FACOLTA' DI INGEGNERIA

Corso di Laurea in Ingegneria Elettronica

PROGETTO DI SISTEMI DI CONTROLLO IN AMBIENTE MATLAB/SIMULINK PER LA PIATTAFORMA LEGO MINDSTORMS

RELATORE:

Ing. Michele Basso …….

CORELATORE:

Dott. Massimo Vassalli ……..

CANDIDATI:

Elena Barani ……..

Piero Danti ……..

Anno Accademico 2007/2008

Indice

Introduzione

Negli ultimi anni la LEGO ha investito molto nel campo della robotica. La linea LEGO MINDSTORMS ha rivoluzionato la concezione delle persone riguardo all'azienda LEGO. I classici mattoncini sono stati affiancati da un prodotto che ha contribuito ad allargare notevolmente la fascia di età dei fan della LEGO. La filosofia dell'azienda è caratterizzata dalla volontà di rendere i prodotti della linea LEGO MINDSTORMS affascinanti, ma utilizzabili, sia da esperti nel campo della robotica sia da principianti in giovane età.

La LEGO ha speso molte energie per la creazione di un percorso didattico adeguato alle varie fasi della crescita del ragazzo. Il punto di forza della linea LEGO MINDSTORMS sta nelle numerose possibilità che mette a disposizione per costruire un robot ideato secondo la propria immaginazione e i propri desideri. La nuova generazione di LEGO MINDSTORMS si basa sul successo del "Robotics Invention System", rinomato in tutto il mondo, ma è di più veloce e facile impiego: nell'arco di qualche ora si può costruire e programmare un robot LEGO. L'interesse degli utenti più esperti è stato stimolato da nuove funzioni tecniche e dalla commercializzazione di sensori ancora più efficienti. L'Hitechnic ha stipulato un accordo di collaborazione con la LEGO che le permette di produrre e vendere accessori hardware certificati dal gruppo LEGO. Questo accordo rappresenta un'apertura, da parte della LEGO, verso la collaborazione con altre aziende e verso la trasformazione del software in open source. Anche Paal Smith Myer, dirigente LEGO, ha mostrato il suo entusiasmo verso questa collaborazione, poiché la ritiene un' opportunità per mostrare le potenzialità della piattaforma LEGO e per aumentare la capacità del prodotto di soddisfare le esigenze dell'utente. In un articolo della rivista PC Magazine "LEGO MINDSTORMS NXT è stato definito come un prodotto all'avanguardia nella tecnologia, in grado di combinare la creatività con la scienza. Il "cuore" del nuovo

prodotto è l'NXT, un microprocessore autonomo da 32 bit che può essere programmato sia da PC che da MAC.

Questa tesi tratta di esperimenti di controllo effettuati mediante l'utilizzo del sensore giroscopio dell'Hi-Technic e dei prodotti della linea LEGO Mindstorms. Si è costruito e si sono studiate le caratteristiche di un Legway, che da un punto di vista fisico può essere modellato come un pendolo inverso con due ruote.L' obbiettivo prefissato è quello di progettare un controllore, che riesca a far stare in equilibrio il Legway, ricorrendo al linguaggio di programmazione MATLAB/Simulink e ai toolbox Real-Time Workshop, Real-TimeWorkshop Embedded Coder, Embedded Coder Robot e Virtual Reality Toolbox. La complessità di questo software è compensata dalle enormi potenzialità che offre per lo sviluppo di un modello complesso e per la sua analisi. Infatti dopo aver modellato il Legway si sono potuti modificare, con estrema velocità e semplicità, i sensori utilizzati per il controllo del sistema fisico scelto. Sono stati realizzati tre Legway diversi: uno con il giroscopio e con il sensore ad ultrasuoni, un altro con il sensore ad ultrasuoni e con quello di luce e un terzo con due sensori di luce ed uno ad ultrasuoni. Grazie all'utilizzo del Bluetooth sono stati raccolti i dati ottenuti durante le simulazioni e questo ci ha consentito di modificare, in modo empirico, alcuni parametri all'interno degli schemi a blocchi costruiti. Utilizzando il toolbox Virtual Reality si è ottenuta una visualizzazione 3-D del modello creato capace di muoversi in un mondo di realtà virtuale, senza ricorrere all'implementazione fisica.

L'esperienza fatta su questo modello piuttosto semplice è stata molto utile perché ci ha permesso di familiarizzare con il Model-Based Design e con i toolbox di MathWorks che insieme costituiscono una tecnica affermata per sistemi di controllo embedded anche molto complessi. Questa tecnica viene anche utilizzata per il controllo di satelliti, di velivoli e di altre applicazioni aereospaziali, per processi di controllo e per macchinari industriali.

La tesi è strutturata come segue:

- Capitolo 2: si descrivono i dispositivi hardware disponibili e si fa una valutazione dei possibili software da utilizzare. Poi si espongono le

caratteristiche del Model-Based Design, dell' ambiente Matlab/ Simulink e dei toolbox utilizzati.

- Capitolo 3: si analizza la struttura del Legway, il suo modello, la tecnica di controllo impiegata, gli schemi a blocchi utilizzati e le modifiche necessarie per passare da un Legway ad un altro che possiede dei sensori diversi.
- Capitolo 4: si riportano i risultati sperimentali ottenuti.

Capitolo 2

2 LEGO Mindstorms

2.1 HARDWARE

Lego Mindstorms è una linea di prodotti LEGO che fonde i tradizionali mattoncini con sensori, attuatori, mattoncini programmabili e pezzi di LEGO-Technic (come assi,travi e ingranaggi) per la costruzione di robot ed altri sistemi automatici. Di questa linea di prodotti esiste una versione educativa: "LEGO Mindstorms for School" che viene fornita con un software di programmazione basato sulla GUI ROBOLAB.

Tutti i tipi di sistemi elettromeccanici esistenti nella realtà posso essere modellati con i Mindstorms.

Il primo prodotto rilasciato dalla LEGO in questo campo è stato l' RCX (1998), antenato dell'attuale NXT(2006): entrambi definiti mattoncini (o brick) programmabili.

L'RCX (Fig. 2.1) contiene un microcontrollore Renesas H8/300 come CPU interna e possiede 32 kB di RAM dove vengono salvati il firmware e i programmi dell'utente. Il brick viene programmato scaricando un programma (che può essere scritto in vari linguaggi di programmazione), da un PC o da un Macintosh, sulla sua RAM attraverso una speciale interfaccia ad infrarossi. Quando l' utente avvia l'applicazione il robot costruito può funzionare in completa autonomia reagendo agli stimoli interni ed esterni in base alle istruzioni contenute nel programma. Oltre alla porta ad infrarossi, ci sono anche tre porte di ingresso per i sensori e tre porte di uscita per i motori. C'è anche uno schermo LCD che mostra: lo stato della batteria, lo stato delle porte I/O, il programma in esecuzione e altre informazioni. Le prestazioni dell'RCX sono molto inferiori rispetto a quelle dell' NXT (qualità dei sensori, velocità, modalità di comunicazione, classi del processore) ma nonostante ciò l'RCX ha costituito un importante passo per lo sviluppo della tecnologia LEGO nel settore della robotica.

Figura 2.1: RCX brick

L'NXT ha un processore a 32 bit Atmel AT91SAM7S256 (classe ARM7) a 48 MHz, con 256kB di flash memory e 64kB di RAM, un coprocessore 8 bit Atmel ATmega48 a 4 MHz, con 4kB di flash e 512 byte di RAM, uno schermo LCD con una risoluzione di 100x64 pixel, una porta USB 2.0 e connettività Bluetooth (per trasferire,senza fili, programmi all'NXT o per offrire un controllo remoto attraverso un cellulare,un palmare o un gamepad) . Il Mindstorms NXT (Fig. 2.2) possiede quattro porte di ingresso e tre di uscita, ma avendo delle connessioni digitali, sarà possibile aumentarne il numero con dei moduli esterni. I connettori non sono gli stessi dell'RCX e utilizzano porte simili ad un connettore RJ-11. Integrato nel mattoncino c'è un altoparlante da 8 kHz che permette di dar voce ai progetti. Il mattoncino richiede 6 batterie di tipo AA oppure la Batteria al Litio della casa.

Il Lego Mindstorms NXT è venduto in due versioni: Retail e Education Base Set. La versione Retail è fornita col software di programmazione NXT-G, invece la versione educativa è venduta con batteria al litio e caricabatterie ma non contiene software. Quest'ultimo è venduto separatamente, con tre licenze distinte (Personal, Classroom, Site).

L’ affermarsi del LEGO Mindstorms è testimoniata dall’ ampia e consolidata comunità di professionisti ed amatori di ogni età coinvolti nella creazione di nuovi progetti, nella condivisione di questi e nel continuo sviluppo di tecniche di programmazione compatibili con i prodotti LEGO.
La LEGO incoraggia gli appassionati che vogliono sviluppare o migliorare il software dell’NXT rendendolo Open Source.

Figura 2.2: Kit base dell' NXT

- **Servo Motori:** Per ogni NXT si possono utilizzare tre diversi servo motori (Fig. 2.3) del peso di 60 grammi ciascuno, che permettono al robot di muoversi. Questi hanno incorporato un sensore di rotazione che misura la velocità e la distanza e le restituisce all’ NXT. In questo modo è possibile controllare il motore con un’accuratezza di un grado. Inoltre si possono sincronizzare più motori alla stessa velocità utilizzando, ad esempio, il blocco “move” nel software LEGO Mindstorms NXT.

Figura 2.3: Motore e interno del motore

- **Sensore di contatto:** Il robot acquisisce il senso del tatto grazie a questo sensore (Fig. 2.4). Quest'ultimo distingue tra due stati : quando viene premuto da qualcosa e quando invece viene rilasciato. Inoltre è anche possibile riconoscere quando il pulsante viene premuto in maniera continuativa.

Figura 2.4: Sensore di contatto

- **Sensore di luce:** E' uno dei due sensori che garantisce al robot il senso della vista (insieme al sensore ad ultrasuoni). Il sensore di luce (Fig. 2.5) permette di distinguere tra un ambiente illuminato ed uno buio, misurando

l'intensità della luce presente. E' anche in grado di distinguere i colori misurdo l'intensità della luce riflessa sulle superfici colorate o neutre (grazie a questa caratteristica un robot dotato di sensore di luce può seguire una linea colorata). Una funzionalità alternativa del sensore è quella di calcolare la distanza dalla superficie di appoggio: prima emettendo una luce di colore rosso e poi misurando la quantità di luce riflessa. L' estrema sensibilità verso la luce e verso il passaggio da una superficie ad una differente rappresenta un ostacolo rilevante nell' utilizzo di questo tipo di sensore. Non è utilizzabile in stanze fortemente illuminate in quanto il sensore non riesce a rilevare variazioni utili e rimane sempre intorno al valore massimo consentito.

Figura 2.5: Sensore di luce

- **Sensore ad ultrasuoni:** Anche questo sensore (Fig. 2.6) serve per il senso della vista infatti permette di determinare la presenza e la distanza, in pollici o in centimetri, di un oggetto e quindi permette di evitare ostacoli prima di un eventuale impatto. Il range di visibilità del sensore varia da 0 cm a 255 cm con una precisione di ± 3 cm. Il principio che utilizza è lo stesso usato dai pipistrelli per orientarsi, cioè emette un'onda e misura il tempo che questa impiega a tornare indietro. Difficoltà rilevanti sono costituite dalle superfici curvilinee, morbide o piccole. Permette anche di determinare il movimento dell'oggetto rilevato. Particolare attenzione deve essere prestata se più sensori ad ultrasuoni operano nella stessa stanza poiché potrebbero interferire.

Figura 2.6: Sensore ad ultrasuoni

- **Sensore di suono:** E' in grado di misurare i livelli di rumorosità sia in dB che in dBA consentendo d'identificare le differenze di tono e di timbro.
    - ✓ dB: il sensore misura con la stessa sensibilità tutti i suoni e per questo prende in considerazione anche quei suoni che non sono percepibili dall'orecchio umano.
    - ✓ dBA: la sensibilità del sensore è adattata alla sensibilità dell'orecchio umano.

Il sensore di suono (Fig 2.7) misura livelli di pressione sonora fino a 90 dB (che corrisponde al rumore prodotto da un tagliaerba). Considerando la molteplicità dei suoni udibili e la complessità dei livelli di pressione del suono la lettura del sensore viene mostrata sul display dell' NXT in percentuale:

- 4-5% stanza silenziosa.
- 5-10% qualcuno che parla in lontananza.
- 10-30% normale conversazione o musica a basso volume.
- 30-100% persone che urlano o musica ad alto volume.

Figura 2.7: Sensore di suono

I sensori finora descritti sono prodotti dalla LEGO mentre l'accelerometro e il giroscopio, presentati successivamente, sono della HiTechnic.

- **Accelerometro:** Misura l'accelerazione sui tre assi x,y e z nel range [-2*g* ; +2*g*] con una suddivisione dell' intervallo tra 0 e *g* in 200 parti. Può misurare l'inclinazione lungo ciascuno dei tre assi. Il valore dell'accelerazione su ciascun asse è aggiornata circa 100 volte al secondo.I tre assi sono assegnati come in Figura 2.8.

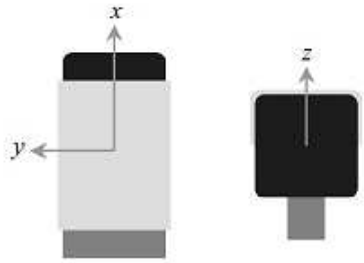


Figura 2.8: Accelerometro

Il valore mostrato sul display dell'NXT rappresenta l'accelerazione o l'inclinazione lungo l'asse x ed è compreso nel range [0 ; 254].

- **Giroscopio:** Questo sensore rileva la rotazione attorno ad un asse e restituisce il valore della velocità angolare in gradi al secondo. Il giroscopio può misurare fino a ±360° al secondo. La frequenza di rotazione può essere rilevata circa 300 volte al secondo. L'asse di misurazione è sul piano verticale, con il giroscopio posizionato come in Figura 2.9.

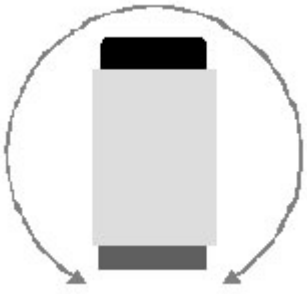


Figura 2.9: Giroscopio

2.2 SOFTWARE

Il firmware dell'NXT è Open Source e questo fa si che ogni sviluppatore possa creare a proprio piacimento un software capace di comunicare con l'hardware del prodotto. E' anche possibile creare nuove periferiche o, grazie a degli adattatori, collegare i sensori standard al mattoncino. Molti appassionati hanno sviluppato un proprio progetto su internet con l'ambizione di migliorare le capacità di questo robot.

Per quanto riguarda la gestione del programma si individuano due categorie: di una fanno parte quei software che compilano e trasferiscono il file contenente il programma nella memoria dell'NXT che poi lo esegue, e dell'altra fanno parte quei software che consentono di caricare nell'NXT l'elaborato, ma lo eseguono tramite un elaboratore esterno.

Esistono anche due tipi d'interfaccia: nel primo rientrano quei programmi dotati di un'interfaccia grafica, costituita spesso da blocchi trascinabili, che rappresentano le funzioni del robot; nel secondo quei software che richiedono la scrittura del programma, tramite linguaggi di programmazione (C, C++, etc).

Analizziamo i principali software:

LEGO NXT-G

E' il programma fornito nel pacchetto base Mindstorms LEGO NXT ed è compatibile sia con MAC-OS che con Windows (l'installazione è molto semplice sia per Vista sia per XP). Questo software, realizzato dalla collaborazione di LEGO e National Instruments, permette, per mezzo di un linguaggio grafico (Fig. 2.10-2.11), la scrittura di programmi anche piuttosto elaborati. Inoltre offre la possibilità di realizzare un file eseguibile direttamente dal mattoncino e possiede un'utility per il download. Per l'upload del programma sull'NXT è possibile utilizzare una connessione via USB o via Bluetooth. Questo linguaggio è molto intuitivo e per tale motivo è necessario poco tempo per imparare a programmare il mattoncino, tanto che LEGO propone il suo utilizzo all'interno di percorsi scolastici. L'interfaccia è molto semplice: sono presenti un numero limitato di

blocchi, divisi per categorie, collegabili tramite elementi che assomigliano molto ai mattoncini componibili con cui è costruito il robot.

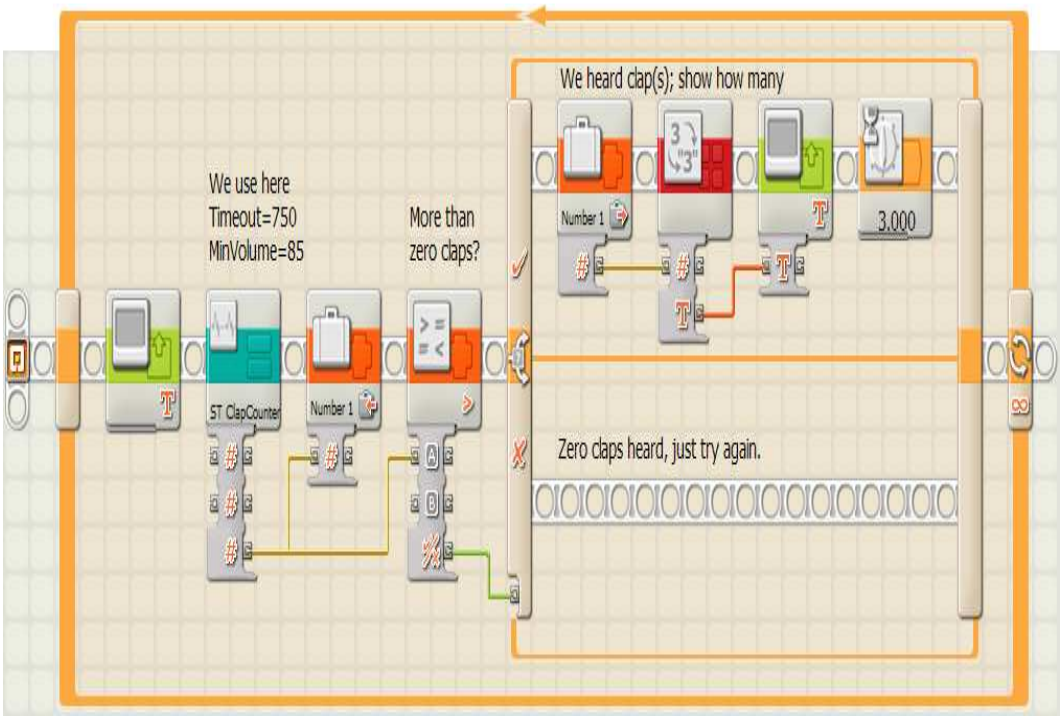


Figura 2.10: Screenshot NXT-G

Tali blocchi, di forma quadrata e di grandezza standard, riportano al loro interno immagini legate alla funzione che essi svolgono. Ogni blocco possiede una funzione diversa e selezionandolo appare un sottomenù dove è possibile impostare una serie di parametri. Inoltre sono presenti dei blocchi specifici per realizzare cicli e switch, che sono molto utili se vogliamo costruire un sistema di controllo.
La gestione di tutte le variabili non è semplice e intuitiva, ma le guide fornite velocizzano l'apprendimento.
Il fatto che il software sia realizzato appositamente per l'NXT limita problemi di connettività e compatibilità. Non è però possibile realizzare controlli troppo complessi perché lo schermo si riempie di una molteplicità di blocchi immersi in un groviglio di fili che compromettono anche la stabilità del sistema (per colpa di un interfaccia abbastanza pesante). I file prodotti con questo programma hanno dimensioni maggiori rispetto a quelli realizzati con altri software (problema da prendere in considerazione dato che la memoria dell'NXT non è di grandi dimensioni). L'esecuzione risulta assai più lenta.

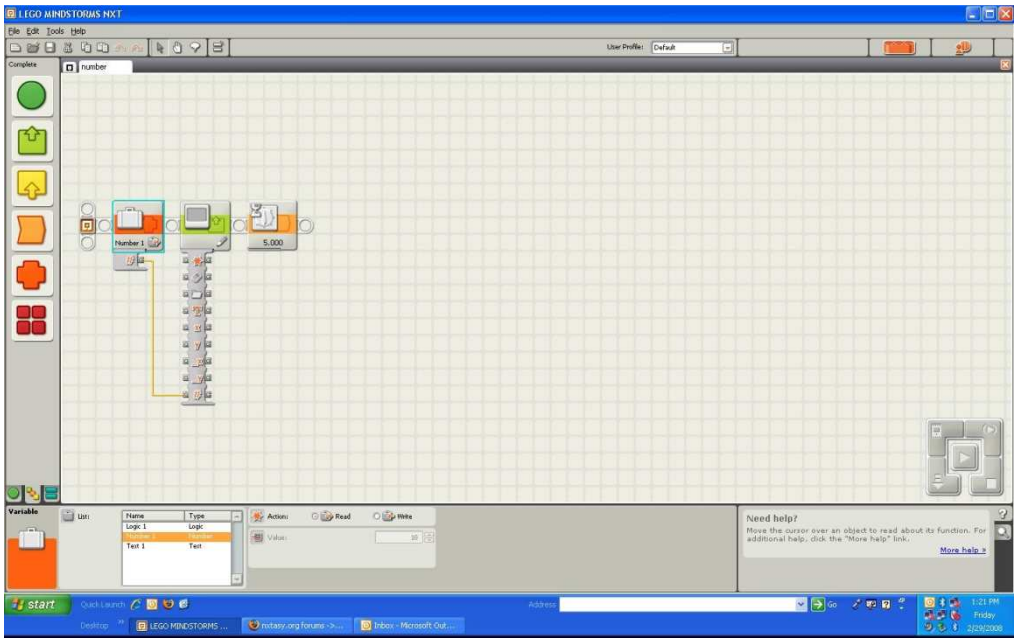


Figura 2.11: Screenshot NXT-G

ROBOLAB

ROBOLAB è la versione di Mindstorms fornita nel kit dell'NXT che permette di utilizzare le sue potenzialità in un contesto scolastico. E' un software molto potente e versatile, provvisto di un'interfaccia grafica installabile sia su Windows che su MacOS.

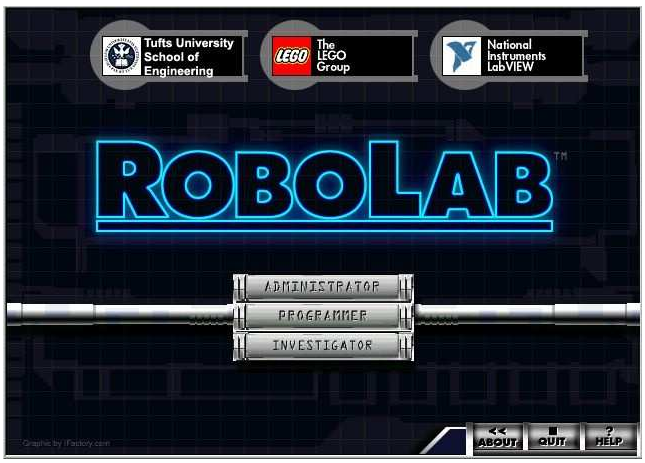


Figura 2.12: Screenshot Robolab

L'ambiente grafico di ROBOLAB (Fig. 2.12) è basato su LabVIEW della National Instruments sia per l'interfaccia sia per le possibilità offerte. Robolab utilizza un proprio firmware che deve essere caricato a bordo al posto del firmware della LEGO. Il software non è gratuito e viene venduto separatamente dall'NXT. ROBOLAB non gestisce gli ingressi in termini di eventi: la struttura del programma è quindi basata su un ciclo infinito di lettura dello stato del sensore con conseguente controllo dei motori.
Destinato all'ambiente scolastico e concepito perché quattro gruppi di lavoro possano contemporaneamente svolgere un'attività diversa, ROBOLAB è caratterizzato dal fatto di avere tre livelli di programmazione: "Pilot","Inventor" e "Investigator", ciascuno dei quali è ulteriormente suddiviso in quattro livelli di difficoltà variabile .

PILOT

Nella sezione "Pilot" di ROBOLAB il numero e l'ordine delle icone corrispondenti alle varie porte dell' NXT sono già preordinati in modo tale da ridurre il numero delle variabili da modificare. Le immagini, alcune delle quali specifiche delle periferiche collegabili all' NXT, rappresentano molto intuitivamente oggetti oppure operazioni quotidiane. La sequenza dei comandi è sempre lineare e al termine della programmazione basta un comando per trasferire le istruzioni nell' NXT e vederle eseguite dal robot.

INVENTOR

Nel contesto operativo che si apre nella sezione di ROBOLAB denominata "Inventor" (Fig. 2.13) i comandi non sono preordinati. Sono disponibili tutte le funzioni e si possono sfruttare a pieno tutte le potenzialità dell' NXT. La facoltà di operare liberamente è un notevole progresso rispetto alle operazioni previste dal livello Pilot, sia perché la bontà del risultato non è più garantita, sia perché si ha la possibilità di disporre di funzioni in grado di creare variabili, iterazioni e subroutines con cui sfruttare il multitasking dell' NXT.

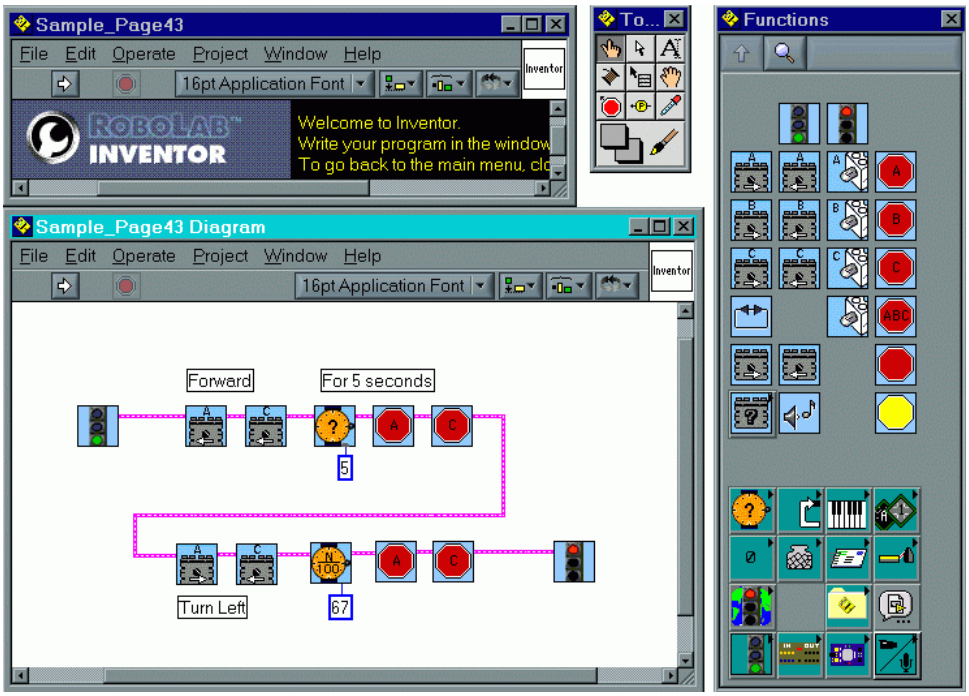


Figura 2.13: Screenshot Robolab Inventor

INVESTIGATOR

Serve per raccogliere,elaborare e visualizzare i dati letti dai sensori.

I sensori ottici, tattili o di temperatura possono essere collegati all'NXT perché esso conservi regolarmente memoria delle variazioni rilevate.

Si possono anche utilizzare sensori di altro tipo, per misurare le variazioni del pH, dell'umidità o della pressione. I dati in tal modo raccolti dall'NXT possono essere prelevati dal mattoncino LEGO e portati nel computer, dove potranno essere elaborati.

NI LabVIEW Toolkit

NI LabVIEW (Fig. 2.14) è un software sviluppato dalla National Instruments come estensione di LabVIEW. Si possono notare molte somiglianze tra LEGO Mindstorms NXT e LabVIEW poiché NI e LEGO hanno collaborato sia per conservare gli elementi principali della programmazione grafica di LabVIEW sia per ottimizzare l'interfaccia utente per i principianti.

NI LabVIEW è drag-and-drop (clicca e trascina) ossia i blocchi , situati nella parte sinistra dello schermo,sono trascinabili nel diagramma. Ogni blocco svolge un'unica funzione, come ad esempio, muovere i motori, mostrare un messaggio

sul display, rilevare un suono o misurare una distanza. E' possibile creare nuovi blocchi di comando compatibili anche con l'NXT-G e questo risulta essere molto utile quando si deve passare da una programmazione di base ad una più complessa.

Combinando una serie di blocchi è possibile programmare il robot in modo che esegua ciò che l'utente desidera.

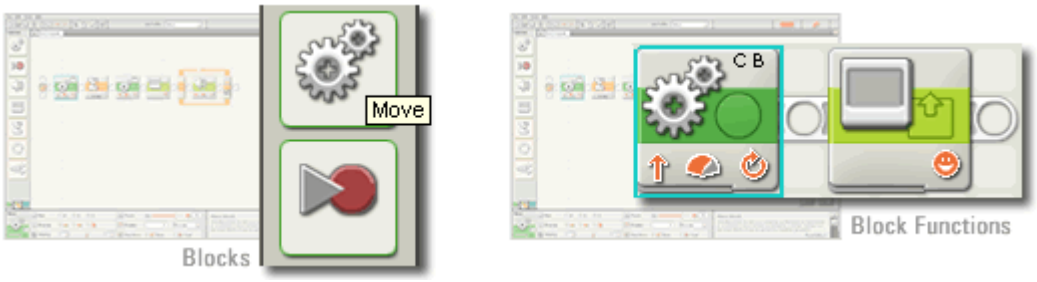


Figura 2.14: Screenshot NI LabVIEW

Dopo aver scritto il programma è necessario scaricarlo sull'NXT utilizzando un collegamento Bluetooth o un cavo USB (Fig. 2.15). Si possono inserire più programmi nel brick e poi selezionare quello desiderato.

Essendo un'estensione necessita del software LabVIEW per funzionare e di una discreta conoscenza dello stesso, il che lo rende molto più complesso dell'NXT-G.

Figura 2.15: Screenshot NI LabVIEW e NXT collegato al PC

Microsoft Robotics Studio

Microsoft Robotics Studio è un software che facilita la creazione di applicazioni nel campo della robotica per numerose piattaforme hardware. Il kit include una semplice, ma potente, piattaforma runtime, facile da utilizzare.

Figura 2.16: Esempio applicativo di Microsoft Robotics Studio

L'interfaccia di lavoro è grafica e si basa sul drag-and-drop. Infatti si possono trascinare i blocchi, che rappresentano le varie funzioni, e connetterli insieme. La caratteristica principale è la possibilità di simulare virtualmente, in 3D (Fig. 2.16), il mattoncino e di testare il programma scritto sulla macchina. E' un software che permette sia il controllo remoto dal PC sia la modalità autonoma. La comunicazione tra PC e robot è realizzata tramite Bluetooth, RF o utilizzando una connessione USB.

NXT OnBrick

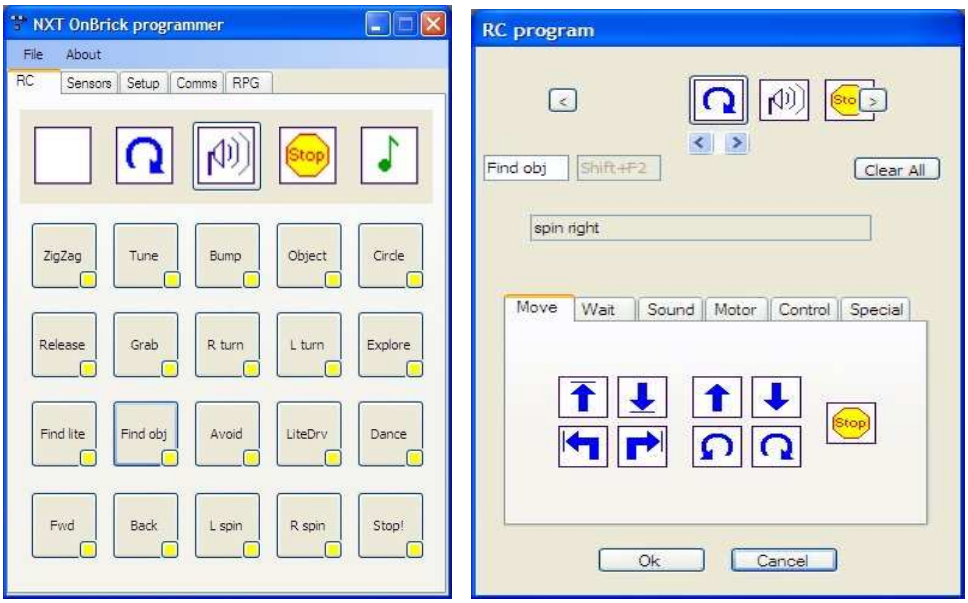


Figura 2.17: Screenshot NXT OnBrick

NXT OnBrick permette il controllo remoto del brick, utilizzando il Bluetooth, sia per PC che per palmari. L'interfaccia grafica è costituita da 20 pulsanti (Fig. 2.17) e questo rende estremamente facile l'esecuzione di alcuni tipi di controllo. L'inserimento dei blocchi grafici avviene in spazi preimpostati a scapito della libertà di programmazione. Sono previste 36 azioni, la maggior parte delle quali con un singolo parametro, come ad esempio il controllo individuale dei motori, l'attesa del verificarsi di eventi rilevati dai sensori e l'emissione di semplici suoni.

RobotC

RobotC è una valida alternativa al software fornito dalla Lego. Sviluppato dalla Carnegie Mellon Robotics, come si intuisce dal nome, si basa sul linguaggio di programmazione C. Può essere utilizzato sia per l'NXT che per l'RCX. Rispetto all' NXT-G può sembrare più complesso in quanto non dispone di un interfaccia grafica (Fig. 2.18) ma le potenzialità sono nettamente superiori; ad esempio permette di inserire i commenti.

Per funzionare necessita di un firmware diverso da quello originale. Quest'ultimo garantisce molti vantaggi tra cui una maggiore velocità e la possibilità di compilare e scaricare direttamente il programma sull' NXT.

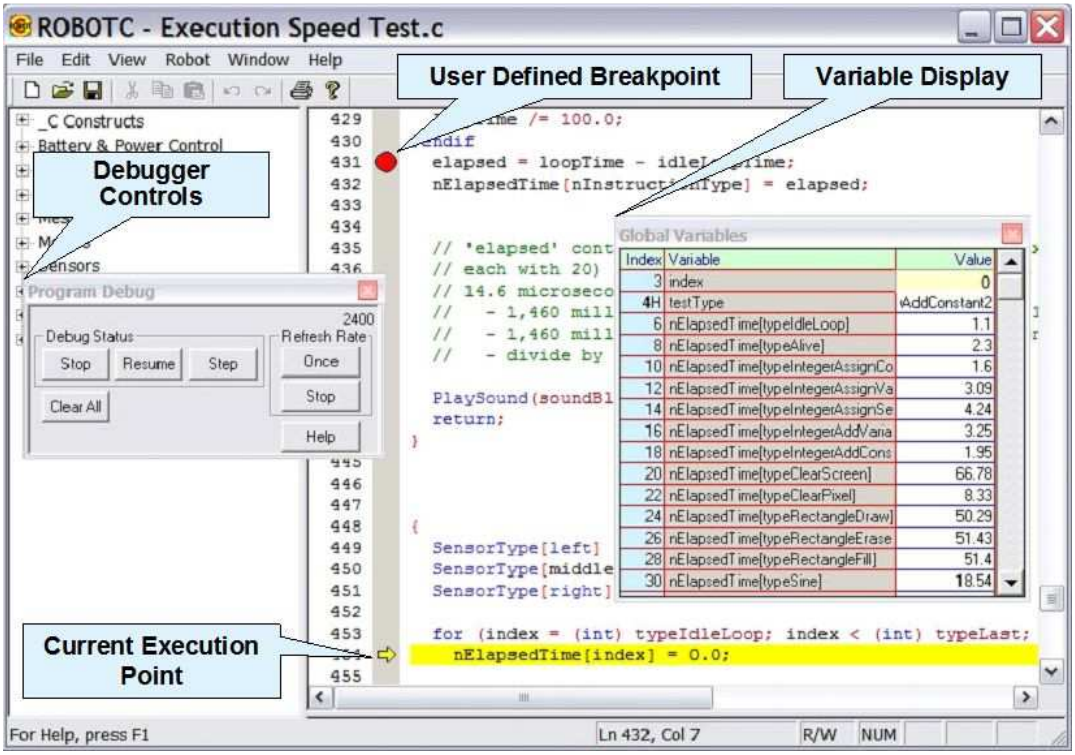


Figura 2.18: Screenshot RobotC

Un ulteriore vantaggio sta nel poter monitorare in tempo reale alcune variabili (valore dell'encoder, valore dei sensori, valori delle variabili di programmazione). RobotC permette anche il controllo da remoto del mattoncino.

NeXT Explorer

Questo software non serve per la programmazione, ma per la gestione della memoria del brick. NeXT Explorer consente infatti il download, l'upload e la cancellazione dei file presenti sull'NXT (molto utile poiché quest'ultimo non è dotato di una memoria molto estesa).

RXEdumper

Questo è un software per la comunicazione e non per la programmazione. E' utilizzato per scrivere programmi per l'NXT (Fig. 2.19) con l'NBC. RXEdumper permette sia la compilazione dei file .nbc in file .rxe sia la decompilazione. Il mattoncino può eseguire solo file .rxe e quindi può essere necessario convertire i file .nbc con RXEdumper, se il programma che utilizza il linguaggio NBC non è in grado di farlo autonomamente. Il file deve essere poi scaricato sul brick tramite altri software, per esempio attraverso NeXT Explorer.

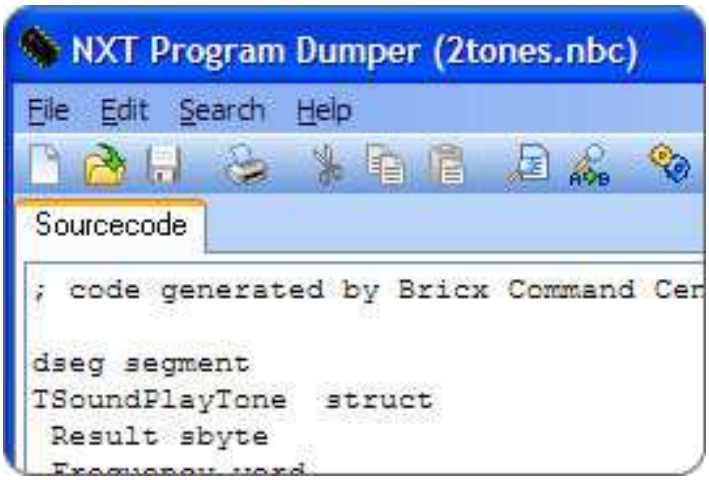


Figura 2.19: Screenshot RXE Dumper

Bricx Command Center

Il BricxCC permette sia la scrittura di programmi attraverso diversi tipi di linguaggi (per esempio NBC ed NXC) sia la compilazione dei file e l'invio di questi all' NXT. Utilizza il firmware fornito dalla LEGO e consente il controllo remoto di alcune funzioni del mattoncino. Inoltre sfrutta il driver LEGO NXT per comunicare attraverso cavo USB o Bluetooth. BricxCC può convertire file MIDI nel formato .rmd dell'NXT. Mediante l'IDE è possibile scaricare programmi sull'NXT, eseguirli e interromperli. Può essere anche utilizzato per decompilare file eseguibili per l'NXT (.rxe, .sys, .rtm) sia tramite una finestra di dialogo, sia trascinandoli sulla finestra del BricxCC (Fig. 2.20). Tuttavia il suo utilizzo non è banale e la possibilità di scrivere con diversi linguaggi è fruibile solo se in possesso di tutte le librerie necessarie. Le potenzialità sono comunque discrete ed essendo un Open Source risulta in continua evoluzione.

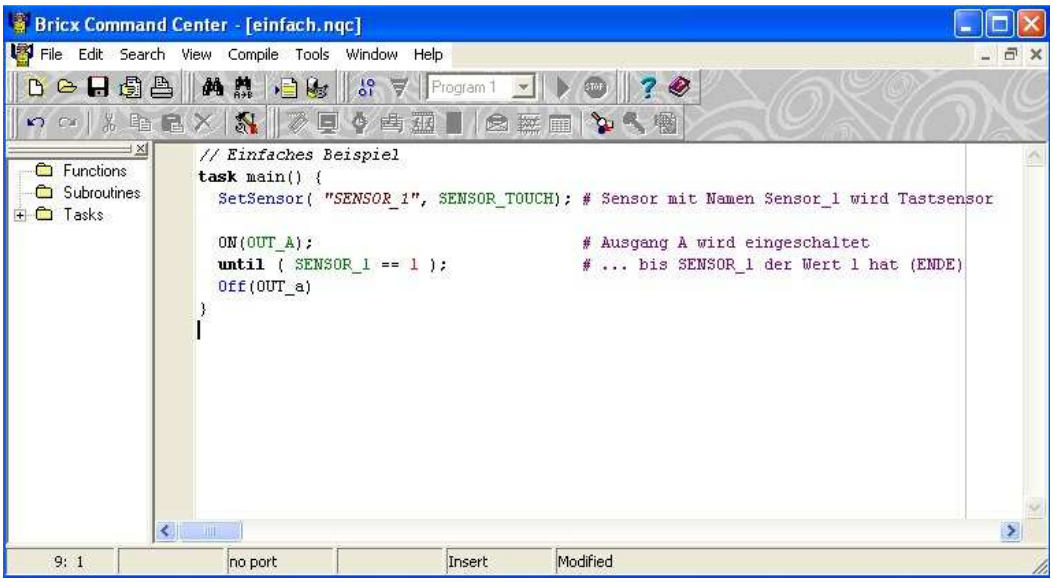


Figura 2.20: Screenshot BricxCC

Next Byte Codes (NBC) è un semplice linguaggio basato sulla sintassi assembly che può essere utilizzato per programmare l'NXT. Serve sia per compilare sia per decompilare file eseguibili sull'NXT. L'NBC è disponibile per Win32, Mac OSX e per Linux. L'NBC si divide in due parti: il linguaggio NBC che descrive la sintassi da utilizzare nella scrittura del programma e l'NBC API che descrive le

funzioni,le costanti e le macro del sistema. L'API è definita in un file speciale conosciuto come "header file" che è incluso all'inizio di ogni programma NBC ma non è incluso automaticamente quando viene compilato un programma.

Not eXactly C (NXC) è un linguaggio di alto livello, simile al C, basato sul compilatore NBC. Utilizza lo stesso firmware dell' NXT-G e questo è molto comodo per gli utenti che vogliono un ambiente sia grafico che testuale (Fig. 2.21) perché non è necessario ricaricare e cambiare il firmware ogni volta che si passa dall'NXT-G all'NXC e viceversa. Inoltre è possibile salvare simultaneamente programmi NXT-G e NXC. L'IDE per NXC è il BricxCC. Una limitazione è rappresentata dal fatto che si possono avere variabili intere, ma non in virgola mobile.

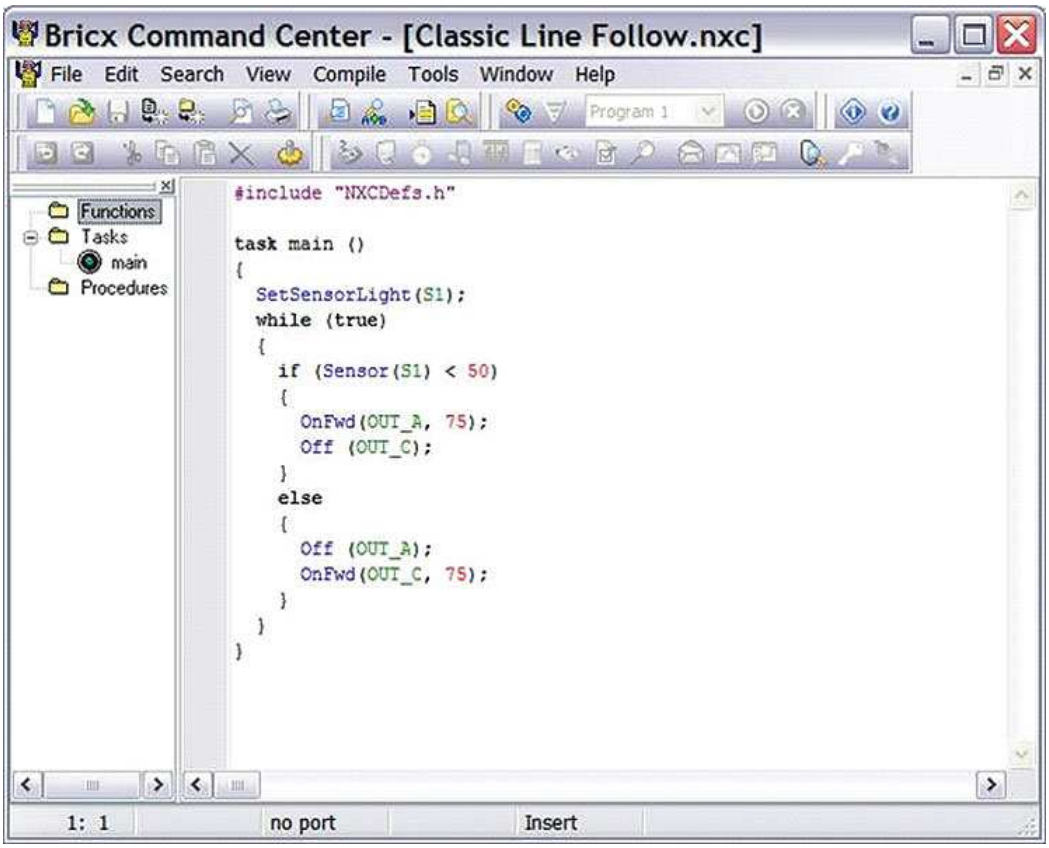


Figura 2.21: Screenshot BricxCC

NXTRC

E' un programma che viene avviato dalla linea di comando ed è utilizzato per la gestione della memoria dell'NXT. Con pochi comandi si riescono ad eseguire tutte le operazioni di base quali download, upload, cancellazione e visualizzazione dei file presenti sull'NXT. Il trasferimento risulta mediamente più veloce rispetto agli altri programmi. Funziona solo su Linux, unicamente con la comunicazione Bluetooth.

pbLUA

Questo linguaggio è utilizzato in molte applicazioni, ma le più importanti riguardano proprio l'NXT LEGO Mindstorms. Ha le seguenti caratteristiche:

- E' scritto in C (Fig. 2.22)
- Può essere compilato sull'NXT
- E' facile da leggere e da scrivere
- Su internet è presente una vasta documentazione ed esiste un newsgroup sull'argomento

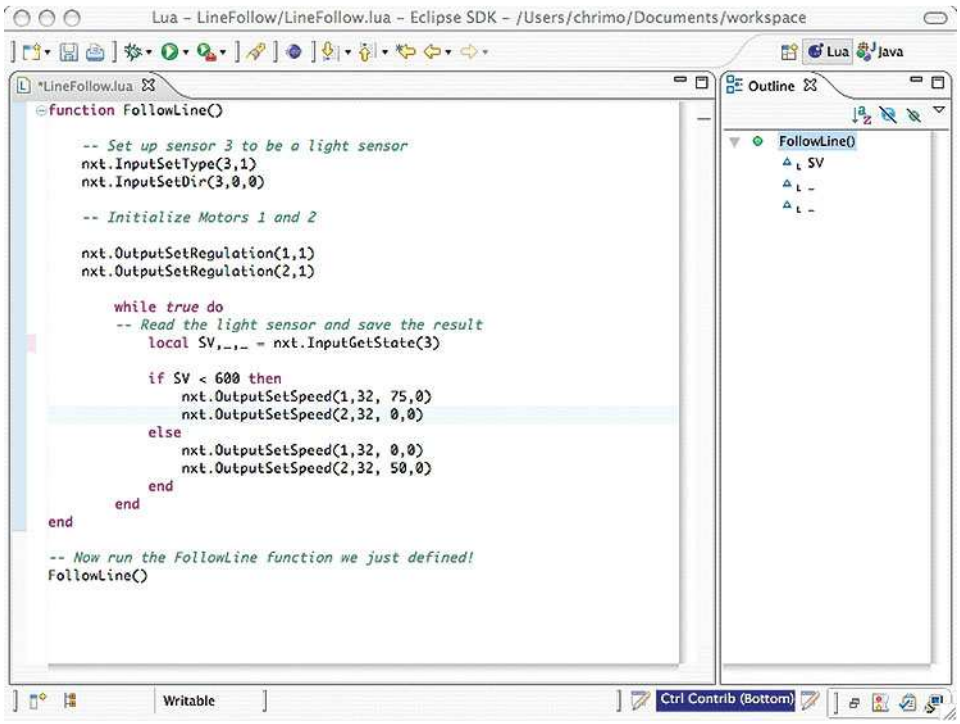


Figura 2.22: Screenshot pbLUA

2.3 SOFTWARE MATHWORKS

In questa sezione verranno illustrate le caratteristiche principali di MATLAB, di Simulink e dei toolbox utilizzati. La nostra scelta è ricaduta sui software MathWorks perché offrono la possibilità di controllare in modo efficiente numerosi sistemi e perché sono largamente impiegati nelle industrie.

2.3.1 MATLAB

MATLAB è un linguaggio di alto livello e un ambiente interattivo per lo sviluppo di algoritmi, per il calcolo numerico, per la visualizzazione e l'analisi di dati. I problemi tecnici, utilizzando MATLAB, possono essere risolti più velocemente che con i tradizionali linguaggi di programmazione, come il C, il C++ e Fortran. Infatti non è necessario gestire i task di basso livello, come ad esempio dichiarare le variabili, specificare il tipo di dati e allocare la memoria. MATLAB è una piattaforma molto efficiente per l'acquisizione di dati da altri file, da altre applicazioni, da banche dati e da dispositivi esterni.
Molto importante per questa tesi è stata la possibilità di importare dati da Microsoft Excel per ottenere una rappresentazione grafica degli esperimenti svolti. Con MATLAB si può compiere l'intero processo di analisi dei dati, dall'acquisizione, alla visualizzazione e l'analisi numerica, fino a produrre una rappresentazione grafica dell'uscita. In MATLAB sono disponibili tutte le funzionalità necessarie per visualizzare i dati prodotti dagli esperimenti con grafici 2-D e 3-D (Fig. 2.23). Questi ultimi possono essere salvati in molti formati, tra cui GIF, JPEG, BMP, EPS.
Sono acquistabili separatamente dei toolbox che estendono l'ambiente MATLAB, permettendo la risoluzione di particolari classi di problemi. Di seguito riportiamo un elenco dei toolbox che abbiamo utilizzato per il controllo del Legway.

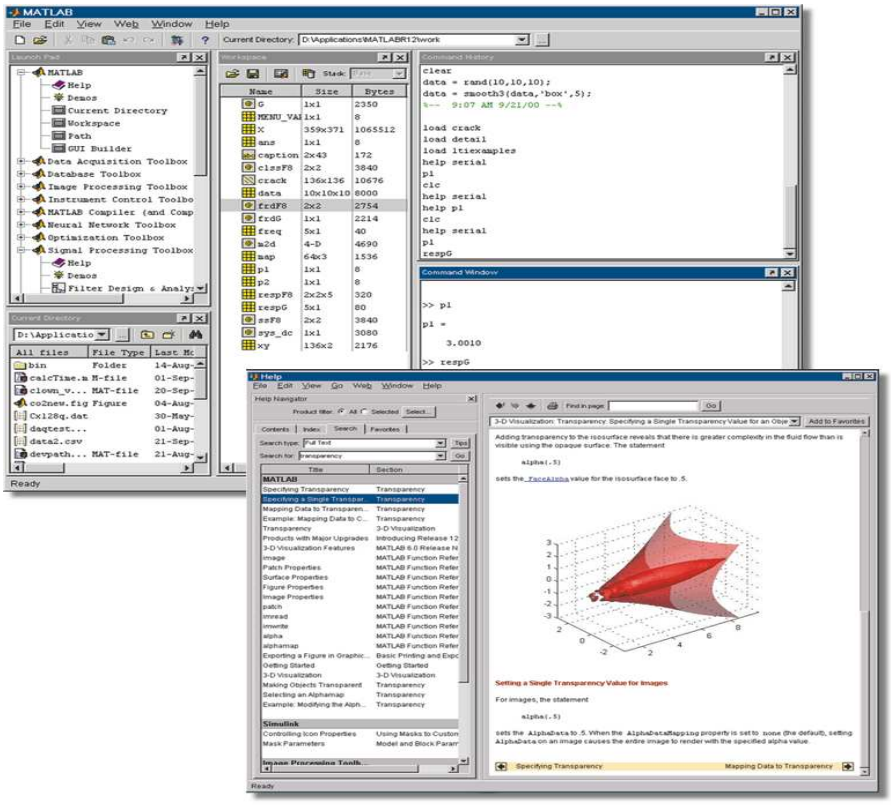


Figura 2.23: Screenshot MATLAB

2.3.2 SIMULINK 7.1

Simulink è un ambiente per simulazioni multi-dominio e un Model-Based Design usato per sistemi dinamici ed embedded. E' un ambiente grafico interattivo e fornisce un set personalizzabile di librerie di blocchi che permettono di progettare, simulare, implementare e testare una grande varietà di sistemi tempo-varianti, di controllarli, di elaborare segnali, video e immagini (Fig. 2.24). Esistono molti add-on che estendono il software di Simulink ad altri domini e che forniscono tools per il progetto, per l'implementazione, per la verifica e la convalida delle specifiche. Simulink è integrato da MATLAB e ciò garantisce l'accesso immediato ad un ampio numero di tools. Questi permettono: lo sviluppo di algoritmi, l'analisi e la visualizzazione delle simulazioni, la creazione di gruppi di elaborazione degli script, la personalizzazione dell'ambiente del modello e la definizione di segnali, parametri e informazioni di prova.

CARATTERISTICHE PRINCIPALI:

- Librerie espandibili di blocchi predefiniti
- Editor con grafica interattiva per assemblare e gestire i diagrammi a blocchi
- Possibilità di gestire progetti complessi suddividendo i modelli gerarchicamente
- Model Explorer per creare e configurare segnali e parametri
- Application programming interfaces (APIs) che permette all'utente di connettersi con altri programmi di simulazione e utilizzare codici embedded
- I blocchi dell' Embedded MATLAB Function per trasferire gli algoritmi di MATLAB in Simulink e nelle implementazioni del sistema embedded
- Modalità per la simulazione (Normal, Accelerator, Rapid Accellerator)
- Debugger grafico e un profiler per esaminare i risultati della simulazione e diagnosticare i comportamenti inaspettati del progetto
- Pieno accesso a MATLAB per analizzare e visualizzare i risultati, per personalizzare l'ambiente del modello e per definire il segnale, i parametri e le informazioni di prova
- Analisi del modello e tools per la diagnosi, per la verifica della coerenza del modello e per l'identificazione di eventuali errori

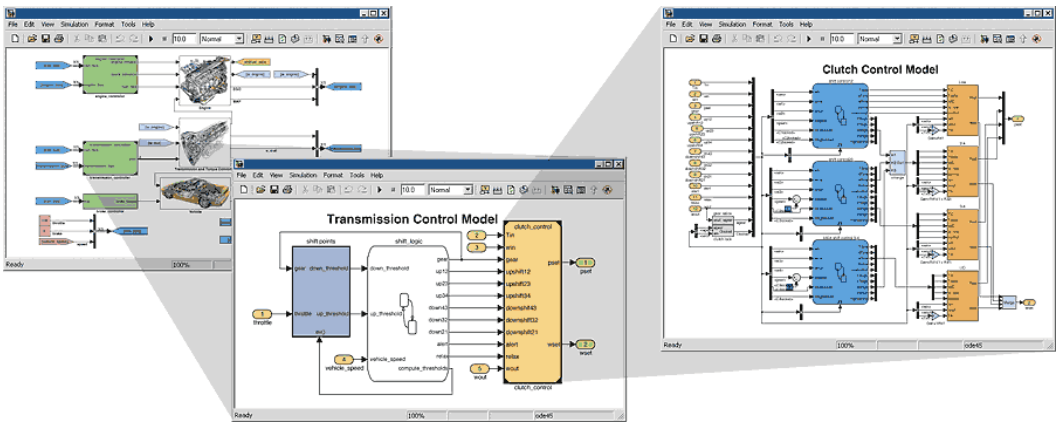


Figura 2.24: Esempi applicativi di progetti realizzati con Simulink

CREARE E LAVORARE CON I MODELLI:

Con Simulink, si può velocemente creare, modellare e mantenere un dettagliato diagramma a blocchi del sistema, usando un set di blocchi predefinito. Simulink fornisce dei tools per modellare gerarchicamente, per gestire dei dati e per personalizzare dei sottosistemi. In tal modo la rappresentazione del sistema è più facile e più veloce, malgrado la sua reale complessità.

SELEZIONARE E PERSONALIZZARE I BLOCCHI:

Simulink include una libreria di funzioni (Fig. 2.25), comunemente impiegate per modellare i sistemi, che contiene:

- Blocchi a dinamica continua e discreta come integratori e derivatori
- Blocchi algoritmici come la Sum, Product, e Lookup Table
- Blocchi strutturali come Mux, Switch, e Bus Selector

Si possono personalizzare questi blocchi predefiniti o crearne di nuovi direttamente da Simulink e inserirli in una libreria propria dell'utente. Set di blocchi addizionali estendono Simulink con funzioni specifiche per il campo aerospaziale, delle comunicazioni, delle radiofrequenze, dell'elaborazione di segnali, dell'elaborazione video, dell'elaborazione di immagini e in altre applicazioni.

INCORPORARE ALGORITMI MATLAB E CODICE:

Dopo aver inserito il codice MATLAB, l'utente può chiamarne le funzioni per analizzare e per visualizzare i dati. Simulink permette di utilizzare il codice Embedded MATLAB per progettare algoritmi embedded, che possono essere utilizzati, insieme al resto del modello, per la generazione del codice. Si può anche inserire codice C, Fortran, ed Ada direttamente nel modello, abilitando l'utente a creare blocchi personalizzati.

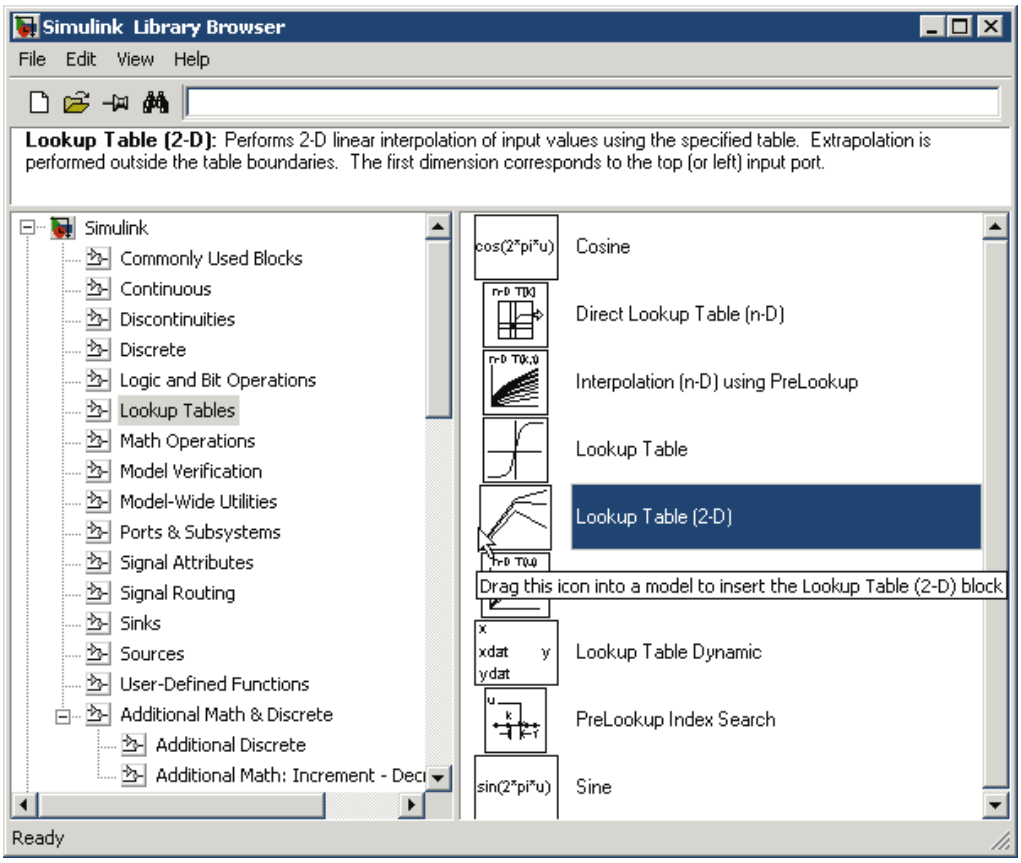


Figura 2.25: Screenshot della libreria di Simulink

SCRIVERE E MODIFICARE IL MODELLO:

Con Simulink, si possono costruire modelli utilizzando la tecnica del drag-and-drop, cioè trascinando i blocchi dalla libreria del browser nell'editor grafico e connettendoli attraverso linee che stabiliscono relazioni matematiche (Fig. 2.26). Il modello può essere organizzato usando i comandi copia, incolla, annulla, allinea, distribuisci e ridimensiona. Le interfacce utente di Simulink danno il completo controllo su ciò che appare sullo schermo. L'utente può aggiungere comandi personalizzati e sottomenù all'editor o al menù principale. Inoltre si possono disabilitare e nascondere i menù o gli oggetti della finestra di dialogo.

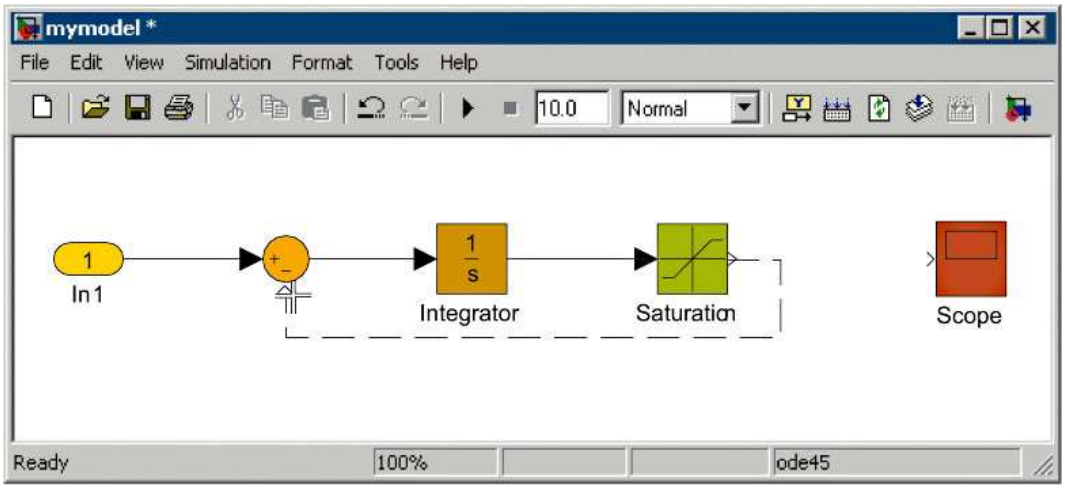


Figura 2.26: Connessione tra blocchi di Simulik

ORGANIZZARE IL PROPRIO MODELLO:

Simulink permette di organizzare il modello in livelli gerarchici utilizzando sottosistemi e referenziando i modelli. I sottosistemi incapsulano un gruppo di blocchi e di segnali in un singolo blocco. L'utente può aggiungere un'interfaccia personalizzata al sottosistema, nascondendone il contenuto del sottosistema e lo facendolo apparire come un blocco atomico con un propria icona e una finestra di dialogo dei parametri.

L' utente può suddividere il modello in componenti e modellarle, simularle e verificarle indipendentemente. Queste componenti possono essere salvate sia come modelli separati sia come sottosistemi in una libreria. I progetti di ogni componente possono essere riutilizzati facilmente mantenendo un archivio delle verifiche e delle revisioni. Organizzando i modelli in questo modo si può selezionare il livello di dettaglio appropriato per la configurazione del lavoro.

SOTTOSISTEMI CONFIGURABILI:

I sottosistemi configurabili permettono di associare varianti del progetto a sottosistemi all'interno del modello. Questa capacità semplifica la creazione e la gestione dei progetti che condividono componenti.

SOTTOSISTEMI AD ESECUZIONE CONDIZIONATA:

I sottosistemi ad esecuzione condizionata dall'ingresso permettono di cambiare un sistema dinamico abilitando o disabilitando sezioni specifiche del progetto attraverso il controllo di segnali logici. Simulink permette di creare segnali che possono abilitare o dare l'avvio all'esecuzione di sottosistemi basati su eventi specifici. I blocchi logici permettono di generare semplici comandi per il controllo del sottosistema. Si possono includere anche controlli logici più complessi (per esempio le macchine a stati modellabili attraverso Stateflow).

DEFINIZIONE E GESTIONE DEI SEGNALI E DEI PARAMETRI:

Simulink abilita l'utente a definire e controllare le caratteristiche dei segnali e dei parametri associati al modello. I segnali sono quantità tempo-varianti rappresentate dalle linee di connessione dei blocchi. I parametri sono coefficienti che aiutano a definire la dinamica e il comportamento del sistema.

I segnali e i parametri possono essere definiti direttamente nel diagramma o in un dizionario separato. Usando Model Explorer (Fig. 2.27), l'utente può gestire il dizionario e ridefinire velocemente il modello cambiando alcuni dati.

Parametri e segnali che possono essere definiti:

- Tipo di dati: single, double, interi con segno e senza segno a 8, 16 or 32 bit; boolean; a virgola fissa
- Dimensioni: scalare, vettore, matrice, or N-D arrays
- Complessità: valori reali o complessi
- Range minimo e massimo, valore iniziale e unità ingegneristiche

I dati in virgola fissa forniscono la possibilità di dimensionare la lunghezza delle parole fino a 128 bit. Questo tipo di dato richiede Simulink Fixed Point software per simulare e generare il codice. L'utente può anche decidere la modalità di campionamento del segnale (sample-based o frame-base) per abilitare un'esecuzione più veloce.

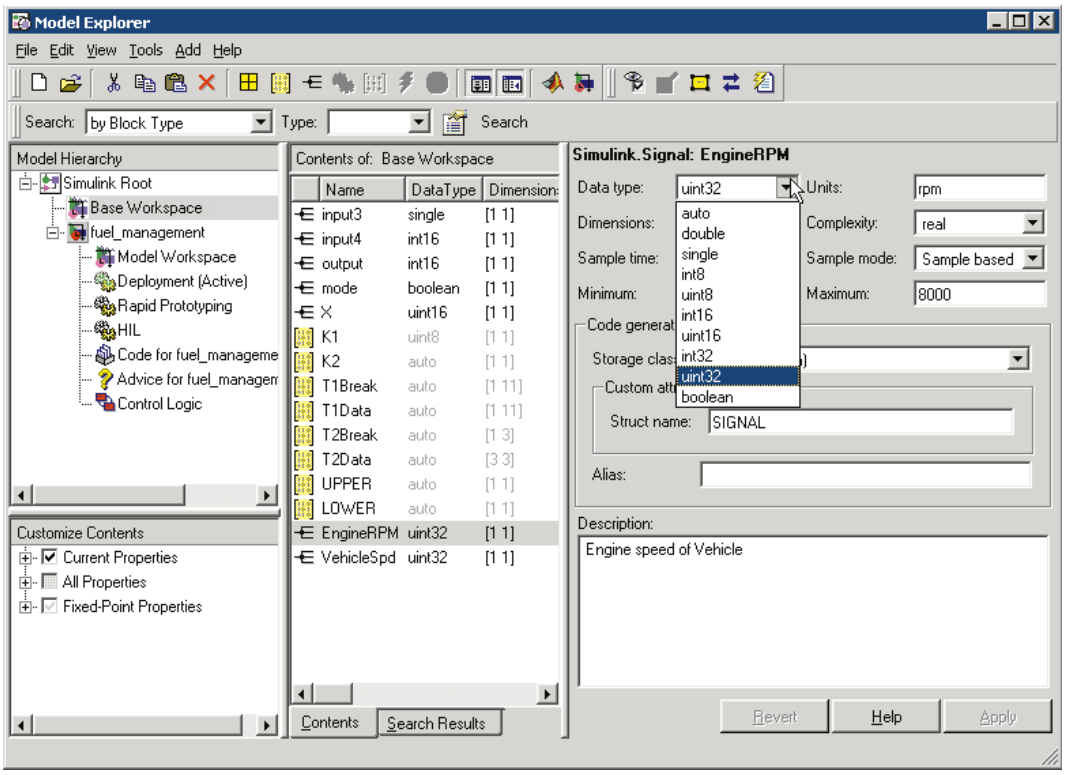


Figura 2.27: Screenshot di Model Explorer

Usando i data-type objects , l' utente può definire data type e segnali bus personalizzati. I segnali bus permettono di definire interfacce trai componenti del progetto.

Simulink permette di definire le specifiche del segnale. Se l'utente non fissa gli attributi , Simulink li determina automaticamente. L'utente può specificare solo le interfacce dei componenti oppure tutti i dati per il modello.

Si può anche restringere il range dei parametri per specificare parte del modello attraverso una gerarchia all'interno degli workspaces , o condividerli attraverso un workspace globale.

ESEGUIRE UNA SIMULAZIONE:

Dopo aver costruito il modello in Simulink, l'utente può simulare la dinamica del comportamento e vedere i risultati in tempo reale. Simulink fornisce molte caratteristiche e tools per assicurare la velocità e l'accuratezza della simulazione.

Infatti include risolutori a passo fisso o variabile, un debugger grafico e un profiler del modello.

UTILIZZARE RISOLUTORI:

I Risolutori sono algoritmi di integrazione numerica che calcolano la dinamica del sistema in funzione del tempo, utilizzando informazioni contenute nel modello. Simulink fornisce risolutori per permettere la simulazione di una grande quantità di sistemi: tempo continuo, tempo discreto, ibridi e sistemi multi-rate di ogni misura.

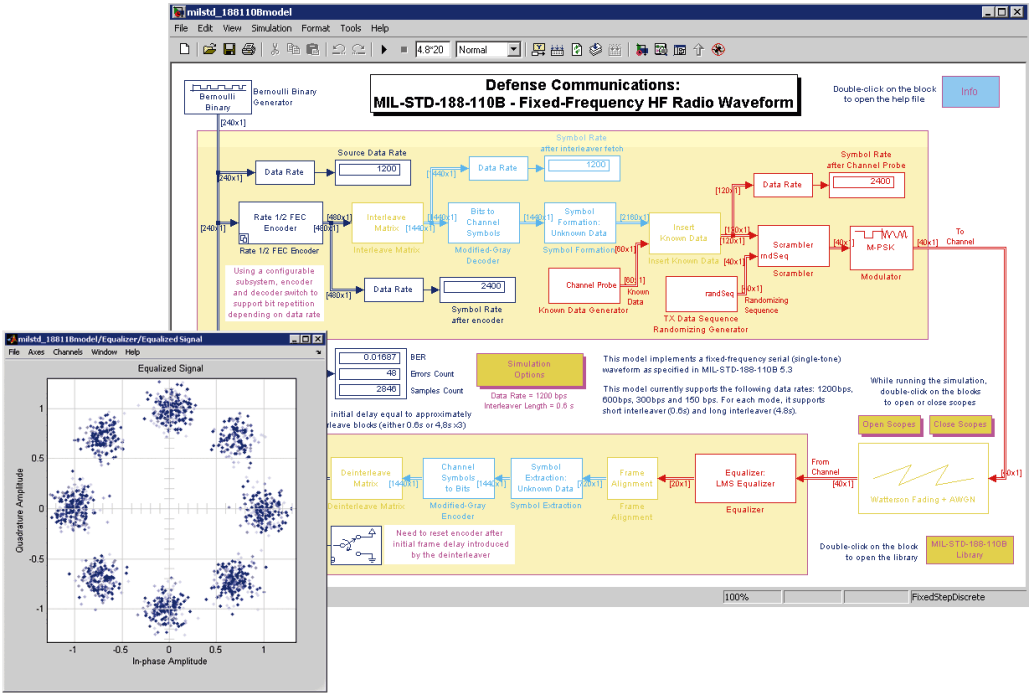


Figura 2.28: Screenshot di un risolutore

Questi risolutori possono simulare stiff systems e state events systems, considerandoli come discontinuità. L'utente può specificare le opzioni di simulazione: tipo e proprietà del risolutore, tempo di inizio e di interruzione della simulazione (Fig. 2.28). Si possono anche settare le informazioni di ottimizzazione e di diagnostica per la simulazione. Differenti combinazioni delle opzioni possono essere salvate con il modello.

DEBUG DI SIMULAZIONE:

Il debugger di Simulink (Fig. 2.29) è un tool interattivo che esamina i risultati della simulazione e controlla i comportamenti inaspettati nel modello Simulink. Questo permette di localizzare velocemente e con precisione i problemi presenti all'interno del modello, simulando un metodo alla volta e analizzando i risultati del metodo in esecuzione.

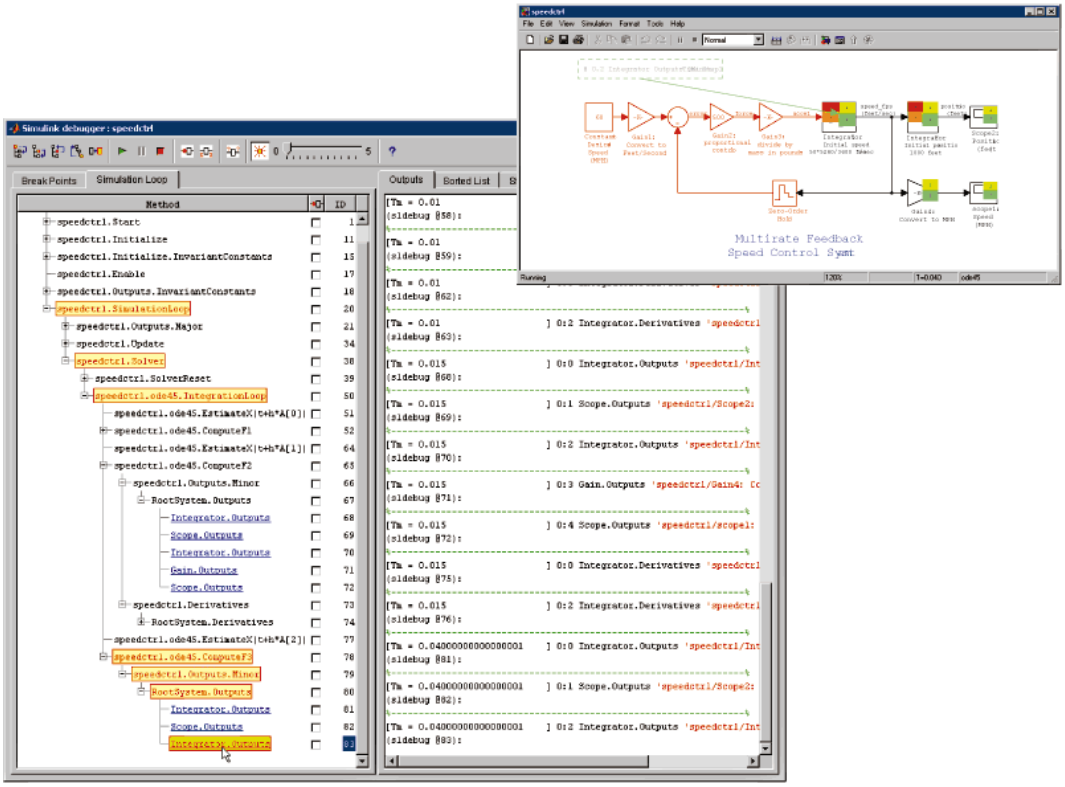


Figura 2.29: Screenshot del debugger GUI

Il debugger permette di impostare i punti di rottura, di controllare l'esecuzione della simulazione, e di mostrare le informazioni del modello. Può essere lanciato da un interfaccia grafica (GUI) o dalla linea di comando di MATLAB. GUI fornisce una semplice visione dello stato dell'esecuzione (attraverso una colorazione del codice). L'utente può anche visualizzare informazioni sullo stato,input e output di un blocco ed altri dati.

ESECUZIONE DI UNA SIMULAZIONE:

Una volta impostate le opzioni della simulazione per il modello, si può eseguire la

simulazione sia dalla linea di comando di MATLAB sia interattivamente utilizzando la Simulink GUI .

Modalità di esecuzione:

- Normal (default): interpreta il modello e lo simula.
- Accelerator: velocizza l'esecuzione del modello compilando il codice quando l'utente ha ancora la possibilità di cambiare i parametri del modello.
- Rapid Accelerator: Simula il modello ancora più velocemente, ma con meno interattività, creando un eseguibile separato da Simulink.

L'utente può anche usare i comandi di MATLAB per caricare e trattare i dati del modello e visualizzare i risultati.

PROFILING E SIMULATION:

Tracciare un profilo della simulazione può essere utile per identificare i punti critici. L'utente può raccogliere i dati della performance mentre la simulazione è in atto e poi generare un resoconto del profilo della simulazione basato sui dati collezionati, che mostrano quanto tempo Simulink necessiti per eseguire ogni simulazione di un metodo.

ANALSI DE I RISULTATI:

Simulink include molti tools per analizzare un sistema, visualizzare i risultati, fare prove e convalidare il modello sotto osservazione.

VISUALIZZAZIONE DEI RISULTATI:

L'utente può visualizzare il sistema vedendo i segnali con i display e gli scope forniti nel software. In alternativa si può costruire un proprio display personalizzato, utilizzando i tools di visualizzazione di MATLAB e di GUI. Si possono anche salvare i dati sui segnali per utilizzi futuri.

Se si vuole capire a fondo come funziona la dinamica del modello si possono vedere scene 3-D di realtà virtuale utilizzando Virtual Reality Toolbox.

TEST E CONVALIDA DEL MODELLO:

Queste operazioni possono essere effettuate tramite dei tools forniti da Simulink. Per esempio il Signal Block Builder permette di creare graficamente forme d'onda da mandare in ingresso al modello per vedere come questo risponde. Utilizzando il Signal & Scope Manager si possono immettere segnali nel modello, salvare e visualizzare i risultati, senza introdurre blocchi aggiuntivi. Simulink fornisce anche blocchi per controllare che gli output di quest'ultimi siano conformi alle richieste del progetto.

L'utente può collegare formalmente le esigenze alle sezioni del modello, può progettare controlli standard del modello personalizzato (Fig. 2.30) ed eseguire un reportage del modello utilizzando il software Simulink Verification and Validation. L' utente può anche generare dei test per verificare che il modello soddisfi gli obiettivi da lui fissati utilizzando il software Simulink Design Verifier. Per gestire e memorizzare le prove indipendentemente dal modello si può usare il software SystemTest.

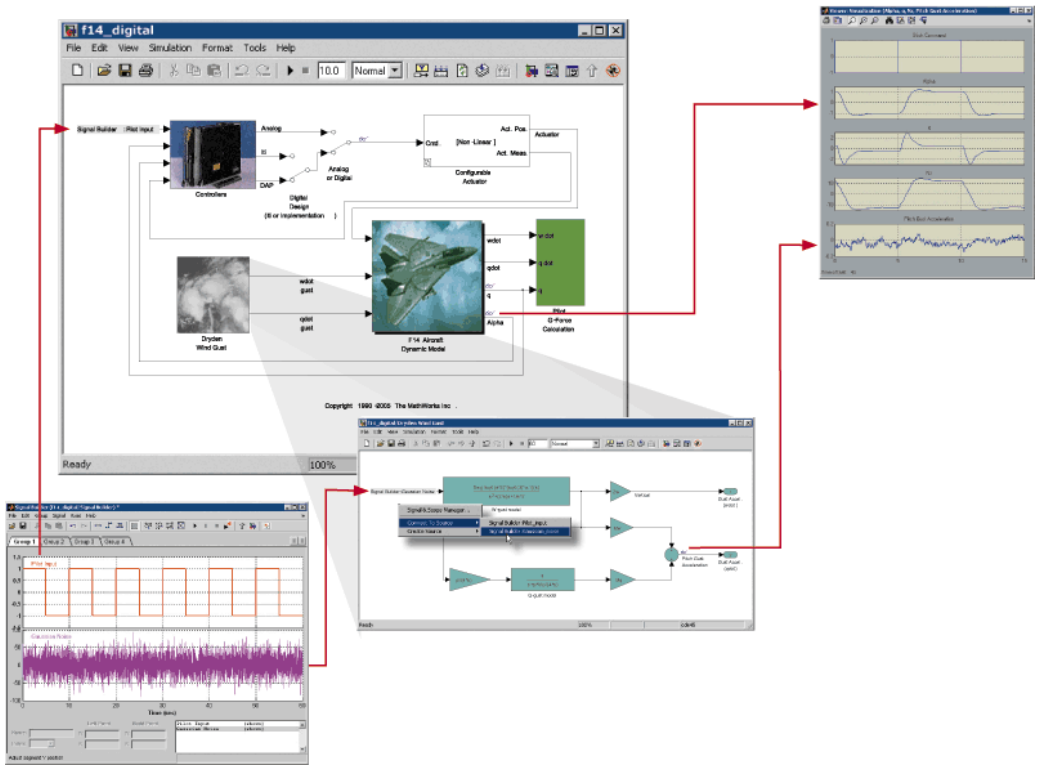


Figura 2.30: Esempio applicativo di un sistema di controllo modellato in Simulink

DOCUMENTARE IL MODELLO:

E' facile aggiungere documentazioni al proprio modello Simulink. Si possono aggiungere direttamente al diagramma annotazioni e link ipertestuali ad altre pagine Web. Descrizioni dettagliate possono essere incluse direttamente tra le proprietà di ogni blocco o del modello. Il DocBlock permette di includere un file testo come se fosse un blocco all'interno del modello. Attraverso un comando si può creare un documento HTML che descrive l' intero modello.

Usando Simulink Manifest Tools si può ottenere una lista dei file che il modello necessita per funzionare e, in aggiunta, comprimere questo file per condividerlo con altri utenti. Usando Simulink Report Generator si possono creare rapporti personalizzati conformi ai documenti standard.

2.3.3 CONTROL DESIGN

E' una tecnica per lo sviluppo software che utilizza modelli che possono essere simulati. I progettisti dell'embedded control system software hanno dovuto affrontare sfide molto impegnative. Oltre a dover sviluppare il progetto a costi ridotti, rispettando i tempi stabiliti, devono garantire performance e caratteristiche soddisfacenti per il prodotto commissionato. I metodi tradizionali di progettazione, di prova e di implementazione del software degli embedded control systems fanno in modo che i progettisti aspettino fino alla fase avanzata del loro lavoro, quando i prodotti effettivi o i prototipi e i real-time embedded targets diventano disponibili, per provare se il software funziona realmente in maniera desiderata. Solo a questo punto il progettista può scoprire gli errori commessi nelle fasi iniziali. In un Model-Based Design per sistemi di controllo, il progettista modella l'impianto e il controllore o una parte di essi, e testa l'algoritmo del controllore attraverso una simulazione da PC o real-time. La simulazione real-time abilita l'utente a verificare l'algoritmo in tempo reale, utilizzando il codice generato dal modello. Questo è chiamato Rapid Prototyping

(RP) per il controllore e l' Hardware In the Loop Simulation (HILS) per l' impianto. La Figura 2.31 spiega il concetto del Model-Based Design applicato ai sistemi di controllo basati su MATLAB.

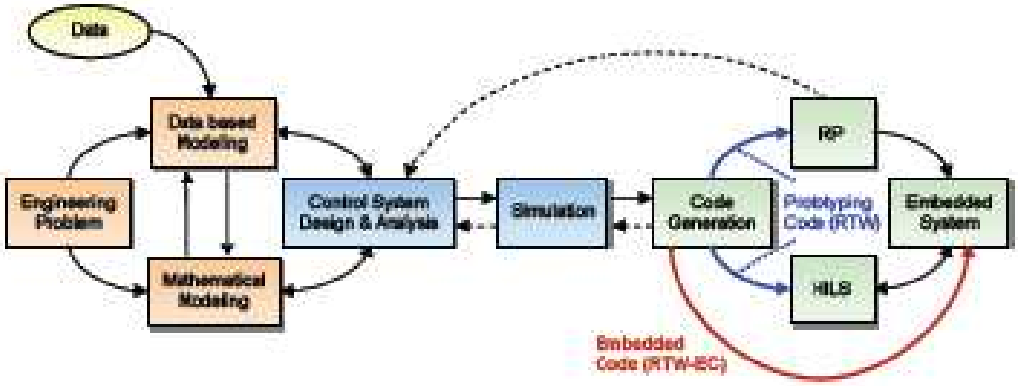


Figura 2.31: Rappresentazione della tecnica Model-Based Design per i sistemi di controllo basati su MATLAB.

MODEL-BASED DESIGN:

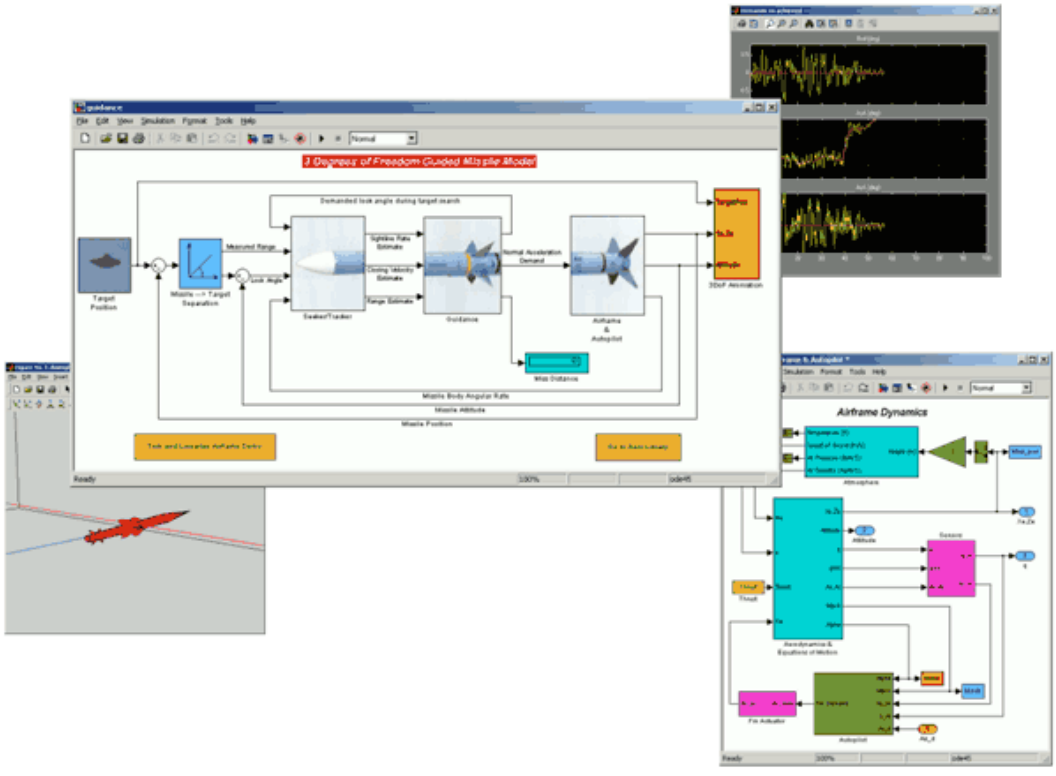


Figura 2.32: Esempi applicativi di progetti realizzati con il Model Based Design

Model-Based Design, insieme ai tools di MathWorks, fornisce una tecnica affermata per creare sistemi di controllo embedded. Questa tecnica è utilizzata per i satelliti, per i velivoli, per molte altre applicazioni aerospaziali, per processi di controllo, per le periferiche dei computer e per i macchinari industriali (Fig. 2.32). Attraverso il Model-Based Design i progettisti possono valutare i loro lavori senza utilizzare prototipi o real-time targets. L'ambiente MathWorks permette agli ingegneri di modellare matematicamente il comportamento di un sistema fisico, di progettare il software, di simulare il modello dell'intero sistema per una previsione accurata e per ottimizzare le performance. Così si riesce a diminuire il lavoro manuale poiché si genera virtualmente software real-time per le prove, per i prototipi e per l'implementazione. I prodotti MathWorks forniscono un ambiente grafico e interattivo col quale si può costruire, gestire e simulare il modello. La natura gerarchica e grafica di questo ambiente permette ai progettisti di comunicare efficacemente diminuendo il rischio di fraintendimento (Fig. 2.33).

MODELLARE E SIMULARE

Per progettare un sistema di controllo embedded e per predire accuratamente le sue performance è necessario comprendere in modo accurato il comportamento dell'intero sistema. MATLAB e Simulink rappresentano il cuore del Model-Based Design per creare modelli matematici di sistemi fisici. L'ambiente grafico di Mathwork permette di utilizzare la tecnica drag-and-drop per trascinare e connettere blocchi così da creare modelli di sistemi dinamici. Questi possono essere a tempo continuo, a tempo discreto, multi-rate o una combinazione di questi tre. L'utente può creare modelli personalizzati o riutilizzare il codice sorgente già scritto incorporando C, Fortran o Ada direttamente nell'ambiente del modello. La struttura e la funzione del sistema possono essere espressi raggruppando gli elementi del modello in ogni possibile combinazione, permettendo così ad ampi team di lavorare simultaneamente al progetto.

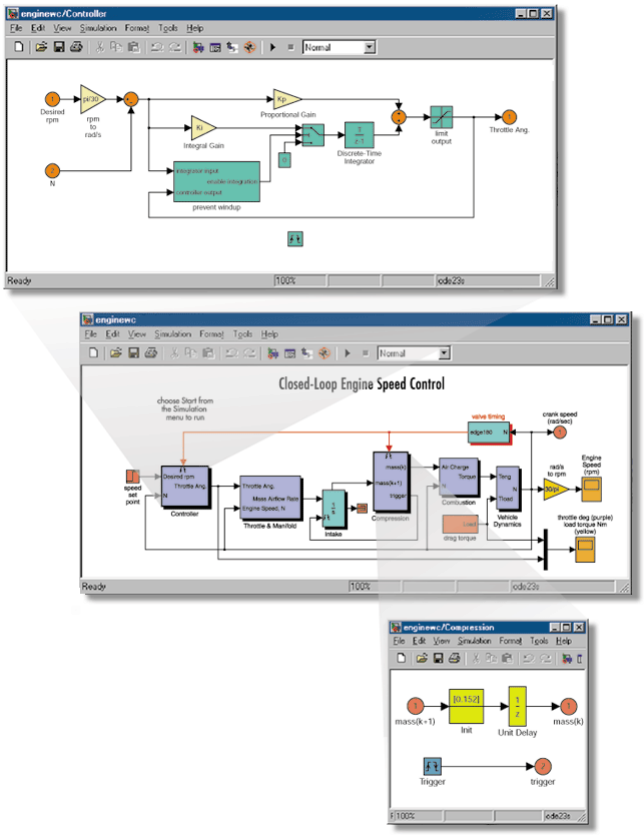


Figura 2.33: Screenshot di un modello gerarchico di un sistema di controllo complesso realizzato con Simulink

Le librerie di questi elementi gerarchici possono essere velocemente create, favorendo il riutilizzo da parte di altri membri del team o per un susseguente progetto. Importante è anche la capacità di modellare graficamente i sistemi event-driven utilizzando i grafici di stato, le tabelle della verità e i diagrammi di flusso. Se il prototipo o il sistema fisico reale sono accessibili e se i dati di input/output possono essere acquisiti da questi, allora il modello matematico può essere generato usando la tecnica di identificazione dei sistemi. Appena gli elementi del modello sono pronti possono essere simulati. La simulazione permette immediatamente di trovare gli errori e di verificare che siano rispettate le specifiche del modello, senza dover aspettare fino ad una fase avanzata del progetto. Andando avanti, il progetto si allarga con l'aggiunta di elementi o con l'aumento della complessità di quelli già presenti. Il progettista può continuare a correggere gli errori attraverso le tecniche di model coverage, performance profiling e debugging interattivo. Quando il modello del sistema fisico raggiunge

il livello di precisione richiesto e la simulazione ha mostrato che il modello è accurato, il sistema di controllo può essere progettato.

CONTROL SYSTEM SOFTWARE DESIGN

Con il modello comportamentale del sistema fisico disponibile il progettista può cominciare il progetto del software del sistema di controllo embedded. L' ambiente di MathWorks per il Model-Based Design supporta svariate tecniche e esigenze di progettazione del sistema di controllo, che spaziano dal semplice al complesso. Ad esempio alcuni progetti possono richiedere l'utilizzo di metodi per il controllo lineare con i quali determinare gli algoritmi e i parametri corretti per il software del sistema di controllo. Utilizzando questa tecnica, con MATLAB e Simulink il progettista può:

- Creare automaticamente i modelli lineari del sistema fisico di cui ha bisogno
- Calcolare i parametri
- Visualizzare i risultati tramite il diagramma di Bode e il luogo delle radici (Fig. 2.34)

Altre applicazioni possono richiedere tecniche meno sofisticate per determinare il corretto progetto del sistema di controllo. Senza tenere conto del metodo usato per la progettazione, l'ambiente MathWorks per il Model-Based Design, consente di utilizzare una simulazione interattiva per valutare velocemente ogni progetto del modello del sistema di controllo congiuntamente al modello del sistema fisico ed evitare quindi il rischio di errori, la spesa e la necessità di prototipi o di sistemi fisici reali.

Una volta completato il progetto si deve considerare l'ambiente di funzionamento desiderato. Il progettista può specificare i dettagli di implementazione del software direttamente nell'ambiente del modello. L' ambiente MathWorks supporta tutti gli aspetti della progettazione del software del sistema di controllo inclusi il processore, l'interfaccia e i comandi standard. La strutture dati di cui si ha bisogno per rispettare gli standard del software o i requisiti dell'interfaccia

dell'ambiente desiderato possono essere definiti come parte del modello del sistema e quindi realizzati quando il software del sistema di controllo embedded viene automaticamente generato. Quando il progetto del software del sistema di controllo è completo si può simulare l'intero modello del sistema.

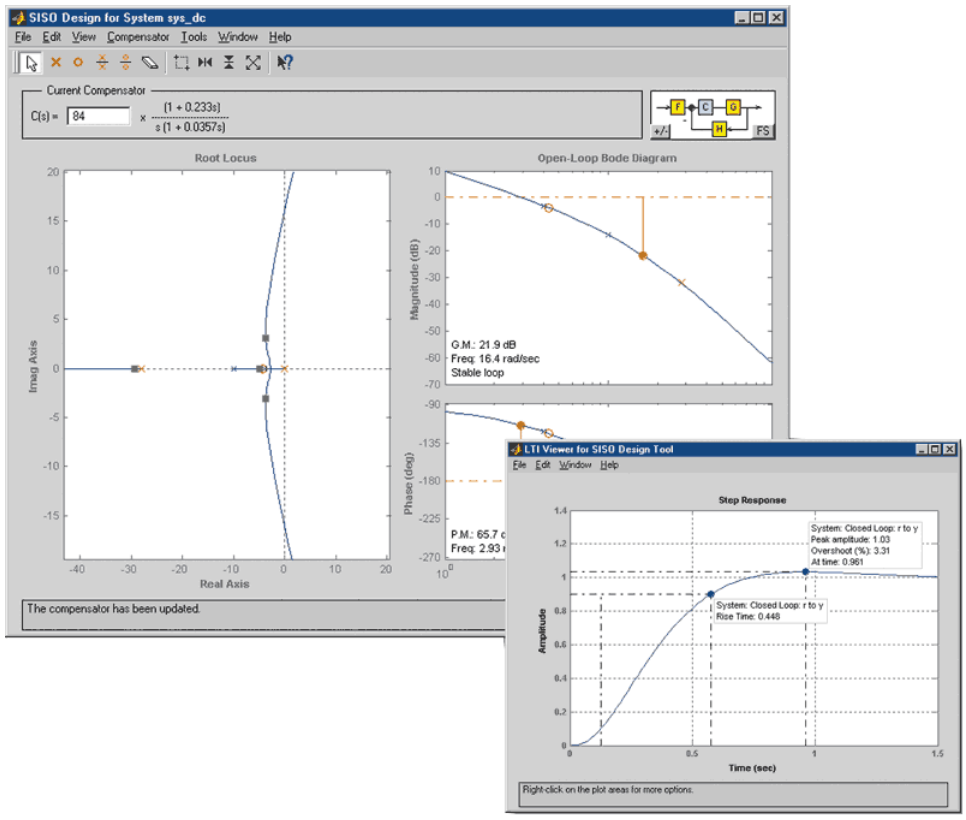
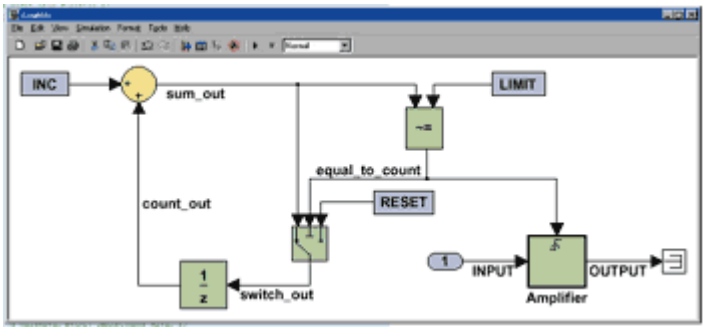


Figura 2.34: Visualizzazione dei risultati del progetto in ambiente MATLAB

Durante la simulazione si possono automaticamente raccogliere informazioni sul profilo, che saranno d'aiuto per mettere in atto la performance e scoprire eventuali errori. Se la performance non soddisfa le aspettative o se si trovano degli errori si può facilmente cambiare il modello per risolvere il problema. Si può simulare nuovamente il modello per avere conferma del cambiamento o della correzione. Una volta che la simulazione dell'intero modello del sistema ha mostrato che il progetto soddisfa i requisiti, si può automaticamente generare un software per fare una prova real-time e per l'implementazione, usando il modello come specifica.

EMBEDDED SOFTWARE TESTING AND IMPLENTATION

Il codice real-time per la prova, per la convalida e per l'implementazione embedded può essere automaticamente generato sul processore desiderato, utilizzando il modello del sistema e Real-Time Workshop (Fig. 2.35). Appena creato, il codice è automaticamente ottimizzato per una veloce esecuzione ed un uso efficiente della memoria. Il codice automaticamente generato dal modello del sistema evita gli errori dovuti alla traduzione manuale del modello in codice e fa risparmiare tempo, permettendo ai progettisti di concentrarsi su compiti più impegnativi. MathWorks fornisce un ambiente software chiamato xPC Target per la prova real-time del prototipo, per la calibrazione e per la convalida di tale codice. xPC Target include un kernel real-time, numerosi dispositivi driver e tutti i software necessari per creare un sistema di controllo del prototipo, per la prova e per la convalida real-time del software. xPC Target può anche essere utilizzato per fornire la funzione di hardware-in-the-loop, usando il codice generato automaticamente dal modello del sistema fisico. La prova hardware-in-the-loop permette al progettista di simulare in tempo reale le caratteristiche ed il comportamento del sistema fisico. Così facendo il prototipo o la produzione del software del sistema di controllo possono essere testati senza che sia necessario l'hardware effettivo o l'ambiente operativo. Dopo che il software embedded è stato completamente testato in real-time, può essere implementato nel processore desiderato. Usando Real-Time Workshop Embedded Coder si può modificare il software embedded per specificare le piattaforme hardware e software desiderate, definendo i formati del codice e delle interfacce e incorporando i dispositivi driver. In seguito si può riutilizzare le informazione della configurazione ogni volta che il codice viene generato. MathWorks ha anche pacchetti preconfigurati che possono aiutare l'utente a generare automaticamente un codice funzionante per diversi ambienti embedded, sia software che hardware. Quando richiesto, il codice generato automaticamente può essere anche testato utilizzando un software standard e delle regole che assicurino l' alta qualità e l’integrità del software. In questo modo è garantita la sicurezza dell'applicazione. Infine si può generare automaticamente la documentazione per l'intero progetto portando l’informazione contenuta nel modello in un formato di testo.




```
/* model step function */
void counter_step(void)
{
  /* local block i/o variables */
  uint8_T rtb_root_count_out;
  uint8_T rtb_root_sum_out;
  uint8_T rtb_root_switch_out;

  /* UnitDelay Block: <Root>/Unit Delay */
  rtb_root_count_out = counter_DWork.root_Unit_Delay_DSTATE;

  /* Sum Block: <Root>/Sum */
  rtb_root_sum_out = (INC) + rtb_root_count_out;

  /* RelationalOperator Block: <Root>/Relational Operator */
  counter_B.root_equal_to_count = (rtb_root_sum_out != (LIMIT));

  /* trigger SubSystem Block: <Root>/Amplifier */
  if ((ZCEventType) (counter_B.root_equal_to_count && !counter_PrevZC.root_Amplifi)) {

    /* Output and update for trigger system: <Root>/Amplifier */
    /* Gain Block: <S1>/Multiply */
    OUTPUT = INPUT * ((int32_T)( K ));
  }
  counter_PrevZC.root_Amplifi = counter_B.root_equal_to_count;

  /* Switch Block: <Root>/Switch */
  if (counter_B.root_equal_to_count) {
    rtb_root_switch_out = rtb_root_sum_out;
  } else {
    rtb_root_switch_out = (RESET);
  }

  /* UnitDelay Block: <Root>/Unit Delay */
  counter_DWork.root_Unit_Delay_DSTATE = rtb_root_switch_out;
}
```

Figura 2.35: Modello Simulink e relativo codice generato automaticamente attraverso Real-Time Workshop

2.3.4 REAL-TIME WORKSHOP

Real-Time Workshop genera ed esegue codice C per lo sviluppo e il controllo di algoritmi modellati con Simulink e con l' Embedded MATLAB (Fig. 2.36-2.37). Il codice risultante può essere utilizzato per molte applicazioni Real-Time e non Real-Time. E' possibile mettere a punto e monitorare il codice generato utilizzando i blocchi di Simulink ed effettuando un'analisi delle proprietà intrinseche. Si può anche interagire con un codice diverso da quello MATLAB e Simulink.

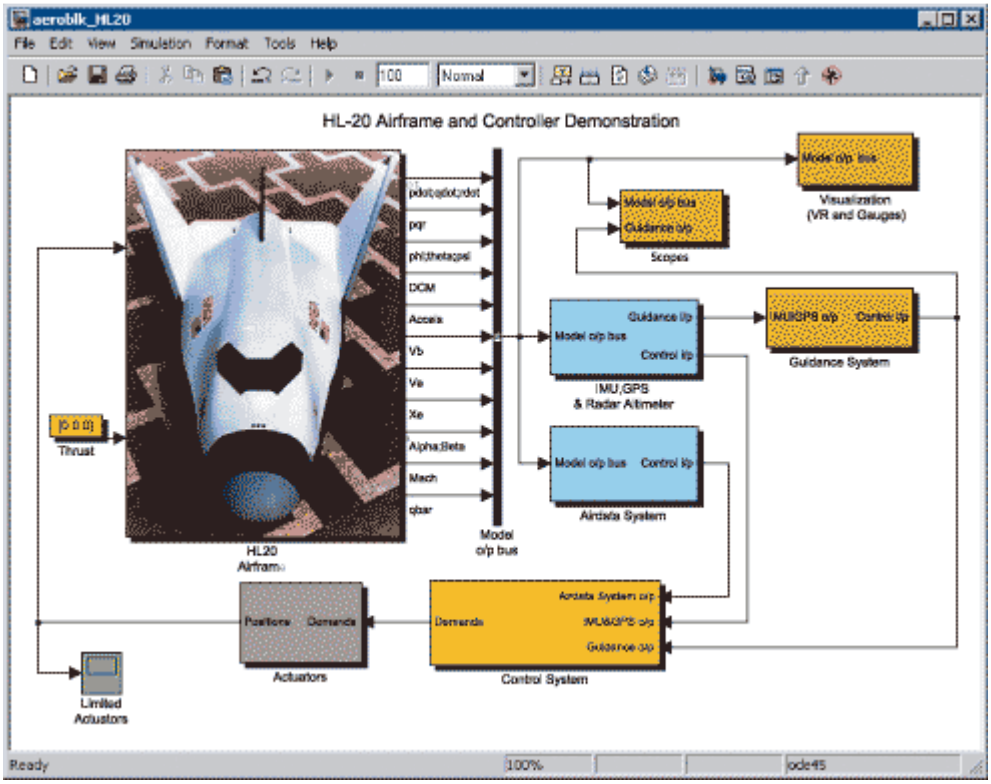


Figura 2.36: Esempio di applicazione implementata con Real-Time Workshop

CARATTERSTICHE PRINCIPALI:

- Genera codici e file eseguibili ANSI/ISO C e C++ per modelli di Simulink discreti, continui o ibridi.
- Usa schemi a blocchi per generare in maniera incrementale il codice per una grande varietà di applicazioni.
- Supporta dizionari di Simulink per dati interi, a virgola fissa e a virgola mobile.
- Genera codice per modelli singol-rate, multi-rate e asincroni.
- Supporta sistemi operativi single-tasking e multi-tasking e ambienti bare-board.
- Rende possibile un'ottimizzazione del codice che migliora la velocità di esecuzione.
- Fornisce funzionalità per la personalizzazione e l'integrazione del codice.

LAVORARE CON RTW:

Real Time Workshop è parte integrante dell' ambiente Simulink, infatti per interagire con Real-Time Workshop si utilizza l'interfaccia grafica Model Explorer. Questo ci dà la possibilità di configurare, in una locazione prefissata di Simulink, le impostazioni del codice generato. Con Model Explorer è possibile:

- Generare il codice per sistemi, o sottosistemi, Simulink.
- Selezionare il target di Real Time Workshop.
- Configurare il target per la generazione del codice.
- Gestire le impostazioni di più configurazioni.

Il Model Advisor di Simulink controlla la configurazione del modello e offre un consiglio su come ottimizzare la configurazione delle impostazioni in base agli obiettivi dichiarati.

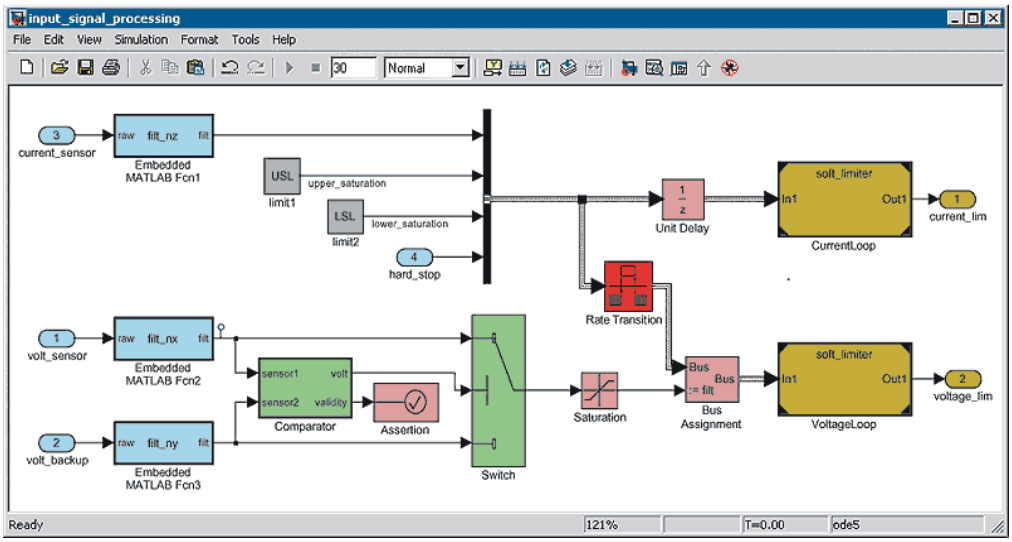


Figura 2.37: Esempio di un sistema realizzato con funzionalità Simulink supportate da Real-Time Workshop

SELEZIONARE GLI OBIETTIVI:

RTW utilizza file target template per tradurre il modello Simulink in codice ANSI/ISO C. Il target template specifica l'ambiente in cui il codice generato verrà eseguito. L'utente può sviluppare il suo target personalizzato o prenderne uno di default supportato da Real-Time Workshop. I target disponibili sono:

- **Generic Real Time Target:** genera codice per la scelta interattiva dei

parametri del modello, acquisisce e mostra sul display i risultati delle simulazioni real-time e alloca dati staticamente.

- **Generic Real Time Malloc Target:** utilizza l'allocazione dinamica della memoria durante la generazione del codice, abilitando l'utente ad includere istanze multiple del modello o più modelli in un eseguibile.
- **S-Function Target:** converte i modelli in S-Function DLLs, rendendone possibile la condivisione per la simulazione.
- **Rapid Simulation Target:** garantisce una piattaforma veloce e flessibile per testare il funzionamento di batch o la simulazione di Monte Carlo, basandosi su una risoluzione con passo fisso o variabile che rende facile la variazione dei parametri e dell'informazione sul segnale in ingresso.
- **Tornado Target:** genera codice per l'esecuzione su VxWorks, un RTOS di Wind River System.

Questi target si possono estendere in modo da creare sia un'interfaccia run-time personalizzata sia file per dispositivi che permettono di accedere all' esecuzione Real-Time Workshop e alle funzionalità di debug.

GENERAZIONE DI CODICE PER MODELLI SIMULINK ED EMBEDDED MATLAB:

Real-Time Workshop genera codice C, per le funzioni Embedded MATLAB, direttamente dalla linea di comando di MATLAB utilizzando la funzione emlc. Real-Time Workshop offre il più completo supporto compatibile con le caratteristiche e le componenti di Simulink, per mezzo di:

- Riferimento al modello che permette la generazione incrementale del codice.
- Blocchi di funzioni Embedded MATLAB in Simulink e funzioni Embedded MATLAB in Stateflow.
- Oggetti Bus che rendono possibile la generazione di strutture nel codice.
- Sottosistemi atomici che rendono possibile il riutilizzo del codice.
- Simulink S-Function che rende possibile la simulazione e l'interfacciamento del codice sorgente.

DEFINIZIONE E CONTROLLO DELLE INFORMAZIONI:

Real-Time Workshop permette di controllare il modo in cui appare l'informazione del modello nel codice generato. Permette inoltre di gestire i dati nei seguenti modi:

- Dichiarando i tipi di dati per mezzo di blocchi specifici.
- Specificando l'area di memoria per regolare e calibrare i parametri o le costanti.
- Specificando l'area di memoria per monitorare e acquisire l'informazione del segnale.

Real-Time Workshop genera il codice a partire da dati salvati nel diagramma o nel dizionario fornito da Simulink Model Explorer.

ESEGUIRE IL CODICE IN AMBIENTE REAL-TIME:

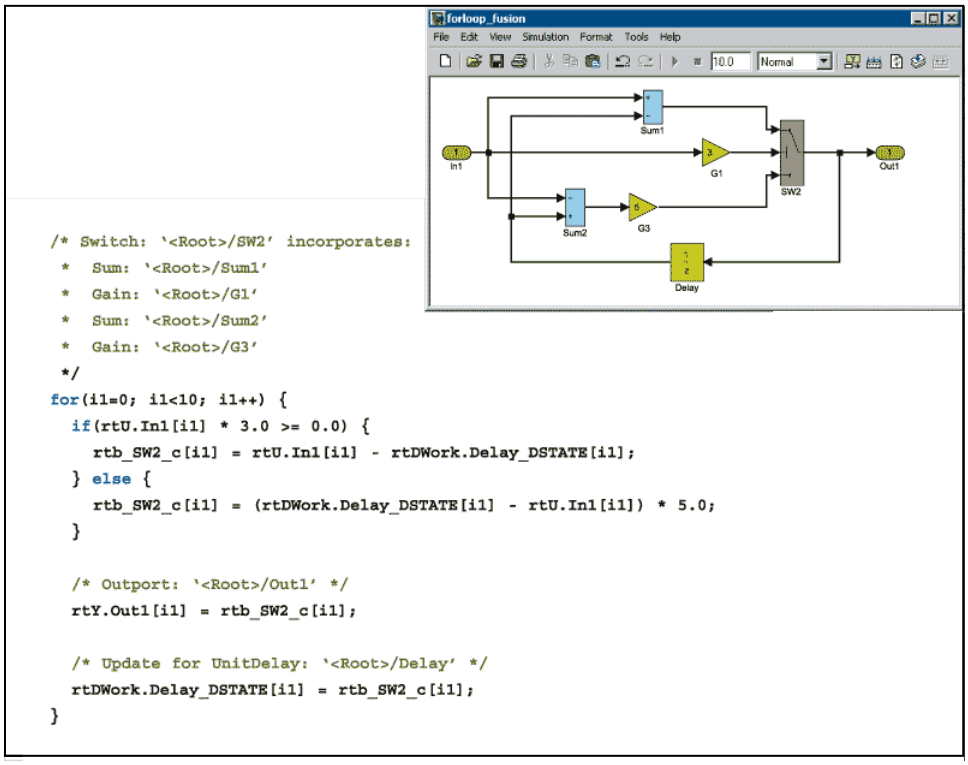
Real-Time Workshop fornisce un'intera struttura per eseguire il codice generato in real-time e per incorporarlo nell'ambiente di esecuzione. Real-Time Workshop genera un codice single-rate o multi-rate basato su un tempo di campionamento periodico, specificato nel modello. Il codice è usato con o senza RTOS, in modalità single tasking, multitasking o asincrona.

- Single tasking: un semplice scheduler invoca il codice generato come se fosse un unico thread di esecuzione e gestisce la prelazione tra le frequenze.
- Multitasking: uno scheduler a frequenza fissata invoca il codice generato gestendo la prelazione tra le frequenze. In ambiente bare-board è possibile gestire il codice con interruzioni annidate mentre in ambiente RTOS si usa il criterio di priorità.
- Asincrono: le frequenze non periodiche ed asincrone sono specificate utilizzando le S-Function di Simulink. Real-Time Workshop traduce queste frequenze in codice basato sull'ambiente di esecuzione. L'utente genera e modella il codice per eventi, quali interruzioni hardware.

REAL-TIME DEPLOYMENT:

Simulink e RTW offrono un set completo di funzionalità indipendenti dai target, per l'utilizzo real-time:

- La possibilità di specificare la priorità per ogni frequenza del modello.
- Contatori e Timers per calcolare il tempo assoluto e trascorso.
- Un blocco Transition Rate per specificare le modalità di trasferimento dei dati tra le varie frequenze (ad es. i semafori e la mutua esclusione), per mantenere l'integrità dei dati e migliorare il rendimento.
- Rilevare e gestire gli errori logici che si verificano.




```
/* Switch: '<Root>/SW2' incorporates:
 *  Sum: '<Root>/Sum1'
 *  Gain: '<Root>/G1'
 *  Sum: '<Root>/Sum2'
 *  Gain: '<Root>/G3'
 */
for(i1=0; i1<10; i1++) {
  if(rtU.In1[i1] * 3.0 >= 0.0) {
    rtb_SW2_c[i1] = rtU.In1[i1] - rtDWork.Delay_DSTATE[i1];
  } else {
    rtb_SW2_c[i1] = (rtDWork.Delay_DSTATE[i1] - rtU.In1[i1]) * 5.0;
  }

  /* Outport: '<Root>/Out1' */
  rtY.Out1[i1] = rtb_SW2_c[i1];

  /* Update for UnitDelay: '<Root>/Delay' */
  rtDWork.Delay_DSTATE[i1] = rtb_SW2_c[i1];
}
```

Figura 2.38: Esempio di schema a blocchi e del relativo codice generato da Real-Time Workshop

OTTIMIZZAZIONE DEL CODICE:

Real-Time Workshop fornisce ottimizzazioni del codice per migliorarne l'efficienza (Fig. 2.38):

- Riutilizzabilità del codice.
- Expression Folding.
- Riutilizzo della memoria del segnale.
- Eliminazione delle dead path.
- Inlining dei parametri.
- Librerie di funzioni per i target a singola precisione e precostituiti (tra cui ISO C e GNU C).

PERSONALIZZARE RTW:

Il codice generato e il suo ambiente di esecuzione possono essere personalizzati con l'aggiunta di codice C, Fortran, Ada e C++.

MONITORING AND TUNING CODE:

Real-Time Workshop permette il monitoraggio e la calibrazione dei blocchi di segnali e dei parametri utilizzando le seguenti interfacce:

Target Based C API: permette al codice scritto dall'utente di accedere ai blocchi di output ed ai parametri al di fuori di MATLAB e di Simulink.

Host Based ASAP2 data exchange file: permette di utilizzare la descrizione standard dei dati di ASAP2 per la misurazione di dati, la calibrazione e la diagnosi di sistemi non appartenenti all'ambiente MATLAB/Simulink.

Simulink External Mode: permette di scaricare i valori di nuovi parametri e di caricare i valori dei segnali per vederli in Simulink o per registrarli nel workspace di MATLAB.

2.3.5 REAL-TIME WORKSHOP EMBEDDED CODER

Real-Time Workshop Embedded Coder genera, a partire da modelli Simulink e Stateflow, un codice C che ha la stessa chiarezza e la stessa efficienza del codice

scritto da programmatori professionisti (Fig. 2.39). Il codice generato è compatto e veloce, caratteristiche fondamentali nei sistemi embedded, per eseguire un test rapido sulle schede di prova, sui microprocessori utilizzati nella produzione di massa e sui simulatori real-time.

Insieme a questo software è fornito un supporto completo per l'integrazione di applicazioni, di funzioni e di dati. Si può utilizzare Real-Time Workshop Embedded Coder per individuare e per verificare le qualità del software. Il codice generato è ANSI/ISO conforme al C e questo fa sì che possa essere eseguito su qualunque microprocessore o sistema operativo real-time (RTOS). I prodotti Embedded, disponibili separatamente, estendono Real-Time Workshop Embedded Coder con specifici pacchetti di supporto.

CARATTERISTICHE PRINCIPALI:

- Genera il codice ANSI/ISO C e C++ a partire da modelli Simulink e Stateflow, con una velocità di esecuzione, un impiego di memoria e una leggibilità paragonabili al codice scritto a mano.
- Ottimizza Real-Time Workshop e Stateflow Coder aggiungendo delle caratteristiche, alla struttura del codice, che sono essenziali per il miglioramento del prodotto.
- Mantiene tutte le proprietà degli oggetti e del dizionario di Simulink, comprese le classi definite in memoria dall'utente, i tipi e gli alias.
- Permette la personalizzazione del codice presente nella libreria e delle funzioni necessarie per la generazione del codice per il processore.
- Opera partizioni coincise del codice multi-rate per uno scheduling efficiente con o senza un RTOS.
- Fornisce numerosi strumenti per chiarire le sue proprietà mediante collegamenti ipertestuali riferiti alla scrittura del codice relativo al modello.
- Verifica il codice importandolo automaticamente in Simulink per testare il software-in-the-loop.
- Genera la documentazione riguardo al codice all'interno di Simulink Model Explorer e produce resoconti sfruttando le capacità di Simulink.

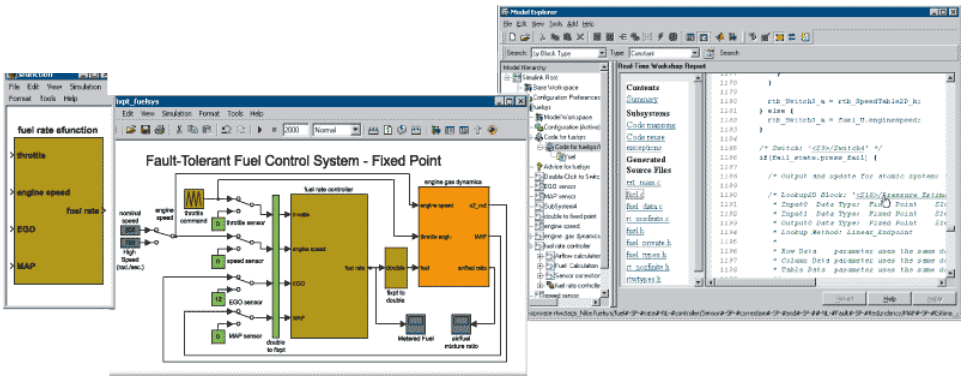


Figura 2.39: Screenshot di un modello a virgola fissa, relativo codice generato e blocco S-Function.

LAVORARE CON REAL-TIME WORKSHOP EMBEDDED CODER:

Real-Time Workshop Embedded Coder fornisce un supporto intrinseco, in Simulink e in Stateflow, per la generazione del codice, includendo modelli a tempo continuo, a tempo discreto e event-based. E' possibile interagire con Real-Time Workshop Embedded Coder e configurare le impostazioni per la generazione del codice utilizzando Simulink Model Explorer (Fig. 2.40). Con Model Explorer è possibile:

- Generare il codice per sistemi, o sottosistemi, Simulink.
- Selezionare il target di Real-Time Workshop.
- Configurare il target per la generazione del codice.
- Gestire le impostazioni di più configurazioni.

Il Model Advisor di Simulink controlla la configurazione del modello ed offre un consiglio su come ottimizzare le impostazioni in base agli obiettivi dichiarati.

SELEZIONARE TARGETS:

Real-Time Workshop Embedded Coder utilizza file di destinazione di tipo template per tradurre il modello di Simulink o di Stateflow in codice C. I target templates specificano l'ambiente nel quale il codice generato verrà eseguito. L'Embedded Real-Time Target, incluso in Real-Time Workshop Embedded Coder, genera codice ANSI/ISO C e può essere configurato sia per variabili in virgola fissa che in virgola mobile.

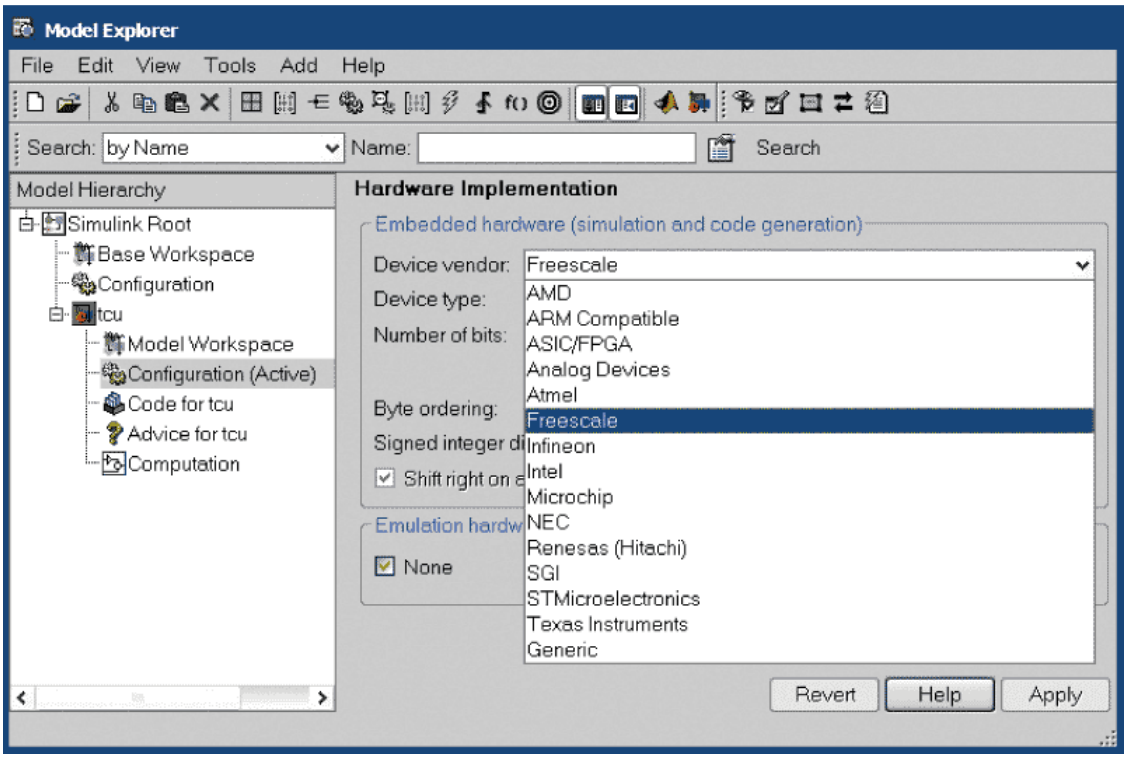


Figura 2.40: Screenshot della scelta del microprocessore per la generazione del codice

Si può anche estendere l' Embedded Real-Time Target per impiegarlo in applicazioni specifiche. E' possibile generare il codice per qualunque microprocessore, specificando le dimensioni degli interi e le caratteristiche dei target necessari oppure scegliendo da una lista di target con impostazioni predefinite.

DEFINIZIONE E GESTIONE DEL CODICE GENERATO:

Simulink Real-Time Workshop Embedded Coder utilizza la seguente classificazione dei dati quando viene generato il codice:

- **Simulink data objects**: fornisce delle classi di memoria predefinite, includendo: const, volatile, exported global, imported global, #define,structure, bitfield e i metodi d'acceso get e set.
- **Module packaging data objects**: fornisce attributi preconfigurati, quali ad esempio segmenti di memoria per calibrare o aggiornare le tabelle di ricerca, per advanced data objects utilizzati tipicamente nella produzione di massa.

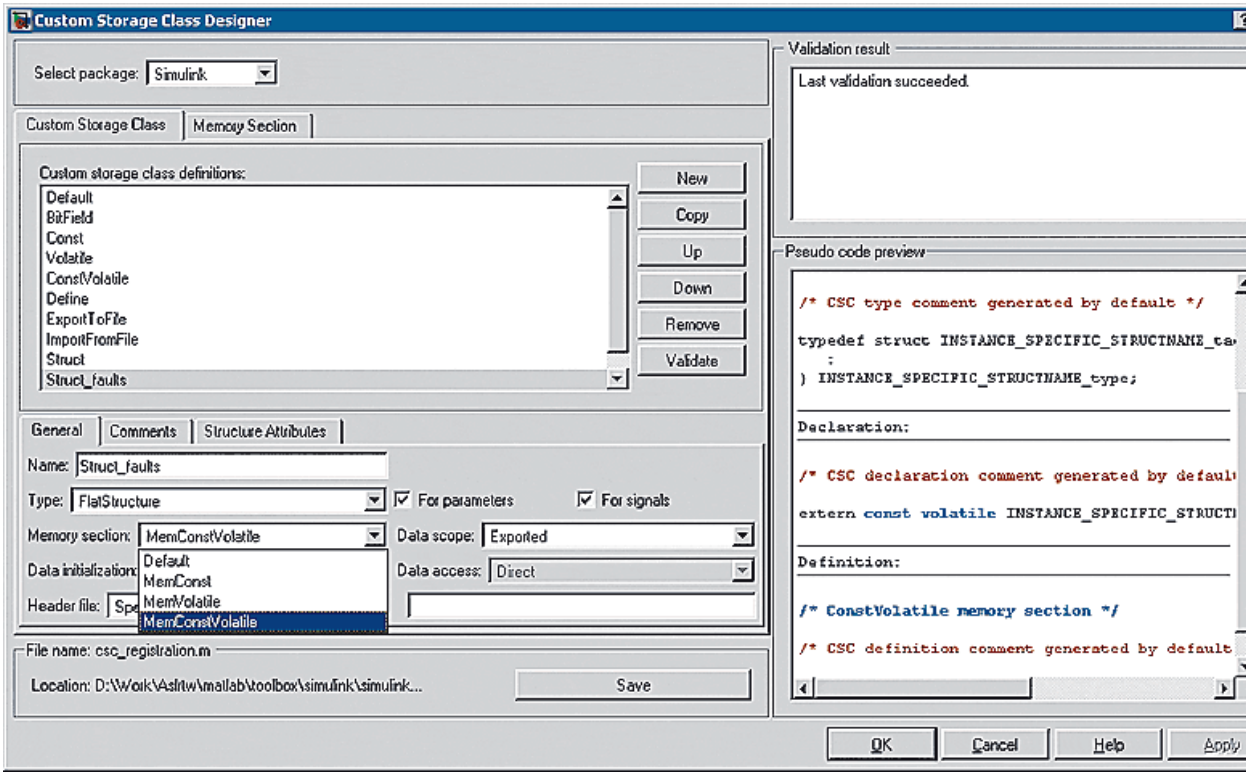


Figura 2.41: Screenshot del Custom Storage Designer

- **Module packaging data objects**: fornisce attributi preconfigurati, quali ad esempio segmenti di memoria per calibrare o aggiornare le tabelle di ricerca, per advanced data objects utilizzati tipicamente nella produzione di massa.
- **User data types**: permette sia di creare tipi astratti per dati complessi sia di ottenere come risultato il controllo preciso di come appare il modello nel codice generato e nell'interfaccia (con dati di qualsiasi complessità) sia di accrescere o di sostituire i tipi predefiniti in Simulink.

I seguenti tools aiutano nel progetto e nell'implementazione di un dizionario:

- **Custom Storage Class Designer**: permette all'utente di creare graficamente le definizioni e le dichiarazioni per importare strutture dati nel codice generato, per esportare dati, per conservare memoria, o

generare automaticamente cambi di dati standards, come ASAM/ASAP2 (Fig.2.41).

- **Simulink Data Object Wizard**: analizza grafici Simulink e Stateflow e popola automaticamente lo spazio di lavoro con appositi data objects (Fig. 2.42).

Real-Time Workshop Embedded Coder permette di scambiare, in Simulink, file ASAP2, abilitando così l'utente a esportare il modello contenente dati di qualsiasi complessità. Inoltre si possono modificare le strutture incorporate per produrre altri meccanismi di cambiamento dei dati.

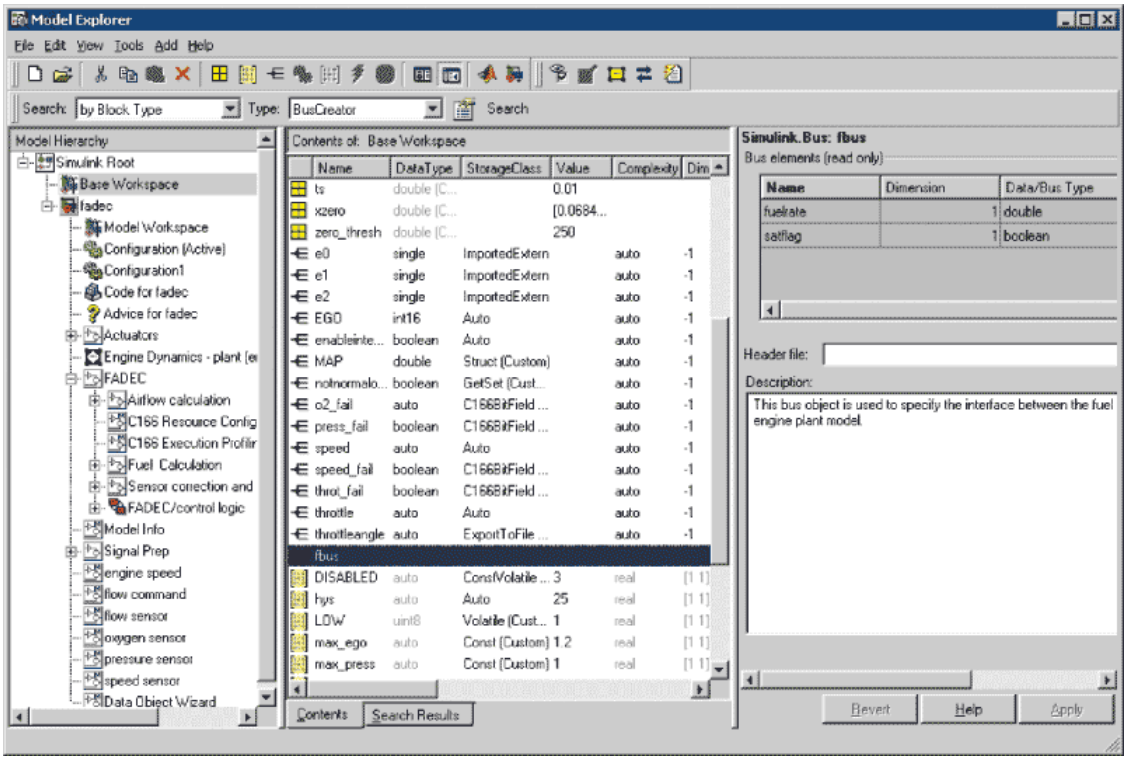


Figura 2.42: Model Explorer popolato da una grande varietà di data objects

ESEGUIRE IL CODICE IN REAL-TIME PRODUCTION ENVIROMENT:

Real Time Workshop Embedded Coder fornisce una struttura completa e ottimizzata per incorporare il codice generato nell'ambiente di esecuzione real-time. Il codice generato è indipendente dai target e può essere eseguito con o senza un RTOS, in modalità singola, multitasking o asincrona.

GENERARE UN PROGRAMMA PRINCIPALE:

Un programma principale estendibile è generato sulla base delle informazioni fornite dall'utente per lo sviluppo del codice nell'ambiente real-time. Questo programma permette di generare e di costruire un eseguibile completamente personalizzato, a partire dal modello.

RAGGRUPPAMENTO DI FREQUENZE:

Real-Time Workshop Embedded Coder genera codice single-rate o multi-rate utilizzando i tempi di campionamento periodici specificati nel modello. Per modelli multi-rate e multitasking impiega una strategia chiamata raggruppamento di frequenze che genera funzioni separate per il lavoro a frequenza base e per ogni lavoro,presente nel modello, ad una sottofrequenza.

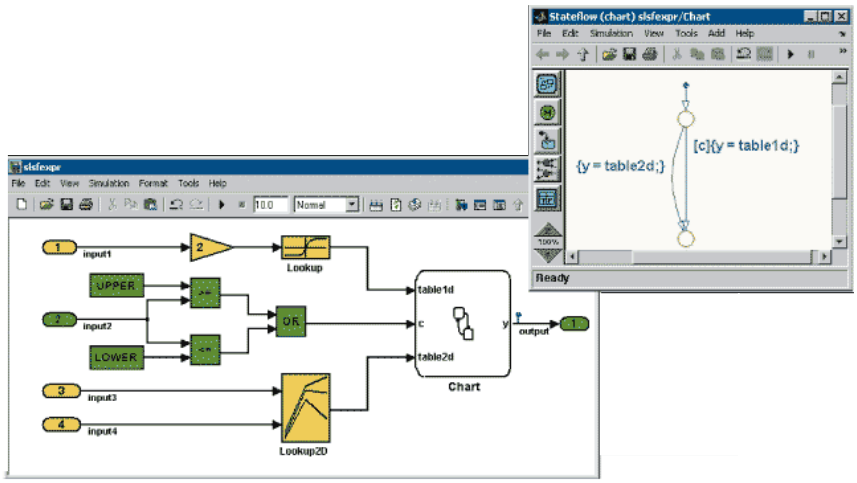
OTTIMIZZAZIONE E IMPACCHETTAMENTO DEL CODICE:

Real-Time Workshop Embedded Coder permette di controllare le interfacce delle funzioni, di preservare le espressioni e di applicare ottimizzazioni su più blocchi per ridurre le dimensioni del codice. I dati possono essere trasmessi al codice generato come dati globali o come argomenti delle funzioni.

Le opzioni di ottimizzazione di Real-Time Workshop Embedded Coder permettono di:

- Generare codice di basso livello (specifico del processore) per funzioni e operazioni matematiche.
- Riutilizzare il codice per ambienti esterni o per esportarlo in modo ereditario.
- Eliminare le inizializzazioni, le terminazioni, le acquisizioni e gli errori di scrittura del codice.
- Unire le funzioni di output/update per ridurre le dimensioni del codice.
- Rimuovere il codice in virgola mobile per le applicazioni che richiedono solo interi.

Real-Time Workshop Embedded Coder fornisce un module packaging features che permette di impacchettare il codice per attenersi agli stili e agli standard di uno specifico software (Fig. 2.43).




```
/* Model step function */
void slsfexpr_step(void)
{
  /* Stateflow: '<Root>/Chart' incorporates:
   *  Logic: '<Root>/Logical Operator'
   *  RelationalOperator: '<Root>/Relational Operator1'
   *  RelationalOperator: '<Root>/Relational Operator'
   *  Constant: '<Root>/Constant'
   *  Constant: '<Root>/Constant1'
   *  Lookup: '<Root>/Look-Up Table'
   *  Gain: '<Root>/Gain'
   *  Lookup2D: '<Root>/Look-Up Table (2-D)'
   */
  if(((UPPER >= rtU.input2) || (rtU.input2 <= LOWER))) {
    output = rt_Lookup(T1Break, 11, input1 * 2.0, T1Data);
  } else {
    output = rt_Lookup2D_Normal(T2Break, 3, T2Break, 3, T2Data,
      input3, input4);
  }
}
```

Figura 2.43: Parti di codice che rappresentano blocchi racchiusi nello switch

Si può sia controllare l'organizzazione interna e il formato di ogni file generato sia determinare come il dato globale è definito e come è referenziato.
Templates e simboli permettono di specificare il contenuto e la disposizione dei commenti e delle sezioni del codice all'interno dei file che contengono il codice generato.

VERIFICA E DOCUMENTAZIONE DEL CODICE:

Real-Time Workshop Embedded Coder offre molte possibilità per verificare il codice generato:

- Il codice generato, sottoforma di un S-function, può essere riportato in Simulink per la prova software-in-the-loop con il modello dell'impianto.
- I commenti e le descrizioni fatte dall'utente per migliorare la leggibilità e la rintracciabilità del codice.
- Un supporto per inserire i requisiti nel codice generato.
- Nomi di identificazione permanenti per minimizzare le differenze di codice tra le modifiche successive del modello.

Real-Time Workshop Embedded Coder fornisce la documentazione del codice in un file HTML che descrive in modo comprensibile le unità del codice e le impostazioni per la configurazione del modello, applicate durante la generazione del codice. Il report include una sezione di riepilogo e una tabella che contiene i file sorgente generati, dotati di link ipertestuali. Se visti in MATLAB Help Browser, questi collegamenti ipertestuali fanno si che i corrispondenti blocchi siano evidenziati nel modello Simulink associato, rendendo il codice facile da tracciare e da rivisitare. E' anche possibile evidenziare il codice a partire dai blocchi per un tracciamento bidirezionale. Il report automaticamente generato in formato HTML mantiene la documentazione aggiornata con la scrittura del codice.

2.3.6 VIRTUAL REALITY TOOLBOX

Il Virtual Relity Toolbox permette di visualizzare e di interagire con le simulazioni di sistemi dinamici in un ambiente di realtà virtuale a 3-D. Il toolbox collega MATLAB e Simulink mediante i grafici della realtà virtuale, abilitando MATLAB o Simulink a controllare la posizione, la rotazione e le dimensioni delle immagini 3-D definite nell'ambiente della realtà virtuale. Il risultato è una rappresentazione animata in 3-D, attraverso la quale è possibile vedere la

composizione e la realizzazione pratica di tutti i sistemi dinamici creati con Simulink.

Si può utilizzare in ambito aerospaziale (Fig. 2.44), biologico, automobilistico e nelle applicazioni meccaniche.

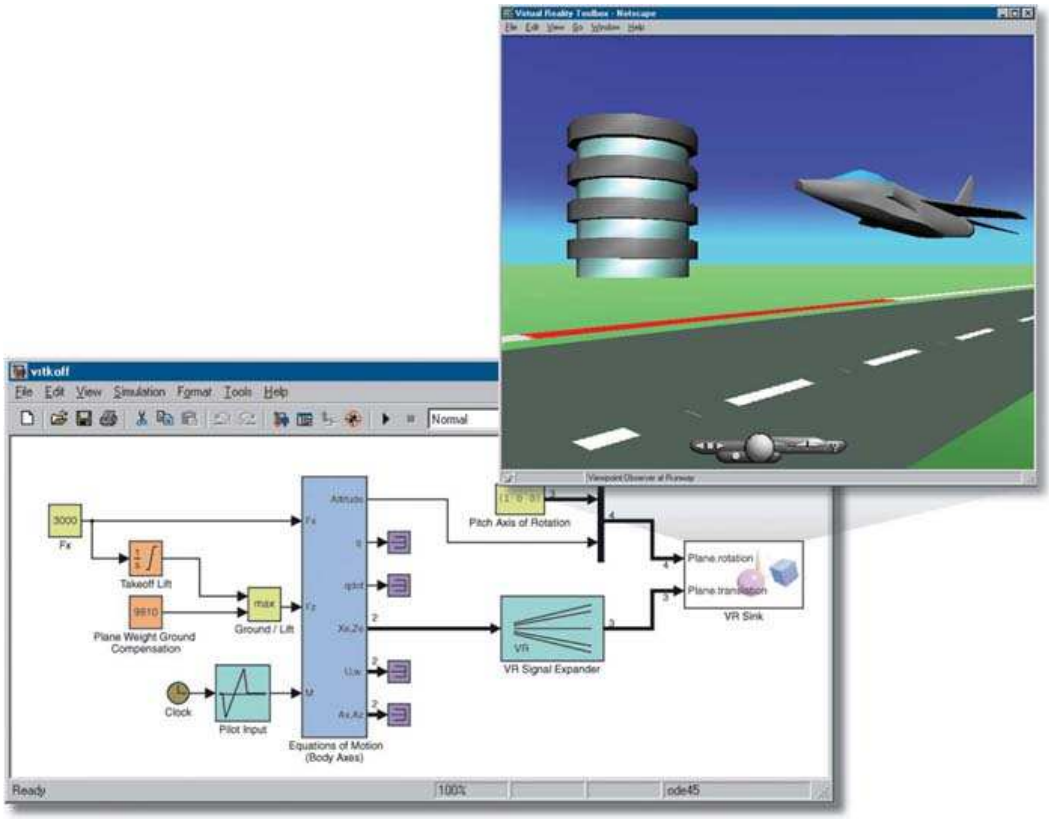


Figura 2.44: Esempio di applicazione del Virtual Reality Toolbox in ambito aerospaziale

CARATTERISTICHE PRINCIPALI:

- Collega i segnali dal Simulink al mondo della realtà virtuale per controllarne alcune proprietà, per esempio il movimento degli oggetti nella realtà virtuale.
- Include tool per vedere e costruire una realtà virtuale.
- Fornisce un'architettura client/server per consentire la comunicazione tra più utenti.
- Produce registrazioni video che nei formati AVI e WRL.
- Interagisce con simulazioni in tempo reale.
- Include funzioni MATLAB per ripristinare e per modificare le proprietà del mondo virtuale.

- E’ possibile connetterlo a dispositivi hardware di input, quali il joystick, Magellan SpaceMouse e Logitech SpaceBall 5000.

COLLEGAMENTO DA SIMULINK AL MONDO VIRTUALE:

Il Virtual Reality Toolbox permette di collegare il modello Simulink ai file della realtà virtuale attraverso un file standard di ricerca. Dopo aver realizzato il collegamento con un file del mondo virtuale, il toolbox fornisce un’interfaccia per trasferire i segnali di Simulink, per controllare i parametri della realtà virtuale, per le immagini e i progetti definiti in quel file. Mediante questo approccio è possibile controllare la posizione, la rotazione e la dimensione di un’immagine utilizzata per rappresentare il movimento e la deformazione di un oggetto fisico. Si possono aggiustare la posizione e l’orientamento di un’immagine per riprodurre il movimento di oggetti nel mondo virtuale. Per esempio è possibile vedere sul display una macchina che si muove sulla strada e si può seguire nel suo percorso mediante un buon controllo dell’immagine (Fig. 2.45) .

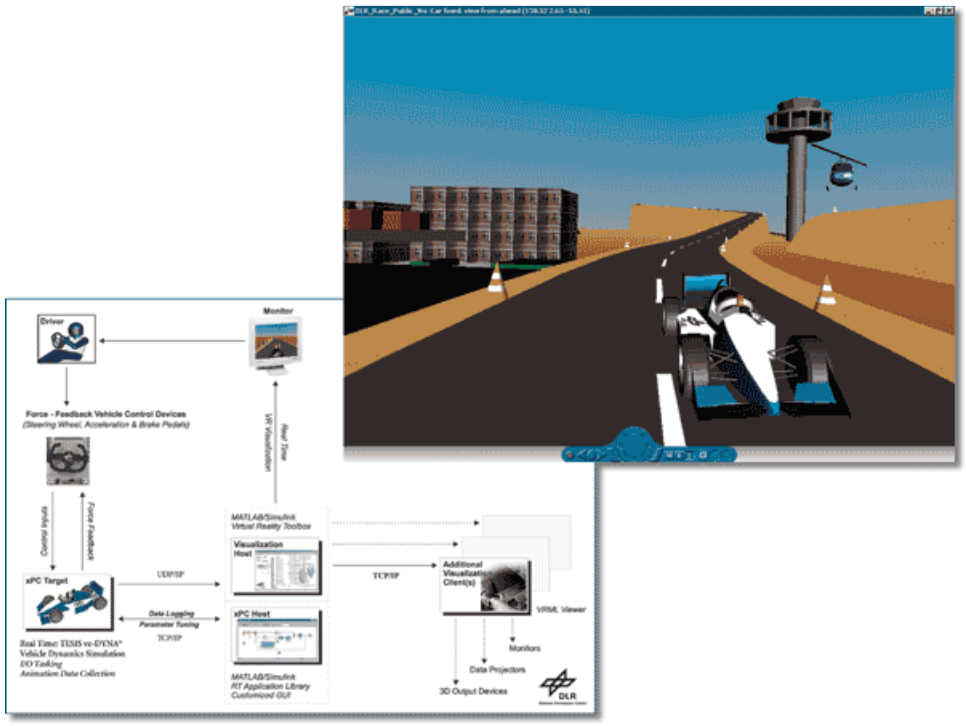


Figura 2.45: Esempio di applicazione del Virtual Reality Toolbox in ambito automobilistico

TOOLS PER VISUALIZZARE E PER COSTRUIRE I MONDI VIRTUALI:

E' possibile creare dei mondi virtuali guidati dall'ambiente Simulink utilizzando lo standard Virtual Reality Modeling Language (VRML). Il Virtual Reality Toolbox fornisce un ambiente di lavoro completo,dotato di:

- un VRML editor per creare mondi di realtà virtuale.
- un VRML viewer per mostrare mondi di realtà virtuale.
- Un opzionale VRML Web browser plug-in per visualizzare mondi virtuali con Windows.

VRML EDITORS:

Il V-Realm Builder nel Virtual Reality Toolbox permette di creare immagini di oggetti fisici utilizzando VRML. E' anche possibile usare il Virtual Reality Toolbox con un'interfaccia che segue gli standard VRML 97. Questa interfaccia permette di importare oggetti 3-D da molti pacchetti CAD, fornendo un processo efficiente per creare modelli visivi dettagliati.

VRML VIEWERS:

Il Virtual Reality Toolbox include tre VRML viewers: il Virtual Reality Toolbox viewer (di default), il blaxxun Contact plug-in per il web browser e l' Orbisnap, un viewer indipendente. Il Virtual Reality Toolbox viewer può essere eseguito su piattaforme multiple. Il blaxxun Contact VRML plug-in permette di vedere l'animazione di un mondo virtuale in un Web browser su Windows. Orbisnap è un viewer VRML indipendente che permette di vedere mondi virtuali o file di animazione già registrati.

AMBIENTE DI LAVORO INDIVIDUALE E COLLABORATIVO:

Il Virtual Reality Toolbox permette di vedere e di interagire con i mondi virtuali simulati su un computer sul quale è in esecuzione Simulink o su computer in rete connessi mediante una LAN o tramite Internet.

In un ambiente di lavoro individuale, nel quale non sono presenti più computer connessi in rete, girano sullo stesso host locale sia il sistema modellato sia la visualizzazione della realtà virtuale. In un ambiente di lavoro collaborativo, il Virtual Reality Toolbox connette il sistema modellato a un Web browser abilitato

al VRML utilizzando TCP/IP, abilitando utenti remoti a vedere i modelli VRML tramite Internet.

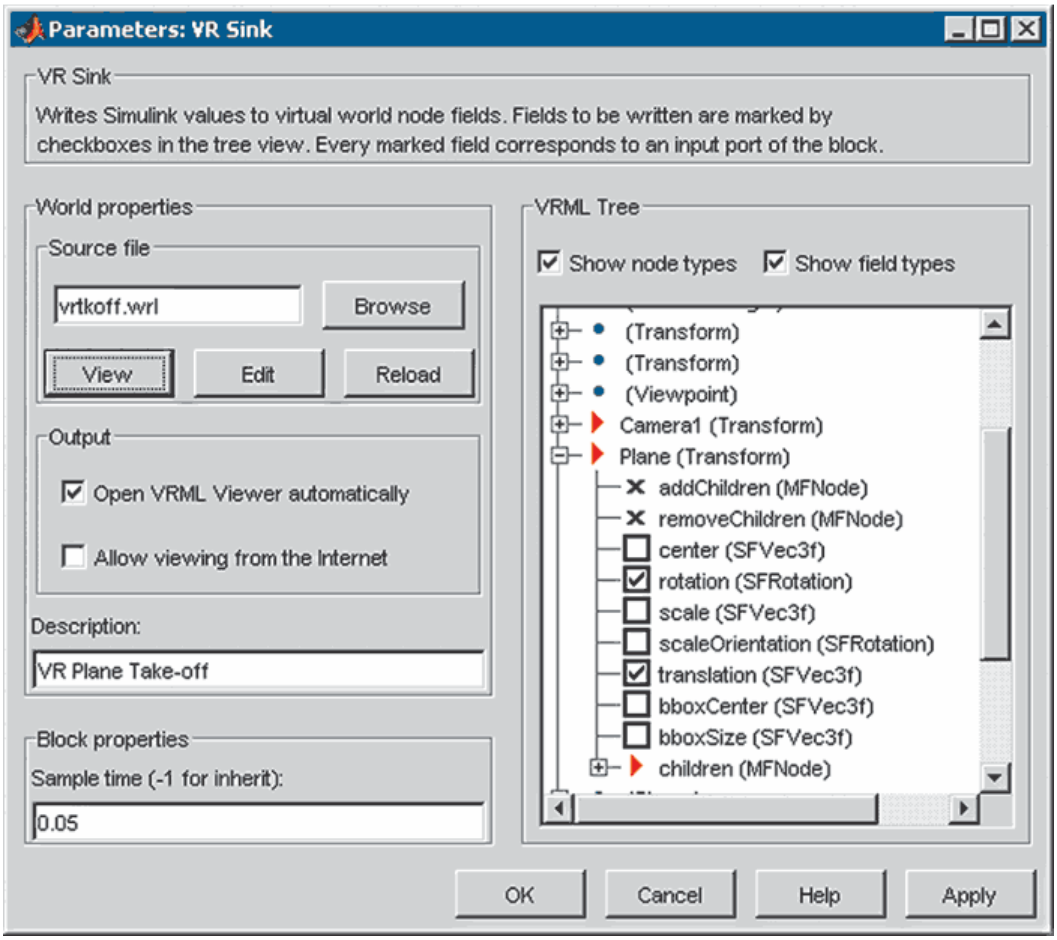


Figura 2.46: Impostazione dei parametri per ottenere la simulazione desiderata

INTERFACCIA MATLAB PER IL VIRTUAL REALITY TOOLBOX:

Il Virtual Reality Toolbox fornisce un'interfaccia MATLAB orientata al mondo della realtà virtuale. Il toolbox utilizza funzioni orientate agli oggetti per creare e controllare il mondo virtuale. Da MATLAB, è possibile fissare e cambiare le posizioni e le proprietà degli oggetti VRML (Fig. 2.46), creare chiamate dall'interfacce grafiche (GUIs) e mappare dati su oggetti virtuali. E' anche possibile vedere il mondo con un VRML viewer, analizzare la sua struttura e fissare nuovi valori per tutti i parametri a disposizione.

SUPPORTO PER IL CODICE C:

Le animazioni del Virtual Reality Toolbox possono essere guidate dal codice C, generato dai modelli Simulink, utilizzando il Real Time Workshop (disponibile separatamente). Questa proprietà è particolarmente utile per le applicazioni con l'hardware-in-the-loop (perché fornisce un'animazione visiva del modello del sistema dinamico e mostra come quest'ultimo s'interfaccia con l'hardware real-time.

SUPPORTO PER I JOYSTICKS E ALTRI HARDWARE D'INPUT:

Il Virtual Reality Toolbox include i blocchi Simulink e le funzioni MATLAB che permettono il controllo dei prototipi virtuali utilizzando controllori di moto 3-D. Comprende:

- Tutti i modelli di dispositivi SpaceMouse (sia seriali che USB)
- Space Traveler controllore di moto
- Logitech SpaceBall 5000
- Force-feedback joysticks

2.3.7 PROGRAMMAZIONE DEL LEGO MINDSTORMS NXT CON MATLAB E CON SIMULINK

Il LEGO MINDSTORMS NXT permette di creare e di controllare robots utilizzando motori e sensori programmabili. Si possono adoperare MATLAB e Simulink per programmare e testare velocemente i robot LEGO MINDSTORMS in svariate circostanze.

CODICE MATLAB PER IL CONTROLLO REMOTO CON LEGO MINDSTORMS:

La modalità di controllo remoto, eseguito sull'host, sfrutta la capacità seriale degli input/output di MATLAB per inviare comandi al LEGO MINDSTORMS NXT attraverso una connessione Bluetooth.

Con il controllo remoto è possibile:

- Iniziare la programmazione immediatamente senza toolbox aggiuntivi.
- Lavorare nell'ambiente MATLAB per lo sviluppo e il debugging.
- Lavorare con la versione per studenti di MATLAB e di Simulink.
- Imparare i rudimenti della programmazione come le iterazioni, le istruzioni condizionali e la programmazione orientata agli oggetti.

CODICE SIMULINK PER IL CONTROLLO EMBEDDED CON LEGO MINDSTORMS NXT:

La soluzione Embedded-Control è sviluppabile utilizzando Simulink, Real-Time Workshop e Real-Time Workshop Embedded Coder; questa viene poi convertita in linguaggio macchina e scaricata sul LEGO MINDSTORMS NXT in modo che vi possa essere eseguita.

Con l'Embedded-Control si possono:

- Programmare i motori e i sensori per farli lavorare contemporaneamente e per ottenere un'esecuzione real-time
- Evitare le limitazioni di range perché il programma viene eseguito direttamente sul LEGO MINDSTORMS NXT.
- Modellare, visualizzare e provare i comportamenti del robot utilizzando il Virtual Reality Tolboox.
- Imparare ad utilizzare gli algoritmi di controllo e il codice generato.

2.3.8 EMBEDDED CODER ROBOT NXT PER LEGO MINDSTORMS

Embedded Coder Robot per LEGO Mindstorms NXT (ECRobot NXT) è un ambiente per il LEGO Mindstorms NXT (Fig. 2.47) che fornisce la capacità di lavorare con il modello dell'NXT. Infatti è possibile sviluppare strategie di controllo, creare un progetto dinamico, simulare e visualizzare le componenti del modello in ambiente grafico 3-D di realtà virtuale. ECRobot NXT permette lo sviluppo degli obiettivi prefissati con Real-Time Workshop Embedded Coder

grazie a OSEK RTOS, un open source che può essere scaricato sull'NXT. Così l'utente può sperimentare la tecnica di Model-Based Design costituita dalla progettazione del modello, dalla simulazione, dalla generazione del codice, dalla decisione degli obiettivi e dalla verifica fisica sull'NXT. ECRobot NXT utilizza software fornito da terze parti per configurare l'ambiente di sviluppo. La maggior parte di tali software, eccetto MATLAB, sono open source.

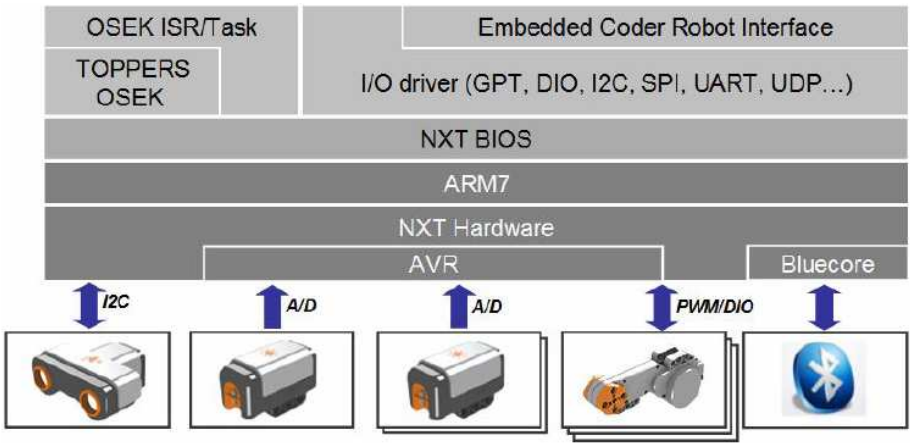


Figura 2.47: Struttura hardware e software dell'NXT

E' necessario installare Cygwin e GNU Make per configurare un ambiente nel quale è possibile effettuare una cross-compilation. GNU ARM è un GCC per la CPU ARM7 (ATMELAT91SAM7S256) che è situato nell'NXT.

A partire dall'nxtOSEK 2.0, nxtOSEK ha supportato un firmware standard per l'enhanced NXT sviluppato da John Hansen. Il firmware standard per l'enhanced NXT può essere utilizzato sia con i programmi applicativi supportati dal firmware standard NXT sia con le applicazioni native di ARM7. Perciò l'Embedded Coder Robot NXT 3.12 o le versioni successive supportano i firmware standard per l'enhanced NXT oltre all'applicazione Flash, originale di nxtOSEK. L'utente può scegliere uno dei tre approcci per scaricare il codice C generato dall'Embedded Coder Robot NXT sull'NXT stesso. Se l'utente vuole utilizzare l'Embedded Coder Robot NXT con altri linguaggi di programmazione deve installare il firmware standard per l'enhanced NXT e NeXTTool; se vuole utilizzare solo l'Embedded Coder Robot NXT o se vuole scrivere un applicazione di dimensione superiore ai 64kB deve installare LibUsb. LibUsb è una libreria C open source per

l'utilizzo dei dispositivi USB. L'installer per Windows è incluso nel pacchetto dell'nxtOSEK.

nxtOSEK è firmware open source sostitutivo per l'NXT e include l'RTOS OSEK TOPPERS che viene compilato con OSEK ECC2.

2.3.9 ECROBOT NXT BLOCKSET:

Dopo aver installato ECRobot NXT, ECRobot NXT Blockset (Fig. 2.48) appare nella libreria di Simulink.

Figura 2.48: Screenshot dell'ECRobot NXT Blockset

LIGHT SENSOR BLOCKS

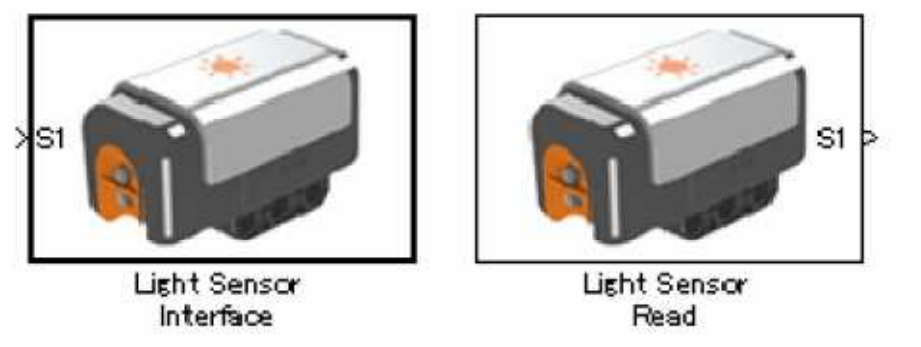


Figura 2.49: Blocchi del sensore di luce

I blocchi dei sensori di luce (Fig 2.49) sono utilizzati per mostrare la luminosità. Valori elevati indicano oscurità o bassa riflessione. Ci sono due blocchi Light Sensor. Light Sensor Interface è un blocco di interfaccia dall'esterno verso il modello. Light Sensor Read è utilizzato per leggere i dati presi dal sensore di luce. Nella simulazione questi blocchi sono dei segnaposto, ma nel codice generato saranno utilizzati per implementare un appropriato dispositivo API. Nella Fig. 2.50 sono riportate le caratteristiche di questo sensore:

Data Type	unt16
Dimension	[1 1]
Data Range	0 to 1023 (raw A/D value)
Port ID	S1/S2/S3/S4

Figura 2.50: Caratteristiche del sensore di luce

TOUCH SENSOR BLOCKS

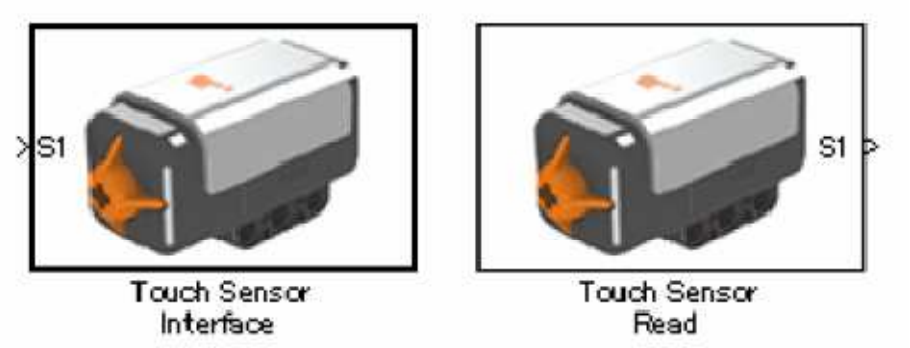


Figura 2.51: Blocchi del sensore di contatto

I blocchi dei sensori di tatto (Fig 2.51) sono utilizzati per rilevare il contatto con un ostacolo. Se il sensore di tatto viene a contatto con un ostacolo, il sensore restituisce il valore 1. Ci sono due blocchi Touch Sensor. Touch Sensor Interface è un blocco di interfaccia dall'esterno verso il modello. Touch Sensor Read è utilizzato per leggere i dati presi dal sensore di tatto. Nella simulazione questi blocchi sono dei segnaposto, ma nel codice generato saranno utilizzati per implementare un appropriato dispositivo API. Nella Fig. 2.52 sono riportate le caratteristiche di questo sensore:

Data Type	unt8
Dimension	[1 1]
Data Range	0(not touched), 1(touched)
Port ID	S1/S2/S3/S4

Figura 2.52: Caratteristiche del sensore di con tatto

SOUND SENSOR BLOCKS

Figura 2.53: Blocchi del sensore di suono

I blocchi dei sensori di suono (Fig 2.53) sono utilizzati per misurare la pressione sonora. Valori più piccoli significano che il suono è alto. Ci sono due blocchi Sound Sensor. Sound Sensor Interface è un blocco di interfaccia dall'esterno verso il modello. Sound Sensor Read è utilizzato per leggere i dati presi dal sensore di suono. Nella simulazione questi blocchi sono dei segnaposto, ma nel codice generato saranno utilizzati per implementare un appropriato dispositivo API. Nella Fig. 2.54 sono riportate le caratteristiche di questo sensore:

Data Type	unt16
Dimension	[1 1]
Data Range	0 to 1023 (raw A/D value)
Port ID	S1/S2/S3/S4

Figura 2.54: Caratteristiche del sensore di suono

REVOLUTION SENSOR BLOCKS

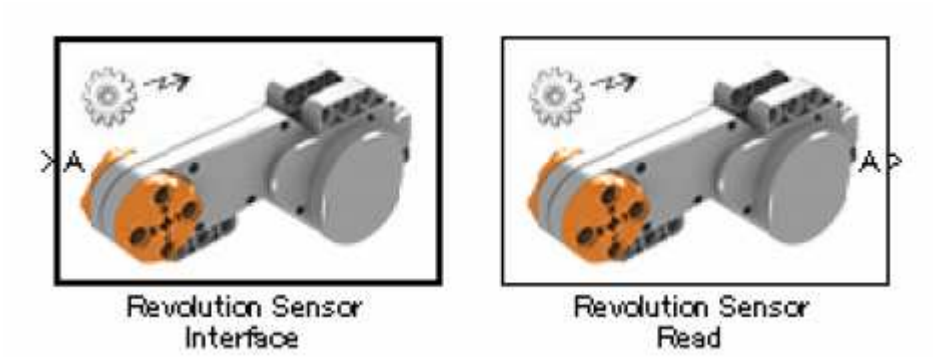


Figura 2.55: Blocchi del sensore di rotazione

I blocchi del sensore di rotazione (Fig 2.55) sono utilizzati per misurare la rotazione del motore. Il sensore di rotazione è formato da due blocchi. Revolution Sensor Interface è un blocco di interfaccia dall'esterno verso il modello. Revolution Sensor Read è utilizzato per leggere i dati presi dal sensore di rotazione. Nella simulazione questi blocchi sono dei segnaposto, ma nel codice

generato saranno utilizzati per implementare un appropriato dispositivo API. Nella Fig. 2.56 sono riportate le caratteristiche di questo sensore:

Data Type	int32
Dimension	[1 1]
Data Range	Range of int32 [deg]
Port ID	A/B/C

. *Figura 2.56: Caratteristiche del sensore di rotazione*

SERVO MOTOR BLOCKS

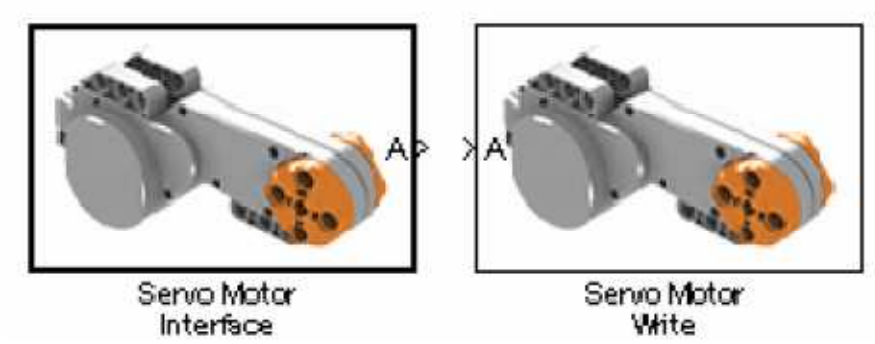


Figura 2.57: Blocchi del servo motore

I blocchi dei servo motori (Fig 2.57) sono utilizzati per controllare i motori. Ci sono due blocchi Servo Motor. Servo Motor Interface è un blocco di interfaccia dall'esterno verso il modello. Servo Motor Write è utilizzato per stabilire i valori necessari per controllare il motore. Nella simulazione questi blocchi sono dei segnaposto, ma nel codice generato saranno utilizzati per implementare un appropriato dispositivo API. Nella Fig. 2.58 sono riportate le caratteristiche del servo motore:

Data Type	int8
Dimension	[1 1]
Data Range	-100 to 100
Port ID	A/B/C
Mode	Brake/Float

Figura 2.58: Caratteristiche del servo motore

BATTERY VOLTAGE BLOCKS

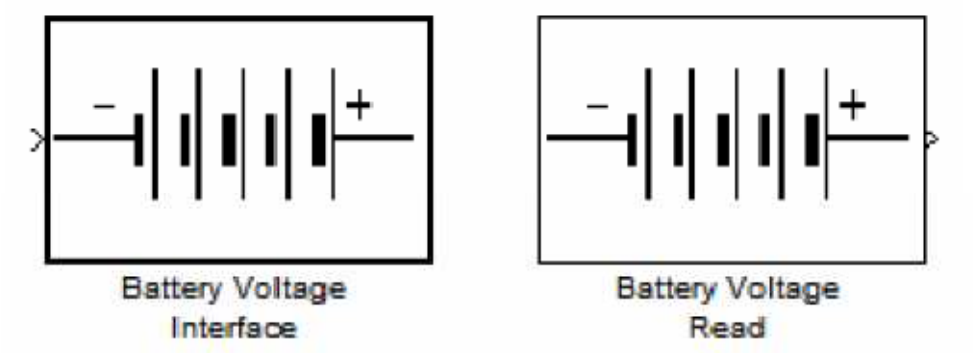


Figura 2.59: Blocchi della batteria

I blocchi Battery Voltage (Fig 2.59) sono utilizzati per rappresentare la tensione dell'NXT. Ci sono due blocchi Battery Voltage. Battery Voltage Interface è un blocco di interfaccia dall'esterno verso il modello. Battery Voltage Read è utilizzato per leggere il voltaggio della batteria. Nella simulazione questi blocchi sono dei segnaposto, ma nel codice generato saranno utilizzati per implementare un appropriato dispositivo API. Nella Fig. 2.60 sono riportate le caratteristiche della batteria:

Data Type	unt16
Dimension	[1 1]
Data Range	0 to possible NXT maximum voltage value in mv (i.e. 9000 = 9.000 V)
Port ID	None

Figura 2.60: Caratteristiche della batteria

SYSTEM CLOCK BLOCKS

Figura 2.61: Blocchi del clock

I blocchi del Clock (Fig 2.61) sono utilizzati per leggere il tick del sistema dell' NXT. Ci sono due blocchi di Clock. Servo Clock Interface non ha né input né output e fornisce, durante la simulazione, un ticchettio per il System Clock Interface. Il blocco System Clock Read è usato per leggere le informazioni del ticchettio di sistema. Questi blocchi saranno utilizzati per implementare un appropriato dispositivo API nel codice generato. Nella realtà, il ticchettio di sistema parte da 0 quando l'NXT viene acceso. (Non parte quando l'applicazione ECRobot viene avviata). Nella Fig. 2.62 sono riportate le caratteristiche del clock:

Data Type	unt32
Dimension	[1 1]
Data Range	0 to maximum of uint32 in mille-second
Port ID	None

Figura 2.62: Caratteristiche del clock

SOUND VOLUME TONE BLOCKS

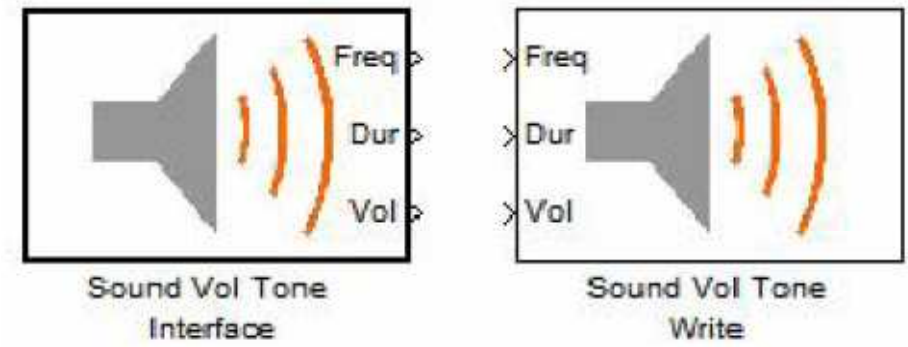


Figura 2.63: Blocchi del Sound Volume Tone

I blocchi Sound Volume Tone (Fig 2.63) sono utilizzati per generare un suono che è determinato dalla frequenza e dalla durata. Nella simulazione questi blocchi sono dei segnaposto, ma nel codice generato saranno utilizzati per implementare un appropriato dispositivo API. Nella Fig. 2.64 sono riportate le caratteristiche del Sound Volume Tone:

Data Type	Freq: unt32, Dur:uint32
Dimension	[1 1]
Data Range	Freq: 31 [Hz] to 14080 [Hz], Dur: 10 to 2560 [msec], Vol: 0 to 100 (0 is mute)
Port ID	None

Figura 2.64: Caratteristiche del Sound Volume Tone

SOUND WAV BLOCKS

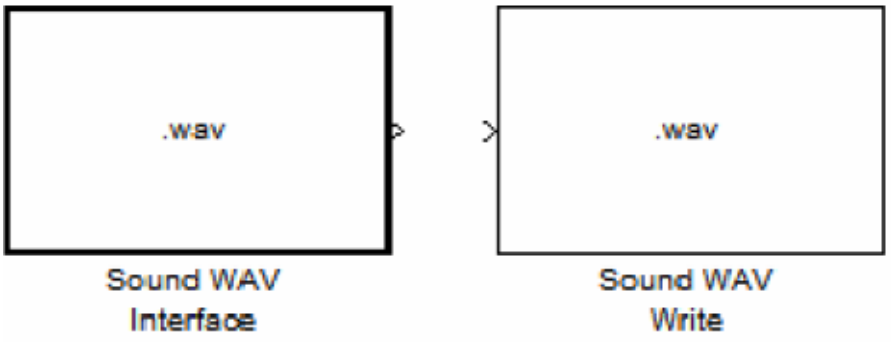


Figura 2.65: Blocchi del Sound WAV

I blocchi Sound WAV Tone (Fig 2.65) sono utilizzati per generare un suono che è salvato come un file WAV. Nella simulazione questi blocchi sono dei segnaposto, ma nel codice generato saranno utilizzati per implementare un appropriato dispositivo API. Nella Fig. 2.66 sono riportate le caratteristiche del Sound WAV:

Data Type	Freq: unt32, Dur:uint32
Dimension	[1 1]
Data Range	-
Port ID	None

Figura 2.66: Caratteristiche del Sound WAV

ULTRASONIC SENSOR READ

Figura 2.67: Blocchi del sensore ad ultrasuoni

Il blocco del Sensore ad ultrasuoni (Fig 2.67) è utilizzato per misurare la distanza da un ostacolo o per rilevare un ostacolo prima che vi sia il contatto. Ci sono due blocchi del sensore ad ultrasuoni: Ultrasonic Sensore Interface è un blocco di interfaccia dall'esterno verso il modello. Il blocco Ultrasonic Sensor Read è utilizzato per leggere i valori provenienti dal sensore. Nella simulazione questi blocchi sono dei segnaposto, ma nel codice generato saranno utilizzati per implementare un appropriato dispositivo API. Nella Fig. 2.68 sono riportate le caratteristiche del sensore ad ultrasuoni:

Data Type	int32
Dimension	[1 1]
Data Range	0 to 255 [cm], -1 (the sensor is not ready for measurement)
Port ID	S1/S2/S3/S4

Figura 2.68: Caratteristiche del sensore ad ultrasuoni

BLUETOOTH RX BLOCKS

Figura 2.69: Blocchi del Bluetooth Rx

I blocchi Bluetooth Rx (Fig 2.69) rappresentano la comunicazione Bluetooth dal PC all'NXT. Ci sono due blocchi Bluetooth Rx. Bluetooth Rx Interface è un blocco di interfaccia dall'esterno verso il modello. Bluetooth Rx Read è utilizzato per leggere i dati. Solo un blocco Bluetooth Rx può essere allocato in un modello. Nella simulazione questi blocchi sono dei segnaposto, ma nel codice generato saranno utilizzati per implementare un appropriato dispositivo API. La connessione Bluetooth è supportata solo dal PC all'NXT e non dall' NXT all'NXT. Nella Fig. 2.70 sono riportate le caratteristiche del Bluetooth Rx:

Data Type	uint8
Dimension	32
Data Range	0 to 255
Port ID	None

Figura 2.70: Caratteristiche del Bluetooth Rx

BLUETOOTH TX BLOCKS

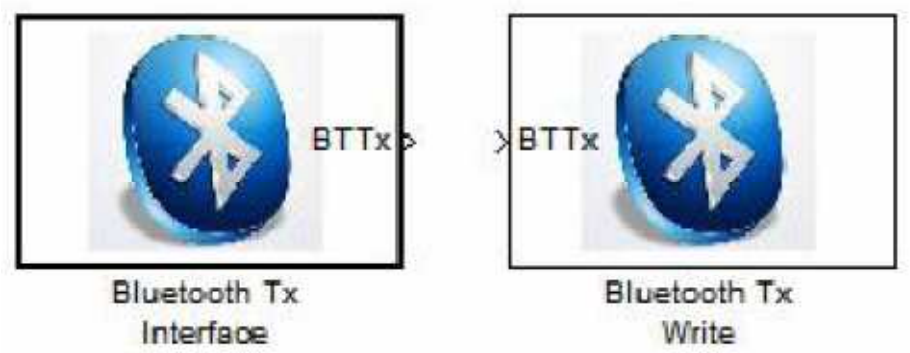


Figura 2.71: Blocchi del Bluetooth Tx

I blocchi Bluetooth Tx (Fig 2.71) rappresentano la comunicazione Bluetooth dall'NXT al PC. Ci sono due blocchi Bluetooth Tx. Bluetooth Tx Interface è un blocco di interfaccia dall'esterno verso il modello. Bluetooth Tx Write è utilizzato per inviare dati. Solo un blocco Bluetooth Tx può essere allocato in un modello. Nella simulazione questi blocchi sono dei segnaposto, ma nel codice generato saranno utilizzati per implementare un appropriato dispositivo API. La connessione Bluetooth è supportata sia dal PC all' NXT che da un NXT ad un altro NXT. Nella Fig. 2.72 sono riportate le caratteristiche del Bluetooth Tx:

Data Type	uint8
Dimension	32
Data Range	0 to 255
Port ID	None

Figura 2.72: Caratteristiche del Bluetooth Tx

NXT GAMEPAD DATA LOGGER BLOCK

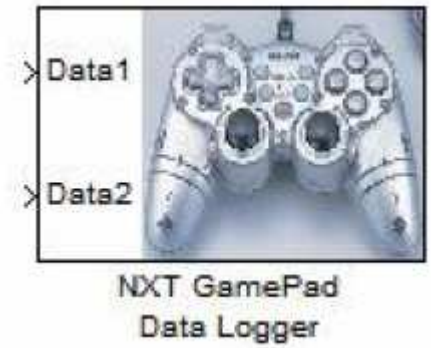


Figura 2.73: Blocco del Data Logger

nxtOSEK ha fornito un tool chiamato NXT GamePad (Fig 2.73). Questo abilita gli utenti a controllare da remoto un NXT (che possiede nxtOSEK) via Bluethoot usando un gamepad analogico. Dalla versione V1.01, l'utente può anche scegliere tra 'Analog Stick Control' and/or 'NXT Data Acquisition' in base all'obbiettivo. L' acquisizione di dati dall' NXT può essere effettuata anche se l'utente non possiede un gamepad. Questo blocco non interferisce con i risultati della simulazione, ma nel codice generato sarà utilizzato per implementare un appropriato dispositivo API.

Data1 e Data2 possono essere utilizzati per prendere i dati dal GamePad analogico. Questi dati possono essere poi salvati in file CVS e ciò è utile per analizzare il comportamento dell'applicazione NXT in MATLAB. NXT GamePad Data Logger block non deve essere utilizzato con il blocco Bluetooth Tx Write o con il blocco NXT GamePad ADC Data Logger all'interno dello stesso modello. Nella Fig. 2.74 sono riportate le caratteristiche del Data Logger:

Data Type	int8
Dimension	[1 1]
Data Range	-128 to 127
Port ID	None

Figura 2.74: Caratteristiche del Data Logger

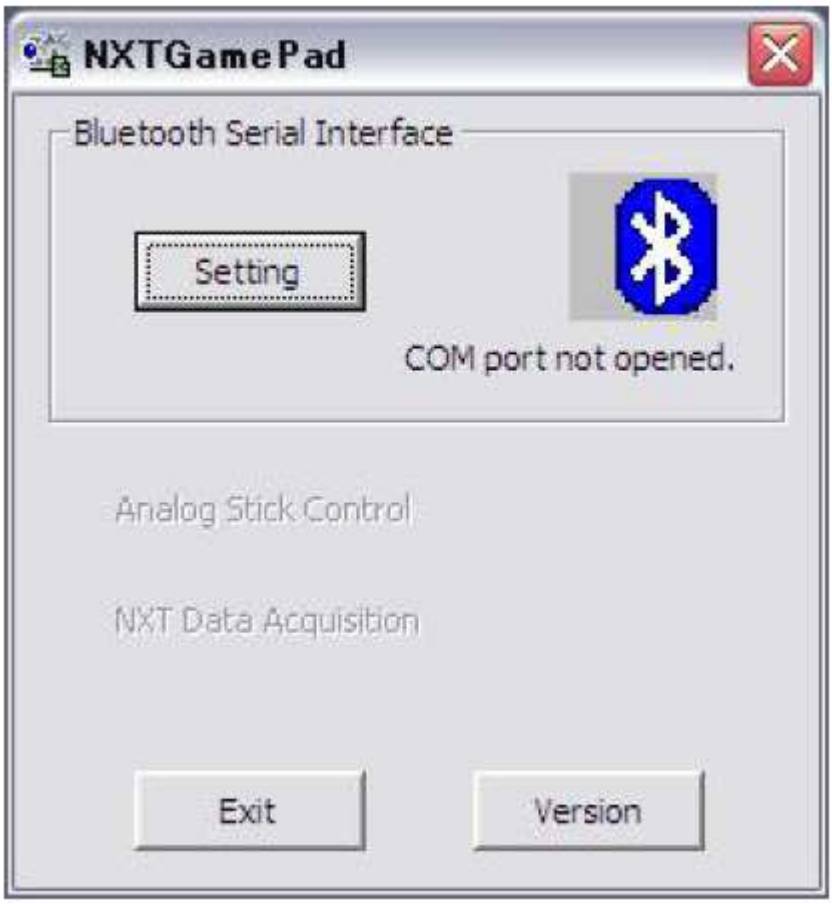


Figura 2.75: Finestra per l'acquisizione di dati tramite Bluetooth

NXT GAMEPAD ADC DATA LOGGER BLOCK

Figura 2.76: Blocco ADC Data Logger

NXT GamePad ADC Data Logger (Fig. 2.76) ha una funzione simile a quella dell'NXT GamePad Data Logger, ma permette all'utente di configurare quattro segnali che possono essere acquisiti e studiati. Il blocco NXT GamePad ADC Data Logger non può essere utilizzato, all'interno di un modello, insieme ai blocchi Bluetooth Tx Write e NXT GamePad Data Logger. Questo blocco richiede la presenza della piattaforma nxtOSEK nel blocco Exported Function-Calls Scheduler per mostrare sullo schermo i valori ADC selezionati dall'utente. Nella Fig. 2.77 sono riportate le caratteristiche dell' ADC Data Logger:

Input name	Data type	Data range	Dimension
Data1/Data2	int8	-128 to 127	1
ADC1/ADC2/ADC3/ADC4	int16	-32768 to 32767	1

Figura 2.77: Caratteristiche del Data Logger

GYRO SENSOR BLOCKS

Figura 2.78: Blocchi del sensore giroscopio

Il blocco del Gyro Sensor (Fig. 2.78) è utilizzato per misurare il numero di gradi al secondo di rotazione. Questo non è un sensore LEGO ma viene prodotto dall' HiTechnic. Gyro Sensor è composto da due blocchi. Il blocco Gyro Sensor Interface è un blocco di interfaccia dall'esterno verso il modello. Il blocco Gyro Sensor Read è utilizzato per leggere i valori provenienti dal sensore. Nella simulazione questi blocchi sono dei segnaposto, ma nel codice generato saranno utilizzati per implementare un appropriato dispositivo API. Nella Fig. 2.79 sono riportate le caratteristiche di questo sensore:

Data Type	unt16
Dimension	[1 1]
Data Range	0 to 1023 (raw A/D value)
Port ID	S1/S2/S3/S4

Figura 2.79: Caratteristiche del giroscopio

ACCELERATION SENSOR BLOCKS

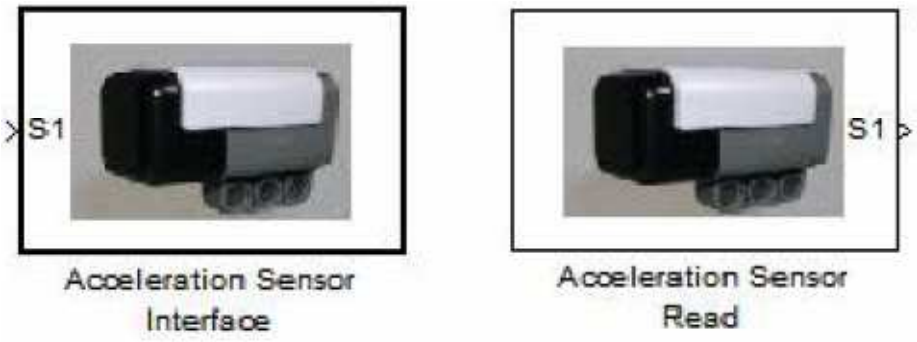


Figura 2.80: Blocchi dell'accelerometro

I blocchi Acceleration Sensor (Fig. 2.80) sono utilizzati per misurare l'accelerazione sui tre assi. Acceleration Sensor non è un prodotto standard della LEGO, ma è un prodotto dell'HiTechnic. Ci sono due blocchi Acceleration Sensor. Acceleration Sensor Interface è un blocco di interfaccia dall'esterno verso il modello. Acceleration Sensor Read è utilizzato per leggere i dati del sensore di accelerazione. Nella simulazione questi blocchi sono dei segnaposto, ma nel codice generato saranno utilizzati per implementare un appropriato dispositivo API. Nella Fig. 2.81 sono riportate le caratteristiche di questo sensore:

Data Type	int16
Dimension	[1 3] (index 1: x axis, index 2: y axis, index 3: z axis)
Data Range	-512 to 511
Port ID	S1/S2/S3/S4

Figura 2.81: Caratteristiche dell'accelerometro

EXPORTED FUNCTION-CALLS SCHEDULER BLOCK

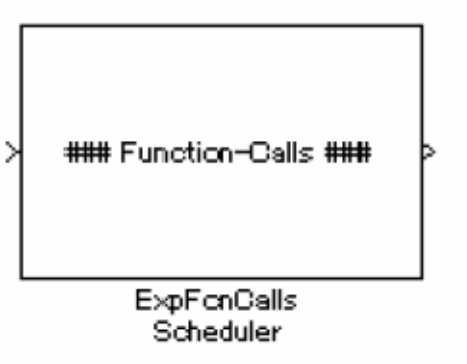


Figura 2.82: Blocco Exported Function-Calls Scheduler

Di solito è richiesto un software, per il controllo del sistema embedded, che permetta di eseguire strategie di controllo diverse in momenti differenti (per esempio durante l'inizializzazione) e con modalità differenti (asincrona, periodica). Exported Function-Calls Subsystem (Fig. 2.82) controlla, durante la simulazione, il tempo di esecuzione del Function-Call Subsystems conformemente ai parametri specificati nella finestra di dialogo Block Parameters (Fig.2.83). Queste informazioni sui parametri saranno utilizzate per generare il file ecrobot_main.c.

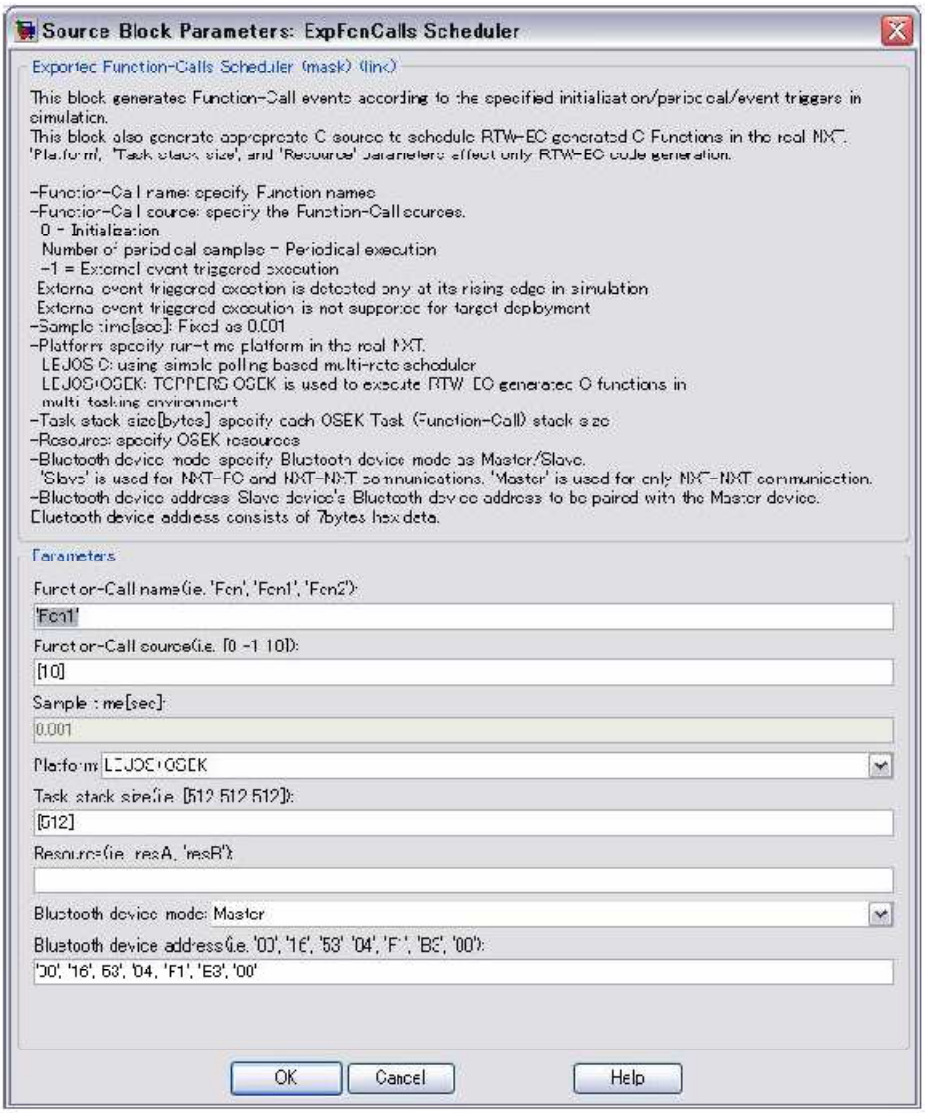


Figura 2.83: Configurazione dei parametri dell' Exported Function-Calls Scheduler

OSEK RESOURCE BLOCK:

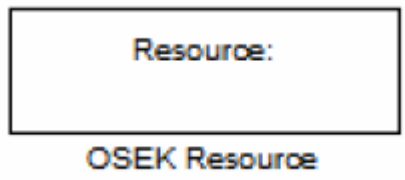


Figura 2.84: Blocco Osek Resource

Il blocco OSEK Resource (Fig. 2.84) è utilizzato per implementare l'OSEK GetResource/ReleaseResource API all' inizio/fine della sezione critica. In Simulink, la sezione critica deve essere racchiusa in un Atomic Subsystem e un blocco OSEK Resource deve essere allocato dentro un Atomic Subsystem con una Resource specificata. Per specificare un Resource identifier nel blocco, esso deve essere definito nel blocco Exported Function-Call Scheduler. Il blocco OSEK Resource non interferisce con i risultati della simulazione ed è solo valido per la generazione del codice.

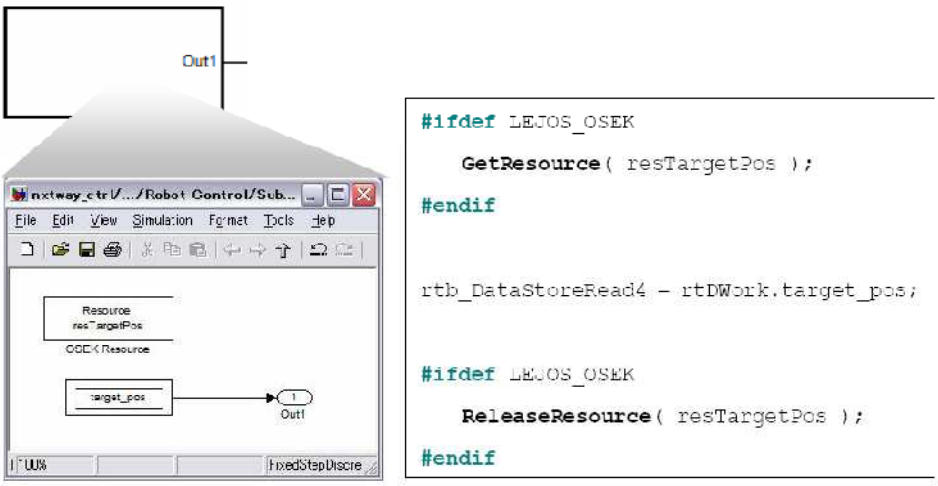


Figura 2.85: Esempio di utilizzo del blocco Osek Resource e codice generato

XY MAP GRAPH BLOCK

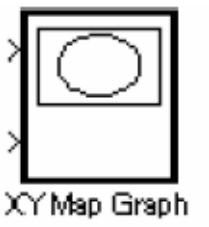


Figura 2.86: Blocco XY Map Graph

Il blocco XY Map Graph (Fig. 2.86) è situato nelle Utilities per la simulazione ed è utilizzato per vedere, durante la simulazione, il percorso di un robot su una

traccia virtuale. XY Map Graph è basato su un blocco Simulink standard che è chiamato XY Graph.

2.3.9 ECROBOT NXT MODELING GUIDELINES

Negli ultimi due anni sono state aggiunte molte caratteristiche nuove a Simulink, Stateflow, Real-Time Workshop e Real Time Workshop Embedded Coder. Questi tools sono i prodotti principali del Model Based Design. Gli NXTBlockset e i modelli demo di ECRobot sono stati progettati seguendo delle linee guida, basate sulle nuove caratteristiche dei prodotti MathWorks.

SEPARAZIONE DELL'IMPIANTO E DEL CONTROLLO ATTRAVERSO IL MODEL REFERENCE:

ECRobot NXT demo utilizza le caratteristiche del Model Reference per dividere il controllore e l'impianto in file di Simulink diversi (Fig. 2.87). Questo approccio permette uno sviluppo parallelo dei processi per il progetto del controllore e dell'impianto. Simulink Buses e Bus Objects sono utilizzati per avere un'interfaccia tra il modello del controllore e quello dell'impianto.

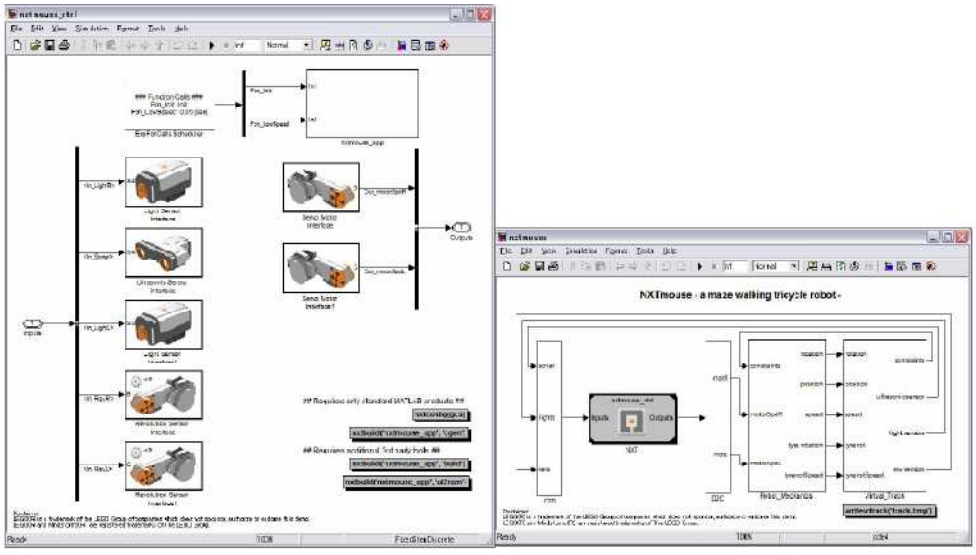


Figura 2.87: Screenshot del modello del controllore e dell'impianto

AMBIENTE DEL SOFTWARE E ARCHITETTURA DELL'APPLICAZIONE NEL MODELLO DEL CONTROLLORE:

Nell'industria del software la separazione dell'ambiente del software (per esempio RTOS) e del software applicativo in obiettivi indipendenti rappresenta una tecnica per raggiungere la riusabilità dei componenti del software e per supportare lo sviluppo su larga scala, mediante il lavoro in team. Questa tecnica può anche essere utilizzata per modellare il controllore con Simulink. Nel modello del controllore fatto con ECRobot NXT, l'ambiente software (scheduler) e il sottosistema dell'applicazione sono separati. In più le interfacce esterne (sensori dell'NXT, motori e dispsitivi di comunicazione) sono rappresentati da specifici blocchi d'interfaccia. All'interno di un sottosistema dell'applicazione ci sono Function-Call Subsystems che vengono eseguiti dal blocco Exported Function-Calls Scheduler. L'architettura del modello di un controllore progettato con ECRobot NXT segue l'architettura di un software reale embedded e per questo il modello del controllore è così tanto leggibile che viene utilizato come specifica eseguibile.

L'EXPORTED FUNCTION-CALLS SCHEDULER CONTROLLA IL TEMPO DI ESECUZIONE DELLE APPLICAZIONI DEI SUBSYSTEMS:

Di solito l'esecuzione di un Function-Call Subsystem è controllata da un Function-Call Generator o da uno Stateflow Event. Stateflow permette all'utente di progettare un complesso sistema dotato di scheduler. In ECRobot NXT, un approccio alternativo consiste nel controllare il tempo di esecuzione utilizzando un Function-Call Subsystem. Il blocco Exported Function-Calls Scheduler (Fig. 2.88) è un C MEX S-Function che fornisce una finestra di dialogo, il Block Parameters (Fig. 2.89), per specificare le informazioni necessarie per controllare il tempo d'esecuzione del Function-Call Subsystems.

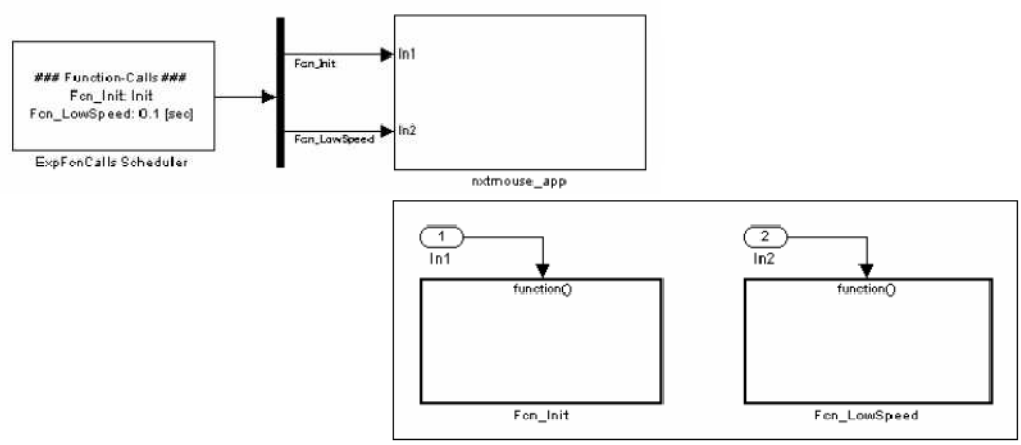


Figura 2.88: Blocco dell'Exported Function-Calls Scheduler e Function-Call Subsystems

Dalla versione R2006a, Simulink e Real-Time Workshop Embedded Coder hanno fornito una funzione "Exporting Function-Call Subsystem code generation" che permette di generare, dal Function-Call Subsystems, funzioni C void-void molto efficienti.

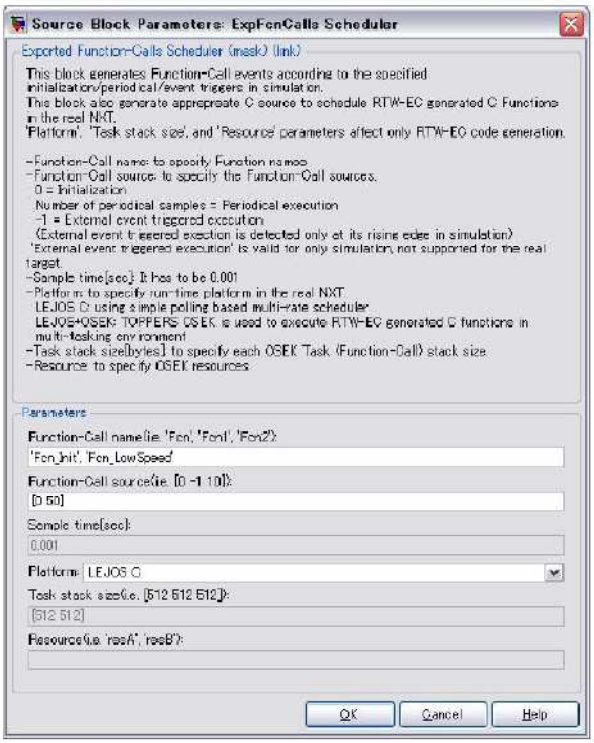


Figura 2.89: Configurazione dei parametri dell' Exported Function-Calls Scheduler

COMUNICAZIONE TRA FUNCTION-CALL SUBSYSTEMS ATTRAVERSO I BLOCCHI DATA STORE READ/WRITE:

Di solito in un modello Simulink l'ordine di esecuzione di ciascun blocco è determinata dal suo contesto. Questo non accade nel Function-Call Subsystem, in cui l'esecuzione è sempre controllata da colui che la chiama (ad es. uno Stateflow Event). Alcune volte queste caratteristiche causano delle difficoltà per il trasferimento di dati tra Function-Call Subsystems. In ECRobot NXT demo, la comunicazione tra Function-Call Subsystems deve essere fatta attravero i blocchi Data Store Read/Write. I blocchi dei dispositivi ECRobot come ad es. Light Sensor Interface/Light Sensor Read sono mascherati dai blocchi Data Store Read/Write e gli identificatori di blocco sono determinati dai Signal Objects che l'utente ha determinato durante l'aggiornamento del modello. L'utilizzo dei blocchi Data Store Read/Write risolve potenziali problemi che si potrebbero verificare nella comunicazione tra Function-Call Subsystems. Tuttavia non bisogna dimenticare che quando si utilizzano i blocchi Data Store Read/Write si elimina il passaggio dei segnali tra i blocchi e Simulink può determinare un ordine di esecuzione inaspettato per i blocchi. Per questo motivo l'utente deve specificare in modo esplicito la priorità d'esecuzione per ogni blocco Data Store Read/Write nella finestra di dialogo delle proprietà del blocco.

NXTCONFIG API PER IMPOSTARE I PARAMETRI:

nxtconfig API (Fig. 2.90) è una funzione MATLAB e che imposta tutti i parametri necessari per il modello del controllore con ECRobot. Questo tipo di automatizzazione è un fattore molto importante per i processi Model-Based Design su larga scala. Da quando è stata rilasciata per la prima volta la demo di ECRobot NXT, numerosi feedbacks sono stati ricevuti. La maggior parte degli utenti appartengono all'ambito accademico e molti di loro hanno espresso il desiderio di utilizzare una maggior varietà di tipi di blocchi e di dati in virgola mobile.

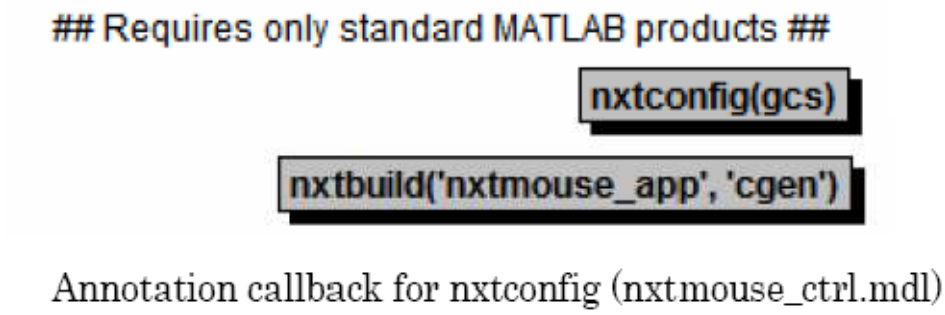


Figura 2.90: Interfaccia per generare il codice

Le loro esigenze sono state accolte e per questo ECRobot NXT V3.03 supporta non solo il codice intero e quello in virgola fissa, ma permette anche di generare codice in virgola mobile (sia single che double). In nxtconfig API il supporto per i numeri in virgola fissa è nel pannello delle opzioni di Real-Time Workshop (Fig. 2.91), perciò l'utente deve aprirlo manualmente quando vuol generare codice in virgola fissa.

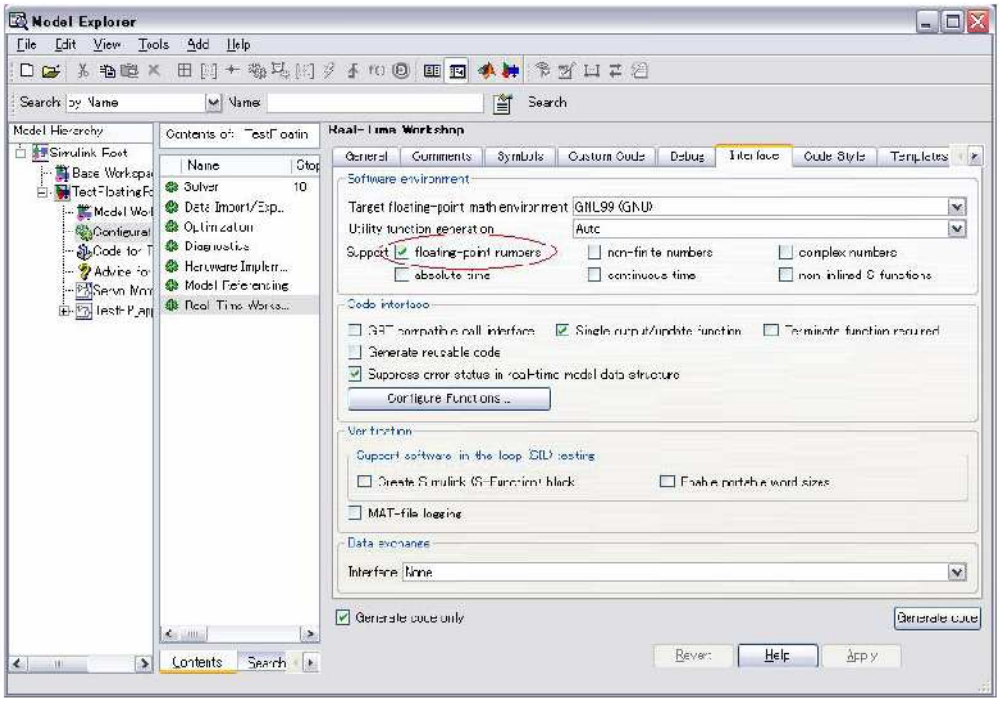


Figura 2.91: Screenshot di Model Explorer per il settaggio delle impostazioni dei numeri a virgola mobile

ATMEL AT91SAM7S256 (processore ARM7) nell'NXT non ha Floating Point Unit e per questo motivo alcuni blocchi di funzioni matematiche richiedono l'utilizzo, nel codice generato, di una libreria matematica in C. Per questo il codice con variabili in virgola mobile occupa una maggior memoria e le sue prestazioni sono peggiori del codice scritto con variabili intere e in virgola fissa.

AMBIENTE PER L'ESECUZIONE DI ECROBOT NXT:

La Fig. 2.92 mostra una visione grafica dell'architettura del LEGO Mindstorm NXT secondo gli sviluppatori del Kit.

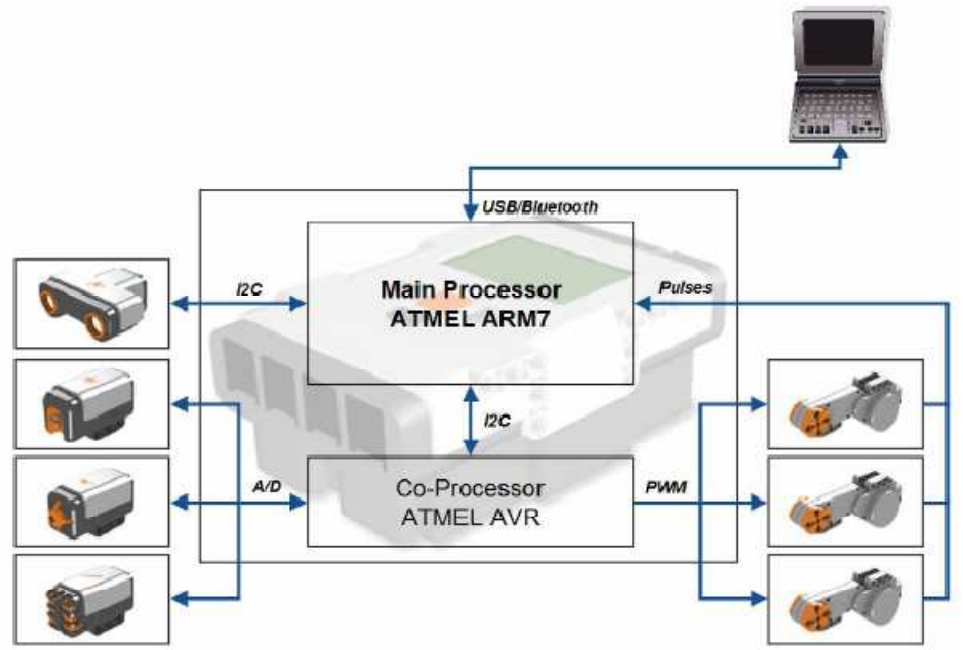


Figura 2.92: Architettura del sistema LEGO Mindstorm NXT

Si può notare chiaramente che la comunicazione tra il processore principale ARM7 (ATMEL AT191SAM7S256) e i sensori è realizzata tramite un co-processore (ATMEL AVR). Le uniche eccezioni sono rappresentate dall'Utrasonic Sensor e dall'acquisizione dei dati dal Servo Motor. Per le applicazioni di ECRobot NXT il fattore più importante per l'accesso ai sensori è la comunicazione con il coprocessore tramite il bus seriale I2C. Questa architettura influenza l'ambiente di esecuzione dell'ECRobot NXT. Il processore principale ARM7 accede ai sensori indipendentemente ogni 2 millisecondi con un Interrupt Service Routine del firmware di LEJOS NXJ che ha un periodo di 1 milli

secondo. I dati provenienti dal servo motore sono catturati direttamente dall'impulso innescato dall'Interrupt Service Routine del firmware Lejos NXJ nel processore principale ARM7. Il sensore ad ultrasuoni comunica con ARM7 attraverso un altro canale I2C.

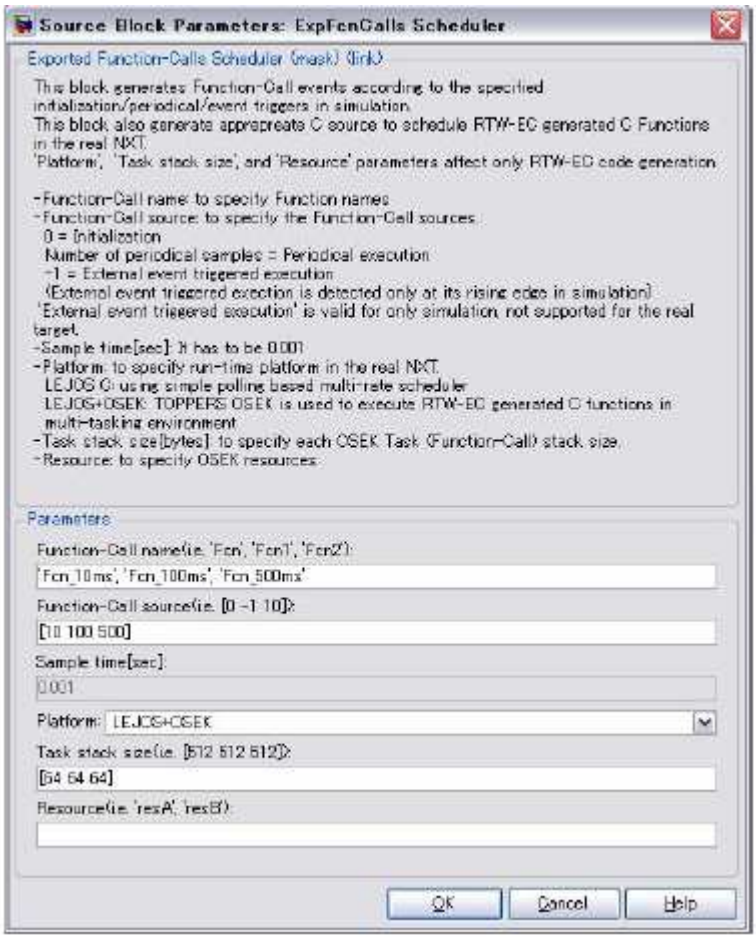


Figura 2.93: Gestione dei parametri dell'Exported Function-Calls Scheduler

Nel file ecrobot_main.c ci sono le definizioni degli OSEK Tasks e le Function-Calls specificate sono mappate nelle OSEK Task (Fig. 2.93). Il tempo di esecuzione di ogni OSEK Task è configurato nei file OSEK OIL (ecrobot.OIL).

ECROBOT INTERFACE API:

Nella directory lejos_osek/ecrobot, ci sono i file sorgente in C dell'interfaccia ECRobot NXT interface (ecrobot_interface.c, ecrobot_interface.h) . Questi file sorgente in C includono un dispositivo APIs che ha accesso ai driver LEJOS NXJ

I/O. Alla maggior parte delle interfacce APIs di ECRobot pubblicate corrispondono in ECRobot i blocchi dei dispositivi dell'NXT.

Per esempio, la funzione C Fcn_10ms è generata da Fcn_10ms Function-Call Subsystem in TestMotor.mdl. In Fcn_10ms Function-Call Subsystem (Fig. 2.94) c'è un blocco Servo Motor Write e il corrispondente dispositivo API pubblicato è implementato nella funzione C Fcn_10ms. Questo dispositivo API è implementato utilizzando un Simulink Signal Object creato dall'utente, che è chiamato NXT.Signal.

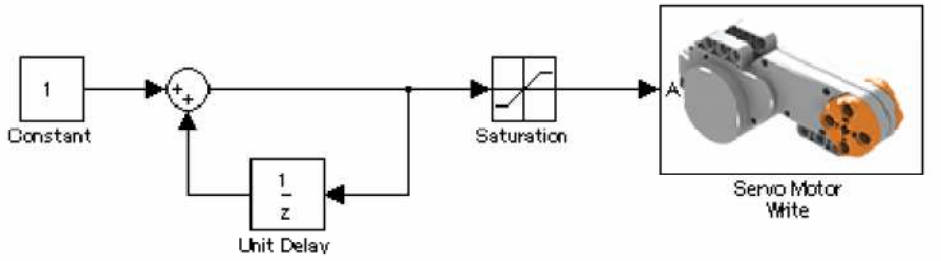


Figura 2.94: Schema a blocchi di Function-Call Subsystems

MOTIVI PER CUI IL ROBOT NON SI MUOVE IN MODO FLUIDO:

Quando viene provato il Model-Based Design Experience in ECRobot demo, si possono incontrare delle difficoltà inaspettate nel controllo dei robot (per es. il robot non si muove su una traiettoria rettilinea, i risultati della simulazione sono diversi per ogni simulazione). Questo comportamento è stato intenzionalmente progettato nel modello dell'impianto del robot. In generale, nei progetti di controllo viene linearizzato l'impianto del sistema, anche se nel mondo reale, qualunque sistema di controllo, seppur semplice, contiene fattori non-lineari che possono causare problemi per il controllo del sistema nel mondo reale. Per esempio, i servo motori della LEGO Mindstorms NXT utilizzano molti meccanismi per realizzare un'elevata forza rotatoria. Questi meccanismi non sono prodotti e assemblati con precisione e ciò emerge in alcune relazioni meccaniche. All'interno del Servo Motore (Fig. 2.95) c'è un blocco che simula queste reazioni meccaniche, permettendo di rilevare anticipatamente eventuali problemi applicativi.

Inoltre nel modello della demo del Robot_Mechanics Subsystem, il controllo effettuato dal servo motore sulla ruota destra e su quella sinistra ha un errore casuale in ogni simulazione. Perciò il robot non si può muovere in modo rettilineo. Questo risultato è molto frequente nell'hardware reale ed è dovuto alle superfici di appoggio, agli errori meccanici del motore, delle ruote e dei meccanismi.

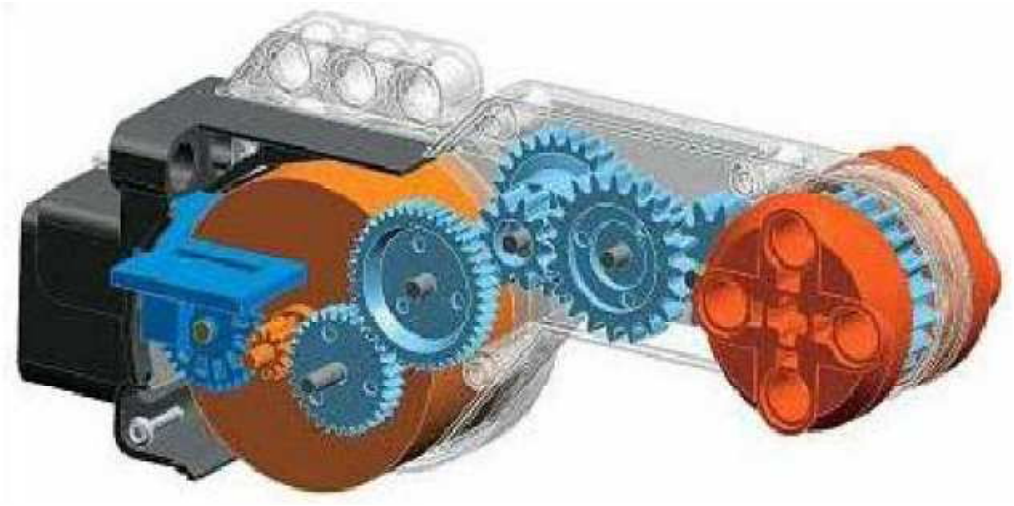


Figura 2.95: Interno del servo motore dell'NXT

COME CARICARE UN'APPLICAZIONE ECROBOT NXT SULLA MEMORIA FLASH DEL BRICK:

ECRobot NXT supporta anche l'upload di programmi sulla memoria FLASH del AT91SAM7S256 dell'NXT. L'utilizzo di questa modalità provoca numerosi problemi, il principale è dovuto alla limitatezza della vita della memoria FLASH che non permette di caricare molti programmi su di essa.

COME PROGETTARE LEGO MINDSTORMS NXT 3-D:

Nell' ambito della LEGO e di Mindstorms, sono stati sviluppati ed esistono numerosi tools e progetti interessanti. Molti di questi sono creati da volontari, appassionati o da progetti open-source.

Capitolo 3

3 ESEMPIO APPLICATIVO NXTWAY

SEGWAY

Nel dicembre del 2001 venne presentato al mondo un curioso prototipo chiamato Ginger (poi in seguito rinominato Segway). Il padre di questa invenzione è il geniale Dean Kamen, scienziato americano nato il 5 aprile 1951. La sua invenzione rappresenta un innovativo mezzo di locomozione individuale dotato di due ruote parallele. È una sorta di monopattino elettrico intelligente che parte, si ferma, fa retromarcia con semplici movimenti del corpo del passeggero-guidatore e gira con l'ausilio di una manopola sul lato sinistro del manubrio. Il segway (Fig. 3.1) per muoversi utilizza il peso della persona trasportata, anticipando le mosse di chi ci sta sopra e riportando il baricentro del corpo in asse attraverso movimenti in avanti ed all'indietro. Il suo funzionamento è quello di "un' estensione del corpo: come un partner in un ballo è capace di anticipare ogni mossa". Per farlo utilizza sensori di rotazione (giroscopi) allo stato solido (MEMS) che sono "capaci di imitare l'equilibrio umano".

Questo mezzo ha avuto un buon impatto mediatico sia per la sua particolarità sia perché è molto ecologico. Essendo di facile utilizzo è stato adottato, in molti stati, anche dalle forze dell'ordine in quanto consente una veloce mobilità, una migliore visibilità e quindi anche una migliore interazione con le persone.

In questa tesi sono stati presi in considerazioni tre modelli di NXTway (Segway costruito utilizzando componenti LEGO). Nel primo (NXTway_LS) è presente un sensore di luce , nel secondo sono presenti due sensori di luce (NXTway_LSS) e nel terzo è presente un sensore giroscopico (NXTWAY_GS).

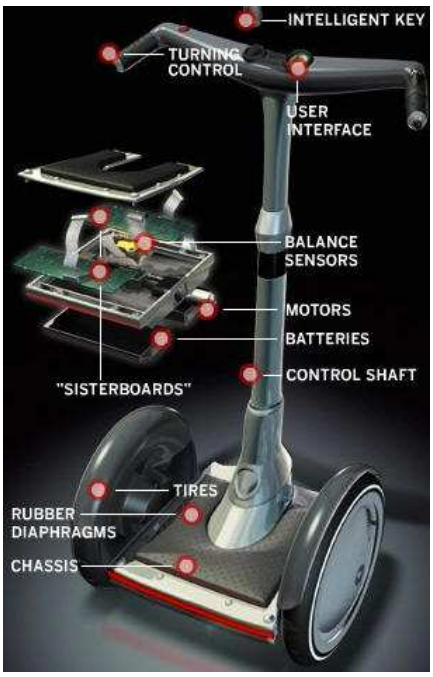


Figura 3.1: segway; legway con un sensore di luce

Caratteristiche principali dell' NXTway:

- Il controllo, ciò che fa in modo che il robot non caschi, è effettuato tramite due motori DC e un giroscopio (o dei sensori di luce)
- Esistono due modalità: controllo autonomo o remoto mediante lo stick analogico del gamepad.

In questo lavoro di tesi presentiamo i principi del Model-Based Design, applicati al controllo dell'equilibrio e del movimento dell'NXT, utilizzando MATLAB/Simulink. I principali punti che verranno affrontati sono:

- Il modello matematico
- Il progetto del controllore
- L'illustrazione del modello
- La simulazione e i risultati sperimentali

3.1 PROGETTO

STRUTTURA:

Il corpo centrale del NXTway è costituito dal brick ed intorno a questo si sviluppa la struttura del Legway creato con pezzi LEGO. Per garantire le migliori prestazioni possibili si deve minimizzare la quantità di mattoncini e di componenti utilizzati. Infatti questi aumentano l'aleatorietà e l'instabilità del progetto. Sui lati del brick vengono montati i due motori DC sotto i quali vengono fissate le ruote in modo che abbiano l'asse di rotazione in comune. Ciò che dà al robot un aspetto antropomorfo è la particolare forma del sensore ad ultrasuoni, posto in alto come fosse una testa con due occhi arancioni. In un primo momento il controllo è stato effettuato solo tramite un sensore giroscopico, posto dietro il sensore ad ultrasuoni (Fig. 3.2). Questo ha dato maggiori benefici: il Gyro Sensor non è sensibile alla luce ed al coefficiente di riflessione della superficie d'appoggio (difetti riscontrati con il sensore di luce). Le principali caratteristiche dei sensori sono espresse in Fig. 3.3.

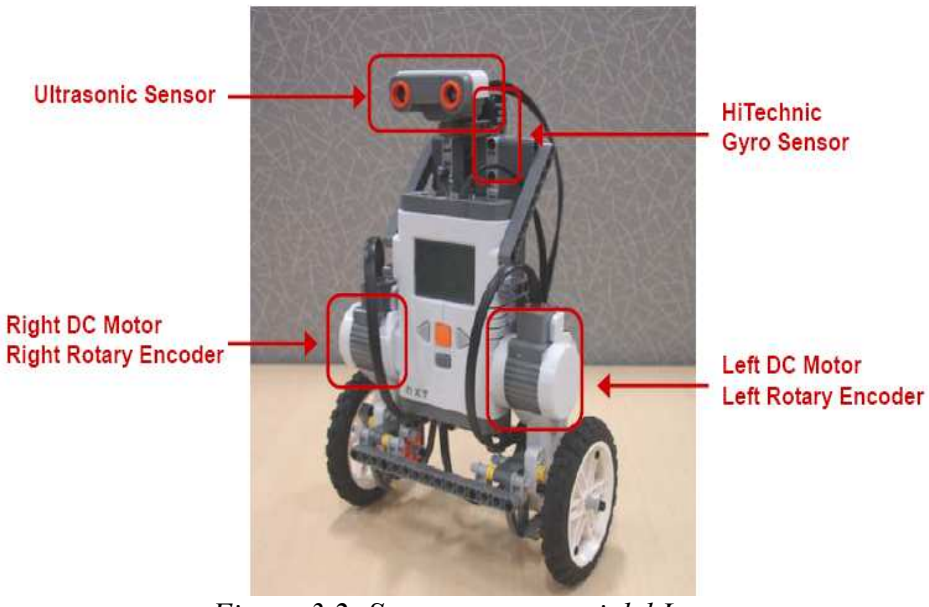


Figura 3.2: Struttura e sensori del Legway

In un secondo momento è stato aggiunto un sensore di luce posto dietro il brick ed attaccato al corpo centrale tramite una barretta orizzontale e una giuntura composta da pezzi LEGO.

Sensor	Output	Unit	Data Type	Maximum Sample [1/sec]
Rotary Encoder	angle	deg	int32	1000
Ultrasonic Sensor	distance	cm	int32	50 (N1)
Gyro Sensor	angular speed	deg/sec	uint16	300

Figura 3.3: Tabella che riassume le caratteristiche dei sensori

Lo schema a blocchi di Simulink è stato modificato e poi ricaricato sull' NXT per garantire un controllo tramite uno o due sensori di luce. In condizioni favorevoli, cioè in una stanza poco illuminata, il robot si muove in modo soddisfacente, e riesce a rimanere in equilibrio, come avviene con il giroscopio.

I problemi maggiori sono emersi quando abbiamo cercato di far rimanere in equilibrio il robot in prossimità di una finestra o su una superficie non uniforme (Fig. 3.4).

Figura 3.4: Legway dotato di due sensori di luce, un sensore ad ultrasuoni e un giroscopio

3.2 MODELLO

L' NXTway (Fig. 3.5) può essere considerato come un pendolo inverso a due ruote. Per tale motivo il modello matematico e le equazioni di moto dell' NXTway hanno forti similitudini con quelle del pendolo inverso (Fig. 3.6).

Figura 3.5: Fotografia dell'NXTway che abbiamo realizzato

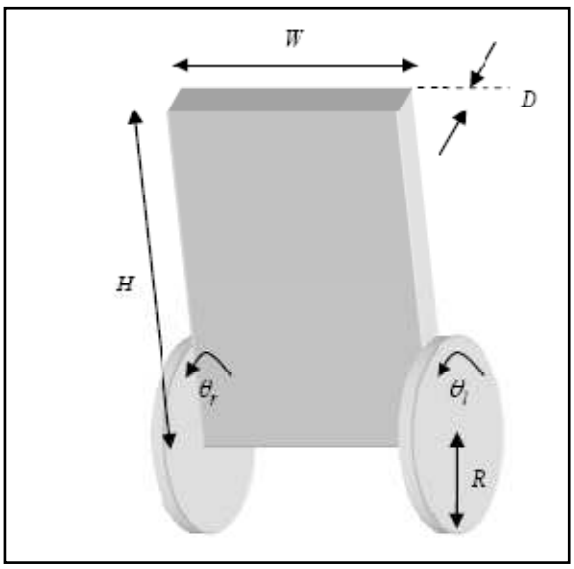


Figura 3.6: Pendolo inverso a due ruote

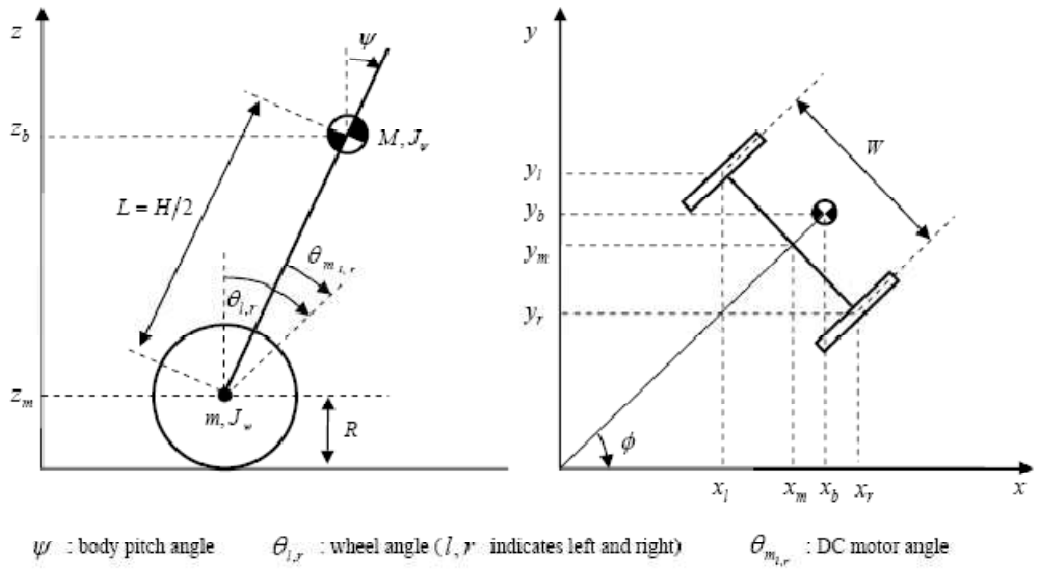


Figura 3.7: Profilo e vista dall'alto del Legway

I parametri fisici utilizzati sono i seguenti:

$g = 9.81$	$[m/\sec^2]$	:	Gravity acceleration
$m = 0.03$	$[kg]$	:	Wheel weight
$R = 0.04$	$[m]$	:	Wheel radius
$J_W = mR^2/2$	$[kgm^2]$	:	Wheel inertia moment
$M = 0.6$	$[kg]$	:	Body weight
$W = 0.14$	$[m]$	:	Body width
$D = 0.04$	$[m]$	:	Body depth
$H = 0.144$	$[m]$	:	Body height
$L = H/2$	$[m]$	:	Distance of the center of mass from the wheel axle
$J_\psi = ML^2/3$	$[kgm^2]$	:	Body pitch inertia moment
$J_\phi = M(W^2 + D^2)/12$	$[kgm^2]$	:	Body yaw inertia moment
$J_m = 1\times10^{-5}$	$[kgm^2]$	:	DC motor inertia moment
$R_m = 6.69$	$[\Omega]$	:	DC motor resistance
$K_b = 0.468$	$[V\sec/rad]$	:	DC motor back EMF constant
$K_t = 0.317$	$[Nm/A]$	:	DC motor torque constant
$n = 1$		:	Gear ratio
$f_m = 0.0022$		:	Friction coefficient between body and DC motor
$f_W = 0$		:	Friction coefficient between wheel and floor.

Essendo difficile calcolare i valori di J_m , n, f_m, f_w sono stati utilizzati i valori più appropriati.

EQUAZIONI DI MOTO DEL PENDOLO INVERSO A DUE RUOTE:

Le equazioni di moto del pendolo inverso a due ruote possono essere ricavate mediante il metodo lagrangiano basato sulle coordinate del sistema nella Fig. 3.7.Se la direzione del pendolo inverso a due ruote all'istante t=0 è nel verso dell'asse x positivo, le coordinate sono le seguenti:

$$(x_m,\ y_m,\ z_m) = (R\theta\cos\phi,\ R\theta\sin\phi,\ R) \tag{3.1}$$

$$(\theta,\ \phi) = \left(\frac{1}{2}(\theta_l + \theta_r)\ \ \frac{R}{W}(\theta_r - \theta_l)\right) \tag{3.2}$$

$$(x_l,\ y_l,\ z_l) = \left(x_m - \frac{W}{2}\sin\phi,\ y_m + \frac{W}{2}\cos\phi,\ z_m\right) \tag{3.3}$$

$$(x_r,\ y_r,\ z_r) = \left(x_m + \frac{W}{2}\sin\phi,\ y_m - \frac{W}{2}\cos\phi,\ z_m\right) \tag{3.4}$$

$$(x_b,\ y_b,\ z_b) = (x_m + L\sin\psi\cos\phi,\ y_m + L\sin\psi\sin\phi,\ z_m + L\cos\psi) \tag{3.5}$$

L'energia cinetica di traslazione T_1, l'energia cinetica di rotazione T_2 e l'energia potenziale U sono:

$$T_1 = \frac{1}{2}m\left(\dot{x}_l^{\,2} + \dot{y}_l^{\,2} + \dot{z}_l^{\,2}\right) + \frac{1}{2}m\left(\dot{x}_r^{\,2} + \dot{y}_r^{\,2} + \dot{z}_r^{\,2}\right) + \frac{1}{2}M\left(\dot{x}_b^{\,2} + \dot{y}_b^{\,2} + \dot{z}_b^{\,2}\right) \tag{3.6}$$

$$T_2 = \frac{1}{2}J_w\dot{\theta}_l^{\,2} + \frac{1}{2}J_w\dot{\theta}_r^{\,2} + \frac{1}{2}J_\psi\dot{\psi}^2 + \frac{1}{2}J_\phi\dot{\phi}^2 + \frac{1}{2}n^2J_m(\dot{\theta}_l - \dot{\psi})^2 + \frac{1}{2}n^2J_m(\dot{\theta}_r - \dot{\psi})^2 \tag{3.7}$$

$$U = mgz_l + mgz_r + Mgz_b \tag{3.8}$$

Il quinto e il sesto termine in T_2 sono l'energia cinetica di rotazione di un'armatura situata a destra e a sinistra di un motore in corrente continua. Il lagrangiano ha la seguente espressione:

$$L = T_1 + T_2 - U \tag{3.9}$$

Come coordinate generalizzate si utilizzano le seguenti variabili:

θ : Average angle of left and right wheel
ψ : Body pitch angle
ϕ : Body yaw angle

Le equazioni di Lagrange sono le seguenti:

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\theta}}\right)-\frac{\partial L}{\partial \theta}=F_\theta \tag{3.10}$$

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\psi}}\right)-\frac{\partial L}{\partial \psi}=F_\psi \tag{3.11}$$

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\phi}}\right)-\frac{\partial L}{\partial \phi}=F_\phi \tag{3.12}$$

Dalle equazioni (3.10)-(3.12) si ricavano le seguenti equazioni:

$$\begin{aligned}&\left[(2m+M)R^2+2J_w+2n^2J_m\right]\ddot{\theta}+\left(MLR\cos\psi-2n^2J_m\right)\ddot{\psi}\\&-MLR\dot{\psi}^2\sin\psi-\left[(2m+M)R^2\theta+MLR\sin\psi\right]\dot{\phi}^2=F_\theta\end{aligned} \tag{3.13}$$

$$\begin{aligned}&\left(MLR\cos\psi-2n^2J_m\right)\ddot{\theta}+\left(ML^2+J_\psi+2n^2J_m\right)\ddot{\psi}-MgL\sin\psi\\&-\left(MLR\,0+ML^2\sin\psi\right)\dot{\phi}^2\cos\psi=F_\psi\end{aligned} \tag{3.14}$$

$$\begin{aligned}&\left[\frac{1}{2}mW^2+J_\phi+\frac{W^2}{2R^2}\left(J_w+n^2J_m\right)+(2m+M)R^2\theta^2+2MLR\theta\sin\psi\right]\ddot{\phi}\\&+2\left[(2m+M)R^2\theta\dot{\theta}+ML^2\dot{\psi}\sin\psi\cos\psi+MLR\left(\dot{\theta}\sin\psi+\theta\dot{\psi}\cos\psi\right)\right]\dot{\phi}=F_\phi\end{aligned} \tag{3.15}$$

Considerando il momento torcente del motore DC e l'attrito viscoso, le forze che entrano in gioco sono le seguenti:

$$\left(F_\theta,\ F_\psi,\ F_\phi\right)=\left(\frac{1}{2}\left(F_l+F_r\right),\ F_\psi,\ \frac{R}{W}\left(F_r-F_l\right)\right) \tag{3.16}$$

$$F_l=nK_t i_l+f_m(\dot{\psi}-\dot{\theta}_l)-f_w\dot{\theta}_l \tag{3.17}$$

$$F_r=nK_t i_r+f_m(\dot{\psi}-\dot{\theta}_r)-f_w\dot{\theta}_r \tag{3.18}$$

$$F_\psi=-nK_t i_l-nK_t i_r-f_m(\dot{\psi}-\dot{\theta}_l)-f_m(\dot{\psi}-\dot{\theta}_r) \tag{3.19}$$

Dove $i_{l,r}$ è la corrente del motore DC. Per controllare quest'ultimo non si può utilizzare direttamente la corrente del motore DC perché esso è basato sul

controllo PWM. Tuttavia si può valutare la relazione tra la corrente $i_{l,r}$ e la tensione $v_{l,r}$ utilizzando le equazioni del motore DC. Generalmente l'equazione del motore DC è la seguente:

$$L_m \dot{i}_{l,r} = v_{l,r} + K_b(\dot{\psi} - \dot{\theta}_{l,r}) - R_m i_{l,r} \tag{3.20}$$

Noi considereremo l'induttanza trascurabile ed approssimabile a zero. Possiamo così ricavare l'espressione della corrente:

$$i_{l,r} - \frac{v_{l,r} + K_b(\dot{\psi} - \dot{\theta}_{l,r})}{R_m} \tag{3.21}$$

Dall'equazione (3.21), la forza può essere espressa utilizzando il valore della tensione del motore:

$$F_\theta = \frac{\alpha}{2}(v_l + v_r) - (\beta + f_w)\dot{\theta} + \beta\dot{\psi} \tag{3.22}$$

$$F_\psi = -\alpha(v_l + v_r) + 2\beta\dot{\theta} - 2\beta\dot{\psi} \tag{3.23}$$

$$F_\phi = \frac{R}{W}\alpha(v_r - v_l) - \left(\beta + \frac{W}{R}f_w\right)\dot{\phi} \tag{3.24}$$

$$\alpha = \frac{nK_t}{R_m}, \quad \beta = \frac{nK_tK_b}{R_m} + f_m \tag{3.25}$$

EQUAZIONI DI STATO DEL PENDOLO INVERSO A DUE RUOTE:

Si possono utilizzare le equazioni di stato basate sulla moderna teoria del controllo linearizzando le equazioni del moto nel punto di equilibrio dell'NXTway. Questo significa che noi consideriamo il limite $\psi \to 0$ ($\sin\psi \to \psi$, $\cos\psi \to 1$) e trascuriamo i termini di secondo ordine come $\dot{\psi}^2$. Le equazioni del moto (3.13)-(3.15) sono quindi approssimate come segue:

$$\left[(2m+M)R^2+2J_w+2n^2J_m\right]\ddot{\theta}+\left(MLR-2n^2J_m\right)\ddot{\psi}=F_\theta \tag{3.26}$$

$$\left(MLR-2n^2J_m\right)\ddot{\theta}+\left(ML^2+J_\psi+2n^2J_m\right)\ddot{\psi}-MgL\psi=F_\psi \tag{3.27}$$

$$\left[\frac{1}{2}mW^2+J_\phi+\frac{W^2}{2R^2}\left(J_w+n^2J_m\right)\right]\ddot{\phi}=F_\phi \tag{3.28}$$

Le equazioni (3.26) e (3.27) contengono ψ e θ , l'equazione (3.28) ha solo Φ. Queste equazioni possono essere espresse nella seguente forma:

$$E\begin{bmatrix}\ddot{\theta}\\ \ddot{\psi}\end{bmatrix}+F\begin{bmatrix}\dot{\theta}\\ \dot{\psi}\end{bmatrix}+G\begin{bmatrix}\theta\\ \psi\end{bmatrix}=H\begin{bmatrix}v_l\\ v_r\end{bmatrix} \tag{3.29}$$

$$E=\begin{bmatrix}(2m+M)R^2+2J_w+2n^2J_m & MLR-2n^2J_m\\ MLR-2n^2J_m & ML^2+J_\psi-2n^2J_m\end{bmatrix}$$

$$F=\begin{bmatrix}\beta+f_w & -\beta\\ -2\beta & 2\beta\end{bmatrix}$$

$$G=\begin{bmatrix}0 & 0\\ 0 & -MgL\end{bmatrix}$$

$$H=\begin{bmatrix}\alpha/2 & \alpha/2\\ -\alpha & -\alpha\end{bmatrix}$$

$$K\ddot{\phi}+I\dot{\phi}=J(v_r-v_l) \tag{3.30}$$

$$I=\beta+\frac{W}{R}f_w$$

$$J=\frac{R}{W}\alpha$$

$$K=\frac{1}{2}mW^2+J_\phi+\frac{W^2}{2R^2}\left(J_w+n^2J_m\right)$$

Noi consideriamo le seguenti variabili di stato x_1, x_2 e **u** come input:

$$\mathbf{x_1}=\left[\theta,\ \psi,\ \dot{\theta},\ \dot{\psi}\right]^T,\ \mathbf{x_2}=\left[\phi,\ \dot{\phi}\right]^T,\ \mathbf{u}=\left[v_l,\ v_r\right]^T \tag{3.31}$$

Di conseguenza possiamo derivare le equazioni di stato del pendolo inverso a due ruote dalle equazioni (3.29) e (3.30).

$$\dot{\mathbf{x}}_1 = A_1\mathbf{x}_1 + B_1\mathbf{u} \tag{3.32}$$

$$\dot{\mathbf{x}}_2 = A_2\mathbf{x}_2 + B_2\mathbf{u} \tag{3.33}$$

$$A_1 = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & A_1(3,2) & A_1(3,3) & A_1(3,4) \\ 0 & A_1(4,2) & A_1(4,3) & A_1(4,4) \end{bmatrix}, \quad B_1 = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ B_1(3) & B_1(3) \\ B_1(4) & B_1(4) \end{bmatrix} \tag{3.34}$$

$$A_2 = \begin{bmatrix} 0 & 1 \\ 0 & -I/K \end{bmatrix}, \quad B_2 = \begin{bmatrix} 0 & 0 \\ -J/K & J/K \end{bmatrix} \tag{3.35}$$

$$A_1(3,2) = -MgLE(1,2)/\det(E)$$
$$A_1(4,2) = MgLE(1,1)/\det(E)$$
$$A_1(3,3) = -\left[(\beta + f_w)E(2,2) + 2\beta E(1,2)\right]/\det(E)$$
$$A_1(4,3) = \left[(\beta + f_w)E(1,2) + 2\beta E(1,1)\right]/\det(E)$$
$$A_1(3,4) = \beta\left(E(2,2) + 2E(1,2)\right)/\det(E)$$
$$A_1(4,4) = -\beta\left(E(1,2) + 2E(1,1)\right)/\det(E)$$
$$B_1(3) = \alpha\left(E(2,2)/2 + E(1,2)\right)/\det(E)$$
$$B_1(4) = -\alpha\left(E(1,2)/2 + E(1,1)\right)/\det(E)$$
$$\det(E) = E(1,1)E(2,2) - E(1,2)^2$$

3.3 CONTROLLO

Per effettuare il controllo dell'NXTway abbiamo utilizzato la teoria moderna del controllo.

INPUTS E OUTPUTS:

L'ingresso di un attuatore è costituito dal PWM duty destro e sinistro del motore DC, sebbene l'input **u** nell'equazione (3.31) è la tensione. Le uscite dei sensori sono l'angolo delle ruote $\theta_{l,r}$ e l'angolo che il centro di massa forma con la verticale ψ (Fig. 3.8).

Figura 3.8: Inputs e outputs

E' facile stimare θ e Φ utilizzando $\theta_{l,r}$. Ci sono due metodi per stimare ψ a partire da $\dot{\psi}$.

- Stimare ψ utilizzando un osservatore basato sulla moderna teoria del controllo.
- Derivare ψ integrando in modo numerico la velocità angolare.

Noi utilizzeremo il secondo metodo per controllare l'NXTway.

STABILITA':

E' facile capire che la posizione di equilibrio dell'NXTway non è stabile. Per ottenere l'equilibrio dobbiamo muovere l'NXTway nella stessa direzione dell'angolo che il centro di massa forma con la verticale. La teoria moderna del controllo fornisce molte tecniche per stabilizzare un sistema instabile.

PROGETTO DEL CONTROLLORE:

L'espressione (3.29) è l'equazione di un sistema massa-molla-smorzatore. La Fig. 3.9 mostra un pendolo inverso a due ruote modellato come un sistema massa-molla-smorzatore.

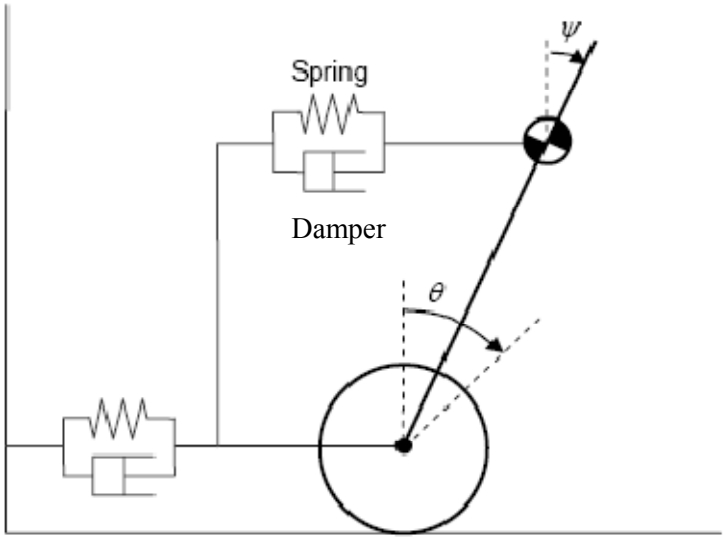


Figura 3.9: Sistema equivalente massa,molla e smorzatore

E' possibile rendere stabile il pendolo inverso a due ruote aggiustando la costante della molla e la costante di attrito dello smorzatore. Per controllare l'NXT si utilizza un servo controllore e si prende come riferimento θ. E' importante non utilizzare nessun altra variabile come riferimento perché altrimenti il sistema diventerebbe incontrollabile. Lo schema a blocchi del servo contollore utilizzato è quello mostrato nella Fig. 3.10:

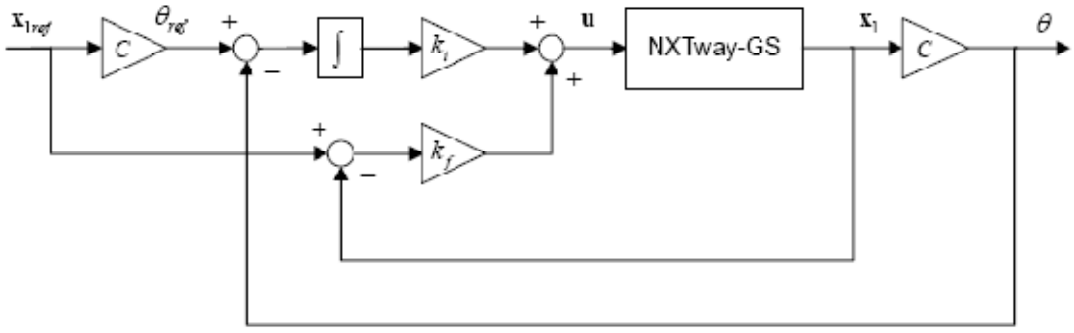


Figura 3.10: Schema a blocchi del servo controllore dell'NXTway_GS

Possiamo calcolare il guadagno in retroazione con il metodo del regolatore lineare quadratico. Vengono scelte empiricamente le seguenti matrici Q ed R:

$$Q = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 6\times10^{5} & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 4\times10^{2} \end{bmatrix}, R = \begin{bmatrix} 1\times10^{3} & 0 \\ 0 & 1\times10^{3} \end{bmatrix}$$

Q(2,2) è un peso per il body pitch angle, e Q(5,5) è un peso per il tempo di integrazione della differenza fra l'angolo medio misurato e il valore desiderato.

Il file param_controller.m calcola il regolatore lineare quadratico e definisce gli altri parametri come in Fig. 3.11:

param_controller.m

```
% Controller Parameters

% Servo Gain Calculation using Optimal Regulator
A_BAR = [A1, zeros(4, 1); C1(1, :), 0];
B_BAR = [B1; 0, 0];
QQ = [
      1, 0,   0, 0, 0
      0, 6e5, 0, 0, 0
      0, 0,   1, 0, 0
      0, 0,   0, 1, 0
      0, 0,   0, 0, 4e2
      ];
RR = 1e3 * eye(3);
KK = lqr(A_BAR, B_BAR, QQ, RR);
k_f = KK(1, 1:4);         % feedback gain
k_i = KK(1, 5);           % integral gain
% suppress velocity gain because it fluctuates NXTway-GS
k_f(3) = k_f(3) * 0.85;
```

Figura 3.11: Screenshot del file param_controller.m

I valori ottenuti per i due guadagni sono:

$k_f = [-0.8703, -31.9978, -1.1566, -2.7887]$

$k_i = -0.4472$

Il valore del guadagno k_f (3) è stato aggiustato per diminuire le oscillazioni dell' NXTway.

3.4 SCHEMI A BLOCCHI

I blocchi principali del modello sono i seguenti:

3.4.1 CONTROLLORE:

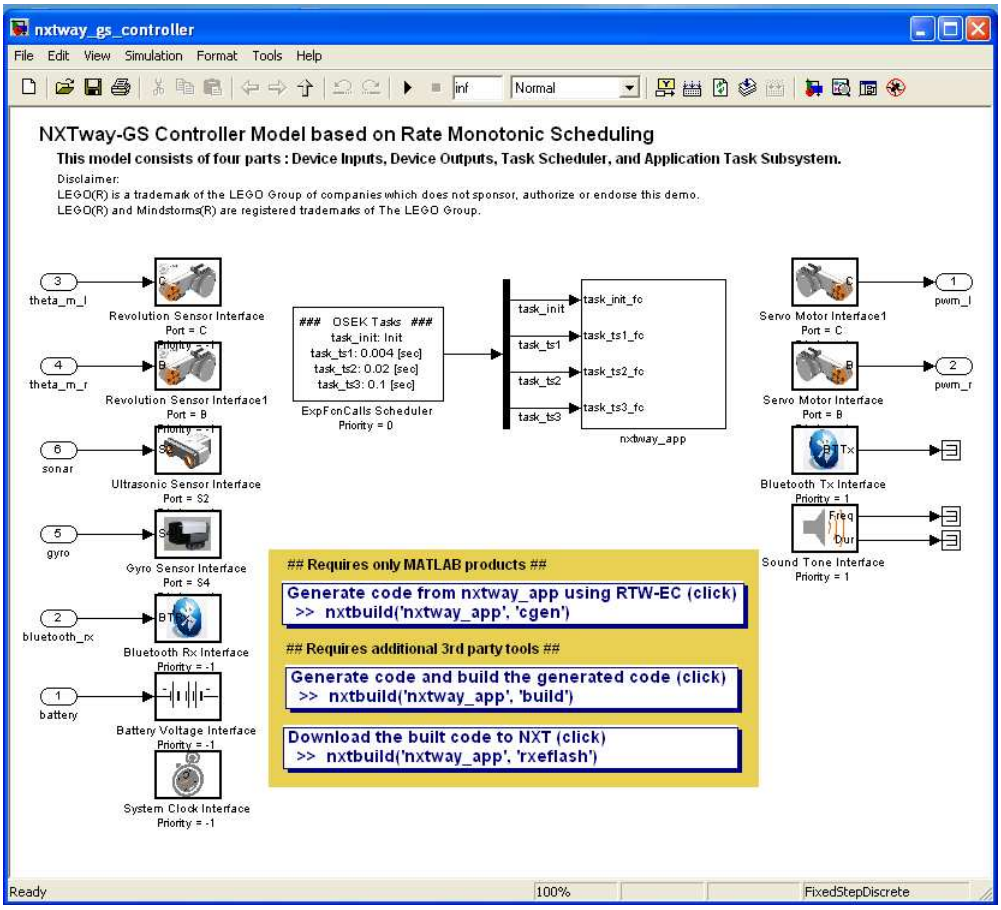


Figura 3.12: Screenshot del controllore

Il controllore (Fig. 3.12-3.14) funziona a tempo discreto, con un tempo di campionamento pari a 1 ms , mentre l'impianto è tempo continuo. Perciò è necessario convertire da tempo continuo a tempo discreto e viceversa inserendo dei convertitori D2C e C2D come riportato in Fig. 3.13:

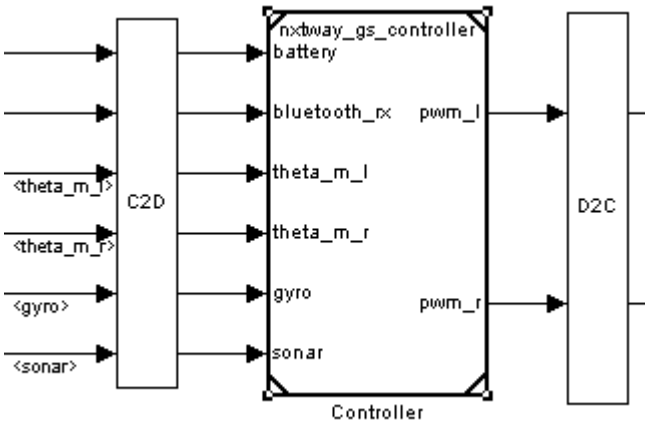


Figura 3.13: Schema a blocchi del controllore e dei convertitori D2C e C2D

3.4.2 CONFIGURAZIONE DEI TASK:

L'NXTway ha 4 task (Fig. 3.14):

Task	Period	Works
task_init	Initialization only	Initial value setting
task_ts1	4 [ms]	Balance & drive control Data logging Gyro offset calibration
task_ts2	20 [ms]	Obstacle check
task_ts3	100 [ms]	Time check Battery voltage averaging

Figura 3.14: Periodo di campionamento e funzione svolta dai task

La durata del task_ts1 è determinata dal massimo numero di campionamenti che può effettuare il giroscopio in un secondo. La durata del task_ts2 è determinata

dal massimo numero di campionamenti che il sensore ad ultrasuoni può effettuare in un secondo.

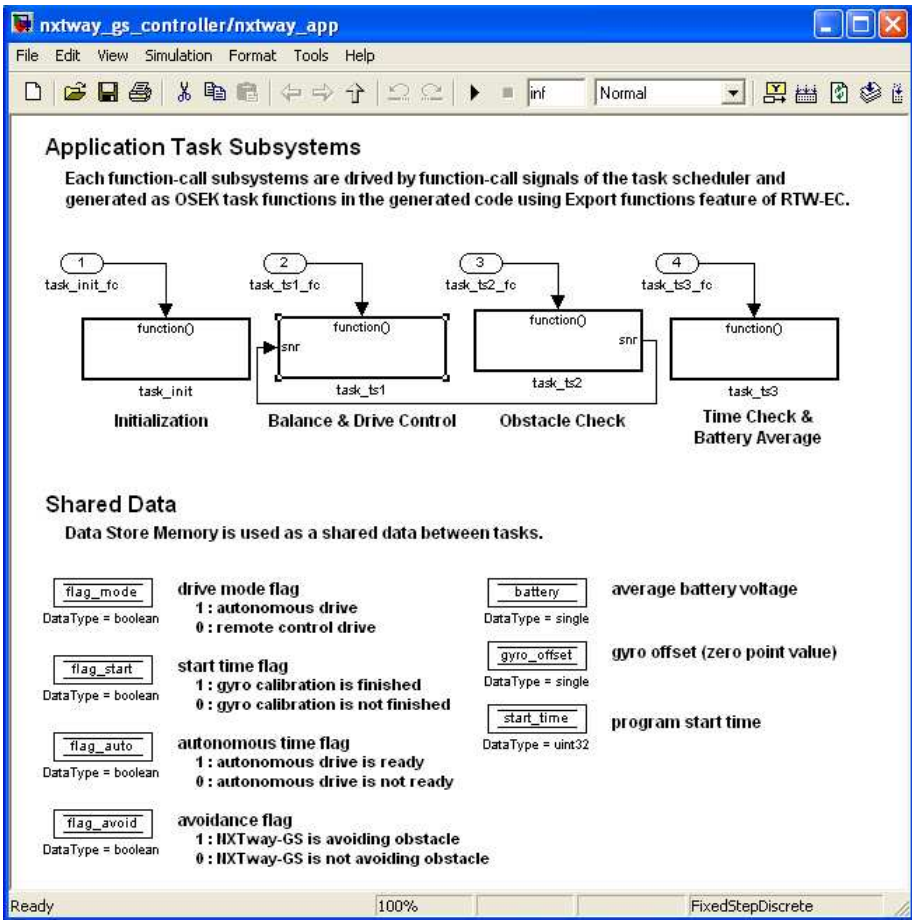


Figura 3.15: Screenshot dell'interno del controllore

3.4.3 TIPI DI DATI:

Per il controllo dell'equilibrio e del movimento abbiamo utilizzato dati in virgola mobile a singola precisione, per ridurre l'errore computazionale.

Il processore ARM7 non prevede processi contenenti numeri in virgola mobile. Ma grazie alla libreria fornita da GCC abbiamo potuto utilizzare software con aritmetica a virgola mobile e con precisione singola.

3.4.4 SCHEDULER E TASKS:

Il blocco ExpFcnCalls Scheduler (Fig. 3.16) deve essere configurato con la dimensione dello stack, con il nome e il periodo del task.

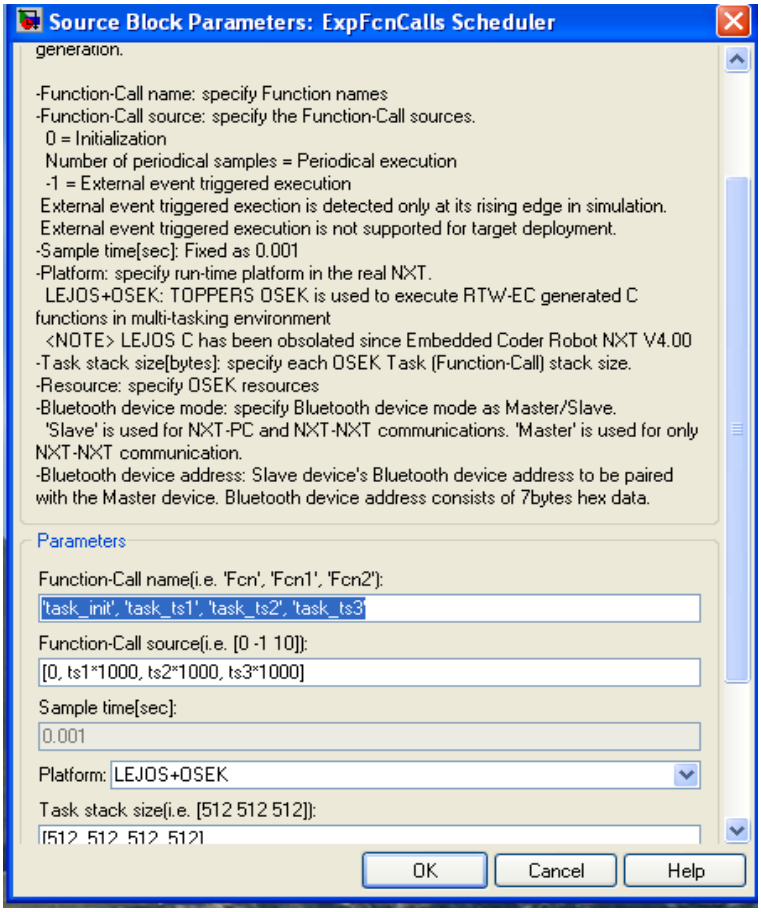


Figura 3.16: Impostazione della dimensione dello stack, del nome e del periodo del task

3.4.5 PRIORITA':

Appena viene progettato il controllore è necessario impostare la priorità (Fig. 3.17) sia dei blocchi riferiti ai dispositivi sia dell'ExpFcnCalls Scheduler per ordinarli con il seguente criterio:

1. Input dei dispositivi
2. Tasks
3. Output dei dispositivi

I numeri più bassi indicano alta priorità e si possono utilizzare anche numeri negativi.

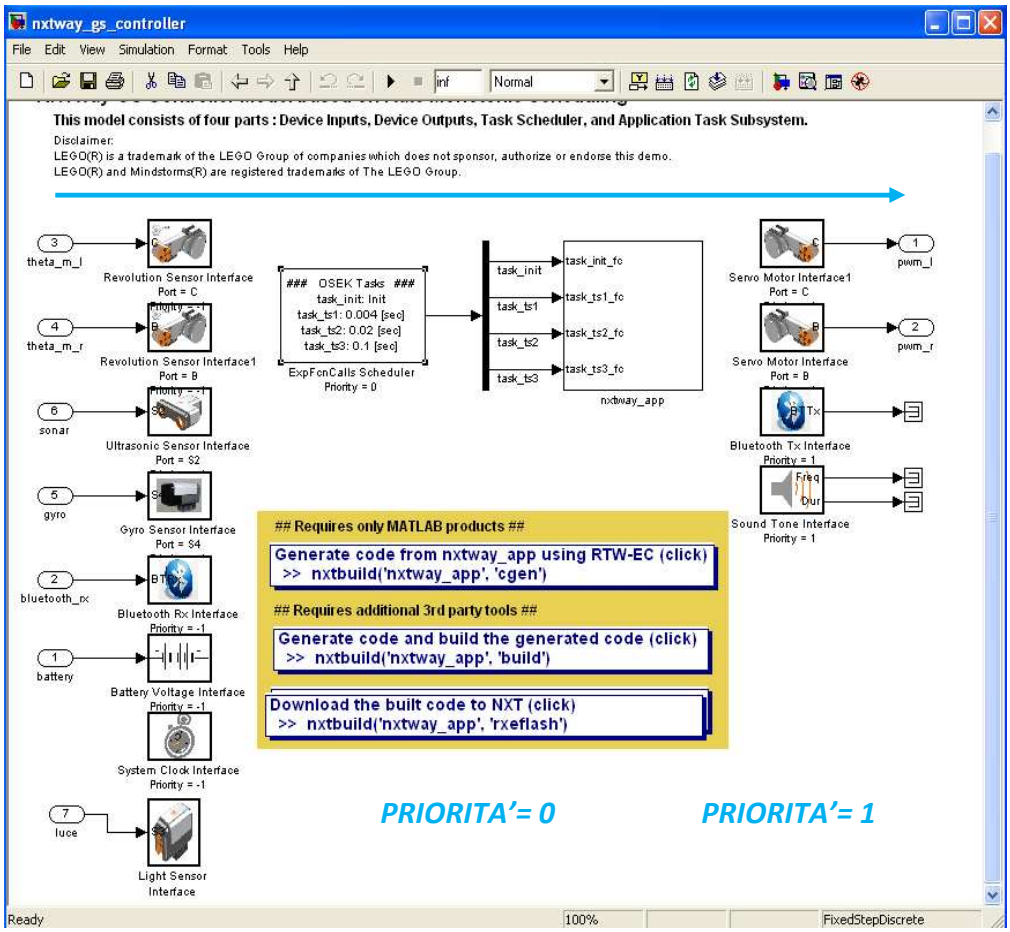


Figura 3.17: Controllore con indice di priorità

3.4.6 DATI CONDIVISI:

Si utilizzano blocchi Data Store Memory (Fig. 3.18) per condividere le informazioni tra più task.

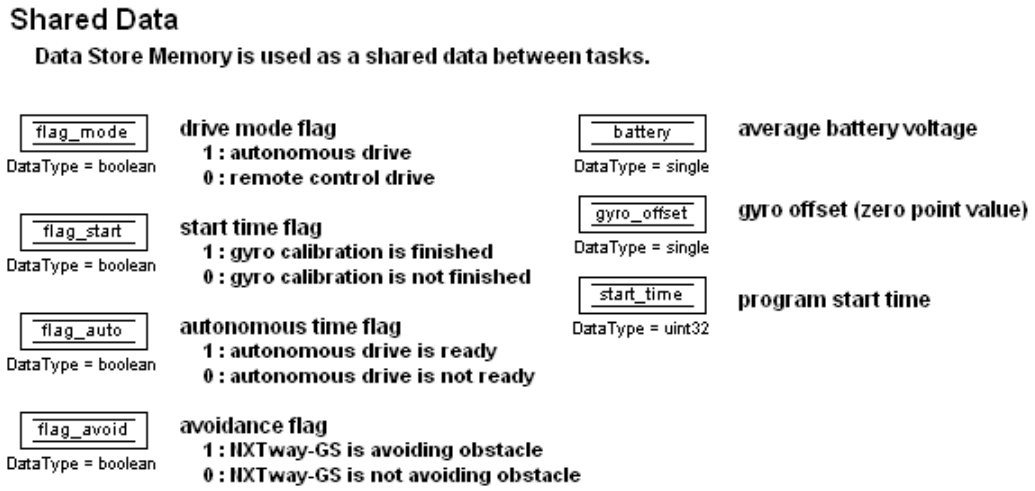


Figura 3.18: Impostazione dei flag, del voltaggio medio della batteria, del gyro offset e del tempo di avvio del progrmma

3.4.7 NXTway:

Questo sottosistema rappresenta un modello matematico per l' NXTway (Fig. 3.19). E' costituito dai sensori, dagli attuatori e da un modello lineare dell' impianto. Converte il tipo dei segnali in double, calcola la dinamica dell'impianto utilizzando un'aritmetica a virgola mobile con precisione double e restituisce i risultati dopo la quantizzazione.

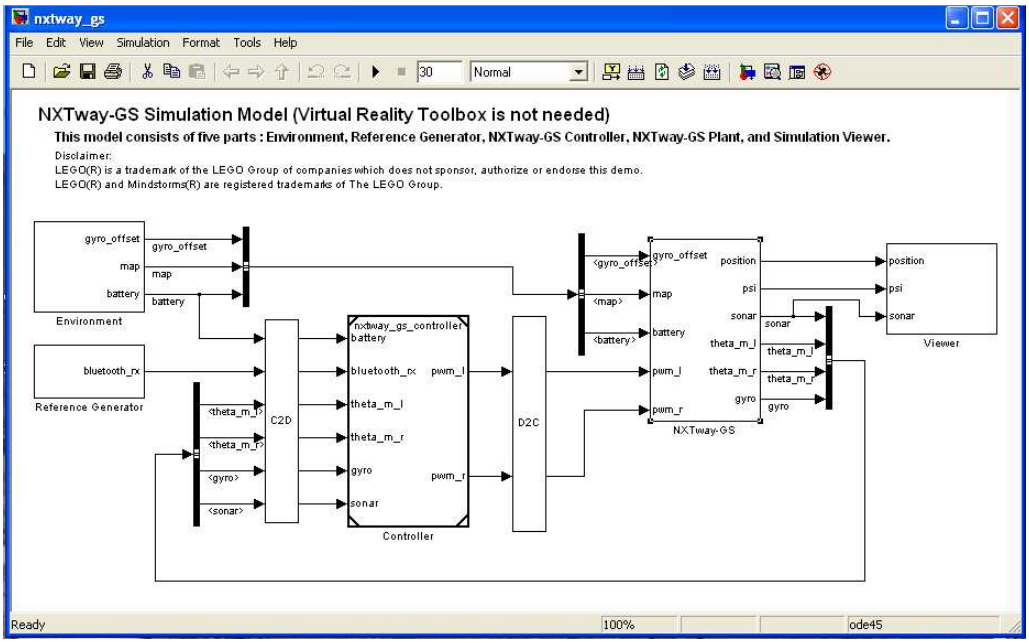


Figura 3.19: Schema a blocchi dell'NXTway_GS

Questo sistema viene utilizzato soltanto per eseguire una simulazione nell' ambiente Simulink. Nella figura sottostante è rappresentato il modello dell'NXTway con il sensore giroscopico. Ogni volta che viene apportata una modifica sul sistema reale (per esempio l' inserimento di sensori differenti) si deve cambiare anche questo blocco per avere una simulazione coerente con il modello fisico.

3.4.8 VIEWER:

Questo sottosistema permette di visualizzare la simulazione. All'interno del viewer una volta abbiamo inserito un blocco XY Graph per vedere sul display la posizione del robot(Fig. 3.21) e un'altra volta il blocco 3D Visualization (Fig. 3.20) che, sfruttando Virtual Reality Toolbox, permette di seguire la simulazione in 3D.

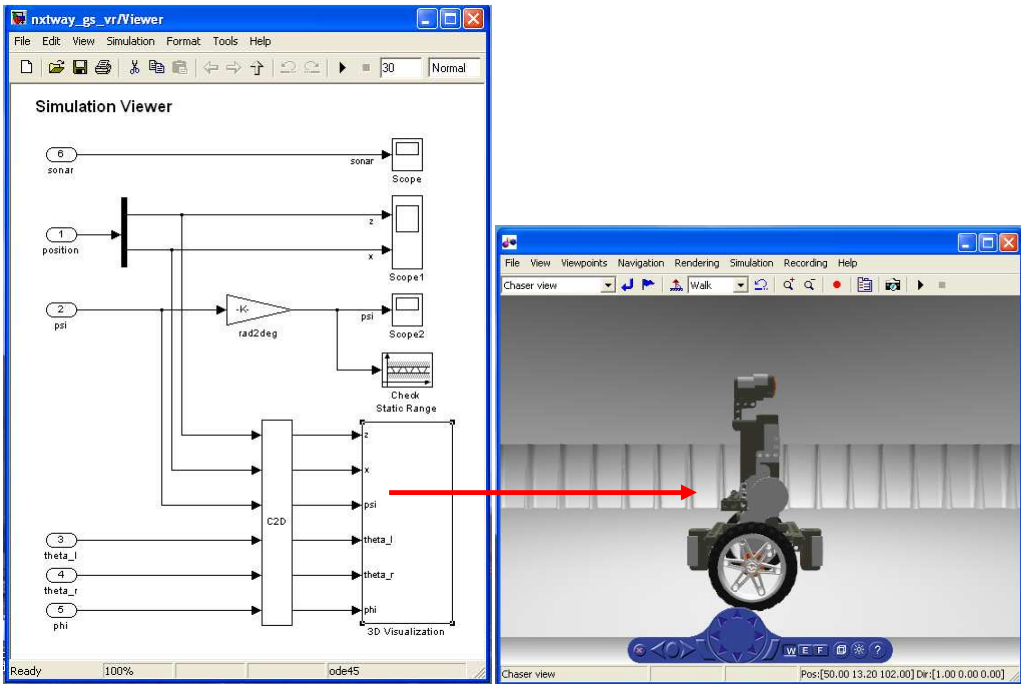


Figura 3.20: Screenshot del blocco Viewer e della simulazione 3-D

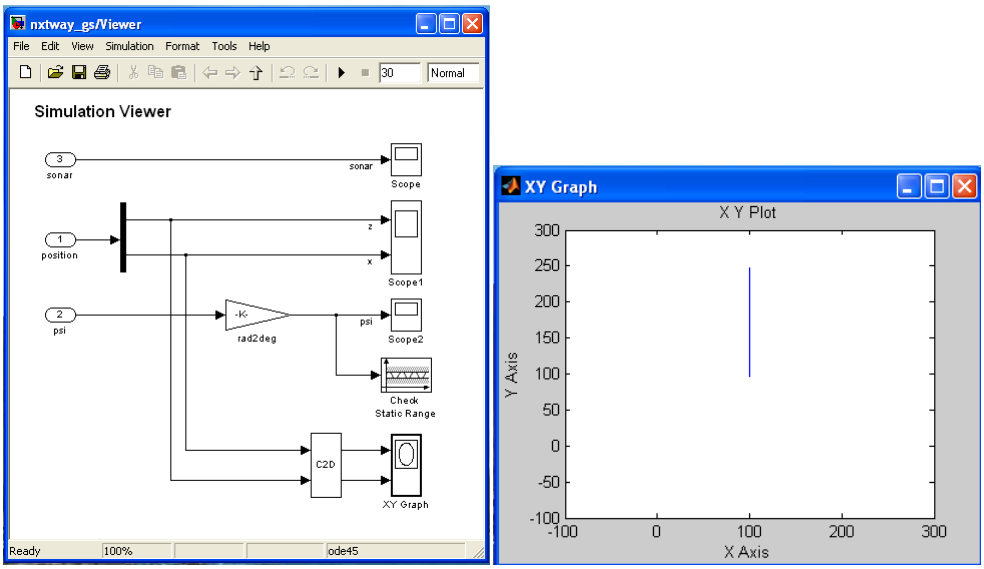


Figura 3.21: Screenshot del blocco Viewer e della simulazione 2-D

Esistono quattro modalità per visualizzare il robot (Fig 3.22-3.23-3.24-3.25):

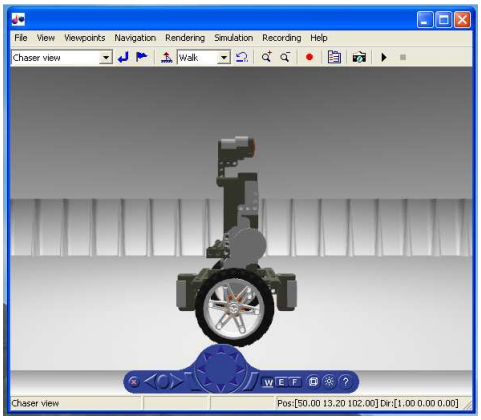


Fig. 3.22: CHASER VIEW: la telecamera segue il robot.

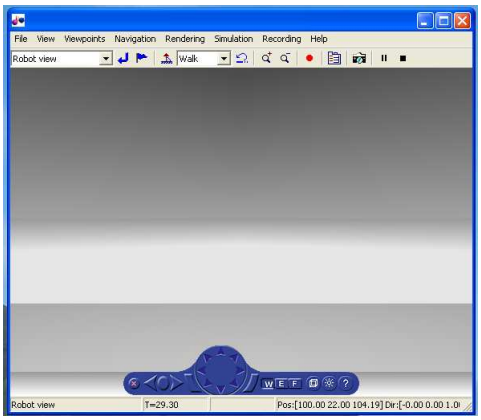


Fig. 3.23: ROBOT VIEW: la telecamera è posizionata sul sensore ad ultrasuoni.

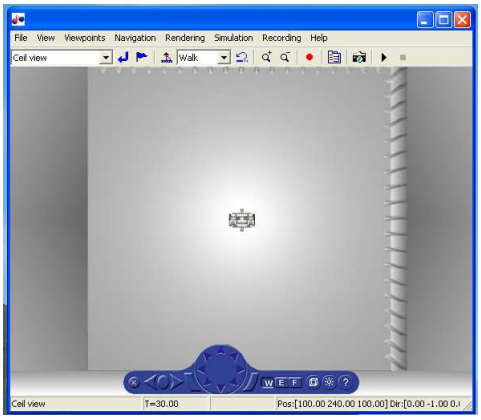


Fig. 3.24:CEIL VIEW: la telecamera è posizionata in

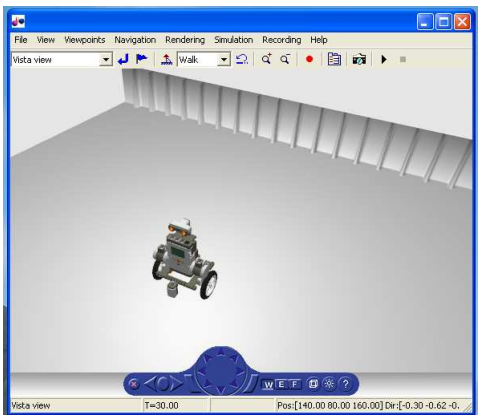


Fig. 3.25:VISTA VIEW: la telecamera si trova in una posizione prefissata.

3.4.9 ATTUATORI:

Gli attuatori calcolano la tensione del motore DC utilizzando il PWM duty ricavato dal controllore. Abbiamo considerato l'attrito viscoso e quello coulombiano, presenti lungo il percorso, come zona morta prima di calcolare la tensione. La massima tensione applicabile al motore DC serve per calcolare il PWM duty. Abbiamo utilizzato la seguente equazione sperimentale che prende in considerazione la tensione della batteria V_B e la massima tensione del motore DC V_{max}.

$V_{max}=0.001089 \times V_B - 0.625$

3.4.10 IMPIANTO:

Il modello dell'impianto (Fig. 3.26) deriva dalle equazioni di stato del pendolo inverso a due ruote. Nell'NXTway con il giroscopio l'impianto è modellato in modo tale che la simulazione inizi dopo la calibrazione del giroscopio, mentre nell'altro non ci sono tempi di attesa.

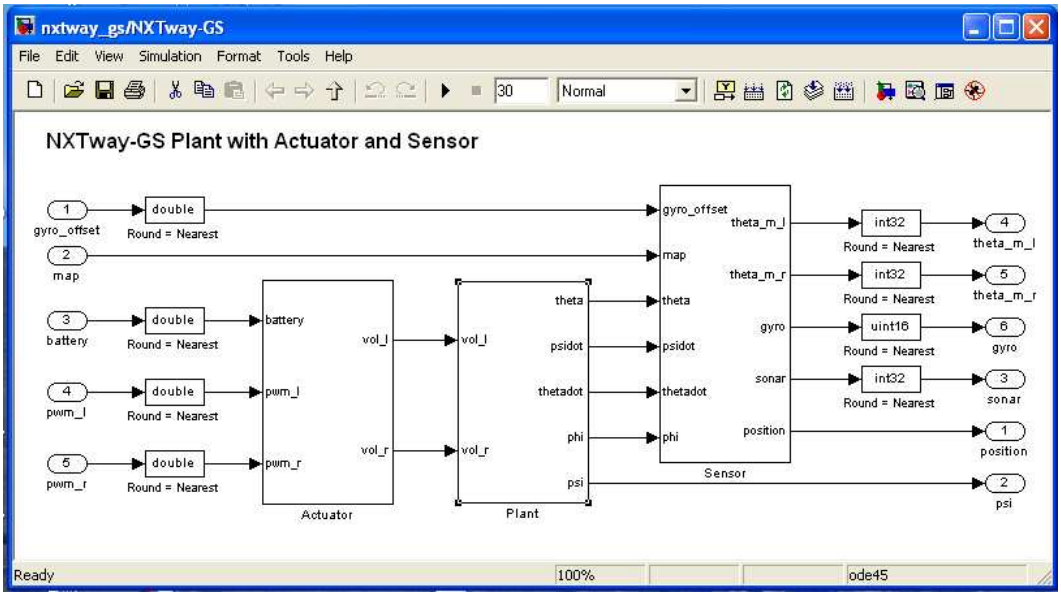


Figura 3.26: Schema a blocchi dell'impianto dell'NXT collegato ai blocchi Sensor e Actuator

3.4.11 SENSORI:

All'interno del blocco Sensor (Fig. 3.27) i valori calcolati nell'impianto vengono trasformati nelle uscite effettive dei sensori. Poiché il costo computazionale per il calcolo della distanza e per il rilevamento degli ostacoli è elevato, abbiamo inserito dei blocchi Rate Transition per ovviare a questo problema.

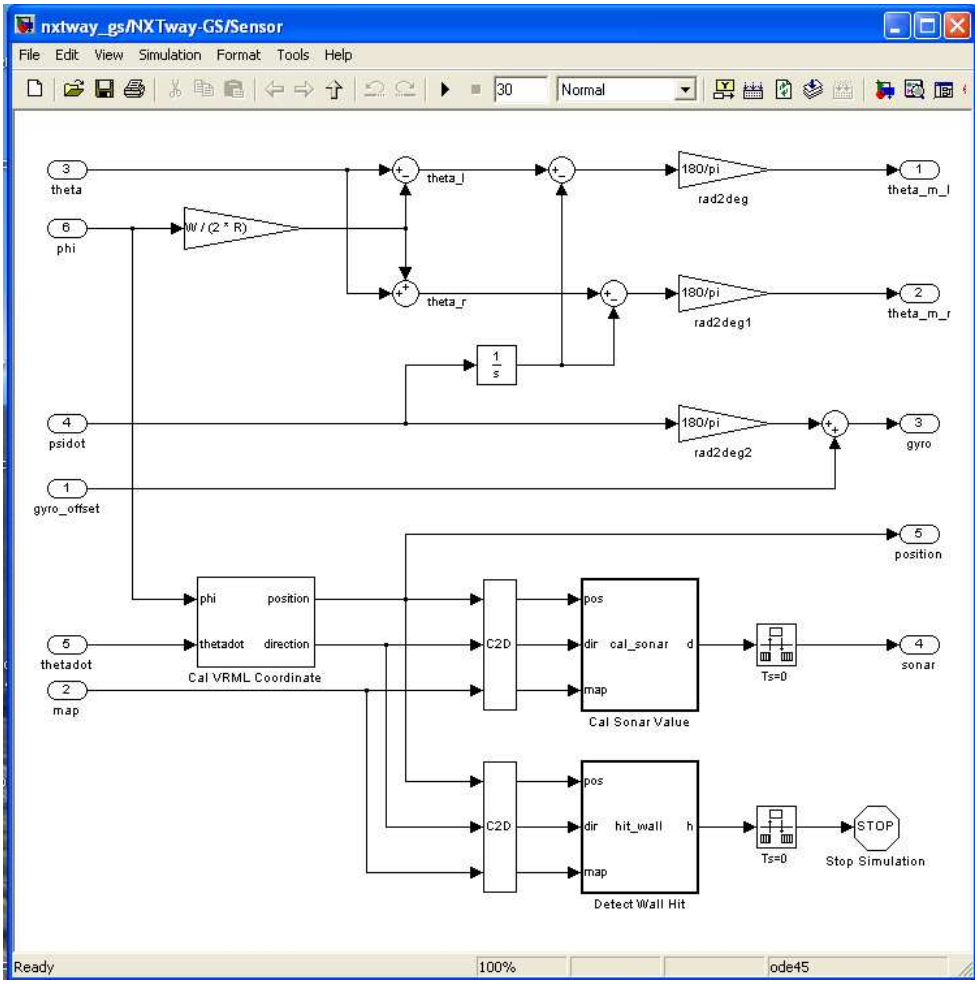


Figura 3.27: Screenshot dell'interno del blocco Sensor

3.4.12 DATA LOGGING

All'interno del controllore abbiamo inserito un blocco NXT GamePad ADC Data Logger (Fig. 3.28) per registrare i valori in uscita dai sensori. Questo blocco può anche prelevare dati interi a 8 e a 16 bit. Per registrare dati in virgola mobile è

necessario quantizzarli dividendoli per valori appropriati (Fig. 3.29) perché il blocco NXT GamePad ADC Data Logger non supporta dati in virgola mobile.

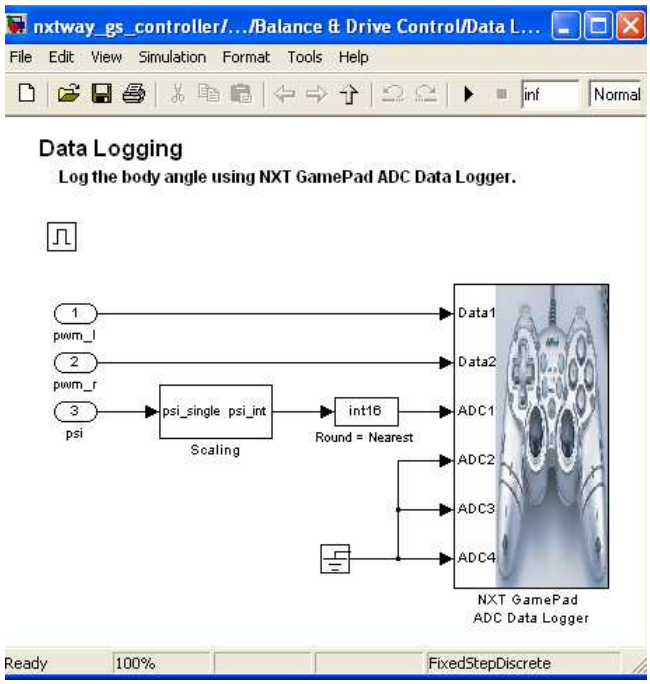


Figura 3.28: Screenshot del blocco Data Logging

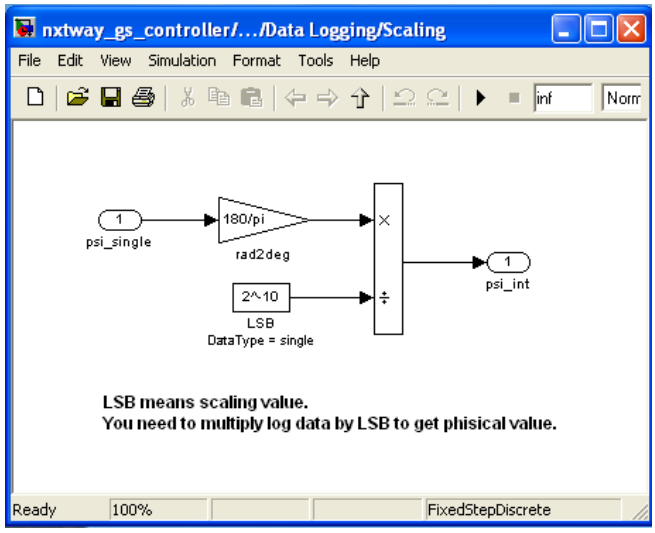


Figura 3.29: Screenshot del blocco Scaling

Come si vede dall'immagine il valore prelevato viene diviso per il valore di fondo scala (LSB).

3.5 MODIFICHE APPORTATE:

3.5.1 CONTROLLO EFFETTUATO MEDIANTE UN SENSORE DI LUCE:

E' necessario porre nell'inizializzazione l'interfaccia Read del sensore di luce (Fig. 3.30).

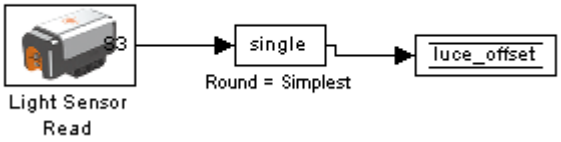


Figura 3.30: Interfaccia Read dl sensore di luce

Nel task_ts1 abbiamo aggiunto un blocco per la calibrazione del sensore di luce (Fig. 3.31).

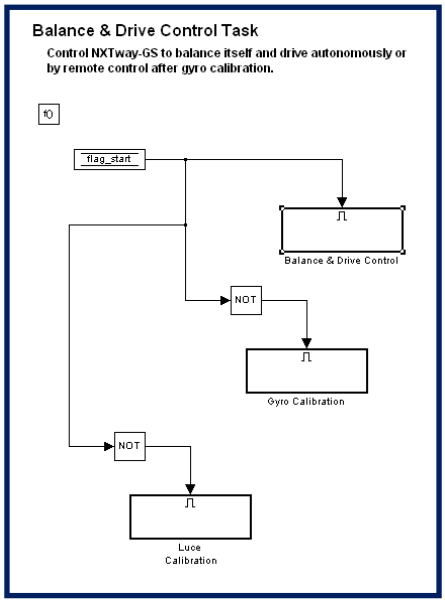


Figura 3.31: Blocco Balance & Drive Control Task modificato con il blocco per la calibrazione del sensore di luce

Questo è costituito dall'interfaccia Light Sensor Read il cui valore viene filtrato mediante un passa basso (Fig. 3.32).

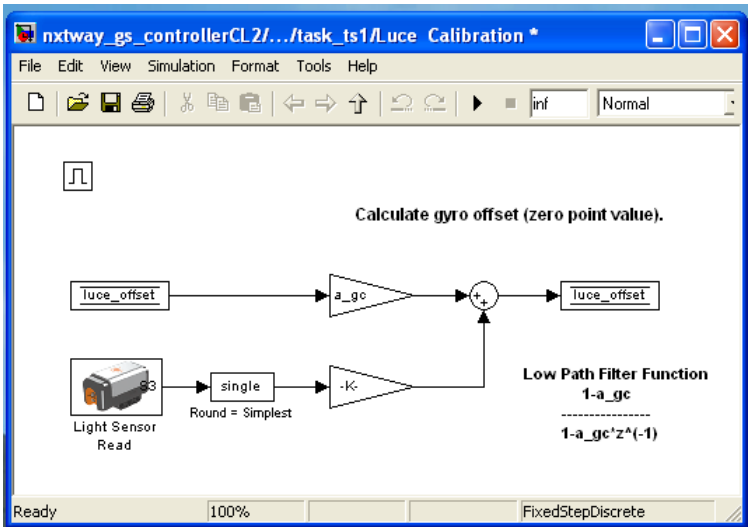


Figura 3.32: Screenshot dell'interno del Blocco Luce Calibration

All'ingresso del controllore abbiamo collegato i dati provenienti dal sensore di luce, al posto di quelli del giroscopio (Fig. 3.33).

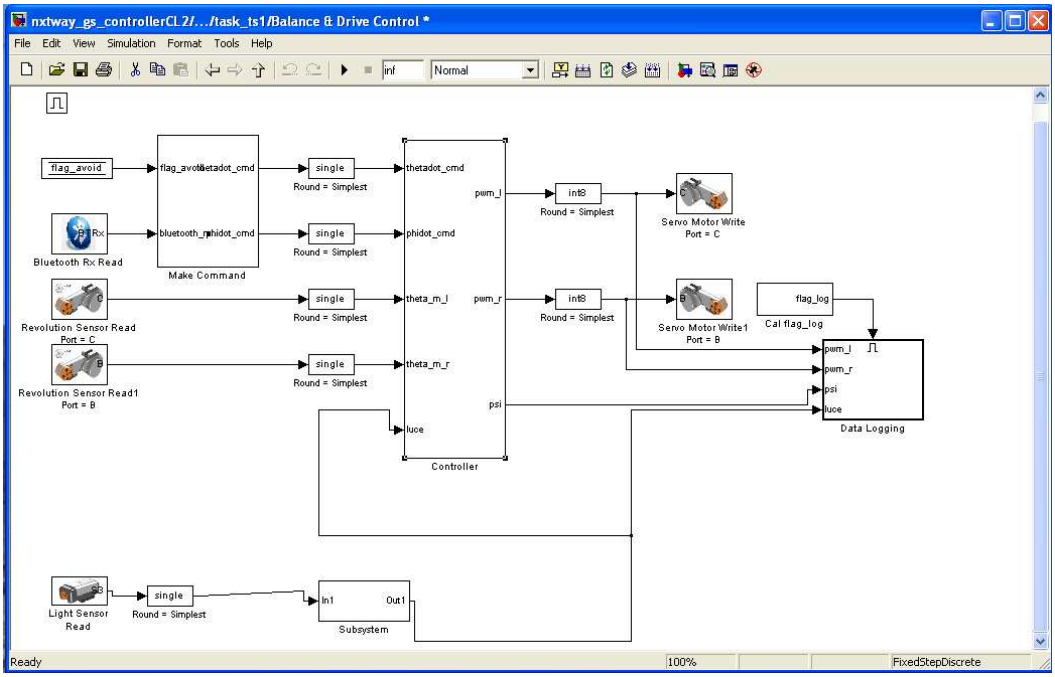


Figura 3.33: Screenshot del blocco Balance & Drive Control con un sensore di luce

Prima di essere inviato al controllore, il valore rilevato dal sensore di luce passa attraverso un Subsystem (Fig. 3.34). Quest'ultimo è costituito da un filtro passa basso , dal blocco Cal luce_offset e da un convertitore che trasforma i gradi in radianti.

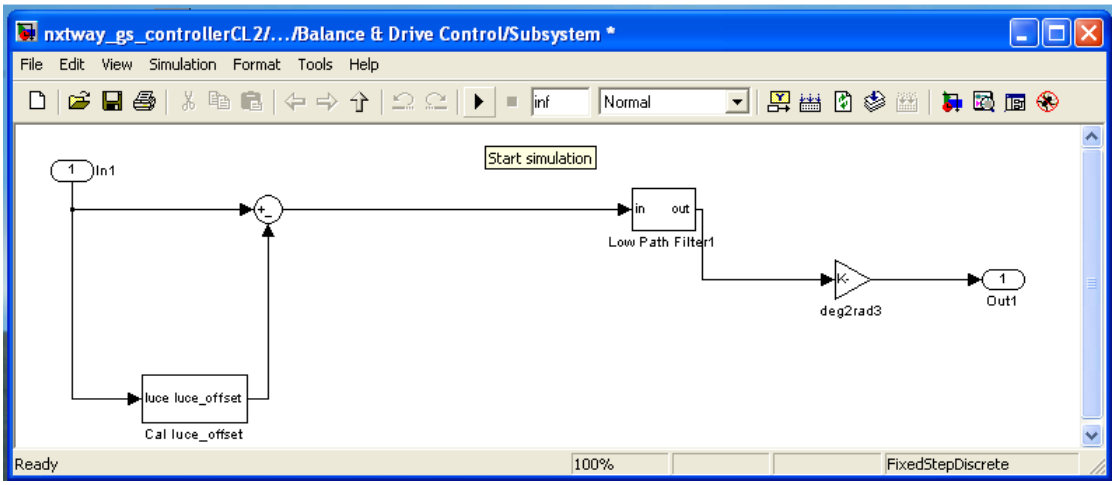


Figura 3.34:Interno del blocco Subsystem

Il blocco Cal luce_offset è un filtro passa basso (Fig. 3.35).

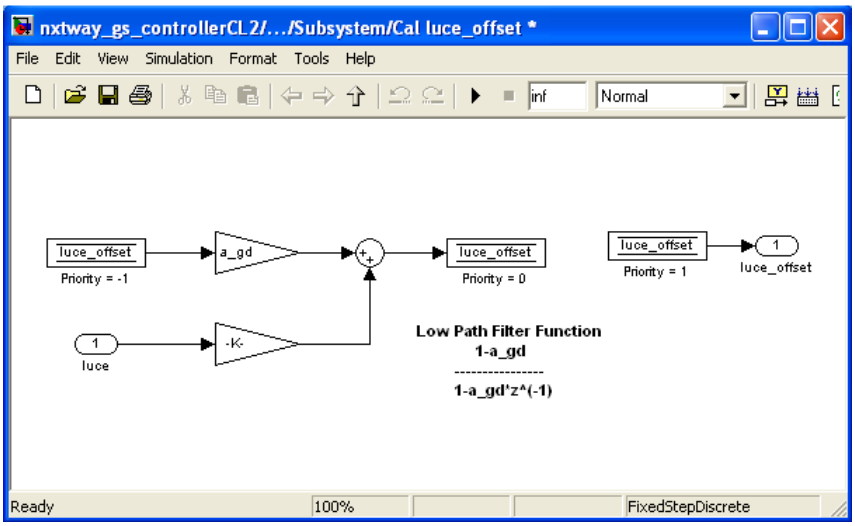


Figura 3.35: Blocco per il calcolo dell'offset del sensore di luce

Il Subsystem contiene anche un filtro passa basso, i cui coefficienti sono stati determinati in modo empirico (Fig. 3.36).

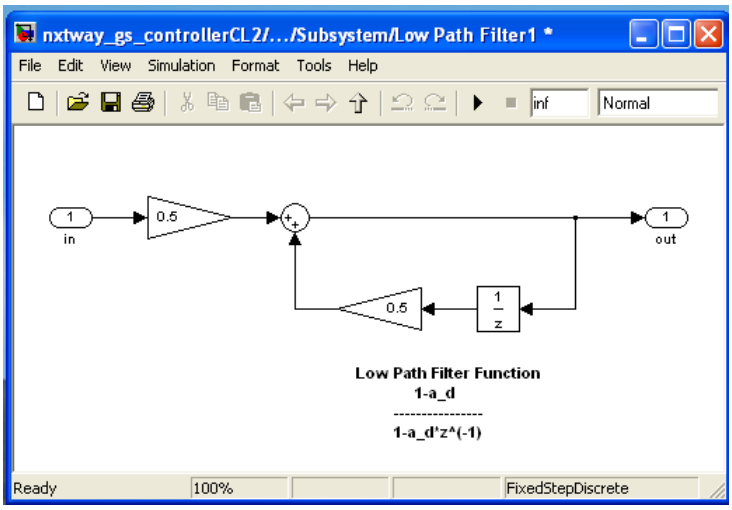


Figura 3.36: Filtro passa basso contenuto nel blocco Subsystem

Analizziamo le modifiche apportate al blocco Cal x1 (Fig. 3.37). Mentre il giroscopio forniva la velocità angolare, il sensore di luce fornisce l'angolo ψ. Per ottenere la velocità angolare è necessario inserire un derivatore discreto.

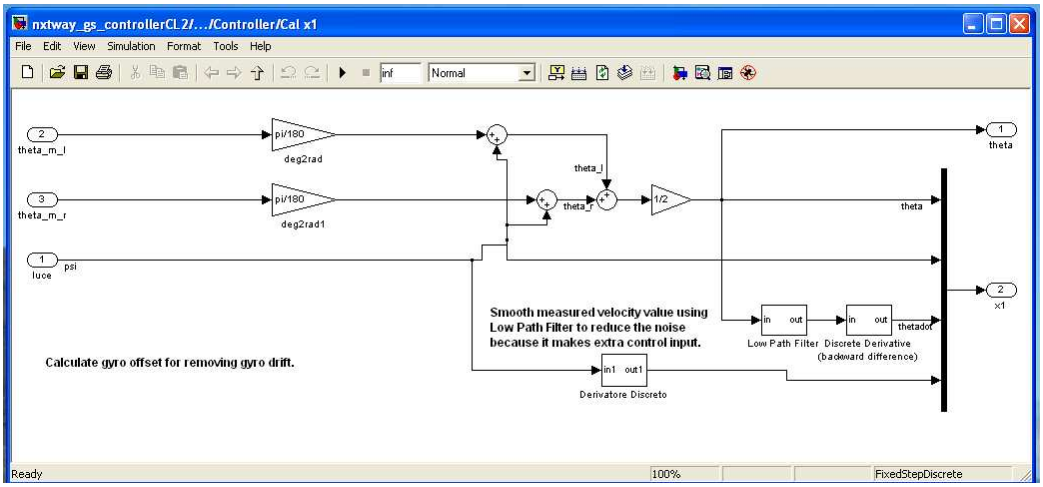


Figura 3.37: Interno del blocco Cal x1

La Fig. 3.38 mostra tale derivatore.

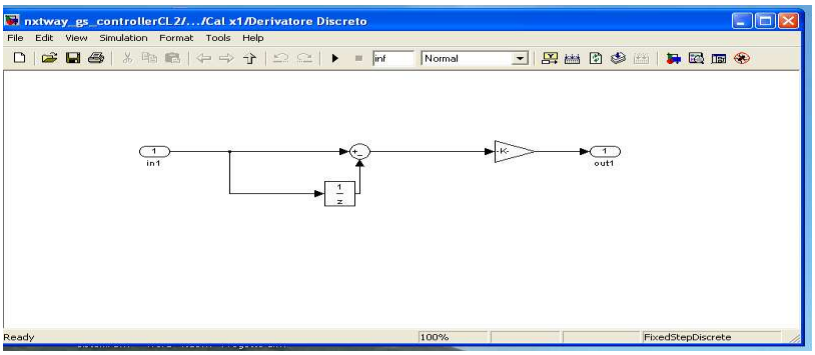


Figura 3.38: Screenshot dell'interno del blocco Derivatore Discreto

3.5.2 CONTROLLO EFFETTUATO MEDIANTE DUE SENSORI DI LUCE:

Questo tipo di controllo (Fig. 3.39) rappresenta un'evoluzione rispetto al modello con un unico sensore di luce. In ingresso al controllore abbiamo i valori forniti da due Light Sensor Read ed elaborati dal blocco Subsystem (luce e luce1).

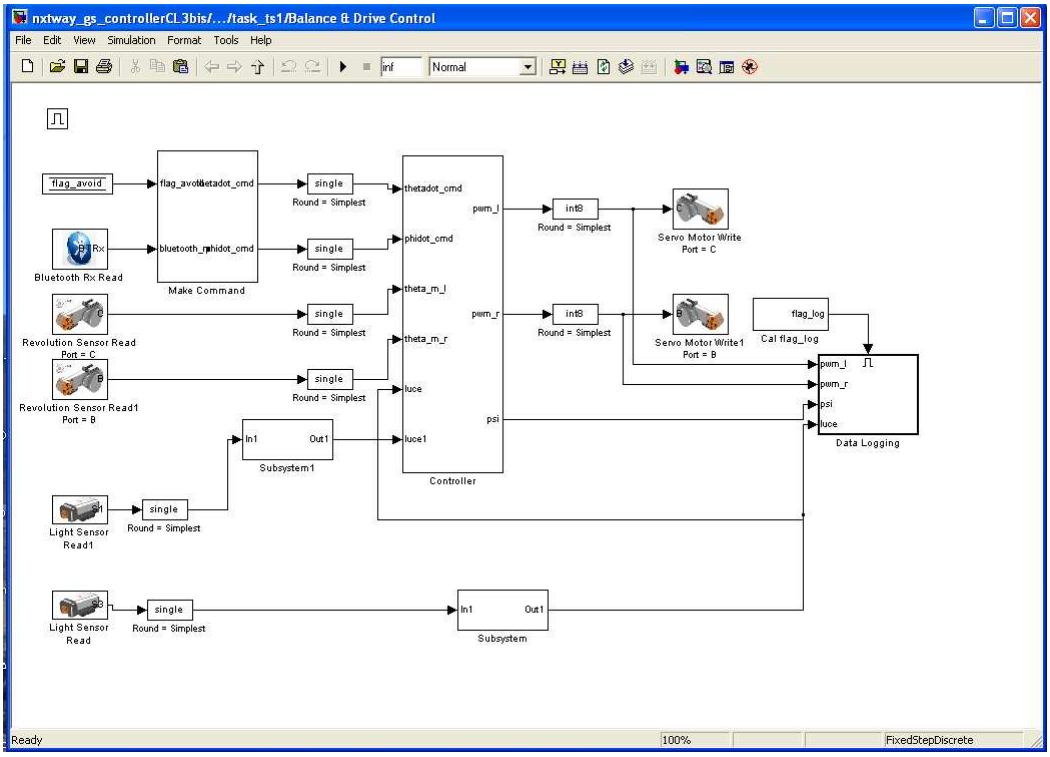


Figura 3.39: Screenshot del blocco Balance & Drive dopo l'aggiunta di due sensori di luce

Quest'ultimo blocco è uguale a quello utilizzato nell'NXTway con un solo sensore di luce.

Il blocco Cal x1 prende in ingresso i valori luce e luce1(Fig. 3.40). Luce1 viene derivato e sottratto a pigreco, luce viene soltanto derivato. Dei due valori così ottenuti viene fatta la media, la quale rappresenta una stima dell'angolo ψ più precisa rispetto a quella ottenuta con un unico sensore.

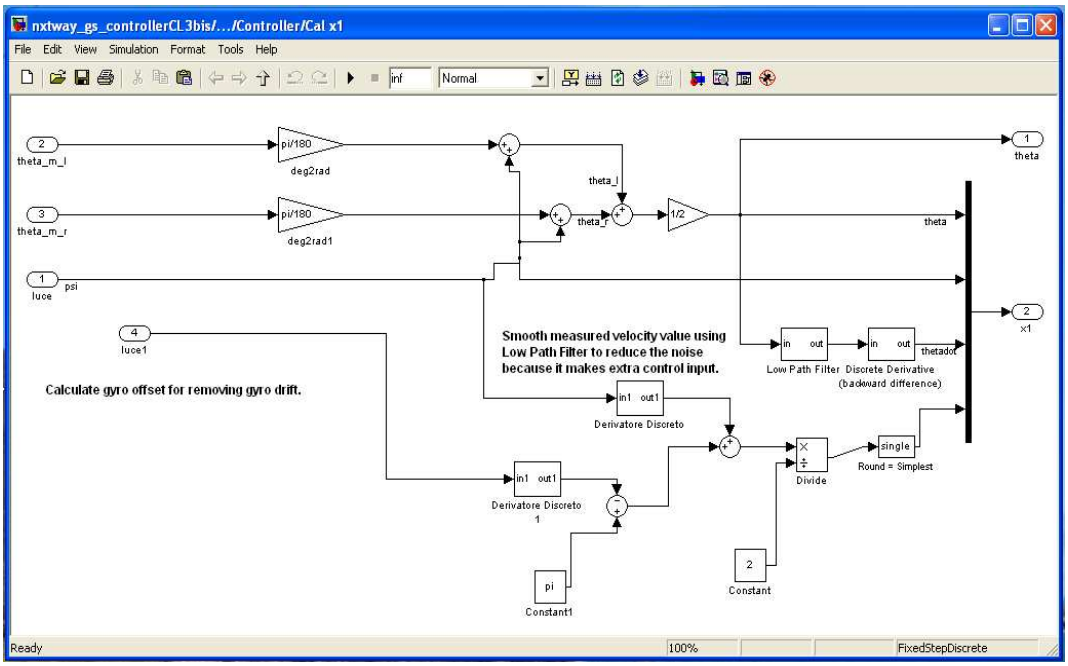


Figura 3.40: Screenshot del blocco Cal x1 dopo l'aggiunta di due sensori di luce

3.5.3 DATA LOGGING DALL'ULTRASUONI E DAL SENSORE DI LUCE:

In ingresso al data logging abbiamo aggiunto sia l'input dell'ultrasuoni sia quello del sensore di luce (sia nell'NXTway_LS che nell'NXTway_LSS) come in Fig. 3.41.

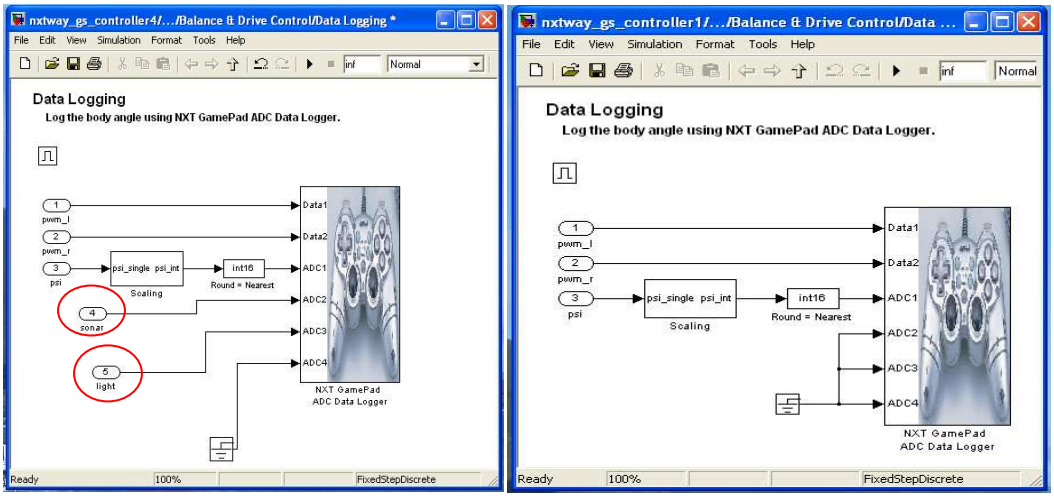


Figura 3.41:Confronto tra lo screenshot del blocco Data Logging con e senza sensore di luce e ultrasuoni.

Per ottenere l'input del sensore ad ultrasuoni è necessario prelevare il valore letto dall'Ultrasonic Sensor Read nel Remote Control Drive (Fig. 3.42-3.43-3.44).

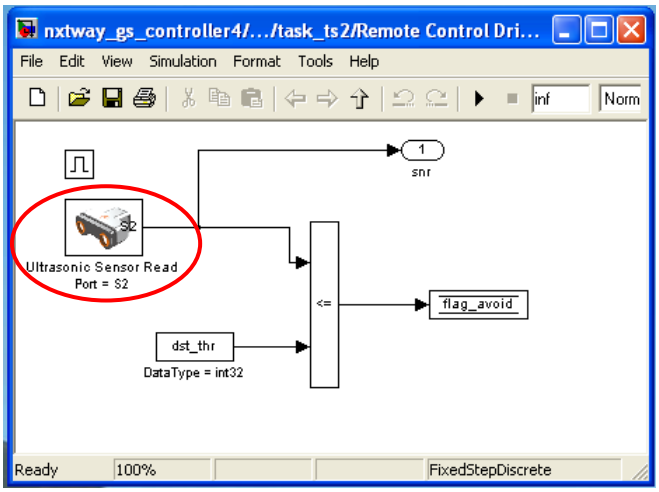


Figura 3.42: Screenshot del blocco Remote Control Drive modificato per acquisire dati dal senosore ad ultrasuoni

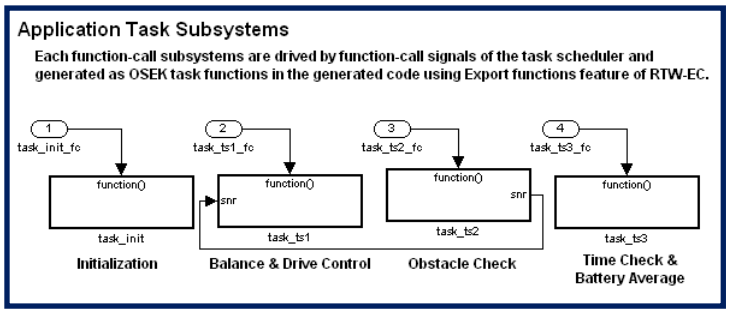


Figura 3.43: Interno del blocco Application Task Subsystems modificato per acquisire dati dal sensore ad ultrasuoni

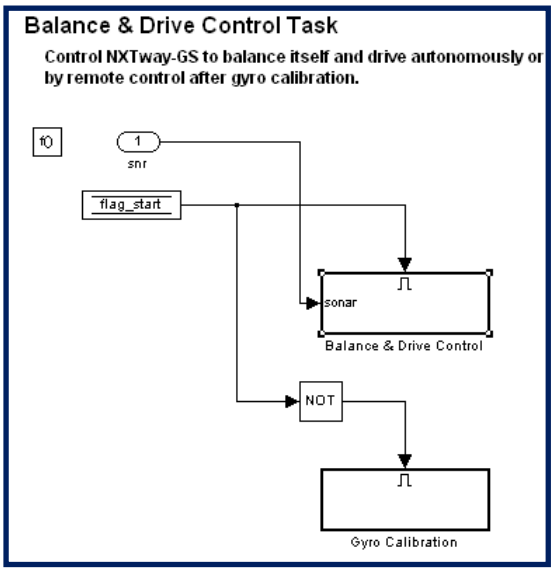


Figura 3.44 Interno del blocco Balance & Drive Control Task modificato per acquisire dati dal sensore ad ultrasuoni

Abbiamo utilizzato tale valore come uscita dal task_ts2 e come ingresso per il task_ts1. Poi lo abbiamo inviato in ingresso al Balance & Drive Control e infine è convertito in un intero.

L'input del sensore di luce si ottiene inserendo un blocco Light Read nel Balance & Drive Control e trasformando il valore ottenuto in un intero.

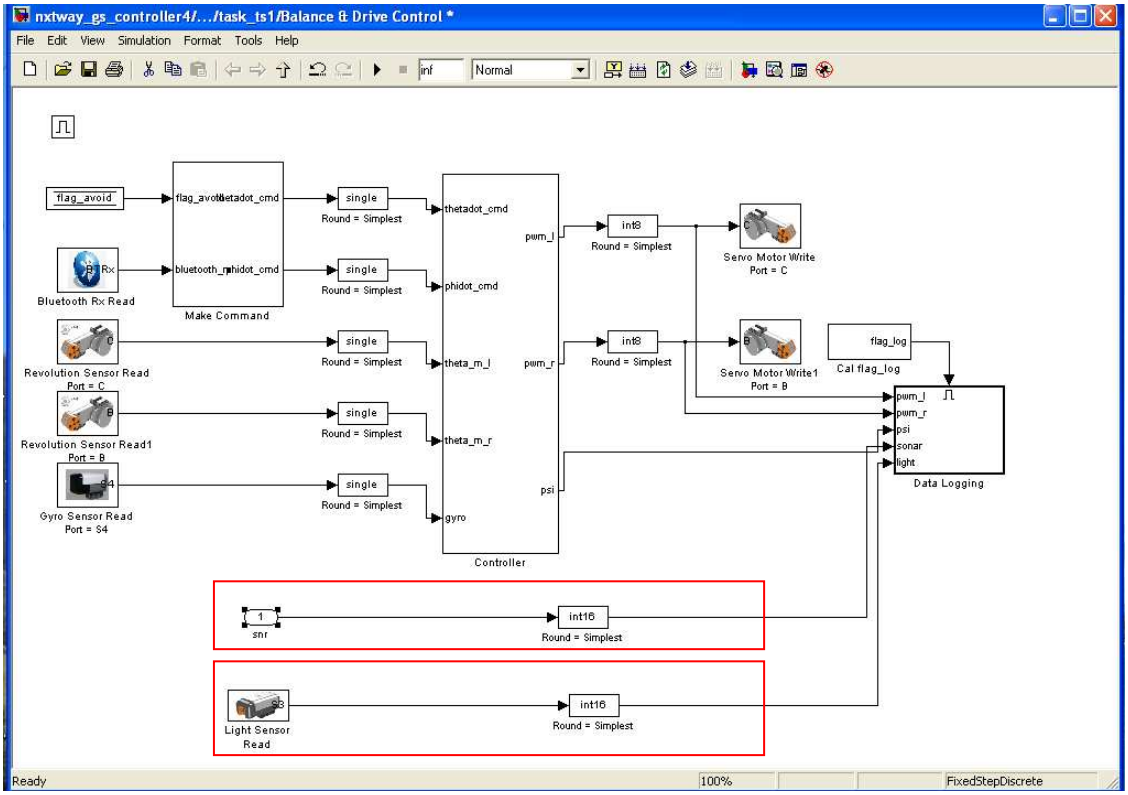


Figura 3.45 Screenshot del Blocco Balance & Drive Control modificato per acquisire dati dal senosore ad ultrasuoni

4 RISULTATI SPERIMENTALI

Grazie all'impiego del blocco Data Logging abbiamo potuto sia modificare alcuni parametri sperimentali del sistema, sia studiare alcune relazioni tra le variabili presenti.

Le seguenti prove sono state effettuate con un NXT dotato di due servomotori, del sensore ad ultrasuoni, di sue sensori di luce e del giroscopio.

PROVA 1:

Il controllo dell'NXT è stato effettuato solo mediante il giroscopio. I valori rilevati della velocità angolare sono stati integrati, in modo da poter misurare l'angolo ψ (Fig. 4.1). Così si è potuto valutare lo scostamento dalla verticale dell'NXT nel tempo.

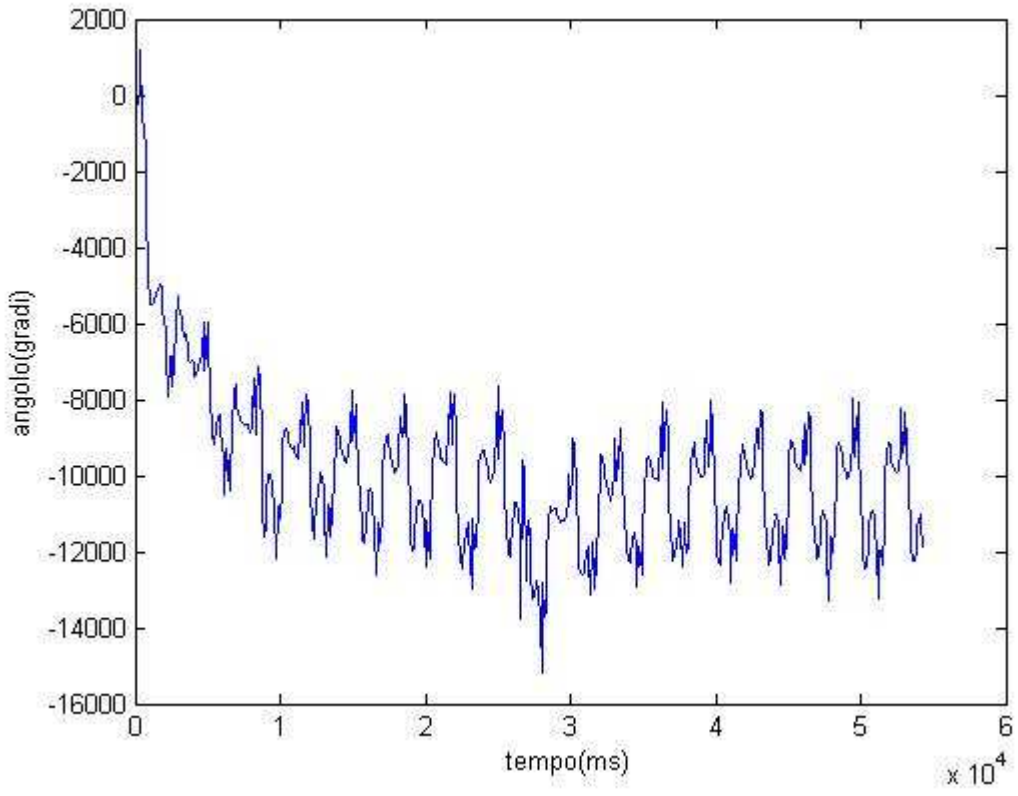


Figura 4.1: Diagramma della variazione dell'angolo in funzione del tempo

Come si vede dalla Fig. 4.2 è presente un fenomeno di drift, ovvero con il passare del tempo l'angolo tende ad allontanarsi sempre di più dalla posizione di equilibrio. Per accertarci di ciò abbiamo deciso di staccare fisicamente i motori.

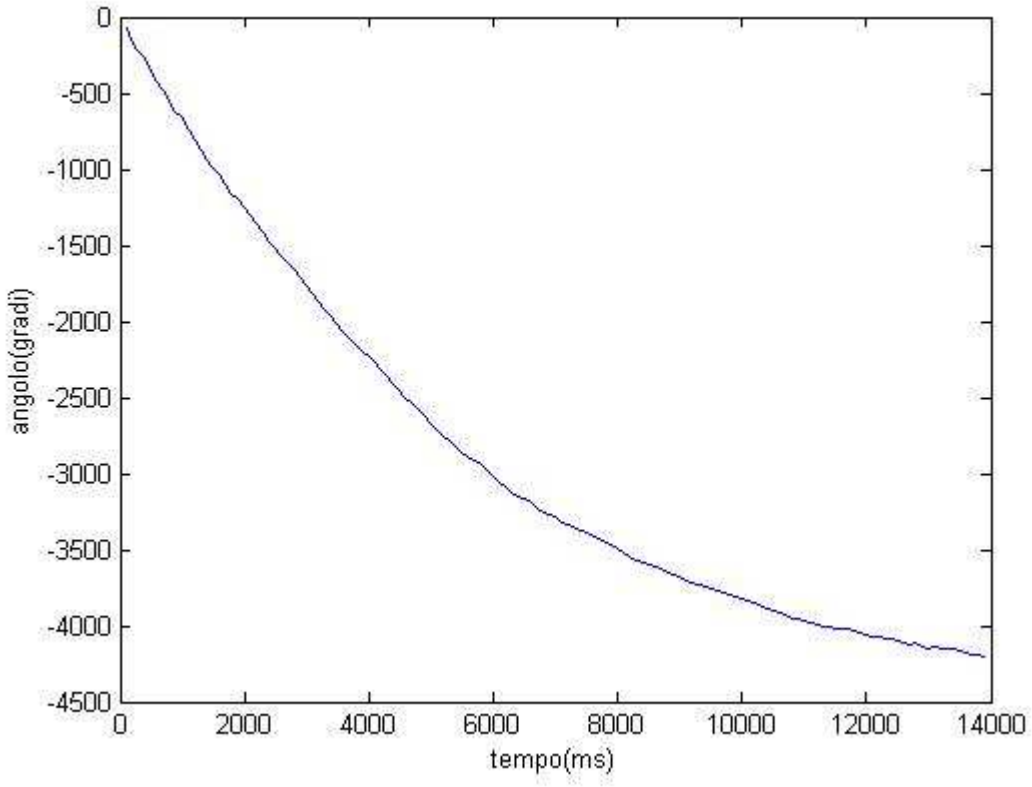


Figura 4.2: Diagramma della variazione dell'angolo in funzione del tempo dopo aver staccato fisicamente i motori

La Fig. 4.2 mostra che l'integratore introduce un fenomeno di deriva. Questo è dovuto sia al fatto che l'integratore ha un numero finito di coefficienti sia al fatto che l'NXT non lavora in virgola mobile.

Per risolvere il problema del drift abbiamo modificato lo schema a blocchi dell'integratore (Fig. 4.3):

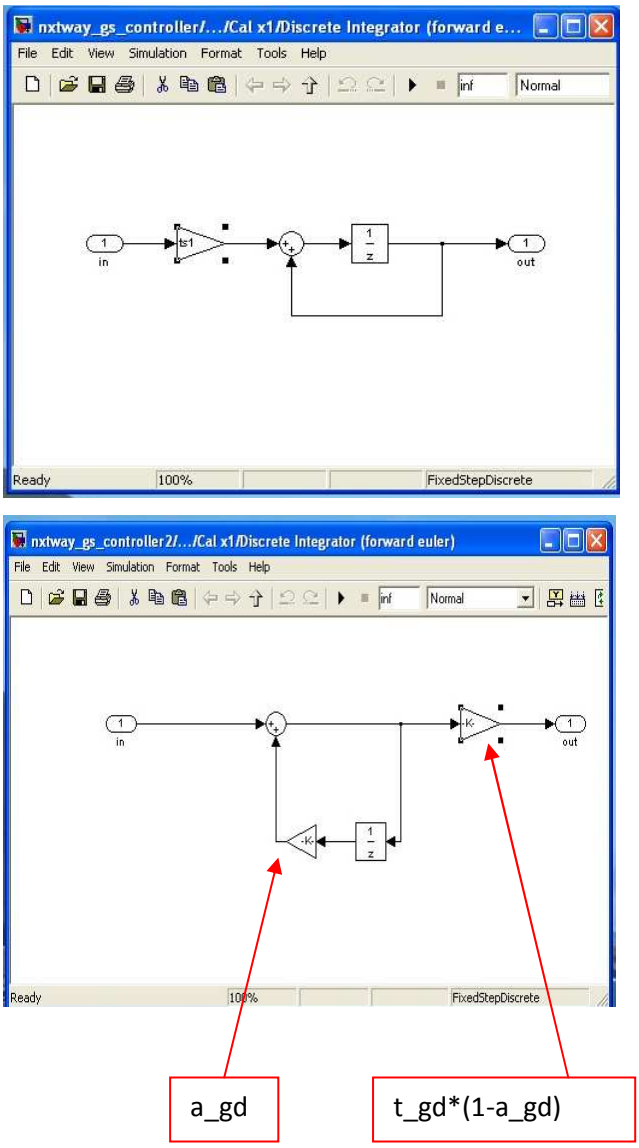


Figura 4.3: Modifiche apportate all'integratore

L'equazione del filtro raffigurato in Fig. 4.3 è la seguente:

$$\frac{t_{gd}\times(1-a_{gd})}{1-a_{gd}\times z^{-1}}$$

Questa funzione di trasferimento rappresenta un derivatore arrestato, che deriva solo a basse frequenze.

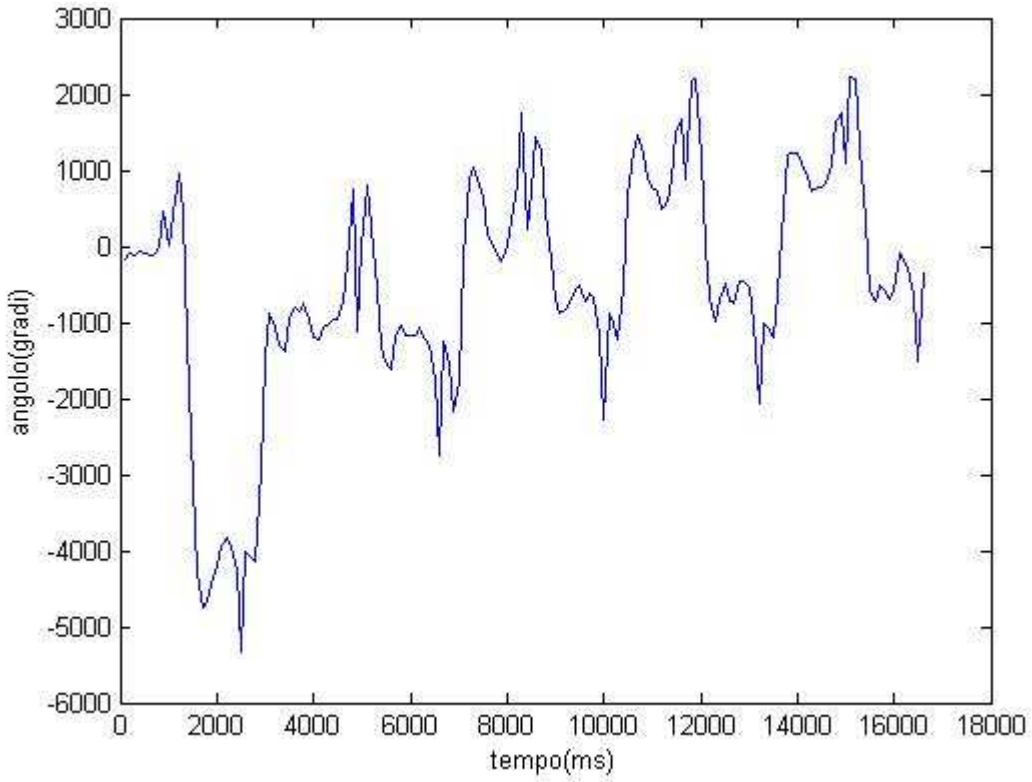


Figura 4.4: Grafico della variazione angolare in funzione del tempo dopo le modifiche apportate

Come si vede dalla Fig. 4.4, siamo riusciti a ridurre notevolmente il fenomeno di drift. Solo adesso possiamo considerare il valore di ψ, calcolato con il giroscopio, come riferimento.

PROVA 2:

Abbiamo inserito un generatore di onda quadra all'interno del blocco Cal Reference (Fig. 4.5) per confrontare i valori dell'angolo ψ ottenuti dal sensore di luce con quelli ottenuti integrando il valore della velocità angolare misurata dal giroscopio.

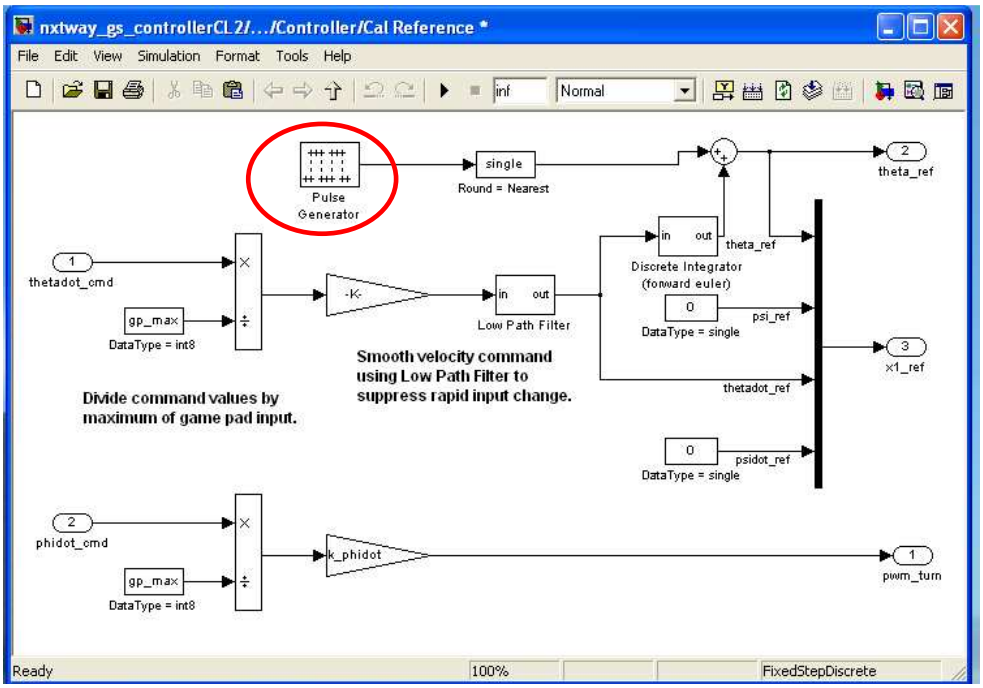


Figura 4.5: Modifiche apportate al blocco Cal Reference

Abbiamo modificato i valori dei coefficienti di luce offset e del Low Path Filter presente nel Subsystem fino ad ottenere la relazione mostrata in Fig. 4.6 tra l'angolo ψ misurato con il giroscopio e quello misurato con il sensore di luce:

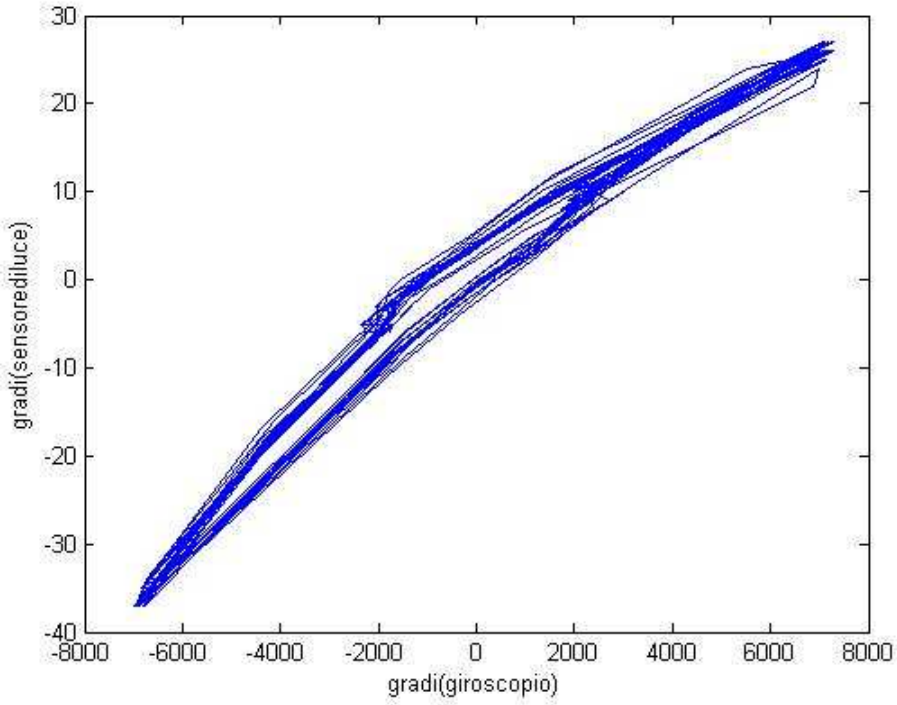


Figura 4.6: Relazione tra l'angolo ψ misurato con il giroscopio e con il sensore di luce

Dalla Fig. 4.6 si nota che è presente un'isteresi. Tale andamento può essere linearizzato prendendo un valore prossimo all'estremo inferiore (-6722; -36) e un valore prossimo all'estremo superiore (7294;27). Il fattore di scala k= 0,0045 permette di ottenere un andamento lineare.

PROVA 3:

A questo punto è stato possibile effettuare il controllo sulla base dei valori misurati dal sensore di luce.

Abbiamo eseguito tre prove su un tavolo bianco, in condizioni diverse:

- Lontano dalla finestra (Fig.4.7):

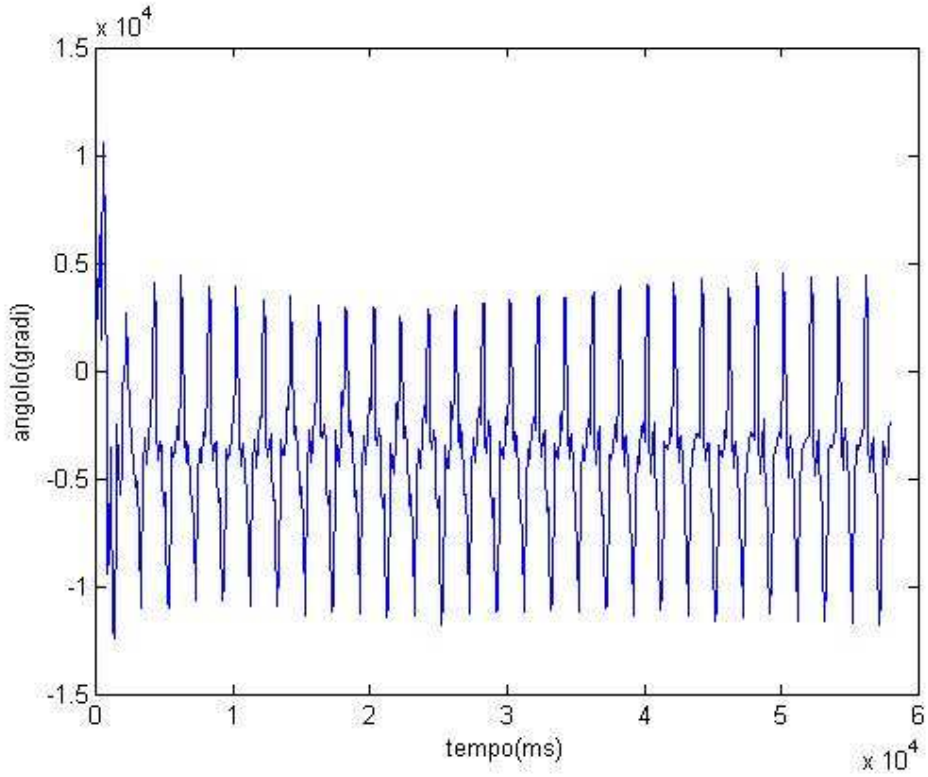


Figura 4.7: Variazione dell'angolo in funzione del tempo

Come possiamo vedere dalla Fig. 4.7 la variazione dell'angolo, nel tempo, rispetto alla verticale è molto piccola (±1°). Il Legway sta in equilibrio.

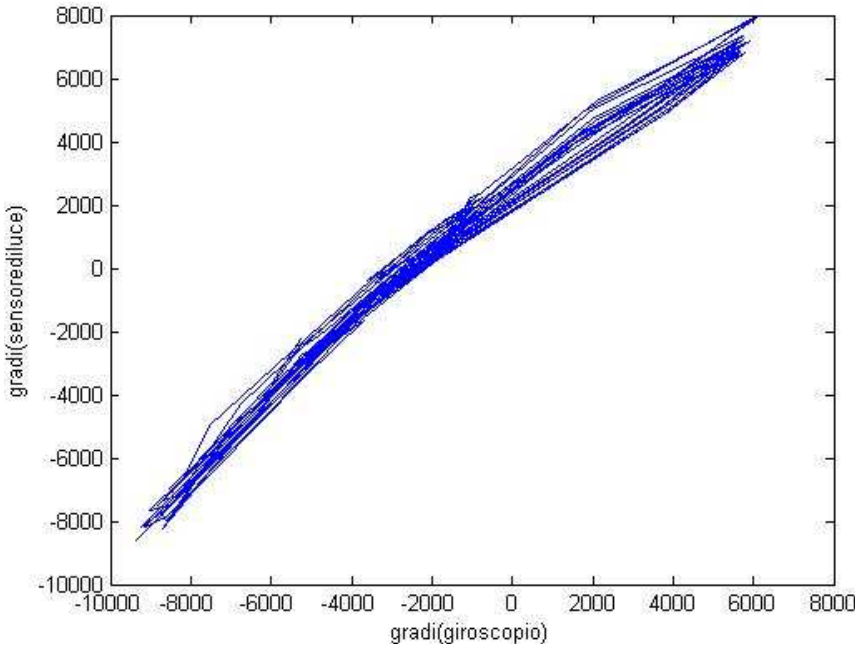


Figura 4.8: Relazione tra l'angolo ψ misurato con il giroscopio e con il sensore di luce

La Fig. 4.8 mostra che l'angolo misurato dal sensore di luce si avvicina molto al valore rilevato dal giroscopio.

- Vicino alla finestra:

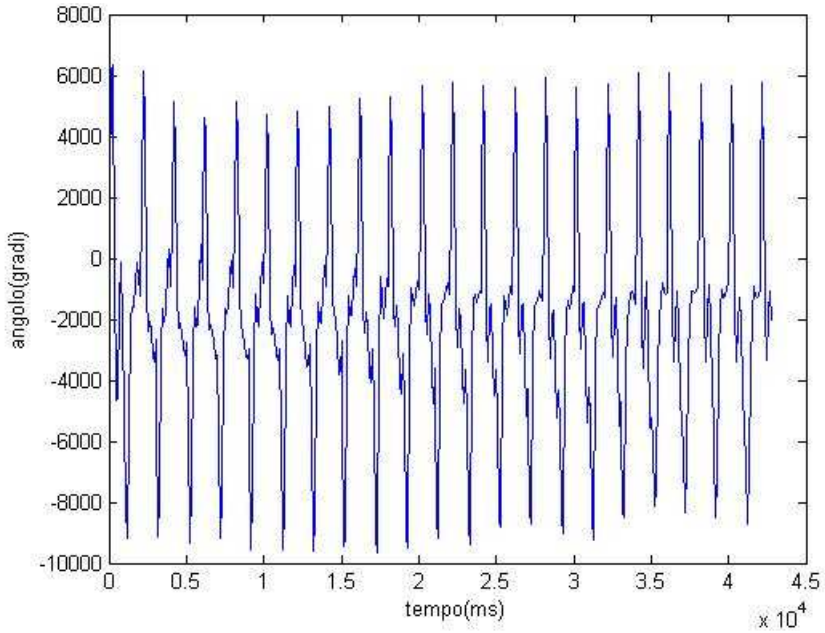


Figura 4.9: Variazione dell'angolo in funzione del tempo

Come possiamo vedere dalla Fig. 4.9 la variazione dell'angolo, nel tempo, è aumentata rispetto all'esperimento effettuato in un luogo oscuro (-8°;6°). Questo è dovuto all'estrema sensibilità del sensore di luce. L'equilibrio del Legway è più precario rispetto alla prova precedente.

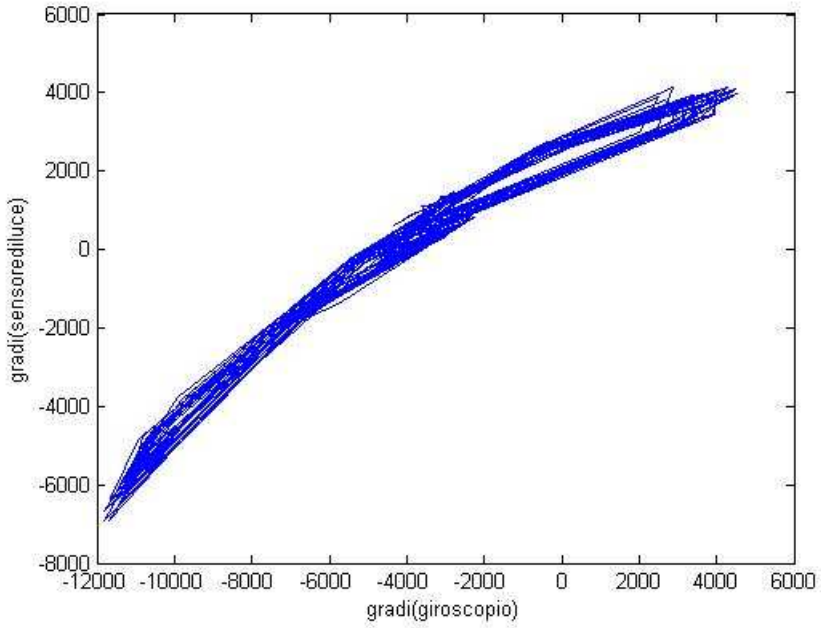


Figura 4.10: Relazione tra l'angolo ψ misurato con il giroscopio e con il sensore di luce

La Fig. 4.10 mostra che l'angolo misurato dal sensore di luce è diverso dal valore rilevato dal giroscopio. Per tale motivo il Legway non sta molto in equilibrio.

- Lontano dalla finestra, ma con la luce accesa:

Come possiamo vedere dalla Fig. 4.11 la variazione dell'angolo, nel tempo, non è molto elevata (-1°;1°). L'equilibrio del Legway è più precario rispetto alla prova precedente.

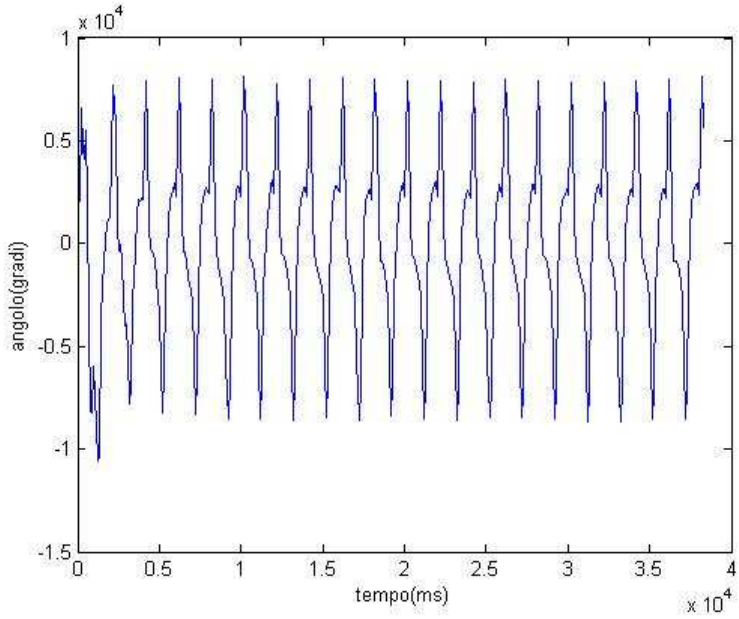


Figura 4.11: Variazione dell'angolo in funzione del tempo

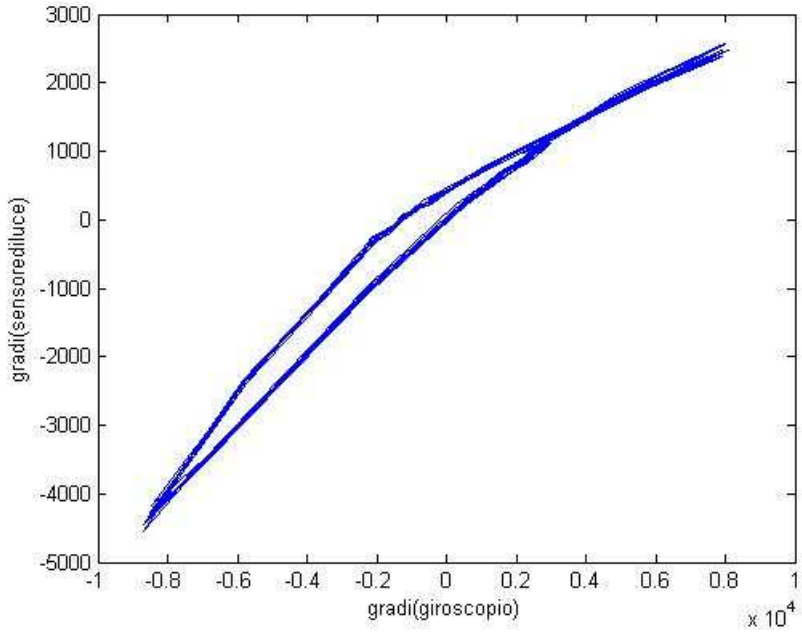


Figura 4.12: Relazione tra l'angolo ψ misurato con il giroscopio e con il sensore di luce

La Fig. 4.12 mostra che l'angolo misurato dal sensore di luce è diverso dal valore rilevato dal giroscopio, soprattutto allontanandosi dall' angolo 0°. Per tale motivo il Legway non sta molto in equilibrio.

Confrontando i valori dell'angolo misurato in un ambiente illuminato (rosso) con quelli ottenuti in un ambiente oscuro (blu), abbiamo cercato di determinare un valore unico per il parametro k, situato all'interno del blocco Subsystem, che potesse garantire la stabilità dell'NXTway_LS indipendentemente dalle condizioni atmosferiche (Fig. 4.13).

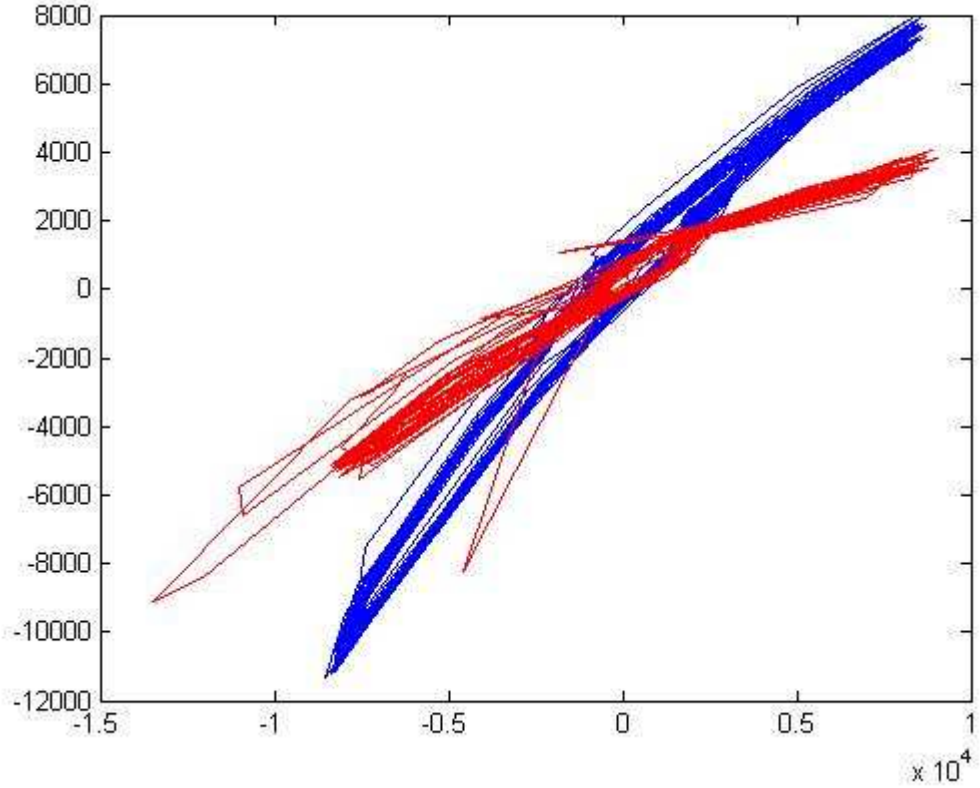


Figura 4.13: Relazione tra l'angolo ψ misurato con il giroscopio e con il sensore di luce in un ambiente illuminati (rosso) e in un ambiente scuro(blu)

Con la stima fatta per il parametro k siamo riusciti a mantenere in equilibrio l'NXTway_LS in tutte le superfici uniformi non troppo riflettenti, in condizioni di luce non troppo elevata.

Abbiamo avuto la conferma che il sensore di luce è molto sensibile alle variazioni della luce ambiente e questo rende impossibile la realizzazione di un NXTway_LS che funziona sia in ambienti illuminati che in ambienti oscuri senza dover modificare i parametri del controllore.

Per tale motivo le prestazioni dell' NXTway_GS, in termini di equilibrio, sono notevolmente superiori a quelle dell' NXTway_LS.

La Fig. 4.14 mostra i risultati ottenuti dopo la variazione del parametro k.

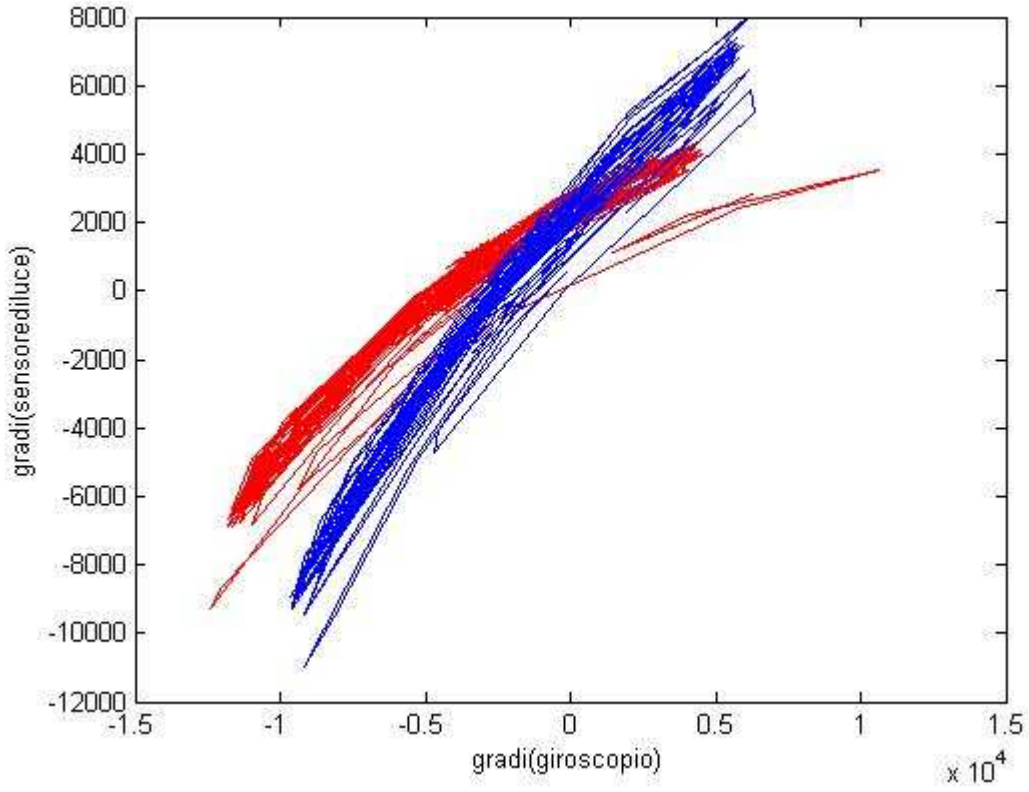


Figura 4.14: Relazione tra l'angolo ψ misurato con il giroscopio e con il sensore di luce in un ambiente illuminati (rosso) e in un ambiente scuro(blu) dopo la variazione del parametro k

PROVA 4:

Infine abbiamo effettuato delle prove con due sensori di luce.

I problemi riscontrati nell'NXTway_LSS sono gli stessi trovati nel caso precedente, poiché la limitazione maggiore è costituita dall'eccessiva sensibilità del sensore di luce.

In un ambiente abbastanza illuminato abbiamo ottenuto il grafico in Fig. 4.15:

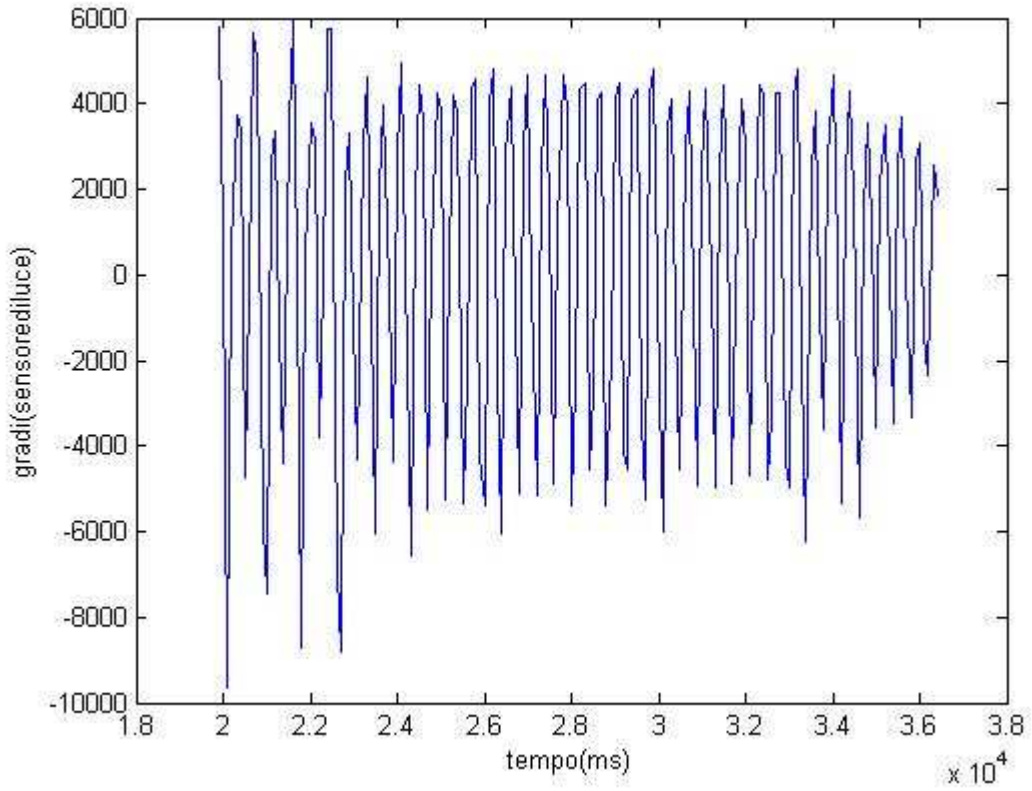


Figura 4.15: Variazione dell'angolo in funzione del tempo

Come possiamo vedere dalla Fig. 4.15 la variazione dell'angolo, nel tempo, non è abbastanza elevata nella fase iniziale e tende a diminuire con il passare del tempo.

Si è poi eseguito una prova su una superficie diversa. Mentre tutte le prove precedenti erano state effettuate su una scrivania bianca, adesso abbiamo posizionato il Legway su un tavolo più scuro, di color marrone nocciola.

La Fig. 4.16 mostra i risultati sperimentali ottenuti in un stanza buia, la 4.17 in una stanza illuminata.

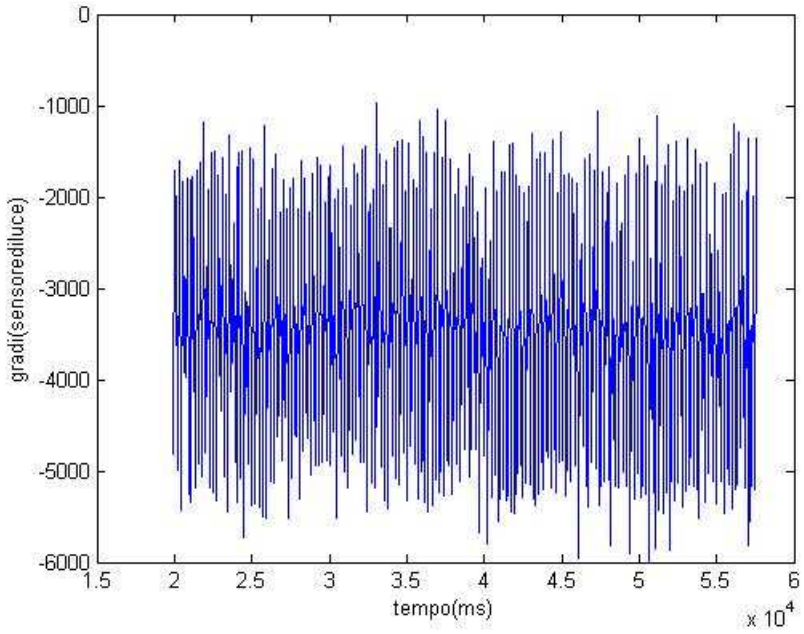


Figura 4.16: Variazione dell'angolo in funzione del tempo

Come possiamo vedere dalla Fig. 4.16 la variazione dell'angolo, nel tempo, è abbastanza elevata e i valori assunti sono tutti negativi (-6;-1). L'equilibrio è comunque abbastanza buono.

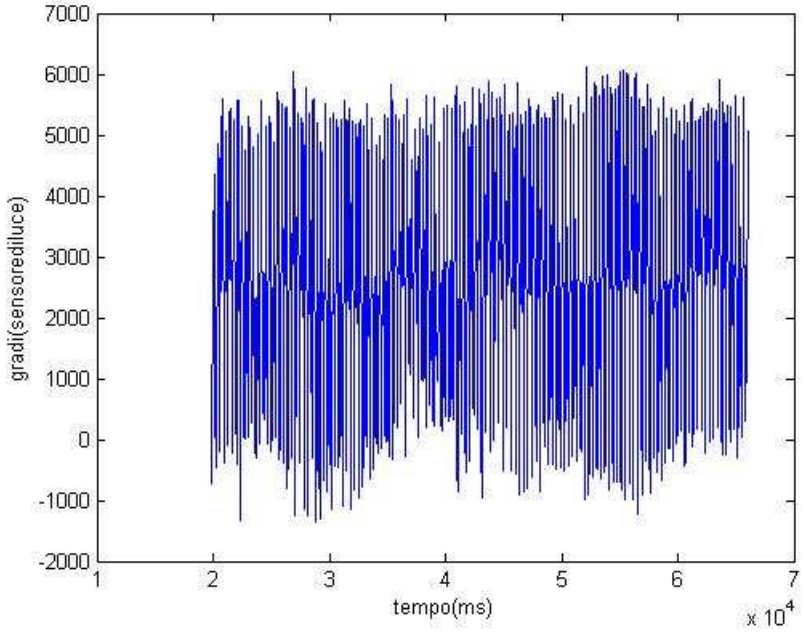


Figura 4.17: Variazione dell'angolo in funzione del tempo

Dalla Fig. 4.17 possiamo notare che la variazione dell'angolo, nel tempo, è abbastanza elevata e i valori assunti sono nel range (-1°;6°). L'equilibrio è più stabile del caso precedente.

Conclusioni

Gli obiettivi che ci eravamo preposti prima di iniziare questa tesi sono stati raggiunti, anche se le idee e le modifiche apportabili sono ancora numerose. La caratteristica innovativa che contraddistingue il nostro lavoro è stata la possibilità di utilizzare Simulink per progettare il controllo e scaricarlo sul Brick. Partendo da un modello di controllo per l' NXTway siamo riusciti a modificarlo per ottenere ciò che volevamo. Nel modello presistente il controllo era effettuato tramite un sensore giroscopio e i dati erano acquisiti solo da questo sensore. Come primo passo abbiamo modificato lo schema a blocchi per poter prelevare i dati dal sensore ad ultrasuoni, che è in grado di rilevare la presenza di ostacoli. In seguito abbiamo aggiunto fisicamente un sensore di luce tramite il quale sono stati effettuati sia il controllo che l'acquisizione di dati. Per ottimizzare la stabilità del robot abbiamo aggiunto un secondo sensore di luce così da calcolare in maniera più precisa l'angolo di spostamento dalla posizione di equilibrio. Prima di modificare lo schema a blocchi del controllore, aggiungendo i sensori di luce, abbiamo dovuto effettuare diverse prove per determinare alcuni parametri di questi sensori: si è introdotto un ingresso virtuale noto ed abbiamo studiato il comportamento del sensore di luce mentre il controllo veniva effettuato dal sensore giroscopio. Questa prova è stata svolta sia in zone di luce che in zone d'ombra in modo da poterne studiare le caratteristiche in condizioni diverse, alla ricerca di un modello unico. Dopo aver calibrato il sensore di luce abbiamo modificato lo schema a blocchi del controllore, prima con l' aggiunta di un solo sensore di luce, poi con due. In queste due situazioni abbiamo raggiunto risultati simili e abbastanza soddisfacenti nonostante la sensibilità di questi sensori.

Un obiettivo raggiungibile in futuro è il controllo dell' NXTway tramite un gamepad o tramite un cellulare compatibile. Per quanto riguarda la comunicazione bluetooth sarebbe molto interessante riuscire a realizzare una connessione tra più NXT. Un NXT può realizzare tre connessioni e per questo si può immaginare di costruire dei gruppi di robot che comunicano tra di loro.

Una caratteristica della linea LEGO Mindostorms è la possibilità di poter dare libero sfogo alla propria fantasia sia per la presenza di moltissimi sensori forniti (

alcuni dei quali prodotti dall' HiTechnic) o realizzati dall'utente (attraverso tutti gli strumenti messi a disposizione per costruirne nuovi) sia per gli innumerevoli software che esistono per programmarli.

Bibliografia

Philippe E. Hurbain homepage:

http://www.philohome.com/nxt.htm

nxtOSEK homepage:

http://lejos-osek.sourceforge.net

[1] MathWorks homepage:

http://www.mathworks.com

[2] Lego education:

http://www.lego.com/education/default.asp

[3] Lego Mindstorms homepages:

http://mindstorms.lego.com/eng/Salzburg_dest/Default.aspx

[4] Analysis of the NXT Bluetooth-Communication Protocol:

http://www.tau.ac.il/~stoledo/lego/btperformance.html

[5] Ryo Watanabe homepage:

http://web.mac.com/ryo_watanabe

[6] Wikipedia homepage:

http://www.wikipedia.org

[7] Bricx Command Center 3.3:

http://bricxcc.sourceforge.net/

[8] RobotC:

http://www.robotc.net/

[9] NXT OnBrick:

http://www.pspwp.pwp.blueyonder.co.uk/science/robotics/nxt/index.html

[10] Robolab:

http://www.lego.com/eng/education/mindstorms/home.asp?pagename=robolab

[11] LabVIEW Toolkit:

http://zone.ni.com/devzone/cda/tut/p/id/4435