UNIVERSITÀ DEGLI STUDI DI BRESCIA
FACOLTÀ DI INGEGNERIA
CORSO DI LAUREA SPECIALISTICA IN INGEGNERIA INFORMATICA
DIPARTIMENTO DI ELETTRONICA PER L'AUTOMAZIONE

STUDIO E REALIZZAZIONE DI UN LABORATORIO DIDATTICO DI ROBOTICA MOBILE UTILIZZABILE REMOTAMENTE VIA INTERNET

Docente:
Prof. Riccardo Cassinis

Studente:
Samuele Gatti

ANNO ACCADEMICO 2009-2010

"La disumanità del computer sta nel fatto che,
una volta programmato e messo in funzione,
si comporta in maniera perfettamente onesta."

Isaac Asimov

Indice

Capitolo 1

Introduzione

La robotica per come viene intesa ai giorni nostri è una scienza di sintesi che raccorda in un unico contesto tutti i saperi e gli ambiti di conoscenza della cultura umana. L'affermazione potrebbe sembrare molto forte, ma ad un'attenta analisi quasi tutte le discipline, in un modo o nell'altro, hanno a che fare con la robotica.
In particolare, l' idea di materia inanimata che diventa servente dell'essere umano va ben al di là della pura essenza tecnologica. A tale proposito l'illustre professor Marco Somalvico ha definito la robotica "una disciplina onnivora", in grado di attingere teorie e metodologie da discipline tradizionalmente slegate tra loro. In un'era nella quale le tecnologie divengono sempre più parte integrante della quotidianità di ogni individuo, l'inserimento della robotica nella programmazione didattica fin dall'infanzia rappresenta non solo una disciplina in grado di affascinare chiunque ne venga a contatto, ma soprattutto uno strumento educativo di enorme portata. Quando si parla di robotica spesso si fa semplicemente riferimento alla realtà industriale, dalle catene di montaggio delle autovetture alle operazioni di saldatura e verniciatura. Ma l'industria rappresenta solo un sottoinsieme dei possibili contesti nei quali la robotica trova applicazione. Vent'anni fa nessuno avrebbe mai pensato che un medico potesse eseguire un intervento chirurgico su un paziente attraverso una procedura di teleoperazione, né tanto meno che un'automobile potesse percorrere un ostico circuito di rally senza ospitare a bordo un pilota. Questi sono solo alcuni degli enormi progressi raggiunti dopo anni di studi e di ricerche nelle quali l'obiettivo

di fondo è sempre stato lo stesso: arricchire una scienza “a servizio dell’uomo”. In un contesto di questo tipo, si capisce bene quanto sia fondamentale il ruolo della formazione degli individui, fin dalla scuola elementare. La scelta degli insegnanti di inserire la robotica all’interno dei percorsi didattici è la chiave di volta per condurre i bambini allo sviluppo di una “forma mentis” adeguata alla comprensione della disciplina in tutte le sue sfaccettature.
L’esperienza degli ultimi anni ha rivelato vincente l’approccio alla robotica mediante l’introduzione di kit di costruzione e la realizzazione di appositi laboratori di didattica. L’idea di poter toccare con mano e di interagire fisicamente con strumenti a basso costo consente lo sviluppo di dinamiche cognitive che non riescono ad emergere attraverso un semplice apprendimento nozionistico. In questo contesto si inserisce la realizzazione di Mobolab, un laboratorio didattico remotamente fruibile tramite Internet. Uno strumento che vuole dare il proprio contributo e inserirsi in maniera attiva nel panorama delle tecnologie per la didattica e per l’insegnamento della robotica nella scuola.

Capitolo 2

Obiettivi

Il lavoro di tesi si pone l'obiettivo di progettare e realizzare un laboratorio didattico all'interno del quale poter vivere delle vere e proprie esperienze di robotica educativa. Dal momento che il laboratorio deve avere come requisito imprescindibile l'utilizzo remoto tramite Internet, è richiesta un'attenta fase di progettazione sulla base della quale comprendere quelle che sono le reali esigenze degli utenti e i loro desideri. Il passo immediatamente successivo prevede la costruzione dell'infrastruttura fisica del laboratorio e la dotazione di tutti gli accorgimenti necessari per rendere l'esperienza il più possibile reale e godibile. Il vero obiettivo di fondo consiste nel voler mettere a disposizione dell'utenza dei robot "giocattolo" a basso costo e consentire a chiunque di manovrare e programmare tali apparecchiature. Lo sforzo più grande risiede nel rendere tutto ciò fruibile attraverso un web browser, nell'ottica in cui ogni singolo utente possa disporre di un calcolatore e di una connessione verso Internet. Quest'ultimo punto prevede lo sviluppo di apposite applicazioni web, che si prefiggono l'obiettivo di offrire diversi livelli di interazione. Innanzitutto si vuole permettere agli utenti l'esplorazione dei robot Lego Mindstorm e la loro manipolazione diretta attraverso un telecomando. Il secondo passo prevede la creazione di un ambiente pensato per la scuola dell'infanzia e orientato al supporto dell'attività di tutti quegli insegnanti che nutrono il forte desiderio di introdurre la robotica nei percorsi didattici fin dalla scuola elementare. L'ultimo passo consiste nel fornire all'utenza strumenti di programmazione avanzata con i quali realizzare

comportamenti più complessi e approfondire la conoscenza sia dei linguaggi di programmazione che dei dispositivi NXT.

Capitolo 3

Stato dell' arte

La realizzazione di un laboratorio didattico di robotica rappresenta uno strumento educativo di enorme portata, il veicolo per accrescere la propria conoscenza attraverso la manipolazione e la costruzione effettiva di oggetti. Lo sviluppo e la diffusione di nuove tecnologie determinano dei cambiamenti nei modi di apprendere e di insegnare. Il contesto attuale delle nuove tecnologie viene ricondotto all'idea di costruzionismo di Papert, pilastro fondamentale per la formazione secondo il quale è possibile offrire il maggior apprendimento con il minimo insegnamento.

3.1 Il costruzionismo

Seymour Papert oggi è considerato il più famoso esperto al mondo su come la tecnologia possa aprire nuovi percorsi all'apprendimento. Ha seguito progetti educativi in ogni parte del mondo, molti dei quali in villaggi sperduti di Paesi in via di sviluppo.
Papert è fondatore del costruzionismo, una nuova interpretazione e rivisitazione delle osservazioni portate dagli psicologi cognitivisti, il cui uso pedagogico viene fatto risalire alla dottrina di Piaget. Secondo Papert il processo di apprendimento è un processo di costruzione di rappresentazioni più o meno corrette e funzionali del mondo con cui si interagisce.
L'idea di costruzionismo può essere ben riassunta dal seguente aforisma:

"se un uomo ha fame gli puoi dare un pesce, ma meglio ancora è dargli una lenza e insegnargli a pescare".

Ad esso Papert aggiunge la seguente considerazione:
"Naturalmente, oltre ad avere conoscenze sulla pesca, è necessario anche disporre di buone lenze, ed è per questo che abbiamo bisogno di computer e di sapere dove sono le acque più ricche".

Si comprende bene che ciò che viene posto al centro dell'attenzione non concerne l'azione di sfamare e il soggetto che la compie, ma chi deve essere sfamato e la necessità di fornirgli le risorse e gli strumenti per poter appagare i propri bisogni non soltanto in questo momento, ma possibilmente per tutto l'arco della sua vita.
Il costruttivismo è quindi un nuovo quadro teorico di riferimento che pone il soggetto che apprende al centro del percorso formativo, in alternativa ad un approccio educativo basato sulla centralità dell'insegnante, depositario indiscusso di un sapere universale, astratto e indipendente dal contesto di riferimento. L'apprendimento non viene visto come un'attività personale, ma come il risultato di una dimensione collettiva d'interpretazione della realtà. La nuova conoscenza viene costruita non solo in base a ciò che è stato acquisito in esperienze passate, ma soprattutto attraverso la condivisione di significati espressi da una comunità di interpreti.

A differenza dell'approccio cognitivista nel quale l'insegnamento è un processo di trasmissione dell'informazione e l'apprendimento di elaborazione ricettiva, il costruzionismo si pone come esperienza situata in uno specifico contesto: il soggetto, spinto dai propri interessi costruisce una propria concezione della realtà attraverso un processo di integrazione delle molteplici prospettive offerte. L'obiettivo che si intende perseguire riguarda l'interiorizzazione di una metodologia d'apprendimento che renda il soggetto progressivamente autonomo nei propri processi cognitivi. Secondo Papert lo scopo dell'istruzione non è quello di alimentare le persone con del sapere codificato ("il pesce"), ma quello di far scoprire al soggetto stesso le specifiche conoscenze di cui ha bisogno ("il pescare").

In questo modo il sapere che viene promosso diviene un aiuto fondamentale nell'acquisizione di altro sapere.

3.2 Il linguaggio Logo

Per comprendere la nascita del linguaggio Logo bisogna fare riferimento all'informatica dei primi anni '80. Per la prima volta si inizia a parlare di home computer, appannaggio di pochi appassionati e con il quale, senza la conoscenza di comandi testuali del sistema operativo e quelli di almeno un linguaggio informatico, non si poteva fare praticamente nulla.
Piegare questi calcolatori ad eseguire qualcosa di utile era un'impresa non da poco, dal momento che i pochi software esistenti erano molto costosi, documentati in un inglese rigido e poco comprensibile. In uno scenario di questo tipo la sola idea di avvicinare un bambino ad un calcolatore sembrava pura fantascienza.

La vera rivelazione si ha con la nascita del linguaggio Logo ad opera di Seymour Papert, un linguaggio di programmazione multilingua ad alto tasso di gratificazione, in grado di disegnare forme geometriche. Una delle grandi potenzialità del Logo è da individuare in quella che oggi viene comunemente definita "interfaccia amichevole"; i primi home computer infatti, all'atto dell'accensione, presentavano una schermata nera ed un piccolo trattino lampeggiante in alto a sinistra nell'attesa di inserimento di caratteri da parte dell'utente. Il Logo presentava una vera e propria interfaccia al cui centro era collocato un oggetto grafico (un triangolino a rappresentazione di una tartaruga) che si spostava a seconda del comando che immesso dall'utente.

Agli inizi del Novecento la pedagogia scientifica porta ad una serie di capovolgimenti profondi: il discente viene portato ad avere un ruolo centrale nell'attività didattica, con particolare attenzione ai suoi tempi e alle sue modalità di apprendimento. Negli anni '60 si giunge all'esplicito rifiuto di una pedagogia "passivizzante" nei confronti dello studente, rifiuto che culmina all'inizio degli anni '70.

L'evoluzione della pedagogia in questo senso si sposta sempre più verso la didattica abbandonando le sue origini filosofiche. L'apprendimento diviene sviluppo dell'intelligenza e non più semplice possesso di conoscenza mnemoniche. Questo nuovo tipo di impostazione conduce al tramonto del pensiero comune secondo cui l'intelligenza possa essere "misurata". La molteplicità della struttura mentale rende unico ogni essere e quindi non può essere oggetto di misurazione con scale e livelli. Non essendo possibile fornire una misura oggettiva dell'intelligenza, è comunque possibile valutarne i prodotti, ovvero le elaborazioni, frutto dell'intelligenza e del processo di apprendimento. Al centro della valutazione non sono più presenti le nozioni memorizzate, bensì le abilità possedute.
L'approccio didattico promosso da Papert è stato spesso sminuito ad una sorta di "imparare facendo". Il costruttivismo e la "filosofia del Logo" non possono essere ridotto a semplici manipolazioni di oggetti. Citando a tal proposito le parole di Papert:

"Chi programma in Logo rifiuta la preoccupazione scolastica di avere risposte giuste e sbagliate; rifiutare giusto e sbagliato non significa che tutto va bene, la vita, il senso della vita non è avere la risposta giusta ma portare avanti le cose! Il concetto riferibile alla cultura del Logo porta a fare in modo che succeda ed è molto più di un principio educativo o pedagogico". È meglio descrivibile come il riflesso di una filosofia del vivere piuttosto che di una teoria dell'educazione. È anche qualcosa di più specifico del costruttivismo nel senso comune attribuito a questo termine.

3.3 Dal Logo al mattoncino programmabile

Descrivere la tartaruga del Logo come un animale artificiale fa intuire che essa può muoversi, ha una direzione di movimento preferenziale e può ripetere una stessa sequenza di movimenti un certo numero di volte. In sostanza l'oggetto che viene rappresentato sullo schermo ha delle analogie con una tartaruga reale e quindi sembra essere molto più famigliare.
La tartaruga virtuale però non ha un vero e proprio corpo e non è possibile

toccarla, ma soprattutto quest'ultima non si accorge degli oggetti che le stanno intorno. Inoltre, la tartaruga ha una vita solitaria e svolge i propri lavori sempre da sola. Per fare in modo che l'animale in questione sia ancora più concreto e reale è possibile dotarlo di occhi, di collocarlo in un mondo popolato da altre tartarughe e dotarlo di un corpo vero, che possa essere toccato ed eventualmente smontato per osservarne e manipolarne il suo interno.
Quest'ultima esigenza ha portato allo sviluppo da parte del MIT (Massachusetts Institute of Technology) prima del Lego/Logo e dello *Logo, successivamente del Lego Mindstorm.

Nel Lego/Logo la tartaruga può registrare lo stato del mondo attorno a se attraverso sensori di vario tipo (di contatto, di luce, ecc...). A differenza della tartaruga geometrica che si sposta eseguendo un'istruzione o una serie di istruzioni che producono sempre lo stesso risultato, la tartaruga "cibernetica" ha un comportamento che non segue uno schema predefinito, ma emerge dall'interazione tra un programma e l'informazione proveniente dal mondo esterno.

Nello *Logo, estensione del Logo sviluppata da Resnick, è possibile gestire un gran numero di tartarughe contemporaneamente. Rispetto al Logo è possibile inoltre l'impiego di una gamma più ampia di sensori e lo sviluppo di veri e propri ambienti dinamici.

Nel Lego Mindstorm la tartaruga virtuale del Logo arriva ad acquisire un corpo realizzato con mattoncini di plastica, ruote meccaniche e componenti elettronici. L'idea di fondo è quella di fornire ai bambini degli strumenti aperti con i quali costruire una tartaruga dotata di ruote e sensori e scrivere un programma in stile Lego/Logo per consentirle di muoversi.
Il prodotto si presenta come una normale scatola di costruzioni contenente pezzi di tipo diverso: ingranaggi, mattoncini, ruote, ecc... Insieme a tali pezzi è presente un mattoncino programmabile che rappresenta un microcalcolatore autonomo. Questo piccolo calcolatore ha proprio le funzioni del cervello di un organismo artificiale e ad esso vengono collegati dei sensori attraverso dei cavi elettrici per consentirgli l'interazione con il mondo esterno.

3.4 La robotica come ambiente di apprendimento

La situazione della scuola italiana vede un interesse sempre più crescente da parte degli insegnanti per i kit che permettono la costruzione e l'impiego di piccoli robot all'interno dei programmi di insegnamento.
Nel corso degli ultimi anni sono state realizzate valide esperienze d'impiego a scuola della robotica, laddove chiaramente si è inteso proporre agli alunni un approccio fortemente costruttivista al sapere, offrendo interazione sia sul piano fisico e materiale, sia sul piano tecnologico e informatico. È proprio la natura di scienza di sintesi che rende la robotica un vero e proprio catalizzatore cognitivo, attorno al quale trovano rinforzo e maggior coinvolgimento tutte le altre attività didattiche.
L'attivazione di un laboratorio di robotica deve avere sempre come obiettivo ben definito la costruzione di competenze reali negli allievi. In questo senso ogni attività deve essere progettata per favorire un apprendimento significativo, favorendo il realizzarsi di un cambiamento del modo di agire del discente attraverso il cambiamento dei suoi modelli di pensiero.
Le Boterf sostiene che

"la competenza risiede nella mobilitazione delle risorse dell'individuo (conoscenze, abilità, atteggiamenti, ecc...) e non nelle risorse stesse, e si configura come un saper agire (o reagire) in una determinata situazione, in un determinato contesto, allo scopo di conseguire una performance, sulla quale altri soggetti (superiori o colleghi) dovranno esprimere un giudizio". Si capisce bene quanto sia didatticamente semplice favorire la crescita delle competenze degli alunni attorno ad un problema robotico.

All'interno di un laboratorio di robotica scatta dunque una dinamica socio-cognitiva che viene ben descritta dal modello di Pfeiffer e Jones, successivamente ripreso e arricchito dallo stesso Le Boterf. Pfeiffer e Jones si rifanno ad un modello di apprendimento attivo nel quale l'allievo svolge attività tratte da problemi concreti riferiti a contesti reali. Il modello prevede un processo di

apprendimento di tipo circolare come rappresentato in figura 3.1.

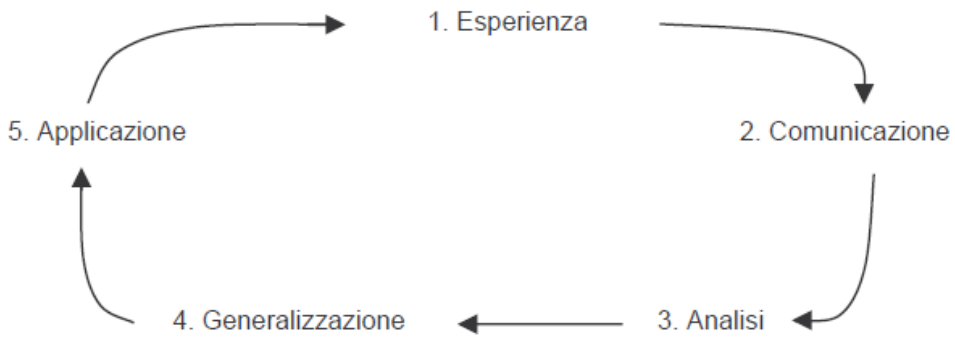

Figura 3.1: Il modello di Pfeiffer e Jones

Le cinque fasi che caratterizzano il modello sono:

- *Esperienza*: rappresenta il momento nel quale il discente viene coinvolto, singolarmente o in gruppo, in particolari attività. I compiti proposti devono essere tratti da attività reali e devono essere significativi per il soggetto che apprende, ovvero legati ad esperienze che lui stesso percepisce come reali e importanti per la propria preparazione.

- *Comunicazione*: rappresenta il momento che prevede la condivisione dell'esperienza con il gruppo. In questa fase emergono le discussioni e il confronto delle proprie idee con quelle degli altri elementi del gruppo.

- *Analisi*: rappresenta il momento in cui il soggetto riflette sulla propria esperienza e la valuta sulla base di criteri espliciti. È un momento personale di rielaborazione originale e creativa, nel quale emergono possibili linee di integrazione tra i modelli esaminati durante la fase di comunicazione.

- *Generalizzazione*: rappresenta il momento nel quale a partire dai risultati dell'analisi vengono elaborati nuovi modelli operativi, l'adozione dei quali porterà a modificare la struttura di pensiero coinvolta nel compito. La fase di generalizzazione rappresenta il momento cruciale dell'intero percorso di attività.

- *Applicazione*: rappresenta il momento nel quale l'allievo deve produrre un nuovo piano d'azione che testerà successivamente in una nuova fase di esperienza. In questo modo l'allievo ritorna sui propri passi per dimostrare che ciò che ha prodotto può essere fatto meglio.

Nella tabella sottostante vengono riportate le analogie tra il modello di Pfeiffer e Jones e il modello di Le Boterf.

Pfeiffer e Jones (1985)	**Le Boterf (2000)**
Esperienza	Esperienza vissuta
Comunicazione	Esplicitazione e narrazione dell'esperienza
Analisi	Concettualizzazione o modellazione
Generalizzazione	Decontestualizzazione degli schemi-modelli
Applicazione	Ritorno alla messa in pratica

Tabella 3.1: Confronto tra il modello di Pfeiffer e Jones e il modello di Le Boterf

Dato quindi un problema e quanto necessario per la sua soluzione viene dato il via all'esperienza, determinando un'esigenza di comunicazione sia all'interno del gruppo che tra gruppi diversi. In questo modo l'analisi del problema e la ricerca di una possibile soluzione portano alla generalizzazione del processo e all'applicazione di quest'ultimo sull'oggetto stesso.
Il robot rappresenta lo strumento concreto ed immediato in grado di far vivere agli allievi l'esperienza tangibile del processo di generalizzazione e applicazione sopra esposto. Con la sua struttura fisica, sensoriale e motoria, viene lanciato ad affrontare un determinato problema e attraverso una performance restituisce un indice di valutazione senza la necessità che, citando le parole di Le Boterf, *"altri soggetti (superiori o colleghi) debbano esprimere un giudizio"*.

Capitolo 4

Lego Mindstorm

Il soggetto principale attorno al quale ruota tutto il lavoro di tesi è il robot Lego Mindstorm NXT. All'apparenza potrebbe sembrare un semplice oggetto di intrattenimento, ma dietro la maschera ludica si svela uno strumento didattico essenziale per l'applicazione del costruttivismo Papertiano.

4.1 Cenni storici

Nel 1988 la collaborazione tra il gruppo Lego e il Massachusetts Institute of Technology (MIT) dà il via allo sviluppo di un "brick intelligente".
La prima versione del mattoncino viene rilasciata nel 1998 con il nome di *Robotics Invention System*, ma la vera e propria nascita del Lego Mindstorm avviene con l'uscita del robot RCX (*Robot Command Explorer*) (figura 4.1). Al suo interno è presente un microcontrollore Hitachi H8/3292 con otto registri ad uso generale e spazio di indirizzamento da 16 bit. L'RCX contiene una memoria on-chip di tipo ROM da 16 kB e una memoria RAM da 512 kB. Inoltre, è presente una memoria esterna per il firmware e i programmi utente della dimensione di 32 kB.
Oltre alla porta ad infrarossi, sono presenti tre porte d'ingresso per i sensori e tre porte d'uscita per i motori. L'RCX è dotato di display LCD attraverso cui è possibile ottenere informazioni circa lo stato della batteria, delle porte e del programma in esecuzione.

Nel 2006 viene rilasciata la nuova versione del prodotto (il kit Lego Mindstorm NXT) e nell'Agosto del 2009 viene lanciata sul mercato l'ultima versione: "il kit Lego Minstorm NXT 2.0" (figura 4.2).

Figura 4.1: Il Lego Mindstorm RCX.

Figura 4.2: Il Lego Mindstorm NXT.

4.2 Lego Mindstorm NXT

4.2.1 Introduzione all'Hardware

Il mattoncino intelligente è dotato di un microprocessore Atmel AT91SAM7S256 (classe ARM7) a 32 bit con 256 kB di memoria flash non volatile e 64 kB di RAM. Include, inoltre, un coprocessore Atmel ATmega48 (classe AVR) a 8 bit con 4 kB di memoria flash e 512 byte di RAM.
L'NXT é dotato di display con risoluzione di 64x100 pixel e supporta due tipologie di comunicazione:

- Wireless: tramite protocollo Bluetooth ad una velocità di 460.8 kBit/s;
- Wired: tramite cavo USB ad una velocità di 12 Mbit/s.

Il mattoncino possiede quattro porte di input e tre porte di output (figura 4.3). Le porte di input acquisiscono i valori misurati dai sensori e sono provviste sia di interfaccia digitale che analogica per il supporto di tutti i tipi di sensore. Le porte di output sono invece dotate di interfaccia digitale.

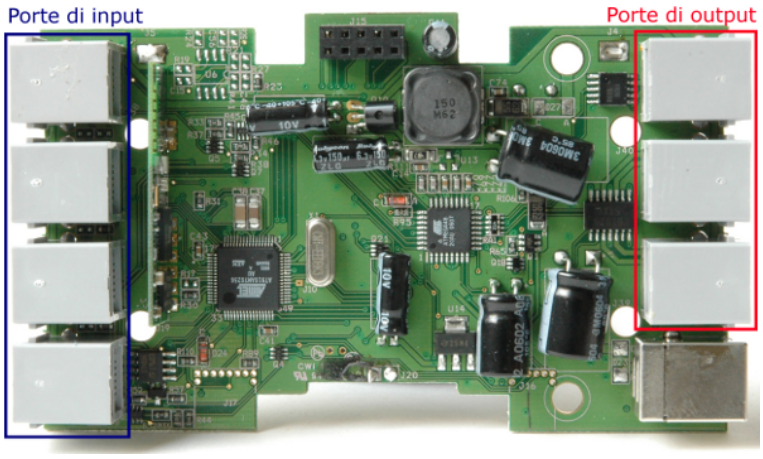

Figura 4.3: La scheda madre dell' NXT.

4.2.2 Motori

Il Lego Mindstorm NXT è dotato di tre servomotori (figura 4.4) che funzionano in corrente continua a 9V. Osservando i motori esternamente si notano: una parte cilindrica contenente il motore vero e proprio, ed una parte allungata contenente la scatola degli ingranaggi di riduzione. Ogni motore contiene un encoder incrementale con risoluzione pari ad 1°. In figura 4.5 è possibile vedere la configurazione interna del motore.

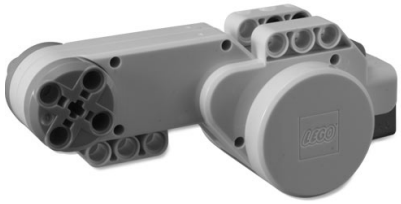

Figura 4.4: Il servomotore.

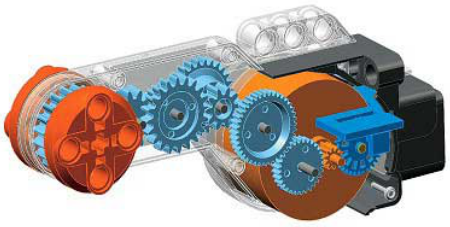

Figura 4.5: L'interno del servomotore.

4.2.3 Sensori

I sensori che vengono forniti con l'NXT appartengono a tre categorie:

- *Analogici attivi*: per effettuare la misura di un fenomeno fisico emettono un'interrogazione e ne misurano la risposta.

- *Analogici passivi*: non interrogano l'ambiente circostante per l'acquisizione dei dati, ma traducono direttamente il fenomeno fisico in segnale elettrico.

- *Digitali*: utilizzano l'interfaccia I^2C.

Sensore di contatto

È un sensore di tipo passivo che viene collegato ad una della quattro porte di input. Il segnale che fornisce è di tipo ON/OFF (0 se rilasciato, 1 se premuto). Inoltre, è possibile settare il sensore per leggere il valore grezzo (da 0 a 1023).

Figura 4.6: Il sensore di contatto.

Sensore di luce

È un sensore di tipo attivo che viene collegato ad una delle quattro porte di input. Il segnale che fornisce è di tipo analogico e proporzionale alla luminosità rilevata. Può funzionare in due diverse modalità:

- *Reflected*: il sensore utilizza un emettitore di luce con il quale illumina gli oggetti e misura la quantità di luce riflessa. Ovviamente le superfici più chiare riflettono una quantità di luce maggiore rispetto alle superfici scure.

- *Ambient*: l'emettitore di luce viene disabilitato e vengono misurati i diversi livelli di luce dell'ambiente circostante.

Figura 4.7: Il sensore di luce.

Sensore di suono

È un sensore di tipo attivo che viene collegato ad una delle quattro porte di input. Il segnale fornito è di tipo analogico e proporzionale al suono registrato. Il sensore può misurare livelli di suono fino a circa 90 dB. Sono presenti due modalità di funzionamento:

- *Normal*: misura anche suoni non udibili dall'orecchio umano;
- *Adjusted*: rileva solo suoni udibili dall'orecchio umano.

Figura 4.8: Il sensore di suono.

Sensore ad ultrasuoni

È un sensore di tipo attivo che viene collegato ad una delle quattro porte di input. Il segnale fornito è di tipo analogico e proporzionale alla distanza dell'oggetto rilevato. Misura distanze in centimetri (o pollici) da 0 a 255 cm con una precisione di circa $\pm 3cm$. La stima della distanza viene effettuata misurando il tempo che intercorre tra l'invio di un ultrasuono, il rimbalzo contro un ostacolo e il ritorno indietro al sensore. La precisione delle misurazioni è

tanto più elevata quanto più l'ostacolo è grande. Il sensore può addirittura non funzionare correttamente con oggetti piccoli, sottili e con superfici non piane.

Figura 4.9: Il sensore ad ultrasuoni.

Sensore di prossimità

Il sensore di prossimità è in grado di rilevare con estrema precisione la presenza e la distanza di un oggetto posto nelle vicinanze. Il sensore utilizza degli impulsi di luce che, eliminando i segnali di disturbo esterni, consentono di ottenere una precisione inferiore ai 2 millimetri, a seconda della forma e della dimensione dell'oggetto da rilevare.

Figura 4.10: Il sensore di prossimità.

Sensore di colore

È un sensore digitale che rileva la colorazione degli oggetti che illumina utilizzando un apposito emettitore di luce. Analizzando le componenti di colore della luce riflessa si determina il *Color Number*, ovvero un numero che identifica il colore all'interno di una gamma di 17 possibili tonalità.

Figura 4.11: Il sensore di colore.

Sensore a infrarossi

Il sensore a infrarossi consente rilevare la radiazione infrarossa emessa da opportune sorgenti e può operare in due modi:

- rilevando segnali a infrarossi modulati
- rilevando segnali a infrarossi non modulati.

Figura 4.12: Il sensore a infrarossi.

Sensore bussola

È un sensore digitale che contiene una bussola in grado di misurare il campo magnetico terrestre e determinare la posizione in gradi rispetto al nord.

Figura 4.13: Il sensore bussola.

Sensore giroscopico

È un sensore analogico che serve per rilevare la velocità angolare lungo un determinato asse. La velocità massima angolare registrabile è pari a 360 gradi/secondo.

Figura 4.14: Il sensore giroscopico.

Sensore di accelerazione

È un sensore digitale che comunica con il brick per mezzo dell'interfaccia I^2C e rileva le accelerazioni lungo i tre assi x, y e z. Tutte le misure di accelerazione sono comprese nell'intervallo che va da -2g a +2g (con g accelerazione gravitazionale terrestre).

Figura 4.15: Il sensore di accelerazione.

4.2.4 I^2C

I^2C (Inter Integrated Circuit) è un sistema di comunicazione seriale utilizzato tra circuiti integrati. Nel 1992 è stata rilasciata la prima versione del protocollo che nel corso del tempo ha subito svariati aggiornamenti. Il bus I^2C trasferisce la informazioni ai dispositivi ad esso collegati attraverso due linee dati: SDA (Serial Data Line) per i dati e SCL (Serial Clock Line) per il clock.
Ogni dispositivo viene identificato da un indirizzo e deve essere impostato o come master o come slave. È possibile che nel bus vi siano più dispositivi master, ma in questo caso solo uno alla volta può controllare il bus.
Il master è il dispositivo che inizia e termina la comunicazione, lo slave può solo ricevere e trasmettere informazioni su richiesta del master. L'indirizzo che identifica ogni periferica inserita nel bus I^2C viene fissato dal produttore in fase di fabbricazione. La Lego ha messo a disposizione delle aziende produttrici di sensori un apposito set di indirizzi in modo tale che non vengano creati dispositivi con lo stesso indirizzo, evitando in questo modo potenziali duplicazioni. Nelle versioni standard del protocollo ogni indirizzo è costituito da 7 bit (10 bit nelle versioni estese). Con 7 bit si possono ipoteticamente indirizzare 128 periferiche (1024 con 10 bit). Una comunicazione sul bus I^2C avviene secondo i seguenti passi:

- Il master verifica che linee SDA e SCL non siano impegnate in un'altra comunicazione. Se il bus è libero allora il master invia alle altre periferiche il messaggio di "bus occupato". In questo modo le altre periferiche si mettono in ascolto per capire con chi vuole comunicare il master.

- il master invia il segnale di sincronizzazione sulla linea SCL.
- il master invia l'indirizzo della periferica con cui vuole parlare.
- il master indica se la comunicazione che intende effettuare è di lettura o scrittura.
- il master attende la risposta da parte della periferica contattata. Nel caso di mancata risposta, libera il bus.
- in seguito all'avvenuto riconoscimento della periferica il master inizia lo scambio dati inviando pacchetti di 8 bit. Per ogni pacchetto inviato viene emesso un segnale che indica l'avvenuta ricezione.
- una volta conclusa la comunicazione, il master libera il bus inviando un segnale di stop.

Per la comunicazione ogni canale è dotato di un buffer di input e di un buffer di output, entrambi della dimensione di 16 byte.

4.3 Bluetooth

Il brick dell'NXT supporta la comunicazione Bluetooth e può connettersi contemporaneamente ad altri tre dispositivi; questi dispositivi possono essere di svariata natura: altri NXT, calcolatori, telefoni cellulari, ecc... Tale funzionalità viene implementata utilizzando il *Serial Port Profile*, ovvero l'equivalente wireless di un cavo seriale. È quindi possibile trasferire programmi e suoni tra i diversi brick e utilizzare la comunicazione Bluetooth per inviare e ricevere informazioni. Per ridurre i consumi che derivano dall'utilizzo di tecnologia Bluetooth, i brick sono dotati di un dispositivo di classe II, che consente di trasmettere fino ad una distanza massima di circa 10 metri.
La modalità di trasmissione tra brick è di tipo master/slave. All'interno di una rete si identifica un dispositivo come master (coordinatore) e agli altri viene assegnato il ruolo di slave (serventi). Non è possibile attribuire contemporaneamente ad un dispositivo la funzione sia di master che di slave. Bisogna

inoltre tenere presente che le unità slave non possono comunicare tra loro direttamente, ma devono sempre passare per l'unità master.
Quando un programma in esecuzione sul brick invia o riceve messaggi, esso specifica una mailbox (numerata da 0 a 9), ovvero un contenitore di messaggi. Se un dispositivo master decide di inviare un messaggio, deve specificare il dispositivo slave al quale desidera inviarlo e la relativa mailbox all'interno della quale depositarlo. Come già menzionato, la comunicazione Bluetooth utilizza un protocollo applicativo detto Serial Port Profile che si appoggia direttamente sul protocollo RFCOMM.
RFCOMM è un protocollo di trasporto di basso livello affidabile, ma può comunque verificarsi una perdita di messaggi dal momento che all'interno di ogni mailbox é possibile depositare fino ad un massimo di cinque messaggi. Infatti, l'operazione di inserimento di un messaggio all'interno di una mailbox piena, determina l'eliminazione del messaggio più vecchio. Inoltre, se il programma in esecuzione sullo slave non effettua un polling frequente dei messaggi ricevuti in ingresso, si rischia che il master mandi in overflow la mailbox e quindi i vecchi messaggi presenti vengano eliminati. La possibile perdita di messaggi avviene anche quando sullo slave non è in esecuzione alcun programma. In questo caso il firmware dello slave ignora i messaggi, che quindi vengono persi.

4.3.1 Il chip BlueCore

L'NXT contiene al suo interno un chip CSR modello BlueCore 4 e una memoria flash esterna da 8 Mbit. Il chip contiene un processore a 16 bit che esegue la versione 3.2 dello stack Bluetooth chiamato *Bluelab*. Il firmware contenuto all'interno del BlueCore integra una macchina virtuale che consente all'utente di controllare ed eseguire piccole applicazioni. Connessa alla macchina virtuale è presente un interprete comandi in grado di decodificare e rispondere alle richieste ricevute dal processore ARM7 attraverso un'apposita interfaccia: UART (Universal Asynchronous Receiver Transmitter). In figura 4.16 viene mostrata l'interfaccia hardware tra il processore ARM7 e il chip BlueCore. L'interfaccia SPI (Serial Peripheral Interface) invece permette l'aggiornamento del chip e

non è attiva durante le normali operazioni dell'NXT. Per reinizializzare correttamente il chip BlueCore e disabilitare la funzionalità di Bluetooth viene utilizzato il pin di Reset. Il pin BC4-CMD indica il tipo di dato da inviare al chip, mentre il pin ARM7-CMD indica il tipo di dato da inviare all'ARM7.
L'interfaccia UART viene invece utilizzata sia per la comunicazione dati che per la comunicazione comandi tra il chip BlueCore e il processore ARM7. In particolare, l'NXT include due modalità di comunicazione:

- Bluetooth versione 2.0 con Serial Port Profile (SPP);
- USB versione 2.0.

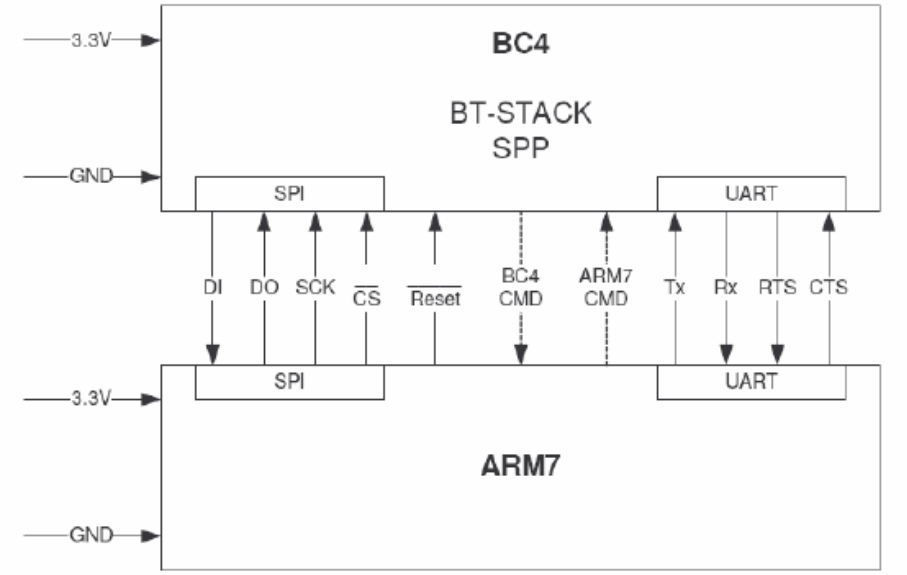

Figura 4.16: Interfaccia hardware presente tra l' ARM7 e il chip BlueCore.

Nella figura seguente (4.17) vengono riportati i principali livelli dello stack tra il software proprietario della Lego e il robot Lego Mindstorm NXT.

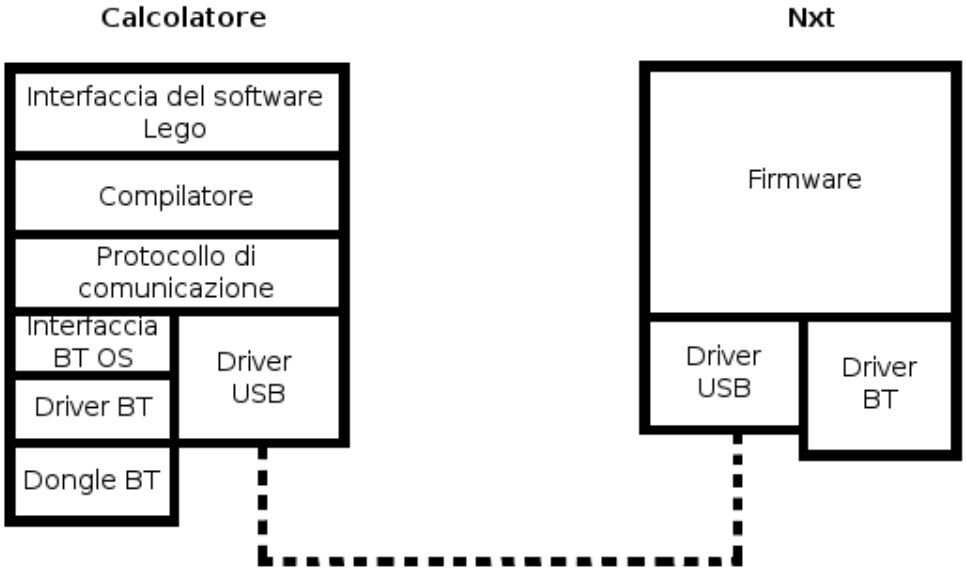

Figura 4.17: Comunicazione tra calcolatore ed NXT.

4.3.2 Comandi diretti

Il protocollo di comunicazione del Lego Mindstorm NXT rende possibile il controllo del brick mediante dispositivi esterni che utilizzino il Serial Port Profile. Lo scopo principale è quello di mettere a disposizione un'interfaccia che consenta di fruire delle funzionalità del brick senza dover necessariamente eseguire programmi specifici sull'NXT. In figura viene mostrata l'architettura generale del protocollo.

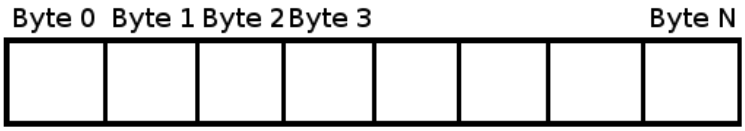

Figura 4.18: Architettura generale del protocollo.

- Byte 0: identifica il tipo di comando, in particolare:
  - 0x00: indica un comando diretto (richiede una risposta);

  - 0x01: indica un comando di sistema (richiede una risposta);
  - 0x02: indica un comando di risposta;
  - 0x80: indica un comando diretto (non richiede una risposta);
  - 0x81: indica un comando di sistema (non richiede una risposta);

- Byte 1-N: identificano il comando stesso o una risposta a seconda del tipo di comando.

La lunghezza di un comando diretto è limitata ad un massimo di 64 byte. Tutti i messaggi Bluetooth necessitano di 2 byte aggiuntivi da apporre davanti al messaggio e servono ad indicare il numero complessivo di byte contenuti nel messaggio stesso. La lunghezza dei pacchetti viene sempre contata non tenendo conto di questi primi due byte. Nella figura seguente viene mostrata la struttura di un messaggio Bluetooth.

Figura 4.19: Struttura di un messaggio Bluetooth.

Di seguito si riportano i principali comandi diretti e la loro struttura.

STARTPROGRAM

Consente di eseguire un programma residente sul brick.

Byte	Contenuto
Byte 0	0x00 oppure 0x80
Byte 1	0x00
Byte 2-21	il nome del programma in formato ASCIIZ, più il terminatore di stringa.

Tabella 4.1: Comando StartProgram.

Se viene richiesto un riscontro allora il pacchetto di risposta ha la seguente struttura:

Byte	Contenuto
Byte 0	0x02
Byte 1	0x00
Byte 2	il byte di stato per segnalare l'eventuale presenza di errori

Tabella 4.2: Risposta a StartProgram.

STOPPROGRAM

Consente interrompere l'esecuzione di un programma attualmente in esecuzione sul brick.

Byte	Contenuto
Byte 0	0x00 oppure 0x80
Byte 1	0x01

Tabella 4.3: Comando StopProgram.

Se viene richiesto un riscontro allora il pacchetto di risposta ha la seguente struttura:

Byte	Contenuto
Byte 0	0x02
Byte 1	0x01
Continua nella prossima pagina	

Byte	Contenuto
Byte 2	il byte di stato per segnalare l'eventuale presenza di errori

Tabella 4.4: Risposta a StopProgram.

PLAYSOUNDFILE

Consente di riprodurre un file audio (formato .rso) residente sul brick.

Byte	Contenuto
Byte 0	0x00 oppure 0x80
Byte 1	0x02
Byte 2	il valore true (impone l'esecuzione in loop del suono) oppure il valore false (impone l'esecuzione singola del suono)
Byte 3-22	il nome del file audio in formato ASCIIZ, più il terminatore di stringa

Tabella 4.5: Comando PlaySoundFile.

Se viene richiesto un riscontro allora il pacchetto di risposta ha la seguente struttura:

Byte	Contenuto
Byte 0	0x02
Byte 1	0x02
Byte 2	il byte di stato per segnalare l'eventuale presenza di errori

Tabella 4.6: Risposta a PlaySoundFile.

PLAYTONE

Consente di riprodurre un tono specificandone frequenza e durata.

Byte	Contenuto
Byte 0	0x00 oppure 0x80
Byte 1	0x03
Byte 2-3	la frequenza del tono in Hertz (nel range da 200 a 14000 Hz)
Byte 4-5	la durata del tono in millisecondi

Tabella 4.7: Comando PlayTone.

Se viene richiesto un riscontro allora il pacchetto di risposta ha la seguente struttura:

Byte	Contenuto
Byte 0	0x02
Byte 1	0x03
Byte 2	il byte di stato per segnalare l'eventuale presenza di errori

Tabella 4.8: Risposta a PlayTone.

SETOUTPUTSTATE

Consente di controllare i motori dell' NXT.

Byte	Contenuto
Byte 0	0x00 oppure 0x80
Byte 1	0x04
Byte 2	la porta alla quale è collegato il motore (0 indica la porta A, 1 indica la porta B e 2 indica la porta C)
Byte 3	il valore di potenza che si desidera fornire al motore (nel range da -100 a 100)
Byte 4	il modo di funzionamento
Byte 5	il modo di regolazione
Byte 6	il verso di rotazione dei motori (nel range da -100 a 100)
Byte 7	lo stato del motore
Byte 8-12	il numero massimo di giri che possono essere effettuati dal motore (se impostato a 0, obbliga il motore a girare all' infinito)

Tabella 4.9: Comando SetOutputState.

Modo	Descrizione
0x01	attiva lo specifico motore
0x02	arresta lo specifico motore
0x04	attiva la regolazione per lo specifico motore

Tabella 4.10: Modi di funzionamento

Modo	Descrizione
0x00	nessuna regolazione viene attivata
Continua nella prossima pagina	

Modo	Descrizione
0x01	viene attivato il controllo di potenza sulla specifica porta di output
0x02	viene attivata la sincronizzazione delle porte di output

Tabella 4.11: Modi di regolazione

Stato	Descrizione
0x00	il motore viene disattivato
0x10	viene aumentata rapidamente la velocità del motore
0x20	il motore viene attivato
0x40	viene decrementata rapidamente la velocità del motore

Tabella 4.12: Stati del motore

Se viene richiesto un riscontro allora il pacchetto di risposta ha la seguente struttura:

Byte	Contenuto
Byte 0	0x02
Byte 1	0x04
Byte 2	il byte di stato per segnalare l'eventuale presenza di errori

Tabella 4.13: Risposta a SetOutputState.

SETINPUTMODE

Consente di impostare i parametri di funzionamento di un sensore collegato alla porta selezionata.

Byte	Contenuto
Byte 0	0x00 oppure 0x80
Byte 1	0x05
Byte 2	la porta alla quale viene connesso il sensore (range da 0 a 3)
Byte 3	il tipo di sensore
Byte 4	il modo di regolazione del sensore

Tabella 4.14: Comando SetInputMode.

Sensore	Descrizione
0x00	nessun sensore configurato
0x01	sensore di contatto
0x02	sensore di temperatura
0x03	vecchio sensore di luce dell'RCX
0x04	vecchio sensore di rotazione dell'RCX
0x05	sensore di luce (LED acceso)
0x06	sensore di luce (LED spento)
0x07	sensore di suono (misura in dB)
0x08	sensore di suono (misura in dBA)
0x09	sensore non utilizzato
0x0A	sensore digitale passivo
0x0B	sensore digitale attivo (per esempio il sensore a ultrasuoni)
0x0C	sensore non compreso tra quelli già citati

Tabella 4.15: Tipologie di sensore.

Modo	Descrizione
0x00	ritorna il valore grezzo (RAW) letto dal sensore
0x20	ritorna il valore 1 (Vero) se il valore grezzo misurato è inferiore al 45% del valore massimo misurabile; ritorna il valore 0 (Falso) se il valore grezzo misurato è superiore del 55% del valore massimo misurabile
0x40	ritorna il numero di transizioni dal valore Vero al valore Falso
0x60	ritorna il numero di transizioni dal valore Falso al valore Vero e poi al valore Falso
0x80	ritorna il valore della misurazione del sensore in percentuale rispetto ad una scala
0xA0	ritorna il valore di temperatura in gradi Celsius
0xC0	ritorna il valore di temperatura in gradi Fahrenheit
0xE0	ritorna il valore relativo al conteggio dei tick di un sensore di rotazione

Tabella 4.16: Modalità di regolazione del sensore.

GETOUTPUTSTATE

Consente di richiedere i parametri con i quali è stata configurata una determinata porta.

Byte	Contenuto
Byte 0	0x00 oppure 0x80
Continua nella prossima pagina	

Byte	Contenuto
Byte 1	0x06
Byte 2	specifica la porta di output (range da 0 a 2)

Tabella 4.17: Comando GetOutputState.

Se viene richiesto un riscontro allora il pacchetto di risposta ha la seguente struttura:

Byte	Contenuto
Byte 0	0x02
Byte 1	0x06
Byte 2	il byte di stato per segnalare l'eventuale presenza di errori.
Byte 3	la porta di output
Byte 4	la potenza configurata sulla porta (valore compreso tra -100 e 100)
Byte 5	il modo cui cui è stata configurata la specifica porta
Byte 6	il modo di regolazione attivo
Byte 7	il verso di rotazione impostato
Byte 8	lo stato del motore
Byte 9-12	il numero massimo di giri imposti al motore
Byte 13-16	il numero di giri effettuati dal motore dopo l'ultimo reset del contatore
Byte 17-20	la posizione (in numero di giri) relativa all'ultimo movimento del motore
Byte 21-24	la posizione (in numero di giri) dall'ultimo reset del sensore di rotazione del motore

Tabella 4.18: Risposta a GetOutputState.

GETINPUTVALUES

Consente di leggere i dati provenienti da una specifica porta di input.

Byte	Contenuto
Byte 0	0x00 oppure 0x80
Byte 1	0x07
Byte 2	specifica la porta di input (range da 0 a 3)

Tabella 4.19: Comando GetInputValues.

Se viene richiesto un riscontro allora il pacchetto di risposta ha la seguente struttura:

Byte	Contenuto
Byte 0	0x02
Byte 1	0x07
Byte 2	il byte di stato per segnalare l'eventuale presenza di errori
Byte 3	la porta sulla quale vengono letti i dati (range da 0 a 3)
Byte 4	specifica se i dati ottenuti sono o meno validi
Byte 5	specifica se è presente una calibrazione del sensore
Byte 6	il tipo di sensore
Byte 7	la modalità di funzionamento del sensore
Byte 8-9	il valore grezzo (RAW) proveniente dal sensore
Byte 10-11	il valore normalizzato proveniente dal sensore
Byte 12-13	il valore scalato in base al modo con il quale è stato configurato il sensore
Byte 14-15	il valore scalato in base al tipo di calibrazione

Tabella 4.20: Risposta a GetInputValues.

RESETINPUTSCALEDVALUE

Consente di eseguire il reset di una porta di input e ricevere come risposta un byte che segnala l'eventuale presenza di errori.

Byte	Contenuto
Byte 0	0x00 oppure 0x80
Byte 1	0x08
Byte 2	specifica la porta di input (range da 0 a 3)

Tabella 4.21: Comando ResetInputScaledValue.

Se viene richiesto un riscontro allora il pacchetto di risposta ha la seguente struttura:

Byte	Contenuto
Byte 0	0x02
Byte 1	0x08
Byte 2	il byte di stato per segnalare l'eventuale presenza di errori

Tabella 4.22: Risposta a ResetInputScaledValue.

RESETMOTORPOSITION

Consente di effettuare il reset di una specifica porta di output e riceve come risposta un byte che segnala l'eventuale presenza di errori.

Byte	Contenuto
Byte 0	0x00 oppure 0x80
Byte 1	0x0A
Byte 2	specifica la porta di output (range da 0 a 2)
Byte 3	contiene un valore booleano che indica la necessità o meno di resettare la posizione relativa all'ultimo movimento

Tabella 4.23: Comando ResetMotorPosition.

Se viene richiesto un riscontro allora il pacchetto di risposta ha la seguente struttura:

Byte	Contenuto
Byte 0	0x02
Byte 1	0x0A
Byte 2	il byte di stato per segnalare l'eventuale presenza di errori

Tabella 4.24: Risposta a ResetMotorPosition.

GETBATTERYLEVEL

Consente di interrogare l'NXT per rilevare informazioni circa lo stato della batteria. Viene fornita la misura della tensione in millivolt oppure una segnalazione d'errore.

Byte	Contenuto
Byte 0	0x00 oppure 0x80
Byte 1	0x0B

Tabella 4.25: Comando GetBatteryLevel.

Se viene richiesto un riscontro allora il pacchetto di risposta ha la seguente struttura:

Byte	Contenuto
Byte 0	0x02
Byte 1	0x0B
Byte 2	il byte di stato per segnalare l'eventuale presenza di errori
Byte 3	la misura della tensione in millivolt

Tabella 4.26: Risposta a GetBatteryLevel.

STOPSOUNDPLAYBACK

Consente di arrestare l'esecuzione di un file audio.

Byte	Contenuto
Byte 0	0x00 oppure 0x80
Byte 1	0x0C

Tabella 4.27: Comando StopSoundPlayBack.

Se viene richiesto un riscontro allora il pacchetto di risposta ha la seguente struttura:

Byte	Contenuto
Byte 0	0x02
Byte 1	0x0B
Continua nella prossima pagina	

Byte	Contenuto
Byte 2	il byte di stato per segnalare l'eventuale presenza di errori
Byte 3-6	un intervallo temporale in millisecondi nel quale il robot viene mandato in modalità sleep

Tabella 4.28: Risposta a StopSoundPlayBack.

LSGETSTATUS

Consente di richiedere lo stato dei dati dell'I^2C. Come risposta viene ricevuto il numero di byte disponibili ed un byte che segnala l'eventuale presenza di errori.

Byte	Contenuto
Byte 0	0x00 oppure 0x80
Byte 1	0x0E
Byte 2	specifica la porta di input (range da 0 a 3)

Tabella 4.29: Comando LSGetStatus.

Se viene richiesto un riscontro allora il pacchetto di risposta ha la seguente struttura:

Byte	Contenuto
Byte 0	0x02
Byte 1	0x0E
Byte 2	il byte di stato per segnalare l'eventuale presenza di errori
Byte 3	il numero di byte pronti per essere letti

Tabella 4.30: Risposta a LSGetStatus.

LSWRITE

Consente di richiedere i dati provenienti dalle porte configurate in modalità "Low Speed I^2C".

Byte	Contenuto
Byte 0	0x00 oppure 0x80
Byte 1	0x0F
Byte 2	specifica la porta di input (range da 0 a 3)
Byte 3	la lunghezza dei dati da trasmettere (in byte)
Byte 4	la lunghezza dei dati da ricevere (in byte)
Byte 5-N	i dati da trasmettere. N rappresenta la lunghezza dei dati da trasmettere + 4

Tabella 4.31: Comando LSWrite.

Se viene richiesto un riscontro allora il pacchetto di risposta ha la seguente struttura:

Byte	Contenuto
Byte 0	0x02
Byte 1	0x0F
Byte 2	il byte di stato per segnalare l'eventuale presenza di errori

Tabella 4.32: Risposta a LSWrite.

LSREAD

Consente di richiedere la lettura dei dati da una porta di input I^2C e ricevere come risposta i dati provenienti dal bus.

Byte	Contenuto
Byte 0	0x00 oppure 0x80
Byte 1	0x10
Byte 2	specifica la porta di input (range da 0 a 3)

Tabella 4.33: Comando LSRead.

Se viene richiesto un riscontro allora il pacchetto di risposta ha la seguente struttura:

Byte	Contenuto
Byte 0	0x02
Byte 1	0x10
Byte 2	il byte di stato per segnalare l'eventuale presenza di errori
Byte 3	il numero di byte letti
Byte 4-19	il numero di dati ricevuti

Tabella 4.34: Risposta a LSRead.

GETCURRENTPROGRAMNAME

Consente di conoscere il programma attualmente in esecuzione sul brick.

Byte	Contenuto
Byte 0	0x00 oppure 0x80
Byte 1	0x11

Tabella 4.35: Comando LSRead.

Se viene richiesto un riscontro allora il pacchetto di risposta ha la seguente struttura:

Byte	Contenuto
Byte 0	0x02
Byte 1	0x11
Byte 2	il byte di stato per segnalare l'eventuale presenza di errori
Byte 3	il numero di byte letti
Byte 3-22	il nome del programma attualmente in esecuzione

Tabella 4.36: Risposta a LSRead.

MESSAGEWRITE

Consente di scrivere un messaggio all'interno di una mailbox.

Byte	Contenuto
Byte 0	0x00 oppure 0x80
Byte 1	0x09
Byte 2	il numero della mailbox nella quale scrivere il messaggio (range da 0 a 9)
Byte 3	la dimensione del messaggio da scrivere
Continua nella prossima pagina	

Byte	Contenuto
Byte 4-N	il contenuto del messaggio. N rappresenta la dimensione del messaggio + 3

Tabella 4.37: Comando MessageWrite.

Se viene richiesto un riscontro allora il pacchetto di risposta ha la seguente struttura:

Byte	Contenuto
Byte 0	0x02
Byte 1	0x09
Byte 2	il byte di stato per segnalare l'eventuale presenza di errori

Tabella 4.38: Risposta a MessageWrite.

MESSAGEREAD

Consente di scrivere un messaggio all'interno di una mailbox.

Byte	Contenuto
Byte 0	0x00 oppure 0x80
Byte 1	0x13
Byte 2	il numero della mailbox remota dalla quale leggere il messaggio (range da 0 a 9)
Byte 3	il numero della mailbox locale dalla quale leggere il messaggio (range da 0 a 9)
Byte 4	specifica se una volta letto il messaggio quest'ultimo debba essere cancellato

Tabella 4.39: Comando MessageRead.

Se viene richiesto un riscontro allora il pacchetto di risposta ha la seguente struttura:

Byte	Contenuto
Byte 0	0x02
Byte 1	0x13
Byte 2	il byte di stato per segnalare l'eventuale presenza di errori
Byte 3	il numero della mailbox locale (range da 0 a 9)
Byte 4	la dimensione del messaggio da leggere
Byte 5-63	il contenuto del messaggio

Tabella 4.40: Risposta a MessageRead.

Va osservato che il byte di stato serve per ricevere informazioni riguardanti la presenza o meno di errori. Se tale byte contiene il valore zero allora non si è verificato alcun tipo di errore. Di seguito, viene riportato l'elenco dei possibili errori che possono essere incontrati utilizzando i comandi diretti.

Byte	Contenuto
0x20	attesa di comunicazione in corso
0x40	la coda della specifica mailbox è vuota
0xBD	impossibile soddisfare la richiesta
0xBE	comando sconosciuto
0xBF	pacchetto danneggiato
0xC0	i dati contengono valori al di fuori del range specificato
0xDD	errore sul bus di comunicazione

Continua nella prossima pagina

Byte	Contenuto
0xDE	memoria libera esaurita nel buffer di comunicazione
0xDF	la connessione specificata non è valida
0xE0	la connessione specificata non è configurata, oppure risulta occupata
0xEC	nessun programma attivo
0xED	dimensione specificata non valida
0xEF	tentativo di accesso ad un campo di una struttura non valido
0xF0	specificato un input/output non valido
0xFB	memoria disponibile insufficiente
0xFF	argomenti errati

Tabella 4.41: Tipologie di errore.

Capitolo 5

Programmazione dell'NXT

La programmazione dell'NXT contempla una grande varietà di linguaggi e di ambienti di programmazione. La rassegna che segue vuole porre l'accento in particolare su alcuni linguaggi, nell'ottica di fornire degli strumenti validi in grado di semplificare e rendere efficiente la comunicazione uomo-macchina.

5.1 Quale linguaggio di programmazione

Il robot è un'oggetto la cui comprensione va ben al di là della sua natura tecnologica. A differenza dei calcolatori, apparsi nell'immaginario comune solo nell'era moderna, i robot hanno attraversato tutte le letterature, fin dall'antichità. Basti pensare che la Bibbia stessa, con la figura di Golem, è considerata la radice della parola robot. L'uomo "programmava" tali creature scrivendo sulla loro fronte la parola *verità*, mentre per riportarle allo stato di materia inanimata era sufficiente che scrivesse la parola *morte*. L'insieme delle parole crea il linguaggio, elemento fondamentale per realizzare l'interazione uomo-macchina.

All'interno di un laboratorio di robotica, in particolare, si sviluppano due forme di interazione:

- **Docente - Discente**

- **Discente - Robot**

L'interazione tra discente e robot prevede che l'apparato robotico venga istruito a comportarsi in un determinato modo attraverso un opportuno linguaggio robotico. In questo senso il robot viene "educato" a svolgere specifici compiti. Ed è proprio il linguaggio l'elemento chiave all'interno dell'attività didattica. Nella realizzazione di una dinamica cognitiva efficiente, al centro della quale porre il discente, lo sforzo maggiore consiste nell'approssimare il linguaggio che caratterizza la relazione Docente-Robot al linguaggio che caratterizza la relazione Docente-Discente.

Per poter introdurre la Robotica come strumento di apprendimento costruttivista fin dalla scuola primaria, è necessario che i discenti acquisiscano piena consapevolezza di ciò che fanno, ovvero che incamerino quella *deep competence* citata da Seymour Papert nel suo *Mindstorms*.

Questa convinzione ha spinto alcuni a ritenere che i linguaggi testuali di programmazione dei robot siano da preferire ai linguaggi iconici. Le esperienze di microrobotica nella scuola hanno mostrato che un linguaggio testuale semplice, ma con regole precise di sintassi, obbliga in qualche modo alla precisione e ad un feedback immediato.

Giovanni Marcianò, in particolare, sostiene che i linguaggi iconici non realizzino un'efficace dialogo uomo-macchina in quanto troppo orientati al robot. A sostegno della sua tesi, mette in campo primitive diverse da quelle presenti nei più comuni linguaggi per la programmazione dei Mindstorm NXT. Queste ultime, in particolare, hanno la caratteristica di essere molto vicine al linguaggio Logo.

Molti studiosi, tuttavia, sostengono la validità dei linguaggi iconici confermando numerose esperienze sul campo, grazie ai quali viene facilitata la comprensione di molti concetti astratti legati al mondo della programmazione.

Concetti come l'"if" o come il "loop" sono spesso difficili da comprendere per via del linguaggio, non a causa del concetto stesso. L'uso dei linguaggi iconici quindi favorisce la comprensione e pone le basi per il successivo utilizzo di linguaggi espressivi più potenti.

Di seguito viene presentata una breve rassegna dei principali linguaggi e ambienti di programmazione.

5.2 NXT-G

NXT-G è un ambiente di programmazione grafico sviluppato dalla National Instruments per Lego. L'applicativo è dotato di un'interfaccia molto semplice ed intuitiva, appositamente pensata per l'utilizzo all'interno di percorsi scolastici e di didattica. La creazione di un programma avviene mediante l'impiego di appositi blocchi, suddivisi in opportune categorie. Ogni blocco è configurabile a piacere attraverso la modifica di svariati parametri ed è caratterizzato da un'immagine che ne identifica la funzione svolta.
Esperienze dirette di utilizzo del software hanno evidenziato una serie di problematiche, non del tutto trascurabili.
Innanzitutto, la creazione di diagrammi di una certa complessità determina la presenza di un numero troppo elevato di blocchi e di interconnessioni che limitano la chiarezza del programma e la sua comprensione. Inoltre, i file prodotti con NXT-G hanno dimensioni molto più grandi rispetto a file realizzati con altri applicativi. Quest'ultimo è un aspetto che non deve assolutamente essere trascurato vista l'esigua disponibilità di memoria di cui è dotato il brick.
Si riportano di seguito le principali sezioni di NXT-G la cui interfaccia utente viene riportata in figura 5.1.

1. **Tutorial**: contiene istruzioni per la costruzione e programmazione dell'NXT;

2. **Il Mio Portale**: consente di accedere al portale della Lego per trovare strumenti, download e informazioni varie;

3. **Barra degli strumenti**: contiene i comandi che vengono selezionati più di frequente dalla barra dei menu;

4. **Area di lavoro**: consente di effettuare la programmazione vera e propria, trascinando i blocchi e posizionandoli nel diagramma di flusso;

5. **Finestra di guida rapida**: fornisce l'accesso alla guida in linea;

6. **Mappa dell'area di lavoro**: consente di avere una panoramica d'insieme dell'area di lavoro;

7. **Barra dei comandi di programmazione**: contiene tutti i blocchi necessari alla programmazione;

8. **Pannello di configurazione**: ogni blocco di programmazione dispone di un pannello di configurazione che ne consente la personalizzazione per gli specifici input e output;

9. **Controller**: attraverso questi pulsanti è possibile scaricare il programma direttamente sull'NXT, oltre a modificare le le impostazioni dell'NXT stesso;

10. **Finestra dell' NXT**: è una finestra di popup contenente informazioni riguardanti la memoria dell'NXT e le impostazioni di comunicazione.

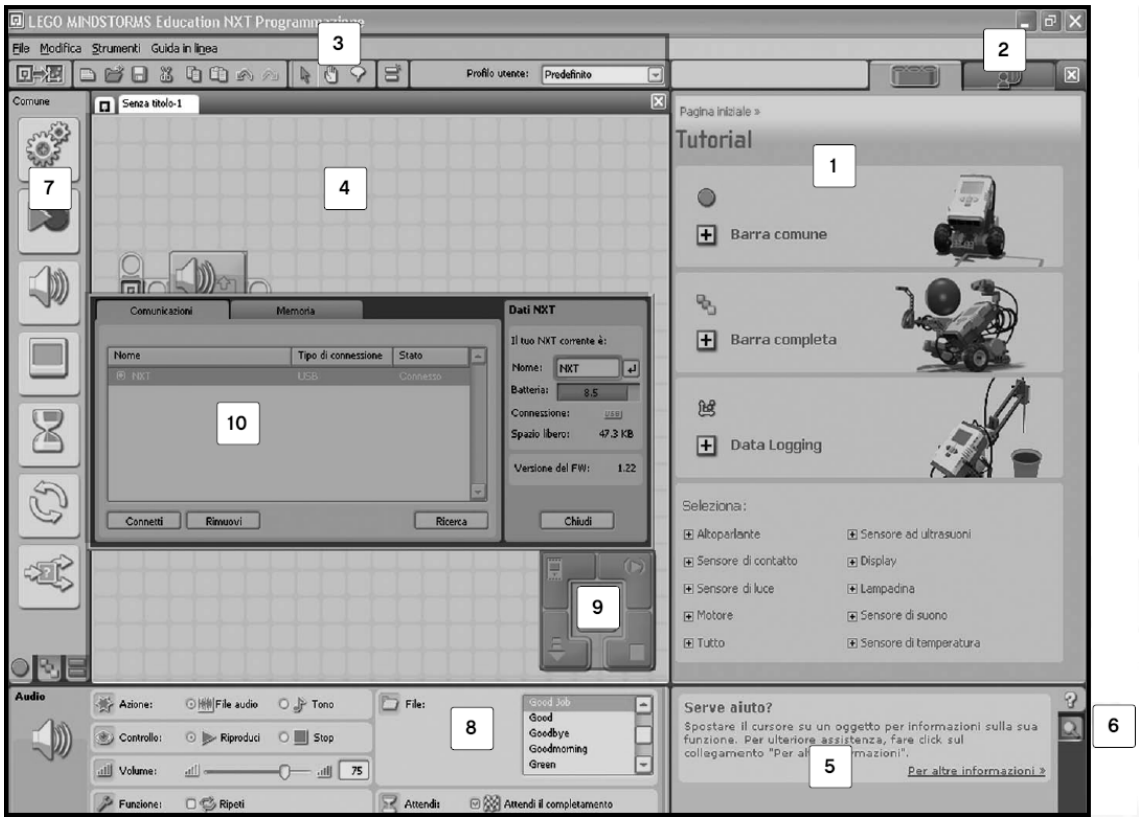

Figura 5.1: L' ambiente di programmazione NXT-G.

5.2.1 Data logging

L'installazione del software NXT-G comprende anche l'installazione dell'applicazione **Data logging**. Con quest'espressione vengono identificate le operazioni di raccolta, memorizzazione e analisi dei dati.
Un classico sistema di data logging osserva un evento o un fenomeno per un determinato periodo di tempo utilizzando sensori collegati ad un calcolatore. Variazioni di temperatura, intervalli e intensità della luce sono solo alcuni dei più comuni esempi di informazioni che possono essere raccolte da un sistema di questo tipo. Applicazioni reali di sistemi di data logging sono le stazioni meteorologiche e le scatole nere degli aeroplani.
Durante un'operazione di data logging vengono eseguite le seguenti tre fasi:

- **Previsione**: determina in anticipo il risultato di un evento o di un fenomeno;
- **Raccolta**: durante il corso dell'esperimento o dell'evento vengono raccolti dei dati;
- **Analisi**: i dati raccolti vengono esaminati e confrontati con i risultati previsti.

In figura 5.2 viene mostrata l'interfaccia utente dell'applicazione Data logging.

1. **Tutorial**: contiene istruzioni per il data logging, basate sui opportuni modelli;
2. **Il Mio Portale**: consente di accedere al portale della Lego per trovare strumenti, download e informazioni varie;
3. **Barra degli strumenti**: contiene funzioni di previsione e di altro tipo utili per l'analisi dei dati;
4. **Grafico**: costituisce la rappresentazione visiva del file dati e risulta utile per creare previsioni, condurre esperimenti ed analizzare i risultati direttamente al suo interno;

5. **Finestra di guida rapida**: fornisce indicazioni utili e dà accesso ad una guida in linea più ampia;
6. **Asse y**: mostra l'unità di misura del sensore;
7. **Asse x**: mostra la durata del tempo dell'esperimento;
8. **Tabella dati**: contiene i valori previsti e quelli registrati dai sensori;
9. **Configurazione esperimento**: ogni esperimento dispone di un pannello di configurazione, che consente di personalizzare il numero e il tipo dei sensori, così come la durata e il tempo di campionamento;
10. **Controller data logging**: consente di comunicare con il brick e di spostare i file dati dall'NXT al calcolatore.

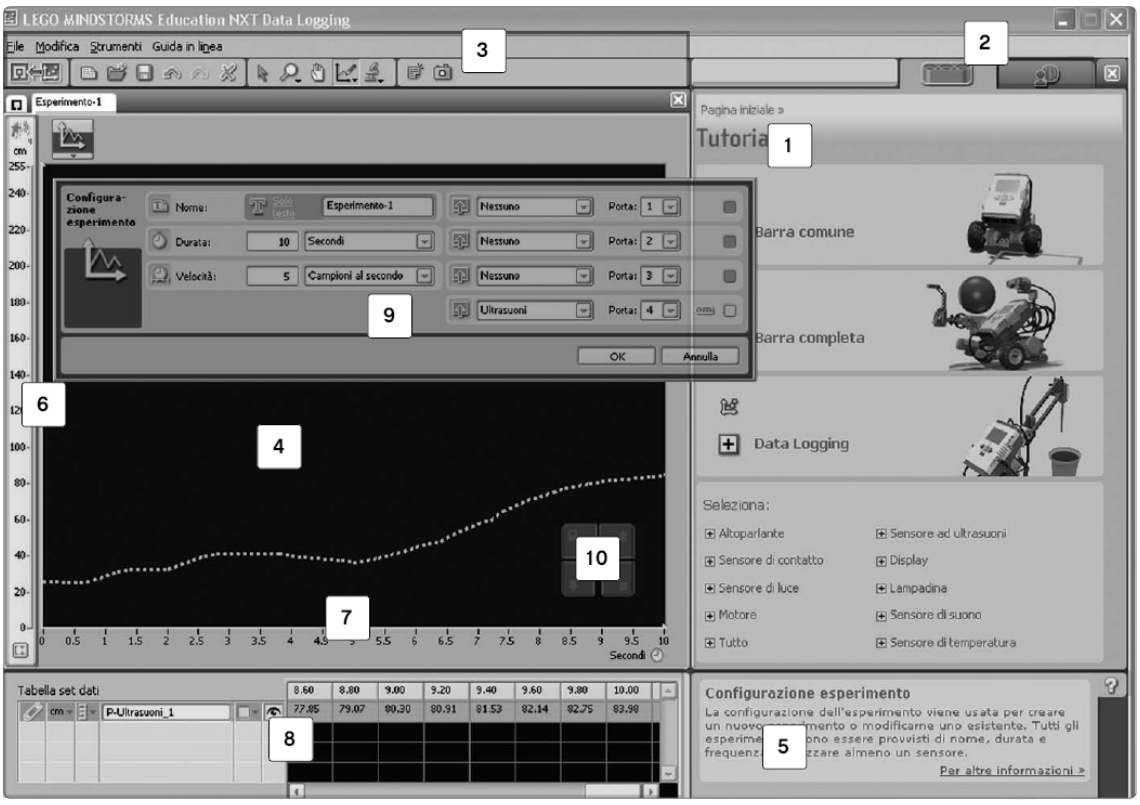

Figura 5.2: L'ambiente di Data logging.

5.3 NQCbaby

Il linguaggio NQCbaby è stato proposto da Giovanni Marcianò. Si tratta di un linguaggio testuale orientato ai bambini, definito mediante macro nel linguaggio NQC.
L'introduzione del Lego Mindstorm RCX nel 1998 ha determinato l'interesse da parte di una folta schiera di persone nell'estendere e arricchire le potenzialità del software originale. Dave Baum è stato uno dei primi innovatori. Ha sviluppato un nuovo linguaggio di programmazione per l'RCX, che utilizza una sintassi molto simile al linguaggio di programmazione C. A causa delle ovvie limitazioni imposte dal firmware dell'RCX, gli sforzi di Baum hanno dato vita ad un linguaggio più ristretto: l'NQC (Not Quite C).
Marcianò sostiene che l'uso di un linguaggio basato sulla lingua italiana e contenente espressioni famigliari sia fondamentale per permettere ai bambini di utilizzare in modo consapevole sia le componenti del robot, sia il linguaggio di programmazione. L'obiettivo connesso alla definizione del linguaggio NQCbaby consiste nel disporre di primitive che facciano riferimento a termini solitamente usati dai bambini.
NQCbaby si impone come strumento che vuole riportare nella didattica la centralità della lingua rispetto al saper fare. Citando a tale proposito una frase di Jerome Bruner:

"*l'insegnamento è enormemente facilitato dal mezzo del linguaggio, che finisce non solo per essere il mezzo per lo scambio, ma lo strumento che lo stesso discente può usare in seguito, per organizzare l'ambiente*".

La struttura del linguaggio NQC prevede l'impiego di macrocomandi grazie ai quali è possibile semplificare la sintassi nativa, come pure attuare una trasposizione linguistica dal vocabolario tecnico inglese ad un vocabolario naturale italiano. Nell'esempio riportato nelle figure sottostanti viene messa in evidenza tale semplificazione.

Nello scrivere il programma il discente viene chiamato ad un esercizio di rigore linguistico. Viene invitato ad essere conciso e preciso, dando ordine al pensiero

```
task main ()
{
  repeat (4)
  {
       OnFwd(OUT_A + OUT_B);
       Wait(100);
       OnRev(OUT_B);
       Wait(50);
  }
  Off(OUT_A + OUT_B);
}
```

Figura 5.3: Esempio di programma in linguaggio NQC.

```
procedura
{
  ripeti (4)
  {
       avanti(100);
       destra(50);
  }
  fermatutto;
}
```

Figura 5.4: Semplificazione della sintassi di NQC mediante NQCbaby.

e strutturando l'articolazione della frase con attenzione e metodo in modo tale che il robot comprenda l'istruzione data.

5.4 Not Exactly C

Come già accennato in precedenza, il firmware presente sull'NXT risulta essere molto diverso da quello presente sull'RCX. La comunità di sviluppatori ha pensato di estendere ulteriormente l'NQC per favorire anche il supporto dell'NXT. Tuttavia, a causa delle notevoli difficoltá e dell' enorme quantiá di tempo speso, i programmatori hanno optato per lo sviluppo di un nuovo compilatore, con l'intento di mantenere la compatibilità con l'NQC e rendere semplice la transizione verso un nuovo linguaggio di programmazione. Il nuovo linguaggio

prende il nome di NXC (Not Exactly C), il cui fondatore è John Hansen.

Un programma scritto in NXC è una sequenza di *task*. Ogni task è costituito da una serie di comandi detti *statement*, ognuno dei quali deve necessariamente terminare con un punto e virgola. Ogni programma deve contenere il task main, senza il quale non è possibile dare il via alla sua esecuzione. All'interno di un programma è possibile inserire fino ad un massimo di 255 task che vengono eseguiti successivamente al task main. In figura 5.5 viene riportata la struttura di un programma scritto in linguaggio NXC.

```
task main ()
{
  statement1;
  statement2;
  …
}
```

Figura 5.5: Struttura di un programma NXC.

I file contenenti il programma scritto in linguaggio NXC hanno estensione **.nxc**. Per permettere al robot di eseguire le istruzioni contenute all'interno del programma è necessario che il file venga compilato e convertito in un file con estensione **.rxe**.
A tale scopo serve **nbc**, il compilatore prodotto da John Hansen, reperibile al seguente indirizzo `http://bricxcc.sourceforge.net/nbc/`.
Digitando da riga di comando la sintassi seguente

```
nbc nomeprogramma.nxc -O= nomeprogramma.rxe
```

l'effetto di tale istruzione, in assenza di errori, consiste nel produrre un file con estensione .rxe nella directory specificata in fase di compilazione. In figura 5.6 viene illustrata l' esecuzione dell'istruzione appena descritta dal Prompt dei comandi di Windows.
L'azione successiva prevede l'upload del file direttamente sul brick mediante **nexttool**, un programma sviluppato da John Hansen e reperibile al seguente

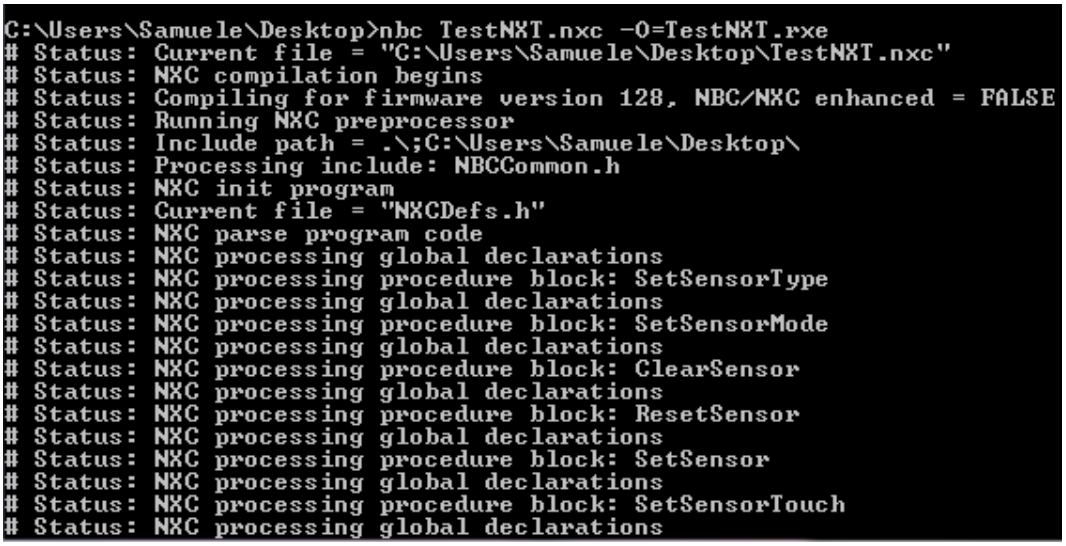

Figura 5.6: Esempio di compilazione di un programma NXC.

indirizzo `http://bricxcc.sourceforge.net/utilities.html`.
Digitando da riga di comando la sintassi seguente

```
nexttool /COM=BTH::nomeNXT::indirizzoBluetooth /BT
-download=nomeprogramma.rxe -run=nomeprogramma.rxe
```

l'effetto di tale istruzione determina l'upload del programma tramite protocollo Bluetooth e la sua successiva esecuzione. In figura 5.7 viene mostrata l'esecuzione dell'istruzione appena esposta.

Figura 5.7: Esempio di upload ed esecuzione di un programma sull'NXT.

5.5 LeJOS

Il robot Lego Mindstorm NXT nasce per essere programmato con il software di programmazione visuale NXT-G. Tuttavia, Lego ha rilasciato il progetto

sotto licenza Open Source, mettendo a disposizione della comunità informatica tutte le specifiche software e hardware del robot. In questo modo, chiunque può creare software e hardware personalizzabili per l'NXT.
La comunità LeJOS (*Lego Java Operating System*) ha dato vita ad una Java Virtual Machine da installare sul robot con l'obiettivo di consentire l'esecuzione di programmi scritti in linguaggio Java direttamente sul brick. Ovviamente, si tratta di una macchina virtuale Java "ristretta". Infatti, le capacità di calcolo e le risorse hardware dell'NXT sono decisamente ridotte rispetto a quelle di un normale calcolatore. Di seguito si riportano alcune tra le caratteristiche distintive di LeJOS:

- **Linguaggio Object Oriented**: LeJOS supporta il paradigma della programmazione orientata agli oggetti e permette di implementare i meccanismi di:
  - *Incapsulamento*: ogni classe risulta essere un'entità ben definita e distinta rispetto al resto del codice. I metodi e i dati al suo interno sono separati da quelli delle altre classi, quindi non possono subire interferenze e non possono a loro volta interferire in parti esterne alla classe non di loro competenza;
  - *Ereditarietà*: consente di derivare nuove classi a partire da classi già definite. In questo modo è possibile creare classi con una gerarchia ben precisa e sviluppare agevolmente codice riutilizzabile;
  - *Polimorfismo*: consente di riscrivere uno stesso metodo in modo che possa lavorare con dati di tipo diverso.
- **Numeri in virgola mobile**: vengono supportati i numeri in virgola mobile e di conseguenza sono possibili calcoli di funzioni trigonometriche, fondamentali per molti algoritmi di robotica;
- **Thread**: viene supportata la programmazione concorrente. La macchina virtuale gestisce i thread secondo uno schema di tipo "preemptive" ed è possibile crearne fino a 255. Tuttavia, possono essere istanziati al massimo cinque thread dal momento che il loro impiego di memoria risulta essere significativo.

- **Array**: viene fornito supporto alla creazione di array multidimensionali, la cui dimensione massima raggiunge i 255 elementi;
- **Modello ad eventi**: LeJOS è in grado di utilizzare il modello ad eventi di Java che include listeners e sorgenti di eventi;
- **Eccezioni**: viene fornito il supporto alle eccezioni e alla loro gestione;
- **Ricorsione**: è possibile realizzare livelli di ricorsione fino ad un numero massimo di dieci.

In figura 5.8 viene riportata l'architettura di LeJOS.

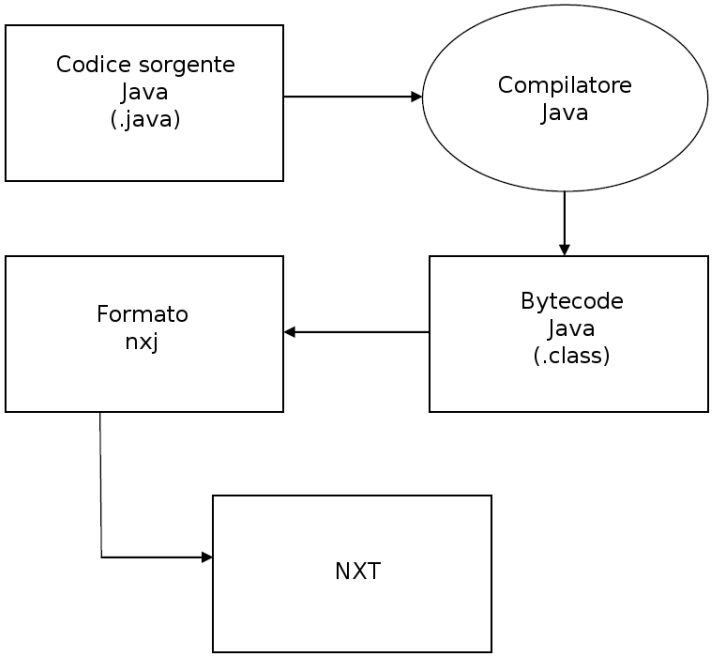

Figura 5.8: Architettura LeJOS.

I file contenenti un programma scritto in linguaggio Java hanno estensione (.java). La compilazione di tali file determina la produzione di altrettanti file con estensione **.class**. LeJOS mette a disposizione **nxjc** un apposito strumento per la compilazione dei file Java.
Digitando da riga di comando la seguente istruzione

`nxjc nomeprogramma.java`
l'effetto è quello di produrre uno o più file .class. L'operazione di compilazione viene illustrata in figura 5.9

Figura 5.9: Esempio di compilazione di un programma LeJOS.

Per poter eseguire l'upload del programma direttamente sul brick è necessario digitare la seguente istruzione

`nxj nomeprogramma`

che provvede a trasformare il file .class in un formato apposito (.nxj) e a caricarlo sull'NXT.

Figura 5.10: Esempio di upload di un programma LeJOS sul brick.

5.5.1 Installazione di LeJOS su Windows

Tutta la documentazione e il software del progetto LeJOS sono reperibili all'indirizzo `http://lejos.sourceforge.net/`.
Alla pagina `http://lejos.sourceforge.net/nxj-downloads.php` sono disponibili i file da scaricare per l'installazione del firmware. Una volta avviata la procedura d'installazione e selezionata la directory di destinazione dei file, viene avviata la procedura di aggiornamento del firmware dell'NXT, che si conclude con il riavvio dell'NXT e il successivo avvio del nuovo firmware LeJOS.

5.5.2 Il menu di LeJOS

In figura 5.11 viene riportato il menu di LeJOS.

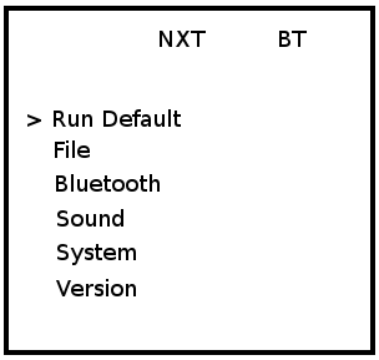

Figura 5.11: Il menu di LeJOS.

- **Run Default**: avvia il programma impostato di default;
- **Files**: visualizza i file presenti nella memoria dell'NXT e consente la loro esecuzione;
- **Bluetooth**: attiva/disattiva la funzionalità di Bluetooth. Consente di cercare, elencare o eliminare i dispositivi associati. Inoltre, è possibile attivare/disattivare la visibilità dell'NXT.
- **Sound**: imposta il volume dei suoni;
- **System**: al suo interno è possibile visualizzare lo stato della memoria e della batteria;
- **Version**: visualizza la versione del firmware LeJOS.

5.5.3 Ripristino del firmware Lego originale

Come già detto in precedenza, con il caricamento del firmware LeJOS sull'NXT non è più possibile eseguire i classici programmi scritti con il software originale fornito dalla Lego. Per poter fare ciò occorre caricare nuovamente il firmware Lego sull'NXT.
Per ripristinare il firmware è necessario avviare sul calcolatore il programma originale Lego (NXT-G), collegare l'NXT alla porta USB e scegliere la voce di menu Strumenti/Aggiorna Firmware NXT.

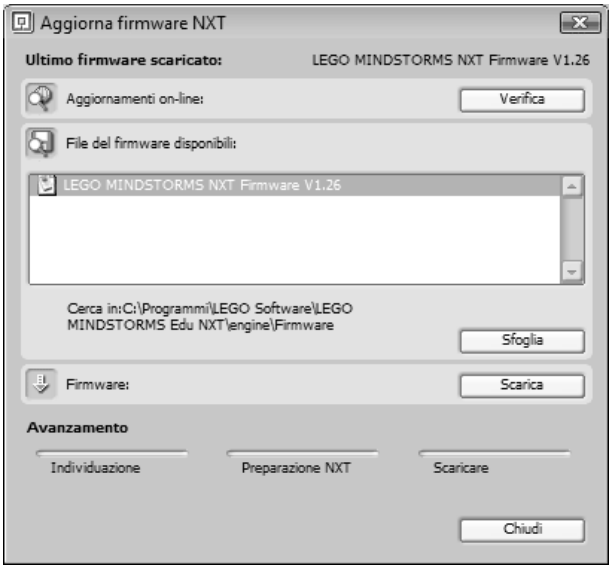

Figura 5.12: La finestra di aggiornamento del firmware Lego.

5.5.4 Strumenti di LeJOS

LeJOS mette a disposizione diversi strumenti richiamabili da riga di comando per la gestione del robot e dei programmi scritti in linguaggio Java. Nella tabella seguente viene riportato l'elenco completo di tali programmi.

Comando	Descrizione
nxjflash	Aggiorna il firmware dell'NXT
nxjc	Compila il programma Java per l'NXT
nxj	Crea il file con estensione `.nxj` (operazione di linking), dopodiché carica i programmi sull'NXT
nxjupload	Carica ed esegue i programmi sull'NXT
nxjbrowse	Visualizza i file presenti nella memoria dell'NXT. Consente di caricare o cancellare file dalla memoria, ottenere informazioni sui file, deframmentare la memoria e impostare il nome dell'NXT.

Continua nella prossima pagina

Comando	Descrizione
nxjmonitor	Consente di monitorare lo stato di funzionamento dell'NXT. In particolare, visualizza graficamente i valori riportati dai sensori, ne legge la posizione e visualizza lo stato di carica della batteria.
nxjdataviewer	Visualizza i dati raccolti con il Data logger.

Tabella 5.1: Strumenti di LeJOS richiamabili da riga di comando.

5.5.5 La Subsumption Architecture

Nel 1986 il professor Rodney Brooks del Massachussetts Institute of Technology introduce per la prima volta il termine di "Subsumption Architecture". Si tratta di un tipo di architettura logica fortemente orientata alla gestione dei comportamenti nell'ambito della robotica autonoma. Secondo tale architettura, la struttura di un comportamento complesso viene decomposta in una sottostruttura di comportamenti più semplici correlati tra loro. Ad ogni comportamento semplice viene associato un livello di astrazione, che rappresenta un obiettivo ben preciso che l'agente deve portare a termine.
L'architettura ha una struttura di tipo bottom-up, nella quale sono gli eventi che si verificano negli strati inferiori dell'architettura ad offrire la possibilità a quelli superiori di essere attivati e portati a termine.
Ogni comportamento viene visto come un'entità dotata di tre caratteristiche funzionali:

- una condizione di attivazione;
- un'attività da svolgere in fase attiva;
- un'attività da svolgere in caso di soppressione.

Per identificare la condizione di attivazione del comportamento ed eseguire la relativa azione, compresa l'eventuale soppressione viene utilizzato un supervisore (un arbitro).

La programmazione che segue il modello dei behaviour consente al programma di mantenere separate tra loro le varie operazioni da eseguire e i controlli necessari per gestire il flusso di esecuzione del programma. In questo modo, la programmazione diventa modulare, mantenendo il codice semplice e leggibile anche in presenza di molti behaviour. Inoltre, ogni behaviour é indipendente dagli altri, quindi in fase di sviluppo potrá essere creato e testato come un parte di programma a sé stante, senza subire le influenze di altre parti di codice.
Un behaviour é un'entità formata da tre parti:

- un test condizionale, che verifica se il behaviour deve essere eseguito;
- una sequenza di operazioni da eseguire quando il behaviour viene attivato;
- una sequenza di operazioni da eseguire quando il behaviour viene disattivato.

Ogni behaviour é contenuto in una classe che implementa l'interfaccia **Behaviour**, che definisce i metodi riportati nella seguente tabella.

Metodo	**Descrizione**
takeControl()	determina se il behaviour deve essere attivato
action()	contiene le operazioni da eseguire quando il behaviour viene attivato
suppress()	contiene le operazioni da eseguire nel momento in cui il behaviour viene disattivato

Tabella 5.2: Metodi che agiscono su un oggetto behaviour.

Il controllo dei behaviour viene operato da una classe apposita, preposta alla verifica del metodo takecontrol() e all'attivazione e disattivazione dei vari behaviour facenti parte del programma, eseguendone i rispettivi metodi action() e suppress() a seconda della situazione. La classe che esegue questo importante compito è la classe **Arbitrator**.

In LeJOS esistono essenzialmente due tipi di behaviour, a seconda del comportamento del codice del loro metodo action():

- behaviour a esecuzione completa: nel momento in cui prendono il controllo, eseguono il codice del proprio metodo action() fino in fondo, cedendo solo allora il controllo all'oggetto Arbitrator;
- behaviour a esecuzione continuata: proseguono la loro esecuzione anche dopo che il metodo action() ha restituito il controllo alla classe Arbitrator.

Nei behaviour ad esecuzione completa, solitamente, non occorre eseguire alcun codice nel metodo suppress(), mentre in quelli ad esecuzione continuata tale metodo è fondamentale, per esempio per fermare i motori o interrompere l'esecuzione di thread eventualmente attivati.

Capitolo 6

Mobolab

Un laboratorio didattico è una struttura fisica e in quanto tale prevede una fase di progettazione che non può e non deve essere trascurata. Il laboratorio racchiude la vera essenza della didattica, un luogo dedicato alla sperimentazione e l'apprendimento.

6.1 L'idea di Mobolab

L'acronimo "Mobolab" nasce da un'idea del prof. Riccardo Cassinis relativamente alla creazione di un laboratorio didattico di robotica mobile remotamente fruibile attraverso Internet.
Il laboratorio intende porsi come valido strumento al servizio dell'insegnamento e, in particolare, vuole rendere concrete delle esperienze di robotica troppo spesso relegate al semplice utilizzo di simulatori. Il successo determinato dai kit robotici e la continua adesione da parte di istituti sparsi in tutto il territorio italiano, è manifesto di un interesse in costante aumento verso tale disciplina. La consapevolezza che nell'era moderna le tecnologie permeino sempre più la nostra società e che, conseguentemente, ogni individuo possa facilmente dotarsi di un calcolatore e di una connessione Internet, giustificano la natura remotabile del laboratorio didattico. Non soltanto un ambiente completamente controllabile da postazione remota, ma un luogo in grado di non dipendere dal-

l'intervento umano, ovvero un ambiente in grado di svolgere le proprie mansioni in maniera completamente autonoma.
Il realismo è il vero punto chiave attorno al quale ruota l'idea di Mobolab. Il laboratorio si interfaccia verso il mondo esterno attraverso un browser, ma a differenza dei software di simulazione, i robot che si muovono all'interno di questo ambiente sono delle entità reali. La simulazione è una rappresentazione interattiva della realtà basata sulla costruzione di un modello del sistema del quale si vuole comprendere il funzionamento. Dal punto di vista didattico la simulazione è una tecnica che può essere utilizzata per comprendere le interrelazioni tra i componenti di un sistema o di un processo, per verificare delle ipotesi relativamente a cosa accadrebbe in un sistema in conseguenza a certe decisioni, e per esaminare le possibili situazioni future. È ampiamente riconosciuto che lo scenario tipico dell'apprendimento sia centrato sulla vita reale e che gran parte di ciò che viene appreso si fondi sull'interazione con la realtà che ci circonda.
L'attrattiva offerta da un oggetto reale è superiore rispetto a quella offerta da un oggetto simulato. L'idea di far muovere un robot che si trova all'interno di un dipartimento, nel sotterraneo di un Università incuriosisce e stimola maggiormente la creatività di chi compie la manipolazione. La scelta di utilizzare dei kit robotici a basso costo ed in particolare l'adozione del kit Lego Mindstorm NXT avvalora e rafforza le considerazioni appena elencate. Robot "giocattolo" realizzati soprattutto per un pubblico giovane e quindi progettati per sopportare i più svariati stress meccanici. In un contesto di questo tipo scompare il timore di dover manovrare un'apparecchiatura costosa e delicata e si entra in un'ottica quasi ludica, nella quale il robot viene visto come un giocattolo da analizzare in tutte le molteplici sfaccettature.
Mobolab rappresenta l'ambiente nel quale realizzare la propria esperienza robotica e si pone nella maniera più assoluta come connubio tra apprendimento e divertimento, la famosa "lenza", per citare un'aforisma Papertiano, che bisogna dare ad un uomo per far sì che impari a pescare.

6.2 La realizzazione del laboratorio

Il laboratorio didattico Mobolab (figura 6.1) si trova all'interno del Laboratorio di Robotica Avanzata dell'Università degli Studi di Brescia.

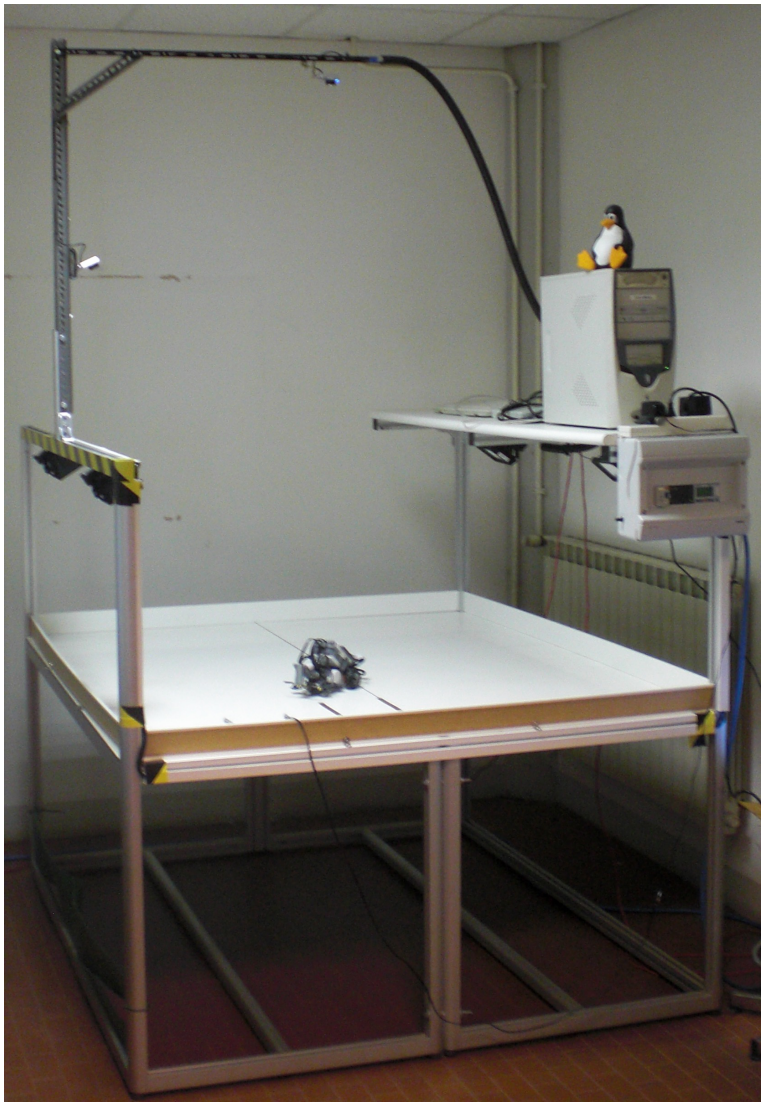

Figura 6.1: Il laboratorio Mobolab.

La superficie di lavoro consta di due tavoli affiancati e collegati in maniera solida tra loro, della dimensione di 1560 x 1800 mm. Il piano è posto a 810 mm dal pavimento ed è circondato da una sponda alta 70 mm di colore bian-

co. La scelta di unire i due tavoli ha consentito di dar vita ad una superficie ideale per permettere ai robot Lego di effettuare i movimenti nella maniera più comoda possibile. Ogni tavolo è dotato inoltre di un'apposita struttura, originariamente adibita al sostegno di una mensola. Per quanto riguarda il tavolo di sinistra, tale mensola è stata rimossa e si è proceduto alla realizzazione di una struttura metallica per il sostegno di due telecamere. Il tavolo di destra invece non ha subito l'asportazione della mensola dal momento che si è ritenuto potesse avere una qualche forma di utilità.
Essendo Mobolab fruibile remotamente attraverso il Web, l'utente deve poter vedere comodamente la superficie di lavoro e pertanto si è provveduto al posizionamento di due telecamere usb Philips modello SPC1005NC.
La prima telecamera è stata posizionata esattamente sulla verticale ad un'altezza di xxx mm dal centro del piano di lavoro in modo da fornire un'immagine il meno possibile distorta relativamente alla posizione dei robot. La seconda telecamera è stata posizionata obliquamente, sul piano perpendicolare del tavolo e passante per il suo asse minore, ad un altezza di xxx mm. L'impiego di due telecamere consente di avere una visione ottimale del piano di lavoro. Mentre la prima telecamera fornisce una visione globale di tutta l'area di lavoro, la seconda telecamera offre una panoramica più ristretta, finalizzata al dettaglio.
L'idea di rendere maggiormente visibile la superficie ha determinato l'installazione di quattro faretti alogeni (figura 6.2) collegati in parallelo ad un controllore logico programmabile (PLC).

Il PLC è un Crouzet Millenium III ed è provvisto di quattro uscite a relè, di cui le prime due controllano i faretti che illuminano il tavolo.
Inoltre, sono stati predisposti due calcolatori:

- una macchina "Linux", sulla quale è installata la release Lenny di Debian, adibita al controllo delle luci, dello stream video proveniente dalle telecamere e avente il ruolo di Webserver per il sito di Mobolab;

- una macchina "Windows", sulla quale è installato il sistema operativo Windows XP, avente il ruolo di Webserver per il controllo dei robot Mindstorm attraverso apposite applicazioni Web.

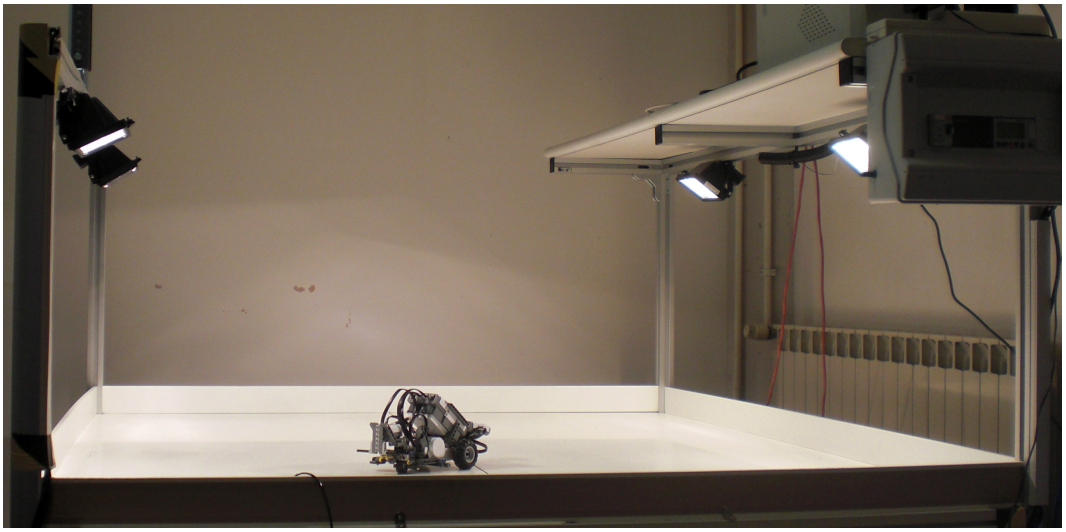

Figura 6.2: I faretti alogeni.

6.3 La gestione delle luci

Come già accennato nella sezione precedente, le luci vengono gestite da un controllore logico programmabile. Tale PLC è dotato di otto ingressi, quattro dei quali sono collegati alla porta parallela del calcolatore Linux. Il collegamento alla porta parallela dà la possibilità di controllare il PLC utilizzando il calcolatore mediante due programmi ("plcd" e "plcc") che regolano il funzionamento delle luci.
In particolare, il programma plcc riceve in ingresso un parametro che causa il passaggio attraverso una sequenza di stati (si veda la figura 6.3):

- lampade spente;
- coppia di lampade 1 accese;
- tutte le lampade accese;
- coppia di lampade 2 accese;
- lampade spente.

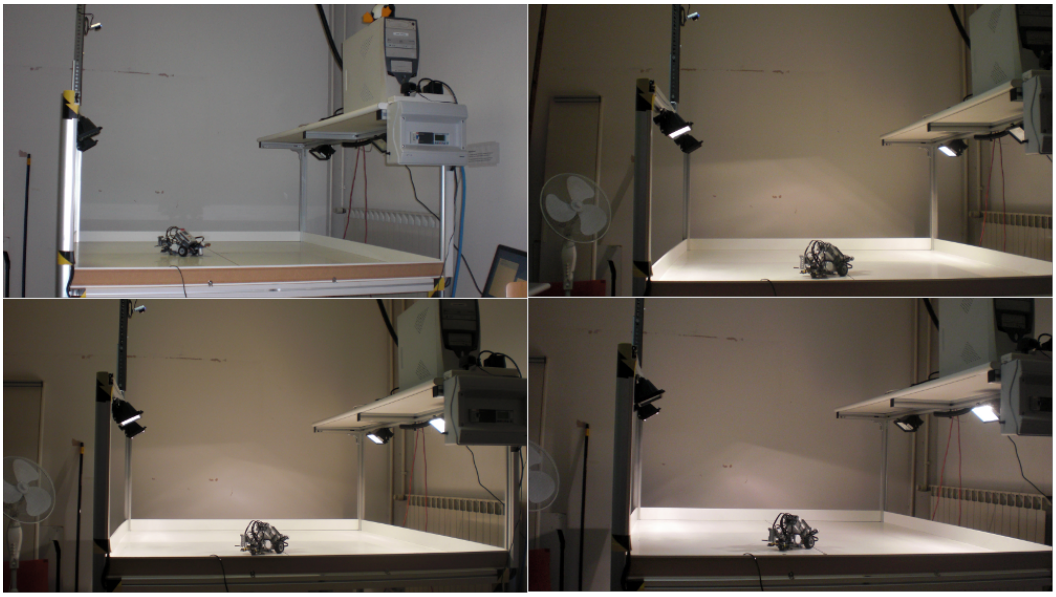

Figura 6.3: Funzionamento dei faretti.

6.4 La gestione delle telecamere

Le due telecamere sono collegate al calcolatore Linux tramite i loro appositi cavi usb e la gestione dello stream video viene gestita dal pacchetto open source **mjpg_streamer**.
Si tratta di un'applicazione a linea di comando per il trasporto di file JPEG, da uno strumento di cattura delle immagini (una webcam) ad un opportuno sistema di visualizzazione (un cosiddetto viewer). Una delle peculiarità di questo pacchetto consiste nello sfruttamento della compressione hardware di cui sono provviste molte telecamere. In questo modo i calcolatori non devono spendere il loro tempo nell'eseguire la compressione dei frame video, con evidente risparmio di risorse. Per quanto riguarda le telecamere Philips installate, è stato eseguito un test dei formati supportati che vengono riportati nella tabella seguente.

Formato	Risoluzione
QSIF (Quarter Source Input Format)	160x120
CGA (Color Graphics Adapter)	320x240
CIF (Common Intermediate Format)	352x288
SVGA (Super Video Graphics Array)	800x600
SXGA (Super Extended Graphics Array)	1280x1024

Tabella 6.1: Formati video supportati.

La scelta del viewer è ricaduta su Cambozola, un'applet che consente di visualizzare in maniera fluida lo stream proveniente da ciascuna delle due telecamere. Il calcolatore al quale sono collegate le due webcam non è una macchina di ultima generazione. Ogni tanto accade che le telecamere si blocchino andando incontro ad una situazione di stallo, scomparendo addirittura dall'elenco dei dispositivi. Per ovviare a questo inconveniente si è pensato di scrivere un demone che si occupa di ricercare il file `/tmp/resetUSB` e, una volta individuato tale file, procedere al reset del sottosistema USB. La seguente procedura prevede che il calcolatore rimanga inattivo per la durata di una quindicina di secondi, al termine della quale viene ripristinato il regolare funzionamento delle telecamere.

6.5 Il sito Web di Mobolab

Per poter usufruire dei servizi messi a disposizione del laboratorio didattico Mobolab è necessario collegarsi tramite Internet al seguente indirizzo:

`http://herbie.ing.unibs.it:8205/~mobolab/`

Il sito Web di Mobolab è stato creato dal prof. Riccardo Cassinis, utilizzando come base un template open source realizzato da Andreas Viklund. Come visibile in figura 6.4 il sito è strutturato nel modo seguente: nella parte alta è presente l'header contenente il logo dell'Università degli Studi di Brescia e il

Figura 6.4: L'interfaccia di Mobolab.

nome del sito.
Sono presenti, inoltre, cinque aree raggiungibili tramite il menu posizionato al di sotto dell'header:

- **Osservazione**: rappresenta una sorta di home page del sito. All'interno di questa sezione viene presentato il laboratorio didattico ed è possibile fruire delle immagini provenienti dalla seconda telecamera che mostrano la superficie di lavoro e i robot disponibili. Si tratta di una pagina del tutto informativa e introduttiva, che illustra le finalità e gli obiettivi del progetto Mobolab.

- **Telecomando**: consente di controllare i robot NXT Mindstorm e i rispettivi sensori attraverso due specifiche applicazioni web: un'applicazione per il controllo del robot Lego e un'altra per il controllo del robot LeJOS.

- **Beebot**: consente di effettuare la programmazione dei robot sfruttando lo stile di interazione *Beebot*, nel quale l'interazione con un'ape elettronica ha come obiettivo l'avvicinamento dei bambini al mondo della robotica.

- **Programmazione iconica**: la programmazione dei robot viene eseguita attraverso la remotazione dell'applicazione NXT-G, per consentire agli utenti l'utilizzo dello strumento software fornito direttamente con i kit Lego Mindstorm, ed esplorare le potenzialità della programmazione iconica.

- **Programmazione testuale**: mette a disposizione degli utenti una modalità di programmazione attraverso cui realizzare programmi scritti in linguaggio NXC (Not eXactly C).

Nella parte destra dell'interfaccia utente di Mobolab è presente un menu laterale che riporta le seguenti sezioni:

- **Risoluzione video**: contiene tre pulsanti che consentono all'utente di selezionare la qualità (alta, media, bassa) dello stream video proveniente dalle due telecamere. Dal momento che il flusso video determina un'occupazione di banda non indifferente, si è pensato di consentire all'utente la possibilità di selezionare risoluzioni diverse a seconda delle esigenze.

- **Collegamento**: consente di determinare la velocità del collegamento Internet e calibrare il sistema sulla base di tale velocità.

- **Illuminazione**: la pressione dei pulsanti presenti all'interno della seguente sezione consente di agire sui quattro faretti per l'illuminazione del piano di lavoro. Come già esposto nel paragrafo relativo alla gestione delle luci, il clic sul pulsante "ACCENDI-SPEGNI" determina l'accensione di una coppia di lampade. Il successivo clic del pulsante realizza l'accensione della coppia di lampade spente; in questo modo, tutte le luci risultano essere attive. Al clic seguente viene spenta la prima coppia di faretti, mentre ad un successivo clic le lampade risultano essere tutte spente. Il pulsante "SPEGNI TUTTO" determina lo spegnimento di tutte le lampade accese. Nel caso di blocco dello stream video è possibile premere il pulsante "RESET VIDEO" che provvede ad effettuare il reset del sottosistema USB e a ripristinare lo stream video.

6.6 La velocità di connessione

Ai fini di un utilizzo ottimale dei servizi forniti da Mobolab è importante la determinazione della velocità di connessione. Più precisamente, l'idea consiste nel calcolare la velocità e, in base a quest'ultima, modificare alcuni aspetti di funzionamento del sistema. Per ora il calcolo della velocità provvede ad impostare la risoluzione con cui vengono trasmesse le immagini video.
Il meccanismo di funzionamento è il seguente:

1. All'apertura della pagina *index.php* viene verificata la presenza di un cookie (lo *SpeedCookie*), il cui valore rappresenta la stima della velocità di download (in Bit/s).

2. In assenza dello SpeedCookie viene caricata la pagina *provavel/misura.html*, che effettua la misura, crea lo SpeedCookie e ricarica la pagina index.php.

3. In funzione del valore dello SpeedCookie viene calcolata una stringa (LO, MI, HI).

4. Viene verificata l'esistenza del cookie *ResolutionCookie* contenente l'ultima velocità impostata. L'assenza di quest'ultimo determina l'esecuzione di un programma (*cgi-bin/resolution_lo*, *cgi-bin/resolution_med* o *cgi-bin/resolution_low*) e il successivo ricaricamento della pagina index.php.

5. L'esistenza del ResolutionCookie determina l'esecuzione di un programma solamente se il suo valore è diverso dalla stringa di velocità calcolata.

Nell'eventualità venga selezionato uno dei tre pulsanti per l'impostazione manuale della risoluzione, allora lo SpeedCookie viene modificato al valore di velocità specificato, in modo tale che al successivo caricamento della pagina non venga rieseguito un nuovo settaggio.
Per rendere più chiaro e comprensibile il funzionamento del meccanismo di determinazione della velocità di connessione è stato realizzato il diagramma riportato in figura 6.5.

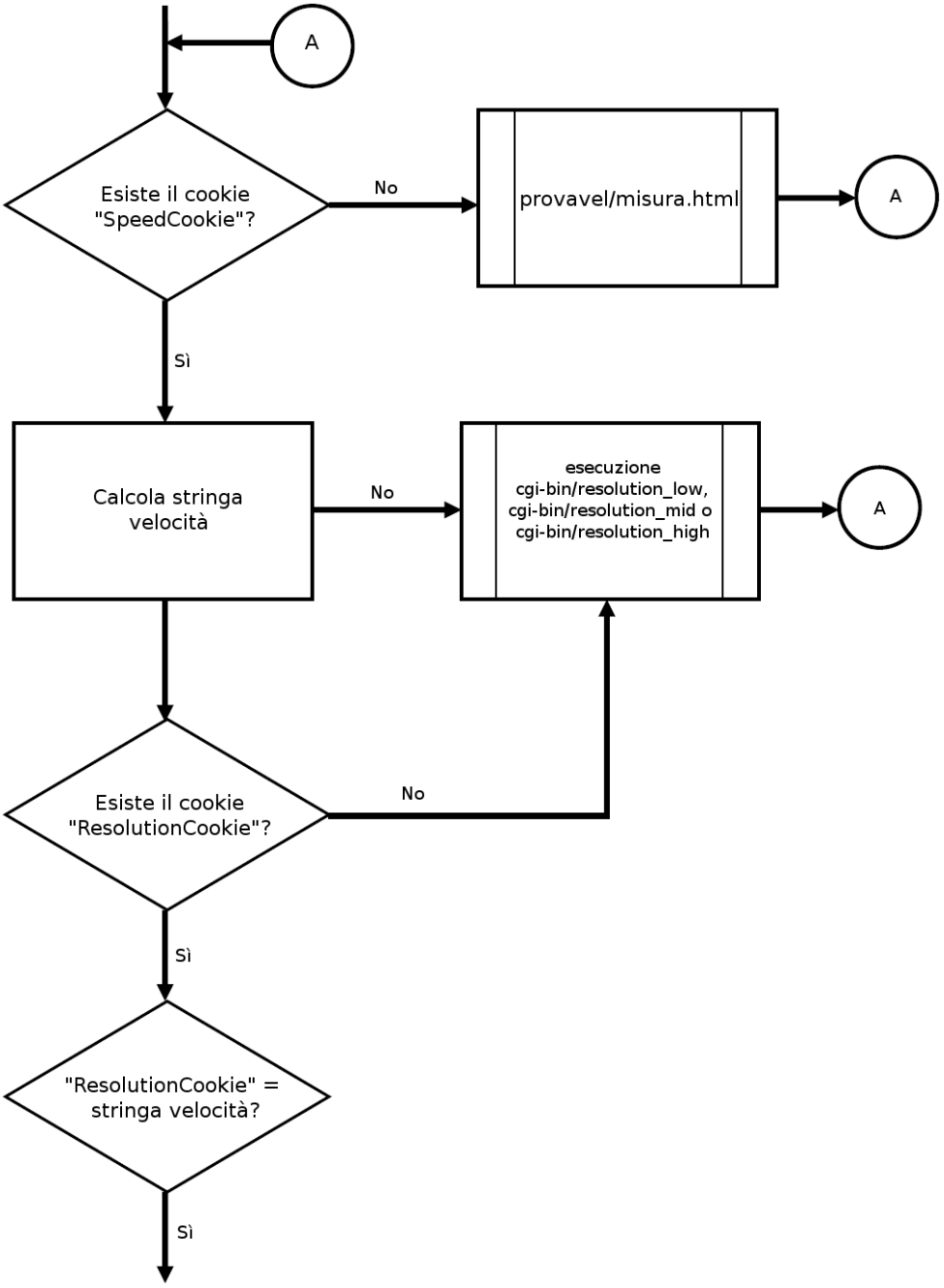

Figura 6.5: Rilevazione automatica della velocità di connessione.

6.7 La stazione di ricarica

I robot Lego Mindstorm NXT sono alimentati da una batteria al litio che offre una buona autonomia. Tuttavia, l'uso prolungato del robot costringe a frequenti operazioni di ricarica della medesima. A tale scopo è stata progettata una stazione di ricarica, visibile in figura 6.6.

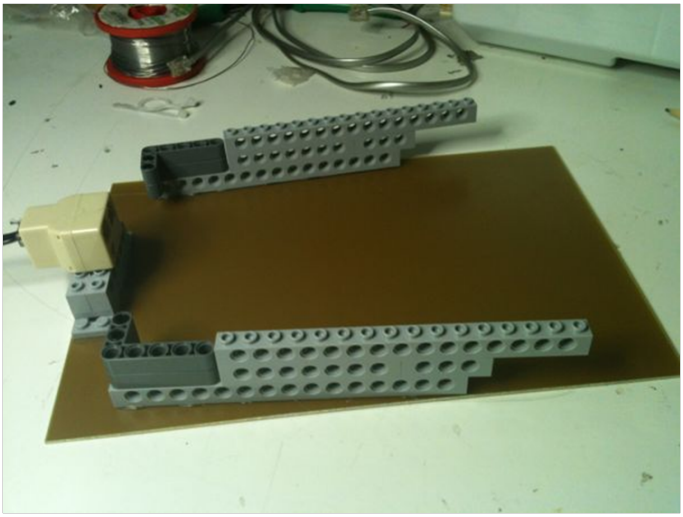

Figura 6.6: La stazione di ricarica.

La parte fissa dei contatti è stata realizzata con uno sdoppiatore per linee telefoniche come si vede in figura 6.7.

Figura 6.7: Lo sdoppiatore.

Tale componente è stato modificato come illustrato in figura 6.8. Anche la forma dei contatti è stata opportunamente modificata per diminuire la forza di reazione in relazione all'inserzione dei contatti mobili.

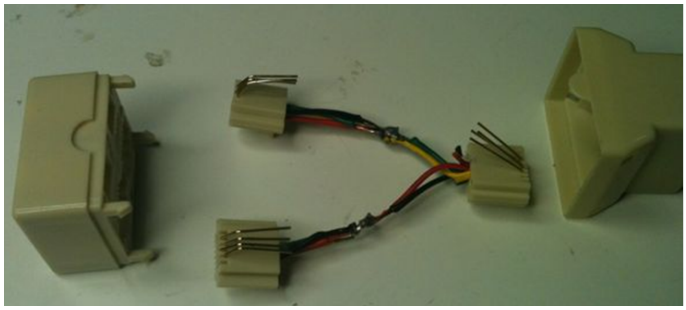

Figura 6.8: Lo sdoppiatore modificato.

La stazione di ricarica presenta due guide laterali che vincolano il robot ad eseguire un rientro corretto. In questo modo i due contatti mobili situati nella parte posteriore dell'NXT vanno ad inserirsi correttamente nei fori dello sdoppiatore consentendo al robot di essere ricaricato in maniera corretta.

6.8 L'accesso al laboratorio

Per poter usufruire dei servizi di Mobolab è necessario effettuare una procedura di registrazione al sito. Pur trattandosi di un contesto didattico ed educativo, intraprendere una politica di sicurezza è sempre una scelta saggia al fine di prevenire possibili azioni di danneggiamento. Il meccanismo di autenticazione viene gestito dal programma **SMF** (Simple Machines Forum). Si tratta di uno script freeware per forum Internet. Lo script, sviluppato dal team di sviluppo Simple Machine, è scritto in linguaggio PHP e utilizza il database MYSQL.
L'utente deve quindi collegarsi al sito di Mobolab e selezionare il pulsante "FORUM". A questo punto è sufficiente effettuare la semplice procedura di registrazione al termine della quale si dispone delle credenziali necessarie per l'accesso ai servizi.
Dal momento che il laboratorio didattico Mobolab è stato pensato come risorsa

disponibile a “chiunque desideri effettuare esperienze di robotica reale”, l'accesse ai servizi viene regolato da un sistema di prenotazione. Sul calcolatore Linux è stato installato il pacchetto *MRBS* (Meeting Rooms Booking System), un'applicazione web per la gestione delle risorse che ricorda molto il Google calendar. L'utente specifica un periodo temporale all'interno del quale utilizzare i servizi del laboratorio. La prenotazione rappresenta il modo più equo di assegnamento delle risorse e permette ad ognuno di poter usufruire dei servizi. Chiaramente le prenotazioni devono essere sottoposte alla supervisione di un amministratore, per evitare che utenti troppo “golosi” occupino le risorse per lunghi archi di tempo a discapito degli altri.

6.9 Il collegamento Bluetooth

Al fine di permettere agli utenti il controllo dei robot Mindstorm NXT in modalità remota, è necessario impostare una connessione Bluetooth. Innanzitutto è opportuno assicurarsi che il calcolatore disponga di connettività Bluetooth. In caso contrario è necessario procurarsi un adattatore Bluetooth USB. Nel contesto di Mobolab, il calcolatore Windows adibito alla gestione del controllo remoto degli NXT non dispone di un'interfaccia radio Bluetooth e pertanto è stato acquistato un'apposito adattatore USB (figura 6.9).

Figura 6.9: Adattatore USB Bluetooth.

Prima di procedere all'operazione di associazione dell'NXT con il calcolatore Windows, è necessaria l'installazione dei driver Lego, che consentono un in-

terfacciamento corretto dei dispositivi NXT al calcolatore. Tali driver sono contenuti nel cd di installazione fornito con il kit di costruzione, ma eventualmente possono essere prelevati dal sito della Lego all'indirizzo

`http://mindstorms.lego.com/en-us/support/files`

La procedura che viene illustrata di seguito si riferisce ad un calcolatore con sistema operativo Windows 7.
Innanzitutto si deve attivare la funzionalità di Bluetooth del dispositivo NXT. Per farlo è sufficiente portarsi nel menù *Bluetooth* e assicurasi che quest'ultimo si trovi sulla configurazione *On*. Inoltre, bisogna verificare che la visibilità del dispositivo sia impostata su *Visible*. Se tutto è stato impostato in maniera corretta, sul display dell'NXT in alto a sinistra compariranno i simboli rappresentati in figura 6.10.

Figura 6.10: Connessione Bluetooth attiva ed NXT visibile.

Il passaggio successivo prevede l'associazione dell'NXT al calcolatore. Per farlo si deve selezionare l'opzione *Aggiungi un dispositivo* dal menu Bluetooth situato in basso a destra della barra delle applicazioni. Viene aperta una finestra (come mostrato in figura 6.11) che mostra tutti i dispositivi Bluetooth rilevati.
A questo punto si deve selezionare il dispositivo NXT desiderato e cliccare il pulsante *Avanti* viene dato avvio alla procedura di "Pairing" per l'associazione NXT-Calcolatore. Sul mattoncino compare una schermata (come riportato in figura 6.12) che richiede l'inserimento di un codice. In particolare,la documentazione prevede che il codice da inserire sia **1234**.
Anche sul calcolatore viene richiesto l'inserimento del codice per il completamento dell'operazione di associazione (come riportato in figura 6.13) a seguito della quale, in assenza di errori, viene terminata la procedura (si veda la figura 6.14).

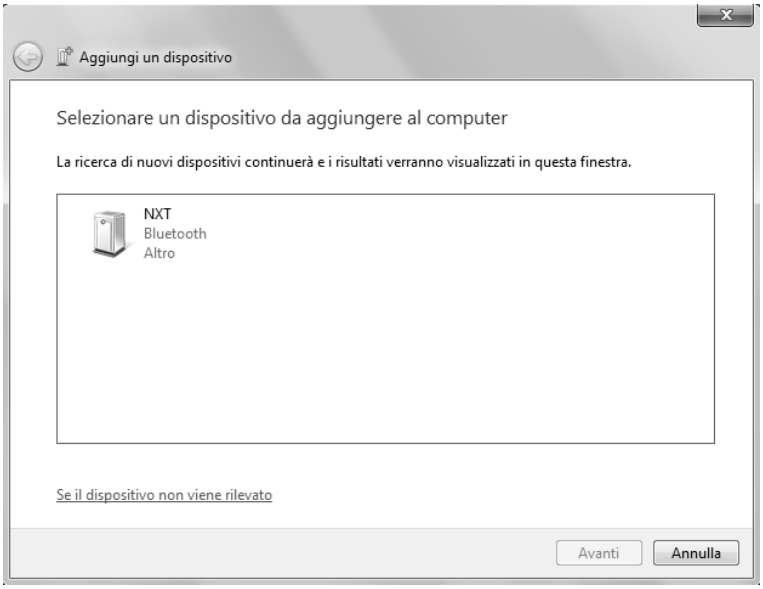

Figura 6.11: Elenco dispositivi Bluetooth rilevati.

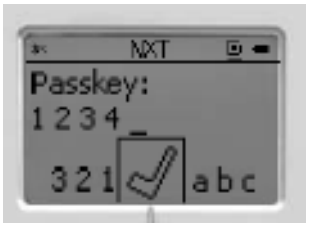

Figura 6.12: Operazione di pairing lato NXT.

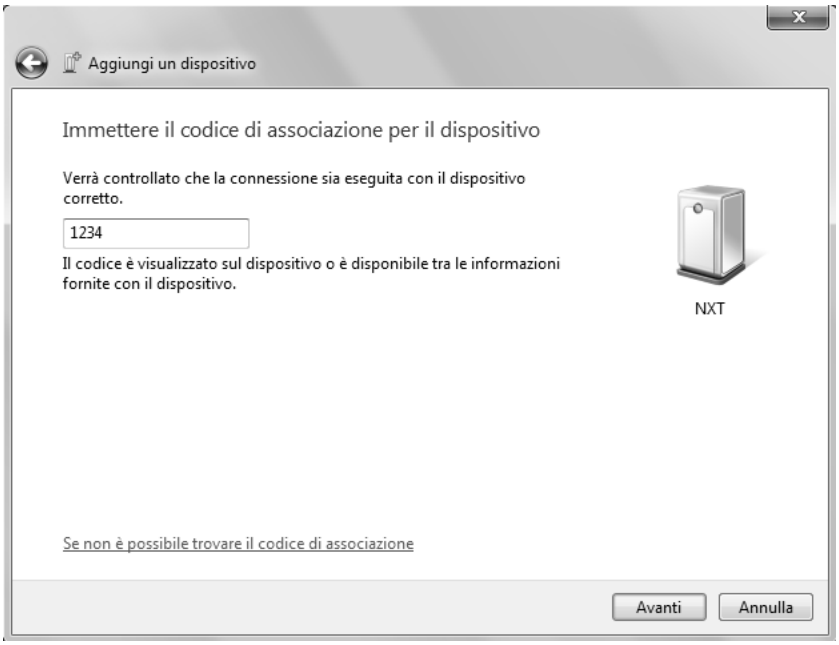

Figura 6.13: Operazione di pairing lato calcolatore.

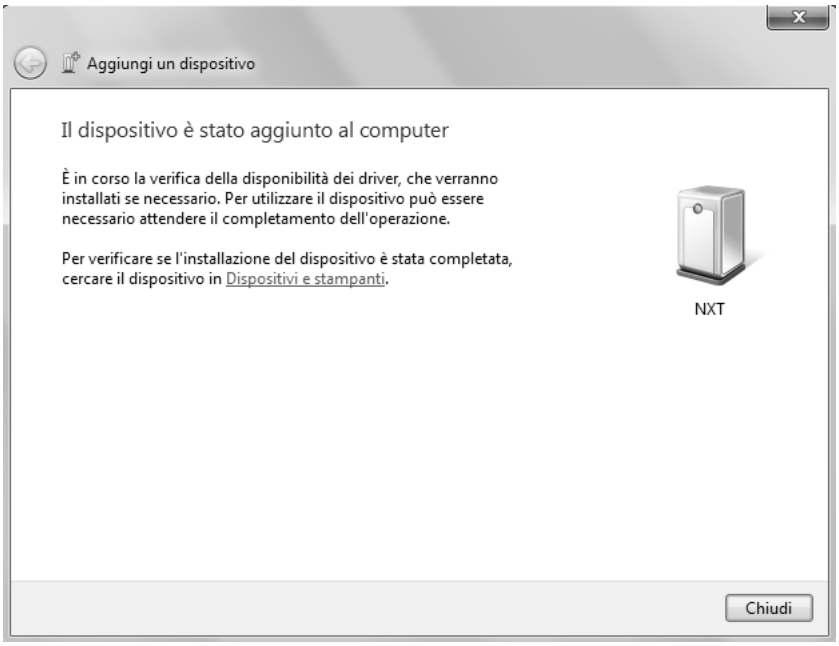

Figura 6.14: Completamento dell'operazione di associazione.

Capitolo 7

Le Applicazioni Web

Per consentire agli utenti di dare vita all'esperienza robotica introdotta nei capitoli precedenti, manca un tassello fondamentale, che si colloca esattamente tra l'utente e il laboratorio didattico Mobolab: l'applicazione web e i suoi servizi. Sono proprio i servizi il vero valore aggiunto di Mobolab, strumenti che mettono nelle mani dell'utente apparecchiature robotiche "reali", con le quali poter finalmente superare i confini di una disciplina che troppo spesso è appannaggio di pochi e viene relegata al mero impiego di simulatori.

7.1 ZK Studio

ZK è un Web Framework open source basato su Ajax, interamente scritto in Java che permette di creare ricche interfacce grafiche per applicazioni web senza la necessità di utilizzare né il linguaggio Javascript, né gli script Ajax.
La parte centrale di ZK è basata su un meccanismo guidato dagli eventi con oltre duecento componenti Ajax e un linguaggio di markup (XUL/XHTML) per il disegno delle interfacce grafiche. ZK è in grado inoltre di integrarsi perfettamente con gran parte delle tecnologie Java EE esistenti sul mercato, in particolare: Spring, Spring Web Flow, JBoss Seam, Enterprise JavaBean, Java Server Page, Java Server Faces e molti altri.

7.1.1 Caratteristiche principali di ZK

L'evoluzione delle applicazioni web, così come vengono intese oggi, ha visto la loro evoluzione da semplici pagine statiche (HTML), a pagine dinamiche (DHTML), a pagine che incorporano tecnologia Ajax (Asynchronous Javascript and XML).
Il punto di forza di Ajax risiede nella possibilità di sviluppare pagine web che abbiano la stessa interattività delle applicazioni Desktop senza l'obbligo di dover interagire con plugin proprietari o con particolari virtual machine. Ajax è fortemente basato su Javascript per la gestione degli eventi lanciati dall'interfaccia grafica e per quanto riguarda la manipolazione dell'interfaccia stessa. Esistono numerosi strumenti per poter integrare questa tecnologia all'interno di un'applicazione web, ma sicuramente il più semplice ed efficace è ZK.
ZK infatti:

- definisce un linguaggio dichiarativo (ZUML) per la realizzazione delle interfacce grafiche che risulta essere molto intuitivo;

- è basato sul lancio di eventi e sull'utilizzo di componenti per consentire alla applicazioni web di disporre dello stesso stile di programmazione usato nella applicazioni Desktop;

- è server-centrico, ovvero tutti i componenti grafici vengono creati sul server rendendone più semplice la comunicazione;

- consente la personalizzazione delle interfacce grafiche attraverso l'uso di fogli di stile (CSS);

- presenta numerosi componenti di terze parti, per esempio: JFreeChart, JasperReports, Google Maps, FCKEditor, Timeline, TimePlot, ExtJS e Dojo;

- consente di scrivere il codice di script che lega la parte business con la parte vista in svariati linguaggi: Java, JRuby, Groovy, Javascript, Phyton, PHP, e molti altri;

- mette a disposizione un ottimo editor WYSIWYG per disegnare e avere una preview delle pagine realizzate;

In linea di massima, i framework per lo sviluppo di applicazioni web sono divisi in due grandi categorie:

- framework che sfruttano la tecnologia Ajax;
- framework che sfruttano le applet come Adobe Flex e Java Applet.

La filosofia di ZK si basa sull'assunzione che le applicazione web debbano sfruttare al meglio la tecnologia Javascript per consentire agli utenti di interagire in maniera rapida senza il ricorso ad una macchina virtuale esterna al browser.
ZK, in particolare, permette agli sviluppatori di programmare direttamente le interfacce grafiche, senza l'utilizzo diretto delle librerie Javascript. Inoltre, i dati vengono visualizzati direttamente dal modello e la vista aggiornata automaticamente ad ogni occorrenza. Quanto appena esposto prende il nome di **Direct RIA** (Rich Internet Application).

In figura 7.1 viene riportata l'architettura di un'applicazione web sviluppata con ZK, il cui flusso di esecuzione viene descritto dalle seguenti fasi:

- l'utente digita l'url ed effettua una richiesta al Webserver che viene servita dal *Loader*;
- il *Loader* carica la pagina specificata e la interpreta per creare correttamente i componenti richiesti;
- il *Loader* presenta il risultato all'interno di una pagina HTML, che viene inviata al browser dell'utente insieme al *Client Engine*;
- il *Client Engine* rimane in ascolto nell'attesa di un evento scatenato dall'utente a seguito del quale viene inviata una richiesta all'*AU Engine*;
- l'*AU Engine* aggiorna, se necessario, il relativo componente;

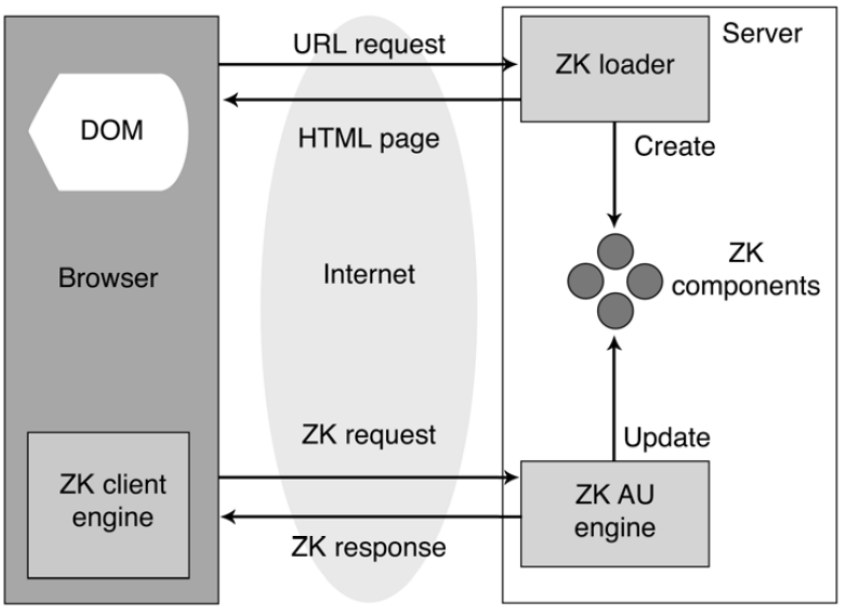

Figura 7.1: L'architettura di ZK.

- se l'applicazione lato server decide di modificare un componente oppure di aggiungerne/rimuoverne altri allora l'*AU Engine* invia tali modifiche al *Client Engine* attraverso delle richieste particolari (le *ZK Responses*);
- le *ZK Responses* servono ad istruire il *Client Engine* e ad aggiornare correttamente il *DOM tree*.

Model View Controller è uno dei pattern architetturali più diffusi nello sviluppo di interfacce grafiche per sistemi software orientati agli oggetti. Il pattern è basato sulla separazione dei compiti tra i componenti software che interpretano tre ruoli principali:

- il **Model** fornisce i metodi per l'accesso ai dati;
- il **View** visualizza i dati contenuti nel Model e gestisce l'interazione con gli utenti;
- il **Controller** riceve i comandi dell'utente e li attua modificando lo stato degli altri due componenti.

In ZK il *Model* rappresenta sia le funzionalità che gli oggetti di business, ovvero la logica applicativa e i relativi oggetti. La parte *View* è un insieme di componenti grafici che compongono la pagina web (il file ZUL) senza la logica di elaborazione degli eventi. La parte *Controller* è una classe Java registrata su un EventListener di uno o più componenti e ha il compito di elaborare gli eventi lanciati dalla parte View.

7.2 L'applicazione Telecomando

Come suggerisce il nome stesso, il telecomando è uno strumento che consente ad un utente il controllo a distanza di un qualche tipo di apparato. Nel caso in questione, l'idea è quella di permettere il controllo remoto dei movimenti dei robot attraverso l'interazione con un'opportuna applicazione web. In questo modo l'utente partecipa ad una manipolazione diretta e realizza un primo livello di esperienza robotica attraverso l'interazione con i robot.
Nel caso in questione, sono state realizzate due applicazioni:

- **Telecomando Lego**: per il controllo del robot con firmware originale Lego;
- **Telecomando Java**: per il controllo del robot con firmware sostitutivo LeJOS.

7.2.1 Il profilo utente

L'applicazione è rivolta in particolare alla categoria degli studenti, anche se più in generale può esercitare attrattiva nei confronti di chiunque desideri intraprendere delle esperienze di robotica educativa.
Innanzitutto è opportuno indagare in che modo l'utente affronta il sistema, analizzandone le seguenti caratteristiche psicologiche:

- lo stile cognitivo è *intuitivo* poiché l'utente tende a lasciarsi guidare dall'intuito nello svolgimento del compito ed è *concreto* dal momento che il compito svolto produce un risultato (la manipolazione diretta del robot);

- l'attitudine dell'utente è *positiva* visto che l'applicazione stimola la curiosità di chi ne fa uso;

- la motivazione è *moderata* dal momento che per effettuare questo tipo di esperienza deve necessariamente utilizzare i servizi offerti dal sistema.

Per quanto concerne le caratteristiche individuali l'utente non deve possedere particolari requisiti di alfabetizzazione dal momento che l'applicazione è sufficientemente autoesplicativa. Non sono richiesti titoli di studio né abilità dattilografiche particolari poiché l'interazione avviene attraverso dei semplici pulsanti. Anche l'alfabetizzazione informatica è bassa poiché è bastevole il solo utilizzo del mouse per l'interazione con l'applicazione.
In generale il livello di esperienza richiesto è basso vista l'intenzionalità di mettere nelle mani dell'utente uno strumento con cui sperimentare.

In relazione alle caratteristiche fisiche, il sistema non è stato studiato per particolari categorie di utenti con precise caratteristiche fisiche e disabilità, tuttavia è rivolto ad una categoria di utenza del tutto generale, senza distinzione di sesso e manualità.

Le caratteristiche di lavoro e dei compiti prevedono che l'utilizzo del sistema sia obbligatorio per interagire con il robot. Per l'utilizzo del sistema non è previsto un addestramento di base, l'utente deve poter interagire in maniera autonoma trovando all'interno dell'applicazione le informazioni di cui ha bisogno. Il compito che si desidera svolgere attraverso l'applicazione non è di per sé importante ai fini dell'applicazione stessa, ma al fine di stimolare nell'utente precise dinamiche in merito all'ambito.

7.2.2 Specifica dei requisiti

L'analisi dei requisiti prevede l'individuazione delle funzionalità che il sistema deve soddisfare. Per quanto riguarda l'applicazione Telecomando, è richiesto che questa sia costituita da un'interfaccia utente intuitiva e semplice, che non necessiti di un apprendimento a priori, ma che risulti essere il più possibile

autoesplicativa. Le funzionalità dell'interfaccia utente, inoltre, devono essere fruibili mediante l'interazione con semplici pulsanti.
Trattandosi di un controllo di movimento, è necessario che l'utente riceva in ogni istante un feedback in relazione ai comandi attuati. Un altro aspetto che non può assolutamente essere trascurato riguarda la visibilità del piano di lavoro. L'utente deve sempre poter vedere ciò che accade sul tavolo. In particolare, le immagini provenienti dalle due telecamere devono avere una dimensione adeguata al fine di consentire un controllo dei movimenti il più possibile ottimale. All'utente deve poi essere consentito l'utilizzo della sensoristica di cui è dotato il robot. In particolare, devono essere forniti sia dei feedback sonori che visivi per quanto riguarda lo stato di attivazione/disattivazione dei sensori e per quanto concerne il loro funzionamento.

7.2.3 Specifica dei casi d'uso

Di seguito vengono presentati i casi d'uso di rilievo, validi sia per l'applicazione Telecomando Lego che per l'applicazione Telecomando Java.

Nome	**Connessione NXT**
Attore	Utente
Scenario principale	1. l'utente accede all'applicazione (pulsante per la connessione abilitato, pulsante per la disconnessione disabilitato) 2. l'utente seleziona il pulsante "Connetti NXT" 3. vengono abilitate la pulsantiera di controllo per i movimenti del robot e la finestra per la gestione dei sensori Fine
Scenario alternativo 1	2a. la connessione non viene stabilita Torna al punto 1
Continua nella prossima pagina	

Nome	**Connessione NXT**
Scenario alternativo 2	2a. l'utente esegue una delle seguenti azioni: chiusura della finestra del browser oppure cambio url 2b. l'utente conferma la scelta effettuata Fine
Scenario alternativo 3	2a. l'utente effettua un'operazione di refresh della pagina web 2b. l'utente conferma la scelta effettuata Torna al punto 1

Tabella 7.1: Connessione NXT.

Nome	**Disconnessione NXT**
Attore	Utente
Scenario principale	1. l'utente invia comandi al robot 2. l'utente seleziona il pulsante di disconnessione 3. la connessione viene abbattuta e sia la pulsantiera di controllo dei movimenti del robot che la finestra di gestione dei sensori vengono disabilitate Fine
Continua nella prossima pagina	

Nome	**Disconnessione NXT**
Scenario alternativo 1	1a. l'utente esegue una delle seguenti azioni: chiusura della finestra del browser oppure cambio url 1b. l'utente conferma la scelta effettuata 1c. la connessione viene abbattuta e sia la pulsantiera di controllo dei movimenti del robot che la finestra di gestione dei sensori vengono disabilitate Fine
Scenario alternativo 2	1a. l'utente effettua un'operazione di refresh della pagina web 1b. l'utente conferma la scelta effettuata 1c. la connessione viene abbattuta e sia la pulsantiera di controllo dei movimenti del robot che la finestra di gestione dei sensori vengono disabilitate Fine

Tabella 7.2: Disconnessione NXT.

Nome	**Controllo NXT**
Attore	Utente
Scenario principale	1. la pulsantiera di controllo del robot e la finestra per la gestione dei sensori sono abilitate 2. viene premuto uno dei pulsanti di controllo Fine
Scenario alternativo 1	2a. l'utente esegue una delle seguenti azioni: chiusura della finestra del browser oppure cambio url 2b. l'utente conferma la scelta effettuata 2c. la connessione viene abbattuta e sia la pulsantiera di controllo dei movimenti del robot che la finestra di gestione dei sensori vengono disabilitate Fine
Scenario alternativo 2	2a. l'utente effettua un'operazione di refresh della pagina web 2b. l'utente conferma la scelta effettuata 2c. la connessione viene abbattuta e sia la pulsantiera di controllo dei movimenti del robot che la finestra di gestione dei sensori vengono disabilitate Fine

Tabella 7.3: Controllo NXT

Nome	**Rientro NXT**
Attore	Utente
Continua nella prossima pagina	

Nome	Rientro NXT
Scenario principale	1. la pulsantiera di controllo del robot e la finestra per la gestione dei sensori sono abilitate 2. l'utente preme il pulsante per il rientro del robot nella stazione di ricarica 3. la connessione viene abbattuta e viene avviata la procedura per il rientro del robot Fine
Scenario alternativo 1	2a. l'utente esegue una delle seguenti azioni: chiusura della finestra del browser oppure cambio url 2b. l'utente conferma la scelta effettuata Fine
Scenario alternativo 2	2a. l'utente effettua un'operazione di refresh della pagina web 2b. l'utente conferma la scelta effettuata Fine
Scenario alternativo 3	2a. l'utente interrompe la procedura di rientro del robot Torna al punto 2

Tabella 7.4: Rientro NXT

7.2.4 Gerarchia delle categorie di pattern

Per consentire di realizzare un'esperienza interattiva di alta qualità si è scelto di utilizzare dei pattern di progettazione. I pattern che si è scelto di prendere in considerazione appartengono alla prima raccolta di Jenifer Tidwell del 1999 intitolata *Common Ground: A Pattern Language for HumanComputer Interface Design*.

I pattern non sono altro che documenti ipertestuali che descrivono artefatti virtuali ed eventi associati cercando di rispondere alle esigenze di usabilità.

In particolare, i pattern proposti dalla Tidwell vengono raggruppati in tre categorie fondamentali:

- **Pattern di contenuto**: modellano la presentazione dell'informazione;
- **Pattern di azione**: modellano l'interazione dell' utente;
- **Pattern di attenzione, layout delle aree di lavoro e navigazione**: consentono all'utente di focalizzare la propria attenzione sul compito da svolgere.

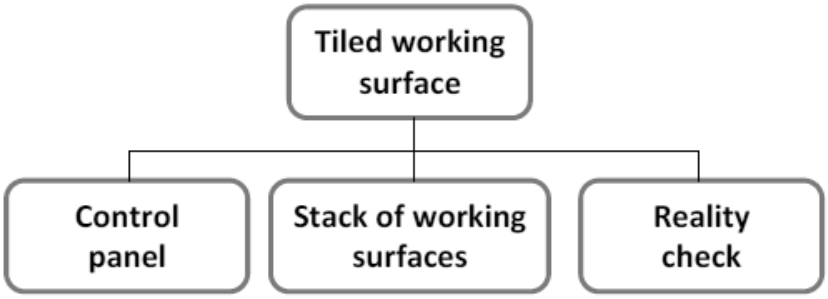

Figura 7.2: La gerarchia dei pattern di alto livello.

Tiled working surface rappresenta l'elemento radice della gerarchia di pattern e prevede che la struttura dell'applicazione sia costituita da diverse aree di lavoro fruibili dall'utente. L'elemento *Control panel* identifica il ruolo di controllo svolto dalla pulsantiera con cui vengono governate le azioni e i movimenti del robot. L'elemento *Stack of working surfaces* rappresenta il pannello di modifica del comportamento del robot e dell'attivazione/disattivazione della sensoristica dello stesso organizzato mediante una struttura a tab. Per quanto concerne l'elemento *Reality check* al fine di prevenire azioni non desiderate vengono attivati opportuni controlli da parte dell'applicazione.
Progress indicator viene utilizzato per indicare lo stato di avanzamento di un'-operazione (per esempio la connessione/ disconnessione del robot). Il pattern *Disabled irrelevant things* concorre a rendere inattive alcune funzionalità in determinati stati dell'applicazione per evitare comportamenti inattesi da parte

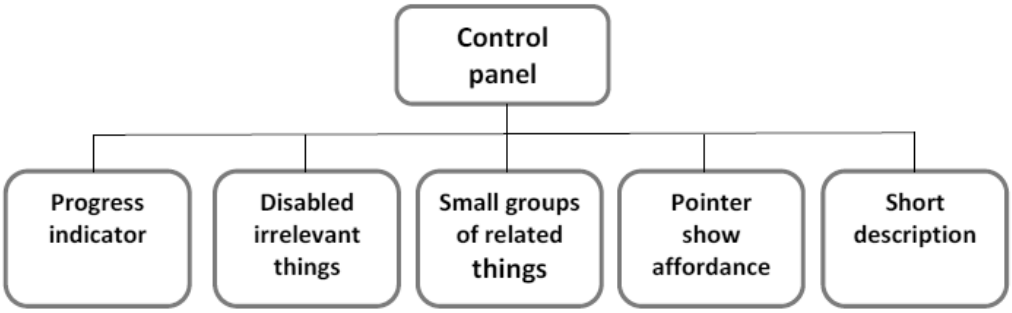

Figura 7.3: La gerarchia dei pattern di medio livello.

del robot. Anche il raggruppamento dei pulsanti segue la filosofia di un pattern, *Small groups of related things* con il fine di dividere la funzionalità in precise categorie e facilitare la fruizione da parte dell'utente. Infine si evidenzia l'impiego dei pattern *Pointer show affordance* che induce un cambiamento del cursore a seconda dell'area di lavoro sulla quale si opera e *Short description* che mostra, a seconda della funzionalità desiderata, una finestra contenente una breve descrizione in merito alla funzionalità stessa.

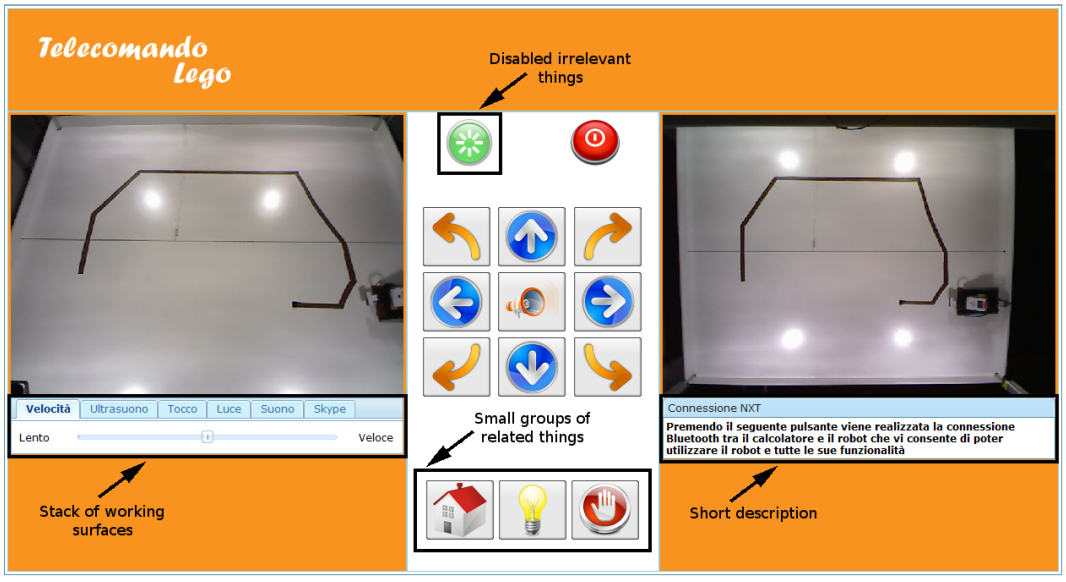

Figura 7.4: Alcuni dei pattern implementati nell'applicazione Telecomando.

7.2.5 Implementazione Telecomando Lego

L'installazione del software proprietario NXT-G determina l'installazione della libreria **Lego Fantom** utilizzata dall'applicativo per il controllo del robot.
L'intenzione è quella di costruire un'applicazione in linguaggio Java che faccia uso delle funzioni contenute in tale libreria. Per realizzare quanto espresso bisogna dichiarare un'interfaccia Java che crei il mapping tra le funzioni Lego della libreria Fantom e il linguaggio Java. Lo strumento che consente di realizzare quanto appena espresso è la **Java Native Access** (JNA), una libreria che permette di effettuare il mapping di tutte le risorse presenti all'interno di una libreria nativa.
Java Native Access utilizza una piccola libreria stub che si occupa di invocare dinamicamente il codice nativo.
Per poter usufruire delle funzioni contenute nella libreria Fantom Lego bisogna dichiarare un'apposita interfaccia che estenda l'interfaccia *StdCallLibrary*.

```
public interface Fantom extends StdCallLibrary {

  Fantom ISTANZA = (Fantom) Native.loadLibrary("fantom",
      Fantom.class);

  Pointer nFANTOM100_createNXT(String risorsa, Stato stato,
      boolean versioneFirmware);

  void nFANTOM100_destroyNXT(Pointer pNXT, Stato stato);

  int nFANTOM100_iNXT_sendDirectCommand(Pointer pNXT,
      boolean richiestaRisposta, Buffer bufferComando, int
      dimBufferComando, byte[] bufferRisposta, int
      dimBufferRisposta, Stato stato);
}
```

Il passo immediatamente successivo consiste nel creare un'istanza della libreria

nativa mediante il metodo *Native.loadLibrary()*. All'interno dell'interfaccia vengono poi definiti i seguenti metodi:

- **nFANTOM100_createNXT**: crea un oggetto che rappresenta lo specifico NXT;
- **nFANTOM100_destroyNXT**: elimina un oggetto NXT;
- **nFANTOM100_iNXT_sendDirectCommand**: invia un comando diretto allo specifico NXT;

A questo punto è stata realizzata la classe **NXT** che definisce l'oggetto NXT e i metodi che agiscono su quest'ultimo. Nel riquadro in basso sono riportati i metodi che realizzano la connessione e la disconnessione Bluetooth di uno specifico dispositivo NXT. La connessione riceve, tra gli argomenti di ingresso, una particolare stringa contenente l'indirizzo Bluetooth del robot. La sintassi **BTH** indica che si tratta di una connessione Bluetooth, **QUI** rappresenta il nome del dispositivo, **00:16:53:0B:5D:2E** l'indirizzo Bluetooth del robot e **4** il numero della porta COM sulla quale è collegato.

```
private Pointer connessioneDiretta() throws
    UnableToCreateNXTException {
     s = new Stato();
     Pointer iNXT = fantom.nFANTOM100_createNXT("BTH::QUI
         ::00:16:53:0B:5D:2E::4", s, false);
     if (Stato.Stati.SUCCESS.equals(s.getStatus())) {
       return iNXT;
     } else {
       throw new UnableToCreateNXTException("Non è possibile
           creare la connessione!");
     }
}

public void disconnettiLego()
{
     fantom.nFANTOM100_destroyNXT(nxtPuntatore, s);
```

```
}
```

I comandi diretti che si è scelto di implementare sono i seguenti: STARTPROGRAM, STOPPROGRAM e MESSAGEWRITE.
Tutti e tre i metodi inviano tramite protocollo Bluetooth specifiche sequenze di byte tramite la funzione di libreria *sendDirectCommand*. I metodi *avviaProgramma()* e *arrestaProgramma()* servono per lanciare o fermare l'esecuzione di uno specifico programma residente sul brick. Il metodo *inviaComando()* si occupa invece di inviare particolari numeri interi all'interno di una determinata mailbox.

```
public void avviaProgramma(String nomeFile) {
    Stato stato = new Stato();
    byte[] dimNomeFile= nomeFile.getBytes();
    ByteBuffer comando= ByteBuffer.allocate(dimNomeFile.
        length+1+1);
    comando.put((byte)0x00);
    comando.put(dimNomeFile);
    comando.put((byte)0x00);
    fantom.nFANTOM100_iNXT_sendDirectCommand(nxtPuntatore,
        false, comando, comando.capacity(), null, 0, stato); }

public void arrestaProgramma() {
    ByteBuffer comando = ByteBuffer.allocate(1);
    comando.put((byte)0x01);
    fantom.nFANTOM100_iNXT_sendDirectCommand(nxtPuntatore,
        false, comando, 1, null, 0, new Stato()); }

public void eseguiComando(int mbox, int com) {
  Stato stato = new Stato();
  ByteBuffer comando = ByteBuffer.allocate(8);
  comando.put((byte)0x09);
  comando.put((byte)mbox);
```

```
    comando.put((byte)0x05);
    comando.put((byte)com);
    comando.put((byte)0x00);
    comando.put((byte)0x00);
    comando.put((byte)0x00);
    comando.put((byte)0x00);
    fantom.nFANTOM100_iNXT_sendDirectCommand(nxtPuntatore,
        false, comando, comando.capacity(), null, 0, stato); }
```

Tali numeri vengono letti da un programma residente sull'NXT, che li interpreta e provvede ad eseguire dei comandi specifici. Quanto appena esposto realizza la parte *Model* del pattern MVC. L'interfaccia grafica dell'applicazione Telecomando Lego viene riportata in figura 7.5.

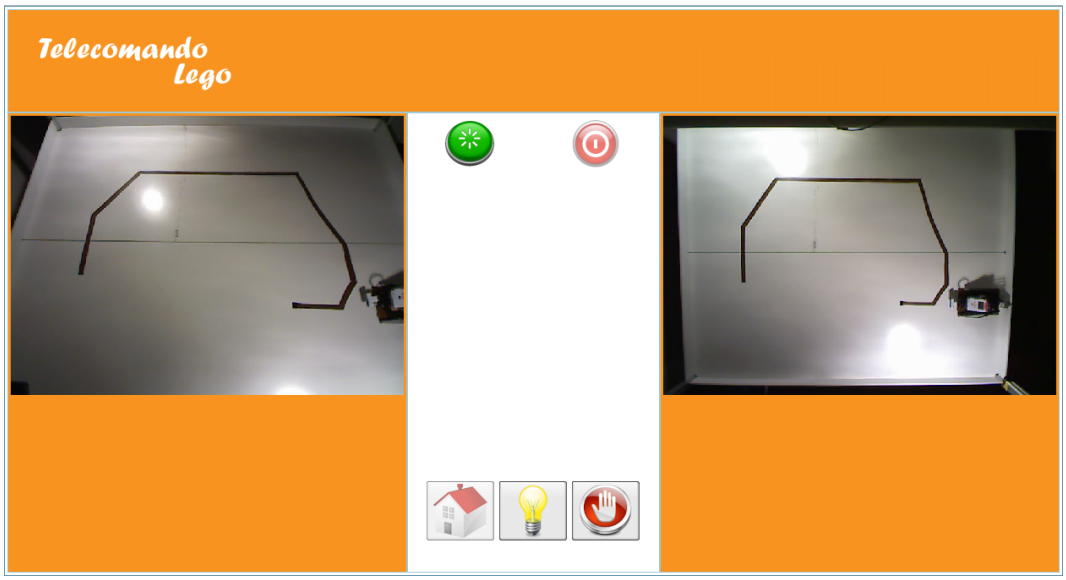

Figura 7.5: L'interfaccia dell'applicazione Telecomando Lego.

Per la realizzazione della parte *View* è stato creato un apposito file index.zul la cui struttura generale viene riportata di seguito.

```
<zk>
  <window>
```

```
        <borderlayout>
                <north/>
                <east/>
                <center/>
                <west/>
                <south/>
        </borderlayout>
    </window>
</zk>
```

Per quanto riguarda la disposizione dei componenti all'interno della finestra è stata adottata una struttura a borderlayout. In alto è stato posizionato l'header contenente il nome dell'applicazione, al centro la pulsantiera per il controllo dei movimenti del robot, a destra l'applet per lo stream delle immagini provenienti dalla seconda telecamera, a sinistra l'applet per lo stream delle immagini provenienti dalla prima telecamera ed un componente a schede per la gestione della velocità e dei sensori del robot.

Per quanto riguarda invece la parte *Controller* è stata creata una classe **MyController** della quale si riportano i metodi più significativi nel riquadro inferiore.

```
public class MyController extends GenericAutowireComposer {

protected Slider velocità, distanza;
protected Button connetti, disconnetti;
protected Radio onUltra, offUltra;
protected Radio onTocco, offTocco;
protected Radio onLuce, offLuce;
protected Label vicino, lontano;
protected Grid pulsantiera;
protected Tabbox parametri;

public void onDisconnettiNXT(Event onClick) {
```

```
  MyDesktopCleanup.lego.arrestaProgramma();
  MyDesktopCleanup.lego.disconnettiLego();
  abilitaComponenti(); }

public void onConnettiNXT(Event onClick) throws
   NXTNotFoundException, UnableToCreateNXTException{
  MyDesktopCleanup.lego = new NXT();
  MyDesktopCleanup.lego.avviaProgramma("TestNXT.rxe");
  disabilitaComponenti(); }
```

La classe MyController implementa l'interfaccia *org.zkoss.zk.ui.util.Composer*. In particolare, la classe MyController estende l'interfaccia *GenericAutowireComposer* che permette di accedere in modo trasparente a tutti i componenti figli della window. Ogni componente ZUML corrisponde proprio ad una classe Java figlia di Component. Per legare l'evento clic del pulsante al metodo che realizza la connessione Bluetooth tra il robot e il calcolatore è stato sufficiente fare il **forward** dell'evento *onClick* al metodo *onConnettiNXT*, analogamente per l'evento clic del pulsante di disconnessione. In questo modo la vista dell'applicazione contiene solo la parte grafica senza il codice per la gestione degli eventi, il quale viene più propriamente collocato nella parte di controllo. Nel riquadro sottostante è riportata una parte del file *index.zul*, in particolare la costruzione dei pulsanti per la connessione/disconnessione dell'NXT al calcolatore.

```
<center>
  <borderlayout>
    <east width="50%" border="none">
    <div align="center">
    <button id="disconnetti" sclass="my-button" tooltiptext=
        "Disconnetti NXT" image="immagini/offDown.png"
        forward="onClick=onDisconnettiNXT" disabled="true" />
    </div>
    </east>
```

```
    <west width="50%" border="none">
    <div align="center">
    <button id="connetti" sclass="my-button" tooltiptext="
        Connetti NXT" image="immagini/on.png" forward="
        onClick=onConnettiNXT" />
    </div>
    </west>
<center border="none" />
```

7.2.6 Implementazione Telecomando Java

Il Telecomando Java, a differenza dell'applicazione appena discussa, non ha richiesto la dichiarazione di un'interfaccia dal momento che il firmware LeJOS viene programmato direttamente in linguaggio Java.
In prima battuta è stata realizzata la classe **NXT** contenente i metodi per la connessione/disconnessione del robot al calcolatore tramite connessione Bluetooth. Il metodo *connettiNXT()* si occupa di eseguire un file batch residente sul calcolatore Windows. In particolare, il file batch manda in esecuzione uno specifico programma residente sull'NXT, il quale resta in attesa di una connessione Bluetooth. Una volta stabilita la connessione, il robot apre uno stream di dati in input per la ricezione dei comandi, mentre l'applicazione web apre uno stream di dati in output per l'invio dei comandi selezionati dall'utente. In sostanza, il programma che gira sull'NXT riceve i comandi inviati dall'utente ed esegue le azioni corrispondenti.

```
public int connettiNXT() throws InterruptedException {
  try {
        Process p = Runtime.getRuntime().exec(new String[]{"
            cmd", "/c", percorso});
        p.waitFor();
        nxtComm = NXTCommFactory.createNXTComm(
            NXTCommFactory.BLUETOOTH);
```

```
            nxtInfo = new NXTInfo(NXTCommFactory.BLUETOOTH, "QUA
                ", "00:16:53:0C:F2:18");
            nxtComm.open(nxtInfo);
            dos = new DataOutputStream(nxtComm.getOutputStream()
                );
            return 0;

           } catch (Exception e) {
            return 1;
        }
}
```

```
public void disconnettiNXT() throws InterruptedException {
  int disconnetti = 9;
  try {
            dos.writeInt(disconnetti);
            dos.close();
            nxtComm.close();

           } catch (Exception e) {

        }
}
```

L'interfaccia grafica dell'applicazione Telecomando Java viene riportata in figura 7.6. Come si può notare la vista è identica a quella dell'applicazione Telecomando Lego.
Per quanto riguarda la parte di controllo, anche in questo caso è stata realizzata un'apposita classe **MyController** della quale vengono presentati i metodi più significativi.

```
public class MyController extends GenericAutowireComposer {

protected Slider velocità, distanza;
```

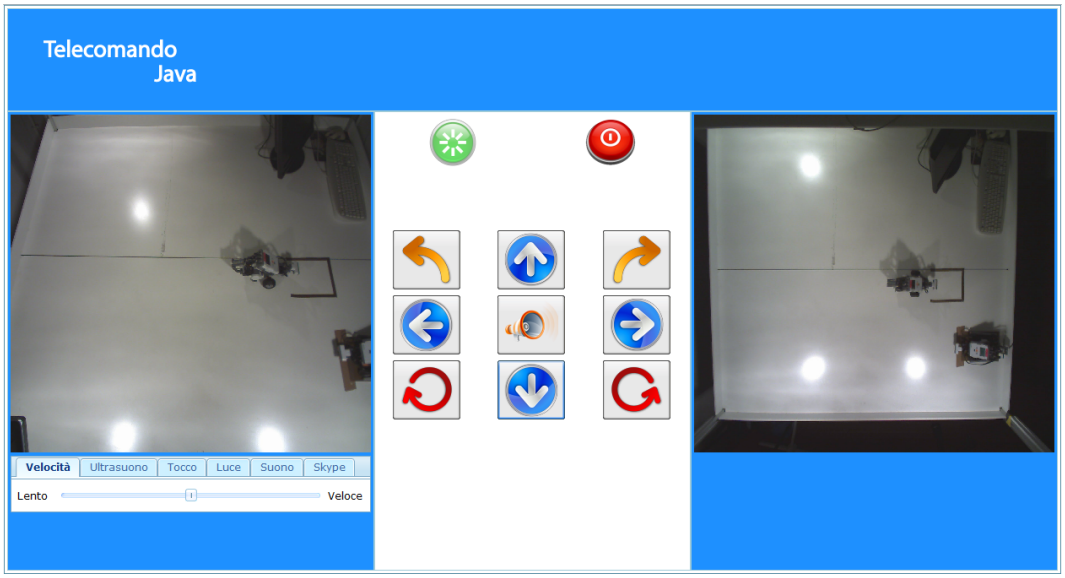

Figura 7.6: L'interfaccia dell'applicazione Telecomando Java.

```
protected Button connetti, disconnetti;
protected Radio onUltra, offUltra;
protected Radio onTocco, offTocco;
protected Radio onLuce, offLuce;
protected Label vicino, lontano;
protected Grid pulsantiera;
protected Tabbox parametri;

public void onDisconnettiNXT(Event onClick) throws
    InterruptedException {
  MyDesktopCleanup.nxt.disconnettiNXT();
  disabilitaComponenti();
}

public void onConnettiNXT(Event onClick) throws
    InterruptedException {
  int i = MyDesktopCleanup.nxt.connettiNXT();
  if(i == 0){
    abilitaComponenti();
```

```
    }
}
```

7.3 L'applicazione BeeBot

Per poter introdurre la robotica nella scuola fin dall'infanzia servono strumenti la cui caratteristica principale sia l'estrema semplicità d'uso. Con il termine Beebot si identifica una piccola ape di plastica, sulla cui schiena sono collocati dei tasti funzione, tramite i quali attivare semplici movimenti (figura 7.7). Il passo in avanti o indietro è fisso e misura 150 mm mentre le rotazioni a destra e a sinistra sono di 90° esatti. La potenzialità di questo oggetto risiede proprio nella sua programmazione, dal momento che non è richiesta la conoscenza di hardware o software particolari, ma è sufficiente utilizzare i tasti posizionati sul suo dorso.

Figura 7.7: Il Beebot.

Quando si parla di programmazione Beebot spesso vengono impiegati dei cartelloni animati, suddivisi in quadrati di circa 150 mm, con l'obiettivo di proporre ambientazioni particolari all'interno delle quali far muovere il robot. I bambini imparano quindi a programmare i movimenti dell'ape attraverso i tasti, per fare in modo che si muova sul cartellone secondo un percorso stabilito. Spesso il Beebot viene posizionato su una casella d'inizio e sul cartellone vengono appositamente inseriti degli ostacoli. Al bambino viene richiesto di impostare un percorso in nel quale l'ape deve raggiungere una casella d'arrivo evitando tali ostacoli.
L'impiego di uno strumento di questo calibro, permette di sviluppare sia la

logica che la percezione spaziale. Il bambino viene chiamato a mettere in atto strategie risolutive, confrontandosi in modo diretto con i concetti di aggiunta e diminuzione delle azioni che possono essere svolte dal robot. La vera forza del Beebot viene ben riassunta dalla seguente frase di Perticari: *il corpo vive le esperienze della mente perché il bambino pensa, agisce per programmare ed esegue con il suo corpo le operazioni, poi riflette e, con la mente e con il linguaggio, opera il confronto tra previsione e ciò che accade veramente.*

Sulla base delle considerazioni appena esposte, è stata prevista la realizzazione di due applicazioni web:

- **Beebot Lego**: per la programmazione Beebot del robot con firmware Lego;
- **Beebot Java**: per la programmazione Beebot del robot con firmware LeJOS.

7.3.1 Specifica dei requisiti

L'applicazione Beebot è rivolta soprattutto alla scuola dell'infanzia e pertanto uno dei requisiti fondamentali impone la realizzazione di un'interfaccia il più possibile semplice e intuitiva. L'utente deve poter interagire con uno strumento di programmazione (una pulsantiera) che ricordi il più possibile il dorso dell'ape Beebot. I pulsanti consentono all'utente di concretizzare un percorso da un punto di partenza ad un potenziale punto di arrivo. L'utente deve poter disporre di uno strumento grafico che lo supporti nella specifica del percorso (una sorta di simulatore). Un altro requisito fondamentale consiste nel mostrare in ogni momento l'esecuzione reale del cammino. L'utente deve vedere i movimenti del robot per verificare che il percorso impostato venga eseguito correttamente.

7.3.2 Specifica dei casi d'uso

Di seguito si riportano i casi d'uso di rilievo, che valgono sia per l'applicazione Beebot Lego che per l'applicazione Beebot Java.

Nome	Programmazione Beebot
Attore	Utente
Scenario principale	1. l'utente accede all'applicazione 2. l'utente imposta un percorso utilizzando l'apposita pulsantiera 3. l'utente clicca il pulsante "Invia" Fine
Scenario alternativo 1	2a. l'utente cancella il percorso impostato Torna al punto 1
Scenario alternativo 2	3a. l'utente effettua un'operazione di refresh della pagina web Torna al punto 1
Scenario alternativo 3	3a. l'utente effettua un'operazione di cambio url oppure chiude la finestra del browser Fine
Scenario alternativo 4	2a. l'utente cambia lo sfondo del simulatore Torna al punto 1

Tabella 7.5: Programmazione del Beebot.

7.3.3 Implementazione Beebot Lego

L'implementazione del Beebot Lego ha previsto la realizzazione di un'apposita classe **MyController**. Quando l'utente crea un percorso, per farlo utilizza i pulsanti AVANTI, INDIETRO, SINISTRA, DESTRA. Ogni passo del percorso viene memorizzato nel vettore *comandiInviati* come si vede dal listato di codice in basso. Il vettore contiene, in ogni posizione occupata, una stringa di testo che rappresenta l'azione che si desidera far eseguire al robot.

```
public void onScriviAvanti(Event onClick){
  indietro.setDisabled(false);
  if(!controllaAvanti()){
    avanti.setDisabled(true);
  } else {
    avanti.setDisabled(false);
    controllaRotazione("AVANTI");
    comandiInviati.add("AVANTI ");
    setBottone(getCasella(), r);
    contatore = contatore + 1;
    onSetPercorso(); }
}

public void onScriviIndietro(Event onClick){
  avanti.setDisabled(false);
  if(!controllaIndietro()){
    indietro.setDisabled(true);
  } else {
    indietro.setDisabled(false);
    controllaRotazione("INDIETRO");
    comandiInviati.add("INDIETRO ");
    setBottone(getCasella(), r);
    contatore = contatore + 1;
    onSetPercorso(); }
}
```

Per prevenire che l'utente crei dei percorsi non validi, sono stati predisposti opportuni controlli che provvedono a disabilitare i pulsanti qualora l'utente raggiunga una condizione tale per cui il successivo passo di computazione conduca il robot al di fuori dell'area di lavoro. Una volta che l'utente ha completato l'inserimento del percorso, viene fatto scorrere il vettore *comandiInviati* e il contenuto di tutti gli elementi viene scritto in un apposito file di testo (Percorso.txt). A questo punto viene invocato il metodo *leggiFile()* che si occupa di leggere le righe del file Percorso.txt e di memorizzarle all'interno del vettore

comandiRicevuti.

```
public void onScriviFile(Event onClick){
  int i;
  String temp = "";
  for(i=0; i < comandiInviati.size(); i++){
    temp = temp + comandiInviati.elementAt(i); }
  temp = temp + "0";
  try {
     FileOutputStream file = new FileOutputStream("C:\\
        Mobolab\\BeeBotLego\\Percorso.txt");
     PrintStream Output = new PrintStream(file);
     Output.println(temp);
     file.close();

    } catch (Exception e) {}
     pulsantiDisattivati();
     leggiFile(); }

public void leggiFile(){
  try {
     FileReader file = new FileReader("C:\\Mobolab\\
        BeeBotLego\\Percorso.txt");
     BufferedReader in = new BufferedReader(file);
     String line;
     do{
        line = in.readLine();
        if(line != null)
          comandiRicevuti.add(line);
        } while(line != null);
     file.close();

    } catch (Exception e){}
     parsingLine(); }
```

Successivamente viene fatto scorrere il vettore *comandiRicevuti* e per ogni elemento memorizzato viene fatto il parsing della stringa contenuta, estraendo ogni singolo comando e memorizzandolo all'interno del vettore *comandi*, per poi mappare gli elementi del suddetto vettore in vere e proprie stringhe di codice da inserire all'interno di un file .nxc (il file Prog.nxc).
Ad ogni comando (AVANTI, INDIETRO, SINISTRA, DESTRA) viene associata la funzione NXC *RotateMotorEx()* che si occupa di far ruotare in maniera opportuna i motori dell'NXT in relazione ai parametri d'ingresso che le vengono passati.
Nel riquadro sottostante viene riportato il metodo Java che si occupa di associare tale funzione ad ogni passo del percorso impostato dall'utente. Tutte queste funzioni vengono inserite all'interno del metodo *main()* del programma per consentirne l'esecuzione da parte dell'NXT.

```
public String interpretaComando(String tmp){
  String cmd = "";
  if(tmp.contains("AVANTI")){
    cmd = "RotateMotorEx(OUT_BC,45,400,0,true,true);"; }
  if(tmp.contains("INDIETRO")){
    cmd = "RotateMotorEx(OUT_BC,45,-400,0,true,true);"; }
  if(tmp.contains("SINISTRA")){
    cmd = "RotateMotorEx(OUT_BC,45,180,-100,true,true);"; }
  if(tmp.contains("DESTRA")){
    cmd = "RotateMotorEx(OUT_BC,45,180,100,true,true);";}
  return cmd;
}
```

Una volta creato il programma è necessario compilarlo, caricarlo sul brick ed eseguirlo. Quest'ultima azione viene realizzata da un apposito file batch che provvede a chiamare il compilatore nbc e a produrre un file .rxe che viene caricato ed eseguito direttamente sul brick.

La realizzazione dell'interfaccia grafica (figura 7.8) ha previsto la creazione di un file index.zul. Analogamente alle applicazioni Telecomando Java e Telecomando Lego, anche in questo caso si è scelto di adottare una struttura a borderlayout. Nella parte superiore è stato collocato l'header contenente il nome dell'applicazione, mentre nella parte sinistra è stata collocata l'applet che mostra le immagini provenienti dalla seconda telecamera. Nella parte destra è stato inserito il simulatore che fornisce una rappresentazione grafica del percorso impostato dall'utente. Si tratta di una griglia divisa in quadranti sulla quale viene impressa la rappresentazione del cammino di un'ape da una casella di partenza ad una casella di arrivo. Nella parte centrale è stata collocata la pulsantiera per la programmazione del percorso ed un'area di testo nella quale viene fornita una rappresentazione testuale dei passi selezionati dall'utente. Sempre nell'area centrale sono presenti due pulsanti. Il pulsante di sinistra consente di eliminare un percorso in fase di costruzione, nell'eventualità l'utente si renda conto di aver commesso degli errori. Il pulsante di destra modifica invece lo sfondo del simulatore in modo da rendere più attraente l'ambientazione.

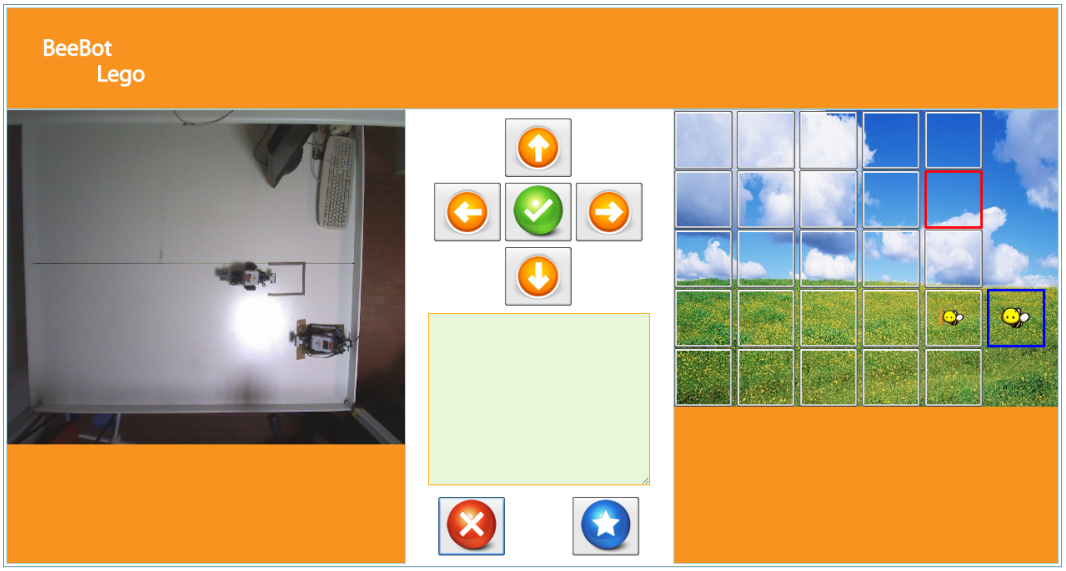

Figura 7.8: L'interfaccia dell'applicazione Beebot Lego.

7.3.4 Implementazione Beebot Java

L'implementazione dell'applicazione Beebot Java è analoga a quella del Beebot Lego. Anche in questo caso è presente una classe **MyController** che dispone di metodi analoghi a quelli dell'applicazione Lego. L'unica differenza risiede nel mapping dei comandi e nella creazione del programma da eseguire. In questo caso il programma che viene creato, compilato, caricato sul brick ed eseguito è a tutti gli effetti un programma Java. Pertanto i comandi di movimento devono essere mappati in opportuni metodi Java e il percorso del robot inserito all'interno di una classe (in questo caso la classe **Beebot**) nella quale deve essere presente il metodo *main()* per l'esecuzione del programma.
In questo caso, ad ogni comando di movimento rettilineo (AVANTI, INDIETRO) viene associato il metodo LeJOS *travel()*, mentre a ciascun comando di rotazione (SINISTRA, DESTRA) viene associato il metodo *rotate()*.

```
public String interpretaComando(String tmp){
  String cmd = "";
  if(tmp.contains("AVANTI")){
    cmd = " tn.travel(20);";   }
  if(tmp.contains("INDIETRO")){
    cmd = " tn.travel(-20);"; }
  if(tmp.contains("SINISTRA")){
    cmd = " tn.rotate(-90);"; }
  if(tmp.contains("DESTRA")){
    cmd = " tn.rotate(90);";   }
  return cmd;
}
```

```
public void mappingComandi(){
 int i;
 String cmd;
 String main =
 "import lejos.nxt.Motor;" +
 "\n import lejos.robotics.navigation.SimpleNavigator;" +
 "\n import lejos.robotics.navigation.TachoPilot;" +
 "\n\n public class Beebot {" +
 "\n\n public static TachoPilot pilot;" +
 "\n public static SimpleNavigator tn;" +
 "\n\n public static void main(String [] args){" +
 "\n\n pilot = new TachoPilot(5.7f,12.2f,Motor.B,Motor.C);"+
 "\n tn = new SimpleNavigator(pilot);" +
 "\n tn.setTurnSpeed(30);" +
 "\n tn.setMoveSpeed(5);" +
 "\n tn.travel(20);";

  try {
     FileOutputStream file = new FileOutputStream("C:\\
        Mobolab\\BeeBotJava\\Beebot.java");
     PrintStream Output = new PrintStream(file);
     Output.println(main);

     for(i=0; i < comandi.size(); i++){
        cmd = interpretaComando(comandi.elementAt(i));
        Output.println(cmd);
     }

     Output.println("\n\n}\n\n}");
     file.close();

     } catch (Exception e) {}
     convertiFile();
}
```

L'interfaccia grafica dell'applicazione Beebot Java (riportata in figura 7.9) è analoga a quella dell'applicazione Beebot Lego.

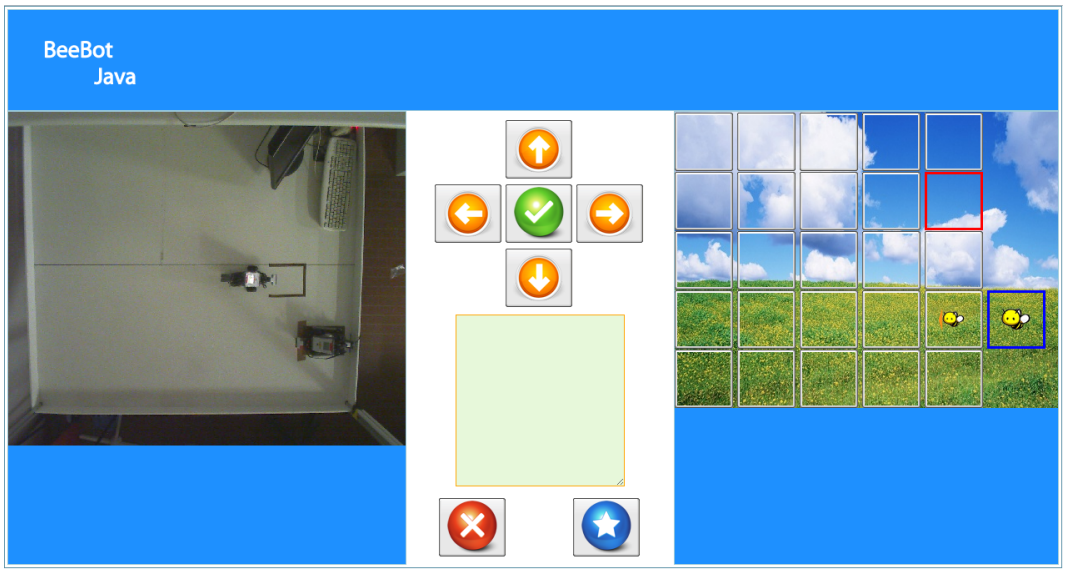

Figura 7.9: L'interfaccia dell'applicazione Beebot Java.

7.4 L'applicazione Programma Lego

L'applicazione Programma Lego è stata sviluppata con l'obiettivo di consentire all'utente la sperimentazione di un livello più avanzato di programmazione dell'NXT. Dopo aver interagito con le applicazioni Telecomando e Beebot, nasce il desiderio di spingersi oltre e realizzare dei comportamenti più complessi che possono essere ottenuti solo attraverso una programmazione testuale del brick. Nel caso in questione è stato preso in considerazione il linguaggio NXC per la programmazione specifica del robot con firmware Lego.

7.4.1 Specifica dei requisiti

Innanzitutto, l'applicazione deve mettere a disposizione uno spazio adeguato per la stesura dei programmi in linguaggio NXC. Ciò consente di non porre vincoli in relazione alla lunghezza del codice e favorisce lo sviluppo di programmi anche molto articolati. L'utente deve ricevere informazioni dall'applicazione, in particolare nell'eventualità si verifichino errori durante la fase di compilazione. A tutela della sicurezza dei robot e del piano di lavoro, deve essere sempre possibile interrompere l'esecuzione di un programma, dal momento che la programmazione stessa può dar luogo a situazioni inattese, come per esempio dei cicli. L'utente deve poter vedere l'esito della propria programmazione, in particolare come per le altre applicazioni deve avere una visione adeguata del piano di lavoro e del robot.

7.4.2 Specifica dei casi d'uso

Di seguito si riportano i casi d'uso di rilievo dell'applicazione Programma Lego.

Nome	**Esecuzione di un programma d'esempio**
Attore	Utente
Scenario principale	1. l'utente accede all'applicazione 2. l'utente seleziona un programma d'esempio dalla finestra "Programmi di esempio" 3. il programma viene caricato nell'area di testo adibita alla scrittura di codice 4. l'utente clicca il pulsante "Esegui programma" Fine
Scenario alternativo 1	4a. l'utente effettua un'operazione di refresh della pagina Torna al punto 1
Scenario alternativo 2	4a. l'utente effettua una delle seguenti operazioni: cambio url o chiusura della finestra del browser Fine
Scenario alternativo 3	3a. l'utente seleziona il pulsante "Cancella tutto" Torna al punto 1
Scenario alternativo 4	4a. l'utente preme il pulsante "Arresta programma" Torna al punto 1

Tabella 7.6: Esecuzione di un programma d'esempio.

Nome	Creazione di un nuovo programma
Attore	Utente
Scenario principale	1. l'utente accede all'applicazione 2. l'utente scrive del codice all'interno dell'area di testo 3. l'utente clicca il pulsante "Esegui programma" Fine
Scenario alternativo 1	3a. l'utente effettua un'operazione di refresh della pagina Torna al punto 1
Scenario alternativo 2	3a. l'utente effettua una delle seguenti operazioni: cambio url o chiusura della finestra del browser Fine
Scenario alternativo 3	3a. l'utente preme il pulsante "Arresta programma" Torna al punto 1

Tabella 7.7: Creazione di un nuovo programma

7.4.3 Implementazione Programma Lego

L'implementazione dell'applicazione Programma Lego ha previsto la realizzazione di una classe **MyController** della quale vengono riportati i metodi più significativi nel listato di codice in basso. I programmi d'esempio sono contenuti all'interno della cartella ProgLego. Se l'utente decide di caricare un programma d'esempio, allora viene preso il contenuto del file relativo e quest'ultimo viene ricopiato all'interno dell'area di testo adibita alla stesura del codice NXC.

In alternativa, come già esposto in precedenza, l'utente può creare un programma da zero scrivendo il codice direttamente nella relativa area di testo. Prima di effettuare la procedura di compilazione è necessario che il testo contenuto all'interno dell'area venga salvato in un opportuno file .nxc. Questo passaggio

viene effettuato dal metodo *onScriviFile()*.

```
public void onLeggiEsempio(Event onClick){
  String nomeFile = programmi.getSelectedItem().getLabel();
  String line;
  String contenuto = "";
  try {
     FileReader file = new FileReader("C:\\Mobolab\\ProgLego
        \\"+nomeFile+".nxc");
     BufferedReader in = new BufferedReader(file);
       do{
        line = in.readLine();
          if(line != null){
          contenuto = contenuto + line + "\n";
          }
        } while(line != null);

        file.close();
        programma.setText(contenuto);

    } catch (Exception e){} }

public void onScriviFile(Event onClick){
  try {
       FileOutputStream file = new FileOutputStream("C:\\
          Mobolab\\ProgLego\\ProgrammaNXC.nxc");
       PrintStream Output = new PrintStream(file);
       Output.println(programma.getText());
       file.close();

    } catch (Exception e) {}
    compilaProgramma();
}
```

La compilazione, il caricamento sul brick, l'esecuzione e l'arresto del programma vengono gestiti da opportuni file batch. La fase di compilazione, in particolare, produce un file di testo contenente l'esito della compilazione. Se sono stati riscontrati degli errori, questi vengono scritti all'interno del file (compilazione.txt) e mostrati a video. Il metodo *onLeggiEsito()* controlla, infatti, che il file compilazione.txt sia vuoto. Se tale condizione è vera allora la compilazione ha avuto successo, mentre al contrario viene mostrata una finestra modale contenente l'elenco degli errori.

```
public void onLeggiEsito(){

 String contenuto = "";
 String line;
 try {
    FileReader file = new FileReader("C:\\Mobolab\\ProgLego
       \\compilazione.txt");
    BufferedReader in = new BufferedReader(file);
    do{
       line = in.readLine();
       if(line != null){
        contenuto = contenuto + line + "\n";
       }
      } while(line != null);

      file.close();
      if(contenuto.equalsIgnoreCase("")){
        eseguiProgramma();
      } else {
        Messagebox.show(contenuto, "Errore durante la fase
           di compilazione", Messagebox.OK, Messagebox.ERROR
           );
    }
    } catch (Exception e){}
}
```

L'interfaccia dell'applicazione viene riportata in figura 7.10. Anche in questo caso è stata adottata una struttura a borderlayout: nella parte superiore è stato inserito l'header contenente il nome dell'applicazione, nella parte di destra è stata collocata l'applet che mostra le immagini provenienti dalla seconda telecamera, nella parte di sinistra sono presenti l'applet che visualizza le immagini provenienti dalla prima telecamera ed una finestra contenente i programmi d'esempio. Nella parte centrale è presente un'area di testo per la scrittura e l'editing dei programmi. Per quanto riguarda invece l'esecuzione e l'arresto dei programmi sono stati predisposti due appositi pulsanti collocati sopra l'area di testo ("Esegui programma" e"Arresta programma").

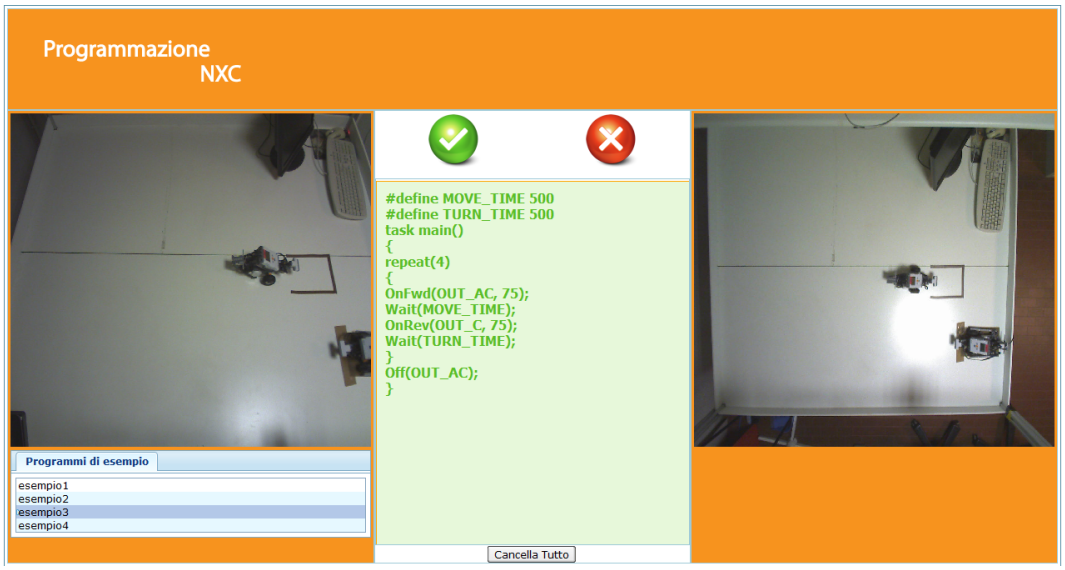

Figura 7.10: L'interfaccia dell'applicazione Programmazione NXC.

7.4.4 La programmazione iconica

Come già accennato nei capitoli precedenti, l'utilizzo del software proprietario della Lego (NXT-G) consente di programmare i robot Mindstorm utilizzando un stile di programmazione basato su icone. NXT-G oltre che essere progettato specificamente per la programmazione degli NXT, rappresenta lo strumento

ideale per un passaggio graduale dallo stile di programmazione Beebot a quello testuale. Per consentire agli utenti l'utilizzo del software si è resa necessaria la remotazione del calcolatore sul quale risiede l'applicazione. Innanzitutto è stato creato un nuovo account utente al quale sono state applicate tutta una serie di restrizioni, nell'ottica di prevenzione da potenziali azioni di danneggiamento. Il passo successivo ha previsto l'installazione di un Server VNC, in particolare, UltraVNC server. Il software utilizza diverse metodologie per l'autenticazione dell'utente, più precisamente mette a disposizione due modalità di login che poggiano, a loro volta, su Windows. E' possibile infatti definire utenti o gruppi di utenti Windows in modo tale che sia concessa loro l'autorizzazione ad effettuare il login da postazioni remote. Le motivazioni che hanno spinto verso la scelta di UltraVNC sono le seguenti: si tratta di un applicativo open source che non richiede l'acquisto di alcun tipo di licenza, ma il vero punto di forza riguarda la possibilità di controllare quest'ultimo anche attraverso un web browser.

Capitolo 8

Conclusioni e sviluppi futuri

La capacità di pensare, organizzare e realizzare un intero lavoro di tesi, richiede impegno e passione, ingredienti fondamentali che, non possono bastare senza il sostegno e la collaborazione delle persone che ci circondano. L'addentrarsi in un mondo come quello educativo richiede un'esperienza che la maggior parte dei docenti maturano dopo anni e anni di insegnamento. Nessuno può resistere al fascino esercitato dal mondo dei bambini, e il solo pensiero di realizzare uno strumento in grado di supportare la loro preparazione può solamente rendere orgoglioso chiunque. Molto lavoro è stato fatto, ma si rischierebbe davvero di peccare di presunzione ritenendolo concluso. Ogni progetto che si rispetti deve sempre dare spazio a ulteriori sviluppi e migliorie.
Innanzitutto, sarebbe interessante poter sfruttare l'idea di Mobolab per realizzare una sorta di RoboCup in miniatura. Il tavolo potrebbe essere predisposto a terreno di gioco per consentire alle scuole di sfidarsi in competizioni sportive utilizzando un semplice web browser. Nell'ottica del divertimento orientato all'apprendimento, la RoboCup rappresenta la manifestazione per eccellenza, in grado di coinvolgere chiunque si avventuri nel suo entourage.
Un'altra idea potrebbe riguardare la costruzione di un manipolatore SCARA per la gestione del piano di lavoro. Un braccio meccanico realizzato con mattoncini Lego, che possa raggiungere ogni punto del tavolo per posizionare particolari elementi, o ovviare a situazioni di stallo, per esempio nell'eventualità un robot subisca un ribaltamento.

In relazione sempre al piano di lavoro si potrebbe pensare di appoggiare uno strato di compensato composto da materiale ruvido per poter consentire ai robot una maggior precisione dei movimenti, dal momento che la superficie attuale risulta essere molto liscia e in alcune occasioni produce uno slittamento dei pneumatici. Sempre in relazione a modifiche dell'infrastruttura potrebbe essere più sicuro l'inserimento di sponde di altezza maggiore per impedire ai robot di cadere dal piano di lavoro e per consentire ai sensori ad ultrasuoni di funzionare in maniera ottimale. Un'ultima importante aggiunta prevede l'installazione di microfoni che permettano di sentire ciò che accade sul tavolo e conseguentemente favorire un controllo ottimale dei dispositivi.

Ringraziamenti

Desidero innanzitutto ringraziare il prof. Riccardo Cassinis, per avermi coinvolto in questo lavoro, per il sostegno e il supporto in ogni singola fase decisionale e per aver riposto in me la sua fiducia. Il doveroso ringraziamento va anche alla dott. Simonetta Siega che ha messo a disposizione la sua grande esperienza nello sviluppo del progetto. Infine non posso che ringraziare di cuore tutti coloro che hanno sempre creduto in me, in particolare i miei genitori e la mia fidanzata Emma, ai quali devo veramente ciò che sono.

Ringrazio di cuore anche tutte le meravigliose persone che ho incontrato lungo il mio percorso universitario, in particolare

Alberto, Alessio, Andrea B., Andrea N., Angelo, Carlo, Davide, Emanuele, Erika, Federica, Flavio, Giancarlo, Giovanni, Luca, Massimo, Michael, Roberto, Umberto.

Grazie davvero di cuore a tutti.

Bibliografia

[1] Avidano M., *Programmare robot con Java*. Hoepli, 2010.

[2] Battegazzore P., *Bee-bot, fare robotica con un giocattolo programmabile a banalità limitata*. 2009.

[3] Benedettelli D., *Programming Lego NXT Robots using NXC*. 2007.

[4] Carbone D., Demo G., Rinaldi R., *Linguaggi per programmare piccoli robot, loro traduttori e altre tecnologie a contorno*. 2008.

[5] Chen H., Cheng R., *ZK Ajax without JavaScript Framework*. 2007.

[6] Hansen J., *LEGO Mindstorms NXT Power Programming: Robotics in C*. 2008.

[7] Hansen J., *NXC Programmer's Guide*. 2010.

[8] Lancia I., Rubinacci F., *Scienze cognitive e aperture pedagogiche*. Franco Angeli, 2009.

[9] Marcianò G., *Usare il linguaggio LOGO per costruire micromondi*. 2004.

[10] Marcianò G., *Costruire microrobot e programmarli*. 2004.

[11] Marcianò G., *Robotica a scuola, Rassegna dell'Istruzione*. Le Monnier, 2005.

[12] Marcianò G., *La Robotica quale ambiente di apprendimento*. 2007.

[13] Marcianò G., *Robotica a scuola*. Lulu, 2008.

[14] Nolfi S., *Dalle Tartarughe Virtuali ai Robot Giocattolo*. 2001.

[15] Oliveira G., Silva R., Lira T., Reis L., *Environment Mapping using the Lego Mindstorms NXT and leJOS NXJ*. 2009.

[16] Tagliagambe S., *Dal cognitivismo al costruzionismo*. 2008.

[17] The Lego Group, *Lego Mindstorm NXT Communication Protocol*. 2006.

[18] The Lego Group, *Lego Mindstorm NXT Direct Commands*. 2006.

[19] The Lego Group, *Lego Mindstorm NXT ARM7 Bluetooth Interface*. 2006.

[20] The Lego Group, *Lego Mindstorm NXT Bluetooth Developer Kit*. 2006.

Elenco delle figure