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

TRABALHO DE CONCLUSÃO DE CURSO

GUARAPUAVA

2015

EDUARTH HEINEN

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

Trabalho de Conclusão de Curso , apresentado na disciplina de Trabalho de Conclusão de Curso 2 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

Co-orientador: Prof. Dr. Diego Marczal

GUARAPUAVA

2015

Ministério da Educação
Universidade Tecnológica Federal do Paraná
Câmpus Guarapuava
Curso Superior de Tecnologia em Sistemas Para a Internet

ATA DE DEFESA DE MONOGRAFIA DE TRABALHO DE CONCLUSÃO DE CURSO DO CURSO DE TSI

No dia 3 de dezembro de 2015, às 17:00 horas, nas dependências da Universidade Tecnológica Federal do Paraná Câmpus Guarapuava, ocorreu a banca de **defesa da monografia** de Trabalho de Conclusão de Curso intitulada: "**RASPIBLOCOS: Ambiente de Programação Didático Baseado em RaspberryPi e Blockly**" do acadêmico **Eduarth Dapper Heinen** sob orientação do professor **Prof. Dr. Eleandro Maschio krynski** do Curso de Tecnologia em Sistemas para Internet.

Banca Avaliadora	
Membro	**Nome**
Orientador	Prof. Dr. Eleandro Maschio krynski
Coorientador	Prof. Dr. Diego Marczal
Avaliador 1	Prof. Me. Andres Jessé Porfirio
Avaliador 2	Prof. Esp. Maurício Barfknecht

Situação do Trabalho	
Situação	(x) Aprovado ( ) Aprovado com ressalvas ( ) Reprovado ( ) Não Compareceu
Encaminhamento do trabalho para biblioteca	(x) Pode ser encaminhado para biblioteca. ( ) Manter sigilo para publicação ou geração de patente.

Guarapuava, 3 de dezembro de 2015.

A Folha de Aprovação assinada encontra-se na Coordenação do Curso (ou Programa).

RESUMO

HEINEN, Eduarth. RASPIBLOCOS: Ambiente de Programação Didático Baseado em Raspberry Pi e Blockly. 50 f. 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 de código aberto, 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. 50 f. 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 como Saleiro et al. (2013) e Iturrate et al. (2013) 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 sob encomenda para essa

finalidade) 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 oferece 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 de código aberto, em que o minicomputador Raspberry Pi atua simultaneamente como hospedeiro da aplicação web, acessível por wi-fi, e controlador dos componentes. O modelo robótico pode ser construído por uma fração do preço de um conjunto Lego Mindstorms, por utilizar componentes de baixo custo e de fácil acesso. 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 ao minicomputador Raspberry Pi;
- Programar blocos de interface para controlar LEDs, motores de corrente contínua, sensores de proximidade e botões;
- Propor a realização de oficinas de robótica com alunos que possuam programação de computadores em seus currículos;

- Divulgar a iniciativa de código aberto, como também disseminar os resultados em publicações e eventos da área.

1.2 ESTRUTURA DA PROPOSTA

Esta seção apresenta a divisão do conteúdo apresentado pela monografia. O Capítulo 2 corresponde à 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. O Capítulo 3 detalha as tecnologias envolvidas e a arquitetura da aplicação. O Capítulo 4 sumariza uma série de passos, ilustrando a dinâmica de utilização do Raspiblocos. O Capítulo 5 detalha as oficinas de programação realizadas com o ambiente. Os Capítulos 6 e 7, respectivamente, trazem os resultados e as considerações finais da pesquisa.

2 RESENHA LITERÁRIA

Apresenta-se, neste capítulo, 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 à 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 na grade (que atua como um 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 do exercício, considerando a grade, 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, os mesmos alunos 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. Os robôs, mesmo sendo feitos com materiais acessíveis, são construídos sobre uma placa projetada especificamente, necessária por causa do tamanho reduzido, facilitando a interconexão entre os componentes. Além disso, as instruções disponíveis na interface são limitadas pela arquitetura simples dos robôs. Estes, pelo *hardware* que possuem, são capazes somente de se moverem 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 aprendiz refere-se a 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 do jogo. Ademais, diversos conteúdos diferentes podem ser apresentados pela simples alteração do cenário ou do tema das perguntas.

Construídos utilizando o Arduino Uno e um *shield* projetado para incluir controlador para motores, módulo *bluetooth* e reguladores de tensão, os robôs do experimento foram programados com as instruções básicas que são capazes de executar: mover-se seguindo uma linha até encontrar uma intersecção, girar a direita ou esquerda e ler etiquetas de rádio frequência. Como ocorre no projeto de Saleiro, a arquitetura dos robôs requer uma placa fabricada especialmente para a conexão dos componentes. Até o presente momento, não haviam sido realizados testes da aplicação e os elementos do enredo ainda não haviam sido completamente integrados

ao ambiente.

O projeto DuinoBlocks, desenvolvido por Alves e Sampaio (2014), é outro exemplo de ambiente que aplica uma linguagem de programação visual para controlar o comportamento de um modelo robótico. Possibilita utilizar blocos semelhantes aos da biblioteca Blockly para programar o controlador Arduino, além de oferecer blocos para estruturas de programação típicas.

Oficinas de programação e robótica, utilizando paralelamente o DuinoBlocks e o ambiente de programação nativo do Arduino, foram realizadas para avaliar a aceitação da proposta. Nestas oficinas, os participantes foram solicitados a "pensarem em voz alta" sobre o funcionamento dos programas, e demonstraram entendimento mais profundo dos programas construídos na interface do DuinoBlocks. Em outra oficina, realizada com um grupo de professores que incluía pessoas com experiência em programação, foram construídos algoritmos mais complexos, atestando a eficiência do ambiente mesmo para usuários familiarizados com a sintaxe de 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 uso, 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 de código aberto, 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 à pesquisa sobre robótica educacional e dificuldades no ensino-aprendizagem de programação de computadores.

2.2.1 Micromundos e robótica educacional

Em contraste com o aprendizado espontâneo que ocorre na infância, durante o qual, sem instrução específica, a criança aprende a falar, a se localizar no espaço e a argumentar; o ambiente de sala de aula, mesmo empregando muito mais tempo e esforço de alunos e professores, não alcança a mesma eficiência. De acordo com Papert (1980), isso ocorre devido ao estímulo do ambiente para a aquisição dos conhecimentos, como durante a infância, 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 modo diferente da aprendizagem natural, alguns conteúdos são apresentadas de maneira dissociada da realidade e da experiência dos alunos. O autor pondera sobre os bloqueios culturais surgidos da frustração que resulta de 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 através dos quais os alunos podem manipular o ambiente, experimentando e construindo assim o próprio conhecimento (MASCHIO, 2007). Um exemplo de micromundo seria um ambiente que possibilitasse manipular a massa de planetas em um sistema planetário, permitindo observar o resultado na gravidade que cada um deles exerce e na órbita que percorre. Papert (1980) 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 ambiente projetado por Papert utiliza robôs que se assemelham a tartarugas e que podem ser programados para se moverem enquanto desenham seu percurso. O autor apresenta uma sucessão de exemplos de aplicações desse ambiente, que variam do ensino de programação à demonstração das leis de Newton em física.

Assim como os micromundos descritos por Papert, o campo de pesquisa da robótica educacional almeja desenvolver 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 robó-

tica na educação, sendo: ***(i)*** habilidades de raciocínio (observação, estimação e manipulação); ***(ii)*** habilidades de processo científico e abordagens de resolução de problemas (como geração de hipóteses, teste de hipóteses e controle de variáveis); e ***(iii)*** interação social, bem como o 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 de estudo e exercício dessa sintaxe. 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. Essa perspectiva inspirou 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. (2003) 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 preciso 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). Para Brooks (1983), a construção de um programa de computador consiste em manter uma série de modelos sobrepostos e conectados de domínios de conhecimento, reconstruindo 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 mentais para o funcionamento da empresa, para o progresso de um pacote até a entrega, passando da 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 idealizados, 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 problemas.

- **A fragilidade do conhecimento e a ineficiência das estratégias utilizadas:** Entre as dificuldades observadas entre programadores inexperientes, relacionadas à sintaxe e à utilização das estruturas de programação, estão: problemas com declaração e atribuição de valores à variáveis, dúvidas sobre a atualização dos contadores durante os ciclos de estruturas de repetição, implementações incorretas de recursão, além de 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 internalizado somente depois de certa experiência. Ademais, Robins et al. (2003) observam que programadores inexperientes frequentemente possuem o conhecimento da sintaxe, porquanto não dominam as estratégias de construção do programa, sendo incapazes de combinar as diversas instruções da linguagem em programas coerentes.

- **O comportamento do aprendiz:** Perkins et al. (1986) distinguem comportamentos característicos entre programadores inexperientes em dois perfis recorrentes: *"stoppers"* e *"movers"*. Estudantes com perfis *"movers"*, ao se depararem com problemas ou dificuldades na representação dos conceitos, modificam e experimentam com o próprio código e com os modelos mentais estabelecidos até encontrarem uma solução. Estudantes com perfis *"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, que estes estudantes apresentam ao encontrar obstáculos.

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;
- Retirada gradual dos apoios ao aprendiz.

Robins et al. (2003) argumentam 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. Entretanto argumentam que, mais cruciais que esse conhecimento, são as estratégias para aplicá-lo. Concluem que se deve delegar mais atenção 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 elementos gráficos como blocos encaixáveis ou fluxogramas de componentes conectáveis para representar instruções de linguagens de programação. Estas metáforas visuais buscam aproximar conceitos de linguagens de programação a 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, bem como 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. Pretende-se, também, impedir a construção de código incoerente pela limitação das conexões possíveis (PASTERNAK, 2009). Com isso, um bloco que representa uma variável, por exemplo, só pode ser conectado a outro que retorne um valor adequado. Dessa forma, eliminam-se quase a totalidade dos erros sintáticos, uma vez que em vez 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*), de acordo com o mesmo autor, correspondem a mais da metade dos erros de compilação cometidos por programadores inexperientes.

3 ARQUITETURA DA SOLUÇÃO PROPOSTA

Este capítulo descreve as tecnologias envolvidas e apresenta a arquitetura proposta para a construção do ambiente.

3.1 Tecnologias envolvidas

Apresentam-se as tecnologias empregadas no funcionamento do ambiente, sendo elas: a biblioteca Blockly (3.1.1), o minicomputador Raspberry Py (3.1.2), a plataforma Node.js (3.1.3), o Framework Express e a biblioteca OnOff (3.1.4).

3.1.1 Blockly

Blockly é um conjunto de bibliotecas destinado à construção de editores para programação visual. Sua aparência e funcionamento foram inspirados no ambiente Scratch, desenvolvido com o propósito de ensinar programação para crianças, por meio da criação de animações e jogos interativos (VANDEVELDE et al., 2013). Assim como Scratch, a biblioteca Blockly 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 em código nas linguagens JavaScript, Python ou Dart.

Como um ambiente de programação visual, Blockly aplica metáforas gráficas para aproximar conceitos de linguagens de programação a tarefas familiares aos usuários. Os programas são construídos encaixando blocos que representam trechos de código sintaticamente corretos. Dessa forma, o ambiente apoia o programador inexperiente em uma das suas principais dificuldades, os erros de sintaxe. Ele assegura, também, a construção de programas coerentes, limitando os encaixes possíveis entre os blocos. Somente blocos graficamente compatíveis podem ser encaixados, como peças de quebra-cabeças, e estas limitações são estabelecidas du-

rante a definição dos blocos, permitindo-se determinar até o tipo de dado que cada instrução deve receber.

A biblioteca também permite que novos blocos sejam adicionados, incorporando capacidades específicas do ambiente, e conferindo novas instruções para a construção de programas. A definição destes blocos acontece pela adição de dois arquivos ao diretório de blocos. Estes descrevem, respectivamente, a aparência (conexões, compatibilidade e comportamento na interface) e o código que resulta da tradução do bloco durante a execução do programa. A *Block Factory* é uma página construída utilizando a biblioteca e disponibilizada junto desta, que oferece blocos para os aspectos e comportamento de um novo bloco. De maneira que, mesmo sem conhecer os métodos da API, o desenvolvedor pode encaixar blocos que representam as características que deseja (cor, formato, comportamento, retorno, opções, campos de texto, blocos compatíveis, entre outros) e receber da página um modelo para a definição do novo bloco, que pode então ser customizado conforme as necessidades do ambiente que está sendo construído. A biblioteca Blockly é aplicada largamente em projetos de programação didática, e entre as referências citadas na Seção (2.1), foi utilizada pelos projetos de Saleiro et al. (2013) e Iturrate et al. (2013).

3.1.2 Raspberry Pi

Raspberry Pi é um minicomputador que reúne *hardware* equivalente ao de um computador convencional em uma placa pouco maior que um cartão de crédito. Uma das versões mais recentes do dispositivo, a Raspberry Pi 2 model B, é construída sobre um processador ARM Cortex-A7 de 900MHz, e possui 1 GB de memória RAM. Tais características permitem que o minicomputador, assim como os modelos anteriores, seja capaz de suportar a execução de um sistema operacional. Esse aspecto é a principal diferença entre o minicomputador e controladores de placa única como o Arduino, que executam código compilado e portanto oferecem vantagens para aplicações em que o tempo de execução das instruções é um ponto crítico.

O conjunto de pinos GPIO (*General Purpose Input Output*), integrado ao Raspberry Pi, permite coordenar uma série de componentes eletrônicos. Controlados pelo sistema operacional, esses pinos funcionam como interruptores, podendo ser ligados ou desligados, transmitindo ou interrompendo a transmissão de corrente elétrica. São capazes, portanto, de controlar diretamente sensores, botões, LEDs e outros componentes eletrônicos. Além desse modo de operação, funcionam também como entrada e saída de dados digitais. Apesar de não conduzirem corrente elétrica suficiente para alimentar motores DC, podem ser usados para acionar placas controladoras aptas a comandar estes motores. O minicomputador oferece ainda entra-

das USB, permitindo a construção de modelos robóticos com reconhecimento de imagem e conexão à rede local ou internet, utilizando câmeras e adaptadores wi-fi compatíveis com USB (VANDEVELDE et al., 2013).

Para que seja possível utilizar o minicomputador como *host* da aplicação e controlador do modelo robótico, é necessário um sistema operacional compatível. Raspbian[1], é uma versão do sistema operacional Debian customizada para o *hardware* do Raspberry Pi. Apoiado oficialmente pela Raspberry Foundation, permite configurar o minicomputador como servidor de aplicações web, enquanto dirige o comportamento das portas GPIO por meio das ferramentas integradas ao sistema operacional. Além do Raspbian, a seção de *downloads* da página do Raspberry Pi disponibiliza imagens de disco dos sistemas operacionais: Ubuntu Mate, Snappy Ubuntu Core, Windows 10 IOT Core, OSMC (*Open Source Media Center*) e outros.

Entre as experiências construídas com o minicomputador, encontram-se telescópios autônomos, *smartphones* e, até mesmo, sistemas que utilizam inteligência artificial no controle de motores de combustão. Por seu relativo baixo custo[2] e popularidade entre acadêmicos e entusiastas da eletrônica e robótica, alcançou, em fevereiro de 2015, a marca de 5 milhões de unidades vendidas[3].

3.1.3 Node.js

Node.js é uma plataforma de aplicações JavaScript, assíncrona e dirigida por eventos. Assim como um servidor web, funciona respondendo requisições *http*. Seu funcionamento baseia-se no *event loop*, ou ciclo de eventos, integrado à plataforma. Este mecanismo recebe e processa requisições, delegando operações de entrada e saída de dados ao sistema operacional, deixando como responsabilidade do programador declarar funções de *callback* a serem executadas na conclusão de cada operação realizada pelo SO. Com isso, o código da aplicação é executado de modo *single threaded*, e as operações de entrada e saída de dados são executadas de maneira assíncrona pelo sistema operacional (CANTELON et al., 2013).

O processo de refinamento da linguagem JavaScript por navegadores modernos, resultou no surgimento do *Google Chrome's V8 JavaScript Engine*[4], interpretador capaz de compilar aplicações JavaScript em linguagem nativa (C/C++), de forma ágil e transparente (BROWN, 2014). A plataforma Node.js surgiu da integração deste interpretador à biblioteca de processa-

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

[2] Raspberry Pi 2 Model B: http://swag.raspberrypi.org/collections/frontpage/products/raspberry-pi-2-model-b (acessado em: 09-12-2015)

[3] Five million sold!: https://www.raspberrypi.org/blog/five-million-sold/

[4] V8 JavaScript Engine: https://code.google.com/p/v8/

mento de eventos *libev*, por Ryan Dahl em 2009 (WANDSCHNEIDER, 2013).

A capacidade de delegar operações de entrada e saída de dados ao sistema operacional de modo assíncrono permite que aplicações Node.js continuem recebendo e tratando requisições enquanto aguardam o *callback* que sinaliza a conclusão das operações delegadas anteriormente. Isto permite que o servidor mantenha conexões abertas e processe requisições consumindo pouca memória. Estas características tornam Node.js a plataforma ideal para a construção de aplicações *DIRT* – acrônimo para *data-intensive real-time* – capazes de responder quase instantaneamente a um grande número de requisições concorrentes (CANTELON et al., 2013).

3.1.4 Framework Express e biblioteca OnOff

Express é um *framework* para a construção de aplicações Node.js. É descrito como minimalista e flexível, e proporciona um conjunto vasto de funcionalidades para a construção de aplicações web (BROWN, 2014). Oferece gerenciamento de *layout*, ferramentas para o tratamento simplificado das requisições, roteamento de endereços, e diversas outras funcionalidades que tornam mais simples a construção de aplicações Node.js (CANTELON et al., 2013). Possui também uma ferramenta de *scaffolding*[5] capaz de gerar automaticamente a estrutura de pastas e arquivos necessária a uma aplicação Node.js.

Para que a aplicação Node.js seja capaz de acionar as portas do minicomputador, foi empregada a biblioteca OnOff. Capaz de controlar ativamente o estado das portas GPIO, também possibilita programar *listeners* que executam funções determinadas ao detectar alterações no estado da porta que monitoram. Estas alterações podem ser causadas pelo pressionar de um botão, pela resposta de um sensor de proximidade ou por qualquer evento em que ocorra retorno de corrente elétrica ao Raspberry Pi através de uma das portas GPIO. A biblioteca mantém seu código aberto e está disponível em um repositório junto da documentação de utilização.

3.2 Arquitetura do ambiente

A arquitetura do ambiente pode ser dividida em quatro camadas: interface, aplicação, controle e componentes. A Figura 1 apresenta essa divisão, destacando os principais elementos de cada camada e a interação entre eles. Na sequência, descreve-se cada camada, bem como o seu papel na arquitetura e as tecnologias que a compõem:

[5]Express application generator: http://expressjs.com/en/starter/generator.html

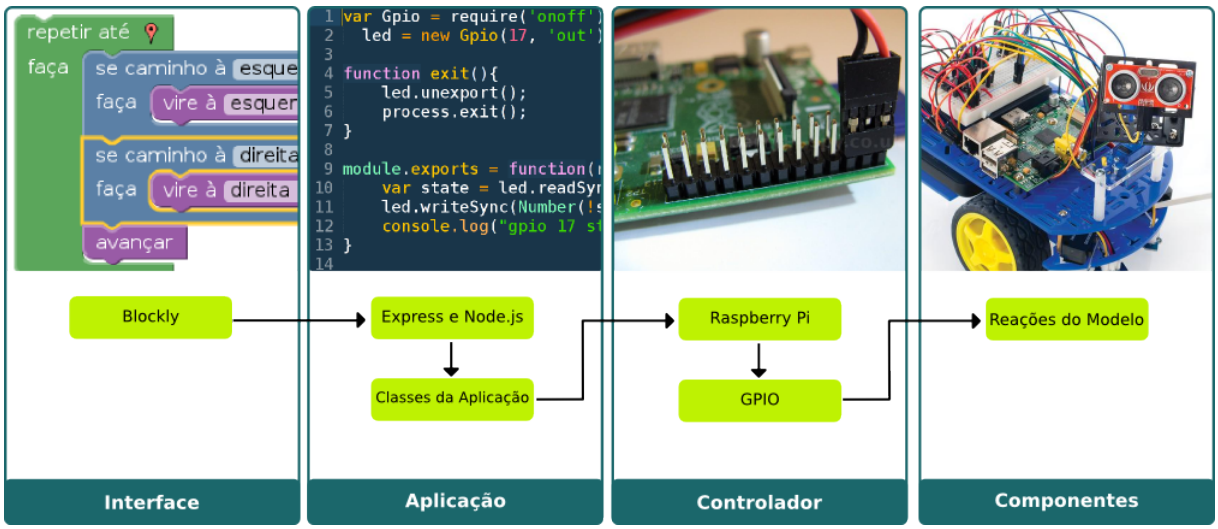


Figura 1: Ilustração das camadas da arquitetura da aplicação.

3.2.1 Camada de interface

A interface da aplicação, apresentada pela Figura 2, consiste de uma página simples que apresenta a área de trabalho onde os programas podem ser construídos, a biblioteca de blocos e o botão "executar programa". Considerando a interação de usuários com diferentes níveis de experiência, a biblioteca Blockly, é empregada de modo que o conjunto de instruções do ambiente seja representado por blocos encaixáveis coloridos. O usuário pode arrastar os blocos da biblioteca para a área de trabalho, alterando as opções necessárias e encaixando novos blocos àqueles previamente posicionados. Clicando-se em "executar programa", os blocos na área de trabalho são traduzidos em código, então enviado ao minicomputador. Este interpreta as instruções recebidas, resultando em reações no modelo robótico.

Além dos blocos oferecidos pela biblioteca, foram adicionados blocos significando instruções próprias do modelo robótico, apresentados na Figura 3, sendo:

- **`GPIO x:`** capaz de alterar diretamente o estado da porta GPIO selecionada no menu *drop-down* representado por *x*;
- **`GPIO x retornar estado:`** pode ser encaixado no bloco que representa uma variável, armazenando o estado da porta selecionada;
- **`Motor DC x y:`** *x* e *y* sendo menus *drop-down* onde o usuário deve selecionar as portas GPIO em que o motor está conectado. Aceita o encaixe dos blocos representando as instruções `mover adiante x ms` e `mover atrás x ms`, sendo *x* o campo para a duração da instrução;

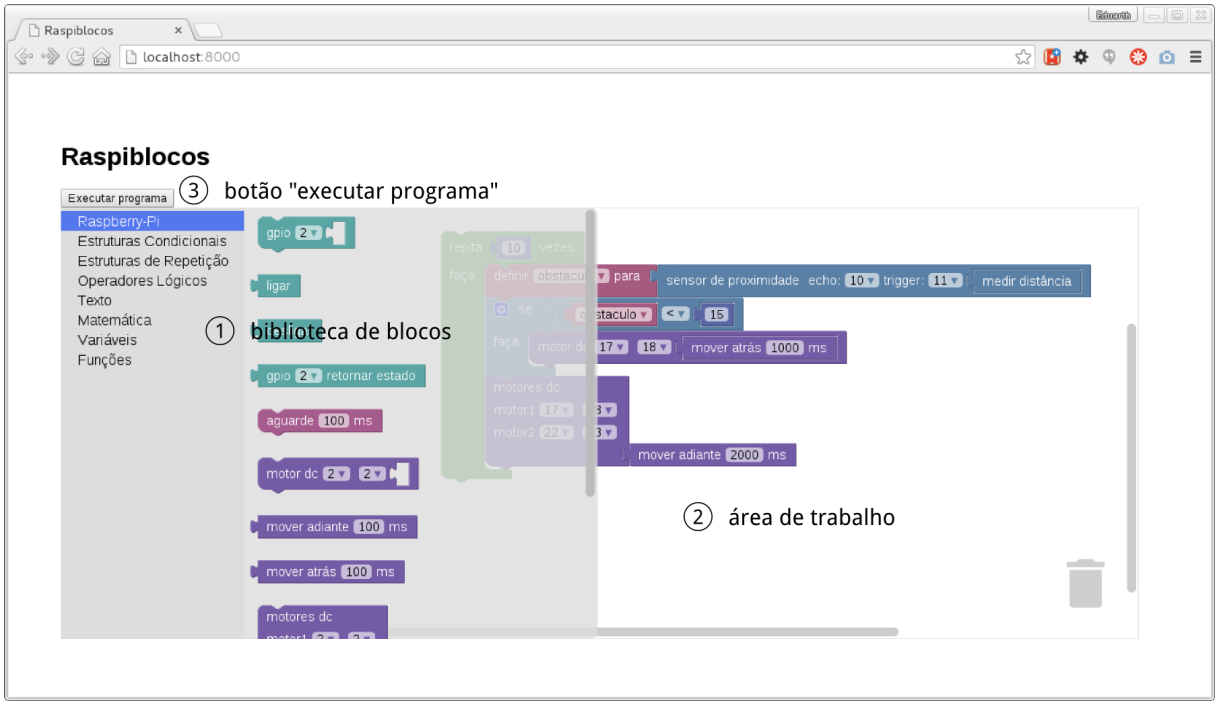


Figura 2: Imagem da interface da aplicação.

- **`Sensor de proximidade echo x trigger y:`** que permite encaixar a instrução `medir distância` e armazenar o resultado em uma variável, assim como o bloco `GPIO x retornar estado`, de maneira que a distância até o obstáculo adiante mais próximo medida pelo sensor seja armazenada na variável;

- **`Botão x:`** aceita o encaixe de uma função, declarada utilizando os blocos com esta finalidade, oferecidos pela biblioteca Blockly. Declara um *listener* que executa a função ao detectar que o botão foi pressionado;

- **`Aguarde x ms:`** que instrui o modelo robótico a aguardar determinado período de tempo antes de interpretar a próxima instrução;

- **`Imprima x:`** que aceita o encaixe de um bloco de variável, imprimindo a distância medida pelo sensor, o estado de uma porta, mensagens de texto armazenadas em variáveis, e outras informações.

3.2.2 Camada de aplicação

Como mencionado na seção anterior, a página inicial da aplicação oferece uma interface de programação visual onde blocos, que representam as instruções que o modelo é capaz

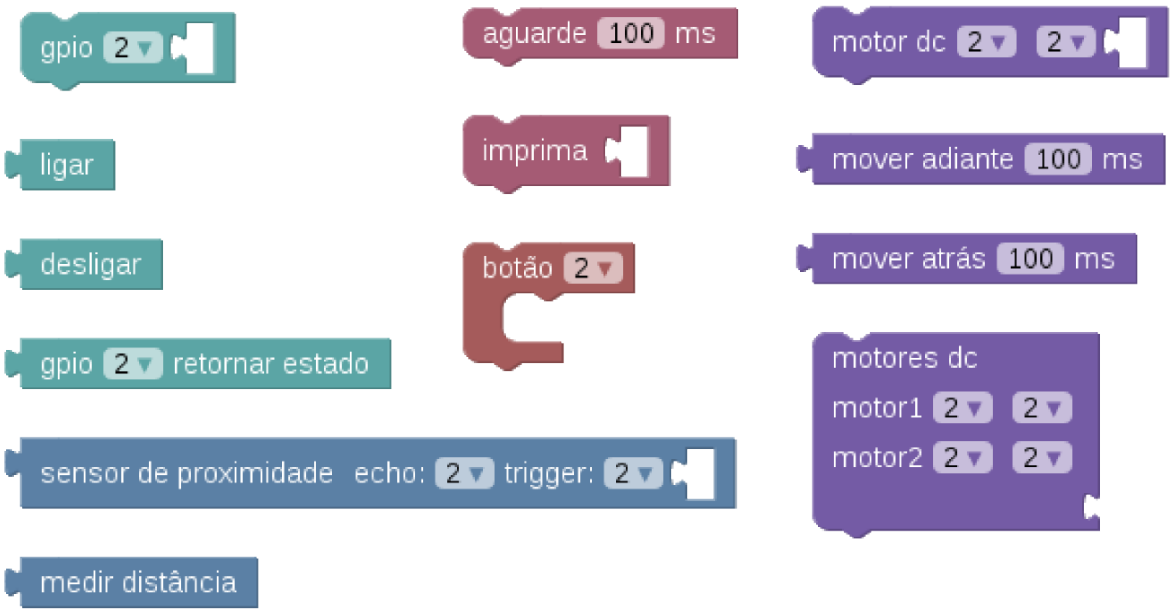


Figura 3: Blocos representando as instruções do modelo.

de realizar e estruturas típicas de linguagens de programação, podem ser empilhados em pequenos programas. Ao executar o programa construído, os blocos são traduzidos em instruções, que são ordenadas e agrupadas em um trecho de código, enviado através da rede wi-fi para a aplicação hospedada no Raspberry Pi. O trecho de código é declarado como uma nova função e executado, acionando os métodos das classes que representam e controlam cada componente.

O principal obstáculo encontrado no desenvolvimento da arquitetura foi sobrepôr o funcionamento nativo da plataforma Node.js. Construída sobre um ciclo de eventos assíncrono, a plataforma é capaz de delegar operações ao sistema operacional e continuar em frente, interpretando as próximas instruções. Por este aspecto, a execução do trecho de código recebido pelo minicomputador não respeitava a sequência das instruções encaixadas na interface. Para solucionar este impasse, foram empregados *generator functions* e *promises*, funcionalidades disponíveis na especificação ECMAScript 6 da linguagem JavaScript e suportadas pelas versões posteriores à 0.11 da plataforma Node.js[6].

Generator functions empregam a palavra-chave `yield` para controlar o fluxo de execução. Quando a função é declarada, obtêm-se um objeto *generator*, que representa o estado atual de execução da função. Este objeto oferece o método `next()` que resume a execução das instruções até encontrar a próxima palavra-chave `yield`. O valor retornado pela instrução que segue a palavra-chave `yield` torna-se o valor de retorno do método `next()`. Isso ocorre de modo que, cada vez que o método `next()` é chamado, retorna-se o resultado da próxima

[6]ES6 (a.k.a. Harmony) Features Implemented in V8 and Available in Node – https://github.com/nodejs/node-v0.x-archive/wiki/ES6-%28a.k.a.-Harmony%29-Fe atures-Implemented-in-V8-and-Available-in-Node

instrução marcada pela palavra-chave `yield`, como se estas palavras-chave indicassem pontos de parada na execução do corpo da função. Além disso, diferente de funções JavaScript típicas, *generator functions* possibilitam que estruturas `try-catch` e a instrução `return` sejam utilizadas como na maioria das linguagens de programação (FLANAGAN, 2011).

Promises foram desenvolvidas para substituir os *callbacks* da linguagem JavaScript. Pela abordagem tradicional, uma função de *callback* é declarada e passada adiante até alcançar a função que retorna a informação que deseja. Usa-se esta como parâmetro da função de *callback*, encaminhando os dados para serem tratados. *Promises*, por sua vez, possuem os métodos `resolve()`, usado para retornar a informação procurada, e `reject()`, que retorna uma mensagem de erro, descrevendo o problema ou comunicando que os dados não puderam ser recuperados. Possui também um atributo que indica o estado da função, podendo ser *pending*, *fullfiled* ou *rejected*. Ao ser declarada, uma *promise* assume um estado de pausa, representado por *pending*, os outros dois estados resultam da conclusão da função e indicam, respectivamente, que atingiu o objetivo ou que encontrou problemas. A principal vantagem das *promises* é a possibilidade de encadear um número indefinido de funções sem depender de *callbacks*. Para isso, *promises* oferecem o método `then()`, que encaminha para a próxima função as informações retornadas pelos métodos `resolve()` ou `reject()`. Desse modo, encadear funções se resume a evocar o método `then()` passando como parâmetro a próxima função que deve ser executada. Além do método `then()`, *promises* oferecem também o método `catch()`, que captura qualquer erro ocorrido na sequência de funções (PARKER, 2015).

Em conjunto, *generator functions* e *promises* são ferramentas poderosas. Combinando a palavra-chave `yield` ao objeto *promise*, o método `next()` da *generator function* é executado automaticamente. Assim, o fluxo de execução entra em pausa pela palavra-chave `yield`, aguarda a conclusão de cada *promise* e então resume automaticamente a execução das instruções. A palavra-chave `yield` antecede as instruções que resultam da tradução dos blocos que controlam portas GPIO, motores, instruções “aguarde” e “retorne estado”, e outros, para assegurar que o modelo aguarde a conclusão da instrução antes de executar a instrução seguinte. A Figura 4 busca apresentar o funcionamento de *promises* e *generator functions* comentando o fluxo de execução de um trecho de código.

Utilizando *generator functions* e *promises*, foram criadas classes que representam portas GPIO, motores de corrente direta, sensores de proximidade e botões. Os métodos disponibilizados por estas classes são acionados pelas instruções interpretadas. Resultando no funcionamento dos componentes eletrônicos conectados às portas do Raspberry Pi. Na definição das

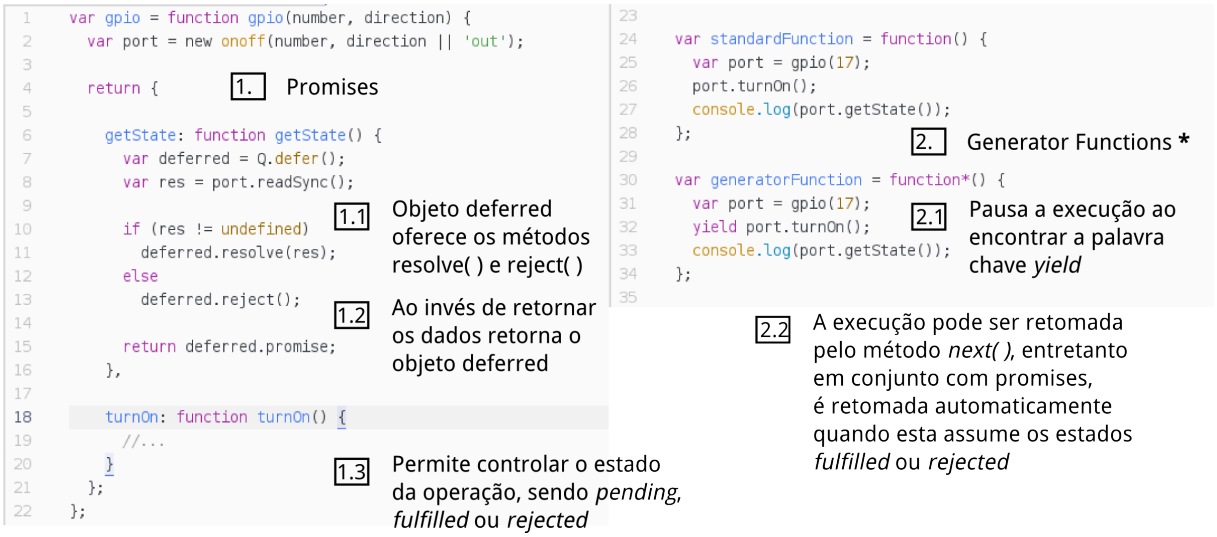


Figura 4: Trecho de código apresentando *promises* e *generator functions*

classes, a biblioteca OnOff[7] é utilizada para controlar o estado das portas físicas do minicomputador. Ela é capaz de ler e escrever valores para o estado das portas, como também de declarar *listeners* que monitoram as portas por alterações.

3.2.3 Camada de controle

A camada de controle transforma as instruções executadas na aplicação Node.js, acionando os métodos das ferramentas integradas ao sistema operacional, que controlam o funcionamento das portas GPIO transmitindo corrente elétrica até os componentes. Esta camada é composta pelo minicomputador Raspberry Pi e portas GPIO integradas, e representa a fronteira entre as instruções da aplicação e as reações do modelo robótico.

A utilização do minicomputador Raspberry Pi é justificada pela possibilidade de hospedar a aplicação web e simultaneamente controlar os componentes do modelo robótico. O adaptador wi-fi USB conectado ao minicomputador oferece uma rede local, através de qual, dispositivos com conectividade a rede e navegador web são capazes de utilizar a aplicação.

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 destas portas, o minicomputador envia correntes elétricas aos componentes nelas conectados. A corrente enviada pelas portas, todavia, não é suficiente para alimentar alguns componentes. Por este 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

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

controlador apenas aciona os componentes, que são alimentados por outra fonte de energia.

3.2.4 Camada de componentes

Os componentes eletrônicos conectados ao Raspberry Pi, tais como motores de corrente direta, sensores de proximidade, botões e LEDs, constituem esta camada da arquitetura. 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 DINÂMICA DE UTILIZAÇÃO DA FERRAMENTA

Esta seção descreve uma série de passos, que exemplificam uma dinâmica de utilização do ambiente. Pressupõe a disponibilidade de um modelo robótico, montado de acordo com as instruções publicadas no repositório da aplicação[1], presentes no Apêndice A, e destaca os pontos de vista do aluno participante e do professor que orienta a oficina como principais atores da dinâmica.

Destaca-se, primeiramente, que o ambiente descrito por este projeto é proposto como ferramenta para realização de oficinas de robótica paralelas à lições teóricas de programação de computadores. A interface de programação visual do ambiente estabelece uma camada de abstração que previne quase completamente a existência de erros de sintaxe e de construção nos programas. Sendo assim, estimula a construção do pensamento computacional, algorítmico e de habilidades em análise e solução de problemas, mas não permite que o aluno lide com os próprios erros de sintaxe, de forma que não seja suficiente para desenvolver proficiência em qualquer linguagem de programação específica.

O material de apoio às oficinas, apresenta conceitos de linguagens de programação, para então desafiar os alunos a utilizarem esse conhecimento na programação do modelo robótico. Um trecho do material está presente no Apêndice B, e os conceitos abordados incluem:

- A interface da aplicação e os blocos disponíveis;
- Os princípios de programação envolvidos em cada desafio;
- Analogias que buscam auxiliar a compreensão dos conceitos, como o uso de trilhos de trem para representar o fluxo de execução com estruturas condicionais ou a analogia da memória do computador como um arquivo;
- Diagramas que detalham a conexão dos componentes, destacando precauções necessárias como assegurar que a voltagem que retorna do sensor de proximidade não danifique a porta GPIO.

[1] Raspiblocos em Github: https://github.com/eduartheinen/raspiblocos

Ainda que o material seja substancialmente detalhado, a dinâmica exposta nesta seção considera uma lição de programação orientada por professor que possua domínio do conteúdo e do funcionamento do modelo robótico. Considera-se que, mesmo com o apoio do material, o aluno possa encontrar obstáculos na utilização do ambiente.

Assume-se que existam dois modelos robóticos, uma turma com dez alunos divididos em dois grupos e um professor como orientador do laboratório. São enumerados, a seguir, uma sequência de passos que descrevem a utilização da ferramenta:

1. O professor apresenta o ambiente aos alunos, destacando as tecnologias envolvidas, descrevendo brevemente o Raspberry Pi e a interface da aplicação;

2. A seguir, apresenta o material da oficina, abrangendo noções de programação e explicando cada conceito por meio dos mecanismos disponíveis no modelo robótico. Por exemplo, variáveis podem ser apresentadas em paralelo ao funcionamento das portas GPIO, permitindo que os alunos armazenem e imprimam seu estado; estruturas condicionais podem ser ilustradas pelo funcionamento do sensor de proximidade, imprimindo mensagens diferentes de acordo com a distância medida. Dessa maneira todo conceito teórico pode ser observado na prática por meio das reações do modelo robótico;

3. Após apresentar os conceitos de programação e o funcionamento dos componentes, o professor propõe que os alunos programem o modelo robótico para cumprir os objetivos determinados pelo desafio da lição;

4. Os alunos acessam a aplicação hospedada no modelo robótico utilizando dispositivos que possuam navegador e rede wi-fi. Cada grupo pode usar um único computador, incentivando, assim, o trabalho em equipe e cooperação na busca por soluções;

5. Na interface da aplicação, os alunos constroem o programa que julgam ser capaz de atender aos requisitos do desafio, executam este programa, observando o comportamento do modelo, e discutem se as reações do modelo cumprem os requisitos ou se é necessário fazer alterações:

   (a) Caso o programa alcance os objetivos, os alunos devem ser encorajados a realizar experiências com o código, buscando novos objetivos e observando como o comportamento do modelo robótico responde às alterações que realizam;

   (b) Caso o programa não alcance os objetivos, os alunos devem ser incentivados a discutir e alterar o programa em busca da solução. O professor pode orientar os alunos, mas eles devem ser capazes de encontrar a solução por conta própria.

A seguir são expostos dois exemplos de conceitos de programação que podem ser apresentados através dos componentes integrados ao modelo, elencando possíveis dificuldades, meios de resolvê-las, e a descrição de um programa que possa alcançar os objetivos do desafio:

1. Supondo que o objetivo da lição seja demonstrar o funcionamento da estrutura condicional `if-else` utilizando o sensor ultrassônico de proximidade, o professor pode desafiar os alunos a construir um programa meça a distância até um obstáculo e acenda um LED caso ela seja menor que 20 cm:
   (a) Possíveis obstáculos seriam:
      i. Problemas para acessar a aplicação: acessar o modelo errado, não encontrar a rede wi-fi da aplicação, errar o endereço ou a porta da aplicação, problemas de configuração de rede do dispositivo usado para acessar a aplicação;
      ii. Problemas na construção do código ou na montagem do modelo: não possuir domínio do conceito de estrutura condicional, selecionar portas erradas nas opções dos blocos, não conectar o sensor de proximidade corretamente;
   (b) Soluções para estes obstáculos poderiam ser:
      i. Orientação do professor na conexão e configuração dos dispositivos;
      ii. Consulta ao material de apoio, que dispõe de diagramas de conexão dos componentes e explicações sobre o funcionamento das estruturas de programação;
   (c) Para alcançar o objetivo, o programa construído deve:
      i. Utilizar o bloco do sensor de proximidade e o bloco que representa uma variável, para armazenar a distância até o obstáculo;
      ii. Testar, com a estrutura condicional `if-else`, se o valor armazenado é menor que 20;
      iii. Caso verdadeiro, acender o LED conectado utilizando o bloco GPIO para alterar o estado da porta;
   (d) Ao executar o programa o comportamento esperado do modelo robótico seria:
      - Se houver um obstáculo a menos de 20 cm em frente do sensor de proximidade, o LED conectado à porta determinada no bloco GPIO do programa deve acender;
      - Caso contrário não deve haver reação;
2. Considerando o desafio de programar o modelo para mover-se por uma sala evitando obstáculos, empregando estruturas de repetição, estruturas condicionais, sensor de proxi-

midade e motores de corrente direta, o professor pode desafiar os participantes a construir um programa movimente o robô pela sala evitando obstáculos:

(a) Possíveis dificuldades seriam:

   i. Problemas para acessar a aplicação, como no exemplo anterior;

   ii. Problemas na construção do código ou na montagem do modelo: inversão das portas de forma que a instrução `mover adiante` faça com que o motor se mova para trás, portas erradas selecionadas nos blocos que representam os motores ou o sensor, e dificuldades para entender como mudar a direção utilizando somente as instruções `mover adiante` e `mover atrás`;

(b) Soluções para estes obstáculos poderiam ser:

   i. A consulta ao material de apoio e orientação do professor, assim como no exemplo anterior;

(c) Para alcançar o objetivo, o programa construído deve:

   i. Utilizar a estrutura *for* para controlar o número de repetições do roteiro: verificar se há obstáculo, decidir sobre seguir em frente ou mudar a direção, e movimentar-se;

   ii. Armazenar em uma variável a distância medida pelo sensor ultrassônico;

   iii. Testar o valor armazenado com a estrutura `if-else`, decidindo entre mover-se em frente ou desviar o obstáculo

      A. Se a distância for menor que 20, acionar somente um dos motores com o bloco `mover atrás`, girando o robô em torno de si mesmo;

      B. Caso contrário, acionar ambos os motores com o bloco `mover adiante`, movendo o robô em frente;

(d) Ao executar o programa o comportamento esperado do modelo robótico seria:

   i. O robô mede a distância até o próximo obstáculo;

   ii. Não havendo obstáculos em frente move-se adiante por alguns segundos e recomeça o ciclo;

   iii. Caso existam obstáculos em frente, o robô gira em torno de si mesmo e recomeça o ciclo.

5 EXPERIMENTO DE ACEITAÇÃO DIDÁTICA

Foram realizadas duas oficinas de programação com o intuito de averiguar a aceitação da ferramenta pelos usuários e levantar hipóteses sobre seu impacto didático. O material de apoio, os participantes, a metodologia das oficinas e os resultados são descritos nas seções seguintes.

5.1 Material de apoio

O material de apoio consiste de um tutorial de programação e de um conjunto de diagramas de montagem dos componentes. O tutorial tem o objetivo de guiar a realização das oficinas, explica conceitos de programação de computadores, relacionando cada um deles ao componente capaz de demonstrar seu funcionamento. Ao final de cada seção do tutorial os aprendizes são desafiados a criarem seus próprios programas, aplicando os conceitos de programação e os componentes ensinados durante a oficina. Os diagramas de construção do modelo robótico, como o ilustrado na Figura 5 são utilizados para expor a conexão dos componentes, e fazem parte do tutorial de programação. Foram produzidos utilizando o programa *Fritzing*, que simplifica a criação destas ilustrações oferecendo imagens conectáveis de diversos componentes eletrônicos.

O conteúdo do tutorial foi inspirado nos livros *Eloquent JavaScript* (HAVERBEKE, 2014), e *Hello World: Computer Programming for Kids and other Beginners* (SANDE; SANDE, 2013), e apresenta:

1. Uma breve descrição do ambiente: apresentando o Raspberry Pi e a interface da aplicação;

2. Introdução sobre o funcionamento dos computadores: que discute as diferenças entre comunicar nossos pensamentos para outro ser humano utilizando linguagem natural e descrever essas intenções, por meio de uma interface ou linguagem de programação, para um computador;

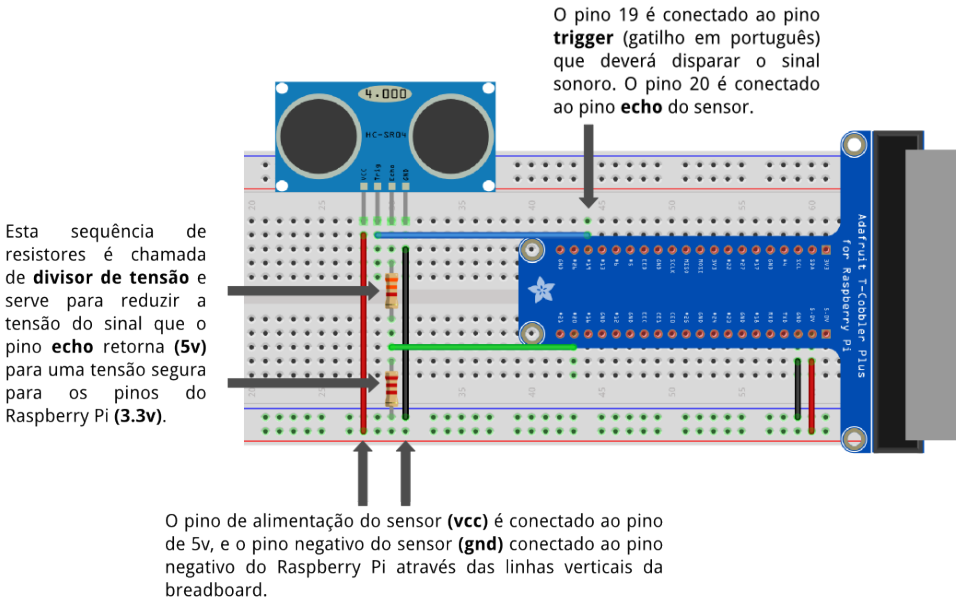


Figura 5: Diagrama de conexão do sensor ultrassônico de proximidade.

3. Introdução sobre linguagens de programação: expondo o modelo de entrada, processamento, e saída de dados;

4. Memória e variáveis: explicadas pela comparação da memória com um arquivo de diversas gavetas, no qual o computador pode armazenar e recuperar informações;

5. Controle das portas GPIO: utilizando o bloco com esta finalidade, seguido de um desafio em que os alunos devem acender e apagar LEDs conectados ao minicomputador;

6. Operadores lógicos e aritméticos: demonstrados utilizando os blocos oferecidos pela biblioteca Blockly em exemplos de operações matemáticas e de precedência entre operadores;

7. Estruturas condicionais e de repetição: demonstrando o funcionamento dos operadores *and* e *or*, e das estruturas condicionais `if-else`, `while-do`, `do-while` e `for` divididas nas seções:

   (a) Se houver obstáculo faça: que apresenta a estrutura `if-else`, detalhes de conexão do sensor ultrassônico de proximidade e o bloco relacionado;

   (b) Enquanto não houver obstáculo faça: que esclarece a estrutura `while-do` e o conceito de *contador* para controlar o número de repetições do comando;

   (c) Faça enquanto não houver obstáculo: explicando a estrutura *do-while* e as diferenças em comparação com a estrutura *while-do*;

(d) Repita *x* vezes: que descreve a estrutura `for`, apresenta diagramas de conexão para os motores de corrente direta, e o bloco na interface capaz de controlá-los;

8. Funções e gatilhos: que explica a declaração de funções utilizando o bloco que representa um botão para declarar *listeners*, que executam uma função determinada ao detectar que o botão foi pressionado.

A última seção do tutorial de programação inclui um breve guia de criação de blocos e programação de classes para o ambiente. E busca incentivar a realização de um projeto final das oficinas, em que os alunos desenvolvam as classes e blocos capazes de controlar um novo componente de sua escolha, e integrem-no ao ambiente.

5.2 Sujeitos

Participaram da pesquisa quatro alunos do Primeiro Período de 2015/2 do curso de Tecnologia em Sistemas para Internet do câmpus Guarapuava da UTFPR. Os alunos frequentaram duas oficinas de programação e robótica apresentadas utilizando o material de apoio e o ambiente descritos neste projeto. Todos os participantes possuíam conhecimento prévio em programação e, portanto, foram solicitados a avaliarem criticamente o conteúdo do tutorial e a realização das oficinas.

5.3 Procedimentos

As oficinas de programação foram realizadas nos dias 15 e 22 de outubro de 2015. A duração de cada oficina foi de 2 horas, durante as quais foram esclarecidos os conceitos envolvido no desafio proposto e realizadas demonstrações do funcionamento dos blocos e mecanismos. Durante a última meia hora de cada oficina foram propostos os objetivos do desafio de programação para que os participantes fossem orientados na construção dos programas.

5.4 Resultados observados

Ao se observar interação com o ambiente e do comportamento dos alunos, pode-se afirmar que os participantes demonstraram interesse pela iniciativa, pelo funcionamento do modelo robótico e pela utilização da interface de programação. Em todas as oficinas os alunos participaram ativamente, questionando e sugerindo alterações ao conteúdo do tutorial e ao funcionamento do ambiente. Entre as sugestões recebidas dos participantes estiveram: exibir na

interface de programação as mensagens que retornam da execução das instruções, utilizar analogias na apresentação dos conceitos e disponibilizar um bloco com uma instrução de pausa.

As oficinas tiveram caráter principalmente de teste de utilização, em detrimento da avaliação do aspecto didático. Todos os participantes possuíam conhecimento prévio, tornando difícil prever quanto conteúdo seria abordado em cada oficina. Por esse motivo, como também pelas restrições de tempo, não foi possível abordar todo o conteúdo do tutorial de programação. Presume-se que o conteúdo completo do material, abrangendo conceitos de programação, componentes e desafios, possa ser apresentado em doze horas-aula.

Como trabalho futuro deve ser avaliada a possibilidade da realização de testes mais abrangentes, com mais participantes e um grupo de controle. Dispondo de três modelos, seria possível convidar 15 alunos das turmas do primeiro semestre e, acompanhando seu desempenho nas disciplinas de programação de computadores em comparação ao desempenho nas oficinas, levantar hipóteses que relacionem a utilização do ambiente e o despenho acadêmico.

6 RESULTADOS

A pesquisa realizada desenvolveu um ambiente e material de apoio que facilitam a realização de oficinas de programação e que:

1. Estimula a exploração, permitindo que o aluno observe o resultado de cada instrução nas reações do modelo robótico;
2. Incentiva o aluno a enfrentar problemas complexos, encorajando-o a modificar seus programas e a observar como as reações do modelo se diferem;
3. Emprega analogias visuais para demonstrar a natureza sequencial da execução das instruções em um programa, além dos padrões de formato, cores e nomeação para indicar instruções que podem ser combinadas;
4. Apoia o aluno no aprendizado de programação pela redução dos erros sintáticos, oferecendo blocos de código predefinido e restrições que evitam a construção de código incoerente;
5. Guia a construção do modelo robótico por meio de diagramas de montagem detalhados, utilizando componentes de baixo custo e sendo, assim, facilmente replicável;
6. Oferece um tutorial de programação criado com o intuito de guiar a realização das oficinas, e que abrange: conceitos de computação e programação, especificações dos blocos disponíveis, diagramas de montagem dos componentes, e que, ao final de cada oficina, desafia o aluno a utilizar os conceitos que aprendeu para controlar o modelo robótico;
7. Estimula os usuários a expandir as instruções, blocos e capacidades do modelo, por manter o código aberto e incentivar a realização de projetos de ampliação do ambiente ao final das oficinas de programação.

6.1 Principais resultados

Pode-se destacar como principais resultados alcançados:

- A concepção e desenvolvimento do ambiente Raspiblocos, abrangendo a definição dos blocos e construção da interface e modelo robótico;
- A realização das oficinas, principalmente por aprimorar o funcionamento do ambiente e da organização das lições;
- E a divulgação da iniciativa por meio da publicação de artigo no CBIE LACLO 2015 (HEINEN et al., 2015).

6.2 Outros resultados alcançados

Além dos resultados diretamente relacionados aos objetivos definidos pelo projeto, pode-se apontar como resultados secundários do projeto:

- A participação no plano de trabalho Oficinas de Robótica e Programação Utilizando o Ambiente RASPIBLOCOS no projeto de extensão Incentivo às Áreas de Ciência, Tecnologia, Engenharia e Matemática por meio da Programação e da Robótica, com duração prevista de setembro de 2015 a julho de 2016. Este projeto busca refinar o ambiente e o material de apoio, e também promover oficinas de programação e robótica para os alunos dos períodos iniciais dos cursos oferecidos no câmpus Guarapuava da UTFPR, posteriormente estendendo-as a alunos dos últimos anos das escolas do município.
- A obtenção da bolsa de apoio à realização do TCC, que deve possibilitar a aquisição de novos modelos robóticos, utilizados na realização de oficinas mais abrangentes.

7 CONSIDERAÇÕES FINAIS

A proposta desta pesquisa, de construir um ambiente didático para a realização de oficinas de programação e robótica, adveio da constatação do potencial destes ambientes da aplicação da tecnologia na educação e das dificuldades inerentes ao aprendizado da programação de computadores. Ao analisar a bibliografia tocante ao tema, observou-se que não somente no ensino de programação de computadores, mas na educação em geral, a aplicação de metodologias de ensino obsoletas faz com que perdurem bloqueios culturais e comportamentos que dificultam a construção do conhecimento.

A pesquisa de Papert (1980) destaca que, durante a infância, a criança adquire uma série de perícias sem qualquer instrução específica, simplesmente interagindo com o ambiente semanticamente rico que a rodeia. Seu cotidiano é repleto de números, cores, sons, e esta abundância de estímulos leva a criança a construir conhecimento enquanto observa e experimenta o ambiente. O autor se refere a este fenômeno como "aprendizagem natural" e argumenta que a educação formalizada que acontece em sala de aula, mesmo exigindo muito mais tempo e esforço de alunos e professores, apresenta desempenho muito inferior ao da aprendizagem espontânea das crianças. Um dos motivos desta condição é a dissociação entre o conhecimento formal e a realidade observada pelos alunos, como por exemplo o princípio da inércia, que declara que todo objeto em movimento permanece em movimento em linha reta e velocidade constante, contraria as observações que o aprendiz fez durante toda sua vida, onde os objetos eventualmente perdem velocidade e param. Para solucionar impasses como este, o autor sugere a criação de micromundos, que são ambientes capazes de demonstrar conceitos que não são naturalmente observáveis, como princípios de física, matemática e programação de computadores.

Projetos de pesquisa como os mencionados na Seção 2.1 são exemplos de iniciativas que deram continuidade à pesquisa de Papert. Tais projetos utilizam materiais acessíveis, como Raspberry Pi, Arduino e componentes eletrônicos de baixo custo, na construção de experiências eletrônicas destinadas à educação, avaliando seus efeitos no aprendizado e comportamento dos alunos. Os resultados apresentados destacam o impacto da robótica e da programação, aliadas

a metodologias de ensino como o *problem based learning* no desenvolvimento de raciocínio lógico, pensamento crítico e cooperação. Apontam também efeitos no aprimoramento de habilidades sociais, pela cooperação com outros alunos, e na motivação em investigar e resolver problemas complexos.

Entretanto permanecem as questões: Como reformar as metodologias obsoletas da educação? Como construir um projeto que seja realmente abrangente e que não acabe esquecido na prateleira de uma biblioteca? Como gerar impacto no aprendizado das pessoas e na sua motivação pra construírem o próprio conhecimento? Primeiro considera-se que a tecnologia deva ficar progressivamente mais transparente, de modo que o aprendiz não perceba as camadas de complexidade entre si e o ambiente que manipula. Considera-se também, que os ambientes devam ser versáteis a ponto de apresentar conceitos distintos, explorando novas aplicações para as mesmas tecnologias. E também que as ferramentas preservem a autonomia do aprendiz, levando-o a construir teorias próprias, fundamentadas no conhecimento que já possui e permitindo que teste estas teorias interagindo e observando o ambiente.

Finalmente é preciso destacar que ambientes de programação didáticos não serão o bastante para gerar mudanças duradouras na educação. É necessário que o aprendiz reflita sobre como se dá o próprio aprendizado, que seja incentivado a desenvolver resiliência, interesse e foco, e que aprenda como motivar a si mesmo. Como também é necessário que os professores façam o possível para tornar as lições interessantes, aplicando metodologias contemporâneas, engajando os aprendizes com atividades significativas para eles, dando senso de propósito e encorajando-os a confiarem na própria inteligência.

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.

FLANAGAN, D. **JavaScript: The Definitive Guide: Activate Your Web Pages**. O'Reilly Media, 2011. ISBN 9781449308162. Disponível em: <https://books.google.com.br/books?id=6TAODdEIxrgC>.

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.

HAVERBEKE, M. **Eloquent JavaScript, 2nd Ed.: A Modern Introduction to Programming**. No Starch Press, Incorporated, 2014. ISBN 9781593275846. Disponível em: <https://books.google.com.br/books?id=mDzDBQAAQBAJ>.

HEINEN, E. et al. Raspiblocos: Ambiente de programação didático baseado em raspberry pi e blockly. In: **Anais do Simpósio Brasileiro de Informática na Educação**. [S.l.: s.n.], 2015. v. 26, n. 1, p. 567.

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.

MASCHIO, E. **Uma abordagem metacognitiva através de multiplas representações externas para o ensino de programação de computadores**. Dissertação (Mestrado), 2007.

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

PARKER, D. **JavaScript with Promises**. O'Reilly Media, 2015. ISBN 9781491930748. Disponível em: <https://books.google.com.br/books?id=9rnBCQAAQBAJ>.

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.

SANDE, W.; SANDE, C. **Hello World!: Computer Programming for Kids and Other Beginners**. Manning Publications Company, 2013. ISBN 9781617290923. Disponível em: <https://books.google.com.br/books?id=sWNfkQEACAAJ>.

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>.

APÊNDICE A – INSTRUÇÕES DE MONTAGEM DO MODELO ROBÓTICO

Curso de Tecnologia em Sistemas para Internet
UTFPR - Câmpus Guarapuava

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

Instruções de montagem do modelo robótico

O modelo robótico

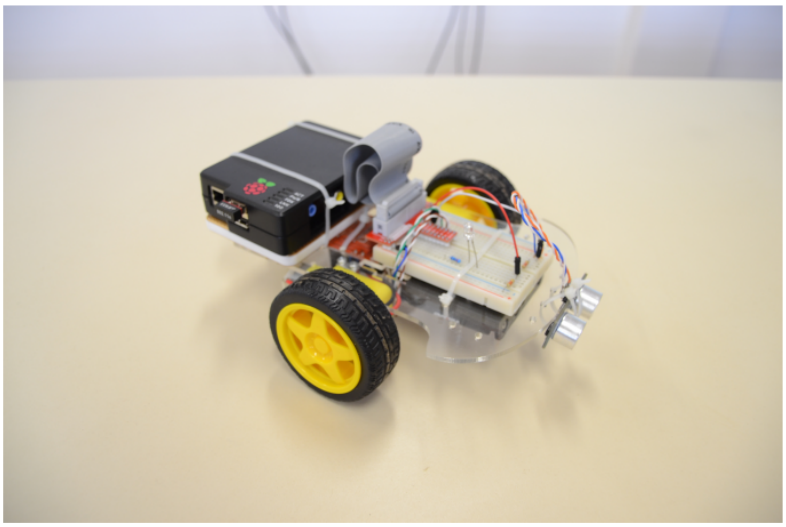


Diagrama de conexão dos motores com controlador L298N

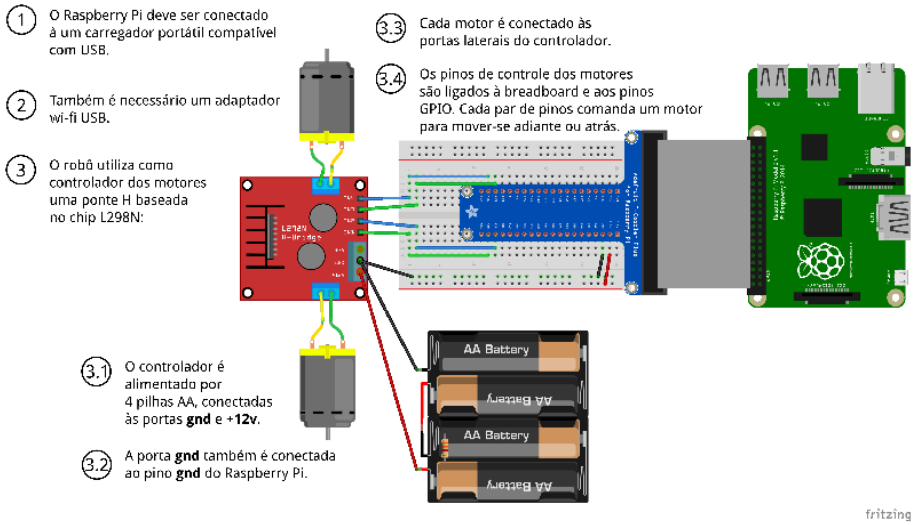
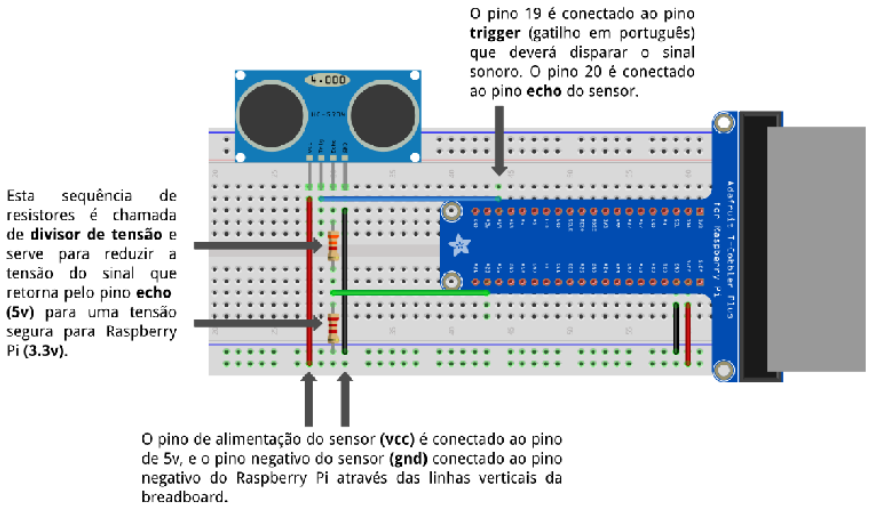


Diagrama de conexão do sensor de proximidade HC-SR04



APÊNDICE B – EXCERTO DO MATERIAL DE APOIO ÀS OFICINAS

Estruturas condicionais

Estruturas condicionais

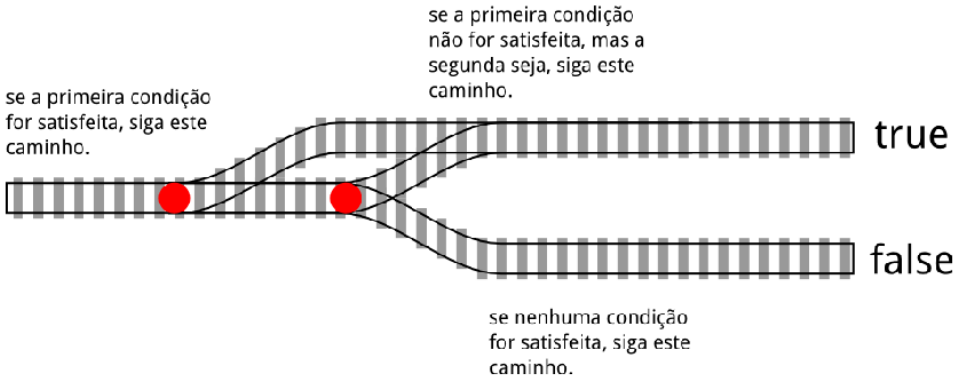
- Alguns programas devem ser capazes de se comportar de maneiras diferentes dependendo do ***input*** ou dos resultados que receberem, como por exemplo;
  - ***Se*** o aluno acertou a resposta adicionar 1 ponto ao contador;
  - ***Se*** o jogador acertou a nave alienígena, destruir a nave e tocar o som de explosão;
  - ***Se*** o arquivo não for encontrado, exibir uma mensagem de erro;

Estruturas condicionais

- A palavra ***if***, é usada para declarar estruturas condicionais na maioria das linguagens de programação;
- Na interface do Raspiblocos, os blocos que permitem declarar estas estruturas encontram-se dentro da categoria **Estruturas condicionais**;

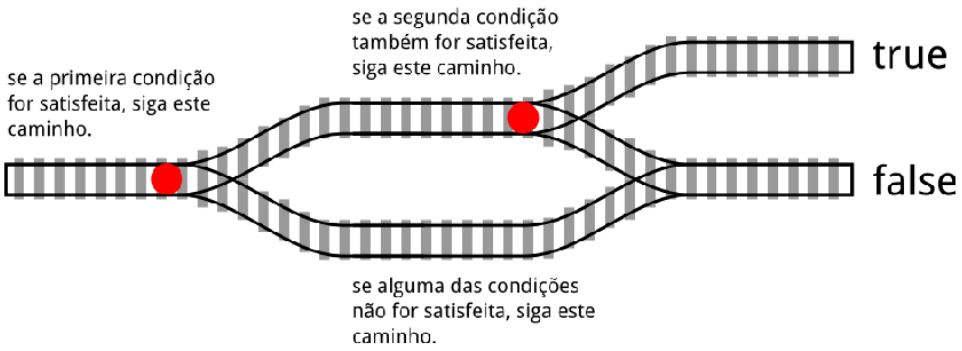
47/58

Estruturas condicionais + OR



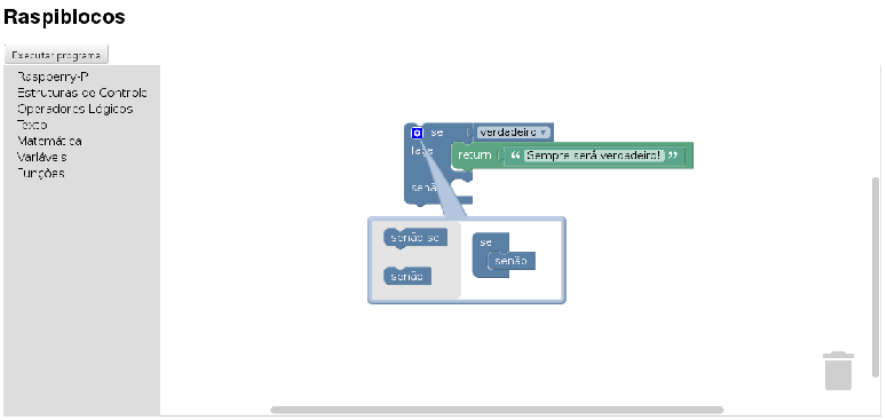
48/58

Estruturas condicionais + AND



49/58

Estruturas condicionais



50/58

Estruturas condicionais

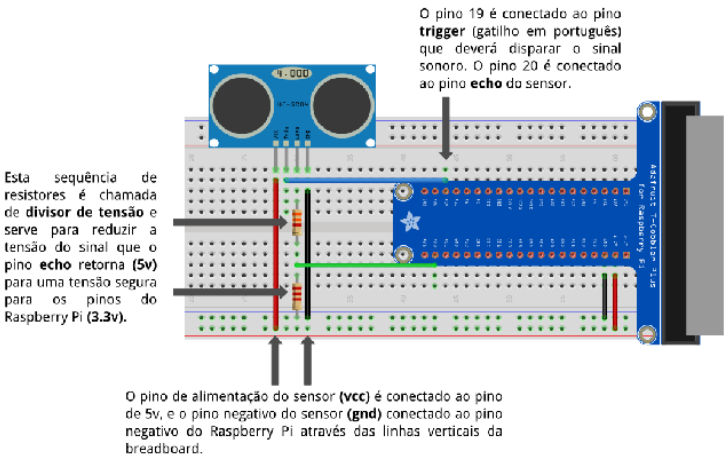
51/58

Se houver obstáculo então...

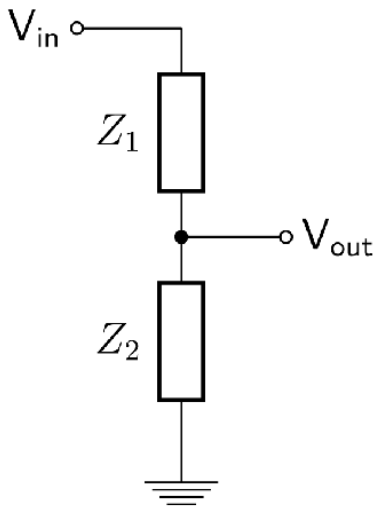
- Sensores ultrassônicos de proximidade funcionam lançando uma onda de som e medindo o tempo que esta onda leva pra retornar;
- O sensor HC-SR04 possui um pino pra alimentação **(vcc)**, um pino negativo **(gnd)**, um pino para disparar a onda sonora **(trig)** e um último para aguardar o retorno do som **(echo)**;

52/58

Se houver obstáculo então...



Se houver obstáculo então...



Se houver obstáculo então...

$$V_{out} = \frac{Z_2}{Z_1 + Z_2} * V_{in}$$

$$V_{out} = \frac{3.3k}{2.2k + 3.3k} * 5v$$

$$V_{out} = 3v$$

55/58

Se houver obstáculo então...