UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ - UTFPR
CÂMPUS GUARAPUAVA
CURSO SUPERIOR DE TECNOLOGIA EM SISTEMAS PARA INTERNET

EDUARTH HEINEN

RASPIBLOCOS: AMBIENTE DE PROGRAMAÇÃO DIDÁTICO BASEADO EM RASPBERRY PI E BLOCKLY

PROJETO DE TRABALHO DE CONCLUSÃO DE CURSO

GUARAPUAVA

2015

EDUARTH HEINEN

RASPIBLOCOS: AMBIENTE DE PROGRAMAÇÃO DIDÁTICO BASEADO EM RASPBERRY PI E BLOCKLY

Projeto de Trabalho de Conclusão de Curso , apresentado a disciplina de Trabalho de Conclusão de Curso 1 do Curso Superior de Tecnologia em Sistemas para Internet da Universidade Tecnológica Federal do Paraná - UTFPR , como requisito parcial para obtenção do título de Tecnólogo em Sistemas para Internet

Orientador: Prof. Dr. Eleandro Maschio

Coorientador: Prof. Dr. Diego Marczal

GUARAPUAVA

2015

RESUMO

HEINEN, Eduarth. RASPIBLOCOS: Ambiente de Programação Didático Baseado em Raspberry Pi e Blockly. 32 f. Projeto de Trabalho de Conclusão de Curso – Curso Superior de Tecnologia em Sistemas para Internet, Universidade Tecnológica Federal do Paraná - UTFPR. Guarapuava, 2015.

Este projeto 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.

Palavras-chave: Educação, Robótica

ABSTRACT

HEINEN, Eduarth. RASPIBLOCOS: Educational Programming Environment Based on Raspberry Pi and Blockly. 32 f. Projeto de Trabalho de Conclusão de Curso – Curso Superior de Tecnologia em Sistemas para Internet, Universidade Tecnológica Federal do Paraná - UTFPR. Guarapuava, 2015.

This project 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 wifis, 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.

Keywords: Education, Robotics

LISTA DE FIGURAS

SUMÁRIO

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.

A popularização de brinquedos programáveis, como a linha LEGO Mindstorms , além de computadores de placa única, como Arduino e Raspberry Pi, 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 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; 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.

1.1 OBJETIVOS

1.1.1 Objetivo geral

Desenvolver um ambiente de programação visual utilizando a biblioteca Blockly e o minicomputador Raspberry Pi, de maneira que reações físicas no modelo eletrônico decorram de instruções representadas pelos blocos encaixáveis na interface.

1.1.2 Objetivos específicos

- Construir uma aplicação web com interface baseada na biblioteca Blockly;
- Integrar a aplicação e o minicomputador Raspberry Pi;
- Programar blocos da interface para controlar leds, motores de corrente contínua, sensores de proximidade e botões;
- Propor a realização de oficinas de programação com alunos que possuam programação de computadores em seus currículos;
- Divulgar a iniciativa open-source, como também disseminar os resultados em publicações e eventos da área.

1.2 ESTRUTURA DA PROPOSTA

Esta seção apresenta a divisão do conteúdo apresentado pelo projeto. O capítulo 2 corresponde a Resenha Literária, dividida em: Seção 2.1, que apresenta o Estado da Arte da área da robótica educacional; e Seção 2.2, que expõe a Fundamentação Teórica, analisando conceitos relacionados ao projeto e os principais componentes da arquitetura. O capítulo 3 descreve os Procedimentos Metodológicos previstos. O processo de desenvolvimento do projeto é abordado no capítulo 4. Este, expõe a Arquitetura da Solução Proposta na Seção 4.1, os Resultados Parciais do projeto na Seção 4.2, e, por fim, o Cronograma e o Estado Atual do projeto frente a ele na Seção 4.3. Concluindo o projeto, o Diferencial Tecnológico é abordado no capítulo 5, e as Considerações Finais, no Capítulo 6.

2 RESENHA LITERÁRIA

O presente capítulo apresenta os trabalhos correlatos e a fundamentação teórica envolvida na concepção do projeto.

2.1 ESTADO DA ARTE

O desenvolvimento de um conjunto de robótica voltado a educação, com um controlador Raspberry Pi e interface baseada em Blockly, foi proposto anteriormente por Saleiro et al. (2013). A equipe de pesquisadores desenvolveu um sistema que 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 90° a 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 as 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 adotas-

sem 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, são construídos sobre uma placa impressa por encomenda, necessária por causa do tamanho reduzido dos robôs, permitindo a conexão entre os componentes. Além disso, as instruções disponíveis na interface são limitadas a arquitetura simples dos robôs. Estes, pelo *hardware* que possuem, são capazes somente de mover-se seguindo as linhas da grade, detectando cruzamentos.

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 a Internet. Com isso, a pesquisa pretendeu possibilitar que estudantes de qualquer lugar do mundo se beneficiassem com o projeto desenvolvido.

O ambiente criado pelos pesquisadores é enriquecido pelo enredo apresentado junto do exercício de programação. Baseado no enredo proposto anteriormente por Dziabenko et al. (2012), o cenário apresentado ao usuário consiste em um pouso forçado em um planeta alienígena. Como integrante da tripulação da nave, o usuário 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, a arquitetura dos robôs requer uma placa impressa sob encomenda para a conexão dos componentes de *hardware*. 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 todosos 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.

2.2 FUNDAMENTAÇÃO TEÓRICA

Esta seção resulta da análise da bibliografia relacionada a pesquisa sobre robótica educacional, dificuldades no ensino-aprendizagem de programação de computadores e componentes da arquitetura do projeto.

2.2.1 Micromundos e robótica educacional

Em contraste com o aprendizado espontâneo que ocorre durante a infância, durante o qual, sem instrução específica, a criança aprende a falar, a se localizar no espaço e a argumentar; o ambiente escolar, mesmo empregando muito mais tempo e esforço dos alunos e professores, não alcança a mesma eficiência. De acordo com Papert, isso ocorre devido ao estímulo do ambiente para a aquisição dos conhecimentos, como ao conversarem com outras pessoas ou

explorarem o espaço onde vivem, as crianças constroem pela experimentação as teorias que baseiam a sua percepção da realidade. No ambiente escolar, de maneira diferente da aprendizagem natural, alguns conteúdos são apresentadas de maneira dissociada da realidade e da experiência dos alunos. O autor discorre sobre os bloqueios culturais surgidos da frustração criada por exigir dos alunos que compreendam conceitos de matemática, física e outras disciplinas complexas, em um ambiente que falha em fornecer senso de propósito e bases tangíveis para que se construa o conhecimento. (PAPERT, 1980)

Papert propõe a criação de micromundos, ambientes semanticamente ricos, que incorporam os conceitos que buscam apresentar, oferecendo meios de interação por meio de quais os alunos podem experimentar e assim construir o próprio conhecimento (KRYNSKI, 2007). Um exemplo de micromundo seria um ambiente que possibilite manipular a massa de planetas em um sistema planetário, permitindo observar o resultado na gravidade que cada planeta exerce e na órbita que percorre. O autor sugere que micromundos apoiam a construção de teorias parcialmente verdadeiras, como parte do processo natural de aprendizado. Durante este processo, as teorias do aprendiz são confrontadas com suas experiências no ambiente com que interage, sendo confirmadas, aprimoradas, substituídas ou descartadas, assim como as teorias que surgem das experiências das crianças durante a infância.

O micromundo projetado por Papert utiliza robôs que se assemelham a tartarugas e que podem ser programados para moverem-se enquanto desenham seu percurso. O autor apresenta uma sucessão de exemplos de aplicações desse ambiente, que variam do ensino de programação a demonstração das leis de Newton em física.

Assim como os micromundos descritos por Papert, o campo de pesquisa da robótica educacional almeja a construção de ambientes que apoiem a experimentação e a construção autônoma de teorias pelo aprendiz. Saygin et al. (2012) afirma que a robótica educacional facilita a realização do aprendizado ativo, aprimorando o pensamento crítico e o raciocínio, além de estimular o interesse dos alunos na abordagem de temas complexos. A revisão bibliográfica realizada por Benitti (2012) sumariza as proficiências desenvolvidas pelo contato com robótica na educação, sendo: ***(i)*** habilidades de raciocínio (observação, estimação e manipulação); ***(ii)*** habilidades de processo científico/abordagens de resolução de problemas (como solução da avaliação, geração de hipóteses, teste de hipóteses e controle de variáveis); e ***(iii)*** interação social/trabalho em equipe (BENITTI, 2012).

2.2.2 Obstáculos no ensino-aprendizagem de programação

Desde seu surgimento, a programação de computadores tem sido aplicada pioneiramente por cientistas, engenheiros e matemáticos na pesquisa científica. Estes pesquisadores usualmente adquiriam a proficiência em programação de computadores pelo estudo da sintaxe de uma linguagem de programação. De maneira que a transmissão deste conhecimento seguiu por anos o mesmo modelo. Conforme a Ciência da Computação avançou como disciplina acadêmica, surgiu a necessidade de tratá-la como uma habilidade independente e não mais como uma ferramenta. Consequentemente inspirando o surgimento da pesquisa sobre os obstáculos na aprendizagem e a busca por métodos de ensino da programação de computadores (FINCHER, 1999).

A revisão literária conduzida por Robins et al. aponta como principais dificuldades de programadores inexperientes:

- **A dificuldade inerente em aprender programação:** Boulay (1986) identifica cinco domínios sobrepostos e potenciais fontes de dificuldades que devem ser superadas por programadores inexperientes: ***(i)*** a consciência da função de um programa de computador e suas aplicações; ***(ii)*** o modelo da *notional machine*, ou seja, a abstração do funcionamento do computador enquanto interpretador de determinada linguagem de programação; ***(iii)*** a sintaxe e a semântica desta linguagem; ***(iv)*** a tradução do programa em esquemas e planos; ***(v)*** e habilidades de planejamento, desenvolvimento, teste e solução de problemas.

- **A construção dos modelos e o desenvolvimento das habilidades necessárias:** Além do modelo da *notional machine*, necessário para a compreensão do funcionamento do computador, é necessário construir os modelos do propósito deste programa e do ambiente em que será aplicado. “Um programa em execução é uma espécie de mecanismo e leva-se um longo tempo para aprender a relação entre um programa na página e o mecanismo que ele descreve"(BOULAY, 1986). Brooks (1983) descreve a construção de um programa de computador como o mantimento de uma série de modelos sobrepostos e conectados de domínios de conhecimento, e posterior reconstrução de aspectos destes modelos no domínio do programa. O desenvolvimento de um programa que controle a situação de pacotes em uma empresa de entregas, por exemplo, envolveria a construção de modelos para o funcionamento da empresa, para o progresso de um pacote até a entrega, passando pela abstração das informações necessárias, para os esquemas e planos que descrevem o programa, por diversos domínios até a construção do domínio do programa. Todos estes devem ser mantidos, relacionados e comparados pelo programador. Além disso, em

situações cujo comportamento do programa difere do esperado, é necessário construir modelos do funcionamento desejado e do funcionamento real do programa de maneira que possam ser comparados em busca de erros ou problemas.

- **A fragilidade do conhecimento e a ineficiência das estratégias utilizadas:** Entre as dificuldades relacionadas a sintaxe e a utilização das estruturas de programação, entre programadores inexperientes, observam-se: problemas com declaração e atribuição de valores a variáveis, dúvidas sobre a atualização dos contadores durante os ciclos de estruturas de repetição, implementações incorretas de recursão e dificuldades em controlar e acompanhar o estado do programa. A concepção do computador como realizador das instruções dentro de um conjunto de regras rígidas, conceito crucial para a aprendizagem da programação, geralmente é obtido somente depois de certa experiência. Ademais, os autores observam que programadores inexperientes frequentemente possuem o conhecimento da sintaxe, porquanto não dominam as estratégias de construção do programa, não sendo capazes de combinar as estruturas da linguagem que conhecem em programas coerentes.

- **O comportamento do aprendiz:** Perkins et al. (1986) distingue comportamentos característicos entre programadores inexperientes em dois perfis principais: *"stoppers"* e *"movers"*. Estudantes com perfis *"movers"*, ao encontrarem problemas ou dificuldades na representação dos conceitos, modificam e experimentam com o próprio código e modelos até encontrarem a solução. Estudantes *"stoppers"*, por sua vez, simplesmente param. Bloqueios culturais, como os observados por Papert no ensino de matemática e física, podem ser a causa das reações emocionais negativas, apresentadas por este perfil de estudantes ao encontrar obstáculos ou dificuldades.

Entre as sugestões de Soloway e Spohrer (1989) para a construção de ambientes que apoiem o ensino-aprendizagem de programação estão:

- linguagens gráficas que tornem explícita a sequência de instruções dos programas;
- padrões de nomeação simples e consistentes;
- modelos do computador em tempo de execução;
- animação dos estados do programa, exibindo ao aprendiz as alterações de estado que seriam invisíveis de outra forma;;
- princípios de design baseados em metáforas espaciais;

- e a gradual retirada dos apoios ao aprendiz.

Robins et al. (2003) argumenta que, antes de analisar as habilidades que diferem principiantes e programadores experientes, é necessário direcionar esforços no sentido de estimular o surgimento de *"effective novices"*, como denominados pelos autores os estudantes que empregam estratégias eficientes na aprendizagem. Os autores observam que grande parte do esforço empregado em lições de programação é direcionado ao ensino da sintaxe e das estruturas de linguagens programação. Argumentando que mais cruciais que o conhecimento são as estratégias para a aplicação deste na compreensão e construção de programas. Concluem que mais atenção deve ser delegada no ensino de estratégias em cursos introdutórios, programando-se "ao vivo" junto dos alunos e discutindo as estratégias empregadas na solução de cada problema, de forma que os estudantes se sintam encorajados e comprometidos com a própria aprendizagem.

2.2.3 Linguagens de programação visual

Linguagens de Programação Visual são recursos que buscam apoiar a construção e a interpretação de programas de computador. Utilizam blocos encaixáveis ou fluxogramas de componentes conectáveis, por exemplo, para representar instruções de linguagens de programação. Estes elementos gráficos atuam como metáforas visuais, aproximando conceitos de linguagens de programação de atividades familiares aos usuários. Reduzem, dessa maneira, a carga cognitiva sobre os alunos, contribuindo para a construção de projetos significativos (PASTERNAK, 2009).

O uso de metáforas visuais, a simplificação da sintaxe e as limitações de conexão, características de linguagens de programação visual, diminuem as barreiras, permitindo ao estudante manter o foco na construção dos programas e na solução dos problemas. Busca-se, também, impedir a construção de código incoerente pela limitação das conexões possíveis (PASTERNAK, 2009). De maneira que um bloco que representa uma variável, por exemplo, só possa ser conectado a outro que retorne um valor coerente. Dessa forma, eliminam-se quase a totalidade dos erros sintáticos, uma vez que 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 sintaxe mais comuns (*missing semicolon*, *unknown variable*, *illegal start of expression*, *unknown class* e *bracket expected*) correspondem a mais da metade dos erros de compilação cometidos por programadores inexperientes.

2.2.4 Componentes da arquitetura da aplicação

Esta seção descreve brevemente os principais componentes que constituem a arquitetura da aplicação.

2.2.4.1 Blockly

Blockly é um conjunto de bibliotecas destinado a construção de editores para programação visual. Sua aparência e funcionamento foram inspirados no ambiente Scratch, desenvolvido com o propósito de ensinar programação a crianças, por meio de atividades como a criação de animações e jogos interativos (VANDEVELDE et al., 2013).

A biblioteca oferece um conjunto de blocos que representam instruções de linguagens de programação, como: declaração de variáveis, operações matemáticas, lógicas e relacionais, estruturas de controle (condicionais e de repetição), entre outras. Estes blocos podem ser empilhados formando pequenos programas, que ao serem executados pelo ambiente, são traduzidos nas linguagens Javascript, Python ou Dart.

Como um ambiente de programação visual, Blockly aplica metáforas para aproximar conceitos de linguagens de programação a tarefas familiares aos usuários. Os programas são construídos encaixando blocos que representam trechos de código sintaticamente correto, apoiando o programador inexperiente em uma das suas principais dificuldades, os erros de sintaxe. Assegura, também, a construção de programas coerentes, limitando os encaixes possíveis entre os blocos. Somente blocos graficamente compatíveis podem ser encaixados e estas limitações podem ser estabelecidas durante a definição dos blocos, permitindo até que o tipo de dado que a instrução deve receber seja especificado.

Blockly também permite que novos blocos sejam adicionados, representando as capacidades do ambiente e disponibilizando novas instruções para a construção de programas. A definição destes blocos é realizada pela adição de dois arquivos ao diretório de blocos. Estes descrevem a aparência (conexões, opções e comportamento na interface) e o código gerado pela tradução do bloco durante a execução do programa. A “Block Factory”, disponibilizada junto da biblioteca, simplifica a criação de novos blocos, permitindo que as características e definições necessárias para a especificação de um bloco (conexões possíveis, cores, estrutura) sejam definidos empilhando blocos que representam cada uma dessas características na interface. A página exibe ao usuário o código que capaz de construir a aparência do novo bloco definido e um modelo de “gerador de código”, como são denominados os arquivos que traduzem a representação gráfica em linguagem de programação.

2.2.4.2 Raspberry pi

Raspberry Pi é um minicomputador que reúne *hardware* equivalente ao de um computador convencional em uma placa pouco maior do que um cartão de crédito. Por isso é capaz de suportar a execução de um sistema operacional, ponto em que difere de controladores de placa única como o Arduino.

O conjunto de pinos GPIO (*General Purpose Input Output*), integrado ao minicomputador, possibilita controlar componentes eletrônicos conectados. Estes pinos podem ser ligados ou desligados pelo sistema operacional, transmitindo corrente elétrica ou interrompendo-a. Dessa forma pode controlar diretamente sensores, botões, leds e outros componentes. Apesar de não conduzir energia suficiente para alimentar motores de corrente direta, pode acionar placas controladoras capazes de comandar estes motores. Oferece ainda entradas USB, por meio das quais dispositivos como adaptadores wi-fi e webcams podem ser conectados, permitindo a construção de modelos robóticos com capacidades de reconhecimento de imagem e conexão wi-fi (VANDEVELDE et al., 2013).

Raspbian[1], é uma versão otimizada para o *hardware* do Raspberry Pi do sistema operacional Debian. Permite configurar o minicomputador como servidor de aplicações web, enquanto dirige o comportamento das portas GPIO através das ferramentas integradas ao sistema operacional. Como observado anteriormente, esta é a principal característica em que o minicomputador difere de controladores de placa única como o Arduino. Nestes, o código executado deve ser previamente compilado, e as aplicações frequentemente dependem de um servidor, como observado na revisão da literatura. Em contrapartida, por não utilizarem um sistema operacional, microcontroladores de placa única oferecem controle em tempo-real dos componentes, aspecto que o Raspberry Pi não assegura.

Pelo seu relativo baixo custo[2] e popularidade entre acadêmicos e hobbistas, o minicomputador atingiu, em junho de 2014, 3 milhões de unidades vendidas. Entre as experiências construídas encontram-se telescópios, *smartphones* e, até mesmo, sistemas que utilizam inteligência artificial no controle de motores de combustão.

2.2.4.3 Node.js e express

Conforme páginas e aplicações web tornaram-se mais dinâmicas, navegadores modernos buscaram refinar suas implementações da linguagem Javascript. Este processo resultou no

[1] Raspbian – https://www.raspbian.org/

[2] Raspberry Pi 2 Model B - £30,00 em http://swag.raspberrypi.org/collections/frontpage/products/raspberry-pi-2-model-b

surgimento do *Google Chrome's V8 Javascript Engine*[3], interpretador capaz de compilar aplicações Javascript em linguagem nativa (C/C++), de maneira transparente e ágil (BROWN, 2014). Em 2009, aliando o interpretador *Chrome's V8* e a biblioteca de processamento de eventos *libev*, Ryan Dahl criou a plataforma Node.js (WANDSCHNEIDER, 2013).

Node.js é uma plataforma de aplicações Javascript, assíncrona e dirigida por eventos, para o lado do servidor. Baseia seu funcionamento em um *event loop*, ou ciclo de eventos. Este recebe e processa requisições, delegando operações de e/s (entrada e saída de dados) ao sistema operacional. Estas, quando concluídas, executam uma função de *callback*, enviando a resposta das operações de volta ao ciclo de eventos. As requisições e o código da aplicação, são executadas de maneira *single threaded*, diferentemente das operações de e/s realizadas de maneira assíncrona pelo sistema operacional (CANTELON et al., 2013).

Por realizar operações de e/s de maneira assíncrona, aplicações Node.js podem continuar tratando requisições enquanto aguardam o *callback* que sinaliza a conclusão das operações. Isto permite que o servidor mantenha conexões abertas e processe inúmeras requisições, consumindo pouca memória. Dessa maneira, Node.js é a plataforma ideal para a construção de aplicações *DIRT* – acrônimo para *data-intensive real-time* – capazes de responder quase instantâneamente a um grande número de usuários concorrentes (CANTELON et al., 2013).

Express é um *framework* para a construção de aplicações Node.js. É descrito como minimalista e flexível, proporcionando um conjunto robusto de funcionalidades para a construção de aplicações web (BROWN, 2014). Torna a construção das aplicações mais simples, pois oferece gerenciamento de layout, utilidades para tratar requisições, transferir arquivos, realizar o roteamento de URLs, entre outros (CANTELON et al., 2013). Possui também uma funcionalidade de *scaffolding*, semelhante a utilizada pela plataforma *Ruby on Rails*, capaz de gerar a estrutura básica de arquivos e pastas para uma aplicação Express.

[3] https://code.google.com/p/v8/

3 PROCEDIMENTOS METODOLÓGICOS

Conforme proposta aprovada no início do ano letivo, o desenvolvimento do projeto foi dividido entre os seguintes passos metodológicos:

1. **Construir a aplicação web inicial:** A aplicação que atuará como interface de programação do experimento está sendo desenvolvida utilizando o framework Express e a plataforma Node.js. A biblioteca Blockly também é utilizada, com a finalidade de prover um ambiente visual de programação por meio de blocos encaixáveis. Tem-se, ao fim desta etapa, um protótipo funcional do ambiente;

2. **Integrar a aplicação ao minicomputador Raspberry Pi:** A biblioteca OnOff oferece um conjunto de métodos que permitem controlar o estado das portas GPIO do Raspberry Pi. A classe responsável por receber as instruções da interface e controlar as portas físicas do minicomputador está sendo construída utilizando estes métodos. Funções como "aguardar x segundos" e "executar rotina" serão adicionadas a mesma classe. Também serão instalados os programas necessários para hospedar a aplicação no minicomputador. Desta etapa resultará um protótipo capaz de acionar leds e motores de corrente direta utilizando estruturas de controle ("se-então", "enquanto-faça") e blocos para as portas GPIO do Raspberry Pi;

3. **Programar blocos para representar componentes e instruções específicos da aplicação:** Por meio da especialização do código responsável pelo controle das portas GPIO, encontram-se em desenvolvimento classes e blocos que representem o funcionamento de leds, motores de corrente direta, botões e sensores de proximidade. Esta camada de abstração permitirá ao usuário fornecer instruções de mais alto nível aos componentes do minicomputador, tais como "andar" para motores, ou "quando a distância for menor do que 30 cm" para sensores de proximidade. Tem sido aplicadas técnicas de Desenvolvimento Dirigido por Testes ou TDD. Esta metodologia, além de fornecer uma ferramenta para testar a aplicação de maneira instantânea e abrangente, ajuda a constituir a documentação inicial do projeto, formada pelas especificações que cada teste descreve. A construção

do modelo robótico tem acompanhado o desenvolvimento. Com isso, a definição de cada classe deve é seguida da adição do respectivo componente ao modelo. O resultado desta etapa consistirá em um protótipo que ofereça blocos com instruções específicas para cada componente eletrônico suportado pelo experimento. Tratando-se do primeiro protótipo testável, em que, utilizando os blocos específicos e as classes para motores de corrente direta e sensores de proximidade, será possível construir e programar um robô capaz de se mover por uma sala evitando obstáculos;

4. **Propor a realização de oficinas de programação:** Será analisada a viabilidade de realizar oficinas com alunos de graduação em cursos que incluam disciplinas de programação em seus currículos, buscando avaliar o potencial didático e refinar o funcionamento da aplicação. A efetivação das oficinas é sujeita a fatores como a disponibilidade de alunos interessados, recursos para aquisição dos equipamentos, além do trâmite de importação destes. Presume-se que poderão haver dificuldades de interpretação das instruções representadas pelos blocos. Assim, as oficinas também devem fornecer indícios desses problemas e de como resolvê-los. Os dados obtidos serão analisados em busca de indícios que apoiem o atributo didático dado ao experimento;

5. **Divulgar a iniciativa open-source, como também disseminar os resultados em publicações e eventos da área:** A aplicação será distribuída como código aberto, procurando atrair mais número de usuários e colaboradores. Em adição, os resultados alcançados na pesquisa serão analisados e submetidos a eventos da área.

4 DESENVOLVIMENTO DA PESQUISA

O ambiente descrito pelo presente projeto, ilustrado pela figura 1, constitui-se de uma aplicação web hospedada em um minicomputador Raspberry Pi, **(1)** acessível por meio de um navegador web e rede wi-fi, **(2)** capaz de controlar o comportamento de um modelo robótico. A interface da aplicação, baseada na biblioteca Blockly, aplica conceitos de programação visual de forma a apoiar a construção de pequenos programas. Blocos coloridos encaixáveis representam estruturas típicas de linguagens de programação e instruções para os componentes eletrônicos do modelo robótico. Este ambiente permite que, através da interface de programação, o aluno possa construir programas que ao serem executados controlem o comportamento do modelo robótico.

Na execução dos programas, as instruções são traduzidos em mensagens enviadas para a aplicação. Cada uma delas contêm o componente e a operação que deve ser realizada. A aplicação interpreta estas mensagens e aciona, na classe que representa o componente, o método que descreve a ação desejada. Tais métodos encapsulam o código responsável por controlar as portas GPIO do minicomputador de maneira que o componente conectado atue como instruído.

As seções subsequentes descrevem a arquitetura da solução proposta (4.1), os resultados parciais vindos da implementação dela (4.2), além do estágio atual da pesquisa frente ao cronograma definido (4.3).

4.1 ARQUITETURA DA SOLUÇÃO PROPOSTA

A arquitetura do ambiente proposto pode ser dividida em quatro camadas: Interface, Aplicação, Controle e Componentes. A Figura 2 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:

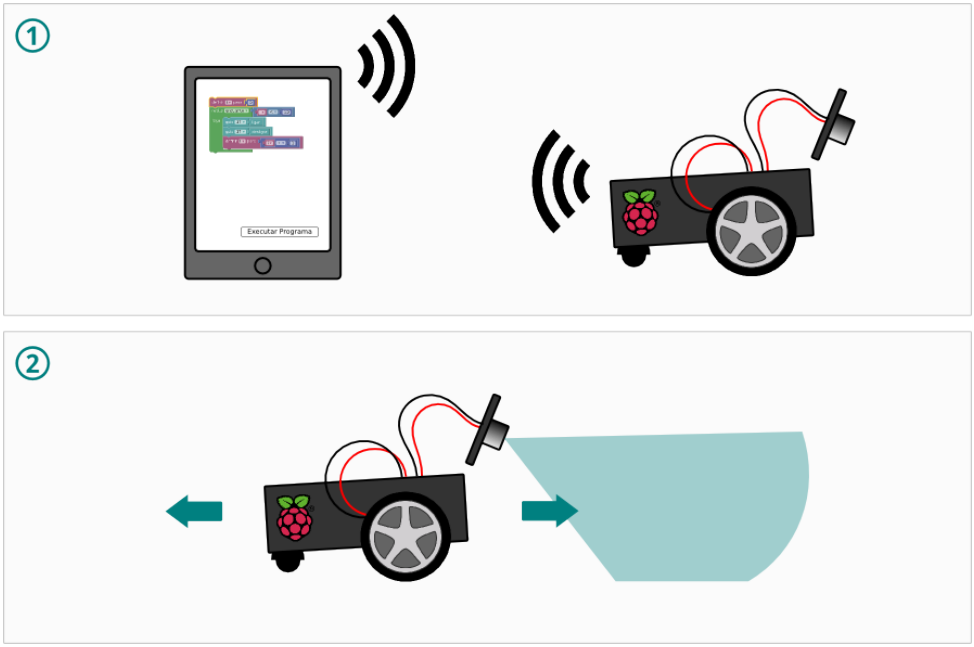

Figura 1: Ilustração do funcionamento do ambiente.

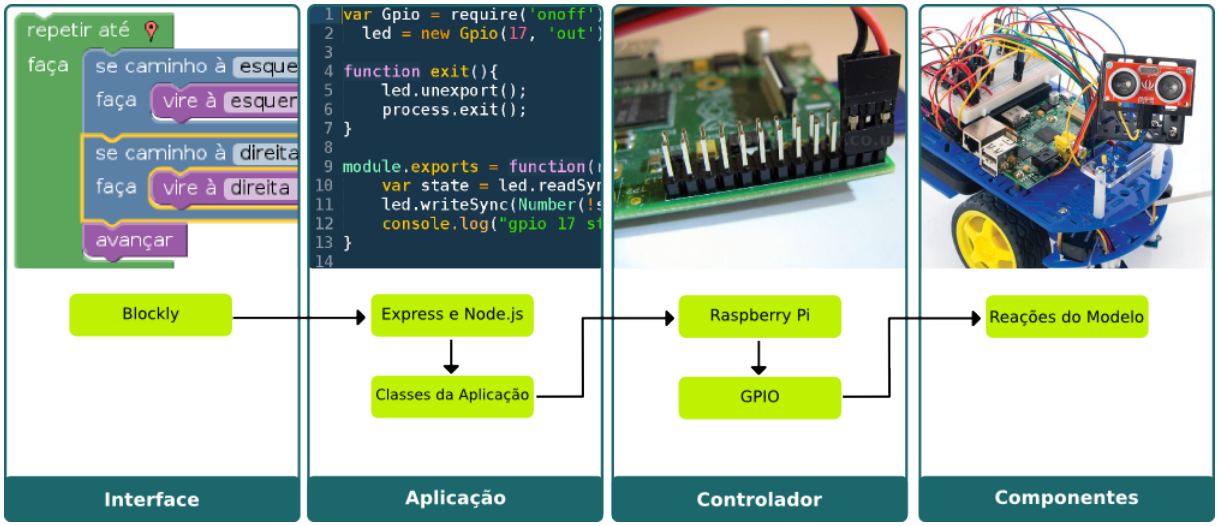

Figura 2: Ilustração da interação entre as camadas da arquitetura da aplicação.

4.1.1 Camada de interface

A interface da aplicação deve ser simples e intuitiva, considerando a interação de usuários com diferentes níveis de experiência com computadores. 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.

Somados aos blocos oferecidos pela biblioteca, blocos significando instruções próprias do modelo robótico serão adicionados, sendo: "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, permitirão que o modelo robótico possa ser programado para que se mova por uma sala desviando obstáculos.

4.1.2 Camada de aplicação:

O uso do minicomputador Raspberry Pi é 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 para as instruções que o modelo robótico é capaz de operar. A execução dos programas construídos aciona os métodos das classes aplicação responsáveis por controlar o componente que representado pela classe. Tais métodos são acionados pela aplicação de acordo com as mensagens recebidas da interface, gerando reações nos componentes eletrônicos do modelo robótico. Para isso, a plataforma Node.js é empregada, em conjunto com o *framework* Express, fazendo com que a aplicação torne-se 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, a biblioteca OnOff[1] é utilizada para controlar o estado das portas físicas do minicomputador. Por meio das funções providas pela biblioteca, é possível tanto verificar quanto alterar o estado das portas, de forma que os componentes conectados sejam acionados.

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

4.1.3 Camada de controle:

A Camada de Controle é responsável por concretizar as instruções programadas na interface, primeiramente convertendo-as em corrente elétrica e, assim, em reações no modelo robótico. Constitui-se de um minicomputador Raspberry Pi e do conjunto de portas GPIO disponíveis.

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.

4.1.4 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 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, as ações possíveis para cada componente serão representadas por blocos encaixáveis na interface. 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.

4.2 OUTROS RESULTADOS PARCIAIS

A construção da aplicação inicial e a integração desta com o minicomputador já foram concluídas. O protótipo atual apresenta a interface baseada na biblioteca Blockly e blocos para controlar o estado das portas GPIO. O bloco para as portas GPIO aceita a conexão de outro que representa as instruções “ligar”, “desligar” e “alterar estado”. Utilizando-os, é possível programar um led conectado em uma das portas do controlador para ligar e desligar por um número determinado de vezes, a Figura 3 apresenta o programa construído.

Esse programa, ao ser executado, faz com que led conectado a porta GPIO selecionada

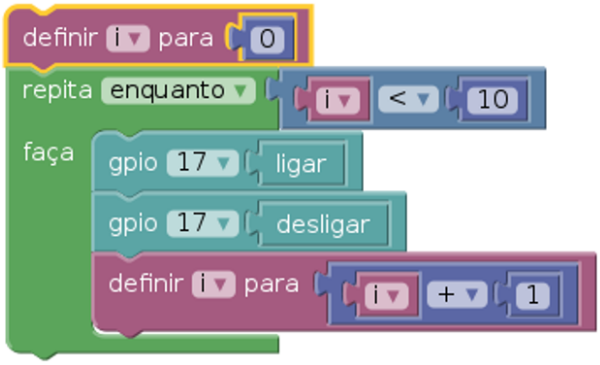

Figura 3: Programa construído utilizando o protótipo da aplicação.

acenda e apague rapidamente. É possível visualizar o componente sendo acionado de acordo com o programa definido na interface, porém a execução ainda é muito rápida (50 ciclos da estrutura de repetição tomam menos de 5 segundos). O funcionamento da instrução "aguarde" precisa ser aprimorado de maneira que permita indicar ao controlador que deve aguardar alguns segundos entre a execução de cada tarefa.

Adicionalmente, registra-se que já foi definido todo o *hardware* que compõe o modelo robótico. Foi realizado um levantamento minucioso de peças de custo acessível e que fossem compatíveis entre si, de modo que a especificação disso permitisse que o modelo seja facilmente replicado. A definição da bateria para a alimentação do conjunto ainda depende da medição do consumo de energia do modelo robótico completo, atrelada à entrega de todos os componentes e a montagem do modelo. Essa tarefa ocupou tempo adicional da pesquisa e a maior parte do *hardware* necessário já foi adquirido, embora uma fração ainda não tenha sido entregue. Segue o detalhamento dos componentes:

- 1x Chassi acrílico para robótica, contendo:

  2x Motores de corrente direta com micro engrenagens;

  2x Rodas plásticas;

- 1x Sensor ultrassônico de distância HC-SR04;
- 1x Controlador para motores de passo e corrente direta L298N;
- 1x Adaptador wi-fi usb Ralink 5370;
- 1x Placa de ensaio para a conexão dos componentes;
- Leds, botões e resistores necessários.

Como resultado subsequente, tendo em vista a disseminação da pesquisa em publicações e eventos da área, foi elaborado e submetido um artigo completo para o CBIE (Congresso Brasileiro de Informática na Educação). Trata-se do maior congresso, em número de participantes, da Computação no Brasil. O artigo foi aprovado na primeira fase de avaliações, sendo que o resultado final será divulgado em 5 de julho de 2015.

4.3 ESTÁGIO ATUAL DE DESENVOLVIMENTO FRENTE AO CRONOGRAMA

Embora se tenha investido tempo adicional no levantamento de componentes de *hardware* para o modelo robótico, como também na escrita do artigo científico, o andamento do cronograma se encontra conforme previsto. Atualmente, o projeto se concentra na programação dos blocos específicos que representam instruções de cada componente. Aprimorando o funcionamento da classe GPIO, adicionada ao ambiente na etapa de desenvolvimento anterior, classes que traduzam o comportamento de motores de corrente direta, botões, leds e sensores encontram-se em desenvolvimento. Os blocos relacionados a essas classes deverão ser adicionados a interface tão logo cada uma delas esteja definida e testada. Como mencionado anteriormente técnicas de TDD já vêm sendo aplicados desde a etapa anterior de desenvolvimento no teste e definição das classes da aplicação. A classe GPIO, a rota que recebe as instruções e as principais funções da aplicação já estão cobertas por testes unitários.

A Figura 4 apresenta o cronograma anteriormente definido para o Trabalho de Conclusão de Curso.

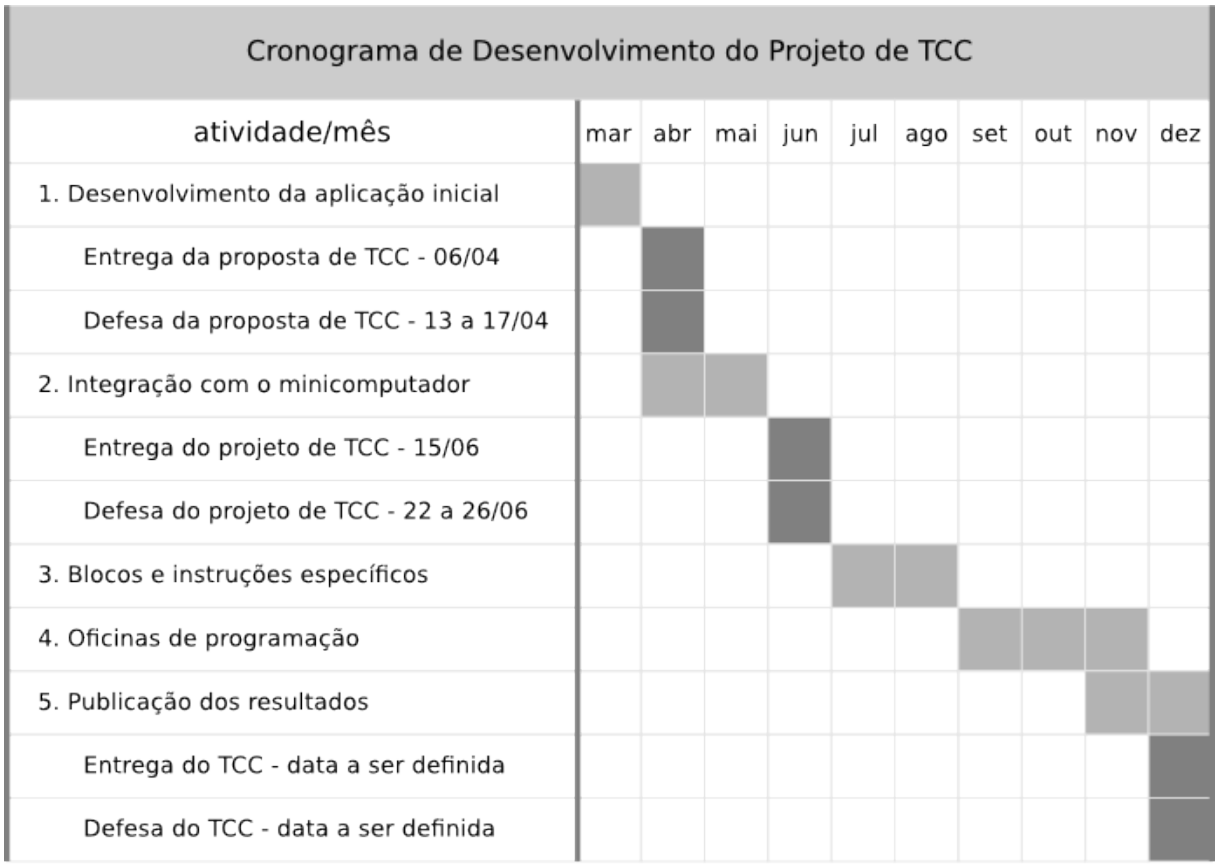

Figura 4: Cronograma de desenvolvimento do Trabalho de Conclusão de Curso.

5 DIFERENCIAL TECNOLÓGICO

A ferramenta apresentada por este projeto difere de tecnologias semelhantes primeiramente por constituir uma iniciativa open-source. Nela, o minicomputador Raspberry Pi controla os componentes do modelo robótico, e atua como hospedeiro de uma aplicação web para programá-lo visualmente. Dessa forma, o usuário pode acessar a interface do ambiente, conectando qualquer dispositivo com navegador web e wi-fi a rede disponibilizada pelo minicomputador, e através dessa interface, programar o comportamento do robô, podendo perceber nas suas ações o resultado das instruções utilizadas.

O código da aplicação deve ser disponibilizado no repositório do projeto sob a Licença Apache de software open-source. Também será disponibilizado um conjunto de tutoriais de utilização do ambiente, ilustrando a conexão e guiando a programação dos componentes. A distribuição livre do código e das especificações para montagem do robô 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.

6 CONSIDERAÇÕES FINAIS

A robótica educativa e a integração da informática em sala de aula têm potencial para mudar a forma como diversas áreas do conhecimento são apresentadas aos estudantes. Os micromundos, sugeridos por Papert, são ambientes que incorporam os conceitos que desejam explorar, oferecendo meios de interação através de quais seus princípios podem ser controlados, permitindo ao aprendiz observar o efeito destas alterações nas interações com o ambiente. Dessa forma, incentivam que o aluno experimente, manipulando as caracteristicas e os conceitos de um ambiente repleto de informação, construindo seu próprio conhecimento. A robótica educativa, influenciada pela pesquisa de Papert, utiliza a tecnologia na criação de experimentos e modelos interativos. É percebida como responsável por incentivar o aperfeiçoamento de habilidades de raciocínio e resolução de problemas, e por alcançar melhores resultados, quando comparada com aulas expositivas tradicionais.

A popularização de minicomputadores e controladores, torna acessível a realização de experiências em robótica e programação. Uma comunidade surgiu ao redor de cada uma dessas tecnologias, compartilhando informações e apoiando a realização de novas experiências. Beneficiando a área robótica na educação pelo decorrente acúmulo e riqueza de informações.

O presente projeto descreveu um ambiente que permite programar o comportamento de um modelo robótico por meio de uma interface de programação visual. Esta interface é acessada utilizando qualquer dispositivo com navegador web e conexão wi-fi. O minicomputador responsável por hospedar a aplicação e controlar os componentes do modelo, oferece uma rede wi-fi pela qual a aplicação web que apresenta a interface pode ser acessada. Os componentes eletrônicos integrados – motores de corrente direta, sensores de proximidade, botões e leds – possibilitam que o modelo robótico seja capaz de se movimentar por uma sala evitando obstáculos, ou navegar através de um labirinto. As ações que cada componente é capaz de realizar, são representadas na interface por blocos que podem ser encaixados uns aos outros formando programas. Estes programas, quando executados, resultam no comportamento do modelo robótico.

A aplicação do ambiente deve ser distribuída sob uma licença *open-source*. Junto do código da aplicação será publicado um conjunto de exemplos de utilização capazes de demonstrar as capacidades do ambiente, guiando a conexão e a programação dos componentes. A utilização destes exemplos apoiará a preparação de oficinas de programação com alunos nos primeiros anos de cursos que integrem programação de computadores em seus currículos. A realização destas oficinas, entretanto, estará condicionada a disponibilidade de recursos e de participantes. Dessa forma o ambiente deve constituir uma solução acessível para a realização de experiências em programação e robótica.

REFERÊNCIAS

ALVES, R. M.; SAMPAIO, F. F. 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**. [S.l.: s.n.], 2014. v. 3, n. 1.

BENITTI, F. B. V. **Exploring the educational potential of robotics in schools: A systematic review**. Elsevier Ltd, abr. 2012. 978–988 p. Disponível em: <http://linkinghub.elsevier.com/retrieve/pii/S0360131511002508>.

BOULAY, B. D. Some difficulties of learning to program. **Journal of Educational Computing Research**, SAGE Publications, v. 2, n. 1, p. 57–73, 1986.

BROOKS, R. **Towards a theory of the comprehension of computer programs**. 1983. 543–554 p.

BROWN, E. **Web Development with Node and Express: Leveraging the JavaScript Stack**. O'Reilly Media, 2014. ISBN 9781491902295. Disponível em: <https://books.google.com.br/books?id=HsLvAwAAQBAJ>.

CANTELON, M. et al. **Node.js in Action**. Manning, 2013. (Running Series). ISBN 9781617290572. Disponível em: <https://books.google.com.br/books?id=JV6rpwAACAAJ>.

DZIABENKO, O.; GARCÍA-ZUBIA, J.; ANGULO, I. Time to play with a microcontroller managed mobile bot. In: IEEE. **Global Engineering Education Conference (EDUCON), 2012 IEEE**. [S.l.], 2012. p. 1–5.

FINCHER, S. What are we doing when we teach programming? **FIE'99 Frontiers in Education. 29th Annual Frontiers in Education Conference. Designing the Future of Science and Engineering Education. Conference Proceedings (IEEE Cat. No.99CH37011**, v. 1, p. 8–12, 1999.

GROUT, V.; HOULDEN, N. Taking computer science and programming into schools: The glyndŵr/bcs turing project. **Procedia-Social and Behavioral Sciences**, Elsevier, v. 141, p. 680–685, 2014.

ITURRATE, I. et al. A mobile robot platform for open learning based on serious games and remote laboratories. In: IEEE. **Engineering Education (CISPEE), 2013 1st International Conference of the Portuguese Society for**. [S.l.], 2013. p. 1–7.

KRYNSKI, E. M. Uma abordagem metacognitiva através de multiplas representações externas para o ensino de programação de computadores. 2007.

PAPERT, S. **Mindstorms: Children, Computers, and Powerful Ideas**. New York, NY, USA: Basic Books, Inc., 1980. ISBN 0-465-04627-4.

PASTERNAK, E. **Visual Programming Pedagogies and Integrating Current Visual Programming Language Features**. Dissertação (Mestrado) — Robotics Institute, Carnegie Mellon University, August 2009.

PERKINS, D. N. et al. Conditions of learning in novice programmers. **Journal of Educational Computing Research**, SAGE Publications, v. 2, n. 1, p. 37–55, 1986.

ROBINS, A.; ROUNTREE, J.; ROUNTREE, N. Learning and Teaching Programming: A Review and Discussion. v. 13, n. 2, p. 137–172, jun. 2003. Disponível em: <http://www.tandfonline.com/doi/abs/10.1076/csed.13.2.137.14200>.

SALEIRO, M. et al. A low-cost classroom-oriented educational robotics system. In: **Social Robotics**. [S.l.]: Springer, 2013. p. 74–83.

SAYGIN, C. et al. Design, development, and implementation of educational robotics activities for k-12 students. In: **Proceedings of 2012 American Society for Engineering Education Annual Conference & Exposition**. [S.l.: s.n.], 2012.

SOLOWAY, E.; SPOHRER, J. **Studying the Novice Programmer**. L. Erlbaum Associates, 1989. (Interacting With Computers : Iwc). ISBN 9780805800029. Disponível em: <https://books.google.com.br/books?id=xskgAQAAIAAJ>.

VANDEVELDE, C. et al. Overview of technologies for building robots in the classroom. In: **International Conference on Robotics in Education**. [S.l.: s.n.], 2013. p. 122–130.

WANDSCHNEIDER, M. **Learning Node.js: A Hands-On Guide to Building Web Applications in JavaScript**. Pearson Education, 2013. (Learning). ISBN 9780133377989. Disponível em: <https://books.google.com.br/books?id=AmMibho26OEC>.