Controles Remotos Alternativos para Lego® Mindstorms® NXT 2.0

Adriana Postal[1], Gustavo Henrique Paetzold[1], Josué Pereira de Castro[1], Tiago Zilio Jesuino[1]
[1]UNIOESTE - Universidade Estadual do Oeste do Paraná
Laboratório de Robótica Inteligente
Rua Universitária, 2069. Jardim Universitário.
Caixa Postal 711 - CEP 85819-110 Cascavel, PR

adriana.postal@unioeste.br, ghp_91@hotmail.com,
josue.castro@unioeste.br, ti_lantra@hotmail.com

Resumo. *O presente trabalho consiste na discussão a respeito do desenvolvimento de novas ferramentas de controle para robôs criados a partir do kit de montagem Lego® Mindstorms® NXT 2.0. As propostas aqui apresentadas visam explorar os recursos não utilizados pelo controle remoto já embutido no software de desenvolvimento em blocos da Lego. Duas alternativas são apresentadas, uma desenvolvida pela plataforma de programação móvel Java™ ME, e outra implementada com bibliotecas básicas, através do ambiente de programação MATLAB®.*

1. Introdução

Quando se trata de conectividade, a área da robótica traz consigo uma grande variedade de possibilidades interessantes de desenvolvimento. Além da criação de robôs independentes, capazes de executar diversas funções através de algoritmos, existe também a possibilidade de se criar uma interface de comunicação entre humano e robô.

O uso de ferramentas para o serviço de controle remoto para robôs já vem sendo explorado há bastante tempo. Na área de automação industrial, por exemplo, algumas tarefas de manufatura de produtos exigem precisão mecânica, muitas vezes por questão de segurança, ao mesmo tempo em que requisitam o manuseio de um funcionário especialista.

Os controles aqui implementados têm como objetivo utilizar as funções e os comandos disponíveis na documentação disponibilizada pela Lego®, explorando as duas possibilidades de conexão, USB™ (*Universal Serial Bus*) [1] e *Bluetooth*® [2], suportadas pela peça de controle central do Lego® Mindstorms® NXT 2.0 [3].

O artigo aqui apresentado é dividido da seguinte forma: a seção 2 descreve o kit de montagem Lego® Mindstorms® NXT 2.0; a seção 3 mostra como é organizada a estrutura de conexão a ser explorada pelos controles; a seção 4 apresenta a implementação do controle modelado no ambiente MATLAB® [4]; a seção 5 apresenta a implementação do controle programado em Java™ [5]; e a seção 6 discute as considerações finais do trabalho.

2. O Lego® Mindstorms® NXT 2.0

O Lego® Mindstorms® NXT é uma linha do brinquedo Lego®, lançada oficialmente em 2006, voltada para a educação tecnológica. Ele possui um módulo controlador de programação intuitiva, também conhecido como *Smart Brick*, que é o "cérebro" do robô,

tendo quatro entradas que manipulam os seus sensores e três saídas que manipulam os seus três servo-motores. Sua arquitetura é simples, possui um microprocessador ARM7 de 32 *bits*, memória Flash de 256 *Kbytes* e memória RAM de 64 *Kbytes*, comunicação *Bluetooth*® 2.0, porta de comunicação USB™ 2.0, uma tela LCD e um alto falante. O *Smart Brick* está ilustrado na figura 1.

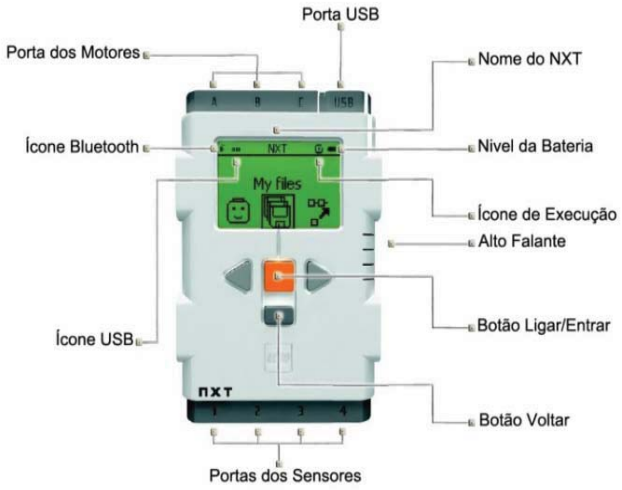

Figura 1. *Smart Brick* controlador do *kit* Lego® Mindstorms NXT® 2.0 [6]

Os controles apresentados neste artigo (utilizando o Java™ ME e o MATLAB®) utilizam o mesmo modelo de robô para a estruturação dos comandos, o *Shooterbot* mostrado na figura 2, que é um dos modelos padrões do *kit* Lego® Mindstorms® NXT® 2.0 para iniciação na montagem de robôs.

Figura 2. *Shooterbot*, modelo base para o desenvolvimento dos controles [7]

O *Shooterbot* possui em sua estrutura um motor destinado ao disparo de projéteis, conectado à porta de saída A do *Smart Brick*, e se move através de duas esteiras posicionadas nas laterais, sendo a esteira da esquerda conectada na porta de saída B, e a esteira da direita na porta de saída C.

3. Estrutura de Conexão e Comunicação no Lego® MINDSTORMS® NXT 2.0

3.1. Hierarquia de Conexão USB™ entre Dispositivos

A conexão USB™ (*Universal Serial Bus*) presente no *Smart Brick* é do tipo "Escravo", ou seja, é ativada através da energia proveniente da porta USB™ do computador, ou de outro dispositivo. Logo essa característica impede a conexão do Lego® diretamente com outros dispositivos que também possuem uma conexão USB™ do tipo "Escravo". Embora o desenvolvimento do controle utilizando o MATLAB® aqui apresentado tenha se dado através da utilização da conexão USB™, a biblioteca RWTH-Mindstorms NXT *toolbox* [8] permite que a conexão deste controle seja estabelecida tanto via *Bluetooth*® quanto via USB™.

3.2. Hierarquia de Conexão *Bluetooth*® entre Dispositivos

O gerenciador de conexões *Bluetooth*® do *Smart Brick* permite no máximo quatro canais de informação simultâneos, organizados em um esquema tradicional de Mestre e Escravo.

O aparelho que será designado como "Mestre" não pode trabalhar como Escravo ao mesmo tempo, e gerenciará toda a informação recebida pelos outros dispositivos. O canal principal de conexão *Bluetooth*® do Lego® NXT é responsável por todo o tráfego de informações entre *Smart Bricks* semelhantes, enquanto os outros três canais disponíveis são destinados a conexões com dispositivos de outra categoria, como celulares e computadores, que podem também trabalhar como "Mestre" ou "Escravo". É através de um destes canais que será estabelecido o canal de troca de informações entre o controle remoto, que foi implementado em Java™, e o robô.

4. Implementação do Controle Remoto Usando o MATLAB®

4.1. Decisões Iniciais de Projeto

A biblioteca utilizada no desenvolvimento deste projeto não é a única biblioteca que fornece ferramentas e suporte para a interação entre o MATLAB® e o Lego® Mindstorms® NXT. Existem outras bibliotecas que permitem interação entre o robô e o software, por exemplo, a *Ecrobot* NXT (*Embedded Coder Robot NXT Demo*) [9], mas devido a algumas dificuldades como complexidade de instalação e de configuração dessas bibliotecas, decidiu-se focar os estudos e a pesquisa para o desenvolvimento deste controle, apenas na biblioteca RWTH – Mindstorms NXT *toolbox*, sendo que esta apresenta uma fácil instalação e configuração.

A principal vantagem no uso da biblioteca RWTH – Mindstorms NXT *toolbox*, é que ela permite combinar aplicações robóticas com operações matemáticas complexas, permitindo visualizações no MATLAB®, abrindo possibilidades ilimitadas para oferecer inteligência artificial ao robô, além de transferir o processamento dos comandos do Lego® Mindstorms® NXT que seria realizado pelo *Smart Brick*, para o computador.

4.2. O software MATLAB®

O MATLAB® é um software interativo de linguagem de alto nível interpretada que permite a execução de tarefas intensivas em relação ao contexto computacional, permitindo resolver problemas de forma mais rápida do que em linguagens de programação tradicionais como o C, C++ e FORTRAN [10].

Este software possui um ambiente interativo de alto desempenho e de fácil utilização, onde os problemas e as soluções são expressos como são escritos matematicamente. Este software trabalha essencialmente com um tipo de objeto, uma matriz numérica retangular podendo conter elementos complexos, sendo adequado àqueles que desejam implementar e testar soluções com facilidade e precisão sem perder tempo com detalhes específicos de linguagem de programação.

4.3. A Biblioteca RWTH–Mindstorms NXT *toolbox*

A biblioteca RWTH – Mindstorms NXT *toolbox* teve seu desenvolvimento motivado pela universidade RWTH Aachen (*Rheinisch-Westfälische Technische Hochschule Aachen*) [11] para estudantes de engenharia elétrica e, portanto para fins de educação. Essa biblioteca possui funções que podem ser classificadas em uma estrutura de múltiplas camadas.

Na primeira camada estão as funções de ajuda e de conversão de parâmetros, palavras e bytes, todos determinados pela documentação dos comandos diretos fornecida pela própria Lego®. Na segunda camada estão os comandos diretos ao NXT que podem ser identificados pelo prefixo *NXT_**, esta camada inclui também funções para empacotamento de comandos *Bluetooth*®. Na terceira camada estão presentes as funções de alto nível para controlar os motores do NXT, seus sensores e a conexão *Bluetooth*®. Na quarta e última camada se encontram funções de alto nível para regulação dos motores, ajustes de precisão e outras utilidades. Essas e outras informações podem ser conseguidas na documentação disponível na página da biblioteca [8].

4.4. A ferramenta para desenvolvimento de interfaces no MATLAB®

O MATLAB®, além de toda a interatividade de um ambiente de programação, possui uma ferramenta especialmente voltada à criação de interfaces gráficas, a famosa GUI (*Graphical Users Interfaces)* [12]. Com esta ferramenta foi possível o desenvolvimento de uma interface simples e amigável de controle para o Lego® Mindstorms® NXT, como será apresentado na seção 4.6.

4.5. Implementação e Desenvolvimento do Controle Remoto

A biblioteca RWTH – Mindstorms NXT *toolbox* em conjunto com o MATLAB® oferece uma grande variedade de opções para o controle e interação com o Lego® Mindstorms® NXT, sendo que um mesmo comando pode ser implementado e executado de maneiras diferentes, sem apresentar diferenças entre o grau de programação entre eles.

Os comandos implementados seguem a estrutura de programação apresentada na figura 3. Foram implementados os quatro comandos básicos de orientação ou direção, sendo eles, `frente`, `direita`, `esquerda` e `atrás`. Além destes, também foi implementado um controle deslizante que funciona para ajustar a velocidade dos motores do robô, um botão para início e fim de uma conexão com o Lego®, além de dois botões extras baseados no *Shooterbot*.

```
NXC_MotorControl(Port, Power, TachoLimit,
SpeedRegulation, ActionAtTachoLimit,
SmoothStart, handle);
```

Figura 3. Estrutura de programação dos comandos dos motores no MATLAB utilizando a biblioteca RWTH – Mindstorms NXT *toolbox*

4.6. A Interface do Controle Remoto

Na interface implementada há um botão "LIGA/DESLIGA" para iniciar e posteriormente finalizar a conexão do computador com o Lego® Mindstorms® NXT, um controle deslizante "velocidade" para regulação da velocidade dos servo-motores, controles de direção que servem para movimentação e controle do robô enviando comandos aos motores B e C, além de dois botões para comandos extras que nesta versão são "Buzina" que envia um comando para que o robô emita um bipe e o botão "Atirar" que envia um comando para que o motor A do robô gire 360º graus. A imagem da interface é apresentada na figura 4.

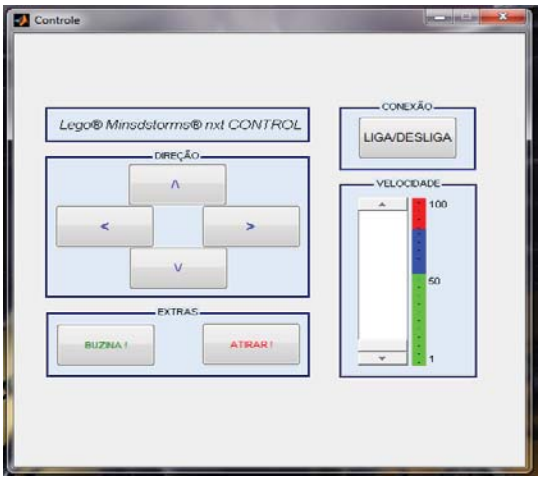

Figura 4. Interface do controle desenvolvido no MATLAB®

5. Implementação do Controle Remoto em Java™ com Conexão *Bluetooth*®

5.1. Decisões Iniciais de Projeto

Através da análise da documentação para desenvolvimento *Bluetooth*® da Lego® [13], e também dos exemplos de sistemas programados na linguagem NXC (*Not eXactly C*) para o robô [14], foram encontradas duas maneiras distintas de se gerenciar o fluxo de instruções da conexão entre os dois dispositivos.

A primeira maneira se dá através da elaboração de um sistema segmentado em duas partes, uma embarcada no dispositivo (programada na linguagem Java™), e outra no robô desejado (programada na linguagem NXC), e a segunda maneira se dá através do protocolo de comunicação direta fornecido pela Lego®, que permite a conexão entre o controle remoto embarcado no periférico e o *firmware* instalado no robô.

A diferença principal entre as duas possíveis abordagens de desenvolvimento do controle remoto é ilustrada na Figura 5.

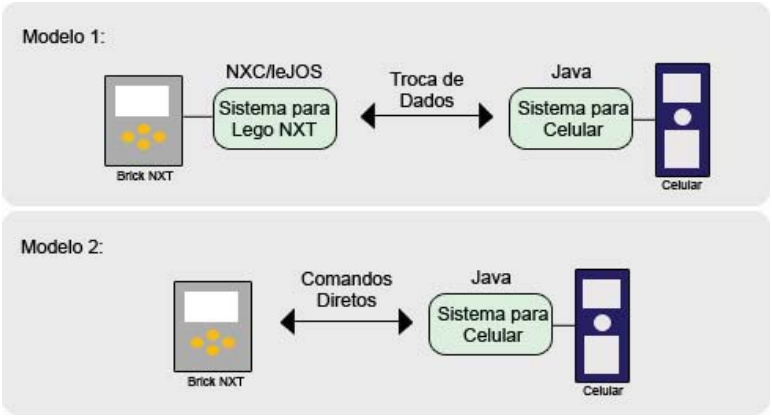

Figura 5. Possíveis modelos de implementação do controle remoto

No modelo 1 existe a necessidade da implementação de ambos os sistemas para o robô e celular, gerando uma série de obstáculos a serem transpostos como compatibilidade de código, sincronização entre os dois sistemas, gerenciamento de memória, etc.

O modelo 2 requer apenas que o sistema para celular seja implementado, deixando a responsabilidade de interpretação dos comandos diretos com o *Smart Brick*. Pela maior simplicidade da implementação, o modelo 2 foi o escolhido para a modelagem do controle.

5.2. Padrões de Comandos Diretos

Para que o *firmware* instalado no *Smart Brick* possa interpretar de maneira correta as informações que foram enviadas do celular, é necessário que os dados estejam organizados de maneira específica.

Através dos parâmetros fornecidos pela documentação encontrada no site da Lego® [3], é possível controlar e monitorar os motores e sensores conectados ao *Smart Brick*. Os dados a serem interpretados devem ser enviados como cadeias de bytes, podendo variar de tamanho de acordo com a instrução especificada. O modelo de cadeia de bytes aceito pelo interpretador é ilustrado na Figura 6.

LSB	MSB	Tipo de Comando	Comando	Dados (4-N bytes)

Figura 6. Padrão de comando direto do Lego® Mindstorms® NXT

Cada byte da cadeia corresponde a uma informação específica do comando direto, que deve ser representada em hexadecimal. Em resumo, o comando direto pode ser descrito da seguinte forma:

-Byte 0 (LSB): Tamanho da cadeia, desconsiderando os dois primeiros bytes.

-Byte 1 (MSB): Índice do byte de maior significância na cadeia.

-Byte 2 (Tipo de Comando): Identificador numérico do tipo da tarefa a ser executada.

-Byte 3 (Comando): Identificador numérico da tarefa.

-Byte 4-N (Dados): Parâmetros referentes à tarefa.

Como exemplo, a Figura 6 ilustra a estrutura da cadeia de bytes que deve ser enviada para que seja requisitado o estado atual da bateria sendo utilizada no *Smart Brick.*

LSB	MSB	Tipo de Comando	Comando
0x02	0x00	0x00	0x0B

Figura 7. Cadeia de bytes para requisição de estado da bateria

O sistema, ao receber a cadeia de bytes descrita na Figura 7, envia como resposta uma cadeia de resposta contendo as informações referentes à carga restante da bateria do dispositivo.

Através da grande variedade de comandos diretos documentados pela Lego®[13], é possível que o controle remoto tenha domínio completo sobre os motores e sensores conectados ao *Smart Brick* em uso.

5.3. Sistema de Busca e Conexão

Para que o aplicativo seja compatível com qualquer robô criado a partir do kit de montagem Lego® Mindstorms® NXT, é necessária a implementação de um sistema de reconhecimento que possa não só identificar quais são os robôs ao alcance do celular, mas também que permita o usuário escolher com qual dos robôs deseja interagir.

O processo de escolha do robô a ser controlado começa com a busca por dispositivos detectáveis, que estejam ao alcance do celular. Através do uso de classes contidas no pacote de bibliotecas Bluecove [15] para programação *Bluetooth*® em Java™, é possível a criação de um sistema de busca descomplicado e eficiente. As classes utilizadas para a implementação do módulo de busca são as seguintes:

- `DiscoveryListener`: Consiste em uma interface Java™, destinada à implementação da classe responsável pelo tratamento das informações dos dispositivos encontrados na busca.
- `DiscoveryAgent`: Classe que, quando instanciada, gera um agente para controlar o processo de busca de dispositivos. Contém o método `startInquiry()`, responsável pelo início do processo de busca de dispositivos.
- `RemoteDevice`: Cada instância da classe `RemoteDevice` contém todas as informações do dispositivo *Bluetooth*® encontrado.

A estrutura de dados escolhida para armazenar as informações adquiridas pelo agente de busca, é composta pelas classes `DeviceList` e `RDElement`:

- `DeviceList`: Lista simplesmente encadeada de elementos `RDElement`, utilizada para armazenar em uma lista todas as informações de identificação dos dispositivos.
- `RDElement`: Elemento da classe `DeviceList`. Armazena uma instância da classe `RemoteDevice` e o nome do dispositivo encontrado.

Utilizando todas as classes acima descritas, o armazenamento das informações se torna simples. Uma vez encontrados, os dispositivos podem ser listados na tela, permitindo então que o usuário possa escolher qual robô deseja controlar.

Após o processo de seleção, é necessário que a conexão entre celular e robô seja feita. Para isso, um segundo conjunto de classes, já contidas no pacote Bluecove, é necessário para que o modelo correto de conexão seja feito:

- `Connector`: Através do método `open()`, abre uma URL de conexão passada por parâmetro.

- `StreamConnection`: Utiliza a conexão criada pela classe `Connector` para instanciar os atributos e métodos necessários para a troca de dados.
- `InputStream`: Classe destinada à entrada de informações através do link *Bluetooth*®.
- `OutputStream`: Classe destinada à saída de informações através do link *Bluetooth*®.

Uma vez criada, a conexão gera automaticamente os objetos de envio e recebimento de dados (`OutputStream` e `InputStream`) através da classe `StreamConnection`.

No momento em que as instâncias das classes `InputStream` e `OutputStream` estiverem acessíveis, a comunicação entre celular e *Smart Brick* já pode ter início. As cadeias de bytes referentes a cada comando direto podem ser enviadas através do método `write()`, contido na classe `OutputStream`, enquanto o método `read()`, da classe `InputStream`, é destinado ao recebimento de informações requisitadas do *Smart Brick.*

5.4. Funcionamento dos Comandos Implementados

Como mencionado em tópicos anteriores, o controle remoto aqui descrito é capaz de controlar qualquer robô criado a partir do kit de montagem Lego® Mindstorms® NXT, porém um padrão de montagem deve ser estabelecido para que o robô escolhido possa se movimentar de maneira adequada utilizando os comandos implementados no controle remoto.

Observando a disposição dos motores no *Shooterbot* (figura 2), um padrão de comandos, descrito na Tabela 1, foi estabelecido para a movimentação dos motores implementados no controle remoto.

Comando implementado	**Sequência de comandos correspondente**
Mover o robô para frente	Girar os motores B e C no sentido horário de maneira sincronizada.
Mover o robô para trás	Girar os motores B e C no sentido anti-horário de maneira sincronizada.
Girar o robô para a esquerda	Girar o motor B no sentido anti-horário, e o motor C no sentido horário.
Girar o robô para a direita	Girar o motor B no sentido horário, e o motor C no sentido anti-horário.
Disparar projétil	Girar o motor A em qualquer sentido.

Tabela 1. Comandos de movimentação implementados no controle remoto

Além de movimentar o robô, o controle é capaz de regular a potência dos motores através de uma variável, permitindo assim que o usuário tenha domínio da velocidade com que o robô atira e se movimenta.

5.5. Modelagem da Interface do Controle Remoto

Uma vez implementadas as estruturas de busca, conexão e comunicação, a ferramenta final a ser associada ao controle é a interface, elaborada no intuito de facilitar a interação entre sistema e usuário.

Para aumentar a iteratividade com que o controle é usado, cada comando deve ser associado a algum botão do teclado, e para determinar tais associações, foi utilizada a simetria entre os botões na estrutura física do celular e as direções de movimento do robô. A correspondência de comandos escolhida é mais claramente representada na Figura 8, ao lado esquerdo da interface gráfica final incorporada no controle remoto.

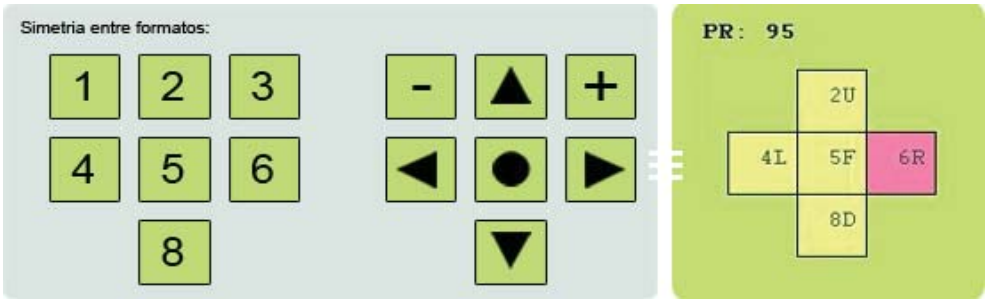

Figura 8. Simetria entre representações e Interface Gráfica do celular

6. Considerações Finais

Ambas as ferramentas de programação utilizadas no desenvolvimento dos controles, MATLAB® e Java™ ME [16], conseguem explorar todo o potencial dos canais de conexão USB™ e *Bluetooth*®, mostrando desempenho, robustez e estabilidade durante a troca de informações.

Apesar da modelagem dos controles ser voltada para a movimentação do robô, as informações adquiridas com os testes demonstram quão grande é o conjunto de possibilidades criadas. A precisão e consistência com que os comandos são executados indicam que robôs mais complexos poderiam ser controlados de maneira eficiente, sem perda de desempenho.

Para projetos consequentes, existe a possibilidade da implementação de interfaces compatíveis com aparelhos que possuem tela sensível ao toque, e também o aproveitamento dos canais de conexão para redistribuição de processamento entre o robô e dispositivos externos, permitindo assim que robôs autônomos possam processar quantidades mais densas de informação.

Referências

[1] USB™ Universal Serial Bus. “Universal Serial Bus”. Disponível em: http://www.usb.org/home. Acessado em: 06 Julho 2011.

[2] Bluetooth® SPECIAL INTEREST GROUP. “Welcome to Bluetooth.org”. Disponível em: https://www.bluetooth.org/apps/content/. Acessado em: 10 Junho 2011.

[3] Lego MINDSTORMS®. “What is NXT”. Disponível em: http://mindstorms.lego.com/en-us/whatisnxt/default.aspx. Acessado em: 28 Junho 2011.

[4] MathWorks®. “MATLAB-The Language Of Technical Computing”. Disponível em: http://www.mathworks.com/products/matlab/description1.html. Acessado em: 06 Julho 2011.

[5] Java™. “O que é Java”. Disponível em: http://www.java.com/pt_BR/download/whatis_java.jsp. Acessado em: 25 Junho 2011.

[6] Smart Brick NXT. Imagem modificada. Disponível em: http://www.robotdiy.com/images/LEGO_Mindstorms_NXT/LEGO_NXT_Detail.gif. Acessado em: 19 setembro 2011.

[7] Shooterbot is a moving vehicle robot that can guard your room and will shoot balls at intruders. Imagem modificada. Disponível em: http://us.mindstorms.lego.com/en-us/history/default.aspx. Acessado em: 19 setembro 2011.

[8] LFB Lehrstuhl für Bildverarbeitung. "RWTH-Mindstorms NXT Toolbox". Disponível em: http://www.mindstorms.rwth-aachen.de/trac. Acessado em: 12 Julho 2011.

[9] MATLAB® Central. "Embedded Coder Robot NXT Demo". Disponível em: http://www.mathworks.com/matlabcentral/fileexchange/13399. Acessado em: 20 Julho 2011.

[10] MathWorks®. "MATLAB Product Description". Disponível em: http://www.mathworks.com/products/matlab/description1.html. Acessado em: 06 Julho 2011.

[11] RWTH Aachen University. "About RWTH". Disponível em: http://www.rwth-aachen.de/aw/zentral/english/Themes/~lw/About_RWTH/. Acessado em: 18 Julho 2011.

[12] MathWorks®. "Creating Graphical User Interfaces". Disponível em: http://www.mathworks.com/help/techdoc/creating_guis/bqz79mu.html. Acessado em: 22 Julho 2011.

[13] Lego MINDSTORMS®. "Bluetooth Developer Kit". Disponível em: http://mindstorms.lego.com/en-us/support/files/default.aspx, Acessado em: 18 Junho 2011.

[14] Next Byte Codes & Not eXactly C. "NXC Sample Programs". Disponível em: http://bricxcc.sourceforge.net/nbc/nxcsamples/index.html, Acessado em: 17 Junho 2011.

[15] Bluecove. "Bluecove Documentation". Disponível em: http://bluecove.org/. Acessado em: 19 Julho 2011.

[16] Knudsen, J. "Beginning J2ME: From Novice to Professional", New York, 2005, Third Edition.