UTILIZAÇÃO DE SENSORES PARA A LOCOMOÇÃO DO ROBÔ CURUMIN EM UM LABIRINTO MUTÁVEL

Karen G. Nunes[1], Luiza M. Oliveira[1], Giovanna H. Tonello[1], Silvia de Castro Bertagnolli[2], Patrícia Nogueira Hübler[2]

[1]INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DO RIO GRANDE DO SUL – CÂMPUS CANOAS
Rua Dra. Maria Zélia Carneiro de Figueiredo, 870
CEP 94412-240 – Canoas – RS

[2] INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DO RIO GRANDE DO SUL – CÂMPUS CANOAS
Rua Dra. Maria Zélia Carneiro de Figueiredo, 870
CEP 94412-240 – Canoas – RS

Resumo O uso da robótica como ferramenta educacional tem sido discutido por diversos autores na literatura. Desse modo, vários tipos de *kits* de robótica foram desenvolvidos para uso por professores e alunos, de forma a facilitar a aprendizagem de conteúdos de disciplinas de cursos de Nível Fundamental, Médio e, até mesmo, de Nível Superior. Este artigo apresenta o *kit* robótico Curumin da XBot. Para exemplificar o seu uso e as potencialidades que ele possui na aprendizagem foi elaborado um estudo de caso de um labirinto mutável, que utilizando somente os sensores do robô, analisa as possibilidades de caminho que o robô pode percorrer. O artigo apresenta parte da documentação dos testes realizados sobre o funcionamento do Curumim usando os seus principais comandos, analisando e corrigindo as imprecisões do mesmo, além de introduzir a programação de um labirinto mutável.

Palavras Chaves: robótica educacional, kit robótico XBot, labirintos mutáveis, programação C/C++.

Abstract: The use of robotics as an educational tool has been debated by many authors in the literature. This way, many kinds of robotics kits were designed to be used by teachers and students to make the learning of content subjects easier in every level of education: Elementary School, Middle School, High School, and even colleges. This article introduces a robotic kit, Curumim from Xbot. To exemplify its use and the potential it got on learning, was elaborated a case study of a mutable labyrinth that uses only the robot's sensors, analyzes the possibilities of route that the robot can go. The article presents some of the documentation from the tests conducted on the operation of Curumim using the key commands, analyzing and correcting the inaccuracies of the same, to support the programming of a mutable labyrinth.

Keywords: educational robotics, robot kit XBot, mutable labyrinth, C/C++ programming.

1 INTRODUÇÃO

A programação de robôs exige a formalização das ideias, a compreensão de outros idiomas, a compreensão de espaço e tempo, entre outros aspectos que contemplam a matemática, a física, o inglês, a computação e a eletrônica, entre outras áreas.

Este trabalho origina-se do projeto de pesquisa "A robótica como ferramenta para qualificar o ensino no IFRS câmpus Canoas" que começou a ser desenvolvido no ano de 2013. O foco do projeto é qualificar o processo de ensino e aprendizagem dos cursos de informática do câmpus Canoas. Para tanto, ele utiliza-se de *kits* de robótica doados ao câmpus através de um convênio com a PETROBRAS, o BNDES e o Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Sul (IFRS) câmpus Canoas.

O câmpus recebeu a doação de dois tipos de *kits* robóticos:

1) KRE, desenvolvido sob encomenda pela empresa Instor, esse *kit* possibilita montar/desmontar e acoplar componentes em sua estrutura, bem como a programação do robô através da linguagem de programação Java e Android. Atualmente, existem 2 unidades deste *kit* na Instituição;

2) Curumin, desenvolvido pela empresa XBot tem como foco apoiar o desenvolvimento do raciocínio lógico e a programação de robôs. A programação ocorre através da linguagem de C/C++. Hoje em dia, existem 07 (sete) unidades deste *kit* para uso pelos alunos da Instituição.

Como o *kit* robótico Curumin é mais simples, em arquitetura de hardware, e como utiliza a linguagem de programação C/C++, a qual já foi estudada pela equipe do projeto, optou-se por iniciar os estudos de robótica utilizando este *kit*. Além disso, a instituição possui um número maior destes *kits* e a equipe do projeto é composta por vários alunos bolsistas do PIBIC-EM/CNPq e por alunos voluntários.

Cabe destacar que, todos os alunos que fazem parte do projeto cursam o 2º e o 3º ano do curso técnico de informática

integrado ao ensino médio. Já as bolsistas que fazem parte da equipe que propôs o presente trabalho são alunas do 3º ano e já estão estudando o modelo de programação orientado a objetos, conhecimento essencial para a programação dos robôs.

Os primeiros contatos com os *kits* robóticos começaram em abril de 2013 e já nas primeiras atividades com o uso da programação em blocos, a qual será descrita nas próximas seções, o robô funcionava. Quando a equipe iniciou as atividades de a programar os robôs sem utilizar os blocos, ela encontrou alguns problemas, pois a linguagem de programação C++ não era conhecida, bem como algumas bibliotecas utilizadas pelo Curumin e pela linguagem C++. Além disso, vários testes foram realizados e os *kits* apresentaram alguns problemas de hardware no uso de sensores e nas rodas que "travavam" durante os deslocamentos. Resolvidos esses problemas a equipe ficou desmotivada com o uso dos *kits*, pois achou que já havia explorado todos os recursos disponibilizados pelo robô.

Então, pensou-se em trabalhar com a ideia de desafios de programação que envolvessem conceitos de física e matemática, através do uso dos robôs; e foi assim que surgiu o estudo de caso que será abordado por este artigo – o labirinto mutável. O objetivo principal este estudo foi estabelecer um algoritmo que visa trazer maior autonomia ao *kit* Curumim, dando-lhe a possibilidade de se locomover em um labirinto mutável através do uso de seus sensores.

Destaca-se que, para realizar esse estudo de caso a equipe vivenciou situações, que exigiam a constante resolução de problemas, o pensamento investigativo e o raciocínio lógico, através da montagem de um labirinto usando papelão reciclado e conceitos de matemática e física. Nesse sentido, os envolvidos com o estudo de caso tiveram como grande desafio o trabalho em equipe, pois não estavam acostumados com este tipo de prática.

A equipe teve que realizar uma análise sobre o modo que o Curumim se comporta ao receber comandos e realizou vários testes. Para uma melhor compreensão de todo o processo usado para o desenvolvimento do estudo de caso todos os resultados dos testes foram tabulados e foram realizados alguns comentários sobre os recursos dos *kits* robóticos Curumin e como eles interferem na sua locomoçãodentro do labirinto.

As próximas seções do artigo apresentam: (i) na seção 2 uma descrição do *kit* robótico Curumin e quais foram os passos seguidos para utilizá-lo; (ii) a seção 3 descreve alguns aspectos relacionados ao uso do ambiente Visual C++ para a programação dos robôs; (iii) na seção 4 é aprensentado o estudo de caso e são detalhados alguns processos utilizados para fazer os robôs reconhecerem o labirinto; (iv) na seção 5 são descritos alguns resultados obtidos com o uso do Curumin, e, finalmente, (v) a seção 6 descreve o labirinto mutável e; (vi) a seção 7 apresenta as conclusões e os trabalhos futuros previstos para o projeto.

2 ROBÔ CURUMIM

O Curumim é um *kit* robótico desenvolvido pela empresa XBot, que consta de uma plataforma robótica e um ambiente para a programação de robôs com o objetivo de promover o desenvolvimento educacional e aprendizado de conceitos nas áreas de lógica, raciocínio lógico, programação estruturada e orientada a objetos para alunos dos mais diversos níveis de ensino (fundamental, médio e até mesmo superior) [XBOTc, 2013].

Além disso, ele pode ser usado como uma plataforma de pesquisa e desenvolvimento multidisciplinar, envolvendo as áreas de computação, física, matemática e eletrônica.

2.1 Iniciação ao Curumim

Quando a equipe estabeleceu os primeiros contatos com o *kit* robótico já havia concluído o primeiro e o segundo anos do ano do curso técnico em informática integrado ao ensino médio, ou seja, já possuía experiência com lógica de programação e programação estrutura com a linguagem de programação C. Estes conhecimentos proporcionaram uma base para o projeto de robótica com uso do *kit* robótico Curumim, que utiliza programação em C++ com conceitos de orientação a objetos.

Cabe destacar que o *kit* já vem de fábrica montado e pronto para ser programado, assim a equipe teve que se preocupar apenas com a parte programável do robô. Para programá-lo é necessário usar o software Curumin ou o Visual C++.

2.1.1 Software Curumim

A XBot disponibiliza em seu site uma série de ferramentas úteis para a programação do robô. Entre elas destacam-se [XBOTa, 2012]:

(i) um software para testar o recebimento de dados que são enviados pela entrada padrão, ou seja, o teclado;

(ii) uma ferramenta para programar em blocos (Figura 1), que permitem que o usuário informe o ponto de início e de fim da execução de ações que podem ser condicionais ou laços de repetição de ações que o robô deve executar;

(iii) recursos para programação, que são úteis para quem deseja controlar o Curumim através de programas C/C++ mais detalhados. Estas ferramentas incluem conjuntos de bibliotecas, arquivos de testes e com exemplos, o ambiente Visual C++ versão 2005,

(iv) documentações, que compreendem apostilas relacionadas tanto ao hardware quanto ao software do *kit* robótico Curumin.

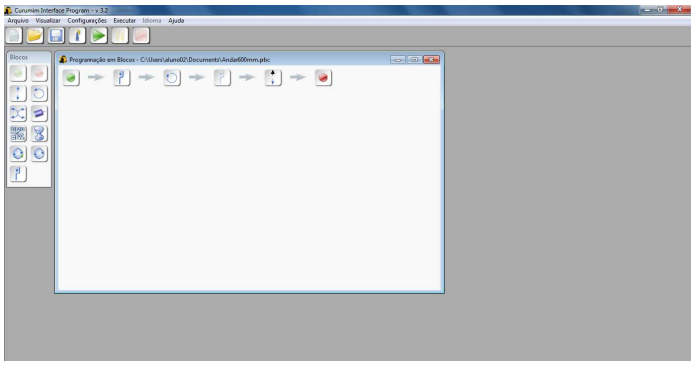

Figura 1 – Ambiente de Programação com Blocos

2.1.2 Comunicação Curumim

O robô recebe os comandos de execução por meio de sinal, o qual por sua vez, é transmitido através do rádio base. Os comandos são provenientes do código que está sendo executado no computador. Para isso, a XBot disponibiliza o *driver* do rádio base para ser instalado na máquina e assim, para que o mesmo consiga passar o código para o Curumim [XBOTb, 2012].

2.1.3 *Kit* Curumim

Quando o Curumim é adquirido vários itens fundamentais fazem parte do *kit* e acompanham o robô (Figura 2) [XBOTb, 2012; XBOTc, 2013]:

(i) o rádio base - que é responsável por enviar os sinais/ações que são entrados no teclado para o robô;

(ii) baterias recarregáveis e carregador de bateria, de forma que o robô possa se locomover sem estar conectado/ligado a uma tomada.

Figura 2 – *Kit* robótico Curumin – Hardware Básico

Esses itens constituem um *kit* quase completo das peças que seriam fundamentais para que o mesmo funcione com todos os recursos possíveis. A placa de vídeo, necessária para que o Curumim consiga reconhecer qualquer tipo de imagem ou proceder com a leitura de QRCodes, não acompanha o *kit* Curumim que é fornecido pela empresa. Para realizar qualquer este tipo de processamento é necessário adquirir a placa separadamente [XBOTb, 2012].

2.1.4 Assistência Técnica

Durante os meses em que pesquisafoi desenvolvida a equipe se deparou com alguns problemas que tiveram que ser resolvidos. Para tanto, ela contou com a ajuda de um funcionário do setor de suporte aos clientes da XBot, que oferece um curso para inicialização de programação do Curumim.

Este curso auxiliou a equipe para que fosse realizado o primeiro contato com o robô. Mesmo após esta formação inicial a equipe continuou contando com a ajuda da assistência técnica da empresa sempre que necessário por meio de chamadas online.

2.2 Sensores do Curumim

O hardware do *kit* Curumin é bem simples, como mencionado previamente (robô, rádio base e baterias). Mas um dos recursos que é muito importante no *kit* e que merece ser destacado são os sensores.

Com os sensores o robô detecta obstáculos, o que possibilita que ele realize desvios ou percorra caminhos diferentes. Para esta pesquisa os sensores são essenciais para a autonomia dos robôs.

O Curumim possui cinco sensores infravermelho, distribuídos da seguinte forma e conforme ilustra a Figura 3 [XBOTb, 2012; XBOTc, 2013]:

- Sensor número 5, sensor frontal ;
- Sensores 1 e 2, chamados de sensores laterais;
- Sensores 3 e 4, denominados sensores traseiros.

Figura 3 - Sensores Infravemelho do *kit* Curumim

3 AMBIENTE DE PROGRAMAÇÃO VISUALC++

A programação em blocos (Figura 1), mesmo sendo útil para quem esta iniciando, possui diversas limitações, que podem ser resolvidas com a linguagem C++. Por exemplo, com a programação em blocos não é possível utilizar o comando else e realizar uma condição sem utilizar algum dos sensores.

No site da Xbot [XBOTc, 2012]é disponibilizado o ambiente Visual C++ 2005, que, nos primeiros testes realizados apresentou alguns problemas de compatibilidade com o Windows 7 já que a versão que encontra-se no site não é a mais recente e nem a mais atualizada.

Então, para a realização da pesquisa, a equipe utilizou o Visual C++ 2008, que também possui problemas de compatibilidade, porém não inviabiliza a sua utilização e a programação dos robôs funcionou corretamente.

3.1 Inserção de bibliotecas e includes

De acordo com o manual disponibilizado, ao criar um projeto é necessário acrescentar manualmente uma série de bibliotecas para que a programação para o Curumim seja possível [XBOTb, 2012].

Além disso, para iniciar um novo programa e utilizar algumas das bibliotecas do Curumin é preciso utilizar alguns *includes*. Sem o uso dessas bibliotecas o programa não é executado. Cabe destacar aqui, que uma das restrições encontradas no ambiente é que se as bibliotecas não forem incluídas na ordem correta o programa não funciona corretamente e o robô não executa as funcionalidades esperadas.

3.2 Principais Funções

No material didático disponibilizado no site do *kit* robótico Curumin [XBOTa, 2012], há explicações das funções utilizadas para o deslocamento do robô com a linguagem de programação C++ (como, por exemplo, *move*(), *rotate*(), *turn*(), etc.), Cada uma dessas funções possui associada documentacao que facilitando o entendimento da semântica e da sintaxe para quem deseja programar o robô sem o uso dos blocos.

É importante observar que é programador quem deve identificar qual função é a mais adequada para a funcionalidade esperada, e toda a configuração da ação esperada pelo robô deve ser informada via parâmetro de função.

Além das funções citadas previamente, as bibliotecas ainda fornecem as funções *setSource()* e *setTarget()*:

- *curumim->setSource(<EndereçoDoRadioBase)*;
- *curumim->setTarget(<EndereçoDoCurumim)*.

Essas funções são usadas para determinar o endereço saída de dados (rádio base) e o endereço do robô, na verdade, elas são responsáveis pela comunicação do Curumim com os comandos enviados pelo computador, o qual recebe os comandos através do teclado.

4 ESTUDO DE CASO: LABIRINTO FIXO

Primeiramente, a equipe elaborou um estudo de caso onde o robô se deslocaria por um labirinto fixo, sem alterações no trajeto. Todo o código em C++ foi desenvolvido de modo que o robô Curumim se movesse através de um labirinto fixo, onde recebia as medidas exatas que ele deveria percorrer.

Durante estes testes foi possível constatar que o robô não possuía a precisão esperada, e que as funções precisavam ser ajustadas a todo momento, de modo que o robô conseguisse concluir todo o percurso sem bater em nenhum obstáculo.

Os problemas encontrados neste estudo de caso foram desde a simples funcionalidade de fazer o robô andar reto até fazer uma rotação de 90º. Neste caso, por exemplo, ao invés de andar reto por uma determinada distância ele percorria uma distância maior ou menor do que a que foi solicitada. Além disso, o robô não andava reto, variando a sua angulação a cada comando solicitado.

Resolvemos realizar uma série de testes na intenção de comprovar que alguns problemas de medidas poderiam ou não afetar a programação do labirinto com sensores.

5 TESTES DE MEDIDAS

Conforme mencionado anteriormente, sentiu-se a necessidade de desenvolver os testes que serao apresentados aqui, após verificar que o Curumim parecia não executar o que lhe era solicitado. A variação nas distâncias visível e imprevisível, podendo variar para mais ou para menos.

A falta de precisão do Curumim é o maior problema para a realização da sua automatização, pois interfere diretamente na ideia de aplicação de um labirinto mutável.

5.1 Andar reto

Os mateirias didáticos que descrevem como usar o robô Curumin recomendam o uso de uma superfície plana para o deslocamento do robô, pois isto em superfícies com muito atrito afetariam o seu desempenho. A equipe resolveu realizar os testes do Curumim andando em linha reta em dois tipos de superfícies: (i) no chão com revestimento em parquet e em cima de uma mesa com revestimento de MDF. Estabeleceu-se que o robô deveria percorrer 600 mm, sendo que está medida foi "marcada" nas duas superfícies.

Os resultados podem ser analisados a partir dos dados tabulados na Tabela 1, onde há a comparação dos resultados obtidos nas duas superfícies, o número de tentativas realizadas e a média obtida.

Tabela 1 - Medidas da função "andar reto" do robô em duas superfícies distintas considerando a medida de 600 mm.

Tentativa	Medida real (em mm)	
	Superfície Parquet	Superfície MDF
1	566	648
2	567	657
3	548	655
4	559	547
5	543	549
Média	556,6	611,2

Como pode ser observado na Tabela 1, as irregularidades do parquet foram responsáveis pelo robô andar menos do que fora solicitado, pois as rodas do robô podem sofrer atrito ou até mesmo trancar. Já na superfície MDF, pode-se notar que, na maioria das vezes, o robô andou mais do que foi programado, pois não há nada que o impeça de percorrer o seu trajeto.

A média obtida nas duas situações será utilizada para o ajuste destas irregularidades obtidas em superfícies diferentes. Assim, quando o robô estiver se deslocando na superfície de parquet, a função será parametrizada de modo que o robô Curumim ande mais para poder realizar a medida correta, já na superfície MDF a função será ajustada para menos. Espera-se com essa decisão evitar que, por exemplo, o robô bata em algum obstáculo.

Outro problema detectado durante as medidas foi a falta do andar reto, pois o mesmo, além de não andar a medida desejada, costuma variar a sua angulação, andando "torto". Então, na realização dos testes anteriores, foram realizadas medidas sobre as variações de angulação e os dados obtidos foram tabuldo na Tabela 2.

Tabela 2 Variação angular do robô após andar 600 mm

Tentativa	Variação na angulação (em graus)	
	Superfície Parquet	Superficie MDF
1	3,5	11,15
2	4,9	1,48
3	1,9	0,87
4	2,05	5,3
5	1,6	4,7
Média	2,79	4,7

Analisando-se a Tabela 2 percebe-se que a variação angular do Curumim é, aparentemente, pequena. Porém, quanto mais comandos de andar reto forem realizados, a sua soma pode resultar em uma batida contra as paredes, podendo inclusive danificar o robô dependendo da velocidade que o mesmo estiver.

Durantes os testes também foi possível observar a direção para qual o robô convergia, porém nas duas situações o robô Curumim só convergiu duas vezes para a esquerda (uma na

superfície parquet e outra na superfície MDF) dentre cinco testes foram realizados em cada uma das superfícies. Assim, conclui-se que a correção deverá ser realizada para a direita.

6 ESTUDO DE CASO: LABIRINTO MUTÁVEL

O labirinto mutável surgiu com da necessidade de que o Curumim deveria conseguir se locomover livremente, sem que a modificação do caminho pudesse alterar o funcionamento do robô. Ou seja, mesmo com a mudança no caminho, o robô Curumim deve realizar testes com os seus sensores para que o mesmo consiga sair sem realizar nenhum tipo de colisão, como, por exemplo, ao se aproximar de uma parede, o mesmo deverá fazer uma curva para desviar do obstáculo até que consiga finalizar o percurso.

A princípio, as condições utilizadas seriam baseadas no esquema ilustrado pela Figura 4.

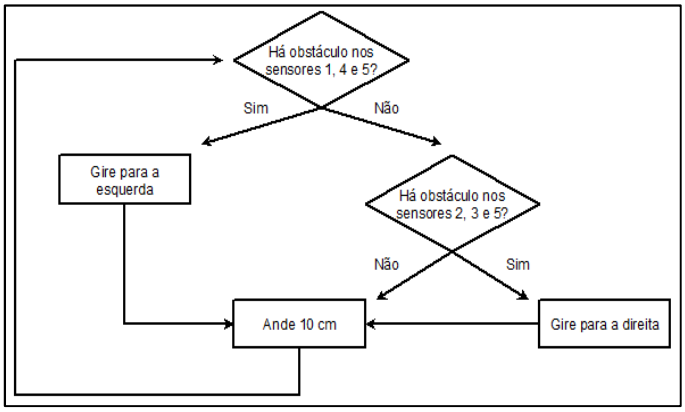

Figura 4 - Esquema em qual o código será baseado

Apesar da má distribuição dos sensores no Curumim, não foi possível percever se há algum ponto cego que pudesse influenciar de forma negativa o código, pois o procedimento para uso dos sensores é bem simples. E a utilização dos resultados testes realizados anteriormente seria para que fosse evitada a colisão do robô nas paredes paralelas a ele, fazendo com que ele rotacione x graus para a direita após o movimento de andar reto.

Durante a programação, a equipe observou que as condições não estavam corretas, pois há possibilidade dos sensores traseiros não detectarem a parede. Então, foram realizadas correções com a condição ilustrada pela Figura 5, abaixo. Esse teste permite que o Curumim analise se há um obstáculo detectado dentro de uma distância de 150 mm no sensor frontal, o 5, e se algum dos sensores laterais esquerdos, o 2 ou o 3 também encontraram. Se verdadeiro, o robô gira para a direita.

```
if((((curumim->sensors(curumim->getTarget(),curumim->getSource(), 4, 250))== 1) && (((curumim->sensors(curumim->getTarget(),curumim->getSource(), 1, 200))== 1) || ((curumim->sensors(curumim->getTarget(),curumim->getSource(), 2, 200))== 1))))
```

Figura 5 – Condição para corrigir a possibilidade dos sensores traseiros não detectarem a parede

Durante alguns testes, as condições estavam corretas, mas o robô confundia os comandos e, às vezes, virava a esquerda ao invés de virar para a direita. Após algumas tentativas, observou-se que que o problema estava na inserção dos sensores traseiros, pois um dos sensores traseiros acabava detectando a parede do labirinto. Assim, resolveu-se excluir os sensores traseiros da programação do labirinto e o código completo está ilustrado na Figura 6:

```
if((((curumim->sensors(curumim->getTarget(),curumim->getSource(), 4, 150))== 1) && (curumim->sensors(curumim->getTarget(),curumim->getSource(), 1, 300))== 1))
        curumim->rotate(curumim->getTarget(), curumim->getSource(), -90, 0);
else{
        if((((curumim->sensors(curumim->getTarget(),curumim->getSource(), 4, 150))== 1) && (curumim->sensors(curumim->getTarget(),curumim->getSource(), 0, 300))== 1))
                curumim->rotate(curumim->getTarget(), curumim->getSource(), 90, 0);
curumim->move(curumim->getTarget(), curumim->getSource(), 100, 0, 300);
```

Figura 6 – Condições finais sem a correção dos problemas encontrados durante o teste de medidas

O código faz com que o robô teste primeiramente se há um obstáculo no sensor 5 e no sensor 2, se a condição é verdadeira, ele gira para a direita. Se a condição for falsa, o robô gira para a esquerda. Após testar as condições, o robô anda 100 mm.

Este código final permite que o robô realize o percurso, em tese, sem nenhum problema. Porém, os problemas na correção da variação angular do Curumim na superfície Parquet acabam inviabilizando a validação do código, pois o Curumim choca-se com a parede após andar alguns milímetros.

6.1 CORREÇÃO DOS PROBLEMAS ENCONTRADOS DURANTE O TESTE DE MEDIDAS

Apesar dos testes terem sido realizados e de, aparentemente, a equipe ter encontrado uma solução para as variações angulares, os consertos foram inviabilizados devido aos limites da função rotate() do Curumim. O problema foi detectado enquanto o robô era programado o robô, pois ele não realizava a rotação de 10º. Então resolveu-se encontrar qual era o límite mínimo de graus necessário para que o robô realizasse o comando e pode-se perceber que o robô só o executava quando a rotação era maior ou igual a 20º.

O que chamou a atenção, pois a utilização de números menores que 20º não gera nenhum tipo de erro na compilação, ele somente não executa. Ao realizar-se uam pesquisa sobre esse fato observou-se que não há nenhuma informação que especifíque um limite mínimo de graus para que o robô realize a rotação.

7 CONSIDERAÇÕES FINAIS

Os estudos de caso elaborados “Labirinto Fixo” e “Labirinto Mutável” possibilitaram o uso de conceitos de física e matemática aliados à programação dos robôs, com isso foi possível integrar conceitos. Além disso, os estudos de caso possibilitaram à equipe compreender que a resolução de problemas não fica restrita à área de programação, e que pode envolver conceitos de áreas afins.

Mesmo possuindo influências negativas no “Labirinto Mutável”, as imprecisões do Curumim tiveram, de certo modo, uma importância positiva no desenvolvimento do trabalho, pois fez com que a equipe procurasse por alternativas que iriam além das disponibilizadas pela empresa XBot. Foi necessário analisar o funcionamento do robô, para só então conseguir programá-lo, tentando corrigir os problemas de modo que o resultado final fosse satisfatório.

Um dos grandes desafios do projeto foi o desenvolvimento do estudo de caso e a resolução dos problemas encontrados através do trabalho em equipe. Isso fortaleceu as relações pessoais e interpessoais, através da troca de conhecimento entre os envolvidos no projeto.

Como trabalho futuro pretende-se utilizar as mesmas soluções utilizadas no kit robótico Curumin no kit robótico KRE, porém utilizando a linguagem de programação Java.

AGRADECIMENTOS

A equipe do projeto gostaria de agradecer ao CNPq por financiar as bolsas PIBIC-EM das alunas Luiza Mostoswiski Oliveira e Karen Givanaz Nunes envolvidas com o projeto, à PETROBRAS e ao BNDES pela doação dos *kits* robóticos e ao IFRS por fornecer apoio financeiro à aquisição de alguns materiais de consumo e permanentes necessários à execução do projeto.

REFERÊNCIAS BIBLIOGRÁFICAS

XBOTa. (2012) “Curumim – Apostila de software V1.1”. Disponível em: www.xbot.com.br/externo/cd_curumim/Apostila_Curumim_Software_v1.1.pdf. Acesso em junho de 2013.

XBOTb (2012) “Curumin - Apostila de Hardware V2.0 Hardware”. Disponível em: www.xbot.com.br/externo/cd_curumim/Apostila_Curumim_Hardware_v2.0.pdf. Acesso em: abril de 2013.

XBOTc. “XBot – Robótica para educação, pesquisa e entretenimento”. Disponível em: http://www.xbot.com.br/. Acesso em: 25 de jul. 2013.