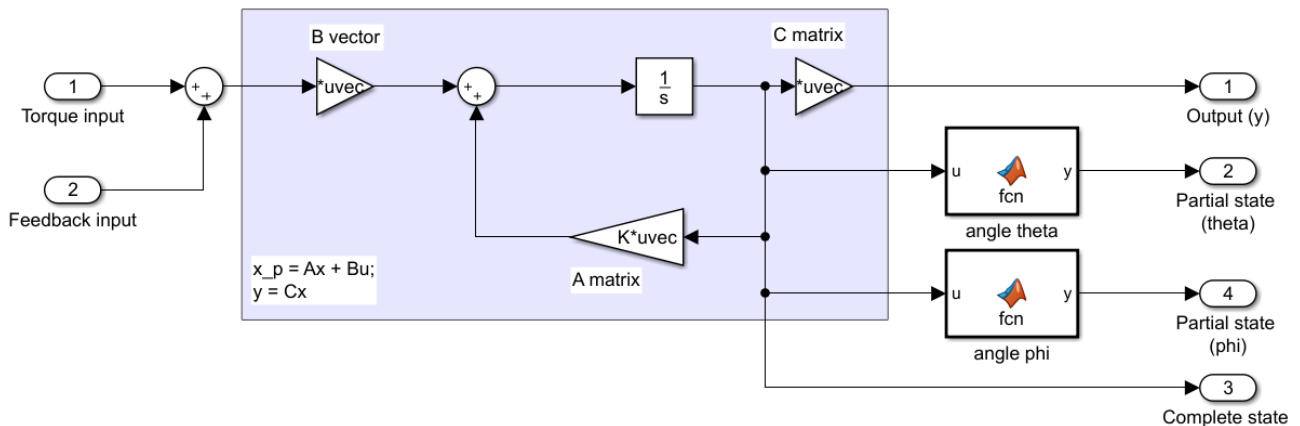


Figure 4.5: Implementazione simulink del sistema linearizzato

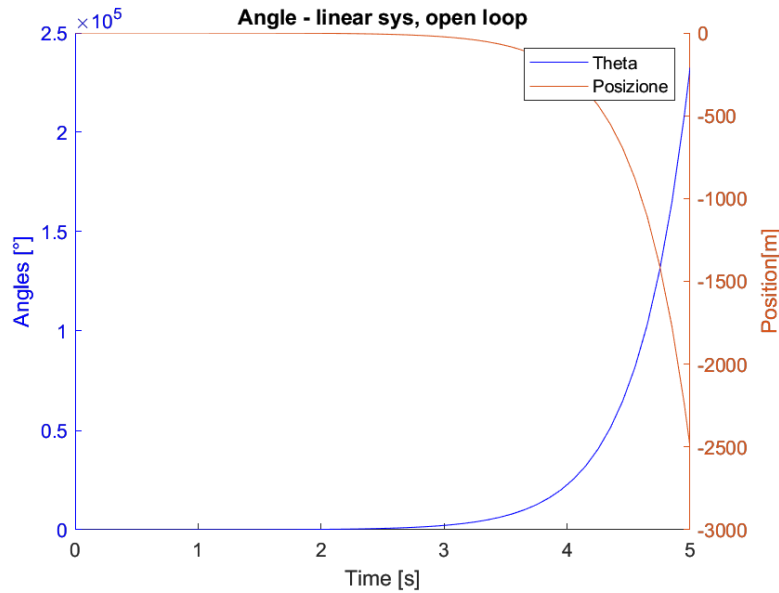


Figure 4.6: Risposta in anello aperto del sistema lineare

In questo caso, la simulazione in anello aperto mostra che il sistema diverge; questo perché la linearizzazione ha senso attorno al punto di equilibrio da cui è stata ottenuta, distante da quel punto il sistema lineare non approssima più il sistema reale ed anche un eventuale controllo ottenuto da esso non garantisce buone performance distante da quel punto. Si nota, in Fig.4.6 che il punto di partenza è 4° e il sistema, lineare, diverge quasi immediatamente; la differenza con la risposta del sistema non lineare in Fig.4.4

Come già verificato in precedenza (sezione 3.2) ci sono due poli reali, uno negativo e uno positivo; questo dimostra come il sistema sia instabile in anello aperto e necessiti di controllo.

4.4 Motore

Il passo successivo è stato quello di andare a modellizzare il motore presente a bordo dello chassis; questo è necessario in quanto si deve tenere conto, in primo luogo, del ritardo che gli attuatori (cioè il motore) introducono nel sistema e che per questo potrebbe diventare instabile.

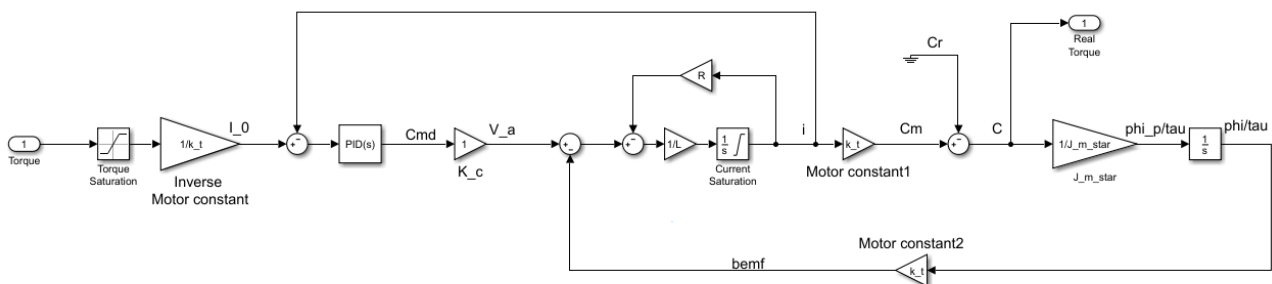


Figure 4.7: Schema a blocchi del motore

I valori delle componenti in Fig.4.7 sono presi dal datasheet del motore. Il controllo del motore DC in questione è stato ottenuto tramite una retroazione in corrente che permette quindi di definire un setpoint alla corrente presente nel motore. Questo si è reso necessario poiché il controllore, attraverso il vettore K e lo stato del sistema, definisce la coppia che il motore dovrebbe erogare. In un motore DC la correlazione tra coppia erogata e corrente esiste ed è ben definita e si tratta di k_t . Il controllore è stato realizzato seguendo metodi già noti in letteratura TODO: scrivere la formula o trovare un posto che la spieghi

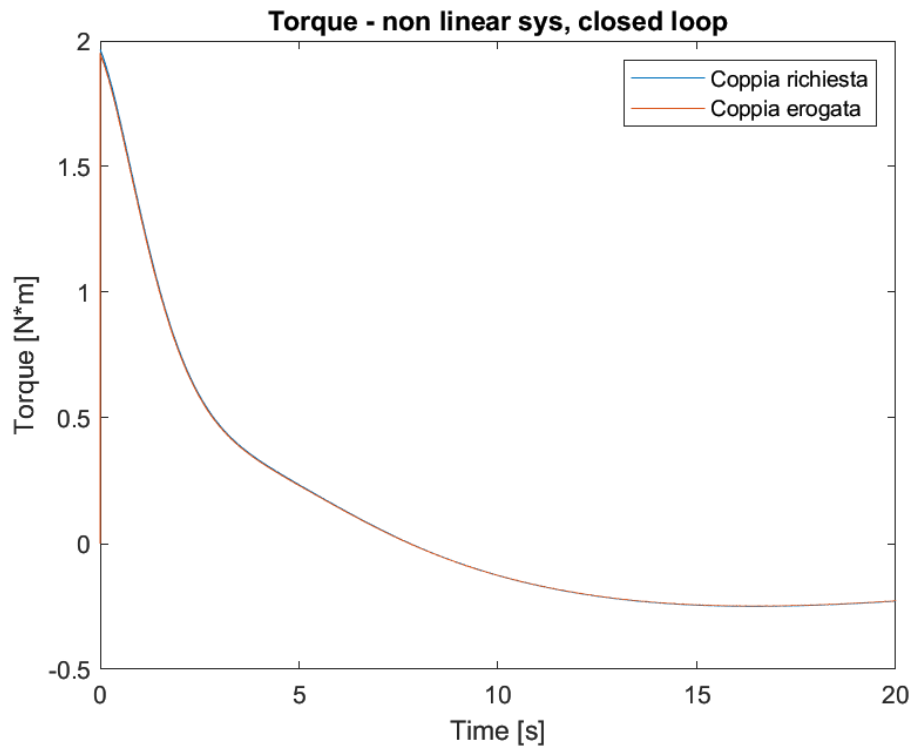


Figure 4.8: Transitorio del motore

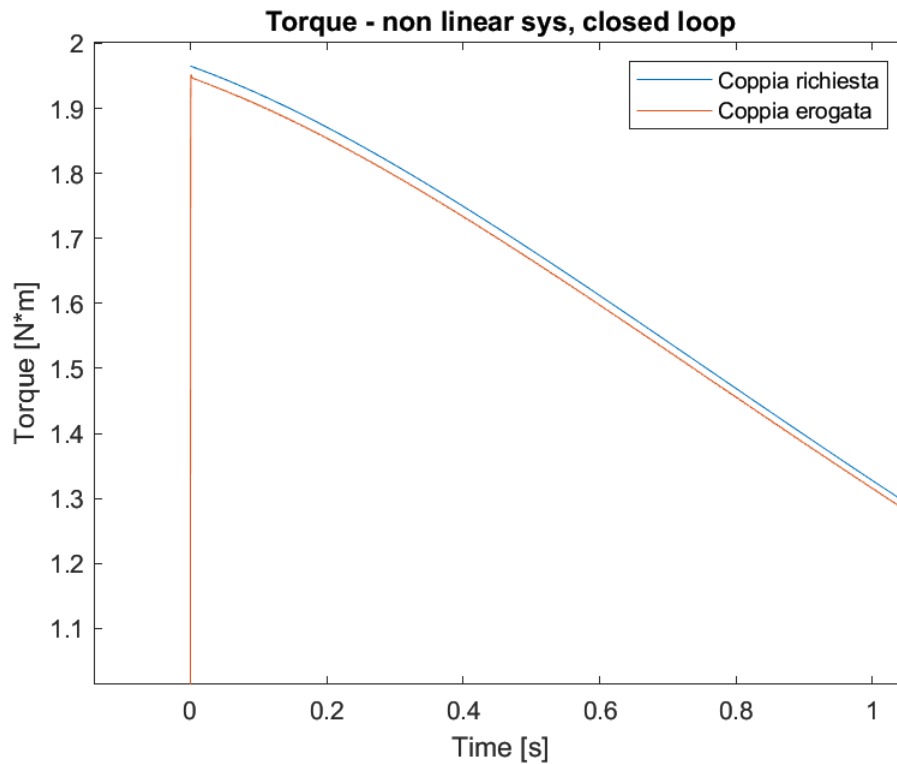


Figure 4.9: Zoom del grafico in figura Fig.4.8

Come si può notare il picco di coppia massimo è minore di 2; questo è dovuto al fatto che, per ragioni esterne, nel motore può scorrere una corrente di massimo 20A e quindi:

$$C_{m,max} = I_{max} \cdot K_{motore} = 20A \cdot 10 \frac{N \cdot m}{A} = 2N \cdot m$$

Un esempio in cui la coppia richiesta supera i $2N \cdot m$ è il seguente:

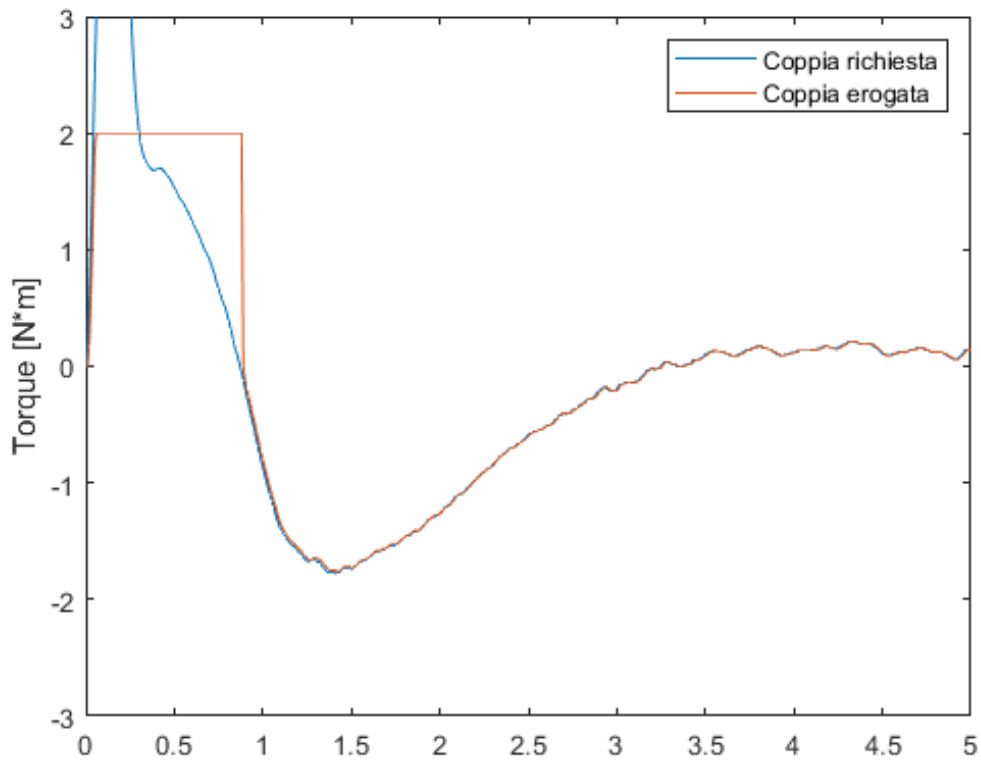


Figure 4.10: Clamp della coppia erogata da parte del motore

In Fig.4.10 è anche possibile notare come l'assenza del blocchetto denominato *Torque saturation*, presente in Fig.4.7, satura l'azione integrale dell'attuatore e inserisce un ritardo non secondario nell'azione di controllo.

4.5 Modello complessivo (al momento)

L'obiettivo di questo paragrafo è quello di fare il punto della situazione del sistema sviluppato fino a questo punto e di sviluppare alcune considerazioni sul lavoro fatto.

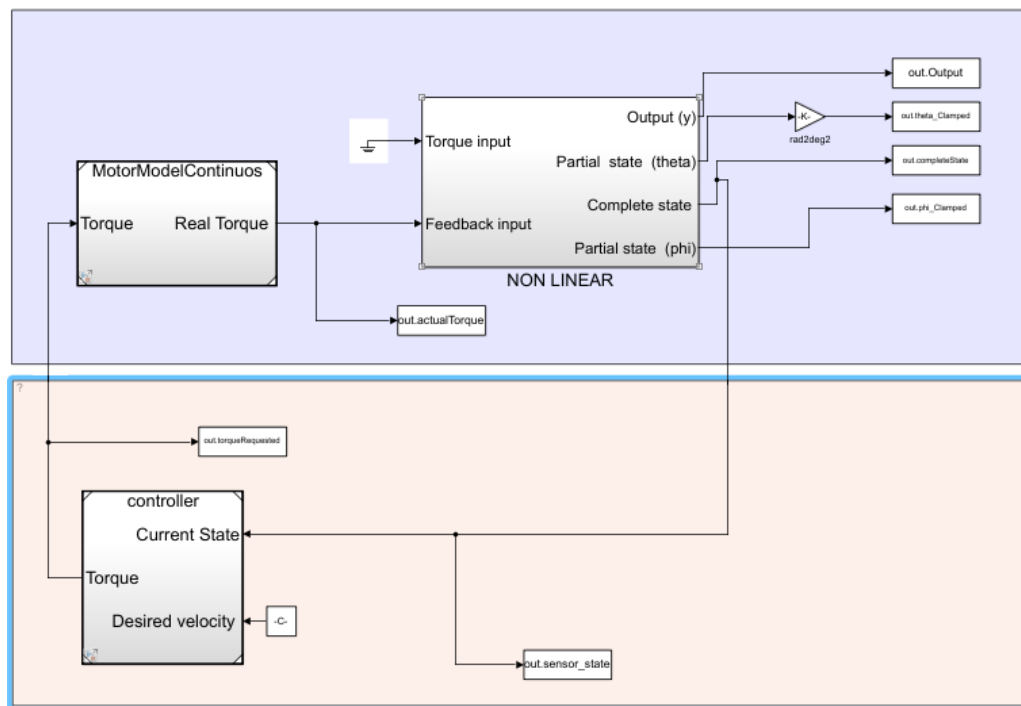


Figure 4.11: Simulink

Come si osserva in Fig.4.11 il sistema al momento comprende tre blocchi:

- Il blocco *non linear*: rappresenta quello che nella realtà sarebbe il sistema reale; si occupa durante la simulazione, dati gli input, di restituire output simili a quelli che si avrebbero in laboratorio utilizzando la macchina vera e propria.
- Il blocco *MotorModelContinuos*: si occupa di simulare la presenza e i transitori dovuti ai motori che nella realtà sono posti a bordo dello chassis.
- Il blocco *controller*: è, dei tre, l'unico blocco che effettivamente dovrebbe essere implementato su un calcolatore. Con i valori simulati dai due blocchi di cui sopra calcola il valore di coppia per il controllo e lo fornisce indietro ai suddetti blocchi per completare la retroazione.

Il sistema al momento è, in linea teorica, a tempo continuo. Nella realtà in un computer e in particolar modo su Simulink la possibilità di far operare i blocchi che simulano il sistema reale a tempo continuo è preclusa. Si è scelto dunque, per quanto fatto finora, di lasciare scegliere al software di Simulink il passo della simulazione ed in particolar modo utilizzare un passo variabile. Questo permette, nei punti in cui le variazioni sono spiccate (ad esempio quando il sistema si avvia) di utilizzare un passo di simulazione anche dell'ordine dei nanosecondi che approssima quasi perfettamente l'esecuzione a tempo continuo.

4.6 Discretizzazione

Come già detto sopra, il controllore è necessario che sia implementato su un calcolatore e quindi verrà eseguito a tempo discreto; per tenere conto di questa caratteristica è necessario discretizzare il controllore e acquisire gli input a tempo discreto:

- all'ingresso del blocco *controller* è stato posto un *Sample & Hold*; questo blocco si occupa di acquisire ogni tempo di sampling T_s il valore in input e mantenerlo inalterato fino alla lettura successiva
- all'uscita del *controller* si è posizionato un altro *Sample & Hold* per le stesse ragioni

- il *controller* è stato trasformato a tempo discreto; TODO inseriamo la cosa dei poli discreti o diciamo che sono uguali essendo un proporzionale? A differenza del controllore della retroazione dello stato, il controllore che chiude la retroazione in velocità va ricalcolato tenendo conto del T_s ; fortunatamente Simulink fa da solo questa conversione se si setta il blocco simulink *Controllore PID* come integratore a tempo discreto fornendo il T_s e la costante moltiplicativa.

Il controllore a tempo discreto ha dunque questo aspetto nel modello simulink implementato:

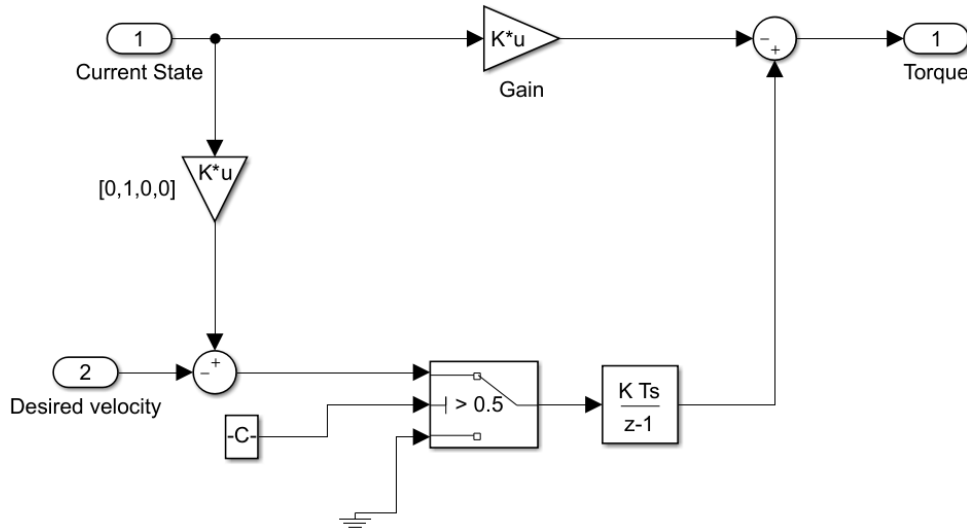


Figure 4.12: Modello Simulink del controllore

Per le stesse ragioni pratiche anche il modello del motore ha subito una conversione: il controllore dell'anello di retroazione in corrente sarà un software eseguito ciclicamente su di un calcolatore e va dunque implementato a tempo discreto. Ricordando che la funzione che la F.d.T. del controllore *PI* a tempo continuo era:

$$\frac{0.001216s + 0.9965}{0.00122s}$$

e applicando il metodo di Tustin da tempo continuo a discreto con un $T_s = 0.001$ si ottiene che:

$$\frac{1.405z - 0.5881}{z - 1}$$

4.7 Sensori

Sempre con l'obiettivo di rendere la simulazione più raffinata e accurata possibile sono stati inseriti all'interno del file simulink anche dei modelli che simulano la presenza dei sensori. Questo è necessario ed importante poiché nella realtà non esiste un modello matematico che genera i dati che poi sono dati in pasto al controllore ma si devono invece usare dei sensori che danno una stima dello stato del sistema. Un sensore presenta due principali caratteristiche:

- Rumore: è stato modellizzato come un *Band-Limited White Noise* con *Noise power* = [0.00000001] e *Sample time* = 0.001.
- Quantizzazione: la lettura dei dati da un sensore, oltre ad avvenire ad un intervallo di tempo minimo e regolare, mostra anche un errore dovuto alla limitata sensibilità del sensore stesso o dell' *ADC* che effettua la lettura. Il blocco simulink *Quantizer* ha l'esatto scopo di simulare questo comportamento presente nei sensori reali.

TODO come spiego che mi arriva in ingresso theta e phi e non le derivate? Sul sistema sono presenti diversi sensori:

- Encoder incrementale: si occupa di misurare la rotazione della singola ruota

- I.M.U. : stima l'inclinazione dello chassis
- Sensore di corrente: misura la corrente presente nel motore

Per come si presenta il sistema, sia l'*encoder* incrementale sia la *I.M.U.* restituiscono rispettivamente una stima di ϕ e θ . Per modellizzare quanto detto finora riguardante i sensori ed in particolare per l'*encoder* e l'*I.M.U.* è stato approntato il seguente blocco simulink:

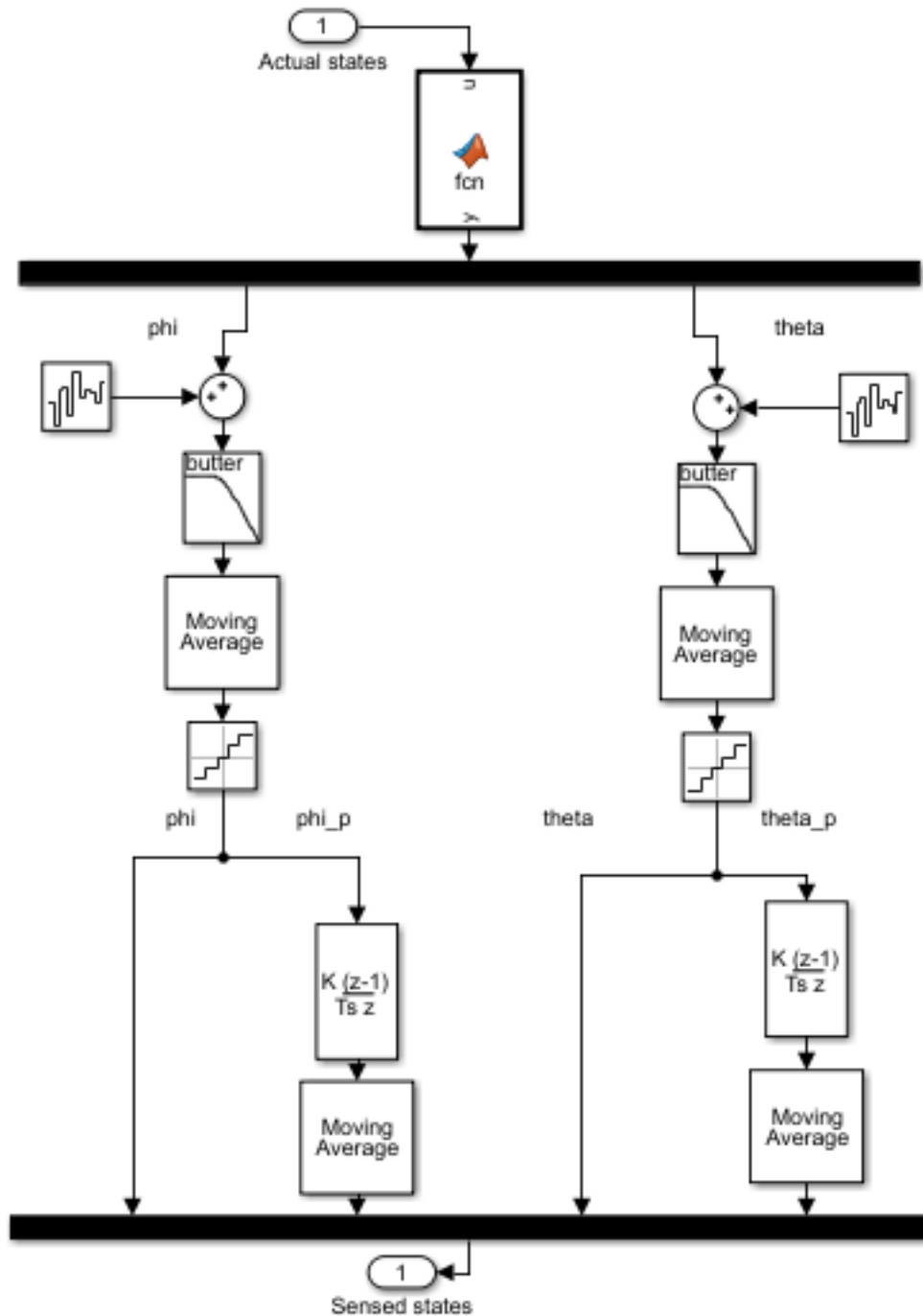


Figure 4.13: Modello Simulink dei sensori *encoder* e *I.M.U.*

Si può notare come in Fig.4.13 siano presenti due blocchi *Moving average* per ciascun sensore; questo si è reso necessario per due ragioni. La prima è che, presentando il sistema del rumore, era necessario rimuoverlo attraverso un filtro digitale; la seconda ragione è che, vista la presenza della quantizzazione, la derivazione a tempo discreto del segnale presenta molti picchi e molti valori nulli; è quindi necessario filtrare questi valori prima di passarli al controllore per evitare picchi di coppie

troppo alti oppure nulli in brevi intervalli di tempo. Si presenta ora la necessità di modellizzare la presenza del sensore di corrente nel motore; per fare ciò si è modificato solamente l'anello di retroazione esterno che permette il controllo del motore in corrente:

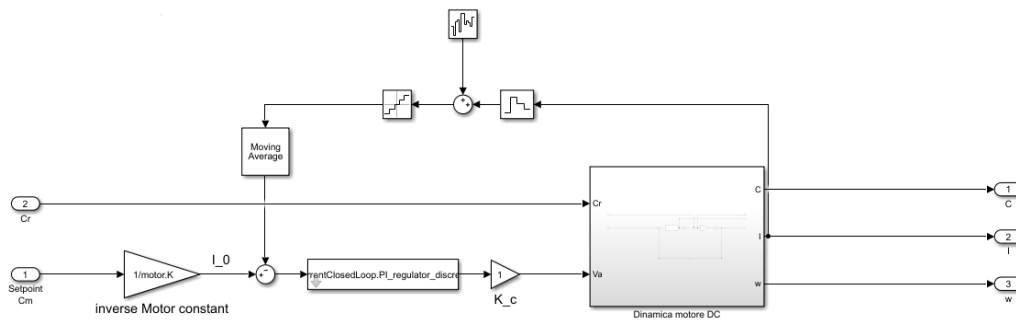


Figure 4.14: Modello Simulink discreto del motore

Si può notare in Fig.4.14 che il controllore è quello discreto presentato prima, mentre è stato inserito anche in questo caso un rumore sulla misura, una quantizzazione e un filtro sul segnale da passare al controllore. Da sottolineare che ognuno dei tre blocchi *Quantizer* ha un valore diverso dovuto al fatto che ognuno dei sensori ha sensibilità diverse:

- Encoder incrementale: presenta una sensibilità di $2^{11} \text{bit} = 2048$ diverse combinazioni e letture
- I.M.U. : sensibilità del giroscopio di $131 \frac{s}{deg}$
- Sensore di corrente: il valore del sensore di corrente è un voltaggio letto da un Arduino Uno con un ADC con sensibilità di 10 bit

4.8 RTOS - XENOMAI

Riportiamo di seguito una breve descrizione del lavoro portato avanti durante questa settimana per quanto concerne la parte RTOS.

- Come prima cosa siamo andati a rivedere il file *.cpp* che avevamo prodotto nelle settimane scorse. In particolare abbiamo dovuto ritoccare la parte relativa alla lettura dei dati dal file, andando ad utilizzare le API presenti in ambiente c invece di quelle relative a cpp. Abbiamo quindi provato e testato la lettura da file tramite il tool DevC++, facilitando così il testing e il debug;
- Siamo andati ad installare una macchina virtuale con installato a bordo **debian 9.8.0** e la patch **Xenomai 3.0.8** ([link](#));
- Essendo debian un OS interamente fruibile da linea di comando, siamo andati a prendere confidenza con l'ambiente, andando a capire l'organizzazione e chiamando gli update/upgrade necessari;
- Prima di andare ad importare il file completo (di cui abbiamo parlato al punto iniziale), siamo andati ad eseguire alcuni esempi iniziali (una sorta di HelloWorld):
 - Abbiamo per primo cosa studiato questo approccio iniziale ([link](#))
 - Siamo andati ad impostare il funzionamento di un task (dummy) periodico, come se si trattasse appunto del main loop di Arduino ([link](#))

- Lo step successivo è stato quello di importare il codice scritto e testato in ambiente DevC++ in xenomai, provandolo e introducendo alcuni dettagli aggiuntivi:

```

GNU nano 2.7.4                                File: Makefile
XENO_CONFIG := /usr/xenomai/bin/xeno-config

CFLAGS := $(shell $(XENO_CONFIG) --posix --alchemy --cflags)
LDFLAGS := $(shell $(XENO_CONFIG) --posix --alchemy --ldflags)

CC := gcc
EXECUTABLE := VAB_RealTime

all: $(EXECUTABLE)

%.c:
    $(CC) -o $@ $< $(CFLAGS) $(LDFLAGS)

clean:
    rm -f $(EXECUTABLE)

[ Read 15 lines ]
Get Help  Write Out  Where Is  Cut Text  Justify  Cur Pos  Prev Page
Exit      Read File  Replace   Uncut Text To Spell  Go To Line Next Page

```

Figure 4.15: Per specificare quale file .c compilare

- Periodo di funzionamento definito tramite questi comandi

```

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2289
2290
2291
2292
2293
2294
2295
2296
2297
2298
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
2370
2371
2372
2373
2374
2375
2376
2377
2378
2379
2380
2381
2382
2383
2384
2385
2386
2387
2388
2389
2390
2391
2392
2393
2394
2395
2396
2397
2398
2399
2400
2401
2402
2403
2404
2405
2406
2407
2408
2409
2410
2411
2412
2413
2414
2415
2416
2417
2418
2419
2420
2421
2422
2423
2424
2425
2426
2427
2428
2429
2430
2431
2432
2433
2434
2435
2436
2437
2438
2439
2440
2441
2442
2443
2444
2445
2446
2447
2448
2449
2450
2451
2452
2453
2454
2455
2456
2457
2458
2459
2460
2461
2462
2463
2464
2465
2466
2467
2468
2469
2470
2471
2472
2473
2474
2475
2476
2477
2478
2479
2480
2481
2482
2483
2484
2485
2486
2487
2488
2489
2490
2491
2492
2493
2494
2495
2496
2497
2498
2499
2500
2501
2502
2503
2504
2505
2506
2507
2508
2509
2510
2511
2512
2513
2514
2515
2516
2517
2518
2519
2520
2521
2522
2523
2524
2525
2526
2527
2528
2529
2530
2531
2532
2533
2534
2535
2536
2537
2538
2539
2540
2541
2542
2543
2544
2545
2546
2547
2548
2549
2550
2551
2552
2553
2554
2555
2556
2557
2558
2559
2560
2561
2562
2563
2564
2565
2566
2567
2568

```

```

GNU nano 2.7.4 File: VAB_RealTime.c

    for(int i = 0; i<WINDOW_WIDTH;i++){
        media_pesata = media_pesata + 1/WINDOW_WIDTH * media_mobile[i];
    }
    return media_pesata;
}

void wrap() {
    printf("START");
    int counter = 0;
    RTIME period = 1000000000;
    rt_task_set_periodic(NULL,TM_NOW,period);
    RTIME now = rt_timer_read();
    RTIME before= now-period;
    while(1){
        now = rt_timer_read();
        read_sensor_phi(counter);
        read_sensor_theta(counter);
        calculate_phi_p(counter);
        calculate_theta_p(counter);
        read_k();
        controllore();
        controllore_motore(counter);
        counter = (counter+1)%WINDOW_WIDTH;
        printf("Actual motor torque: %f\n", actual_cm);
        printf("Elapsed time: %f\n",Ts);
        before= now;
    }

    return;
}

```

Figure 4.16: Definizione e temporizzazione del *WHILE*

- Salvataggio dei parametri temporali now ad inizio del ciclo e update del parametro before alla fine del ciclo: questo per facilitare, in qualsiasi punto del codice, l'accesso ai parametri temporali per eventuali necessità di calcolo; inoltre abbiamo utilizzato questi parametri per definire l'intervallo di tempo su cui andare a calcolare la derivata;
- Abbiamo seguito questo approccio nel ciclo principale eseguito a frequenza fissa: lettura file e sensori → algoritmo/elaborazione → produzione dei risultati di interesse;

```

Gain value parsed: -> K0 = 0.000000
Gain value read: -> K1 = 0
Gain value parsed: -> K1 = 0.000000
Gain value read: -> K2 = 60
Gain value parsed: -> K2 = 60.000000
Gain value read: -> K3 = 0
Gain value parsed: -> K3 = 0.000000
Actual motor torque: -130.967766
Elapsed time: 0.001000
tempGainFile is open
Gain value read: -> K0 = 0
Gain value parsed: -> K0 = 0.000000
Gain value read: -> K1 = 0
Gain value parsed: -> K1 = 0.000000
Gain value read: -> K2 = 60
Gain value parsed: -> K2 = 60.000000
Gain value read: -> K3 = 0
Gain value parsed: -> K3 = 0.000000
Actual motor torque: -131.039428
Elapsed time: 0.001000
tempGainFile is open
Gain value read: -> K0 = 0
Gain value parsed: -> K0 = 0.000000
Gain value read: -> K1 = 0
Gain value parsed: -> K1 = 0.000000
Gain value read: -> K2 = 60
Gain value parsed: -> K2 = 60.000000
Gain value read: -> K3 = 0
Gain value parsed: -> K3 = 0.000000
Actual motor torque: -131.111109
Elapsed time: 0.001000
root@xenomai308:/home/Laboratorio_SistemiMeccatroniciIII/functions/C-XENOMAI#

```

Figure 4.17: Esecuzione a tempo fissato delle funzioni di lettura e calcolo

- Il thread relativo al server opc - ua non è ancora stato gestito: esso sarà ovviamente lasciato in secondo piano rispetto al thread principale temporizzato alla frequenza specifica;

Part I

Allegati

Appendix A

Appendice A

Riportiamo di seguito il codice prodotto per la gestione della comunicazione tramite protocollo OPC-UA.

```
10 ##### LIBRARY IMPORT #####
11 from opcua import Server
12 from random import randint
13 from datetime import datetime
14 import os
15 import time
16 import socket
17 #####
18
19 ##### GET LOCAL MACHINE IP #####
20 s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
21 s.connect(("8.8.8.8", 80))
22 id = s.getsockname()[0]
23 s.close()
24 #####
25
26 ##### SET SERVER INSTANCE #####
27 server = Server()
28 url = "opc.tcp://" + id + ":4840"
29
30 server.set_endpoint(url)
31
32 name = "V.A.B. - Server Raspberry PI"
33 addspace = server.register_namespace(name)
34
35 node = server.get_objects_node()
36 Param = node.add_object(addspace, "Gain parameters")
37 #####
38
39 ##### FILE READING #####
40 fileName_temp = "GainParametersToController.txt"
41 fileName_confirmed = "GainParametersConfirmed.txt"
42
43 lineHeaders = ["K_phi ", "K_phi_p ", "K_theta ", "K_theta_p "]
44 initialValueGains = [0, 0, 0, 0]
45
46 if os.path.exists(fileName_confirmed):
47     print("Get initial values from file and create a temp one")
48     file_object_def = open(fileName_confirmed, "r+")
49     lines = file_object_def.readlines()
50     count = 0
51     # Strips the newline character
52     for line in lines:
53         print(line.strip())
54         values = line.split()
55         if count <= 3:
56             initialValueGains[count] = int(values[1])
```

```

57         else:
58             timestamp = str(values[0]) + str(values[1])
59             count = count + 1
60             file_object_def.close()
61
62             file_object_temp = open(fileName_temp, "w")
63             for i, header in zip(initialValueGains, lineHeaders):
64                 print(header + str(i))
65                 file_object_temp.write(header + str(i) + "\n")
66
67             # Time stamp from confirmed file
68             file_object_temp.write(timestamp)
69             file_object_temp.close()
70     else:
71         print("Creating a new files and set parameters to zero")
72         file_object_temp = open(fileName_temp, "w")
73         file_object_def = open(fileName_confirmed, "w")
74
75         for i, header in zip(range(4), lineHeaders):
76             print(header + "0")
77             file_object_temp.write(header + "0\n")
78             file_object_def.write(header + "0\n")
79
80         # current date and time
81         now = datetime.now()
82         timestamp = datetime.timestamp(now)
83         dt_object = datetime.fromtimestamp(timestamp)
84         file_object_temp.write(str(dt_object))
85         file_object_def.write(str(dt_object))
86
87         file_object_temp.close()
88         file_object_def.close()
89     #####
90
91     ##### PARAMETERS INIT #####
92     K_phi = Param.add_variable(addspace, "K_phi", initialValueGains[0])
93     K_phi_p = Param.add_variable(addspace, "K_phi_p", initialValueGains[1])
94     K_theta = Param.add_variable(addspace, "K_theta", initialValueGains[2])
95     K_theta_p = Param.add_variable(addspace, "K_theta_p", initialValueGains[3])
96     submit_to_controller = Param.add_variable(addspace, "Submit change to controller",
97         False)
98     submit_to_file = Param.add_variable(addspace, "Store definitively in file", False)
99     shut_down = Param.add_variable(addspace, "SHUT DOWN SERVER", False)
100
101     K_phi.set_writable()
102     K_phi_p.set_writable()
103     K_theta.set_writable()
104     K_theta_p.set_writable()
105     submit_to_controller.set_writable()
106     submit_to_file.set_writable()
107     shut_down.set_writable()
108     #####
109
110     server.start()
111
112     print("Server started at {}".format(url))
113
114     K1 = K_phi.get_value()
115     K2 = K_phi_p.get_value()
116     K3 = K_theta.get_value()
117     K4 = K_theta_p.get_value()
118
119     while True:
120         save_on_temp_file = submit_to_controller.get_value()
121         save_on_definitively_file = submit_to_file.get_value()
122         exit = shut_down.get_value()

```

```

122
123     if save_on_definitively_file:
124         # Write on file that store gains considered stable
125         K1 = K_phi.get_value()
126         K2 = K_phi_p.get_value()
127         K3 = K_theta.get_value()
128         K4 = K_theta_p.get_value()
129
130         gains = [K1, K2, K3, K4]
131         file_object_conf = open(fileName_confirmed, "w")
132         for header, gain in zip(lineHeaders, gains):
133             file_object_conf.write(header + str(gain) + "\n")
134
135         # current date and time
136         now = datetime.now()
137
138         timestamp = datetime.timestamp(now)
139         dt_object = datetime.fromtimestamp(timestamp)
140         file_object_conf.write(str(dt_object))
141         file_object_conf.close()
142
143         submit_to_file.set_value(False)
144         print("Confirmed data")
145
146     if save_on_temp_file:
147         # Write on file that is read from controller on Raspberry
148         K1 = K_phi.get_value()
149         K2 = K_phi_p.get_value()
150         K3 = K_theta.get_value()
151         K4 = K_theta_p.get_value()
152
153         gains = [K1, K2, K3, K4]
154         file_object_temp = open(fileName_temp, "w")
155         for header, gain in zip(lineHeaders, gains):
156             file_object_temp.write(header + str(gain) + "\n")
157
158         # current date and time
159         now = datetime.now()
160
161         timestamp = datetime.timestamp(now)
162         dt_object = datetime.fromtimestamp(timestamp)
163         file_object_temp.write(str(dt_object))
164         file_object_temp.close()
165
166         submit_to_controller.set_value(False)
167         print("Submitted data to temp file that will be read from raspberry")
168
169     if not save_on_temp_file and not save_on_definitively_file:
170         # No change submitted
171         K = [K1, K2, K3, K4]
172         print(K)
173         time.sleep(2)
174
175     if exit:
176         print("Shut down server...")
177         break
178
179 if os.path.exists(fileName_temp):
180     print("Removing temp file...")
181     os.remove(fileName_temp)
182 else:
183     print("The temp file does not exist")
184 print("Server stopped")

```


Bibliography

- [1] *Controllo tramite retroazione dello stato, Controlli automatici, Fabio Previdi* https://cal.unibg.it/wp-content/uploads/controlli_automatici/Lez06.pdf