AQUISIÇÃO DE DADOS E CONTROLE ON/OFF DEDICADOS AO TREINAMENTO DE REDE NEURAL ARTIFICIAL

THOMAS LUCAS IRIGOITE BARROCO[1]; JONATAS LEMUEL BISPO ZOTTIS[2]; FERNANDO BALTAR VERONEZ[3]; MARCELO ESPOSITO[4]

[1]Universidade Federal de Pelotas – lucasbarroco@globomail.com
[2]Universidade Federal de Pelotas – lemueltra@gmail.com
[3]Universidade Federal de Pelotas – fernandoveronez_@hotmail.com
[4]Universidade Federal de Pelotas – marcelo.esposito@ufpel.edu.br

1. INTRODUÇÃO

Fazendo uso da linguagem de programação Java (2016), do ambiente integrado de desenvolvimento Eclipse e das ferramentas de desenvolvimento do Lejos (2016), desenvolveu-se um sistema de aquisição de dados com o objetivo principal de realizar o treinamento de uma rede neural artificial (RNA) usando o algoritmo *backpropagation* (HAYKIN, 2001).

Ao todo nove variáveis foram medidas ou calculadas e tiveram seu comportamento dinâmico analisado. Todas as variáveis dizem respeito a um robô seguidor de linha montado com peças do kit Lego Mindstoms Education 9797. Destas variáveis, somente cinco foram efetivamente utilizadas no treinamento da rede neural, três na camada de entrada e duas na camada de saída da RNA.

As nove variáveis de interesse foram: o ângulo de giro do motor A *angA*, o ângulo de giro do motor B *angB*, a luminosidade do local onde o robô está naquele momento *luzA* em %, a potência enviada ao motor A *potA* em %, a potência enviada ao motor B *potB* em %, a velocidade do eixo do motor A *velA* em rad/s, a velocidade do eixo motor B *velB* em rad/s e os tempos de amostragem do ângulo de giro dos motores *tempA* e *tempB*, que foram utilizados para determinar as velocidades individuais dos motores. As variáveis *angA*, *angB* e *luzA* foram facilmente medidas usando funções do Lejos. Os dados referentes às variáveis *tempA* e *tempB* foram obtidos utilizando uma função nativa da linguagem Java (System.currentTimeMillis();), que retorna o valor em milissegundos de um intervalo de tempo de interesse. Os valores de *potA* e *potB* são os valores de potência determinados pela estrutura de controle *ON/OFF*.

A ferramenta Lejos oferece algumas opções para a aquisição de dados de potência dos motores, no entanto nenhuma das opções implementada correspondeu ao fenômeno físico observado durante os testes. Isso justifica o banco de dados criado a partir dos valores de saída do controle *ON-OFF*, antes mesmo de o efeito ser observado na pista. Para a medida das velocidades nos eixos dos motores o mesmo problema foi constatado. Devido a esta discrepância com as funções do Lejos, a alternativa utilizada foi o cálculo das velocidades usando a variação do ângulo de giro de cada um dos motores pelo intervalo de tempo decorrido. Durante a elaboração da sequência de cálculo da velocidade, percebeu-se que o robô ao se deslocar poderia apresentar valores negativos de velocidade como consequência da aquisição do ângulo de giro dos motores (encoder incremental). Quando um motor era acionado o outro recuava alguns milímetros, ocasionando um deslocamento negativo (subtração da posição inicial pela posição ao término do movimento). A solução foi descartar o deslocamento negativo que ocorre quando o robô recua em sua trajetória, ou seja, a velocidade é igual à zero nesta situação.

2. METODOLOGIA

A primeira etapa do trabalho foi desenvolver um algoritmo de controle alternativo as redes neurais artificiais (RNAs), possibilitando a aquisição de dados e o treinamento da RNA. Devido a simplicidade, a estrutura de controle escolhida foi a do tipo *ON/OFF*. Para a elaboração deste algoritmo, foi utilizada a linguagem de programação orientada a objetos, Java (2016). Suas características mais importantes são a portabilidade, a segurança, a sintaxe similar a de outras linguagens de programação de grande utilização como, por exemplo, C e C++ e a facilidade na criação de programas multitarefa, dentre outras propriedades.

Foi utilizado o ambiente de desenvolvimento integrado (IDE) Eclipse. O Eclipse Mars é uma ferramenta de código aberto e apresenta como características a praticidade na implementação do algoritmo, amplo suporte ao desenvolvedor e fácil desenvolvimento e utilização de *plug-ins* e *frameworks*, facilitando a elaboração dos códigos em Java.

A estrutura física do robô seguidor de linha foi elaborada utilizando o kit Lego Mindstoms Education 9797. O kit é formado por um conjunto de peças e instrumentos que reúnem blocos de construção tradicionais Lego, motores, vigas, sensores e engrenagens, além de vários outros materiais que aliados a microprocessadores (Atmel de 32 bits), possibilitam a criação de inúmeros e sofisticados projetos de robótica. A Figura 1, mostra em detalhes o robô montado e utilizado neste trabalho.

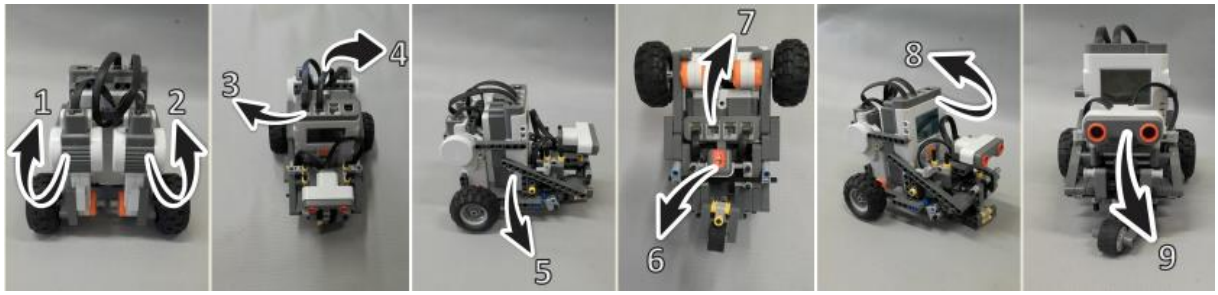

Figura 1 - Estrutura do Robô. (1) motor A, (2) motor B, (3) portas de ligação dos motores, (4) cabos de alimentação e comunicação, (5) blocos de construção Lego, (6) sensor de luminosidade, (7) portas de conexão dos sensores, (8) módulo de processamento e controle NXT Bricks Lego, (9) sensor ultrassônico (não utilizado neste projeto).

Para que fosse possível a execução dos algoritmos desenvolvidos em Java, no microprocessador do Lego NXT, foram usadas as ferramentas Lejos (2016). Dentre as ferramentas estão: um *firmware* substituto para o *firmware* original do Brick Lego NXT, uma biblioteca de classes Java que possibilitam a utilização de funções do Brick em códigos Java, uma ferramenta que realiza a ativação do *firmware* do Lejos, o *upload* e a depuração de programas e um *plug-in* para Eclipse, que permite a utilização das ferramentas já citadas no ambiente de desenvolvimento integrado. Juntamente com outras várias funções disponíveis o Lejos torna-se a ferramenta ideal para a interação Java / Lego Mindstorms.

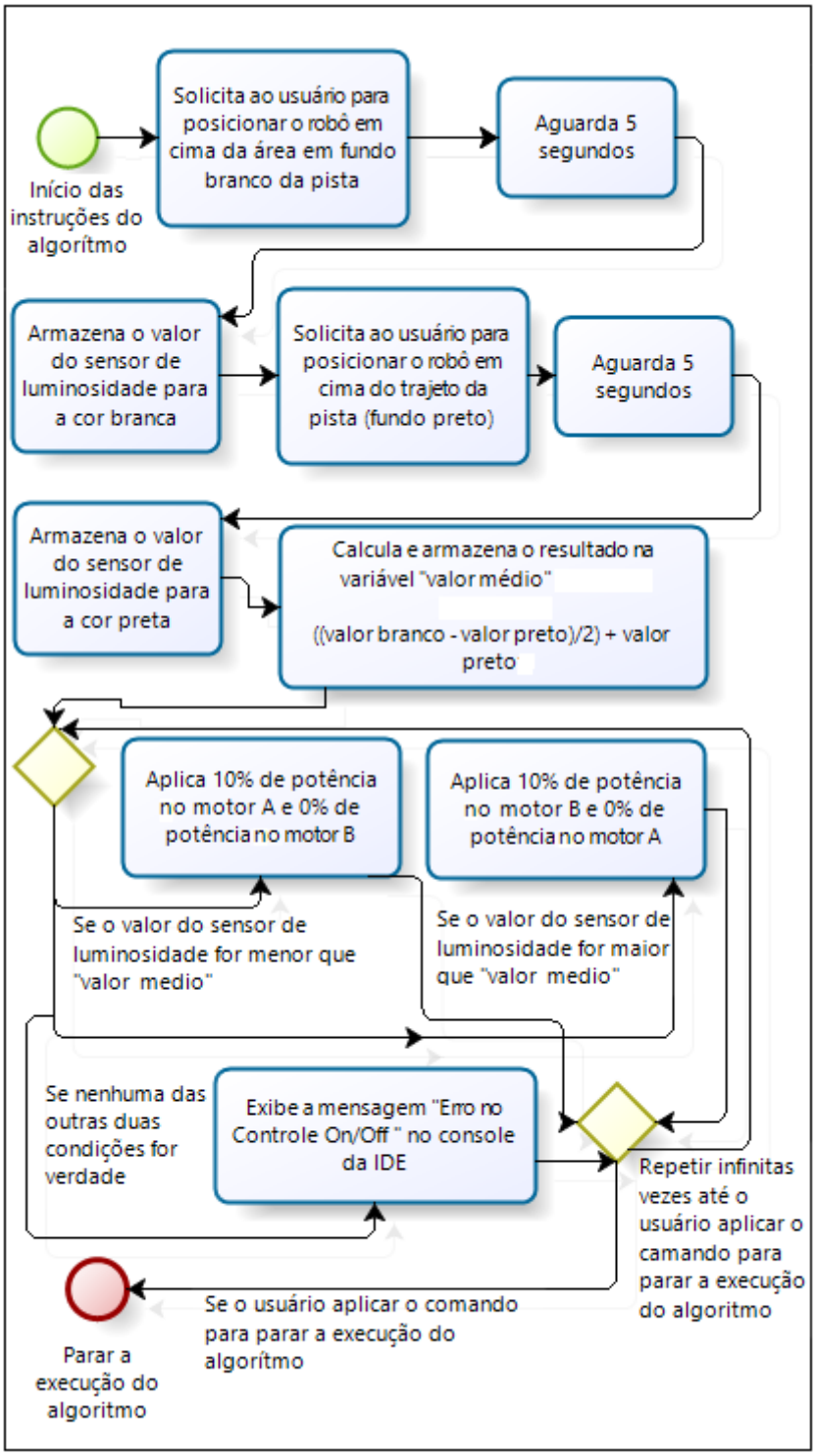

Figura 2 - Estrutura de controle *ON/OFF*.

Baseados em exemplos de algoritmos para o seguidor de linha, os autores criaram um novo projeto no Eclipse e o configuraram para a compilação do algoritmo no computador pessoal do desenvolvedor. Usando comunicação serial conectou-se via cabo o computador com o Brick Lego, isso dispensou a prática de *upload* do algoritmo para a memória do microprocessador do Lego NXT.

Para se valer da biblioteca de classes em Java do Lejos, estas foram inseridas no algoritmo de controle *ON/OFF* utilizando o comando "import lejos.nxt.*;". Após inúmeros testes, erros e correções estabeleceu-se um algoritmo que segue a orientação do diagrama de fluxo mostrado na Figura 2.

3. RESULTADOS E DISCUSSÃO

O robô seguidor de linha foi conectado e devidamente posicionado sobre um banner com o traçado da pista de kart do Sapiens Parque de Florianópolis/Santa Catarina. A pista foi impressa na cor preta e o plano de fundo na cor branca. Com o programa executando no Eclipse, observou-se que o robô seguia perfeitamente a linha em todas as partes do desenho do circuito automobilístico, exceto em 2 pontos críticos, as curvas 6 e 10 mostradas na Figura 3.

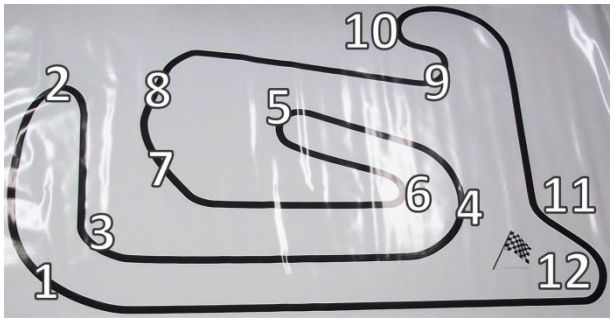

Figura 3 – Pista de testes do robô.

Com o objetivo de corrigir este problema foi inserida uma função para a troca de lado da linha a ser seguida. Percebeu-se que o robô ao executar o controle *ON/OFF* se posicionava sempre à esquerda da linha preta que delimita o caminho a ser seguido. Esse comportamento é comum em robôs seguidores de linha e é costumeiramente chamado de escolha do lado predileto.

As curvas foram os maiores obstáculos a serem superados pelo robô e no caso das curvas 6 e 10 o robô tornava-se instável, perdendo o contato com a linha preta. A função de troca do lado predileto foi implementada utilizando-se um sensor de toque ligado diretamente ao robô. Via código foi possível enviar um comando para a troca de lado, tanto do lado esquerdo para o direito como do lado direito para o esquerdo, toda vez que o sensor fosse pressionado.

4. CONCLUSÕES

Todos os objetivos propostos foram atendidos com êxito. Dois encoders incrementais (sensores para o ângulo de giro e cálculo da velocidade), um fototransistor (sensor de luminosidade) e dois circuitos tipo ponte H (acionamento dos motores) foram usados em conjunto com um sistema microprocessado para a confecção do sistema de aquisição de dados. Respeitando o teorema da amostragem e com a estratégia de controle tipo ON/OFF o robô móvel foi capaz de dar uma volta completa na pista utilizada nos testes, fornecendo dados suficientes para o treinamento de uma rede neural.

5. REFERÊNCIAS BIBLIOGRÁFICAS

HAYKIN, S., Redes Neurais: Princípios e Prática. 2ª ed. Bookman, Porto Alegre, 2001.

JAVA. **Java no ambiente Eclipse**. Acessado em 18 jul. 2016. Online. Disponível em: http://www.sun.com/java/.

LEJOS. **Eclipse com *firmware open source***. Acessado em 18 jul. 2016. Online. Disponível em: http://www.leJOS.org/.

VERONEZ, F. B.; BARROCO, T. L. I.; ZOTTIS, J. L. B.; ESPOSITO, M. Configuração do algoritmo de treinamento de rede neural artificial para o controle de um robô móvel. In: **II Congresso de Ensino de Graduação da Universidade Federal de Pelotas**, 2016, Pelotas. Anais do II CEG 2016.

ZOTTIS, J. L. B.;BARROCO, T. L. I.; VERONEZ, F. B.; ESPOSITO, M. Rede neural artificial aplicada no controle em tempo real de um robô móvel. In: **II Congresso de Ensino de Graduação da Universidade Federal de Pelotas**, 2016, Pelotas. Anais do II CEG 2016.