Università degli Studi di Padova

FACOLTÀ DI INGEGNERIA
Corso di Laurea in Ingegneria Informatica

Tesi di laurea triennale

Sviluppo di alcuni esempi robotici educativi : caratterizzazione ingegneristica

Candidato:
Camerin Matteo
Matricola 563828

Relatore:
Michele Moro

Anno Accademico 2010–2011

Indice

Sommario **5**

1 Introduzione **7**
1.1 Il progetto TERECoP 8
1.2 Strumenti utilizzati 9
1.2.1 Tecnologia Mindstorm NXT 9
1.2.2 Il linguaggio NXT-G 10
1.3 Prerequisiti necessari 10

2 Tecnologia Lego Mindstorm NXT **11**
2.1 Brick NXT 11
2.2 Accelerometro HiTechnic NXT 12
2.3 Sensore ad ultrasuoni 13
2.4 Software NXT-G (LEGO MINDSTORMS Education NXT Programming 2.0) 15
2.4.1 Struttura del programma 15
2.4.2 Problematiche dovute all'utilizzo di NXT-G versione 2.0 . 17
2.4.3 Motivazioni della scelta 20

3 Motivazioni didattiche delle esperienze con robot educativi **23**
3.1 Il Discovery on film festival 23
3.1.1 La Discovery Arena 25
3.2 Argomenti didattici collegati ai Robot 26

4 Il verme **31**
4.1 In natura 31
4.2 La teoria 33
4.3 Simulazione con LEGO ®MINDSTORMS® NXT 35
4.4 Il programma 40

5 La "visione stereoscopica" **43**
5.1 In natura 43
5.2 La teoria 44
5.3 Simulazione con LEGO ®MINDSTORMS® NXT 48
5.4 Il programma 57

6 L'equilibrio **63**
6.1 In natura 63
6.2 La teoria 64
6.3 Simulazione con LEGO ®MINDSTORMS® NXT 67
6.4 Il programma 74

Conclusioni **79**

Conclusioni **81**

Sommario

La seguente tesi ha lo scopo di illustrare la progettazione, la costruzione e la programmazione di tre robot educativi basati sulla tecnologia LEGO®MINDSTORMS®NXT che prendono spunto da comportamenti e possibilità proprie di alcuni esseri viventi nella natura che ci circonda. In particolare si rappresenteranno il movimento del verme, la "visione stereoscopica" di un pipistrello ed un esempio semplificato di come mantenere un corpo in equilibrio. Lo scopo di queste rappresentazioni,ed in particolare di tutta robotica educativa, è quello di utilizzare dei semplici robot dimostrativi per introdurre agli studenti fondamentali nozioni teoriche usando però un approccio più leggero e soprattutto più interessante, permettendo loro inoltre di verificare sul campo la validità di quanto viene loro insegnato.

Nella camminata del verme verrà utilizzata una struttura molto semplice composta da una coppia di ruote governata da un motore alle due estremità ed uno snodo libero nel mezzo. I due motori, con il loro movimento alternato, faranno compiere al robot il movimento a fisarmonica tipico di alcuni vermi.

Per quanto concerne la "visione stereoscopica" il robot costruito è in grado di determinare la posizione di un oggetto posto nel raggio d'azione dei propri sensori ultrasonici, indicando, tramite una visualizzazione a schermo ed a un segnale sonoro, se l'oggetto si trova a destra, leggermente a destra, davanti, leggermente a sinistra o a sinistra. Nella simulazione viene utilizzato un automa che si muove avanti ed indietro su una linea parallela rispetto a quella dove è situato il visore stereoscopico. Tale movimento verrà percepito dal visore che indicherà quindi quale posizione è stata assunta dall'oggetto tra le 5 definite.

Nell'ultimo esempio verrà riprodotta in modo semplificato, la capacità di mantenersi in equilibrio del corpo umano e di certi animali. Verrà utilizzato un robot composto da un normale carro cingolato con sopra montata una coppia di motori. Questi due motori serviranno per far muovere a destra, sinistra, avanti o indietro una colonna verticale che rappresenta il tronco del corpo, sopra la quale

sarà montato un accelerometro per percepire le pendenze cui il carro è sottoposto. La colonna sarà in grado di autobilanciarsi grazie all'applicazione delle leggi del pendolo inverso. Per mantenere in equilibrio tutto il sistema, verrà utilizzato un controllore PI.

Per la realizzazione degli esempi si utilizzerà il kit LEGO ®MINDSTORMS® NXT. La programmazione verrà effettuata utilizzando l'ambiente di programmazione NXT-G.

Capitolo 1

Introduzione

Il seguente progetto è stato concepito per creare degli esempi di robot educativi allo scopo di introdurli nelle scuole per favorire l'insegnamento di diversi concetti didattici. Questo lavoro si è originato a valle della conclusione del progetto TERECoP, un progetto didattico internazionale partito nell'ottobre dell'anno 2006, nel quale è inserito anche il Dipartimento di Ingegneria dell'Informazione dell'Università di Padova.

La seguente tesi è stata eseguita con lo scopo di progettare, costruire e programmare 3 robot educativi, ognuno dei quali rappresenta un aspetto particolare di alcuni esseri viventi presenti in natura. L'idea di base che si cela dietro alla progettazione di questi robot, è quella di insegnare dei concetti teorici utilizzando esempi robotici costruiti ad hoc. L'utilizzo di robot nel campo dell'istruzione, ha come obbiettivo principale, quello di far crescere l'interesse dello studente, nei confronti delle diverse materie che questo si trova a dover affrontare tutti i giorni. È facile immaginare che uno studente, trovandosi nella situazione di dover costruire e programmare dei semplici robot, sia più propenso ad apprendere le nozioni teoriche che si celano dietro il funzionamento di questi automi. Questa possibilità di verificare sul campo la validità delle nozioni teoriche, permette anche una più semplice comprensione dei concetti da parte degli studenti; infatti molto spesso è più semplice capire il funzionamento di una cosa osservandone il comportamento, piuttosto che cercando di immaginarne il funzionamento in base alle nozioni teoriche assimilate.

Il livello di difficoltà nella costruzione di questi robot, è abbastanza ridotto e prevede l'utilizzo della tecnologia LEGO ®MINDSTORMS®. Questa tecnologia è stata sviluppata per consentire la realizzazione di semplici robot, senza la necessità di possedere delle competenze tecniche particolari. La facilità di utilizzo di questi strumenti,

ne consente l'uso anche agli studenti delle scuole elementari, ecco perchè questa tecnologia è l'ideale per questo tipo di progetto.

Gli esempi di questa tesi sono stati creati per rappresentare comportamenti e possibilità proprie di alcuni esseri viventi presenti in natura.

Il primo esempio riproduce il movimento di camminata tipico del verme. In questa rappresentazione verrà costruito un robot in grado di muoversi nel terreno imitando il movimento peristaltico tipico dei vermi.

Il secondo esempio prevede la costruzione di un robot capace di riconoscere la posizione degli oggetti, utilizzando lo stesso principio che caratterizza la "vista" dei pipistrelli. Come accade per i pipistrelli infatti, anche questo robot emette onde ultrasoniche per rilevare gli oggetti nello spazio. Tramite l'utilizzo combinato di due sensori ultrasonici, questo robot è in grado di distinguere le posizioni degli oggetti nel proprio campo visivo, indicando all'utente se un oggetto si trova: molto a destra, leggermente a destra, in posizione centrale, leggermente a sinistra, molto a sinistra. "vedere" gli oggetti in movimento.

Il terzo ed ultimo esempio, simula le capacità di mantenersi in equilibrio tipiche dell'uomo e di alcuni animali. Per questa rappresentazione verrà utilizzato un robot composto da un carro cingolato con sopra montata una colonna verticale. La simulazione prevede che il robot, grazie all'utilizzo di due motori e un accelerometro, mantenga la colonna in equilibrio durante la sua marcia lungo un percorso con diverse pendenze frontali e laterali.

Lo scopo della realizzazione di questi esempi è quello di utilizzare la robotica per rappresentare nella pratica, alcuni importanti concetti teorici. Utilizzando questi robot infatti è possibile trattare argomenti come ad esempio i teoremi sui triangoli, oppure è possibile introdurre tutta una serie di argomenti di fisica come quello del "pendolo inverso".

1.1 Il progetto TERECoP

TERECoP (Teacher Education on Robotics-Enhanced Constructivist Pedagogical Methods) è un progetto finanziato dalla comunità europea che ha l'obiettivo di diffondere l'utilizzo dei robot nelle scuole come aiuto all'apprendimento scolastico. Questo progetto prende ispirazione dalle teorie costruttiviste dello psicologo e pedagogista svizzero Jean Piaget e dalla filosofia didattica costruzionista del matematico sudafricano Seymour Papert. Le teorie di Pi-

aget sostengono che l'apprendimento non sia tanto il risultato di un passaggio di conoscenze, ma un processo attivo di costruzione della conoscenza basato su esperienze empiriche ricavate dal mondo reale e collegate a preconoscenze uniche e personali (Piaget, 1972). La filosofia di Papert, invece, introduce l'idea che il processo di apprendimento risulti decisamente più efficace qualora vengano introdotti artefatti cognitivi, ovvero oggetti e dispositivi che si basino su concetti familiari allo studente.

Il progetto TERECoP applica i concetti appena citati utilizzando i robot, i quali vengono realizzati grazie al sistema LEGO ®MINDSTORMS® NXT. Questo sistema si sposa benissimo con il progetto TERECoP, infatti oltre a non richiedere ingenti investimenti monetari , consente anche di realizzare strutture robotiche in tempi limitati e con un'assoluta semplicità sia nella costruzione(mattoncini, ruote, motori, sensori ed ingranaggi sono facili da assemblare) sia nella programmazione (il software di base per controllare i robot offre un'interfaccia grafica molto semplice ed intuitiva). Questi strumenti rendono possibili nuovi tipi di esperimenti scientifici, grazie ai quali lo studente può comprendere attraverso l'esperimento pratico i fenomeni fisici della vita quotidiana.

1.2 Strumenti utilizzati

1.2.1 Tecnologia Mindstorm NXT

Il kit LEGO ®MINDSTORMS® fornisce una serie di elementi che consentono di costruire dei semplici robot. Il kit fornisce un Brick NXT che è la componente principale del robot ed è la parte che verrà poi programmata per far compiere all'automa le azioni per cui è stato concepito. Nel kit inoltre vengono forniti 3 servomotori e 4 sensori (di luce, ad ultrasuoni, di tatto e di suono). Verrà utilizzato inoltre un accelerometro che però attualmente va acquistato separatamente al kit. Tutti questi elementi verranno impiegati per la costruzione della parte hardware dei robot.

1.2.2 Il linguaggio NXT-G

NXT-G è un linguaggio iconico per programmare il robot LEGO Mindstorm NXT basato sulla tecnologia LabVIEW della National Intruments. L'intuitiva interfaccia grafica fornita dal software ne rende la programmazione veloce e di facile utilizzo. Questo programma verrà utilizzato per tutte e tre le rappresentazioni presenti nella tesi. Ulteriori dettagli su NXT-G saranno forniti nel capitolo seguente.

1.3 Prerequisiti necessari

Per una corretta comprensione degli argomenti trattati in questa tesi, il lettore deve essere in possesso di una buona conoscenza nel campo della fisica, dell'algebra e della geometria. In particolare un lettore deve essere in possesso dei seguenti requisiti:

- conoscenze di base nel campo della dinamica dei corpi rotatori
- buone conoscenze dei tre principi fondamentali della dinamica e delle loro applicazioni
- conoscenze generali sulle forze di attrito e sui loro funzionamenti
- buona dimestichezza con i teoremi geometrici sui triangoli
- ottime conoscenze di algebra.

Nel caso un lettore non sia in possesso dei requisiti appena citati, la comprensione di alcuni passaggi presenti in questa tesi, può risultare difficile.

Capitolo 2

Tecnologia Lego Mindstorm NXT

2.1 Brick NXT

Il brick NXT (figura 2.1) è il cervello del robot ®MINDSTORMS® e ad esso possono essere collegati 4 sensori e 3 servomotori. Il mattoncino intelligente viene programmato tramite computer per far eseguire al robot i compiti per cui è stato creato.

Figura 2.1: Brick NXT.

Note tecniche:

- Microcontrollore ARM7 a 32 bit
- FLASH a 256 Kbyte, ARM a 64 Kbyte
- Microcontrollore AVR a 8 bit
- FLASH a 4 Kbyte, ARM a 512 byte
- Tre porte d'uscita destinate al collegamento dei motori

- Quattro porte d'ingresso destinate alla connessione dei sensori
- Quattro tasti per poter interagire con il brick.
- Dispaly LCD con 100x64 pixel per la visualizzazione dei dati
- Alimentazione: batteria a 6AA
- Altoparlante per l'emissione di suoni con qualità del suono pari a 8KHz dotato di un canale con una risoluzione pari ad 8 bit ed una velocità di campionamento di 2-16KHz
- Connessione Bluetooth (Bluetooth Class II V2.0 compliant) e porta USB (12 Mbit/s) per il collegamento dell'NXT al computer permettendo il download dei programmi da eseguire oltre alla possibilità di effettuare aggiornamenti del firmware.

2.2 Accelerometro HiTechnic NXT

(a) *Il sensore.* (b) *Orientamento degli assi.*

Figura 2.2: L'accelerometro HiTechnic NXT.

L'accelerometro della HiTechnic che viene utilizzato in questa tesi, serve a misurare le accelerazioni dei tre assi (X,Y,Z) che vengono subite dal robot su cui è montato. Questo sensore è in grado di misurare accelerazioni da -2G a + 2G fornendo valori nel range [-400, +400]. Questo accelerometro può essere collegato al brick NXT come un qualsiasi altro sensore che viene fornito con il kit LEGO e utilizza il protocollo di comunicazione digitale I2C. Il sensore è in grado di misurare le accelerazioni con una frequenza di 100 volte al secondo. Il sensore è in grado di misurare anche le accelerazioni quando il robot è fermo grazie alla forza di gravità. L'accelerometro infatti percepisce le accelerazioni fornite da tale forza e quindi può effettuare le misurazioni anche trovandosi in una situazione di assoluta immobilità. Il sensore calcola le accelerazioni misurando l'inerzia di una massa quando viene sottoposta ad un'accelerazione. La

massa viene sospesa ad un elemento elastico, mentre un qualche tipo di sensore ne rileva lo spostamento rispetto alla struttura fissa del dispositivo. In presenza di un'accelerazione, la massa, che è dotata di una propria inerzia, si sposta dalla propria posizione di riposo in modo proporzionale all'accelerazione rilevata, coerentemente con la seconda legge del moto di Newton $F = m \cdot a$. Ovviamente si ha la necessità che l'elemento elastico abbia una costante di allungamento K lineare rispetto alla forza F. La relazione sarà del tipo $F = K \cdot s$ dove s è lo spostamento. Il sensore trasforma questa variazione di posizione in un segnale elettrico acquisibile dai sistemi di misura. È necessario sapere ai fini di un corretto utilizzo del sensore in fase di programmazione, che questo non necessita di alcun tipo di calibrazione.

Per poter utilizzare l'accelerometro nel linguaggio di programmazione NXT-G è necessario importare il blocco, relativo a tale sensore, tramite l'apposita funzione "*procedura guidata di importazione/esportazione blocchi*" presente negli strumenti del programma. Ovviamente il blocco dell'accelerometro NXT deve essere precedentemente scaricato dal sito della "HiTecnic", la casa produttrice di tale sensore, prima di poter essere importato.

2.3 Sensore ad ultrasuoni

Figura 2.3: Sensore ad ultrasuoni

Questo tipo di sensore è stato concepito allo scopo di misurare le distanze, in centimetri o in pollici , da oggetti posti davanti al sensore all'interno di un certo angolo di 'visione' . Il principio di funzionamento è molto semplice: il sensore emette onde ultrasoniche a frequenze che variano nel range da 40 a 250KHz e che risultano quindi non udibili all'orecchio umano, il quale percepisce onde con

frequenze inferiori a 20kHz. Le onde emesse dal dispositivo vagano nell'aria fin che non trovano una superficie sulla quale possono riflettersi e tornare quindi verso il sensore. Una volta che l'onda riflessa raggiunge il sensore, questo calcola la distanza dall'oggetto con questa formula:

$$L = \frac{V * \Delta t}{2} \tag{2.1}$$

dove V è la velocità di propagazione delle onde e Δ t è il tempo che intercorre dalla trasmissione alla ricezione dell'onda.

Come l'occhio umano anche questo sensore ha un "campo visivo" oltre al quale non può andare. Nel caso specifico il sensore ha un angolo di visione di circa 30° e può misurare distanze non superiori a 255 cm (se non dovesse esserci nessun oggetto entro questa distanza, il sensore ritorna il valore 255).

Bisogna considerare, quando si utilizza questo tipo di sensore, che ci sono molte cause che possono portare a misurazioni errate o imprecise. Le principali problematiche sono date dall'ambiente: infatti la velocità delle onde viene influenzata sia dalla temperatura che dall'umidità. L'umidità influisce sul segnale attenuandolo, riducendo così la distanza che l'ultrasuono può raggiungere. La temperatura invece influisce sul calcolo della distanza in quanto la velocità del suono dipende dalla temperatura del mezzo trasmissivo. La velocità del suono in aria si può determinare dalla formula:

$$v(T) = 0,3261 \cdot \sqrt{1 + (\frac{T}{273})} \tag{2.2}$$

dove $v(T)$ è misurata in $\frac{m}{s}$ e T è la temperatura dell'aria in °C. Si nota quindi che al variare della temperatura, varia anche la velocità dell'onda. Bisognerebbe quindi tarare lo strumento in base alla temperatura ambientale rilevata all'istante di misurazione.

Oltre a questi motivi, misurazioni errate possono essere provocate da rumori di fondo, dalla presenza di più sensori di questo tipo nell'ambiente collegati a robot indipendenti (le onde potrebbero essere scambiate) ed infine anche dal tipo di superficie dell'oggetto di cui si vuole la misura(se l'oggetto non è perpendicolare al ricevitore l'onda riflessa potrebbe non raggiungere il ricevitore). In fase di programmazione si dovranno applicare dei filtri alle misurazioni per scartare valori fuori dai range ammessi dal problema, evitando così che si presentino sequenze di valori molto fluttuanti che provocherebbero comportamenti fastidiosi.

2.4 Software NXT-G (LEGO MINDSTORMS Education NXT Programming 2.0)

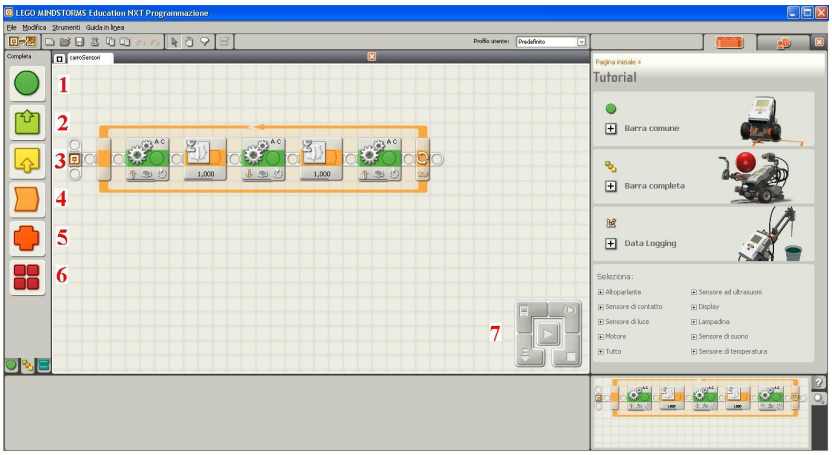


Figura 2.4: Schermata del programma NXT-G

Per la programmazione di tutti e tre i robot è stato utilizzato il Software NXT-G, un linguaggio iconico basato sul paradigma della cosiddetta *Graphic Language*. Questo stile di programmazione è contraddistinto dall'assenza di codice scritto sotto forma di testo; infatti la definizione di ogni componente avviene tramite icone ed oggetti grafici, che solo in alcuni casi, richiedono la specifica di parametri in forma testuale, comunque forniti all'interno di specifiche interfacce grafiche. L'utilizzo della programmazione grafica semplifica la gestione dei vari componenti come motori o sensori, oltre a facilitare la definizione delle azioni da far compiere al robot. Vediamo ora nel dettaglio quali sono le caratteristiche principali di NXT-G.

2.4.1 Struttura del programma

In figura 2.4 possiamo notare come un programma creato con NXT-G non sia altro che una sequenza di blocchi collegati tra loro. Il software LEGO MINDSTORMS Education, per favorire la ricerca dei vari blocchi al programmatore, li suddivide in sei categorie :

- *blocchi comuni*: (in Figura 2.4 rappresentati dal numero 1) questa categoria racchiude i blocchi che più comunemente vengono utilizzati nei programmi NXT-G, come l'icona del movimento, quella del display, del riproduttore di suoni o anche

quella che permette di ripetere ciclicamente una sequenza di blocchi di programma.

- *blocchi azione*: in Figura 2.4 rappresentati dal numero 2) in questa classe troviamo tutti quei blocchi che fanno svolgere delle azioni al robot come il blocco di movimento, il riproduttore di suoni, il blocco display o quello per inviare messaggi via bluetooth.

- *blocchi dei sensori*: in Figura 2.4 rappresentati dal numero 3) qui possiamo trovare tutti i blocchi che permettono di gestire i vari sensori disponibili con il kit come quello sonoro, il sensore di luce o quello di contatto. Non troveremo invece qui quei sensori che vengono forniti separatamente, come ad esempio l'accelerometro che verrà usato successivamente.

- *blocchi di flusso*: in Figura 2.4 rappresentati dal numero 4) di questa categoria fanno parte tutti i blocchi vanno ad alterare il flusso del programma, come il blocco iterazione, quello di interruzione(per esempio lo switch), il blocco attesa e quello di stop.

- *blocchi dati*: in Figura 2.4 rappresentati dal numero 5) di questa classe fanno parte tutti i blocchi dati, variabili e costanti, oltre a tutti i blocchi che rappresentano le operazioni aritmetiche, logiche o di comparazione.

- *blocchi avanzati*: in Figura 2.4 rappresentati dal numero 6) nell'ultima categoria troviamo quei blocchi che non rientrano in tutte quelle descritte precedentemente, qui troviamo blocchi per gestire le stringhe(ad esempio la conversione da numero a stringa) e tutti quei blocchi che vengono integrati successivamente all'installazione del programma, come ad esempio quello dell'accelerometro che deve essere integrato successivamente all'installazione.

Ogni blocco di ciascuna delle categorie appena citate, offre la possibilità di impostare dei parametri manualmente, sempre con l'ausilio di un'interfaccia grafica. Un utente può quindi inserire il valore delle variabili, scrivere del testo a schermo scegliendo anche la posizione esatta su cui visualizzare la stringa, oppure potrebbe inserire il numero di giri da far fare ad un motore.

Tutti i blocchi descritti fin qui hanno anche la possibilità di interagire tra loro. L'iterazione tra elementi diversi del programma è rappresenta da una linea di colore giallo, chiamata 'collegamento

dati', che collega quei blocchi che necessitano di interagire tra loro. L'iterazione tra blocchi consente di eseguire operazioni matematiche tra una o due variabili, consente di far visualizzare il valore di una variabile a schermo e permette l'assegnazione di valori alle variabili.

Un semplice esempio di codice NXT-G è rappresentato in figura 2.5. In questo codice viene assegnato un valore alla variabile numerica 'X' tramite interfaccia grafica. A questa variabile viene sottratto un valore, sempre fornito tramite interfaccia grafica e il risultato di questa operazione viene prima di tutto convertito in stringa e poi viene visualizzato sul display. Tutta questa operazione viene ripetuta fino a che l'utente non interrompe il programma tramite gli appositi tasti presenti sul brick NXT, in quanto è presente un ciclo infinito che racchiude tutti gli elementi del programma.

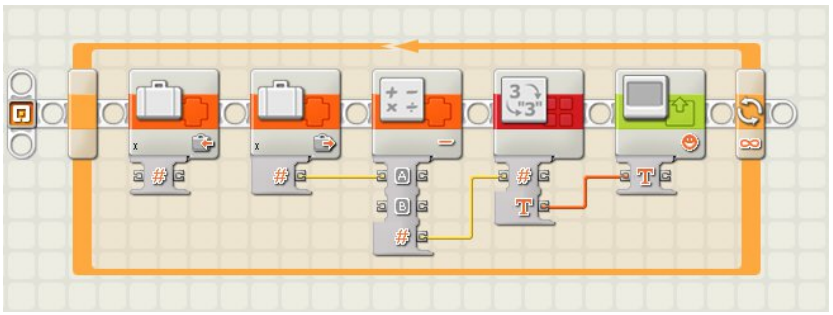


Figura 2.5: Semplice esempio di codice NXT-G

2.4.2 Problematiche dovute all'utilizzo di NXT-G versione 2.0

La semplicità della programmazione NXT-G, come spesso accade, comporta una serie di svantaggi di cui bisogna tenere conto quando si va ad utilizzare questo tipo di linguaggio. In primo luogo bisogna considerare che NXT-G è un linguaggio iconico e l'ambiente di programmazione risulta piuttosto oneroso da gestire in termini computazionali. Se si vanno a creare delle applicazioni abbastanza complesse, si nota immediatamente che diventa molto difficile utilizzare il programma senza l'ausilio di una macchina abbastanza potente. Infatti appena il numero dei blocchi comincia a diventare elevato, lo spostamento delle icone all'interno della schermata inizia a diventare poco fluido, rendendo impossibile la programmazione e provocando, in alcuni casi, anche il blocco del programma, con la necessità quindi di chiudere l'applicazione forzatamente, perdendo tutte le modifiche fatte fino al momento dell'ultimo salvataggio, fatto che provoca sempre un certo disagio.

Un'ulteriore problematica di NXT-G è data dai tempi di risposta lenti rispetto agli altri tipi di linguaggio come ad esempio NXC. Questo problema può essere rilevante quando si vanno a realizzare applicazioni di tipo *"real time"* che necessitano di tempi di risposta prossimi allo 0. Inoltre questi ritardi nella risposta tipici della programmazione NXT-G, portano ad avere dei rallentamenti nella lettura dei sensori e questo problema in alcuni casi è il motivo principale per cui si sceglie un altro linguaggio. La spiegazione di questo ritardo rispetto agli altri codici la si può trovare andando ad esaminare la dimensione dei file generati dai vari linguaggi. Un file medio prodotto in NXT-G si aggira intorno alle 450 righe di bytecode contro una media di 70-80 righe prodotte da NXC. Questo fatto introduce un'altra problematica che riguarda la dimensione dei file che risulta essere molto maggiore rispetto a quelle degli altri linguaggi. Questo fatto può risultare problematico visto che il Brick NXT non è dotato di una memoria molto estesa.

Uno dei più grandi problemi che ha accompagnato in passato gli utilizzatori di NXT-G , riguarda la gestione delle operazioni matematiche ed in particolare della divisione. Fin dalla versione 1.0 infatti, questo linguaggio ha sempre supportato solo variabili e operatori matematici a valori interi, creando non pochi problemi a chi doveva realizzare applicazioni di precisione. Il problema principale è dovuto alle modalità con cui NXT-G arrotonda i risultati. La scelta dei programmatori di questo linguaggio è stata quella di troncare i numeri togliendo semplicemente la parte decimale, tenendo come valore finale la sola parte intera. Così facendo però vengono introdotti degli errori anche piuttosto significativi. Supponiamo per esempio che nel nostro programma ad un certo punto debba essere effettuata una divisione di questo tipo:

$$\frac{49}{25} = 1,96 \tag{2.3}$$

il risultato della divisione 2.2 viene arrotondato al valore 1 dal programma. Andiamo ora a calcolare l'errore relativo commesso in fase di troncamento:

$$e_R = \frac{1,96 - 1}{1,96} = 48,98\% \tag{2.4}$$

Un errore di quasi il 50% risulta essere inaccettabile soprattutto quando si devono effettuare applicazioni che necessitano di una certa precisione.

Per risolvere il problema, chi scrive il programma deve essere consapevole di dover aggiungere opportune operazioni di ordinamento,

scalamento e di arrotondamento in modo da non perdere nei calcoli cifre significative. Per fare questo l'operazione di divisione deve essere effettuata utilizzando la seguente formula:

$$\frac{a}{b} = \frac{(\frac{a \cdot 100}{b}) + 50}{100} \tag{2.5}$$

Grazie a questa formula è possibile eseguire le divisioni intere arrotondando il risultato e non troncandolo come invece viene fatto utilizzando la divisione intera del linguaggio NXT-G. Testando questa formula con i numeri dell'esempio precedente si ottiene:

$$\frac{49}{25} = \frac{(\frac{4900}{25}) + 50}{100} = \frac{246}{100} = 2 \tag{2.6}$$

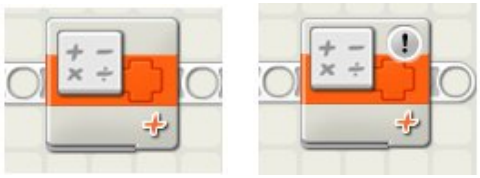
che è il risultato che ci si aspetta a seguito di un arrotondamento. Se invece si utilizza la formula 2.5 per eseguire la divisione:

$$\frac{30}{25} = \frac{(\frac{3000}{25}) + 50}{100} = \frac{170}{100} = 1 \tag{2.7}$$

il risultato viene arrotondato per difetto, ovvero al valore intero inferiore e anche in questo caso l'arrotondamento risulta corretto.

Il problema appena descritto però non si presenta nella versione del software ®MINDSTORMS® NXT-G 2.0 che viene utilizzato per la realizzazione degli esempi presenti in questa tesi. Nella versione 2.0 infatti, le variabili sono di tipo *"floating point"* mentre non sono più presenti le variabili di tipo intero. Il fatto di aver sostituito le variabili intere con quelle di tipo "floating point" si è rivelato anch'esso un problema nella realizzazione dell'esempio della "visione stereoscopica", soprattutto in assenza di un blocco NXT in grado di fornire la parte intera di un numero. In quel esempio infatti, è sorto il problema inverso rispetto a quello descritto in precedenza, ovvero c'era la necessità di arrotondare il risultato di una divisione per troncamento, mantenendo la sola parte intera. In assenza della funzione matematica che fornisce la parte intera, si è creata la necessità di cercare una soluzione alternativa per sopperire a questa limitazione di questa versione NXT-G. Grazie ad una ricerca nel web è stato possibile risolvere questo problema.

Da questa ricerca è emerso che nella versione 2.0 in realtà, sono presenti alcuni blocchi, apparentemente non visibili, che consentono di risolvere il problema. Se per esempio si apre un sorgente realizzato con una versione di NXT-G 1.x, includente un blocco di operatori matematici, si nota che questo viene visualizzato con un'icona diversa rispetto a quella che si otterrebbe prendendo il blocco matematico dalla paletta dei blocchi dati.



(a) *Icona del blocco floating point.* (b) *Icona del blocco intero.*

Figura 2.6: Icone dei blocchi matematici nelle due versioni.

In figura 2.6 sono riportate le due icone relativa al blocco matematico intero(figura b) e a quello di tipo floating point(figura a). Si può notare come nell'icona della versione intera sia presente un punto esclamativo (!) a differenza dell'icona del blocco matematico presente nelle palette della versione 2.0. Per aggiungere questo tipo di blocchi all'interno delle palette della versione 2.0 è sufficiente creare un file vuoto nella directory "BlockRegistry', con il nome del blocco e con estensione ".txt". Per esempio nel caso del blocco matematico :

[LEGO MINDSTORMS directory]/engine/EditorVIs/BlockRegistry/Numerin Operations.txt

dove al posto di "LEGO MINDSTORMS directory" va sostituito il path della cartella nella quale è installato il programma MINDSTORM NXT-G 2.0.

Questo soluzione risulta essere molto utile e rende NXT-G molto più flessibile; infatti così facendo in ogni momento un programmatore può utilizzare, a sua discrezione, o la divisione intera o quella floating point eliminando totalmente tutti i problemi che si possono creare con la divisione.

2.4.3 Motivazioni della scelta

Dopo aver letto il paragrafo precedente è lecito porsi la domanda "ma allora perché usare NXT-G come linguaggio di programmazione se presenta tutti questi aspetti apparentemente negativi? ". La risposta è che questo tipo di linguaggio ha una facilità di programmazione che non troverete in nessun altro linguaggio scritto. Infatti NXT-G è così semplice ed intuitivo che può essere utilizzato fin dall'età di otto anni. L'interfaccia grafica di cui è dotato lo rende molto intuitivo; inoltre il codice prodotto risulta di facile comprensione agli occhi di chi guarda un programma di cui non è stato l'autore. Chiunque si sia trovato almeno una volta nella situazione di dover lavorare su un codice scritto da qualcun altro, non può che apprezzare questa

caratteristica, visto che molto spesso si devono perdere delle ore per capire cosa realmente faccia un programma.

Una delle ragioni principali, se non la più importante, che ha portato alla scelta di NXT-G come linguaggio per lo sviluppo di questa tesi, è stata sicuramente la difficoltà di realizzare l'esempio del *"verme"* con il linguaggio NXC. Infatti, una volta programmato il robot rappresentante il verme con NXC, si è notato che il movimento del robot risultava discontinuo nel momento in cui si passava dall'utilizzo di un motore, all'altro. Questo effetto risultava poco gradevole e soprattutto differiva in modo vistoso dal movimento a fisarmonica tipico dei lombrichi. Considerando che gli esempi contenuti in questa tesi hanno lo scopo di rappresentare il più fedelmente possibile alcune realtà della natura, non era accettabile un movimento poco fluido del robot perché venivano a mancare le condizioni per introdurre l'argomento dello studio dei grafici seno e coseno, visto che la funzione disegnata dal robot risulta discontinua.

Come si è potuto vedere quindi dei validi motivi per non scegliere NXT-G come linguaggio di programmazione per questi esempi non si presentano visto che, nello sviluppo di questa tesi non ci sono programmi così complicati da diventare ingestibili per questo software.

Capitolo 3

Motivazioni didattiche delle esperienze con robot educativi

In questo capitolo si vogliono descrivere le motivazione per cui si è scelto di realizzare queste esperienze con i robot educativi. In primo luogo questi robot sono stati creati per essere utilizzati come supporto all'attività didattiche nelle scuole. Infatti consentono di introdurre vari argomenti che verranno successivamente trattati nel dettaglio. Un'altra motivazione che ha portato allo sviluppo di questi esempi, è stata la partecipazione del Dipartimento di Ingegneria dell'Informazione dell'Università di Padova al Discovery on Film festival presso il Museo Civico di Rovereto.Come ormai da anni accade infatti, l'università di Padova collabora alla realizzazione di del Discovery on Film festival e gli esempi trattati in questa tesi, sono stati creati allo scopo di essere presentati all'interno della manifestazione.

3.1 Il Discovery on film festival

Il Discovery on film festival è una manifestazione organizzata dal Museo Civico di Rovereto con il patrocinio di ASI(Agenzia Spaziale Italiana) e del ministero dell'università e della ricerca. Il Museo non intraprende solo l'attività di custode delle tracce d'un passato più o meno recente, ma si è fatto promotore della ricerca in ogni campo, compreso nel settore dell'innovazione tecnologica più avanzata. È proprio a questo scopo che nel 2001 è nata la mostra di documentari *Discovery on film*. Una rassegna che vede per quattro giornate Rovereto e il suo Museo protagonisti del dibattito scientifico internazionale, grazie alla proiezione di audiovisivi provenienti dal vasto repertorio del Prix Leonardo (il maggiore festival del film scientifico, in programma annualmente a Parma), ma grazie anche ad incontri

con grandi personalità della ricerca mondiale come ad esempio il padre della robotica Antal Bejczy, l'organizzatore dei Mondiali di calcio per robot Enrico Pagello, il progettista di robot per l'esplorazione d' ambienti estremi Gianmarco Veruggio (che ha dato occasione ai presenti di telecomandare l'automa sottomarino Romeo, ideato per immergersi nei mari artici al largo delle isole Svalbard) e il ricercatore Paolo Fiorini (uno tra gli ideatori del Path Finder per la Nasa).

Figura 3.1: Manifesto Discovery on Film 2011

L'obbiettivo di questa mostra è quello di avvicinare la gente, ed in particolare i ragazzi delle scuole, alla scienza e alla tecnologia. Per questo motivo anche il Museo Civico di Rovereto, come il Dipartimento di Ingegneria dell'Informazione dell'Università di Padova, partecipa al progetto didattico internazionale TEREcOP che come già spiegato in precedenza, sviluppa una struttura di supporto per corsi di formazione degli insegnanti, al fine di aiutarli a realizzare attività formative di tipo costruttivista con l'uso della robotica. Per fare questo ogni anno collaborano allo svolgimento della manifestazione collaborano varie aziende che operano nel campo della ricerca e della robotica, mettendo a disposizione del museo i risultati prodotti dal loro lavoro. All'interno della mostra infatti, sono predisposti degli stand dove dei rappresentanti delle aziende mettono in mostra al pubblico alcuni dei loro principali prodotti, utilizzando filmati e dimostrazioni vere e proprie per illustrare gli sviluppi tecnologici che sono stati prodotti dalle loro ricerche. Oltre alle aziende

anche varie università e scuole, contribuiscono alla realizzazione del *Discovery on film* festival. Tra le varie università partecipa al festival come collaboratrice anche l'Università degli Studi di Padova. Da ormai molto tempo infatti, l'università partecipa alla mostra portando il proprio contributo con esempi di robotica educativa con lo scopo di spiegare a docenti e studenti delle scuole, che importante contributo possa dare la robotica allo svolgimento delle attività didattiche. Nell'edizione 2011 l'Università di Padova ha partecipato alla mostra portando una serie di esempi di robot ispirati alla natura. In particolare sono stati presentati i robot che sono descritti in questa tesi, ovvero il verme, il robot rappresentante la visione stereoscopica e quello riguardante l'equilibrio. La presentazione di questi esempi è stata fatta all'interno di una struttura, messa a disposizione dal museo, adibita per favorire la presentazione dei robot, la *Discovery Arena*.

3.1.1 La Discovery Arena

La Discovery Arena è una struttura, realizzata in occasione del Discovery film festival, dove vengono effettuate diverse dimostrazioni nell'arco dei cinque giorni della mostra. L'arena è stata concepita sia con lo scopo di creare le condizioni ideali per permettere l'esibizione dei vari robot, sia con l'obbiettivo di favorire la visione alle circa cento persone che possono essere contenute nella struttura, in modo che queste possano cogliere anche il minimo particolare di tutte le esibizioni. Il campo gara infatti è costantemente monitorato da un paio di telecamere che proiettano le immagini su due schermi, consentendo così a chi assiste alle esibizioni, di cogliere anche il minimo movimento dei robot che sarebbe impossibile notare dall'esterno del campo gara. All'interno dell'arena si susseguono esibizioni di vario genere come ad esempio gare di *FIRST® LEGO® League*, e di *Robocup*. La FLL è un concorso internazionale promosso dalla LEGO con lo scopo di favorire creatività e spirito di gruppo dei ragazzi dai 10 ai 16 anni. I ragazzi vengono inseriti in una sana competizione che li vede impegnati nel costruire dei robot con i kit della LEGO, che verranno poi giudicati da una giuria. La Robocup invece è una competizione "calcistica" per robot creata con lo scopo di realizzare entro il 2050 una squadra di robot umanoidi in grado di sfidare la nazionale campione del mondo.

Oltre alle esibizioni appena citate, all'interno dell'arena sono stati presentati anche i robot descritti in questa tesi. In due diversi momenti del festival infatti, è stato possibile mostrare il funzionamento di tutti e tre i robot spiegandone, oltre al lato ingegneristico

riguardante la progettazione, anche tutte le possibili applicazioni didattiche nel quale possono essere utilizzati questi esempi.

Figura 3.2: La Robocup

Durante la presentazione degli esempi è stato anche possibile effettuare una discussione con il pubblico presente composto prevalentemente da ragazzi delle scuole. La presentazione infatti, era presieduta da una presentatrice che aveva il compito di far interagire il pubblico, con coloro che effettuavano le presentazioni. Dai dibattiti emersi durante la presentazione al pubblico, è stato possibile capire che l'utilizzo dei robot per l'educazione scolastica, sarebbe accettato in modo molto positivo sia da docenti che da studenti. Infatti da alcuni degli ragazzi che hanno partecipato al dibattito, è emerso sarebbe molto più interessante andare a studiare i fenomeni matematici e fisici che vengono insegnati nelle scuole, se questi si potessero rappresentare con dei robot da loro stessi costruiti.

3.2 Argomenti didattici collegati ai Robot

Uno degli obbiettivi che hanno portato alla creazione dei robot trattati in questa tesi è quello di aiutare la comprensione e lo studio di alcuni concetti ai ragazzi delle scuole. Andiamo ora ad analizzare quali sono gli argomenti trattabili utilizzando i tre robot che sono stati creati.

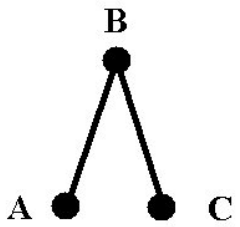


Figura 3.3: Schema della struttura del robot verme

Utilizzando il primo dei tre esempi, quello della camminata del verme, possono essere introdotti tre argomenti molto utili nel campo della matematica e della fisica. Infatti, se andiamo ad analizzare le posizioni assunte dal punto B di figura 3.3 , possiamo notare che queste disegnano un grafico che rappresenta un arco di coseno ripetuto.

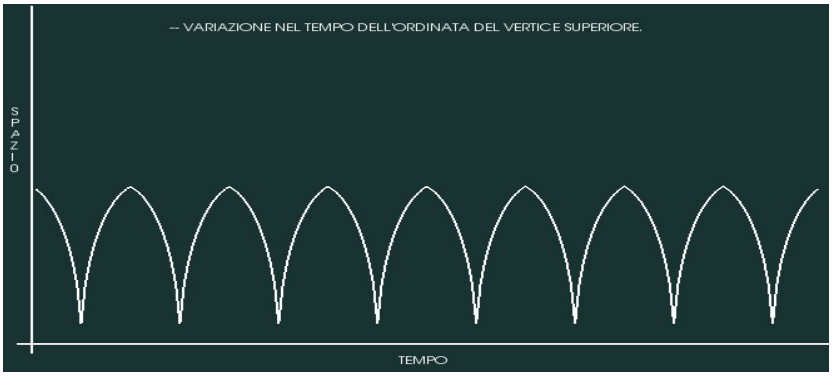
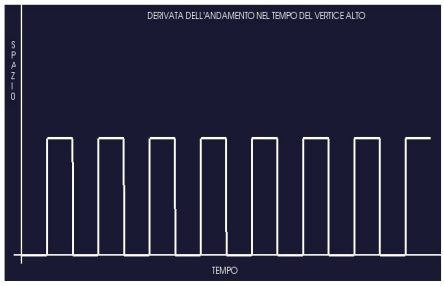
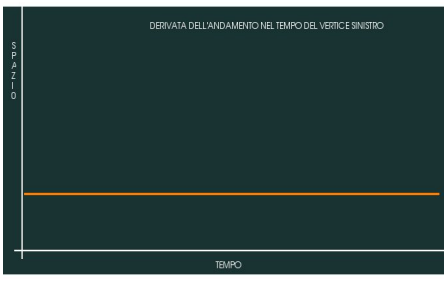


Figura 3.4: Grafico andamento del punto B

Oltre a questo si può anche introdurre il concetto di derivata studiando il moto del verme. Infatti si possono studiare le velocità dei diversi punti del verme, notando che la velocità del punto B è esattamente la metà rispetto a quella del punto A. Questo risultato lo si può vedere in modo evidente nei grafici di figura 3.5. Per realizzare questi grafici è stato necessario derivare nel tempo lo spostamento dei vari punti, ottenendo così la velocità degli stessi.Come si vede in figura 3.5 tale velocità risulterà costante per il punto B, mentre per le posizioni A e C sarà uguale a zero o al doppio di quella del punto B a seconda della movimento che il robot sta compiendo.



(a) *Grafico della velocità del punto A*



(b) *Grafico della velocità del punto B.*

Figura 3.5: Grafici delle velocità dei punti A e B.

Servendoci del secondo robot, quello della "visione stereoscopica", è possibile introdurre una serie di teoremi sui triangoli che rappre-

sentano la base per tutta la geometria. Quando si va a definire il raggio d'azione del visore stereoscopico infatti, vengono utilizzati dei teoremi fondamentali della geometria, come ad esempio il teorema di Pitagora, che servirà per definire precisamente in quale area gli oggetti saranno reperibili dai due sensori utilizzati nella costruzione del robot. Dalla figura 3.6 si può notare infatti che i campi visivi dei sensori disegnano delle aree a forma di triangolo.

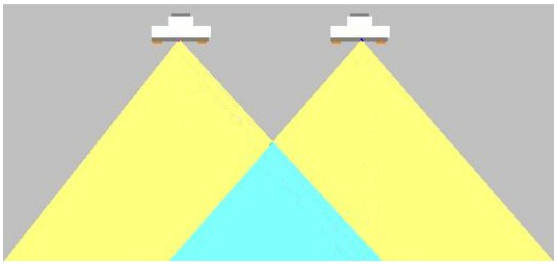


Figura 3.6: Campo visivo dei sensori

Per andare a calcolare le aree dei triangoli che vengono formati dai campi visivi dei sensori è necessario utilizzare i seguenti teoremi sui triangoli:

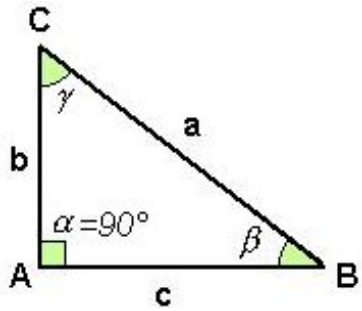


Figura 3.7: Triangolo rettangolo

Il teorema di Pitagora che dice: *"In un triangolo rettangolo, l'area del quadrato costruito sull'ipotenusa è equivalente alla somma delle aree dei quadrati costruiti sui due cateti."* che in formule diventa:

$$a = \sqrt{b^2 + c^2} \tag{3.1}$$

E il teorema sui triangoli rettangoli che dice : *"In un triangolo rettangolo, la misura di un cateto è uguale al prodotto della misura dell'ipotenusa per il seno dell'angolo opposto oppure per il coseno dell'angolo adiacente."* che in formule diventa:

$$b = a \cdot \sin\beta \qquad b = a \cdot \cos\gamma \qquad c = a \cdot \sin\gamma \qquad c = a \cdot \cos\beta \tag{3.2}$$

L'ultimo degli esempi trattati in questa tesi può essere utilizzato per trattare una serie di argomenti che molto spesso vengono affrontati nelle scuole. Per prima cosa si può introdurre l'argomento del *pendolo inverso* argomento importante quando si va a studiare la fisica. L'idea su cui si basa il robot che riproduce l'equilibrio infatti, è esattamente rappresentata dal moto del pendolo inverso. In seguito verrà trattato in modo più approfondito tale argomento. Oltre a quello appena descritto anche un altro argomento può essere affrontato andando ad utilizzare l'esempio sull'equilibrio, quello degli integrali. Per bilanciare il robot infatti viene utilizzato un controllo tipico dei sistemi retroazionati, il PI. Questo controllore viene utilizzato aggiungendo al segnale in uscita una parte proporzionale ed una integrale. Tramite lo sviluppo di questa seconda parte si possono introdurre gli integrale agli studenti in un modo molto più interessante rispetto a quello tradizionale.

Capitolo 4

Il verme

Figura 4.1: Il verme

4.1 In natura

Il robot che andremo ad analizzare in questo capitolo è stato concepito con lo scopo di riprodurre la "camminata" del verme. Più precisamente si andrà a riprodurre il moto del verme, infatti la riproduzione fedele della "camminata" sarebbe un lavoro molto complicato e assolutamente inutile visti i fini didattici di questa esperienza. La struttura corporea del verme in realtà, è molto più complessa di quando possa sembrare. Esso infatti è composto da circa centocinquanta anelli collegati tra loro che si muovono tutti in modo indipendente durante la camminata; pertanto per una riproduzione fedele di questo moto sarebbe necessaria una struttura meccanica molto complessa accompagnata da una programmazione molto più particolare e precisa rispetto a quella utilizzata nel robot che in questo capitolo si va a descrivere. Lo spostamento del verme è

caratterizzato dal continuo alternarsi tra la fase di allungo e quella di contrattura del corpo. Queste due fasi combinate tra loro rendono il movimento di "camminata" simile al movimento delle onde del mare. Di questo moto sono responsabili tutti gli anelli che compongono il corpo del verme: infatti la "camminata" ha inizio quando l'ultimo di questi anelli inizia a muoversi fornendo la spinta iniziale, generando una sorta di reazione a catena. È infatti dopo la prima spinta che il secondo anello inizia a muoversi verso l'alto mettendo a sua volta in moto il terzo anello. Questa sorta di reazione a catena avrà fine solamente quando tutti gli anelli avranno completato il loro movimento, portando così il verme nella configurazione "b" di figura 4.2 . Una volta terminata la fase di contrattura inizia il processo inverso ovvero quello di allungo che riporta il verme nella configurazione iniziale ovvero quella di "a" di figura 4.2. Ovviamente queste operazioni si susseguiranno fino a che il verme non avrà raggiunto la meta desiderata.

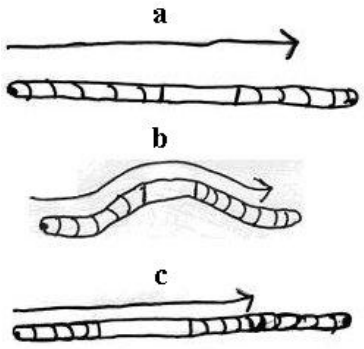


Figura 4.2: Il movimento del verme

Per rendere possibile questo tipo di movimento, ognuno dei centocinquanta anelli componenti la struttura del verme, è dotato di quattro paia di setole orientate di norma all'indietro. Queste setole hanno un ruolo fondamentale per il movimento del verme: infatti, tramite la loro struttura, riescono a far presa sul terreno permettendo così l'avanzamento al verme. Senza l'ausilio di queste setole la "camminata" diverrebbe difficile vista la natura viscosa del verme, che creerebbe una bassa forza di attrito tra il corpo ed il terreno, provocando uno scivolamento all'indietro invece che in avanti.

Figura 4.3: Le setole del verme

Figura 4.4: Posizione di partenza

4.2 La teoria

Andiamo ora ad analizzare la teoria alla base della camminata del verme analizzando tutti gli elementi che entrano in gioco durante l'esecuzione di questo moto. Come già riportato in precedenza la camminata del verme è caratterizzata da un continuo alternarsi tra la fase di allungo e quella di contrattura. La fase di allungo ha inizio quando il verme si trova nella configurazione di figura 4.4.

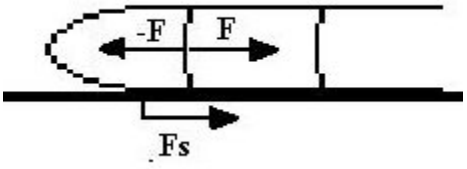


Figura 4.5: La risultante delle forze

A questo punto l'ultimo degli anelli che compone il verme, ovvero quello in "coda", inizia la fase di spinta. In questa fase viene generata una forza(F in figura 4.5) che viene impressa sul penultimo anello, spingendolo in direzione concorde a quella della "camminata". Come insegna il terzo principio della dinamica, " *Ad ogni azione corrisponde una reazione uguale e contraria.*" . Anche in questo caso la forza impressa dalla coda verso il penultimo anello genera una

reazione uguale e contraria, ovvero vi sarà una forza(-F in figura 4.5) impressa dal penultimo anello che spingerà la coda all'indietro. Se esistessero solamente le forze appena descritte, il verme rimarrebbe sempre fermo nella posizione iniziale. È invece grazie alla forza di attrito, ed in particolare la "forza di attrito statico ", che il verme riesce a muoversi sul terreno. L'attrito è una forza dissipativa che si esercita tra due superfici a contatto tra loro opponendosi al loro moto relativo. In questo particolare esempio la forza di attrito(f_s) si genera tra la superficie su cui cammina il verme ed il verme stesso. L'intensità di tale forza è data dalla formula 4.1:

$$f_s = \mu_s \cdot N \tag{4.1}$$

con f_s forza di attrito, μ_s coefficiente di attrito e N intensità normale della forza. La forza d'attrito che viene a generarsi è esattamente di uguale intensità rispetto a quella di reazione generata dalla spinta della coda, l'unica cosa che varia è il verso che risulta essere opposto ovvero:

$$|-F| = |-f_s| \tag{4.2}$$

Applichiamo ora la seconda legge di Newton per calcolare la forza risultante tra i due anelli:

$$\sum_{k=0}^{n} F_k = m \cdot a \tag{4.3}$$

nel nostro caso:

$$\sum_{k=0}^{n} F_k = F + (-F) + f_s \tag{4.4}$$

per la formula 4.2 la risultante diventa:

$$\sum_{k=0}^{n} F_k = F \tag{4.5}$$

ovvero la risultante sarà data dalla sola forza F in quanto le altre forze si annullano a vicenda. Essendo la sommatoria di tutte le forze applicate tra i due anelli non nulla, possiamo dedurre, dalla seconda legge di Newton, che il penultimo anello subirà un'accelerazione e quindi si allungherà in avanti. Questo fatto provocherà una reazione a catena nella quale ogni anello applicherà una spinta sul successivo, fino a che tutti gli anelli si saranno allungati portando il verme nella configurazione di figura 4.6.

Figura 4.6: posizione di allungo

Una volta finita la fase di allungo inizia quella di contrattura, ovvero la parte anteriore del verme richiama a se la parte posteriore. In questo caso l'anello in testa al verme eserciterà una forza di richiamo nei confronti del secondo anello, generando a sua volta una forza di reazione uguale e contraria che spinge la testa in avanti. Anche in questo caso entra in gioco la forza di attrito che si contrappone alla forza di reazione azzerandone l'effetto. Così facendo la risultante delle forze è ancora una volta la prima forza generata, ovvero quella di richiamo esercitata dall'anello di testa. In questo modo verrà generata una reazione a catena dove ogni anello del corpo tirerà verso di se quello immediatamente successivo riportando l'intero verme nella configurazione di figura 4.4. Le due fasi di allungo e contrattura si susseguiranno fino a che il verme non avrà raggiunto la destinazione desiderata.

4.3 Simulazione con LEGO ®MINDSTORMS® NXT

Figura 4.7: Il robot riprodotto con il software Lego Digital Design

La simulazione della camminata del verme viene effettuata utilizzando il robot di figura 4.7. Come si può vedere dalla figura

l'automa è dotato, in ognuna delle due estremità, di una coppia di ruote(punti A e C in figura 4.7) ciascuna delle quali è governata da un servomotore; inoltre è presente uno snodo centrale (punto B di figura 4.7) libero di muoversi in base al movimento effettuato dal robot.

Il meccanismo di funzionamento prevede che il robot inizialmente si trovi nella posizione di figura 4.7, potendo così dare inizio alla camminata partendo dalla fase di allungo. Questa fase ha inizio quando viene azionato il motore del punto C di figura 4.7, il quale farà muovere le ruote in direzione frontale facendo così allungare il robot. È molto importante che durante la fase di allungo il motore del punto A venga mantenuto frenato simulando così l'attrito che, come visto, gioca un ruolo fondamentale nel movimento peristaltico.

Si assuma lo snodo B ideale (quindi senza attrito) e i due bracci del robot rigidi e con massa nulla. A seguito di queste premesse si può affermare che mantenendo i motori in folle, il robot rimane in uno stato di quiete: infatti avendo trascurato la massa dei due bracci, le uniche forze che agiscono sul robot sono le due forze peso delle ruote che però vengono contrastate dalla forza di reazione esercitata dal suolo su cui sono appoggiate.

Quando viene azionato il motore C, viene generata una coppia motrice che fa variare istantaneamente la velocità angolare delle ruote da 0 a $\omega = \frac{V}{r}$. A questo punto si generano delle forze interne lungo i due bracci del robot come si vede in figura 4.8.

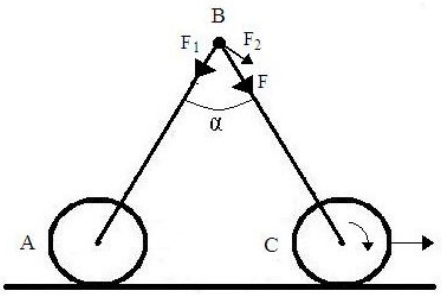


Figura 4.8: Le forze agenti sul robot con $\alpha < 90°$.

Come si vede in figura 4.8 l'inizio del rotolamento del punto C genera una forza F che si sviluppa lungo il braccio 2 del robot. Questa forza genera a sua volta due forze: "F_1" che si sviluppa lungo il braccio 1 e "F_2" con componente perpendicolare rispetto al braccio 1. L'intensità delle due forze è data da:

$$F_1 = F \cdot \cos\alpha \tag{4.6}$$

$$F_2 = F \cdot \sin\alpha \tag{4.7}$$

Fino a che l'angolo α assume valori inferiori a 90° le due forze F_1 e F_2 hanno direzioni e versi concordi con quelli di figura 4.8. Con questo verso la forza F_2 attrae verso destra il punto B generando a sua volta, una forza che si sviluppa lungo il braccio 1 e che fa muovere il punto A nella stessa direzione del punto B, impedendo così lo svolgimento della fase di allungo. Quando l'angolo è esattamente 90°, la forza F_1 si annulla e rimane la sola forza F_2 che assume lo stesso verso e la stessa direzione della forza F.

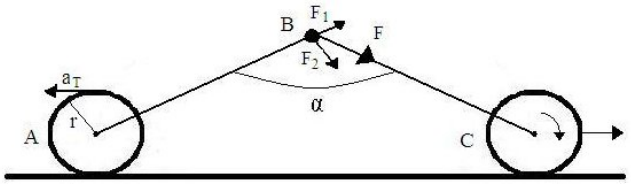


Figura 4.9: Le forze agenti sul robot con $\alpha > 90°$.

Per valori dell'angolo α maggiori di 90°, la forza F_1 cambia direzione e verso, provocando quindi un cambio di direzione della forza F_2 come si vede in figura 4.9. Per angoli maggiori di 90° infatti, la formula 4.6 assume valori negativi, a causa del cambio di segno della funzione $\cos\alpha$. Questo cambio di segno fa assumere alla forza F_2 la direzione riportata in figura 4.9. Con questa direzione e questo verso, la forza F_2 spinge il punto B verso il basso, generando a sua volta una forza che si sviluppa lungo il braccio 1 e che spinge il punto A verso sinistra, provocando quindi la caduta del robot.

Nelle considerazioni precedenti si è visto come il punto "A" sia sottoposto a delle forze che ne provocano lo spostamento. Dalla presenza di queste forze si deduce, in accordo con la seconda legge di Newton ($F = m \cdot a$), che le ruote del punto A sono sottoposte ad un'accelerazione lineare 'a' come si vede in figura 4.9. La componente tangenziale(a_T) di questa accelerazione, implica che le ruote del punto "A" subiscono un'accelerazione angolare(α) pari a:

$$\alpha = \frac{a_T}{r} \tag{4.8}$$

dove 'r' rappresenta il raggio della ruota. L'accelerazione angolare che si crea, fa girare le ruote del provocando così il movimento del punto "A". Mantenendo però frenato il motore posizionato in questo punto, l'accelerazione angolare viene contrastata dalla forza frenante esercitata dal motore, quindi le ruote rimangono ferme e la fase di allungo procede senza errori.

Una volta terminata la fase di allungo ha inizio la fase di contrattura. In questa fase il punto A del robot deve avvicinarsi al punto C, il quale invece non deve muoversi dalla posizione in cui si trova. Quando inizia la fase di contrattura il robot si trova nella configurazione di figura 4.10 b) e deve tornare alla configurazione di partenza. Per fare ciò viene messo in moto il motore del punto A mentre, per il motivo spiegato in precedenza,il motore C si mantiene in modalità frenata.

(a) *Robot in posizione di contrattura.* (b) *Robot in posizione di allungo.*

Figura 4.10: Robot nelle due posizioni principali.

Le azioni che vengono fatte fare al robot sono le stesse che sono state descritte in precedenza per la fase di allungo, l'unica differenza è che in questo caso sarà il motore del punto "A" a muoversi, mentre quello del punto "C" rimane fermo.

Questa rappresentazione del verme porta con se alcune limitazioni dovute alla tecnologia impiegata per costruirlo. A differenza dei vermi infatti, questo robot non può compiere una completa elongazione del corpo per motivi fisici. Va infatti considerato che il peso del robot è molto superiore rispetto al peso del verme, inoltre a differenza di quanto accade per i vermi, in questa riproduzione la spinta viene generata da due soli motori posti alle estremità del robot invece di essere fornita da ogni singola parte del verme. Queste differenze introducono una problematica di cui bisogna tenere conto in fase di programmazione.

L'elemento del robot che crea i maggiori problemi è senza dubbio il brick, ovvero la mente del nostro automa. Questo oggetto infatti è molto pesante ed ingombrante e la forza generata dal suo peso va ad incidere molto sui gradi massimi di inclinazione che si possono

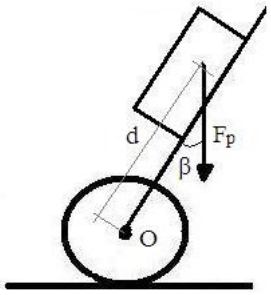


Figura 4.11: Forze generate dal peso del brick

far fare al robot in fase di allungo. In questa fase, il motore "C" è in movimento mentre il motore "A" in modalità frenata per impedire alle ruote del punto "A" di muoversi. Il brick NXT, come si vede in figura 4.11, genera una forza peso perpendicolare al suolo di intensità:

$$F_p = m \cdot g \tag{4.9}$$

con 'm' massa del brick e 'g' accelerazione gravitazionale. Questa forza peso spinge il braccio 1 verso il basso, generando un momento torcente nel punto "O" che provoca un'accelerazione angolare nelle ruote . Il momento torcente generato dalla forza peso dipende dall'angolo β infatti:

$$\tau = F_p \cdot d \cdot \sin\beta \tag{4.10}$$

dove 'd' è la distanza tra il centro 'O' e il centro di massa del brick e 'β' è l'angolo che si forma tra la forza peso e l'inclinazione del braccio. Se aumenta l'inclinazione del robot, aumenta l'angolo β quindi, il momento torcente calcolato nella formula 4.10 aumenta di intensità e di conseguenza viene incrementata anche l'accelerazione angolare. L'aumento di questa accelerazione comporta un maggiore sforzo, che il motore deve compiere, per mantenere ferme le ruote. Dopo una certa inclinazione, il momento torcente assume un valore tale da non consentire più al motore, posto nel punto "A" di mantenere frenate le ruote. A questo punto le ruote poste nel punto "A" accelerano verso sinistra mentre il punto "C" continua il suo movimento verso destra. Questo fatto provoca l'allungamento totale del robot con conseguente caduta dello stesso. In fase di programmazione quindi, sarà necessario regolare con attenzione l'angolo di apertura massimo da far compiere all'automa, in modo tale che la forza frenante dei motori possa compensare la forza generata sulle ruote, dal peso del brick.

4.4 Il programma

La simulazione della camminata del verme verrà realizzata facendo compiere al robot una fase di allungo seguita da una fase di contrattura. Questa sequenza di azioni verrà eseguita in un ciclo per cinque volte facendo compiere al robot un movimento in avanti. Una volta terminata questa sequenza di cinque cicli, verrà eseguita un'altra serie di cinque cicli per far tornare il robot in posizione iniziale. Questo movimento avanti ed indietro dell'automa verrà inserito in un ulteriore ciclo infinito facendo cosi continuare la simulazione fino a che un utente non ne interrompa la corsa tramite l'utilizzo dei tasti presenti nel brick NXT.

Figura 4.12: Codice NXT-G del robot

Analizzando il programma nel dettaglio si può vedere dalla figura 4.12 come questo risulti molto semplice e conciso. Il codice NXT-G prodotto infatti prevede il solo utilizzo di 4 blocchi azione dei motori e di 3 blocchi iterazione. La sequenza di azioni eseguite è molto semplice ma richiede che il robot sia posizionato in posizione di contrattura prima di far partire il programma, in caso contrario vi sarà un mal funzionamento che porterà la simulazione al fallimento. Una volta messo in posizione di contrattura, verrà azionato il motore A facendogli compiere un numero di gradi di rotazione in senso antiorario. Va precisato che in qualunque istante quando un motore è in funzione l'altro viene mantenuto frenato per i motivi descritti precedentemente. Una volta completato il movimento del motore inizierà il movimento del motore B il quale compierà lo stesso numero di gradi di rotazione realizzati dal motore A, ma li farà nel verso opposto vista la diversa collocazione dei motori nel robot. Il movimento alternato di questi due motori si trova all'interno di un ciclo che farà eseguire questa sequenza di azioni per cinque volte. Una volta terminato questo ciclo il robot si troverà ancora nella configurazione di partenza ma la posizione sul terreno sarà variata di una distanza pari a cinque volte la lunghezza del robot nella sua massima elongazione possibile.

A questo punto è necessario far tornare il robot nella posizione iniziale pertanto sarà necessario ripetere le stesse operazioni di prima solamente variando la direzione dei motori. In questo caso verrà per prima cosa fatto ruotare il motore C in verso antiorario per un numero di gradi pari a quelli fatti compiere in precedenza ai due motori. Una volta fermato il motore C viene azionato il motore A con le stesse condizioni ma questa volta con verso orario. Una volta eseguite le cinque iterazioni di questo ciclo il robot sarà riportato alla posizione iniziale. Il ciclo che racchiude tutti gli elementi del programma serve soltanto per far continuare il robot nella propria simulazione fino a che qualcuno non termini l'esecuzione del programma, andando ad agire sui tasti del brick NXT.

L'unica problematica emersa durante la programmazione di questo esempio, è stata la scelta relativa al numero di gradi da far compiere ai motori durante la simulazione. L'elemento da considerare per quanto riguarda questa scelta è rappresentato dalla superficie sulla quale si farà muovere il robot. È infatti importante sapere su quale materiale correranno le ruote perché in base a questo vi sarà una variazione della forza di attrito esercitata dal suole sulle ruote. Ovviamente più attrito verrà impresso dal terreno e maggiore potrà essere l'allungo del robot. Per uniformarsi alla maggior parte delle superfici possibili si è scelto, dopo una serie di prove pratiche, di far compiere ai motori un movimento di 600 gradi. Questa decisione è il giusto compromesso tra la necessità di far muovere il robot su superfici con basso coefficiente d'attrito e la volontà di simulare quanto più realmente possibile il movimento del verme, il quale come è stato visto si distende completamente durante la camminata.

Capitolo 5

La "visione stereoscopica"

In questo capitolo verrà studiata la realizzazione di un robot capace di "vedere" gli oggetti in movimento, rilevandone la direzione dello spostamento. Per creare questo esempio si è preso spunto dalla "vista" dei pipistrelli, anche se come vedremo di seguito, è improprio utilizzare il termine "vista" quando si parla di questi mammiferi.

5.1 In natura

Analizziamo ora il meccanismo che consente ai pipistrelli di orientarsi nello spazio. A differenza di quanto accade per gli esseri umani, i pipistrelli non sono dotati di occhi in grado di rilevare gli oggetti nello spazio. Questo tipo di mammifero infatti non vede gli oggetti attorno a se, ma ne rileva la presenza grazie all'utilizzo di segnali sonori. Per orientarsi nello spazio infatti, i pipistrelli inviano continuamente delle onde ultrasoniche, le quali vengono prodotte della laringe e vengono inviate nell'ambiente tramite il naso o, più comunemente, dalla bocca aperta. La frequenza dei suoni prodotti va da quattordicimila a più di centomila Hz, molto al di là delle capacità uditive dell'orecchio umano, che percepisce suoni con una frequenza che va da venti a ventimila Hz. Una volta inviate le onde sonore nell'ambiente il mammifero attende l'eco prodotta dalle onde una volta che queste urtano un oggetto. L'eco prodotta viene percepita ed analizzata dal sistema uditivo, il quale riesce a costruire una sorta di mappa dell'ambiente circostante. L'apparato uditivo dei pipistrelli è così sofisticato che riesce a distinguere l'eco prodotta da due oggetti diversi, consentendo a questi mammiferi di rilevare oggetti di natura diversa. Le complessità di questo meccanismo sono dovute al fatto che gli echi riflessi dalla vegetazione sono segnali stocastici molto complessi: dal punto di vista del segnale acustico una

pianta è una matrice tridimensionale di foglie che riflette il segnale emesso dal pipistrello.

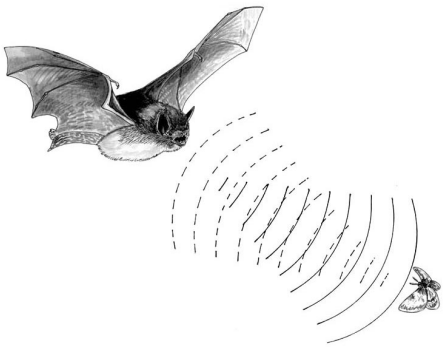


Figura 5.1: Forze generate dal peso del brick

Il segnale di ritorno è dunque una sovrapposizione di molti echi, per questo motivo è ancora difficile capire come possano essere interpretati correttamente dall'apparato uditivo dei pipistrelli, consentendo al mammifero di distinguere un frutto di un albero dalle sue foglie. Grazie a questo meccanismo di riconoscimento, detto anche "ecolocalizzazione", il pipistrello è in grado, durante i suoi voli apparentemente caotici e confusi, di eseguire una sorta di "mappatura" dell'ambiente circostante, rilevando gli ostacoli e le eventuali prede che si presentano lungo la sua strada.

5.2 La teoria

Lo scopo di questa sessione è quello di analizzare più nel dettaglio i fenomeni che stanno dietro alla "vista dei pipistrelli". Come è stato detto in precedenza i pipistrelli emettono degli ultrasuoni per orientarsi nello spazio.

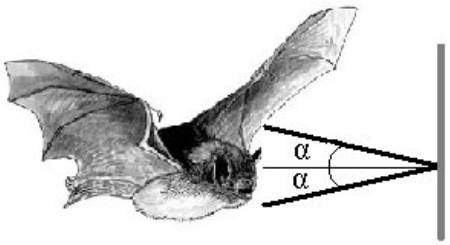


Figura 5.2: Riflessione degli ultrasuoni

Queste onde sonore vagano nell'aria fino a che non incontrano un corpo rigido che, invece di assorbirle, le riflette. Le onde riflesse ritornano verso la sorgente con una forma diversa e con un ritardo t . Se questo ritardo è superiore al decimo di secondo questo fenomeno fisico viene definito "l'eco " dell'onda. Conoscendo il tempo di ritardo t è possibile quindi calcolare la distanza di un oggetto dall'emettitore di onde ad ultrasuoni. Le onde sonore urtano gli oggetti con un angolo α come si vede in figura 5.2, conoscendo questo angolo ed il tempo che l'onda impiega per tornare all'emettitore è dunque possibile calcolare la distanza D:

$$D = \frac{V \cdot t \cdot \cos \alpha}{2} \tag{5.1}$$

Nel caso l'angolo di contatto tra l'onda e la superficie riflettente sia prossimo allo 0 la formula 5.1 diventa:

$$D = \frac{V \cdot t}{2} \tag{5.2}$$

Grazie a questi 'calcoli' è dunque possibile per i pipistrelli conoscere le distanze dagli oggetti. Come è stato visto in precedenza però questi mammiferi non riconoscono solamente le distanze ma riescono anche a valutare le differenze tra gli oggetti semplicemente studiandone le onde riflesse. Questo meccanismo si basa essenzialmente sull'analisi delle variazioni delle caratteristiche delle onde riflesse, è infatti notando le differenze con il segnale trasmesso che si possono dedurre le caratteristiche dell'oggetto urtato.

Per analizzare le differenze tra le diverse onde sonore è necessario introdurre alcuni concetti fondamentali nel campo dell'acustica. Il primo di questi è il concetto di velocità di propagazione dell'onda sonora, la quale viene misurata in metri al secondo($\frac{m}{s}$) ed è definita come lo spazio percorso dal fronte d'onda nell'unità di tempo. Essa dipende fortemente dalle caratteristiche del mezzo nel quale viene trasmesso il segnale e può variare considerevolmente a seconda dei diversi elementi su cui si propaga. Infatti la velocità di propagazione del segnale sonoro nell'aria è di circa 300 $\frac{m}{s}$ contro i 1500 $\frac{m}{s}$ raggiunti nei tessuti umani e gli addirittura 4000 $\frac{m}{s}$ ottenuti propagandosi nelle ossa. La velocità di propagazione di un'onda dipende essenzialmente da due caratteristiche della stessa, la frequenza (F) e la lunghezza d'onda (λ). È grazie a queste due informazioni infatti che si può ricavare la velocità di un segnale considerato che la velocità è data da:

$$V = \lambda \cdot F \tag{5.3}$$

La velocità di propagazione è il fattore che influenza un altro concetto fondamentale nel campo dell'acustica, quello di impedenza. L'impedenza acustica viene definita come la resistenza del mezzo al passaggio degli ultrasuoni. Questa caratteristica, comunemente contraddistinta dal simbolo Z, è definita come il prodotto tra la densità del mezzo trasmissivo ρ e la velocità di propagazione dello stesso V in formule:

$$Z = \rho \cdot V \tag{5.4}$$

Una volta introdotti questi concetti è possibile studiare il meccanismo con il quale vengono analizzati i diversi echi prodotti dagli urti degli ultrasuoni emessi dai pipistrelli. Quando un'onda sonora colpisce un oggetto, la superficie di contatto con l'oggetto si può considerare come la linea di confine tra due zone con impedenza acustica Z. Quando un'onda urta un oggetto vengono generati due fenomeni, uno detto "riflessione" e l'altro "rifrazione". Il primo di questi dice che quando un ultrasuono colpisce la superficie di contatto tra due zone con impedenza differente, il segnale inviato viene riflesso in misura proporzionale alla diversa impedenza tra le i due mezzi. Il segnale riflesso è dotato di energia "incidente" (I) e l'intensità di questo segnale è data dal prodotto tra la sua energia ed il coefficiente di riflessione che in formule diventa:

$$E_i = I \cdot r \tag{5.5}$$

dove il coefficiente di riflessione è dato dalla formula:

$$r = (\frac{Z_1 - Z_2}{Z_1 + Z_2})^2 \tag{5.6}$$

con Z_1 e Z_2 impedenze delle due zone. In figura 5.3 si vede come l'onda venga riflessa dalla superficie di contatto con un angolo di uscita(α_i) uguale a quello di ingresso. Se l'ultrasuono colpisce una superficie con un angolo α_i troppo elevato, l'intero segnale verrà riflesso e non vi sarà rifrazione.

Il secondo fenomeno generato dall'urto degli ultrasuoni sugli oggetti è rappresentato dalla rifrazione. Come abbiamo appena visto E_i rappresenta l'energia del segnale riflesso dalla superficie di contatto, ma non tutta l'energia di cui era dotato l'ultrasuono di partenza. La restante energia, a meno di una piccola parte che viene ceduta nell'urto, diventa l'energia del segnale di rifrazione ovvero:

$$E_t = E_s - E_i \tag{5.7}$$

con E_s energia del segnale sonoro iniziale.

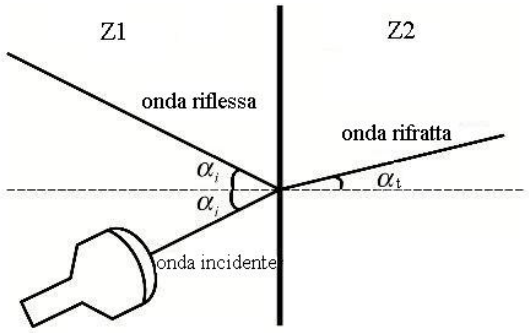


Figura 5.3: Riflessione e rifrazione degli ultrasuoni

L'energia del segnale trasmesso nel secondo mezzo è data dal prodotto tra l'energia incidente ed il coefficiente di trasmissione t ovvero:

$$E_t = I \cdot t \tag{5.8}$$

dove il coefficiente di trasmissione è dato dalla formula:

$$t = \frac{4 \cdot Z_1 \cdot Z_2}{(Z_1 + Z_2)^2} \tag{5.9}$$

L'angolo di trasmissione questa volta non è uguale a quello con cui l'ultrasuono a urtato la superficie, va calcolato utilizzando il Teorema di Snell che dice:

$$\frac{\sin \alpha_i}{V_1} = \frac{\sin \alpha_t}{V_2} \tag{5.10}$$

dove V_1 e V_2 rappresentano rispettivamente le velocità di propagazione delle due zone Z1, Z2 di figura 5.3.

Per riconoscere quindi i diversi oggetti tra loro i pipistrelli analizzano le differenza tra gli ultrasuoni trasmessi e quelli ricevuti come feedback. Un'onda, come abbiamo visto, viene riflessa con intensità diversa da oggetto ad oggetto a causa della diversa impedenza degli stessi. È secondo questo criterio che il pipistrello diversifica gli oggetti, anche se il meccanismo utilizzato è ancora oggi difficile da capire.

5.3 Simulazione con LEGO ®MINDSTORMS® NXT

(a) *Il robot principale.* (b) *Il robot "oggetto'.*

Figura 5.4: I due robot della simulazione.

Per la simulazione della "visione stereoscopica" verranno utilizzati due robot, uno che fungerà da visore(figura 5.4 a) mentre l'altro(figura 5.4 b) avrà il solo compito di muoversi avanti ed indietro nel campo visivo del primo robot in modo tale che questo possa percepirne il cambiamento di posizione. Il robot che rappresenta l'oggetto in movimento è un semplicissimo carro con quattro ruote sul quale è stata fissata una lastra di alluminio posta verticalmente su uno dei due lati. La lastra di alluminio è stata utilizzata per favorire il riflesso degli ultrasuoni inviati dai due sensori usati in questa rappresentazione. Se non avessimo usato questa lastra, le misurazioni sarebbero state falsate dalla geometria irregolare del carro con le ruote. Il carro infatti non è una forma lineare come può essere una parete, pertanto un'onda potrebbe sbattere contro le ruote, ovvero nella parte più vicina agli "occhi", mentre l'onda dell'altro sensore potrebbe scontrarsi con il brick nxt che rappresenta invece la parte più interna, facendo credere così al robot numero uno che l'oggetto si trovi in una posizione più laterale di quanto in realtà sia. La lastra di alluminio consente inoltre di evitare problemi relativa all'angolo d'urto delle onde, infatti per la natura dei sensori l'angolo con cui gli ultrasuoni vanno a colpire gli oggetti deve essere abbastanza ridotto. Se così non fosse una volta urtato l'oggetto, l'onda verrebbe riflessa con un angolo tale da non riuscire a raggiungere il ricevitore del sensore ad ultrasuoni e verrebbe quindi dispersa nell'ambiente.

L'elemento principale di questo esempio è ovviamente il robot costruito per rilevare gli spostamenti degli oggetti, ovvero il visore.

Questo automa ha una struttura molto semplice, infatti è composto soltanto da tre parti fondamentali: il brick NXT e i due sensori ad ultrasuoni posti ad una distanza A di 30 cm uno dall'altro. Questa distanza è stata calibrata in modo tale che l'area C(in figura 5.5) in comune tra i due sensori sia abbastanza grande per avere una sufficiente visione stereoscopica, ma allo stesso tempo non sia troppo grande da produrre un'area di visibilità comune ai due sensori troppo piccola.

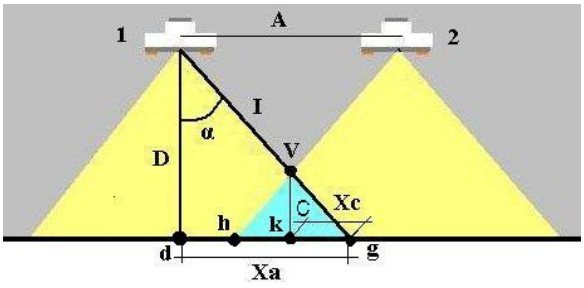


Figura 5.5: campo visivo del robot

L'attività principale svolta durante la costruzione di questo automa è stata sostanzialmente quella di trovare la giusta dimensione dell'area in comune tra i due campi visivi dei sensori. Gli elementi da considerare nella definizione di questa zona comune sono la distanza tra i sensori A, l'angolo di visione massimo dei sensori α e le distanza massima D_{max} e minima D_{min} nella quale gli oggetti vengono rilevati da entrambi i sensori con buona affidabilità.

Non potendo modificare i due parametri α e A in quanto dipendono dalla struttura del robot e dei sensori, il calcolo della dimensione dell'area comune si riduce alla sola determinazione delle due distanze D_{min} e D_{max}. Per determinare questi parametri, è necessario calcolare i due valori delle distanze X_a e X_c (di figura 5.5) le quali, assumendo un angolo α di 30°, sono date da:

$$X_a = D \cdot \tan \alpha \tag{5.11}$$

$$X_c = X_a - \frac{A}{2} \tag{5.12}$$

Essendo $\tan 30 = \frac{\sqrt{3}}{3}$ la 5.12 diventa:

$$X_c = \frac{D \cdot \sqrt{3}}{3} - \frac{A}{2} \tag{5.13}$$

La calibrazione della distanza minima, viene effettuata in modo da garantire al robot, una buona "visione stereoscopica"; questo

significa che la larghezza ($2X_c$) dell'area comune, non dovrà essere inferiore ad un certo valore. In questo caso si è scelto:

$$X_c \geq \frac{A}{6} \tag{5.14}$$

Sostituendo quindi la 5.13 nella 5.14 si ottiene:

$$\frac{D \cdot \sqrt{3}}{3} - \frac{A}{2} \geq \frac{A}{6} \tag{5.15}$$

e quindi:

$$D \geq \frac{2 \cdot A}{\sqrt{3}} \tag{5.16}$$

Essendo la distanza A tra i due sensori pari a 30 cm, è possibile calcolare la distanza minima, sostituendo il valore di A nella formula 5.16 ottenendo quindi:

$$D \geq 34,64cm \tag{5.17}$$

Va considerato che i sensori NXT non lavorano con valori decimali per questo motivo il valore appena calcolato va arrotondato al valore intero successivo ovvero 35 cm che sarà dunque il nostro valore D_{min}.

Una volta calcolato il limite minimo è necessario definire un valore massimo oltre il quale non è possibile andare. La scelta di porre un limite massimo alla distanza D è dovuta alle caratteristiche dei sensori utilizzati. I sensori ad ultrasuoni NXT infatti, nonostante riescano a misure distanze fino a 255 cm, non garantiscono una grande affidabilità quando si va oltre una certa soglia. Questo scarsa affidabilità è dovuta a due motivi: il primo è che con l'aumentare della distanza aumenta anche il disturbo che l'ambiente esterno introduce sul segnale ad ultrasuoni, il secondo è dovuto alla mancata ricezione da parte del ricevitore, del segnale riflesso che si crea quando l'onda ad ultrasuoni inviata urta un oggetto. La probabilità che si verifichi il problema della mancata ricezione del segnale riflesso, aumenta con l'aumentare della distanza D.

Per calcolare il valore D_{max} quindi sarà necessario fare delle prove per vedere fino a che distanza gli errori dovuti alla perdita del segnale trasmesso si presentano con una frequenza accettabile. In fase di programmazione poi sarà necessario filtrare gli eventuali errori di misura che di tanto in tanto possono presentarsi per i motivi appena citati. Dopo una serie di prove è stato possibile verificare che oltre una distanza D di 60 cm, il numero di misurazioni errate assume

frequenze importanti, pertanto questo sarà il valore D_{max} oltre al quale non si andrà durante questa esperienza.

Una volta definiti tutti i parametri relativi alle dimensioni del campo visivo del robot, si può procedere con la fase relativa al calcolo della posizione degli oggetti.

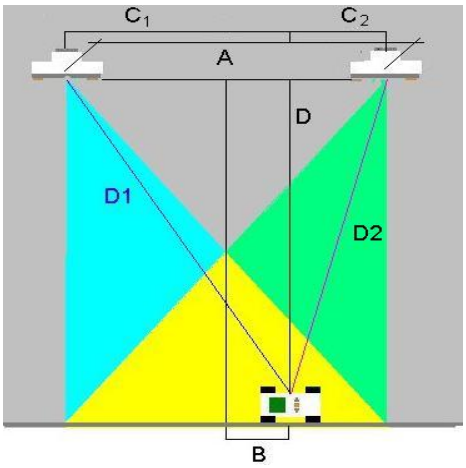


Figura 5.6: Rilevazione degli oggetti.

Osservando la figura 5.6 si può notare che questa operazione equivale al ricavare il valore del segmento B utilizzando i dati a disposizione. Gli unici valori di cui si è a conoscenza sono la distanza A e le due misure $D1$ e $D2$ fornite dai sensori. Sarà dunque necessario ricavare una formula per il calcolo automatico della posizione che utilizzi le variabili appena citate. Per prima cosa si procede con il calcolo dei due valori delle distanze C_1 e C_2:

$$C_1 = \sqrt{D1^2 - D^2} = \frac{A}{2} + B \tag{5.18}$$

$$C_2 = \sqrt{D2^2 - D^2} = \frac{A}{2} - B \tag{5.19}$$

da queste due espressioni si possono ricavare le due distanze $D1$ e $D2$ che saranno quindi:

$$D1^2 = (\frac{A}{2} + B)^2 + D^2 \tag{5.20}$$

$$D2^2 = (\frac{A}{2} - B)^2 + D^2 \tag{5.21}$$

Sottraendo la 5.21 alla 5.20 e si ottiene:

$$D1^2 - D2^2 = (\frac{A}{2} + B)^2 - (\frac{A}{2} - B)^2 \tag{5.22}$$

ricavando la distanza B la 5.22 diventa:

$$B = \frac{D1^2 - D2^2}{2 \cdot A} \tag{5.23}$$

il calcolo della distanza B dunque, dipende dalle sole variabili A,$D1$ e $D2$. Per confermare la veridicità di questa affermazione si considera la situazione in cui l'oggetto i trova esattamente davanti al sensore 2. In questo caso il valore di C è 0, mentre B diventa $\frac{A}{2}$, e dunque:

$$D2^2 = D1^2 - A^2 \tag{5.24}$$

utilizzando la formula 5.23 per il calcolo della posizione e sostituendo la distanza $D2$ appena calcolata si ottiene:

$$B = \frac{D1^2 - D2^2}{2 \cdot A} = \frac{D1^2 - D1^2 + A^2}{2 \cdot A} = \frac{A}{2} \tag{5.25}$$

che è esattamente il valore che ci si aspettava.

È interessante osservare come si comporta la differenza (ΔD) tra le due distanze misurate quando l'oggetto si trova in posizione molto laterale rispetto ai due sensori, ovvero quando B assume valori elevati. Partendo dal calcolo delle due distanze:

$$D1 = \sqrt{(\frac{A}{2} + B)^2 + D^2} \tag{5.26}$$

$$D2 = \sqrt{(\frac{A}{2} - B)^2 + D^2} \tag{5.27}$$

si può procedere facendo la differenza tra le due:

$$\Delta D = D1 - D2 = \sqrt{(\frac{A}{2} + B)^2 + D^2} - \sqrt{(\frac{A}{2} - B)^2 + D^2} \tag{5.28}$$

e calcolando quindi il limite per B si ottiene:

$$\lim_{B \to \infty} \Delta D = \lim_{B \to \infty} |(\frac{A}{2} + B)| - |(\frac{A}{2} - B)| \tag{5.29}$$

$$\frac{A}{2} + B - (B - \frac{A}{2}) = A \tag{5.30}$$

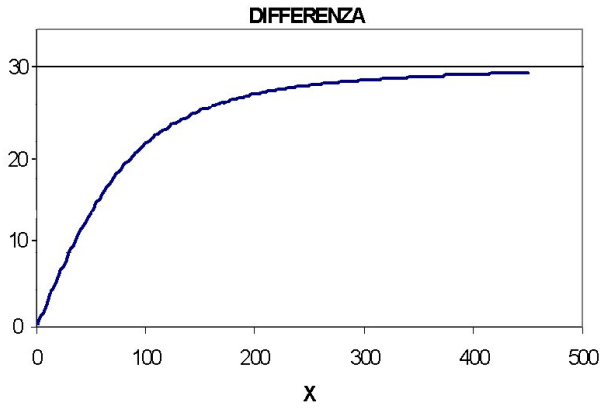


Figura 5.7: campo visivo del robot

In figura 5.7 è riportato il grafico relativo alla differenza tra le due distanze $D1$ e $D2$. Si può notare come per valori di X grandi la funzione tenda al valore 30 ovvero A.

Una volta terminata la parte riguardante la definizione del campo visivo e trovata la funzione per ricavare la posizione degli oggetti nello spazio, è necessario fissare le modalità con cui rappresentare le diverse posizioni degli oggetti riconosciute dal robot. In questo esempio si è deciso di visualizzare nel display del brick NXT le posizioni rilevate dal robot, con una modalità di visualizzazione molto semplice ed intuitiva. Prima di procedere con la visualizzazione però, è stato necessario limitare un po' la "definizione" del robot. Questa scelta è stata necessaria dopo aver verificato che riproducendo fedelmente ogni diversa posizione rilevata dal robot, si viene a creare una situazione di instabilità dovuta a due fattori che sono la sensibilità e la precisione dei sensori NXT. Questi sensori infatti hanno una sensibilità di misurazione di un centimetro ed una precisione non elevatissima, pertanto è molto facile che vengano fatti errori di qualche centimetro durante le misurazioni, ad esempio se un oggetto si trova ad una distanza di 35,1 cm dal sensore e vengono fatte due misurazioni successive, è lecito aspettarsi che la misura fornita possa essere prima 35 e poi 34 cm.

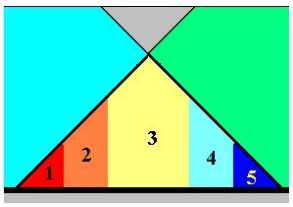


Figura 5.8: Divisione del campo visivo

Per questo motivo si è deciso di rendere meno sensibile il robot dividendo il campo visivo in cinque fasce come rappresentato in figura 5.8 e definite in questo modo:

- MOLTO A SINISTRA (MSX):l'oggetto si trova nella ZONA 1

- SINISTRA (SX): l'oggetto si trova nella ZONA 2

- CENTRO (CE): l'oggetto si trova nella ZONA 3

- DESTRA (DX): l'oggetto si trova nella ZONA 4

- MOLTO A DESTRA (MDX): l'oggetto si trova nella ZONA 5

Se si divide il campo visivo nel modo appena descritto, diventa necessario dividere in cinque parti anche l'insieme dei valori restituiti dalla funzione della formula 5.23. Per fare ciò bisogna andare a vedere quale range di valori viene restituito dalla funzione quando si lavora alle distanze D_{max} e D_{min}.

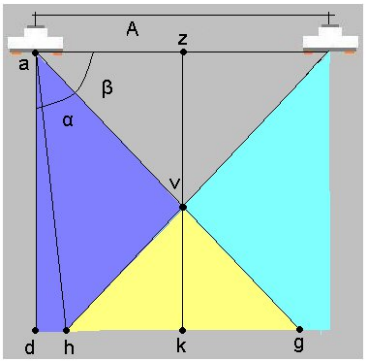


Figura 5.9: Divisione del campo visivo

Il range di valori sarà dato dalla differenza tra le distanze(arrotondate al valore intero precedente) misurate da uno dei due sensori nei due punti estremi del campo visivo (g o h in figura 5.9). Questa operazione coincide con il calcolo della differenza tra le due distanze $\bar{ag}$ e $\bar{ah}$. Queste distanze dipendo dai valori D e A infatti:

$$\bar{av} = \frac{\bar{az}}{\cos\beta} \tag{5.31}$$

essendo:

$$\beta = 90° - \alpha = 60° \tag{5.32}$$

la 5.31 diventa:

$$\bar{av} = \frac{\frac{A}{2}}{\frac{1}{2}} = A \tag{5.33}$$

quindi:

$$\bar{zv} = \bar{av} \cdot \sin\beta = A \cdot \frac{\sqrt{3}}{2} \tag{5.34}$$

$$\bar{vk} = D - \bar{zv} = D - (A \cdot \frac{\sqrt{3}}{2}) \tag{5.35}$$

$$\bar{hk} = \bar{vk} \cdot \tan\alpha = (D - (A \cdot \frac{\sqrt{3}}{2})) \cdot \frac{\sqrt{3}}{3} = (\frac{\sqrt{3}}{3} \cdot D) - \frac{A}{2} \tag{5.36}$$

$$\bar{dh} = \bar{dk} - \bar{hk} = \frac{A}{2} - (\frac{\sqrt{3}}{3} \cdot D) + \frac{A}{2} = A - (\frac{\sqrt{3}}{3} \cdot D) \tag{5.37}$$

Essendo $\bar{hk} = \bar{kg}$ si può definire $\bar{dg}$ come :

$$\bar{dg} = \bar{dk} + \bar{hk} = \frac{A}{2} + (\frac{\sqrt{3}}{3} \cdot D) - \frac{A}{2} = \frac{\sqrt{3}}{3} \cdot D \tag{5.38}$$

Le due distanze $\bar{ah}$ e $\bar{ag}$ saranno quindi:

$$\bar{ag} = \sqrt{D^2 + \bar{dg}^2} = \sqrt{D^2 + (\frac{\sqrt{3}}{3} \cdot D)^2} = \sqrt{\frac{4 \cdot D^2}{3}} = \frac{2 \cdot D}{\sqrt{3}} \tag{5.39}$$

$$\bar{ah} = \sqrt{D^2 + \bar{dh}^2} = \sqrt{D^2 + (A - (\frac{\sqrt{3}}{3} \cdot D))^2} \tag{5.40}$$

A questo punto sostituendo alla variabile D delle formule 5.39 e 5.40 rispettivamente i valori D_{max} e D_{min}, si ottengono le due distanze misurate dai sensori nei due estremi h e g:

$$\bar{ag} = \frac{2 \cdot 35}{\sqrt{3}} = 40,42cm \rightarrow 40cm \tag{5.41}$$

$$\bar{ah} = \sqrt{35^2 + (30 - (\frac{\sqrt{3}}{3} \cdot 35))^2} = 36,34cm \rightarrow 36 \tag{5.42}$$

e la loro differenza è:

$$40 - 36 = 4 \tag{5.43}$$

quindi la funzione restituisce 4 valori per ognuna delle due zone (destra e sinistra) più il valore 0 che indica la posizione centrale, pertanto alla distanza minima l'insieme dei valori restituiti dalla funzione che calcola la posizione ha cardinalità:

$$(2 \cdot 4) + 1 = 9 \tag{5.44}$$

Quando si lavora alla distanza massima invece le due distanze $\bar{ag}$ e $\bar{ah}$ diventano:

$$\bar{ag} = \frac{2 \cdot 60}{\sqrt{3}} = 69,28cm \rightarrow 69cm \tag{5.45}$$

$$\bar{ah} = \sqrt{60^2 + (30 - (\frac{\sqrt{3}}{3} \cdot 60))^2} = 60,18cm \rightarrow 60 \tag{5.46}$$

e la loro differenza è:

$$69 - 60 = 9 \tag{5.47}$$

in questo caso invece la funzione restituisce 9 valori per ognuna delle due zone pertanto l'insieme dei valori restituiti dalla funzione ha cardinalità:

$$(2 \cdot 9) + 1 = 19 \tag{5.48}$$

Una volta calcolati i due range, si può procedere con la divisione dei valori in modo da formare le cinque aree definite in precedenza. Osservando i dati appena calcolati, si è deciso di procedere alla seguente divisione: un oggetto viene considerato in zona MSX se il risultato ritornato dalla funzione 5.27 è minore di -5, mentre per valori compresi nell'intervallo [-5, -3] ci si trova nella zona SX; un oggetto si trova nella zona centrale CE se l'equazione ritorna valori in modulo minori di 3, mentre appartengono alla zona DX i valori compresi nell'intervallo [3, 5] ed infine verranno considerati in posizione MDX tutti i casi in cui l'equazione ritorna valori maggiori di 5.

Le zone appena descritte vengono rappresentate nel display del brick NXT come cinque cerchi posti su una stessa linea dello schermo, mentre la posizione è rappresentata da una X che si sposta da un cerchio all'altro a seconda della posizione assunta dagli oggetti durante la simulazione. Questo tipo di visualizzazione dell'output è

ottimo se il robot viene utilizzato da un numero ristretto di utenti, se invece viene esibito ad una pubblico più numeroso, come è accaduto in occasione del Discovery on Film Festival, allora è necessario cambiare le modalità di visualizzazione. Per rendere percepibili ad un pubblico più vasto le diverse posizioni rilevate dal robot si è pensato di utilizzare il riproduttore di suoni presente nel brick NXT. Per ognuna delle cinque posizioni infatti, è stata associata una tonalità di suono facilmente distinguibile, in questo modo anche solo ascoltando il robot, è facile percepire i cambiamenti di posizione degli oggetti. Le modalità di scelta delle diverse tonalità di suono verranno descritte nella sessione successiva.

5.4 Il programma

La simulazione della visione stereoscopica verrà effettuata muovendo il robot con le ruote avanti ed indietro nel campo visivo del robot "visore", il quale rileverà la posizione assunta dall'oggetto e la visualizzerà nel display del brick NXT. Durante tutta la simulazione il robot "visore" emetterà un suono la cui tonalità varierà a seconda della posizione assunta dal carro nel campo visivo, rendendo ancora più facile percepire gli spostamenti.

La simulazione prevede che il robot con le ruote sia posto centralmente rispetto alla posizione del robot "visore" e con le ruote parallele allo stesso come si vede in figura 5.10.

Figura 5.10: Posizione dei due robot durante la simulazione.

Inoltre per poter funzionare correttamente è necessario che il robot che produce il movimento sia posto ad una distanza maggiore di 30 cm e minore di 60 cm dal robot "visore".

La programmazione del robot ha prodotto un codice molto semplice come si può vedere in figura 5.11. Il programma infatti prevede che il robot muova i motori A e C in avanti per un totale di 500

gradi, provocando così lo spostamento verso destra del robot. Una volta terminato questo movimento verrà invertito il verso dei motori A e C , i quali verranno fatti muovere per 1000 gradi portando il robot ad attraversare tutto il campo visivo del robot "visore".

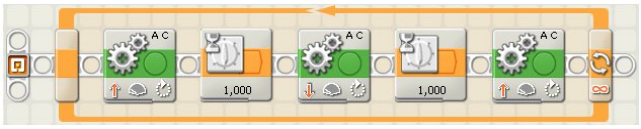


Figura 5.11: Codice che fa muovere il robot oggetto.

Una volta terminato anche questo movimento verrà di nuovo invertito il verso dei motori per far percorrere al robot i restanti 500 gradi che lo separano dal ritorno alla posizione iniziale. Tutte queste operazioni verranno inserite in un ciclo infinito, pertanto il movimento verrà ripetuto fino a che un utente non fermerà l'esecuzione del programma utilizzando i tasti presenti nel brick del robot.

Per quanto riguarda la programmazione del robot "visore" invece il codice NXT-G prodotto è un po' più complesso. Il programma infatti dovrà svolgere diverse operazioni per permettere la corretta esecuzione della simulazione. Per prima cosa il programma provvederà alla lettura delle misure dei due sensori ad ultrasuoni, filtrando qualora sia necessario, le misurazioni anomale.

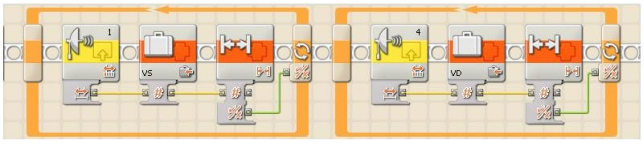


Figura 5.12: Codice relativo alla lettura dei sensori.

In figura 5.12 è riportata la parte di programma che si occupa delle operazioni di acquisizione e filtraggio dei dati. Si può notare dall'immagine come i due blocchi che leggono i valori dei sensori, siano messi all'interno di un ciclo. Questo ciclo è utilizzato per filtrare le misure errate e per impedire,

che la distanza tra i robot sia inferiore ai 30 cm; infatti la condizione per uscire da entrambi i cicli è che la misurazione fatta dai due sensori non ritorni valori minori di 30 e maggiori di 100. Il limite massimo è stato posto per eliminare le misurazioni errate dovute alla perdita dell'onda riflessa: infatti in questo caso il sensore ritornerà il valore 255. Nel caso la misurazione non rientrasse nei parametri stabiliti l'operazione verrà ripetuta fino a che il valore di ritorno del sensore non sia idoneo a soddisfare la condizione di uscita del ciclo.

Una volta letti e salvati nelle due variabili VD e VS i valori letti dei due sensori (destro e sinistro), è possibile procedere con i calcoli necessari alla determinazione della posizione.

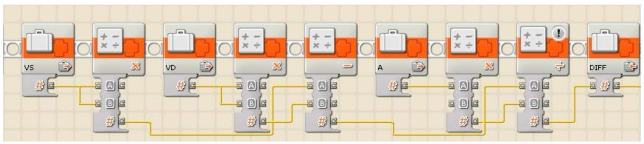


Figura 5.13: Codice per il calcolo della posizione degli oggetti.

In figura 5.13 è riportato il pezzo di codice che effettua i calcoli per determinare la posizione dell'oggetto. La formula utilizzata per determinare la posizione è la stessa del capitolo precedente ovvero:

$$B = \frac{D1^2 - D2^2}{2 \cdot A} \tag{5.49}$$

dove nel codice $D1$ è rappresentato dall variabile VS , $D2$ dalla variabile VD e B dalla variabile $DIFF$.

Una volta effettuato il calcolo del valore B viene effettuato un altro aggiustamento per consentire al robot di visualizzare in modo corretto le posizioni degli oggetti.

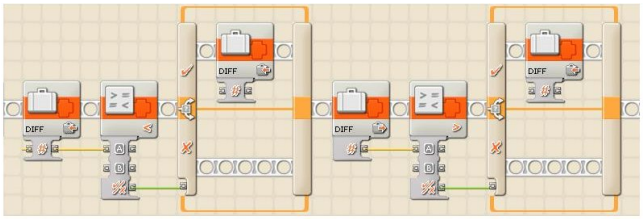


Figura 5.14: Codice che filtra i valori errati della variabile DIFF.

Il codice di figura 5.14 provvede ad assegnare alla variabile $DIFF$ il valore -8 se il valore della variabile stessa è inferiore a -8, mentre assegna il valore otto se invece il valore della stessa variabile assume valori maggiori di +8. Le motivazioni di questa operazione verranno spiegate in seguito quando ci si occuperà della fase di visualizzazione.

Il codice riportato in figura 5.15 serve per evitare che l'imprecisione dei sensori produca degli sbalzi anomali nel calcolo del valore $DIFF$.

Questa parte di programma infatti non consente al robot di effettuare bruschi cambiamenti di posizione: infatti, se il valore $DIFF$ dovesse differire di due posizioni rispetto al calcolo precedente, verrebbe sostituito con il valore precedente. Questo vincolo è utile

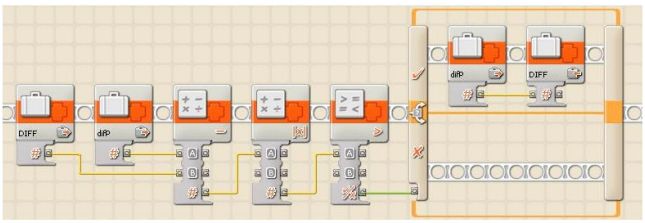


Figura 5.15: Codice per eliminare le oscillazioni anomale.

solamente per il tipo di simulazione che si andrà ad effettuare: infatti lo spostamento del robot con le ruote è di tipo lineare, quindi se vi fosse una grande differenza tra due valori $DIFF$ calcolati in successione, saremmo in presenza di un errore dovuto alle misurazioni dei sensori.

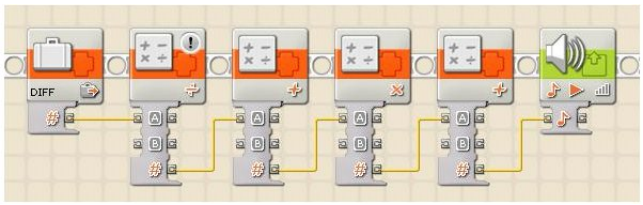


Figura 5.16: Codice che produce il segnale audio.

Prima di passare alla parte relativa alla visualizzazione, il programma si occupa della parte relativa alla riproduzione dei suoni. Infatti il robot dovrà assegnare alle 5 aree di divisione del campo visivo cinque suoni facilmente distinguibili. Dopo una serie di prove si è scelto di mantenere una distanza tra le varie tonalità di 140 Hz, assegnando come valore alla zona più a sinistra (MSX) il valore 400Hz. Ad ogni iterazione il codice di figura 5.16 assegnerà il rispettivo valore di tonalità utilizzando la seguente formula :

$$(((\frac{DIFF}{3}) + 2) \cdot 140) + 400 \tag{5.50}$$

dove la divisione per 3 è fatta ancora una volta utilizzando il blocco matematico della versione 1.0. La realizzazione della formula precedente si basa sul fatto che la variabile $DIFF$ può assumere valori interi compresi nell'intervallo [-8, 8]. Dividendo questa variabile per 3 si ha come effetto quello di ridurre l'intervallo a [-2, 2]. A questo punto sommando il valore 2 alla variabile $DIFF$ si ottiene come effetto quello di scalare l'intervalle facendolo diventare [0, 4]. A questo punto con la moltiplicazione per 140 e la somma del valore 400 si assegnano rispettivamente alle cinque zone (da sinistra

a destra) i valori 400, 540, 680, 820 e 960 Hz. Assegnando queste tonalità alle diverse zone del campo visivo è facile distinguere anche solo ascoltando il cambiamento della posizione degli oggetti.

Una volta terminata la parte relativa alla riproduzione sonora, è possibile procedere con la visualizzazione a display delle posizioni. Per prima cosa vengono visualizzate in basso del display le due distanze VS e VD che serviranno per definire la distanza iniziale alla quale si posiziona il robot.

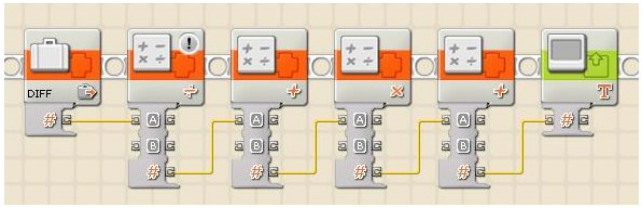


Figura 5.17: Codice per la visualizzazione della posizione a schermo.

Il segmento di codice di figura 5.17 rappresenta la funzione per visualizzare a schermo la posizione rilevata dal robot. Nel display verranno visualizzati cinque cerchi posti sulla stessa linea dello schermo, che rappresenteranno le cinque diverse posizioni. La locazione dell'oggetto viene rappresentata da una "X" che si sposta di cerchio in cerchio. La posizione della "X" è determinata dalla funzione :

$$(((\frac{DIFF}{3}) + 2) \cdot 22) + 4 \tag{5.51}$$

Le operazioni eseguite sono le stesse di quelle eseguite per il calcolo della tonalità eseguito in precedenza, con l'unica differenza che il valore in questo caso viene scalato di 22 e traslato di 4 in modo da centrare la posizione della "X" nei cinque cerchi che verranno visualizzati nel display. Anche in questo caso si può notare dalla figura 5.17 che nel codice la divisione viene effettuata utilizzando il blocco aritmetico della versione 1.0. Questa scelta è necessaria a causa dell'assenza nella versione 2.0 di un blocco che calcola la parte intera di un valore. Utilizzando una divisione con valori decimali non è possibile dividere l'area in cinque zone, infatti il risultato dell'equazione 5.49 non sarebbe mai lo stesso in caso di valori diversi di $DIFF$, impedendo così di centrale la "X" nei cerchi del display. Per esempio prendendo come valori della variabile $DIFF$ 4 e 5 e sostituendoli nella formula 5.49 e utilizzando la divisione intera si ottengono rispettivamente i valori:

$$(((\frac{4}{3}) + 2) \cdot 22) + 4 = 70 \tag{5.52}$$

$$(((\frac{5}{3}) + 2) \cdot 22) + 4 = 70 \tag{5.53}$$

e dunque entrambi appartengono alla zona DX, mentre se si utilizza la divisione decimale della versione 2.0 i risultati diventerebbero :

$$(((\frac{4}{3}) + 2) \cdot 22) + 4 = 77,3 \tag{5.54}$$

$$(((\frac{5}{3}) + 2) \cdot 22) + 4 = 84,6 \tag{5.55}$$

e dunque entrambi i valori non apparterrebbero a nessuna delle cinque zone.

Figura 5.18: Codice per la visualizzazione dei cerchi nello schermo.

Una volta calcolata la posizione in cui visualizzare la "X" non resta che creare i cinque cerchi rappresentanti le diverse zone. In figura 5.18 è presente il codice con il quale ad ogni iterazione vengono creati i cinque cerchi che rappresenteranno le diverse posizioni. L'ultima delle icone di figura 5.18 rappresenta l'attesa tra una misurazione e la successiva. Questo tempo di attesa è stato fissato dopo una serie di prove, in quanto effettuare le misurazioni con una frequenza troppo elevata risulterebbe controproducente, inoltre diventerebbe difficile capire esattamente la posizione assunta dagli oggetti se questi si muovono ad una velocità troppo elevata. Il valore scelto per il tempo di attesa è di 300 millisecondi e consente al sistema la giusta sensibilità ai cambiamenti di posizione degli oggetti, infatti con questo tempo il robot non risulterà troppo sensibile agli spostamenti come descritto in precedenza, ma allo stesso tempo sarà in grado di reagire ai movimenti in tempi brevi.

Capitolo 6

L'equilibrio

In questo capitolo si andrà a studiare la realizzazione di un robot che simula, in modo semplificato, il meccanismo di mantenimento dell'equilibrio del corpo umano. L'idea di fondo è quella di introdurre alcuni importanti concetti fisici che si presentano quando si studia l'equilibrio del corpo umano, senza però dover trattare tutta la vasta teoria che si cela dietro lo studio di questo fenomeno.

6.1 In natura

In questa sessione si andrà ad analizzare il meccanismo che consente al corpo umano di mantenersi in equilibrio durante il compimento delle varie azioni nel corso della giornata. Nell'uomo l'equilibrio si può considerare come un insieme di aggiustamenti automatici ed inconsci che permettono, contrastando la forza di gravità, di mantenere una posizione o di non cadere durante l'esecuzione di un gesto. Il corpo umano infatti lavora continuamente nell'arco di una giornata, per contrastare le forze che agiscono su di esso. L'obbiettivo di tale lavoro è quello di mantenere il baricentro del corpo all'interno dell'area ricoperta della base d'appoggio che è rappresentata dai piedi. Se infatti la proiezione del baricentro uscisse dalla zona di terreno dove sono appoggiati i piedi, il corpo cadrebbe a terra spinto dalla forza di gravità.

A differenza di quanto accade nel mondo animale, l'uomo è alla continua ricerca dell'equilibrio per via della sua caratteristica bipodalica, infatti reggendosi su due soli arti, la base di appoggio risulta essere molto più piccoli rispetto a quanto accade per i quadrupedi e quindi diventa molto più facile che la proiezione del baricentro sul terreno, esca dall'area occupata dai piedi.

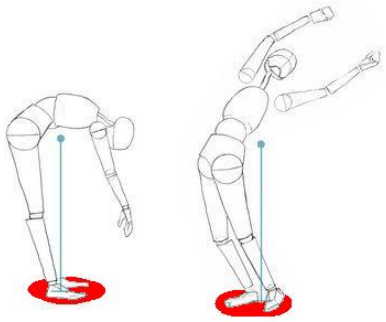


Figura 6.1: Codice NXT-G

Per questo motivo il corpo umano è dotato di un sistema scheletrico composto da numerosi segmenti che costantemente e reciprocamente devono assestarsi per contrastare la gravità. Il movimento di questi segmenti viene comandato dal cervello, il quale , in base ai dati che gli vengono forniti sulla posizione del corpo e sulle forze che agiscono sullo stesso, comanda i muscoli per far muovere le varie parti del corpo in modo da bilanciare le forze.

6.2 La teoria

L'obbiettivo di questa sessione è quello di studiare in modo più approfondito i vari meccanismi che consentono al corpo umano di mantenere l'equilibrio. L'equilibrio è una qualità che consente all'uomo di mantenere o di recuperare una determinata posizione, sia essa statica o dinamica. Questa qualità non è rappresentata da una situazione di riferimento predefinita, ma deriva da un continuo adattamento tonico-posturale da parte del corpo, alle situazioni che si presentano. In ogni momento dunque ogni parte del corpo dovrà muoversi in modo da contrastare le forze che altrimenti lo spingerebbero fuori dal proprio baricentro. In fisica infatti un corpo si definisce in equilibrio se la verticale passante per il baricentro del corpo, cade all'interno della base di appoggio. Nel caso del corpo umano la superficie di appoggio è rappresentata dai piedi, essendo le uniche parti del corpo a contatto con il terreno quando ci si trova in posizione verticale. Non essendo dotato di quattro punti di appoggio come accade per la maggior parte degli animali, il corpo umano deve continuamente lavorare per mantenersi in posizione eretta. Per mantenere l'equilibrio quindi sarà necessario che:

$$\sum_{k=0}^{n} F \tag{6.1}$$

abbia come risultante un vettore diretto verso la superficie d'appoggio. Nel caso in cui la risultante delle forze fosse rappresentata da un vettore con direzione esterna rispetto alla superficie d'appoggio, il corpo dovrebbe reagire per compensare la forza che si genera. In figura 6.2 si vede una situazione in cui il corpo provvede a bilanciare le forze che si generano quando la superficie su cui si trova assume un'inclinazione maggiore, in questo caso la forza F, generata dal cambio di pendenza viene compensata spostando il baricentro in avanti mantenendo così il corpo in equilibrio.

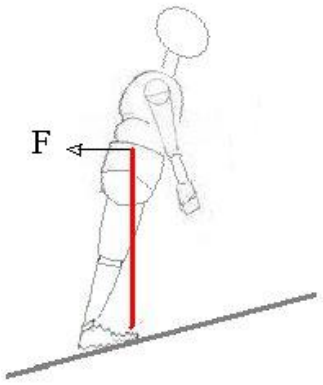


Figura 6.2: Codice NXT-G

Una volta definite le condizioni necessarie al mantenimento dell'equilibrio è necessario descrivere come il corpo umano percepisce di essere in una posizione stabile. L'elemento che permette di capire all'organismo se si trova in posizione di equilibrio è la forza di gravità G. Per orientarsi il corpo umano utilizza questa forza come riferimento assoluto, tanto che quando si trova in posizione eretta, il corpo si dice orientato in direzione della gravità.

Per mantenersi in equilibrio, il corpo necessità di sapere in ogni momento, senza l'ausilio della vista, l'esatta posizione di ognuna delle sue parti. Questa caratteristica è chiamata "propriocezione" e rappresenta la capacità del sistema nervoso di percepire in ogni momento la configurazione assunta dai vari elementi che compongono la struttura motoria del corpo ovvero: il capo, il tronco, le gambe, i piedi e i collegamenti tra questi elementi ovvero la colonna vertebrale, il bacino, le ginocchia e le caviglie. La "propriocezione" è possibile grazie alla presenza in tutto il corpo di recettori sensoriali :

esterocettori cutanei, propriocettori, enterocettori e ricettori vestibolari. Questi recettori hanno la funzione di ricevere i diversi stimoli esterni ed interni all'organismo e di trasmetterli, traducendoli in impulsi elettrici, attraverso le fibre nervose afferenti, al sistema nervoso centrale. Alcuni di questi recettori formano strutture complesse tali da essere considerati organi detti di senso, come ad esempio l'occhio o l'orecchio. Quest'ultimo elemento rappresenta una parte fondamentale per l'equilibrio del corpo umano; All'interno di esso infatti è collocato l'apparato vestibolare.

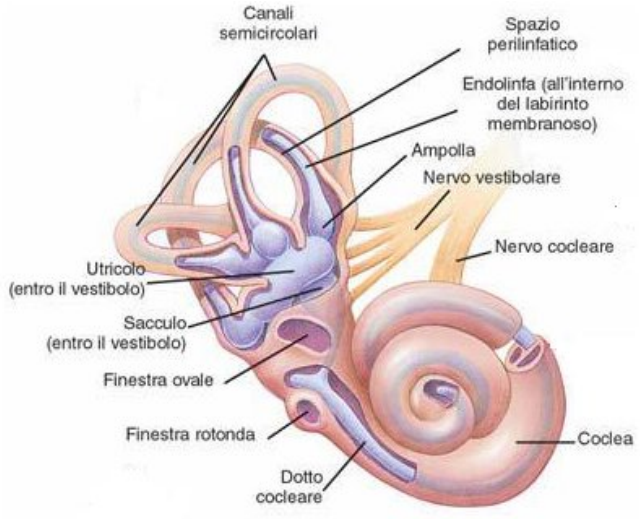


Figura 6.3: Codice NXT-G

Questo apparato, come si vede in figura 6.3 è un organo abbastanza complesso composto da varie parti. Alcune di queste hanno enorme importanza nella percezione delle accelerazioni subite dal corpo umano. L'apparato vestibolare è predisposto per rispondere al meglio ad accelerazioni rapide e di breve durata mentre si lascia facilmente ingannare da accelerazioni lunghe o inconsuete (ciò spiega i capogiri che si hanno quando si ruota più volte su se stessi e ci si ferma di colpo). Il "Dotto Cocleare" è un canale che collega il "Sacculo" alla "Coclea"e contiene dei microcristalli, i quali consentono ai recettori sensoriali, posti nelle pareti del Sacculo, di percepire le accelerazioni verticali alle quali viene sottoposto un uomo. Il Sacculo a sua volta, è in comunicazione con "l'Utricolo", il quale sempre grazie allo stesso meccanismo, fornisce informazioni sulle accelerazioni orizzontali. L'utricolo, inoltre, rappresenta lo sbocco comune dei tre canali semicircolari del labirinto. All'interno di questi sono presenti dei recettori sensoriali che percepiscono i movimenti rotatori di tes-

ta e corpo (ovvero le accellerazioni angolari). Questi sistemi, tutti insieme, forniscono al cervello informazioni sulla posizione della testa e del corpo nello spazio. Queste informazioni arrivano ai "nuclei vestibolari ", situati nel tronco encefalico, che rappresentano il vero organo dell'equilibrio. Ad essi infatti arrivano le informazioni di tutti i recettori sensoriali posturali (vestibolo, esterocettori cutanei, propriocettori e esterocettori visivi) e qui vengono elaborate, assieme alla sostanza reticolare e sotto il controllo del cervelletto, oltre che della corteccia cerebrale, consentendo così al sistema dell'equilibrio, detto sistema tonico posturale, di svolgere il suo compito, ossia di garantire il corretto assetto posturale sia statico che dinamico. Nella creazione, memorizzazione e rappresentazione mentale della posizione verticale del corpo, le informazioni provenienti dal otricolo e dal sacculo sono considerate primarie rispetto a quelle di tipo visivo e propriocettivo. A differenza dell'elaborata e complessa informazione derivante dagli esterocettori cutanei e dai propriocettori, l'orecchio e l'occhio trasmettono all'encefalo una percezione diretta dell'ambiente esterno. Il 90% delle informazioni arriva infatti all'encefalo tramite questi ultimi due canali. Però, affinchè le informazioni derivanti dal sistema vestibolare possano essere interpretate dal sistema posturale, devono essere costantemente comparate e integrate con quelle derivanti dagli altri recettori periferici (visivi, cutanei e propriocettori), in particolare con quelle pressorie derivanti dal piede, unico riferimento fisso nella stazione eretta.

6.3 Simulazione con LEGO ®MINDSTORMS® NXT

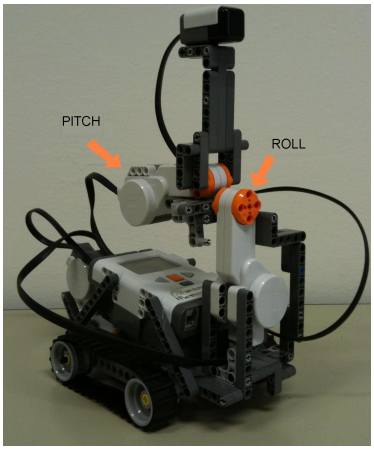


Figura 6.4: Il robot rappresentante l'equilibrio

Il robot che simula la capacità del corpo umano di mantenersi in equilibrio, ha come caratteristica fondamentale quella di avere una struttura relativamente semplice, come si vede in figura 6.4. Questa caratteristica rende più facile capire i meccanismi che si celano dietro a questo automa, in quanto ogni elemento meccanico che consente il mantenimento dell'equilibrio, è facilmente visibile. Il robot in questione è un semplice carro cingolato che viene fatto muovere in linea retta. Questa limitazione è dovuta al fatto che al brick NXT non è possibile collegare più di tre servomotori, quindi, dovendone utilizzare due per la parte relativa all'equilibrio, non è stato possibile collegare un motore ad ogni singolo cingolo, consentendo così al robot di poter svolgere anche cambi di direzione. Sul carro cingolato è montata una coppia di motori collegati tra loro, che hanno il compito di mantenere in equilibrio una colonna verticale sopra la quale è situato un accelerometro. I due motori (Roll e Pitch di figura 6.4) così configurati servono a compensare le accelerazioni , laterali e frontali, subite dalla colonna, e lo fanno muovendola nelle quattro direzioni: avanti, indietro, destra, sinistra. Così facendo la colonna può mantenersi in equilibrio durante il movimento del carro, sia che ci si trovi in salita, discesa o ad un'inclinazione laterale. La simulazione prevede che il robot, avanzando nel suo movimento, venga sottoposto a diverse inclinazioni, laterali e frontali, posizionando sempre la torre in posizione perpendicolare rispetto al suolo, mantenendo così il robot in equilibrio.

Durante la realizzazione di questo robot sono emersi una serie di problemi dovuti alle caratteristiche del materiale utilizzato. Infatti lavorando con la tecnologia Mindstorm bisogna tener presente che i servomotori, pur demoltiplicati, non sono in grado di garantire una grande forza in frenata, per questo motivo la colonna non può avere un peso troppo elevato altrimenti i motori non saranno in grado di mantenerla in posizione. Inoltre per non rischiare che la colonna cada, a causa della scarsa forza frenante, sarà necessario che i motori non spostino la colonna con una velocità angolare troppo elevata. Questo ultimo problema avrà delle ripercussioni nei tempi di risposta del sistema: infatti, se diminuisce la velocità, aumenta il tempo impiegato per portare la colonna in posizione e dunque la frequenza delle correzioni dovute al controllo in retroazione diminuisce, facendo diventare il sistema meno preciso.

La questione che ha causato i maggiori problemi in fase di realizzazione, però, è stata la necessità di mantenere ridotto il peso della colonna. Va ricordato che fanno parte della torre i due motori che ne gestiscono il movimento, i quali hanno un peso molto elevato. Questo peso se distribuito nella struttura in modo erra-

to, porterebbe alla caduta della colonna non appena la verticale passante per il baricentro della stessa, esce dalla base di appoggio.

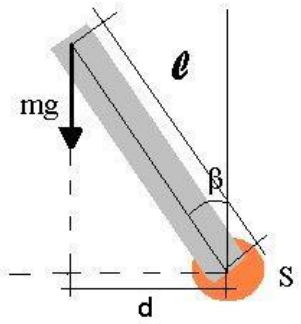


Figura 6.5: Forze che contribuiscono al movimento della colonna

Per ridurre questo problema è necessario trovare una configurazione dei motori, tale da mantenere il baricentro della struttura, quanto più possibile vicino al centro della parte rotante del motore di ROLL. La struttura della colonna infatti, può essere vista come un esempio di pendolo inverso, ovvero vi è un'asta(in questo caso la colonna) libera di muoversi e vincolata solo all'estremo inferiore ad uno snodo che può muoversi per bilanciare le oscillazioni della parte superiore.

La forza che contribuisce al movimento, come si vede in figura 6.5, è esclusivamente la forza peso. In questo caso infatti l'accelerazione angolare "α" del punto "S", è data da:

$$\alpha = \frac{\tau}{m \cdot l \cdot d} \tag{6.2}$$

dove m è la massa della colonna e τ il momento torcente delle forze rispetto al punto fisso. Il momento torcente di una forza dipende dal suo braccio, che nel caso della forza peso generata dalla colonna, è rappresentato dal valore della distanza "d". Con l'aumentare di questo parametro, aumenta anche il momento torcente, pertanto aumenta anche la l'accelerazione subita dal punto S. Considerando la seconda legge di Newton:

$$\sum_{k=0}^{n} \vec{F}_k = m \cdot \vec{a} \tag{6.3}$$

si deduce che l'aumento dell'accelerazione, implica un aumento della forza che il motore deve applicare per mantenere ferma la colonna. L'elemento che consente di diminuire tale forza, è il parametro d, che è dato da:

$$d = l \cdot \sin\beta \tag{6.4}$$

Il parametro d quindi dipende, oltre che dall'angolo β, anche dalla distanza l. Va ricordato che la lunghezza l è la distanza che divide il punto "S" dal baricentro della colonna. In questo caso quindi per ridurre il parametro d e di conseguenza ridurre anche l'accelerazione angolare "α", basterà avvicinare il baricentro della colonna, al punto S. La migliore configurazione per minimizzare la problematica relativa all'eccessivo peso della colonna quindi, è quella utilizzata nella realizzazione di questo robot, ovvero quella in cui il motore di PITCH si trova all'altezza della parte mobile del motore di ROLL: infatti, così facendo, una buona parte del peso si trova ad una distanza d prossima allo 0 e dunque il momento della forza peso generata è pari a 0.

Per mantenere in posizione di equilibrio il robot è necessario effettuare continuamente un controllo in retroazione. Il termine controllo definisce l'azione svolta per portare e mantenere ad un valore prefissato, un parametro fisico di un impianto o di un processo che nel nostro caso sarà rappresentato dalla posizione della colonna. Indicando con r(t) il valore che si vuole far assumere alla variabile controllata, con y(t) il valore effettivamente assunto da tale grandezza, è possibile definire la funzione errore come:

$$e(t) = r(t) - y(t) \tag{6.5}$$

Lo scopo dell'azione di controllo è quello di applicare la migliore scelta possibile per la funzione $u(t)$ (detta variabile di controllo) in modo da rendere il sistema asintoticamente stabile, minimizzare il valor medio dell'errore r, ridurre al minimo il tempo di risposta del sistema. Passando ora alla trasformata di Laplace, è possibile rappresentare il problema del controllo in retroazione con lo schema di figura 6.6.

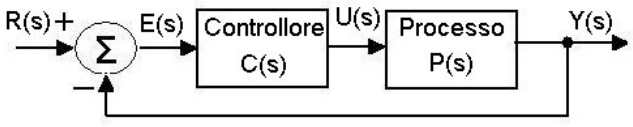


Figura 6.6: Schema a blocchi di un sistema retroazionato

Le principali componenti di tale schema sono: il rivelatore di errore $E(s) = R(s) - Y(s)$, il controllore che ha il compito di trasformare il segnale d'errore in un segnale $U(s)$ che agisce sul processo sottoposto a controllo ed il processo stesso P(s). Supponiamo che

sia il controllore (blocco C) sia il processo (blocco P) possano essere schematizzati come sistemi lineari e stazionari, caratterizzati ognuno da una funzione di trasferimento, rispettivamente C(s) e P(s). La funzione di trasferimento del sistema retroazionato diventa:

$$T(s) = \frac{Y(s)}{R(s)} = \frac{C(s) \cdot P(s)}{1 + (C(s) \cdot P(s))} \tag{6.6}$$

Il problema generale del controllo si riduce quindi a determinare, per una certa funzione di trasferimento del processo P(s), la migliore funzione di trasferimento del controllore C(s) che ottimizza la T (s).

Nel nostro caso però il sistema è un po' più complesso come si vede in figura 6.7

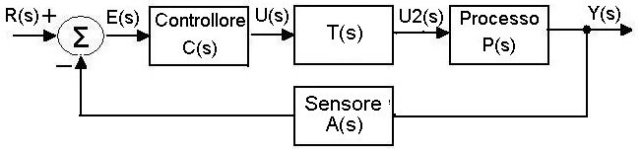


Figura 6.7: Schema a blocchi di un sistema retroazionato

In questo caso nella retroazione è presente il blocco A che rappresenta il sensore che effettua le misurazioni, dei valori assunti dalla variabile da controllare. Il sensore utilizzato nel nostro caso misura le accelerazioni quindi, il rilevatore di errore E(s), fornisce un valore che rappresenta un'accelerazione al controllore C(s). L'uscita U(s) fornita dal controllore, rappresenta a sua volta un'accelerazione. In questo sistema però, la variabile da controllare è la posizione della colonna infatti: si vuole che la torre, in ogni momento, sia posta in posizione perpendicolare rispetto al piano su cui si trovava il robot inizialmente. La variabile da controllare rappresenta quindi una variazione angolare e dunque, l'ingresso U2(s) del Processo P(s), dovrà essere di questo tipo. Tra il controllore C e il processo P perciò, è stato inserito il blocco T che porta l'uscita del controllore nella forma richiesta in ingresso dal processo.

Nello schema di figura 6.7 il blocco processo, è rappresentato dal movimento del motore che regola la posizione della colonna. Tale motore viene controllato in posizione dal sistema retroazionato, poiché deve compensare gli spostamenti subiti dalla colonna. Oltre alla posizione, il motore viene anche controllato in velocità da un controllore interno di tipo PID. Tale controllo ha lo scopo di far mantenere la velocità costante al motore, anche nel caso in cui questo si trovi sotto sforzo per cause esterne.

Per quanto riguarda il controllore, in questo caso si è scelto di utilizzare un controllore di tipo PI, ovvero l'utilizzo combinato di due funzioni, quella Proporzionale e quella Integrale. La componente derivativa, che è quella che interviene nel caso di brusche variazioni, è stata omessa allo scopo di evitare inutili instabilità, a scapito di una certa prontezza che, in queste dimostrazioni, è richiesta in misura limitata.

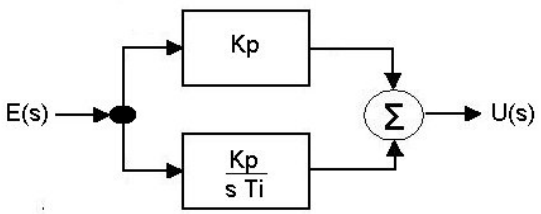


Figura 6.8: Schema del controllore PI

In figura 6.8 è riportato lo schema di tale controllore: è facile vedere come l'uscita U(s) sia data dalla somma delle due componenti, quella proporzionale e quella integrale ovvero:

$$U(s) = (K_p \cdot E(s)) + (\frac{K_p}{s \cdot T_I} \cdot E(s)) \tag{6.7}$$

dove K_p e T_I sono due costanti positive. Il problema del progetto di un controllore PID si riduce alla scelta dei valori più opportuni per i parametri K_p e T_I. Tale scelta non è banale perché richiede la conoscenza dettagliata delle proprietà del processo che si vuole controllare. Nel caso della progettazione del controllore PI utilizzato in questo robot, si è deciso di calibrare le costanti per via sperimentale, provando nella pratica quali valori siano più opportuni.

Nel caso del nostro controllore è stato necessario effettuare una serie di prove per trovare il valore migliore delle costanti K_p e T_I. Dopo una serie di prove sono stati scelti i valori:

$$K_p = 0,5 \tag{6.8}$$

$$T_I = 0,5 \tag{6.9}$$

Nella costruzione del robot bisogna ricordare che andranno utilizzati non uno, ma ben due controllori PI che però sono sostanzialmente indipendenti. Le variabili da controllare infatti, sono sia l'angolo di spostamento sull'asse X, che quello sull'asse Y. Per questo motivo lo schema a blocchi del sistema diventa quello di figura 6.8.

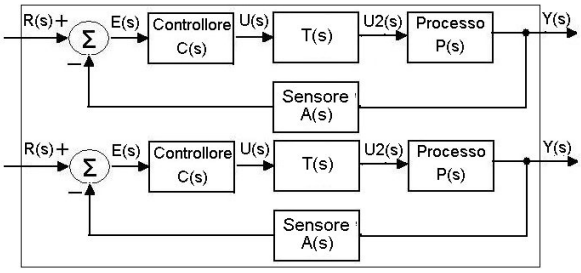


Figura 6.9: Schema dei due controllori PI relativi agli spostamenti sugli assi X e Y

In questo caso verrà utilizzato, per entrambi i controllori, gli stessi parametri K_p e T_I calcolati in precedenza. Una volta calcolati i parametri relativi ai controllori PI si può procedere con i calcoli relativi al movimento della colonna.

Figura 6.10: Configurazione del'accelerometro

Per calcolare i movimenti da far fare ai due motori è necessario misurare le accelerazioni che vengono subite dal robot. Per fare ciò viene utilizzato l'accelerometro NXT posto in configurazione concorde a quella di figura 6.10. Ai fini di bilanciare la colonna, verranno misurate solo le accelerazioni lungo gli assi X e Y mentre non verranno considerate quelle sull'asse Z. Per bilanciare tali accelerazioni sarà necessario calcolare l'angolo di inclinazione che si crea quando la torre viene mossa dalle forze che agiscono su di essa. Il calcolo dell'angolo va fatto per entrambi gli assi X e Y ed è dato rispettivamente dalle formule:

$$\alpha = \arcsin(\frac{A_x}{g}) \tag{6.10}$$

$$\beta = \arcsin(\frac{A_y}{g}) \tag{6.11}$$

con A_x e A_y rispettivamente accelerazione nell'asse X e nell'asse Y rilevate dal sensore e "g" accelerazione gravitazionale.

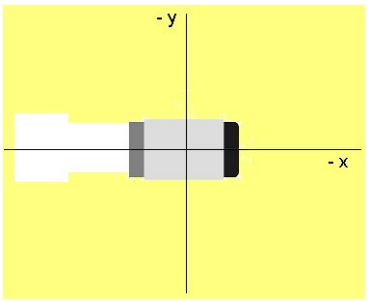


Figura 6.11: Orientamento dell'accelerometro

Una volta calcolato il valore di uscita u(t) per entrambi i controllori, sarà possibile far muovere i motori per portare la torre in posizione di equilibrio. Prima di fare ciò è però necessario definire in quale direzione far muovere i due motori. Osservando la figura 6.10, dove si vede l'accelerometro visto dall'altro, si nota come per un angolo di inclinazione sull'asse X positivo, il motore di PITCH dovrà far ruotare la colonna in avanti per mantenerla in equilibrio. Per quanto riguarda l'angolo di ROLL se l'angolo percepito è positivo allora il motore dovrà ruotare la colonna verso la parte sinistra del robot. I versi dei motori andranno gestiti in fase di programmazione in base alla configurazione degli stessi nel robot.

6.4 Il programma

La fase di programmazione relativa al robot descritto in questo capitolo, ha avuto come attività preliminare, lo studio dei dati forniti dall'accelerometro NXT della Hitecnic. È importanti infatti conoscere la struttura ed il significato dei dati forniti dal sensore, per non commettere errori. Il sensore utilizzato fornisce l'accelerazione nei tre assi X, Y e Z in una gamma da -2G a +2G con una sensibilità di 200 valori per ogni G. il sensore quindi fornisce valori interi nell'intervallo [-400 , 400]. Se il sensore ritorna un valore di 200 quindi significa che l'accelerometro è sottoposta all'accelerazione di un G.

Una volta definita la struttura dei dati ricevuti in input, si può procedere con la fase di programmazione.

Per prima cosa è necessario effettuare una misurazione delle accelerazioni degli assi X e Y quando il robot si trova fermo nella configurazione iniziale, memorizzandole entrambe in una variabile.

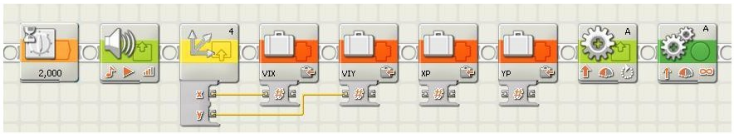


Figura 6.12: Codice che inizializza le variabili.

Nel codice NXT-G di figura 6.12 si vede come i valori delle accelerazioni iniziali vengano memorizzati nelle due variabili VIX e VIY. La memorizzazione di questi valori nelle due variabili servirà per poter avere un elemento di confronto, infatti se non venissero conservati tali valori, non sarebbe possibile determinare l'errore nelle successive misurazioni.

Una volta definite ed inizializzate tutte le variabili, il programma entra in un ciclo infinito in modo da ripetere continuamente il controllo retroazionato. All'inizio di ogni iterazione il programma effettua una lettura del sensore NXT per conoscere l'eventuale spostamento della colonna.

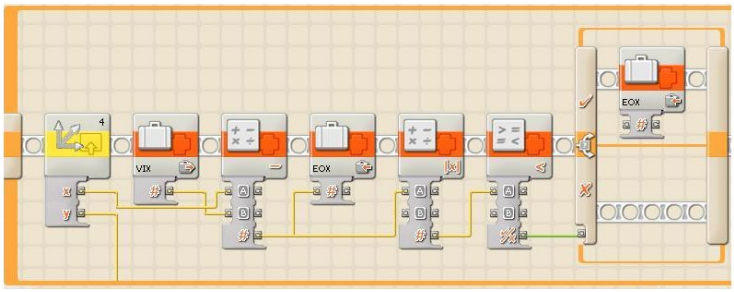


Figura 6.13: Codice che esegue la lettura dei valori del sensore, calcola l'errore e ne filtra i valori.

Una volta letto il valore dell'accelerazione come si vede in figura 6.13, viene calcolato l'errore ovvero :

$$EOX = x - VIX \tag{6.12}$$

con "x" valore dell'accelerazione nell'asse X misurato dal sensore e VIX valore dell'accelerazione misurato con il robot in posizione iniziale. La stessa operazione viene effettuata per l'accelerazione sull'asse delle Y ovvero :

$$EOY = y - VIY \tag{6.13}$$

Nel codice di figura 6.13 gli ultimi 3 blocchi servono per diminuire la sensibilità del sistema. Infatti nel caso in cui il modulo dell'errore sia rappresentato da un valore minore di 15, il programma set-

ta la variabile EOX a 0, considerando quindi nulla la variazione rispetto alla posizione iniziale. Questa operazione viene eseguita per evitare che il sistema diventi instabile, infatti gli errori il cui valore è inferiore di 15, rappresentano variazioni angolari molto ridotte e trascurabili ai fini del mantenimento dell'equilibrio.

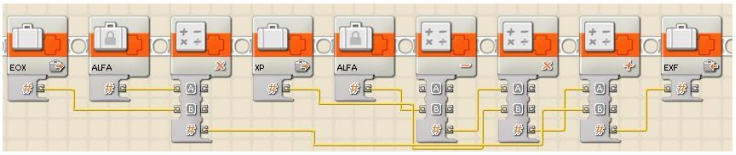


Figura 6.14: Codice del relativo controllore PI.

Se venissero considerate invece, si assisterebbe ad una continua e fastidiosa oscillazione della colonna, in quanto l'errore non sarebbe quasi mai uguale a zero. Questo tipo di filtro andrà applicato ad entrambi gli errori EOX e EOY per garantire la stabilità totale del sistema.

Una volta calcolati e filtrati i valori dei due errori è necessario calcolare il valore dell'uscita relativa al sistema di controllo PI. In figura 6.14 è riportato il codice relativo al calcolo del valore di uscita del controllore PI utilizzato per controllare il posizionamento sull'asse X della colonna. Ovviamente analogo sarà il codice per quanto riguarda l'asse delle Y. Per effettuare il calcolo dell'uscita del controllore PI è necessario conoscere l'errore EOX e l'errore calcolato nell'iterazione precedente, che viene memorizzato nella variabile XP (inizializzata al valore 0 all'avvio del programma). Il calcolo dell'uscita è dato da :

$$EXF = (EOX \cdot ALFA) + (XP \cdot (1 - ALFA)) \tag{6.14}$$

con $ALFA$ uguale a 0,5, e rappresentante il valore della costante K_p calcolata nella sessione precedente. Il valore EXF a questo punto rappresenta l'uscita del nostro controllore. Il risultato di tale calcolo però è relativo alla differenza di accelerazione tra la posizione attuale e quella iniziale. Per poter equilibrare la torre è necessario conoscere il valore dell'angolo relativo a tale accelerazione, il quale viene fornito dalla formula :

$$\alpha = \arcsin \frac{A}{g} \tag{6.15}$$

dove A è l'accelerazione misurata. A questo punto sorge un problema relativo al calcolo di tale valore con il programma NXT-G. Questo infatti non possiede la funzione "arcsin" e dunque non

è possibile applicare la formula 6.15. In NXT-G però è possibile importare un blocco che calcola il valore dell'arcotangente quindi essendo :

$$\arcsin x = 2 \cdot \arctan\left(\frac{x}{1+\sqrt{1-x^2}}\right) \tag{6.16}$$

la 6.15 diventa :

$$\alpha = 2 \cdot \arctan\left(\frac{\frac{A}{g}}{1+\sqrt{1-(\frac{A}{g})^2}}\right) \tag{6.17}$$

con

$$\frac{A}{g} = \frac{EXF}{200} \tag{6.18}$$

La divisione per 200 è dovuta al fatto che il sensore ritorna 200 valori ogni g di accelerazione.

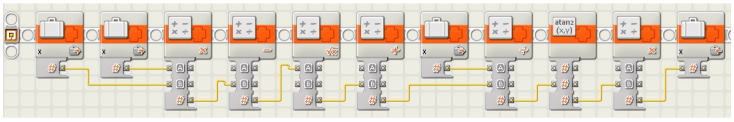


Figura 6.15: Codice per il calcolo della funzione arcoseno.

Il risultato della formula 6.17 è memorizzato nella variabile $ANGX$ e per calcolarlo è stato creato un blocco chiamato "arcoseno2" il cui codice è riportato in figura 6.15. Tale blocco verrà riutilizzato per il calcolo dell'angolo relativo all'asse Y il cui valore sarà salvato nella variabile $ANGY$.

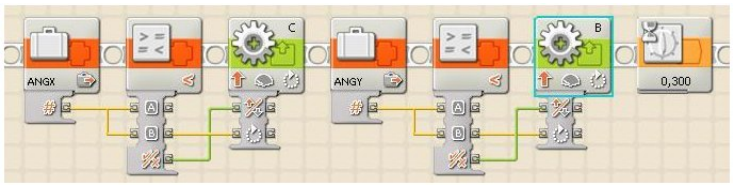


Figura 6.16: Codice che aziona i motori che muovono la colonna.

Una volta trovati i valori dei due angoli sarà possibile procedere al movimento dei motori di PITCH E ROLL. In figura 6.16 viene riportata la parte di codice relativa al movimento dei due motori. I motori lavorano in serie in quanto, farli lavorare in parallelo richiederebbe un lavoro di sincronizzazione tra i due motori e la lettura dei sensori, di difficile gestione.

Per prima cosa si fa muovere il motore di PITCH che viene collegato alla porta C. Il verso del motore dipende dal valore della variabile $ANGX$, infatti se la variabile assume valori negativi, significa che la colonna necessita di essere spinta verso l'indietro e quindi il motore C deve essere mosso in senso orario per un numero di gradi uguale a $ANGX$. In caso contrario, ovvero per valori di $ANGX$ positivi, la colonna deve essere spinta in avanti e quindi il motore de girare in senso antiorario. Una volta terminata tale operazione si muove il motore di ROLL che è collegato alla porta B. In questo caso per valori della variabile $ANGY$ negativi il motore dovrà spingere la colonna verso la parte destra del robot e quindi dovrà compiere una rotazione di $ANGY$ gradi in senso orario. In caso di valore dell'angolo minore di 0 il motore verrà mosso in senso antiorario.

L'ultimo blocco di figura 6.15 rappresenta un'attesa di 300 millisecondi. Questa attesa è necessaria per la stabilità del sistema. Il suo valore infatti è stato deciso dopo una serie di prove dalle quali è emerso che per valori inferiori a quello stabilito, il sistema commette degli errori e diventa instabile. Se invece vengono usati valori maggiori di quello utilizzato, il tempo di reazione agli eventi del sistema diventa troppo elevato e dunque il robot non riesce a mantenere l'equilibrio.

Conclusioni

Nonostante alcuni limiti dovuti alla tecnologia MINDSTORM utilizzata nello sviluppo di questa tesi, è stato possibile raggiungere tutti gli obbiettivi iniziali che erano stati prefissati.

Il robot che simula il movimento del verme riproduce in modo soddisfacente, il movimento peristaltico tipico dei vermi. L'unica limitazione di questa simulazione è dovuta all'apertura massima del robot in quanto, non è stato possibile far distendere completamente il robot durante la fase di allungo, a causa delle limitazioni tecnologiche del materiale utilizzato. A parte questa piccola limitazione, il movimento del robot risulta fluido e rappresenta in modo abbastanza fedele il movimento per il quale è stato creato.

Per quel che riguarda il "visore stereoscopico" si può dire che il robot prodotto, grazie anche al buon sistema di filtri utilizzato, riconosce in modo soddisfacente gli oggetti che si muovono in modo lineare nel suo raggio d'azione. L'unico limite di questo robot è dovuto al fatto che , può non essere percepita la variazione di posizione all'interno del campo visivo, se l'oggetto passa improvvisamente da una zona ad un'altra non immediatamente vicina. È possibile ovviare a tale problema eliminando l'ultimo filtro impiegato, ma questo porterebbe a delle oscillazioni fastidiose nell'output prodotto dal robot.

Anche la terza simulazione può considerarsi in svolta in modo positivo. Il robot costruito è in grado di mantenere abbastanza precisamente la posizione di equilibrio della colonna. Questo esempio è senza dubbio quello che più ha risentito dei limiti tecnologici della tecnologia mindstorm. La scarsa forza frenante dei motori, ha portato alla necessità di ridurre la velocità dei movimenti di aggiustamento della colonna . Questo fatto ha portato ad un aumento dei tempi di risposta del sistema e dunque è stato necessario rimuovere la componente derivativa del controllore PID. Eliminando questa componente il robot può incontrare delle difficoltà in caso di bruschi cambi di inclinazione, ma essendo questi degli esempi educativi si può affermare che il risultato ottenuto con la costruzione di questo

robot è soddisfacente.

Ringraziamenti

Voglio ringraziare innanzitutto il professor Moro per la disponibilità dimostrata, durante l'intero sviluppo di questa tesi, nel correggere ed indirizzare il mio lavoro ai fini di raggiungere gli obbiettivi che ci eravamo inizialmente imposti.

Ringrazio tutti i colleghi con cui ho affrontato la mia carriera universitaria, senza i quali probabilmente adesso non starei scrivendo questa tesi. Tutti loro infatti sono stati un valido aiuto nella preparazione degli esami, ma soprattutto mi hanno fatto trascorrere degli anni indimenticabili.

Per ultima, ma non per questo meno importante, ringrazio la mia famiglia per avermi sempre sostenuto e sopportato durante tutto il mio percorso di studi....