

Java & Lego Mindstorms

Daniela Ruggeri <http://www.jia.it>

Stefano Sanna <http://www.jugsardegna.org>



Java Italian Association

LEGO Mindstorms

- LEGO... non ha bisogno di presentazioni! :-)
- Mindstorms Robotics Invention System
 - Sviluppato in collaborazione con il MIT, il RIS è il primo esempio di “intelligent brick” totalmente programmabile
 - CPU Hitachi 8-bit (16 Mhz)
 - 16 Kb di ROM, 32 Kb di RAM
 - 3 porte di input per i sensori
 - 3 porte di output per i motori
 - un display LCD a 5 caratteri e una porta di comunicazione a infrarossi.



LEGO Java Operating System

Versione stabile:

lejos_win32_2_1_0.zip

Versione con bug:

lejos.3.0.0-RC2-win32.zip

scaricabili dal sito <http://lejos.sourceforge.net/>



DIRECTORY LEJOS

- **bin** contiene gli eseguibili
- **classes** sorgenti java
- **common** contiene i files che sono usati per generare il codice per il linker e la virtual machine.
- **docs** documentazione
- **examples** esempi
- **gameboy_impl** file per la piattaforma Nintendo Game Boy
- **jtools** codice java base per il computer e codice del linker leJOS, usa i file di common/*.db.



DIRECTORY LEJOS

- **rcx_impl** contiene codice C specifico per il firmware RCX.
- **regression** contiene tutta una serie di test di regressione per testare leJOS
- **tools** contiene codice C per creare tools LeJOS
- **unix_impl** Questa directory contiene il codice necessario a creare un tool di emulazione Unix.
- **vmsrc** contiene codice C per la virtual machine.



Librerie jar

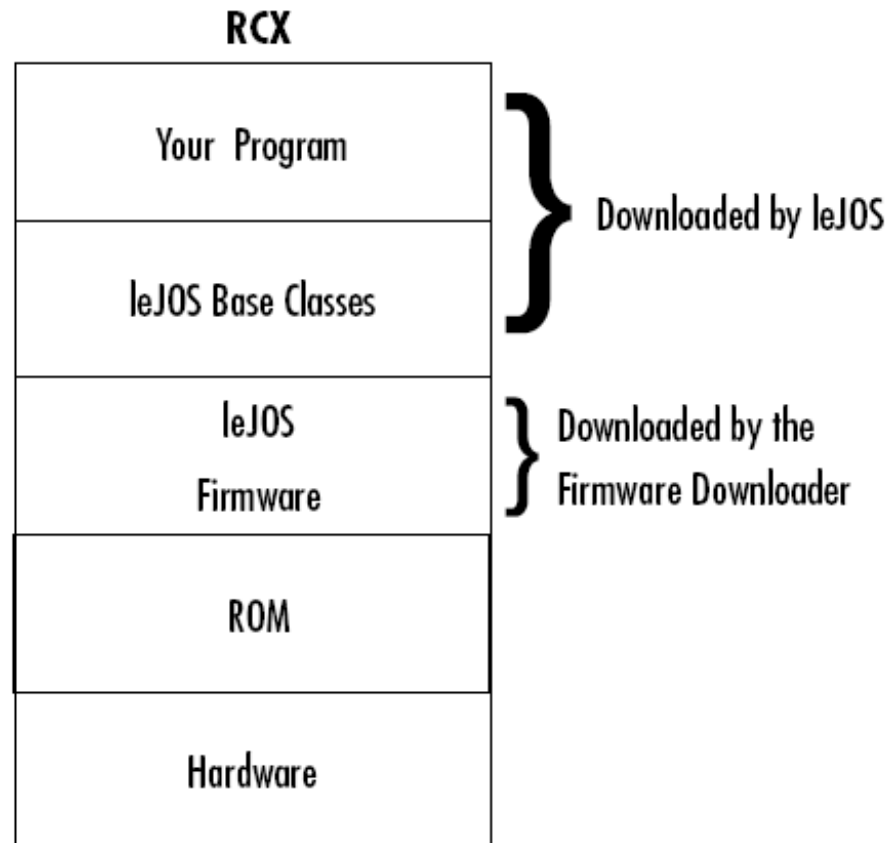
- ***rcxrcxcomm.jar*** – gestione sensori, motori e comunicazione da RCX al pc
- ***pcrcxcomm.jar*** – gestione comunicazione da pc a RCX

API Lejos

Packages

java.io	Supporto per Input/Output.
java.lang	Classi del linguaggio base Java
java.net	Supporto per Networking/sockets
java.util	Utilità
javax.servlet.http	Classi per gestire un Web Server con <u>lejos</u>
josx.platform.rcx	Classi per gestire sensori RCX, motori RCX ecc.
josx.rcxcomm	Classi per gestire la comunicazione tra RCX e il PC
josx.robotics	Classi e interfacce per gestire azioni e comportamenti dei robot in maniera parametrizzata
josx.util	Altre classi di utilità

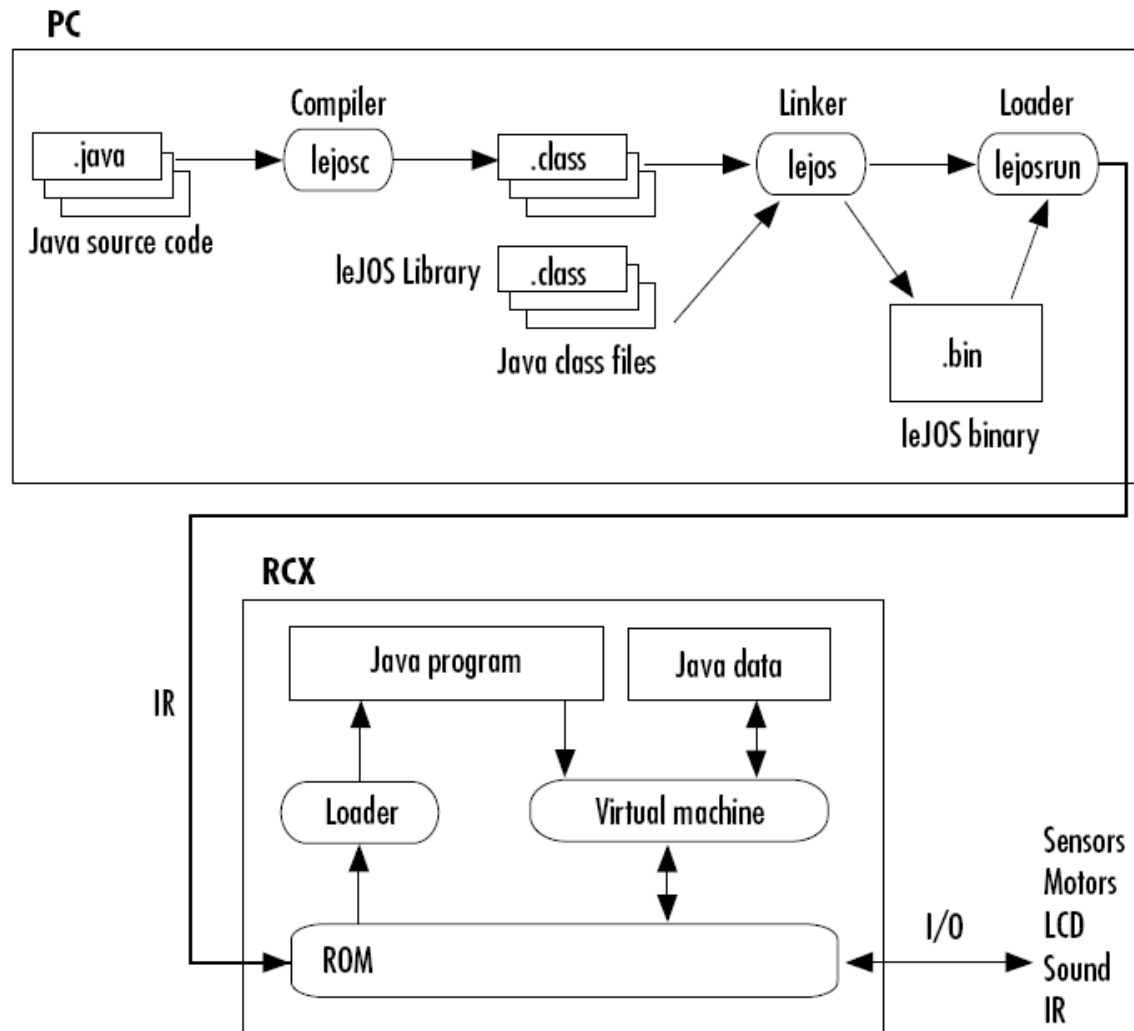
Architettura leJOS



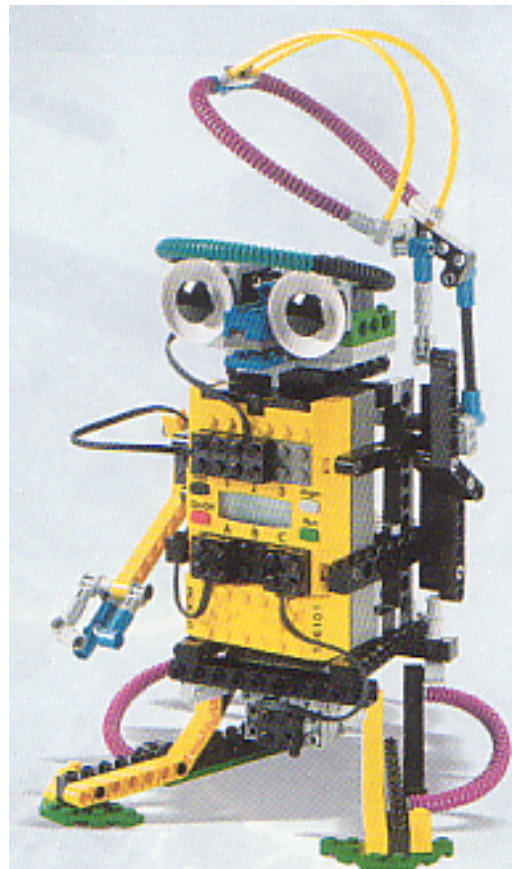
Comandi per RCX

- **lejosfirmidl** (nuovo **firmidl**) fa caricare il firmware in RCX
- **lejosjc** (nuovo **lejosjc**) compila i sorgenti java
- **lejosrun** trasferisce il programma binario in RCX tramite la torre
- **lejos** trasforma i programmi in file binari e li trasferisce in RCX tramite la torre IR Tower

Trasferimento codice



Inventorbot (robby)



Specifiche di programma

1. Fare in modo che il robot emetta dei suoni e alzi il cappello quando viene toccata la mano
3. Fare in modo che il robot emetta suoni diversi e giri in sensi diversi a secondo la durata della pressione esercitata sulla mano
5. Fare in modo che il robot saluti se viene colpito da una luce in faccia



Che serve sapere di Java?

- La base del linguaggio
- Gli eventi a delega
- I thread.



Calcolo del tempo

- **josx.util.Timer**

Oggetto Timer per gestire un intervallo di tempo

- **josx.util.TimerListener**

Interfaccia ascoltatrice usata con la classe
Timer

In particolare il metodo **timedOut()**
dell'interfaccia viene chiamato allo scadere
del tempo fissato.



Cambiamento stato sensore

- **osx.platform.rcx.Sensor**

Astrazione di un sensore. In particolare **Sensor.S1** rappresenta il sensore collegato alla porta 1

- **josx.util.SensorListener**

Interfaccia ascoltatrice usata con la classe **Sensor**

Il metodo **stateChanged(Sensor aSource, int aOldValue, int aNewValue)** per cambiamento stato del sensore.

- **osx.platform.rcx. SensorCostant**

Interfaccia usata con la classe **Sensor** di tutte le costanti riguardanti un sensore



Altre classi usate

- **osx.platform.rcx.Motor**
Astrazione di un motore. In particolare **Motor.A** e **Motor.C** rappresentano i motori collegati alle porte A e C
- **osx.platform.rcx.Button**
Astrazione di un bottone RCX. In particolare nell'esempio è utilizzato **Button.RUN** ma esistono anche **Button.VIEW** e **Button.PRGM**
- **osx.platform.rcx.Sound**
Classe usata per emettere suoni

Realizzazione calcolo del tempo

// questa variabile vale true se il tempo trascorso
 è minore di 3 secondi

```
boolean timeLess3 = true;
```

/* Creazione istanza della classe Timer
 associandola alla lista ascoltatrice che
 conterrà

il tempo che passa */

```
    Timer timer = new Timer(3000,new  

  TimerListener(){
```

// trascorsi 3000 millisecondi viene chiamato
 questo metodo

```
    public void timedOut() {  

      timeLess3 = false;  

    } });
```

Stati Sensore di contatto

```

Sensor.S1.activate(); // attiva il sensore collegato alla porta 1
Sensor.S1.setPreviousValue (0); // imposta valore iniziale del
sensore a 0 (rilasciato)
// imposta tipo di sensore (sensore di contatto) e valori in
output (booleani)
Sensor.S1.setTypeAndMode (SENSOR_TYPE_TOUCH,
SENSOR_MODE_BOOL);
Sensor.S1.addSensorListener(new SensorListener() {
// questo metodo viene chiamato quando il sensore cambia
dallo stato premuto a quello
// rilasciato e viceversa.
public void stateChanged(Sensor aSource,
int aOldValue,
int aNewValue) {
if (aOldValue == 1 || aNewValue == 0) {
// sensore in stato di premuto. Start del timer
timeLess3=true;
timer.start();
return;    }
}

```

Durata meno di 3 secondi

```
timer.stop();
try {
    if (timeLess3) {
        Sound.playTone(2100,300);
        Motor.C.forward();
        Thread.sleep(1000);
        Motor.C.stop();
        Sound.beep();
        Motor.A.forward();
        Thread.sleep(2000);
        Motor.A.stop();
        Motor.C.backward();
        Thread.sleep(1000);
        Motor.C.stop();
    }
}
```

Durata più di 3 secondi

```
else {  
    Sound.systemSound(true,3);  
    Motor.C.backward();  
    Thread.sleep(1000);  
    Motor.C.stop();  
    Sound.systemSound(true,4);  
    Motor.A.backward();  
    Thread.sleep(2000);  
    Motor.A.stop();  
    Motor.C.forward();  
    Thread.sleep(1000);  
    Motor.C.stop();  
    Sound.systemSound(true,5);  
    timeLess3=true;  
}
```

Stati Sensore ottico

```
Sensor.S3.activate(); // attiva il sensore collegato alla porta 1
Sensor.S3.setPreviousValue (0); // imposta valore iniziale del sensore a 0

// imposta tipo di sensore (sensore ottico) e valori in output (percentuale)
Sensor.S3.setTypeAndMode (SENSOR_TYPE_LIGHT, SENSOR_MODE_PCT);

Sensor.S3.addSensorListener(new SensorListener() {
// questo metodo viene chiamato quando il sensore è colpito dalla luce
public void stateChanged(Sensor aSource,
    int aOldValue,
    int aNewValue) {
    if (aNewValue > 80) {
        Motor.A.forward();
        Thread.sleep(2000);
        Motor.A.stop();
    }
}
```

Comandi shell per RCX

```
set classpath=.;c:\lejos\lib\pcrcxcomm.jar;  
c:\lejos\lib\rcxrcxcomm.jar
```

```
set path=c:\jdk1.4.0\bin;c:\lejos\bin;%path%
```

```
set RCXTTY=usb
```

```
lejos Inventorbot.java
```

```
lejos Inventorbot
```

LEGO Mindstorms NXT

- Caratteristiche hardware:
 - Doppio microcontrollore:
 - ARM7 32bit, 256 Kbyte FLASH, 64 Kbyte RAM
 - AVR 8bit, 4 Kbyte FLASH, 512 Byte RAM
 - 4 ingressi, 3 uscite
 - Interfacce USB e Bluetooth
 - Display grafico 100x64px
 - Audio playback 8KHz
 - Possibilità di realizzare network di quattro NXT (master + 3 slave)



NXT'reme: è tutto open!

- LEGO mette a disposizione della comunità degli sviluppatori risorse per implementare soluzioni originali:
 - Aggiornamenti firmware
 - Documentazione dettagliata dell'interfaccia hardware e delle porte I/O
 - Documentazione dettagliata del protocollo di comunicazione attraverso Bluetooth
 - Strumenti di sviluppo



WebNXT: controllare un robot via web

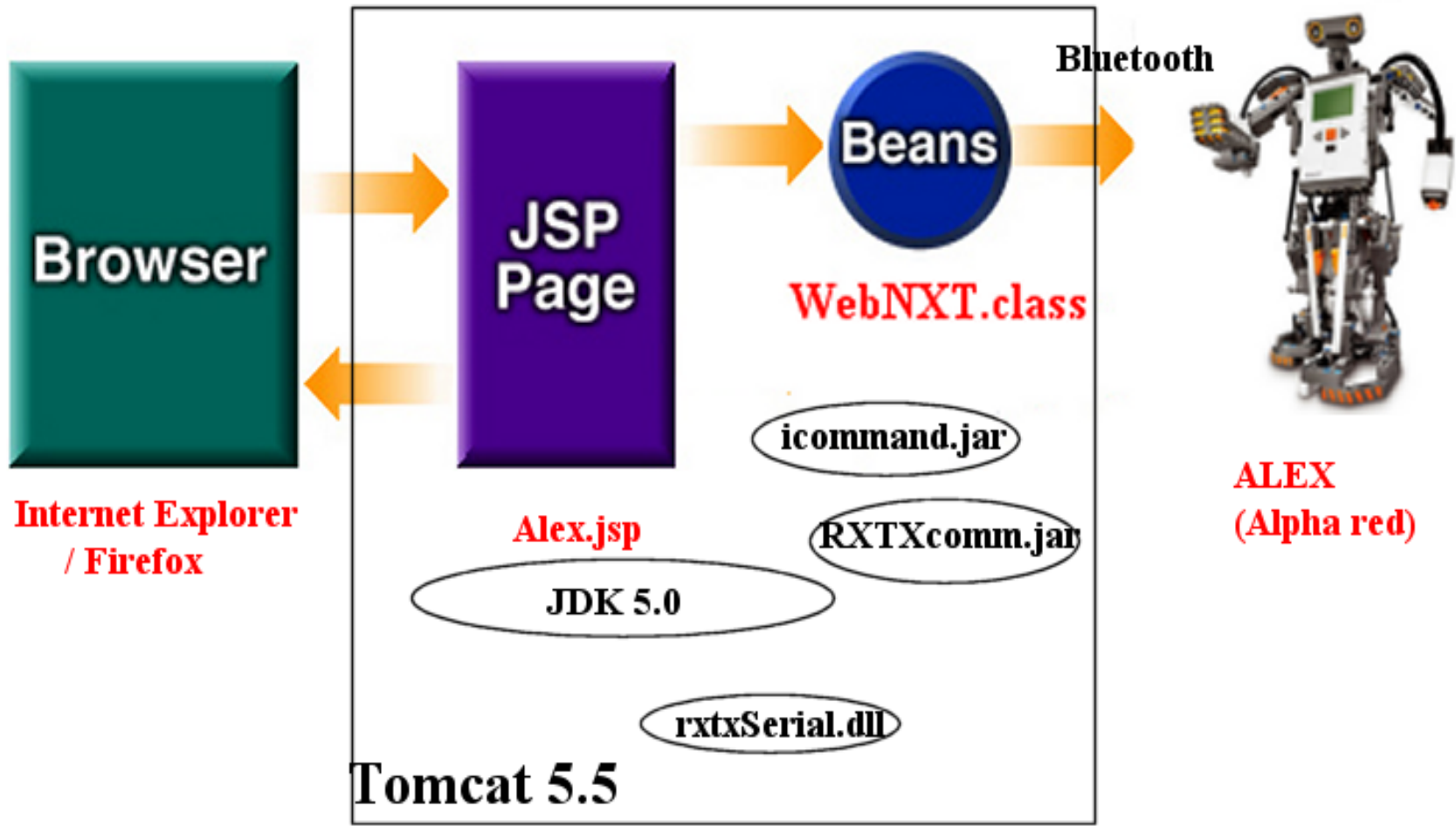
- Obiettivo: controllare da remoto un robot attraverso un browser web
- Cosa serve
 - l'ultima versione Java Standard Edition
 - un JSP/Servlet container
 - una maschera JSP + un Bean Java
 - alcune librerie Java per comunicare con il robot via Bluetooth
 - un browser web



Java Server Pages

- Tecnologia della Sun. Simile a Active Server Pages della Microsoft
- JSP permette di creare pagine HTML e DHTML.
- Utile per costruire pagine che contengono informazioni presenti su altre pagine, prelevate da un database acquisite da input, ecc. (per esempio interagire via bluetooth con un robot)

Architettura



Prodotti usati

- Java Standard Edition *versione 5.0*
<http://java.sun.com/javase/downloads/index.jsp>
- JSP/Servlet container *Tomcat 5.5*
<http://tomcat.apache.org/>
- *Alex.jsp* + il bean *WebNXT.java*
- Librerie java *icommand.jar* e *RXTXcomm.jar* e libreria nativa *rxtxSerial.dll*
<http://lejos.sourceforge.net/>
<http://www.rxtx.org>
- Il browser *Internet Explorer* o *Firefox*



Cosa è necessario fare

- Creare collegamento bluetooth con NXT
- Impostare nel file **icommand.properties** parametro **NXTCOMM**=<porta bluetooth>
- Copiare il file **icommand.properties** nella directory Tomcat Creare sotto la directory copiare il servizio robot sotto la directory webapps di tomcat
- Copiare la libreria nativa rtxSerial.dll nella directory \bin di Tomcat
- Preparare la webapplication robot con Alex.jsp e WebNXT.class Far partire Tomcat
- Caricare la pagina
<http://localhost:8080/robot/Alex.jsp>

WebNxt: Classi usate

- ***command.nxtcomm.NXTCommand***
Classe per la spedizione e ricezione comandi da /per l'NXT
- ***command.nxtcomm.NXTProtocol*** Classe contenente i comandi da/per l'NXT

Classe NXTCommand

Classe che contiene tutti i comandi da/per NXT spediti in questo formato

- **Byte 0** = parte LSB lunghezza messaggio
- **Byte 1** = parte MSB lunghezza messaggio
- **Byte 2** = Tipo comando (es. comando *sistema con risposta*)
- **Byte 3** = Comando (es. *riproduci un suono contenuto in un file*)
- **Byte 4...Byte n** (dati necessari al comando, es. *nome del file contenente il suono da riprodurre*)

Tipi di comandi (NXTProtocol)

Tipo	Byte	Descrizione
DIRECT_COMMAND_REPLY	0x00	Comando diretto con risposta
SYSTEM_COMMAND_REPLY	0x01	Comando di sistema con risposta
REPLY_COMMAND	0x02	Comando di risposta
DIRECT_COMMAND_NOREPLY	0x80	Comando diretto senza risposta
SYSTEM_COMMAND_NOREPLY	0x81	Comando di sistema senza risposta

Comandi di sistema (NXTProtocol)

Tipo	Byte	Descrizione
OPEN_READ	0x80	Apri file in lettura
OPEN_WRITE	0x81	Apri file in scrittura
READ	0x82	Leggi
WRITE	0x83	Scrivi
CLOSE	0x84	Chiude connessione
DELETE	0x85	Cancella file
FIND_FIRST	0x86	Trova il primo file nella directory e con il filtro specificati
FIND_NEXT	0x87	Trova il prossimo file nella directory e con il filtro specificati
GET_FIRMWARE_VERSION	0x88	Ritorna informazioni sul firmware
SET_BRICK_NAME	0x98	Imposta nome NXT
GET_DEVICE_INFO	0x9B	Ottiene informazioni sul dispositivo NXT
DELETE_USER_FLASH	0xA0	Cancella le informazioni nella memory Flash
POLL_LENGTH	0xA1	Ottiene la lunghezza di messaggio di risposta ad un comando
POLL	0xA2	Ottiene il messaggio di risposta ad un comando

Comandi diretti (NXTProtocol)

Tipo	Byte	Descrizione
START_PROGRAM	0x00	Lancia un programma nell'NXT
STOP_PROGRAM	0x01	Ferma un programma nell'NXT
PLAY_SOUND_FILE	0x02	Riproduce un suono di un file
PLAY_TONE	0x03	Riborduce un tono
SET_OUTPUT_STATE	0x04	Imposta parametri per un motore
SET_INPUT_MODE	0x05	Imposta parametri per un sensore
GET_OUTPUT_STATE	0x06	Ottiene parametri di un motore
GET_INPUT_VALUES	0x07	Ottiene parametri di un sensore
RESET_SCALED_INPUT_VALUE	0x08	Azzera impostazioni di un sensore

Comandi diretti (NXTProtocol)

Tipo	Byte	Descrizione
MESSAGE_WRITE	0x09	Spedisce un messaggio ad una casella inbox NXT
RESET_MOTOR_POSITION	0x0A	Azzerata contagiri del motore
GET_BATTERY_LEVEL	0x0B	Ritorna il livello della batteria
STOP_SOUND_PLAYBACK	0x0C	Ferma la riproduzione di un suono
KEEP_ALIVE	0x0D	Lascia acceso l'NXT
LS_GET_STATUS	0x0E	Ritorna lo stato del sensore a ultrasuoni
LS_WRITE	0x0F	Imposta informazioni al sensore a ultrasuoni
LS_READ	0x10	Legge informazioni di un sensore a ultrasuoni
GET_CURRENT_PROGRAM_NAME	0x11	Ottiene il nome del programma che sta girando

Es. Ottiene versione firmware

Comando spedito

- **Byte 0** = 0x02 LSB
- **Byte 1** = 0x00 MSB
- **Byte 2** = 0x01 tipo comando *sistema con risposta*
- **Byte 3** = 0x88 Comando GET_FIRMWARE_VERSION

Comando ricevuto

- **Byte 0** = 0x06 LSB
- **Byte 1** = 0x00 MSB
- **Byte 2** = 0x02 tipo comando *risposta*
- **Byte 3... Byte 8** = Dati Firmware

La pagina Alex.jsp

```
<jsp:useBean id="robotBeanId" scope="session"
  class="lego.WebNXT" />
```

```
<form method="post" action="Alex.jsp">
```

```
<% if (request.getParameter("chiudi") != null)
  robotBeanId.close();
  %>
```

```
<jsp:setProperty name="robotBeanId" property="*" />
<input type="submit" name="avanti" value="Avanti" >
<input type="submit" name="stop" value="Stop"><br>
<input type="submit" name="indietro" value="Indietro">
```

```
<input type="submit" name="saluta" value="Saluta"><br>
<input type="submit" name="chiudi" value="Chiudi
  Connessione Bluetooth"><br>
</form>
```

Risultato web

Test Alex NXT

Avanti

Stop

Indietro

Saluta

Chiudi Connessione Bluetooth



Comando Avanti e Indietro

```

public void setAvanti(String newValue)
    throws Exception {
    System.out.println("avanti: ");
    NXTCommand.setVerify(true);
    Motor.B.forward();
    Motor.C.forward();
}

public void setIndietro(String newValue)
    throws Exception {
    NXTCommand.setVerify(true);
    Motor.B.backward();
    Motor.C.backward();
}
  
```

Comando Saluta e Chiudi

```

public void setSaluta(String newValue)
    throws Exception {
    NXTCommand.setVerify(true);
    Motor.A.forward(1000);
    Speaker.playSoundFile("ciao.rso",false);
}
public void close() throws Exception {
    NXTCommand.close();
    Thread.sleep(10000);
}
  
```



JNXT: un controller gestuale

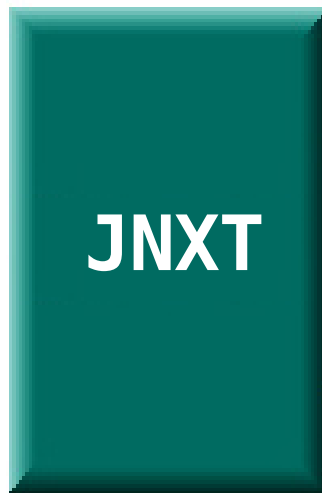
- Obiettivo: controllare un robot attraverso gesti (o tratti) anziché con dei pulsanti:
 - per rendere più immediata l'interazione uomo-macchina (il robot interpreta i gesti “umani” dell'utente anziché costringere questi a imparare un nuovo telecomando!)
 - rendere i robot pilotabili da utenti con disabilità motorie
- Componenti software:
 - Applicazione Java ME su PDA Windows Mobile
 - Libreria di comunicazione verso l'interfaccia Bluetooth (Serial Port Profile)

Cosa occorre

- Windows Mobile:
 - gestisce l'input attraverso touch screen
 - permette il collegamento a dispositivi Bluetooth
- Java ME CLDC/MIDP
 - permette di scrivere applicazioni per PDA
 - permette di riutilizzare numerose librerie scritte per i telefoni cellulari
- Applicazione e libreria di comunicazione:
 - JNXT per l'interazione gestuale
 - iCommand (scritto da Brian Bagnall) per il controllo low-level del NXT

Il touchscreen rileva i segni tracciati dall'utente

Il robot riceve i comandi via Bluetooth



JNXT traduce i tratti in comandi di alto livello (avanti, indietro...) e utilizza iCommand per i controlli a basso livello sui motori (Motore A avanti lentamente, Motore B indietro...)

JNXT: alcune problematiche

- Tecniche:
 - Porting di iCommand su CLDC
 - Scelta della VM
 - Supporto JSR 82 (Bluetooth API) su Windows Mobile
 - Emulazione input pen-based su desktop
- Semantiche:
 - Attribuzione di un significato ai tratti sullo schermo
 - Visualizzazione dei riscontri ottenuti dai sensori

Porting di iCommand su CLDC

- iCommand è modellato utilizzando esclusivamente classi Java 1.1: il porting è stato molto semplice:
 - Properties:
 - si è optato per una Hashtable popolata automaticamente in fase di inizializzazione della MIDlet
 - SerialPort:
 - Su Windows Mobile: CommConnection sostituisce le classi di JavaComm
 - Su Symbian OS: StreamConnection sarà utilizzata come connessione sullo schema btsp
 - La libreria per l'algebra in virgola mobile è ancora in fase di scelta (probabilmente verranno scritte alcune classi ad hoc)



Scelta della VM

- IBM offre una coppia di runtime environment (CDC/PP + CLDC/MIDP) di ottima qualità per Windows Mobile, Linux e PalmOS (solo CLDC)
- Vantaggi (versione CLDC):
 - molto economica
 - GUI ben integrata con il sistema operativo
 - supporto: JSR 75, JSR 135, CommConnection
 - input pen-based



Supporto JSR 82

- La VM di IBM non fornisce supporto alla JSR 82: non è possibile gestire lo stack Bluetooth all'interno dell'applicazione Java ME
- Il supporto nativo alle connessioni seriali permette di sfruttare i servizi del sistema operativo per il discovery e il pairing dei dispositivi

Bluetooth Serial Port Profile




Start 23.57

domenica 18 giugno 2006

- Imposta informazioni su utente
- Nessun messaggio da leggere
- Nessuna attività
- Nessun appuntamento programmato

Nuovo

- Attiva Bluetooth
- Collegamenti Bluetooth
- Impostazioni Bluetooth



Impostazioni Blueto 9.46

Bluetooth: Servizi

Servizi

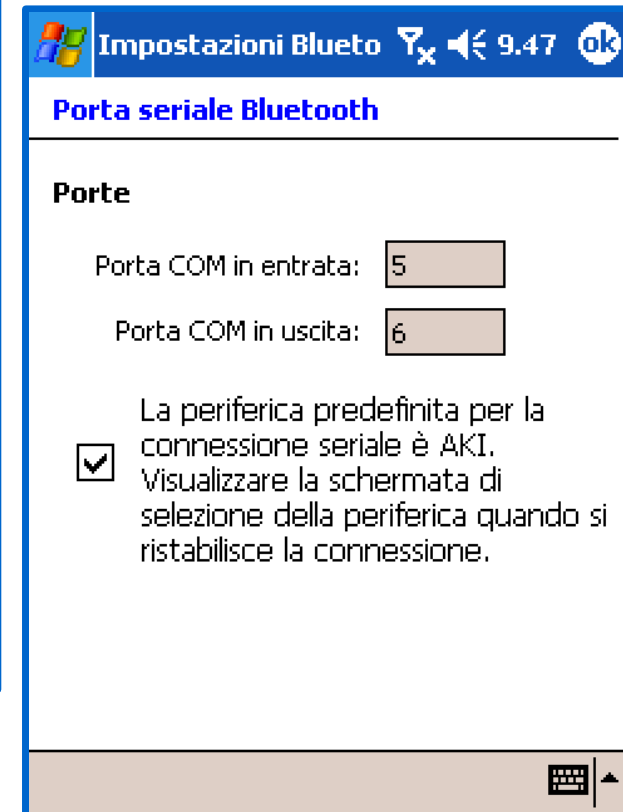
- Trasferimento file
- Scambio di informazioni
- Porta seriale**
- Server di rete personale

Impostazioni servizio

- Attiva servizio
- Richiede autorizzazione
- Richiede autenticazione (passepartout)
 - Richiede crittografia

Avanzate...

Generali | Accessibilità | Servizi | Informazioni su



Impostazioni Blueto 9.47

Porta seriale Bluetooth

Porte

Porta COM in entrata: 5

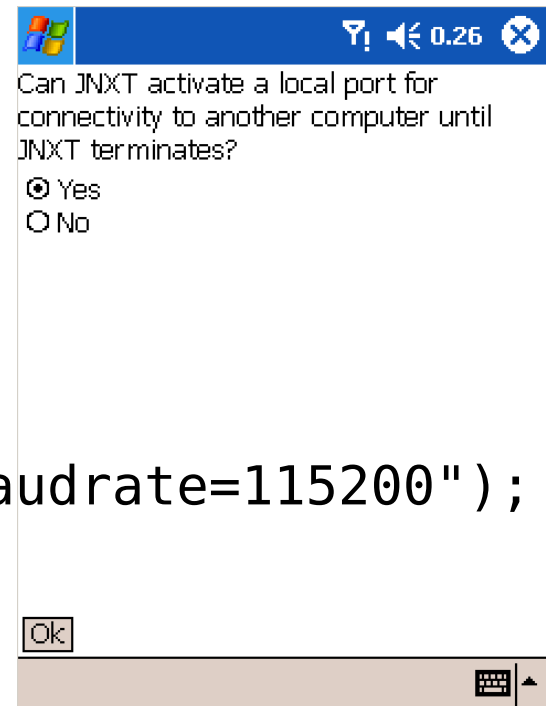
Porta COM in uscita: 6

La periferica predefinita per la connessione seriale è AKI. Visualizzare la schermata di selezione della periferica quando si ristabilisce la connessione.

Accesso alla porta seriale

- L'attivazione della CommConnection attiva la richiesta di autorizzazione da parte dell'utente e la scelta del device Bluetooth da utilizzare:

```
CommConnection commPort;  
commPort = (CommConnection)  
    Connector.open("comm:COM6;baudrate=115200");
```



Note sul Wireless Toolkit

- Il WTK fornisce emulatori “phone oriented”, senza supporto per l'input via penna. Per abilitarlo è sufficiente impostare un flag all'interno del file

WTK_DIR\wtllib\devices\DEVICE_NAME\DEVICE_NAME.properties

ad esempio:

c:\WTK23\wtllib\devices\DefaultColorPhone\DefaultColorPhone.properties

touch_screen=true

Semantica dei gesti /1



Vai avanti

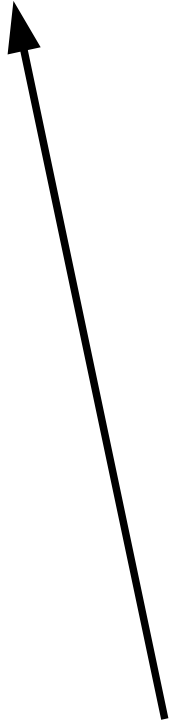


Torna indietro

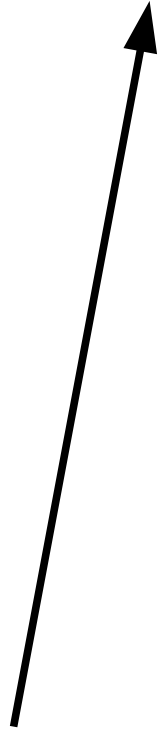


STOP

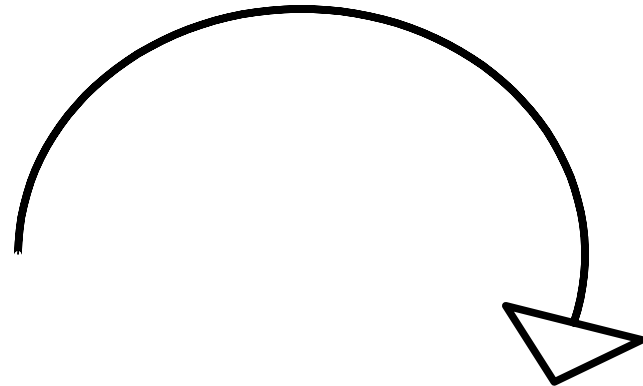
Semantica dei gesti /2



Avanti-sinistra

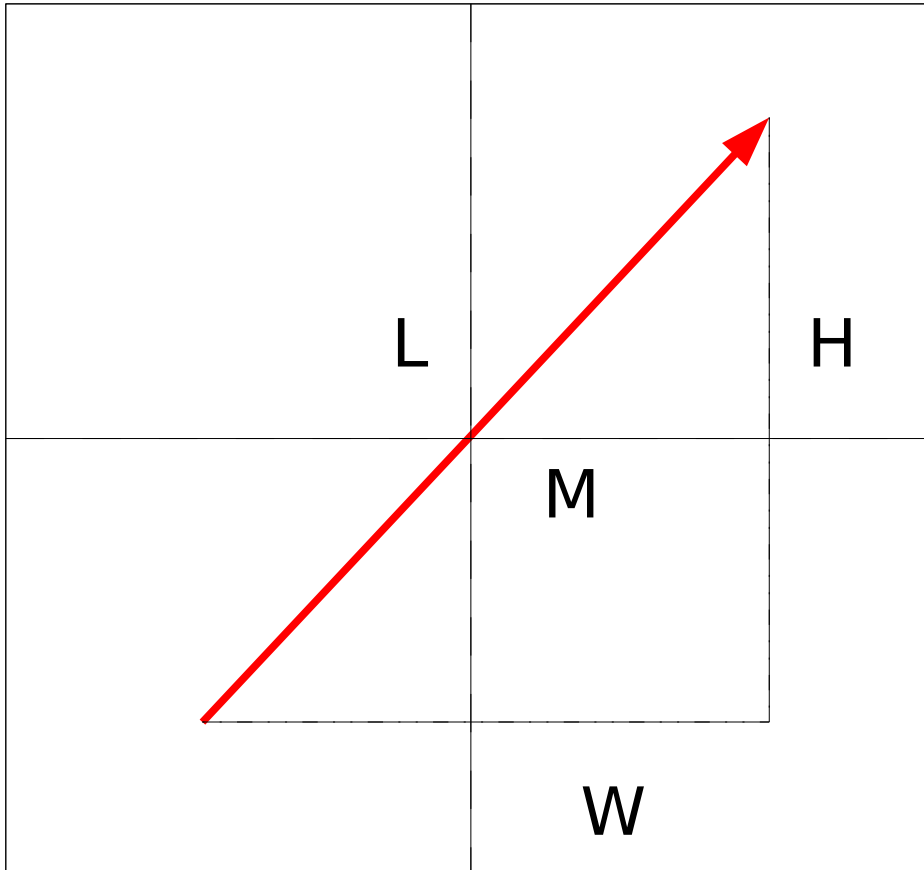


Avanti-destra



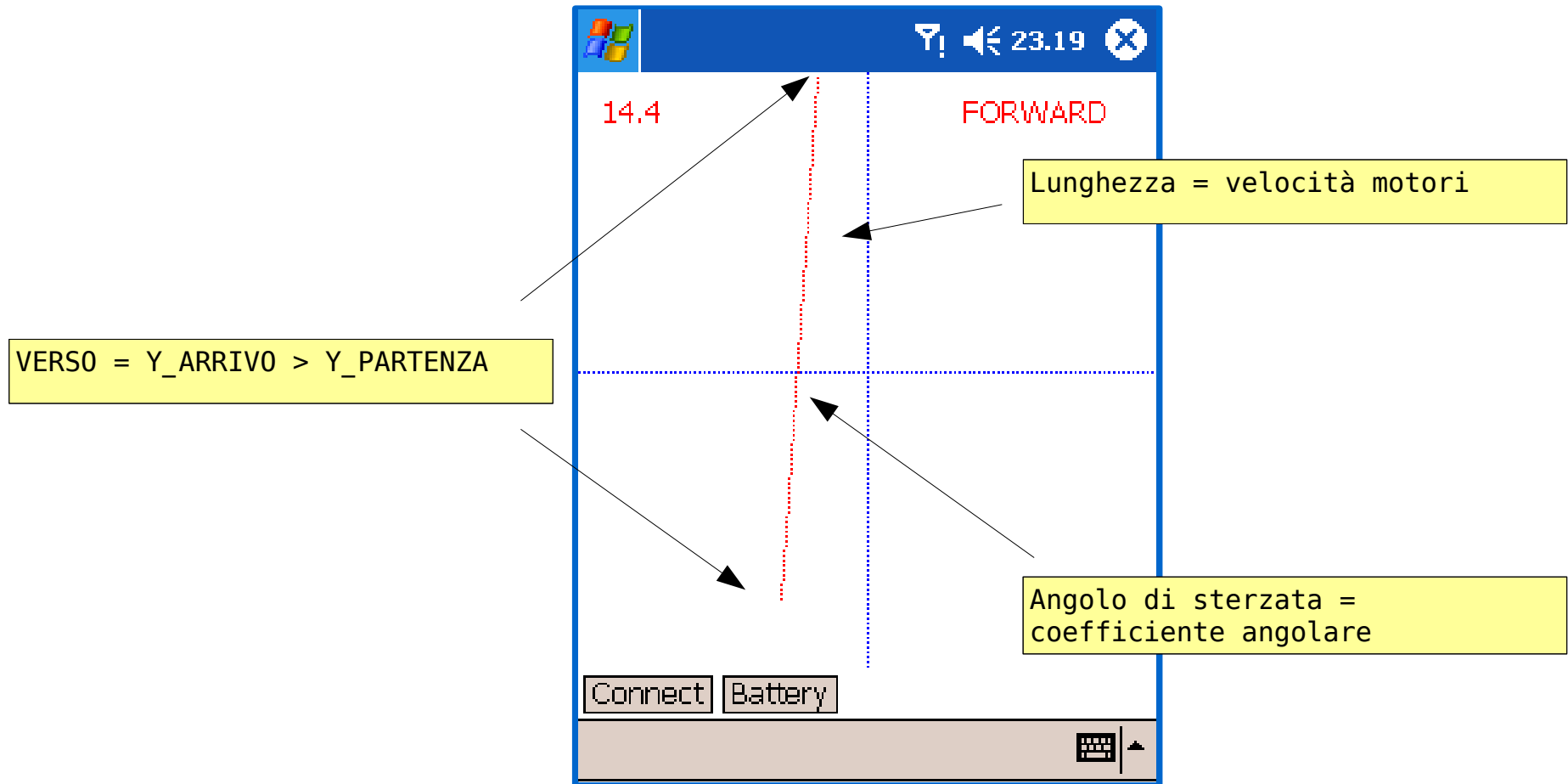
Torna indietro

Semantica dei gesti



- L: lunghezza della retta
- $M = H/W$:
coefficiente
angolare della retta
- $X2-X1$: direzione
(destra-sinistra)
- $Y2-Y1$: direzione
(avanti-indietro)

JNXT: semantica dei gesti



Pen-based input su MIDP

- Il Canvas su MIDP in ambiente PDA supporta la gestione degli eventi relativi al puntatore (penna):
 - **hasPointerEvents()** e **hasPointerMotionEvents()** permettono di conoscere a runtime le caratteristiche del dispositivo
 - **pointerPressed()**, **pointerDragged()** e **pointerReleased()** sono i metodi di callback invocati quando lo stato del puntatore cambia. I metodi ricevono le coordinate x,y del punto al quale l'evento si riferisce

Il codice su JNEXT /1

```

public void pointerPressed(int x, int y) {
    firstX = x;          firstY = y;
    released = false;
}

public void pointerReleased(int x, int y) {
    float Y = (firstY - y);          float X = (x - firstX);
    float m = Y / X;
    lastM = m;
    getDirection(m);
    lastX = x;
    lastY = y;
    repaint();
}

public void pointerDragged(int x, int y) {
    lastX = x;          lastY = y;
    repaint();
}
    
```

Il codice su JNXT /2

```
private void getDirection(float m) {  
  
    boolean forward = lastY < firstY;  
  
    boolean right = lastX > firstX;  
  
    m = Math.abs(m);  
  
    if (m < 0.8) {  
  
        lastMessage = "STOP";  
  
        listener.stop();  
  
    }  
}
```

Evoluzioni

- JNXT sarà sviluppato per supportare:
 - feedback sul PDA in base alle rilevazioni dei sensori (flash, colori, vibrazione)
 - possibilità di tracciare percorsi curvilinei
 - sostituzione della penna con un accelerometro
 - maggiore interazione con altri dispositivi Bluetooth: RFID reader, SMS machine...

Riferimenti JNXT

- Pagina di JNXT:
 - <http://www.gerdavax.it/jnxt>
- iCommand:
 - <http://lejos.sourceforge.net/>
- IBM Virtual Machine
 - http://www.ibm.com/developerworks/websphere/zones/wireless/weme_eval_runtimes.html
- Java ME su PDA
 - http://www.gerdavax.it/data/PDA_JMDF2006.pdf

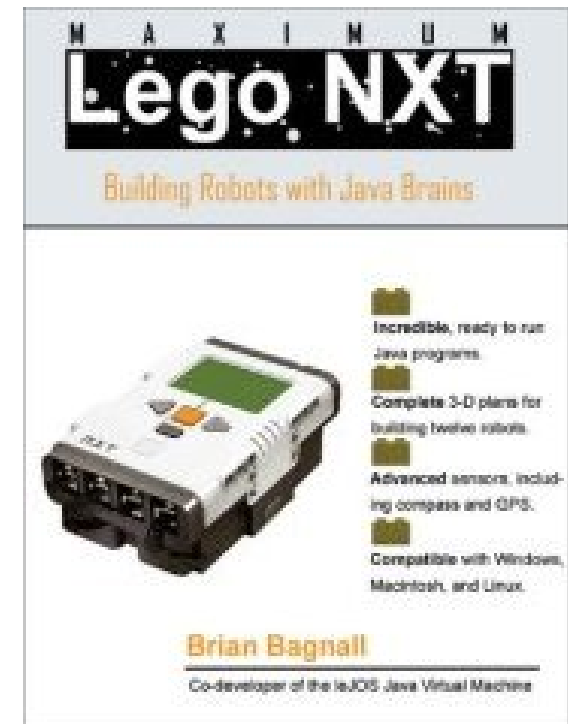
Un libro di prossima uscita

- **Maximum Lego NXT: Building Robots with Java Brains**

Brian Bagnall

Variant Press, 2007

ISBN: 0973864915





Grazie per l'attenzione.