# Estudo de Caso Usando o Framework LeJOS

**Nilton M. de Souza[1], Dalton M. Tavares[1]**

Departamento Ciências da Computação – Universidade Federal de Goiás
Avenida Dr. Lamartine Pinto de Avelar, 1120, Setor Universitário, Catalão – GO – Brasil

nilton.mendes.2@gmail.com, dalton.tavares@catalao.ufg.br

***Abstract.*** *This paper describes how to use the LeJOS framework in the scope of an experience that deploys a route, using elements called WayPoints. WayPoints determine "points" in space, previously marked by the user, describing a path to be followed. The LeJOS is and open source project that allows the use of the Java language to control the LEGO® Mindstorms® NXT kit. It has a powerful API that allows the easy implementation of activities, using sensors and motors. The project uses a simple mobile robot, made from the LEGO® kit, showing how the logic and coding was made in order to reach the objective.*

***Resumo.*** *Este artigo descreve a utilização do framework LeJOS no âmbito de um experimento que desenvolve uma rota, através de elementos chamados WayPoints. WayPoints determinam "pontos" no espaço, previamente marcados pelo usuário, formando uma trajetória a ser seguida. O LeJOS é um projeto open source que permite a utilização da linguagem Java para manipulação do kit LEGO® Mindstorms® NXT. O LeJOS possui uma poderosa API que permite implementar várias atividades facilmente, como manipulação de sensores e motores. O projeto utiliza um robô móvel simples, que é feito a partir do kit LEGO®, mostrando como foi feita a programação e a lógica utilizada para alcançar o objetivo.*

## 1. Introdução

LeJOS é um *framework* para desenvolvimento de aplicações usando o kit LEGO® Mindstorms® NXT. Para desenvolver este estudo de caso foi usado um kit LEGO® Education modelo 9797, o qual contempla um bloco inteligente, rodas, pneus, engrenagens e peças para montagem de protótipos, motores e sensores. Neste caso, o kit foi usado para montar um pequeno robô móvel, a ser utilizado no experimento. As peças fundamentais para a montagem de qualquer protótipo são três motores, quatro sensores (som, ultra-sônico, intensidade luminosa e toque), o bloco inteligente com uma pequena tela de LCD, cabos para conectar os motores e sensores ao bloco inteligente, um cabo USB para conectar o bloco inteligente ao computador e interface *Bluetooth*.[Oliveira, Thiago et al. 2008]

O *framework* LeJOS proporciona a utilização da linguagem JAVA, com várias bibliotecas e recursos distribuídos, em conjunto com uma máquina virtual bem simples, a qual é inserida no bloco inteligente substituindo o *firmware* original fornecido pela LEGO®. Além disso, também existem algumas bibliotecas para a contra parte hospedada em um computador, incluindo vários exemplos de programas para a operação do robô

ou dispositivo, controlado via computador ou via programa hospedado no bloco inteligente NXT [Lew 2010].

O LeJOS detém uma API que possui bibliotecas que fornecem suporte ao sensor luminoso, sensor ultra-sônico, sensor de toque, sensor sonoro, aos três motores e o estabelecimento de conexão via USB ou Bluetooth. Outro recurso importante é o suporte a *multithreading*, muitas vezes necessário para que duas ou mais atividades concorrentes sejam realizadas.[LeJOS Team 2008c] Um exemplo seria o uso concorrente de dois sensores e dois motores, onde cada um compõe uma tarefa diferente, sendo representados cada um em uma *thread* específica.

Conexões via USB e *Bluetooth* são muito importantes, pois permitem a expansão dos recursos nativos do bloco inteligente. Este é muito limitado comparado a um computador ou *smartphones* atuais. Dessa forma, parte do processamento pode ser realizado fora do bloco inteligente (ex. em um notebook ou PC) e informações de interesse podem ser trocadas via USB ou *Bluetooth*, garantindo o controle do dispositivo pretendido. Essas conexões também são utilizadas para enviar os programas ao bloco inteligente.

Utilizando o *framework* LeJOS e o kit de desenvolvimento LEGO® Mindstorms® NXT, este artigo irá apresentar um estudo de caso envolvendo um planejador de trajetória simples por meio da determinação manual de pontos. Para tanto, será apresentada uma breve revisão bibliográfica sobre o *framework* LeJOS na seção 2, o estudo de caso será descrito na seção 3 e uma breve conclusão e propostas de trabalhos futuros serão discutidos na seção 4.

## 2. Organização do *Framework* LeJOS

O LeJOS permite a programação de robôs LEGO® Mindstorms® NXT/RXC utilizando a linguagem JAVA. Este *framework* surgiu de uma evolução do TinyVM, que é uma pequena máquina virtual para Java e um *firmware* para substituir o sistema original do LEGO® Mindstorms® NXT [LeJOS Team 2008b]. .

Para instalação, o próprio site do LeJOS fornece o *link* para *download* do pacote, através de um repositório no site *sourceforge*. Este código está disponível para Mac OS, Linux e Windows necessitando apenas de alguns pacotes descritos na seção 3.

O pacote contém:

- Um *firmware* para substituir o *firmware* original presete no bloco inteligente do kit LEGO® Mindstorms®;
- Uma biblioteca que auxilia no desenvolvimento de aplicativos para PC e para o bloco inteligente NXT;
- Um endereçador (*linker*) que transforma o código java, para uma linguagem compreendida pelo bloco inteligente do NXT;
- Ferramentas para compilação, envio de arquivos, modificação do *firmware* e *stream* entre o computador e o NXT;
- Vários exemplos de programas e códigos .

## 3. Estudo de Caso

O LeJOS possui uma vasta documentação para programação em sua linguagem, instalação, configuração, além de tutoriais iniciais[1]. Para o contexto desse artigo, será usada a versão para Linux Ubuntu 12.04 em conjunto com a IDE Eclipse® configurada para o desenvolvimento em JAVA e um *plugin* específico para o LeJOS. Essa configuração facilita a criação, compilação e envio dos programas para o bloco inteligente.

Além da máquina virtual Java e do software LeJOS, também deverão ser instalados o "`ant`" e a biblioteca "`lib-usb`". O `ant`[2] é um *maker*, baseado em Java, seu objetivo é construir projetos em Java a fim de serem executados em qualquer máquina que tenha uma máquina virtual java. A `lib-usb` é importante pois permite uma conexão via *usb* entre dispositivos sem que exista um *driver* instalado.

Após a instalação do ambiente de desenvolvimento, deve ser criado um usuário (ex. `lego`) e a regra de operação do dispositivo para o gerenciador de dispositivos (`udev`). A regra criada permite a descoberta do bloco inteligente via barramento USB e a definição do grupo de acesso e permissões (Listagem 1).

```
BUS=="usb", SYSFS{idVendor}=="03eb", GROUP="lego", MODE="0660"
BUS=="usb", SYSFS{idVendor}=="0694", GROUP="lego", MODE="0660"
```

**Listagem 1. Regras para o `udev` definidas em `/etc/udev/rules.d/lego-mindstorms-nxt.rules`.**

O estudo de caso proposto visa apresentar um planejador de trajetória simples. Em uma implementação posterior, pode ser realizado o planejamento de movimento (*motion planning*). O planejamento de movimento leva em consideração duas tarefas bem definidas: planeja o caminho (*path planning*) e planeja a trajetória (*trajectory planning*). O planejamento de caminho diz respeito a geração de um caminho livre de colisões em um ambiente com obstáculos e sua otimização com relação a algum critério. Um planejador de trajetória deve fornecer movimentos pré-programados ao longo de um caminho planejado [Hachour 2008].

O planejador de trajetória pretendido deve permitir que seja traçada uma rota manualmente, inserindo pontos em um padrão de *waypoints* dados três pontos X, Y e uma orientação (ou *heading*) que consiste na posição em relação a frente do carro, evitando que ele percorra um dado grau de rotação pelo pior caminho, o que poderia prejudicar a execução do caminho traçado. Posteriormente a rota armazenada deve ser executada a partir dos pontos gravados e suas orientações. Uma representação do protótipo do veículo é apresentada na Figura 1.

O protótipo construído possui duas rodas frontais, e dois motores conectados a cada uma, e uma pequena roda traseira. O motores são conectados por meio de cabos RJ11 ao bloco inteligente que os controlam, e as rodas são ligadas por pequenas engrenagens. A estrutura do veículo é feita de pequenos componentes plásticos do kit.

1 Para maiores informações acesse http://lejos.sourceforge.net/. Acesso em: 14/01/2013.
2 Para maiores informações acesse http://ant.apache.org/manual/. Acessado em: 17/01/2013

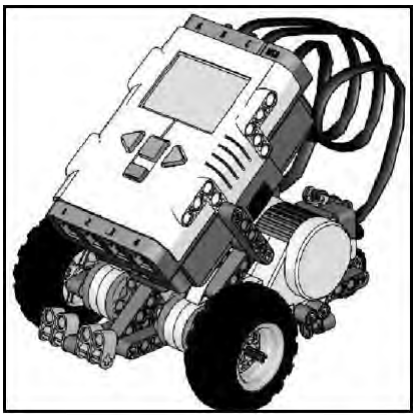

**Figura 1: Protótipo do veículo**

A seção 3.1 fornece as bases para o experimento proposto usando o *framework* LeJOS. Para tanto, serão descritos os fundamentos de operação dos motores e as funções utilizadas para o planejamento de trajetória.

## 3.1. Planejamento de Trajetória no LeJOS

Para se desenvolver um planejador de trajetória simples usando o LeJOS, deve-se levar em conta a leitura dos sensores e motores do kit LEGO® Mindstorms® NXT. Estes são diretamente conectados a portas específicas, sendo os motores ligados às portas, A, B e C e os sensores às portas 1, 2, 3 e 4. Não existe uma especificação formal de uma dada ordem para a ligação de motores ou sensores, porém, o programador deve saber a qual porta um dado dispositivo está relacionado. Para o estudo de caso em questão, serão apresentados os fundamentos de operação dos motores utilizando o *framework* LeJOS.

### 3.1.1. Manipulando Motores

Conforme discutido anteriormente, os motores do kit LEGO® Mindstorms® NXT podem estar associados a três portas: A, B ou C. As classes utilizadas para manipular os motores, na maioria dos casos, são a `lejos.nxt.NXTRegulatedMotor` e a `lejos.nxt.Motor`.

`NXTRegulatedMotor` é a abstração de um motor e `Motor` possui três instâncias de `NXTRegulatedMotor` (A, B ou C), cada uma relacionada a uma porta destinada a um motor e o acesso a cada uma é dado por `Motor.A`, `Motor.B` e `Motor.C`. Algumas das propriedades previstas para os motores são:

- `forward()`: rotaciona o motor para frente, até que seja invocada a parada mediante o uso de `flt()` ou `stop()`;
- `backward()`: Rotaciona o motor para trás, até que seja invocada a parada mediante o uso de `flt()` ou `stop()`;
- `stop()`: para o motor;
- `flt()` ou `flt(boolean imediato)`: para o motor e o desliga, colocando-o em modo flutuante. No segundo caso, o argumento indica se a instrução deve ou não ser executada imediatamente;
- `getAcceleration()`: retorna a aceleração atual do motor;

- `getPosition()`: retorna a posição que o motor está tentando manter;
- `getTachoCount()`: fornece a contagem atual de rotação do motor (valor do tacômetro);
- `getSpeed()`: retorna a velocidade do motor;
- `resetTachoCount()`: zera a contagem de rotação do motor (tacômetro);
- `rotateTo(int angulo)`: movimenta o motor até dado ângulo a partir de 0;
- `setAcceleration(int aceleracao)`: define a aceleração do motor;
- `setSpeed()`: define a velocidade do motor.

A algumas operações que podem ser realizadas com motores, como por exemplo, a rotação para frente, para trás, a verificação da aceleração e o ajuste de velocidade. Outra questão de suma importância para o controle de um robô, de um modo geral, diz respeito a sua operação remota. Dessa forma, é possível conectar o robô por intermédio do bloco inteligente com o computador, via *Bluetooth* e controlá-lo usando uma interface simples[Oliveira, Geraldo 2009].

O controle remoto dos motores (Listagem 1) é realizado usando uma classe `Motor` semelhante ao exemplo mostrado na Listagem 2. Cada motor é do tipo `NXTRemoteMotor`, o qual é semelhante a `NXTRegulatedMotor`, entretanto, este é utilizado no computador para controlar remotamente os motores A, B e C.

```
import lejos.nxt.Motor;
public class TesteMotor { //Operações com motores porta A
      public static void main(String[] args) throws Exception {
            Motor.A.forward();//Rotação para frente
            Motor.A.flt();//Para o motor, coloca em modo flutuante
      }
}
```

**Listagem 1 . Demonstração de código para controle remoto de motores**

### 3.1.2 Planejamento de Trajetória no LeJOS

A implementação utilizou o conceito de *waypoints* no LeJOS (Listagem 3). *Waypoints* são marcações realizadas nos eixos X, Y e dotadas de um fator H, em graus, o qual determina uma orientação em relação a frente do veículo simulado. Esta orientação refere-se a informação pré-determinada sobre as “curvas”, para que o robô não se perca ou se movimente em graus excessivos [LeJOS Team 2008a].

Para desenvoler essa atividade foi instanciado um piloto do tipo `DifferentialPilot` que é a abstração de um carro com duas rodas, e dois motores. Este precisa apenas de informação quanto a dimensão das rodas e quais motores serão utilizados à direita e esquerda. Essa classe já implementa diversas tarefas que um carro faz, como andar, virar, parar entre outras.

Também fez-se o uso da classe `OdometryPoseProvider`, a qual é responsável por designar a posição para o carro. Ela também funciona como um *listener* de

movimento.Ao final, foi usada a classe `Navigator` que controla o robô através de um caminho (*path*), em uma sequencia de *waypoints,* até que alcance o último ponto e pare. Para construir o *path* são adicionados pontos através do método `addWaypoint(new Waypoint(X, Y, H))`. Em que os pontos X e Y são pontos de um plano cartesiano que determina sua localização. Exemplo de como funciona o plano cartesiano na Figura 2.

Como este exemplo é bem simples, os pontos de *heading* foram lançados manualmente, mas o cálculo para encontrar esse valor é dado pela posição do *waypoint* atual até o seguinte, formando uma reta. Esta reta forma um ângulo com a atual posição da frente do robô, até estar na direção desta reta, que é a mesma direção do próximo ponto. Para obter o ângulo utilizam-se a equação do coeficiente angular da reta (Equação 1), que dá o valor do coeficiente da reta em relação a tangente e a equação para o cálculo do arco da tangente (Equação 2).

$$tgz = \frac{y_B - y_A}{x_B - x_A}$$

**Equação 1. Cálculo do coeficiente angular da reta**

$$\arctan z = \sum_{n=0}^{\infty} \frac{2^{2n}(n!)^2}{(2n+1)!} \frac{z^{2n+1}}{(1+z^2)^{n+1}}$$

**Equação 2. Fórmula do arco tangente**

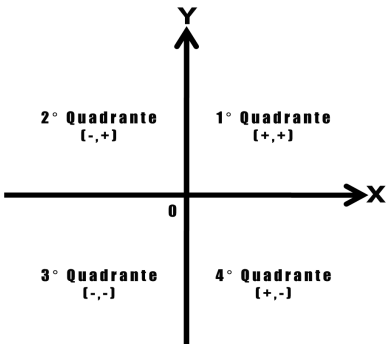


**Figura 2: Plano Cartesiano**

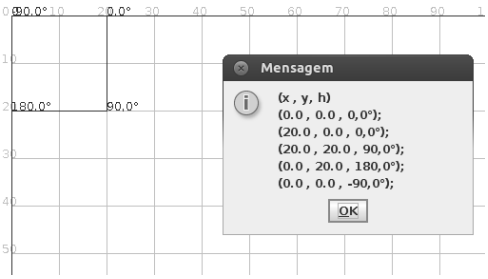


**Figura 3: Demonstrativo de pontos em uma trajetória**

```
import lejos.nxt.Button;
import lejos.nxt.Motor;
import lejos.nxt.LCD;
import lejos.robotics.navigation.DifferentialPilot;
import lejos.robotics.navigation.Navigator;
import lejos.robotics.navigation.Waypoint;
import lejos.robotics.navigation.Pose;
import lejos.robotics.pathfinding.Path;
import lejos.robotics.localization.OdometryPoseProvider;
public class NavigatorTest {
      Navigator nav;
      public static void main(String[] args) throws IOException {
            System.out.println("Pressione um Botão");
            Button.waitForAnyPress();
            DifferentialPilot pilot = new DifferentialPilot(6.5f,
            11.5f, Motor.A, Motor.B);
            OdometryPoseProvider pp = new OdometryPoseProvider(pilot);
            pilot.addMoveListener(pp);
            Navigator nav = new Navigator(pilot);
            //Adiciona waypoints (X, Y, H)
            nav.addWaypoint(new Waypoint(0, 0, 0));
            nav.addWaypoint(new Waypoint(20.0, 0.0,90));
            nav.addWaypoint(new Waypoint(20.0, 20.0,180));
            nav.addWaypoint(new Waypoint(0,20,-90));
            nav.addWaypoint(new Waypoint(0,0,0));
            //Segue os pontos marcados acima.
            nav.followPath();
            Button.waitForAnyPress();
            nav.followPath();
            System.out.println(nav.getPath());
            Button.waitForAnyPress();
      }
}
```

**Listagem 2. Implementação do planejador de trajetória usando *waypoints***

Este será o valor do *heading* para este trajeto entre os dois pontos. Em Java pode-se usar o método `Math.atan2(`$y_B - y_A$`,` $x_B - x_A$`)`, o qual retorna o grau de inclinação da reta. Um exemplo gráfico é apresentado na Figura 3.

## 4. Conclusão

A utilização da ferramenta LeJOS para o desenvolvimento desta pesquisa foi adequada, dada a sua vasta documentação e um fórum bastante ativo. O problema a ser resolvido não é um problema complexo, considerando que a API LeJOS implementa as funções necessárias, facilitando o processo de desenvolvimento, bastando apenas utilizar a fundamentação matemáticas para finalizar a pesquisa.

Como proposta de trabalho futuro, pretende-se criar um planejador de caminho como parte de um estudo para o desenvolvimento de um planejador de movimento completo, o qual terá uma contra parte no PC e uma no NXT. No PC será implementada uma tela semelhante a apresentada na Figura 2 ,e ao final da marcação dos pontos, estes serão enviados para o NXT, o qual deve criar o *path* a ser seguido. O experimento deve usar sensores luminosos e ultra-sônico, pois o caminho traçado poderá ter obstáculos. Dessa forma, o veículo estará apto a prever o risco de colisão e desviar, sem perder sua localização, deverá retornar a trajetória assim que possível.

### Referências

Lew, M.W., Horton, T.B. e Sherriff, M.S. (2010) "Using LEGO MINDSTORMS NXT and LEJOS in an Advanced Software Engineering Course", In: *23rd IEEE Conference on Software Engineering Education and Training (CSEE&T),* pp.121-128.

LeJOS Team (2008a) "LeJOS NXJ API", Disponível em: http://lejos.sourceforge.net/nxt/nxj/api/index.html. Acessado em: 23/01/2013.

LeJOS Team (2008b) "NXJ technology", http://lejos.sourceforge.net/nxj.php. Acessado em: 23/01/2013.

LeJOS Team (2008c) "The LeJOS NXT tutorial", Disponivel em: http://lejos.sourceforge.net/nxt/nxj/tutorial/index.htm. Acessado em: 23/01/2013.

Oliveira, Geraldo, Silva, Ricardo, Lira, Tiago, Reis , Luis Paulo. (2009) "Environment Mapping using the Lego Mindstorms NXT and leJOS NXJ", Disponível em: http://paginas.fe.up.pt/~ei04085/psite/LejosPaper.pdf, Acessado em: 20/01/2013.

Oliveira, Thiago Costa, Gonçalves, Nelson M. A.,Ribeiro, João M. F. S. (2008) "Exploração da plataforma de programação leJOS para robôs Lego MindStorms: Uma Abordagem à Robótica Evolucionária". Disponível em: http://www.di.uminho.pt/ensino/licenciaturas/lei/laboratorios-de-informatica-iv/47109_47119_47131.pdf. Acessado em: 18/01/2013.

Hachour, O. (2008) "Path planning of Autonomous Mobile robo", In: International Journal Of Systems Applications, Engineering & Development, 2ª Edição. http://www.naun.org/multimedia/UPress/saed/saed-45.pdf, Janeiro.