DESENVOLVIMENTO DE APLICAÇÕES EMBARCADAS PARA LEGO RCX-2.0 UTILIZANDO SOFTWARE LIVRE

Lady Daiana O. MAIA (1), Vandermi João da SILVA (1), Ricardo E. V. S. ROSA(2), José P. QUEIROZ-NETO (2, 3), Vicente F. de LUCENA Jr (1, 2).

(1) UFAM – Universidade Federal do Amazonas, CETELI – Centro de Pesquisa e Desenvolvimento em Tecnologia Eletrônica e da Informação, Av. General Rodrigo Otávio Jordão Ramos, 3000; CEP 69077-000; Manaus – AM; Tel.: (92)3647 4495, Fax.: (92)3647 4494, e-mail: ladydaiana@gmail.com.br, { vicente, vandermi}@ufam.edu.br
(2) UFAM – PPGEE – Faculdade de Tecnologia (FT); Programa de Pós-Graduação em Engenharia Elétrica, Tel.: (92) 3647 4495, Fax.: (92)3647 4494; email: ricardoerikson@gmail.com,
(3) Centro Federal de Educação Tecnológica do Amazonas – CEFET, Av. Sete de Setembro, 1975 – Centro CEP: 69020-120 – Manaus/Amazonas, FAX (92) 3613-3530, e-mail: pinheiro@cefetam.edu.br

RESUMO

A automação e a robótica estão cada vez mais presentes no mundo moderno. Isto é, desde aplicações tecnologicas embarcadas básicas com uso domésticos, hospitalares e industriais, até sistemas de automação de foguetes espaciais desenvolvidos para cumprir missões críticas. Com a queda de preços de microcontroladores e microprocessadores, surgem no mercado diversos *kits* de desenvolvimento, que podem ser incorporados em projetos acadêmicos ou industriais. Este trabalho, apresenta uma implementação embarcada no *Kit Lego Mindstorm* RCX-2.0, adequando técnicas de engenharia de *software* e mostrando as fases de projeto, implementação e testes, necessárias para o bom desenvolvimento de sistemas reais. O experimento foi desenvolvido no laboratório de robótica e automação industrial da Universidade Federal do Amazonas, usando somente softwares livres. Durante o experimento foi construída uma pista em forma de labirinto e funções para direcionar o robô até a saída do percurso. O robô foi construído em forma de um triciclo, equipado com sensor de luminosidade que captura mudanças no meio físico e as envia para se tomar a decisão do caminho a seguir. O objetivo principal do trabalho é mensurar as dificuldades e limitações técnicas de projetos envolvendo sistemas embarcados, e contribuir para melhorar o interesse de alunos pelas disciplinas de Linguagem de Programação e Engenharia de Software.

Palavras-chave: Robótica, Sistemas embarcados, Software livre, Lego RCX 2.0.

1. INTRODUÇÃO

A robótica é definida como ligação inteligente entre a percepção e ação, sendo necessário certo grau de inteligência para realização de uma determinada tarefa e envolve uma interação física entre o sistema e o meio onde a tarefa está sendo realizada (PIO, 2006).

O Kit Lego Mindstorms é um conjunto de robótica para área educacional, permitindo ao usuário criar robôs com peças mecânicas simples e programá-los para realizar tarefas complexas. O Kit é composto por blocos de montar, motores, sensores e o *Robotic Command Explorer* – RCX, um microprocessador que é o cérebro do sistema . O emprego da robótica em ambientes educacionais já provou ser uma ferramenta adequada para o desenvolvimento de atividades que envolvam criar, projetar e planejar, favorecendo assim o processo de ensino-aprendizagem e ainda ampliar a integração entre diferentes áreas de conhecimento (BAGGNAL, 2007).

Estudos realizados usando robôs no ensino de inteligência artificial e tecnologia robótica (NOURBAKHSH, 2000) e ainda a robótica móvel como instrumento de apóio a aprendizagem de computação (PIO, 2006), tem gerado interesse na incorporação de projetos com robôs no ensino de programação e engenharia de software (JAZDI, 2006).

A Engenharia de Software para sistemas embarcados carece de estudos mais aprofundados e de métodos mais robustos, que permitam integrar o desenvolvimento do software, em paralelo ao projeto de hardware, portanto é necessário usar pequenos dispositivos através de experimentos e técnicas de projeto de software/hardware e desenvolver novas técnicas para resolver esses problemas. Dessa forma, pode-se entender o problema separadamente, desenvolver os requisitos de hardware e software, culminando com a transformação dos projetos em um produto único (LUCENA, 2006).

Por outro lado, o software livre, vem se tornando uma ferramenta facilitadora da aprendizagem, por permitir o acesso ao código fonte de diversas aplicações inclusive, sistemas operacionais, compartilhando o conhecimento, com isso é possível usar ferramentas de baixo custo, com o propósito de diminuir os custos de projeto, principalmente quando se trata de projetos educacionais. A principal vantagem quando se usa essa abordagem, é a possibilidade de customização do código fonte, aplicando técnicas de reuso, para aproveitar somente as partes interessantes para o projeto (CESAR,2008).

Neste sentido pretende-se apresentar a aplicação embarcada para navegabilidade no labirinto utilizando o Lego Mindstorms para mensurar as dificuldades e limitações técnicas de projetos envolvendo sistemas embarcados e software livres e ainda contribuir para melhorar o interesse de alunos pelas disciplinas de Linguagem de Programação e Engenharia de Software.

Na seção 2, são apresentadas as ferramentas livres e a metodologia desenvolvimento utilizados para a implementação do robô. A seção 3 apresenta as fases do projeto, bem como uma descrição dos passos seguidos durante a implementação do projeto do robô. Os resultados e testes são mostrados na seção 4 e por fim, a seção 5 traz as considerações finais.

2. METODOLOGIA E FERRAMENTAS LIVRES

O desenvolvimento do projeto foi realizado no laboratório de robótica e automação industrial da Universidade Federal do Amazonas e optou-se pela utilização de ferramentas livres, devido às vantagens do mesmo em relação aos custos e benefícios.

O projeto utilizou as seguintes etapas: (i) escolha das ferramentas livres e o ambiente de programação; (ii) o estudo do modelo de Engenharia de software e a criação da arquitetura do sistema; (iii) a construção do robô Lego; (iv) a implementação da aplicação proposta, passando por todas as fases do projeto de software; e (v) testes realizados no robô relacionados a aplicação embarcada.

O uso de ferramentas livres no projeto foi devido o *Software* livre está cada vez mais consolidado e robusto, sendo possível a construção de um software, usando partes de outros softwares, de forma a aproveitar e otimizar o uso do código, sem necessidade de construir tudo do zero (CESAR, 2008) . A vantagem do uso de *Software* Livre é a possibilidade que se tem para modificá-lo e distribuí-lo compartilhando o conhecimento

adquirido. Nesse tópico, serão apresentados os *Softwares* Livres utilizados no projeto e uma breve descrição de cada um deles.

2.1. Sistema Operacional Linux UBUNTU

A palavra Ubuntu é de origem Africana e traduzida para o português significa "humanidade para com os outros" (UBUNTU, 2008). O sistema operacional Ubuntu desenvolvido pela comunidade Linux, utiliza o núcleo do Linux que foi criado por Linus Torvalds e que hoje tem uma ramificação muito grande no mundo dos Sistemas Operacionais.

O Ubuntu é licenciado por *General License Public* - GPL, para qualquer pessoa utilizar, estudar, modificar e distribuir de acordo com os termos da licença, e a sua escolha se deu pelas seguintes vantagens: robustez e confiabilidade, por ter código aberto, compatibilidade, segurança, rapidez e estabilidade. (UBUNTU-BR, 2008)

2.2. BrickOs

O BrickOS licenciado por GPL é uma alternativa *open source* para sistema operacional do kit Lego *Mindstorms*, com a possibilidade de adaptação do código fonte e customização. Foi desenvolvido a partir do padrão *Portable Operating System Interface* – POSIX, conjunto de normas que tem por objetivo garantir a portabilidade do código-fonte de um programa apartir de um sistema operacional (POSIX, 2008) , como solução para as limitações encontradas no software próprio do kit denominado *Not Quit C* - NQC , é multitarefa e permite um maior controle sobre a CPU do RCX (BAGNALL, 2007).

De acordo com Tavares (2004) esse sistema operacional possui funcionalidades avançadas de tempo real, pois implementa o gerenciamento de memória através de paginação, controle de processos, comunicação inter-processos, semáforos, escalonador de processos com prioridade, suporte ao fluxo do controle e controle total sobre o display, dentre outras funcionalidades. Além disso, o BrickOS permite amplo uso da memória, sendo que o limite para o número de variáveis, de funções, de processos e de sub-rotinas é o próprio tamanho da memória . Destaca-se devido às vantagens, de acordo com a comparação na Tabela 1.

Tabela 1 - Comparação entre os Sistemas

Sistemas	BrickOS	NQC
Controle de Hardware	Completo	Via Biblioteca
Linguagem	C / C++	C
Velocidade de Execução	Máxima	Média
Sistema Operacional RCX	Próprio	Lego
Sistema Operacional PC	Linux	Windows/Linux

Para a programação do RCX, o BrickOS utiliza os compiladores gcc e g++, esses compiladores, na realidade, contém bibliotecas padrão e suportam linguagens C/C++ , que suportam um conjunto restrito de funções, ou seja, é possível programar em C e C++ com algumas restrições de sintaxe e de hardware do RCX como a memória de 32 K (BRICKOS, 2008). A escolha desse sistema para ser utilizado nesse projeto foi motivada devido às características citadas, pois é possível considerar o BrickOS como um dos melhores sistemas alternativos para o RCX (SYSTEM BRICKOS, 2008).

2.3. IDE Eclipse com C/C++ Development Tooling – CDT

Eclipse é uma IDE bastante utilizada atualmente, pois permite integrar várias ferramentas em um único ambiente de desenvolvimento. É uma ferramenta bastante flexível e foi desenvolvida em Java pela comunidade Eclipse, utiliza *Standard Widget Toolkit* - SWT como biblioteca gráfica e possui o seu desenvolvimento baseado em *plug-ins.*

Embora o Eclipse seja principalmente um ambiente de desenvolvimento Java, sua arquitetura garante apoio a outras linguagens de programação como Java, C, C++, Phyton, PHP, Pearl, Cobol (LOSANO, 2003). A escolha dessa plataforma se deu por algumas vantagens, dentre elas destacam-se:

- É Software Livre;
- Auxilia no desenvolvimento e construção de aplicações;
- Depuração integrada em tempo real, ajudando a encontrar os erros de compilação;
- Ambiente amigável;
- Por ser uma plataforma portável e extensível, o qual permite adicionar outras ferramentas.

O CDT é um projeto open source licenciado sob a GPL, executado exclusivamente na linguagem de programação Java como um conjunto de *plug-ins* para o Eclipse *Platform* SDK. Com este *plugin* pode-se apoiar o desenvolvimento de aplicações em linguagem C / C + +, juntamente com a edição avançada e apoio a depuração (ECLIPSE,2008).

3. FASES DO PROJETO

O projeto foi desenvolvido em duas fases distintas. Primeiro construiu-se a parte mecânica do robô, em seguida foram feitos testes de avaliação para escolha do modelo, por fim foram feitas as análises de requisitos, arquitetura de desenvolvimento e testes do software. O robô foi construído a partir das peças oferecidas no *Kit Lego Mindstorn* RCX 2.0, e consistem basicamente de dois motores, um sensor de luminosidade, uma unidade de processamento e três rodas. O projeto de software consistiu em construir uma aplicação capaz de automatizar o dispositivo para percorrer um labirinto até encontrar a saída.

3.1. Construção do robô usando o kit Lego RCX 2.0

Primeiramente foi pensado em montar um robô tipo tanque de guerra, contudo, devido a grande quantidade de peças e a dirigibilidade do robô, o protótipo foi descartado. Através de pesquisas bibliográficas, foi possível identificar um protótipo em forma de triciclo, que posteriormente foi adaptado para um trator tipo arado, que se comportou muito bem no quesito de estabilidade e controle, sendo este o utilizado no projeto. A Figura 1 apresenta o robô construído pela equipe e utilizado no experimento.

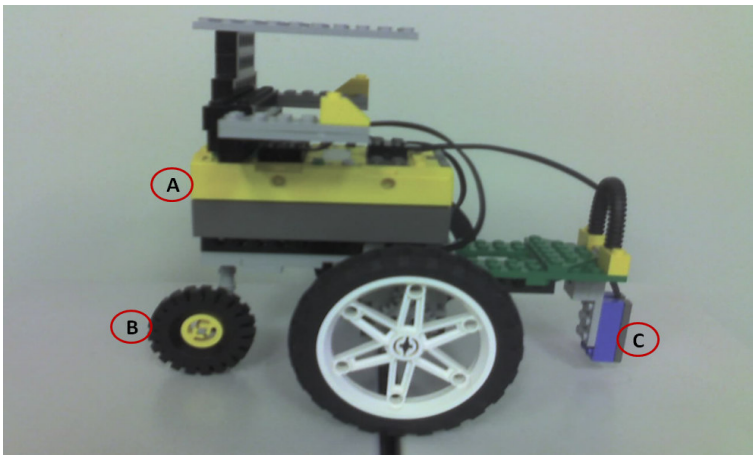

Figura 1- Robô Trator, montado com kit Lego Mindstorms

Observando a Figura 1 pode-se notar que na parte frontal do robô, foi adicionado o sensor de luminosidade (C), de forma a captar a intensidade das cores o mais próximo possível. Isto foi uma decisão de projeto, tendo em vista que a precisão do sensor não é confiável se a leitura for feita de uma distância maior. A roda traseira possui movimentos de tração para a esquerda e direita (B), com o intuito de facilitar a dirigibilidade do equipamento, permitindo que faça as curvas, previstas para sua navegabilidade. O cérebro do robô (A), responsável por processar as informações, interliga os motores e o sensor, completando o dispositivo.

3.2. Análise de Requisitos

Após a construção da parte mecânica, partiu-se para a análise de requisitos do *software*, baseado em premissas pré-determinadas para o experimento. O projeto foi desenvolvido com base nos requisitos principais que consistia em um sistema de controle que permitiu a navegação em um labirinto composto por um circuito desenhado em papel, usando padrões de escalas de cinza. O labirinto continha uma entrada e uma saída, identificadas nas escalas de cinza, possuía também entroncamentos e bifurcações para dificultar o percurso pelo robô.

Após a avaliação do problema, buscou-se desenvolver o documento de requisitos do software, iniciando pelos requisitos funcionais, agrupados por classes como se vê nos principais requisitos descritos a seguir.

a) Os requisitos de confiabilidade foram identificados através do mapeamento do comportamento esperado do sensor de luminosidade, e foram definidos de acordo com as funcionalidades de navegação. O sensor deverá ler em escala de cinza e de acordo com a cor lida, o sistema tomará a decisão de seguir em frente, virar direita, esquerda ou parar quando identificasse fim de labirinto.

b) Os requisitos de segurança foram identificados, a partir da avaliação do protótipo e somente foram necessários para proteger o equipamento quanto a possíveis colisões acidentais, ligando os motores somente após 5 segundos do *start* do software e parando-os após o cumprimento da tarefa.

c) A configuração *Infrared Data Association* - IRDA através de um *script shell* usando o console do Linux , foi o único ponto mapeado como requisito Interface Homem Máquina -IHM do Sistema.

3.3. Arquitetura do Sistema

O modelo de arquitetura adotado para facilitar a interoperabilidade dos módulos e a possibilidade de possuir diferentes interfaces de acesso ao sistema é o modelo em camadas, a sua escolha foi realizada para facilitar o uso do Sistema Operacional BrickOS e suas bibliotecas em C e C++.

Dividido em quatro camadas, o modelo é composto pela camada de Sistema Operacional Linux, seguida da camada do Sistema Operacional BrickOs, bibliotecas em linguagem de programação C, finalizando com a camada de aplicação, onde é desenvolvido o software de controle do dispositivo. A vantagem deste tipo de modelo é a possibilidade de realizar diferentes implementações para cada uma das camadas, tornando-as interoperáveis. Cada camada só utiliza serviços da camada imediatamente abaixo dela e fornece serviços para a camada imediatamente acima dela.

Figura 2. Modelo de camadas utilizado na arquitetura do sistema.

A Figura 2 apresenta a arquitetura do projeto, com o modelo de camadas. A camada de aplicação do software de controle foi implementada, usando os recursos das demais camadas do sistema operacional BrickOs, dentre as possibilidades de interfaces estão o uso de comandos de display, sensores e motores do Kit Lego. A segunda camada de bibliotecas da linguagem C e C++ provêem funções e classes para acesso direto ao *hardware*, permitindo o uso através das chamadas de funções e métodos do sistema operacional.

A camada de sistema BrickOS , concentra todos os módulos de operação do Kit Lego inclusive o *firmware,* que é carregado junto com a aplicação, prover o controle da CPU H8300-Hitachi, que é o cérebro do robô Lego RCX. Por ser de código livre, mudanças podem ser aplicadas desde que o autor seja referenciado, de acordo com os termos da licença *Mozzila.*

A camada do sistema operacional Linux foi utilizada somente em tempo de projeto e pode ser substituída por qualquer distribuição, inclusive o CygWin, uma distribuição Linux que funciona em sistemas operacionais proprietários. Nesta camada o *Gnu C Compile* (GCC) e o *Gnu* ++ (G++), foram instalados como ambiente de desenvolvimento, integrado ao Eclipse através do *plugin* CDT. Sendo responsável pela comunicação da torre de transmissão USB e o RCX que usa arquitetura de comunicação da Lego.

3.4. Diagramas funcionais

O *software* de controle do robô foi desenvolvido usando a modelagem estruturada e implementado usando linguagem de programação C. Através dos diagramas desenvolvidos é possível mostrar as funcionalidades e módulos do sistema antes de sua codificação, permitindo modificar os diagramas sem impactar na implementação. O modelo em Cascata que é um modelo utilizado na Engenharia de Software, foi escolhido para o projeto, contudo, durante a fase de implementação obteve-se problemas devido ao ajuste de *hardware* (motores, sensores e engrenagens), sendo necessário a realização de experimentos empíricos para testes dos dispositivos, esses testes realizados não estavam previstos nesse modelo de engenharia.A modelagem estruturada utilizada no desenvolvimento permite mostrar através de modelos quais as funcionalidades do sistema e como essas funcionalidades podem ser transformadas em módulos do sistema

3.4.1. Diagrama de Contexto - DC

O DC pode ser observado na Figura 3, vê-se que o sistema interage com uma entidade externa (Sensor), que captura um padrão de cor e envia para um processo principal (Proc_Mov), que por sua vez o processa e envia os comandos para direcionar o robô. As entidades representadas (Mv_Dir, Mv_Esq, Mv_Frente , Mv_Re e Mv_180) , podem ser consideradas como módulos externos ao sistema, recebendo somente os comandos através de parâmetros. Esses parâmetros dependem exclusivamente da leitura do sensor, por isso essa é a parte crítica do sistema. Uma leitura errada significa um dado errado e como conseqüência, um movimento errado por isso, programou-se um conjunto de padrões de cinza específicos para cada movimento, tratados em funções separadas.

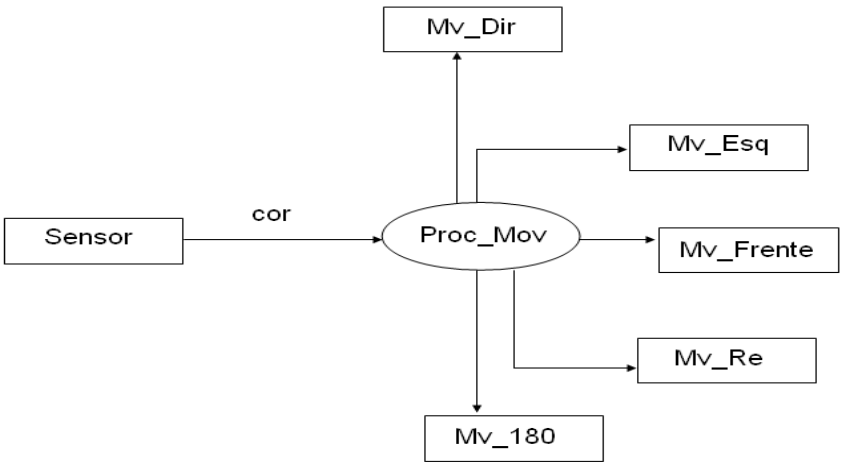

Figura 3. Diagrama de Contexto da Aplicação para o Robô Lego.

3.4.2. Diagrama de Fluxo de Dados - DFD

A partir do diagrama de contexto, foi definido o DFD com a abstração das principais possíveis funções do sistema de controle do robô, a Figura 4 apresenta o diagrama com a relação entre a entidade externa Sensor, que passa um parâmetro de cor para o processo (1. Proc_Mov) , que após determinar a velocidade e a direção , informa ao processo (2. Movimenta) , o qual executa o movimento previsto, (esquerda, direita, frente, ré ou giro de 180º), passa ao processo (3. Mensagem) a mensagem de texto que deverá ser vista no display do Lego Robô.

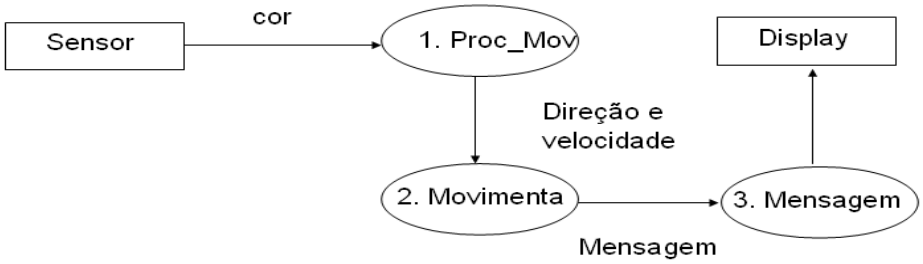

Figura 4. Diagrama de Fluxo de Dados da Aplicação para o Robô Lego.

3.4.3. Diagrama de Estados - DE

A Figura 5 mostra o Diagrama de Estados do sistema onde pode se notar que o estado lendo_cor, pode usar ou descartar a cor lida, possibilitando assim que a lógica do sistema possa ser programada para rastrear a cor recebida do sensor de acordo com o padrão de cores que o sistema se guiará para percorrer o labirinto.

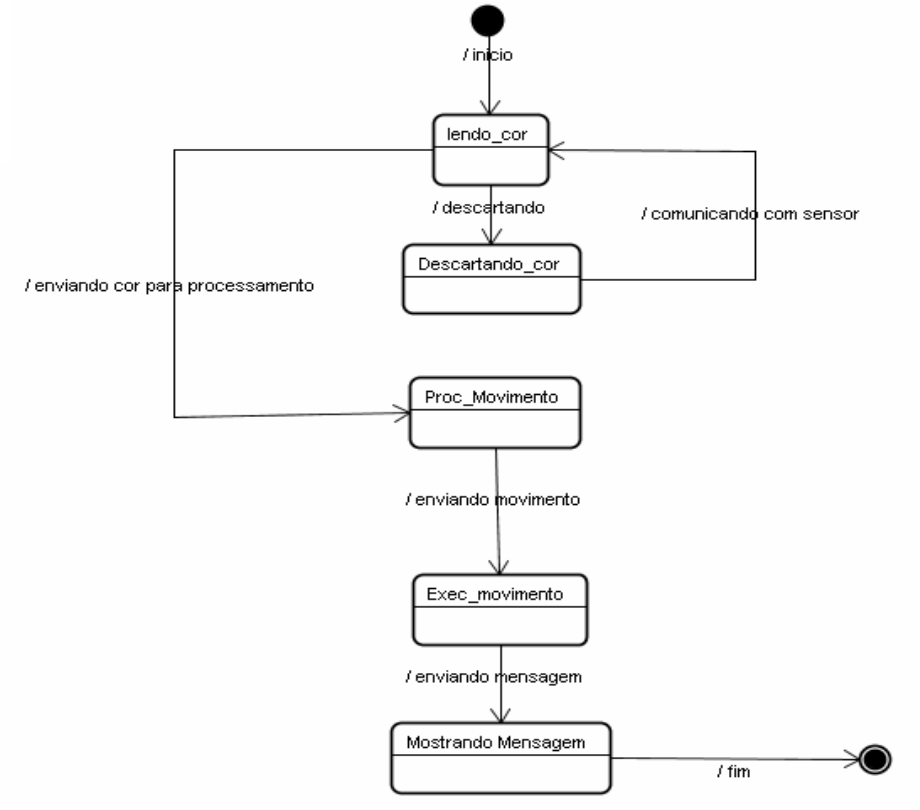

Figura 5. Diagrama de Estados da Aplicação para o Robô Lego.

3.5. Implementação do projeto

Para a implementação do projeto foram criados módulos de acordo com as funções, os quais são necessários na implementação dos elementos descritos nesse projeto, os módulos serão detalhados a seguir:

a) Módulo Ler_cor: Este módulo verifica a cor lida e contém todo o código responsável pela interpretação da cor recebida através do sensor de luz, possui funções para tratar diferentes tons de escala de cinza para tomada de decisão e navegação do labirinto. Tendo a responsabilidade em fornecer suporte para o módulo Movimenta que permite a mobilidade do robô.

b) Módulo Proc_Mov: Todo o código responsável pelo processamento do movimento do robô foi implementado neste módulo, nele é implementado a lógica dos movimentos previstos para que o robô consiga seguir o labirinto e encontrar a saída. O objetivo é permitir a navegação do robô no labirinto através da identificação das cores, além disso, verificar o status dos motores e trocar suas velocidades e suas direções quando cores definidas como obstáculos forem encontradas.

c) Movimenta: O módulo Movimenta, foi construído para executar os movimentos processado em Proc_Mov e gerar uma mensagem para módulo de Mensagem, a finalidade do módulo é dar mobilidade e direção para o robô.

d) Mensagem: Além dos módulos apresentados, construiu-se também um módulo de mensagem, com o objetivo de informar no display do robô, qual a direção a ser tomada, de acordo com a leitura e processamento do sensor.

4. TESTES E RESULTADOS

Os testes realizados objetivam garantir dois itens principais, a qualidade do software de controle do robô Lego RCX e as funcionalidades do robô para verificar se o objetivo foi atingido. Os testes sobre as funcionalidades da aplicação foram realizados de acordo com as especificações de requisitos do projeto relacionados à comunicação entre o computador e o RCX e as funcionalidades da aplicação, que consistem nos caso de teste de comunicação e caso de teste de funcionalidade, cada caso de teste consistiu em 15 tentativas.

Tabela 2 – Planilha de Resultados dos Testes

DESCRIÇÃO DE TESTE	**DADOS**
INFORMAÇÕES DO ROBÔ	RCX 2.0
LINGUAGEM UTILIZADA	C
Comunicação entre o computador e o RCX	**Resultado +**
Reconhecimento da torre de transmissão de dados USB	**100%**
Transferência do firmware para o RCX	**100%**
Transferência da aplicação lego_robo para o módulo RCX	**100%**
Funcionalidade da aplicação	**Resultado +**
Função start()	**100%**
Função turn_left()	**90%**
Função turn_right()	**90%**
Função reverse()	**65%**
Função end_of_maze()	**80%**

Os resultados dos testes realizados apresentados na Tabela 2 referente ao caso de teste de comunicação, que possui o objetivo de verificar se é possível transmitir o firmware e aplicação para o módulo Lego RCX através da comunicação usando o infravermelho entre a torre de transmissão USB e o módulo infravermelho do RCX, entre o sistema Operacional Linux e o sistema BrickOS. O outro caso de teste é o das funcionalidades da implementação que consiste em verificar se as funções de deslocamento pra frente, para esquerda, pra direita, função ré e função girar 180º obtiveram o resultado esperado.

De acordo com os resultados descritos na planilha de testes do robô, observa-se que os testes realizados com o sensor de luminosidade, não atingiram os resultados esperados, tendo em vista que, o sensor tem uma baixa precisão na identificação das cores, sofrendo interferências de acordo com a iluminação do ambiente. Foi utilizado um padrão de cores em escala de cinza, variando a altura do sensor em relação à cor detectada no labirinto, verificou-se que a partir de uma altura maior que três centímetros em relação ao solo, perdiam-se a faixa de cor necessária para identificar os obstáculos do labirinto. No entanto, para resolver esse problema, utilizou-se uma peça da cor preta como máscara, diminuindo a intensidade da luz direta no sensor e manteve-se a altura do sensor em no máximo três centímetros do solo, com isso, obteve-se uma melhoria na funcionalidade reverse, aumentando o seu percentual para próximo de 100%.

5. CONSIDERAÇÕES FINAIS

O trabalho apresentou uma aplicação embarcada para o Kit Lego Mindstorms, utilizando tecnologias livres e técnicas de Engenharia de Software. A principal limitação encontrada para o desenvolvimento da aplicação é a falta de um modelo específico consolidado para tratar os processos tanto de software quanto de hardware, que é necessário para o desenvolvimento de softwares embarcados.

No entanto, utilizamos um modelo conceituado da Engenharia de Software para o desenvolvimento dessa aplicação embarcada, o Modelo Cascata, e observou-se que não é apropriado para este tipo de aplicação, tendo em vista que há necessidade de um fluxo de processo que cubra os testes de hardware que impactam diretamente em mudanças no processo de desenvolvimento de software embarcados. Outras limitações encontradas foram às relacionadas ao sensor, pois o sensor de luminosidade possui pouca precisão na identificação do padrão de cores e o seu tempo de leitura é pequeno interferindo na identificação do cruzamento do labirinto.

A aplicação desenvolvida demonstra o uso dessas tecnologias e valida à importância dos métodos de Engenharia Software. Com isso, percebe-se que o estudo da robótica pode ser empregado em várias áreas da educação, auxiliando no processo de ensino-aprendizagem, como nas disciplinas de programação e engenharia de software ou outras. Podendo assim, proporcionar uma motivação maior e até mesmo reduzir a evasão dos alunos e principalmente a aplicação dos conceitos vistos em sala de aula. Os resultados do experimento, estão disponíveis, no grupo de pesquisa dos autores, que pode ser contatado através dos e-mails contidos no inicio deste artigo.

REFERÊNCIAS

PIO, J.L.; Castro, T. H.; Júnior, A. N. C. **A Robótica Móvel como Instrumento de Apoio à Aprendizagem Computacional**. 2006. XVII Simpósio Brasileiro de Informática na Educação. **Anais.** Pag. 197 a 206.

TAVARES, D. M.; ANTUNES, V. A.; GONÇALVES, L. M. **Em Evidência o Potencial e Limitações dos Compiladores NQC e BrickOS e seus Respectivos Sistemas Operacionais**. Revista de Informática Teórica e Aplicada, pag. 79-98, V.10, 2004.

LOSANO, F. **Eclipse no Mundo Open Source**. Disponível em: <http://web.teccomm.les.inf.puc-rio.br:8080/eclipse/files/EclipseDay2/palestras/eclipseoss.pdf> Acessado em: 31. Jul. 2008

NOURBAKHSH, I., **Robots and Education in the classroom and in the museum: On the study of robots, and robots for study.** Proceedings of the Workshop for Personal Robotics for Education, IEEE International Conference on Robots and Automation, IEEE Press, 2000.

CESAR, **CRUISE:Componet Reuse in Software Engineering**. Disponível em: <http://www.cesar.org.br/> Acessado em: 01. Jul. 2008.

BAGNALL, B., Maximum Lego NXT. **Building Robots with JavaTM Brains**. Ed. Press Variant. 2007

ECLIPSE, **Comunidade Eclipse**, Disponível em: <www.eclipse.org> . Acessado em < 15. Jul. 2008>.

BRICKOS, **BrickOS Home Page**, Disponível em <www.brickos.sourceforge.net>. Acessado em < 20. Jul.2008>

BARRIUSO, J. M., Castellano, E., Cebrián, J., Garcia, J., Haro, M. J., Herreros, M., Pérez, I., Valiente, J. L.,& Vidosa, I**.. Experiencias com robots en aulas de secundaria**. Portugal, 2004.

UBUNTU, **Learn moure about Ubuntu**. Disponível em:< http://www.ubuntu.com/products/whatisubuntu> Acessado em: 01. Ago. 2008.

UBUNTU-BR, **Comunidade Ubuntu**. Disponível em: < http://pt.wikipedia.org/wiki/Ubuntu_Linux> Acessado em: 15. Jul. 2008.

SYSTEM BRICKOS, **BrickOs Operating System**. Disponível em: <http://en.wikipedia.org/wiki/BrickOS_operating_system>. Acessado em: 01. Ago. 2008.

POSIX, ***Portable Operating System Interface***. Disponível em: <http://pt.wikipedia.org/wiki/POSIX>. Acessado em: 17. Jul. 2008.

LUCENA, Jr, V. F. ; BRITO, Alysson ; GÖHNER, P. ; JAZDI, Nasser . **A Germany-Brazil Experience Report on Teaching Software Engineering for electrical Engineering Undergraduate Students**. In: 19th Conference on Software Engineering Educationand Training (CSEE&T), 2006, Honolulu. Software Engineering Education and Training CSEET 2006, 2006. v. 1. p. 69-76.

JAZDI, Nasser ; Lucena Jr, V. F. ; GÖHNER, P. . **Unibral, an Education and Research Cooperation between Brazil and Germany**. In: 36th Frontiers In Education Conference, 2006, San Diego. Proceeedings of the 36th Frontiers In Education Conference, 2006. p. 20-25.

LUCENA Jr, V. F. ; BRITO, Alysson ; JAZDI, Nasser . **Uma Metodologia Germano-Brasileira para o Ensino de Engenharia de Software para Alunos de Engenharia Elétrica**. In: COBENGE - Congresso Brasileiro do Ensino de Engenharia, 2006, Passo Fundo -RS. Anais do Evento, 2006.

AGRADECIMENTOS

Agradecemos a Coordenação de Aperfeiçoamento de pessoal de Nível Superior – CAPES e a Fundação de Amparo a Pesquisa ao Estado do Amazonas- FAPEAM, pelo financiamento de bolsas de estudos com a finalidade de formação de recursos humanos.