DuinoBlocks: Um Ambiente de Programação Visual para Robótica Educacional[1]

Rafael Machado Alves, Fábio Ferrentini Sampaio, Marcos da Fonseca Elia

Instituto Tércio Pacitti de Aplicações e Pesquisas Computacionais (NCE)
Universidade Federal do Rio de Janeiro (UFRJ) – Cx Postal 2324 – Rio de Janeiro – RJ

rafamachadoalves@ufrj.br, {ffs,melia}@nce.ufrj.br

Abstract. *This paper presents a visual programming environment called DuinoBlocks developed for the Arduino robotics hardware. The 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. Tests conducted with the environment have shown that teachers feel more comfortable in working with this environment compared with the textual language of the Arduino (Wiring).*

Resumo. *Este trabalho apresenta o ambiente de programação visual DuinoBlocks desenvolvido para o hardware de robótica Arduino. O DuinoBlocks é capaz de rodar em máquinas com diferentes sistemas operacionais, inclusive nos computadores pessoais do Programa PROUCA do Governo Federal. Os testes realizados com o ambiente têm demonstrado que professores se sentem mais confortáveis em trabalhar com esse ambiente em comparação com a linguagem textual do Arduino (Wiring).*

1. Introdução

No futuro, os objetos do dia-a-dia terão sensores que serão capazes de se comunicar através da internet com o intuito de facilitar a vida de seus usuários na execução das tarefas do cotidiano. Esse novo paradigma de acoplar a internet a diferentes produtos é denominado *Internet das Coisas* [Atzori et al. 2010]. Em se tratando de cidades, uma análise dos dados coletados pelos sensores serve para tomar melhores decisões, coordenar recursos, antecipar problemas e resolvê-los de forma proativa (p. ex. manipular atuadores automaticamente) [da Silva et al. 2012]. No Brasil, onde o assunto ainda é novidade em termos práticos, os desafios para o futuro são a busca de soluções inteligentes para a cidade lutar contra problemas recorrentes como o trânsito, diminuindo horas que são perdidas em engarrafamentos, e as enchentes, evitando tragédias causadas pelas chuvas.

Do ponto de vista do nosso grupo de pesquisa em tecnologias no ensino, a concepção e implementação - no futuro - de projetos urbanísticos inovadores para *Cidades Inteligentes*, implica, necessariamente, o desenvolvimento de competências e habilidades dos nossos alunos para trabalhar com inovações, robótica e outros recursos tecnológicos.

[1] Projeto Financiado pelo CNPq No. 550.400/2011-7

Na atividade com *Robótica Educacional* (RE) o esforço do educando é empregado na criação de soluções oriundas do cotidiano dos alunos, sejam essas compostas por hardware e/ou software. As soluções visam à resolução de um problema proposto, podendo o mesmo ser real, promovendo assim a transformação do ambiente escolar em uma oficina de inventores.

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, desconectado 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 [Mendelson et al. 1990]. 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 (2009), 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. Trata especificamente do desenho e implementação de um ambiente de programação visual denominado DuinoBlocks, baseado na plataforma robótica Arduino (2013). Desta forma, caminha ao encontro da proposta do quarto Grande Desafio da Sociedade Brasileira de Computação [Baranauskas e Souza 2006], o qual aponta para a construção de sistemas que favoreçam o uso de tecnologias na educação por professores, melhorando a qualidade do ensino e ampliando o acesso participativo e universal do cidadão brasileiro ao conhecimento.

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 é apresentado o contexto e os principais requisitos para o desenvolvimento do DuinoBlocks; 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

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 (2007), Filho (2008), Miranda (2010), Santos (2010) e Sasahara (2007) 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.

2.1 O Projeto Arduino

O projeto Arduino prevê uma plataforma de hardware e software abertos de fácil utilização, acessível não somente à especialistas na área de eletrônica, mas também hobbystas ou qualquer pessoa interessada na criação de objetos ou ambientes interativos. Uma vez programado, o Arduino controla uma gama de componentes eletrônicos (motores, *displays*, etc.) com base em instruções recebidas através de sensores como os de luminosidade e temperatura.

O ambiente de programação nativo do Arduino – *Wiring* – é composto por uma linguagem baseada em comandos com características semelhantes à linguagem C e um conjunto de opções que auxiliam na depuração e comunicação (*upload*) com a placa de hardware Arduino (2013).

2.2 O Programa PROUCA

O Programa Um Computador por Aluno (PROUCA) é um projeto do Governo Federal com 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 fazem uma chamada via Edital para que grupos de pesquisa no Brasil apresentem propostas que contemplem o uso dos laptops adquiridos pelas escolas parceiras do PROUCA [Sampaio e Elia 2012]. O projeto Uca na Cuca de pesquisa científica, pesquisa tecnológica e inovação pedagógica na área de RE é um dos selecionados pelo referido Edital.

O projeto Uca na Cuca prevê cinco metas, dentre elas a criação de um novo ambiente de desenvolvimento de programas (IDE) para o kit Arduino, incorporando uma VPL mais amigável. Esse ambiente é denominado DuinoBlocks e seus principais requisitos e características estão descritos nas Seções 4 e 5.

3. Trabalhos Relacionados

Foi realizada uma análise acerca dos softwares disponíveis para programação gráfica de robôs, com o intuito de comparar as suas funcionalidades equivalentes - identificando aquelas que melhor se ajustam ao contexto da proposta - bem como apontar a ausência de funcionalidades essenciais.

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. Quanto à comunicação, os softwares de programação gráfica para o Arduino se classificam desta forma: **Ambientes autônomos**: Minibloq, Ardubloq e ModKit. **Ambientes não autônomos**: Squeak Etoys, S4A, Firefly.

A seguir serão comentados os softwares que mais influenciaram o desenvolvimento do DuinoBlocks.

MINIBLOQ[2]: 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[3]: 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 (W*iring*), possui suporte multilíngue e é codificado em Java. Os blocos são representados por figuras (como no Minibloq).

S4A[4]**:** Uma adaptação para a robótica do Scratch, um dos ambientes de programação gráfica mais populares desenvolvido no MIT Media Lab (2013). Outra adaptação do Scracth, porém não para robótica, é o BYOB (*Build Your Own Blocks*) que possui uma poderosa ferramenta para o usuário criar seus próprios blocos. Os blocos são representados por texto. É codificado em Smalltalk.

MODKIT[5]: Ambiente de programação para microcontroladores que permite programar o hardware Arduino e outros compatíveis. Os blocos são 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. 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.

Vale ressaltar que, com exceção do ModKit, nenhum dos ambientes tratados acima salvam seus projetos na nuvem.

4. Requisitos do Sistema

O levantamento de requisitos teve inicio na revisão da literatura e análise dos softwares que utilizam 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, as considerações feitas pelos professores que

[2] Site do MiniBloq: http://blog.minibloq.org/
[3] Site do ArduBlock: http://blog.ardublock.com/
[4] Site do S4A: http://seaside.citilab.eu/
[5] Site do ModKit: http://www.modk.it/

participaram dos nossos cursos de REBC (ocorridos nas escolas parceiras do Projeto), também foram de grande importância na identificação das necessidades do público alvo.

A seguir são detalhados os dois principais requisitos do ambiente DuinoBlocks:

Ambiente multiplataforma: O primeiro 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[6]. A alternativa econômica e em linha com as tendências atuais no desenvolvimento de software [Shuai 2010], foi a de implementar um ambiente multiplataforma que rodasse na nuvem.

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 nos navegadores das máquinas do PROUCA.

O requisito definido acima traz no seu bojo três outros importantes aspectos positivos: 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; reduz a necessidade de espaço de memória ou disco nos laptops, uma vez que os projetos podem ser salvos na nuvem; e introduz facilidades para o compartilhamento e colaboração dos projetos desenvolvidos pelos alunos e professores.

Linguagem de programação amigável: Professores e alunos hoje já estão relativamente bem familiarizados com interfaces de manipulação direta como o Windows e distribuições Linux. 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 amigável para o hardware Arduino. Desta forma, ao invés de pensarmos num ambiente tradicional (textual) de programação onde o usuário deve saber de antemão a sintaxe exata dos comandos de um programa em construção, pensou-se na utilização de blocos gráficos representando os comandos da linguagem e com um formato (visual) tal que seriam capazes de se conectar somente a alguns outros blocos de comandos de forma a evitar problemas de análise léxica e sintática.

5. O Ambiente DuinoBlocks

As funcionalidades do ambiente DuinoBlocks foram idealizadas levando-se em conta as necessidades relatadas na Seção 4 (Requisitos do Sistema) e outros ambientes de programação visual como o Scratch (2013) Diferentes aspectos de usabilidade do ambiente foram projetados segundo referências as Leis da Simplicidade proposta por Maeda (2006).

[6] Os laptops das escolas do PROUCA, parceiras do projeto Uca na Cuca, possuem o sistema operacional Meego - uma distribuição Linux. Por sua vez, os computadores que os alunos por ventura tenham em suas residências têm 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.

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 Crinnion (1991).

A Figura 1 mostra o layout do ambiente DuinoBlocks, com um exemplo de algoritmo (Pisca Led) bastante conhecido na RE. O layout é dividido em 5 painéis: o rodapé (contém botões que controlam a disposição dos painéis); o cabeçalho (barra de ferramentas com botões que realizam ações no software); o esquerdo (contém uma lista de blocos separados por categorias e subcategorias); o central (área de construção do algoritmo); e o direito (contém a tradução textual do algoritmo em Wiring).

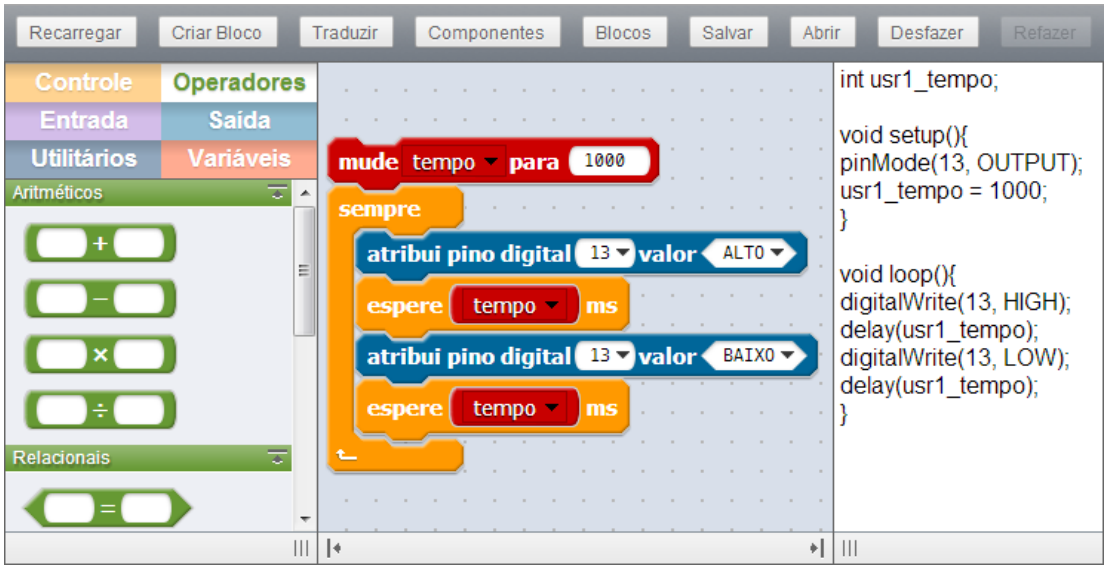

Figura 1. Layout do Ambiente DuinoBlocks.

A seguir é detalhado como são abstraídas as principais estruturas lógicas de programação do ambiente DuinoBlocks.

- *Categoria dos blocos.* Os blocos são agrupados por categorias. Cada categoria é indicada por uma cor. Algumas categorias possuem subcategorias. Atualmente o DuinoBlocks possui seis categorias: *Controle* (cor abóbora), *Operadores* (cor verde), *Entrada* (cor roxa), *Saída* (cor azul claro), *Utilitários* (cor azul escuro) e *Variáveis* (cor vermelha).

- *Tipos de bloco*. A forma do bloco indica os possíveis encaixes corretos sintaticamente, determinando também o seu tipo.

Os blocos passíveis de empilhamento (*Blocos de Pilha*) possuem um entalhe acima do seu nome e uma saliência logo abaixo. Os blocos "sempre" e "atribui pino digital" na Figura 1, são exemplos desse tipo.

Os blocos utilizados para serem encaixados dentro de outros blocos (*Blocos de Retorno*) podem ser de três subtipos diferentes: o *Numérico* possui bordas arredondadas e retorna um número (ex. blocos aritméticos no painel esquerdo do layout na Figura 1); o *Lógico* possui bordas pontiagudas e retorna um valor lógico (V ou F); e o *Alfanumérico* possui bordas quadradas e retorna caracteres.

- *Entrada de dados dos blocos*: alguns blocos possuem entradas de dados. Estas podem ser preenchidas encaixando outros blocos, via teclado, ou escolhendo uma opção em um

combobox (quando o conjunto de entradas possíveis é conhecido). O seu formato é diferente para cada tipo de entrada permitido. (ex. entradas nos blocos azuis na Figura 1)

- *Criação de variáveis*: para criar uma variável o usuário escolhe, na categoria *Variáveis*, seu nome (sem se preocupar com regras de nomeação) e seu tipo (lógica, numérica ou alfanumérica). Ao criar uma variável o ambiente fornece novos blocos: um para obter o seu valor e outro para alterá-lo (ex. blocos vermelhos manipulando a variável "tempo" na Figura 1).

- *Criação de blocos pelo próprio usuário*: o ambiente permite a criação de procedimentos (sub-rotinas) pelo usuário. Para tal o usuário escolhe um nome para o procedimento, define sua categoria e tipo e edita o procedimento como se tivesse criando um novo programa no DuinoBlocks. O ambiente, de forma automática, disponibiliza-o para uso no painel esquerdo do layout.

- *Recursividade*: para realizar uma recursão, cria-se um novo bloco (procedimento) e dentro do mesmo faz-se uma chamada ao procedimento criado.

O sistema possui outros módulos além do principal "Blocos" de programação gráfica: O módulo "Componentes", que apresenta um grau de abstração maior ao especializar comandos para determinados componentes (LEDs, buzina, servo, motor, display de 7 segmentos, LCD, chave de contato, potenciômetro, LDR, termistor e joystick. - Figura 2); e o módulo "Editor de Bloco", para criação de procedimentos (Figura 3).

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 "Hardwares".

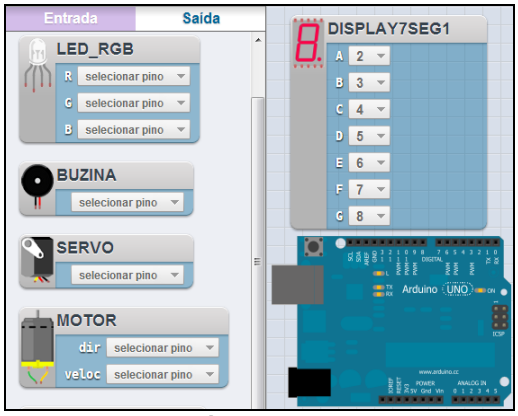

Figura 2. Módulo Componentes

Figura 3. Módulo Editor de Bloco

5.1. Arquitetura e Implementação do Sistema

A escolha da tecnologia de desenvolvimento do DuinoBlocks levou em consideração o esforço necessário para a programação da sua interface. Desta forma, optou-se pela biblioteca Pyjamas[7] (escrita em Python) e o IDE Eclipse. Outra vantagem de tais escolhas foi a possibilidade de se gerar código javascript de forma automática, permitindo a sua execução em qualquer navegador web ou rodar sem alterações como uma aplicação desktop.

[7] Site do *Pyjamas*: pyjs.org

Cada bloco da VPL é uma abstração de um comando que equivale a instruções textuais de baixo nível. A Figura 4 mostra a estrutura HTML e a formatação CSS básicas para a construção de um bloco (baseado no ModKit).

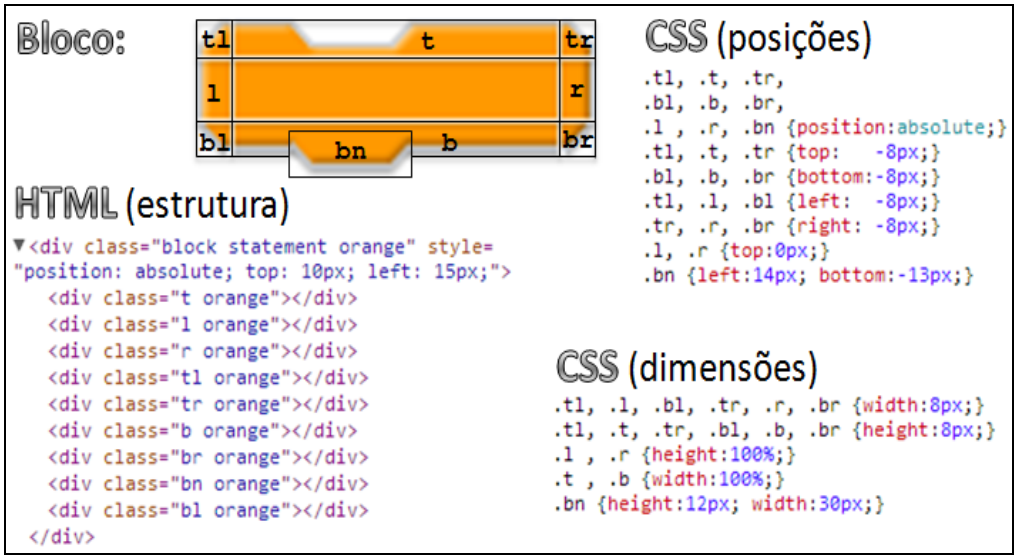

Figura 4. Construção básica de um bloco

A Figura 5 mostra o diagrama de classes resumido do projeto. O pacote *block* trata das funcionalidades dos diferentes tipos de blocos. O pacote *arguments* possui as classes para o gerenciamento dos diferentes tipos de entrada de dados. Na biblioteca *Pyjamas* é mostrado as principais classes e seus métodos para manipular um objeto arrastável. A classe *BlockList* trata do painel esquerdo da interface - que contém as categorias com as listas de blocos - e a classe *BlocksPad* do painel central – responsável pela criação/gerenciamento de algoritmos (programação feita pelos usuários).

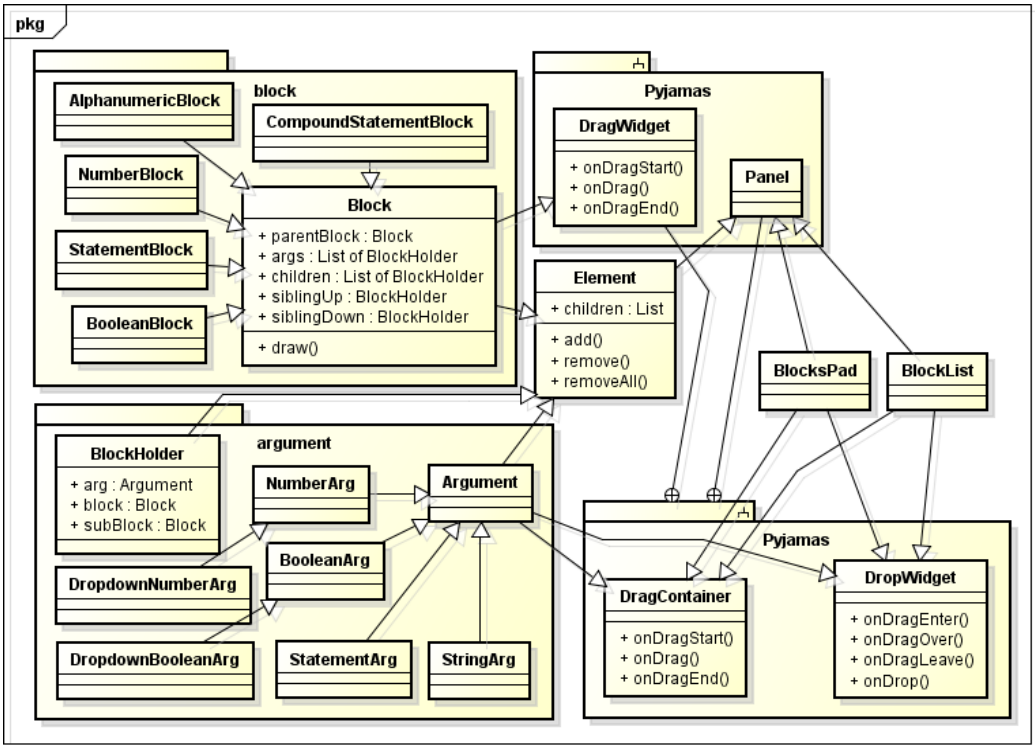

Figura 5. Diagrama de Classes Resumido do DuinoBlocks.

Por se tratar de um ambiente com VPL, entendemos que não é coerente o mecanismo de tratamento de erros e o fornecimento de ajuda do ambiente serem apresentados na forma textual. Por esse motivo o DuinoBlocks possui os módulos de "Tratamento de Erro" e de "Ajuda" também na forma visual.

6. Avaliação do DuinoBlocks

O projeto Uca na Cuca vem promovendo no município de Piraí cursos de capacitação de professores do ensino fundamental e médio intitulado "Formação em Robótica Educacional com Hardware Livre". Tais cursos têm caráter teórico-prático e a aplicação dos mesmos forneceu importantes subsídios para a especificação do ambiente DuinoBlocks (apresentadas na Seção 4).

Vale ressaltar que os cursos citados também demonstraram a viabilidade do trabalho com REBC utilizando os computadores do PROUCA. Ao mesmo tempo percebeu-se um grande interesse por parte dos professores participantes e de alunos do ensino fundamental que trabalharam com os mesmos.

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 e a se organizar em duplas. Cada dupla utilizou um kit de robótica. 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 num diário de bordo, além de responderem a questionários impressos e questionamentos orais [Cohen 2005].

Uma vez que o objetivo da equipe proponente da oficina era o de testar o ambiente DuinoBlocks, não foi dada grande ênfase na montagem física dos experimentos, que por vezes foram entregues semi-prontos. Os algoritmos, por sua vez, foram, no início, elaborados de forma conjunta (professor e cursistas) e à medida que os participantes ganhavam experiência, os desafios de programação eram propostos sem ajuda dos instrutores.

6.1. Resultados da Avaliação

Partimos de um processo inicialmente diretivo, onde o algoritmo (projetado com *datashow*) era explicado e ao mesmo tempo construído junto com os participantes que o

copiavam. Depois exploratório, onde eram solicitadas alterações no algoritmo construído. E por último, já no final do curso, para um processo concreto de plena programação. É certo que os participantes ainda fazem programações singelas, ainda sim os resultados mostram que os participantes saíram de um processo de imitação para o de elaboração mental por meio das primeiras programações de seus protótipos.

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 *drag-and-drop* 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:* [Someren et al 1994]) 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.

7. Conclusões e Direções Futuras

O DuinoBlocks é um ambiente que estende os recursos de programação do Arduino para permitir ao iniciante – preferencialmente professor e alunos do ensino básico - programar um dispositivo robótico a fim de enriquecer o processo de ensino-aprendizagem.

A versão atual do ambiente 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.

Atualmente os esforços da equipe de desenvolvimento têm se concentrado na organização do layout, e na correção de pequenos problemas de usabilidade detectados durante os experimentos.

Como trabalhos futuros a serem implementados ainda no 1º. Semestre de 2013 está previsto:

- o desenvolvimento de funcionalidades que permitam salvar programas na nuvem;

- a implementação de uma funcionalidade que permita o usuário carregar o programa escrito em DuinoBlocks diretamente no hardware Arduino, sem que seja necessário copiar o código textual traduzido pelo DuinoBlocks para colar no IDE Arduino.

- o desenvolvimento de uma comunidade web em torno do ambiente DuinoBlocks voltada ao incentivo do compartilhamento de programas.

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.

Referências

Atzori, L. Iera, A. Morabito, G. (2010) "The Internet of Things: A survey". Elsevier Computer Networks.

Albuquerque, A. P.; Melo, C. M.; César, D. R. e Mill, D. (2007) "Robótica Pedagógica Livre: Instrumento de Criação, Reflexão e Inclusão Sócio-digital". Em *XVIII Simpósio Brasileiro de Informática na Educação. São Paulo.*

Arduino. Em: <arduino.cc>. Acesso em: 29 de março de 2013.

Baranauskas, M.C.C. e Souza, C.S. (2006) "Desafio nº 4: Acesso Participativo e Universal do Cidadão Brasileiro ao Conhecimento". In: *Computação Brasil, ano VII*.

Cohen, L. Manion, L. Morrison, K. (2005) "Research Methods in Education". 5th ed. Taylor & Francis e-Library.

Crinnion, J. (1991). "Evolutionary Systems Development, a practical guide to the use of prototyping within a structured systems methodology". Plenum Press, New York.

Filho, D. A. M. e Gonçalves, P. C. (2008) "Robótica Educacional de Baixo Custo: Uma Realidade para as Escolas Brasileiras". Em *XXVIII Simpósio Brasileiro de Informática na Educação. Belém do Pará.*

Freire, P. "Pedagogia do oprimido". Rio de Janeiro: Paz e Terra, 1970.

Maeda, J. (2006) "The Laws of Simplicity", MIT Press.

Mendelson, P.; Green, T. R. G.; Brna, P. (1990) "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*.

Miranda, L. C.; Sampaio, F. F. e Borges, J. A. dos S. (2010) "RoboFácil: Especificação e Implementação de um Kit de Robótica para a Realidade Educacional Brasileira". Em *Revista Brasileira de Informática na Educação, Volume 18, Número 3.*

MIT Media Lab. Laboratório de Mídia do Instituto de Tecnologia de Massachusetts – EUA. Em: <http://www.media.mit.edu/> Acesso em: 11 de abril de 2013.

Pasternak, E. “Visual Programming Pedagogies and Integrating Current Visual Programming Language Features”. Carnegie Mellon University Robotics Institute. Thesis Master's Degree. 2009. Disponível em: <http://www.ri.cmu.edu/pub_files/2009/8/Thesis-1.pdf>. Acesso em: 11 de abril de 2013.

Sampaio e Elia (2012). “Projeto um computador por aluno: pesquisas e perspectivas”. Disponível em: <http://www.nce.ufrj.br/ginape/livro-prouca>. Acesso em: 11 de abril de 2013.

Santos, F. L., Nascimento, F. M. S., Bezerra, R. M. S. (2010) “REDUC: A Robótica Educacional como Abordagem de Baixo Custo para o Ensino de Computação em Cursos Técnicos e Tecnológicos” Em *XVI Workshop Sobre Informática na Escola – WIE. Belo Horizonte.*

Sasahara, L. R. e Cruz, S. M. S. (2007) “Hajime – Uma nova abordagem em robótica educacional”. Em *XVIII Simpósio Brasileiro de Informática na Educação. São Paulo.*

Scratch. Em: <http://scratch.mit.edu/>. Acesso em: 11 Abr 2013.

Shuai, Z., et al (2010). “Cloud computing research and development trend”. IEEE Computer Society.

Someren, M. W. van, Barnard, Y. F., Sandberg, J. A. C. (1994) “The Think Aloud Method. A practical guide to modelling cognitive processes”. Academic Press, London. Disponível em: <http://staff.science.uva.nl/~maarten/Think-aloud-method.pdf>. Acesso em: 11 Abr 2013.

da Silva, V. H. S. F., Alvaro, A. (2012) “Uma Plataforma para Cidades Inteligentes baseada na Internet das Coisas”, In: *VIII Simpósio Brasileiro de Sistemas de Informação (SBSI)*, São Paulo, SP.