Universidade Federal do Rio de Janeiro

RAFAEL MACHADO ALVES

DISSERTAÇÃO DE MESTRADO

DUINOBLOCKS: DESENHO E IMPLEMENTAÇÃO DE UM AMBIENTE DE PROGRAMAÇÃO VISUAL PARA ROBÓTICA EDUCACIONAL

RIO DE JANEIRO
2013

Instituto de Matemática

Instituto Tércio Pacitti de Aplicações
e Pesquisas Computacionais

RAFAEL MACHADO ALVES

DUINOBLOCKS: DESENHO E IMPLEMENTAÇÃO DE UM AMBIENTE DE PROGRAMAÇÃO VISUAL PARA ROBÓTICA EDUCACIONAL

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Informática, do Instituto de Matemática e Instituto Tércio Pacitti de Aplicações e Pesquisas Computacionais, Universidade Federal do Rio de Janeiro, como requisito parcial para obtenção do título de Mestre em Informática.

Orientador: Fábio Ferrentini Sampaio, Ph.D.

RIO DE JANEIRO
2013

A474 Alves, Rafael Machado.
DUINOBLOCKS: desenho e implementação de um ambiente de programação visual para robótica educacional. / Rafael Machado Alves.-- 2013.
112 f.: il.

Dissertação (Mestrado em Informática) – Universidade Federal do Rio de Janeiro, Instituto de Matemática, Instituto Tércio Pacitti de Aplicações e Pesquisas Computacionais, Programa de Pós-Graduação em Informática, Rio de Janeiro, 2013.

Orientador: Fábio Ferrentini Sampaio

1. Robótica Educacional. 2. Linguagem de Programação Visual. 3. Computador por Aluno. - Teses. I. Sampaio, Fábio Ferrentini, (Orient.). II. Universidade Federal do Rio de Janeiro, Instituto de Matemática, Instituto Tércio Pacitti de Aplicações e Pesquisas Computacionais, Programa de Pós-graduação em Informática. III. Título

CDD.

RAFAEL MACHADO ALVES

DUINOBLOCKS: DESENHO E IMPLEMENTAÇÃO DE UM AMBIENTE DE PROGRAMAÇÃO VISUAL PARA ROBÓTICA EDUCACIONAL

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Informática, do Instituto de Matemática e Instituto Tércio Pacitti de Aplicações e Pesquisas Computacionais, Universidade Federal do Rio de Janeiro, como requisito parcial para obtenção do título de Mestre em Informática.

Aprovada em: Rio de Janeiro, 20 de dezembro de 2013.

_____

Prof. Fábio Ferrentini Sampaio, Ph.D., NCE e PPGI/UFRJ
(Orientador)

_____

Prof. Marcos da Fonseca Elia, Ph.D., NCE e PPGI/UFRJ

_____

Prof. Carlo Emmanuel Tolla de Oliveira, Ph.D., NCE/UFRJ

_____

Prof. Leonardo Cunha de Miranda, D.Sc., DIMAp/UFRN

DEDICATÓRIA

À minha família e amigos.

AGRADECIMENTOS

Primeiramente agradeço a Deus pelas oportunidades proporcionadas e por guiar meus passos nesta jornada tão importante em minha vida.

Ao meu orientador, o Prof. Ph.D. Fábio Ferrentini Sampaio pelas orientações esclarecedoras, norteando a minha trajetória acadêmica durante o mestrado.

Aos membros das bancas de Qualificação e Acompanhamento, Prof. Ph.D. Marcos da Fonseca Elia e Prof. Ph.D Carlo Emmanuel Tolla de Oliveira pelas valiosas sugestões. E ao Prof. D.Sc. Leonardo Cunha de Miranda por disponibilizar espaço em sua agenda para participação na Banca de Defesa deste trabalho de pesquisa.

Aos colegas do Projeto Uca na Cuca, Marcos Castro, Serafim Brandão, Armando Luiz, Cesar Bastos e Rodrigo Guedes pelo apoio e colaboração.

Aos colegas de graduação e trabalho, os analistas de sistemas Anderson Resende e Daniel Gustavo pela flexibilização de horário e companheirismo. E ao colega de graduação, o designer gráfico Elias Sant'ana pela contribuição na criação de imagens.

Agradeço à minha mãe Penha Machado, ao meu pai Hélcio Alves e à minha futura esposa Walnéa Alves, o apoio de vocês foi e está sendo muito importante para mim.

A todos da minha família que de alguma forma me apoiaram durante esses anos e pela paciência em tolerar a minha ausência.

E a todos que ajudaram direta e indiretamente para a realização deste trabalho.

“Nunca ore suplicando cargas mais leves e sim ombros mais fortes.”

(Phillips Brooks)

RESUMO

ALVES, Rafael Machado. DUINOBLOCKS: desenho e implementação de um ambiente de programação visual para robótica educacional. 2013. 112 f. Dissertação (Mestrado em Informática) – PPGI, Instituto de Matemática, Instituto Tércio Pacitti de Aplicações e Pesquisas Computacionais, Universidade Federal do Rio de Janeiro, Rio de Janeiro, 2013.

O presente trabalho tem como objetivo a proposição e o desenvolvimento de um ambiente com linguagem de programação visual que permita 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. Para alcançar tais objetivos, foi realizado um estudo sobre o mecanismo de elaboração de algoritmos em ambientes de programação visual, sobretudo, os voltados para o hardware Arduino. Além disso, foram realizadas pesquisas de campo com o público-alvo durante a realização de cursos de robótica educacional. Tais estudos proporcionaram um maior entendimento das necessidades dos usuários, viabilizando o desenho e a implementação do ambiente denominado DuinoBlocks. Este ambiente é capaz de rodar em máquinas com diferentes sistemas operacionais, inclusive nos computadores pessoais do PROUCA. 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 padrão do Arduino (Wiring).

Palavras-chave: Robótica Educacional, Linguagem de Programação Visual, Um Computador por Aluno.

ABSTRACT

ALVES, Rafael Machado. DUINOBLOCKS: desenho e implementação de um ambiente de programação visual para robótica educacional. 2013. 112 f. Dissertação (Mestrado em Informática) – PPGI, Instituto de Matemática, Instituto Tércio Pacitti de Aplicações e Pesquisas Computacionais, Universidade Federal do Rio de Janeiro, Rio de Janeiro, 2013.

This research work presents the proposal and the development of a computational environment with visual programming language that enables beginners to program the Arduino robotic device. These users are, preferentially, teachers and students from brazilian public schools partners of the Programa Um Computador por Aluno (PROUCA) of the Federal Government. To reach such objectives, a study about the development of algorithms for visual programming environments was conducted. In addition, field surveys were conducted with the target audience during courses in educational robotics. Such studies provided a better understanding of users' needs, enabling the design and implementation of the environment called DuinoBlocks. This environment is able to run on machines with different operating systems, including personal computers of the PROUCA. Tests conducted with the environment have demonstrated that teachers feel more comfortable working with this environment in comparison with the textual language of the Arduino (Wiring).

Keywords: Educational Robotics, Visual Programming Language, One Laptop per Child Brazil.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE QUADROS

LISTA DE ABREVIATURAS E SIGLAS

ACE	Application, Creation and Evaluation
BAT	Batch
bps	bits por segundo
CAPES	Coordenação de Aperfeiçoamento de Pessoal de Nível Superior
CNPq	Conselho Nacional de Desenvolvimento Científico e Tecnológico
CSBC	Congresso da Sociedade Brasileira de Computação
CSS	Cascading Style Sheets
DBK	DuinoBlocks file extension
DIMAp	Departamento de Informática e Matemática Aplicada
EAD	Educação a Distância
GPL	General Public License
GTE	Guia de Tecnologias Educacionais
HTML	HyperText Markup Language
IDE	Integrated Development Environment
IHC	Interação Humano-Computador
NCE	Instituto Tércio Pacitti de Aplicações e Pesquisas Computacionais
JSON	JavaScript Object Notation
LabVad	Laboratório Virtual de Atividades Didáticas
LCD	Liquid Crystal Display
LDR	Light Dependent Resistor
LED	Light Emitting Diode
MEC	Ministério da Educação
MIT	Massachusetts Institute of Technology

MRDS	Microsoft Robotics Developer Studio
OBR	Olimpíada Brasileira de Robótica
OLPC	One Laptop per Child
PCN	Parâmetros Curriculares Nacionais
PPGI	Programa de Pós-Graduação em Informática
PROUCA	Programa Um Computador por Aluno
RE	Robótica Educacional
REBC	Robótica Educacional de Baixo Custo
RPM	Red Hat Package Manager
S4A	Scratch for Arduino
SBA	Sociedade Brasileira de Automática
SBC	Sociedade Brasileira de Computação
SEED	Secretaria de Ensino a Distância
SEMIS	Seminário Integrado de Software e Hardware
SH	Shell Script
TI	Tecnologia da Informação
TIAE	Tecnologias da Informação Aplicadas à Educação
TIC	Tecnologia da Informação e Comunicação
UCA	Um Computador por Aluno
UFRJ	Universidade Federal do Rio de Janeiro
UFRN	Universidade Federal do Rio Grande do Norte
URL	Uniform Resource Locator
USB	Universal Serial Bus
V	Volt
VPL	Visual Programming Language
ZPD	Zona Proximal de Desenvolvimento

SUMÁRIO

1 INTRODUÇÃO

Este capítulo apresenta uma visão geral da utilização da robótica em sala de aula (Seção 1.1) e as questões relacionadas às barreiras iniciais da programação em robótica educacional dos kits robóticos acessíveis a professores e alunos (Seção 1.2). Apresenta ainda a relevância do presente trabalho (Seção 1.3), seguido dos seus objetivos, motivações e justificativas (Seção 1.4). Encerra-se apresentando a organização da dissertação (Seção 1.5).

1.1 Contexto Geral

Com o rápido avanço dos novos recursos tecnológicos observado nos últimos anos, a sociedade atual, cada vez mais conectada, encontra-se envolta por Tecnologias da Informação e Comunicação (TIC), inclusive no meio educacional.

O uso da tecnologia 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*. (2004), 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, é o esmiuçar da teoria através da prática. Neste processo a RE desenvolve competências como: raciocínio lógico, formulação e teste de hipóteses, habilidades manuais e estéticas, investigação e compreensão, resolução de problemas por meio de erros e acertos, aplicação das teorias formuladas a atividades concretas, utilização da criatividade em diferentes situações e capacidade crítica [ZILLI, 2004].

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, diferentes trabalhos são direcionados à aplicação da robótica em sala de aula [SAMPAIO e ELIA 2011; ALVES *et al*., 2012; PINTO *et al*., 2012;

ZANETTI *et al*., 2012]. Há também uma tendência promissora a partir do uso de Simuladores Virtuais [CHELLA, 2012; FERNANDES, 2013] e Laboratórios Remotos [VICTORINO *et al*., 2009; da CRUZ *et al*., 2009; SOUZA *et al*., 2011; SAMPAIO e ELIA 2011] 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 [da SILVA, 2009; RIBEIRO *et al*., 2011; PINTO, 2011] em contextos formais e não formais de ensino. Entretanto, conforme Miranda e outros (2010) ainda existem poucos trabalhos preocupados em apresentar alternativas às barreiras iniciais da programação em RE.

1.2 Caracterização do Problema

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 Robótica Educacional.

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. Elas reduzem a carga cognitiva sobre os estudantes que aprendem sua primeira linguagem de programação [PASTERNAK, 2009].

Neste sentido, este trabalho procura diminuir às barreiras iniciais da programação em RE no contexto nacional através de um ambiente com VPL. Desta forma, abre-se a possibilidade de criação de uma gama de projetos educacionais, atualmente inviabilizados pela ausência de ferramentas adequadas ao contexto,

promovendo assim o desenvolvimento de competências e habilidades dos nossos alunos para trabalhar com inovações, robótica e outros recursos tecnológicos necessários à formação dos cidadãos do século XXI.

1.3 Relevância

O MEC divulga desde 2008 os benefícios da RE por meio do Guia das Tecnologias Educacionais (GTE). O documento tem como objetivo selecionar e pré-qualificar produtos e propostas com vistas a promover a melhoria da qualidade da educação básica [GTE, 2013]. No guia de 2011/2012 [COGETEC, 2011] são encontrados pelo menos três tópicos envolvendo a robótica. No primeiro deles, é citado a empresa Brink Robótica [BRINK MOBIL, 2013], fabricante de kits robóticos produzidos para cada nível escolar de acordo com os Parâmetros Curriculares Nacionais (PCN). No segundo, encontra-se o Projeto de Alfabetização Tecnológica [PETe, 2013] que se fundamenta no uso da robótica para desenvolver um programa de formação pautado na exploração conceitual de conteúdos curriculares. Em um terceiro, são apresentadas “Soluções Tecnológicas para Robótica Educacional Utilizando Materiais Recicláveis e Sucata” onde os componentes eletrônicos são adquiridos das mais diversas formas.

A robótica educacional também é tratada em competições de caráter nacional e internacional. Dentre esses eventos encontra-se a Olimpíada Brasileira de Robótica [OBR, 2013], evento nacional realizado anualmente com apoio das sociedades científicas SBC (Sociedade Brasileira de Computação) e SBA (Sociedade Brasileira de Automática). A OBR utiliza a temática da robótica, tradicionalmente de grande aceitação junto aos jovens, para estimulá-los às carreiras científico-tecnológicas, identificar jovens talentosos e promover debates e atualizações no processo de ensino-aprendizagem brasileiro.

Este trabalho oferece aos professores novas oportunidades de exploração da robótica educacional como elemento motivador da aprendizagem, assim como o GTE e a OBR. Além disso, esta proposta tira vantagem dos dispositivos de baixo

custo que hoje facilitam a aquisição de componentes de hardware por professores e escolas, no momento em que o Governo Federal vem estimulando a utilização de novas propostas de ensinar e aprender com tecnologias através do PROUCA (2013).

Desta forma, caminha ao encontro da proposta do quarto Grande Desafio da Sociedade Brasileira de Computação [BARANAUSKAS e SOUZA, 2006] que 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.

1.4 Motivações, Objetivos e Justificativas

O desenvolvimento deste trabalho surgiu pelo interesse do autor nos benefícios que a robótica pode proporcionar ao ensino, aliada a potencialização de seu uso e a expansão de seus limites de atuação proporcionados pela inserção da programação. Este interesse está relacionado à formação do pesquisador como profissional da área de programação de computadores, por sua atuação na área de desenvolvimento de softwares educacionais e pela busca em contribuir para o ensino no Brasil.

Motivado também pela ausência de ferramentas direcionadas ao contexto específico do Programa Um Computador por Aluno (PROUCA), propõe-se um ambiente que permita a criação de projetos em RE de forma intuitiva nos laptops distribuídos, sendo seu público-alvo, preferencialmente, professores e alunos das escolas públicas brasileiras parceiras do PROUCA. Cabe ressaltar no entanto que a referida proposta não se restringe a este domínio, podendo ser utilizada de forma mais ampla, em outros contextos que apresentem características semelhantes.

O presente trabalho tem como objetivo propor e implementar um ambiente com linguagem de programação visual – denominado DuinoBlocks – com vistas a facilitar o acesso à programação do dispositivo robótico Arduino (2013) por usuários

iniciantes, bem como promover novas possibilidades pedagógicas de utilização dos recursos computacionais do PROUCA.

Para avaliar a proposta do DuinoBlocks, pretende-se submeter o software a testes com o público-alvo verificando a sua aceitabilidade. Espera-se que este trabalho possa contribuir para que professores e alunos sejam capazes de entender e elaborar algoritmos com o uso intuitivo da linguagem de programação visual.

1.5 Organização da Dissertação

Esta dissertação está organizada em sete capítulos, além desta introdução. As condições para a utilização e o desenvolvimento do DuinoBlocks estão divididos nos três capítulos subsequentes (Capítulos 2, 3 e 4). No segundo capítulo são apresentados os pré-requisitos tecnológicos e educacionais do contexto de utilização do DuinoBlocks. O terceiro capítulo trata da revisão da literatura, no que diz respeito ao uso de programação no meio educacional e faz uma análise de ambientes que utilizam VPL em robótica e, no quarto capítulo, os principais requisitos para o desenvolvimento do DuinoBlocks são apresentados.

No quinto capítulo descreve-se o ambiente proposto e implementado. O sexto capítulo apresenta a metodologia empregada para avaliar o DuinoBlocks e os seus resultados. No sétimo e último capítulo conclui-se o trabalho, tecendo as considerações finais e propondo trabalhos futuros. Em seguida temos a bibliografia, os apêndices e os anexos.

2 PRÉ-REQUISITOS TECNOLÓGICOS E EDUCACIONAIS

Este capítulo objetiva descrever o contexto de utilização do DuinoBlocks. Ele está organizado em duas seções. A Seção 2.1 (Projeto Arduino) trata dos pré-requisitos tecnológicos apresentando o projeto Arduino como uma tecnologia eletrônica de baixo custo para a implementação de trabalhos em robótica educacional. A Seção 2.2 (Programa Um Computador por Aluno) trata dos pré-requisitos educacionais, apresentando o projeto de pesquisa em que este trabalho está inserido.

2.1 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 *et al*. (2007), Sasahara e da Cruz (2007), Filho e Gonçalves (2008), Miranda *et al*. (2010) e Santos (2010) apresentam bons resultados em se tratando de Robótica Educacional de Baixo Custo (REBC), democratizando assim o acesso às tecnologias. A REBC utiliza materiais alternativos (sucatas), recursos de hardware e software livres, tais como o projeto Arduino, como forma de viabilizar economicamente projetos na área de RE.

Criado em 2005 na Itália, o projeto Arduino (2013) constitui 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. Segundo David Mellis (2009), 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 (Figura 1) é uma placa eletrônica baseada em um circuito de entradas e saídas simples, microcontrolada e desenvolvida sobre uma biblioteca escrita em C/C++. O microcontrolador[1] da família ATMEL AVR (2013) presente na placa pode ser programado com a linguagem de programação Arduino, baseada na linguagem Wiring (2013) e com o ambiente de desenvolvimento Arduino, baseado no ambiente Processing (2013).

[1] Microcontrolador - circuito integrado que contém todas as funções básicas de um computador.

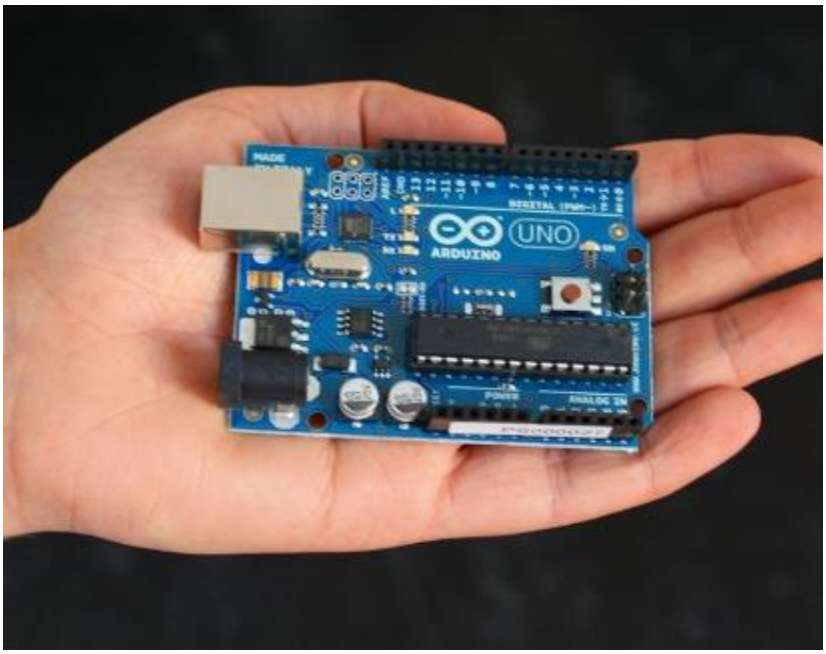


Figura 1: Modelo Uno do Hardware Arduino[2] (2013).

Uma vez programado, o Arduino pode interpretar as variáveis do ambiente que o cerca por meio da recepção de sinais elétricos de sensores (p. ex. sensor de luminosidade, temperatura e movimento), bem como pode interagir com o mundo externo, através de atuadores (p. ex. LED[3], motores e *displays*[4]). A Figura 2 mostra, de forma esquemática, o funcionamento do Arduino ao manipular atuadores com base em dados provenientes de sensores.

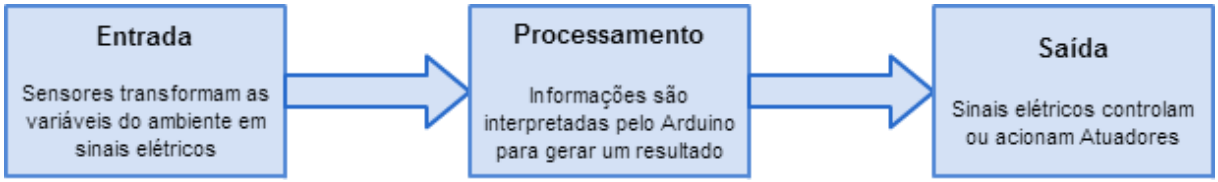


Figura 2: Diagrama de Funcionamento do Arduino

O ambiente de programação Arduino (Figura 3) é uma aplicação multiplataforma desenvolvida em linguagem Java e liberada sob a licença GPL [GENERAL PUBLIC LICENSE, 2013]. Inclui um conjunto de opções que auxiliam na depuração e comunicação (*upload*) com a placa de hardware Arduino.

---

2 Outros modelos de hardwares são encontrados em: http://arduino.cc/en/Main/Products.

3 Diodo emissor de luz, também conhecido pela sigla em inglês LED (*Light Emitting Diode*).

4 *Display* (ou mostrador, em português) é um dispositivo para a apresentação de informação de modo visual.

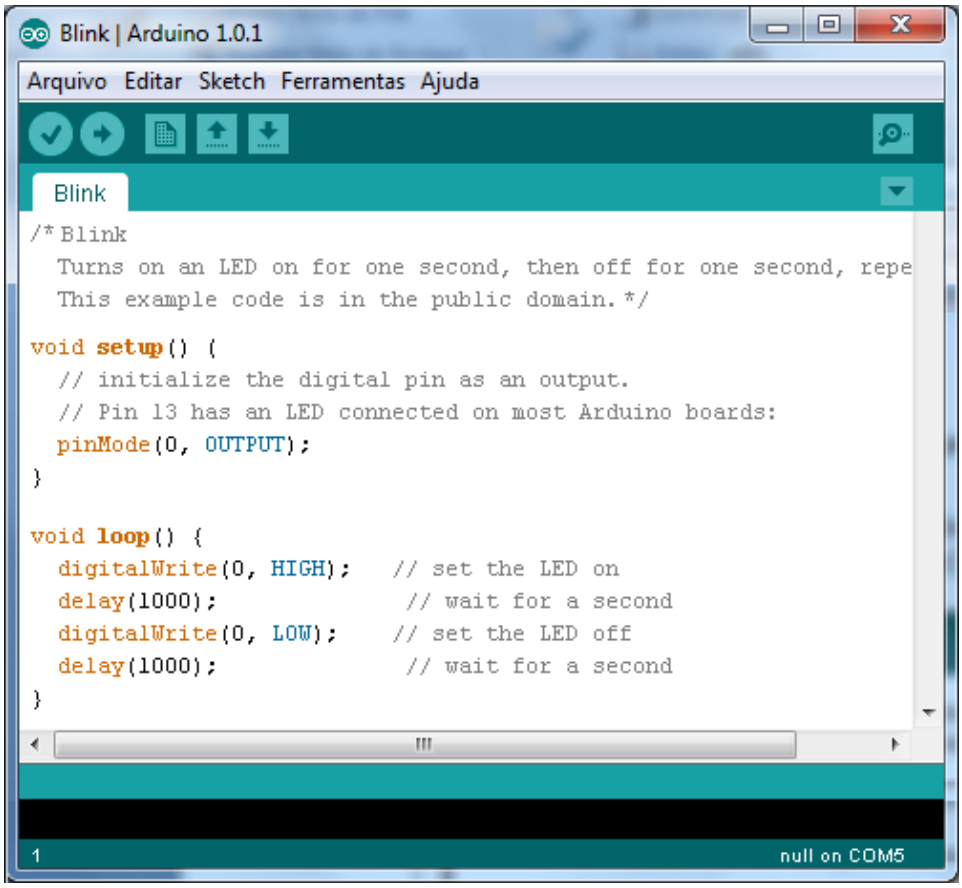


Figura 3: Ambiente de Programação Arduino

Para conexão do hardware Arduino com o computador que possui o ambiente de desenvolvimento, é utilizada uma interface serial com conexão USB. Na plataforma Windows a conexão do Arduino é identificada como uma porta serial tipo COM (p. ex. COM3) e na plataforma Linux, como um dispositivo serial ttyUSB (p. ex. ttyUSB1). O cabo utilizado para interconexão é do tipo USB (o mesmo utilizado para impressoras e scanners) [ALVES *et al*., 2012].

A alimentação elétrica padrão da placa Arduino é 5V (necessária ao funcionamento do microcontrolador) e pode ser obtida de duas formas: do próprio cabo USB ou através de uma fonte externa, tais como baterias ou eliminadores de pilha ligados ao conector de energia externa. Nos casos de alimentação elétrica com fonte externa, pode-se aplicar tensões entre 7 e 20V, pois a placa possui um dispositivo regulador de voltagem que faz a adequação necessária para entregar a tensão de 5V ao microcontrolador [ALVES *et al*., 2012].

Dentre as vantagens de utilizar o projeto Arduino como uma tecnologia eletrônica para a implementação de trabalhos em RE, destaca-se, primeiramente, o fato de ser uma plataforma aberta e possuir baixo custo em comparação a outras plataformas de microcontroladores como a LEGO MINDSTORMS (2013). Em segundo, a facilidade de utilização por não especialistas em eletrônica, como os professores da educação básica.

2.2 Programa Um Computador por Aluno

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 (Figura 4) portáteis novos e de baixo custo, com conteúdos pedagógicos [PROUCA, 2013].

No final de 2011, o Ministério da Educação, através da SEED, CAPES e CNPq, faz uma chamada via edital para que grupos de pesquisa no Brasil apresentassem propostas que contemplassem o uso dos laptops adquiridos pelas escolas parceiras do PROUCA [SAMPAIO e ELIA 2012].

Figura 4: Laptop Classmate distribuído pelo PROUCA[5]

[5] Fonte da imagem: <http://br-linux.org/wparchive/2008/zumo-apresenta-o-novo-classmate-pc.php>.

Um dos projetos selecionados pelo referido edital foi o projeto Uca na Cuca [UCA NA CUCA, 2013] que propõe ações inovadoras, reflexivas e práticas, sobre o uso da robótica educacional em sala de aula, a serem desenvolvidas com enfoque de pesquisa científica básica e tecnológica, implicando no alcance das seguintes metas:

1. **Curso de formação em robótica educacional com hardware livre**: proposta, desenvolvimento, aplicação e avaliação de uma metodologia para formação de professores em robótica educacional [PINTO, 2011];
2. **Produção de kit didático centrado na plataforma Arduino**: testar, validar e documentar materiais didáticos que deverão estar pronto para o reuso em rede web e/ou para ser distribuído;
3. **Adaptação para o PROUCA e plataforma Arduino de uma linguagem de programação visual "ProgrameFácil" desenvolvida no GINAPE**: propõe o desenvolvimento de uma linguagem para funcionar nos computadores do PROUCA tomando com ponto de partida o "ProgrameFácil" [MIRANDA, 2006];
4. **Ambiente virtual de acesso remoto de atividades didáticas em Robótica Educacional – LabVad/RobEd**: ambiente web para acesso remoto a práticas com robótica educacional, possibilitando estudos a distância e sem custos iniciais na aquisição do Kit de Robótica [da SILVA *et al*., 2014];
5. **Avaliação sistêmica da intervenção** nas escolas parceiras [PINHEIRO *et al*., 2013].

O presente trabalho está inserido na meta 3 do projeto supracitado que prevê a criação de um novo ambiente de desenvolvimento de programas (IDE) para o hardware Arduino, incorporando uma VPL mais intuitiva ao público alvo. Esse ambiente é denominado DuinoBlocks e seus principais requisitos e características estão descritos nos Capítulos 4 e 5.

3 REVISÃO DA LITERATURA

Neste capítulo são apresentados os referenciais teóricos (Seção 3.1) e uma discussão sobre o uso da programação visual na educação (Seção 3.2). São descritos os trabalhos relacionados e o estudo realizado sobre o mecanismo de elaboração de algoritmos em ambientes de programação visual, sobretudo, os voltados para o hardware Arduino (Seção 3.3). Encerra-se com as considerações sobre os trabalhos relacionados (Seção 3.4).

3.1 Referencial Teórico

O referencial teórico utilizado para nortear este trabalho está baseado principalmente nos estudos de Seymour Papert e José Armando Valente, a respeito de como a programação pode ser utilizada em benefício do processo de construção do conhecimento dos estudantes. Utilizou-se também as teorias socioconstrutivistas de Lev S. Vygotsky e Paulo Freire, sobre a importância da interação e da realidade social em sala de aula.

O objetivo da programação na Robótica Educacional não é ensiná-la por si mesma, mas de utilizá-la em proveito de algum outro fim educacional. Neste sentido, a utilização da programação em RE objetiva proporcionar um ambiente de aprendizado baseado na resolução de problemas e na elaboração de projetos. Na Figura 5 Valente (1998) sugere o ciclo de ações que acontece na interação aprendiz-computador na situação de programação. O presente trabalho busca tornar a "*descrição da solução do problema através de uma linguagem de programação*" mais intuitiva e fácil de utilizar.

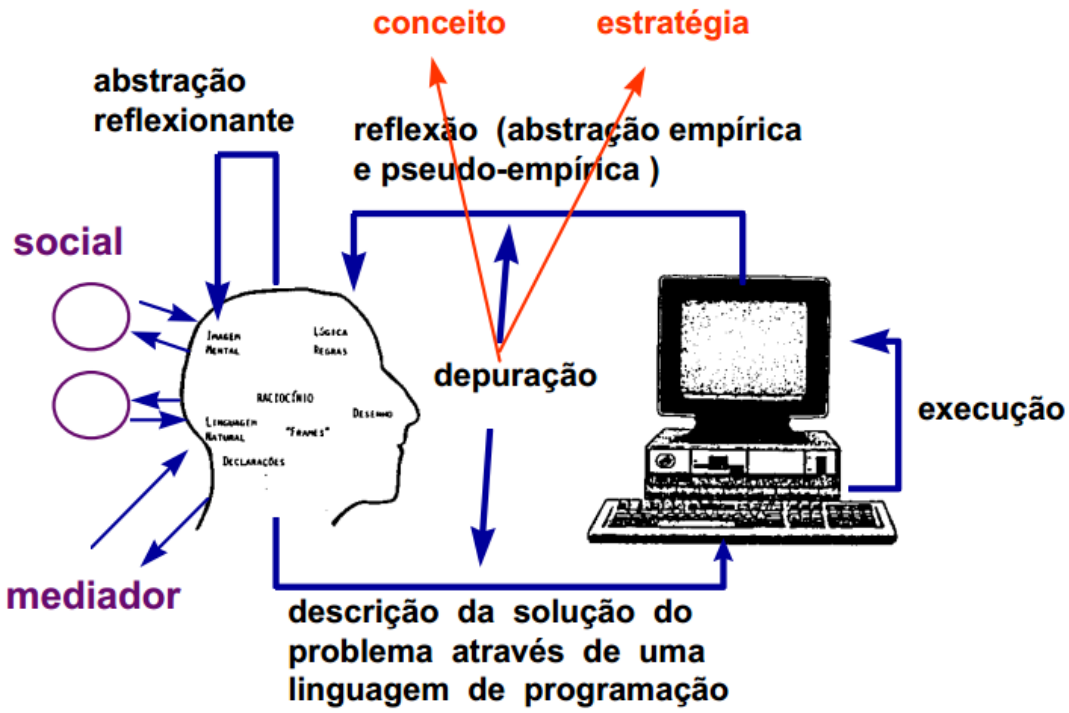


Figura 5: Ações entre a interação aprendiz-computador na situação de programação [Valente, 1998]

Além das ações que o aluno realiza, o diagrama da Figura 5 mostra os elementos sociais que permeiam e suportam a sua interação com o computador. Seus principais contribuidores são:

- Jean Piaget, com estudos sobre os diferentes níveis de abstração: empírica, pseudo-empírica e reflexionante [PIAGET, 1995] acarretados aqui por meio da reflexão realizada pelo aluno sobre o que ele intenciona e o que é produzido pelo computador;

- Lev Vygotsky, com estudos sobre a importância da interação social em sala de aula, descrevendo um modelo de mediação, segundo o qual o mediador é efetivo quando ele age dentro da Zona Proximal de Desenvolvimento (ZPD), definida como

  > a distância entre o nível de desenvolvimento atual, determinado pela resolução de problema independente e o nível de desenvolvimento potencial determinado através da resolução de problema sob auxílio do adulto ou em colaboração com colegas mais capazes [VYGOTSKY, 1978, p. 86].

  Neste sentido, o mediador deve observar os níveis de desenvolvimento do aluno durante a descrição da solução do problema para intervir dentro da ZPD. Assim, se o mediador atuar no nível de desenvolvimento atual, não acrescentará em nada, uma vez que o aluno já domina o assunto proposto. Atuando além do nível potencial de desenvolvimento, o aluno não estará apto para realizar o proposto.

- Seymour Papert, um dos pioneiros no estudo sobre a introdução da informática na educação, difundiu a utilização do computador por meio da linguagem de programação LOGO[6]. Ele parte do princípio que os alunos carregam uma bagagem de conhecimentos a serem aproveitados na construção de novas estruturas cognitivas. Portanto, ao usar a programação, o aluno coloca em prática suas intenções e seus conhecimentos internalizados para a construção de novos saberes. Programar, na visão de Papert significa:

---

6 Site http://el.media.mit.edu/logo-foundation/index.html

> nada mais, nada a menos, comunicar-se com o computador, numa linguagem que tanto ele [computador] quanto o homem podem entender [PAPERT, 1985, p. 18].

Sistematizado por Papert, as ideias do Construcionismo valorizam a construção do conhecimento por meio da interação do sujeito com objetos resultantes de sua cultura. Ele infere, ainda, que os alunos tornam-se criadores de conhecimento, deixando de ser apenas receptores de um conhecimento pronto e acabado e passando a criá-lo com o uso do computador.

Segundo esta premissa é o aluno quem diz ao computador o que deve ser feito, por meio da linguagem de programação. Esta concepção difere da maioria dos softwares educacionais encontrados no mercado, que se constituem em programas prontos, fechados e baseados numa concepção instrucionista de aprendizagem, isto é, por meio de instruções, o programa diz para a criança o que dever ser feito [POCRIFKA e SANTOS, 2009].

- Paulo Freire, sobre a importância de aspectos da realidade social dentro sala de aula. No contexto Um Computador por Aluno, o educando está inserido em um ambiente social e não está isolado da sua comunidade. O contexto social pode ser utilizado como fonte de suporte intelectual e afetivo, ou mesmo de problemas e situações a serem resolvidos, pois

  > o aluno se interessa mais pelo problema contextualizado do que quando este é "fabricado" pelo professor, no sentido de facilitar a explicação de um determinado conceito [FREIRE *apud* VALENTE, 1970].

Neste sentido, a aprendizagem torna-se mais significativa quando se utiliza elementos do ambiente social em que o aluno está inserido como fonte de ideias de problemas a serem resolvidos e quando é fruto de seu próprio esforço [MAISONNETTE, 2002].

Diante deste contexto e partindo da ideia de um laptop conectado à Internet, disponibilizado na escola para cada estudante e educador, o presente trabalho pretende oportunizar ao professor a utilização da robótica, como tema problematizador, embutida na elaboração de algoritmos para a resolução de problemas cotidianos, promovendo o poder de pensamento do aluno. Assim, sob o

ponto de vista pedagógico, o ambiente proposto é destinado ao desenvolvimento prático de conceitos curriculares, por meio da interação do aluno com os objetos desse ambiente.

3.2 Programação Visual na Educação

As primeiras VPL surgiram na década de 1960, com aperfeiçoamentos nos anos 1970 e 1980. O objetivo era de permitir 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 VPLs para fins específicos [PASTERNAK, 2009].

Na ciência da computação, os ambientes de programação visual são utilizados como uma abordagem diferenciada ao ensino introdutório de programação de computadores em cursos superiores de áreas tecnológicas [MÉLO *et al.*, 2011; AURELIANO e TEDESCO, 2012; BRANDÃO *et al.*, 2012]. Estas 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 ensinos fundamental e médio, a programação visual vem sendo proposta, principalmente, para tornar a aprendizagem de conceitos curriculares mais lúdica e motivadora [BARBERO *et al.*, 2011; MARTINS, 2012; SCAICO *et al.*, 2012].

Atualmente, novas VPL vêm sendo desenvolvidas para crianças de 5 a 7 anos de idade, como é o caso do ScratchJr. A ideia aqui é dar a oportunidade a esse público infantil de explorar as possibilidades educacionais da programação de computadores. [FLANNERY *et al*., 2013; KAZAKOFF e BERS, 2013].

As linguagens visuais não são para todos os usuários, ou para qualquer situação. São muitas vezes restritas a um único domínio, tornando-as difíceis de adaptar a outras áreas e aos planos de aula dos educadores. Contudo, tais linguagens são poderosas ferramentas para introduzir os estudantes de todas as idades à programação e ao pensamento crítico.

3.3 Trabalhos Relacionados

Esta pesquisa utiliza ideias apresentadas no trabalho de Miranda (2006), o qual especificou e implementou artefatos de hardware e software (Figura 6) de baixo custo para um kit de RE. No entanto, o foco do presente trabalho é o desenvolvimento de uma linguagem visual intuitiva e fácil de ser utilizada, principalmente no contexto PROUCA, por usuários não especialistas em programação de computadores.

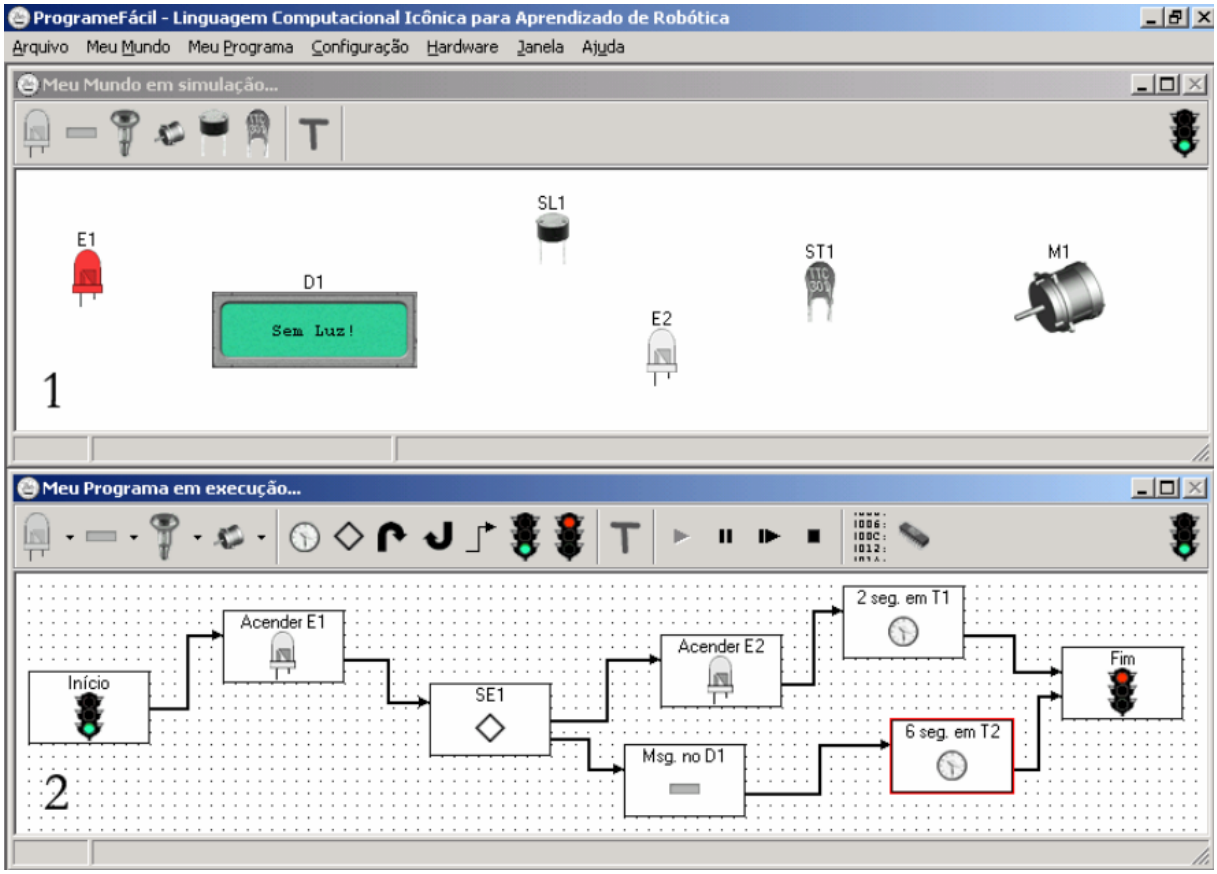


Figura 6: ProgrameFácil – Linguagem Computacional Icônica para Aprendizado de Robótica

O desenvolvimento do DuinoBlocks é inspirado nas ideias apresentadas no trabalho de Pinto (2011) que focou na formação de professores na área de RE sustentada sob dois pilares: um pedagógico, com a aplicação de uma arquitetura interativa apoiada nas TIC e outro tecnológico, com a proposta de utilização de tecnologias livres de hardware, como o projeto Arduino, objetivando o acesso de instituições públicas de ensino a modernas plataformas de programação, seja pelo fator custo, seja pela facilidade de programação por não especialistas em

informática e eletrônica, como os professores da educação básica. Pinto aponta como trabalhos futuros de sua pesquisa a necessidade do

> Desenvolvimento/aplicação de um ambiente de programação icônico para utilização junto a placa eletrônica Arduino. Observar as possibilidades de uso das linguagens “Programe Fácil” e “Scratch” [PINTO, 2011 p, 128]

Uma análise acerca dos softwares disponíveis para programação gráfica de robôs (Figura 7) foi realizada 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.

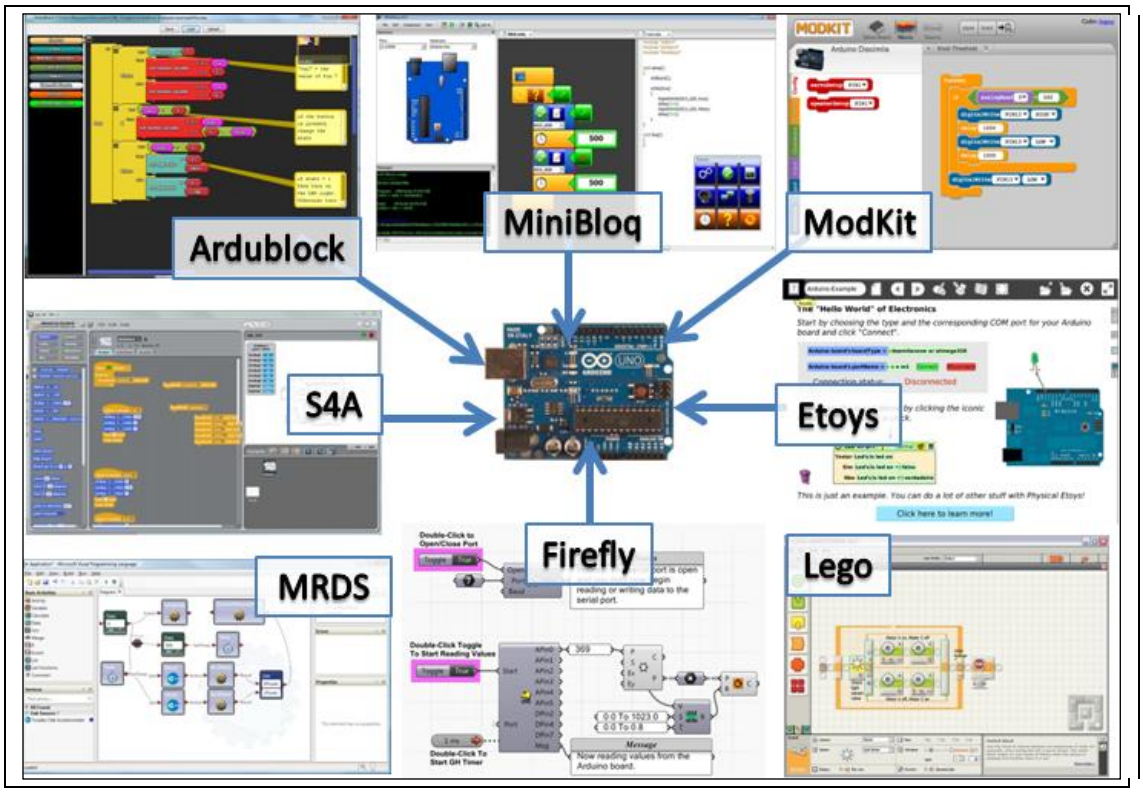


Figura 7: Ambientes de programação gráfica para robótica[7]

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. A segunda abordagem é aquela em que o algoritmo tem uma estrutura de **diagrama**, onde o fluxo de execução segue por arestas e nós.

[7] Os ambientes Microsoft Robotics Developer Studio (MRDS) e Lego Mindstorms não utilizam a plataforma Arduino.

Quanto à representação gráfica do algoritmo, as principais linguagens estudadas são classificadas da seguinte forma:

- **Empilhamento**: Ardublock, Minibloq, ModKit, S4A e Squeak Etoys;
- **Diagrama**: 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 de uso, 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 são classificados da seguinte forma:

- **Ambientes autônomos**: Minibloq, Ardubloq e ModKit.
- **Ambientes não autônomos**: Squeak Etoys, S4A e Firefly.

A seguir serão comentados os principais aspectos dos softwares que mais influenciaram o desenvolvimento do DuinoBlocks. O estudo não é realizado de modo sistemático, ou seja, não segue um quadro de referência em que é avaliado cada

software considerado em relação a um determinado aspecto. A análise é feita em forma resumo, contendo informações sobre os recursos que cada software oferece.

3.3.1 MiniBloq

Um dos principais objetivos do MiniBloq[8] (Figura 8) é levar a computação física e plataformas de robótica para a escola primária. Dentre suas características destacam-se o gerador de código textual e a verificação de erros em tempo real, permitindo ao usuário uma percepção imediata de suas ações sobre ambiente. A portabilidade que permite executar o software direto de um pendrive sem a necessidade de instalação, o suporte multilíngue com tradução para o Português e o suporte para Linux (lançados recentemente) são exemplos de funcionalidades buscadas para o DuinoBlocks. O software, codificado em C++, possui uma identidade visual própria, pois os desenhos de seus blocos e os mecanismos de empilhamento não se assemelham com outros softwares.

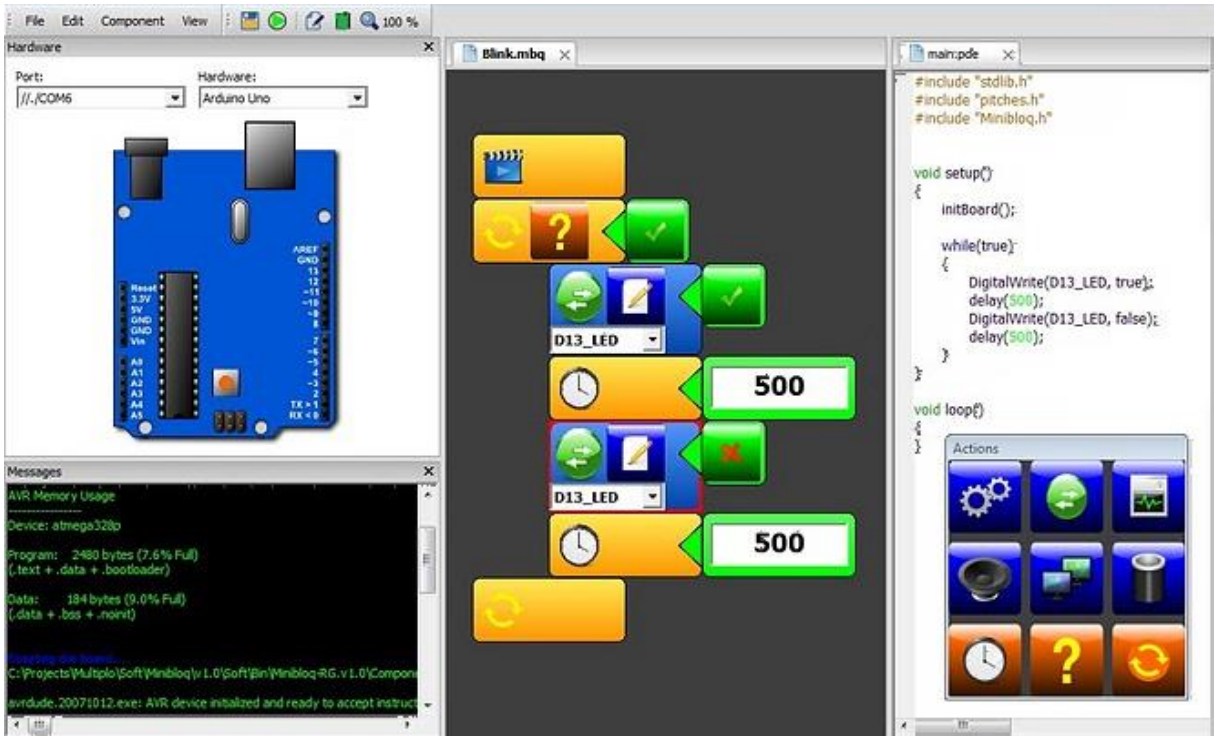


Figura 8: Layout do MiniBloq

Criado pelo argentino Julián da Silva Gillig, o MiniBloq descende de outro programa desenvolvido por ele, o MiniBloques, concebido para ensinar robótica e

[8] Site do MiniBloq: http://blog.minibloq.org/

tecnologia para crianças. Publicado em 1997, o programa foi utilizado em escolas primárias e secundárias da Argentina. Em 2007, Julián começou outro projeto a fim de desenvolver uma plataforma de software para robótica e o MiniBloq foi um dos seus componentes. Mas, o MiniBloq cresceu como um projeto independente, vindo a ser financiado em campanha do Kickstarter, site de financiamento coletivo, em 2011.

3.3.2 S4A

Desenvolvido pelo Citilab, o S4A[9] ou Scratch for Arduino (Figura 9) é uma adaptação para a robótica do Scratch. O Scratch é um dos ambientes de programação gráfica mais populares desenvolvido no MIT Media Lab (2013). Outra adaptação do Scratch, porém não para robótica, é o BYOB (*Build Your Own Blocks*) que possui uma poderosa ferramenta para o usuário criar seus próprios blocos.

O S4A, codificado em Smalltalk, amplia os recursos do Scratch com a inclusão de novos blocos para controlar sensores e atuadores conectados ao Arduino. Seu principal objetivo é atrair pessoas para o mundo da programação e proporcionar uma interface de alto nível para programadores Arduino.

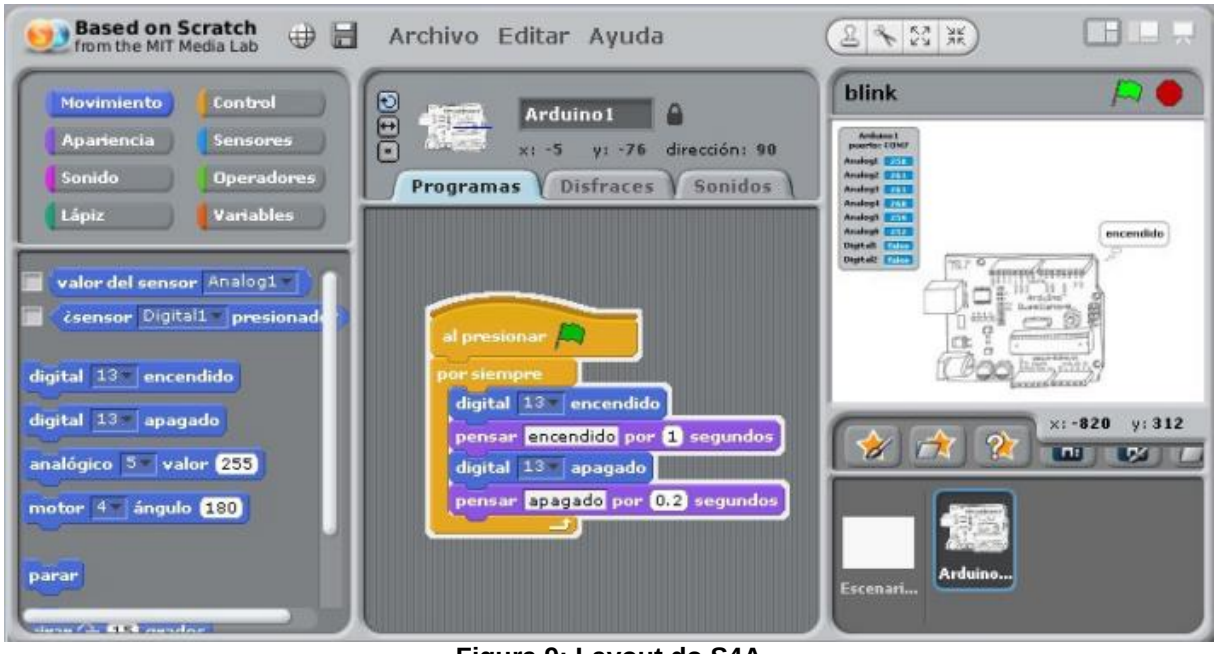


Figura 9: Layout do S4A

S4A interage com Arduino enviando os estados dos atuadores e recebendo os estados dos sensores em uma determinada frequência. A troca de dados é

[9] Site do S4A: http://s4a.cat/

realizada através de um programa específico (*firmware*) previamente instalado no hardware.

3.3.3 ArduBlock

O ArduBlock[10] (Figura 10) é um 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.

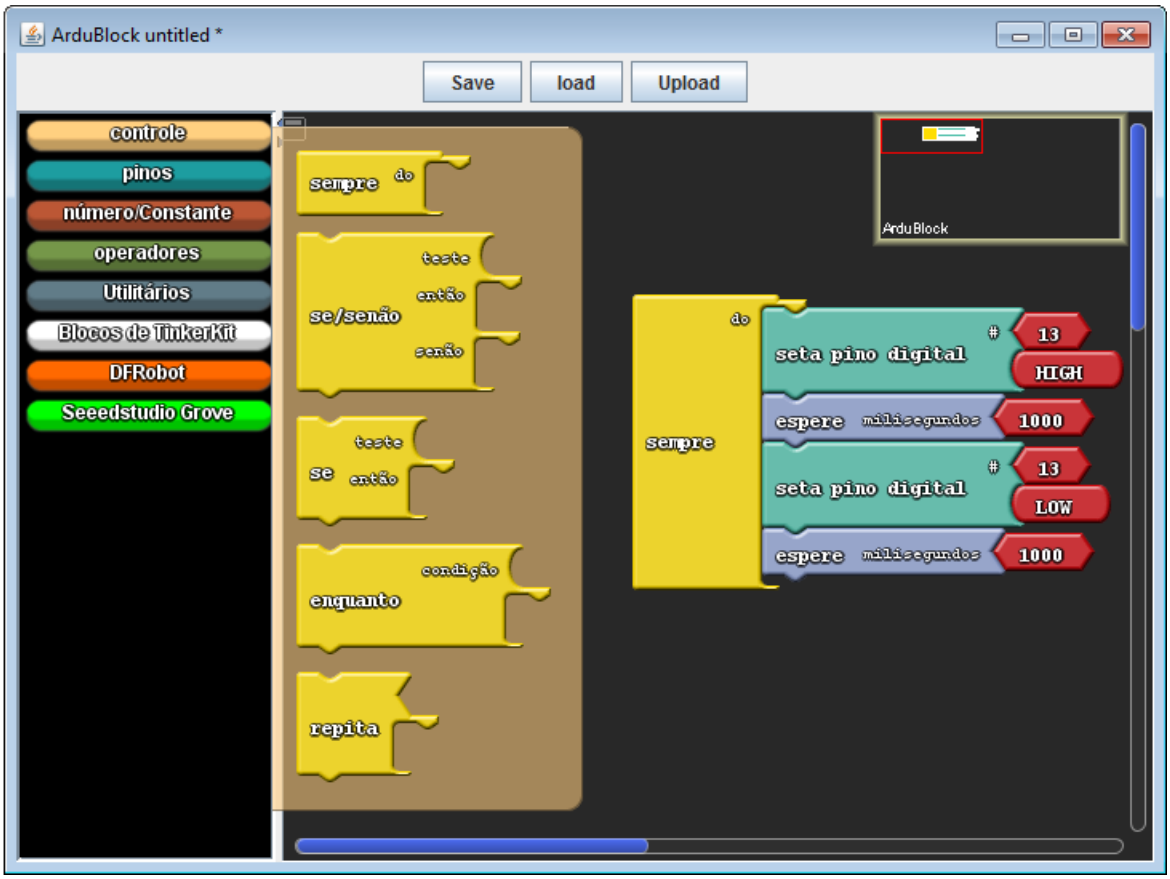


Figura 10: Layout do ArduBlock

Utiliza a biblioteca OpenBlocks[11], um framework extensível para sistemas de programação em blocos gráficos desenvolvido como parte do projeto StarLogo TNG[12] do MIT e igualmente empregado como a base para o Android App Inventor[13].

---

10 Site do ArduBlock: http://blog.ardublock.com/
11 Página do OpenBlocks: http://education.mit.edu/openblocks
12 Página do StarLogo TNG: http://education.mit.edu/projects/starlogo-tng
13 Site do Android App Inventor: http://appinventor.mit.edu/

De acordo com David Li, seu desenvolvedor, utilizar uma linguagem derivada de C não é a melhor maneira de aprender a programar, sendo assim, ele desenvolveu o ArduBlock. Para Li, a programação visual é como livros ilustrados para crianças e em um determinado ponto, elas vão começar a ler o livro sem imagens [ARDUBLOCKGROUP, 2013], referindo-se a transição da programação visual para textual a medida que o usuário ganha mais experiência.

3.3.4 ModKit

O ModKit[14] (Figura 11) é um ambiente de programação para microcontroladores que permite programar o Arduino e outros hardwares compatíveis. O desenho de seus blocos é inspirado no Scratch. O ambiente 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.

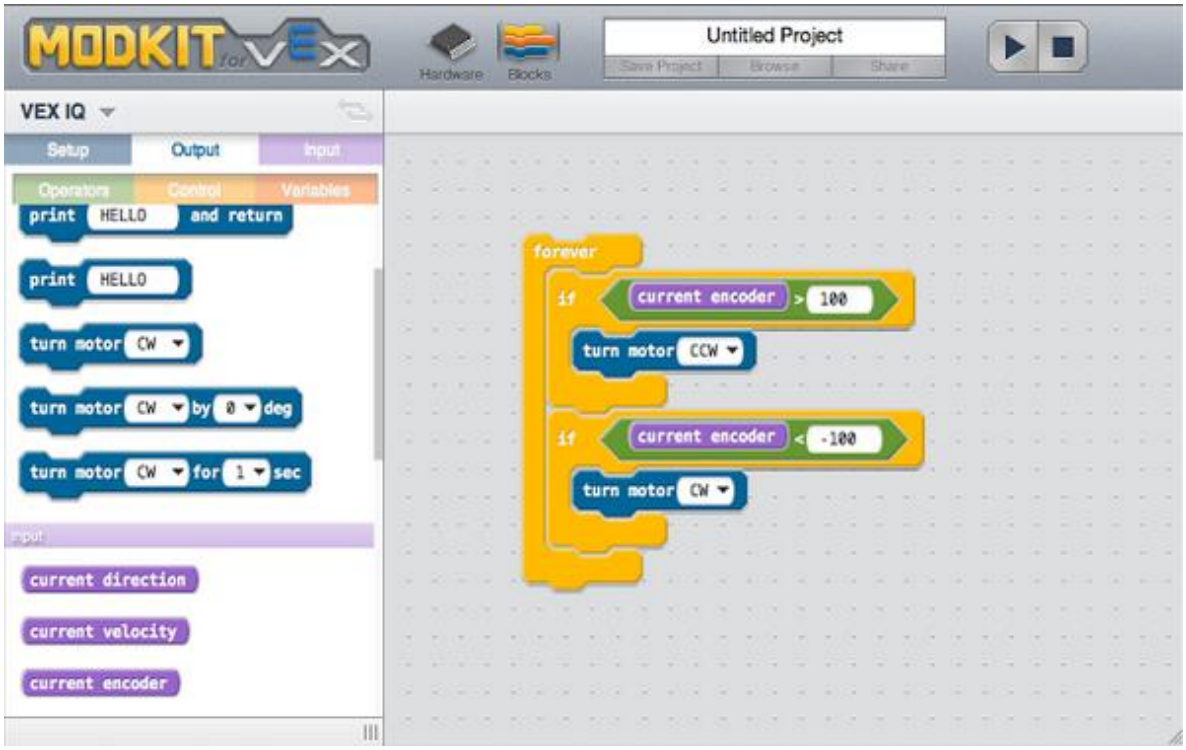


Figura 11: Layout do ModKit

14 Site do ModKit: http://www.modk.it/

3.4 Considerações sobre os Trabalhos Relacionados

A partir do estudo aqui apresentado, foi possível levantar as características operacionais e de implementação que o DuinoBlocks deve possuir. Para cada software supracitado 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 são relatadas a seguir:

- A presença de categorias 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 o fato de como o programa 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 o efeito imediato do código escrito e, por ser derivado do Scratch, a disposição de um contexto visual, para a criação de animações, que os outros não oferecem;

- Vale ressaltar que, com exceção do ModKit, nenhum dos ambientes tratados acima 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.

Percebe-se assim, a partir do estudo realizado sobre programação visual voltada para robótica, 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

Este capítulo descreve os procedimentos realizados para o levantamento dos requisitos do ambiente DuinoBlocks (Seções 4.1, 4.2 e 4.3) seguido da apresentação dos casos de uso do sistema (Seção 4.4).

4.1 Levantamento de Requisitos

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 [CRINNION, 1991]. 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.

O levantamento de requisitos teve início na revisão da literatura em RE e análise de softwares com VPL (Capítulo 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 (Seção 4.2). 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 (Seção 4.3).

4.2 Principais Requisitos

A coleta de opiniões de *stakeholders*[15] como professores em processo de formação em RE e especialistas na área de tecnologias no ensino e robótica ajudou a compreender as reais necessidades dos potenciais usuários. Os três principais requisitos levantados que o ambiente DuinoBlocks deve prover são:

- Requisito de portabilidade: Ambiente Multiplataforma (Subseção 4.2.1);
- Requisito de facilidade de uso: Linguagem de Programação para Iniciantes (Subseção 4.2.2);
- Requisitos de eficiência e desempenho: Adequação às Limitações do Classmate (Subseção 4.2.3).

[15] *Stakeholder* significa público estratégico. São pessoas que de alguma forma são afetadas pelo sistema e têm direta ou indiretamente influência nos requisitos do sistema.

4.2.1 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. Os laptops das escolas do PROUCA, parceiras do projeto Uca na Cuca, possuem o sistema operacional Meego [MEEGO, 2013] - uma 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 [ZHANG, 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 (*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 [SOUSA *et al*., 2010]. Terceiro, introduz facilidades para o compartilhamento e colaboração dos projetos desenvolvidos pelos alunos e professores.

4.2.2 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 [HUNDHAUSEN *et al*.,

2006]. 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. 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 lidar com problemas de natureza léxica e sintática.

4.2.3 Adequação às Limitações do Classmate

As funcionalidades do sistema devem ser adequadas as limitadas especificações de hardware e periféricos do laptop Classmate (2013). 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 alta 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 tempo de execução das ações e a quantidade de memória requerida.

4.3 Aplicação de Teste com Ambiente de Programação Arduino

O projeto UCA na CUCA vem promovendo no município de Piraí o curso de capacitação intitulado “Formação em Robótica Educacional com Hardware Livre”, destinado a professores do ensino fundamental e médio das escolas públicas do Estado do Rio de Janeiro. Os cursistas da segunda turma, que iniciou em 13 de março de 2012, foram professores que não possuíam formação em informática nem conhecimentos em programação. Eles atuavam no ensino fundamental nas áreas de

Matemática, Ciências e História. O curso vinha utilizando o hardware e o software padrão textual do Arduino como tecnologia eletrônica de baixo custo para a implementação dos trabalhos em robótica educacional. Com o propósito de diagnosticar os pontos fortes e fracos, bem como, as dificuldades dos professores na utilização da linguagem de programação Wiring do IDE Arduino, foi utilizado um Teste [COHEN *et al*., 2005] envolvendo a resolução de um problema específico. A seguir são elencados e apresentados em forma de tópicos o conjunto de procedimentos pré-determinados que foram adotados.

- **Fases do Curso**

O curso é semipresencial apoiado por uma plataforma virtual de aprendizagem e dividido em três camadas de acordo com o MHI-3C (Modelo Hierárquico de Interatividade em três camadas) [PINTO, 2011] descrito brevemente a seguir:

1° Pesquisador-Professor:
   i. Fóruns: Tecnologia na Educação (EAD)
   ii. Oficina de robótica educacional (três encontros presenciais)
2° Professor-Professor:
   i. Discussão para criação de atividades didáticas
   ii. Um encontro presencial extra
3° Professor-Aluno:
   i. Aplicação das atividades didáticas com os alunos
   ii. Encontro presencial: Apresentação de trabalhos

O teste foi aplicado no encontro presencial extra na segunda camada do curso. Os professores submetidos ao teste já haviam tido contato com o ambiente de programação do Arduino através de programas-exemplo mostrados na primeira camada do curso.

- **Registro**

Para a resolução do teste foi utilizado o laptop Classmate, o mesmo utilizado no curso. Durante a realização do teste, um programa de *screencast*[16] estava rodando em background no computador. Esta abordagem proporcionou um registro compreensivo dos comportamentos, das atitudes e dos diálogos ocorridos na intervenção [COHEN *et al*., 2005]. Após o teste foram analisadas as ações dos professores que estavam gravadas em um arquivo de áudio e vídeo.

- **Grupos**

Atividades em RE apoiam o trabalho em grupo, logo a aplicação do teste individualmente estaria em desacordo com esta realidade. Por outro lado, grupos com três ou mais integrantes poderiam prejudicar a confiabilidade dos resultados. Desta forma, os professores foram divididos em duplas e caso o número de professores presentes fosse ímpar, um grupo teria três professores.

- **Combinação de Métodos**

O caminho que é feito até que se chegue ao resultado é o foco principal do teste e mais importante que o produto final. Desta forma, os professores foram estimulados a verbalizarem todos os seus pensamentos durante a execução do teste, técnica *Think Aloud* [van SOMEREN *et al*., 1994]. Eles também foram orientados a falarem um membro do grupo por vez para não prejudicar a captura de áudio.

- **Orientações**

As seguintes informações foram passadas aos professores cursistas:

i. **Pesquisa**: O teste aplicado não consiste em um método de avaliação do curso. Consiste em uma etapa da pesquisa de dissertação de mestrado

[16] *Screencast* é o registro (gravação) da saída do vídeo gerado por computador em atividade. Pode ou não conter o áudio integrado. Fonte: http://pt.wikipedia.org/wiki/Screencast.

sobre o desenvolvimento de uma linguagem de programação visual para a robótica educacional no contexto UCA.

ii. **Confidencialidade**: As informações recolhidas neste teste, assim como os dados pessoais serão tratados com confidencialidade e serão utilizados exclusivamente para este estudo.

- **Escolha da Tarefa**

A primeira tarefa do curso envolvendo a programação textual foi com o exemplo chamado *Blink* ou "Pisca LED", muito conhecido no meio da robótica (faz um LED ficar piscando). Ele requer estruturas de dados simples em seu algoritmo (Figura 12). O algoritmo já havia sido apresentado aos professores e explicado o seu código, durante as aulas presenciais na camada 1 do curso. O programa foi executado e constatado o entendimento da turma, comparando o resultado esperado com o resultado real.

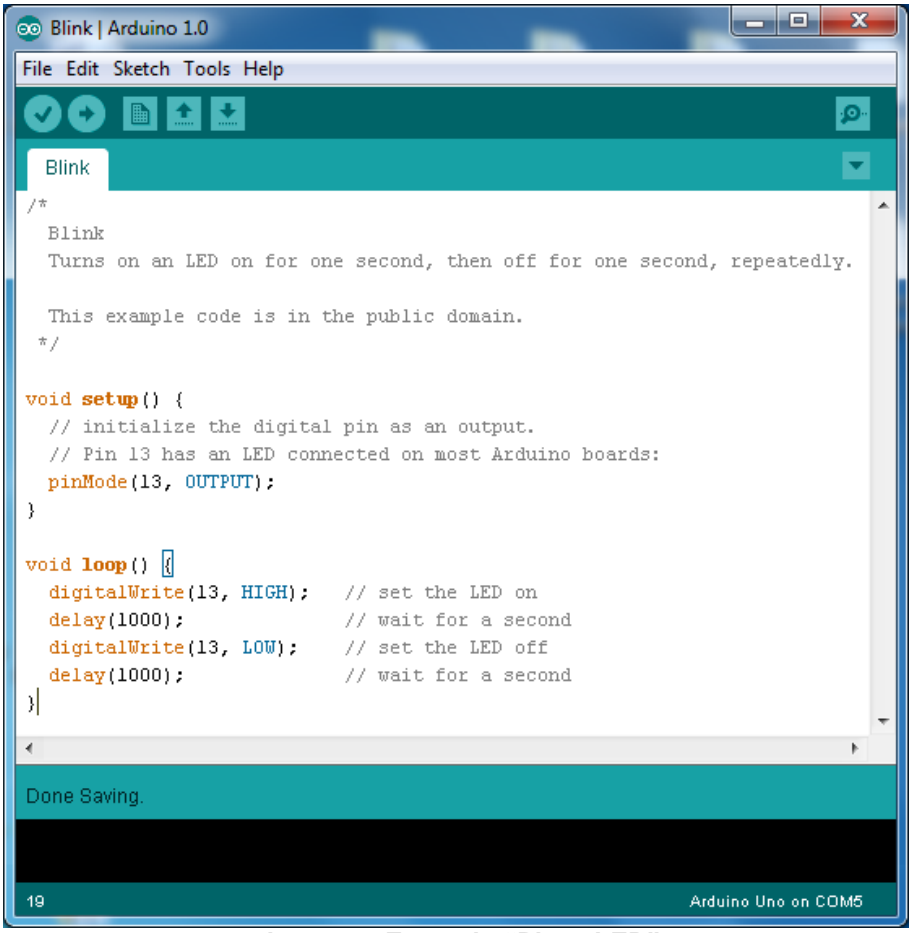


Figura 12: Exemplo "Pisca LED"

Este mesmo exemplo, mas com uma pequena modificação, foi utilizado no teste. Porém, desta vez foi solicitado que eles construíssem o desafio desde o início.

- **Enunciado**

O enunciado do teste foi fornecido de forma oral: “Elaborem um algoritmo que faça um led conectado no pino digital 11 piscar com uma frequência de meio segundo”.

- **Etapas da Aplicação**

O teste foi planejado em etapas que previam o fornecimento de informações a medida que houvesse um prolongamento no tempo de execução das tarefas.

A eletrônica do projeto foi fornecida montada, já que o teste focou a elaboração textual do algoritmo. Os professores foram orientados a não utilizar os exemplos prontos contidos na biblioteca do ambiente Arduino. A aplicação do teste contemplou as seguintes etapas de orientação:

1. Entregar o editor de código do ambiente em branco e solicitar aos professores que fizessem a tarefa desde o início. Aguardar alguns minutos.

2. Passado alguns minutos e constatado um prolongamento no tempo de execução do teste, fornecer o trecho de código apresentado na Figura 13a. Aguardar alguns minutos.

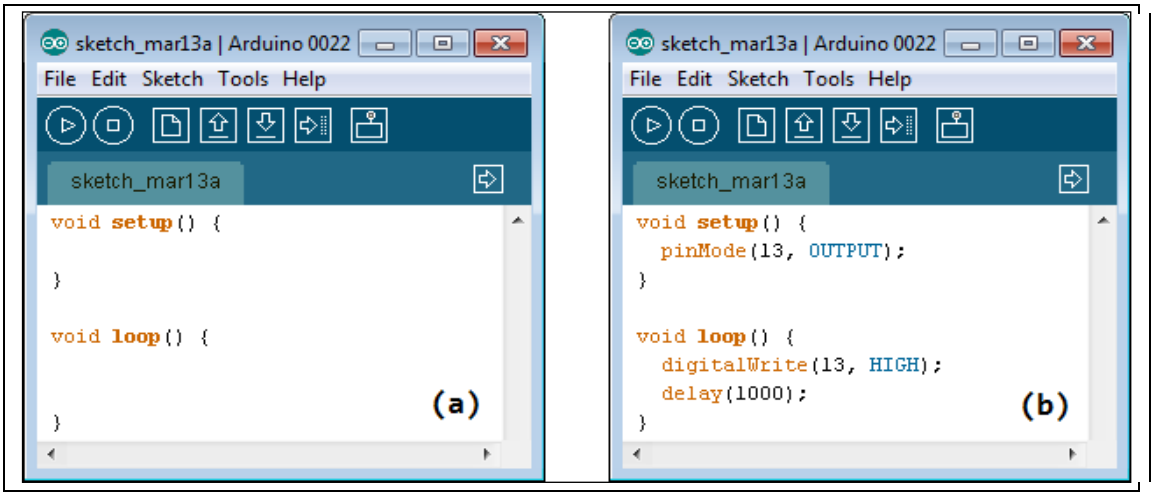


Figura 13: Informações previstas a serem fornecidas

3. Permanecendo as dificuldades na execução do teste, fornecer o trecho de código apresentado na Figura 13b.

As informações foram fornecidas textualmente exatamente como mostradas nas figuras e não foram acompanhadas de explicações (deste modo os professores são forçados a digitar os comandos e a buscar por ajuda no próprio software).

- **Métricas da Avaliação**

Com a realização deste teste, procurou-se saber algumas questões a respeito da usabilidade do software como:

i. Os professores foram capazes de realizar as tarefas?
ii. Quais foram as intenções na realização de cada tarefa?
iii. As informações relevantes foram encontradas?
iv. Quanto tempo demorou na execução das etapas?
v. Os caminhos seguidos foram os mais eficientes?
vi. Os professores sabem o que estão fazendo?
vii. Que problemas encontraram?

- **Objetivos**

O intuito desta experiência foi testar a hipótese de que os professores teriam uma dificuldades em construir o algoritmo para a solução deste problema. E que esta dificuldade, a princípio, poderia ser explicada por alguns dos seguintes motivos:

i. A programação textual desfavorece o não especialista;
ii. Os comandos do software estão em inglês;
iii. Descobrir os comandos existentes não é trivial;
iv. Utilizar a interface não é intuitivo.

4.1.2 Resultados da Aplicação do Teste

Os testes foram realizados com dois grupos de dois professores cada. A última versão disponível do software Arduino foi utilizada. Na época não havia suporte multilíngue. Quanto à interface, não houve dificuldades em relação à utilização dos menus e botões do software como: *Verify* que verifica os erros no programa; U*pload* que carrega o programa na placa eletrônica; menu *Tools*, itens de menu *Board* e *Serial Port* onde é especificado o modelo de hardware usado e a porta serial em que esta conectado.

Para um dos grupos, a primeira dica foi fornecida sem que nenhum comando tivesse sido digitado. Logo no início do teste, um professor relatou: "*Se eu visualizar o comando, eu tenho mais ou menos a ideia do que ele faz*" referindo-se ao fato de não se lembrar dos comandos necessários para representar a solução da tarefa proposta. Em seguida o mesmo completa: "*partindo da visualização dele [comando] eu consigo fazer alguma coisa, partindo do zero eu nunca peguei para programar*" demonstrando uma dependência em relação a algum algoritmo de exemplo disponível para tentar adaptá-lo.

Ao se deparar com um erro de compilação, a dupla foi questionada sobre o entendimento da mensagem de erro que aparece no console do software. Eles disseram não compreender, alegando o fato de a mensagem estar no idioma inglês. Os erros de compilação que ocorreram durante o teste foram: ausência de ponto-e-vírgula; comandos digitados fora das chaves (escopo) do método; comandos digitados com espaços entre as letras; constantes digitadas em minúsculo (Figura 14), demonstrando a dificuldade encontrada principalmente quanto à sintaxe da linguagem textual.

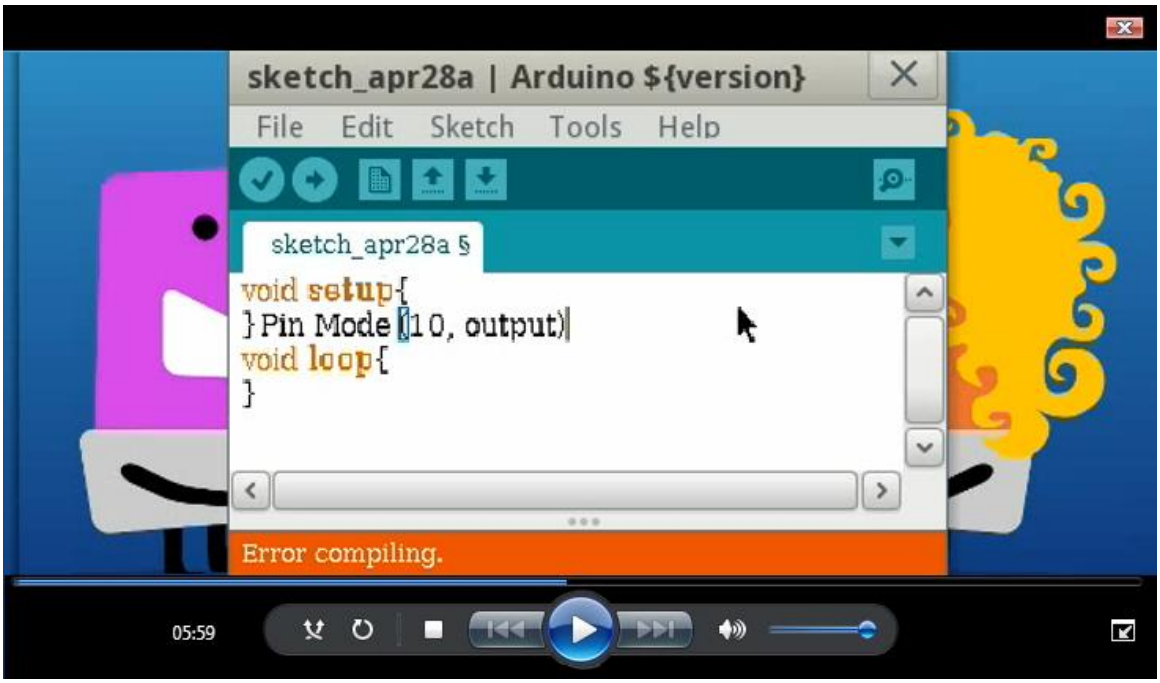


Figura 14: Erros de Sintaxe Durante a Compilação do Programa

Apesar dos erros de sintaxe, a captura do áudio permitiu perceber que em determinados momentos os professores verbalizaram corretamente os passos que queriam realizar. Em um trecho da gravação um professor informa ao outro que está inserindo o tempo que o LED vai ficar aceso, mas como pode ser visto na Figura 15 suas intenções não são representadas corretamente na linguagem de programação Wiring. Neste caso o professor acrescentou equivocadamente o parâmetro “500” na função “*digitalWrite*”, quando o correto seria adicionar um comando “*delay*”.

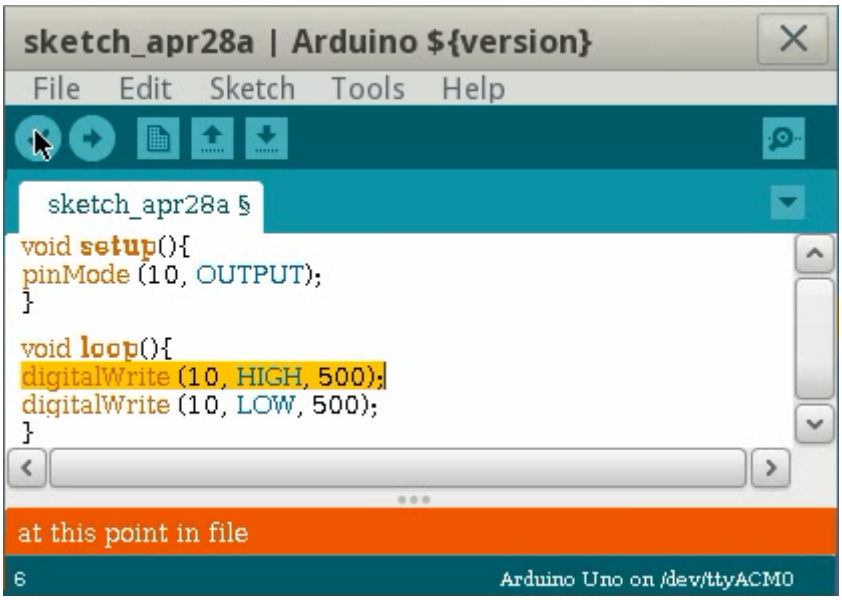


Figura 15: Passagem de parâmetros errada

A análise do vídeo demonstrou a importância do recurso *Syntax Highlighting*[17] na legibilidade do algoritmo. Em diferentes momentos da gravação o professor percebe que digitou errado ao notar a formatação do texto e rapidamente torna a digitar correto. A Figura 16 mostra os instantes 6m:13s e 6m:18s, intervalo em que o professor percebe que digitou a constante "*high*" em minúsculo, quando o correto é "*HIGH*" em maiúsculo. Esta percepção foi facilitada pelo *feedback* que o editor de código fornece ao destacar as palavras-chaves.

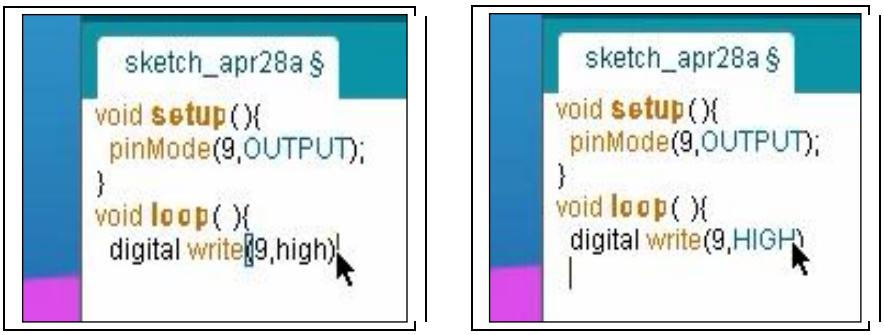


Figura 16: Momento em que é percebido o erro de digitação

Os grupos levaram em média dez minutos até completar o teste. A interface do software se mostrou relativamente simples para as ações básicas que a tarefa exigia. Os dois grupos solicitaram as dicas previstas, pois não lembravam como utilizar os comandos. A partir do teste realizado, foi possível levantar algumas necessidades dos usuários, ajudando a definir os seguintes requisitos para o DuinoBlocks:

- Ter os blocos de comando categorizados e dispostos no layout;
- Conter ajuda sobre a utilização dos blocos;
- Disponibilizar os parâmetros de um comando sempre que o conjunto de parâmetros possíveis for conhecido;
- Manter a simplicidade nos recursos que realizam a comunicação com o hardware;
- Fornecer um tratamento de erros mais compreensivo.

[17] Realce de sintaxe é uma funcionalidade disponível em alguns editores de código fonte que exibe o texto em uma formatação específica para cada categoria de termos.

4.4 Casos de Uso

A fim de apresentar as funcionalidades propostas e a interação usuário-sistema foi elaborado o diagrama de caso de uso mostrado na Figura 17. Apesar de o ambiente DuinoBlocks ser voltado para dois perfis de usuários diferentes, a saber, professor e aluno, estes são identificados como um único ator (Usuário) no sistema, pois não foram encontrados motivos para que houvesse uma distinção nos papéis desempenhados por eles na interação com o sistema.

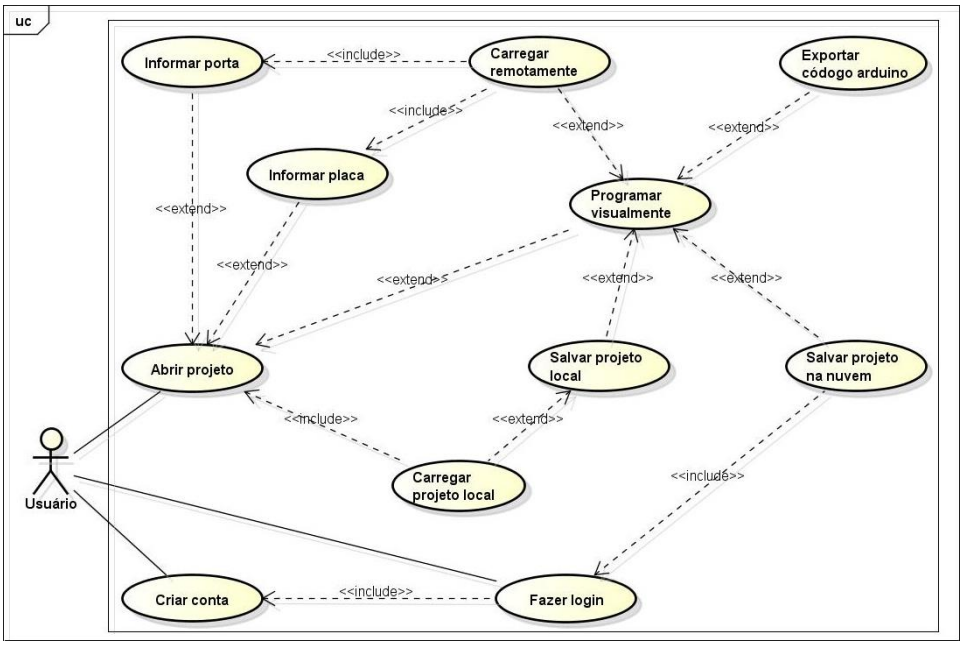


Figura 17: Diagrama de Caso de Uso.

A seguir são detalhados os casos de uso que o ambiente DuinoBlocks deve prover e seu comportamento em certas situações, descrevendo como os usuários interagem com o sistema.

Tabela 1: Caso de Uso – Criar conta

Precondições	Nenhuma.
Fluxo Normal	1. O usuário informa dados.

Tabela 2: Caso de Uso – Fazer login

Precondições	1. Ter executado o caso de uso “Criar conta”
Fluxo Normal	1. O usuário informa email e senha.
Fluxo Alternativo: Esqueceu Senha	1. O usuário informa o email e clica em “Esqueci senha”. 2. O sistema envia um email ao usuário com a senha.

Tabela 3: Caso de Uso – Programar visualmente

Precondições	1. Ter executado o caso de uso “Abrir projeto”.
Fluxo Normal	1. O usuário cria um novo programa.
Fluxo	1. O usuário edita um programa salvo.

Tabela 4: Caso de Uso – Carregar remotamente

Precondições	1. Ter executado o caso de uso "Programar Visualmente". 2. Ter conectado a placa (hardware Arduino) na porta USB. 3. Ter executado o caso de uso "Informar Placa". 4. Ter executado o caso de uso "Informar Porta".
Fluxo Normal	1. O usuário clica em carregar. 2. O sistema carrega o programa na placa.
Fluxo Alternativo: Informação Incompleta.	2a. O usuário não executa as precondições 2 e/ou 3. 2b. O sistema envia aviso de ausência. Retorna ao passo 1.
Fluxo Alternativo: Informação Errada.	3a. O usuário informa placa e/ou porta errada. 3b. O sistema fracassa ao carregar e envia aviso de erro. Retorna ao passo 1.

Tabela 5: Caso de Uso – Salvar projeto na nuvem

Precondições	1. Ter executado o caso de uso "Fazer login". 2. Ter executado o caso de uso "Programar visualmente".
Fluxo Normal	1. O usuário salva o projeto aberto na nuvem.

Tabela 6: Caso de Uso – Salvar projeto local

Precondições	1. Ter executado o caso de uso "Programar visualmente".
Fluxo Normal	1. O usuário clica em "Salvar projeto no computador". 2. O sistema abre uma janela e o usuário especifica o caminho do arquivo.

Tabela 7: Caso de Uso – Informar placa

Precondições	1. Ter executado o caso de uso "Abrir projeto".
Fluxo Normal	1. O usuário informa o modelo de placa arduino que esta conectada.

Tabela 8: Caso de Uso – Informar porta

Precondições	1. Ter executado o caso de uso "Abrir projeto".
Fluxo Normal	1. O usuário informa a porta USB que a placa esta conectada.

Tabela 9: Caso de Uso – Exportar código arduino

Precondições	1. Ter executado o caso de uso "Programar visualmente".
Fluxo Normal	1. O usuário clica em "Exportar código arduino".

Tabela 10: Caso de Uso – Abrir projeto

Precondições	Nenhuma.
Fluxo Normal	1. O usuário clica em "Criar novo projeto".
Fluxo Alternativo: Editar	1. O usuário executa o caso de uso "Fazer login". 2. O usuário seleciona um projeto salvo e clica em "Editar projeto".
Fluxo Alternativo: Carregar	1. O usuário executa o caso de uso "Carregar projeto local".

Tabela 11: Caso de Uso – Carregar projeto local

Precondições	1. Ter executado o caso de uso "Salvar projeto local".
Fluxo Normal	1. O usuário clica em "Carregar projeto local".

5 AMBIENTE DUINOBLOCKS

O propósito deste capítulo é apresentar o layout, as principais funcionalidades e a arquitetura do ambiente desenvolvido.

5.1 Implementação

As funcionalidades do ambiente DuinoBlocks foram idealizadas levando-se em conta as necessidades relatadas no Capítulo 4 (Requisitos do Sistema) e outros ambientes de programação visual tratados na Seção 3.3 (*Trabalhos Relacionados*). Diferentes aspectos de usabilidade do ambiente foram projetados segundo referências às Leis da Simplicidade proposta por Maeda (2006).

No desenvolvimento do DuinoBlocks foi empregado o modelo evolutivo de engenharia de software detalhado por Crinnion (1991). Tal modelo propõe o desenvolvimento de um protótipo inicial que vai sendo submetido a avaliações e refinamentos, levando à criação de sucessivas versões, até alcançar a funcionalidade desejada.

5.2 Layout

A Figura 18mostra o layout do ambiente DuinoBlocks, com um exemplo de algoritmo (Pisca LED) bastante conhecido na RE. O layout é dividido em cinco regiões, detalhadas a seguir.

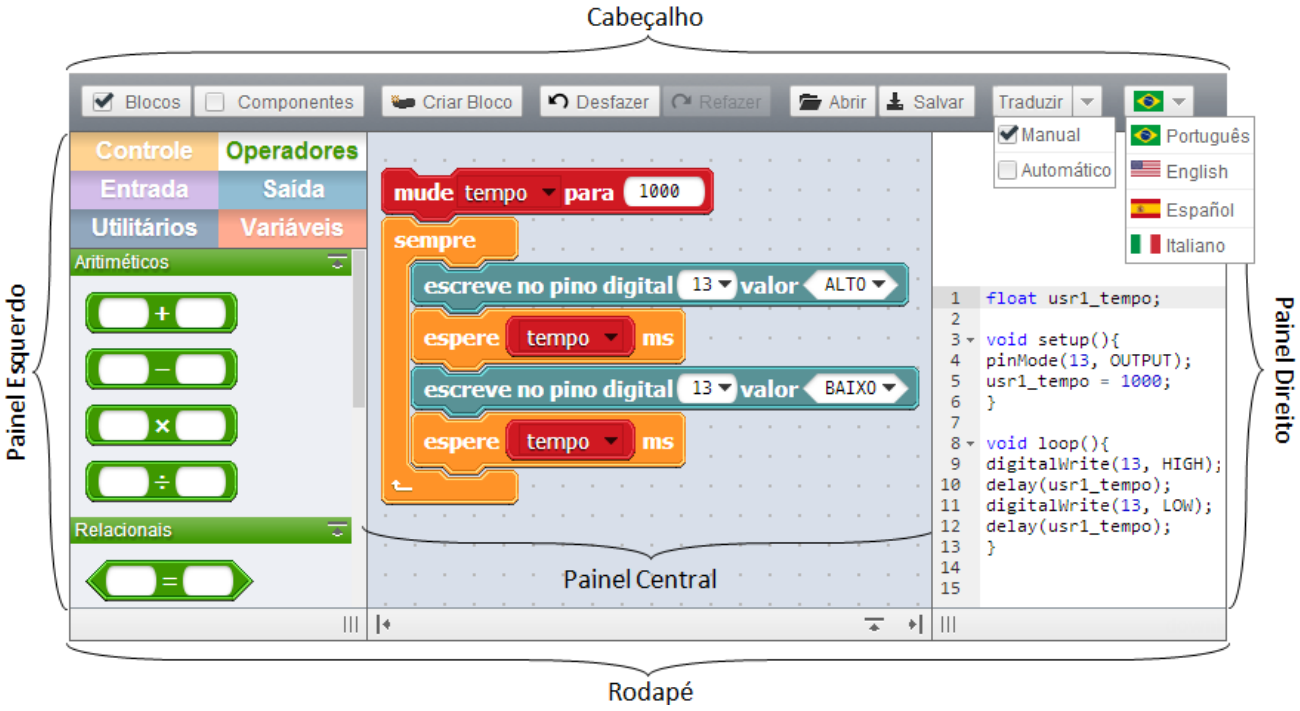


Figura 18: Layout do Ambiente DuinoBlocks

- *Cabeçalho*. Barra de ferramentas com botões, onde são apresentadas as ações de mais alto nível do ambiente.

- **Blocos / Componentes**: alternam entre os módulos Blocos e Componentes (Seção 5.8);
- **Salvar / Abrir**: salva e abre o algoritmo visual em um arquivo no formato JSON com extensão DBK;
- **Desfazer / Refazer:** retrocede e avança uma ação realizada pelo usuário;
- **Criar Bloco:** módulo de criação de blocos, permitindo ao usuário criar o seu próprio bloco (Seção 5.7);
- **Traduzir:** converte o algoritmo visual em algoritmo textual;
- **Idioma:** suporte multilíngue, permitindo selecionar outros idiomas do software.

- *Painel Esquerdo*. Contém uma lista de blocos separados por categorias e subcategorias (Seção 5.3);

- *Painel Central.* Área de construção do algoritmo visual;

- *Painel Direito*. Contém a tradução textual em Wiring do algoritmo construído no Painel Central;

- *Rodapé*. Contém botões que controlam a disposição dos outros painéis, a saber: ocultar/mostrar e redimensionar os painéis Esquerdo, Direito e Comunicação (Figura 19). 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.

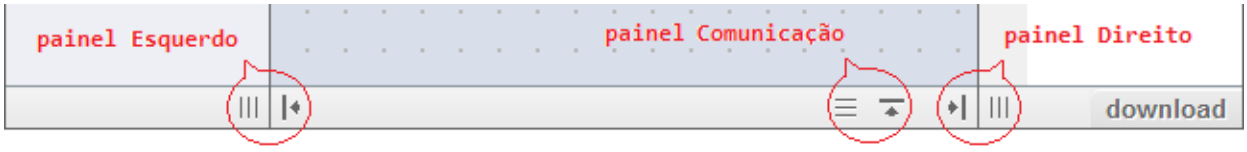


Figura 19: Ícones do Rodapé responsáveis pela manipulação dos paineis Esquerdo, de Comunicação e Direito

Nas seções seguintes, são detalhadas as principais estruturas lógicas da programação do ambiente DuinoBlocks.

5.3 Categorias e Subcategorias dos Blocos

Os blocos de comando estão dispostos em seis categorias (Figura 20). Cada categoria é indicada por uma cor específica, que favorece a localização de um bloco e a identificação da categoria à qual pertence. As cores utilizadas são baseadas no Scratch. Além disso, as categorias possuem subcategorias que proporcionam melhor organização do layout. O agrupamento de blocos em subcategorias não foi verificado nos softwares similares estudados.

Atualmente o DuinoBlocks possui as seguintes categorias e subcategorias:

- **Controle (cor abóbora)**: blocos que realizam repetições e desvios de fluxo. Suas subcategorias são: “Loops” e “Controles de Fluxo”;

- **Operadores (cor verde)**: blocos para construir expressões matemáticas. Contém as seguintes subcategorias de blocos: “Aritméticos”, “Relacionais” e “Lógicos”;

- **Entrada (cor roxa)**: blocos que realizam leituras analógica ou digital de uma porta do hardware Arduino. Possui a subcategoria “Genéricos” que contém blocos que realizam leituras de uma porta, independente do componente conectado. Quando se utiliza um componente de entrada no módulo de Componentes (Seção 5.8), é criada automaticamente uma subcategoria, identificada pelo nome do componente, contendo blocos específicos que o manipulam;

- **Saída (cor azul claro)**: blocos utilizados para realizar escritas analógica ou digital em uma porta do hardware Arduino. Possui a subcategoria “Genéricos” que contém blocos que realizam escritas em uma porta, independente do componente conectado. Nas situações em que é utilizado um componente de saída no módulo de Componentes, cria-se dinamicamente uma subcategoria, cuja nomeação se dá de acordo com o

componente utilizado, contendo blocos específicos para sua manipulação (análogo à categoria Entrada);

- **Utilitários (cor azul escuro)**: blocos para realizar ações extras. Suas subcategorias são: “Comandos Alfanuméricos”, “Comunicação Serial”, “Funções Matemáticas”, “Funções Trigonométricas” e “Outros”;

- **Variáveis (cor vermelha)**: criação de variáveis (Seção 5.6). Suas subcategorias são criadas dinamicamente, uma para cada tipo de variável: “Lógicas”, “Numéricas” e “Alfanuméricas”.

Figura 20: Categorias e Subcategorias

As categorias ainda podem conter a subcategoria “Meus Blocos” gerada automaticamente pelo usuário (Seção 5.7 - Criação de blocos pelo próprio usuário).

5.4 Tipos de Blocos

A forma do bloco indica os possíveis encaixes corretos sintaticamente, determinando também o seu tipo. Existem dois tipos de blocos:

- **Blocos de Pilha** (Figura 21): são passíveis de empilhamento. Possuem um entalhe no topo e uma saliência na base para serem conectados e reunidos em pilhas. Podem ser de dois subtipos diferentes:
    - **Simples**: somente suporta a inserção de outros blocos de pilha no topo ou na base;
    - **Composto**: tem a forma em “C”, permitindo a inserção de outros blocos de pilha, além do topo e da base. Pode ter dois níveis de composição.

Figura 21: Blocos de Pilha

- **Blocos de Retorno** (Figura 22): são utilizados para serem encaixados dentro de outros blocos, não passíveis de empilhamento. Podem ser de três subtipos/formatos diferentes e apenas se encaixam nas cavidades de mesmo formato:
    - **Lógico**: possui extremidades pontiagudas. Devolve um valor lógico (Verdadeiro ou Falso).
    - **Numérico**: possui extremidades arredondadas. Devolve um número (p. ex. blocos da subcategoria Aritméticos na categoria Operadores);

- **Alfanumérico**: possui extremidades quadradas. Devolve caracteres.

Figura 22: Blocos de Retorno[18]

5.5 Entrada de Dados dos Blocos

Alguns blocos possuem entradas de dados. O seu formato é diferente para cada tipo de entrada permitida. Existem quatro tipos de entradas de dados diferentes. O Quadro 1 apresenta um exemplo para cada tipo de entrada de dados:

- **Entrada de Pilha**: uma saliência na base ou um entalhe no topo de um bloco de pilha. Aceita somente encaixe de outros blocos de pilha (empilhamento);
- **Entrada Lógica**: cavidade com extremidades pontiagudas. Aceita valores lógicos (Verdadeiro ou Falso);
- **Entrada Numérica**: cavidade com extremidades arredondadas. Aceita números;
- **Entrada Alfanumérica**: cavidade com extremidades quadradas. Aceita caracteres.

Dependendo do tipo da entrada de dados, seu preenchimento é realizado por algumas das três formas a seguir:

- **Via teclado**: digitando números nas entradas numéricas (Figura 23) ou digitando caracteres nas entradas alfanuméricas (Figura 24).

[18] Nas figuras, a cor dos blocos foi propositadamente especificada em cinza, pois seu tipo/forma independe da categoria a qual pertence.

Figura 23: Entrada numérica

Figura 24: Entrada alfanumérica

- **Via menu de opções**: quando o conjunto de entradas possíveis é conhecido. A seta no canto direito da cavidade (Figura 25) indica que a entrada de dados pode ser preenchida via menu de opções. Ao clicar na seta, um menu é mostrado para escolher a opção desejada (Figura 26).

Figura 25: Entrada de dados via menu

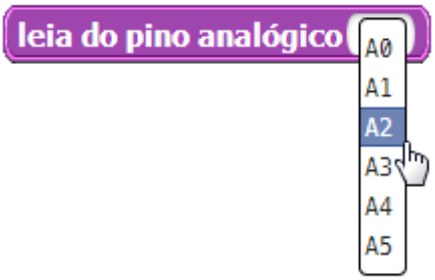
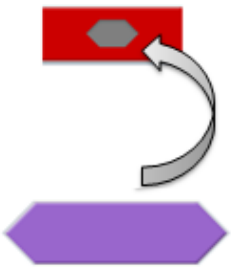
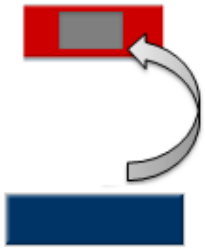


Figura 26: Menu de opções

- **Encaixando outro bloco**: o Quadro 1 mostra os instantes antes, durante e depois do encaixe de diferentes blocos nas correspondentes entradas de dados. Durante o arrasto de um bloco sobre uma entrada correspondente, esta é sinalizada com uma cor cinza, indicando que o encaixe é permitido.

	Entrada de Pilha	Entrada Lógica	Entrada Numérica	Entrada Alfanumérica
Antes				
Durante				
Depois				

Quadro 1: Encaixando blocos

No Scratch é possível inserir um bloco lógico (extremidades pontiagudas) ou numérico (extremidades arredondadas) em uma entrada alfanumérica (cavidade com extremidades quadradas). No DuinoBlocks, estes encaixes não são permitidos, uma vez que, para evitar desentendimento ao realizar encaixes, a correspondência entre blocos e cavidades de mesmo formato é seguida a rigor.

No DuinoBlocks, a inserção de valor lógico ou numérico em uma entrada de dados alfanumérica é feita via utilização de blocos conversores. A Figura 27 mostra os blocos conversores. O bloco I converte um valor numérico em alfanumérico, o bloco II converte um valor lógico em alfanumérico e o bloco III converte um valor alfanumérico em numérico.

Figura 27: Blocos "converte" da subcategoria "Comandos Alfanuméricos" em "Utilitários"

A Figura 28 apresenta um exemplo de conversão. Nela, o bloco numérico "temperatura" é encaixado antes ao bloco "converte" para depois o conjunto ser encaixado à entrada alfanumérica do bloco "escreve no monitor serial".

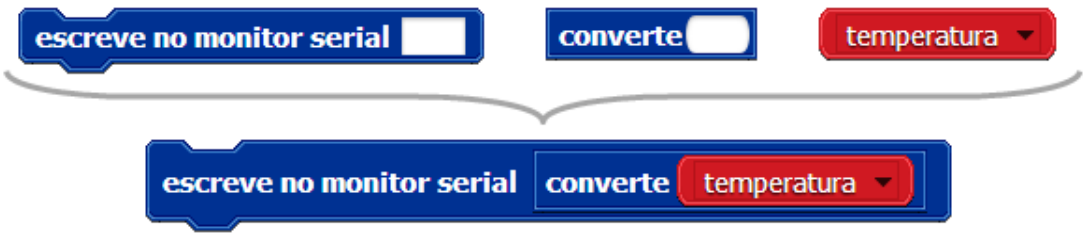


Figura 28: Exemplo de utilização de um bloco conversor numérico para alfanumérico

5.6 Criação de Variáveis

Para criar uma variável (Figura 29) o usuário escolhe, na categoria Variáveis, seu nome (sem se preocupar com regras de nomeação) e seu tipo (lógica, numérica ou alfanumérica).

Figura 29: Criação de variáveis

Ao criar uma variável o ambiente fornece novos blocos: um para obter o seu valor e um ou mais para alterá-lo (Figura 30).

Figura 30: Blocos responsáveis pela manipulação de variáveis numéricas

5.7 Criação de Blocos pelo Usuário

O ambiente permite que o usuário crie novos blocos. Esta funcionalidade abstrai o conceito de procedimento (sub-rotina) de uma linguagem textual. Para tal, o usuário clica no botão “Criar Bloco” e o ambiente abre a janela “Novo Bloco” (Figura 31). Nesta janela, o usuário escolhe um nome para o novo bloco, define sua categoria e tipo. E de forma automática, o ambiente gera uma nova subcategoria chamada “Meus Blocos”, dentro da categoria especificada, contendo o bloco criado.

Figura 31: Criação de Blocos

Em seguida, abre o módulo “Editor de Bloco” (Figura 32) para especificar as ações do novo bloco, como se tivesse criando um novo procedimento.

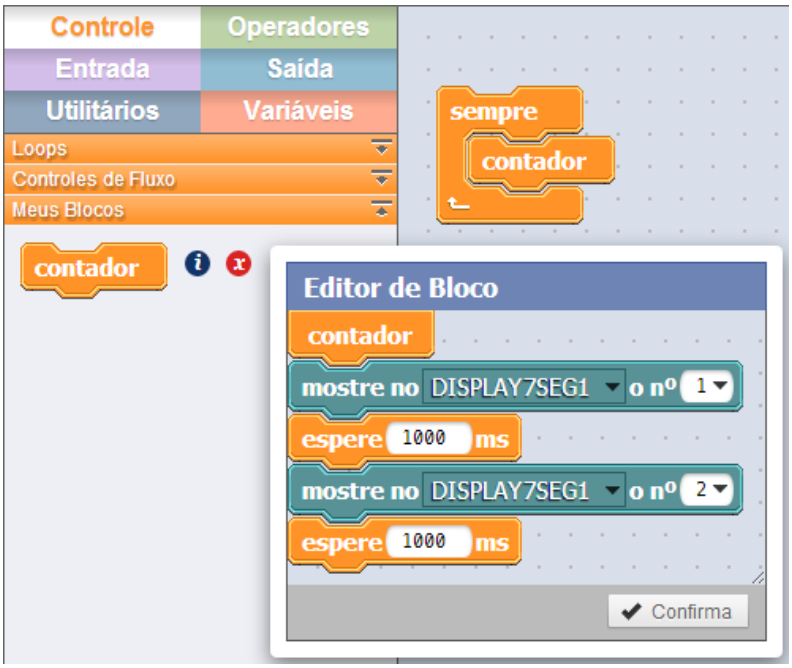


Figura 32: Módulo Editor de Bloco

5.8 Módulo Componentes

O módulo “Componentes” (Figura 33) apresenta um grau de abstração maior ao especializar comandos para determinados componentes de hardware.

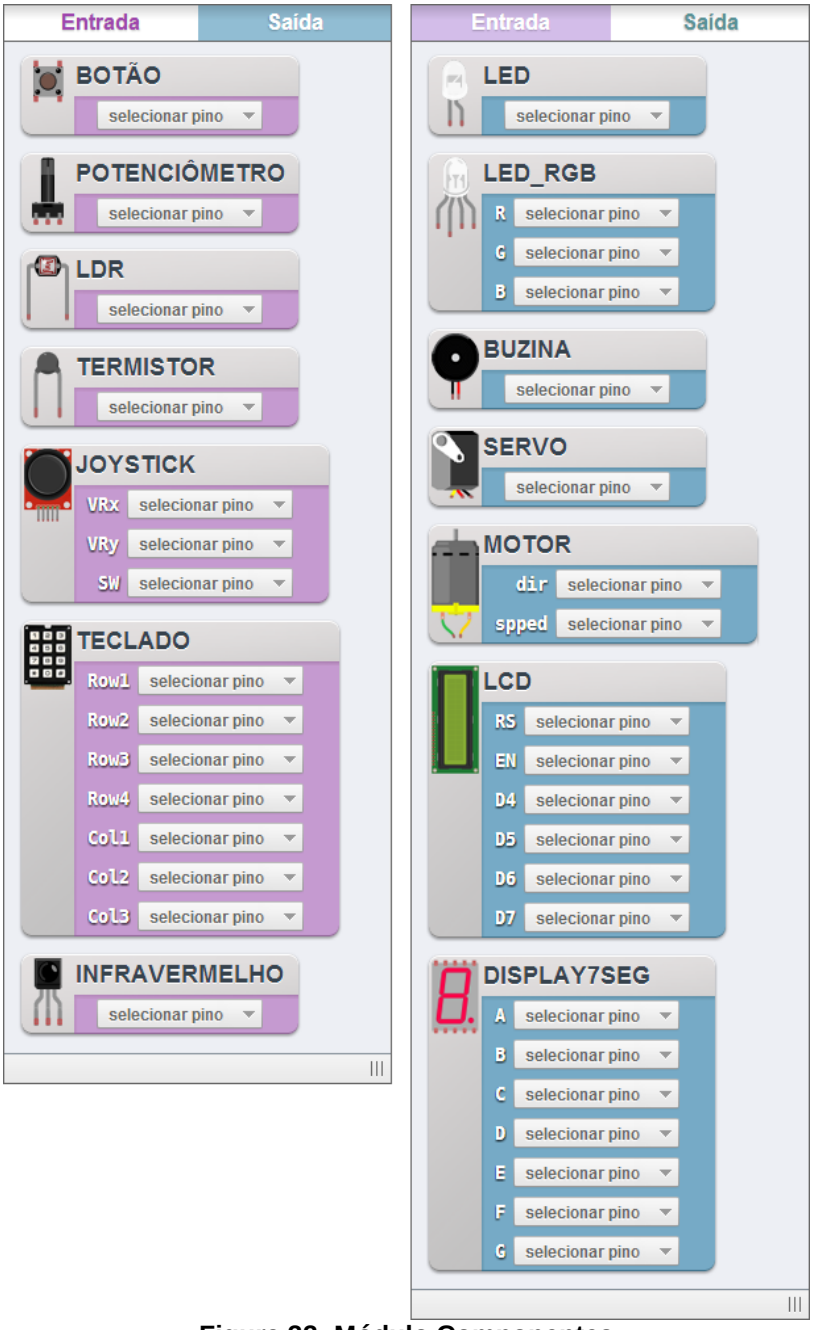


Figura 33: Módulo Componentes

Os componentes inseridos no DuinoBlocks são os mais utilizados nos kits de robótica para iniciantes disponíveis no mercado e estão organizados em duas categorias:

- **Entrada:** contém sensores e outros componentes eletrônicos, que transformam as variáveis do ambiente em sinais elétricos e os fornecem para o hardware Arduino. Atualmente, o ambiente DuinoBlocks possui sete componentes de Entrada: Botão, Potenciômetro, LDR[19], Termistor[20], Joystick, Teclado e Infravermelho;

- **Saída:** contém componentes eletrônicos como atuadores, controlados ou acionados por sinais elétricos oriundos do hardware Arduino. Atualmente, o ambiente DuinoBlocks possui sete componentes de Saída: LED, LED_RGB, Buzina, Servo, Motor, LCD[21], Display numérico.

5.9 Módulo de Comunicação com o Hardware

O Módulo de Comunicação contém os recursos necessários para carregar o código traduzido em Wiring no hardware Arduino diretamente do navegador. O carregamento é realizado através do *Plugin* CodeBender (Figura 34) instalado no navegador e do módulo de comunicação, incorporado no código fonte do DuinoBlocks. Nesta solução, não há a necessidade de instalar um IDE, localizar, atualizar e gerenciar as bibliotecas e os *drivers* corretos, tarefas que demoram e requerem conhecimento técnico da tecnologia.

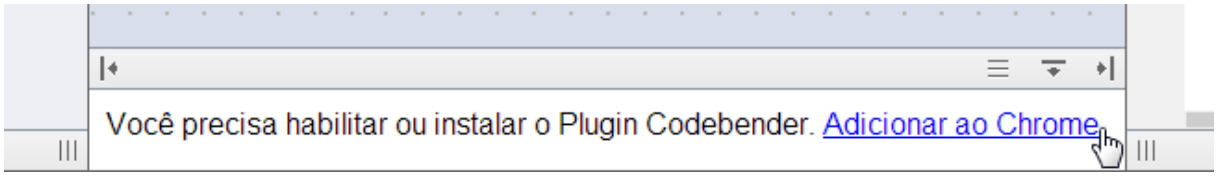


Figura 34: Instalação do *Plugin* CodeBender

Depois de instalar o *plugin*, são disponibilizados no layout do painel de Comunicação (Figura 35) recursos para o usuário realizar os casos de usos “Informar Porta” (Tabela 8) e “Carregar Remotamente” (Tabela 4). O caso de uso “Informar Placa” (Tabela 7) foi omitido, sendo necessário desenvolver o módulo de Hardware para disponibilizar esta funcionalidade. O Arduino UNO, modelo de placa mais utilizado, é selecionado por *default*.

19 LDR (*Light Dependent Resistor*) – sensor de luminosidade
20 Termistor – sensor de temperatura
21 LCD (*Liquid Crystal Display*) – display de caracteres

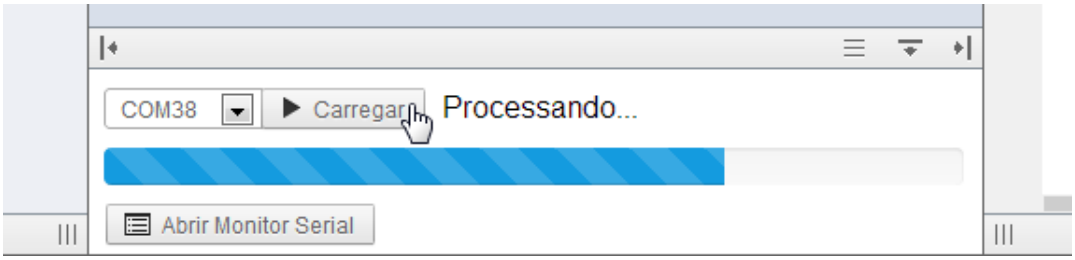


Figura 35: Carregamento do hardware Arduino

A Figura 36 mostra o monitor serial, usado para a comunicação entre a placa Arduino e o computador. A taxa de transmissão foi fixada em 9600 bps.

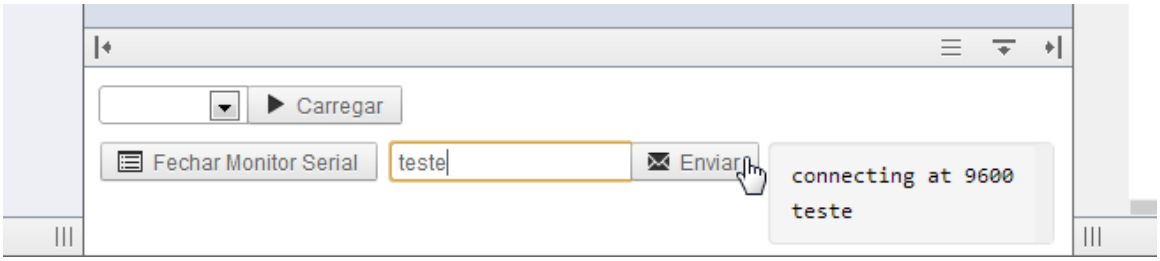


Figura 36: Monitor serial

5.10 Troca de Operador

No processo de elaboração de um algoritmo, é comum criar, testar, depurar, alterar e tornar a testar uma expressão matemática diversas vezes até se chegar no resultado desejado. Nas linguagens textuais a simples troca de um operador em uma expressão é uma ação rápida. Ao criar os primeiros programas no DuinoBlocks percebeu-se que esta simples ação se mostrava dificultosa na linguagem visual, pois era necessário remover um bloco e adicionar outro para alterar um operador.

Para resolver este problema, foi adicionada uma funcionalidade para que fosse possível trocar o operador de um bloco, sem a necessidade de removê-lo. Ao clicar no operador, é aberto um popup contendo os operadores equivalentes para que o usuário possa escolher o operador substituto (Figura 37).

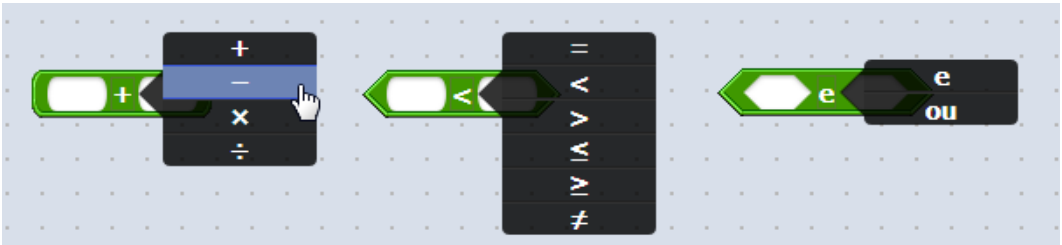


Figura 37: Troca de operadores

Vale ressaltar a preocupação em aproximar-se do modelo mental do usuário [JOHNSON-LAIRD, 1983], tonando familiar os símbolos das operações, ao utilizar exatamente a representação padrão matemática. Diferentemente de outros softwares, como o Scratch, que identificam as operações de multiplicação com um asterisco "*" e divisão com uma barra "/", o DuinoBlocks procura utilizar representações do conhecimento de seu público-alvo.

5.11 Outras Funcionalidades

A fim de se explorar ao máximo as características visuais da linguagem do DuinoBlocks, optou-se por apresentar as funcionalidades de "Tratamento de Erro" (Figura 38) e de "Ajuda" (Figura 39) também na forma visual.

Figura 38: Tratamento de Erro

A Figura 38 mostra uma janela informando ao usuário que o bloco "se" possui um dado ausente. Aqui o erro é tratado como ausência de dados. Esta janela é aberta após o usuário clicar em "traduzir".

A Figura 39 mostra uma ajuda para o bloco "sempre". Esta ajuda é apresentada clicando no ícone " ***i*** " (Figura 40) ao lado do bloco no painel Esquerdo.

Figura 39: Ajuda do bloco “sempre”

Figura 40: Ajuda

5.12 Arquitetura 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 (2013) 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.

5.12.1 Desenho dos Blocos

A técnica de desenho dos blocos foi baseada no ambiente de programação para microcontroladores ModKit. A Figura 41 mostra as partes que compõem um bloco do tipo pilha.

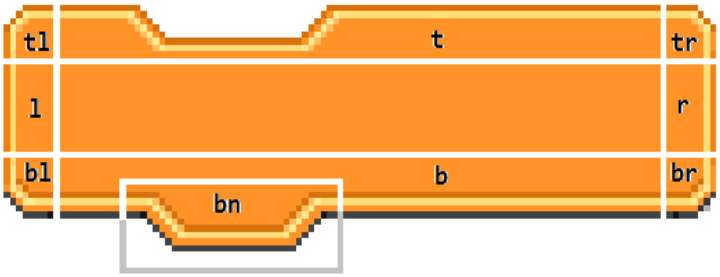


Figura 41: Estrutura de um bloco do tipo pilha

Este padrão de estrutura, que consiste em dividir os blocos em quadrantes, permite desenhar todos os tipos de blocos da linguagem apenas modificando

poucos parâmetros. Outro aspecto importante desta técnica é a facilidade proporcionada nos cálculos de redimensionamento dos blocos. A Figura 42 mostra o código HTML necessário para desenhar um bloco do tipo pilha de cor laranja.

```
▼<div class="block statement orange">
   <div class="t"></div>
   <div class="l"></div>
   <div class="r"></div>
   <div class="tl"></div>
   <div class="tr"></div>
   <div class="b"></div>
   <div class="br"></div>
   <div class="bn"></div>
   <div class="bl"></div>
 </div>
```

Figura 42: Código HTML de um bloco do tipo pilha

Cada parte do bloco é representada por uma div HTML e uma classe CSS. A Figura 43 e a Figura 44 mostram, respectivamente, o código CSS do posicionamento e do dimensionamento das partes de um bloco do tipo pilha.

```
.tl, .t, .tr,
.bl, .b, .br,
.l , .r, .bn {position:absolute;}
.tl, .t, .tr {top:    -8px;}
.bl, .b, .br {bottom:-8px;}
.tl, .l, .bl {left:   -8px;}
.tr, .r, .br {right: -8px;}
.l, .r {top:0px;}
.bn {left:14px; bottom:-13px;}
```

Figura 43: Código CSS do posicionamento de cada parte de um bloco do tipo pilha

```
.tl, .l, .bl, .tr, .r, .br {width:8px;}
.tl, .t, .tr, .bl, .b, .br {height:8px;}
.l , .r {height:100%;}
.t , .b {width:100%;}
.bn {height:12px; width:30px;}
```

Figura 44: Código CSS do dimensionamento de cada parte de um bloco do tipo pilha

A Figura 45 mostra o código CSS responsável pela formatação do plano de fundo de cada parte de um bloco do tipo pilha. Por sua vez, a Figura 46 mostra a imagem (orange.png) que contém o plano de fundo de cada parte de um bloco qualquer de cor laranja.

```
.block.orange {background-color:#f90;}
.orange.tl, .orange.t, .orange.tr,
.orange.l ,            .orange.r ,
.orange.bl, .orange.b, .orange.br,
.orange.bn {background-image:url('orange.png');}
.tl {background-position:    0px   0px;}
.t  {background-position:  -26px   0px;}
.tr {background-position: -990px   0px;}
.l  {background-position:-1000px   0px;}
.r  {background-position:-1012px   0px;}
.bl {background-position:    0px -12px;}
.b  {background-position:  -11px -12px;}
.br {background-position: -990px -12px;}
.bn {background-position:  -15px -41px;}
```

Figura 45: Código CSS do plano de fundo de cada parte de um bloco do tipo pilha

Figura 46: Imagem contendo o plano de fundo de todas as partes de um bloco qualquer de cor laranja

5.12.2. Estrutura Lógica dos Empilhamentos

A Figura 47 mostra um exemplo de pilha de blocos. No diagrama da Figura 48 é apresentada como esta pilha fica estruturada logicamente no código fonte. Esta estrutura é determinante nas ações de encaixe e desencaixe de blocos e na tradução do algoritmo visual em textual.

Os blocos de pilhas possuem referências para o bloco encaixado em seu topo (*SiblingUp*) e o bloco encaixado em sua base (*SiblingDown*). Os argumentos dos comandos (*Arg*) e os filhos (*Child*) de um bloco composto são mantidos em uma lista de referências. Por sua vez, cada bloco possui uma referência ao bloco em que está encaixado (*Parent*). E os blocos que manipulam uma variável guardam a sua referência (VarName).

Figura 47: Exemplo de uma pilha de blocos

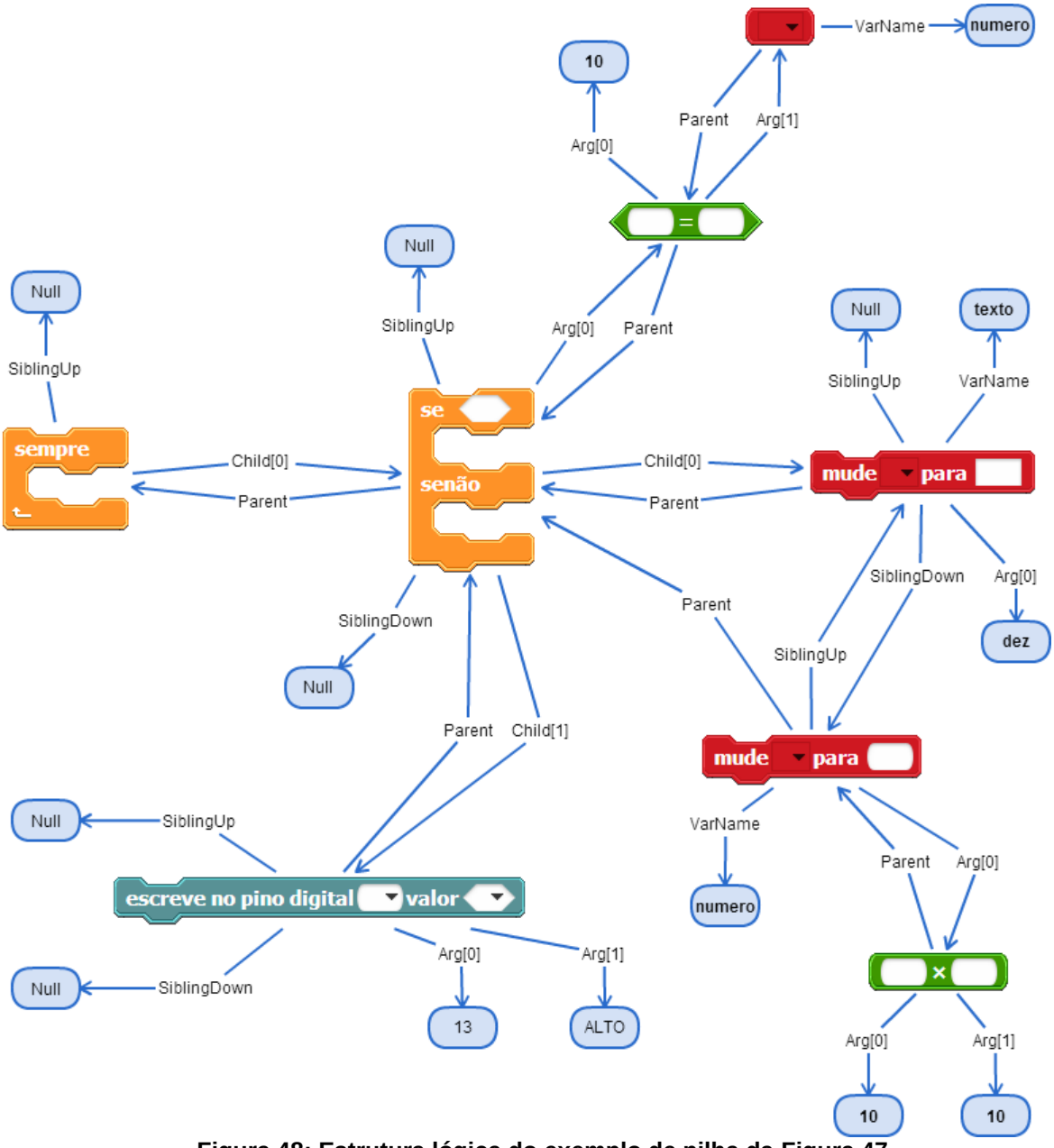


Figura 48: Estrutura lógica do exemplo de pilha da Figura 47

5.12.3. Projeto DuinoBlocks

A Figura 49 mostra o diagrama de classes resumido do projeto DuinoBlocks. 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).

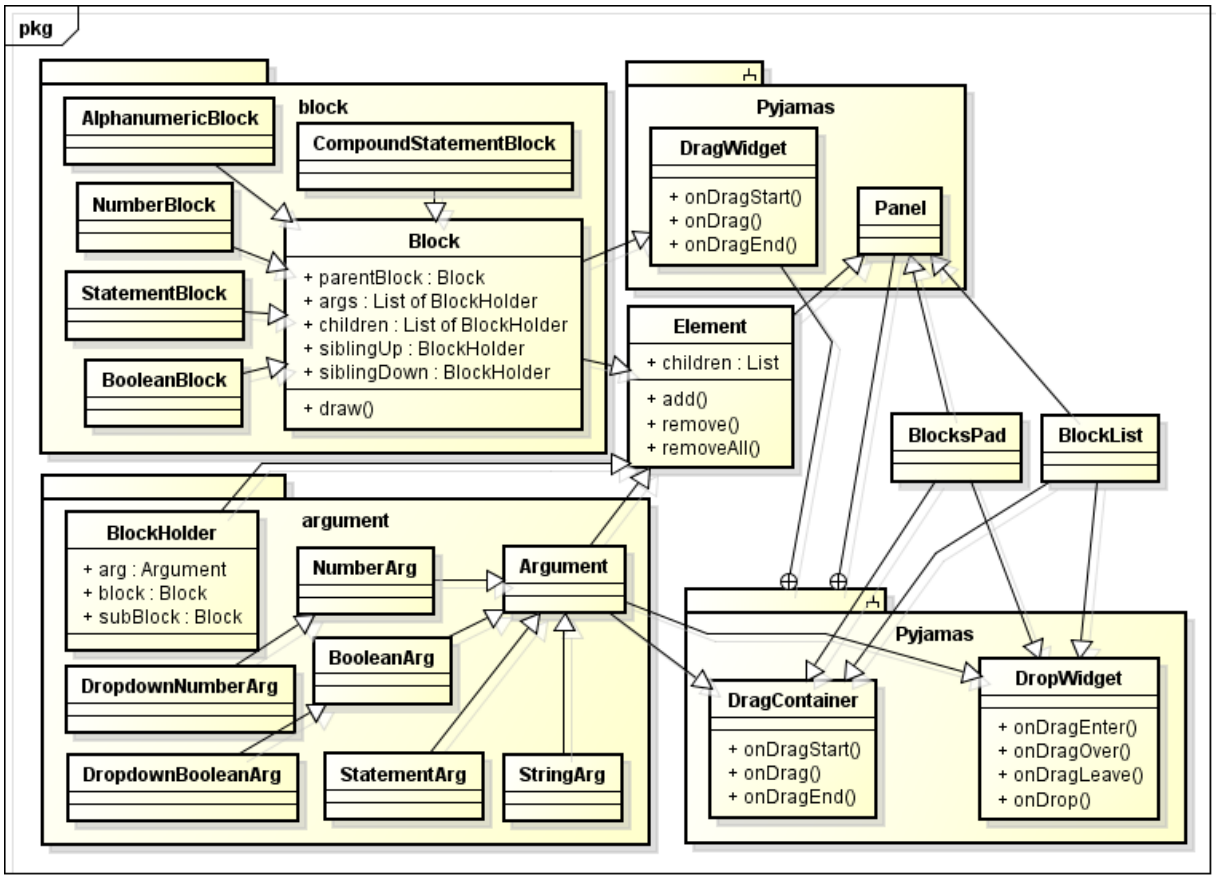


Figura 49: Diagrama de Classes Resumido do DuinoBlocks

5.13 Ambiente de Desenvolvimento

No desenvolvimento do DuinoBlocks, foi priorizado a utilização de softwares livres. O Quadro 2 apresenta os recursos utilizados no desenvolvimento do ambiente.

Recurso	Finalidade	End. 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 *translate* do código fonte 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.

5.14 Executando o DuinoBlocks no Modo Off-line

A Figura 50 mostra o erro que é gerado ao abrir o arquivo HTML do ambiente DuinoBlocks localmente com o navegador Chrome.

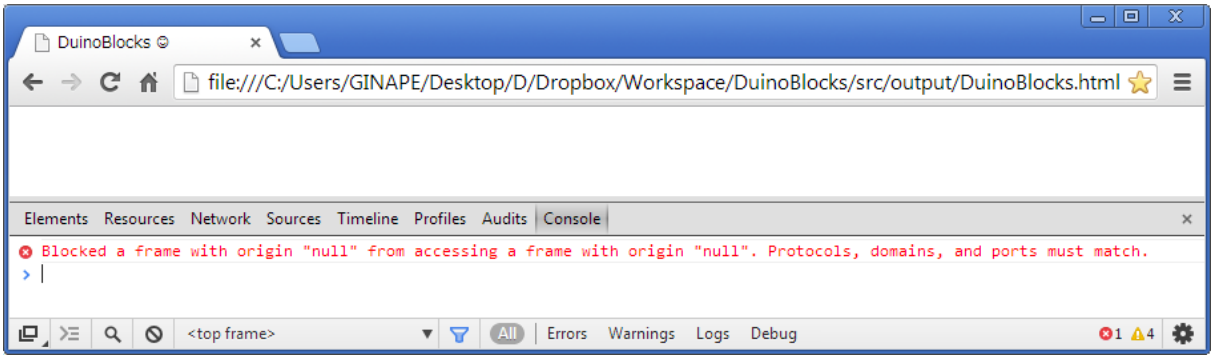


Figura 50: Erro ao executar o DuinoBlocks com o Chrome

Este erro é decorrente de uma proteção do Chrome que impede que a *tag iframe* inserida pela biblioteca Pyjamas no HTML da página, acesse outros arquivos

como, por exemplo, as imagens e os estilos CSS. Este problema não ocorre ao abrir o arquivo HTML do ambiente DuinoBlocks localmente utilizando os navegadores Firefox e Internet Explorer.

A solução para contornar o problema foi criar um arquivo BAT com um código que abre o navegador com a proteção desabilitada. A Figura 51 mostra o código Shell Script do arquivo BAT necessário para executar o DuinoBlocks com o navegador Chrome no sistema operacional Windows. A primeira linha indica o caminho do diretório onde o Chrome está instalado. A segunda linha indica o caminho do arquivo HTML do DuinoBlocks. A terceira significa que o Chrome será aberto com a diretiva especificada habilitada.

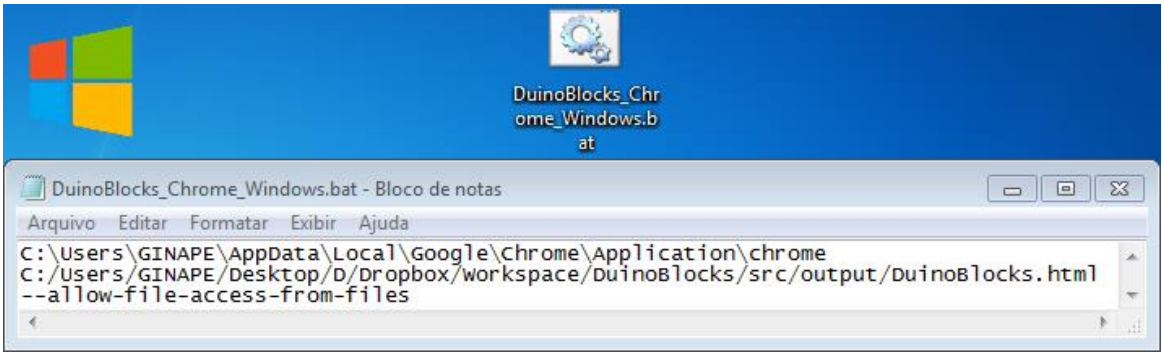


Figura 51: Código para executar o DuinoBlocks no modo off-line com o Chrome no Windows

Analogamente na Figura 52 mostra o código Shell Script do arquivo SH necessário para executar o DuinoBlocks com o navegador Chromium no sistema operacional Meego. A 1º linha indica o caminho do diretório onde o Chromium está instalado. A 2º linha indica o caminho do arquivo HTML do DuinoBlocks. A 3° significa que o Chromium será aberto com a diretiva especificada habilitada.

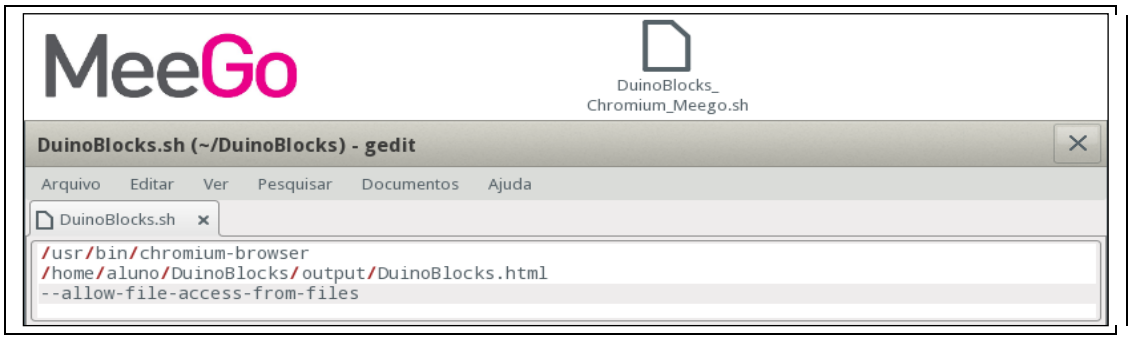


Figura 52: Código para executar o DuinoBlocks no modo off-line com o Chromium no Meego

6 AVALIAÇÃO DO DUINOBLOCKS

Para avaliar a proposta do DuinoBlocks e colher subsídios sobre a viabilidade de uso do ambiente, foram realizadas experimentações com o público-alvo em dois momentos. No primeiro, o DuinoBlocks foi avaliado em oficina de REBC (Seções 6.1 e 6.2) e no segundo em curso de formação em RE (Seções 6.3 e 6.4).

6.1 Avaliação do DuinoBlocks em Oficina de REBC

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. Os participantes da oficina foram 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 à 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 (Figura 53) fornecido pelo projeto Uca na Cuca. Em termos práticos, as aulas abordavam a elaboração de experimentos com montagem de componentes e a sua programação.

Figura 53: Kit Arduino para iniciante da loja virtual RoboCore

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 *et al*., 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 (Figura 54), - muitas vezes 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 a ajuda dos instrutores.

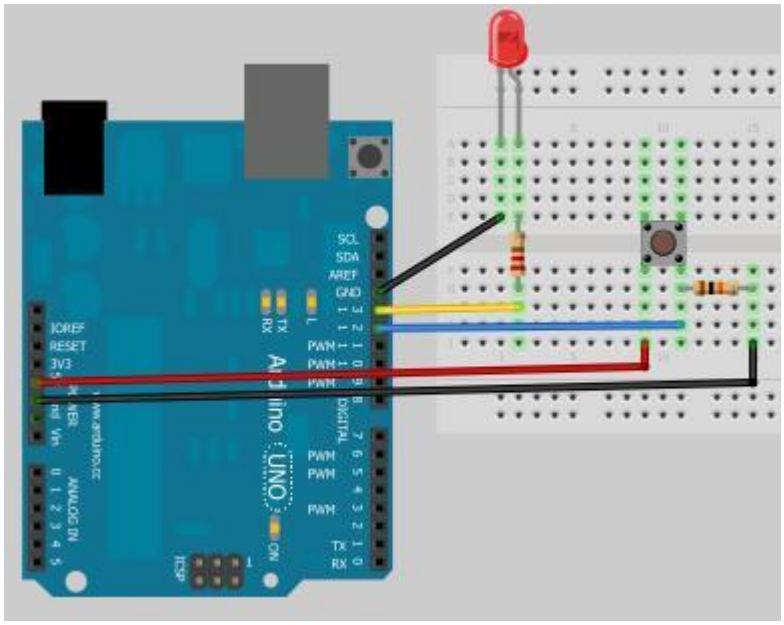


Figura 54: Montagem física de um experimento utilizado na Oficina

6.1.1 Resultados da Avaliação do DuinoBlocks em Oficina de REBC

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. Numa segunda etapa, de caráter semi-exploratória, os participantes eram solicitados a fazer alterações no algoritmo construído. No terceiro e último momento, já ao final do curso, os desafios de programação propostos eram elaborados sem ajuda dos instrutores.

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 leva em conta as coordenadas do ponteiro do mouse. Assim, há situações em que arrastar uma fenda de um bloco até a saliência de outro, não necessariamente resulta em um encaixe. A Figura 55 ilustra a tentativa, sem sucesso, de encaixar o bloco “espere” dentro do bloco “sempre”. Apesar da fenda de um bloco estar próxima da saliência do outro, o encaixe não ocorrerá ao soltar o bloco arrastado pela sua extremidade direita.

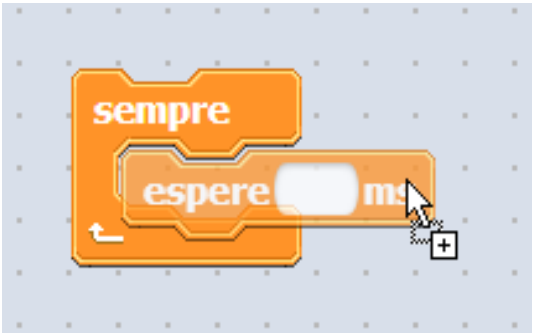


Figura 55: Tentativa, sem sucesso, de encaixar dois blocos

Já na Figura 56 é ilustrado onde deve ser posicionado o ponteiro do mouse em relação ao bloco “sempre” ao arrastar o bloco “espere” pela extremidade direita. Este movimento, levar o ponteiro do mouse até uma fenda ou entalhe, não é natural para o usuário, mas sim, aproximar uma fenda a um entalhe conforme acontece na Figura 55.

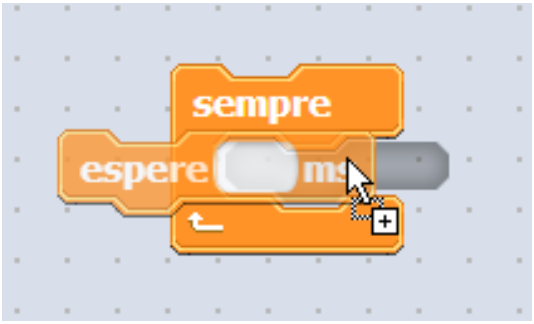


Figura 56: Encaixe não intuitivo de dois blocos

A fim de auxiliar o usuário no processo de encaixe de blocos decidiu-se apresentar uma área cinza, fornecendo um feedback visual para indicar uma região passível de encaixe e também emitir um som, produzindo um feedback auditivo para

indicar a realização de um encaixe. Contudo, ainda se faz necessário realizar uma atualização no DuinoBlocks que modifique a forma de encaixe dos blocos, deixando esta ação mais intuitiva.

Ainda durante a elaboração dos algoritmos foi possível perceber que os participantes não tentaram fazer encaixes impossíveis, demonstrando que a lógica necessária para realizar encaixes (correspondência entre blocos e cavidades de mesmo formato) foi entendida.

No que se refere à organização dos blocos, o agrupamento dos mesmos em categorias e em subcategorias e a utilização de cores para cada uma delas, ajudaram na localização dos blocos, visto que, nenhum cursista demonstrou dificuldades para encontrar um bloco representante de uma ação (comando) específica.

No último encontro da Oficina (após duas semanas de intervalo) os participantes foram solicitados a "pensar em voz alta" (técnica *think aloud* [van SOMEREN *et al*., 1994]) sobre o propósito dos algoritmos apresentados em Wiring e em DuinoBlocks. Ao discursarem sobre os programas em Wiring, conseguiam relatar o objetivo de algumas partes isoladas 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 mais completo sobre a ação a ser executada.

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 cursistas 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 Avaliação do DuinoBlocks em Curso de Formação em RE

O projeto Uca na Cuca vem promovendo no município de Piraí-RJ cursos de capacitação de professores do ensino fundamental e médio intitulado "Formação em Robótica Educacional com Hardware Livre" [SAMPAIO e ELIA, 2011]. Tais cursos

têm caráter teórico-prático e a aplicação dos mesmos forneceram importantes subsídios para a especificação do ambiente DuinoBlocks, como apresentado no Capítulo 4 (Requisitos do Sistema).

Nas suas primeiras versões foi utilizado apenas o IDE Arduino, com sua programação textual em linguagem Wiring, para a implementação dos projetos em RE. Na terceira turma do curso foi introduzido o ambiente de programação visual DuinoBlocks, com o propósito de avaliar, principalmente, a sua usabilidade.

O curso iniciou 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:

1. Introdução a Robótica Educacional, Eletrônica Básica e Apresentação do Arduino;

2. Programação Textual Wiring.

As atividades da terceira aula 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 última aula do curso foram realizados novos experimentos utilizando o módulo de componentes do ambiente (ver Seção 5.8).

Nas Tabelas 12, 13 e 14 estão descritas as atividades criadas para verificar a capacidade de raciocinar sobre a criação de um algoritmo a partir da adoção do DuinoBlocks. Nestas três atividades são descritos o objetivo, com uma breve contextualização, os materiais utilizados, a montagem dos componentes e o algoritmo de uma provável resolução.

Tabela 12: Atividade 1 – Acender LED

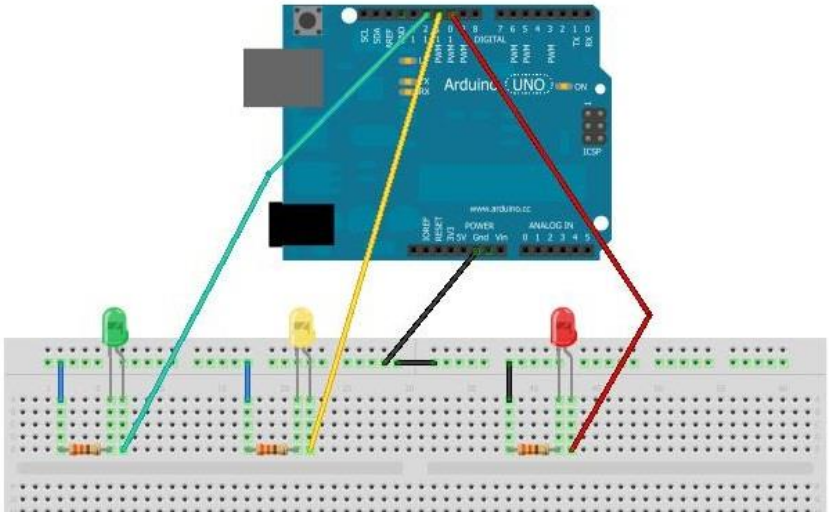
Objetivo	Simular o funcionamento de um semáforo de trânsito.
Material	3x LED (1 verde, 1 amarelo e 1 vermelho); 3x resistores de 330 ohms; Arduino, Protoboard e Fios condutores.
Montagem	
Algoritmo	

Tabela 13: Atividade 2 – Acionar uma Buzina com um Potenciômetro

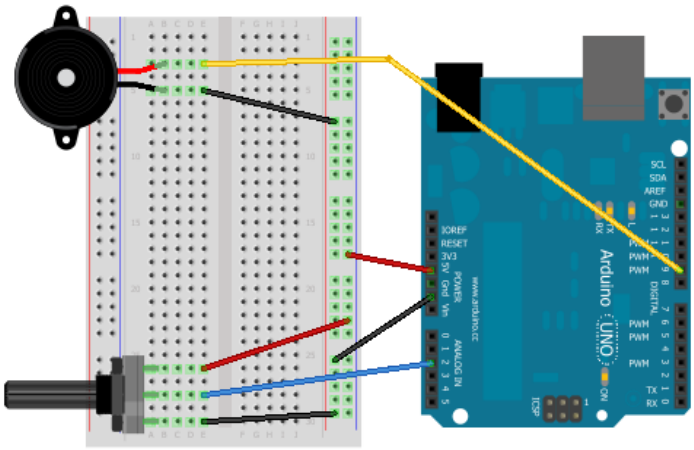
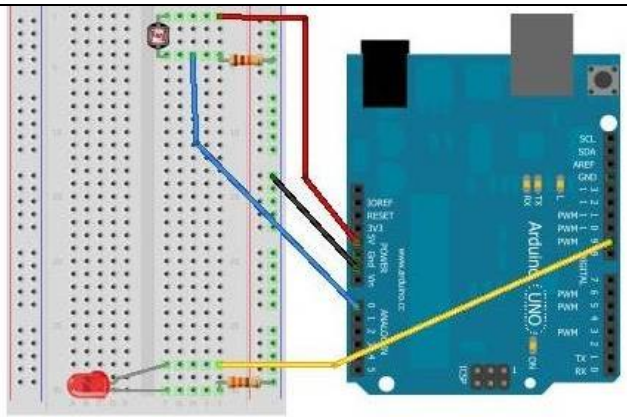
Objetivo	Simular a regulagem do volume de um rádio.
Material	1x Potenciômetro; 1x Buzina; Arduino, Protoboard e Fios condutores.
Montagem	
Algoritmo	

Tabela 14: Atividade 3 – Ligar um LED com um LDR

Objetivo	Simular o acionamento automático da iluminação pública quando anoitece.
Material	1x LED; 1x LDR; 2x resistores de 330 ohms; Arduino, Protoboard e Fios condutores.
Montagem	
Algoritmo	

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

Em comparação com as turmas anteriores, os projetos aqui executados foram construídos com uma curva de aprendizagem menor em relação ao ambiente de programação, sendo possível realizar experimentos mais complexos a partir da adoção do DuinoBlocks. Experimentos que não eram oportunos nas turmas anteriores como utilizar um display de caracteres, servo e teclado foram possíveis de serem realizados devido a abstração de complexidade dos comandos no DuinoBlocks.

Na atividade descrita na Tabela 12, um grupo de professores chamou a atenção ao construiu uma solução mais elaborada do que a esperada. Além dos comandos básicos para simular o funcionamento de um semáforo de trânsito, o algoritmo continha um trecho reponsável por fazer o LED amarelo ficar piscando. Este fato demonstrou que aqueles usuários sem conhecimento de programação foram capazes de utilizar estruturas lógicas de repetição (*loops*) sem a necessidade de explicações prévias.

Na atividade da Tabela 13 foi necessário fornecer uma explicação técnica para os grupos realizarem corretamente a divisão por quatro contida na solução. Nas leituras analógicas é obtido um valor entre 0 e 1023 e nas escritas analógicas é atribuido um valor entre 0 e 255, ou seja, são 1024 valores possíveis na leitura e 256 (quatro vezes menor) na escrita. E na atividade da Tabela 14 um grupo solicitou informações extras e foi fornecido uma explicação em pseudo-código (ex.: se esta escuro então ascende senão apaga) para ajudar na construção a solução.

O grupo com integrantes que tinham conhecimento em programação realizou as três atividades sem nenhuma dificuldade e em tempo menor que os outros. Eles utilizaram variáveis nas soluções e esta funcionalidade não havia sido explicada. Para estas atividades foi possível perceber que o DuinoBlocks não apresentou limitações mesmo para usuários de ambientes textuais de programação.

O módulo de comunicação com o hardware (Seção 5.10) que permite o usuário carregar um programa escrito em DuinoBlocks diretamente no hardware

Arduino ainda estava sendo desenvolvido. A necessidade desta funcionalidade foi priorizada após a oficina de REBC (Seção 6.1) e confirmada durante esta avaliação, pois foi possível perceber que os participantes demonstravam dificuldades em copiar o código textual traduzido pelo DuinoBlocks para colar no IDE Arduino, ação que prejudicava a fluidez da experimentação.

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.

7 CONCLUSÕES E TRABALHOS FUTUROS

Este capítulo apresenta as conclusões da pesquisa, tecendo as considerações finais e propondo trabalhos futuros.

7.1 Conclusões

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 é 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 demostrando sua acessibilidade aos usuários e apesar de não terem habilidades técnicas, ainda são capazes de criar protótipos com um mínimo de esforço.

O fato de o software ser escrito em Python, utilizando a biblioteca Pyjamas trouxe grandes vantagens em relação à programação da interface. A principal desvantagem é a ausência de um suporte adequado para a importação de outras bibliotecas escritas em Javascript, sendo necessário criar soluções improvisadas para estender as funcionalidades do software.

Até a segunda turma do curso promovido pelo projeto Uca na Cuca, os laptops Classmate não continham o software Arduino na lista de programas e o mesmo era instalado separadamente. Nesta solução, o Arduino era instalado e executado por linhas de comando através do terminal do sistema operacional, ações que dificultavam a utilização do hardware em sala de aula pelos professores. Após uma parceria entre as equipes do projeto e a empresa Metasys, foi gerada uma requisição para que o Arduino viesse instalado no Meego, fazendo parte dos programas carregados juntos com o sistema operacional. Com esta solução, o software passou a ser executado, mais facilmente, com um duplo clique em seu ícone. O Anexo 1 contém o tutorial, utilizado na terceira turma do curso, com os passos da instalação do Arduino na lista de programas do Meego.

Para que as funcionalidades que utilizam recursos de HTML5, como Abrir/Salvar, pudessem funcionar, foi necessário atualizar o navegador Chromium nos laptops Classmates. A atualização do navegador avançou da versão 11 para a versão 25. O Apêndice 1 contém o tutorial com os passos da atualização do Chromium no sistema operacional Meego.

7.2 Artigos Acadêmicos

- **Uso do hardware livre Arduino em ambientes de ensino-aprendizagem** – Minicurso apresentado na Jornada de Atualização em Informática e Educação (JAIE), evento integrante do Congresso Brasileiro de Informática da Educação (CBIE) [ALVES *et al*., 2012];

- **DuinoBlocks: Um Ambiente de Programação Visual para Robótica Educacional** – Artigo completo apresentado na Seminário Integrado de

Software e Hardware (SEMISH), evento integrante do Congresso da Sociedade Brasileira de Computação (CSBC) [ALVES *et al*., 2013].

7.3 Trabalhos Futuros

Durante o desenvolvimento do DuinoBlocks, percebeu-se que haveria diversas funcionalidades relevantes ao projeto inicial, porém que fugiriam do escopo e/ou do tempo disponível para implementação. Dentre as funcionalidades previstas que não foram desenvolvidas, são priorizadas:

- O desenvolvimento de funcionalidades que permitam salvar/abrir programas na nuvem;
- O desenvolvimento de uma comunidade web em torno do ambiente DuinoBlocks voltada ao incentivo do compartilhamento de programas;
- Disponibilizar o vocabulário de palavras em site gerenciador de tradução colaborativa para que interessados, ao redor do mundo, possam contribuir com a tradução em sua língua;
- Liberar o projeto em repositório de código fonte online para que outros desenvolvedores possam criar novas funcionalidades, além de identificar e corrigir *bugs*;

Atualmente os esforços têm se concentrado na correção de pequenos problemas de usabilidade detectados durante os experimentos. As melhorias previstas a serem implementadas são:

- Tornar a forma de encaixe dos blocos intuitiva;
- Desfazer/Refazer mais eficiente, pois quando contém muitos blocos esta ação é demorada;
- Compatibilidade do layout com o navegador Firefox;
- Implementar o módulo de hardware, permitindo utilizar outros modelos de placa Arduino, além do Uno.

- Concluir as imagens do módulo de ajuda, exemplificando a utilização de cada bloco.

Como outra vertente de pesquisa, propõe-se explorar as possibilidades pedagógicas do ambiente DuinoBlocks, testando-o em sala de aula com os alunos a fim de avaliar a efetividade do aprendizado de conceitos curriculares contextualizados através da robótica.

REFERÊNCIAS

ALBUQUERQUE, A. P. ; MELO, C. M. ; CÉSAR, D. R. ; MILL, D. Robótica pedagógica livre: instrumento de criação, reflexão e inclusão sócio-digital. In: SIMPÓSIO BRASILEIRO DE INFORMÁTICA NA EDUCAÇÃO,18., 2007. São Paulo, **Anais …** São Paulo: CEIE, 2007.

ALVES, R. M. ; SILVA, A. L. C. ; PINTO, M. C. ; SAMPAIO, F. F. ; ELIA, M. F. Uso do hardware livre Arduino em ambientes de ensino-aprendizagem. In: CONGRESSO BRASILEIRO DE INFORMÁTICA DA EDUCAÇÃO, 2012, Rio de Janeiro; JAIE - JORNADA DE ATUALIZAÇÃO EM INFORMÁTICA E EDUCAÇÃO, 2012, Rio de Janeiro. **Anais...** Rio de Janeiro: CEIE, 2012.

ALVES, R. M. ; SAMPAIO, F. F. ; ELIA, M. F. DuinoBlocks: um ambiente de programação visual para robótica educacional. In: CSBC - XXXIII CONGRESSO DA SOCIEDADE BRASILEIRA DE COMPUTAÇÃO, 33., 2013. Maceió; SEMISH - SEMINÁRIO INTEGRADO DE SOFTWARE E HARDWARE, 40., 2013, Maceió. **Anais …** Maceió: SBC, 2013.

ARDUBLOCKGROUP. Disponível em <https://groups.google.com/forum/#!forum/ardublock>. Acesso em: nov. 2013.

ARDUINO. Disponível em: <http://arduino.cc/playground/Portugues/HomePage>. Acesso em: nov. 2013.

ATMEL AVR. Disponível em: <http://www.atmel.com/products/microcontrollers/avr/>. Acesso em: nov. 2013.

AURELIANO, V. C. O. ; TEDESCO, P. C. A. R. Avaliando o uso do Scratch como abordagem alternativa para o processo de ensino-aprendizagem de programação. In: XXXII CONGRESSO DA SOCIEDADE BRASILEIRA DE COMPUTAÇÃO, 32., 2012, Curitiba. **Anais …** Curitiba: SBC, 2012.

BARANAUSKAS, M. C. C. ; SOUZA, C. S. Desafio nº 4: acesso participativo e universal do cidadão brasileiro ao conhecimento. **Computação Brasil**, Porto Alegre, ano 7, 2006.

BARBERO, A.; DEMO, B. ; VASCHETTO, F. A contribution to the discussion on informatics and robotics in secondary schools. In: INTERNATIONAL CONFERENCE ON ROBOTICS IN EDUCATION, 2., 2011, Vienna. **Proceedings …** Vienna: Austrian Society for Innovative Computer Sciences, 2011.

BORKULO , S. P. van. **The assessment of learning outcomes of computer modelling in secondary science education**. 2009. Thesis (D.Sc) – University of Twente, Enschede, 2009.

BRANDÃO, L. O. ; BRANDÃO, A. A. F. ; RIBEIRO, R S. iVProg – uma ferramenta de programação visual para o ensino de algoritmos. In: CONGRESSO BRASILEIRO DE INFORMÁTICA NA EDUCAÇÃO. 2012, Rio de Janeiro; WORKSHOP DE INFORMÁATICA NA ESCOLA, 18., 2012, Rio de Janeiro. **Anais...** Rio de Janeiro: SBC, 2012.

BRASIL. MINISTÉRIO DA EDUCAÇÃO. **GTE Guia de Tecnologias Educacionais**. Brasília, Disponível em: <http://portal.mec.gov.br/arquivos/pdf/posconjur1.pdf>. Acesso em: nov. 2013.

BRASIL. MINISTÉRIO DA EDUCAÇÃO. SECRETARIA DE EDUCAÇÃO BÁSICA. COGETEC. **Guia de tecnologias educacionais 2011/2012**. Brasília: MEC, 2011. 196 p. Organização COGETEC. Disponível em: <http://portaldoprofessor.mec.gov.br/pdf/guias_2011_Web.pdf>. Acesso em: nov. 2013.

BRINK MOBIL. Disponível em: <http://www.brinkmobil.com.br>. Acesso em: nov. 2013.

CHELLA, M. T. SimRobô: simulador para robótica com propósito educacional. In: WORKSHOP DE ROBÓTICA EDUCACIONAL, 3., Fortaleza, 2012. **Anais …** Fortaleza: SBC, 2012.

CLASSMATE. Disponível em: <http://www.instituicaodoseculo21.com.br/classmate.html>. Acesso em: nov. 2013.

COHEN, L. ; MANION, L. ; MORRISON, K. **Research methods in education**. 5. ed. London: Taylor & Francis e-Library, 2005. Disponível em: <http://www.mums.ac.ir/shares/nurse/nurse_coll/EDO/manabe%20Amuzeshi/research%20method%20in%20education.pdf>. Acesso em: nov. 2013.

CRINNION, J. **Evolutionary systems development:** a practical guide to the use of prototyping within a structured systems methodology. London: Pitman Publishing, 1991.

CRUZ, M. K. ; HAETINGER, W. ; HORN, F. ; CARVALHO, D. V. ; ARAÚJO, G. H. Controle de kit de robótica através de laboratório remoto pela internet: uma aplicação para a formação docente e para a educação básica. In: SIMPÓSIO BRASILEIRO DE INFORMÁTICA NA EDUCAÇÃO, 20., 2009, Florianópolis, **Anais …** Florianópolis: SBC, 2009.

FERNANDES, C. **Um simulador de ambiente de robótica educacional em plataforma virtual**. 2013. Dissertação (Mestrado em Ciências) – Programa de Pós-Graduaçaõ em Engenharia Eletrica e de Computação, Centro de Tecnologia, Universidade Federal do Rio Grande do Norte, Natal. 2013.

FLANNERY, L. P. ; KAZAKOFF, E. R. ; BONTÁ, P. Dedigning Scratch Jr: support for early childhood learning through computer programming. In: INTERNATIONAL CONFERENCE ON INTERACTION DESIGN AND CHILDREN, 12., 2013, New York. **Proceedings …** New York: ACM, 2013.

FREIRE, P. **Pedagogia do oprimido**. Rio de Janeiro: Paz e Terra, 1970.

GENERAL PUBLIC LICENSE. Disponível em: <http://www.gnu.org/licenses/gpl.html>. Acesso em: nov. 2013.

HUNDHAUSEN, C. D. ; FARLEY, S. ; BROWN, J. L. Can direct manipulation lower the barriers to programming and promote positive transfer to textual programming? an experimental study. In: IEEE SYMPOSIUM ON VISUAL LANGUAGES AND HUMAN-CENTRIC COMPUTING, 2006.Brighton. **Proceedings …** Los Alamitos: IEEE, 2006. p. 157-164.

JOHNSON-LAIRD, P. **Mental models**. Cambridge, MA: Harvard University Press, 513p, 1983.

FRANCISCO JÚNIOR, N. M. ; VASQUES, C. K. ; FRANCISCO, T. H. A. Robótica educacional e a produção científica na base de dados da capes. **Revista Electrónica de Investigación y Docencia (REID)**, [S.l.], n. 4, p. 35-53, 2010. ISSN 1989-2446. Disponível em: <http://www.ujaen.es/revista/reid/revista/n4/REID4art2.pdf>. Acesso em: nov. 2013.

KAZAKOFF, E. R. ; BERS, M. U. Designing new technologies for early childhood: results from the initial pilot studies of ScratchJr**.** SRCD SOCITY FOR RESEARCH IN CHILD DEVELOPMENT, 2013. Seattle. **Poster presented at SRCD Society for Research in Child Development**. Seattle: SRCD, 2013.

LEGO mindstorms. Disponível em: <http://mindstorms.lego.com>. Acesso em: nov. 2013.

MAEDA, J. **The laws of simplicity**. Cambridge: MIT Press, 2006.

MAISONNETTE, R. **A utilização dos recursos informatizados a partir de uma relação inventiva com a máquina:** a robótica educativa. Proinfo – Programa Nacional de Informática na Educação, 2002. Disponível em: http://www.proinfo.gov.br/upload/biblioteca/192.pdf Acesso em: nov. 2013.

MARTINS, A. R. Q. **Usando o Scratch para potencializar o pensamento criativo em crianças do ensino fundamental**. 2012. Dissertação (Mestrado em Educação) – Programa de Pós-Graduação em Educação, Universidade de Passo Fundo, Passo Fundo, 2012.

MEDEIROS FILHO, D. A. ; GONÇALVES, P. C. Robótica educacional de baixo custo: uma realidade para as escolas brasileiras. In: CONGRESSO DA SOCIEDADE BRASILEIRA DE COMPUTAÇÃO, 28., 2008, Belém do Para. WORKSHOP SOBRE INFORMÁTICA NA ESCOLA. 2008, Belém do Para. **Anais …** . Belém do Para: SBC, 2008.

MEEGO. Disponível em: <http://www.metasys.com.br/index.php?option=com_content&view=article&id=417&Itemid=173&lang=pt>. Acesso em: nov. 2013.

MELLIS, D. O hardware em código aberto. **Info Exame**, 09 março, 2009. Entrevista. Disponível em: <http://info.abril.com.br/professional/tendencias/hardware-livre-leve-e-solto.shtml>. Acesso em: nov. 2013.

MÉLO, F. É. N. ; CUNHA, R. R. M. ; SCOLARO, D. R. ; CAMPOS, J. L. Do Scratch ao Arduino: uma proposta para o ensino introdutório de programação para cursos superiores de tecnologia, In: CONGRESSO BRASILEIRO DE EDUCAÇÃO EM ENGENHARIA, 39., 2011, Blumenau. **Anais …** Blumenau: ABENGE/FURB, 2011.

MENDELSON, P. ; GREEN, T. R. G. ; BRNA, P. Programming languages in education: the search for an easy start. In: HOC, J. ; GREEN, T. ; GILMORE, D. ; SAMWAY, R. (Eds) **Psychology of programming.** London: Academic Press, 1990. p. 175-200.

MIRANDA, L. C. de. **RoboFácil:** especificação e implementação de artefatos de hardware e software de baixo custo para um kit de robótica educacional. Dissertação (Mestrado em Informática) – Programa de Pós-Graduação em Informática, Instituto de Maatemática, Núcleo de Computação Eletrônica, Universidade Federal do Rio de Janeiro, Rio de Janeiro, 2006. Disponível em: <http://www.nce.ufrj.br/ginape/>. Acesso em: nov. 2013.

MIRANDA, L. C. ; SAMPAIO, F. F. ; BORGES, J. A. S. RoboFácil: especificação e implementação de um kit de robótica para a realidade educacional brasileira. **Revista Brasileira de Informática na Educação**, Florianópolis, v. 18, n. 3, p. 45-58, 2010.

OLIMPÍADA brasileira de robótica. Disponível em: <http://www.obr.org.br/>. Acesso em: nov. 2013.

PAPERT, S. **Logo:** computadores e educação. São Paulo: Brasiliense, 1985.

PASTERNAK, E. **Visual programming pedagogies and integrating current visual programming language features**. 2009. Dissertation (Master's Degree) – Robotics Institute, Carnegie Mellon University, Pittsburgh, 2009. Disponível em: < http://www.ri.cmu.edu/pub_files/2009/8/Thesis-1.pdf >. Acesso em: nov. 2013.

PETe. **Planejamento em educação tecnológica**. Disponível em: <http://www.pete.com.br/>. Acesso em: nov. 2013.

PIAGET, J. **Abstração reflexionante:** relações lógico-aritméticas e ordem das relações espaciais. Porto Alegre: ArtMed, 1995.

PINHEIRO, R. G. P. ; ELIA, M. F. ; SAMPAIO, F. F. Avaliando as competências escolares através da Prova Brasil usando ferramenta web. In: CONGRESSO BRASILEIRO DE INFORMÁTICA NA EDUCAÇÃO – CBIE, 2., 2013, Campinas; WORKSHOP DE INFORMÁTICA NA ESCOLA – WIE, 19., 2013, Campinas. **Anais …** Campinas: SBC, 2013.
,

PINTO, M. C. **Aplicação de arquitetura pedagógica em curso de robótica educacional com hardware livre**. 2011. Dissertação (Mestrado em Informática) – Programa de Pós-Graduação em Informática, Instituto de Mataemática, Instituto Tércio Pacitti de Aplicações e Pesquisas Computacionais, Universidade Federal do Rio de Janeiro, Rio de Janeiro, 2011. Disponível em: <http://www.nce.ufrj.br/ginape/>. Acesso em: nov. 2013.

PINTO, M. C. ; ELIA, M. F. ; SAMPAIO, F. F. Formação de professores em robótica educacional com hardware livre Arduino no contexto um computador por aluno. In: WORKSHOP DE INFORMÁTICA NA ESCOLA. 18., 2012, Rio de Janeiro: **Anais …** Rio de Janeiro, SBC, 2012.

POCRIFKA, D. H. ; SANTOS, T. W. Linguagem logo e a construção do conhecimento. EDUCERE - CONGRESSO NACIONAL DE EDUCAÇÃO, 9., 2009, Curitiba. **Anais …** Curutiba: PUCPR, 2009. Disponível em: <http://www.pucpr.br/eventos/educere/educere2009/anais/pdf/2980_1303.pdf>. Acesso em: nov. 2013.

PROCESSING. Disponível em: <http://processing.org/>. Acesso em: nov. 2013.

PROUCA. **Programa um computador por aluno**. Disponível em: <http://www.bndes.gov.br/SiteBNDES/bndes/bndes_pt/Institucional/Apoio_Financeiro/Programas_e_Fundos/prouca.html>. Acesso em: nov. 2013.

PYJAMAS. Disponível em: <pyjs.org>. Acesso em: nov. 2013.

RIBEIRO, P. C. ; MARTINS, C. B. ; BERNARDINI, F. C. A Robótica como ferramenta de apoio ao ensino de disciplinas de programação em cursos de computação e engenharia. In: SIMPÓSIO BRASILEIRO DE INFORMÁTICA NA EDUCAÇÃO, 22., 2011, Aracaju. **Anais …** Aracajú, SBC, 2011.

SAMPAIO, F. F.; ELIA, M. F. **Projeto Uca na Cuca:** robótica educacional na sala de aula. 2011. Disponível em: <http://www.nce.ufrj.br/GINAPE/publicacoes/Projetos/Proj_UCAnaCUCA.pdf>. Acesso em: nov. 2013.

SAMPAIO, F. F. ; ELIA, M. F. **Projeto um computador por aluno:** pesquisas e perspectivas. 2012. Disponível em: <http://www.nce.ufrj.br/ginape/livro-prouca>. Acesso em: nov. 2013.

SANTOS, F. L. ; NASCIMENTO, F. M. S. ; BEZERRA, R. M. S. REDUC: a robótica educacional como abordagem de baixo custo para o ensino de computação em cursos técnicos e tecnológicos. In: WIE - WORKSHOP DE INFORMÁTICA NA ESCOLA, 16., 2010, Belo Horizonte. **Anais …** Belo Horizonte: SBC, 2010.

SASAHARA, L. R. ; CRUZ, S. M. S. Hajime – uma nova abordagem em robótica educacional. In: SIMPÓSIO BRASILEIRO DE INFORMÁTICA NA EDUCAÇÃO, 18., 2007, São Paulo. **Anais …** São Paulo: SBC, 2007.

SCAICO, P. D. ; LIMA, A. A. ; SILVA, J. B. B. ; AZEVEDO, S. ; PAIVA, L. F. ; RAPOSO, E. H. S. ; ALENCAR, Y. ; MENDES, J. P. Programação no ensino médio: uma abordagem de ensino orientado ao design com Scratch. In: WIE - WORKSHOP DE INFORMÁTICA NA ESCOLA, 18., 2012, Rio de Janeiro, **Anais ...** Rio de Janeiro: SBC, 2012.

SCHONS, C. ; PRIMAZ, E. ; WIRTH, G. A. P. Introdução a robótica educativa na instituição escolar para alunos do ensino fundamental da disciplina de língua espanhola através das novas tecnologias de aprendizagem. In: WORKSHOP DE COMPUTAÇÃO DA REGIÃO SUL, 1., 2004. Florianópolis. **Anais …** Florianópolis: [s.n.], 2004.

SILVA, A. F. **RoboEduc:** uma metodologia de aprendizado com robótica educacional. 2009. Tese (Doutorado em Ciências) – Programa de Pós-graduação em Engenharia Eletrica, Centro de Tecnologia, Universidade Federal do Rio Grande do Norte, Natal, 2009.

SILVA, A. L. C. ; SAMPAIO, F. F. ; ELIA, M. F. ; BRANDÃO, S. Laboratório remoto de robótica educativa – LabVad. In: EURO AMERICAN ASSOCIATION ON TELEMATICS AND INFORMATION SYSTEMS. 7., 2014, Valparaiso. **Proceedings …** New York: ACM, 2014.

SOUSA, F. R. C. ; MOREIRA, L. O. ; MACHADO, J. C. Computação em nuvem: conceitos, tecnologias, aplicações e desafios. In: **ERCEMAPI 2009**. Versão revisada e estendida em setembro de 2010. Cap. 7. Disponível em: <http://www.ufpi.br/ercemapi/arquivos/file/minicurso/mc7.pdf>. Acesso em: nov. 2013.

SOUZA, M. B. ; NETTO, J. F. M. ; ALENCAR, M. A. S. ; SILVA, M. M. Arcabouço de um ambiente telerobótico educacional baseado em sistemas multiagentes. In: XXII SIMPÓSIO BRASILEIRO DE INFORMÁTICA NA EDUCAÇÃO, 22. 2011, Aracajú. **Anais …** Aracajú: SBC, 2011.

UCA na cuca. Disponível em: <http://www.nce.ufrj.br/ginape/ucanacuca/>. Acesso em: nov. 2013.

VALENTE, J. A. Por quê o computador na educação? In: _____. **Computadores e conhecimento:** repensando a educação. Campinas: UNICAMP, 1993. p. 24-44.

VAN SOMEREN, M. W. ; BARNARD, Y. F. ; SANDBERG, J. A. C. **The think aloud aethod. A practical guide to modelling cognitive processes**. London: Academic Press, 1994. Disponível em: <http://staff.science.uva.nl/~maarten/Think-aloud-method.pdf>. Acesso em: nov. 2013.

VICTORINO, L. ; ELIA, M. F. ; GOMES, A. ; CASTRO, M. ; BASTOS, C. Laboratório virtual de atividades didáticas – LabVad. In: WORKSHOP DE INFORMÁTICA NA ESCOLA, 15, 2009, Bento Gonçalves. **Anais …** Bento Gonçalves: SBC, 2009. Disponível em: <http://www.lbd.dcc.ufmg.br/colecoes/wie/2009/022.pdf>. Acesso em: nov. 2013.

VYGOTSKY, L. ; COLE, M. ; JOHN-STEINER, S. ; SCRIBNER, S. ; SOUBERMAN, L. **Mind in society:** the development of higher psychological processes. Cambridge, MA: Harvard University Press, 1978.

WIRING. Disponível em: <http://wiring.org.co/>. Acesso em: nov. 2013.

ZANETTI, H. A .; SOUZA, A. L. D. ; D'ABREU, J. V. ; BORGES, M. A. Uso de robótica e jogos digitais como sistema de apoio ao aprendizado. In: JAIE - JORNADA DE

ATUALIZAÇÃO EM INFORMÁTICA NA EDUCAÇÃO, 2013, Campinas. **Anais …** , Campinas: SBC, 2013.

ZHANG, S. ; ZHANG, S. ; CHEN, X. ; HUO, X. **Cloud romputing research and development trend**. In: INTERNATIONAL CONFERENCE ON FUTURE NETWORKS, 2., 2010, Sanya, Hainan, China. **Proceedings …** Los Alamitos: IEEE, 2010.

ZILLI, S. R. **A robótica educacional no ensino fundamental:** perspectivas e prática. 2004. Dissertação (Mestrado em Engenharia da Produção) – Programa de Pós-Graduação em Engenharia da Produçaõ, Universidade Federal de Santa Catarina, Florianópolis, 2004.

ANEXOS

Anexo 1 - Tutorial de Instalação do Arduino no Meego

Os procedimentos deste tutorial visam à instalação do IDE Arduino no sistema operacional Meego fornecido pela Metasys. Neste processo, o classmate deve estar conectado à internet.

Passo 1: Abrir terminal como administrador

1.1. Busque na barra superior a seção "**Aplicativos**"

1.2. Clique na barra "**Acessórios**"

1.3. Clique no item "**Terminal**" (Figura 57)

Figura 57: Abrindo o Terminal para Instalar o Arduino

1.4. Digite "**su**" na tela do terminal e pressione enter (Figura 58)

1.5. Será solicitada uma senha. Digite "**metasys**" (sem as aspas) e pressione enter

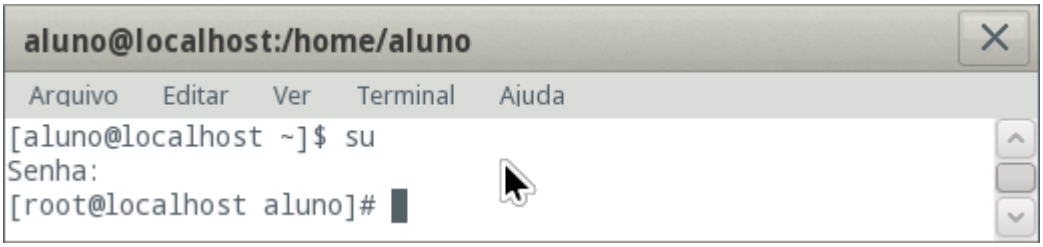


Figura 58: Entrando como administrador no Terminal para Instalar o Arduino

Passo 2: Instalação do Arduino

2.1. Digite o comando abaixo para adicionar o repositório e pressione enter (Figura 59).

sudo smart channel --add arduino-metasys-netbook type=rpm-md baseurl=http://update.metasys.com.br/repository/extra/arduino-metasys-netbook

2.1.1. Irá aparecer a mensagem abaixo. Digite “**s**” (sem aspas) e pressione enter:

Incluir este canal? (s/N):

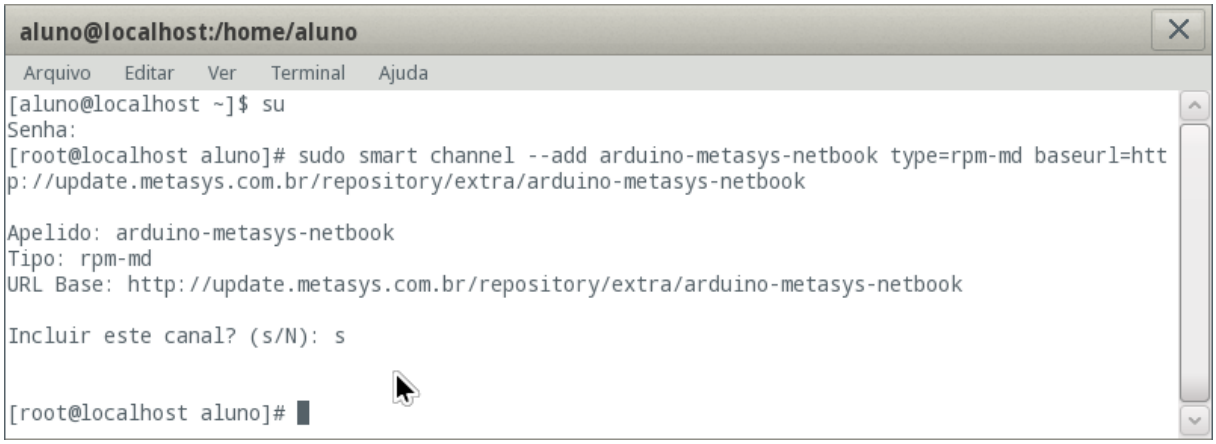


Figura 59: Adicionando o repositório

2.2. Digite o comando abaixo atualizar os repositórios e pressione enter (Figura 60).

sudo smart update

2.2.1. Irá aparecer a mensagem abaixo. Digite “**s**” (sem aspas) e pressione enter:

Confirma mudanças? (S/n):

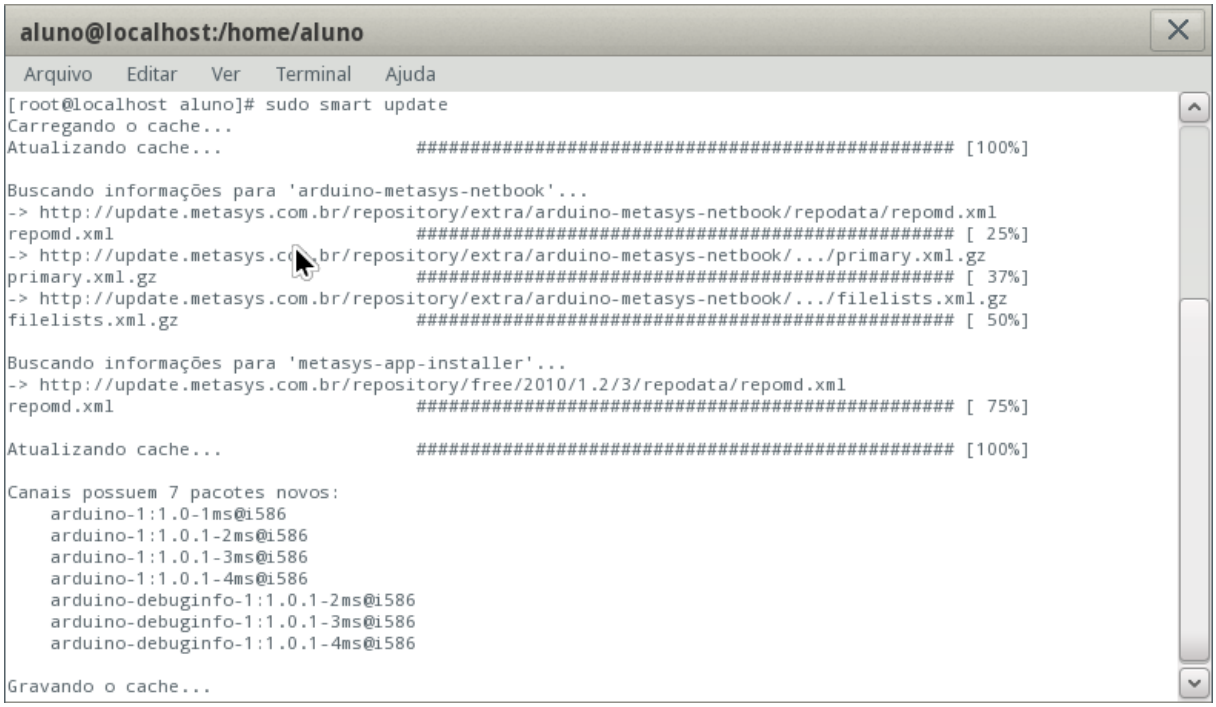


Figura 60: Atualizando repositório

2.3. Digite o comando abaixo para instalar o Arduino e pressione enter (Figura 61)

sudo smart install arduino

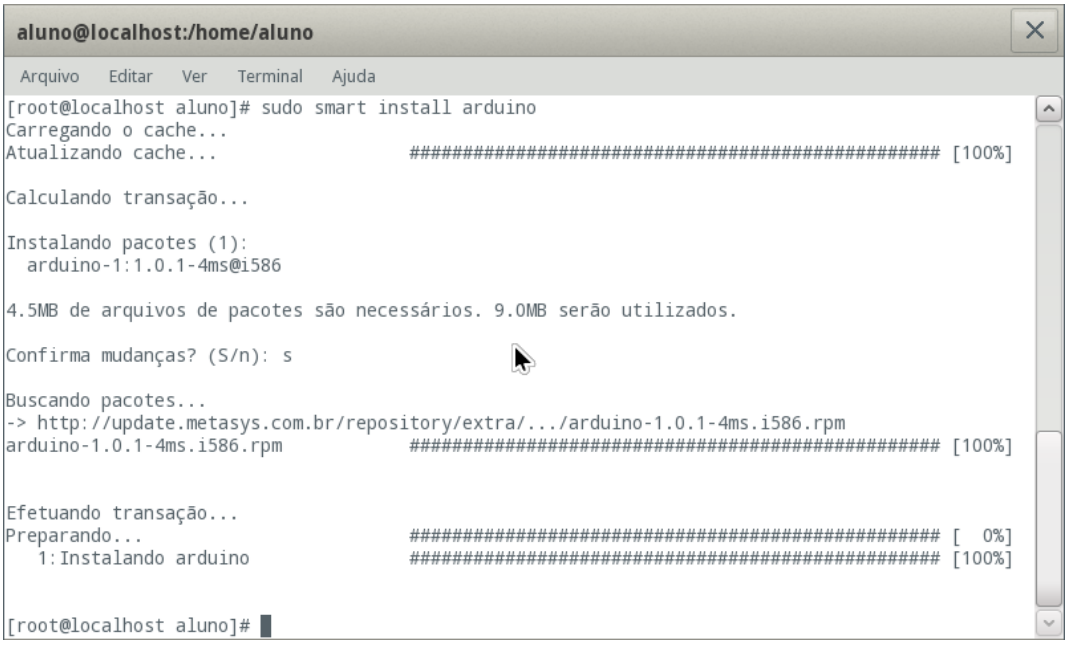


Figura 61: Instalando o Arduino.

Passo 3: Executando o IDE Arduino

3.1. Busque na barra superior a seção "**Aplicativos**"

3.2. Clique na barra "**Desenvolvimento**"

3.3. Clique em "**Arduino**" (Figura 62)

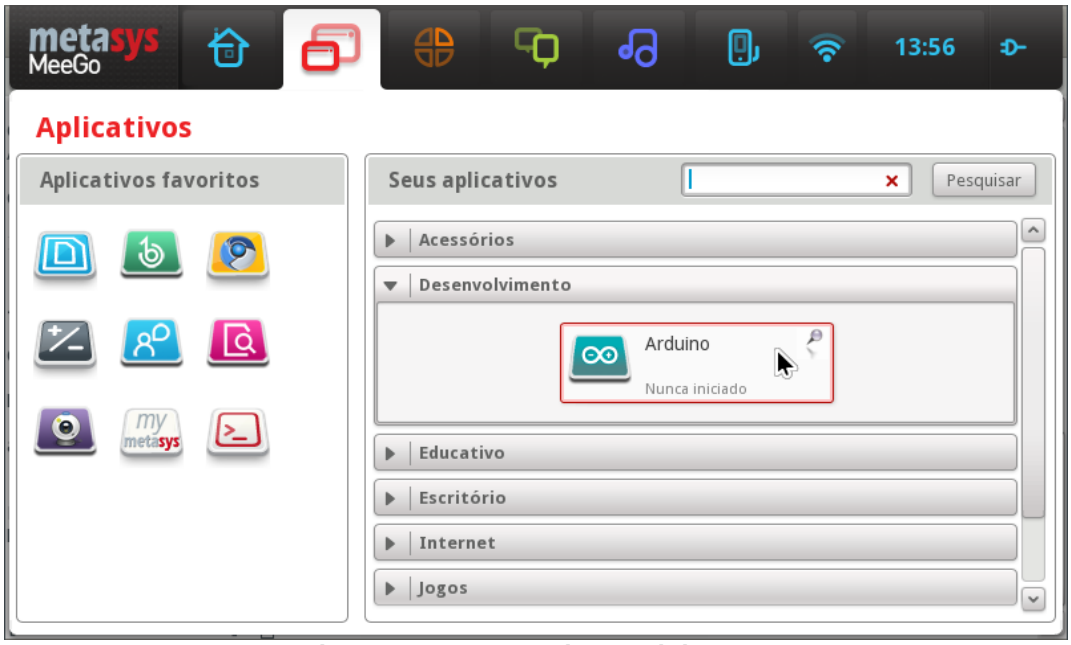


Figura 62: Executando o Arduino

Passo 4: Configurando o Idioma para o Português

4.1. Com o software Arduino aberto, clique no menu “**File**”

4.2. Clique no sub-menu “**Preferences**”

4.3. No campo “**Editor language**” selecione “**Português (Portuguese - Brazil)”** e clique em OK (Figura 63)

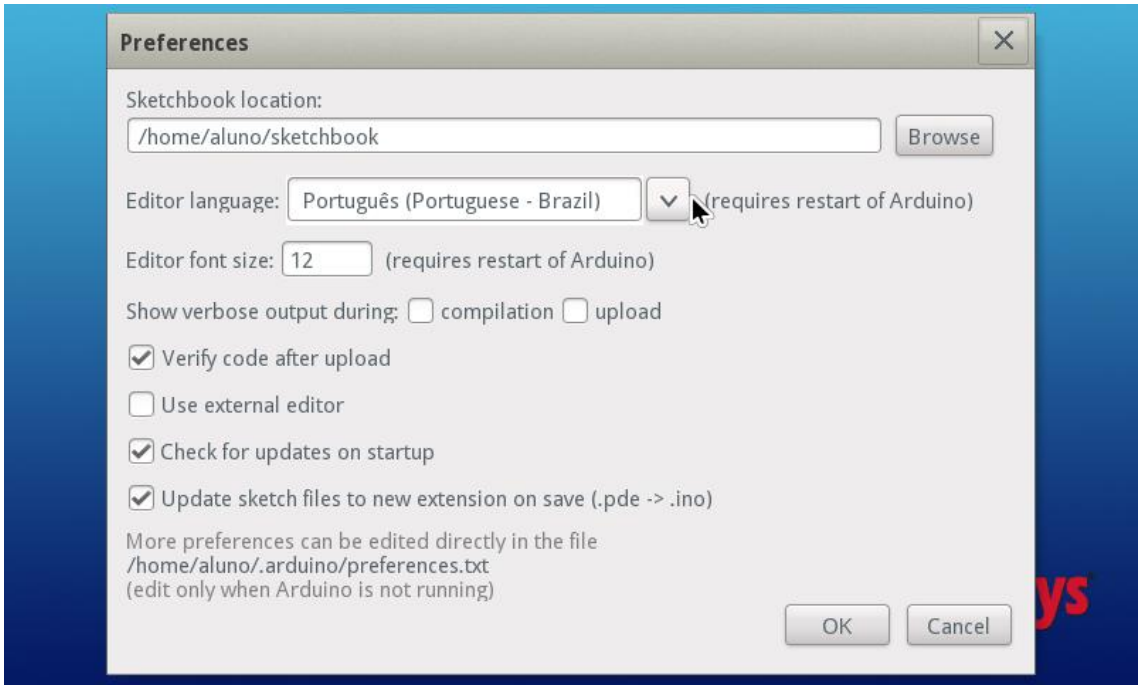


Figura 63: Configurando o Idioma

4.4. Para as mudanças terem efeito é necessário abrir o programa novamente (Passo 3). A Figura 64 mostra o Arduino instalado e configurado para português.

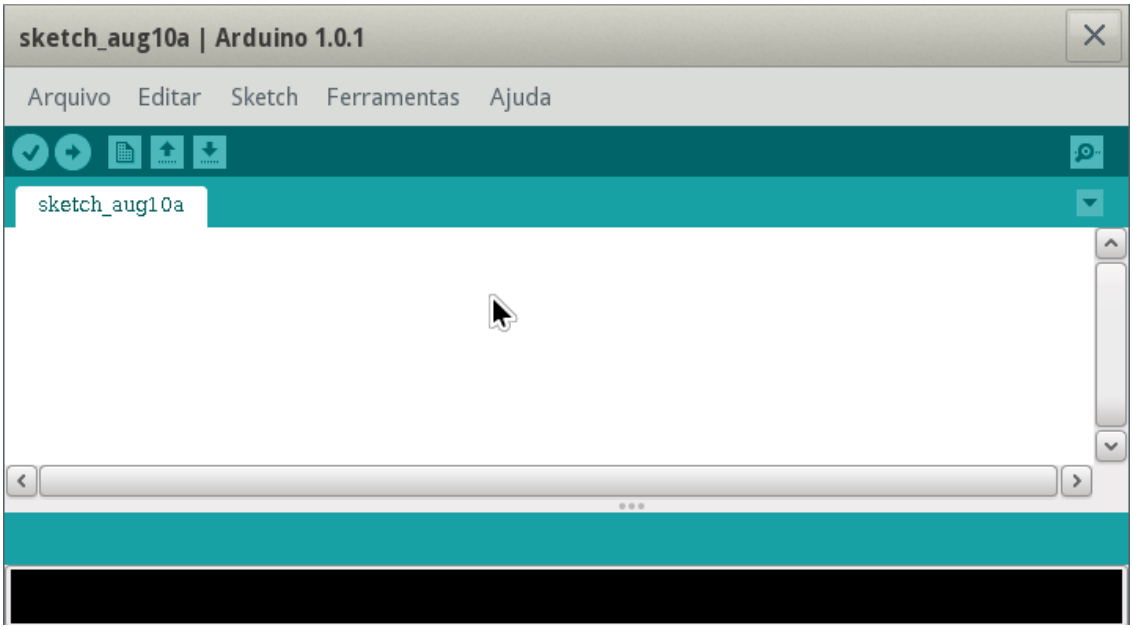


Figura 64: Arduino em português instalado no Meego

OBS: Caso necessite **remover** a instalação execute os seguintes passos:

1. Abra terminal como administrador
2. Digite o comando abaixo para remover o repositório (Figura 65):

Sudo smart channel –remove arduino-metasys-netbook

2.1. Irá aparecer a mensagem abaixo. Digite "**s**" (sem aspas) e pressione enter:

Remover canal 'arduino-metasys-netbook'? (s/N):

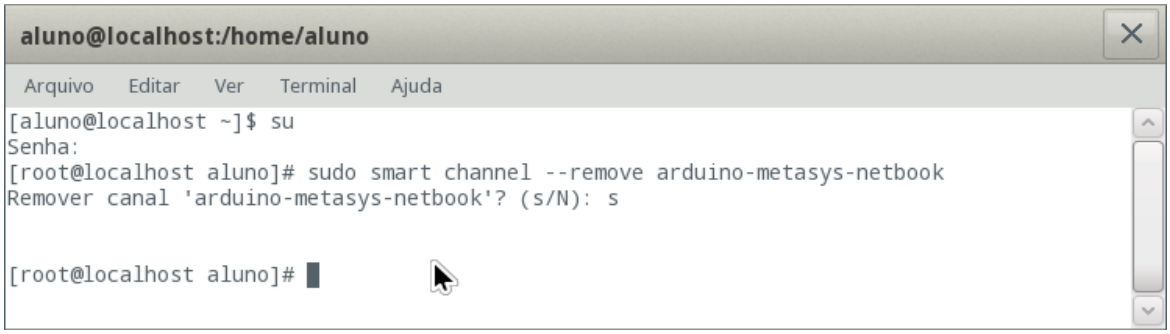


Figura 65: Removendo repositório

3. Digite o comando abaixo para desinstalar o Arduino (Figura 66):

3.1. Irá aparecer a mensagem abaixo. Digite "**s**" (sem aspas) e pressione enter:

Confirma mudanças? (S/N):

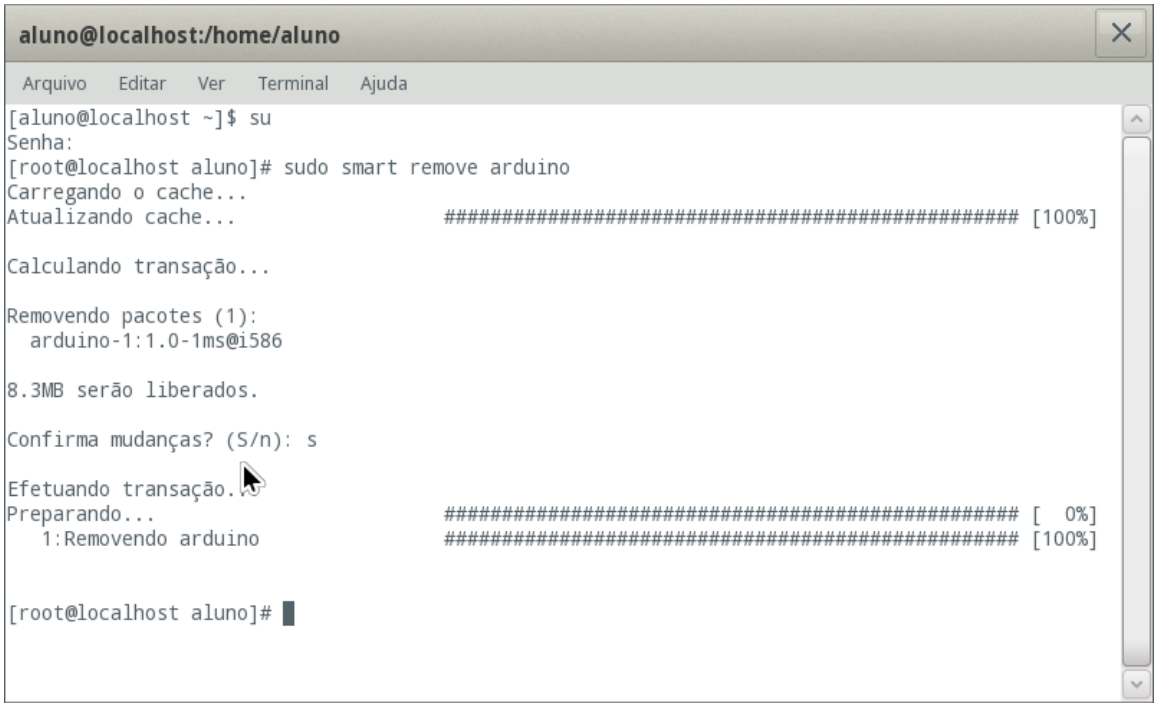


Figura 66: Desinstalando Arduino

APÊNDICES

Apêndice 1 - Tutorial de Atualização do Chromium no Meego

Os procedimentos deste tutorial visam à atualização do navegador Chromium no sistema operacional Meego para a utilização plena dos recursos do DuinoBlocks. Este tutorial é adaptado do original fornecido pela equipe da Metasys. Neste processo, o classmate deve estar conectado à internet.

Passo 1: Baixar os arquivos necessárias à atualização

1.1. Baixe o arquivo “chromium.rar” disponibilizado no Google Drive do projeto Uca na Cuca. Descompacte o arquivo em um pendrive, nele contém uma pasta de mesmo nome. Dentro desta pasta há 10 arquivos com extensão RPM para instalar.

https://docs.google.com/file/d/0B3DaNEmxS_0daUdwSUxBOXlQcms/edit?usp=sharing

Passo 2: Abrir terminal como administrador

2.1. Busque na barra superior a seção "**Aplicativos**"

2.2. Clique na barra "**Acessórios**"

2.3. Clique no item "**Terminal**"

2.4. Digite "**su**" na tela do terminal e pressione enter (Figura 67)

2.5. Será solicitada uma senha. Digite "**metasys**" (sem as aspas) e pressione enter

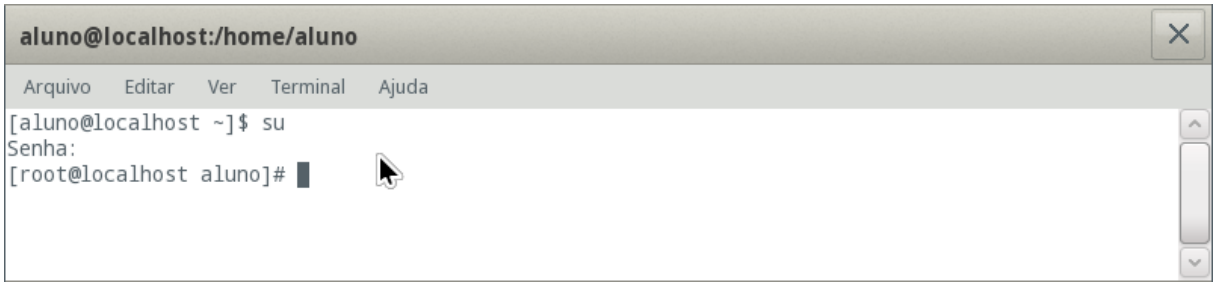


Figura 67: Entrando como administrador no Terminal para Atualizar o Chromium

Passo 3: Acessando os arquivos baixados

3.1. Digite o comando abaixo para acessar a pasta “chromium” que esta no pendrive. Substitua “nome_do_pendrive” pelo nome do pendrive que contém os arquivos baixados.

cd /media/nome_do_pendrive/chromium/

3.2. Digite o comando “**ls**” (sem aspas) e pressione enter para visualizar e certificar-se do conteúdo da pasta (Figura 68).

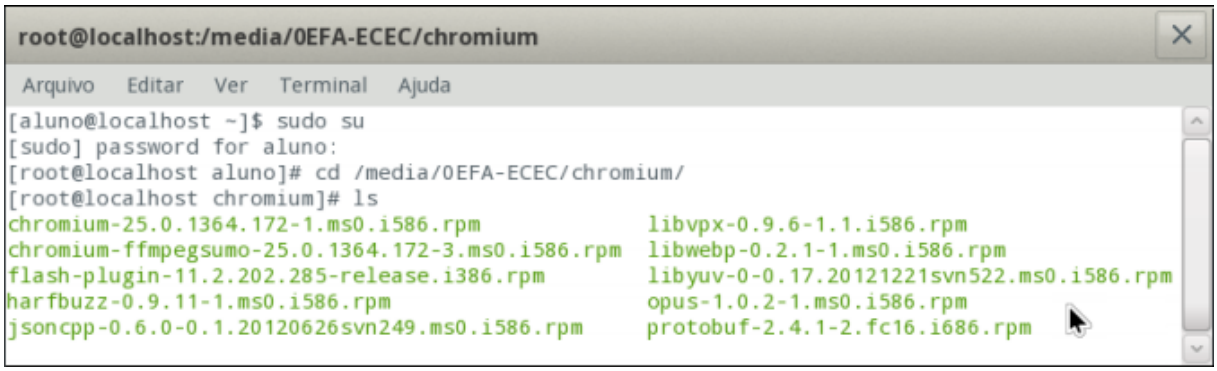


Figura 68: Acessando os arquivos necessários à atualização do Chromium

Passo 4: Atualização do Chromium

4.1. Digite o comando abaixo para atualizar o Chromium (Figura 69).

rpm -Uvh *

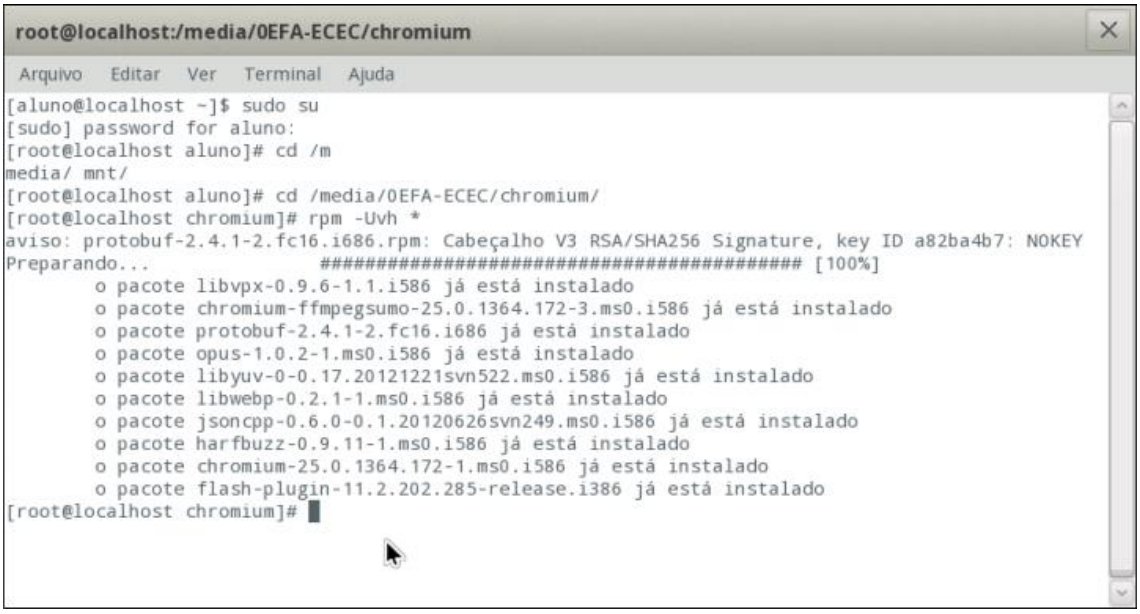


Figura 69: Atualizando o Chromium

4.2. Aguarde até que todos os 10 pacotes sejam instalados. Após a instalação feche o terminal ou digite duas vezes o comando:

exit

UNIVERSIDADE FEDERAL DO RIO DE JANEIRO
PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA

CCMN - Bloco C - Cidade Universitária - Ilha do Fundão
Rio de Janeiro - RJ CEP: 21941-916
www.ppgi.ufrj.br