Desenvolvimento do Ambiente de Programação para Movimentação de um Robô Móvel Baseado em Blocos Lógicos

Adriana Herden, Marcos R. B. Vallim, Márcio H. Rodrigues, Marcel D. A. Siqueira, André L. S. Moscato
UTFPR - Universidade Tecnológica Federal do Paraná
Centro de Experimentação Ninho de Pardais
Av. Alberto Carazzai, 1640. Centro
CEP 86.300-000 Cornélio Procópio, PR

herden@utfpr.edu.br, mvallim@utfpr.edu.br, marciohr13@gmail.com, marceldanilo@gmail.com, andremoscato@yahoo.com.br

Resumo. *Este artigo mostra o desenvolvimento do ambiente de programação para um kit de robótica educacional baseado em blocos lógicos que representam as ações de um robô móvel. O trabalho faz parte de um projeto de pesquisa chamado "Ninho de Pardais", que visa implementar um centro de experimentação em tecnologias educacionais. O projeto conta com uma equipe multidisciplinar de professores e estudantes.*

1. Introdução

O projeto "Ninho de Pardais" começou em 2007 a partir de uma oportunidade criada com a abertura da "Chamada Pública MCT/FINEP/FNDCT – PROMOVE – Engenharia no Ensino Médio 05/2006", a qual visava apoio financeiro à implementação de projetos inovadores, a fim de promover a interação das ciências da engenharia com o ensino em escolas de nível médio [1].

Neste contexto, o projeto inovou desde o início promovendo oficinas de robótica educacional para alunos da rede pública de ensino. Por meio das oficinas houve o despertar do interesse e da criatividade por parte dos alunos, de maneira lúdica, durante as atividades realizadas com os kits comerciais. Por fim, a integração com a rede pública de ensino foi expressa pela implantação de núcleos de robótica nas escolas estaduais da região conhecida como Norte Pioneiro do Estado do Paraná. Assim, o centro de experimentação de tecnologias educacionais, no aspecto social, ultrapassa os limites da universidade e atinge espaços nas escolas. Por fim, os resultados obsevados, até o momento, são mais alunos no ensino superior nos cursos de ciência e tecnologia.

Dentre uma das metas científicas do projeto Ninho de Pardais destaca-se a criação de um kit de robótica educacional próprio, o qual se caracteriza em um módulo de controle implementado em FPGA (*Field Programmable Gate Array*), elementos mecânicos de baixo custo e um software de ambiente de programação. O kit está em desenvolvimento e conta com uma equipe interdisciplinar reunindo alunos bolsistas e voluntários de iniciação científica (superior ou ensino médio), de estágios, de extensão,

de trabalho de conclusão de curso das áreas de elétrica, mecânica e informática. Em resumo, o ambiente de programação inicialmente é responsável por capturar dos usuários, via interface gráfica, as ações de movimento para o robô móvel, e posteriormente compilar e transmitir esta programação via conexão USB (*Universial Serial Bus)* para o microcontrolador do robô.

De forma geral, os componentes do ambiente de programação disponibilizam as ações que movimentarão um robô e o habilitarão para participar dos desafios propostos. Estas ações comuns aos robôs móveis foram identificadas após uma análise de usabilidade realizada em três kits comerciais de robótica educacional. A partir desta análise foram identificados os blocos de programação, que devem ser capaz de expressar a lógica para a movimentação do robô. Os blocos de programação serviram como estratégia para aprofundar e representar o entendimento da lógica de cada ação do robô móvel.

No restante do documento, é apresentado na seção 2 o uso dos blocos lógicos como estratégia de entendimento, na seção 3 a composição do Kit Ninho de Pardais, na seção 4 o ambiente de programação, na seção 5 o método de avaliação, na seção 6 os resultados até o momento e discussões sobre o trabalho. Por fim, na seção 7 as conclusões e na seção 8 os agradecimentos.

2. Blocos Lógicos como Estratégia de Entendimento

No início dos estudos optou-se por entender os kits de robótica educacional existentes no mercado. Em geral, os kits de robótica educacional são caros, pois a maioria deles é importada. A escolha dos kits baseou-se na participação dos fabricantes neste mercado. Por meio de pesquisa, foram identificadas as empresas como a Lego®, a FischerTechnik®, e por fim uma empresa nacional chamada PNCA®.

De cada uma destas empresas foi escolhido um tipo de kit para análise. Os kits analisados foram: (1) kit Lego Mindstorms NXT; (2) kit Robo Mobile e o (3) kit Alpha. Todos estes possuem elementos eletrônicos, mecânicos e de software. E diferenciam-se no ambiente de programação, especialmente nos modelos prontos de programação de desafios, na forma de transmissão das configurações dos motores e sensores, nos elementos mecânicos entre outros.

Durante a análise dos kits comerciais o foco estava na identificação das funcionalidades para o ambiente de programação. Os métodos de análise de usabilidade utilizados foram avaliação heurística e percurso cognitivo, e para interpretação dos resultados foi utilizada a análise de características [2]. Em resumo, o estudo auxiliou na identificação das características principais que poderiam estar presentes no Kit Ninho de Pardais. Algumas destas características são: criar modelos mentais para as tarefas do usuário, usar metáforas e ícones, mostrar os modelos de programação em vários formatos, entre outros.

Destaca-se ainda outra estratégia utilizada para o entendimento das funcionalidades do ambiente de programação, que foi o conceito de programação por blocos, que caracteriza a linguagem visual NXT-G. Nesta linguagem ao invés de

escrever um programa em texto, trabalha-se em ícones que representam unidades independentes de instruções de programa. Estes blocos podem ser arrastados e dispostos conforme uma lógica desejada, seguindo a mesma ordem para execução [3].

Os primeiros blocos criados foram aprimorados baseando-se na programação gráfica da empresa Lego, especificamente no software Lego Mindstorms Edu NXT. Baseando-se em kits de robótica analisados foram concebidos os blocos de programação fundamentais para a programação básica do robô, a saber: bloco declaração, bloco mover, bloco decisão, bloco espera, bloco contador, bloco saltar.

3. Kit de Robótica Educacional Ninho de Pardais

O projeto Ninho de Pardais se caracteriza pela integração multidisciplinar das áreas de informática, engenharia elétrica e engenharia mecânica, especialmente para a criação do seu próprio kit de robótica educacional.

O módulo de software está em fase de desenvolvimento das interfaces, e já dispõe de boa usabilidade, baseando-se em ícones e metáforas. Visto que as funcionalidades para movimentar o robô já foram implementadas em linguagem VB.NET e respectivamente testadas [4]. O protótipo de um módulo de hardware foi desenvolvido, embarcando funções correspondentes de acesso ao hardware do robô. Sua tarefa básica é de controle de motores e aquisição paralela de sinais de sensores.

3.1. Componentes da Área de Computação

Os elementos de computação estão relacionados à análise e desenvolvimento do ambiente de programação, que é responsável por recuperar a programação que o usuário faz antes de transmiti-la para o módulo de controle, ou seja, o hardware com os componentes eletrônicos que controlam os movimentos do robô. O robô construído é do tipo móvel, logo, o controle deve ser prioritariamente autônomo. Assim, as instruções anteriormente programadas no ambiente são transmitidas para ele por meio de conexão USB, e em seguida o robô é desconectado, executando o controle embarcado.

Existem dois módulos de software do ambiente de programação, e estes são complementares, pois abordam funcionalidades diferentes. Um módulo de ações para o robô foi baseado na identificação de blocos lógicos, ou também chamado de blocos de programação. Ele trata de comandos básicos digitados pelos usuários como acionamento de motores, leitura de sensores, controle de variáveis e estrutura de controle. O outro módulo foi baseado na comunicação entre o computador pessoal e o microcontrolador do robô, a fim de transmitir um conjunto de instruções via USB, determinando a característica de autonomia do robô móvel.

Inicialmente foi estudada a geração de um arquivo com extensão "`.c`" para ser compilado e gravado no microcontrolador do protótipo. Depois foram estudadas algumas APIs USB para Windows como `JSR-80` e `jUSB` para transmissão dos arquivos. As implementações seguiram o paradigma de orientação a objetos e optaram pelo uso do `ARRAY LIST`, a fim de permitir que o compilador do processador embarcado no FPGA lesse um arquivo texto contendo a sequenciação de comandos

digitados pelo usuário. Foram realizados testes de leitura deste arquivo gerado para verificar a validade da interpretação do FPGA. A Figura 1 mostra parte de um método para adicionar uma *String* no `ARRAY LIST`.

```
Try
    NIPAR.IncluirLista.addlista("usleep" & "(" & intespera & ")" & ";")
    Return True
Catch ex As Exception
    Return False
End Try
```

Figura 1. Função para Adicionar uma String no ArrayList

Atualmente o projeto está em desenvolvimento com ênfase no estudo e implementação dos blocos de programação identificados. Cada bloco é responsável por ações dos sensores e atuadores do robô.

3.2. Componentes da Área de Engenharia Elétrica

Os elementos de eletrônica estão relacionados ao uso de uma placa para realizar as simulações do software. A placa de desenvolvimento DE2 é uma placa para estudos e desenvolvimento de sistemas com FPGA. É composta de um FPGA da Altera®, o Cyclone II 2C35, e diversos periféricos, como chaves, LEDs, interfaces de comunicação USB e Serial, slot para cartão SD, módulo de comunicação por infravermelho, expansão de pinos, display LCD 16x2, memórias SRAM e FLASH, entre outros.

A plataforma de robótica proposta neste projeto é baseada em FPGA, dispositivo que pode ser configurado para uso em robótica com um sistema semelhante ao que se tem dentro de um computador, isto é, um processador que executa instruções especificadas por um usuário, lê dados de entrada e controla saídas. Os objetivos desta plataforma são de processar sinais de sensores, executar rotinas pré-programadas e controlar motores.

Para se alcançar isto, inicialmente há a construção de um núcleo de processador com um sistema Nios II, gerado pelo SOPC *Builder* (*System on a Programmable Chip Builder*) que é um componente do Quartus II capaz de gerar sistemas microprocessados. Depois este é instanciado em um projeto de hardware reconfigurável, e dentro deste, são desenvolvidos módulos de controle de motores e de sensoriamento. Logo após, o projeto é compilado e configurado no FPGA, e o sistema fica pronto para receber e executar uma programação, o que é realizado pelo Nios II *Flash Programmer*, que é um aplicativo executado no modo *command-line*, e tem um papel fundamental na integração de softwares.

Cada um dos blocos lógicos criados cumpre funções específicas no arquivo gerado em linguagem C, e fazem a interface com os recursos de hardware da plataforma. O bloco da Figura 2 é responsável, dentro de seu contexto, por propulsionar um robô que usa dois motores (canais ‘A’ e ‘B’), com uma curvatura determinada pelo botão *Trackbar* (notado por “Steering”) e com determinada velocidade (ajustada em ‘Power’). Além desses parâmetros, características de duração e reação podem ser escolhidas em nível, isto é, quanto o robô executará (tempo, medida de ângulo e ilimitado) e como os

atuadores reagirão após isso, permitindo-se estados de ‘brake’ (frenagem instantânea) e ‘float’ (permite inércia).

Figura 2. Bloco de programação do Lego Mindstorms Educations NXT

Este tipo de bloco é fundamental para a programação do robô, e se fez necessário na versão de testes do ambiente de programação desenvolvido no projeto do kit próprio. Na Figura 3 é mostrada a interface gráfica feita em VB.NET correspondente ao bloco de movimentação do robô, baseado Figura 2. Este bloco foi elaborado partindo-se do código em linguagem C mascarado pela interface e a definição das funções de acordo com os recursos de hardware.

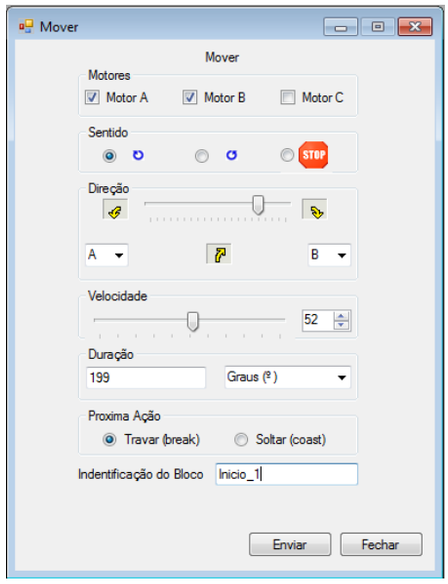

Figura 3. Bloco Mover em VB.NET

A ação de propulsão é realizada a partir da escolha de dois dos três motores disponíveis. Um sentido comum de rotação é adotado e uma determinada velocidade. No *TrackBar*, pode ser escolhida uma determinada curvatura, possibilitando ao robô fazer desde curvas suaves ou até mesmo girar sobre o próprio eixo com apenas um bloco de programação. Esta curvatura é interpretada com um equacionamento simples, e é uma função do Nios II baseada no valor lido no componente *TrackBar*. Esta dará valores de -10 a 10, indicando a rotação sobre o próprio eixo no sentido horário, anti-horário ou em linha reta.

Este bloco permite ainda o movimento do robô com dois motores em sincronia, ou acionamento simples de um ou três motores. O primeiro modo é o aplicado para movimentos com direção controlada, ajustada pelo campo 'Direção' do bloco. Para os dois modos, além dos parâmetros básicos de velocidade e sentido, são oferecidos modos de regime em 'livre', 'graus', 'rotações' e 'tempo'. Quando não é escolhido o movimento livre, pode-se escolher o modo de parada dos motores, com 'Travar' ou 'Soltar'. O primeiro freia instantaneamente o motor, e o segundo permite o movimento inercial.

3.3. Componentes da Área de Engenharia Mecânica

Os componentes mecânicos do Kit do Ninho de Pardais diferem-se quanto à identificação, classificação e fabricação. Estes elementos foram classificados a partir da aplicação de técnicas da engenharia reversa [5] nos kits comerciais de robótica, de acordo com as principais funções de cada elemento e tipos de encaixe. Foi possível conhecer todos os tipos de encaixe, a saber: (i) quanto à posição [ângulos, planos, direções e sentidos]; (ii) quanto à forma [interferência ou elementos de união] e (iii) quanto aos perfis [cilíndrico, cruzeta e quadrada].

Para desenvolver os elementos do kit básico foi necessário criar um elemento-chave, que servirá de base para todas as medidas dos outros elementos. Este pode ser inserido no grupo dos Elementos Estruturais.

Os elementos estruturais são responsáveis pela sustentação do robô e dos elementos que promovem o movimento. Com o estudo dos kits observou-se uma grande quantidade de elementos estruturais, e para melhor entendimento de todos esses elementos foi criada uma subdivisão dos elementos estruturais, sendo: (a) elementos estruturais básicos; (b) elementos estruturais bidimensionais e tridimensionais.

Grande parte dos elementos não se encaixa por interferência, então os elementos de ligação têm a função de realizar o encaixe entre dois ou mais elementos. Os elementos de ligação foram divididos em dois grupos de acordo com sua forma: (a) elementos de ligação de perfil uniforme; (b) elementos de ligação de perfil aleatório.

Já os elementos motores são responsáveis por promover o movimento da estrutura. Eles transformam movimentos lineares em circulares, circulares em lineares, realizam troca da direção de movimento e criam relações de movimento. Geralmente são encaixados por interferência.

4. Ambiente de Programação

Esta seção aborda como foi analisado e desenvolvido o ambiente de programação, focando a maneira como deve ser montada uma estrutura de programação para que o usuário possa da forma mais simples aplicar sua própria programação para realizar a movimentação do robô, utilizando sua própria lógica de interpretação. Optou-se pela plataforma .NET como o paradigma de programação orientada a objetos.

No estudo da movimentação do robô foi proposto que a programação do movimento fosse dividida em três passos, sendo: (i) a escolha de motores que atuarão no

movimento, (ii) o sentido do movimento que deverá ser realizado e (iii) a direção e velocidade que deverão ser definidas para este movimento. Para a escolha dos motores que atuarão no movimento do robô foi definido que um, dois ou até três motores podem ser escolhidos para ser realizada a movimentação desejada, conforme disponibilidade dos atuadores físicos (motores). Caso dois motores sejam definidos como atuadores do sistema, é habilitada a opção de escolha da direção que pode ser realizado esse movimento. Já na escolha do sentido, foi definido que pode ser escolhido um entre três sentidos possíveis, que são: para frente, para trás ou permanecer em repouso (parado).

Para a seleção de motores foi utilizado um componente da programação chamado *CheckBox*, o qual permite ao usuário fazer checagem dos motores que são representados pelas letras A, B e C. A movimentação do robô só pode ser feita se pelo menos um motor for selecionado, então em tempo de execução sempre um motor ficará selecionado para o usuário automaticamente, caso ele tente desmarcar todas as opções. Assim, a Figura 4 mostra uma função que foi elaborada para verificar as caixas de checagem referentes aos motores, retornando um número inteiro com a quantidade de motores selecionados.

```
Private Function verificaQtdMotores()
    If chkMotorA.Checked = True And chkMotorB.Checked = True And chkMotorC.Checked = True Then
        Return 3
    ElseIf (chkMotorA.Checked = True And chkMotorB.Checked = True) Or
           (chkMotorA.Checked = True And chkMotorC.Checked = True) Or
           (chkMotorB.Checked = True And chkMotorC.Checked = True) Then
        Return 2
    Else
        Return 1
    End If
End Function
```

Figura 4: Função que Verifica a Quantidade de Motores Selecionados

Para a representação da escolha do sentido foi utilizado o *RadioButton*, que é um componente utilizado em um grupo específico e que permite a seleção de apenas uma entre as demais opções do grupo. No caso da programação, foram disponibilizados três botões indicando o sentido do movimento. Sendo que se o sentido escolhido for para frente, as velocidades dos motores são positivas, se o sentido for para trás, as velocidades dos motores são negativas, e caso for parado, as velocidades são nulas.

A Figura 5 mostra uma função que recebe por parâmetro o motor selecionado, verificando qual o *RadioButton* que foi selecionado para indicar o sentido, e adiciona as informações obtidas em um `Array List` com nome de *Lista*. Por meio do método `addlista()` são identificados o motor e a velocidade definida pelo usuário através de um componente numérico.

```
Private Sub verificarSentido(ByVal motor As String)
    If rbtnSentFrente.Checked = True Then
        NIPAR.Lista.addlista("motor_" & motor & " = " & nudVelocidade.Value & ";")
    ElseIf rbtnSentRev.Checked = True Then
        NIPAR.Lista.addlista("motor_" & motor & " = " & nudVelocidade.Value * -1 & ";")
    Else
        NIPAR.Lista.addlista("motor_" & motor & " = " & 0 & ";")
    End If
End Sub
```

Figura 5: Função que Verifica o Sentido do Movimento

A direção do movimento pode ser realizada de três maneiras possíveis. Um dos movimentos possíveis é o movimento retilíneo (linha reta), no qual os motores atuarão com velocidades iguais. Outros movimentos possíveis seriam movimentos curvilíneos, sendo tanto para a esquerda quanto para a direta. Para a realização do movimento curvilíneo foi definido que um motor deve atuar com a velocidade menor que o outro, consequentemente o resultado é uma curva no movimento. A escolha de qual motor deve atuar a favor da direção é definido pelo usuário.

Para a escolha da velocidade, existe uma combinação de valores positivos em uma escala de 1 (um) a 100 (cem). De acordo com a velocidade definida pelo usuário, ela deve ser aplicada aos motores que atuam no movimento, com variação de um para outro, caso o movimento seja curvilíneo. Existe uma restrição na escolha de velocidade, que acontece quando o sentido de seu movimento é permanecer em repouso (parado).

```
Private Sub calcularVelocidade(ByVal motor As String)
    Dim velocidade As Integer
    If trkDirecao.Value < 0 Then
        If rbtnSentFrente.Checked = True Then
            velocidade = nudVelocidade.Value * (1 + trkDirecao.Value / 5)
            NIPAR.Lista.addlista("motor_" & motor & " = " & velocidade & ";")
        ElseIf rbtnSentRev.Checked = True Then
            velocidade = (nudVelocidade.Value * (-1)) * (1 + trkDirecao.Value / 5)
            NIPAR.Lista.addlista("motor_" & motor & " = " & velocidade & ";")
        Else
            NIPAR.Lista.addlista("motor_" & motor & " = " & 0 & ";")
        End If
    Else
        If rbtnSentFrente.Checked = True Then
            velocidade = nudVelocidade.Value * (1 - trkDirecao.Value / 5)
            NIPAR.Lista.addlista("motor_" & motor & " = " & velocidade & ";")
        ElseIf rbtnSentRev.Checked = True Then
            velocidade = (nudVelocidade.Value * (-1)) * (1 - trkDirecao.Value / 5)
            NIPAR.Lista.addlista("motor_" & motor & " = " & velocidade & ";")
        Else
            NIPAR.Lista.addlista("motor_" & motor & " = " & 0 & ";")
        End If
    End If
End Sub
```

Figura 6: Função que Verifica a Direção

Nos casos dos movimentos curvilíneos, quando o movimento for para a esquerda a sua curvatura será definida por o menor valor possível, ou seja, quanto mais negativo for o valor, maior a curvatura. Da mesma forma quando a curva for para direita, mas se tratando de um valor positivo, quanto maior o valor, maior será sua curvatura. A Figura 6 mostra uma função que recebe o motor que está atuando no movimento e verifica o valor do *TrackBar* de Direção, como também o sentido do movimento, aplicando uma fórmula desenvolvida para descobrir qual a real velocidade que tem que ser aplicada ao motor para que o robô realize a curva.

5. Método de Avaliação do Ambiente de Programação

A avaliação está relacionada à integração da placa de desenvolvimento DE2 com o software em execução. Neste contexto, os testes foram baseados no padrão controle de ambiente, a fim de observar sensores e atuadores após a intervenção do usuário [6].

Na Figura 7 o teste mostra como os dados inseridos na interface são interpretados pelo FPGA, e o resultado aparece quando os LEDs correspondentes são acessos. No

exemplo o usuário define um valor de 30 para “motor a” e “motor b”, sendo que o sentido é horário e anti-horário respectivamente.

Figura 7. Interface do bloco Atuadores e Funcionamento dos dados no FPGA

Os testes de implementação usando `ARRAY` mostram que existe limitação na quantidade de funções disponíveis ao usuário. Então a utilização de `ARRAY LIST` foi considerada melhor, pois permitiu ao usuário inserir tantas funções quanto fossem necessárias em sua programação.

6. Resultados e Discussões

Para a validação do código-fonte gerado pelo ambiente de programação, foram realizadas algumas simulações de movimentos possíveis do robô, de acordo com a interface gráfica com o usuário. O código-fonte desenvolvido para a simulação foi gravado em um arquivo .c e compilado pelo sistema Nios II.

A ideia de programação por blocos se mostrou eficiente após uma breve explanação sobre as funções dos blocos a usuários voluntários. As funções, por aludirem graficamente as ações físicas do robô, são compreendidas rapidamente, tornando-se ferramentas práticas ao usuário programador. Dessa forma, são eliminadas as dificuldades de uma linguagem de programação propriamente dita aos olhos de um usuário leigo. Isso atende o propósito do projeto quanto à adequação da linguagem ao uso com alunos de ensino médio que não possuem conhecimento de programação.

Um importante trabalho futuro é a construção de um mecanismo que apareça paralelamente à programação que o usuário faz, mostrando os blocos já preenchidos em um esquema de fluxograma. Isso permitiria que o usuário obtivesse uma visão geral da programação e facilitaria a elaboração de algoritmos mais extensos.

7. Conclusão

Observou-se um impacto científico e tecnológico na comunidade participante, pois o projeto agrega atualmente nove alunos de ensino médio e oito alunos de graduação. Também se destaca a visibilidade que o grupo ganhou participando efetivamente em eventos técnico-científicos com a publicação de aproximadamente trinta artigos.

As atividades do projeto também ganharam maior visibilidade, adquirindo um caráter de ser um projeto de divulgação científica e tecnológica, e atingido uma parte da população que normalmente não têm acesso a esse tipo de informação.

No aspecto do desenvolvimento de ambiente de programação ainda restam atividades que resolvam alguns problemas de usabilidade como o ajuste da intensidade de luminosidade (tons), e também diferenciação entre meta-compilação e *download* do programa para o robô móvel. As simulações do software foram feitas em uma placa Altera DE2 e os resultados mostram que não houve erros de sintaxe no código-fonte, pois apresentaram sucesso dos movimentos do robô. Vale ressaltar que quando não ocorrer o movimento no robô, provavelmente uma revisão deverá ser feita na programação do usuário.

As simulações realizadas demonstram que o sistema Nios II, que tem licença paga, foi capaz de executar um programa do usuário, ler um sensor e controlar adequadamente motores. O desafio neste momento é a migração para o uso de núcleos gratuitos de microcontroladores em lógica reconfigurável, pois o baixo custo do kit é um dos objetivos deste projeto. A análise de custo-benefício também é o objeto de estudo atualmente no módulo mecânico, pois a fabricação em baixo custo não poderá ser realizada em detrimento a qualidade das peças fabricadas.

Núcleos de robótica foram implantados nas escolas estaduais da região e cada escola recebeu na forma de cessão de uso um kit de robótica Lego NXT e um *notebook* para o desenvolvimento de suas atividades. Futuramente o Kit Ninho de Pardais também será utilizado em oficinas de robótica na rede pública de ensino.

8. Agradecimentos

Este trabalho é financiado pela FINEP – Financiadora de Estudos e Projetos (REF 4971/2006). O mesmo conta com o apoio da Fundação de Apoio à Educação, Pesquisa e Desenvolvimento Científico e Tecnológico da UTFPR (FUNTEF), da Fundação Araucária de Apoio ao Desenvolvimento Científico e Tecnológico do Paraná e do Conselho Nacional de Desenvolvimento Científico e Tecnológico - CNPq Brasil.

Referências

[1] ABENGE. “Programa de Modernização e Valorização das Engenharias – PROMOVE”, Revista de Ensino de Engenharia, Vol. 22, No. 2, 2003, pp.1-5.

[2] Benyon, D.,Interação humano computador, cap. Avaliação, Pearson Prentice Hall, São Paulo, 2.ed., 2011, p 149-165.

[3] Bishop, O., Programming Lego Mindstorms NXT, Syngress, Perth, 2008.

[4] Wakefield, C., VB. NET: guia do desenvolvedor, Alta Books Ltda, São Paulo, 2002.

[5] Ferneda, A. B., Integração de Metrologia CAD E CAM: Uma contribuição ao estudo da Engenharia Reversa, “dissertação de mestrado”, Universidade de São Paulo – Escola de Engenharia de São Carlos, São Carlos, 1999.

[6] Sommerville, I., Engenharia de software, cap. Software Embutido, Pearson Prentice Hall, São Paulo,9.ed., 2011, p. 375-394.