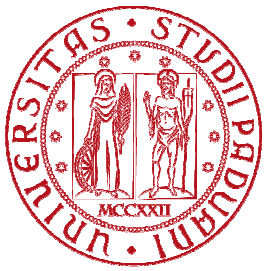

Università degli Studi di Padova

FACOLTÀ DI INGEGNERIA

Corso di Laurea in Ingegneria Informatica

Emulazione robot didattico Bee-Bot tramite Mindstorms NXT

Laureando:

Andrea Pasqualin

Matricola 524182

Relatore:

Prof. Michele Moro

Anno Accademico 2010-2011

Indice

1 Sommario

Lo scopo di questo progetto vuole essere l'emulazione del comportamento e delle funzionalità del robot didattico Bee-Bot™ tramite il kit Lego® Mindstorms® NXT, con l'ausilio di un ambiente software grafico.
Il risultato della tesi è un programma, multipiattaforma, sviluppato in linguaggio Java, che costituisce allo stesso tempo un'interfaccia grafica di programmazione e un ambiente di simulazione.
Il software è in grado di produrre dalla programmazione grafica un codice sorgente NXC, compilarlo attraverso gli strumenti BricxCC ed inviare il relativo file eseguibile al robot NXT.

2 Introduzione

La robotica educativa rappresenta un innovativo strumento didattico che ha reso possibile l'apprendimento in ambiente ludico. Essa trae origine dal paradigma costruttivista dello psicologo e pedagogista Jean Piaget, successivamente rivisitato dalla teoria costruzionista del matematico Seymour Papert. Secondo la filosofia costruttivista la vita consiste in un processo cognitivo che nasce esclusivamente dalla propria esperienza: è il soggetto stesso che crea la realtà, partecipando in maniera attiva alla sua costruzione. Papert introduce, in aggiunta, l'utilizzo di artefatti cognitivi ovvero oggetti tangibili (costruzioni, programmi di computer, composizioni musicali, ecc.) per migliorare e facilitare l'apprendimento. L'invenzione del linguaggio di programma Logo, prima, e, recentemente, la produzione di kit robotici, alla portata dei bambini, sono stati elementi fondamentali per lo sviluppo di questa nuova disciplina. Lo stesso Papert scrive "In tempi più recenti siamo riusciti a costruire computer abbastanza piccoli da poter essere inseriti nei modelli stessi. La differenza è sostanziale; ora l'intelligenza si trova in realtà all'interno del modello, non in un computer fuori scala. Inoltre i modelli possono essere autonomi. Possono muoversi a piacimento senza un cordone ombelicale. Tutto insomma appare più reale". (S. Papert, "I bambini e il computer", Rizzoli, 1994). Le nuove ricerche sul campo hanno permesso di sviluppare e rafforzare nel bambino abilità quali concentrazione, ragionamento, attenzione, memoria, presa di decisione, collaborazione, lavoro di gruppo. Il tutto nel suo pieno benessere psicofisico.

Ciò ha reso possibile l'adozione, sempre più frequente, da parte dei docenti di metodi di insegnamento integrativi alla didattica tradizionale e l'espansione dei vari campi di applicazione.

In quest'ottica si è diffusa sempre più la presenza dei kit didattici Lego® Mindstorms® NXT, nelle scuole di vario grado. NXT è programmabile a differenti livelli di complessità, supporta differenti paradigmi di programmazione e può essere impiegato in differenti livelli educativi (a diverse età).

Inoltre possiede semplici, ma significative possibilità di espansione. Questo avviene connettendo sensori aggiuntivi o interfacciando la piattaforma robotica ad altri dispositivi per consentire l'elaborazione e il controllo remoti.
I tempi di apprendimento per iniziare a lavorare con NXT sono molto brevi mentre l'assemblaggio del robot è intuitivo e non richiede particolari strumenti.
Un'altra caratteristica è la familiarità per i giovani utenti: l'utilizzo dei mattoncini lego è già ben noto e questo permette di trasformare il compito da svolgere in un semplice gioco.

Le caratteristiche precedenti, assieme ad un costo relativamente contenuto, ad un compromesso tra complessità e possibilità di espansione tramite un approccio bottom-up hanno fatto sì che NXT venisse scelto come robot di riferimento in numerosi progetti educativi.

Lo scopo di questo elaborato è rendere NXT, normalmente utilizzato verso gli ultimi anni della scuola primaria, uno strumento impiegabile anche da alunni più giovani, attraverso l'ausilio di un'interfaccia software, che permette di emulare le caratteristiche del robot Bee-Bot, destinato all'utilizzo nei primi anni di scuola primaria.

2.1 Strumenti utilizzati

Gli strumenti utilizzati per la realizzazione del progetto sono stati molteplici:

Java

Java è un linguaggio di programmazione orientato agli oggetti, creato da James Gosling e altri ingegneri di Sun Microsystems. Java è un marchio registrato di Oracle.

I motivi principali per cui è stato scelto questo linguaggio per realizzare NXTBBEmu sono: la sua portabilità, ovvero può essere utilizzato su diversi sistemi operativi senza dover produrre un codice sorgente specifico per ognuno, e la relativa semplicità del linguaggio rispetto ad altri linguaggi di programmazione.

Per lo sviluppo del software è stata adottata la versione 1.6 di Java e l'eseguibile è stato testato su piattaforme Windows.

JCreator

JCreator (http://www.jcreator.com/) è un ambiente di sviluppo integrato per Java sviluppato dalla Xinox Software. In questo progetto è stata utilizzata la versione Lite Edition (LE) distribuita con licenza proprietaria freeware.

Si è scelto di utilizzare questo IDE per scopi didattici.

Le funzionalità di JCreator sono state integrate con strumenti esterni open source per l'analisi del codice quali: Checkstyle (http://checkstyle.sourceforge.net/), PMD (http://pmd.sf.net/), e FindBugs (http://findbugs.sourceforge.net/).

NXC

Not eXactly C (NXC) é un linguaggio di programmazione ad alto livello, simile a C, successore del Not Quite C (NQC), utilizzato su RCX.

Per compilare i programmi NXC, che hanno estensione ".nxc", si utilizza il compilatore NBC, sul quale è basato questo linguaggio di programmazione.

Sia NXC che NBC funzionano sul firmware originale della Lego, non é quindi necessario cambiare il firmware dell'NXT.

NBC e NXC sono programmi open source disponibili per Mac, Linux e Windows rilasciati secondo i termini della licenza Mozilla Public License (MPL).

Bricx Command Center

Bricx Command Center, versione 3.3, è un IDE per Sistemi Operativi Windows utilizzabile per programmare il brick LEGO Mindstorms NXT usando i linguaggi .nxc e NBC. Questo ambiente di sviluppo comprende il compilatore NBC utilizzato da NXTBBEmu per produrre i file eseguibili .rxe.

iText

iText è una libreria open source per creare e manipolare i file pdf con Java, sviluppata da Bruno Lowagie e Paulo Soares. Ai fini dei progetto è stata utilizzata la versione 5.0.5 distribuita secondo i termini della licenza Affero General Public License version 3.

Altri strumenti

Sono stati utilizzati inoltre strumenti grafici quali GIMP (http://www.gimp.org/), Inkscape (http://www.inkscape.org/), ImageMagick (http://imagemagick.org/), per la creazione e la manipolazione delle immagini inserite nell'interfaccia grafica. Questi programmi sono distribuiti con licenze GNU GPL o compatibili.

Le icone utilizzate in NXTBBEmu appartengono al Tango Desktop Project (http://tango.freedesktop.org/) e le altre immagini, invece, fanno parte delle Open Clip Art Library (http://ww.openclipart.org/); entrambe le librerie di immagini sono distribuite con licenza pubblico dominio.

3 Bee-Bot™

3.1 Descrizione

Bee-bot è un robot per la didattica educativa della TTS Group Ltd. Progettato per essere un robot per superfici piane, è adatto all'uso nella scuola dell'infanzia e nella scuola primaria grazie alla elevata semplicità di utilizzo.

Figura 3.1: Bee-Bot™

Bee-Bot appare come una piccola ape di plastica, sulla cui schiena sono presenti dei tasti, con i quali è possibile programmare ed eseguire dei semplici movimenti, molto precisi.

Possono essere memorizzati programmi fino a 40 operazioni, ognuna delle quali può essere un movimento in avanti/indietro, una rotazione di 90° a destra/sinistra, o una pausa. Il passo in avanti o indietro è fisso e misura 15 centimetri.

Ad ogni movimento corrisponde un lampeggiamento degli occhi del robot, accompagnato, a scelta, con un beep sonoro, che si diversifica alla conclusione di una sequenza.

Figura 3.2: Tasti di Bee-Bot

Tasti di programmazione

↑	Avanza di 15 cm
↓	Arretra di 15 cm
←	Ruota a sinistra di 90°
→	Ruota a destra di 90°
PAUSE	Effettua una pausa di 1 secondo

Tasti funzione

CLEAR	Pulisce la memoria
GO	Esegue il programma

3.1.1 I tasti AVANTI e INDIETRO

I comandi AVANTI e INDIETRO sono frecce arancioni poste l'una in direzione opposta all'altra; la prima, rivolta verso il muso dell'ape; l'altra, verso la parte posteriore. Ogni movimento in avanti o indietro provoca uno spostamento del robot di 150 mm nella direzione richiesta. Una sequenza fatta con un numero N di AVANTI permette di avanzare di quel numero preciso di passi.

3.1.2 I tasti DESTRA e SINISTRA

I comandi DESTRA e SINISTRA sono rappresentati da frecce arancioni poste l'una in direzione opposta all'altra; ogni rotazione fa ruotare l'unità di 90°.

3.1.3 Il tasto PAUSE

Il comando PAUSE provoca l'arresto dell'unità per un secondo.

3.1.4 Il tasto CLEAR

Il tasto CLEAR cancella la sequenza di operazioni memorizzate.

3.1.5 Il tasto GO

Il comando GO viene dato mediante il tasto verde, rotondo, posto al centro della piccola tastiera sulla schiena dell'ape. Si differenzia dagli altri tasti per colore, forma e dimensione.

Alla pressione del tasto GO, il robot eseguirà i comandi nell'ordine in cui sono stati memorizzati, con una breve pausa tra un comando e l'altro.

All'accensione la memoria viene cancellata e l'eventuale pressione del tasto GO in questo momento causerà unicamente l'emissione di un suono, senza che il robot compia alcun movimento.

Al completamento della sequenza di comandi l'unità si fermerà e riprodurrà un suono.

Premere Go durante l'esecuzione di una sequenza di comandi fermerà la sequenza.

3.2 Specifiche tecniche

Avanzamento/arretramento:	150mm ±8 mm
Rotazioni destra/sinistra:	90° ± 4°
Pausa:	1 secondo ±15%
Velocità dei movimenti:	circa 65 mm/sec (dipende dalle condizioni delle batterie)
Batterie:	3 batterie alcaline di tipo AA
Autonomia:	circa 8 ore (dipende principalmente dal numero di movimenti eseguiti)
Certificazioni di sicurezza:	CE
	EN71
	EMC
	EN50088
	EN60825

3.3 Attività possibili con Bee-Bot

Bee-Bot è stato progettato con l'obiettivo di consentire ai bambini di avvicinarsi con il gioco al mondo della robotica, aiuta a sviluppare capacità di problem solving e di pensiero logico, a contare, a visualizzare i percorsi nello spazio, a leggere mappe e ad apprendere le basi dei linguaggi di programmazione.

Per raggiungere questi obiettivi diventano indispensabili i "tappetini" che rappresentano lo spazio sul quale Bee-Bot può muoversi. Si può scegliere se produrre da sé questi tappetini, facendo diventare anche questo un momento di apprendimento, o utilizzare le versioni commercializzate. Le versioni attualmente commercializzate dispongono di una griglia, compatibile con le misure del robot, costituita da quadrati di 15 centimetri di lato.

Su queste griglie può essere valutata la percezione spaziale degli alunni (in particolare destra – sinistra e il concetto di direzione relativa), la capacità di ordinare una sequenza composta da almeno pochi step, la capacità di definire un percorso per andare da un punto A ad un punto B.

3.4 Software Focus on Bee-Bot

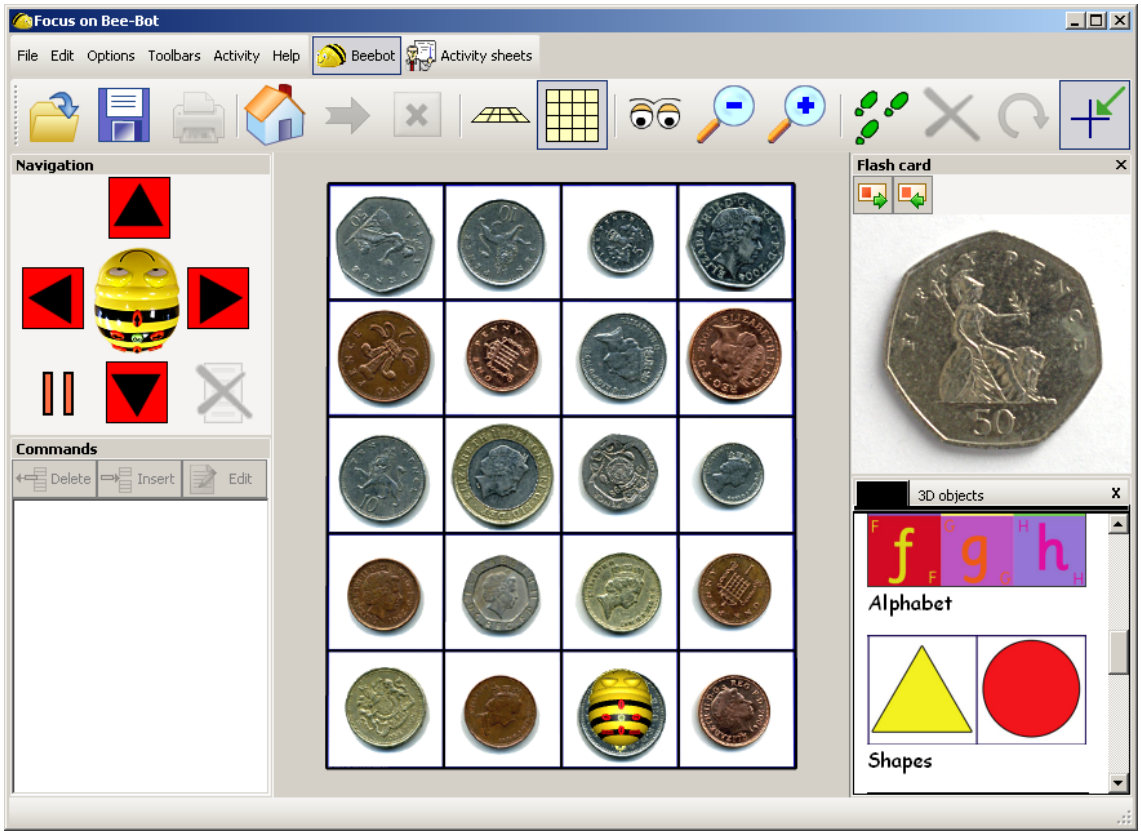

Figura 3.2: Focus on Bee-Bot

Bee-Bot dispone inoltre di un software di simulazione, Focus on Bee-Bot, dove le stuoie commercializzate vengono fedelmente rappresentate come attività. Focus on Bee-Bot è in grado di visualizzare simulazioni bidimensionali e tridimensionali, permette l'inserimento dei comandi in maniera analoga a quanto avviene con il robot fisico, ma essendo Bee-Bot sprovvisto di interfacce per la comunicazione con altri dispositivi, non ne consente il trasferimento della lista di comandi inseriti.

4 LEGO® Mindstorms® NXT

4.1 Descrizione

LEGO Mindstorms NXT è un kit robotico programmabile sviluppato dalla Lego e rilasciato alla fine di Luglio 2006.

Figura 4.1: Brick NXT

Il componente principale del kit è il computer a forma di mattone chiamato "brick NXT". Questo computer riceve segnali da un massimo di quattro sensori e controlla fino a tre servomotori elettrici, attraverso cavi RJ12. Il brick è dotato di un display LCD monocromatico di 100x64 pixel e dispone di quattro tasti che possono essere utilizzati per navigare l'interfaccia utente a menù gerarchici. Esso ha anche un altoparlante che può riprodurre file sonori campionati a 2-16 kHz.

4.2 Specifiche tecniche

Il brick NXT presenta le seguenti specifiche Hardware

- Processore centrale Atmel® 32-bit ARM® processor, AT91SAM7S256, è una CPU RISC progettata dalla ARM, basata sull'architettura ARM v4T ed è stata studiata per dispositivi mobili e a bassa potenza. Questo processore supporta istruzioni a 32-bit e a 16-bit attraverso, rispettivamente, i set di istruzioni ARM e Thumb. Inoltre il microprocessore è equipaggiato con 256 KB di memoria flash e 64 KB di memoria RAM.

- Coprocessore: Atmel® 8-bit AVR processor, ATmega48 a 4 MHz con 4 KB di memoria flash e 512 Byte di memoria RAM. ATmega48 è un processore della famiglia di processori RISC AtmelAVR. La sua caratteristica principale è quella di utilizzare una memoria flash interna per memorizzare il contenuto del programma.

- Controller comunicazione wireless Bluetooth CSR BlueCoreTM 4 v2.0 +EDR System che lavora alla frequenza di 26 MHz. Il dispositivo bluetooth ha memoria flash esterna di 8 Mbit e una memoria RAM di 47 KB. Supporta il Serial Port Profile (SPP).

- Porta USB 1.1 full speed (velocità di trasmissione 12 Mbit/s).

- 4 porte di input con interfaccia RJ12 a 6 fili alle quali è possibile collegare i sensori o altri dispositivi di input, sia analogici, sia digitali.

- 3 porte di output con interfaccia RJ12 a 6 fili alle quali è possibile collegare i motori o altri dispositivi di output.

- Display grafico LCD con matrice 100x64 pixel e un'area di visualizzazione di 26x 40.6 mm.

- Altoparlante con una risoluzione di 8 bit, supporta frequenze di campionamento di 2-16 kHz.

- 4 tasti gommati per gestire l'interfaccia utente.

- Alimentazione attraverso 6 batterie alcaline AA oppure tramite una batteria agli ioni di litio da 1400 mAh.

Il brick NXT include un firmware open source rilasciato dalla Lego che supporta una macchina virtuale secondo le tecnologie Labview di National Instruments.

Oltre al brick NXT il kit educativo del robot Lego Mindstorms NXT include:

- 431 parti Lego Technic
- 3 servomotori
- un sensore di contatto (Touch Sensor) che rileva la sua pressione e il suo rilascio
- un sensore di suoni (Sound Sensor) che misura il livello sonoro in base a dei modelli sonori preimpostati
- un sensore di luce (Light Sensor) che misura l'intensità della luce
- un sensore ad ultrasuoni (Ultrasonic Sensor) per misurare le distanze

Inoltre si possono acquistare dei sensori compatibili supplementari, ad esempio:

- un rilevatore di infrarossi (IRSeeker Sensor)
- un sensore di comunicazioni infrarossi (IRLink Sensor)
- un sensore di accelerazione/accelerometro (Acceleration-Tilt sensor)
- un giroscopio (Gyro Sensor)
- una bussola (Compass Sensor)
- un sensore di colori (Color Sensor)

4.3 Servomotori

I componenti di cui si compone il sistema di attuazione del robot sono i tre servomotori, alimentati con correnti continua e controllati tramite modulazione a larghezza di impulso (PWM), aventi le seguenti caratteristiche:

- Tensione di alimentazione 9V (DC)
- Velocità massima 170 rpm (117 rpm a 9V)
- Potenza meccanica a 9V 2,03W
- Potenza elettrica a 9V 4,95W
- Efficienza a 9V 41%
- Assorbimento a 9V 0.55A
- Corrente di No-Load 60mA
- Coppia a 9V 16,7 N*cm
- Coppia in stallo 50 N*cm
- Corrente di stallo 2A
- Peso 80 gr

Tensione di alimentazione	Momento torcente	Velocità di rotazione	Corrente	Potenza meccanica	Potenza elettrica	Efficienza
4.5 V	16.7 N.cm	33 rpm	0.6 A	0.58 W	2.7 W	21.4 %
7 V	16.7 N.cm	82 rpm	0.55 A	1.44 W	3.85 W	37.3 %
9 V	16.7 N.cm	117 rpm	0.55 A	2.03 W	4.95 W	41 %
12 V	16.7 N.cm	177 rpm	0.58 A	3.10 W	6.96 W	44.5 %

Tabella 4.1: Alcuni dati sperimentali relativi ai servomotori

Fonte: http://www.philohome.com/motors/motorcomp.htm

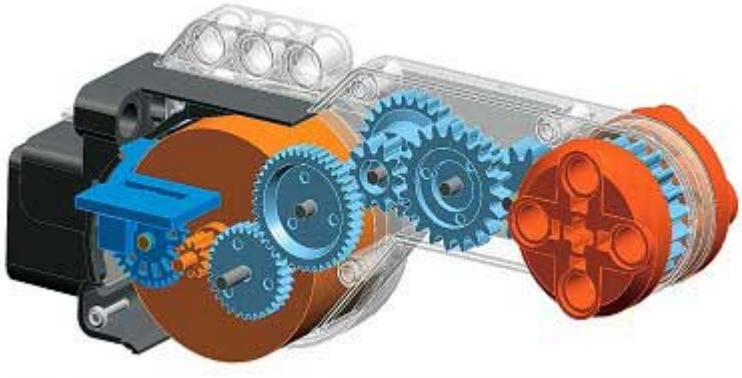

Figura 4.2: servomotore dell'NXT

Ogni servomotore ha al suo interno un sensore di rotazione che permette di misurare la velocità e/o la distanza percorsa da un certo punto.
I sensori di rotazione misurano la posizione del motore in gradi con una precisione di ± 1 grado. Una rotazione completa equivale a 360 gradi.
I sensori di rotazione, unitamente alla capacità dell'NXT di controllare a coppie i motori per sincronizzarli, permette un controllo preciso dei movimenti del robot.

4.4 Aspetti geometrici

Nell'ambito di questo progetto è stato utilizzato il robot in una versione semplificata, priva di sensori, della sua configurazione Tribot.

Nella configurazione Tribot il robot è assemblato come un veicolo a trazione differenziale, avente due ruote attive con lo stesso asse di rotazione, controllate separatamente e una ruota passiva eccentrica (caster), che si allinea automaticamente e rapidamente nella direzione di spinta dello chassis e svolge la funzione di punto di appoggio per il bilanciamento statico (senza influenzare la mobilità dello chassis).

Successivamente la ruota eccentrica è stata sostituita da una sfera per cercare di ottenere un minore errore nella rotazione eliminando il problema del riallineamento.

Il robot, così configurato (indipendentemente dalla ruota passiva adottata), può ruotare sul posto senza muovere il centro delle ruote se le velocità angolari delle due ruote sono uguali e contrarie, come dimostrato dalle seguenti considerazioni (*Teacher Education on Robotics-Enhanced Constructivist Pedagogical Methods*, 2009).

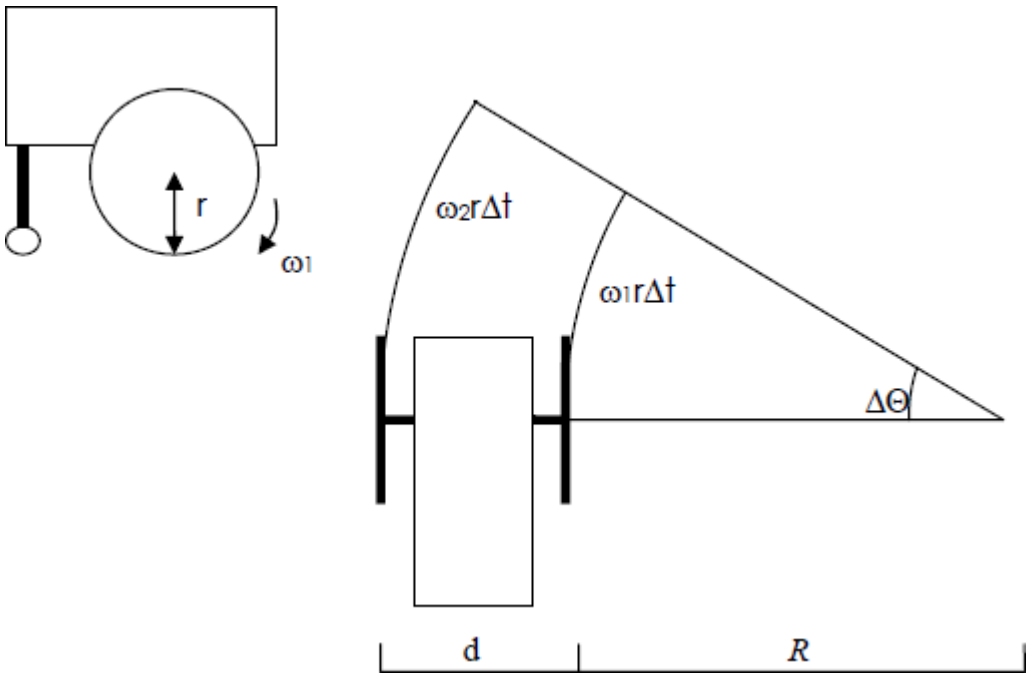

Figura 4.3: Movimento di rotazione

Considerata una rotazione in verso orario delle ruote e supposto che questa rotazione faccia muovere il robot in avanti, se la ruota$_1$ ha velocità angolare ω_1 e la ruota$_2$ ha velocità angolare ω_2, con $\omega_2 > \omega_1$, misurate in rad/s, può essere facilmente verificato che il robot ruota a destra e le due ruote tracciano due traiettorie circolari concentriche.

Definito con r il raggio comune delle ruote e con d la distanza fissa tra le due ruote in un breve movimento di Δt unità di tempo le due ruote tracceranno un arco di circonferenza lungo rispettivamente $\Delta A_1 = \omega_1 r \Delta t$ e $\Delta A_2 = \omega_2 r \Delta t$

Sia R la distanza tra la ruota$_1$ e l'ipotetico centro dei due archi di circonferenza e $\Delta\Theta$ l'angolo corrispondente ai due archi, risulta che

$$\Delta A_1 = \omega_1 r \Delta t = r\theta_1 = R\Delta\Theta$$

$$\Delta A_2 = \omega_2 r \Delta t = r\theta_2 = (R+d)\Delta\Theta$$

Pertanto

$$\Delta\Theta = \frac{\omega_1 r \Delta t}{R} = \frac{\omega_2 r \Delta t}{R+d}$$

$$\Omega = \frac{\Delta\Theta}{\Delta t} = \frac{\omega_1 r}{R} = \frac{\omega_2 r}{R+d}$$

$$\frac{\omega_1}{R} = \frac{\omega_2}{R+d}$$

$$R = \frac{d\omega_1}{\omega_2 - \omega_1}$$

Pertanto il raggio R non dipende dal raggio r delle ruote ma soltanto dalla velocità angolare e dalla distanza tra le due ruote.

Quando $\omega_2 \to \omega_1$, $R \to \infty$, il movimento risulta rettilineo e la prima equazione diventa semplicemente $\Delta A = \omega r \Delta t = r\Delta\theta$

Quando $\omega_1 = 0$, $R = 0$ e il robot gira attorno alla ruota$_1$ che non si sta muovendo

Quando

$$\omega_1 = -\omega_2 \quad (\omega_2 > 0)$$

Risulta $R = -\frac{d}{2}$ e il robot ruota attorno al punto medio tra le due ruote indipendentemente dalla potenza applicata.

La velocità angolare e l'angolo percorso dal robot (ignorando il segno) sono in questo caso

$$\Omega = \frac{2|\omega_1| r}{d}$$

$$\Theta = \frac{|\theta_1| r}{R} = \frac{2|\theta_1| r}{d}$$

Il problema ora diventa imporre le due velocità angolari richieste dalla traiettoria desiderata, ma fortunatamente le API di NXC forniscono i comandi adeguati.

5 Manuale utente

5.1 Premessa

L'emulatore NXTBBEmu è stato creato con lo scopo di fornire uno strumento in grado di utilizzare la flessibilità del robot NXT per svolgere una funzione analoga a Bee-Bot. L'assunto è che, almeno per il momento, non sono previste estensioni hardware ed in particolare un sostituto dei pulsanti pertanto la programmazione del robot avviene attraverso il personal computer.
È da ritenersi, in quest'ottica, un'interfaccia grafica di programmazione.
NXTBBEmu integra inoltre un simulatore bidimensionale del comportamento del robot; all'utente è lasciata facoltà di simulare gli effetti dei comandi impartiti, prima dell'invio del file eseguibile al robot.
Il programma è multilingua e dotato di selezione automatica della lingua da utilizzare a seconda di quella del sistema operativo in uso; è comunque possibile impostare una lingua diversa da un'apposita finestra di configurazione.

5.2 Pacchetto software

Il pacchetto software non necessita di una particolare procedura d'installazione per poter essere utilizzato: infatti è sufficiente copiare i file forniti in una cartella dedicata.
Questa cartella dovrà contenere:

- l'archivio java eseguibile *NXTBBEmuGUI.jar*
- l'archivio java eseguibile *iText.jar*
- una sottocartella *mats* contenente file di attività aggiuntive per il programma.
- una sottocartella *UserGuides* contenente i manuali utente del programma in varie lingue.

Successivamente alla personalizzazione delle impostazioni verrà salvato nella cartella d'installazione il file *custom.properties* contenente queste modifiche.

I requisiti dell'applicazione sono i seguenti:

- risoluzione video di almeno 1024x600, preferibile superiore
- Java Runtime Environment 6 o superiore (http://www.java.com/)
- eseguibile nbc, nella versione stand alone o contenuto in BricxCC (http://bricxcc.sourceforge.net/)
- driver di comunicazione per il robot NXT installati (http://mindstorms.lego.com/en-us/support/files/default.aspx#Driver)

5.3 Guida all'utilizzo di NXTBBEmu

Figura 5.1: Finestra principale di NXTBBEmu

Per quanto riguarda l'aspetto grafico si è cercato di realizzare un'interfaccia user-friendly, semplice e intuitiva.

L'interfaccia grafica è costituita da:

a. una barra dei menù
b. una barra degli strumenti
c. un pannello di navigazione
d. un pannello di controllo della simulazione
e. un pannello dei comandi
f. un pannello di simulazione
g. un pannello di selezione degli obiettivi
h. un pannello di selezione delle attività

5.3.1 Barra dei menù

La barra dei menù permette di accedere alla maggior parte delle funzioni dell'interfaccia grafica.

- Menù file
    - Nuova lista: permette di creare una nuova lista senza nome, chiedendo di salvare eventuali file aperti.
    - Apri lista: permette di aprire un documento (in formato .txt), chiedendo di salvare eventuali file aperti.
    - Salva lista: permette di salvare la lista comandi attuale in formato .txt
    - Salva NXC: esporta la lista comandi attuale in un file .nxc.
    - Stampa Pdf: permette di "stampare" un resoconto dell'attività corrente/della simulazione.
    - Esci: esce dall'interfaccia, chiedendo di salvare eventuali file aperti.
- Menù Simulazione
    - Avvia simulazione.
    - Sospendi simulazione.
    - Ferma simulazione.
- Menù Strumenti
    - Compila e carica su robot: compila la lista attuale e carica sul robot il relativo file eseguibile .rxe. Il robot confermerà l'avvenuta ricezione del file attraverso l'emissione di un segnale acustico.
    - Compila, carica su robot ed esegui: compila la lista attuale, carica sul robot il relativo file eseguibile .rxe e lo esegue; prestare particolare attenzione al cavo usb quando si utilizza questa funzione, essa è indicata principalmente per la connessione bluetooth.
- Menù Opzioni
    - Griglia: permette di attivare/disattivare la visualizzazione della griglia sull'attività corrente; l'impostazione di default è inserita nel file di descrizione dell'attività.

- Schermo intero: ingrandisce la finestra dell'interfaccia a schermo intero.
- Lingua: permette di selezionare la lingua del programma.
- Impostazioni: permette di modificare le impostazioni del programma, effettuare la taratura del robot.

- Menù aiuto
  - Manuale utente: apre il file della guida.
  - ?: visualizza le informazioni relative al programma in uso.

5.3.2 Barra degli strumenti

La barra degli strumenti permette di raggiungere i comandi di utilizzo più comune senza dove ricorrere alla barra dei menù, fornendo un accesso immediato.


Crea una nuova lista di comandi


Apre una lista di comandi salvata in precedenze


Salva la lista dei comandi in formato testuale

Esporta la lista dei comandi in formato .nxc


Crea un resoconto pdf della simulazione


Invia il programma all'NXT


Invia il programma all'NXT e ne avvia l’esecuzione

Fa ritornare nella posizione iniziale il robot visualizzato

Attiva/disattiva la visualizzazione della griglia

Ingrandisce la finestra a schermo intero

Seleziona la lingua

Modifica le opzioni di configurazione

Esce dal programma

5.3.3 Pannello di navigazione

Figura 5.2: Pannello di navigazione

Il pannello di navigazione è il componente attraverso il quale viene realizzata la programmazione grafica: la pressione di un tasto comporta l'aggiunta della relativa azione in coda alla lista dei comandi. I tasti sono dotati di descrizione e di icone che dovrebbero escludere qualsiasi interpretazione ambigua del loro significato ed effetto. La stessa interfaccia grafica viene riproposta dalle finestre di dialogo "Modifica" e "Inserisci", le quali mostrano qual è l'azione di riferimento tramite la barra del titolo.

Figura 5.3: Finestra di dialogo "Modifica"

Figura 5.4: Finestra di dialogo "Inserisci"

5.3.4 Pannello di controllo della simulazione

Figura 5.5: Pannello di controllo della simulazione

Raccoglie i controlli della simulazione. I tasti non riportano una descrizione testuale (tranne il tooltip), ma sono rappresentati da icone internazionalmente riconosciute.

Il tasto stop  ferma l'esecuzione e riporta il robot nella posizione iniziale per l'attività corrente.

Il tasto play  avvia la simulazione della lista di operazioni corrente e la riprende nel caso sia in pausa.

Il tasto pausa  sospende la simulazione, nella posizione in cui si trova il robot alla pressione del tasto, mantenendo visibile la traccia delle operazioni effettuate.

Lo slider permette di selezionare la velocità di riproduzione della simulazione; la modifica della velocità non ha effetto sul codice inviato al robot.

5.3.5 Pannello dei comandi

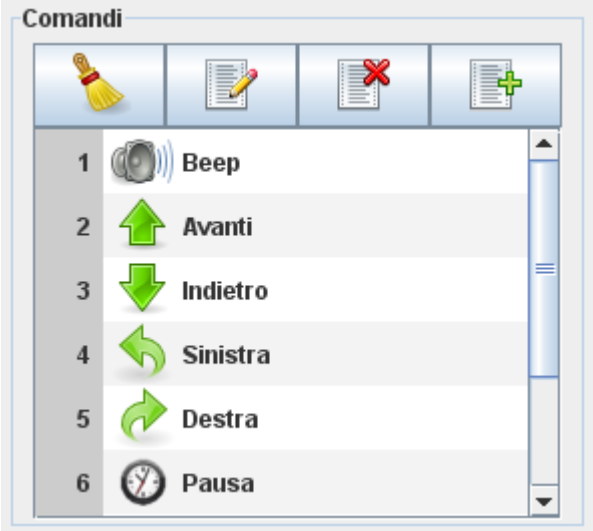

Figura 5.6: Pannello dei comandi

Questo pannello visualizza la lista dei comandi aggiornata in tempo reale e ne permette la modifica. Permette di cancellare la lista corrente oppure selezionare ogni singola azione per rimuoverla, modificarla o inserire un'altra azione prima della selezionata attraverso i tasti al di sopra della lista oppure con il menù contestuale che compare alla pressione del tasto destro. Durante la simulazione viene selezionata l'azione riprodotta in quel momento.

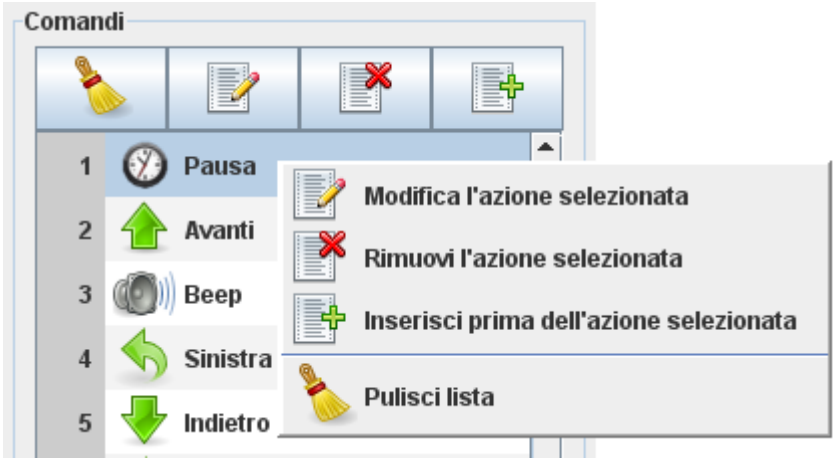

Figura 5.7: Menù contestuale

Le opzioni “Modifica” e “Inserisci” appaiono come le finestre di dialogo già presentate.
L’opzione “Elimina” invece comporta la visualizzazione della seguente finestra di dialogo, esclusivamente testuale.

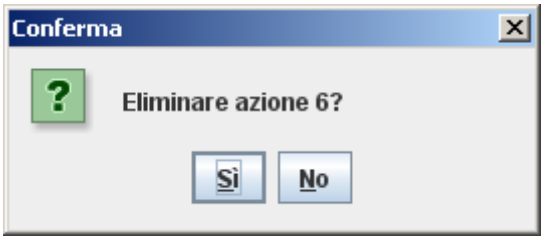

Figura 5.8: Finestra di dialogo “Elimina”

Mentre la simulazione è in pausa, se viene eliminata, aggiunta o modificata un’azione facente nella parte di lista già simulata, alla pressione del tasto play la simulazione verrà fatta ripartire dall’inizio.

5.3.6 Pannello di simulazione

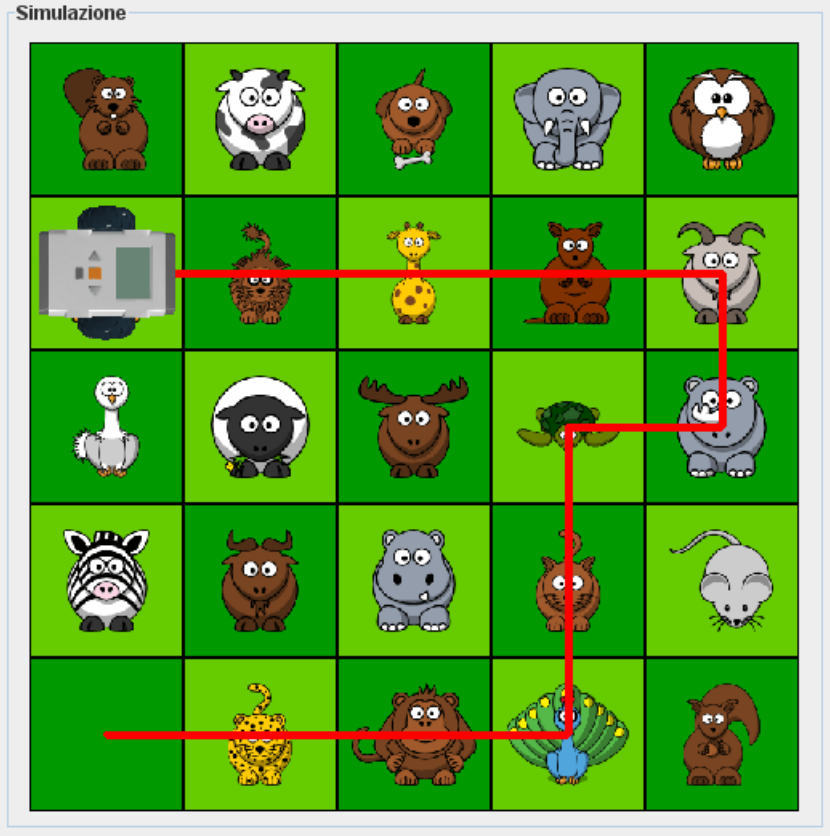

Figura 5.9: Pannello di simulazione

Il pannello di simulazione è il pannello centrale del'interfaccia. Mostra all'utente l'attività per la quale si sta costruendo la lista dei comandi, riproduce su di essa la simulazione e visualizza una traccia del percorso conseguente ai comandi inseriti. Permette di visualizzare una griglia sopra l'immagine, nel caso questa ne sia sprovvista, per agevolare l'utente. Il pannello si dimensiona automaticamente a seconda della risoluzione dello schermo per occupare la maggior area di visualizzazione possibile.

→

Figura 5.10: Rappresentazione del robot con indicato il verso di percorrenza del comando avanti

5.3.7 Pannello di selezione degli obiettivi

Figura 5.11: Pannello di selezione degli obiettivi

È un pannello costituito da quatto menù a tendina ognuno dei quali indica un obiettivo da conseguire. Un obiettivo si considera raggiunto se si esegue una pausa o un beep quando ci si trova su di esso. La selezione di uno o più obiettivi è facoltativa. Al termine della simulazione viene visualizzata una finestra di dialogo che indica se e quando (con quale azione della lista) gli obiettivi sono stati raggiunti.
Per resettare rapidamente gli obiettivi impostati è possibile premere sul tasto corrispondente all'attività attualmente caricata nel pannello di selezione dei mat.

5.3.8 Pannello di selezione delle attività

Figura 5.12: Pannello di selezione delle attività

Pannello che permette di scegliere l'attività da svolgere fra quelle incluse nel programma e quelle presenti nella cartella esterna indicata nelle opzioni. Le attività vengono visualizzate in ordine alfabetico, divise in due gruppi, il primo contiene le attività esterne aggiunte dall'utente, il secondo contiene le attività incluse nel programma.

5.4 Selezione della lingua

Il programma seleziona automaticamente la lingua a seconda della lingua impostata nel sistema operativo in uso; è possibile aprire la finestra di selezione della lingua dal menù opzioni oppure premendo l'apposito tasto della toolbar.
Dopo aver selezionato la lingua desiderata è necessario riavviare il programma.

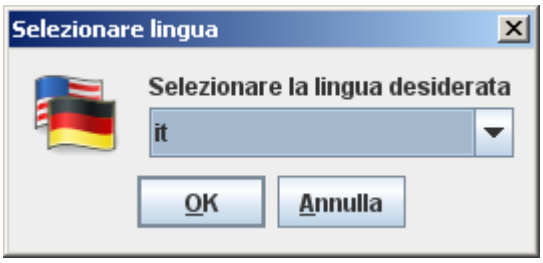

Figura 5.13: Finestra di selezione della lingua

5.5 Configurazione

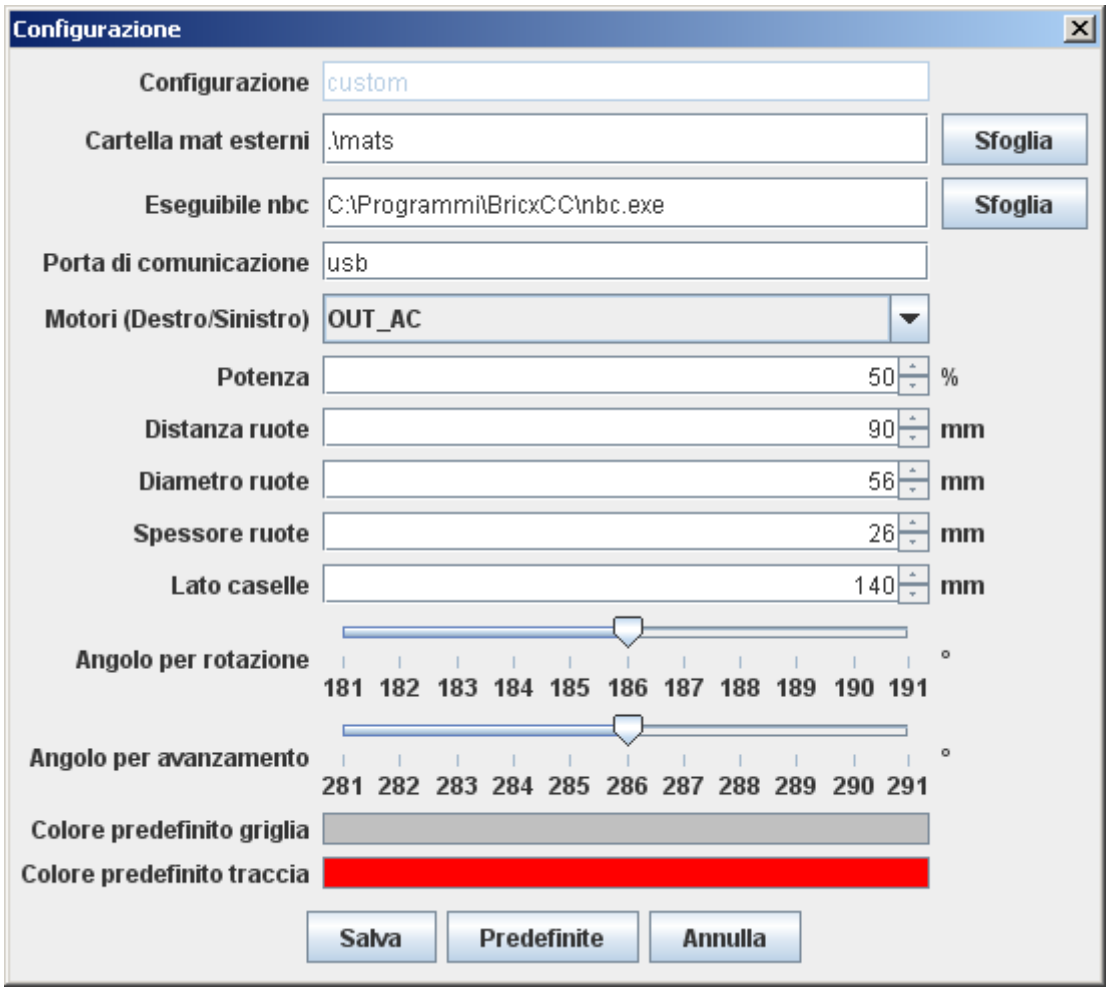

Figura 5.14: Finestra di configurazione

Dalla finestra di dialogo di configurazione è possibile:

- selezionare la cartella contenente le attività esterne aggiunte dall'utente (richiede il riavvio del programma);
- impostare il percorso dell'eseguibile nbc, per compilare e caricare sul robot le operazioni da eseguire;
- impostare la porta di comunicazione con il robot
- selezionare le porte a cui sono collegati i motori
- impostare la potenza dei motori

- effettuare la taratura del robot, modificando gli slider che rappresentano l'angolo di rotazione e di avanzamento, in seguito all'inserimento dei dati relativi al dimensionamento del robot e del tappetino su cui si svolge l'attività.
- impostare i colori predefiniti di griglia e traccia

5.6 Stampa resoconto in pdf

È possibile stampare un resoconto della simulazione avente il seguente schema

- Nome attività
- Stato iniziale dell'attività
- Obiettivi (se selezionati)
- Lista numerata dei comandi, interrotta, in tanti punti quanti sono gli obiettivi raggiunti, da un messaggio indicante il raggiungimento dell'obiettivo seguito dall'immagine della cella che lo rappresenta
- Stato finale attività

La stampa è consentita soltanto quando si è portato a termine la riproduzione della simulazione e solo se il robot non si trova al di fuori della superficie dell'attività

5.7 Salvataggio in formato .txt

Dopo aver inserito le operazioni che il robot deve svolgere è possibile salvarle in formato txt, per poterle ricaricare successivamente.

I comandi saranno preceduti da un commento indicante la lingua in cui sono stati salvati in comandi, la data e l'ora di salvataggio e l'attività per cui quei comandi erano stati inseriti.

Nel caso in cui la lista dei comandi sia stata salvata in una lingua diversa da quella in uso all'apertura del file, il programma effettuerà una traduzione automatica.

5.8 Salvataggio in formato NXC

NXC è il linguaggio utilizzato in questo progetto per programmare il brick NXT.

Dopo aver inserito le operazioni che il robot deve svolgere è possibile salvarle in formato NXC.

Il file salvato, sarà basato su un paradigma di programmazione procedurale e avrà la seguente struttura:

- Definizione delle costanti utilizzate dal programma .nxc (potenza motori, angoli di rotazione dei motori);

- Definizione delle subroutine relative alla azioni che può compiere il robot (forward, backward, left, right, pause, beep);

- Task main che contiene la traduzione dei comandi inseriti, tramite l'interfaccia, nelle chiamate alle subroutine precedenti. Il task main si conclude con l'istruzione per lo spegnimento dei motori.

```
#define POWER_PCT 50
#define ANGLE 227
#define MOTORS OUT_AC


sub forward() {
  // ISTRUZIONI PER FAR AVANZARE IL ROBOT DI UNA CELLA
}

sub backward() {
  // ISTRUZIONI PER FAR AVANZARE IL ROBOT DI UNA CELLA
}

sub pause() {
  // ISTRUZIONI PER ESEGUIRE UNA PAUSA DI 1 SECONDO
}

sub right() {
// ISTRUZIONI PER RUOTARE IL ROBOT DI 90° A DESTRA
}

sub left() {
// ISTRUZIONI PER RUOTARE IL ROBOT DI 90° A SINISTRA
}

sub beep() {
// ISTRUZIONI PER RIPRODURRE UN SUONO PREDEFINITO
}


task main() {

  forward();
  forward();
  left();
  backward();
  right();

  Off(OUT_AC);
}
```

Schema file NXC

Il file .nxc prodotto potrà essere compilato ed eventualmente modificato separatamente con BricxCC.

Eventuali modifiche al file .nxc salvato non saranno prese in considerazione da NXTBBEmu che al fine di creare i file eseguibili per il robot si basa unicamente sulla lista di comandi impartiti dall'interfaccia grafica e a partire da quest'ultima crea un file .nxc temporaneo in maniera trasparente all'utente.

5.9 Descrizione file attività (mat)

Ogni attività è costituita da un'immagine e da un file di testo .txt contenente i seguenti parametri

NAME	Nome dell'attività
FILENAME	Nome del file immagine
ROWS	Righe dell'immagine
COLUMNS	Colonne dell'immagine
INITIALCELLROW	Riga a cui appartiene la cella iniziale
INITIALCELLCOLUMN	Colonna a cui appartiene la cella iniziale
ORIENTATION	Orientamento del robot in gradi
GRID	Valore booleano che definisce se la griglia va disegnata o no (true o false)
GRIDCOLOR	Colore della griglia espresso tramite numero esadecimale
TRACKCOLOR	Colore della traccia espresso tramite numero esadecimale

L'identificazione delle celle per una griglia (n · m), dove n rappresenta il numero di righe ed m il numero di colonne, segue lo schema sottostante

	0	1	2	…	m-1
0	(0, 0)	(0, 1)	(0, 2)	…	(0, m-1)
1	(1, 0)	(1, 1)	(1, 2)	…	(1, m-1)
2	(2, 0)	(2, 1)	(2, 2)	…	(2, m-1)
…	…	…	…	…	
n-1	(n-1, 0)	(n-1, 1)	(n-1, 2)		(n-1, m-1)

```
NAME=animals
FILENAME=animals.png
ROWS=5
COLUMNS=5
INITIALCELLROW=4
INITIALCELLCOLUMN=0
ORIENTATION=90
GRID=false
GRIDCOLOR=0xc0c0c0
TRACKCOLOR=0xcc0000
```

Esempio di file descrittore di attività

Tutte le attività che si desidera aggiungere al programma devono essere contenute nella stessa cartella e verranno caricate all'avvio del programma.

Le immagini possono essere nei formati più comuni: .png, .jpg o .gif.
Nella realizzazione dell'immagine si consiglia di utilizzare come lato di ogni singola cella 150 o 180 pixel.

5.10 Esecuzione file eseguibile su NXT

Per trasferire il programma al robot NXT occorre che quest'ultimo sia acceso e correttamente connesso al personal computer, si suggerisce l'utilizzo del cavo di connessione usb.
Il trasferimento del file può essere avviato dal menù strumenti oppure dagli appositi tasti della barra degli strumenti.
Una finestra di dialogo segnalerà il corretto trasferimento del file, inoltre l'avvenuta ricezione, sarà confermata dal robot con l'emissione di un segnale acustico.

Al fine di far eseguire al robot i comandi impostati occorre posizionarlo nella casella iniziale curandosi di sovrapporre il punto medio dell'asse che congiunge le due ruote attive con il punto d'incontro delle diagonali della casella. Aiutarsi con i lati della cella per allineare il robot nella direzione del moto.

Il posizionamento deve avvenire prima del trasferimento del programma al robot, se si desidera avviare l'esecuzione direttamente da NXTBBEmu, altrimenti potrà avvenire successivamente e il programma potrà essere avviato dal menù del brick NXT.

5.11 Sessione di lavoro

Di seguito è riportata, a titolo d'esempio, un'intera sessione di lavoro.

5.11.1 Fase iniziale

All'apertura del programma è possibile selezionare il mat desiderato tramite il pannello di selezione delle attività.

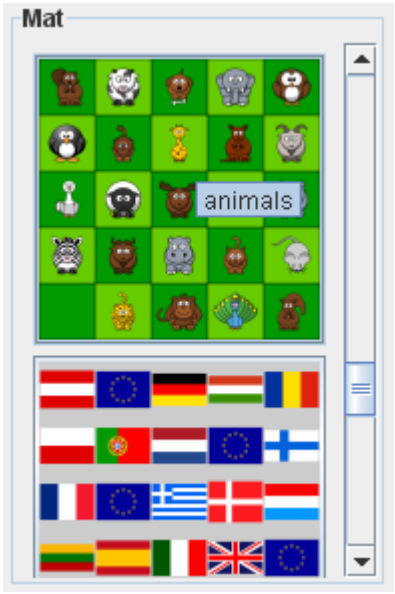

Figura 5.15: selezione mat

Al termine del caricamento del mat è possibile selezionare uno o più obiettivi. La selezione è necessaria solo se si desidera che sia il programma a verificare il raggiungimento degli obiettivi. È possibile selezionare gli obiettivi anche appena prima di avviare la simulazione, ma è sconsigliato in quanto non si avrebbe un riferimento visivo degli obiettivi da raggiungere.

Figura 5.16: selezione obiettivi

5.11.2 Programmazione

Selezionati gli obiettivi è possibile iniziare a inserire in sequenza i comandi per raggiungerli attraverso il pannello di navigazione. In caso l'utente ne abbia necessità può modificare o eliminare i comandi inseriti o aggiungerne in posizioni intermedie.
Si ricorda che un obiettivo è da considerarsi raggiunto soltanto se, mentre ci si trova su di esso, viene eseguita una pausa o un beep.

Figura 5.17: lista comandi inseriti

5.11.3 Simulazione

È possibile avviare la simulazione al termine dell'inserimento dei comandi oppure dopo averne inseriti soltanto alcuni per verificarne la correttezza.

Se l'utente lo desiderasse potrebbe non eseguire la simulazione ed inviare direttamente il file eseguibile al robot.

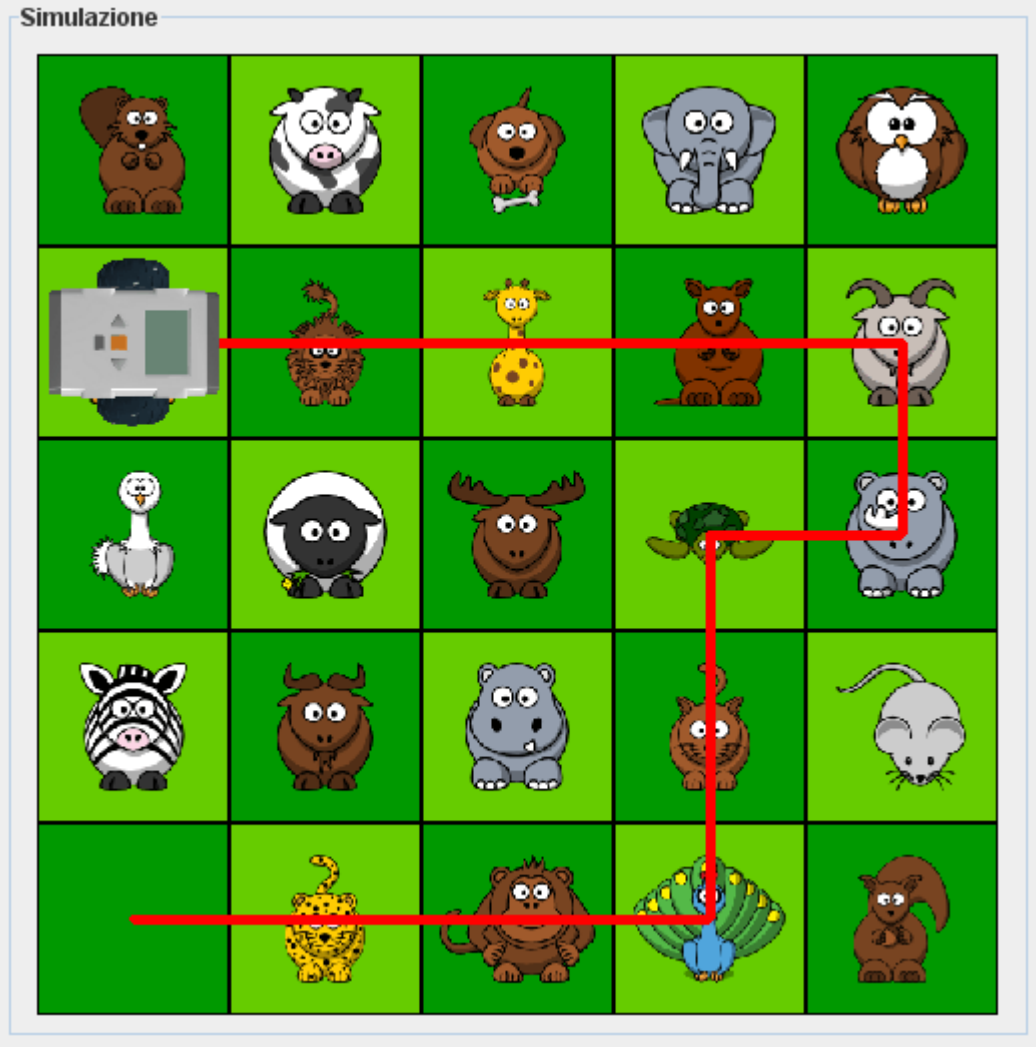

Figura 5.18: simulazione

Al termine della simulazione verrà visualizzata una finestra di dialogo indicante gli obiettivi raggiunti e quale sia l'ultima azione che ne ha determinato il raggiungimento.

Figura 5.19: obiettivi raggiunti

5.11.4 Esecuzione

Dopo aver inserito i comandi e al termine della simulazione, se è stata avviata, l'utente può inviare un file eseguibile al robot preventivamente connesso al computer e acceso.

È facoltà dell'utente scegliere se inviare solamente il file oppure inviare il file e far sì che questo venga avviato immediatamente dopo in automatico. Nel secondo caso è opportuno che il robot sia già stato posizionato nella cella di partenza.

Nel caso in cui il trasferimento vada a buon fine verrà visualizzato un messaggio di conferma e il robot emetterà un segnale acustico; in caso contrario verrà visualizzato un messaggio di errore.

Figura 5.20: Trasferimento eseguito

Figura 5.21: Errore di trasferimento dati

Se non si è scelto di avviare la simulazione dal computer, sarà necessario avviare il programma direttamente dal menù del robot, dopo averlo posizionato nella cella iniziale.

5.11.5 Output

Soltanto a simulazione terminata sarà possibile “stampare” un riepilogo in pdf della sessione appena eseguita.

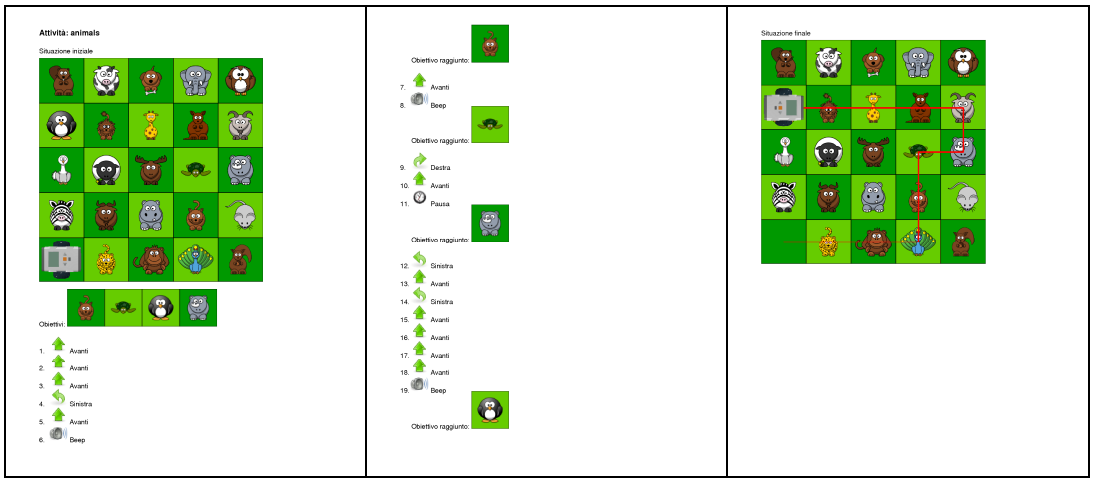

Figura 5.22: animals.pdf

È possibile salvare la lista dei comandi inseriti in formato testuale per poterli ricaricare in un momento successivo.

```
#it
#giovedì 17 febbraio 2011 16.50.03
#animals
Avanti
Avanti
Avanti
Sinistra
Avanti
Beep
Avanti
Beep
Destra
Avanti
Pausa
Sinistra
Avanti
Sinistra
Avanti
Avanti
Avanti
Avanti
Beep
```

animals.txt

In aggiunta il programma fornisce la possibilità di esportare la lista dei comandi in formato NXC.

```
#define MOTORS OUT_AC
#define POWER_PCT 50
#define BF_ANGLE 286
#define ROT_ANGLE 186
sub forward()
{
  ResetAllTachoCounts(MOTORS);
  RotateMotorEx(MOTORS, POWER_PCT, BF_ANGLE, 0, true, false);
}

sub backward()
{
  ResetAllTachoCounts(MOTORS);
  RotateMotorEx(MOTORS, -POWER_PCT, BF_ANGLE, 0, true, false);
}

sub pause()
{
  OffEx(MOTORS, RESET_ALL);
  Wait(1000);
```

```
}

sub right()
{
  ResetAllTachoCounts(MOTORS);
  RotateMotorEx(MOTORS, POWER_PCT, ROT_ANGLE, 100, true, true);
  Wait(50);
}

sub left()
{
  ResetAllTachoCounts(MOTORS);
  RotateMotorEx(MOTORS, POWER_PCT, ROT_ANGLE, -100, true, true);
  Wait(50);
}

sub beep()
{
  OffEx(MOTORS, RESET_ALL);
  PlayToneEx(262, 400, 3, FALSE);
  Wait(1000);
}


task main()
{

  forward();
  forward();
  forward();
  left();
  forward();
  beep();
  forward();
  beep();
  right();
  forward();
  pause();
  left();
  forward();
  left();
  forward();
  forward();
  forward();
  forward();
  beep();
  Off(MOTORS);
}
```

animals.nxc

6 Manuale tecnico

6.1 Impostazione generale

L'intera applicazione è racchiusa nel package nxtbbemu.

Nella cartella del package si trovano tutte le classi che compongono l'applicazione e il file *default.properties* dov'è salvata la configurazione predefinita. Sono inoltre presenti le sottocartelle indicate nella seguente tabella.

icons	contenente le icone utilizzate nel programma
languages	contenente i file di lingua
mats	contenente i mat interni
nxc	contenente i file necessari a costruire i file .nxc
tribot	contenente le immagini con cui è rappresentato il robot

Le classi che compongono il package sono

NXTBBEmuConf	gestisce i file di configurazione dell'applicazione
NXTBBEmuConfJDialog	implementa la finestra di dialogo che permette di modificare i parametri di configurazione
NXTBBEmuFileChooser	implementa i FileChooser utilizzati dall'applicazione
NXTBBEmuFrame	costituisce la classe principale dell'applicazione, implementa il frame, i pannelli e i metodi per gestirli
NXTBBEmuGUI	include la classe di test per il metodo
NXTBBEmuImageRenderer	definisce l'aspetto grafico della lista di comandi
NXTBBEmuRes	gestisce le traduzioni dell'applicazione
CmdExecJDialog	implementa la finestra di dialogo che cattura l'output di nbc.exe, contiene il metodo main
ComboBoxRenderer	definisce l'aspetto grafico dei menù a tendina (combobox)
SpringUtilities	Fornisce metodi per creare layout basati su form o griglie, è utilizzata per disegnare il layout di NXTBBEmuConfJDialog
Utilities	Raccoglie metodi di utilità generale per il package

Le classi, il cui nome non contiene il prefisso NXTBBEmu sono da ritenersi di utilità generale.

La classe NXTBBEmuFrame contiene le seguenti classi interne, che costituiscono i componenti grafici dell'interfaccia

MenuBar	gestisce la barra dei menù
ToolBar	gestisce la barra degli strumenti
GenericNavigationPanel	implementa un generico pannello di navigazione, è utilizzata sia per l'intercaccia principale, sia per le finestre di dialogo modifica e inserisci
SimCtrlPanel	implementa gli strumenti di controllo della simulazione
CmdsPanel	implementa il pannello contenente la lista dei comandi
SimulationPanel	implementa il pannello della simulazione che tramite un OverlayLayout sovrappone AnimationPanel a BackgroundPanel
BackgroundPanel	implementa il componente che disegna l'immagine del mat sullo schermo e l'eventuale griglia sopra di essa
AnimationPanel	implementa il componente che disegna gli spostamenti del robot e la loro traccia
TargetsPanel	implementa il pannello con gli obiettivi da raggiungere
MatsPanel	implementa il pannello di scelta dei mat

NXTBBEmuFrame include inoltre queste altre classi interne:

NXTBBEmuFrameListener	implementa l'ascoltatore di eventi principale
MatSelector	implementa l'ascoltatore di eventi per il pannello MatsPanel
NavJDialog	implementa le finestre di dialogo "Modifica" e "Inserisci"
PopupMenu	Implementa il menù contestuale sulla lista dei comandi

Figura 6.1: Schema pannelli

La disposizione dei pannelli nel frame è gestita da un GridBagLayout.

6.2 Descrizione Listener principale

La classe NXTBBEmuFrame contiene un Listener che gestisce gli eventi della barra dei menù, della toolbar, del pannello di navigazione, del pannello di controllo simulazione e del pannello dei comandi

Per creare un'istanza di questo ascoltatore di eventi è sufficiente invocare new NXTBBEmuFrameListener(LSTNR_OPTION, "chiamante").

Il listener gestisce gli eventi generati attraverso due costrutti switch-case, uno dei quali risulta disattivato durante l'esecuzione della simulazione, in modo da disattivare la gestione di quegli eventi che interferirebbero con essa.

6.3 Descrizione simulazione

La simulazione è realizzata dal pannello AnimationPanel tramite la sovrascrittura del metodo paintComponent(Graphics g), che si occupa di ridisegnare il componente.

Da un punto di vista logico, il metodo disegna la posizione dell'immagine che rappresenta il robot sopra l'immagine del mat basandosi su di una tabella che memorizza orientamento, coordinate x ed y, tempo di persistenza dell'immagine, valore booleano che definisce se deve essere riprodotto il beep.

Orientamento	Coordinata x	Coordinata y	Tempo di persistenza frame	Beep
int [°]	int	int	int [ms]	boolean

Schema tabella simulazione

La tabella ha dimensioni variabili: all'apertura del programma la tabella contiene una sola riga, che descrive la posizione iniziale del robot; all'avvio della simulazione la tabella assume una dimensione pari a 6 * numero_azioni + 1(posizione iniziale), la variabile booleana che indica che la simulazione è attiva viene impostata a true e il metodo paintComponent inizia a rappresentare la simulazione; al termine della simulazione la tabella rimane immutata e la variabile booleana viene impostata a false.

Ogni singolo comando è rappresentato da 6 spostamenti/rotazioni del robot, e di conseguenza da 6 righe della tabella; trattandosi di un'animazione, e non di un filmato, un tale frame rate è sufficiente per garantire la persistenza della visione e di conseguenza un aspetto fluido della simulazione con tempo di permanenza di ogni singolo frame di 100 ms.

Per mantenere un'uniformità nell'implementazione anche le azioni "Pausa" e "Beep" sono rappresentate da 6 righe, nelle quali non cambiano orientamento e coordinate, ma vengono aumentati i tempi di persistenza. Per quanto riguarda l'azione "Beep", è impostata a true soltanto una delle 6 righe per garantire l'esecuzione di un singolo suono.

Nel programma ogni colonna di questa tabella è implementata come singolo array.

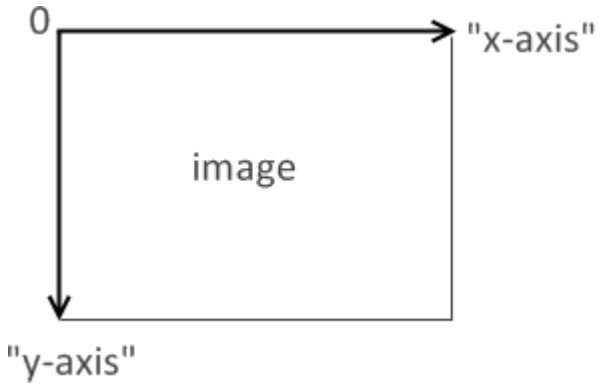

Figura 6.2 :Disposizione asse cartesiano nei pannelli Java

La tabella viene costruita incrementalmente aggiungendo le variazioni rispetto alla riga precedente, secondo lo schema sottostante.

		↑ 0		→ 90		↓ 180		← 270				
	°	x	y	x	y	x	y	x	y	ms	suono	Note
Avanti	0	0	- Δy	+ Δx	0	0	+ Δy	- Δx	0	0	false	1
Indietro	0	0	+ Δy	- Δx	0	0	- Δy	+ Δx	0	0	false	1
Sinistra	- ΔA	0	0	0	0	0	0	0	0	0	false	1, 2
Destra	+ ΔA	0	0	0	0	0	0	0	0	0	false	1, 3
Pausa	0	0	0	0	0	0	0	0	0	+ Δt	false	4
Suono	0	0	0	0	0	0	0	0	0	+ Δt	true	4, 5

Il valore ΔA è fisso è dipende dal numero di immagini che rappresentano una rotazione di 90°; nel caso specifico di questa implementazione è di 15°.

Il valore Δx è uguale a Δy, essendo le celle, su cui si muove il robot, quadrate; varia a seconda della dimensione della finestra.

Il valore Δt è variabile, dipende dalla velocità di simulazione impostata.

Note

1 incremento o decremento rispetto al valore precedente

2 se dopo l'operazione angolo < 0 allora angolo = 360 + (angolo)

3 se dopo l'operazione angolo > 360 allora angolo = angolo % 360

4 Incremento rispetto alla pausa di default

5 (0 < indice < n/2) false, (n/2) true, (n/2 + 1 <indice < n) false

6.4 Dimensionamento immagine attività (mat)

Il programma effettua un ridimensionamento automatico dell'immagine del mat in modo da ottenere l'immagine, i cui lati siano multipli di 6, che occupa il maggior spazio possibile nel pannello di simulazione. I lati devo essere multipli esatti di 6 al fine di riprodurre spostamenti in avanti o indietro esatti.

6.5 Interfaccia multilingua

L'applicazione è stata predisposta all'internazionalizzazione. La localizzazione avviene attraverso le classi java.util.Locale e java.util.ResourceBundle, che consentono una flessibile gestione del problema

La classe java.util.Locale consente all'applicazione di conoscere in quale viene lingua viene eseguita. La classe java.util.ResourceBundle è invece il contenitore degli oggetti locale-specific ovvero degli oggetti che assumono valori differenti in base al locale. Le risorse locale-specific presenti nell'applicazione sono, ad esempio, le stringhe che rappresentano una label dell'interfaccia grafica; l'applicazione internazionalizzata, tramite la classe NXTBBEmuRes, reperirà la stringa dal ResourceBundle specifico per il locale. Il locale corrente viene definito in base a quello del sistema operativo in uso, se l'utente non ha definito una specifica scelta nella finestra di selezione della lingua.

```
...
public class NXTBBEmuRes {
	private Locale currentLocale;
	private final ResourceBundle resourcebundle;

	public NXTBBEmuRes() {
		if (new NXTBBEmuConf().getProperty("LOCALE").length() > 0) {
			currentLocale =
				new Locale(new
NXTBBEmuConf().getProperty("LOCALE"));
		} else {
			currentLocale = Locale.getDefault();
		}
```

```
            resourcebundle =
                  ResourceBundle.getBundle("nxtbbemu.languages.Res",
currentLocale);
      }

      public NXTBBEmuRes(String language) {
            currentLocale = new Locale(language);
            resourcebundle =
                  ResourceBundle.getBundle("nxtbbemu.languages.Res",
currentLocale);
      }

     public String getString(String name) {
            return resourcebundle.getString(name);
      }
}
```

La stringa desiderata verrà ottenuta invocando il metodo getString("chiave_stringa"), su di un'istanza della classe NXTBBEmuRes.

```
NXTBBEmuRes res = new NXTBBEmuRes();

String s = res.getString("chiave_stringa");
```

6.6 Gestione configurazione

La configurazione è salvata tramite file .properties: file di testo contenenti un elenco di coppie chiave/valore, dove la chiave è un identificativo associato ad una determinata stringa e valore è la stringa stessa.

La configurazione predefinita è salvata nel file custom.properties contenuto all'interno del jar, mentre la configurazione personalizzata viene salvata nella cartella da cui viene eseguito il jar (che non è necessariamente quella in cui si trova il jar).

La gestione dei file di configurazione avviene attraverso la classe NXTBBEmuConf.

Al momento del caricamento della configurazione, se non viene trovato il file custom.properties, verrà caricato il file *default.properties*.

```
try {
      properties.load(new FileInputStream("custom.properties"));
} catch (FileNotFoundException  fnfe) {
      loadDefaults();
} catch (IOException ioe) {
```

```
        System.err.println(ioe);
}
```

La proprietà desiderata verrà ottenuta invocando il metodo getProperty (“proprietà”), su di un’istanza della classe NXTBBEmuConf..

```
NXTBBEmuConf conf = new NXTBBEmuConf ();

String locale = conf.getProperty("LOCALE");
```

6.7 Descrizione stampa su file .pdf

La creazione del pdf viene effettuata tramite la libreria iText: con essa è possibile creare molto velocemente e facilmente report anche complessi contenenti tabelle ed altri tipi di formattazione.
È sufficiente creare un oggetto di tipo Document, ottenere un’istanza di PDFWriter ed iniziare ad aggiungere elementi al documento.

iText è composto da una struttura gerarchica di elementi; l’elemento di testo base è il Chunk che è una porzione di testo che condivide lo stesso stile. Poi c’è il Phrase che può combinare diversi Chunk con font diversi, mentre il Paragraph può combinare più Chunk e Phrase tra di loro aggiungendo anche informazioni sul Layout come ad esempio i margini. Un altro elemento è l’Anchor che serve per definire dei riferimenti, ad esempio per creare degli shortcut da un Indice. Inoltre abbiamo le List per definire degli elenchi puntati e/o numerati, le Image per inserire le immagini, nei più disparati formati e le Table per inserire le tabelle, per le quali è possibile specificare lo stile dei bordi.

```
Document document = new Document();
try {
        PdfWriter.getInstance(document, new FileOutputStream(outFile));

        document.open();

        com.itextpdf.text.Font titleFont = new com.itextpdf.text.Font();
        titleFont.setSize(14);
        titleFont.setStyle(com.itextpdf.text.Font.BOLD);
```

```
        com.itextpdf.text.Font paragraphFont = new com.itextpdf.text.Font();
        paragraphFont.setSize(com.itextpdf.text.Font.DEFAULTSIZE);
        paragraphFont.setStyle(com.itextpdf.text.Font.NORMAL);

        Paragraph paragraph = new Paragraph();
        paragraph.setFont(paragraphFont);

        paragraph.add(new Chunk(res.getString("NXTBBEmuFrame_Activity") +
": " + matName + "\n\n", titleFont));
        paragraph.add(new
Chunk(res.getString("NXTBBEmuFrame_OriginalSituation") + "\n"));

        com.itextpdf.text.Image startImage =
com.itextpdf.text.Image.getInstance(startBImage, null);
        Dimension pdfDim = Utilities.imageResize(450, 400, 0, 0, 1, 1, matImg ,
0);
        startImage.scaleAbsolute((float) pdfDim.getWidth(), (float)
pdfDim.getHeight());
        paragraph.add(startImage);

        int achievedTargets = 0;

        if (selectedIndexesSum > 0) {
            Phrase phrase = new Phrase("\n\n\n\n" +
res.getString("NXTBBEmuFrame_Targets") + ": ");

            for (int i = 0; i < miniatureComboBox.length; i++) {
                int k = miniatureComboBox[i].getSelectedIndex();
                if (k > 0) {
                    phrase.add(new
Chunk(com.itextpdf.text.Image.getInstance(images[k].getImage(), null), 0,
0));
                }
            }
            paragraph.add(phrase);
        }

        paragraph.add(new Chunk("\n\n\n"));

        com.itextpdf.text.List list = new com.itextpdf.text.List(true,
runningCmdsList.length);

        for (int i = 0; i < runningCmdsList.length; i++) {
            Phrase listPhrase = new Phrase();
            listPhrase.add(new
Chunk(com.itextpdf.text.Image.getInstance(getImageIcon(runningCmdsList[i]).
getImage(), null), 0, 0));
            listPhrase.add(new Chunk(" " + runningCmdsList[i] + " "));
            listPhrase.add(new Chunk("\n\n"));
            list.add(new ListItem(listPhrase));
            if ((actionTargetTable != null) &&
(actionTargetTable.containsKey(i))) {
                com.itextpdf.text.List sublist = new
com.itextpdf.text.List(false, 1);
```

```
                    sublist.setListSymbol("");
                    Phrase sublistPhrase = new Phrase();
                    sublistPhrase.add(new Chunk("\n\n" +
res.getString("NXTBBEmuFrame_TargetAchieved") + ": "));
                    sublistPhrase.add(new
Chunk(com.itextpdf.text.Image.getInstance(images[actionTargetTable.get(i)].g
etImage(), null), 0, 0));
                    sublistPhrase.add(new Chunk("\n\n\n"));
                    sublist.add(new ListItem(sublistPhrase));
                    list.add(sublist);
                    achievedTargets++;
               }
          }

          paragraph.add(list);

          if (achievedTargets == 0) {
               paragraph.add(new
Chunk(res.getString("NXTBBEmuFrame_NoTargetsAchieved") + "\n\n\n"));
          }

          paragraph.add(new
Chunk(res.getString("NXTBBEmuFrame_FinalSituation") + "\n"));

          com.itextpdf.text.Image image =
com.itextpdf.text.Image.getInstance(bufferedImage, null);
          image.scaleAbsolute((float) pdfDim.getWidth(), (float)
pdfDim.getHeight());
          paragraph.add(image);

          document.add(paragraph);
          document.close();
```

6.8 Interfacciamento eseguibile nbc

L'applicazione utilizza le classi java.lang.Process e java.lang.Runtime per eseguire il compilatore nbc, catturarne l'output e il valore d'uscita restituito.

La cattura dell'output è da considerarsi un'operazione aggiuntiva, non necessaria al fine di compilare i file .nxc; l'output catturato è intenzionalmente nascosto all'utente al fine di non creare confusione, per visualizzarlo è necessario ingrandire la finestra di dialogo che compare dopo aver richiamato l'eseguibile nbc da interfaccia grafica.

```
JTextArea outputTA = new JTextArea();

...

private void execCmd() {
```

```
    Runtime r = Runtime.getRuntime();
    Process p;
    BufferedReader is;
    String line;

    try {
        JScrollBar verticalScrollBar = scrollingArea.getVerticalScrollBar();
        verticalScrollBar.setValue(verticalScrollBar.getMinimum());
        outputTA.setText("Output: \n");
        p = r.exec(command);

        is = new BufferedReader(new
InputStreamReader(p.getInputStream()));

        while ((line = is.readLine()) != null) {
            outputTA.append(line + "\n");
        }
        outputTA.append("\n\n");
        is.close();
        int processExitValue = p.exitValue();
        if (processExitValue == 0) {
            outputTA.append("   " + new
NXTBBEmuRes().getString("CmdExecJDialog_DownloadOK"));
        } else {
            outputTA.append("   " + new
NXTBBEmuRes().getString("CmdExecJDialog_CheckConnection"));
        }

    } catch (Exception exc) {
        outputTA.append(exc.getMessage() + "\n\n");
        outputTA.append("   " + "Error");
        System.err.println(exc.getMessage());
    }
}
```

6.9 Creazione output .txt

NXTBBEmu esporta la lista dei comandi anteponendo nel file 3 righe di commento indicanti la lingua in cui i comandi sono stati salvati, la data e l'ora del salvataggio e il mat a cui si riferivano. Il commento relativo alla lingua viene utilizzato da programma per ricarica la lista dei comandi; i restanti commenti attualmente vengono visualizzati all'utente in un messaggio di avvenuto caricamento al termine dell'apertura del file.

Dopo aver scritto i commenti sul file, viene accodata la lista dei comandi tramite il metodo Utilities.defaultListModelToFile.

```
try {
      PrintWriter pw = new PrintWriter(new FileWriter(file));
      pw.println("#" + res.getString("language"));
      pw.println("#" + dateTimeInstance.format(new Date()));
      pw.println("#" + matName);
      pw.close();
} catch (IOException ioe) {
      System.err.println(ioe);
}

Utilities.defaultListModelToFile(cmdsListModel, file, true);
```

6.10 Creazione output .nxc

NXTBBEmu è in grado di esportare la lista dei comandi in codice NXC strutturato secondo un paradigma di programmazione procedurale. Al fine di ottenere il file .nxc sono stati inclusi nell'applicazione dei file contenenti le varie parti di cui si compone il codice sorgente: l'intestazione HEADER contiene la definizione delle procedure, relative alle operazioni che il robot deve compiere, e l'apertura del task main(); FORWARD, BACKWARD, LEFT, RIGHT, BEEP e PAUSE contengono unicamente le chiamate alle procedure; TRAILER contiene il comando di spegnimento dei motori e la chiusura del task main().

Il programma si occupa di inserire, prima dell'intestazione, un preambolo contenente la definizione delle costanti necessarie al file NXC, i cui valori vengono letti dal file di configurazione e sono determinati dalle operazioni di taratura dell'NXC.

Successivamente vengono scritte le chiamate alle procedure determinate dalla lista comandi attuale e al termine il file viene chiuso con TRAILER.

```
#define MOTORS OUT_AC
#define POWER_PCT 50
#define BF_ANGLE 286
#define ROT_ANGLE 186
```

Esempio di preambolo

```
sub forward()
{
  ResetAllTachoCounts(MOTORS);
  RotateMotorEx(MOTORS, POWER_PCT, BF_ANGLE, 0, true, false);
}

sub backward()
{
  ResetAllTachoCounts(MOTORS);
  RotateMotorEx(MOTORS, -POWER_PCT, BF_ANGLE, 0, true, false);
}

sub pause()
{
  OffEx(MOTORS, RESET_ALL);
  Wait(1000);
}

sub right()
{
  ResetAllTachoCounts(MOTORS);
  RotateMotorEx(MOTORS, POWER_PCT, ROT_ANGLE, 100, true, true);
  Wait(50);
}

sub left()
{
  ResetAllTachoCounts(MOTORS);
  RotateMotorEx(MOTORS, POWER_PCT, ROT_ANGLE, -100, true, true);
  Wait(50);
}

sub beep()
{
  OffEx(MOTORS, RESET_ALL);
  PlayToneEx(262, 400, 3, FALSE);
  Wait(1000);
}


task main()
{
```

Contenuto di HEADER

```
  forward();
```

Contenuto di FORWARD

```
  backward();
```

Contenuto di BACKWARD

```
  left();
```

Contenuto di LEFT

```
  right();
```

Contenuto di RIGHT

```
  beep();
```

Contenuto di BEEP

```
  pause();
```

Contenuto di PAUSE

```
  Off(MOTORS);
}
```

Contenuto di TRAILER

Le funzioni NXC utilizzate per controllare i servomotori sono le seguenti

void **OffEx** (byte outputs, const byte reset)

Spegne i motori (arrestandoli) e resetta i contatori specificati

Parametri:

outputs Definisce le porte di output

reset resetta i contatori specificati

void **RotateMotorEx** (byte outputs, char pwr, long angle,
char turnpct, bool sync, bool stop)

Fa ruotare i servomotori per il numero specificato di gradi

Parametri:

outputs	Definisce le porte di output
pwr	Definisce la potenza di esercizio dei motori. Se negativa inverte la direzione
angle	Stabilisce l'angolo di rotazione
turnpct	Rapporto di curvatura. La direzione del robot dipende da questo parametro
sync	Sincronizza i motori, deve essere impostato a true se turnpct è diverso da 0
stop	Specifica se i motori devono essere bloccati al termine della rotazione o lasciati proseguire per inerzia

Si è inoltre provveduto a resettare i contatori prima dell’esecuzione di ogni comando

7 Conclusioni

Questa prima versione di NXTBBEmu consente di emulare ad un livello soddisfacente le funzionalità di Bee-Bot.

Lo sviluppo di questo software ha richiesto, una buona conoscenza di programmazione di interfacce grafiche in Java.

Gli sviluppi futuri del software possono essere svariati, si suggerisce in particolar modo l'utilizzo di un robot dotato di sensori per permettere spostamenti più accurati.

Un'altra possibilità di espansione futura potrebbe prevedere l'interfacciamento hardware di una tastiera, dotata dei tasti necessari per la programmazione in loco dell'NXT, aggiungendo nel software la capacità di fare un reverse engineering e recuperare quanto programmato nel robot, in modo da creare una piattaforma di apprendimento più completa.

Un'altra possibilità di espansione futura potrebbe essere l'integrazione di un strumento in grado di creare i file di attività per l'applicazione, costituiti da immagini vettoriali, che possono diventare dei veri e propri tappetini stampati, e da file descrittivi più articolati, magari xml, che descrivano il contenuto di ogni singola cella, per generare obiettivi più complessi.

8 Bibliografia

[1] LEGO® MINDSTORMS® NXT Hardware Development Kit, Version 1.00

[2] John Hansen. Not eXactly C Programmer's Guide, http://bricxcc.sourceforge.net/nbc/nxcdoc/NXC Guide.pdf

[3] Daniele Benedettelli. Programming LEGO NXT Robots using NXC, http://bricxcc.sourceforge.net/nbc/nxcdoc/NXC_tutorial.pdf

[4] Dimitris Alimisis, *Teacher Education on Robotics-Enhanced Constructivist Pedagogical Methods*, 2009, ASPETE, ISBN 978-960-6749-49-0

[5] Patrizia Battegazzore, *Bee-bot, fare robotica con un giocattolo programmabile a banalità limitata.* In A. Andronico, L. Colazzo (Eds), Didamatica 2009 – Informatica per la didattica, 2009, ISBN 978-88-8443-277-3.