DuinoBlocks: Desenho e Implementação de um Ambiente de Programação Visual para Robótica Educacional

DuinoBlocks: Design and Implementation of a Visual Programming Environment for Educational Robotics

Rafael Machado Alves

Programa de Pós-Graduação em
Informática – Universidade Federal
do Rio de Janeiro – PPGI/UFRJ
rafamachadoalves@hotmail.com

Fábio Ferrentini Sampaio

Programa de Pós-Graduação em
Informática – Universidade Federal
do Rio de Janeiro – PPGI/UFRJ e
Instituto Tércio Pacitti de Aplicações
e Pesquisas Computacionais –
NCE/UFRJ
ffs@nce.ufrj.br

Marcos da Fonseca Elia

Programa de Pós-Graduação em
Informática – Universidade Federal
do Rio de Janeiro – PPGI/UFRJ e
Instituto Tércio Pacitti de Aplicações
e Pesquisas Computacionais –
NCE/UFRJ
melia@nce.ufrj.br

Resumo

O presente trabalho apresenta o desenho e implementação do ambiente com linguagem de programação visual DuinoBlocks. O referido ambiente tem por objetivo facilitar aos usuários iniciantes programarem o dispositivo robótico Arduino. Estes usuários são, preferencialmente, professores e alunos das escolas públicas brasileiras, parceiras do Programa Um Computador por Aluno (PROUCA) do Governo Federal. No processo de desenvolvimento do DuinoBlocks foi realizado levantamento bibliográfico sobre mecanismos de elaboração de algoritmos em ambientes de programação visual, sobretudo, os voltados para o hardware Arduino. Foram também feitos estudos de campo com o público-alvo durante a realização de cursos de robótica educacional. Tais estudos contribuíram para um maior entendimento das necessidades dos usuários. Testes realizados com as primeiras versões do ambiente têm demonstrado que professores se sentem mais confortáveis em trabalhar com esse ambiente em comparação com a linguagem textual padrão do Arduino (Wiring).

Palavras-Chave: robótica e educação, linguagem visual de programação, Arduino, TI no ensino.

Abstract

This paper presents the design and implementation of DuinoBlocks, a visual programming environment developed for the Arduino robotics hardware. The aims of the environment is to facilitate the process of learning programming Arduino hardware. DuinoBlocks is able to run on different machines with different operating systems including the PROUCA personal computers distributed by the Federal Government of Brazil to different schools. During the process of developing DuinoBlocks some bibliographic studies about visual programming languages were carried on. Tests conducted with the environment have shown that teachers and others feel more comfortable in working with this environment compared with the textual language of the Arduino (Wiring).

Keywords: robotics in education, visual programming language, Arduino, TI in education.

1 Introdução

O uso das tecnologias da informação e da comunicação (TIC) na educação é um campo amplo tendo em vista as possibilidades que apresenta a fim de tornar o processo ensino-aprendizagem mais criativo e estimulante. Neste sentido, diferentes iniciativas vêm sendo propostas para uma aproximação consciente das TIC no processo educacional. Dentre elas, destaca-se a Robótica Educacional (RE) ou Robótica Pedagógica, uma estratégia de caráter interdisciplinar, desafiadora e lúdica para a promoção da aprendizagem de conceitos curriculares. Para Schons et al. [1], a robótica pedagógica

constitui nova ferramenta que se encontra à disposição do professor, por meio da qual é possível demonstrar na prática muitos dos conceitos teóricos, às vezes de difícil compreensão, motivando tanto o professor como principalmente o aluno.

Através da Robótica o aluno pensa, manuseia, constrói, executa, constata o que está certo, depura o que está errado e reexecuta, ou seja, trabalha os conceitos teóricos através da prática. Neste processo a RE desenvolve competências como: raciocínio lógico, formulação e teste de hipóteses, habilidades manuais e estéticas, aplicação das teorias formuladas a atividades concretas, utilização da criatividade em diferentes situações e capacidade crítica [2].

A atividade com robótica educacional é dinâmica e motivadora, onde o esforço do educando é empregado na criação de soluções, sejam essas compostas por hardware e/ou software. As soluções visam à resolução de um problema proposto, podendo o mesmo ser do cotidiano, promovendo assim a transformação do ambiente escolar em uma oficina de inventores.

No Brasil e no exterior, diferentes trabalhos são direcionados à aplicação da robótica em sala de aula [3, 4, 5, 6, 7]. Há também uma tendência promissora a partir do uso de Simuladores Virtuais [8, 9] e Laboratórios Remotos [3, 10, 11, 12] objetivando estudos a distância e sem custos iniciais na aquisição de kits robóticos. Alguns outros trabalhos abordam metodologias de aprendizado e avaliação do uso da RE [13, 14, 15, 16] em contextos formais e não formais de ensino. Entretanto, conforme Miranda e outros [17] ainda existem poucos trabalhos preocupados em apresentar alternativas às barreiras iniciais da programação em RE para crianças que, por exemplo, não consolidaram o seu modelo hipotético-dedutivo.

É possível utilizar a robótica em sala de aula sem o uso da programação. Entretanto, os projetos construídos ficam com um escopo muito limitado e, de certo modo,

desconectados dos problemas reais. Desta forma, compreende-se que a inserção da programação abre a possibilidade de criação de sistemas inteligentes e autônomos capazes de reagir a um estímulo, expandindo os limites de atuação da RE.

Contudo, a linguagem de programação da maioria dos kits de robótica acessíveis às nossas Escolas são textuais, dificultando o trabalho do professor e do aluno, muitas vezes iniciantes em programação [18]. Por sua vez, as Linguagens de Programação Visual (ou VPL, sigla em inglês para *Visual Programming Language*) fornecem uma metáfora que ajuda o usuário a criar uma determinada ação (programa) com um mínimo de treinamento. Segundo Pasternak [19], elas reduzem a carga cognitiva sobre os estudantes que aprendem sua primeira linguagem de programação.

Este trabalho promove a Robótica Educacional de Baixo Custo com VPL no contexto do Programa Um Computador por Aluno do Governo Federal [20]. Trata especificamente do desenho e implementação de um ambiente de programação visual denominado DuinoBlocks, baseado na plataforma robótica Arduino [21].

As próximas seções estão organizadas da seguinte forma: na Seção 2 é apresentado o contexto para o desenvolvimento do presente trabalho; em seguida, na Seção 3, encontra-se uma análise de ambientes que utilizam VPL em robótica; na Seção 4, os principais requisitos para o desenvolvimento do DuinoBlocks são apresentados; na Seção 5 é descrito o ambiente proposto e implementado; na Seção 6 é apresentada uma avaliação do DuinoBlocks e os primeiros resultados obtidos; por fim, na Seção 7 conclui-se o artigo, tecendo as considerações finais e trabalhos futuros.

2 Contexto de Utilização do DuinoBlocks

2.1 Laboratório Virtual Didático (LabVad)

Este trabalho faz parte de uma linha de pesquisa em RE iniciada em nossa instituição com a dissertação de Miranda [22], que especificou e implementou artefatos de hardware de baixo custo e de software embarcado para um kit de RE. Victorino e outros [10] deram continuidade quando formularam uma arquitetura para um laboratório virtual didático (LabVad) sustentada em dois pilares: um pedagógico com o desenvolvimento de um modelo para formação docente em RE de características interativas e outro tecnológico, com a proposta de utilização de

tecnologias livres de hardware, como por exemplo o projeto Arduino, com o claro objetivo de facilitar o acesso de instituições públicas de ensino a modernas plataformas de programação em robótica educativa, seja pelo fator custo, seja pela facilidade de programação por não especialistas em informática e eletrônica, como os professores da educação básica.

A partir de então, a consolidação do LabVad vem se dando paulatinamente por meio de contribuições bastante significativas de outras pesquisas de dissertação de mestrado: Pinto [15] com a construção do modelo preconizado em 2007 [10] para a formação docente, constituído por três camadas sequenciais: pesquisadores-professores, professores-professores e professores-alunos (Modelo 3C); Alves [23] propôs o ambiente de desenvolvimento VPL DuinoBlocks, objeto do presente artigo; enquanto Souza e outros [24] vêm avançando na construção de uma plataforma de hardware e software para gerenciar o acesso remoto de uma aplicação RE usando o Arduino.

2.2 O Programa PROUCA

Outra circunstância que envolve a proposta DuinoBlocks é a sua estreita associação com o Programa Um Computador por Aluno (PROUCA) do Governo Federal. O Programa tem o propósito de promover a inclusão social das crianças brasileiras da rede pública de ensino mediante a aquisição de computadores portáteis novos e de baixo custo, com conteúdos pedagógicos. No final de 2011 o Ministério da Educação, através da SEED, CAPES e CNPq fizeram uma chamada via Edital para que grupos de pesquisa no Brasil apresentassem propostas voltadas para o uso dos laptops adquiridos pelas escolas parceiras do PROUCA¹.

O ambiente DuinoBlocks de desenvolvimento de programas (IDE), para o kit Arduino e que incorpora uma VPL mais amigável, é resultante de uma das cinco metas do projeto Uca na Cuca [3] de pesquisa científica, pesquisa tecnológica e inovação pedagógica na área de RE selecionado, dentre outros, pelo Edital citado.

O contexto do PROUCA impõe a priori certos requisitos ao DuinoBlocks: compatibilidade com o laptop ClassMate, o sistema operacional MeeGo (embarcados nos ClassMates) e o uso da robótica educacional de baixo-custo (REBC), além de outros requisitos pedagógicos tais como a faixa etária do público-alvo formada por alunos do ensino fundamental de escolas públicas.

2.3 O Projeto Arduino

O elevado custo de kits comerciais voltados para a RE ainda contribui para a pouca atividade desta no cenário da educação pública brasileira. Contudo, autores como Albuquerque e outros [25], Filho [26], Miranda e outros [17], Santos [27] e Sasahara e da Cruz [28] apresentam bons resultados em se tratando de Robótica Educacional de Baixo Custo (REBC), democratizando assim o acesso as tecnologias. A REBC utiliza materiais alternativos (sucatas), recursos de hardware e software livres, tais como o projeto Arduino, como forma de se viabilizar economicamente projetos na área de RE.

Criado em 2005 na Itália, o projeto Arduino [21] constitui uma plataforma de hardware e software abertos de fácil utilização, acessível não somente a especialistas na área de eletrônica, mas também a hobbystas ou a qualquer pessoa interessada na criação de objetos ou ambientes interativos. Segundo David Mellis [29], um dos criadores do projeto:

Nós queríamos que outras pessoas estendessem a plataforma para adequá-la às suas necessidades. Para isso, elas deveriam ter acesso ao código-fonte do software e ao projeto do hardware. Além disso, como era uma plataforma nova, ser de código aberto deu confiança às pessoas. Elas sabiam que poderiam continuar expandindo a plataforma mesmo que o desenvolvedor original desistisse dela.

O projeto ficou conhecido mundialmente, com aplicações em diferentes segmentos (p. ex. música, artes, educação e meio ambiente) e com comunidades espalhadas no planeta trocando experiências sobre suas aplicações.

O hardware Arduino é uma placa eletrônica utilizando circuitos de entradas e saídas simples, microcontrolada e desenvolvida sobre uma biblioteca escrita em C/C++. O microcontrolador da família ATMEL AVR, presente na placa, pode ser programado com a linguagem de programação Arduino, baseada na linguagem Wiring².

3 Linguagens Visuais

As primeiras VPL surgiram na década de 1960, com aperfeiçoamentos nos anos 1970 e 1980. O objetivo era de facilitar aos cientistas e entusiastas da computação acesso à programação. Entretanto, devido às limitações dos recursos gráficos da época, a sua disseminação ficou prejudicada. Com o aumento do poder computacional das máquinas, hoje encontramos disponíveis diferentes VPL

¹ <http://resultado.cnpq.br/5415026289677057>

² <http://wiring.org.co/>

para fins específicos [19].

Na ciência da computação, os ambientes de programação visual são utilizados hoje em dia como uma abordagem diferenciada ao ensino introdutório de programação de computadores em cursos superiores de áreas tecnológicas [35], [36], [37]. Essas propostas, em geral, buscam conduzir o aluno a um aprendizado facilitado, consistente e voltado a aplicações práticas, diminuindo as dificuldades na elaboração de um raciocínio estruturado para a solução de um problema computacional. Já nos ensinamentos fundamental e médio, a programação visual vem sendo proposta, principalmente, para tornar a aprendizagem de conceitos curriculares mais lúdica e motivadora [39], [40], [41].

Atualmente, novas VPL vêm sendo desenvolvidas para crianças de 5 a 7 anos de idade, como é o caso do ScratchJr [42]. A ideia aqui é dar a oportunidade a esse público infantil de explorar as possibilidades educacionais da programação de computadores [43].

Para cada um dos softwares que mais influenciaram o desenvolvimento do DuinoBlocks foram levantadas informações básicas sobre os recursos que ele oferece, comparando-se as suas funcionalidades equivalentes e identificando-se as características desejáveis para o desenvolvimento do presente projeto. Algumas características relevantes dos ambientes estudados, bem como a ausência de funcionalidades importantes, são apresentadas no final desta seção¹.

Dentre as VPL estudadas, duas abordagens diferentes de representação de algoritmos foram observadas. A primeira é caracterizada pela formação de empilhamentos ordenados, em que o fluxo de execução se dá de cima para baixo. Nesse grupo foram estudadas as linguagens Ardublock, Minibloq, ModKit, S4A e Squeak Etoys. A segunda abordagem é aquela em que o algoritmo tem uma estrutura de diagrama, onde o fluxo de execução segue por arestas e nós. Aqui avaliou-se as linguagens LEGO Mindstorms, Microsoft Robotics Developer Studio e Firefly.

Apesar deste trabalho não visar a utilização de softwares comerciais, ainda assim, estes foram estudados como fonte de referência. Já os softwares não comerciais, além de serem parcialmente compatíveis ao contexto da proposta, permitem que desenvolvedores façam o reaproveitamento de código-fonte. Quanto à licença, os softwares tratados foram classificados da seguinte forma: Softwares não comerciais: Minibloq, Ardublock, Squeak Etoys e S4A. Softwares comerciais: ModKit, LEGO

Mindstorms, Firefly e Microsoft Robotics Developer Studio.

Em relação à comunicação entre o computador e o hardware Arduino, existem dois tipos de ambientes de programação. Os não autônomos, em que os projetos desenvolvidos necessitam de uma conexão constante com o computador. Em contrapartida os ambientes autônomos - solução buscada para este trabalho - permitem que, uma vez programado, o hardware fique independente do computador.

- Ambientes não autônomos: Squeak Etoys, S4A e Firefly²

- **Squeak Etoys**³: ambiente computacional livre e de código aberto. Foi desenvolvido focando o uso por crianças utilizando estratégias de programação orientada a objetos. O ambiente permite a incorporação de mídias e provê suporte multilíngue.
- **S4A**⁴: Pode ser visto como uma derivação do ambiente Scratch [30] incorporando funcionalidades (blocos de programação) para trabalhar com robótica.
- **FireFly**⁵: Ambiente gráfico com o qual é possível programar diferentes dispositivos de hardware populares como celulares, câmeras e controladores de jogos (Wii Nunchuck e Microsoft Kinect).

- Ambientes autônomos: Minibloq, Ardublock e ModKit.

- **MiniBloq**⁶: Um de seus principais objetivos é levar a computação física e plataformas de robótica para a escola primária. Dentre suas características destacam-se: o gerador de código e a verificação de erros em tempo real; portabilidade; tradução para o português e o suporte para Linux (lançados recentemente); codificado em C++.
- **ArduBlock**⁷: Utilitário gráfico, cuja missão é gerar código compatível para o IDE Arduino que o reconhece como um "plugin". Implementa as principais funções da linguagem de programação do Arduino (Wiring), possui suporte multilíngue e é codificado em Java. Os blocos são representados

¹ Uma descrição mais detalhada dos ambientes aqui apresentados pode ser encontrada na dissertação de Mestrado do 1º autor deste trabalho em [23].

² Os ambientes aqui selecionados e estudados foram aqueles dedicados a fazer a ponte entre os mundos digital e físico, simplificando o processo de prototipagem de objetos interativos.

³ Site do Squeak Etoys: <http://www.squeakland.org/>

⁴ Site do S4A: <http://seaside.citilab.eu/>

⁵ Site do FireFly: <http://fireflyexperiments.com/>

⁶ Site do MiniBloq: <http://blog.minibloq.org/>

⁷ Site do ArduBlock: <http://blog.ardublock.com/>

por figuras (como no Minibloq).

- **ModKit¹**: Ambiente de programação para microcontroladores que permite programar o hardware Arduino e outros compatíveis. Suas principais características são: blocos de programação inspirados no Scratch; é executado no navegador e requer um widget (ainda não disponível para Linux) na área de trabalho para se comunicar com o hardware; os projetos desenvolvidos podem ser salvos na nuvem; a detecção automática de hardware e a versão desktop foram lançadas recentemente; sua versão gratuita não permite utilizar recursos básicos como a criação de variáveis.

A seguir apresentamos um resumo comentado das principais características dos ambientes aqui apresentados seguidos, no Quadro 1, de uma síntese dos mesmos:

- A presença de categorias de comandos voltadas para kits robóticos é, definitivamente, um ponto a favor do Ardublock sobre o S4A. Com o Ardublock, é possível trabalhar em um nível mais elevado de abstração na programação do hardware (ou seja, programação direcionada ao dispositivo) ao invés de se preocupar com uma porta lógica do Arduino como acontece no S4A;

- Uma diferença do S4A em relação aos outros softwares é a modo como o programa carregado no hardware é gerenciado. Com o S4A o computador tem de permanecer ligado e conectado ao hardware constantemente através de um cabo USB, enquanto que com outros softwares esta restrição não se aplica, uma vez que o programa é traduzido e posteriormente carregado na memória do Arduino. Entretanto, suas vantagens são:

- Efeito imediato do código escrito – A inserção/remoção ou alteração de um parâmetro de comandos do programa do usuário implicam, de forma instantânea, na execução do código carregado no hardware.
- Kit robótico capaz de interagir com o ambiente Scratch – O ambiente Scratch e o kit robótico

podem trocar informações/dados, permitindo novas possibilidades de projetos.

- Com exceção do ModKit, nenhum dos ambientes tratados anteriormente salvam seus projetos na nuvem. Apesar de ser uma modificação do Scratch e este possuir recursos para salvar seus projetos na nuvem, os projetos criados no S4A não são compartilhados na comunidade virtual do Scratch, por conta dos termos de uso do ambiente.

Assim, a partir do estudo bibliográfico realizado sobre programação visual voltada para robótica percebeu-se a ausência de um ambiente que congregasse ao mesmo tempo as seguintes características: ambiente web de fácil aprendizado, multilíngue, voltado para o contexto educacional e com acesso gratuito.

4 Requisitos do Sistema

A compreensão completa dos requisitos é fundamental para um desenvolvimento de software bem-sucedido. Porém, a identificação inicial de todos os requisitos, na realidade, não é sequencial [31]. Neste projeto a identificação das necessidades dos usuários foi realizada durante todo o seu ciclo de vida, mas com uma maior intensidade no seu começo.

Desta forma, o levantamento de requisitos foi feito em diferentes momentos, tendo iniciado com as condições de contorno contextuais descritas na Seção 2 (projetos LabVad, PROUCA e Arduino) e com a revisão da literatura de softwares com VPL (Seção 3). A equipe estendida do projeto Uca na Cuca, em diferentes seminários e reuniões, também discutiu sobre as principais características a serem implementadas no software. Por fim, pesquisas de campo realizadas durante os cursos de REBC promovidos pelo projeto, também foram de grande importância na identificação das necessidades do público-alvo.

A coleta de opiniões de *stakeholders*, como professores em processo de formação em REBC e

	Usa Arduino	Licença	Plataforma	Execução Autônoma	Algoritmo	Idioma
DuinoBlocks	sim	não comercial	web	autônomo	pilha	multilíngue
MiniBloq	sim	não comercial	desktop	autônomo	pilha	multilíngue
Ardublock	sim	não comercial	desktop	autônomo	pilha	multilíngue
S4A	sim	não comercial	desktop	não autônomo	pilha	multilíngue
Squeak Etoys	sim	não comercial	desktop	não autônomo	pilha	multilíngue
Firefly	sim	comercial	desktop	não autônomo	diagrama	inglês
ModKit	sim	comercial	web	autônomo	pilha	inglês
Lego Mindstorms	não	comercial	desktop	autônomo	diagrama	inglês
MRDS	não	comercial	desktop	autônomo	diagrama	inglês

Quadro 1: Síntese comparativa entre os diferentes ambientes analisados

¹ Site do ModKit: <http://www.modk.it/>

especialistas na área de tecnologias no ensino e robótica, ajudou a compreender as reais necessidades dos potenciais usuários. Os três principais requisitos levantados nesta fase que o ambiente DuinoBlocks deve prover são:

- **Requisito de portabilidade:** Ambiente Multiplataforma.

Um importante aspecto levado em consideração no desenvolvimento do DuinoBlocks foi a existência de diferentes sistemas operacionais disponíveis para alunos e professores nos diferentes equipamentos utilizados por eles. Os laptops das escolas do PROUCA possuem o sistema operacional Meego [38], ou alguma outra distribuição LINUX. Por sua vez, os computadores que os alunos por ventura tenham em suas residências têm – com uma boa chance – o sistema operacional Windows. Já os computadores distribuídos aos professores da rede pública de ensino nos diferentes estados do Brasil vêm também com alguma versão Linux.

A alternativa econômica e em linha com as tendências atuais no desenvolvimento de software, foi a de implementar um ambiente multiplataforma que rodasse na nuvem [32].

Uma vez que o acesso à web ainda é precário em diversas escolas do Brasil e que o número de residências conectadas ainda é pequeno, pensou-se que o DuinoBlocks também pudesse rodar localmente (off-line) nos navegadores das máquinas do PROUCA.

O requisito definido acima traz no seu bojo três outros importantes aspectos positivos. Primeiro, elimina a complexidade de uma eventual instalação, configuração ou atualização do sistema, possibilitando aos usuários o acesso ao DuinoBlocks sem a necessidade de conhecimento sobre a tecnologia utilizada. Segundo, reduz a necessidade de espaço de memória ou disco nos laptops, uma vez que os projetos podem ser salvos na nuvem e as máquinas dos usuários não necessitam ter alta capacidade de processamento [33]. Terceiro, introduz facilidades para o compartilhamento e colaboração dos projetos desenvolvidos pelos alunos e professores.

- **Requisito de facilidade de uso:** Linguagem de Programação para Iniciantes.

Professores e alunos hoje já estão relativamente bem familiarizados com interfaces de manipulação direta como o Windows e distribuições Linux. Tais interfaces diminuem as barreiras à programação textual [34]. A utilização dos recursos gráficos e mecanismos de arrastar e soltar presentes nestes ambientes é, portanto, a 1ª escolha na definição de um ambiente de programação para iniciantes centrada no hardware Arduino.

Da mesma forma, diferentes pesquisas sobre ensino-aprendizagem de programação com linguagens visuais [35, 36, 37] têm apontado para a superioridade destas em relação as linguagens textuais na formação de novos usuários. Em tais ambientes visuais não é necessário, por exemplo, conhecer de antemão a sintaxe exata dos comandos de um programa em construção. O formato dos blocos da linguagem facilitam o encadeamento léxico e lógico dos programas, diminuindo a carga cognitiva dos (futuros) programadores.

- **Requisitos de eficiência e desempenho:** Adequação às Limitações do ClassMate.

As funcionalidades do sistema devem ser adequadas as limitadas especificações de hardware e periféricos do laptop ClassMate [44]. As principais limitações da última versão deste equipamento são:

- O tamanho reduzido da tela de 8.9": o layout do sistema deverá ser adequado a essa pequena dimensão e de preferência ajustável para que também se adeque às telas de maior tamanho e resolução.
- A ausência de mouse: a usabilidade do sistema deverá considerar o uso do touchpad.
- As limitações de processamento e armazenamento: no desenvolvimento do sistema deve-se atentar para o tempo de execução das ações e a quantidade de memória requerida.

5 O Ambiente DuinoBlocks

Neste capítulo são apresentados as principais características do DuinoBlocks quanto à sua concepção, implementação e interface, definidas a partir das condições e requisitos apresentados nas seções anteriores.

5.1 Concepção e Implementação

Em relação aos aspectos de usabilidade do ambiente, os mesmos foram projetados segundo referências às Leis da Simplicidade proposta por Maeda [45].

Por se tratar de um sistema com requisitos baseados em pesquisas, foi empregado no desenvolvimento do mesmo o modelo evolutivo de engenharia de software detalhado em [31].

A escolha da tecnologia de desenvolvimento do DuinoBlocks levou em consideração o esforço necessário para a programação da sua interface e a utilização de software livre. Desta forma, optou-se pela biblioteca Pyjamas escrita em Python e o IDE Eclipse. Outra vantagem de tais escolhas foi a possibilidade de geração

de código Javascript de forma automática, permitindo a sua execução em qualquer navegador web ou rodar sem alterações como uma aplicação desktop. O Quadro 2 apresenta os recursos utilizados no desenvolvimento do ambiente.

O aspecto gráfico de cada elemento de programação do ambiente (blocos – ver abaixo) utiliza, na sua construção, uma estrutura HTML e a formatação CSS básica (baseado no ambiente ModKit). Utilizou-se cores “fortes” para cada categoria de blocos a fim de realçar a diferença na funcionalidade dos mesmos.

Após programar uma tarefa em DuinoBlocks o usuário pode converter o seu programa para Wiring usando o tradutor do ambiente, ou enviá-lo diretamente para a placa Arduino. Nesse caso o ambiente se encarrega – de forma transparente para o usuário – da conversão do programa e envio à placa Arduino. Aqui é preciso que o DuinoBlocks esteja conectado à web, uma vez que utiliza ferramentas na nuvem para compilar o código Wiring e carregá-lo no hardware.

Recurso	Finalidade	Endereço Eletrônico
Eclipse	IDE de desenvolvimento	www.eclipse.org
Python 2.7	Compilador Python	www.python.org/getit
PyDev	Plugin do Eclipse para a linguagem Python	pydev.org
Pyjamas	Biblioteca para construção da interface	pyjs.org
Ardublock	Referência de código fonte	github.com/taweili/ardublock
Java2Python	Conversão do pacote <i>translate</i> do Ardublock	github.com/natural/java2python
CodeBender	Módulo de Comunicação com o Hardware	codebender.cc
Astah	Criação de caso de uso e diagrama de classes	astah.net
Cacoo	Criação de diagramas	cacoo.com
Fritzing	Utilização das imagens dos componentes	fritzing.org
Chrome	Navegador web para teste e depuração	www.chromium.org
DropBox	Disponibilização de link público para testes	www.dropbox.com
ClassMate PC	Testes com o computador do PROUCA	pt.wikipedia.org/wiki/Classmate_PC
Paint.NET	Edição de imagens	www.getpaint.net

Quadro 2: Ferramentas e recursos utilizados

As principais estruturas lógicas de programação do ambiente DuinoBlocks foram abstraídas a partir do conceito de Blocos: estrutura visual equivalente às instruções textuais na linguagem de baixo nível Wiring (linguagem padrão de programação do Arduino).

As Figuras 1 e 2 apresentam, respectivamente uma classificação dos Blocos existentes no ambiente e a sua representação gráfica.

O agrupamento *Categoria* permite distinguir a funcionalidade de um bloco. Atualmente o ambiente possui seis categorias: Controle (cor abóbora), Operadores (cor verde), Entrada (cor roxa), Saída (cor azul claro), Utilitários (cor azul escuro) e Variáveis (cor vermelha).

O formato dos blocos também permite identificar os possíveis encaixes corretos do ponto de vista sintático. Assim, se dois blocos não se encaixam é porque não existe uma construção sintática em Wiring equivalente.

O agrupamento *Empilhamento* permite distinguir se um determinado bloco pode ser empilhado com outros. Blocos empilháveis possuem entalhes externos que facilitam o seu encadeamento com outros. Exemplos desses blocos são aqueles utilizados para representar comandos do tipo “Sempre” (loop), leitura/escrita em um

determinado pino digital, entre outros.

Figura 1: Classificação dos Blocos no ambiente DuinoBlocks

Os blocos de *Retorno* são aqueles que podem ser encaixados dentro de outros blocos. São divididos em três subtipos diferentes: o Numérico possui bordas

arredondadas e retorna um número (ex. blocos aritméticos); o Lógico possui bordas pontiagudas e retorna um valor lógico (V ou F); e o Alfanumérico possui bordas quadradas e retorna caracteres.

Alguns blocos permitem que alguns de seus parâmetros sejam ajustados pelos usuários. Tais blocos possuem o que chamamos de *Entrada de Dados*. Estas podem ser preenchidas encaixando outros blocos, via teclado (Entradas genéricas), ou escolhendo uma opção em um Combobox (quando o conjunto de entradas possíveis é conhecido). A posição no bloco onde é permitida a entrada de um dado possui formatos diferentes, dependendo do tipo de dado permitido (Numérico, Lógico ou Alfanumérico).

Variáveis podem ser criadas pelo usuário. Elas têm escopo global, podendo ser utilizadas como entrada de dados de outros blocos ou receberem valores de acordo com o seu tipo (Numérico, Lógico ou Alfanumérico).

O ambiente DuinoBlocks conta ainda com um conjunto de funções matemáticas, trigonométricas e de conversão de valores, agrupadas na sub-categoria *Utilitários*.

O sistema permite ainda especializar comandos para determinados componentes tais como: LEDs, buzina, servo, motor, display de 7 segmentos, LCD, chave de contato, potenciômetro, LDR, termistor e joystick (Figura 4).

Figura 2: As seis diferentes categorias de blocos do ambiente DuinoBlocks

5.2 A Interface

A interface do DuinoBlocks utiliza os recursos de arrastar e soltar na manipulação das suas diferentes opções e na criação dos programas.

A fim de facilitar o entendimento e manipulação do ambiente pelo usuário, optou-se por colocar no desktop do ambiente todas as funcionalidades disponíveis.

A Figura 3 mostra o layout do ambiente DuinoBlocks, com um exemplo de algoritmo (Pisca Led) bastante conhecido na RE. O layout é dividido em 6 painéis apresentados a seguir:

- O Cabeçalho contém botões que realizam ações no software, a saber:
 - Blocos / Componentes: alternam entre os módulos

Blocos e Componentes (Figuras 3 e 4);

- Abrir / Salvar: abre e salva o projeto em um arquivo com extensão DBK;
- Desfazer / Refazer: retrocede e avança uma ação realizada pelo usuário;
- Criar Bloco: permite ao usuário criar o seu próprio bloco (procedimentos);
- Traduzir: converte o algoritmo visual em textual (linguagem Wiring);
- Idioma: suporte multilíngue.

- O Rodapé contém botões que controlam a disposição dos outros painéis, a saber: ocultar/mostrar e redimensionar. A implementação destas ações foram motivadas pela baixa resolução de tela do laptop

ClassMate. Inclui também o botão “download” para baixar o código textual traduzido.

- O Esquerdo contém uma lista de blocos separados por categorias e subcategorias;
- O Central é a área de construção do algoritmo;
- O Direito contém a tradução textual do algoritmo em Wiring.
- O Inferior contém os recursos necessários para carregar o programa no Arduino diretamente do navegador. Não há a necessidade de instalar um IDE e gerenciar as bibliotecas e os drivers corretos, tarefas que

demoram e requerem conhecimento técnico da tecnologia.

O ambiente permite ainda a utilização do recurso de Recursividade. Para tanto, cria-se um novo bloco (procedimento) e dentro do mesmo faz-se uma chamada ao procedimento criado.

Coerente com os princípios que norteiam o desenvolvimento das VPL, as mensagens de erro e o fornecimento de ajuda são apresentados num formato visual (utilizando-se elementos gráficos do ambiente) e sensíveis ao contexto em que foram solicitadas (Figuras 5 e 6).

Figura 3: Layout do Ambiente DuinoBlocks

Figura 4: Módulo de componentes

Figura 5: Módulo “Tratamento de erro”

Figura 6: Módulo “Ajuda”

O DuinoBlocks foi inicialmente concebido para programar o Arduino Uno. Porém, ele pode ser estendido para ser usado com qualquer versão de placa Arduino com a implementação de um módulo de “Hardware”.

O diagrama de classes e mais detalhes de implementação podem ser encontradas em [23].

6 Avaliação do DuinoBlocks

O grupo GINAPE¹, responsável pelo projeto Uca na Cuca, vem promovendo cursos de capacitação em Robótica Educacional de Baixo Custo para uma sequência de turmas com cerca de 15 professores dos

diferentes níveis de ensino [5].

Os cursos iniciais serviram como importantes fontes de subsídios para a definição dos requisitos do DuinoBlocks. Os dois últimos, contudo, foram oferecidos com um protótipo do DuinoBlocks em funcionamento, permitindo o planejamento de uma proposta para avaliação das funcionalidades e da interface do ambiente, bem como analisar a sua viabilidade em Escolas da Rede Pública de Pirai.

Embora esses dois cursos, de caráter teórico-prático, enquanto instrumento de pesquisa tenham alcances limitados, pois foram feitos com amostras escolhidas por conveniência e sem o controle de outros fatores intervenientes, a aplicação dos mesmos forneceu importantes subsídios para a especificação do ambiente DuinoBlocks. Segue uma apresentação do planejamento e da aplicação desses estudos, bem como dos instrumentos de análise dos resultados obtidos.

6.1. Oficina de Robótica Educacional de Baixo Custo

No primeiro trimestre de 2013 o DuinoBlocks foi submetido a testes com o objetivo de avaliar principalmente a sua usabilidade. Foi realizada uma oficina de REBC para os alunos de um curso de Pós-Graduação em Informática na Educação, onde os participantes eram professores de diferentes áreas com idades entre 23 e 50 anos, sendo que apenas um deles tinha experiência em programação de computadores. A oficina iniciou com seis participantes e terminou com quatro, devido a mudanças na agenda de dois deles.

A oficina teve carga horária de 12 horas divididas em quatro aulas. Os participantes foram orientados a levar seus laptops (rodando Windows e OS) e a se organizar em duplas. Cada dupla utilizou um kit de robótica fornecido pela nossa equipe.

Em termos práticos, as aulas abordavam a elaboração de experimentos com montagem de componentes e a sua programação. Na primeira aula do curso os alunos utilizaram somente a linguagem textual do Arduino (Wiring). Nos encontros seguintes foi também incluída a linguagem gráfica do DuinoBlocks. Durante a utilização dos dois ambientes de programação, os participantes observados foram instruídos a registrar suas impressões em um diário de bordo, além de responderem a questionários impressos e questionamentos orais [46]. Os responsáveis pela Oficina também registraram suas impressões em relatórios ao final dos encontros.

Considerando que o objetivo principal dessa oficina era o de testar o ambiente DuinoBlocks frente à linguagem textual Wiring, decidiu-se não dar grande

¹ Site do GINAPE: <http://www.nce.ufrj.br/ginape/>

ênfase na montagem física dos experimentos (que, por vezes, foram entregues semiprontos). Decidiu-se também que os algoritmos, inicialmente elaborados de forma conjunta (professor e cursistas), passassem a receber menos ajuda dos instrutores à medida que os participantes ganhavam mais experiência.

6.1.1. Resultados da Avaliação

Partiu-se de um processo inicialmente diretivo, onde o algoritmo (apresentado com projetor multimídia) era explicado e ao mesmo tempo construído junto com os participantes que o copiavam. Na sequência utilizou-se uma abordagem mais exploratória, onde os participantes eram solicitados a efetuar alterações no algoritmo construído. E por último, já no final do curso, trabalhou-se numa abordagem expressiva, onde os alunos criavam seus programas a partir de desafios (simples) propostos.

Logo nas primeiras interações com o DuinoBlocks, foi observado que a maioria dos participantes teve dificuldades para encaixar blocos. Esta dificuldade decorre do fato que os eventos de “arrastar e soltar” da biblioteca utilizada são baseados no ponteiro do mouse e por esse motivo, há situações em que arrastar uma fenda de um bloco até o entalhe de outro não necessariamente resulta em um encaixe. Desta forma, será realizada uma atualização que modifica a forma de encaixe dos blocos para deixar esta ação mais intuitiva.

Durante a elaboração dos algoritmos foi possível perceber que os participantes não tentaram fazer encaixes impossíveis, demonstrando assim, que a lógica dos blocos foi entendida. No que se refere à organização dos blocos, o agrupamento em categorias (uma cor para cada) e em subcategorias ajudaram na localização dos blocos, visto que, nenhum deles demonstrou dificuldades para encontrar um bloco representante de uma ação (comando) específica.

No último encontro da oficina (após 2 semanas de intervalo) os participantes foram solicitados a “pensar em voz alta” (técnica *think aloud* [47]) sobre o propósito de algoritmos apresentados em Wiring e em DuinoBlocks. Ao discursarem sobre os programas em Wiring, conseguiam relatar o objetivo de algumas partes do programa, sem – aparentemente – conseguir perceber o todo. No entanto, quando apresentados a algoritmos em DuinoBlocks, conseguiam responder corretamente sobre a funcionalidade dos mesmos, demonstrando um entendimento completo.

Vale ressaltar que o nível de complexidade dos programas solicitados aos participantes foi relativamente simples. Mas ainda assim os resultados obtidos apontam para o fato dos mesmos serem capazes de partir de um processo de “cópia com alterações” para o de elaboração

mental com o apoio do ambiente de programação visual DuinoBlocks.

6.2. Curso de Formação em Robótica Educacional

A utilização do ambiente DuinoBlocks foi introduzida na 3ª turma do curso de capacitação de professores do ensino fundamental e médio intitulado “Formação em Robótica Educacional com Hardware Livre” em Pirai [3]. Tais cursos têm caráter teórico-prático, combinando momentos presenciais e à distância - via ambiente virtual de aprendizagem Moodle configurado para este fim. A carga horária total prevista era de 48 horas, sendo que os encontros presenciais (4 ou 5 encontros no total) aconteciam, aproximadamente, a cada 15 dias. Ao final do curso os grupos deveriam apresentar uma atividade didática envolvendo o uso da robótica.

A 3ª turma do curso iniciou em maio de 2013 com doze participantes, sendo cinco profissionais da área de redes de computadores com experiência em programação, um profissional de TI e seis professores de diferentes áreas, iniciantes em programação.

Nas aulas do curso, os participantes foram organizados em grupos por bancada, sendo que os cursistas experientes em programação trabalharam juntos. Pediu-se a todos que mantivessem a mesma organização de grupos durante todos os encontros.

Em cada bancada havia um laptop ClassMate e um kit de robótica fornecido pelo projeto Uca na Cuca. As aulas práticas abordavam a elaboração de experimentos com montagem de componentes e a sua programação.

Nas duas primeiras aulas foram mantidas as atividades previstas no cronograma dos cursos anteriores. Nelas foram abordados os seguintes assuntos: introdução a Robótica Educacional, eletrônica básica e apresentação do Arduino com programação textual Wiring.

As atividades da terceira aula presencial foram alteradas e, no lugar de prosseguir com a programação Wiring, foram realizados os mesmos experimentos da aula anterior, porém com a programação em DuinoBlocks. Na 4ª aula presencial do curso foram realizados novos experimentos utilizando o módulo de componentes do ambiente. A apresentação dos trabalhos finais aconteceu durante um 5º encontro presencial (já em agosto de 2013), contando com a presença do Prefeito e da Secretária de Educação da Cidade.

Os dados foram coletados a partir do diário de bordo elaborado pelos participantes, das respostas dadas às tarefas propostas em texto impresso e das anotações feitas pelo professor e equipe do GINAPE/UFRJ presentes nos encontros.

6.2.1 Resultados da Avaliação do DuinoBlocks em Curso de Formação em RE

Em comparação com as turmas anteriores, os projetos executados com o DuinoBlocks foram construídos com um grau de dificuldade menor em relação ao ambiente de programação Wiring, sendo possível realizar experimentos mais complexos (p. ex. atividades que envolviam o uso de display de caracteres, servo-motor e teclado, não trabalhadas com as turmas anteriores).

Um outro aspecto interessante foi o fato dos cursistas aparentarem uma facilidade maior para “descobrir a funcionalidade” dos comandos do DuinoBlocks. Numa atividade que tinha como objetivo simular um semáforo, verificou-se que um dos grupos sem conhecimento prévio de programação foi capaz de inferir o funcionamento do bloco de repetição (loop) e empregá-lo corretamente no seu trabalho.

O grupo com integrantes com conhecimento prévio em programação realizou as atividades propostas sem nenhuma dificuldade e em tempo menor que os outros. Foram capazes de empregar variáveis nas suas soluções, mesmo sem esta funcionalidade ter sido apresentada. O ambiente também não apresentou limitações para este tipo de público, demonstrando sua viabilidade de uso por especialistas.

O módulo de comunicação com o hardware (permite o usuário carregar de forma automática um programa escrito em DuinoBlocks diretamente no hardware Arduino) foi utilizado nesta versão do curso, facilitando ainda mais o desenvolvimento e testes dos programas dos usuários.

Os computadores disponíveis nas salas de aula de Pirai têm algumas características bastante peculiares: telas menores, rodam o ambiente operacional MeeGo (uma versão Linux), utilizam o navegador Chromium. Mesmo sob estas condições, o ambiente DuinoBlocks rodou sem problemas nessas máquinas.

7 Conclusões e Trabalhos Futuros

À medida que os projetos de robótica são aplicados em sala de aula e difundidos pelos grupos de discussão na web (ex.: fóruns, blogs e wikis), novas demandas vão surgindo. Assim, novas tecnologias são criadas para suprir estas necessidades como, por exemplo, a solução da presente pesquisa cujo objetivo foi o de desenvolver um ambiente de programação visual com o propósito de facilitar o processo de programação do hardware Arduino por não especialistas.

O DuinoBlocks é um ambiente que estende os

recursos de programação do Arduino para permitir ao iniciante – preferencialmente professores e alunos da educação básica – programar, com facilidade, um dispositivo robótico. Do ponto de vista educacional espera-se que tal contribuição possa ajudar a enriquecer o processo de ensino-aprendizagem permitindo, por exemplo, que professores possam demonstrar, via atividades práticas e de forma mais contextualizada, conceitos teóricos ainda hoje apresentados apenas com o uso de textos didáticos e quadro negro.

O potencial do projeto Uca na Cuca e, particularmente do ambiente de programação visual DuinoBlocks, não é o de simplesmente apoiar o processo de ensino e aprendizagem, mas o de transformar o ambiente escolar em uma oficina de inventores, onde os estudantes possam trazer seus conhecimentos pessoais e interesses para a sala de aula, utilizando-os para o desenvolvimento de competências e habilidades necessárias aos cidadãos do século XXI, presentes nos Parâmetros Curriculares Nacionais.

O ambiente proposto é um apoio à experimentação, no computador, de atividades com RE. Sua versão atual já é capaz de rodar na nuvem, bem como na máquina do usuário com qualquer sistema operacional. Em ambas as situações o acesso ao ambiente é feito via navegador web.

Os testes com o DuinoBlocks vêm demonstrando que seus usuários – mesmo iniciantes – têm conseguido dominar o ambiente e desenvolver programas em tempos menores quando comparados com o uso da linguagem nativa do Arduino.

Como trabalhos futuros a serem implementados estão previstos:

- a finalização do módulo que permite conectar o ambiente a outras placas Arduino diferentes da UNO;
- o desenvolvimento de funcionalidades que permitam salvar programas na nuvem;
- o desenvolvimento de uma comunidade web em torno do ambiente DuinoBlocks voltada ao incentivo do compartilhamento de atividades didáticas e programas.
- conforme descrito na seção 4 do trabalho, a revisão dos trabalhos relacionados foi feita com o intuito de ser parte do processo de definição dos requisitos da ferramenta DuinoBlocks, procurando assim preencher lacunas eventualmente encontradas. O sucesso - alcançado pelo nosso produto em geral e dessa estratégia em particular - será avaliado por meio do seu uso com o público-alvo, formado mormente por professores e alunos do ensino fundamental.

Agradecimentos

Esta pesquisa foi parcialmente financiada pelo CNPq através do Edital CNPq/CAPES/SEED-MEC nº 76/2010 – PROUCA.

Referências

- [1] C. Schons, E. Primaz, G. A. P. Wirth. Introdução a Robótica Educativa na Instituição Escolar para alunos do Ensino Fundamental da disciplina de Língua Espanhola através das Novas Tecnologias de Aprendizagem. In: *I Workshop de Computação da Região Sul*, 2004.
- [2] S. R. Zilli. A Robótica Educacional no Ensino Fundamental: Perspectivas e Prática. Dissertação de Mestrado, Universidade Federal de Santa Catarina, 2004.
- [3] F. F. Sampaio, Elia M. F. Projeto Uca na Cuca: robótica educacional na sala de aula. 2011. Disponível em: <www.nce.ufrj.br/GINAPE/publicacoes/Projetos/Proj_UCAnaCUCA.pdf>. Acesso: dez. 2014.
- [4] R. M. Alves, A. L. C. Silva, M. de C. Pinto, F. F. Sampaio, M. F. Elia. Uso do hardware livre Arduino em ambientes de ensino-aprendizagem. In: *JAIE - Jornada de Atualização em Informática e Educação. CBIE - Congresso Brasileiro de Informática da Educação*. Rio de Janeiro, Brasil, 2012.
- [5] M. C. Pinto, M. F. Elia, F. F. Sampaio. Formação de professores em robótica educacional com Hardware Livre Arduino no contexto Um Computador por Aluno. In: *XVIII Workshop de Informática na Escola*. Rio de Janeiro, 2012.
- [6] H. A. Zanetti, A. L. D. Souza, J. V. D'Abreu, M. A. Borges. Uso de robótica e jogos digitais como sistema de apoio ao aprendizado. In: *JAIE - Jornada de Atualização em Informática na Educação*. Campinas, São Paulo, 2013.
- [7] D. Alimisis. Designing robotics-enhanced constructivist training for science and technology teachers: the TERECoP Project. In J. Luca & E. Weippl (Eds.), *Proceedings of World Conference on Educational Multimedia, Hypermedia and Telecommunications* 2008(pp. 288-293). Chesapeake, VA: AACE.
- [8] M. T. Chella. SimRobô: simulador para robótica com propósito educacional. In: *III Workshop de Robótica Educacional*, Fortaleza, 2012.
- [9] C. Fernandes. Um Simulador de Ambiente de Robótica Educacional em Plataforma Virtual. Dissertação de Mestrado, Universidade Federal do Rio Grande do Norte, 2013.
- [10] L. Victorino, M. F. Elia, A. Gomes, M. C. Pinto, C. Bastos. Laboratório Virtual de Atividades Didáticas – LabVad, In: *XV Workshop de Informática na Escola – WIE*. Bento Gonçalves, Rio Grande do Sul, 2009. Disponível em: <www.lbd.dcc.ufmg.br/colecoes/wie/2009/022.pdf>. Acesso: dez. 2014.
- [11] M. K. Da Cruz, W. Haetinger, F. Horn, D. V. De Carvalho, G. H. Araujo. Controle de Kit de Robótica através de Laboratório Remoto pela Internet: uma Aplicação para a Formação Docente e para a Educação Básica. In: *XX Simpósio Brasileiro de Informática na Educação*, Florianópolis, 2009.
- [12] M. B. Souza, J. F. M. Netto, M. A. S. Alencar, M. M. Silva. Arcabouço de um Ambiente Telerobótico Educacional Baseado em Sistemas Multiagentes. In: *XXII Simpósio Brasileiro de Informática na Educação*. Aracaju, 2011.
- [13] A. F. da Silva. RoboEduc: Uma Metodologia de Aprendizado com Robótica Educacional. Tese de doutorado, Universidade Federal do Rio Grande do Norte, 2009.
- [14] P. C. Ribeiro, C. B. Martins, F. C. Bernardini. Robótica como Ferramenta de Apoio ao Ensino de Disciplinas de Programação em Cursos de Computação e Engenharia. In: *XXII Simpósio Brasileiro de Informática na Educação*. Aracaju, 2011.
- [15] M. de C. Pinto. Aplicação de arquitetura pedagógica em curso de robótica educacional com hardware livre. Dissertação de Mestrado, UFRJ, Rio de Janeiro, 2011. Disponível em: <www.nce.ufrj.br/ginape/>. Acesso: dez. 2014.
- [16] M. Jacek. Some thoughts on robotics for education. *Proceeding of American Association of Artificial Intelligence Symposium on Robotics and Education*. Lund University. 2001.
- [17] L. C. Miranda, F. F. Sampaio, J. A. dos S. Borges. RoboFácil: Especificação e Implementação de um Kit de Robótica para a Realidade Educacional Brasileira. *Revista Brasileira de Informática na Educação*, Volume 18, Número 3, 2010.

- [18] P. Mendelson, T. R. G. Green, P. Brna. Programming languages in education: the search for an easy start. In: *Hoc, J., Green, T., Gilmore, D. & Samway, R. (eds) Psychology of Programming*, 175-200, London: Academic Press, 1990.
- [19] E. Pasternak. Visual Programming Pedagogies and Integrating Current Visual Programming Language Features. Master's Degree. Carnegie Mellon University. Robotics Institute, 2009. Disponível em: <www.ri.cmu.edu/pub_files/2009/8/Thesis-1.pdf>. Acesso: dez. 2014.
- [20] PROUCA. Programa Um Computador por Aluno. Disponível em: <www.bndes.gov.br/SiteBNDES/bndes/bndes_pt/Institucional/Apoio_Financeiro/Programas_e_Fundos/prouca.html>. Acesso: nov. 2013.
- [21] ARDUINO. Disponível em: <playground.arduino.cc/>. Acesso: dez. 2014.
- [22] Miranda L. C. RoboFácil: Especificação e Implementação de Artefatos de Hardware e Software de Baixo Custo para um Kit de Robótica Educacional. Dissertação de Mestrado, UFRJ, Rio de Janeiro, 2006. Disponível em: <www.nce.ufrj.br/ginape/>. Acesso: dez. 2014.
- [23] R. M. Alves. DuinoBlocks: Desenho e Implementação de um Ambiente de Programação Visual para Robótica Educacional. Dissertação de Mestrado, UFRJ, Rio de Janeiro, 2013.
- [24] LabVad portal. Disponível em: <146.164.3.24/ucanacuca/labvad/>. Acesso: dez. 2014.
- [25] A. P. Albuquerque, C. M. Melo, D. R. Cesar, D. Mill. Robótica Pedagógica Livre: Instrumento de Criação, Reflexão e Inclusão Sócio-digital. In: *XVIII Simpósio Brasileiro de Informática na Educação*. São Paulo, 2007.
- [26] D. A. M. Filho, P. C. Gonçalves. Robótica Educacional de Baixo Custo: Uma Realidade para as Escolas Brasileiras. In: *XXVIII Simpósio Brasileiro de Informática na Educação*. Belém do Pará, 2008.
- [27] F. L. Santos, F. M. S. Nascimento, R. M. S. Bezerra. REDUC: A Robótica Educacional como Abordagem de Baixo Custo para o Ensino de Computação em Cursos Técnicos e Tecnológicos. In: *XVI Workshop de Informática na Escola – WIE*. Belo Horizonte, 2010.
- [28] L. R. Sasahara, S.M. S. da Cruz. Hajime – Uma nova abordagem em robótica educacional. In: *XVIII Simpósio Brasileiro de Informática na Educação*. São Paulo, 2007.
- [29] D. Mellis. O hardware em código aberto. Entrevista para revista Info Exame, março, 2009. Disponível em: <info.abril.com.br/professional/tendencias/hardware-livre-leve-e-solto.shtml>. Acesso: dez. 2014.
- [30] Scratch – Site oficial. Disponível em <scratch.mit.edu>. Acesso: dez. 2014.
- [31] J. Crinnion. Evolutionary systems development: a practical guide to the use of prototyping within a structured systems methodology. Pitman Publishing, London, 1991.
- [32] S. Zhang, X. Chen, X. Huo. Cloud Computing Research and Development Trend. In: *Second International Conference on Future Networks*, 2010.
- [33] F. R. C. Souza, L. O. Moreira, J. C. Machado. Computação em Nuvem: Conceitos, Tecnologias, Aplicações e Desafios. Publicado no ERCEMAPI 2009. Versão revisada e estendida em Set. 2010. Capítulo 7. Disponível em: <www.ufpi.br/ercemapi/arquivos/file/minicurso/mc7.pdf>. Acesso: nov. 2013.
- [34] C. D. Hundhausen, S. Farley, J. L. Brown. Can Direct Manipulation Lower the Barriers to Programming and Promote Positive Transfer to Textual Programming? An Experimental Study. *Visual Languages and Human-Centric Computing*, 2006.
- [35] F. É. N de Mélo, R. R. M. da Cunha D. R. Scolaro, J. L. Campos. Do Scratch ao Arduino: Uma Proposta para o Ensino Introdutório de Programação para Cursos Superiores de Tecnologia, In: *XXXIX Congresso Brasileiro de Educação em Engenharia*, Blumenau, Brasil, 2011.
- [36] V. C. O. Aureliano, P. C. Tedesco. Avaliando o uso do Scratch como abordagem alternativa para o processo de ensino-aprendizagem de programação. In: *XXXII Congresso da Sociedade Brasileira de Computação*. Curitiba, Brasil, 2012.
- [37] L. O. Brandão, A. A. F. Brandão, R. de S. Ribeiro. iVProg – Uma Ferramenta de Programação Visual para o Ensino de Algoritmos. In: *Workshops do Congresso Brasileiro de Informática na Educação*. Rio de

Janeiro, 2012.

- [38] MEEGO. Disponível em: <www.metasys.com.br/index.php?option=com_content&view=article&id=417&Itemid=173&lang=pt>. Acesso: dez. 2014.
- [39] A. Barbero, B. Demo, F. Vaschetto. A Contribution to the Discussion on Informatics and Robotics in Secondary Schools. In: *Proceedings of the 2nd international conference on Robotics in Education*, RiE2011. Vienna. Italy, 2011.
- [40] A. Martins. Usando o Scratch para potencializar o pensamento criativo em crianças do ensino fundamental. Dissertação de Mestrado, Universidade de Passo Fundo, 2012.
- [41] P. D. Scaico, A. A. Lima, J. B. B. da Silva, S. Azevedo, L. F. Paiva, E. H. S. Raposo, Y. Alencar, J. P. Mendes. Programação no Ensino Médio: Uma Abordagem de Ensino Orientado ao Design com Scratch. In: *XVIII Workshop Sobre Informática na Escola - WIE*. Rio de Janeiro. 2012.
- [42] L. P. Flannery, E. R. Kazakoff, P. Bontá. Designing ScratchJr: Support for early childhood learning through computer programming. In: *Proceedings of the 12th International Conference on Interaction Design and Children*. ACM, New York, NY, USA, 2013.
- [43] E. R. Kazakoff, M. U. Bers. Designing New Technologies for Early Childhood: Results From the Initial Pilot Studies of ScratchJr. *Poster presented at SRCD Society for Research in Child Development*. Seattle, Washington. 18-20 April, 2013.
- [44] CLASSMATE. Disponível em: <www.instituicaodoseculo21.com.br/classmate.html>. Acesso: nov. 2013.
- [45] J. Maeda. *The Laws of Simplicity*. MIT Press, 2006.
- [46] L. Cohen, L. Manion, K. Morrison. *Research Methods in Education*. 5th ed. Taylor & Francis e-Library, 2005. Acesso: dez 2014.
- [47] M. W. van Someren, Y. F. Barnard, J. A. C. Sandberg. *The Think Aloud Method. A practical guide to modelling cognitive processes*. Academic Press, London. 1994.