FACOLTÀ DI INGEGNERIA

Corso di Laurea in Ingegneria Meccanica

SVILUPPO DI SISTEMI DI CONTROLLO SU PIATTAFORMA LEGO MINDSTORMS

RELATORE:

Ing. Michele Basso .....

CORELATORI:

Dott. Franco Quercioli .....

Dott. Massimo Vassalli .....

CANDIDATI:

Iacopo Finocchi .....

Niccolò Monni .....

Anno Accademico 2006/2007

Indice

Elenco delle figure

Capitolo 1

Introduzione

Questa tesi tratta di esperimenti di controllo sviluppati attraverso l'utilizzo dei soli componenti LEGO. Gli esperimenti si sono svolti presso il Consiglio Nazionale delle Ricerche (CNR) di Firenze, nei laboratori dell'Istituto Sistemi Complessi (ISC). L'esperienze di cui sopra sono state concepite a scopo didattico, al fine di scoprire e testare le potenzialità del nuovo NXT Mindstorms LEGO.

La presenza del LEGO all'interno degli esperimenti del CNR non è un evento casuale; infatti già da molti anni viene posta un'attenzione particolare all'utilizzo di questi componenti, ritenuti di solito esclusivamente ludici, per la realizzazione di prove scentifiche. Già negli anni '90 il lavoro "Play in Optics with LEGO", portato avanti dal laboratorio di ottica dell'ISC, suscitò un interesse tale da ricevere l'attenzione della stampa specialistica e non. Fu presentato al "Fith International Topical Meeting on Education and Training in Optics" e apparve in molte riviste del settore, tra le quali "Opto & Laser Europe", "Daily Telegraph", "The Guardian" nella rubrica online "Technology". Naturalmente lo scopo finale di questi esperimenti non è il semplice divertimento, ma la diffusione della conoscienza scientifica ad un numero più ampio di persone. IL LEGO, infatti, rende possibile una comunicazione a tutti i livelli, da studenti ad appassionati di scienza, non richiede nozioni di base e riesce a coinvolgere e far appassionare l'utilizzatore. La stessa casamadre, seguendo la propria teoria dell'Imparare facendo, promuove l'integrazione del LEGO all'interno di percorsi didattici, proponendo una serie

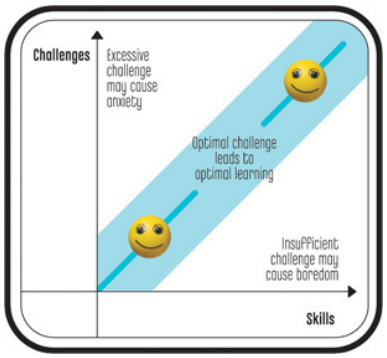


Figura 1.1: Rapporto tra difficoltà ed interesse del ragazzo

di step da seguire insieme ai ragazzi: "Noi riteniamo che il migliore insegnamento avvenga quando i bambini hanno l'opportunità di esplorare il mondo da loro stessi, ma in un ambiente guidato. Costruire attivamente le cose nel mondo della fisica, aiuta il bambino a migliorare le proprie conoscenze. Queste nuove conoscenze rendono capaci i bambini di trovare sempre nuove sofisticate soluzioni ai problemi, creano nuove abilità e più sfide in un sistema auto-rinforzante."[1] Ispirandosi a questo principio, LEGO ha strutturato un percorso adeguato alle varie fasi della crescita del ragazzo, in modo da evitare di incontrare sfide eccessivamente difficili, che possano portare ad un allontanamento dalla scienza.

Il percorso prevede, come già detto quattro fasi:

- *Connect*: In questa fase i ragazzi vengono posti davanti al problema, ed invogliati a risolverlo.
- *Construct*: In questa fase i ragazzi si confrontano, pianificano e costruiscono la soluzione del problema
- *Contemplate*: È una fase molto importante durante la quale i ragazzi esaminano i loro progetti confrontandoli e modificandoli, comprendendo ciò che hanno fatto
- *Continue*:In questa ultima fase si ha l'evoluzione del progetto con l'aggiunta di ulteriori difficoltà, di nuovi quesiti.

[1]Da http://www.lego.com/education/default.asp

Proprio per cercare di realizzare una diffusione ad un livello generale, l' ISC si è posto come obbiettivo la realizzazione di un sito internet, dove collocare tutti i vari esperimenti realizzati, anche quello illustrato in questa tesi. Tale sito dovrà essere dinamico, permettendo all'utente di contribuire al suo sviluppo con la propria creatività, proponendo nuovi esperimenti o alcune modifiche a quelli già esistenti. Questa tesi si propone quindi il ruolo di "apri pista" per una serie di esperimenti di complessità crescente che sicuramente verranno portati avanti nel tempo. Sono, ad esempio, già state poste le basi per far interagire più unità NXT tra loro al fine di studiare il comportamento di più unità simultaneamente.

In questo lavoro sono stati analizzati due tipi di processi fisici: il pendolo di Furuta, una particolare applicazione del pendolo inverso, e il Legway, che prende spunto dal mezzo di locomozione commerciale Segway.

La scelta di queste strutture come argomento di studio, è derivata sia dalla necessità di determinare le prestazioni del LEGO NXT, per il quale risultava necessario un esperimento di non banale soluzione, sia da considerazioni pratiche: il pendolo di Furuta è un oggetto di largo uso nella sperimentazione di metodologie di controllo, esso infatti presenta delle caratteristiche dinamiche relativamente semplici da comprendere (da qui il suo vasto uso anche in ambito didattico) ma tutt'altro che banali da regolare (motivando la sua presenza come "test bed" anche in molti laboratori di ricerca). Il pendolo di Furuta è presente nel laboratorio di automazione di Santa Marta. Risulta interessante cercare di ricreare lo stesso esperimento attraverso il solo utilizzo del LEGO che presenta sia un utilizzo più semplice, sia un costo più contenuto.

Il Legway è risultato la scelta più logica dopo il fallimento dell'esperimento sul pendolo; presenta infatti un comportamento e una dinamica similari a quelli del pendolo stesso, ma presenta una struttura dinamica notevolmente semplificata.

La tesi è strutturata come segue:

Capitolo 2 Si esamina l'hardware LEGO Mindstorms NXT in tutte le sue parti.

Capitolo 3 Si descrivono i principali software disponibili per gestire l'NXT.

Capitolo 4 Si descrive il tentativo di realizzazione del pendolo di Furuta, distinguendo le due strutture realizzate, i programmi utilizzati e il motivo per cui l'esperimento è stato abbandonato.

Capitolo 5 Si analizza la struttura meccanica costruita, il comportamento dinamico, le problematiche derivanti dall'hardware a disposizione e tutte le altre fasi che hanno portato alla realizzazione del Legway.

Capitolo 2

NXT Mindstorms

Nell'agosto del 2006 LEGO ha lanciato sul mercato un articolo atto a divenire più uno strumento istruttivo utilizzabile nelle varie scuole, che un vero e proprio giocattolo: il LEGO Mindstorm NXT. Questo nuovo sistema ridefinisce la categoria di robot creati a partire dal 1998, dal ramo robotico della compagnia, introducendo nuove possibilità di interazione e di sviluppo per i consumatori. La prima generazione di robot venne definita LEGO Mindstorm RCX (fig 2.1).

Figura 2.1: LEGO Mindstorm RCX

Le caratteristiche erano nettamente inferiori rispetto al nuovo NXT (classe del processore, modalità di comunicazione, qualità sensori, velocità) ma ha avuto l'importante ruolo di apri pista, per un futuro sviluppo. "Con il lancio di LEGO MINDSTORMS (1998), abbiamo cambiato radicalmente il modo in cui le persone concepiscono e giocano con le costruzioni LEGO e abbiamo aiutato la propagazione di un livello accessibile e raggiungibile per gli utilizzatori di robot.", dice Jørgen Vig Knudstorp(CEO di LEGO Group), "Otto anni più tardi, siamo pronti a migliorare il singolo prodotto più venduto nella storia dell'azienda. Abbiamo sviluppato il nuovo set di strumenti, fornendo nuovi componenti e capacità con cui proporre sfide alle nostre dinamiche comunità di fan. Inoltre il nuovo software, il nuovo design e la possibilità di creare un robot funzionante in appena 30 minuti, hanno avvicinato una nuova generazione di giovani appassionati di robotica".

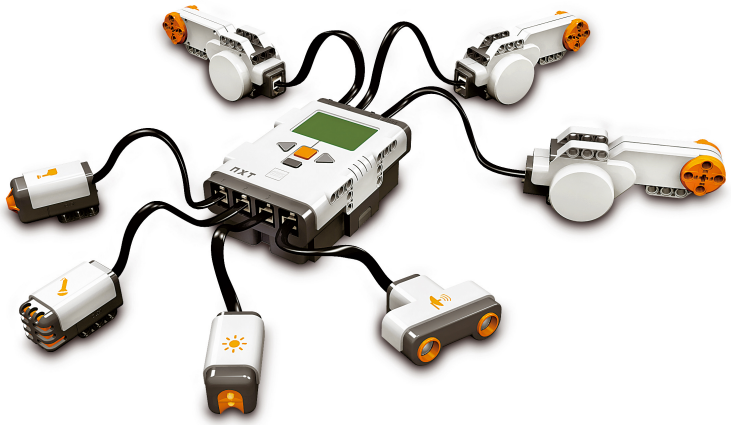


Figura 2.2: LEGO Mindstorm NXT

Il fatto che il settore della robotica sia quello che ha registrato il maggior numero di vendite nella storia della LEGO, può dare un'idea di quanto sia estesa l'applicazione di questi prodotti, sia in percorsi didattici che per uso privato. La rete stessa è una conferma evidente a tale fenomeno; basta infatti digitare Mindstorm NXT su un qualsiasi motore di ricerca per accedere ad un numero elavatissimo di blog, siti privati, siti appartenenti a scuole o università, sezioni derivate dalla stessa LEGO, che riportano esperimenti, test sui sensori, nuovi software per la programmazione, etc.

LEGO Mindstorm NXT

Il kit base, fornito da LEGO è composto da:

- una unità di calcolo intelligente
- tre attuatori
- quattro sensori(ottico, sonoro, ultrasuoni, pressione)
- 519 celebri "mattoncini", appartenenti alla serie LEGO TECHNIC.
- software intuitivo per la progettazione e la stesura di programmi di controllo NXTG

2.1 Dati tecnici

2.1.1 Intelligent Brick

Di seguito riportiamo le specifiche tecniche del brick NXT:

- processore a 32 bit Atmel AT91SAM7S256 (classe ARM7) a 48 MHz
- 256KB memoria flash
- 64KB RAM
- Interfaccia bluetooth v2.0+EDR (chipset CSR BlueCore 4 versione 2, clockato a 26 MHz, con propri buffer RAM e firmware stack Bluelab 3.2) velocità teorica massima 0,46 Mbit/sec (per trasferire il software o per controllare il robot da remoto);
- Display LCD bianco e nero da 100x64 pixel (il pixel è circa 0,4x0,4mm);
- speaker mono 8 bit fino a 16 KHz;
- tastiera con quattro tasti in gomma;
- Porta USB 2.0;
- Interfaccia per permettere lo sviluppo di periferiche da parte di terze parti;

2.1.2 Attuatori

Il kit base fornito da LEGO prevede 3 servo motori del peso di 60g ciascuno, le cui caratteristiche meccaniche, dalle prove che abbiamo eseguito, risultano essere discrete. Ogni motore ha al sul interno un encoder (sensibilità ± 1 grado) che assicura il corretto posizionamento del rotore.

Figura 2.3: Encoder motore

Come qualsiasi altro motore elettrico possiamo individuare le equazioni che governano la sua dinamica.

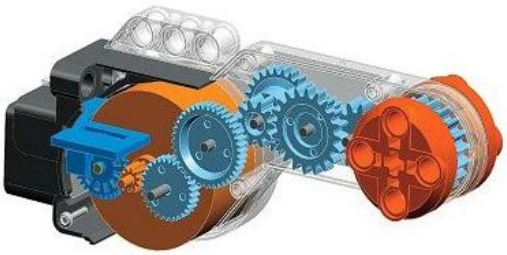


Figura 2.4: Interno motore

Di seguito riportiamo un calcolo delle caratteristiche elettriche e meccaniche dei motori:

Equazione del circuito

$$\begin{cases} L_a \frac{di_a}{dt} + R_a i_a + e_b = e_a \\ e_b = K_b \omega \end{cases}$$

Equazione di moto

$$\begin{cases} J\frac{d\omega}{dt} + B\omega = \tau_a - \tau_d \\ \tau_a = K_\tau i_a \end{cases}$$

In condizioni stazionarie($\omega = costante$ e $i_a = costante$) queste equazioni differenziali si semplificano nelle seguenti equazioni lineari:

$$\begin{cases} R_a i_a + K_b \omega = e_a \\ K_\tau i_a - B\omega = \tau_d \end{cases}$$

In forma matriciale:

$$\begin{bmatrix} R_a & K_b \\ K_\tau & -B \end{bmatrix} \begin{bmatrix} i_a \\ \omega \end{bmatrix} = \begin{bmatrix} e_a \\ \tau_d \end{bmatrix}$$

Da esperimenti risulta che:

$$\begin{bmatrix} i_a \\ \omega \end{bmatrix} = \begin{bmatrix} 7.2115 \times 10^{-3} & 2.9949 \\ 2.0294 & -43.840 \end{bmatrix} \begin{bmatrix} e_a \\ \tau_d \end{bmatrix}$$

Eseguendo i calcoli:

$$\begin{bmatrix} e_a \\ \tau_d \end{bmatrix} = \begin{bmatrix} 7.2115 \times 10^{-3} & 2.9949 \\ 2.0294 & -43.840 \end{bmatrix}^{-1} \begin{bmatrix} i_a \\ \omega \end{bmatrix}$$

$$= \begin{bmatrix} 6.8562 & 0.46838 \\ 0.31739 & -0.0011278 \end{bmatrix} \begin{bmatrix} i_a \\ \omega \end{bmatrix}$$

Dunque abbiamo una stima dei parametri del motore:

$$R_a = 6.8562(\Omega), \tag{2.1}$$

$$K_b = 0.46839(V/(rad/s)), \tag{2.2}$$

$$K_\tau = 0.31739(N \cdot m/A), \tag{2.3}$$

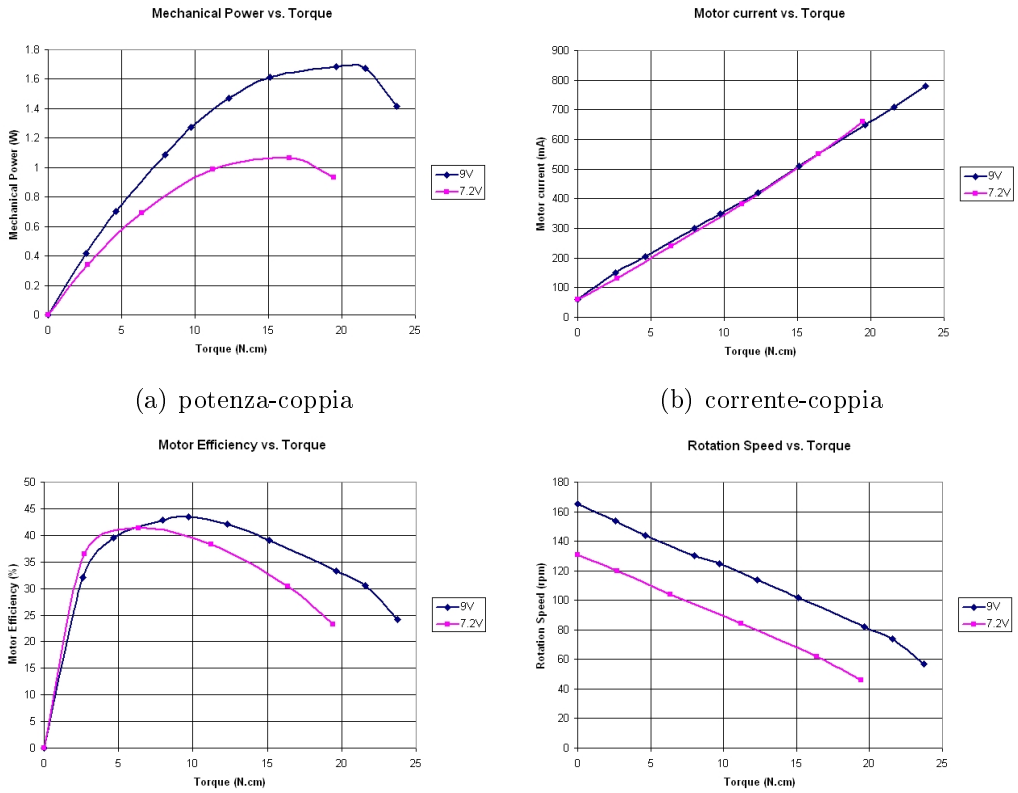
$$B = 1.1278 \times 10^{-3}(N \cdot m/(rad/s)), \tag{2.4}$$

Riassumiamo in tabella le principali specifiche:

La velocità massima e la corrente in assenza di carico sono rispettivamente 170 giri/min e 60 mA.

	
V fornita	9V
n max a vuoto	170 giri/min
I a vuoto	60 mA
I stallo	2 A
Coppia di stallo	50 Ncm

Sono state studiate anche le caratteristiche di risposta dei motori, sia quando l'NXT è alimentato con batterie alcanile oppure con al batteria fornita a parte (NiMH) (fig 2.5).



(a) potenza-coppia

(b) corrente-coppia

(c) efficienza-coppia

(d) velocità-coppia

Figura 2.5: Carattarestiche dei servomotori

2.1.3 Sensori

Il kit fornito da LEGO contiene 4 sensori :

- pressione
- luce
- ultrasuoni
- suono

Sensore di pressione

È il sensore più semplice e conferisce all' NXT il senso del tatto. Esso si attiva nel caso in cui il pulsante venga premuto (o rilasciato).

Figura 2.6: Sensore di pressione

Sensore di luce

É uno dei due sensori (insieme a quello ultrasonico) che dà il senso della vista all'NXT. È in grado di:

- Distinguere tra il buio e la luce
- Misurare l'intensità della luce in un ambiente
- Misurare l'intensità della luce riflessa sulle superfici (colorate o neutre)
- Riconoscere colori differenti

Figura 2.7: Sensore di luce

Oltre a misurare l'intensità della luce in un ambiente, questo sensore è anche in grado di calcolare la distanza dalle superfici, emettendo una luce di colore rosso e misurando la quantità riflessa. Una difficoltà nell'utilizzo di questo tipo di sensore è dovuta alla sua estrema sensibilità nella misura della luce. Si riscontrano forti differenze di lettura al passare da una superficie opaca ad una lucida, da una bianca ad una colorata. Inoltre non risulta sfruttabile direttamente in stanze fortemente illuminate; infatti in luoghi dove è presente una elevata luminosità il sensore non riesce a rilevare variazioni utili (rimane sempre intorno al valore massimo). Questo sensore può essere utilizzato anche per distinguere colori diversi tra loro(es. bianco-nero, rosso-blu).

Sensore ad ultrasuoni

È uno dei due sensori (insieme a quello di luce) che dona il senso della vista all'NXT. È in grado di

- determinare la distanza da un oggetto
- misurare la distanza da questo
- determinare movimenti

Il principio su cui si basa e lo stesso che usano i pipistrelli per orientarsi: misura cioè il tempo che un onda impiega a colpire un oggetto e tornare indietro. È facile intuire che si possono incontrare difficoltà nel riconoscimento di superfici sferiche (una palla) o costituite di materiale morbido (tipo stoffa) o anche di superfici fini e piccole.

Figura 2.8: Sensore ad ultrasuoni

Il sensore può misurare le distanze sia in centimetri che in inch, ha un range di funzionamento da 0cm a 255cm con una sensibilità di $\pm 3cm$. Proprio a causa di una sensibilità decisamente non spinta, non è stato possibile utilizzare questo tipo di sensore per per il calcolo delle distanze nei nostri scopi.

Sensore di suono

Conferisce all'NXT la capacità di sentire, ed è in grado di lavorare in due modalità:

- dBA: [adjusted decibel] la sensibilità del sensore è adeguata a quella dell'orecchio umano.
- dB: [decibel] il sensore determina i decibel standard; tutti i suoni sono misurati con la medesima sensibilità (risulteranno anche quei suoni troppo alti o troppo bassi per l'orecchio umano)

Figura 2.9: Sensore di suono

Il limite massimo di lettura è 90dB (il rumore prodotto da un tagliaerba). Vista la moltitudine di suoni udibili e la difficoltà nella loro campionatura,

l'uscita che si ottiene dal sensore è in percentuale. Approsimativamente si ha:

% rumore	tipologia rumore
4-5	silenzio in una stanza
5-10	dialogo di persone non vicine al sensore
10-30	normale conversazone vicino al sensore o musica ad un livello normale
30-100	persone che gridano o musica a volume elevato

Tabella 2.1: Range valori letti dal sensore di suono

Il sensore può comunque essere programmato in modo da avere un valore di soglia sotto il quale non si ha nessun segnale di uscita.

Capitolo 3

Studio software

La LEGO ha deciso di rendere opensource il firmware dell'NXT in modo tale che ogni sviluppatore abbia la possibilità di creare a proprio piacimento un software capace di dialogare con l'hardware del prodotto. Non solo, è possibile addirittura creare nuove periferiche o, grazie a degli adattatori collegare dei sensori "standard" al mattoncino. Questo ha fatto si che su internet si siano sviluppati svariati progetti con lo scopo di aumentare le capacità di questo robot.
Il primo problema che ci siamo trovati ad affrontare è stato dunque quello di scegliere il software più adeguato ai nostri scopi.
L'analisi del software è stata condotta secondo due criteri:

- Gestione del programma: si individuano due categorie: una prima di cui fanno parte quei software che compilano e trasferiscono il file contenente il programma nella memoria dell'NXT che poi lo esegue, e una seconda di cui fanno parte quei software che consentono di caricare a bordo l'elaborato, ma lo eseguono tramite un eleboratore esterno (quindi da remoto).

- Tipo di interfaccia: anche qui possiamo individuare due categorie: nella prima rientrano quei programmi dotati di un interfaccia grafica, costituita spesso da blocchi trascinabili, che rappresentano le funzioni del robot. Della seconda fanno parte invece quei software che comportano la scrittura completa dell'elaborato e richiedono quindi una conoscenza di base di linguaggi di programmazione (C, C++, etc).

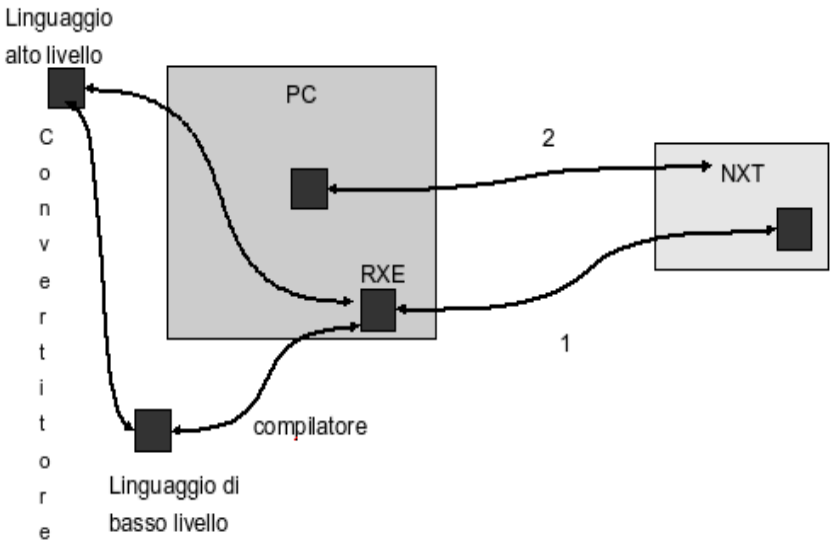


Figura 3.1: schema funzionamento

Con questa premessa possiamo passare ad un analisi più approfondita dei principali software.

3.1 LEGO NXT-G

Come prima analisi abbiamo ritenuto utile osservare il programma fornito di base, nel pacchetto Mindstorms LEGO NXT: LEGO NXT-G, compatibile con Windows e con MAC-OS.
Questo software, realizzato dalla collaborazione di LEGO e National Instruments, per mezzo di un linguaggio grafico permette la scrittura di programmi anche piuttosto elaborati. Permette di realizzare un file eseguibile direttamente dal mattoncino e possiede un utility per il download. È molto intuitivo e chiunque può riuscire in poco tempo ad imparare a programmare il mattoncino, tanto che LEGO propone il suo utilizzo all'interno di percorsi scolastici, e a tale scopo sono presenti dei tutorial e diversi esempi.
L'interfaccia si presenta molto intuitiva, con un numero limitato di blocchi

divisi per categorie, collegabili tramite elementi che assomigliano molto ai mattoni componibili con cui è costruito il robot.(fig. 3.2)

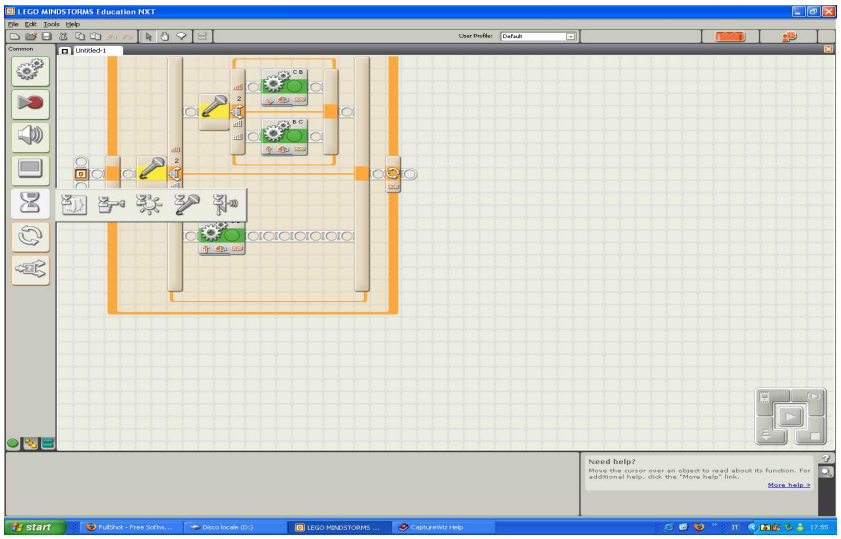


Figura 3.2: Screenshot NXTG

3.1.1 Blocchi

Sono 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 (es. nel blocco "Movimento" che aziona i motori, possiamo scegliere quali motori attivare, la velocità eccetera).
La gestione di tutte le variabili non è così semplice e intuitiva, ma le guide presenti sono di notevole aiuto e velocizzano l'apprendimento. Sono presenti dei blocchi specifici per realizzare cicli e switch molto utili se vogliamo costruire un sistema di controllo.

3.1.2 Considerazioni personali

Il software in questione presenta alcuni vantaggi quali l' intuitività e la semplicità di utilizzo; il fatto che sia realizzato appositamente per l'NXT limita

problemi di connettività e compatibilità. Purtroppo peró questa soluzione presenta svariati problemi che la rendono improponibile per i nostri scopi: la realizzazione di controlli non troppo complessi fa sì che lo schermo si riempia di blocchi immersi in una giungla di fili compromettendo anche la stabilità del sistema (per colpa di un interfaccia abbastanza pesante).

I file prodotti con questo programma sono molto più "grossi" di quelli realizzati con altri software (problema non banale dato che la memoria dell'NXT non è di grandi dimensioni), ma soprattutto l'esecuzione risulta assai più lenta.

Questo ci porta a considerare questo software inadatto ai nostri scopi, in quanto non possiamo inserire ulteriori limitazioni di natura informatica a quelle già presenti nell'hardware.

3.2 Robolab

Robolab è un secondo software sviluppato dalla Lego diretto principalmente ad un uso educativo, con l'obiettivo principale di utilizzare i prodotti Mindstorms nelle scuole medie e superiori. Da qui il bisogno di un software sempre molto intuitivo ma che allo stesso tempo permetta la creazione di programmi di maggiore complessità.
Analizzando Robolab si nota la somiglianza con l'originale LabVIEW, non solo per l'interfaccia ma anche per le possibilità offerte. Le necessità di programmazione dell'NXT sono ben sviluppate, e come imput e output sono presenti i sensori e gli attuatori dell'NXT.
L'interfaccia è molto leggera e di facile utilizzo, soprattutto per utenti di LabVIEW. Il risultato della programmazione è un file che viene automaticamente compilato e scaricato a bordo del brick, anche se l'esecuzione può essere supervisionata dalla macchina. E' possibile creare grafici in tempo reale.
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, funziona su Windows e MacOS.(fig. 3.3

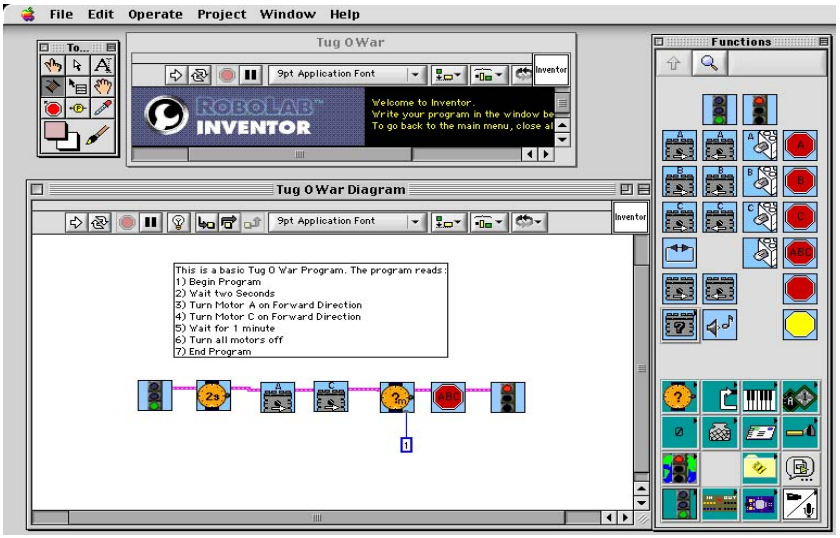


Figura 3.3: Screenshot Robolab

3.3 NI LabVIEW Toolkit

National Instruments ha sviluppato per proprio conto un add-on gratuito per il programma LabVIEW. Non è altro che un gruppo di "blocchi" contenenti gli input e gli output dell'NXT a cui si aggiunge un utility (purtroppo molto lenta) che permette il download dei programmi e il controllo da remoto (real-time) del brick.
Il software permette di sfruttare tutte le potenzialità di LabVIEW per programmare il mattoncino e quindi permette un notevole incremento delle prestazioni dello stesso. Permette, a differenza del software fornito dalla LEGO, di creare nuovi blocchi di comando che possono essere introdotti anche nel NXT-G, e questo risulta essere molto utile quando si deve passare da una programmazione di base ad una più complessa.
Essendo un add-on necessita del software LabVIEW per funzionare e di una discreta conoscenza dello stesso, il che lo rende molto più complesso dell'NXT-G.(fig. 3.4

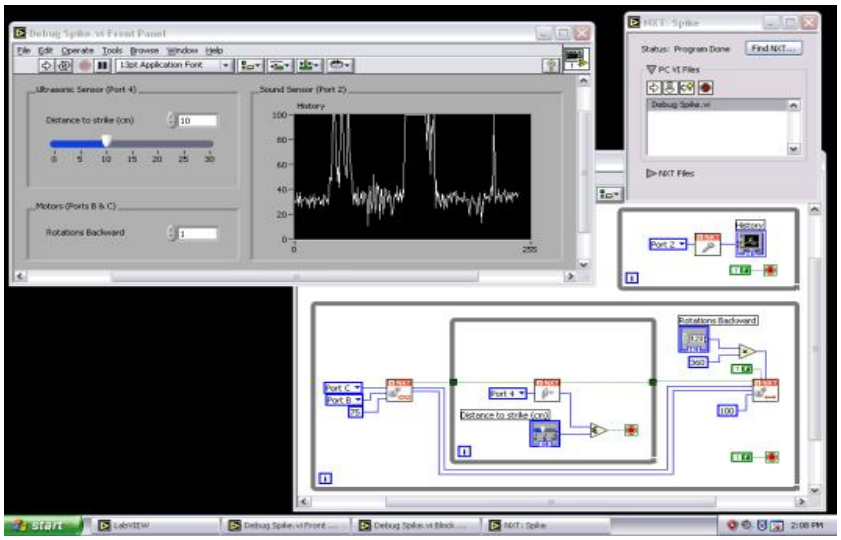


Figura 3.4: Screenshot NI Labview Toolkit

3.4 Microsoft Robotics Studio

MS Robotics Studio è un software per il controllo remoto; Microsoft ha creato la possibilità di interfacciarlo con il LEGO NXT. Questo programma non nasce per l'NXT, ma bensì come software per la programmazione e la simulazione virtuale di robot a livello superiore. Il suo utilizzo non è nè semplice nè intuitivo anche se l'interfaccia di lavoro rimane grafica. La caratteristica principale è la possibilità di simulare virtualmente la presenza del mattoncino e di testare il programma scritto sulla macchina.
Necessita di un software proprio e consente la comunicazione solo via bluetooth.(fig. 3.5)

3.5 NXT OnBrick

NXT OnBrick permette il controllo remoto del brick. Si tratta di una programmazione estremamente banale, con la quale, attraverso un facile linguaggio grafico, si ha la possibilità di eseguire alcuni controlli. L'inserimento dei blocchi grafici avviene in spazi pre-impostati a scapito della libertà di programmazione. Il programma viene scaricato sul mattoncino con possibil-

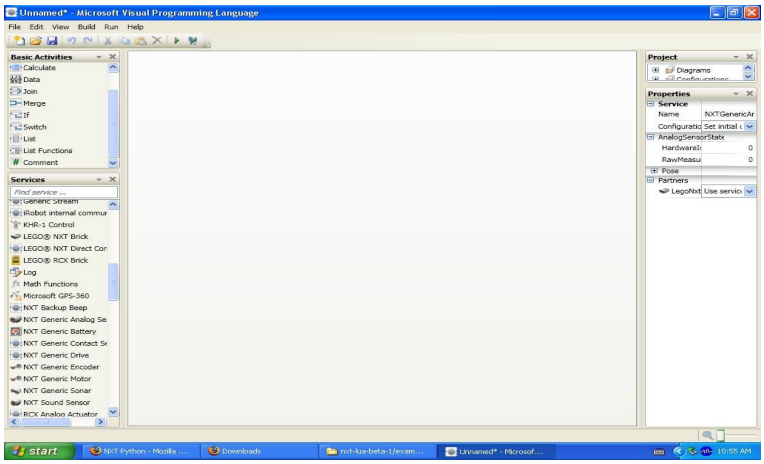


Figura 3.5: Screenshot Microsoft Robotics Studio

ità di scelta sia della comunicazione bluetooth che USB. Non risulta possibile eseguire cicli. Una curiosità è la possibilità di installarlo su un PDA.(fig. 3.6)

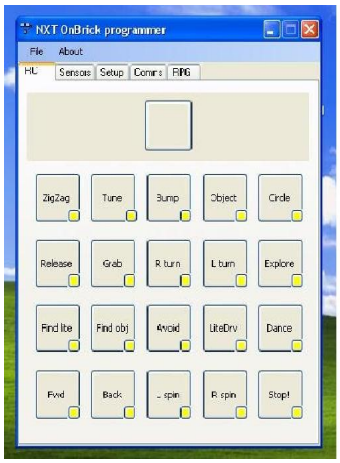


Figura 3.6: Screenshot NXT OnBrick

3.6 RobotC

RobotC è una valida alternativa al software fornito dalla LEGO. Sviluppato dalla Carnegie Mellon Robotics Academy, il programma si basa sul linguag-

gio base C; l'interfaccia non è grafica, ma è necessario scrivere direttamente i comandi, alcuni dei quali sono già preimpostati in modo da velocizzare la stesura. Ovviamente questo tipo si scrittura risulta più impegnativa e più lenta, ma le potenzialità sono nettamente superiori.
Per poter funzionare necessità di un proprio firmware, che essendo migliore di quello fornito dalla LEGO, consente velocità maggiori; è in grado di compilare e scaricare direttamente il programma sul brick. RobotC permette inoltre il monitoraggio real-time di alcune variabili dell'NXT (valore encooder motore, valore dei sensori, valore variabili di programmazione) e il controllo da remoto del mattoncino.
L'unica considerazione negativa, oltre alla difficoltà della tipologia di scrittura, è che questo non è un software libero, ma deve essere acquistato.(fig. 3.7)

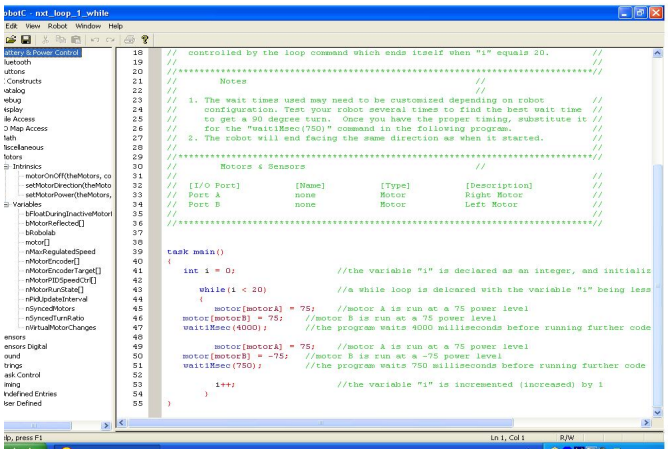


Figura 3.7: Screenshot RobotC

3.7 NeXT Explorer

Non è un software per la programmazione, ma bensì per la gestione dei file presenti sul brick: Consente infatti il download, l'upload e la cancellazione dei file presenti sull'NXT. Risulta utile perché avendo a disposizione una

memoria limitata è necessario gestire i file in maniera veloce e semplice.(fig. 3.8)

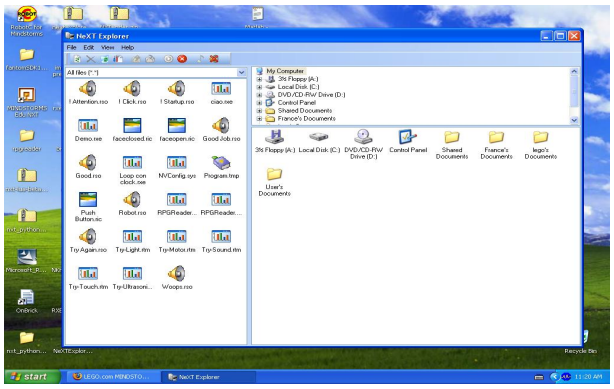


Figura 3.8: Screenshot Next Explorer

3.8 RXEdumper

Come il precedente è solo un software per la comunicazione e non consente la stesura di programmi. In particolare RXEdumper permette la compilazione dei file .nbc (provenienti da software che utilizzano questo tipo di linguaggio, ma che non sempre sono in grado di convertire i file direttamente in .rxe, cioè i file eseguibili dal mattoncino) in file .rxe, oppure viceversa. Il file deve poi essere inviato al brick, tamite altri software, quale ad esempio NeXT Explorer.(fig. 3.9)

3.9 Bricx Command Center

Il BricxCC permette la scrittura di programmi attraverso diverse tipologie di linguaggi (NBC, NXC, etc.), la compilazione dei file e l'invio di questi all'NXT. Utilizza il firmware fornito dalla LEGO e consente il controllo da remoto di alcune funzioni del mattoncino. Tuttavia il suo utilizzo non è banale e la possibilità di utilizzare diversi linguaggi è fruibile solo se in possesso di tutte le librerie necessarie. Le potenzialità sono comunque discrete ed essendo un opensource risulta in continua evoluzione.

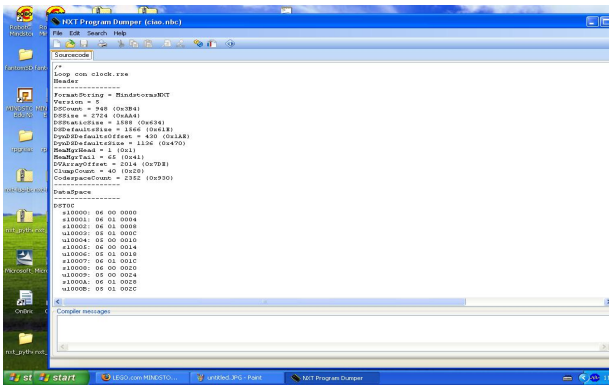


Figura 3.9: Screenshot RXEdumper

Il linguaggio NXC risulta particolarmante utile, perché è simile al C, ma allo stesso tempo meno formale, e ciò comporta una velocità di scrittura più veloce ma non per questo banale.(fig. 3.10)

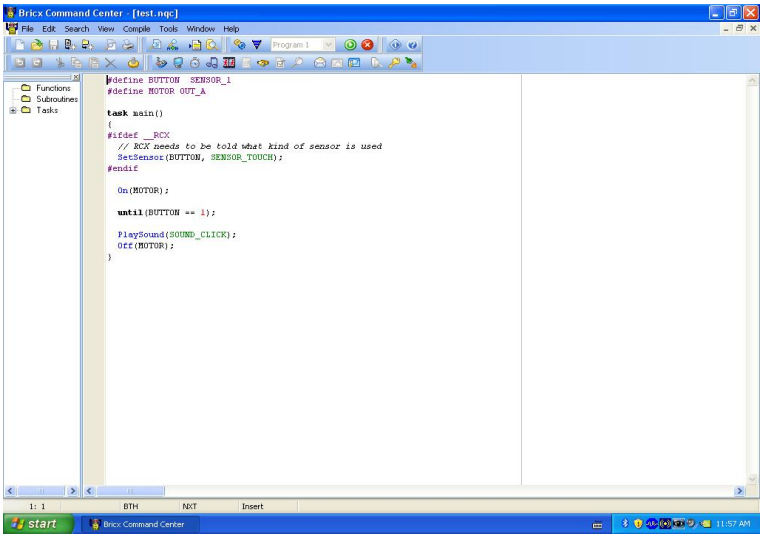


Figura 3.10: Screenshot BricxCC

3.10 Embedded Code Robot

Sono librerie che si aggiungono al Simulink; necessita quindi di MatLab per funzionare, ma è in grado di incrementare notevolmente le potenzialità dell'NXT. L'unica difficoltà riscontrata è nella compilazione del file ottenuto con questa programmazione in .rxe, cioè nei file eseguibili dal brick. Infatti, al momento dell'inizio di questa analisi non esisteva un metodo per interfacciare i due programmi, ed era possibile solo una simulazione virtuale.

3.11 NXTRC

Programma da linea di comando 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.

3.12 Scelta del Software

L'analisi dei software presenti in rete è stata senza dubbio difficoltosa sia per la grande quantità di programmi (gratuiti o meno) presenti, sia perché non sempre questi erano facilmente comprensibili e utilizzabili, o comunque necessitavano di approfondite analisi per comprenderne le reali potenzialità. Infatti, essendo spesso software gratuiti, creati e sviluppati in modo amatoriale o semi-amatoriale, non tutte le funzioni risultavano perfettamente operative e molte volte anche la stessa installazione si presentava complessa e molto articolata.

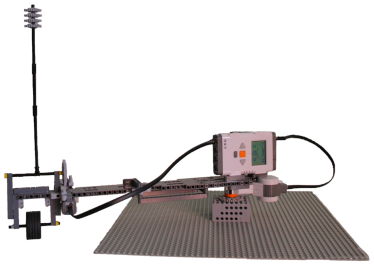
Lo studio dei software ha avuto un duplice obietttivo: trovare una modalità di programmazione più veloce e dinamica rispetto a quella fornita dalla LEGO che desse anche la possibilità di un dialogo con altri software, oltre che porre le basi per arrivare in un futuro ad un controllo Real Time, di robot NXT tramite computer remoto (come già presente per altre piattaforme nel laboratorio di automazione di Santa Marta). A tale scopo abbiamo cercato di interfacciare l'NXT con il linguaggio matlab, per creare la possibilità di

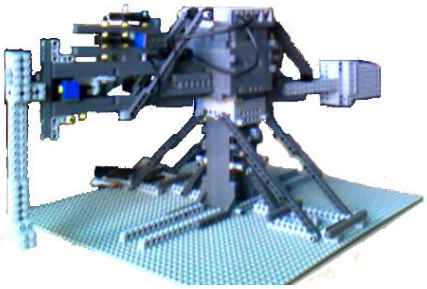
un legame con il laboratorio di Santa Marta che utilizza Simulink e Matlab per la gestone dei propri esperimenti.

Capitolo 4

Pendolo di Furuta

Il primo esperimento che abbiamo tentato di analizzare è stato la costruzione del pendolo di Furuta. Il nostro scopo era quello di ricreare la struttura presente all'interno del laboratorio di automazione di Santa Marta, attraverso l'utilizzo dei soli componenti LEGO. Il software utilizzato è quello fornito dalla LEGO, il LEGO NXT-G; questa scelta è stata dettata da un duplice motivo: non era stata ancora sviluppata un'interfaccia di comunicazione tra un linguaggio di scrittura e l'NXT di facile utilizzo; in ultimo dovevano ancora essere chiarire le effettive potenzialità del software (se risultasse attinente alle nostre richieste, la stesura e la compilazione del programma verrebbero notevolmente semplificate).





4.1 Pendolo di Furuta

4.1.1 Cenni di analisi cinematica e dinamica

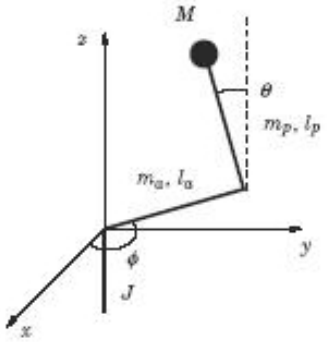


Figura 4.1: Pendolo di Furuta

Il sistema è composto da due corpi rigidi connessi tra di loro:

- un pilastro centrale, con momento di inerzia **J**, rigidamente connesso ad un braccio orizzontale di lunghezza **La**, di massa (linearmente distribuita) **Ma**.
- un pendolo di lunghezza **Lp**, di massa (linearmente distribuita) **Mp**.

Preso in considerazione un punto P e definendo **Ra** e **Rp** le distanze dall'origine quando P appartiene, rispettivamente al braccio e al pendolo, si possono definire la velocità e la posizione di P rispetto all'origine.

$$R(Ra,Rp) = [Rx(Ra,Rp), Ry(Ra,Rp), Rz(Ra,Rp)]^T$$

$$V(Ra,Rp) = [Vx(Ra,Rp), Vy(Ra,Rp), Vz(Ra,Rp)]^T$$

dove

$$Rx(Ra, Rp) = Ra \cos\phi - Rp \sin\phi \sin\vartheta$$
$$Ry(Ra, Rp) = Ra \sin\phi + Rp \cos\phi \sin\vartheta$$
$$Rz(Ra, Rp) = Rp \cos\vartheta$$

È possibile definire anche l'energia del pendolo di Furuta, e da questa determinare la lagrangiana e l'equazione di moto. Si riporta solo i risultati finali, perché i passaggi intermedi esulano dalle finalità di questa tesi.

$$(\alpha + \beta \sin^2 \vartheta)\ddot{\phi} + \chi \cos\vartheta\ddot{\vartheta} + 2\beta \cos\vartheta \sin\vartheta\dot{\phi}\dot{\vartheta} - \chi \sin\vartheta\dot{\vartheta}^2 = \tau$$
$$\chi \cos\vartheta\ddot{\phi} + \beta\ddot{\vartheta} - \beta \cos\vartheta \sin\vartheta\dot{\phi}^2 - \delta \sin\vartheta = \xi$$

Dove sono state introdotte le costanti geometriche:

$$\alpha = J + (M + \frac{1}{3}Ma + Mp)La^2 \qquad \beta = (M + \frac{1}{3}Mp)Lp^2$$
$$\chi = (M + \frac{1}{2}Mp)LaLp \qquad \delta = (M + \frac{1}{2}Mp)gLp$$

4.1.2 Struttura

La struttura, creata con lo scopo di garantire rigidità senza creare condizioni fisiche limitative per l'esperimento, è composta dalle seguenti parti:

- Colonna centrale rigida (a)
- Braccio centrale bilanciato (b)
- Pendolo con massa il più possibile concentrata nell'estremità (c)
- Trasduttore (d)

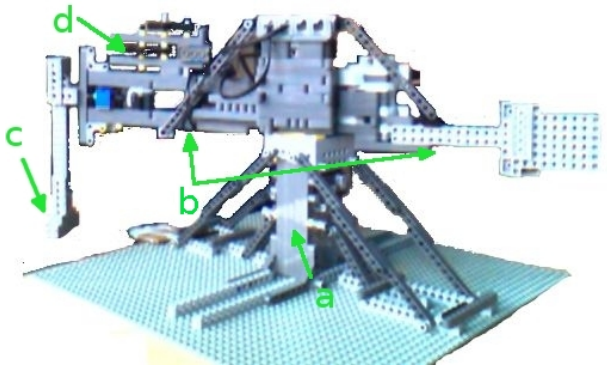


Figura 4.2: Prima struttura del pendolo di Furuta

La difficoltà principale è nella creazione di parti solide che non siano sensibili alle deformazioni elastiche dei componenti in modo da poter considerare il pendolo come una struttura rigida, condizione necessaria per poter impostare un controllo efficace. Per la rotazione del braccio sono utilizzati due motori.
La differenza principale sta nel posizionamento dell'elaboratore. Nel laboratorio di Santa Marta il controllo è realizzato tramite computer, esterno: I cavi per la lettura dei sensori e il controllo del motore passano all'interno della struttura. Nel nostro esperimento risulta impossibile pensare di mantenere il brick a terra: la sua posizione deve essere necessariamente solidale con il braccio per evitare di aggrovigliare i cavi di motori e sensori. Questo particolare crea sicuramente un ulteriore problema vista la massa dell'unità. Una possible soluzione potrebbe essere quella di far coincidere la posizione del centro di massa del brick con l'asse di rotazione dei motori in modo da annullare la componente del momento di inerzia dovuta alla distanza tra l'asse di rotazione e il baricentro. Rimane comunque il fatto che i motori devono trascinare una massa notevole e questo non può che peggiorare la reattività del sistema.

Colonna centrale I componenti LEGO non consentono di attaccare la colonna alla piastra d'appoggio su cui deve essere ancorata tutta la struttura. Il fissaggio è predisposto attraverso tiranti laterali assecondando anche la necessita' di impedire la torsione della colonna. Alcuni puntoni sono inseriti internamente.

Braccio Non potendo relizzare una struttura a sbalzo (per evitare una freccia eccessiva), la figura del braccio si rifà a quella di una gru da cantiere con tiranti e contrappeso. Tuttavia, l'elasticità dei materiali provoca lo stesso incovenienti; risulta ottimale creare una struttura leggermente precaricata (anche se i componeti LEGO non risultano essere molto efficaci per questo utilizzo).

Pendolo Semplici asticelle collegate in modo da fornire resistenza alla flessione (una sola si flette durante la rotazione a causa del massa) e una massa, collegata all'estremità, di valore piuttosto elevato per rallentare la dinamica del sistema.

Trasduttore Assolve il delicato compito di rilevare la posizione del pendolo. La scelta del tipo di trasduttore cade tra un encoder e un sensore di luce. L'encoder è del vecchio modello di NXT, l'RCX. Tali sensori devono essere montati sul braccio, in prossimità del pendolo.

4.1.3 Collegamento motori

Il collegamento dei motori con la colonna centrale e con il braccio è un particolare che richiede attenzione in quanto è fondamentale che sia solido per realizzare un controllo efficace. Sorgono così numerosi problemi:

- Ancoraggio fisico dei motori
- Reattività e velocità
- Attrito tra braccio e colonna e stabilità
- Flesso-torsione degli assi motore e salto delle ruote dentate

4.1.4 Ancoraggio fisico

La fisionomia dell'NXT non consente un fissaggio immediato, ma costringe alla creazione di una camicia esterna in grado di contenere i due motori. L'idea è quella di creare una scatola rigida in grado di garantire un collegamento stabile con la colonna centrale. Lo sforzo dovuto al peso della struttura non deve gravare sui due motori ma sulla scatola: i due attuatori devono esclusivamente fornire la coppia necessaria alla rotazione. Con questa scelta si riesce a rendere ininfluente il gioco tra rotore e statore lungo la direzione trasversale. Per unire i due motori si può pensare di utilizzare un asse comune oppure un sitema di ruote dentate. Questa seconda scelta risulta la più appropriata, fornendo anche una possibilità di regolazione del rapporto coppia/velocità.

4.1.5 Reattività e velocità

La reattività del motore LEGO considerato isolato dalla struttura è sufficientemente elevata, tuttavia aumentando la massa in movimento e dovendo eseguire continui e repentini cambi di direzione, questa non risulta più adeguata alle esigenze. Sfruttando gli ingranaggi usati per ancorare i due motori è possibile ovviare a questo incoveniente: se si utilizzano due ingranaggi di diametro inferiore a quello utilizzato per il collegamento alla colonna centrale, si ottiene un aumento della reattività anche se a scapito di una diminuzione della velocità massima.

4.1.6 Attrito braccio-colonna

Per evitare un attrito eccessivo e il rischio di impuntamento tra la gabbia e il supporto centrale durante la rotazione del braccio, sono utilizzati dei componenti LEGO che presentano una superficie lucida. Scaricando il peso della struttura sulla gabbia si ottiene un aumento della stabilità, perché se il peso fosse scaricato direttamente sui motori (come accadeva in strutture precedenti) si avrebbero oscillazioni lungo il piano verticale, dovute ai giochi presenti tra la parte rotorica (A) e quella statorica (B) del motore.

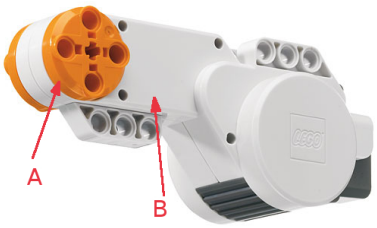


Figura 4.3: Statore e rotore motore

4.1.7 Flesso-torsione degli assi motore e salto delle ruote

Particolare attenzione deve essere posta nel collegamento con l'ingranaggio centrale. È vero che il peso è distribuito sulla gabbia, ma la coppia, che si scarica sulla colonna centrale attraverso l'ingranaggio, proviene dalle due

ruote dentate laterali, vincolate ai rispettivi motori. Si possono verificare due situazioni critiche tra loro interconnesse:

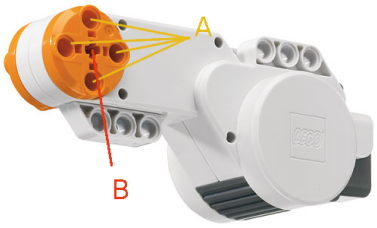


Figura 4.4: Fori motore NXT

- Flesso torsione degli alberi dei motori causata da un elesticità troppo elevata del collegamento tra il rotore e l'ingranaggio. La soluzione è aumentare i vincoli tra i due compenenti sfruttando tutti i fori presenti nel rotore (non utilizzare solo il collegamento contrale A, ma anche i quattro fori circolari B)(fig 4.4).

- Salto delle ruote motrici: può verificarsi sia a causa della flesso torsione degli alberi, sia per un'eccessiva coppia (ad esempio durante i cambi di direzione). Per ovviare a questo problema si pongono delle battute laterali che impediscono all'ingranggio di "saltare".

4.1.8 Trasduttore

La scelta del tipo di trasduttore e il suo collocamento all'interno della struttura riveste un ruolo fondamentale nella creazione del controllo. I requisiti fondamentali sono una buona sensibilità e un buon tempo di risposta. Testando i vari sensori a disposizione si nota subito che gli unici due in grado di soddisfare queste richieste sono:

- Sensore di luce

- Sensore di rotazione (RCX)

4.1.9 Sensore di rotazione

La LEGO non mette a disposizione alcun sensore per leggere gli angoli, ma possono essere acquistati da costruttori privati quali HiTechnic e mindsensors.com. Tuttavia, volendo utilizzare i componenti già in nostro possesso, la scelta è ricaduta sul sensore di rotazione LEGO fornito con l'RCX presente all'interno del laboratorio di ottica dell'Istituo Sistemi Complessi perché utilizzato in esperimenti precedenti.

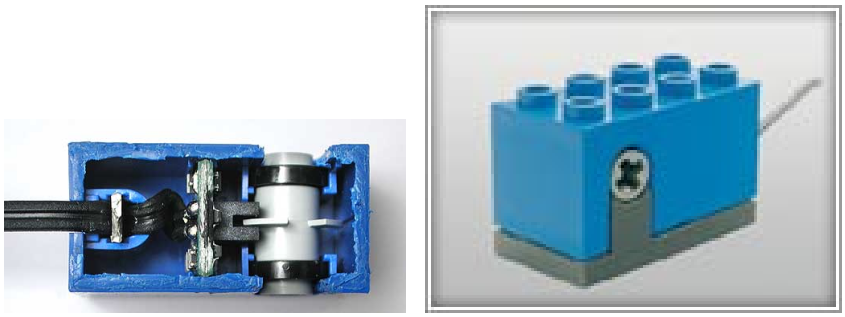


Figura 4.5: Sensore di rotazione

Caratteristiche sensore

Le caratteristiche di questo sensore sono piuttosto scadenti. Un giro completo viene diviso in 16 parti, circa 22,5 gradi,e ciò rappresenta una sensibilità eccessivamente bassa. La velocità angolare massima è 500 giri/min. Utilizzare il sensore senza nessuna modifica risulta quindi proibitivo e inutile ai fini del controllo che si vuole realizzare.

Collocazione fisica

Essendo un sensore nato per il modello RCX il collegamento fisico con i componenti dell'NXT non risulta immediato. La sua posizione è sul braccio ma non può essere fissato con il semplice utilizzo dei tasselli tipici del LEGO. Risulta necessario creare una struttura in grado di sorreggerlo.

Il migliore utilizzo si otterrebbe collegando direttamente il sensore con il pendolo, ma non potendo contare su una buona sensibilità, si deve ovviare "aumentando" il numero di giri del pendolo. Questo è possibile se si fanno

Figura 4.6: Ancoraggio sensore di rotazione

seguire all'albero del pendolo una serie di ingranaggi moltiplicatori; il limite superiore della moltiplicazione è dettato, oltre che dall'impossibilità fisica di avere una catena di ingranaggi troppo lunga, anche dalla risoluzione massima del sensore di 500 giri/min. Con un rapporto di moltiplicazione di circa 22 si ottiene una sensibilità di circa 1 grado. Tuttavia, a causa degli errori relativi propri alla risoluzione e alla velocità massima, risulta necessario l'utilizzo di due sensori di rotazione: uno per la lettura dei giri del pendolo moltiplicati per il coefficiente desiderato e uno per leggere il numero dei giri del pendolo reali. Il primo trova utilizzo in un range ristretto, intorno alla posizione di equilibrio instabile, mentre il secondo, deve essere sfruttato quando ci si allontana da tale posizione: una caduta del pendolo può provocare una velocità di rotazione dell'albero, al secondo sensosre, superiore a 500 giri/min, portando incertezza sulla misura. Nasce così la necessità di una lettura insensibile a tale errore, che dia la possibilità al controllo di riportare il pendolo in posizione di equilibrio instabile. Raggiunta nuovamente tale configurazione, viene utilizzato il sensore di rotazione ridotto. La risposta al gradino di questo sensore mostra un buon comportamento anche se i limiti fisici sono evidenti.

4.1.10 Sensore di luce

Il sensore di luce è quello già descritto per l'utilizzo nel Legway, tuttavia il suo utilizzo è notevolmente diverso.
Il pendolo è libero di ruotare a 360 gradi, quindi il sensore deve essere posizionato in modo da leggere sì l'inclinazione del pendolo, ma non ne deve impedire il movimento. Inoltre può essere utilizzato finchè il movimento non

è linearizzato.

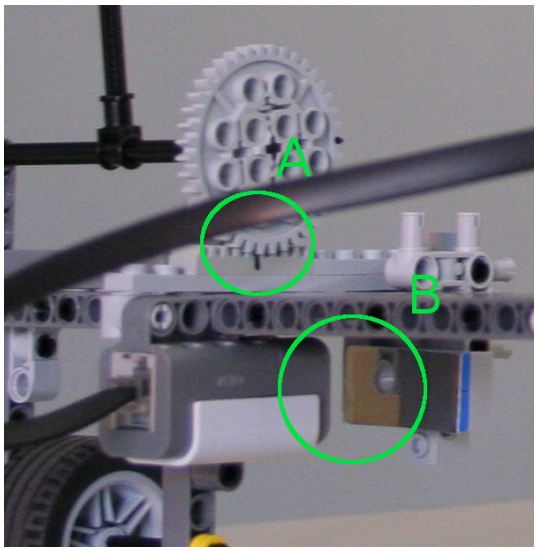


Figura 4.7: montaggio sensore di luce

L'immagine 4.7 è riferita alla nuova struttura del pendolo di Furuta, tuttavia il principio di funzionamento è rimasto invariato passando da una struttura all'altra. Per rendere rettilineo un movimento circolare è stato necessario utilizzare un accoppiamento ingranaggio (posto sull'albero proveniente dal pendolo) cremagliera (particolare A in 4.7). Questo sistema prende spunto dallo sterzo delle prime automobili, nelle quali la ruota dentata muovendosi fa scorrere la cremagliera. Sono necessari due sensori di luce per un funzionamento corretto (in entrambi i sensi). Inoltre sugli estremi della cremagliera sono posizionati degli specchi, rivolti verso il sensore, che invece sono ancorati al braccio (particolare B in 4.7). Per usufruire a pieno di questi sensori è necessario schermarli il più possibile dalla luce proveniente dall'ambiente esterno.

4.1.11 Scelta del trasduttore

La scelta del tipo di trasduttore non è stata immediata. Nonostante il sensore di rotazione necessiti di un moltiplicatore, risulta di più facile utilizzo rispetto a quello di luce, comportando, inoltre un peso complessivo inferiore per il braccio. Tuttavia l'evidente carenza dal punto di vista delle caratteristiche

ci ha portati all'utilizzo del sensore di luce. Le prestazioni di quest'ultimo sono notevolmente superiori anche per quanto riguarda la resistenza offerta alla rotazione del pendolo che risulta minimizzata rispetto a quella causata dagli ingranaggi. Tuttavia la nostra modalità di utilizzo non permette un funzionamento automatico del pendolo: la posizione iniziale, cioè del pendolo in equilibrio instabile corrispondente all'ingranaggio al centro della cremagliera, deve essere creata manualmente. Se lo si volesse evitare, si dovrebbe creare una ruota dentata solo in parte, in modo che l'ingranamento con la cremagliera avvenga sempre nella stessa posizione, ma in questa prima fase di studio abbiamo ritenuto sufficiente la nostra struttura. Entrambe le configurazioni dei trasduttori non risultano comunque molto appropriate: durante il funzionamento, le oscillazione del braccio creano effetti negativi sulla lettura del sensore, ma nonostante ciò sono le migliori configurazioni che siamo riusciti ad ottenere con i sensori in nostro possesso.

4.2 Programma di controllo

Come già detto, il software di scrittura è il LEGO Minsdstorms NXT-G. Trovandosi ad utilizzare un programma grafico si può pensare che la stesura sia semplice ed immediata, ma questo è vero finchè si rimane ad un livello basso di difficoltà, perché aumentando le richieste di prestazioni si notano subito le limitazioni di tale software:

- E' necessario avere più cicli di clock, eseguti in parallelo.
- Si devono impostare limiti precisi sui motori e sui tempi di funzionamento .
- Si ha la necessità che tutto il controllo sia veloce e dinamico.

Anche solo con queste richieste si nota subito una crescità esponenziale della difficoltà, dei tempi di scrittura e delle dimensioni del programma, con chiare ripercussioni sulla capicità di realizzare un controllo veloce. Nonostante l'aggiunta di alcuni blocchi disponibili on-line (ad esempio la funzione seno e coseno) questo software mostra notevoli limitazioni, che hanno impedito la sua applicazione e hanno portato al suo abbandono.

4.3 Nuova struttura

Nonostante tutte le accortezze con le quali abbiamo realizzato la struttura, per limitare l'influenza delle limitate caratteristiche meccaniche dei componenti LEGO sulla stabilità della struttura, non è stato possibile realizzare un controllo con questa prima configurazione, e le ragioni sono molteplici:

- Il braccio oscilla vistosamente lungo la verticale
- I motori non riescono a fornire l'accelerazione necessaria per mantenere in equilibrio il pendolo, o almeno ci riescono solo per piccoli spostamenti nell'intorno della posizione di equilibrio instabile. Quando il pendolo cade con una velocità troppo elevata il sistema non riesce più a recuperare tale inclinazione.
- L'intera struttura braccio-pendolo-sensore risulta pesante, non rigida e inadeguata ad ottenere una risposta rapida e precisa.

Tutte queste considerazioni ci hanno portato alla creazione di una nuova struttura (fig 4.8) che eliminasse tutte queste caratteristiche dannose, o almeno cercasse di attenuarne gli effetti.

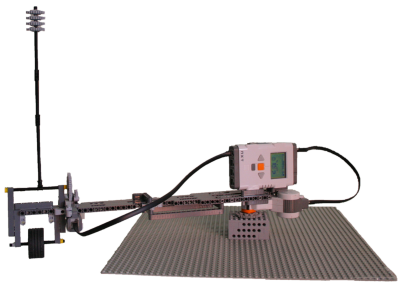


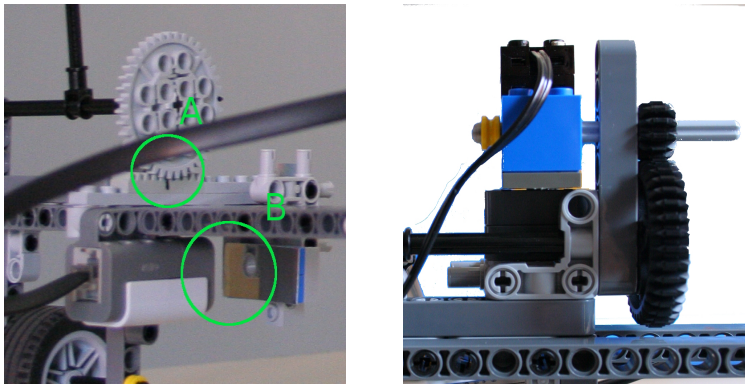
Figura 4.8: Nuova struttura pendolo di Furuta

Si nota subito che il braccio e il pendolo sono notevolmente più snelli, e il braccio stesso inoltre non ha problemi di freccia, perchè non si trova più a sbalzo ma poggia su una ruota. Questo non provoca nessuna modifica alla dinamica del pendolo, l'unica limitazione è sulla rotazione completa dello

stesso, ora impedita. Si è preferito cercare di mantenere in equilibrio il pendolo in questa nuova configurazione, lasciando in secondo piano il problema di cercare l'oscillazione adatta a portare il sistema in posizione di equilibrio instabile.
Tutta la struttura risulta più solida; la colonna centrale non è più in grado di torcersi per l'effetto del momento del motore che adesso è in grado di soddisfare da solo alla richiesta di coppia. Il peso del brick si scarica a terra tramite la colonna, mentre il peso del pendolo e del carrello con il sensore, viene sorretto dalla ruota.

4.3.1 Trasduttore

I tipi di trasduttori a disposizione non sono differenti dai precedenti sensori di luce e di rotazione.



(a) Montaggio sensore di luce

(b) Montaggio sensore di rotazione

Figura 4.9: Metodi li lettura dell'angolo

Testandoli entrambi si ottengono le medesime prestazioni, e visto che il tentativo è quello di migliorare il comportamento della attuale configurazione rispetto alla precedente, si preferisce adottare il sensore di luce che risulta più sensibile. Il tipo di montaggio, come si vede nella figura, é similare al precedente: il moto del pendolo è trasformato da rotatorio in rettilineo tramite un accoppiamento ingranaggio-cremagliera; su quest'ultima è ancorato uno specchietto che si avvicina e si allontana al sensore di luce vincolato

al braccio(vedi figura). Si crea così un legame tra l'angolo e l'intensità di luce.

4.3.2 Nuovo software

Nel cambiare la struttura è stato abbandonato anche il software NXT-G per passare al BricxCC. Come linguaggio di programmazione è utilizzato l'NXC, eliminando così elimato tutti i difetti relativi al software presenti precedentemente. I vari problemi che si presentano adesso sono dovuti alla struttura meccanica e ai componenti.

4.4 Abbandono del progetto

Attraverso tutti i cambiamenti e le modifiche apportate si è cercato di ottenere una configurazione in grado di sopportare la dinamica del pendolo e che permettesse di studiare tale fenomeno. Tuttavia i risultati non sono stati quelli sperati.

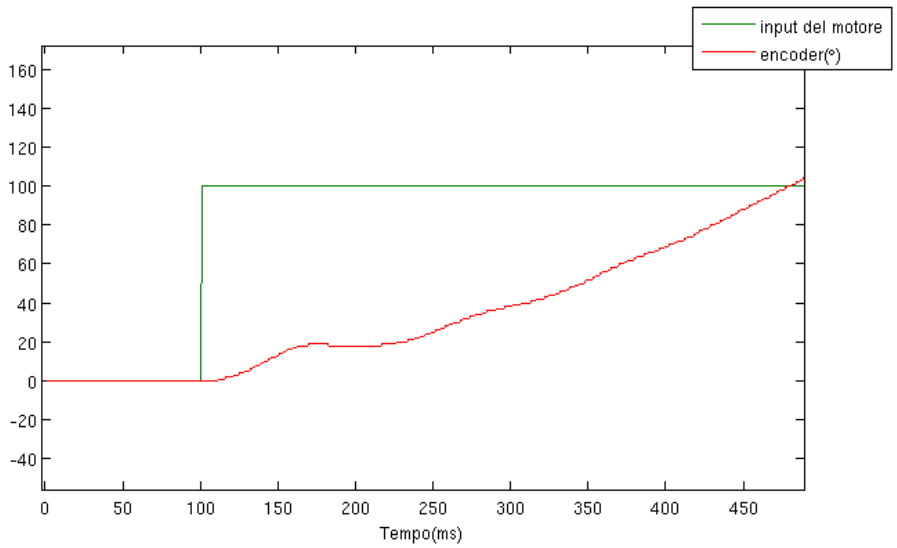


Figura 4.10: Risposta al gradino del pendolo

Nonostante aver alleggerito le varie parti componenti il pendolo di Furuta, e averle unite tra loro tramite connessioni solide non siamo risuciti a

mantenere in posizione di equilibrio instabile il pendolo.
Analizzando il comportamento della struttura ci siamo resi conto che l'insieme di tutti i giochi presenti (al motore, tra i pezzetti LEGO, al pendolo), l'elasticità e la flessibilità dei componenti stessi, rendono impossibile creare un sistema controllabile, e la risposta del pendolo al gradino conferma questa teoria (fig. 4.10). Si nota infatti una risposta oscillante dell'intero sistema; tale oscillazione compromette senza dubbio la possibilità di controllo dell'equilibrio.

È quindi impossibile (o forse lo è stato per noi) con il solo utilizzo dei componenti LEGO e dei sensori in nostro possesso, creare e controllare un pendolo di Furuta.

Capitolo 5

Segway

La principale attività svolta all'interno di questa tesi, riguarda la costruzione di un robot capace di mantenere l'equilibrio su due ruote montate sullo stesso asse: il Legway. Tale struttura prende spunto dal Segway, il nuovo mezzo di locomozione che sta prendendo sempre più campo.

(a) Legway (b) Segway

Figura 5.1: Legway e Segway

Il Segway (il cui nome significa "cambiamento senza intoppi da uno stato ad un altro") nacque da un'idea di Dean Kamen, il quale vide un giovane uomo su una sedia a rotelle che lottava per salire su un marciapiede. Pensando a questo realizzò che il problema non era la sedia a rotelle inefficace, ma il fatto che il mondo era costruito per le persone che possono stare in equilibrio. Così, insieme alla sua squadra, creò l' Independence IBOT Mobility

System: un mezzo di mobilità auto-bilanciato, che permette agli utilizzatori di salire scale e marciapiedi nonchè di muoversi su sabbia e rocce. Questa prima creazione ha dato l'idea per un successivo sviluppo del Segway a livello industriale, e la sua applicazione trova sempre più campo perché è in grado di "trasformare" una persona in un pedone potenziato, permettendole di andare più lontano, muoversi più velocemente e portare più carico.

Come prima cosa è stata creata una struttura di base, che sarà poi sviluppata e migliorata progressivamente; l'obbiettivo è quello di ottenere una struttura molto semplice ma allo stesso tempo solida e con il baricentro in posizione predeterminata. La scelta migliore per avere una misura di posizione affidabile si otterrebbe con un sensore giroscopico, ma in questo caso si preferisce utilizzare i sensori forniti nel kit di base(descritti precedentemente), la scelta ricade quindi sul sensore di luce, che garantisce un'ottima sensibilità anche se possiede grosse limitazioni che si è cercato di aggirare in qualche modo. Nel passo successivo si eseguirà un' analisi fisica della struttura in maniera qualitativa per avere un'idea di quali sono i parametri fondamentali su cui possiamo agire nel nostro controllo, dopodichè si passerà alla stesura del software di controllo e alla determinazione dei parametri in maniera sperimentale. Come ultimo passo si costruirà un joystick per inviare comandi di movimento al robot tramite bluetooth.

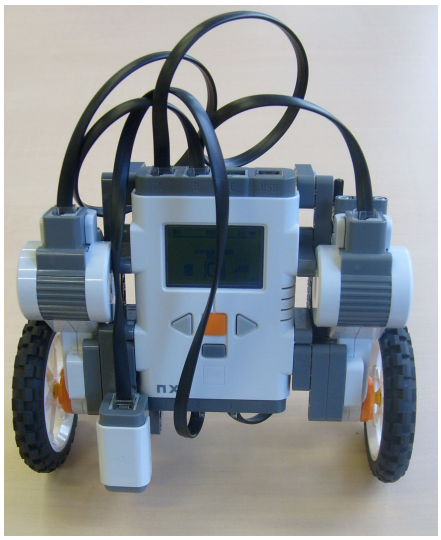
Esaminiamo adesso più dettagliatamente queste fasi.

5.1 Struttura

La struttura finale del Legway si ottiene attraverso molti step evolutivi. Originariamente la struttura ha una certa forma, ma sia attraverso prove sperimentali, sia attraverso considerazioni logiche, questa verrà modificata, al fine di ottenere le migliori possibilità di controllo e gestione del robot.

Come si può vedere dalle figure 5.2, sono state apportate diverse modifiche. Prima fra tutte si evidenzia la necessità di rendere equidistanti i due sensori di luce dal terreno di appoggio. Infatti, sfruttando la differenza dei due sensori, è necessario avere un riferimento comune; e a questo problema se ne aggiunge un secondo che consiste nel creare una struttura che presenti un baricentro in grado di mantenerla il più possibile nell'intorno della pro-

(a) Modello vecchio



(b) Modello finale

Figura 5.2: Modifiche alla struttura

pria posizione di equilibrio instabile (fig. 5.3). Risulta inutile avere sensori equidistanti dal terreno e una posizione di equilibrio sbilanciata in avanti: si ritroverebbero gli stessi inconvenienti della prima struttura. Per realizzare il bilanciamento abbiamo inserito un contrappeso nella parte posteriore.

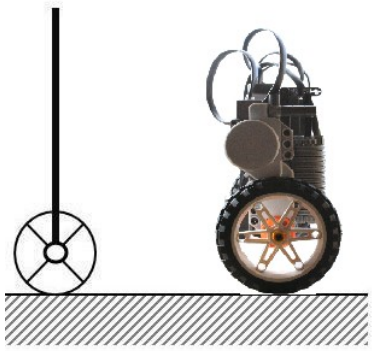


Figura 5.3: Equilibrio instabile

La struttura del Legway è minimale: i pezzi utilizzati sono volutamente pochi in quanto ogni collegamento fatto con il LEGO introduce dei giochi che minano la solidità del sistema.

Il Brick costituisce il corpo centrale del robot, la sua massa relativamente elevata ci aiuta ad aumentare l'inerzia del sistema e quindi rallentarne la dinamica. Ai lati del Brick si collegano i due motori con le ruote, in modo che abbiano l'asse di rotazione comune. Proprio sulle ruote arriva un'altra importante differenza con la vecchia struttura: il raggio delle ruote stesse. Attraverso prove sperimentali si può verificare che le ruote di diametro maggiore permettono un controllo migliore del sistema, inoltre la struttura costruita con queste modalità risulta più stabile e compatta.
Un particolare tentativo è stato fatto cercando di allontanare i sensori dal brick al fine di aumentarne la sensibilità di lettura, in quanto aumentando il raggio si cercava di amplificare la lettura della variazione di luce. Questa struttura è stata però subito scartata, perché risultava impossibile creare un legame solido che impedisse il dondolamento del sensore. La lettura risultava sì migliore, ma l'affidabilità del sistema diveniva pessima.
Le due ruote sono svincolate tra loro, mentre nella vecchia struttura erano legate da un albero, è risultato più conveniente assicurare l'equirotazione dei motori per via software (ciò rende possibile anche l'implementazione del comando con l'introduzione del joystick, in grado di far ruotare il robot).

5.2 Analisi Dinamica

Il funzionamento dinamico è semplice: ad una variazione(positiva o negativa dell'inclinazione) letta dai sensori di luce, deve corrispondere un momento dato dalle ruote, in grado di risollevare il Legway; questa coppia deve essere proporzionale all'angolo di inclinazione in modo da evitare che il controllo stesso sia la causa della caduta nel verso opposto.

In tabella sono riportati i significati dei vari simboli

Il comportamento del Legway è rappresentato dalla seguente equazione di moto:

$$\{J + T'(\theta)MT(\theta)\}\dot{\omega} + \{D + T'(\theta)M\dot{T}(\theta,\omega)\}\omega + T'(\theta)f_g = E_p e_m$$

dove i parametri dell'equazione corrispondo a :

$$\theta = \begin{bmatrix}\theta_1\\ \theta_2\end{bmatrix}, \quad \omega = \begin{bmatrix}\omega_1\\ \omega_2\end{bmatrix}, \quad f_g = \begin{bmatrix}0\\ M_1 g\\ 0\end{bmatrix}.$$

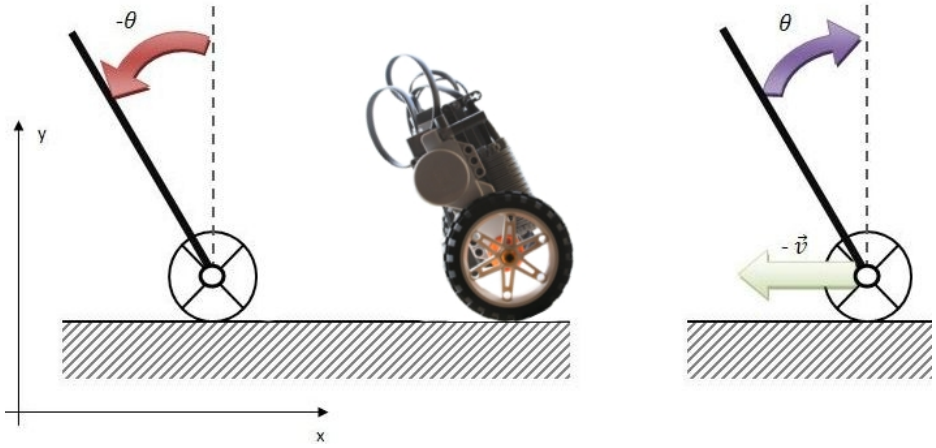


Figura 5.4: Movimento Legway indietro

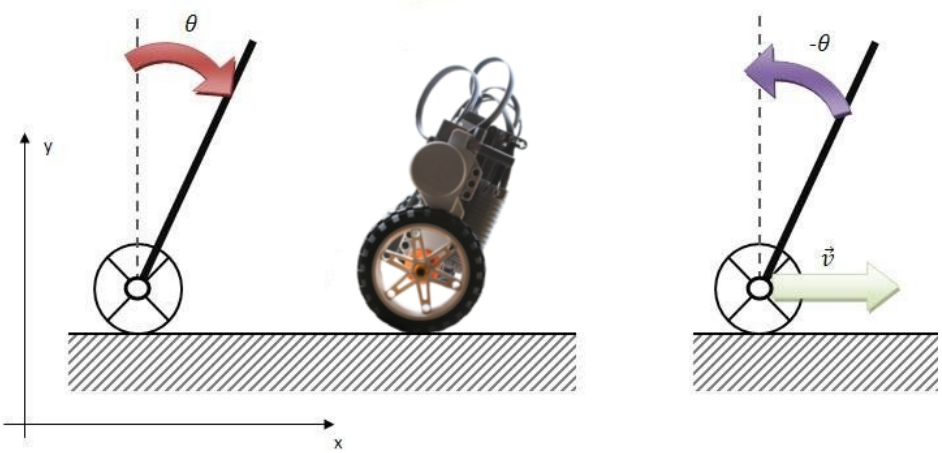


Figura 5.5: Movimento Legway avanti

Grandezza	Significato
M_1 (Kg)	Massa del corpo
M_2 (Kg)	Massa della ruota
J_1 $(Kg * m^2)$	Momento d'inerzia del corpo
J_2 $(Kg * m^2)$	Momento d'inerzia delle ruote
J_m $(Kg * m^2)$	Momento d'inerzia del motore
d_2 $(N * m/rad * s)$	coefficiente di resistenza viscosa dell'asse di rotazione delle ruote
l (m)	distanza tra il centro di massa e l'asse delle ruote
r (m)	raggio delle ruote
Φ_1 (rad)	angolo tra il centro di massa e la verticale
Φ_2 (rad)	angolo di rotazione delle ruote
Φ_m (rad)	angolo di rotazione del motore
ω_1 (rad/s)	velocità angolare corpo
ω_2 (rad/s)	velocità angolare ruota
ω_m (rad/s)	velocità angolare motore
f (N)	forza all'asse delle ruote
f_x (N)	componente orizzontale di f
f_y (N)	componente verticale di f
f_h (N)	Forza d'attrito
f_v (N)	Risultante punto d'appoggio
g (m/s^2)	accelerazione di gravità
τ_d $(N * m)$	Momento torcente alla ruota
τ_m $(N * m)$	Momento torcente sull'asse motore
R_m (ω)	resistenza uscita motore
G_m (S)	conduttanza uscita motore
K_τ $(N * m/A)$	costante meccanica motore
K_b $(V/rad * s)$	costante elettrica motore
e_m (V)	tensione uscita motore

Tabella 5.1: Parametri per la descrizione comportamento dinamico

$$M = \begin{bmatrix} M_1 & 0 & 0 \\ 0 & M_1 & 0 \\ 0 & 0 & M_2 \end{bmatrix}, J = \begin{bmatrix} J_1 + 2J_m & -2J_m \\ -2J_m & 2(J_2 + J_m) \end{bmatrix},$$
$$D = 2(d_2 + K_\tau K_b G_m) \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}, E_p = 2K_\tau G_m \begin{bmatrix} -1 \\ 1 \end{bmatrix}$$
$$T(\theta) = \begin{bmatrix} l\cos\theta_1 & r \\ -l\sin\theta_1 & 0 \\ 0 & r \end{bmatrix}, \quad \dot{T}(\theta, \omega) = \begin{bmatrix} -\omega_1 l\sin\theta_1 & 0 \\ -\omega_1 l\cos\theta_1 & 0 \\ 0 & 0 \end{bmatrix}$$

Anche se sono state identificate le equazioni di moto, l'analisi rimane ad un livello puramente qualitativo. Non si calcolano i reali momenti di inerzia o le varie posizioni dei baricentri, in quanto un'analisi così dettagliata sarebbe senza dubbio molto difficoltosa sia a causa della carenza di informazioni riguardanti i componenti dell'NXT in nostro possesso, sia perché lo studio prevede una continua evoluzione della struttura stessa. Abbiamo invece eseguito prove più utili ai nostri scopi, ad esempio testare i tempi di risposta dei motori o la risposta al gradino del sensore di luce. Queste ultime prove ci hanno permesso di capire se esistesse davvero la possibilità di realizzare un controllo con i sensori e i componenti in nostro possesso, o se occorra provvedere all'acquisto di nuovi sensori più sensibili e veloci.

5.2.1 Raccolta dati

Prima di effettuare delle prove dobbiamo trovare un sistema di acquisizione dei dati. Si utilizzano due metodi:

- **Scrittura** Il linguaggio utilizzato permette la scrittura di 4 file contemporaneamente: sfruttando questa possibilità si registrano le variabili interessate e, alla fine della prova, si trasferisce il file su un elaboratore(si ha lo svantaggio di dover stabilire prima la dimensione del file).

- **Invio** Attraverso la connessione bluetooth, si inviano i dati al computer in tempo reale, quest'ultimo si occupa di leggerli e scriverli su un file.

Il file generato viene poi letto ed elaborato utilizzando il software Matlab, con il fine di creare dei grafici.

5.2.2 Tempi di risposta del motore

Determinare i tempi di risposta dei motori è necessario ai fini della stesura del controllo. È inutile creare un controllo estremamente sofisticato se poi i motori sono "indifferenti" alla variazioni del segnale comando.
La prova si ottiene inviando un segnale gradino all'attuatore, e misurando la risposta dell'encoder presente negli stessi motori in funzione del tempo conteggiato utilizzando il Current Tick dell'NXT. (fig. 5.6)

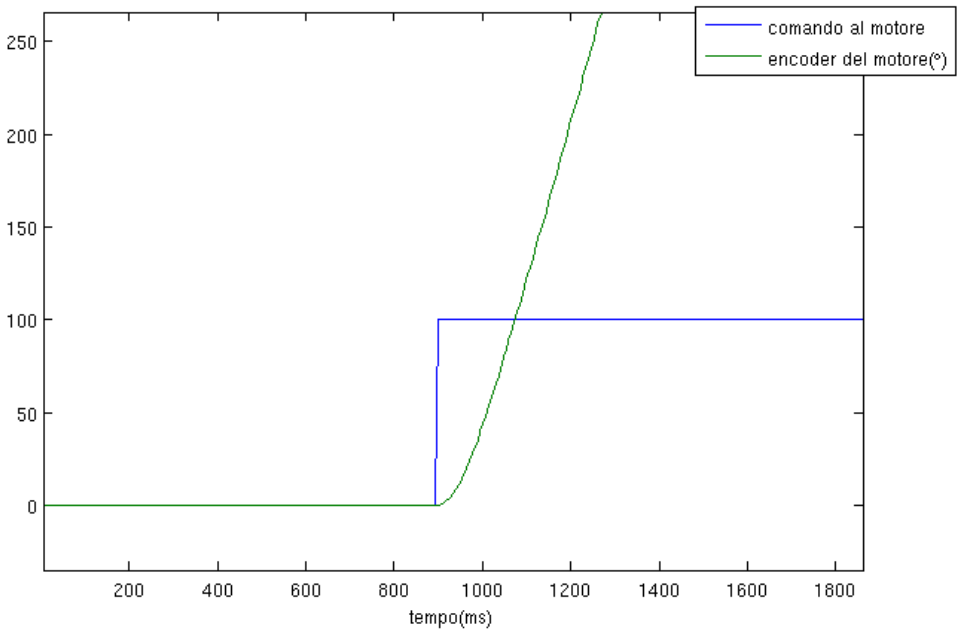


Figura 5.6: Risposta al gradino del motore

Una considerazione immediata è che i tempi di risposta del motore sono buoni e quindi non devono pervenire problemi di controllo derivanti da questo fenomeno. Un fenomeno che invece potrebbe creare problemi di stabilità è la presenza di una zona morta. Andando a pilotare il motore con un comando p che parte da zero e incrementa di una unità ogni secondo, si osserva che l'attuatore non crea coppia fino a che p non supera il valore 8, con $-100 \leq p \leq 100$. (fig. 5.7 e 5.8)

La presenza della zona morta può creare insensibilità al segnale di comando. Tuttavia durante il funzionamento del Legway tale fenomeno non

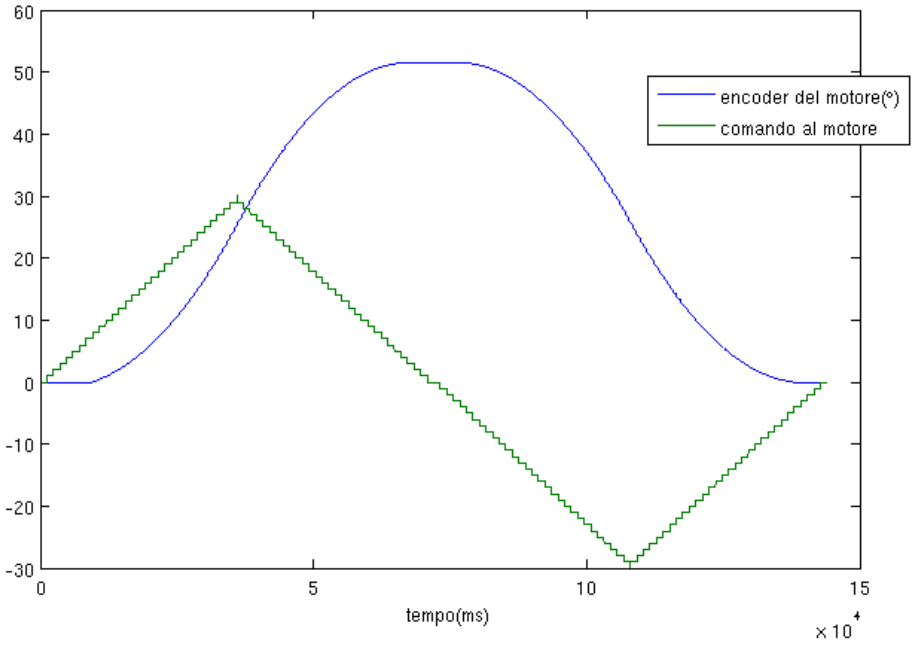


Figura 5.7: Zona morta del motore

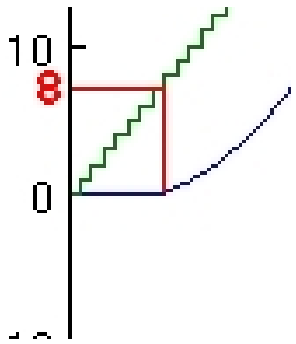


Figura 5.8: Ingrandimento zona morta del motore

comporta effetti negativi. Infatti il range nell'intorno di zero, nel quale il segnale di comando non produce alcuna uscita di coppia al motore, aiuta la stabilità del sistema rendendo il Legway indifferente ai disturbi (generati dai sensori di luce) che lo allontanerebbero dalla posizione di equilibrio instabile. La lettura dei sensori di luce è infatti molto "sporca": l'elevata sensibilità causa inevitabilmente variazioni anche piccole del comando, anche quando si raggiunge la posizione di equilibrio instabile, durante la quale non si vuole invece nessun comando agli attuatori.

5.3 Sensore di luce

Lo studio delle caratteristiche del sensore di luce si articola su più livelli. È necessario conoscere sia il comportamento dinamico del sensore (tempo di risposta, risposta al gradino), sia la risposta che la coppia di sensori da in presenza di diversi tipi di superfici (lucida, opaca, pietra, legno, etc.) e di condizioni di illuminazione (artificiale, naturale, riflessa, etc.) In un primo momento, per evitare o almeno diminuire l'influenza della luce esterna, sono stati posti dei filtri per l'infrarosso sopra i led di lettura della luminosità, leggendo in questo modo solo la luce generata dal sensore stesso e riflessa dalla superficie. Successivamente, a seguito di un miglioramento del controllo, è stato possibile togliere tali filtri. Risulta utile verificare la risposta dei sensori di luce in funzione delle varie superfici di lavoro.

5.3.1 Stima dell'influenza delle superfici e della luce esterna

Dovendo lavorare su superfici di natura variabile, con intesità e angolazione di luce indefinita, è conveniente analizzare il valore di luce rilevato dai due sensori. Le modalità di esecuzione di tali prove sono due:

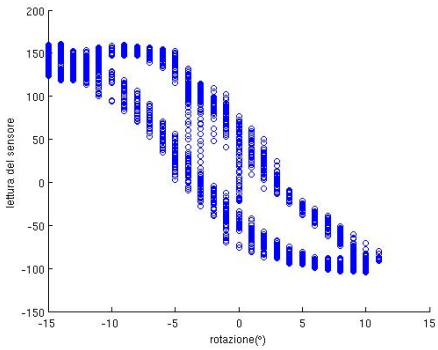
- Muovere manualmente il Legway mantenendo ferme le ruote simulando in questo modo il funzionamento normale. Il controllo range in cui muovere l'NXT è visivo (i limiti sono dettati dal contatto dei sensori con la superficie).

- Mantenere fermo il Legway in posizione orizzontale e muovere la superficie d'appoggio ricreata artificialmente. In questo modo è possibile creare un programma di controllo che imponga un valore univoco agli angoli di inclinazione.

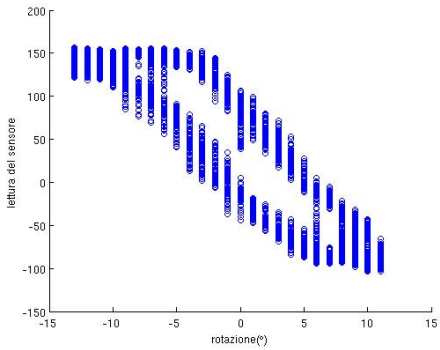
Per entrambe le tipologie di esecuzione le inclinazioni sono ripetute più volte all'interno della stessa prova al fine di rendere ininfluente il rumore sulla misura. Analizziamo i grafici ottenuti con la prima modalità.

Prova numero 1

La superficie d'appoggio è un tavolo lucido di legno betulla (chiaro) non illuminato direttamente dal sole. La luce presente nella stanza è quella naturale proveniente dall'esterno.(Prove a e b) Osservando i grafici appare evidente



(a) Prova a

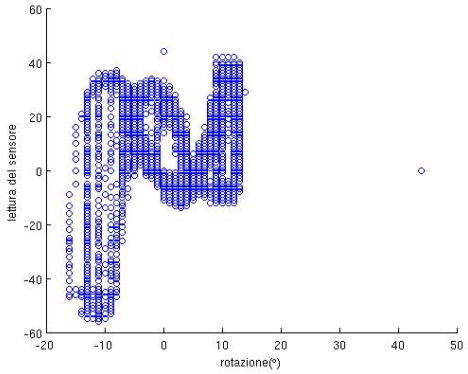


(b) Prova b

il fenomeno dell'isteresi, che si ripeterà anche per le prove successive. Il significato di tale forma verrà spiegato più avanti.

Prova numero 2

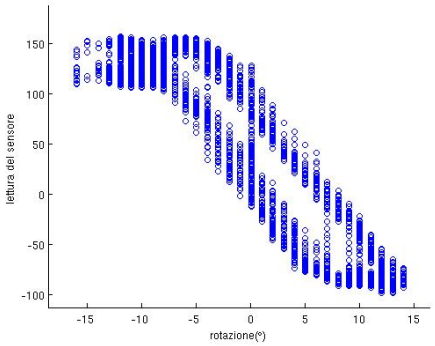
La superficie d'appoggio è un tavolo lucido di legno di betulla (chiaro) esposto direttamente alla luce solare.(Prova c) Il comportamento non è lineare: la luce solare riflessa dal tavolo lucido satura i sensori provocando notevoli problemi al controllo e alla stabilità del Legway.



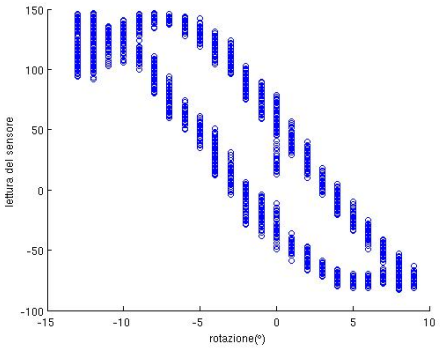
(c) Prova c

Prova numero 3

La superficie d'appoggio è un tavolo lucido di legno di betulla (chiaro) non illuminato direttamente dal sole, la luce nella stanza è artificiale. (Prove d ed e)



(d) Prova d



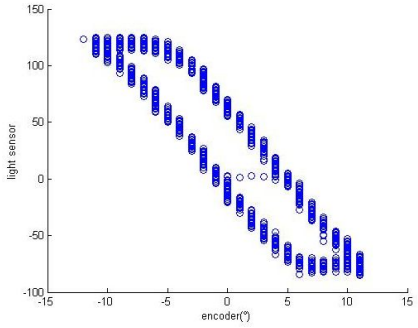
(e) Prova e

La presenza dell'illuminazione artificiale non altera sensibilmente il grafico. Tuttavia questo è vero finchè l'illuminazione rimane perpendicolare rispetto al tavolo. Se i raggi provenissero con una certa angolazione, infatti, potrebbero investire in maniera differente i due sensori provocando differenze nelle risposte degli stessi.

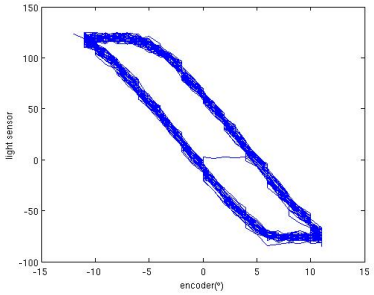
Analizziamo adesso i dati ottenuti con la seconda modalità.

Prova numero 4

La superficie utilizzata è un foglio bianco. Il range dell'angolo di inclinazione è compreso tra -10 e $+10$.



(f) Rappresentazione con punti



(g) Rappresentazione con linee

In questo caso è assente l'incertezza all'apice dell'isteresi causata principalmente dal contatto del sensore di luce con il piano di appoggio che provoca l'otturazione, anche solo parziale, del led di acquisizione.

Le considerazioni che si possono trarre osservando i grafici è che il colore e la composizione del materiale del piano di appoggio sono determinanti ai fini del controllo, in quanto responsabili del valore della risposta dei sensori. Nella situazione attuale il coefficiente di amplificazione relativo ai sensori di luce, all'interno dell'algoritmo di controllo, non tiene di conto di questo fenomeno. Tuttavia è prevedibile per il futuro un'implementazione del controllo durante la quale il robot acquisisce informazioni relative all'ambiente e modifica i relativi coefficienti, annullando i problemi che ne derivano. L'isteresi dei grafici è invece attribuibile al gioco presente nell'accoppiamento statore-rotore del motore che impedisce all'encoder di seguire fedelmente la variazione dell'angolo d'inclinazione, quindi non dipende propriamente dal sensore, ma da come è stata effettuata la misura.

5.3.2 Risposta al gradino

Per creare il gradino è stata utilizzata una lampada a led in modo che la generazione di luce potesse essere considerata istantanea. Il sensore è stato posto davanti alla pila e una volta avviato il programma dal brick, dopo pochi

istanti, è stata attivata la luce. Considerando la connotazione ludica dello strumento, il risultato ha superato le nostre aspettative: il salto avviene in 1-2 millisecondi e si stabilizza in circa 10 millisecondi.

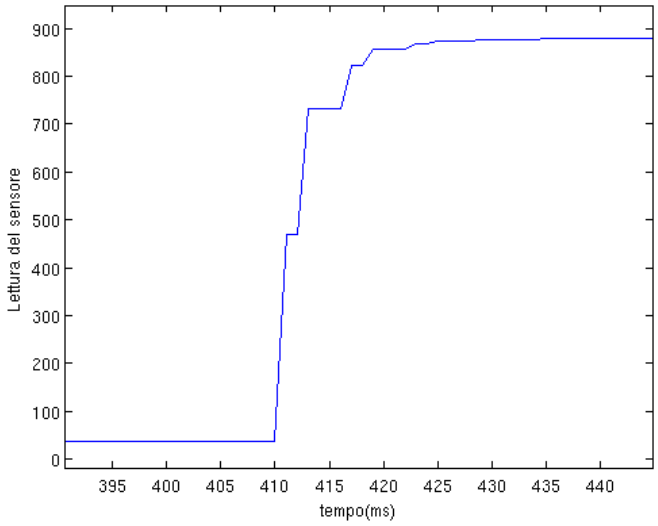


Figura 5.9: Risposta al gradino del sensore di luce

5.4 Software

Precedentemente sono stati descritti i principali software che consentono di lavorare con l'NXT. La varietà di questi programmi ci consente di trovare un buon compromesso tra le nostre esigenze.
I parametri fondamentali della nostra ricerca sono:

- La dimensione dei file compilati deve risultare ridotta compatibilmente alla complessità del comando
- L'esecuzione del programma deve risultare più veloce possibile
- Il linguaggio di programmazione deve essere comprensibile(senza sforzi eccessivi)
- Possibilmente la compilazione e il trasferimento del file a bordo non deve richiedere tempi eccessivi

- Si preferisce utilizzare software opensource
- Se possibile si vorrebbe utilizzare matlab per la programmazione

Allo stato attuale delle cose, nessun software sembra riuscire a soddisfare l'ultimo punto. Abbiamo deciso quindi di utilizzare un linguaggio non grafico che si avvicini il più possibile a qualcosa di già conosciuto, nel nostro caso il C. Con questa scelta si hanno tre possibili candidati: NQC, NXC, RobotC. Si sceglie di utilizzare l'NXC per l'ottima documentazione fornita e per il fatto che è libero (a differenza di RobotC). Il software BrickxCC sembra la scelta più comoda per la compilazione e il trasferimento del file, ma avendo a disposizione una macchina con sistema operativo GNU/linux risulta impossibile utilizzarlo. Optiamo allora per una soluzione separata di compilazione e trasferimento utilizzando per la prima parte il compilatore NXC/NBC fornito da John Hansen e per la seconda il software NXTRC.

5.4.1 Introduzione al linguaggio NXC

Prima di iniziare a scrivere il programma vediamo di comprendere come si utilizza questo nuovo linguaggio, almeno per le cose che ci interessano da vicino (per approfondimenti rimandiamo al manuale scritto da John Hansen)[1]. L'NXC si divide in due parti: il linguaggio, che descrive la sintassi da utilizzare, e le API(Application Programming Interface) che descrivono funzioni, costanti e macro che possono essere usate.

Linguaggio

Analogamente al C ed al C++ l'NXC è "case-sensitive"("xYz" è diverso da "Xyz").
Spazi e linee vuote non hanno significato per quanto riguarda la compilazione, servono solo per rendere il testo più leggibile.
Per inserire commenti si utilizzano i comandi "/*" di inizio commento, e "*/" di fine commento, oppure si utilizza il comando "//" che ordina al compilatore di ignorare ciò che segue nella singola linea.

[1] John Hansen, Not Exactly C (NXC) Programmer's guide

Costanti numeriche Le costanti possono essere scritte sia in forma decimale o esadecimale: scrivendo semplicemente i caratteri si ottengono numeri decimali; per avere numeri esadecimali dobbiamo aggiungere il prefisso "0x".

Task L'NXT supporta il multi-threading, quindi ad ogni task in NXC corrisponde un thread.

```
task pippo()
{
  // corpo del task
}
```

In un programma ci deve sempre essere un task con nome "main" che viene avviato all'inizio del programma.
Si possono avere fino a 256 task.

Variabili Tra diversi tipi di variabili utilizzabili noi utilizziamo le "int"(16 bit con segno), "unsigned long"(32 bit senza segno) e "string" (per le stringhe di testo). Per dichiarare le variabili si usa la seguente sintassi:

```
int x;          //dichiara x
string y,z;     //dichiara y e z
```

Queste possono essere di due tipi: globali o locali. Le prime si dichiarano all'inizio del programma (fuori dai task) e possono essere utilizzate da tutti i task, mentre le altre vanno dichiarate all'interno del task e sono utilizzabili solo all'interno di quello.

Strutture di controllo Una struttura di controllo consente di raggruppare una serie di istruzioni semplicemente includendole tra due parentesi graffe.

```
{
        x = 1;
        y = 2;
}
```

Questo strumento si rivela fondamentale per la creazione di un programma se abbinato a delle condizioni. Ad esempio si potrebbe voler eseguire le istruzioni contenute nella struttura solo se una variabile assume un determinato valore; questo si ottiene con la seguente sintassi:

```
        if (condizione) conseguenza else alternativa
```

ad esempio

```
if (x==1) y = 3 ; else y = 4;
if (x==1) {
y = 1;
z = 2;
}
```

La condizione "while" viene invece utilizzata per creare cicli(loop) e la si utilizza con la seguente sintassi;

```
while (condizione) corpo
```

durante l'esecuzione viene controllata la condizione, se è vera viene eseguito il corpo, poi viene ricontrollata la condizione e così via fin tanto che questa deventa falsa, in questo modo il corpo viene saltato e si passa alle istruzioni successive.[2]
Esempio

```
while(x < 10) {
        x = x + 1;
        y = y * 2;
}
```

Altri tipi di condizioni possono essere "for", "repeat", "switch" ecc...

Espressioni Questo tipo di istruzioni sono formate da "valori" e "operatori". I primi sono divisi in due tipologie, variabili e costanti numeriche. I valori vengono rappresentati in NXC come interi. Per le costanti vengono utilizzati 32 bit con segno, di conseguenza non si può superare la cifra 2147483647 e non si può scendere sotto la cifra -2147483648 per non incorrere in problemi di overflow(per le nostre necessità questi limiti non sono un problema).

Ci sono due variabili speciali predefinite: "vero" (a cui viene assegnato il valore 1) e "falso" (valore 0). Alcune espressioni danno questi risultati.

Per combinare i valori si utilizzano gli operatori. A noi interessano quelli fondamentali della matematica[3]; scritti in ordine di precedenza:

- moltiplicazione e divisione: *,/
- somma e prodotto: +,-

[2]Se la condizione fosse sempre vera il ciclo verrebbe ripetuto all'nfinito

[3]da notare che il carattere "=" non viene utilizzato come operatore ma come "assegnazione"

- maggiore, minore, maggiore/minore o uguale: >,<,>=,<=
- uguale a, diverso da: ==, !=

le parentesi servono per cambiare le precedenze:

```
x = 2 + 3 * 4;      // x = 14
y = (2 + 3) * 4;    // y = 20
```

NXC API

Finora abbiamo anzalizzato la sintassi "interna" dell'NXC, adesso dobbiamo vedere quali sono i comandi per interagire con l'hardware dell'NXT: sensori, attuatori, LCD, altoparlante e comunicazione.

Sensori L'NXT supporta fino a quattro periferiche di input usate contemporaneamente, in NXC sono state definite le costanti S1, S2, S3 e S4 per identificare le porte che le collegano. I valori di ogni sensore sono definiti con le seguenti costanti: SENSOR_1, SENSOR_2, SENSOR_3 e SENSOR_4. Per utilizzare un sensore peró questo non è sufficiente perché l'NXT non può conoscere a priori il sensore collegato ad ogni porta. Quindi dobbiamo definire il tipo di sensore e il modo in cui devono essere rappresentati i dati. La sintassi delle istruzioni sopra citate è la seguente:

- SetSensorType(costante porta, costante tipo)
- SetSensorMode(costante di porta, costante di modo)

Esistono 13 costanti di "tipo", noi utilizzeremo la configurazione che identifica il sensore di luce con la lampadina accesa. La costante di modo predefinita per il sensore di luce è "SENSOR_MODE_PERCENT" che dá in uscita un valore compreso tra 0 e 100, noi abbiamo però deciso di utilizzare la costante "SENSOR_MODE_RAW" che utilizza per intero il segnale a 10 bit del sensore (che varia tra 0 e 1023), questo per avere una maggiore sensibilità. Esempio:

```
SetSensorType(S2, SENSOR_TYPE_LIGHT_ACTIVE);
SetSensorMode(S2, SENSOR_MODE_RAW);
```

Queste istruzioni in genere si usano una volta sola all'inizio del programma, ma non è esclusa la possibilità di cambiare tipo o modo in qualsiasi momento.

Si può utilizzare il valore del sensore in qualsiasi espressione richiamandolo con i comandi SENSOR_1, SENSOR_2 ecc...

```
x = SENSOR_1;
//leggi il sensore 1 e immagazina il suo valore in x
```

Attuatori Si possono utilizzare fino a tre attuatori contemporaneamente, che funzionano sia da motori che da sensori di posizione. Le costanti OUT_A, OUT_B e OUT_C identificano le rispettive uscite, combinabili anche come OUT_AB, OUT_AC, OUT_BC e OUT_ABC.

Possono essere comandati con un ingresso di potenza (che varia per livelli da 0 a 100, valori negativi per il senso contrario), oppure si può impostare una determinata rotazione da compiere (in gradi), abbinata ad una potenza.

La funzione più semplice per muovere i motori con la potenza è il seguente:

OnFwd(uscita, potenza)

questo comando da una tensione costante al motore, che si muove con una velocità di regime costante, senza accorgimenti.

Per mezzo dell'encoder montato all'interno degli attuatori è possibile conoscere in ogni momento la posizione (con la precisione di un grado) relativa a quella iniziale(all'avvio del programma). La funzione che si utilizza a questo scopo è:

MotorRotationCount(uscita)

che ci da un valore espresso in gradi.

Altoparlante Risulta a volte comodo utilizzare dei segnali acustici per segnalare ad esempio che una operazione è stata eseguita correttamente all'interno del programma, oppure per segnalare la chiusura di un ciclo.

Per eseguire un singolo tono si utilizza la seguente funzione:

PlayTone(frequenza, durata)

con frequenza in hertz(Hz) e durata in millisecondi.

È anche possibile eseguire file musicali e melodie con la funzione:

PlayFile(nomefile)

I file da riprodurre devono essere dentro la memoria dell'NXT e devono essere in formato ".rso" oppure ".rmd"

Comunicazione L'NXT è dotato di comunicazione bluethooth che consente di scambiare informazioni con altri dispositivi. L'utilità più sfruttata consiste nel download dei file da un computer al mattoncino, ma l'NXC permette di sfruttare questo componente anche all'interno dei programmi.

Ad esempio possiamo inviare valori o stringhe di testo ad un calcolatore in maniera relativamente semplice con le funzioni:

- **SendRemoteNumber(connessione, box, valore)**
- **SendRemoteString(connessione, box, valore)**

dove il valore di "connessione" deve essere 0 perché la comunicazione viene avviata dall'esterno e il "box" è un altro valore a scelta. Ovviamente per funzionare sul computer utilizzato deve esserci un sofware che crea la connessione e legge quello che viene inviato.

Leggermente più complicato risulta far dialogare due o più NXT. I robot connessi possono essere al massimo 4, ma ci deve sempre essere uno e un solo "master" (cioè quello che crea la connessione attraverso il menù del Brick), gli altri("slave") non possono scambiare dati tra di loro ma comunicano solo con il master.

Il master deve assegnare un numero di connessione(1,2,3) ad ogni slave con cui poi riesce ad identificarli. In ogni slave ci sono delle caselle(box) numerate che possono contenere un valore per volta e possono essere modificate e lette sia dallo slave che dal master. Queste box sono tutte identiche ma per praticità faremo la distinzione tra OutBox(OB) che utilizziamo per inviare dati e InBox(IB) per ricevere dati[4].

Le funzioni principali da utilizzare dal master sono:

[4]Da notare che una stessa casella usata per trasferire file dal master allo slave prenderà nome OB nel programma del master e IB nel programma dello slave, e viceversa

- **SendRemoteNumber(connessione, OB, valore)** per scrivere valori nella OB
- **ReiceiveRemoteNumber(connessione, IB, variabile)** che legge il valore presente nella IB e lo immagazina nella variabile scelta

Per quanto riguarda lo slave:

- **ReiceiveRemoteNumber(IB, svuota, valore)** legge il valore presente nella IB e lo immagazina nella variabile scelta, la variabile "svuota" può avere i valori "true" o "false" e consente di cancellare il contenuto della box dopo la lettura
- **SendResponseNumber(OB, valore)** per scrivere valori nella OB

5.4.2 Struttura del software di controllo

Il software che controlla il Legway è costituito da tre task:

- **main**: serve per le operazioni di inizializzazione e fa partire in parallelo gli altri
- **command**: comanda i motori e si occupa della comunciazione bluetooth
- **speed**: legge i sensori e calcola i differenziali

Qualsiasi operazione svolta dalla CPU richide un certo tempo per essere eseguita, non c'è modo di conoscere a priori quanto dura ogni singola istruzione ma, se vengono messe all'interno di una struttura di controllo, si può leggere un timer all'inizio e alla fine e avere una stima del tempo totale impiegato. Con questo metodo ci si accorge che, mettendo tutte le operazioni in una struttura, risulterebbe molto difficile avere un controllo efficace a causa di tempi piuttosto rilevanti.

Da questo problema nasce l'idea di dividere le istruzioni in due task, uno più veloce e uno più lento, in modo da dare precedenza alle operazioni più "urgenti".

Variabili Per semplicità tutte le variabili presenti sono "globali". Quasi tutte le variabili utilizzate sono di tipo "int"(intero a 16 bit con segno). Si sceglio il tipo "unsigned long" (intero a 32 bit senza segno) per le variabili con cui misuriamo il tempo perché l'unità è il millisecondo. Con una variabile a 16 bit ogni 65 secondi (nel caso senza segno) circa si ha un azzeramento che crea problemi.

```
 int a, b, v, d, p, Vref, Vref2, s, z, w0, g, x,, v2, y, l;
unsigned long t0, t1, t;
```

task main Come già accennato, in NXC viene avviato per primo il task con nome "main". Per prima cosa dobbiamo definire i sensori di luce:

```
SetSensorType(S2, SENSOR_TYPE_LIGHT_ACTIVE)
SetSensorMode(S2, SENSOR_MODE_RAW)
SetSensorType(S3, SENSOR_TYPE_LIGHT_ACTIVE)
SetSensorMode(S3, SENSOR_MODE_RAW)
```

in questo modo i sensori di luce 2 e 3 vengono attivati, e con luce accesa i dati vengono interpretati come un segnale proporzionale all'intensità di luce che puó variare da 0 a 1023.

Si dá un tempo di attesa per evitare che il programma prosegua senza aver terminato l'operazione precedente; 100 millisecondi sono sufficienti.

```
Wait(100)
```

Ci sono varie problematiche che impediscono ai sensori, anche se montati simmetricamente, di non avere identico valore iniziale. Queste possono essere:

- **Posizionamento**: l'avviamento viene eseguito a mano e risulta quindi impossibile ricercare ogni volta l'esatta posizione verticale, di conseguenza i sensori non avranno la stessa distanza da terra.
- **Ambiente**: la luce esterna può avere una direzione diversa da quella ideale, oppure la superficie di appoggio può presentare irregolarità.
- **Sensori stessi**: non è detto che siano perfettamente uguali.

Una possibile soluzione consiste nel normalizzare la differenza con un valore memorizzato all'avvio, si legge dunque la differenza dei sensori:

```
Vref = SENSOR_2 - SENSOR_3
```

Si prende anche un tempo di riferimento, necessario perchè il timer si attiva con l'accensione dell'NXT e non si azzera all'inizio del programma.

```
t1 = CurrentTick()
```

Dopo queste operazioni preliminari possiamo avviare in parallelo gli altri due task con il comando:

```
Precedes(speed, command)
```

Ecco il task completo:

```
task main() {
SetSensorType(S2, SENSOR_TYPE_LIGHT_ACTIVE);
SetSensorMode(S2, SENSOR_MODE_RAW);
SetSensorType(S3, SENSOR_TYPE_LIGHT_ACTIVE);
SetSensorMode(S3, SENSOR_MODE_RAW);

Wait(100);

Vref = SENSOR_2 - SENSOR_3;
t1 = CurrentTick();
Precedes(speed, command);
}
```

task speed In questa parte del software vengono letti o calcolati le variabili necessarie al controllo:

- posizione angolare assoluta
- rotazione delle ruote rispetto al robot
- derivata rispetto al tempo della posizione
- derivata rispetto al tempo della rotazione delle ruote

Chiaramente risulta impossibile calcolare le derivate con un sistema a tempo discreto; le possiamo però approssimare con un rapporto incrementale del tipo $\frac{\Delta p}{\Delta t} = \frac{p(t_2)-p(t_1)}{t_2-t_1}$ se si riesce a determinare l'intervallo di tempo. Più piccolo risulta questo intervallo e più si tende alla reale derivata.

Dovendo eseguire l'operazione ciclicamente si utilizza un loop infinito con all'interno un'attesa che serve a scandire l'intervallo di tempo desiderato del nostro rapporto incrementale. Il tempo di campionamento scelto è 5ms, che risulta un valore ottimale visto che il tempo di clock per il *taskcommand* è 20ms (in un ciclo del command avvengono 4 cicli del *taskspeed*).

```
task speed() {
while(1)    {
...
...
Wait(5)
}
}
```

Il primi comandi del ciclo calcolano la differenza tra i due sensori di luce e l'encoder delle ruote, questi valori vengono memorizzati su altre due variabili subito prima dello "Wait". Questo ci consente di disporre, nello spazio tra le due istruzioni, di due valori dello stesso segnale a tempi diversi(distanti di 5 millisecondi in questo caso).

```
while(1){
b = SENSOR_2 - SENSOR_3
...     //qua posso utilizzare b(t) e d = b(t-5)
d = b
Wait(5)
}
```

Questo ci consente di calcolare il rapporto incrementale nel modo visto precedentemente, il suo utilizzo puro tuttavia, non risulta ottimale ai fini del controllo. Non tenendo di conto dei cicli precedenti, ogni calcolo andrebbe a considerare solo i valori letti dai sensori in quel preciso momento. In presenza di rumore si avrebbe un risultato non solo funzione della condizione reale, ma anche del rumore che risulta dominante. Si sceglie così di utilizzare un filtro del tipo: $v(k) = \alpha \cdot v(k-1) + (1-\alpha) \cdot \frac{\Delta p}{\Delta t}$, in modo da eliminare eventuali disturbi e ottenere una misura affidabile.

```
v =  v/4 + 150*(b - d)
```

I coefficienti numerici sono stati determinati sperimentalmente, per via iterativa. Tutti questi comandi eseguiti con i valori proveniente dai sensori di luce vengono eseguiti anche con i valori provenienti dagli encoder dei motori. Ecco il task completo:

```
task speed() {
```

```
while(1)    {
b = SENSOR_2 - SENSOR_3
z = (MotorRotationCount(OUT_B)+MotorRotationCount(OUT_A))/2
v =  v/4 + 150*(b - d)
v2 = v2*9/10 + 20 * (z - y)
d = b
y = z
Wait(5)
}
}
```

task command Questo task è responsabile di:

- generare il segnale di potenza dei motori
- porre un limite superiore e inferiore al valore massimo del motore
- acquisire i comandi del joystick

La prima operazione svolta è rivolta al fissaggio del tempo di clock. Il valore assegnato per il ciclo, come già detto, è 20 ms e risulta un tempo che media tra la necessità di un controllo reattivo e quella di eseguire tutti i calcoli.

```
while(1) {
t0 = t + 20
t = CurrentTick() - t1
...
...
while(t0 > t){
t = CurrentTick()
}
}
```

Questo algoritmo verrà chiarito nella sezione successiva.

La compenente principale di questo task è la generazione del segnale di comando p. Questo è la composizione di tutte la variabili acquisite o calcolate, pesate da coefficienti determinati sperimentalmente per via iterativa.

```
p = (b - Vref + z/10  + a/3 )*6/5 + v/45  +  v2/8
```

Le variabili sono tutte note, tranne "a"; questa non è necessaria ai fini del controllo: ha lo scopo di far muovere il Legway avanti e indietro e proviene dal joystick, inviata tramite collegamento bluetooth.

Se per qualche motivo il robot non viene inizializzato in posizione perfettamente verticale, avremo una "$Vref$" sbagliata, che conterrà un errore e. Per colpa di e il robot comincerà a "camminare" nella direzione in cui è sbilanciato, senza qualcosa che controbilanci e non si ferma. Mentre il Legway si muove, z cresce (o diminuisce a seconda della direzione) finchè il suo valore (corretto con una costante) non controbilancia perfettamente e; a questo punto il comando non è più sbilanciato e il robot si ferma in posizione di equilibrio. Non solo, ma ogni volta che z varia (ad esempio spostando manualmente il Legway), il controllo automaticamente tenterá di riportare z al valore che bilancia e. Con lo stesso criterio si ottiene lo spostamento comandato in avanti e indietro: se a, che finora abbiamo supposto essere uguale a 0, dovesse cambiare, z dovrà cambiare per bilanciare anche questo nuovo squilibrio; se a aumenta costantemente il legway comincerà a muoversi con velocità costante(apparte un periodo di transizione iniziale).

Il controllore di base utilizzato è di tipo PD, si utilizza quindi, oltre alla posizione, anche il differenziale v calcolato nel task precedente.

Se calcolassimo p solo con le variabili descritte fin ora, il robot non farebbe alcuna distinzione tra l'essere fermo o in movimento. Un aumento della velocità media di movimento, porterebbe ad una diminuzione di coppia non previsto dal controllore. Si ricorre dunque ad un controllo sulla velocità stimata delle ruote $v2$, calcolata nel task precendente. Per determinare questa stima, è necessario un alto coefficiente di filtraggio, per evitare di interferire con quella componente di velocità che varia con una frequenza più alta, indispensabile al controllo.

La rotazione a destra e a sinistra viene invece governata con il seguente algoritmo:

```
g = (p + x/5)
l = (p - x/5)
```

dove x è la seconda variabile inviata dal joystick che si va a sommare al comando p calcolato prima, e genera due controlli distinti per il motore di destra e di sinistra del Legway, g e l.

```
OnFwd(OUT_A, l)
```

```
OnFwd(OUT_B, g)
```

L'ultima parte di algoritmo appartenente a questo task riguarda l'imposizione del limite superiore e inferiore al valore di g e l. Infatti un problema riscontrato sugli attuatori NXT è l'instabilità per valori comando di 128 e -128, corrispondenti alla saturazione.

```
if(l > 100) {
l = 100
}
if (l<-100) {
l = -100
}
if(g > 100) {
g = 100
}
if (g<-100) {
g = -100
}
```

Visto che comunque la saturazione dei motori arriva per valori di ± 100 è inutile impostare un limite superiore a ± 127. Questo problema deriva dai numeri di bit destinati ai motori (128 corrisponde a 2^7): si crea infatti incertezza sul valore da attribuire al segnale di comando quando è raggiunto e superato il valore massimo.

Il task completo:

```
task command() {
while(1) {
t0 = t + 20
t = CurrentTick() - t1
ReceiveRemoteNumber(IB1, false, a)
ReceiveRemoteNumber(IB2, false, x)
p = (b - Vref + z/10 + a/3 )*6/5 + v/45 + v2/8
g = (p + x/5)
l = (p-x/5)
if(l > 100) {
l = 100
}
if (l<-100) {
l = -100
}
```

```
if(g > 100) {
g = 100
}
if (g<-100) {
g = -100
}
OnFwd(OUT_A, l)
OnFwd(OUT_B, g)
while(t0 > t) {
t = CurrentTick()
}
}
}
```

5.4.3 Tempo di campionamento e saturazione del conteggio

Per considerare affidabile un qualsiasi sistema di controllo è necessario conoscere e poter impostare il tempo di campionamento. Senza un'informazione precisa sul tempo di clock, infatti, qualsiasi controllo perde di significato, in quanto il processo non è in condizioni di ripetibilità, almeno non in maniera sicura. Sia il LEGO NXT che il software NXC non offrono la possibilità di impostare un tempo di campionamento, e non presentano nessuna funzione utile al suo calcolo. Si è reso necessario, quindi, escogitare una modalità per rendere costante e sicuro il tempo di clock. La nostra soluzione è stata quella di far leggere il tempo all'inizio del ciclo t ($t = CurrentTick()$), sommare a questo il tempo di clock ottenedo t_0 ($t_0 = t + T_{clock}$), e permettere la chiusura del ciclo solo dopo aver verificato che il contatore sia maggiore o uguale di t_0.

```
while(1) {
t0 = t + 20
t = CurrentTick() - t1
...
while(t0 > t) {
t = CurrentTick()
}
```

La scelta del valore del T_{clock} non è stata immediata; sono state eseguite numerose analisi sperimentali al fine di determinare un tempo sufficientemente piccolo da poter permettere un controllo fluido, ma abbstanza grande

da poter permettere il completamento del programma. L'impostazione di un determinato T_{clock} è anche in funzione della complessità del programma e della quantità di operazioni richieste al mattoncino. Aumentare la "difficoltà" del programma richiede tempi di calcolo più lunghi, il che comporta a sua volta il rischio di non avere più un controllo sicuro sul tempo di campionamento (potrebbe verificarsi che il tempo del ciclo risulta maggiore del T_{clock} scelto). Un buon compromesso si è rivelato un T_{clock} di 20ms.

Per il conteggio del tempo ($CurrentTick()$) viene utilizzato un contatore interno che si attiva al momento dell'accensione del mattoncino (e non all'avviamento del programma). Questo contatore veniva letto dal programma con una variabile a 16 bit unsigned, e quinidi dopo 65536 conteggi, circa 65 secondi (1000 unità formano un secondo), si riazzerava. Proprio per questo fenomeno e per come abbiamo creato il campionamento è insorto un problema: il segway durante il suo funzionamento, a intervalli di tempo non regolari, si bloccava smettendo senza una ragione apparente di funzionare. In realtà il fenomeno non èra casuale: quando il tempo t assume il valore $t = 65516$ il software somma il T_{clock}, nel nostro caso 20 ms, portando t_0 al valore di 65536. Una volta completate le operazioni entrava nel ciclo while $(t_0 > t)$ che assicura la non uscita da questo finchè t non supera il valore di t_0. Ma avendo il valore limite di 65536 èra impossibile per il contatore avere un numero maggiore. Ciò si traduceva in un loop infinto che impediva al Legway di continuare a funzionare.

Sono state individuate due possibili soluzioni a questo problema. La prima è l'introduzione (come già fatto per i motori) di un limite superiore al contatore: questo comporta sì la perdita di informazioni sul campionamento ma assicura la continuità di funzionamento. Ponendo inoltre come valore massimo un numero prossimo a 65536, l'incertezza nel clock diventa minima. Abbiamo imposto come valore di soglia 65500. Se t risulta maggiore o uguale a tale numero t_0 assume il valore 1. Quindi, finchè il CurrentTick non avrà superato il valore 1 non avremo nessun controllo sul tempo del ciclo, senza peró compromettere la stablità del sistema, perché nel caso peggiore questo effetto dura 36 ms. La seconda invece sfrutta i bit a disposizione per il conteggio, attribuendo al Current Tick non più 16bit ma bensì 32bit. In questo modo non si modifica in nessun modo l'algoritmo di controllo (il Legway

potrebbe ancora entrare in un loop infinito), si aumenta semplicemente il valore massimo del conteggio. Se 16 bit corrispondono a circa 65 secondi; 32 bit corrispondono a circa 1193 ore, è dunque impensabile che si raggiungano certi tempi di funzionamento. Questa seconda soluzione è stata individuata successivamente rispetto alla prima ma è senza dubbio migliore.

5.5 Joystick

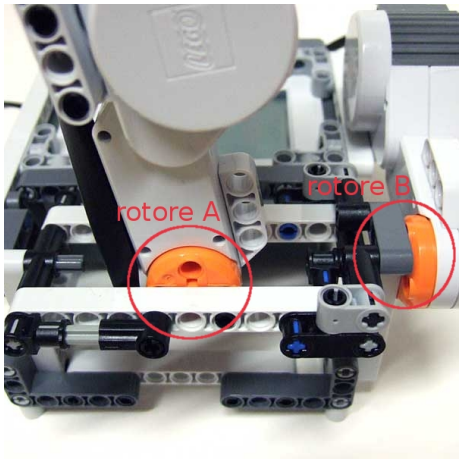
Il Legway è risultato un ottimo esperimento, il controllo è stabile e l'equilibirio può essere mantenuto senza limiti di tempo. Tuttavia, sarebbe molto piú interessante se vi fosse la possibilità di pilotarlo e quindi di far variare la sua posizione senza influenzare la stabilità. Questo aspetto si può tradurre in pratica con la creazione di un joystick. Per realizzare questo controllo è necessario:

- realizzare la struttura meccanica del joystick
- creare la comunicazione bluetooth tra Legway e joystick

5.5.1 Struttura meccanica joystick

Le specifiche richieste per un joystick sono semplici.

- leggere la rotazione lungo il piano xz
- leggere la rotazione lungo il piano yz



La misurazione di questi angoli avviene sfruttando gli encoder degli attuatori. La prima lettura corrisponde ad una inclinazione in avanti o indietro del motore A (fig 5.10) e deve, quindi, far avanzare o arretrare il Legway. La seconda lettura invece è legata alla rotazione a destra o sinistra del motore A, che essendo connesso al motore B, ne provoca la rotazione del rotore (fig. 5.10). Questa particolare struttura meccanica riesce a minimizzare i momenti torcenti che si vengono a creare e che risultano fastidiosi durante l'utilizzo del joystick. Il sensore di touch (indicato con C in figura) può essere utilizzato per compiere operazioni occasionali (allarme, stop, etc.); nel nostro algoritmo è stato utilizzato per comandare l'emissione di un suono.

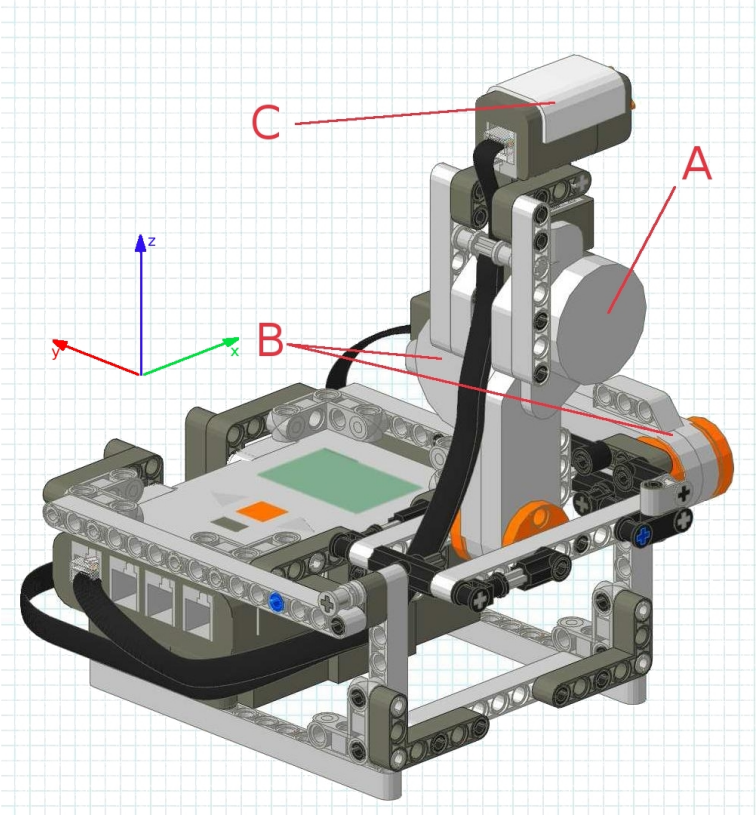


Figura 5.10: joystick LDD

5.5.2 Software e comunicazione

Il programma che viene eseguito dal joystick è molto semplice, la problematica principale consiste nella comunicazione tra i due NXT, di cui abbiamo già posto le basi. Il Joystick è definito come "master" e il legway come "slave"[5]. È presente un singolo task che esegue un ciclo infinito che si ripete ogni circa 80 ms. Prima dell'ingresso al ciclo il programma esegue un controllo che testa la connessione con l'altro NXT. All'interno si eseguono le seguenti operazioni:

- Lettura dei due motori che rappresentano l'asse x e y
- Invio dei dati letti al legway
- Generazione di un feedback ai motori proporzionale alla lettura degli stessi.

Listato in appendice (A-2).

[5]Questo viene fatto utilizzando il menù dell'NXT prima di avviare il programma

Capitolo 6

Conclusioni e obiettivi futuri

Le finalità della nostra tesi si possono dire raggiunte, anche se non è stato possibile completare l'esperimento relativo al pendolo di Furuta. Abbiamo studiato e testato le potenzialità dell'NXT cercando di ottimizzarne le prestazioni introducendo componenti aggiuntivi come ad esempio il software di scrittura, e gettando le basi per un futuro sviluppo di esperimenti più complessi.

In questo lavoro ci siamo concentrati nel creare il controllo di un singolo robot, il passo successivo potrebbe essere quello di creare più unità capaci di interagire tra loro, svolgendo compiti semplici ma diversi, in modo da ottenere un lavoro di gruppo di complessità superiore. In questa ottica si può immaginare che sia un robot a coordinare gli altri, si è visto che un NXT è capace di creare fino a 3 connessioni, per un totale di 4 robot; oppure, con una visione più ampia, il coordinamento potrebbe essere svolto da un'unità esterna(non è stato identificato un limite di connessioni tra un computer e gli NXT).

La semplicità con cui questi oggetti possono essere creati, modificati o smontati costituisce uno dei principali vantaggi del loro utilizzo: se una struttura presenta dei problemi, questa può essere modificata o addirittura smontata e rimontata con la tipica semplicità di un gioco per bambini, il risparmio di tempo e risorse è notevole. Alla luce di questo potrebbe essere interessante l'utilizzo di queste apparecchiature nella fase progettuale di macchine anche sofisticate: la realizzazione di un modello a basso costo con il

Lego potrebbe identificare errori di progetto o problematiche non modellate, almeno di carattere macroscopico, che potrebbero anche essere risolte con un processo iterativo di modifica strutturale.

Nel periodo in cui è stato svolto il nostro lavoro, è stato migliorato e completato il software Embedded Code Robot: quel programma che permette di utilizzare Simulink per l'utilizzo dell'NXT, di cui abbiamo parlato nel secondo capitolo. Con i nuovi sviluppi sembra adesso possibile interfacciare Matlab con il Brick, una cosa che ci eravamo prefitti di fare all'inizio dei lavori. Non abbiamo testato questo software, ma potrebbe rivelarsi interessante per gli sviluppi futuri.

Si ritiene dunque che questo strumento possa essere considerato come qualcosa di più di un semplice giocattolo: sicuramente un ottimo aiuto all'educazione e forse anche una piccola risorsa in campo scentifico.

Appendice A

Listati

A.1 Legway

```
 #include "NXCDefs.h"

int a, b, v, d, p, Vref, Vref2, s, z, w0, g, x,, v2, y, l;
unsigned long t0, t1, t;

#define BT_CONN 1
#define IB1 5
#define IB2 6
#define OB 1

task speed() {
while(1)    {
b = SENSOR_2 - SENSOR_3
z = (MotorRotationCount(OUT_B)+MotorRotationCount(OUT_A))/2
v =  v/4 + 150*(b - d)
v2 = v2*9/10 + 20 * (z - y)
d = b
y = z
Wait(5)
}
}

task command() {
```

```
while(1) {

t0 = t + 20
t = CurrentTick() - t1
ReceiveRemoteNumber(IB1, false, a)
ReceiveRemoteNumber(IB2, false, x)

p = (b - Vref + z/10 + a/3 )*6/5 + v/45 + v2/8
g = (p + x/5)
l = (p-x/5)
if(l > 100) {
l = 100
}
if (l<-100) {
l = -100
}

if(g > 100) {
g = 100
}
if (g<-100) {
g = -100
}
OnFwd(OUT_A, l)
OnFwd(OUT_B, g)
SendResponseNumber(OB, p)
while(t0 > t) {
t = CurrentTick()
}

}
}


task main() {
SetSensorType(S2, SENSOR_TYPE_LIGHT_ACTIVE)
SetSensorMode(S2, SENSOR_MODE_RAW)
SetSensorType(S3, SENSOR_TYPE_LIGHT_ACTIVE)
SetSensorMode(S3, SENSOR_MODE_RAW)
Wait(100)
Vref = SENSOR_2 - SENSOR_3
```

```
t1 = CurrentTick()
Precedes(speed, command)
}
```

A.2 Joystick

```
#include "NXCDefs.h"
#define BT_CONN 1
#define OB1 5
#define OB2 6
int p, a, x, b;
sub BTCheck(int conn){
    if (!BluetoothStatus(conn)==NO_ERR){
        TextOut(5,LCD_LINE2,"Error");
        Wait(1000);
        Stop(true);
    }
}
task main(){
SetSensor(S1, SENSOR_TOUCH);
BTCheck(BT_CONN);
while(1) {
b = MotorRotationCount(OUT_A);
x = -MotorRotationCount(OUT_B);
a = a + b/10;
x = x * 2;
ResetScreen();
NumOut(0,LCD_LINE1, a);
NumOut(0,LCD_LINE3, x);
NumOut(0,LCD_LINE5, p);
//OnFwd(OUT_A, p*2);
SendRemoteNumber(BT_CONN, OB1, a);
Wait(30);
SendRemoteNumber(BT_CONN, OB2, x);
Wait(30);
if(SENSOR_1==1) {
RemotePlaySoundFile(BT_CONN, "bullethit.rso", false);
}
Wait(20);
}
}
```

Bibliografia

[1] *Lego Mindstorms homepage*, http://www.mindstorms.com.

[2] *Lego education*, http://www.lego.com/education/default.asp

[3] Philippe Hurbain, *LEGO Mindstorms NXT*, http://www.philohome.com/nxt.htm

[4] Steve Hassenplug, *NXT Programming Software*, http://www.teamhassenplug.org/NXT/NXTSoftware.html

[5] Ryo Watanabe, http://web.mac.com/ryo_watanabe

[6] *Robolab*, http://www.lego.com/eng/education/mindstorms/home.asp?pagename=robolab

[7] *LabVIEW Toolkit for LEGO MINDSTORMS NXT*, http://zone.ni.com/devzone/cda/tut/p/id/4435

[8] *Microsoft Robotics Studio Developer Center*, http://msdn2.microsoft.com/en-us/robotics/default.aspx

[9] *NXT OnBrick*, http://www.pspwp.pwp.blueyonder.co.uk/science/robotics/nxt/index.html

[10] *RobotC*, http://www.robotc.net/

[11] *Bricx Command Center 3.3*, http://bricxcc.sourceforge.net/

[12] *What is Embedded Coder Robot NXT?*, http://lejos-osek.sourceforge.net/ecrobot_nxt.htm

[13] John Hansen, *NXT Power Programming*, 2008

[14] Emmanuele Serra, *Realizzazione di Sistemi di Controllo Prototipali mediante Lego Mindstorms*, Tesi, UNIVERSITÀ DEGLI STUDI DELL'AQUILA, A.A. 2005/2006

Ringraziamenti

Desideriamo ringraziare tutte le persone che ci sono state vicine durante la preparazione di questa tesi. In special modo vogliamo ringraziare il Prof. Michele Basso, il Dott. Massimo Vassalli e il Dott. Franco Quercioli, che ci hanno supportato e incoraggiato, fornendoci nuovi stimoli e motivazioni nei momenti più difficili.

Vogliamo inoltre ringraziare per l'ospitalità e la disponibilità del personale dell'Istituto Sistemi Complessi, in particolare il dott. Giampiero Puccioni e il dott. Bruno Tiribilli, che ci hanno subito accolto, mettendo a nostra disposizione apparecchiature e fornendo un prezioso supporto tecnico tutte le volte che se ne è verificata la necessità.

Un pensiero particolare va anche a Philippe Hurbain e John Hansen, sempre pronti a rispondere alle nostre e-mail e a Chris Rogers, professore della Tufts University (Medford, MA), che durante la visita al CNR si è mostrato attento e disponibile.

***Iacopo**: Voglio ringraziare la mia famiglia che mi ha sopportato in questo periodo difficile, tutti i miei amici e i Flying Bisc, sempre presenti e capaci di far vedere il lato positivo delle cose, componente essenziale per giungere a questo risultato. Per ultima ringrazio la Martina che comunque mi ha dato la forza per completare questo lavoro.*

***Niccolò**: Ringrazio la mia famiglia, la mia ragazza e tutti i miei amici per l'aiuto fornito, in particolare i compagni con cui ho condiviso questa avventura.*