RASPIBLOCOS: Ambiente de Programação Didático Baseado em Raspberry Pi e Blockly

Eduarth Heinen[1], Eleandro Maschio[1], Diego Marczal[1], Pedro Lealdino Filho[2]

[1]Coordenação do Curso de Tecnologia em Sistemas para Internet
Universidade Tecnológica Federal do Paraná (UTFPR)
Guarapuava – PR – Brasil

[2]Université Claude Bernard, Lyon I
Villeurbanne – France

eheinen@alunos.utfpr.edu.br, {eleandrom,marczal}@utfpr.edu.br

pedro.lealdino@etu.univ-lyon1.fr

Abstract. *This article details a visual programming environment able to translate instructions, represented by building blocks, into actions performed by components of a concrete robotic model. The environment's interface was implemented with Blockly library, and the robotic model with the single board computer Raspberry Pi. As a differential technology, it is an open-source initiative, where the same piece of hardware acts as a host for a web application, accessible through wi-fis, and as a controller of the electronic components. With this, the experiments can be easily replicated. The tool provides experiments in programming and robotics build by the students, acting as a motivational and enhancement factor in learning.*

Resumo. *Este artigo detalha um ambiente de programação visual capaz de traduzir instruções, representadas por blocos encaixáveis, em reações nos componentes de um modelo robótico físico. A interface do ambiente foi implementada com a biblioteca Blockly, e o modelo robótico com o minicomputador Raspberry Pi. Como diferencial tecnológico, trata-se de uma iniciativa open-source, em que o mesmo minicomputador atua como hospedeiro de uma aplicação web, acessível por wi-fi, e controlador dos componentes. Com isso, os experimentos são facilmente replicados. A ferramenta promove que sejam construídas experiências em programação e robótica por alunos, atuando como fator motivacional e de aprofundamento no ensino.*

1. Introdução

Existem indícios de que a aplicação de metodologias de ensino, que envolvam a construção autônoma de teorias pelos alunos, alcance melhores resultados do que abordagens baseadas em aulas expositivas [Papert 1980]. Considerando isso, pesquisadores têm proposto o uso da robótica na construção de ambientes que sirvam de contexto para a formulação e validação de hipóteses. Projetos nesse sentido buscam utilizar a robótica e a tecnologia na educação com o objetivo de estimular o interesse e o aprendizado dos alunos.

DOI: 10.5753/cbie.sbie.2015.567

A popularização de brinquedos programáveis, como a linha LEGO Mindstorms[1], além de computadores de placa única, como Arduino[2] e Raspberry Pi[3], simplificou a construção de modelos robóticos, incentivando o surgimento de projetos de pesquisa na área da Informática na Educação. Tais projetos, integrando robôs e noções de programação na apresentação de conteúdo aos alunos, buscam avaliar o impacto e a influência dessas tecnologias no aprendizado. Entre os resultados observados em experimentos na área, destacam-se o aumento da participação e da motivação dos alunos, assim como um maior desenvolvimento de habilidades ligadas ao raciocínio lógico e à resolução de problemas [Saygin et al. 2012, Saleiro et al. 2013].

A carência de profissionais nos campos da Ciência, Tecnologia, Engenharia e Matemática[4] é outro fator apontado pelos autores como motivação para a pesquisa. O contato com experiências que integrem tecnologia e robótica em lições escolares é percebido como um incentivo ao interesse dos alunos por essas áreas [Iturrate et al. 2013, Vandevelde et al. 2013].

Entre as ferramentas disponíveis para a realização de lições de robótica em sala de aula, a linha de brinquedos programáveis LEGO Mindstorms mostra-se tão versátil que tem sido utilizada em competições [Grout and Houlden 2014], mas possui preço restritivo. Soluções de baixo custo, como a placa eletrônica programável Picoboard, por outro lado, são acessíveis, mas oferecem recursos limitados. Projetos de pesquisa, em sua maioria, descrevem soluções que dependem de um requisito específico (como uma placa impressa por encomenda) ou representam soluções proprietárias, utilizadas por uma determinada universidade ou iniciativa em andamento.

Neste contexto, a corrente pesquisa apresenta um ambiente capaz de traduzir programas, representados por blocos encaixáveis, em reações no modelo robótico físico. Para isso, o ambiente provê uma interface em que blocos coloridos denotam instruções e possibilitam a construção de pequenos programas. Cada programa, ao ser executado, envia instruções para componentes eletrônicos conectados a um minicomputador Raspberry Pi, de maneira que o aluno observe o resultado do experimento por meio do comportamento apresentado pelo modelo robótico.

Destaca-se, como maior diferencial tecnológico, que se trata de uma iniciativa *open-source*, em que o minicomputador Raspberry Pi atua simultaneamente como hospedeiro da aplicação web, acessível por *wi-fi*, e controlador dos componentes. Com isso, os experimentos podem ser facilmente replicados. Ademais, considerando a crescente popularidade do Raspberry Pi, a distribuição livre atingiria potencialmente maior abrangência, atraindo mais usuários e colaboradores para o projeto.

2. Referencial Teórico e Trabalhos Correlatos

O desenvolvimento de um conjunto de robótica voltado à educação, com um controlador Raspberry Pi e interface baseada em Blockly[5], foi proposto anteriormente por Saleiro et al. [2013]. A equipe de pesquisadores desenvolveu um sistema que

[1]LEGO Mindstorms: `www.lego.com/en-us/mindstorms`
[2]Arduino: `www.arduino.cc`
[3]Raspberry Pi: `www.raspberrypi.org`
[4]Do inglês: *Science, Techonology, Engineering and Mathematics* (STEM).
[5]Blockly: `developers.google.com/blockly`

utiliza o minicomputador para controlar múltiplos robôs. Cada pequeno robô, chamado "robô infante", é capaz de se mover através de uma grade de linhas pretas e fundo branco. A interface da aplicação oferece blocos de instruções como "seguir em frente" e "girar 90º à direita", permitindo que o usuário construa um pequeno roteiro para levar o robô até os objetivos marcados no mapa.

Os autores apontam que esse conjunto pode ser utilizado por professores em lições de diversas disciplinas, simplesmente alterando os objetivos e o contexto do exercício para corresponder ao conteúdo da lição. Além de motivar os alunos a aprenderem o conteúdo abordado, então percebido como um requisito necessário para que o robô possa atingir os objetivos do exercício, os experimentos possibilitam desenvolver habilidades como o raciocínio matemático e a dedução lógica.

A metodologia aplicada pelos pesquisadores incluiu lições de Geografia e de reciclagem de lixo eletrônico, com turmas de terceiras e quartas séries do ensino fundamental, em escolas de Faro, Portugal. Uma lição de Geografia, por exemplo, definiu como objetivos, no mapa, as posições das principais montanhas de Portugal em relação a Faro. Programando a movimentação dos robôs para que alcançassem esses objetivos, os alunos puderam construir conhecimentos sobre sua localização relativa às montanhas. Da mesma maneira, observando o resultado de cada instrução, definindo roteiros e resolvendo problemas decorrentes, conseguiram desenvolver habilidades de raciocínio matemático, dedutivo, abdutivo e indutivo. Os pesquisadores relataram que a utilização do experimento nas lições proporcionou que os alunos adotassem a postura de formuladores de hipóteses, buscando soluções colaborativas e discutindo-as abertamente.

A aplicação utilizada no experimento de Saleiro et al. [2013], entretanto, não está disponível ao público, e os robôs, mesmo sendo feitos com materiais acessíveis, dependem de uma placa impressa por encomenda. Além disso, o projeto de Saleiro é focado em alunos do ensino básico e, portanto, as instruções disponíveis na interface são bastante simples, e também limitadas às capacidades dos microrrobôs.

Outra experiência com objetivos semelhantes foi desenvolvida por Iturrate et al. [2013]. Nela, um robô navega através de um labirinto de acordo com as instruções recebidas do computador que controla o experimento. As imagens do laboratório onde o labirinto e o robô se encontram são transmitidas pela Internet até o usuário da aplicação. O conceito de laboratórios remotos, aplicado na pesquisa, busca oferecer uma alternativa para a democratização do acesso ao conhecimento produzido nas universidades, conectando os experimentos à Internet. Com isso, a pesquisa pretendeu possibilitar que estudantes de qualquer lugar do mundo se beneficiassem com o projeto desenvolvido.

O experimento de Iturrate et al. [2013] baseia-se no modelo proposto anteriormente por Dziabenko et al. [2012]. Trata-se de um jogo cujo enredo introduz o aluno ao cenário de um pouso forçado em um planeta alienígena. Como integrante da tripulação da nave, ele deve programar o robô para recuperar as peças espalhadas pelo cenário. Ao alcançar cada um dos objetivos, representados no labirinto por etiquetas de rádio frequência (RFID), uma questão relacionada ao tema abordado pela lição é apresentada ao aluno. Dessa forma, assim como no experimento de

Saleiro, os alunos percebem o conteúdo como requisito para alcançar os objetivos jogo. Ademais, diversos conteúdos diferentes podem ser apresentados pela simples alteração do cenário ou do tema das perguntas.

Como os robôs são construídos sobre minicomputadores Arduino Uno, a aplicação que controla o experimento de Iturrate envia a eles somente instruções previamente programadas. Como ocorre no projeto de Saleiro, esses robôs dependem de uma placa impressa sob encomenda. Os autores, entretanto, assumem a possibilidade de expandir as instruções disponíveis na aplicação. Até o presente momento, não haviam sido realizados testes da aplicação e a interface ainda não oferecia todos os recursos necessários para a contextualização das lições.

O projeto DuinoBlocks, desenvolvido por Alves e Sampaio [2014], é outro exemplo de ambiente que aplica linguagem de programação visual para controlar o comportamento de um modelo robótico. Os pesquisadores realizaram duas oficinas de Robótica Educacional com professores de áreas variadas. Na abordagem, os participantes foram antes apresentados ao ambiente de programação nativo do Arduino e, depois, ao ambiente desenvolvido pelos pesquisadores. Ao serem solicitados a "pensarem em voz alta" sobre o funcionamento dos programas, os participantes demonstraram entendimento mais profundo daqueles construídos com o auxílio da interface visual desenvolvida no projeto. Em outra oficina, realizada com um grupo de professores que incluía pessoas com conhecimento em programação, experimentos mais complexos foram realizados, sendo constatada a eficiência do ambiente mesmo para usuários familiarizados com a sintaxe das linguagens de programação.

Os projetos mencionados constituem ferramentas com potencial para o emprego multidisciplinar da robótica na educação. Entre as vantagens observadas, evidencia-se a facilidade de configuração e de utilização, apontada como objetivo de todos os projetos, bem como a preocupação dos pesquisadores em criar contextos para apoiar os objetivos dos robôs.

Considerando que os projetos analisados possuem limitações, argumenta-se que uma iniciativa *open-source*, que utilize uma interface de programação visual no controle dos componentes de um modelo robótico, atingiria potencialmente maior abrangência. Do ponto de vista didático, destaca-se que as lições poderiam ser mais complexas, permitindo tratar conceitos de linguagens de programação e de eletrônica, direcionando o projeto também aos alunos de graduação. Com isso, o interesse deles pelos campos da Ciência, Tecnologia, Engenharia e Matemática, bem como a aquisição de conhecimentos em programação, continuariam sendo motivados.

3. Fundamentos da Solução Proposta

A ferramenta detalhada neste artigo consiste de uma aplicação web hospedada em um minicomputador Raspberry Pi, que utiliza uma interface de programação visual no controle dos componentes eletrônicos nele conectados. A arquitetura proposta pode ser dividida em quatro camadas: Interface, Aplicação, Controle e Componentes. A Figura 1 apresenta essa divisão, destacando os elementos das camadas e a interação entre eles. Na sequência, descreve-se cada camada, bem como o seu papel na arquitetura e as tecnologias que a compõem:

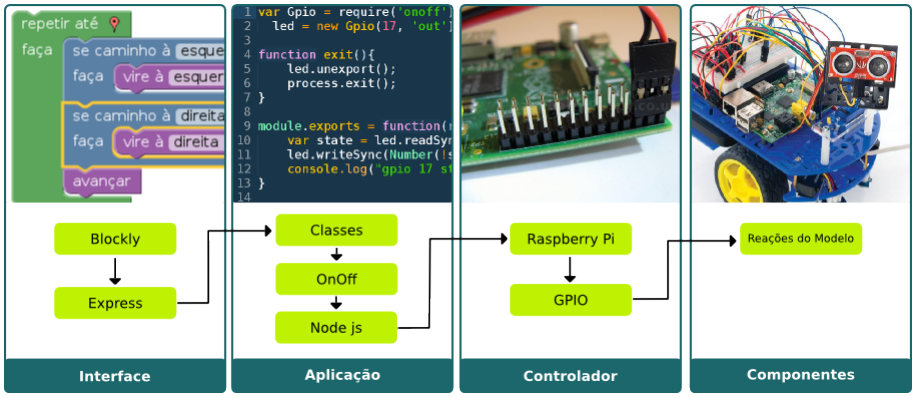

Figura 1. Arquitetura e interação entre os componentes da ferramenta

3.1. Camada de Interface

A interface da aplicação deveria ser simples e intuitiva, considerando a interação de usuários com diferentes níveis de experiência com computadores. Nesse sentido, a aplicação pode ser acessada por meio de um navegador web, conectando-se à rede *wi-fi* oferecida pelo minicomputador. Utiliza-se a biblioteca de programação visual Blockly na interface, de maneira que as instruções disponíveis no ambiente sejam representadas por blocos encaixáveis. Os programas compostos por esses blocos são executados pelo minicomputador, resultando em reações no modelo robótico.

Blockly é uma biblioteca destinada à construção de editores para programação visual. A aparência e o funcionamento dela são inspirados no ambiente Scratch, desenvolvido com o propósito de ensinar programação a pessoas sem experiência com computadores, por meio de atividades como a criação de animações e jogos interativos [Vandevelde et al. 2013].

Ambientes como Blockly, Scratch e Alice são exemplos que se amparam no conceito de linguagem de programação visual[6]. Nesse sentido, buscam simplificar a construção de programas empregando componentes gráficos para representar elementos instrucionais da linguagem utilizada. Com isso, instruções como declaração de variáveis, operações matemáticas, lógicas e relacionais, estruturas de controle (condicionais e de repetição), definição de funções, entre outras, podem ser encaixadas, compondo pequenos programas. Em adição, a biblioteca oferece a possibilidade de executar os programas criados, traduzindo-os em código equivalente nas linguagens Javascript, Python e Dart.

A Figura 2 apresenta um exemplo de programa construído utilizando interface da ferramenta. Esse programa tem a finalidade de fazer o led do microrrobô piscar 10 vezes. As conexões destacadas relacionam-se com: **(1)** instruções anteriores e subsequentes; **(2)** blocos que representam valores ou o retorno de uma função; e **(3)** instruções executadas dentro do escopo do bloco atual.

Ao limitar as conexões possíveis, linguagens de programação visual buscam impedir a construção de código incoerente [Pasternak 2009]. Dessa maneira, um bloco que representa uma variável, por exemplo, só pode ser conectado a outro que

[6] *Visual Programming Languages* (VPL).

Figura 2. Exemplo de programa construído utilizando a interface da ferramenta

retorne um valor compatível. Ainda, eliminam-se quase a totalidade dos erros sintáticos porque, ao invés de redigir as instruções, o usuário simplesmente encaixa trechos predefinidos de código. Isso se mostra interessante uma vez que os cinco erros de compilação mais comuns (*missing semicolon*, *unknown variable*, *illegal start of expression*, *unknown class* e *bracket expected*) correspondem a mais da metade daqueles cometidos por novatos. Por fim, complementando os motivos citados, o mesmo autor justifica que essas linguagens reduzem as barreiras cognitivas e auxiliam na criação de projetos relevantes, porque oferecem apoio semântico quanto ao propósito de cada instrução e sobre o estado do programa.

Considerando isso, a biblioteca Blockly foi escolhida para compor a interface de programação visual na arquitetura proposta. Somados aos blocos oferecidos pela biblioteca, outros foram adicionados, significando instruções próprias do modelo robótico. São instruções específicas como: "mover para frente" e "mover para trás" para motores; "medir distância" para sensores de proximidade; "acender" e "apagar" para leds; e "quando pressionado faça..." para botões. Essas instruções, e respectivos componentes, permitem que o modelo robótico possa ser programado para que se mova por uma sala desviando obstáculos.

3.2. Camada de Aplicação

O uso do minicomputador Raspberry Pi foi motivado pela sua capacidade de hospedar a aplicação web e de, simultaneamente, controlar os componentes do modelo robótico. Por meio de um adaptador *wi-fi*, o minicomputador fornece uma rede de internet sem-fio, de maneira que qualquer dispositivo conectado possa acessar a aplicação, utilizando um navegador web.

A aplicação web, ao ser acessada, apresenta a interface de programação visual com os blocos das instruções que o modelo robótico é capaz de executar. Os programas construídos com essas instruções, quando executados, enviam à aplicação mensagens contendo o componente eletrônico e a ação que deve ser realizada. As classes da aplicação, representando cada um desses componentes, oferecem métodos que encapsulam o código responsável por realizar suas respectivas ações. Tais métodos são acionados pela aplicação de acordo com as mensagens recebidas da interface, gerando reações nos componentes eletrônicos do modelo robótico. Para isso, foi necessário empregar a plataforma Node.js, em conjunto com o *framework* Express, fazendo com que a aplicação fosse dirigida por eventos. Na abordagem, a ocorrência

de um evento (mensagem recebida) aciona um comportamento relacionado (método de uma classe da aplicação).

Na definição das classes citadas, utilizou-se a biblioteca OnOff[7] para controlar o estado das portas físicas do minicomputador. Estas portas são um conjunto de pinos, chamados GPIO (*General Purpose Input Output*), que podem ser ligados, transmitindo corrente elétrica, ou desligados. Por meio das funções providas pela biblioteca OnOff, é possível tanto verificar quanto alterar o estado das portas, de forma que os componentes conectados sejam acionados.

No processo de implementação, foram aplicadas técnicas de Desenvolvimento Dirigido por Testes (*Test Driven Development*, TDD). Elas consistem na definição de testes automatizados para funcionalidades que ainda não foram implementadas. Cada funcionalidade é, então, programada usando os respectivos testes como critérios de validação. Dentre outras vantagens, tais técnicas auxiliam na compreensão prévia do que será desenvolvido e na diminuição do tempo destinado à busca manual de problemas. Assim, pode-se testar rapidamente o funcionamento do programa, tendo auxílio na busca por alterações indevidas e na refatoração do código. Obtém-se, deste modo, potenciais aumento de produtividade e de homogenização da qualidade do software produzido [Erdogmus et al. 2005].

3.3. Camada de Controle

A Camada de Controle concretiza as instruções programadas na interface, primeiramente convertendo-as em corrente elétrica e, depois, em reações no modelo robótico. Constitui-se de um minicomputador Raspberry Pi e do conjunto de portas GPIO disponíveis. Foi optado pelo Raspberry Pi porque ele reúne *hardware* equivalente ao de um computador convencional em uma placa pouco maior do que um cartão de crédito. O minicomputador possui recursos suficientes para suportar a execução de um sistema operacional e também oferece um conjunto de pinos capazes de controlar componentes eletrônicos. Entre os projetos desenvolvidos utilizando essa tecnologia, encontram-se telescópios, *smartphones* e, até mesmo, sistemas que utilizam inteligência artificial no controle de motores de combustão.

Os métodos das classes da Camada de Aplicação utilizam a biblioteca OnOff para controlar o estado das portas GPIO do minicomputador. Ao alterar o estado dessas portas, o minicomputador envia correntes elétricas aos componentes nelas conectados. A corrente enviada pelas portas, todavia, não basta para alimentar alguns componentes. Por esse motivo, o minicomputador e os componentes eletrônicos conectados precisam da alimentação externa de uma bateria integrada ao modelo robótico. Assim sendo, a corrente enviada pelas portas do controlador apenas aciona os componentes, que são alimentados por outra fonte de energia.

3.4. Camada de Componentes

Os componentes eletrônicos conectados ao Raspberry Pi, tais como motores de corrente direta, sensores de proximidade, botões e leds, formam esta camada. O conjunto proporciona que o modelo robótico construído consiga se movimentar através

[7]Repositório da biblioteca OnOff: `github.com/fivdi/onoff`

de uma sala, desviando obstáculos. Foram escolhidos componentes genéricos e acessíveis, de modo que tanto a montagem quanto a replicação do modelo robótico fossem facilitadas. Conforme antedito, blocos encaixáveis na interface representam as ações possíveis para cada componente. Os blocos denotam instruções que, quando executadas, são recebidas pelo minicomputador e traduzidas em correntes elétricas. Então, as portas GPIO transmitem essas correntes, realizando (alter)ações físicas no modelo robótico.

4. Resultados

A ferramenta descrita pelo artigo compreende um modelo robótico e uma aplicação web para programá-lo visualmente. Os componentes do modelo, conectados ao minicomputador Raspberry Pi, são: dois motores com rodas, um sensor de proximidade, um led e um botão. A interface da aplicação provê blocos que representam estruturas de controle e instruções para comandar cada componente. Empilhando esses blocos, consegue-se programar o comportamento do modelo robótico. A Figura 3 traz o exemplo de um programa construído pela interface da aplicação.

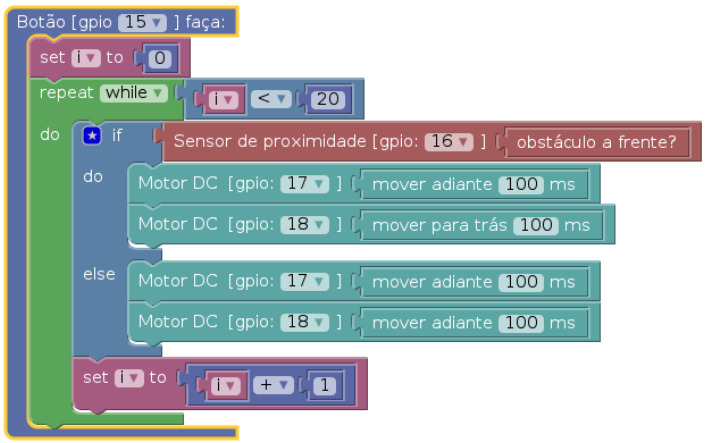

Figura 3. Exemplo de programa definido utilizando a interface da aplicação.

O exemplo instrui o comportamento do modelo robótico para que verifique se existe algum obstáculo a sua frente, girando sobre seu próprio eixo, caso positivo, ou seguindo em frente, caso negativo. O bloco mais externo representa o acionamento de um botão conectado à porta GPIO número 15. As instruções contidas nesse bloco, como indicado pela semântica dos formatos e rótulos, são disparadas mediante o pressionamento do botão físico no modelo robótico. A primeira delas define uma variável, utilizada na sequência para contar as repetições de uma estrutura *while-do*. Dentro desta estrutura, o programa verifica o resultado da instrução "obstáculo à frente", ligada ao bloco que representa um sensor de proximidade conectado à GPIO 16. Caso o sensor indique que exista um obstáculo, as instruções "mover adiante" e "mover para trás" são utilizadas para acionar, em direções opostas, os motores conectados às portas GPIO 17 e 18 (fazendo com que o modelo robótico gire sobre seu próprio eixo). Do contrário, ou seja, não havendo obstáculos, os motores são acionados para que movam o modelo adiante.

A fim de incentivar a experimentação da ferramenta, bem como o emprego didático em oficinas, o código da aplicação será disponibilizado em conjunto com as especificações para a montagem do modelo robótico, além de tutoriais contendo exemplos de utilização. O material tem o objetivo de orientar a conexão e posterior programação dos componentes, apresentando complexidade progressiva (dos leds e botões até os motores e sensores de proximidade). Mesmo sendo o primeiro material de divulgação, já se fornecem condições para que usuário programe o modelo robótico, de modo que ele se movimente desviando obstáculos. Proporciona-se isso por meio do conhecimento oferecido pelas experimentações feitas com os componentes individuais, remetendo o saber a uma resolução integradora.

5. Considerações Finais e Trabalhos Futuros

Foi apresentada, pelo artigo, uma ferramenta didática *open-source*, voltada à experimentação em robótica educacional. Nela, um minicomputador Raspberry Pi controla os componentes do microrrobô, além de atuar como hospedeiro de uma aplicação web para programá-lo visualmente. Objetivou-se que os experimentos pudessem ser facilmente replicados, inclusive considerando a montagem do microrrobô. Por meio da distribuição livre, espera-se atrair mais usuários e colaboradores para o projeto.

O projeto se encontra em fase final de desenvolvimento. A arquitetura descrita está totalmente implementada e funcional. Atualmente, são investidos esforços em ajustes que atendam às peculiaridades de cada componente de *hardware* conectado ao Raspberry Pi. Na sequência, serão revisados os tutoriais e exemplos que exploram a ferramenta. Prevê-se que a versão inicial do RASPIBLOCOS seja divulgada no primeiro semestre de 2016.

O código da aplicação e o conjunto de exemplos serão disponibilizados sob a Licença Apache[8] de software *open-source*, no repositório do projeto. A distribuição livre do código e das especificações para montagem do microrrobô deve encorajar interessados a utilizarem e contribuírem para o desenvolvimento da ferramenta. Potencializa-se, desta forma, a realização de experimentos em robótica educacional, a visibilidade da iniciativa, bem como a maturidade e expansão do projeto.

Por meio da coleta de informações diante da utilização dos tutoriais, será delineada a proposta de oficinas utilizando a ferramenta. A metodologia das oficinas considera aquelas aplicadas nos experimentos de Saygin et al. [2012], Saleiro et al. [2013] e Dziabenko et al. [2012]. As propostas integram conceitos como ciclo de aprendizagem ativa, aprendizagem baseada em problemas e *serious games*.

A metodologia proposta para a realização das oficinas inicialmente prevê as quatro etapas seguintes: **(1)** apresentar os objetivos e os conceitos envolvidos na realização da oficina, e avaliar o conhecimento em programação dos alunos; **(2)** orientar os alunos na programação individual dos componentes, progredindo em complexidade; **(3)** desafiar os alunos a integrarem esse conhecimento, fazendo o modelo robótico desviar obstáculos; e **(4)** realizar pesquisas de opinião e reavaliar o conhecimento dos alunos.

[8]Licença Apache: `www.apache.org/licenses/LICENSE-2.0`

Referências

Alves, R. M. and Sampaio, F. F. (2014). Duinoblocks: Desenho e implementação de um ambiente de programação visual para robótica educacional baseado no hardware arduino. In *Anais dos Workshops do Congresso Brasileiro de Informática na Educação*, volume 3.

Dziabenko, O., García-Zubia, J., and Angulo, I. (2012). Time to play with a microcontroller managed mobile bot. In *Global Engineering Education Conference (EDUCON), 2012 IEEE*, pages 1–5. IEEE.

Erdogmus, H., Morisio, M., and Torchiano, M. (2005). On the effectiveness of the test-first approach to programming. *IEEE Transactions on Software Engineering*, 31(3):226–237.

Grout, V. and Houlden, N. (2014). Taking computer science and programming into schools: The glyndŵr/bcs turing project. *Procedia-Social and Behavioral Sciences*, 141:680–685.

Iturrate, I., Martin, G., Garcia-Zubia, J., Angulo, I., Dziabenko, O., Orduña, P., Alves, G., and Fidalgo, A. (2013). A mobile robot platform for open learning based on serious games and remote laboratories. In *Engineering Education (CISPEE), 2013 1st International Conference of the Portuguese Society for*, pages 1–7. IEEE.

Papert, S. (1980). *Mindstorms: Children, Computers, and Powerful Ideas*. Basic Books, Inc., New York, NY, USA.

Pasternak, E. (2009). Visual programming pedagogies and integrating current visual programming language features. Master's thesis, Robotics Institute, Carnegie Mellon University.

Saleiro, M., Carmo, B., Rodrigues, J. M., and du Buf, J. H. (2013). A low-cost classroom-oriented educational robotics system. In *Social Robotics*, pages 74–83. Springer.

Saygin, C., Yuen, T., Shipley, H., Wan, H., and Akopian, D. (2012). Design, development, and implementation of educational robotics activities for k-12 students. In *Proceedings of 2012 American Society for Engineering Education Annual Conference & Exposition*.

Vandevelde, C., Saldien, J., Ciocci, C., and Vanderborght, B. (2013). Overview of technologies for building robots in the classroom. In *International Conference on Robotics in Education*, pages 122–130.