PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA

Universidade Federal do Rio de Janeiro

DISSERTAÇÃO DE MESTRADO

Estudo Exploratório Sobre o Uso da Robótica Educacional no Ensino de Programação Introdutória

André Rachman Dargains

dargains@ufrj.br

Orientador: Fábio Ferrentini Sampaio, Ph.D.

Área de Pesquisa:

Informática, Educação e Sociedade

Instituto de Matemática

Instituto Tércio Pacitti de Aplicações e Pesquisas Computacionais

PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA

Universidade Federal do Rio de Janeiro

ESTUDO EXPLORATÓRIO SOBRE O USO DA ROBÓTICA EDUCACIONAL NO ENSINO DE PROGRAMAÇÃO INTRODUTÓRIA

André Rachman Dargains
dargains@ufrj.br

Orientador:
Fábio Ferrentini Sampaio, Ph.D.

Área de Pesquisa:
Informática, Educação e Sociedade

RIO DE JANEIRO

2015

Dargains, André Rachman

Estudo exploratório sobre o uso da robótica educacional no ensino de programação introdutória / André Rachman Dargains. Rio de Janeiro: UFRJ, 2015.

Orientador: Fábio Ferrentini Sampaio, Ph.D.

Dissertação (Mestrado em Informática) – Universidade Federal do Rio de Janeiro, Instituto de Matemática, Instituto Tércio Pacitti de Aplicações e Pesquisas Computacionais, 2015.

ANDRÉ RACHMAN DARGAINS

ESTUDO EXPLORATÓRIO SOBRE O USO DA ROBÓTICA EDUCACIONAL NO ENSINO DE PROGRAMAÇÃO INTRODUTÓRIA

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

Aprovada em: Rio de Janeiro, 21 de outubro de 2015.

_____

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

_____

Claudia Lage Rebello da Motta, D. Sc. (NCE e PPGI/UFRJ)

_____

Carlo Emmanoel Tolla de Oliveira, Ph. D. (NCE/UFRJ)

_____

José Antônio dos Santos Borges, D. Sc. (NCE/UFRJ)

_____

Kate Cerqueira Revoredo, D. Sc. (UNIRIO)

Agradecimentos

A realização dessa dissertação de mestrado só foi possível devido a importantes incentivos e apoios de amigos, colegas e professores, com quem tive o prazer de conviver durante o período de estudos.

Primeiramente ao professor Fábio Ferrentini Sampaio, pela sua orientação, total apoio, disponibilidade, pelo saber que transmitiu de bom grado, pelas opiniões e críticas que me fizeram crescer, total colaboração na busca de soluções para problemas que surgiram ao longo da realização deste trabalho, e por todas as palavras de incentivo.

Agradeço aos meus pais, que sempre apoiaram minhas decisões e incentivaram meus sonhos, desde pequeno.

Agradeço a todos os professores que me acompanharam durante a graduação, em especial ao Prof. Marcos da Fonseca Elia, responsáveis pelo aperfeiçoamento deste trabalho.

Agradeço também aos colegas de mestrado da UFRJ, que sempre me incentivaram durante as aulas e pesquisas acadêmicas. Agradeço especialmente a Cristiane e Viviane, que abriram de boa vontade as portas de suas escolas e deram todo o apoio para que este trabalho pudesse ser realizado.

Por último, dirijo um agradecimento especial à minha esposa, Mariana, pelo seu apoio incondicional, incentivo, amizade e paciência demonstradas e total compreensão nos momentos mais difíceis.

Resumo

DARGAINS, André Rachman. Estudo Exploratório Sobre o Uso da Robótica no Ensino de Programação. 2015. Dissertação (Mestrado em Informática) – Instituto de Matemática, Instituto Tércio Pacitti de Aplicações e Pesquisas Computacionais, Universidade Federal do Rio de Janeiro, Rio de Janeiro, 2015.

Embora computadores cada vez mais versáteis e outros avanços tecnológicos permeiem nossas vidas, poucos ainda sabem ler e escrever a língua dos computadores. Muitas pesquisas e iniciativas públicas e particulares almejam reverter esse quadro, no entanto as dificuldades são desencorajadoras. Este trabalho tem como objetivo unir três pilares da busca por um melhor ensino de programação: a robótica educativa, a abordagem construcionista e a Taxonomia de Bloom revisada, uma ferramenta de avaliação, em um curso de programação introdutória, com o intuito de identificar métodos e estratégias para o crescimento da qualidade do ensino de programação introdutória. Para isso, duas aplicações foram realizadas com alunos de ensino médio em escolas públicas cariocas, e seus resultados analisados e comparados com uma disciplina de programação oferecida por uma universidade federal local. Espera-se, com esse estudo, a possibilidade de construir novas propostas e identificar caminhos para o fomento da educação em ciência, tecnologia, engenharia e matemática em escolas públicas brasileiras.

Abstract

DARGAINS, André Rachman. Estudo Exploratório Sobre o Uso da Robótica no Ensino de Programação. 2015. Dissertação (Mestrado em Informática) – Instituto de Matemática, Instituto Tércio Pacitti de Aplicações e Pesquisas Computacionais, Universidade Federal do Rio de Janeiro, Rio de Janeiro, 2015.

Although more and more versatile computers and other technological advances permeate our lives, few people know how to read and write the language of computers. Many researches and public and private initiatives seek to change this situation. The difficulties, however, are discouraging. This work aims to unite three pillars of the search for a better educational programming: educational robotics, the constructionist method and the revised Bloom's Taxonomy in an introductory programming course to identify methods and strategies for the growth of introductory programming learning. To achieve this, two applications were conducted with high school students in Rio de Janeiro's public schools with no programming background, and the results were analyzed and compared with a programming course offered by a local federal university. The results of this study are expected to open ways for new proposals and identify ways for the promotion of science, technology, engineering and mathmatics education in Brazilian public schools.

Lista de Figuras

Lista de Quadros

Lista de Tabelas

Sumário

1. INTRODUÇÃO

Este Capítulo tem como objetivo estruturar o contexto do trabalho, a problemática e a justificativa para sua composição, bem como expor de modo sucinto seus demais Capítulos.

1.1 O contexto atual

Tornou-se comum referir-se aos jovens como "nativos digitais", devido a sua aparente fluência com as tecnologias digitais (Prensky, 2001). E, de fato, muitos deles sentem-se muito à vontade em enviar mensagens de texto, jogar online e navegar na web. Mas será que isso realmente os faz fluentes nas novas tecnologias? Embora os jovens interajam com mídias digitais a todo tempo, poucos podem criar os seus próprios jogos, animações ou simulações. É como se eles pudessem "ler", mas não "escrever".

Segundo Resnick (2007), a fluência digital não exige apenas a capacidade de conversar, navegar e interagir, mas também a capacidade de projetar, criar e inventar novas mídias. Para isso, é necessário aprender algum tipo de linguagem de programação. Nas palavras de Flannery:

> Vejo a programação de computadores como uma extensão da escrita. A capacidade de programar permite "escrever" novos tipos de coisas - histórias interativas, jogos, animações e simulações. E, como acontece com a escrita tradicional, há fortes razões para que todos possam aprender a escrever em código, mas eu vejo razões muito mais profundas e mais amplas para aprender. No processo de aprender a programar, as pessoas aprendem muitas outras coisas. Eles não estão apenas aprendendo a programar, elas estão programando para aprender. Além de aprender ideias matemáticas e computacionais (como variáveis e condicionais), eles também estão aprendendo estratégias para a resolução de problemas, elaboração de projetos, e comunicação de ideias. Estas habilidades são úteis não apenas para os profissionais de TI, mas para todos, independentemente da idade, origem, interesses ou ocupação. (Flannery et al., 2013)

Está se tornando cada vez mais claro que a simples introdução de tecnologia no processo educativo não é suficiente para garantir a sua integração, já que a tecnologia por si só não leva à mudança. Pelo contrário, é na própria maneira pela qual os professores usam a tecnologia que mudanças poderão ocorrer na educação (Carr et al., 1998). Para Zhao (2003), a fim de que os professores se tornem fluentes na tecnologia educacional, é necessário que eles consigam ir além da mera competência no uso das ferramentas disponíveis. Ou seja, não basta apenas saber utilizar uma ferramenta; deve-se estudar, aprender e entender como utilizá-la para alcançar todo o seu potencial.

O recente aumento do interesse em aprender e, consequentemente, em ensinar a programar, é refletido no aumento da disponibilidade de portais voltados ao

assunto, como a Kahn Academy[1], CodeAcademy[2] e Code.org[3]. Muitos desses portais tem-se centrado principalmente em novas oportunidades de trabalho e de carreira, encorajados pelo rápido crescimento do número de postos de trabalho para os programadores e profissionais de TI, com a demanda superando em muito a oferta[4].

Uma revisão sistemática de literatura sobre o processo de ensino-aprendizagem de programação para iniciantes, publicados nos últimos 10 anos no SBIE[5] e WIE[6], feita por Aureliano e Tedesco (2012), apontou para um crescimento gradual no interesse da comunidade científica sobre o assunto (Figura 1.1).

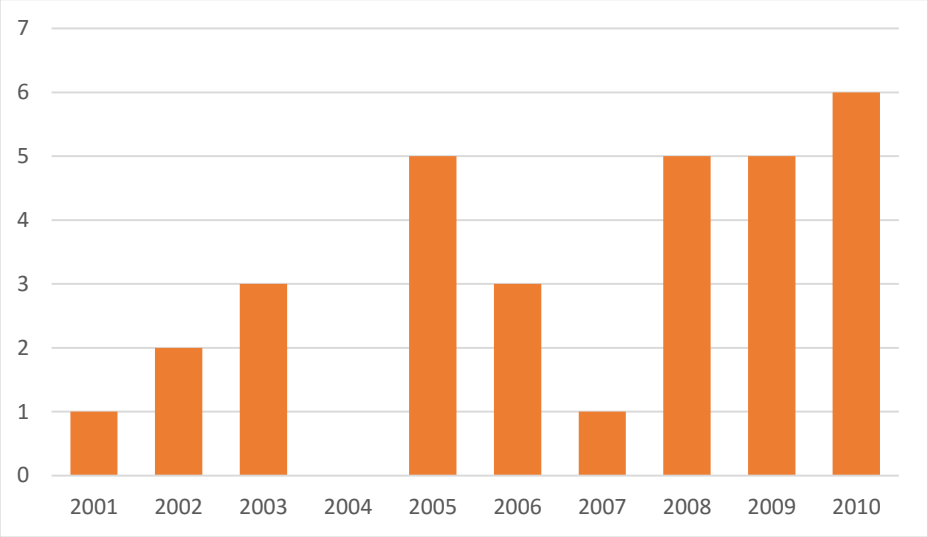


Figura 1.1: Crescimento do interesse na pesquisa do ensino-programação introdutória (Aureliano e Tedesco, 2012)

[1] http://pt.khanacademy.org/computing/computer-programming
[2] http://www.codeacademy.com
[3] http://www.code.org
[4] De acordo com a SOFTEX (2009, p.31, p.47), organização responsável pelo programa do governo para promoção da excelência do software brasileiro, o número de empresas de software e serviços de tecnologia da informação vem crescendo a partir de 2003, a uma taxa média anual de 4,8%, tendo no ano de 2009 cerca de 70 mil empresas de TI (Tecnologia da Informação) e crescimento médio anual do número de pessoas ocupadas de 12,6% com cerca de 540.000 profissionais de TI em 2009.
[5] Simpósio Brasileiro de Informática na Educação
[6] Workshop de Informática na Educação

No entanto, as disciplinas de algoritmos, programação e estrutura de dados, normalmente localizadas no início dos currículos dos cursos de graduação em computação e informática, consistem em grande obstáculo para os acadêmicos iniciantes, sendo causa de altos índices de reprovação e evasão (Ferreira, 2005).

No que se refere às dificuldades específicas enfrentadas por parte dos acadêmicos, podem ser citadas: a confusão existente em relação à significação da informação recebida e suas interdependências, causando desinteresse pela disciplina (Smith, 1981); a dificuldade de interpretação dos enunciados e de abstração das informações contidas nos problemas (Falkembach et al., 2003) e, consequentemente, a falta de habilidade nas resoluções de tais problemas (Olsen, 2005). Em sala de aula, essas situações são comumente encontradas em turmas constituídas por grupos de aprendizagem heterogêneos, cujos indivíduos apresentam ritmos distintos de aprendizagem e de dificuldades, exigindo uma demanda de interação para atendimento, acompanhamento, mediação e avaliação por parte do professor muitas vezes impossível de ser alcançada (Raabe e Silva, 2005).

Segundo Neto e Cechinel (2006), é comum deparar-se com o relato abaixo:

> O professor pode iniciar a aula solicitando que os alunos resolvam um exercício cuja solução implementada com parâmetros é mais simples e otimizada que outra solução que não utiliza este recurso, como o seguinte: Crie um programa que leia dois números, calcule o fatorial de cada um deles e imprima o resultado da soma dos fatoriais. Por ainda desconhecerem os conceitos de parâmetros, os alunos devem gerar algoritmos sem parametrização. Após explicar o conteúdo, o professor pode apresentar uma solução com parâmetros, para mostrar aos alunos as facilidades oferecidas pelos parâmetros em termos de reutilização de código. A aula, que parece perfeita, termina com a tradicional pergunta do professor: Alguma dúvida?. Com raras exceções, os alunos respondem que não possuem dúvidas e, professor e alunos deixam a sala de aula com a certeza do dever cumprido (Neto e Cechinel, 2006).

Os autores prosseguem colocando que, embora não existam pesquisas que registrem esse fato, análises empíricas podem ser realizadas para encontrar formas de melhorar a qualidade do ensino de programação.

Considerando todas as informações aqui colocadas, podemos chegar à conclusão de que a programação não é um assunto fácil de ser estudado. Ela exige a compreensão correta de conceitos abstratos, com os quais muitos estudantes têm problemas de aprendizagem, devido à sua própria natureza

(Lahtinen et al., 2005). Além disso, muitas vezes não há recursos suficientes e os alunos sofrem com a falta de conhecimentos necessários que deveriam já ter sido aprendidos. Ademais, os grupos de estudantes de programação em uma sala de aula geralmente são grandes e heterogêneos e, portanto, é difícil projetar o material didático de um modo que seria benéfico para todos. Tais características muitas vezes levam a altas taxas de abandono em cursos de programação.

1.2 Justificativa

A capacidade de programar oferece muitos benefícios importantes como a expansão do alcance do que se pode criar e expressar com o computador, ao mesmo tempo ampliando a gama do que se pode aprender. Ainda, segundo autores como Wing (2006), essa habilidade apoia o desenvolvimento do "pensamento computacional", ajudando a conhecer importantes estratégias de resolução de problemas e de design (como a modularização e design iterativo) que passam para domínios de não programação. Uma vez que a programação envolve a criação de representações externas de seus processos de resolução de problemas, ela também oferece ao programador oportunidades para refletir sobre o seu próprio pensamento (diSessa, 2000).

No entanto, aprender a programar pode ser muito difícil para iniciantes, não só para jovens como também pessoas de qualquer faixa etária (Kelleher e Pausch, 2005). Além dos desafios de aprender a formar soluções estruturadas para os problemas e entender como os programas são executados, programadores novatos também precisam aprender uma sintaxe rígida e comandos que podem ter nomes aparentemente arbitrários ou talvez confusos. Enfrentar esses desafios simultaneamente pode ser opressivo e muitas vezes desanimador para esses iniciantes.

Nos últimos anos, a comunidade internacional interessada no ensino de computação e de TICs na educação tem apresentado diferentes trabalhos focando no ensino de programação ainda nos primeiros anos do ensino regular (Friedrich et al., 2012; Goh e Aris, 2007; Val e Pastor, 2012; Flannery, 2013), ao

mesmo tempo em que procuram alternativas pedagógicas e computacionais para ensiná-la (Major et al., 2011; Chiou, 2004).

Segundo Neto e Cechinel (2006), ainda hoje, não existe consenso sobre um conjunto universal de estratégias indicadas para a apresentação, estudo e prática desses conteúdos, obrigando desta forma cada Instituição a percorrer seus próprios caminhos na busca de soluções. Com o intuito de amenizar esta situação, vários professores e pesquisadores vêm trabalhando diretamente na realização de experimentos práticos, assim como na construção de ferramentas e metodologias para facilitar o processo de ensino-aprendizagem desta área.

Um problema constante no ensino da programação é a própria avaliação e comparação de desempenho de diferentes práticas de ensino. Como colocado por de Jesus e Raabe:

> Entre os muitos desafios enfrentados por um professor de programação introdutória está a avaliação. Como um professor poderia mensurar objetivamente a aprendizagem de seus alunos? Supondo que os alunos obtenham ótimo desempenho, como saber, por exemplo, se o desempenho não foi causado por testes demasiadamente fáceis? O professor poderia comparar o desempenho de sua turma com uma turma de referência. Mas, nesse caso, como saber se a turma de referência é realmente referência? (de Jesus e Raabe, 2009).

Segundo Barker e Ansorge (2007), *early adopters* de robótica na sala de aula têm relatado muitos elogios. No entanto, há uma clara falta de pesquisa quantitativa sobre a forma como a robótica pode facilitar o aprendizado de disciplinas STEM[7] para estudantes. A maioria das pesquisas envolvendo robótica na sala de aula foi conduzida com alunos do ensino médio e de faculdade, com resultados dependentes da percepção de professores ou estudantes, em vez de projetos de pesquisa rigorosos baseados em dados de desempenho do aluno. A utilização de uma taxonomia, como propõe este trabalho, tem como objetivo neutralizar essa falha.

O levantamento de literatura apropriada para o embasamento teórico deste trabalho encontrou inúmeras pesquisas com o objetivo de atacar os problemas relacionados ao ensino de programação. As buscas focaram em três aspectos

---

[7] Sigla em inglês que significa Ciências, Tecnologia, Engenharia e Matemática, disciplinas comumente consideradas como as principais em uma educação científica abrangente.

do problema: dificuldades e motivação dos alunos, a falta de uma ferramenta para construção de material, mensuração e comparação de resultados e um método apropriado para o ensino de programação.

Para isso, a revisão de literatura deste trabalho foi feita acerca de três tópicos chave: o uso da robótica educativa como facilitadora do ensino/aprendizagem, o uso da Taxonomia de Bloom como norteadora na construção de cursos e medição de resultados, e experiências da abordagem construcionista como motivadora e catalizadora do aprendizado, todos no âmbito do ensino de programação. Pesquisas que apresentam as dificuldades no ensino da programação, suas causas e possíveis soluções também foram levantadas, como literatura de apoio ao presente trabalho.

A revisão, no entanto, não constatou a existência de pesquisas que mostrassem simultaneamente os três pontos mencionados acima. Para que o potencial do uso da robótica no ensino de programação possa ser aproveitado em uma proposta de inovação curricular é necessário levar em consideração a forma adequada de utilizá-la e a avaliação dos resultados da aprendizagem. A estratégia de uso da robótica educacional como parte do processo de aprendizagem deve estar associada a estratégias de avaliação, que não só estimule a aprendizagem como também avalie consistentemente o ganho na aprendizagem.

1.3 Hipótese e objetivos

Este trabalho, portanto, tem como objetivo unir os três pilares discutidos anteriormente na aplicação de um curso de programação introdutória para alunos do ensino médio, com o uso da robótica educacional, utilizando a metodologia construcionista para a construção e norteadora de suas aulas, e a Taxonomia de Bloom [8]revisada para desenhar o material didático e provas, avaliar os ganhos cognitivos dos participantes e comparar os resultados com outras pesquisas.

[8] A Taxonomia de Bloom será tratada com maior detalhamento no Capítulo 3.

Espera-se, assim, investigar a possibilidade de alunos, sem conhecimento prévio de programação, alcançarem os níveis superiores da Taxonomia de Bloom revisada, por meio de um curso de programação introdutória utilizando a robótica educativa como apoio. Uma reposta satisfatória para essa pergunta pode significar um passo importante na direção da aplicação de uma proposta unificada para o ensino de programação, adaptável ao currículo do nível médio de escolas brasileiras.

Para tal, foi realizado um estudo exploratório dividido em duas etapas: a primeira, chamada Oficina, trata de uma aplicação piloto de ensino de programação introdutória com o uso da robótica educacional. Seu objetivo foi coletar experiências e averiguar a possibilidade de sucesso de uma segunda execução, muito mais ampla. Nessa etapa já foram utilizados os insumos do trabalho, como a Taxonomia de Bloom revisada, o conteúdo didático aplicado no curso de Computação de uma universidade e os métodos de coleta comentados mais à frente.

A segunda etapa do trabalho utiliza todos os dados coletados durante a execução da Oficina para construir a segunda aplicação, o Curso. Embora os aspectos cruciais tenham sido mantidos da primeira para a segunda etapa, alguns foram alterados conforme a experiência vivida no estudo. A principal diferença entre as etapas é a duração da aplicação realizada em cada uma, que passou de oito horas em cinco aulas na primeira etapa para trinta horas em doze aulas durante o curso.

1.4 A organização do trabalho

Esta dissertação está organizada em cinco Capítulos resumidamente descritos abaixo:

O Capítulo 1, de introdução, tem como objetivo estruturar o contexto do trabalho, a problemática e a justificativa para sua composição, bem como expor de modo sucinto seus demais Capítulos.

O Capítulo 2 apresenta a revisão de literatura realizada sobre os três pontos discutidos no Capítulo 1, notadamente o uso da robótica educativa, o uso da

Taxonomia de Bloom e experiências da abordagem construcionista, todos no âmbito do ensino de programação. Pesquisas que apresentam as dificuldades no ensino da programação, suas causas e possíveis soluções também foram levantadas, como literatura de apoio. Essa busca foi composta para o embasamento teórico da dissertação, onde são apresentados resumidamente os caminhos trilhados e resultados alcançados das pesquisas mais relevantes ao tema deste trabalho.

O Capítulo 3 introduz a Taxonomia de Bloom, ferramenta norteadora na construção do material didático utilizado nas aplicações deste trabalho e na avaliação e comparação de seus resultados.

O Capítulo 4 apresenta a metodologia usada nesta pesquisa, onde são introduzidos os objetivos gerais e específicos, a abordagem de construção das aplicações e as ferramentas utilizadas.

O Capítulo 5 apresenta a análise e discussão dos dados coletados das aplicações através dos diários do professor, provas e questionários de opinião, discutidos à luz do referencial teórico do trabalho e da revisão de literatura realizada no Capítulo 2.

O Capítulo 6 conclui o trabalho com considerações sobre todo o processo e propõe aprofundamento para pesquisas futuras.

2. O ENSINO DE PROGRAMAÇÃO

Este Capítulo apresenta a revisão de literatura realizada sobre os três pontos discutidos no Capítulo 1, nomeadamente o uso da robótica educativa, o uso da Taxonomia de Bloom e experiências da abordagem construcionista, todos no âmbito do ensino de programação. Pesquisas que apresentam as dificuldades no ensino da programação, suas causas e possíveis soluções também foram levantadas, como literatura de apoio.

2.1 Dificuldades no ensino de programação

Diferentes autores comentam sobre a reputação da programação de ser difícil de aprender (Kelleher e Pausch, 2005). Alguns problemas comuns que impedem a formação de um maior número de programadores são a grande evasão devido à dificuldade dos cursos (Lahtinen et al., 2005). Embora a programação possa ser uma habilidade muito útil e uma carreira recompensadora, é geralmente aceito que um novato levará aproximadamente 10 anos até tornar-se um expert (Robins et al., 2003).

Por definição, novatos não têm muitos dos pontos fortes dos experts. Estudos revisados por Winslow (1996), por exemplo, concluíram que os programadores iniciantes estão limitados ao conhecimento superficialmente organizado, faltam modelos mentais detalhados, falham em aplicar o conhecimento relevante e ao revisar um programa, usam a abordagem de leitura “linha a linha” ao invés de analisarem “pedaços” ou estruturas do programa, tornando-o mais significativo. Estudos coletados por Soloway e Spohrer (1989) mostram déficits de compreensão dos programadores iniciantes em vários tópicos específicos de linguagem de programação (tais como variáveis, laços, matrizes e recursão), através de deficiências em seu planejamento e testes de código e em questões mais gerais, relativas à utilização de planos de programas. Segundo Wiedenbeck (1999), alunos novatos são muito limitados e concretos em sua compreensão de programas, faltando-lhes o uso da abstração e interpretação de problemas.

Ao longo dos anos esse cenário não apresenta grandes alterações. Entre os motivos apontados por pesquisas mais recentes estão: a falta de contextualização do processo de aprendizagem (Figueiredo e Afonso, 2006), a natureza do método de ensino tradicional, baseado em leituras e sintaxes específicas de linguagem de programação (Lahtinen et al., 2005), e as dificuldades na compreensão dos conceitos básicos de programação, como variáveis, tipos de dados ou endereços de memória (Miliszewska e Tan, 2007 e Lahtinen et al., 2005), descritos como conceitos abstratos, sem uma representação equivalente na vida real.

Trautman (2015) indica as quatro fases que todo estudante de programação enfrenta atualmente:

- Fase 1 – Início facilitado: é fácil encontrar referências, guias, tutoriais e ajuda na literatura e em sites de internet. Tudo parece simples e os programas são construídos sem muitas dificuldades.
- Fase 2 – Queda brusca: o primeiro baque de autoconfiança acontece quando o estudante percebe que não é capaz de construir qualquer programa que deseja, do zero. A ajuda fica mais escassa e qualquer avanço parece ser mais devido à sorte que ao próprio conhecimento.
- Fase 3 – Planalto: com o material de referência cada vez mais escasso e programas cada vez mais complexos, o estudante atinge um planalto no seu processo de aprendizagem. Os tópicos exigem mais tempo para serem dominados e os avanços são difíceis de serem mensurados corretamente.
- Fase 4 – Ascenção: ao chegar a esse ponto, o estudante sente-se confiante para construir um programa complexo e já apresenta uma bagagem de conhecimento relativamente vasta. Os programas criados começam a ficar robustos e eficientes.

Segundo o autor, muitos aspirantes a programador cessam seus avanços na fase 3, ao perceberem que o aprendizado da programação poderia ser mais difícil que previamente estimado. A postura gerada pelas inúmeras fontes de ensino de programação de que a programação será sempre simples e auxiliada por outrem é posta em xeque devido à carência de material de referência e aumento da complexidade dos programas vivenciados à medida que o estudante evolui no aprendizado.

Combinando esses fatores, temos uma nova geração de estudantes de ciência da computação, para quem os computadores têm sido uma presença constante em suas vidas, uma ferramenta importante, mas não se sentem motivados a aprender a programa-los (Lethbridge et al., 2007).

2.2 Ensino construcionista

Segundo Arendt (2003), o construcionismo (Papert, 1991) consiste na hipótese mestra piagetiana de que não existem estruturas cognitivas inatas, sendo estas construídas pelo sujeito, no decorrer de suas ações no meio; a ênfase está na priorização dos objetivos do aluno, experiências e estratégias metacognitivas. Reeves (1998) afirma que alunos atingem um estado de equilíbrio cognitivo através da reconstrução de conceitos, esquemas, modelos mentais, e outras estruturas cognitivas em face de novas informações e experiência que podem entrar em conflito com construções anteriores. Essa linha de pensamento está diametralmente oposta ao formato tradicional de ensino, chamado de instrucionismo (Reeves, 2006).

Adeptos da metodologia instrucionista salientam a importância de metas e objetivos que existem para além do aluno (Quadro 2.1). Essas metas e objetivos são retirados de um domínio de conhecimento, por exemplo, álgebra, ou extraído a partir de observações dos comportamentos de especialistas dentro de um determinado domínio, por exemplo, de programadores. Uma vez que as metas e objetivos são delineados, eles são sequenciados em hierarquias de aprendizagem, geralmente representando uma progressão da ordem de ensino inferior a superior. Em seguida, a instrução direta é projetada para resolver cada um dos objetivos da hierarquia, muitas vezes empregando estratégias instrucionais derivados de psicologia comportamental (Rieber, 1992). Segundo Reeves (1998), relativamente pouca ênfase é colocada sobre o aluno, por si só, que é geralmente visto como um receptor passivo de instruções.

Quadro 2.1: Diferenças entre as abordagens instrucionista (sala de aula tradicional) e construcionista

Instrucionismo	Construcionismo
O currículo é iniciado com partes de um todo. Enfatiza habilidades básicas.	O currículo enfatiza conceitos amplos, iniciando com o todo e expandindo para incluir as partes.
Aderência estrita ao currículo estático é altamente recomendada.	Busca de questões e interesses dos alunos é altamente recomendada.
Materiais são em sua maioria livros texto e de atividades.	Materiais incluem fontes primárias de materiais e materiais manipulativos.
O aprendizado é baseado na repetição.	O ensino é interativo, desenvolvido a partir do que o aluno já aprendeu.
Professores disseminam informação para os alunos; alunos são recebedores passivos de conhecimento.	Professores dialogam com os alunos, ajudando-os a construir seu próprio conhecimento.
O papel do professor é baseado na autoridade.	O papel do professor é interativo, baseado na negociação.
Avaliação através de provas e respostas corretas.	Avaliação inclui os trabalhos do aluno, observações e pontos de vista, assim como provas. O processo é tão importante quanto o produto.
O conhecimento é inerte.	O conhecimento é dinâmico, sempre alterado a partir das experiências.
Alunos trabalham na maioria do tempo sozinhos.	Alunos trabalham na maioria do tempo em grupos.

A abordagem construcionista exige uma multiplicidade de perspectivas, de modo que os alunos tenham uma gama completa de opções para construir seu próprio conhecimento. Na educação científica, essa abordagem pode proporcionar aos alunos oportunidades para redescobrir as teorias atualmente aceitas de uma determinada ciência, bem como teorias rivais que podem, eventualmente, substituir as atuais posições (Papert, 1991). Nesse caso, professores podem fornecer orientação ou motivação para ajudar os alunos nas suas descobertas, mas não dirigem excessivamente o processo de aprendizagem. Segundo Papert (1991):

> O construcionismo compartilha com o construtivismo a visão de que aprender é “construir estruturas de conhecimento” através da internalização progressiva de ações. [...] Em seguida, adiciona a ideia de que isso acontece de forma especialmente feliz em um contexto onde o aluno está conscientemente empenhado na construção de uma entidade, seja ela um castelo de areia na praia ou uma teoria do universo. (PAPERT, 1991)

Segundo Ackermann (2006), devido ao seu maior foco na aprendizagem através da prática, a abordagem de Papert ajuda a entender como as ideias se formam e se transformam quando expressas através de diferentes meios de comunicação, quando atualizadas em contextos particulares e quando trabalhadas por mentes individuais.

Ackermann afirma ainda que, na visão de Papert, mergulhar em situações desconhecidas, ao custo de experimentar uma sensação momentânea de perda, também é uma parte crucial do aprendizado. Somente quando um aluno realmente viajou através de um mundo, através da adoção de diferentes perspectivas, ou colocando diferentes "óculos", pode começar um diálogo entre as experiências locais e inicialmente incompatíveis.

Orey (2010) discute o construcionismo em seu livro, e apresenta uma lista de potenciais benefícios decorrente do seu uso:

- **Maior motivação**: Os alunos podem escolher os seus temas, a extensão do conteúdo, e o modo de apresentação, e eles mesmos constroem seus projetos para atender seus próprios interesses e habilidades. Esses tipos de atividades são altamente motivadores para os alunos.

- **O aumento da capacidade de resolver problemas:** A metodologia construcionista incentiva os alunos a se envolverem em contextos complexos e mal definidos. Desde o início, os alunos identificam os seus temas e problemas, para então procurar as possíveis soluções. Ao participar de trabalho tanto independente quanto colaborativo, os alunos melhoram suas habilidades de resolução de problemas, desenvolvendo assim suas habilidades de pensamento crítico.
- **Melhoria das habilidades de pesquisa:** O construcionismo oferece uma conexão real com o contexto. Alunos realizam pesquisas utilizando vários recursos de informação e, ao localizar os recursos por si próprios (ao contrário do instrucionismo, onde todas as informações entendidas como necessárias são apresentadas pelo professor), suas habilidades de pesquisa se desenvolvem e melhoram.
- **Aumento da colaboração:** Nas fases de atividades, os alunos criam e organizam seus grupos. Eles compartilham conhecimentos e constroem artefatos colaborativamente. Através da colaboração, eles desenvolvem habilidades de comunicação sociais e obtém múltiplas perspectivas.
- **Aumento da habilidade de gerenciar recursos:** Um curso construcionista bem desenhado oferece aos alunos experiência em organização de projetos e gerenciamento de tempo com recursos apropriados.

O foco do construcionismo na prática em sala de aula sugere a utilização de instrumentos concretos no processo de ensino-aprendizagem. O encorajamento de estudantes na construção de circuitos eletrônicos a partir de componentes básicos leva-os a buscar soluções concretas a problemas práticos, gerando significados por meio de experiências e ações (Valente, 1993; Papert, 1994). Para chegarem a uma solução final satisfatória, empregam e desenvolvem habilidades de formulação e teste de hipóteses, raciocínio lógico, resolução de problemas por meio de erros e acertos, entre outras (Zilli, 2004). A robótica tem se tornado uma ferramenta muito utilizada pela comunidade científica, explorada extensivamente na Seção 2.3 deste trabalho.

Papert (1980) acrescenta que as crianças são motivadas pelo controle que elas têm quando utilizam ferramentas tecnológicas, como computadores, que lhes permitem realizar todas as decisões tomadas sobre os seus projetos, através da programação. As crianças são inclusive motivadas, durante o processo, a aprender sobre outros tópicos que estão de alguma forma relacionados ao processo de descoberta da solução. Ainda segundo o autor, os alunos geralmente são capazes de aprender por conta própria um conceito de engenharia, pré-requisito para utilizar a tecnologia ou para atingir seus objetivos de projeto, uma vez que o conjunto seja demasiado interessante e significativo para eles.

Esta abordagem resulta em uma melhor retenção de conteúdo em longo prazo do que a abordagem tradicional (Norman e Schmidt, 1992), maior motivação (Albanese e Mitchell, l993), e o desenvolvimento de habilidades para resolução de problemas (Hmelo et al., 1997). Outras pesquisas também indicam que a educação experiencial melhora o desenvolvimento social e acadêmico de crianças, incentivando a interação social e a aprendizagem cooperativa (Deen et al., 2001; Slavin, 2000).

Papert (1980) constatou que a robótica educacional é uma excelente maneira de colocar a teoria construtivista em prática. Em seus estudos, ele mostra que crianças que aprendem com a robótica são capazes de imaginar-se no lugar do robô e entender como a programação de um computador funciona: os participantes do estudo foram capazes de transferir a sua compreensão do mundo real para a compreensão da lógica e princípios matemáticos. Papert acreditava que o que faz com que muitos conceitos sejam difíceis para as crianças entenderem é a falta de materiais do mundo real que demonstram tais conceitos. Ele aceitava que os robôs programáveis são flexíveis e poderosos o suficiente para serem capazes de demonstrar ideias que antes não apresentavam analogias simples do mundo real.

2.3 Uso da robótica educativa no ensino de programação

A programação não é uma disciplina trivial de se aprender ou ensinar. Entre os motivos apontados por pesquisas sobre as dificuldades no seu ensino destacam-se a falta de contextualização do processo de aprendizagem (Figueiredo e Afonso, 2006), a natureza do método de ensino tradicional, baseado em leituras e sintaxes específicas da linguagem sendo estudada (Lahtinen et al., 2006), e as dificuldades na compreensão dos conceitos básicos de programação, como variáveis, tipos de dados ou endereços de memória, descritos como conceitos abstratos, sem uma representação equivalente na vida real (Lahtinen et al., 2005; Miliszewska e Tan, 2007).

Muitos autores têm então buscado alternativas para tornar esse aprendizado mais compreensível e eficiente. A robótica, uma dessas alternativas, já foi classificada como facilitadora de aprendizagem de princípios científicos e matemáticos através da experimentação de materiais concretos (Rogers e Portsmore, 2004), incentivadora de classes baseadas em resolução de problemas (Rogers e Portsmore, 2004; Nourbakhsh et al., 2005; Robinson, 2005) e promotora de aprendizagem cooperativa (Nourbakhsh et al., 2005; Beer et al., 1999).

Concomitantemente, outros estudos sobre o uso da robótica na sala de aula têm apontado para um alto grau de interesse e envolvimento dos alunos, fomentando o interesse em carreiras de matemática e ciências (Rogers e Portsmore, 2004; Robinson, 2005).

Diferentes iniciativas documentam os ganhos diretos do uso da robótica no ensino de programação de computadores. Uma revisão sistemática de literatura, conduzida por Major e seus colegas em 2011 (Major et al., 2011) apontou mais de 60 estudos unindo o ensino de programação introdutória e robótica, publicados nos últimos 14 anos em congressos e revistas internacionais. Os resultados dos trabalhos, em sua grande maioria, apontaram para ganhos efetivos no processo de aprendizagem de programação utilizando a robótica versus o ensino tradicional de programação. A revisão também mostra que em

68% dos trabalhos foram usados materiais de robótica físicos, contra 20% de trabalhos que utilizaram simuladores e 12% que apresentaram ambos. Esse resultado, portanto, aponta para uma preferência popular para o uso de circuitos, placas e fios, ou seja, material físico para o uso da robótica em aplicações didáticas.

Um argumento para a preferência por materiais físicos no ensino com robôs é que os alunos veem os robôs como brinquedos e diversão (Mauch, 2001). De fato, um kit de robótica amplamente usado não só pela comunidade científica, mas também em escolas é desenvolvido pela Lego[9], um conhecido fabricante de brinquedos de blocos de construção para crianças. Alunos que usam este kit podem construir e programar robôs usando os mesmos materiais que eles têm em casa. Isso faz com que tudo o que aprendam com os kits pareça divertido também (Barker e Ansorge, 2007).

Fagin e Merkle (2003) e Barnes (2002) usaram robôs para ajudar a ensinar as linguagens de programação Java e ADA, respectivamente, em salas de aula. A principal ênfase em seus cursos foi de ensinar as sintaxes e suas estruturas básicas enfocando os aspectos de engenharia e mecânica de robôs. Outros cursos que também utilizam a robótica têm-se centrado na construção e programação dos robôs em si (Nourbakhsh et al., 2005; Beer et al., 1999).

Goh e Arris (2007) basearam-se na plataforma Lego MindStorms (Lego, 2015) para ministrar um curso focado na construção de robôs de competição. Separados em grupos, o objetivo dos alunos foi o de se organizarem em funções distintas (programador, engenheiro, designer) e montar o robô mais performático da turma. Diversos ganhos foram notados, como trabalho em equipe, pesquisa de informações por conta própria e a ruptura da visão da programação e robótica como cadeiras muito complexas.

Chiou (2004) usou a robótica para atacar o crescimento do desinteresse por cursos baseados em matemática, ciências e tecnologia. Ele argumenta que a implementação não pode ser mal planejada, ou o interesse pela robótica diminuirá rapidamente à medida que o fator novidade deixa de existir. O autor

[9] mindstorms.lego.com

afirma ainda que a robótica educacional precisa ser implementada com grande cuidado para que não seja causado o efeito contrário, ou seja, inadvertidamente afastar os alunos interessados.

Moore (1999) usou o tema robótica como um "gancho" para captar a atenção dos seus alunos de quinta série em outras disciplinas, encorajando seus alunos a pensar de forma crítica sobre robôs. De acordo com Moore (1999) os alunos são capazes de construir e programar robôs, entender os conceitos de geometria, escrever e compartilhar histórias com colegas e comparar sistemas de tecnologia com os sistemas do corpo humano. O estudo, no entanto, não fornece uma avaliação quantitativa do programa de robótica. Rogers e Portsmore (2004) também ensinaram jovens estudantes utilizando robôs, e projetaram um currículo utilizando robôs LEGO que ensina alunos da quinta série sobre engenharia.

Stager (2009) explora o uso do construcionismo em uma classe, afirmando que há pelo menos cinco caminhos para o uso adequado da robótica educacional em sala de aula:

- **Robótica como disciplina** - Robótica é ensinada como uma disciplina própria. Competições de robótica populares, como a *First LEGO League*, são exemplos dessa abordagem.
- **Ensinar conceitos STEM específicos** – A robótica pode ser usada para ensinar conceitos científicos físicos, tais como: máquinas simples, força, torque, potência, fricção, vantagens mecânicas; conceitos de ciência da computação como programação, depuração e feedback; conceitos matemáticos como frações, variável, operações aritméticas, etc.
- **Unidades temáticas** - Estudantes constroem e programam robôs para modelar máquinas e sistemas, como aeroportos, fábricas, parques de diversões ou uma cidade. A expectativa é que as disciplinas escolares tradicionais e seus conceitos subjacentes sejam abordados ou incorporados a estes temas.
- **Temas curriculares** – A robótica é usada como um meio para resolver problema específicos ligados a um assunto do currículo formal. Um exemplo poderia ser: "Identificar um problema na África Subsaariana e

construir um robô para resolver esse problema." O realismo da solução pode ser subordinado a pensar sobre a natureza do problema.

- **Estilo livre** – A robótica e programação de computadores são usados como material de construção como parte do laboratório intelectual do aluno e do seu veículo de autoexpressão. O aluno pode usar os materiais para fazer o que quiser. Poderosas ideias emergem nas atividades desse tipo de contexto.

Stager continua afirmando que há o florescimento de uma nova teoria pedagógica, baseada em quatro fatores críticos: bons questionamentos, materiais apropriados, tempo suficiente para realizar a atividade proposta e uma cultura de apoio ao aluno. Segundo o autor, essa nova abordagem não está restrita à robótica ou à computação.

Embora diferentes análises apresentando as variadas conquistas da robótica educacional tenham mostrado que a robótica é eficaz no ensino da programação, foi notado que cada trabalho utiliza uma ferramenta para orientação da construção, avaliação e comparação de resultados. O efeito é uma vasta gama de estratégias diferentes empregadas sobre o assunto, incapacitando a comunidade internacional de poder compará-las e eleger os melhores caminhos. Surge então a necessidade de alinhamento de uma estratégia de construção e avaliação para que os trabalhos possam ser comparados entre si e os projetos mais atraentes, replicados com facilidade.

2.3.1 Arduino

Introduzida em 2005, a plataforma Arduino foi projetada no Instituto de Design Interativo de Ivrea[10] para fornecer uma maneira barata e fácil a amadores, estudantes e profissionais para a criação de dispositivos que interagem com seu ambiente através de sensores e atuadores.

O Arduino permite aos usuários criar protótipos eletrônicos funcionais, tanto objetos ligados a um computador quanto funcionando individualmente. A plataforma é capaz de ler a partir de uma vasta gama de sensores, controlar um

[10] https://interactionivrea.org/

amplo espectro de dispositivos de saída, e comunicar com software em execução num computador ou através de uma rede.

Há muitos passos necessários para realizar a mais básica das tarefas com um microcontrolador: a escolha de um microcontrolador em particular, pensar no circuito necessário para usá-lo, comprar as peças necessárias, montá-las, fazer o download do software necessário para programar o microcontrolador, descobrir uma forma do microcontrolador conectar-se ao computador, instalar os drivers necessários, comprar ou construir um dispositivo externo para programar o microcontrolador, aprender a escrever o código do microcontrolador (o que pode exigir a leitura uma folha de dados com centenas de páginas), escrever o código, descobrir quais são os argumentos de linha de comando necessários para compilar e fazer o upload do código, etc.

O papel do Arduino é eliminar ou mitigar tantos passos acima quanto possível, com uma combinação de hardware e software.

Figura 2.1: Placa Arduino modelo UNO R3, usada neste trabalho

O modelo usado neste trabalho, o UNO R3 (Figura 2.1), é composto por uma placa de hardware livre projetado em torno de um microcontrolador Atmel ATmega328, que opera a 5 V com 2 Kb de RAM, 32 Kb de memória flash para armazenar programas e 1 Kb de EEPROM para armazenar parâmetros. A velocidade de clock é de 16 MHz, que se traduz em execução de cerca de 300.000 linhas de código fonte C por segundo (Quadro 2.2).

Quadro 2.2: Uso de memória na placa Arduino

Memória	Tamanho	Tipo	Uso
Flash	32.768 bytes	Não volátil	Guarda o código fonte do programa e o bootloader do Arduino.
SRAM	2048 bytes	Volátil	Espaço operacional para acesso de variáveis e funções.
EEPROM	1024 bytes	Não volátil	Repositório permanente para dados do usuário, como preferências.

A placa tem 14 pinos de entrada/saída digital e 6 pinos de entrada analógica, que permitem ao usuário anexar várias placas de extensão, chamadas shields. Há também um conector USB para conexão com computador e uma tomada para ligar a uma fonte externa de alimentação 6-20 V (por exemplo, uma bateria de 9 V), ao executar um programa enquanto não estiver conectado ao computador via cabo USB (Figura 2.2).

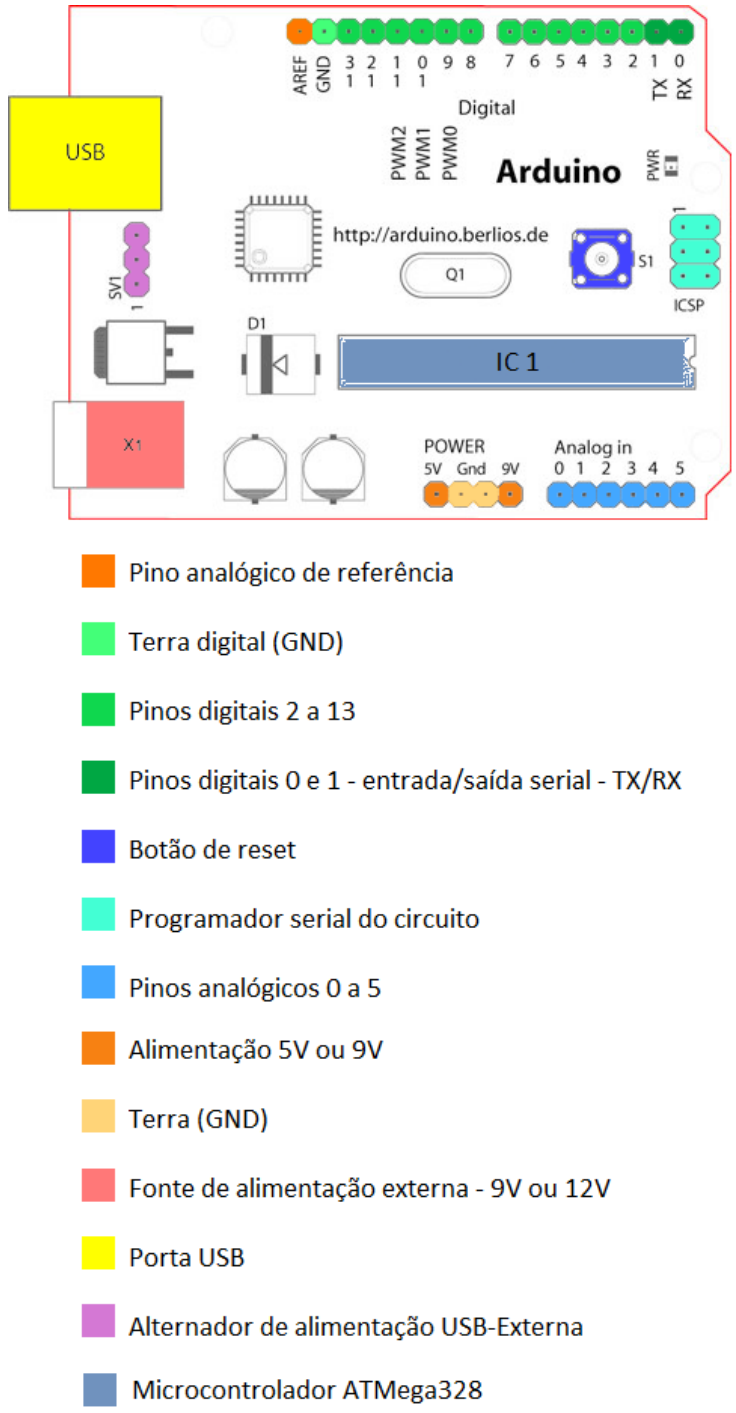


Figura 2.2: As diferentes partes da placa Arduino UNO R3 (imagem retirada de http://arduino.cc/en/Guide/Board?from=Tutorial.ArduinoBoard)

O Arduino conta ainda com um ambiente de desenvolvimento integrado simples (IDE), executável em diferentes sistemas operacionais, permitindo aos usuários escreverem programas utilizando a linguagem Wiring (um subset do C e C++).

Segundo Evans (2011), a plataforma Arduino é por si só muito útil para projetos de micro controladores, mas isso apenas não é suficiente para impulsionar sua popularidade e ampla adoção. Para impulsionar sua adoção, a empresa criadora evita manter fechados o desenho da placa de interface e o ambiente de desenvolvimento, trazendo como consequência um profundo enraizamento na prática emergente de hardware livre. Ao contrário de softwares open source, dos quais Linux é geralmente o exemplo frequentemente citado, um hardware open source busca a colaboração onde os objetos físicos são o resultado. Isso gera um modelo de desenvolvimento distribuído com contribuidores residentes em diferentes partes do mundo. Ao contrário de sistemas fechados, projetos de código aberto permitem uma liberdade individual aos usuários para acessar o código ou design fonte de um projeto, fazer melhorias e redistribuir essas melhorias para a comunidade em geral.

Segundo Mellis e outros (2007):

> Desde o início, Arduino nasceu como um projeto colaborativo entre diferentes universidades e indivíduos. Sempre foi nosso objetivo de maximizar o impacto dentro do mundo acadêmico, tentando levantar questões sobre como nós projetamos artefatos interativos. Uma das principais questões a abordar é o que se refere à propriedade intelectual. Design de interação física é uma disciplina jovem. Acreditamos que uma boa maneira de fazê-la crescer para acomodar as necessidades da sociedade é procurar uma maneira de licenciar os resultados que os torna disponível para outras pessoas usarem. Optamos por fazer toda a parte plataforma do movimento de cultura livre, liberá-los sob licenças permissivas (Mellis et al., 2007).

Uma grande variedade de desenvolvedores selecionou o Arduino como uma plataforma de desenvolvimento para todos os tipos de sistemas computacionais (Rubio et al., 2013). O intuito dos criadores era de projetar uma placa muito fácil de usar: seu público alvo inicial era de artistas e designers, e não de programadores. Além disso, graças ao fato do Arduino ter seu código aberto, ele é apoiado por uma vasta comunidade de usuários que compartilham suas ideias, projetos e soluções.

Uma característica importante do Arduino é que o usuário pode criar um programa de controle no computador, transferi-lo para o microcontrolador usando a interface de programação Wiring (Figura 2.3) e ele será executado automaticamente. Removendo a conexão do cabo USB para o PC, o programa

ainda será executado normalmente cada vez que o botão de reset for pressionado. Mesmo ao remover a bateria e guardar a placa Arduino em um armário durante seis meses, quando voltar a ligar a bateria, o último programa armazenado será executado. Isso significa que é necessário conectar a placa no computador para desenvolver e depurar um programa, mas uma vez que é feito, o computador deixa de ser necessário para executar o programa.

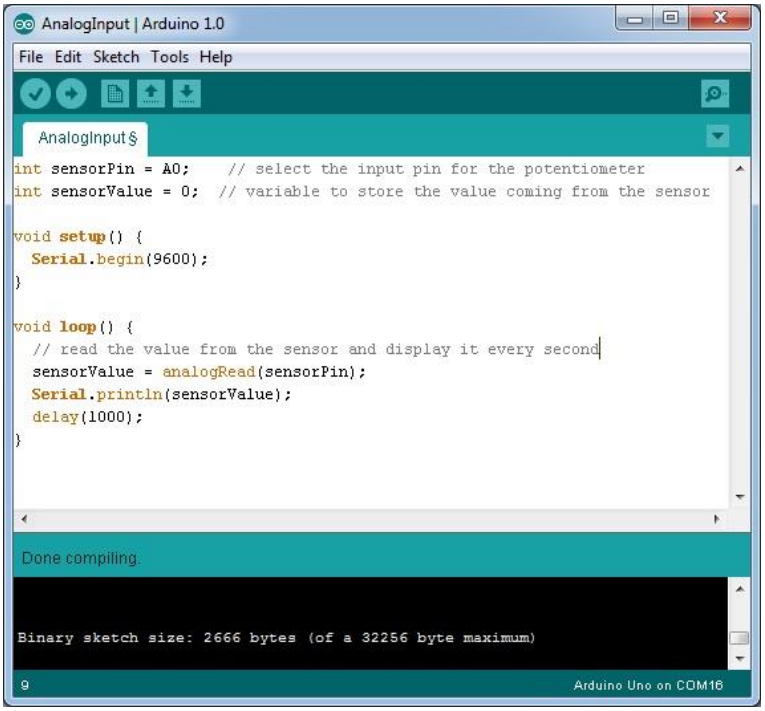


Figura 2.3: Wiring, ambiente de programação nativo do sistema Arduino

Neste trabalho foi utilizado um kit específico, disponível no mercado, contendo diferentes componentes para construção de circuitos básicos (resistores, leds, botões, fios, protoboard, etc.), além de uma placa Arduino modelo UNO R3. Os kits podem ser montados por um preço consideravelmente baixo, uma vez que o Arduino segue o conceito de hardware de código aberto e os componentes eletrônicos contidos no kit serem baratos e facilmente encontrados.

A comunidade virtual usuária do sistema Arduino é extremamente ativa. A principal ferramenta de divulgação para Arduino é através de um website[11], editado de forma colaborativa por quinze pessoas de vários países. Um wiki[12] publicamente editável fornece um espaço para os usuários do Arduino para escrever tutoriais, fornecer exemplo de código, fazer upload de projetos de circuitos e compartilhar projetos criados com Arduino. No fórum[13], também publicamente disponível, os usuários podem pedir ajuda em uma grande variedade de questões, recebendo conselhos e sugestões de como fazer sua placa funcionar, depurar seu código, descobrir quais componentes usar para uma determinada tarefa, etc. O projeto não se limita ao inglês; as pessoas também têm escrito tutoriais em português, alemão, japonês, chinês e muitas outras línguas.

O Arduino, portanto, é indicado em diferentes estudos (Rubio et al., 2012; Mellis et al., 2007; Araújo et al., 2013) como uma boa escolha de microcontrolador no ensino/aprendizado de robótica. O Arduino é facilmente encontrado no mercado a um baixo custo e, por ser uma peça de hardware open source, possui muitas versões alternativas de diferentes níveis de qualidade, criadas por diversas empresas, disponíveis por preços ainda mais acessíveis.

2.3.2 Ambientes visuais de programação

Valentim (2000) apresenta um problema comum entre as linguagens de programação: embora estejamos acostumados a pensar e a construir algoritmos de forma gráfica, os computadores não são programados utilizando desenhos ou fluxogramas.

O exemplo utilizado é o da construção do algoritmo para o cálculo de fatorial de um número N inteiro. O cálculo é obtido através da multiplicação de N pelos seus antecessores até se chegar ao número 1. Sendo assim, o fatorial de 5 é obtido por 5 x 4 x 3 x 2 x 1. Segundo o autor:

---

[11] www.arduino.cc
[12] http://playground.arduino.cc/
[13] http://forum.arduino.cc/

> Observa-se que há uma ação que se repete dentro do procedimento de cálculo: a multiplicação é realizada várias vezes. Para solucionarmos esta questão será empregado um laço. Na programação de computadores toda vez que precisamos repetir N vezes uma determinada sequência de comandos, utilizamos um laço ou estrutura de repetição. O laço possibilita ao programador repetir uma parte do programa quantas vezes forem necessárias. (Valentim, 2000)

Uma solução do cálculo, representada em formato de fluxograma, é apresentada na Figura 2.4:

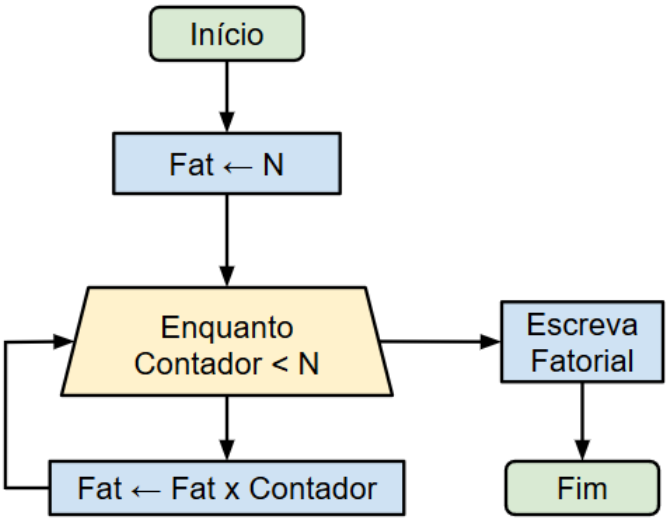


Figura 2.4: Fluxograma do algoritmo para cálculo do fatorial de um número inteiro N

No ensino de programação é comum utilizar o pseudocódigo antes da introdução de linguagens propriamente ditas. O pseudocódigo é uma linguagem de programação simples que permite que o aluno se concentre inicialmente na solução do problema proposto sem ter que dominar a sintaxe de uma linguagem de programação (de Souza, 2009). O algoritmo representado acima, no formato de código, poderia ser escrito na linguagem de programação em pseudocódigo como mostrado na Figura 2.5:

```
    fat, i, n: INTEIRO;
ESCREVA ("DIGITE UM NÚMERO: ");
LEIA (n);
Fat := n;
PARA i DE 1 ATÉ (n – 1)  FAÇA
      fat := fat * i;
FIM_PARA;
ESCREVA (fat);
```

Figura 2.5: Algoritmo do cálculo do fatorial de N

Na figura 2.6 pode-se observar que no programa há várias palavras adicionais, tais como "program", "var", "begin", "end". Elas são uma exigência do compilador, que nada mais é do que um programa ou conjunto de programas que tem por objetivo traduzir um programa escrito em uma linguagem de alto nível, o código-fonte, em um programa expresso em uma linguagem de baixo nível.

```
Program FATORIAL;
var
   I, N, FAT: integer;
Begin
      Writeln('Digite um número:');
      readln (N);
      FAT := N;
      For i:=1 to  (N-1)  do
          FAT := FAT * I;
       writeln('O fatorial de  ', N, ' equivale a:  ', FAT);
end;
```

Figura 2.6: Algoritmo do cálculo do fatorial de N escrito na linguagem Pascal

Esta tradução envolve a análise sintática, a qual tem por objetivo verificar se o programa está escrito dentro das regras da linguagem de programação do compilador, por exemplo, na linguagem Pascal todo comando deve terminar com um ponto e vírgula, caso contrário o compilador deverá emitir uma mensagem de erro. Após, realizada esta análise, o compilador partirá para a análise semântica, que envolve a verificação de contexto, por exemplo, em Pascal não podemos somar o número 3 com o valor booleano verdadeiro, ou usar uma

variável X que não esteja declarada, somente após a passagem pelas 2 verificações o programa será convertido para a linguagem de máquina, e poderá ser executado pelo computador.

Gerada a partir desse contexto, a programação por demonstração é uma técnica que visa aproximar o usuário cada vez mais do ambiente de programação, sem que seja necessário aprender uma linguagem específica. Ferreira et al. (2010) afirmam:

> A aplicação da programação por demonstração permite que os usuários, por meio de um ambiente baseado em interface gráfica, aprendam a programar por meio de exemplos, especificando “o que” deverá ser feito, sem ter a preocupação de “como” será feito.

Segundo Smith (2000), os estudantes que aprendem a programar utilizando a técnica de programação por demonstração tendem a obter melhores resultados do que aqueles que aprenderam pelo método tradicional. A facilidade que as linguagens textuais oferecem para descrever um determinado problema ou algoritmo está mais relacionada com a maneira pela qual os computadores operam e não com o processo cognitivo de programação (Brown & Kimura, 1994).

A técnica de programação por demonstração apresenta um alto grau de sinergia com a metodologia de programação visual (Coura, 2006; Ferreira et al., 2010). Enquanto a linguagem textual exige do aluno o conhecimento das especificidades de cada linguagem de programação que deseja utilizar, a linguagem visual aborda apenas o processo de construção do algoritmo. Ao remover as questões necessárias de sintaxe das linguagens de programação, os alunos passam a focar na semântica do programa. Além disso, a representação visual de um problema está muito mais próxima com a forma pela qual a solução é obtida ou entendida se comparada à representação textual (Evangelista, 2001).

Na linguagem de programação, entende-se como sintaxe a forma como as instruções de uma linguagem são escritas, mas sem atender ao seu significado. Enquanto no C++ os blocos de comando que serão executados são limitados por “{ }”, em Pascal são limitados por “begin” e “end”. A sintaxe é a gramática das linguagens de programação, o conjunto de regras que devem ser atendidas

para que o compilador – e consequentemente o computador – entenda e execute o código escrito.

A semântica, por outro lado, corresponde à descrição do significado das instruções válidas de uma linguagem. Por exemplo, a sintaxe da instrução *if* da linguagem C++ é: *if* (condição) {instruções} e sua semântica é: “se o valor da expressão for verdadeiro, as instruções incorporadas serão executadas pelo programa”. Portanto, quando um aluno de programação tem como única preocupação a semântica de seu programa, ele consegue direcionar seus esforços para o aprendizado da lógica de programação, comum a todas as linguagens.

No intuito de desenvolver as competências básicas do aprendizado de programação, existe uma discussão na comunidade científica sobre quais as melhores estratégias para ensiná-la. No entanto, ainda não existe um consenso sobre o melhor método para ensino e avaliação na educação básica. Alguns trabalhos advogam a favor das linguagens visuais, orientadas ao design, como o Scratch (Aureliano e Tedesco 2012; Scaico et al., 2012; Araújo et al., 2013) e plataformas que sigam essa linha, como App Inventor (Gomes e Melo, 2013).

Scratch (Maloney et al., 2010) é um ambiente produzido pelo Lifelong Kindergarten Group do Massachusetts Institute of Technology Media Lab. O software foi disponibilizado em maio de 2007 para download em www.scratch.mit.edu. Tanto o website quanto o software, possuem versões em português. Através do software Scratch é possível trabalhar os seguintes conceitos específicos de programação: sequência, iteração, condição, variáveis, execução paralela, sincronia, interação em tempo real, lógica booleana, números randômicos, tratamento de evento e criação de interfaces.

Além dos conceitos de programações descritos acima, o software Scratch proporciona através de seus comandos “variáveis”, “operadores”, “sensores” e “controle” os recursos necessários para realizar, entre outras possibilidades, operações matemáticas com ou sem substituições de variáveis, construções de figuras geométricas, manipulação das coordenadas cartesianas, raciocínio lógico usando condicionalidades do tipo “se, senão” e movimentos de objetos/scripts. Pode-se ainda elencar como potencialidades do software, o

desenvolvimento da criatividade, a manipulação de mídia, construções de programas que coordenam simultaneamente animações, textos, músicas, sons e gráficos, além de permitir o compartilhamento de suas produções no sítio próprio da web.

Aureliano e Tedesco (2012) investigaram se o ambiente de programação visual Scratch possibilita um melhor desempenho para os alunos iniciantes em programação em comparação à linguagem textual tradicional. Os autores concluíram que a abordagem com Scratch gerou melhores resultados, mas que é necessária mais investigação sobre o tema. Scaico (2012) apresentou o relato de experiência de ensino de programação com Scratch, através de uma abordagem de ensino que estimula a criatividade. O artigo relata uma olimpíada de programação, atividade realizada para motivar os alunos e que serviu como ferramenta de avaliação.

O trabalho de Gomes e Melo (2013) segue apostando no paradigma de programação visual como estratégia de ensino de programação. As autoras apresentam uma proposta metodológica e o relato de experiência de ensino de programação por meio do ambiente de programação App Inventor[14], criado, mantido e hospedado na universidade MIT. No trabalho de Gomes e Melo, a grande motivação foi permitir que os participantes aprendessem lógica de programação enquanto desenvolviam seus próprios aplicativos.

2.4 Uso de taxonomias em cursos de programação

Viu-se necessário a adoção de uma métrica para avaliação e comparação dos ganhos cognitivos dos alunos durante os cursos, e uma diretriz a ser adotada para a construção da apresentação dos conceitos e exercícios aos alunos. Para entender melhor como isso se dá, foram levantados estudos propondo cursos ou oficinas de programação que usem algum tipo de taxonomia como norteador para sua construção.

[14] http://www.appinventor.mit.edu/

Howard (1996) propôs identificar claramente os objetivos para cada lição, e atribuí-los a um determinado nível da taxonomia. A maioria das aulas tem uma série de objetivos de conhecimento, mas o alcance de níveis superiores varia durante o curso. Ao demarcar o nível mais alto alcançado em cada aula em um gráfico, o autor consegue discernir a evolução do curso de acordo com a profundidade do conhecimento que os alunos assimilaram.

Scott (2003) afirma que a avaliação deve medir o nível alcançado por cada aluno, e a nota deve depender de suas conquistas. Em particular, ele nota que o seu ensino tem vindo a cobrir os níveis 3 (aplicação) e 6 (de avaliação) da Taxonomia.

Buck e Stucki (2001) delineiam uma abordagem pedagógica de dentro para fora com base na Taxonomia de Bloom para o desenvolvimento cognitivo. Este framework permite aos alunos compreender os conceitos básicos antes de serem convidados a aplicá-los.

Lahtinen e Ahoniemi (2005) mostram preocupação com o uso de taxonomias para o projeto de visualizações para ajudar os alunos a compreender a programação, não só nos níveis cognitivos elementares, mas também o suficiente para apoiar o seu progresso através do curso sugerido no seu trabalho. Os autores olham para cada nível de taxonomia de Bloom e discutem os tipos de material visual que seriam relevantes para apresentar e interagir com o material em cada nível, resultando em uma categorização dos exemplos de visualização dos programas.

Doran e Langan (1995) informam sobre um projeto que implementou uma abordagem baseada em cognição (usando a Taxonomia de Bloom) para os dois primeiros anos de um curso de computação, utilizando sequenciamento estratégico (espiral) e os níveis de mestria associados de tópicos chave. O projeto também investigou o uso de laboratórios fechados, com feedback frequente e uso precoce de trabalho em equipe. Eles usaram micro objetivos do curso mapeados em níveis específicos da Taxonomia de Bloom.

Johnson e Fuller (2007) relatam dois estudos de cursos de ciência da computação realizados pelos alunos do primeiro ano do curso de ciência da computação em uma universidade: um painel de avaliação avaliado por

instrutores, e entrevistas com os instrutores em cada curso. Uma conclusão significativa destes estudos é que o nível mais significativo para muitos dos cursos estudados é o nível de aplicação; aplicação de técnicas para a criação de programas parece ser o cerne do estudo da computação. No entanto, para problemas de aplicação complexos os alunos precisam usar as habilidades que estariam classificados nos níveis de análise/síntese/avaliação. Os autores propõem um novo patamar de "aplicação alta" para assuntos como computação. Isto abrange a atividade cognitiva que é destinada a resolver um problema, mas que precisa das habilidades tradicionalmente de "nível mais alto" que envolvam os alunos no nível de análise/síntese/avaliação.

Para finalizar, Kramer (2007) identifica a abstração como uma habilidade fundamental para muitas áreas da ciência da computação, e discute em seu trabalho o modelo de Piaget do desenvolvimento cognitivo (Piaget e Inhelder, 1969). Seu argumento é baseado em estudos que mostram que uma percentagem significativa da população em geral não desenvolve suas habilidades cognitivas a ponto de fazer uso significativo dos processos operacionais formais.

Kramer ainda argumenta que levar alunos até esta etapa é um pré-requisito para os que estudam diferentes aspectos da computação, e que devemos conceber cursos que garantam que os alunos alcancem este estágio de aprendizado cognitivo antes de ensinar outros tópicos de computação. Para Kramer, a capacidade de abstração é tão importante que ele sugere a mensuração dessa competência como uma forma de selecionar os alunos para cursos de computação.

2.5 Considerações sobre a Revisão de Literatura

Revisões sistemáticas de literatura apontam para um crescimento cada vez maior de artigos sobre educação STEM (Ciências, Tecnologia, Engenharia e Matemática, do inglês), através de estudos sobre programação e robótica. Enquanto a programação tem sido apontada como habilidade cada vez mais necessária para sociedade contemporânea, a robótica educacional ganha espaço como facilitadora no ensino da programação.

Devido à grande gama de pesquisas sobre os assuntos, o escopo deste trabalho contemplará apenas trabalhos que envolvam diretamente um ou mais dos três pilares discutidos no Capítulo de introdução: ensino de programação com o uso da robótica educacional, uso da Taxonomia de Bloom revisada para desenho e avaliação de resultados, e uso da metodologia construcionista para montagem das aulas e exercícios.

Percebemos, a partir das Seções anteriores, que os assuntos abordados são extensamente discutidos pela comunidade científica em diferentes formatos e por razões distintas. Os objetivos variam desde explorar a eficácia de uma determinada metodologia de ensino a testar hipóteses promissoras. Considerando toda essa gama de trabalhos, no entanto, não encontramos estudos que liguem os pontos anteriormente discutidos. Esse fato deixa visível a necessidade de maior estudo – almejada por esta pesquisa – para aprofundamento dos conhecimentos nesta área.

3. A TAXONOMIA DE BLOOM

Este Capítulo introduz a Taxonomia de Bloom, ferramenta norteadora na construção das aulas, exercícios e provas utilizados nas aplicações deste trabalho e na avaliação dos ganhos cognitivos dos participantes do estudo. Na Seção 3.2 os domínios da Taxonomia são apresentados, e na Seção 3.3 a versão revisada da Taxonomia, que foi utilizada neste trabalho, é discutida.

3.1 A busca por uma ferramenta de organização didática e avaliação confiável

Avaliação é a coleta sistemática de evidências por meio das quais se determinam mudanças que ocorrem nos alunos e como elas ocorreram (Krathwohl et al., 1973). Sendo que, é formativa toda a avaliação que ajuda o aluno a aprender e a se desenvolver, que participa da regulamentação das aprendizagens e do desenvolvimento no sentido de um projeto educativo (Perrenoud, 1999).

Langsch (1999) afirma que existem várias técnicas para realização da avaliação, mas é importante definir qual seu objetivo, ou seja, se é apenas uma mera medida ou uma referência para contribuir em futuras aprendizagens. Pois, através da análise de resultados podem ser traçadas novas estratégias e caminhos de ensino-aprendizagem. Uma das técnicas propostas por Langsch é a taxonomia de aprendizagem ou educacionais.

Segundo Fuller (2007), taxonomias de aprendizagem descrevem e classificam os estágios em dimensões cognitivas, afetivas e emocionais que uma pessoa pode passar durante um processo de aprendizagem. Taxonomias de aprendizagem podem ser usadas para definir os objetivos do currículo de um curso, de modo que ele não seja apenas descrito na base dos tópicos a serem abordados, mas também em termos do nível desejado de entendimento para cada tópico (Moon, 2002).

Essas taxonomias também são amplamente utilizadas para descrever as etapas de aprendizagem pelo que o aluno passa em um determinado tópico. Por exemplo, um aluno pode ser capaz de recitar de cor o que é recursão, mas não é capaz de implementar um algoritmo recursivo. Um professor pode ter em vista que os seus alunos aprendam um tópico em um determinado nível em uma taxonomia (por exemplo, os alunos podem vir a ser capazes de compreender o conceito de recursão sem necessariamente aplicá-lo). Uma vez que isso tenha sido feito, o professor pode avaliar os alunos no nível escolhido através de uma escolha adequada de perguntas ou exemplos (Lister e Leaney, 2003).

Sabemos que é comum alunos - e algumas vezes até nós mesmos - dizerem não ter dúvida alguma sobre um determinado tópico passado em classe e,

mesmo assim, mostrando dificuldades ao fazer um exercício que contém esse tópico. Surge aí uma incompreensão por parte do educador, pois esse tópico foi abordado em aula, os alunos pareceram entender, e mesmo assim não conseguem aplicar o novo conhecimento em um exercício de fixação ou com pequenas variações.

Segundo Scott (2003), uma taxonomia pode explicar a razão desses acontecimentos tão comuns: acompanhar o professor durante suas explicações de um tópico ou técnica implica na categoria “compreensão” da taxonomia, enquanto que aplicar aquele conhecimento em um exercício acarreta o uso da categoria “síntese”.

Taxonomias de aprendizado têm sido também utilizadas em muitos outros contextos, tais como introduzir os alunos a uma taxonomia de aprendizagem para sensibilizá-los e melhorar o seu nível de compreensão e de suas técnicas que estudam (Cukierman e McGee Thompson, 2007). Elas também são usadas para estruturar exercícios de cursos baseados em ou com uso assistido de computadores (Lahtinen e Ahoniemi, 2005; Hernán-Losada et al., 2004).

Taxonomias educacionais são ferramentas importantes na avaliação da realização de alunos em cursos e na construção de objetivos de aprendizado (Fuller et al., 2007). No entanto, taxonomias não são triviais de serem usadas e em alguns momentos pesquisadores podem até discordar entre si de algumas classificações (Johnson e Fuller, 2007). Isso pode, inclusive, explicar a baixa taxa de adesão e penetração das taxonomias educacionais. Para Whalley e outros (2006), muitas vezes as descrições dos níveis de uma taxonomia são difíceis de serem interpretados no contexto dos exercícios de programação. Thompson e outros (2008) relatam uma situação onde foram observadas discrepâncias significativas entre as classificações sugeridas por diferentes professores para uma mesma questão de um teste.

Não obstante, pesquisadores vem tentando encontrar formas de medir e avaliar os ganhos cognitivos de suas aplicações, e diversos deles utilizaram a Taxonomia de Bloom com altos graus de sucesso. Jesus e Raabe (2009), em seu trabalho, apresentam uma série de pesquisas que utilizaram tal taxonomia. McCracken e seus colegas (2001) realizaram um estudo conjunto entre

universidades de alguns países com o objetivo de avaliar os alunos de programação introdutória. Nesta pesquisa foi elaborado um instrumento de avaliação através de um trabalho conjunto entre os professores envolvidos. De forma muito semelhante, Lister (2004) realizou experimentos em sete países com o objetivo de testar alunos iniciantes na leitura e entendimento de código de programas. Novamente, os pesquisadores utilizaram um instrumento padrão de avaliação, de maneira que a comparação entre o desempenho das diferentes turmas pôde ser realizada. O mesmo instrumento de avaliação utilizado por estes autores foi reutilizado e aprimorado no trabalho de Whalley (2006). Os instrumentos de avaliação utilizados nas pesquisas de Whalley e Lister foram elaborados segundo a taxonomia de Bloom, uma estrutura conceitual concebida para auxiliar a definição de objetivos de aprendizagem.

3.2 A Taxonomia de Bloom e seus domínios

A Taxonomia de Bloom foi criada por Bloom e seus colegas (Bloom et al., 1956) com o intuito de facilitar a troca de questões em testes entre professores de várias universidades, ao garantir que cada questão avaliaria os mesmos objetivos de aprendizagem.

A Taxonomia de Bloom refere-se a uma classificação dos diferentes objetivos que os educadores estabelecem para que seus estudantes os alcancem. Ela divide os objetivos educacionais em três domínios distintos: cognitivo, afetivo e psicomotor. O domínio cognitivo engloba o conhecimento e pensamento, o domínio afetivo o sentimento e as relações interpessoais, e o domínio psicomotor a manipulação física de objetos. O desenvolvimento em cada domínio é medido através de níveis ou categorias organizadas em hierarquias lineares, de forma que para alcançar uma categoria, o aluno deve dominar as categorias anteriores. Portanto, dentro dos domínios, a aprendizagem nos níveis superiores é dependente da obtenção do conhecimento prévio e habilidades em níveis inferiores. O objetivo da taxonomia de Bloom é de motivar os educadores a se concentrarem em todos os três domínios, criando uma forma mais holística de educação.

Segundo Filatro (2009), a Taxonomia de objetivos educacionais desenvolvida por Bloom e seus colegas influenciou significativamente a sistemática de planejamento pedagógico, na medida em que criou uma linguagem comum e padronizada para identificar e classificar as atividades educacionais.

A seguir estão descritos cada um dos domínios da taxonomia de Bloom e suas respectivas categorias.

3.2.1 Domínio Cognitivo

Habilidades no domínio cognitivo tratam do conhecimento, compreensão e pensamento crítico sobre um tema específico. Segundo Scott (2003), a educação tradicional tende a enfatizar as habilidades neste domínio, principalmente as primeiras categorias.

O domínio cognitivo apresenta seis níveis (Figura 3.1), subindo em complexidade ao mover-se através das categorias de ordem mais baixas às mais altas. O desenvolvimento de uma categoria não implica no deslocamento imediato para um nível superior. No entanto, segundo Bloom, um estudante precisa dominar as categorias-requisito de uma categoria para poder então alcançá-la.

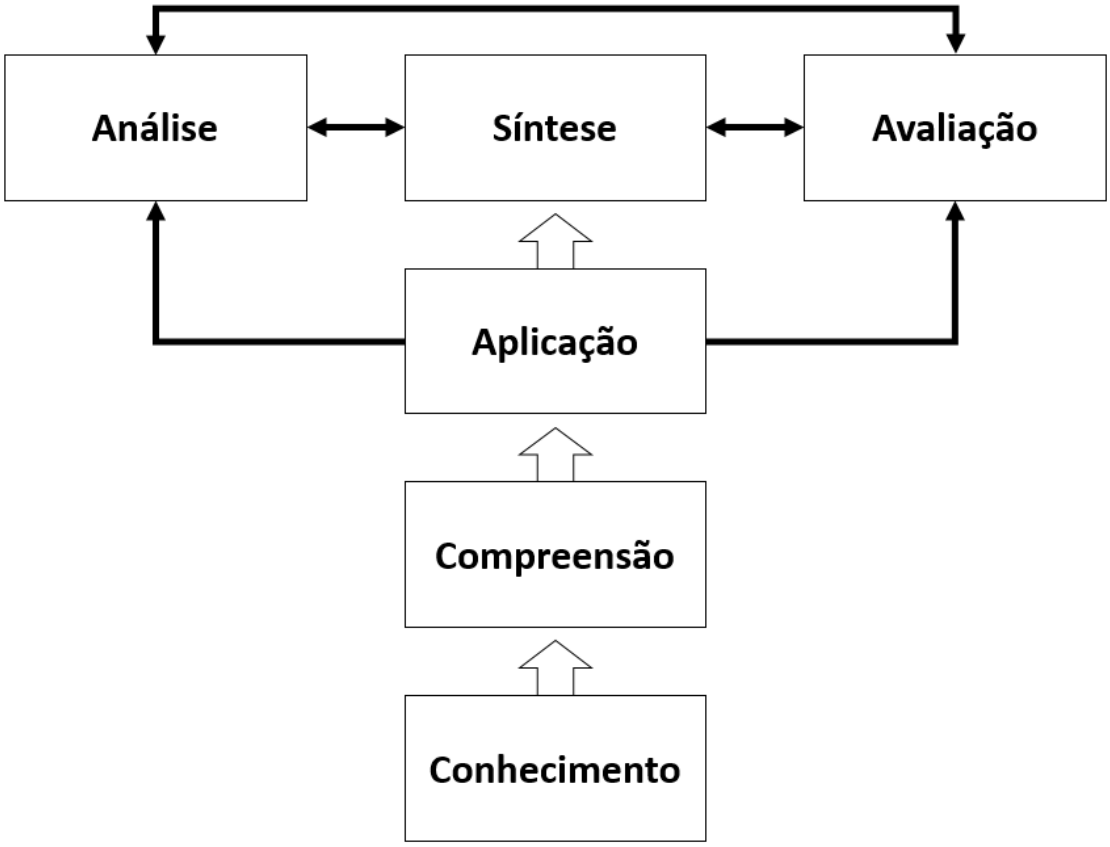


Figura 3.1: Relacionamento entre as categorias do Domínio Cognitivo na Taxonomia de Bloom (Hall e Johnson, 1994)

Conhecimento

Essa é a categoria que menos exige do aluno, pois basta que ele memorize algumas informações, mesmo que não consiga compreender completamente o seu significado ou aplicá-la para resolver um problema. Objetivos desta categoria são úteis nos momentos iniciais da aprendizagem, quando a pessoa deve, primeiramente, armazenar informações que serão recuperadas posteriormente para desenvolver processos mais complexos. O aluno apresenta memória de tópicos aprendidos ao recordar fatos, termos, conceitos básicos e respostas.

Alguns verbos que refletem este nível de conhecimento e podem ser utilizados nas avaliações são: citar, nomear, listar, reproduzir, repetir e apresentar.

Exemplos de questões na computação:

- Identificar elementos específicos em um trecho de código.
- Reconhecer a implementação de um determinado conceito.

- Reconhecer a descrição mais apropriada para um determinado conceito.
- Lembrar-se de um conceito, processo, algoritmo, etc.
- Listar operadores de acordo com a ordem de precedência.
- Definir o propósito de um método construtor.
- Descrever um determinado padrão de projeto.
- Citar os nomes dos tipos de loops em uma linguagem de programação.
- Listar N métodos que executem operações de entrada e saída de dados.

Compreensão

Demonstrar entendimento de fatos e ideias, através da organização, comparação, tradução, interpretação, dando descrições, e indicando as principais ideias. Nos objetivos desta categoria não basta que o aluno memorize informações; é necessário que ele compreenda o significado e a importância das mesmas.

Alguns verbos que refletem este nível de conhecimento e podem ser utilizados nas avaliações são: explicar, relacionar, traduzir, transformar, descrever e associar.

Exemplos de questões na computação:

- Escrever em pseudocódigo, fluxograma ou em linguagem natural um programa que calcule uma fórmula bem conhecida.
- Completar partes faltantes de um programa utilizando fragmentos de código.
- Explicar com palavras o comportamento de um trecho de código.
- Predizer valores de variáveis depois da execução de um trecho de código.
- Traduzir um algoritmo de uma forma de representação para outra.
- Explicar um conceito, algoritmo ou padrão de projeto.
- Apresentar exemplos de um conceito, algoritmo ou padrão de projeto.

Aplicação

Uso do conhecimento adquirido para resolver problemas em situações novas, aplicando esse conhecimento para lidar com fatos, técnicas e regras de uma

maneira diferente. Nesta categoria é preciso, além de compreender determinados conceitos, generalizá-los a fim de aplicá-los a novas situações.

Alguns verbos que refletem este nível de conhecimento e podem ser utilizados nas avaliações são: aplicar, prever, demonstrar, preparar, resolver e modelar.

Exemplos de questões na computação:

- Implementar um programa utilizando como exemplo um código que resolva um problema semelhante.
- Implementar ordenação de vetores não numéricos com alunos que já tenham ordenado vetores numéricos.
- Executar mentalmente expressões seguindo as regras de precedência.
- Resolver um problema familiar, mas com dados ou ferramentas não familiares.
- Modificar o código de um laço do tipo *for* para um do tipo *while*.

Análise

Examinar e quebrar informações em partes, identificando motivos ou causas, além de ser capaz de fazer inferências e encontrar evidências para apoiar generalizações. Esta categoria enfoca o desdobramento de um conceito em suas partes constitutivas e a percepção de suas inter-relações e de seus modos de organização. Por isto que, para analisar determinado conceito, é preciso previamente compreendê-lo como um todo e ser capaz de generalizar a sua aplicação. Esta etapa é fundamental para que se chegue à última categoria da taxonomia, que é a capacidade de avaliação ou julgamento.

Alguns verbos que refletem este nível de conhecimento e podem ser utilizados nas avaliações são: examinar, determinar, categorizar, diagnosticar, dividir, classificar e organizar.

Exemplos de questões na computação:

- Dividir uma tarefa de programação em suas partes componentes.
- Organizar as partes componentes para atingir um objetivo geral.
- Identificar componentes críticos para o desenvolvimento.
- Identificar componentes ou requisitos não importantes.

- Diferenciar um método construtor dos demais métodos de uma classe.

Síntese

Aqui o aluno é capaz de construir uma estrutura ou padrão de diversos elementos. Também se refere o ato de colocar as peças em conjunto para formar um todo (Omari, 2006). Compilar informações juntas de uma maneira diferente pela combinação de elementos em um novo padrão ou propor soluções alternativas. Esta categoria possibilita o conhecimento sobre como unir partes já conhecidas e estudadas em um todo, com características não percebidas anteriormente. Esta categoria é responsável por dotar as pessoas com um comportamento criador.

Alguns verbos que refletem este nível de conhecimento e podem ser utilizados nas avaliações são: combinar, conceber, integrar, produzir, inventar, projetar, reorganizar, formular.

Exemplos de questões na computação:

- Propor algoritmo, processo ou estratégia alternativa para um problema.
- Hipotetizar que uma nova combinação de algoritmos resolverá o problema
- Construir um programa utilizando algoritmos inventados.
- Aplicar algoritmos conhecidos em uma combinação não familiar para o aluno.

Avaliação

Apresentar e defender opiniões, fazendo julgamentos sobre a informação, a validade de ideias ou a qualidade do trabalho com base em um conjunto de critérios. É a capacidade de julgamento sobre valores, ideias, materiais, métodos, enfim, tudo o que pode ser apresentado a uma pessoa com um determinado propósito. Foi disposta no último estágio da taxonomia, porque é preciso conhecer, compreender, aplicar, analisar e sintetizar para ter a capacidade de avaliar.

Alguns verbos que refletem este nível de conhecimento e podem ser utilizados nas avaliações são: concluir, avaliar, criticar, justificar, deduzir, recomendar e comparar.

Exemplos de questões na computação:

- Determinar se um código satisfaz os requisitos definindo uma estratégia de teste apropriada.
- Criticar a qualidade de um código baseando-se em boas práticas de programação ou critérios de eficiência do código.
- Avaliar qual de dois algoritmos que resolvem a mesma tarefa é mais adequado.
- Encontrar um erro de lógica em um trecho de código dado.

3.2.2 Domínio Afetivo

As habilidades no domínio afetivo descrevem a maneira como as pessoas reagem emocionalmente e sua capacidade de sentir a dor ou alegria de outros seres vivos. Objetivos afetivos normalmente focam na conscientização e crescimento em atitudes, emoções e sentimentos.

Há cinco níveis no domínio afetivo que se deslocam através dos processos de ordem inferiores aos superiores:

Recebendo

Refere-se à experiência de escuta; disponibilidade para ouvir e estar ciente. Essa categoria é expressa através da escuta e prestando atenção a outros com uma atitude de respeito e bom comportamento.

Palavras-chave: reconhece, pede, atencioso, cortês, obediente, segue, dá, escuta, entende.

Exemplos: Ouça os outros com respeito. Ouvir e lembrar o nome das pessoas recém-introduzidas.

Respondendo

Refere-se à experiência de estar ativamente envolvido em atividades, com o conteúdo, colegas e professores nos processos de aprendizagem. Inclui atenção e reação às interações associadas à aprendizagem. Essa categoria é alcançada ao completar tarefas em locais de aprendizagem e demonstrar vontade de

responder em tempo real ou de forma assíncrona, expondo clara motivação para alcançar uma experiência de aprendizagem satisfatória.

Palavras-chave: respostas, assistências, auxílios, cumpre, conforma, discute, cumprimenta, ajuda, etiqueta, executa, apresenta, diz.

Exemplos: Participa de discussões em classe. Dá uma apresentação. Questiona novas ideias, conceitos, modelos, etc., a fim de entendê-los completamente. Conhece as regras de segurança e as pratica.

Valorizando

Refere-se ao valor que o aluno atribui a um fenômeno ou comportamento, incluindo os níveis complexos de comprometimento de aplicar essas proposições de valor. Reflete-se em um conjunto de valores internalizados que são demonstrados através de crenças e sensibilidade para diferenças individuais e culturais. Alunos que alcançam essa categoria utilizam os valores para estabelecer o compromisso de prosseguir e compreender mudanças sociais.

Palavras-chave: aprecia, ame, dá valor, demonstra, inicia, convida, junta-se, justifica, propõe, respeita, compartilha.

Exemplos: Demonstra crença no processo democrático. É sensível a diferenças individuais e culturais (valor de diversidade). Mostra a capacidade de resolver problemas. Propõe um plano de melhoria social e segue com compromisso.

Organizando e conceitualizando valores

Refere-se à capacidade de conceituar o refinamento ou desenvolvimento de valores como parte da identidade individual. Alunos que alcançam essa categoria são capazes de organizar valores como prioridades através de diferenciação e avaliação em caso de valores conflitantes. Essa categoria leva a um processo contínuo de consideração, aplicação, sintetização, e incorporação de valores para a identidade individual.

Palavras-chave: compara, refere, sintetiza.

Exemplos: Reconhece a necessidade de um equilíbrio entre a liberdade e um comportamento responsável. Explica o papel do planejamento sistemático na resolução de problemas. Aceita padrões éticos profissionais. Cria um plano de

vida em harmonia com as habilidades, interesses e crenças. Prioriza tempo de forma eficaz para atender às necessidades da organização, família e eu.

Internalizando valores

Refere-se ao desenvolvimento de um sistema de orientação interna de valores que é consistente, penetrante, previsível e característico do aluno. Esse sistema de valores incorpora o desenvolvimento durante toda a vida e serve como plataforma sobre a qual desenvolvimento contínuo e refinamento pode ser posicionado reposicionado ao longo da vida.

Palavras-chave: age, discrimina, influencia, modifica, realiza, qualifica, pergunta, revisa, serve, soluciona, verifica.

Exemplos: Mostra a auto-suficiência ao trabalhar de forma independente. Coopera em atividades em grupo (demonstra trabalho em equipe). Usa uma abordagem objetiva na resolução de problemas. Exibe um compromisso profissional para a prática da ética em uma base diária. Revisa decisões e mudanças de comportamento à luz de novas evidências. Valoriza as pessoas pelo que elas são, e não pela sua aparência.

3.2.3 Domínio Psicomotor

Habilidades no domínio psicomotor descrevem a habilidade de manipular fisicamente uma ferramenta ou instrumento como uma mão ou um martelo. Objetivos psicomotores geralmente se concentram em mudança e/ou desenvolvimento de comportamento e/ou habilidades.

Bloom e seus colegas nunca criaram subcategorias de competências no domínio psicomotor, mas desde então outros educadores criaram suas próprias taxonomias psicomotoras. Simpson (1972) propôs os seguintes níveis: percepção, definição, resposta guiada, mecanismo, resposta ostensiva complexa, adaptação e originação.

Percepção

A habilidade de usar sinais sensoriais para orientar a atividade motora. Isso varia de estimulação sensorial, através da seleção de sinalização, a tradução.

Palavras-chave: escolher, descrever, detectar, diferenciar, distinguir, identificar, isolar, referir, selecionar.

Exemplos: Detecção de sinais de comunicação não-verbais. Estimar onde a bola vai cair depois de ser lançada e, em seguida, mover-se para o local correto para pegar a bola. Ajustar o calor do fogão à temperatura correta pelo cheiro e sabor dos alimentos. Ajustar a altura das forquilhas de uma empilhadeira comparando onde as forquilhas estão em relação ao estrado.

Definição

Prontidão para agir. Inclui conjuntos mentais, físicos e emocionais. Estes três conjuntos são disposições que predeterminam a resposta de uma pessoa a situações diferentes (às vezes chamado de mindset).

Palavras-chave: começar, mostrar, explicar, movimentar, prover, reagir, voluntariar.

Exemplos: Conhecer e agir de acordo com uma sequência de etapas de um processo de fabricação. Reconhecer suas habilidades e limitações. Mostrar desejo de aprender um novo processo (motivação). NOTA: Esta subdivisão do Domínio Psicomotor está intimamente relacionada com a subdivisão "Respondendo aos fenômenos" do domínio afetivo.

Resposta guiada

Os estágios iniciais no aprendizado de uma habilidade complexa que inclui imitação e tentativa e erro. Adequação de desempenho é alcançado através da prática.

Palavras-chave: copiar, traçar, reagir, reproduzir, responder.

Exemplos: Executar uma equação matemática como demonstrado. Seguir instruções para a construção de um modelo. Responder a sinais do instrutor ao aprender a operar uma empilhadeira.

Mecanismo

Este é o estágio intermediário no aprendizado de uma habilidade complexa. Respostas aprendidas se tornaram habituais e os movimentos podem ser realizados com alguma confiança e proficiência.

Palavras-chave: montar, calibrar, construir, desmontar, mostrar, fixar, corrigir, manipular, medir, consertar, misturar, organizar, esboçar.

Exemplos: Usar um computador pessoal. Reparar uma torneira vazando. Dirigir um carro.

Resposta complexa ostensiva

O desempenho habilidoso de atos motores que envolvem padrões de movimentos complexos. A proficiência é indicada por um desempenho rápido, preciso e altamente coordenado, exigindo um mínimo de energia. Esta categoria inclui a realização sem hesitação e desempenho automático. Por exemplo, os jogadores muitas vezes proferem sons de satisfação ou palavrões assim que batem em uma bola de tênis ou chutam uma bola de futebol, porque eles sabem, a partir da sensação do ato, que resultado será produzido.

Palavras-chave: montar, construir, calibrar, construir, desmontar, mostrar, fixar, corrigir, manipular, medir, consertar, organizar, esboçar. NOTA: As palavras-chave são as mesmas que Mecanismo, mas têm advérbios e adjetivos que indicam que o desempenho é mais rápido, melhor, mais preciso, etc.

Exemplos: Manobrar um carro em uma vaga de estacionamento paralelo apertada. Operar um computador com rapidez e precisão. Mostrar competência ao tocar piano.

Adaptação

Habilidades estão bem desenvolvidas e o indivíduo pode modificar os padrões de movimento para se adequar às exigências especiais. Exemplos: Responder de forma eficaz a experiências inesperadas. Modificar instruções para atender às necessidades dos alunos.

Palavras-chave: se adaptar, alterar, mudar, reorganizar, reordenar, revisar, variar.

Executar uma tarefa com uma máquina que não estava inicialmente prevista para fazer (máquina não está danificada e não há perigo na execução da nova tarefa).

Originação

A criação de novos padrões de movimento para atender a uma determinada situação ou problema específico. Os resultados de aprendizagem enfatizam a criatividade com base em habilidades altamente desenvolvidos.

Palavras-chave: organizar, construir, combinar, compor, criar projetos, iniciar, fazer, originar.

Exemplos: Construir uma nova teoria. Desenvolver uma nova e abrangente programação de treinamento. Criar uma nova rotina de ginástica.

3.3 A Taxonomia Revisada

Uma recente reavaliação da taxonomia de Bloom por Anderson e seus colegas (2001) sugere que os dois ou três primeiros níveis da hierarquia podem ser planos. Os autores também propuseram que a taxonomia deve ser bidimensional, com as (ligeiramente reconfiguradas) categorias originais de *Lembrar*, *Entender*, *Aplicar*, *Analisar*, *Avaliar* e *Criar* formando a dimensão do processo cognitivo e factual, conceitual, procedimental e metacognitivo formando a dimensão do conhecimento.

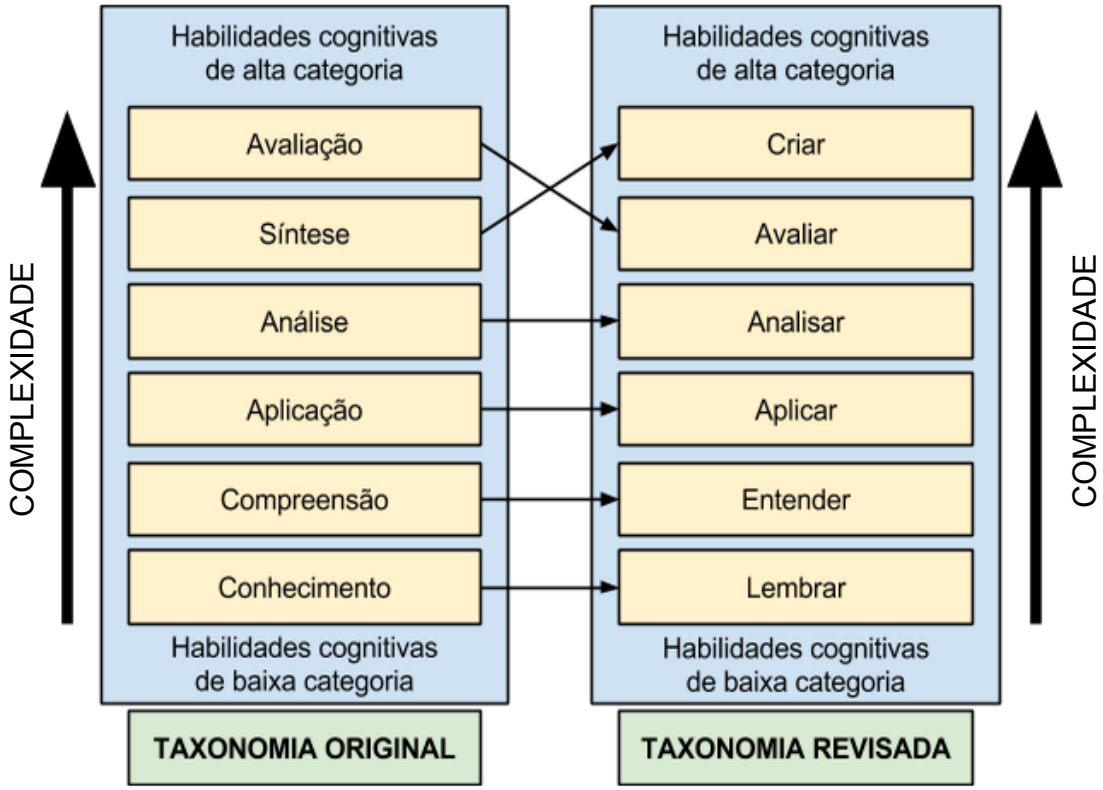


Figura 3.2: Revisão das categorias da Taxonomia e sua hierarquia

Krathwohl (1964) afirma que na Taxonomia revisada existem duas dimensões: a dimensão do Conhecimento, que engloba as subcategorias da categoria Conhecimento na taxonomia original, e a dimensão dos Processos Cognitivos, que abrange as seis categorias da Taxonomia original, porém renomeadas, em alguns casos apenas para suas formas verbais. A categoria Conhecimento tornou-se *Lembrar*, Compreensão tornou-se *Entender*, Síntese tornou-se *Criar* (e foi promovida para a categoria mais alta da hierarquia), Aplicação, Análise e Avaliação tornaram-se respectivamente *Aplicar*, *Analisar* e *Avaliar* (Figura 3.2). A estrutura de evolução em níveis da Taxonomia original, no entanto, foi mantida na sua versão revisada, embora tendo sofrido as modificações mencionadas acima. Outra medida tomada foi a associação de verbos às categorias, de forma a facilitar seu uso, conforme apresentado no Quadro 3.1.

Quadro 3.1: Categorias da Taxonomia revisada e seus verbos associados

Categoria	Processo cognitivo
Lembrar	Reconhecer, recordar
Entender	Interpretar, exemplificar, classificar, sumarizar, inferir, comparar, explicar
Aplicar	Executar, implementar
Analisar	Diferenciar, organizar, atribuir
Avaliar	Checar, criticar
Criar	Generalizar, planejar, produzir

Segundo Fuller e seus colegas (2006), essas taxonomias não definem uma sequência de instruções, mas sim níveis de desempenho que podem ser desejados para qualquer tipo de conteúdo. É esperado que um aluno atuante em um nível cognitivo superior seja capaz de operar nos níveis mais baixos na hierarquia. Isso poderia ser interpretado como o reconhecimento de um processo de aprendizagem sequencial. No entanto, a Taxonomia não descarta o uso de uma abordagem iterativa para aprender o conteúdo.

Os autores da Taxonomia revisada reconhecem que há uma possível sobreposição em termos de complexidade cognitiva entre as categorias de nível superior da hierarquia. No entanto, o ponto médio de cada uma das categorias de nível mais elevado é considerado como sendo mais complexo do que a categoria inferior (Svec 2005, Krathwohl et al., 1964). Por exemplo, o processo

cognitivo de *explicar* na categoria *Entender* pode exigir uma carga cognitiva superior a *executar* na categoria *Aplicar*, em alguns contextos. Embora Fuller (2007) reconheça que a Taxonomia original ainda é a mais usada, ela critica essa versão, dizendo que as categorias nem sempre são fácies de serem aplicadas, que há uma sobreposição significativa entre elas e que existem debates na academia sobre a ordem em que as categorias análise, síntese e avaliação são hierarquizadas.

Whalley (2006) diz que mesmo para professores de computação experientes, muitas das descrições dos níveis da taxonomia em certas categorias são difíceis de ser interpretados no contexto dos exercícios de programação. Esse problema também é apontado por Fuller (2007), embora afirme que a Taxonomia revisada solucionou diversas questões da versão original, tornando-a mais indicada para utilização como ferramenta norteadora em trabalhos científicos.

Podemos agora revisitar o problema colocado por Neto e Cechinel (2006) no Capítulo 1 deste trabalho para encontrar uma explicação à luz da Taxonomia de Bloom revisada:

> O problema ocorre porque o professor ministra uma aula que faz com que o aluno atinja apenas o segundo nível da taxonomia e espera que os alunos generalizem seu conhecimento a ponto de aplicá-los em outros problemas, o que é uma característica dos objetivos de aprendizagem da terceira categoria. Isto ocorre porque o professor desconhece até que ponto a aula ministrada é capaz de levar o aluno a raciocinar e, como consequência, não sabe se este é o ponto onde ele espera que o aluno chegue. Por isto, na próxima aula ele solicitará que o aluno resolva outro problema qualquer através de um programa que utilize parâmetros. Ao perceber as dificuldades dos alunos para aplicar o novo conceito, o professor pode pensar que isto se deve à dificuldade inerente à área. Certamente, ele não procurará em si próprio, ou em suas aulas, os motivos que levaram o aluno a não obter sucesso na resolução do novo exercício (Neto e Cechinel, 2006).

Podemos inferir que a Taxonomia de Bloom – e sua versão revisada – são extremamente úteis para o desenho das aulas de um curso e avaliação do aprendizado dos alunos. Usaremos neste trabalho, portanto, a versão revisada da Taxonomia como norteadora na construção e análise dos resultados obtidos após a aplicação.

Essa versão da Taxonomia foi escolhida para esta pesquisa por já ter sido mostrada como a mais indicada em cursos de programação (Horward et al., 1996; Johnson e Fuller, 2007; Scott, 2003; Jesus e Raabe, 2009).

Cabe lembrar que todos os três domínios da Taxonomia são importantes para a avaliação completa do crescimento de um indivíduo (Reeves, 2006). No entanto, devido à complexidade de utilizar todos os domínios e o foco no aprendizado cognitivo dos alunos de programação, a escolha de não contemplar os domínios afetivo e psicomotor foi tomada neste trabalho. A pesquisa procura se aprofundar no uso do domínio cognitivo da Taxonomia de Bloom revisada para um curso de programação, como ferramenta norteadora na construção do curso, do material didático, dos exercícios e provas, assim como para a comparação posterior dos resultados obtidos com outros trabalhos que seguiram esse tema.

4. METODOLOGIA

Este Capítulo apresenta a metodologia usada neste trabalho. Em primeiro lugar são apresentadas as hipóteses da pesquisa e suas questões principais. Em seguida, os eixos empregados no desenvolvimento da proposta são explorados. Na Seção 4.3 os aspectos das aplicações, como a sua construção, postura do professor e escolha de abordagem metodológica e os instrumentos de coleta de dados empregados são discutidos. Na Seção 4.4 é descrita a comparação das aplicações com o curso de Computação I, oferecido pela Universidade Federal do Rio de Janeiro (UFRJ). Finalmente, a Seção 4.5 apresenta as ferramentas utilizadas para a realização deste estudo.

4.1 Hipóteses/perguntas-chave

O objetivo da presente pesquisa é avaliar os possíveis ganhos do uso da robótica educativa no ensino de programação introdutória para alunos do ensino médio sem conhecimento prévio de programação.

Em vista disso, propõe-se a elaboração de um curso de programação, separado em duas intervenções, para alunos do ensino médio, sem conhecimento prévio de programação, com uso intenso da robótica educativa como facilitadora do ensino. Discutida em detalhes na Seção 4.3.3, a metodologia de estudo exploratório propõe a coleta de dados qualitativa e quantitativa, para que ao final do curso seja possível analisar os resultados e avaliar as ferramentas escolhidas. Embora ambos os tipos de coleta e análise sejam utilizados no estudo exploratório, o foco da pesquisa manteve-se na análise qualitativa das aplicações.

Assim, duas intervenções foram planejadas: no primeiro momento, uma oficina com o conteúdo condensado, servindo de piloto para avaliar o método de curso, material, tempo de aula e capacidade do professor, entre outros fatores. Após o término da Oficina piloto, mudanças estruturais seriam feitas na proposta levando em conta o feedback coletado durante a Oficina, de forma a proporcionar uma melhor experiência aos alunos e professor e um resultado final mais rico e assertivo.

Para construção das aplicações, avaliação e comparação dos resultados com outros trabalhos, a Taxonomia de Bloom revisada foi utilizada como ferramenta norteadora.

Objetivo geral:

Investigar a possibilidade de alunos, sem conhecimento prévio de programação, alcançarem os níveis superiores da Taxonomia de Bloom revisada, por meio de um curso de programação introdutória utilizando a robótica educativa como apoio.

Objetivos específicos:

- Construir e aplicar uma Oficina de programação introdutória com uso da robótica educacional;
- Construir e aplicar um Curso de programação introdutória com uso da robótica educacional, baseado nos dados coletados durante a Oficina;
- Investigar e analisar o desempenho dos alunos durante a Oficina e o Curso;
- Investigar a percepção dos alunos sobre o uso da robótica educacional e o ambiente de programação visual DuinoBlocks;
- Investigar a integração da robótica educacional com um curso de programação introdutória;
- Comparar os resultados das aplicações com a disciplina de Computação I, oferecida na UFRJ.

Questões de pesquisa:

- Alunos do ensino médio são capazes de aprender corretamente a ementa de um curso universitário de programação, utilizando robótica?
- Esses alunos são capazes de alcançar os níveis superiores da Taxonomia de Bloom revisada?
- A disponibilidade de utilizar materiais físicos influencia no desempenho dos alunos?

4.2 Abordagem

Tendo em vista uma melhoria no processo de ensino e aprendizagem de programação, este trabalho propõe a criação de um curso de programação com o uso de robótica a partir de três eixos norteadores: currículo acadêmico de ensino de programação utilizado pela Universidade Federal do Rio de Janeiro (UFRJ); programa do curso/oficina a partir da Taxonomia de Bloom revisada, e uma abordagem construcionista na preparação dos exercícios, desafios e tarefas propostas.

4.2.1 Eixo 1: Uso do currículo acadêmico da UFRJ

Os tópicos abordados nas aplicações baseiam-se no currículo acadêmico utilizado na disciplina de Computação I da UFRJ, que engloba as diretrizes elaboradas pelo CNE (Conselho Nacional de Educação)[15] em relação ao perfil dos egressos nos cursos (suas competências e habilidades necessárias para egresso), a ementa dos cursos e tópicos abordados, metodologia de ensino e carga horária necessária. A intenção de tal escolha foi a de garantir futuras comparações com o curso da UFRJ e de outras Universidades que seguem o currículo de referência da SBC (Sociedade Brasileira de Computação) para cursos de graduação em Bacharelado em ciência da Computação e Engenharia da Computação[16].

A ausência de pesquisas que utilizam um currículo oficial restringe a capacidade de comparação de resultados, e é uma das limitações apontadas no Capítulo 2 deste trabalho.

4.2.2 Eixo 2: Uso da Taxonomia de Bloom Revisada

Na concepção da proposta deste trabalho, viu-se necessário a adoção de uma métrica para avaliação dos ganhos cognitivos dos alunos participantes, bem como uma diretriz a ser adotada para a construção da apresentação dos conceitos e exercícios envolvidos. Tal métrica também poderia permitir, futuramente, a avaliação de provas e exercícios aplicados em outros cursos semelhantes, colaborando no processo de comparação de estratégias de ensino-aprendizagem de programação. Seguindo alguns autores que já abordaram esta questão (Howard et al., 1996; Scott, 2003; Johnson e Fuller, 2006), optou-se pelo uso da Taxonomia de Bloom revisada.

---

[15] O documento oficial com as diretrizes curriculares nacionais do curso de graduação em Ciência da Computação e similares pode ser acessado no portal do Ministério da Educação, através do endereço eletrônico http://portal.mec.gov.br/index.php?option=com_docman&task=doc_download&gid=11205&Itemid=

[16] O documento com o currículo de referência da SBC pode ser acessado através do endereço eletrônico http://www.sbc.org.br/index.php?option=com_jdownloads&Itemid=195&task=finish&cid=183&catid=36

Segundo Scott (2003), um dos problemas mais comuns encontrados por educadores é quando os alunos avançam em diferentes velocidades através de diferentes áreas do conhecimento, seja devido às suas habilidades ou motivação. Portanto o professor deve começar nas categorias inferiores da Taxonomia e trabalhar seu caminho através das categorias superiores. Se os exercícios e questões de prova tenderem a abordar todas as seis categorias da Taxonomia, eles poderão dar uma forma muito mais confiável de mensuração do aprendizado.

Considerando que cursos de Ciência da Computação se limitam (Scott, 2003), quase sempre, às três primeiras categorias da referida Taxonomia (*Lembrar*, *Entender* e *Aplicar*), as aplicações e os exercícios propostos foram construídos de forma a ir além desses níveis iniciais, perpassando os 6 níveis propostos. O objetivo aqui é o de poder desenvolver todos os níveis cognitivos da Taxonomia nos alunos, resultando em uma experiência de aprendizado mais completa.

4.2.3 Eixo 3: Abordagem Construcionista

Durante a concepção da Oficina houve sempre um esforço de construir o material didático em uma direção mais prática, ou seja, manter os alunos a todo o momento ocupados programando e construindo experimentos. A teoria é necessária para que o conhecimento sobre um determinado tópico seja aprofundado, porém, obedecendo a diretrizes da abordagem construcionista, a introdução de novos tópicos durante as aulas foi planejada através da apresentação de problemas práticos, cuja solução depende da exploração, tentativa e erro, até a descoberta do novo conhecimento.

A estratégia pedagógica adotada privilegia a liberdade dos estudantes para explorarem não só o ambiente de programação com a linguagem visual DuinoBlocks, como também o kit de robótica. Soluções corretas, distintas da esperada, devem ser incentivadas tanto quanto as respostas do “gabarito”. Segundo Heinzen (1994), essas respostas alternativas evidenciam o poder da criatividade. Conforme a sugestão do autor, a frustração pode facilitar a performance cognitiva pelo aumento da motivação para solução de problemas e estimulação intelectual. A frustração produz instigação para um número de

diferentes tipos de respostas, pois “existe frequentemente a tendência de um sujeito aumentar seus esforços após uma falha, de tentar uma variedade de respostas alternativas, apresentando algum tipo de emotividade” (p. 137).

Petty (2002) apresenta, em seu trabalho, algumas estratégias do uso da abordagem construcionista:

- "Ensinar através de questões", ou descobrimento guiado.
- Explicar as tarefas que exigem que alunos expressem sua compreensão para os outros, e para desenvolver esse entendimento antes de expressá-lo.
- Faça questões do tipo "diagnóstico", e use as respostas erradas para explorar e corrigir mal-entendidos - “questionamento socrático".
- Use tarefas e perguntas provocativas utilizando verbos localizados nas categorias mais altas da Taxonomia de Bloom, ao invés de questões simples, uma vez que as categorias superiores exigem maior raciocínio e processamento intelectual.
    - Análise: “por quê”.
    - Síntese: "como" você poderia...
    - Avaliação: Questões de julgamento.

  Essas questões de ordem superior exigem que o aluno construa suas próprias concepções do novo material. Não é recomendado trabalhar com o material até que ele esteja corretamente conceituado, portanto questões que exigem raciocínio forçam a conceitualização.

- Use estudos de casos que relacionam o tema com a vida real ou experiências anteriores.
- Use trabalhos em grupo para que os estudantes discutam o material, de modo que a deliberação entre membros do grupo e o ensino compartilhado ocorra.
- Use mapas mentais e resumos que apontem a relação entre as partes de um tema para o todo, uma vez que o aprendizado envolve a construção de padrões mentais. Enfatize também a relação do tópico vistos em um determinado momento a outros temas do curso.

- Ensine competências no contexto do assunto. Pense em você como um professor de habilidades que usa conteúdo para ensinar essas habilidades.
- Estimulação aumenta a taxa de aprendizagem. Portanto, use recursos multissensoriais ricos, atividades animadas e gere um senso de diversão sempre que possível.

Outro aspecto importante dessa proposta trata dos exercícios desenvolvidos em sala. Ao contrário de muitos cursos de introdução à programação, onde as atividades práticas não têm significado aparente para os alunos, na proposta da Oficina em tela, tais atividades foram, na medida do possível, ao encontro do interesse dos aprendizes. A partir de conversas em sala sobre outros interesses da turma, os exercícios propostos nas aulas seguintes procuravam relacionar a robótica com alguns desses tópicos.

4.3 Aspectos das aplicações

4.3.1 Construção

Duas aplicações foram realizadas neste trabalho. A primeira, de curta duração, foi realizada como um piloto para a segunda aplicação, de maior duração (Quadro 4.1). Com o intuito de evitar o desalinho da leitura, chamaremos, a partir de agora, a primeira aplicação de Oficina e a segunda aplicação de Curso.

Quadro 4.1: Principais diferenças entre as duas aplicações deste trabalho

Aspecto	Oficina	Curso
Duração	4 semanas	14 semanas
Início	08/05/2014	25/08/2014
Término	05/06/2014	18/11/2014

Local	Colégio Estadual José Leite Lopes	Colégio Pedro II
Número de alunos	10	12
Número de aulas	5	11
Duração das aulas	1 hora e 30 minutos	3 horas

O fato dos alunos progredirem de forma distinta, seja por questões de habilidade ou motivação, foram levados em conta na montagem dos exercícios propostos em aula. Eles foram pensados de forma a serem sempre iniciados explorando as categorias mais baixas da Taxonomia. A medida que os participantes dominam o conteúdo envolvido referente àquela categoria, são sugeridos pequenos incrementos no problema proposto, tornando-os mais complexos a ponto de abordar categorias superiores (Scott, 2003). Essa estratégia permite a observação da evolução de cada aluno em relação às categorias da Taxonomia, e consequentemente a adaptação do ritmo da aula para que todos consigam acompanha-la.

Em adição ao processo de criação como parte integrante da abordagem construcionista, a autorreflexão é o passo que liga todo o processo e traz significado para as outras partes. Ele dá aos alunos uma oportunidade para pensar e avaliar o seu próprio processo de pensamento e de seu papel na criação do projeto (Bers et al., 2002). Tais aspectos podem também levar a uma compreensão mais profunda do tema e das etapas e ferramentas envolvidas na criação do projeto relacionado.

Um exemplo de como o construcionismo foi utilizado no Curso pode ser visto no uso das protoboards para montagem de circuitos. A placa de ensaio sem solda (ou protoboard) é uma ferramenta essencial para uma rápida prototipagem de circuitos eletrônicos. Ela funciona conectando, internamente, os fios que são posicionados nos furos corretos (Figura X). Ao aprender o funcionamento da protoboard, os alunos constroem, por tentativa e erro, seu conhecimento. A

medida que esse conhecimento se torna enraizado, o aluno passa a conseguir utilizar de forma cada vez mais eficiente os materiais de aula – no caso da protoboard, os alunos descobrem novas formas de organizar seu circuito, utilizando ao máximo sua estrutura em matriz.

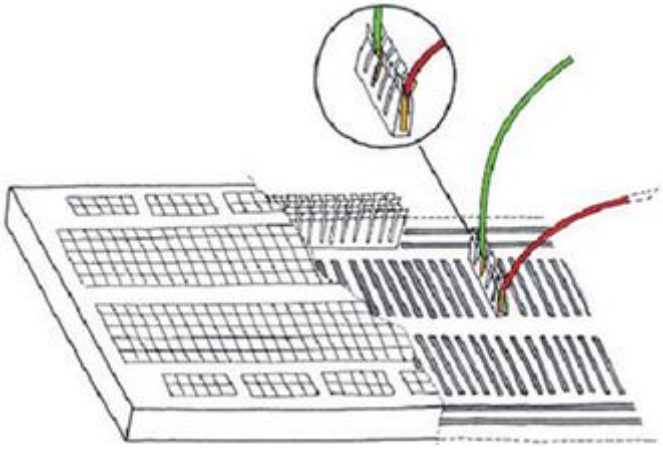


Figura 4.1: Corte representativo de uma parte da protoboard.

A vantagem pedagógica de se usar uma protoboard durante as atividades de aula é de que o aluno fica livre para experimentar diferentes ligações ao montar seu circuito. Ao utilizar uma protoboard, não é necessário qualquer trabalho de solda ou conexão entre os componentes, fazendo com que a única preocupação do aluno seja de conectar os fios de forma correta ao montar o circuito. Por essa mesma razão a protoboard também pode ser reutilizada inúmeras vezes, enquanto que placas normais onde os componentes são soldados tornam-se únicas uma vez montadas. O aluno tem, então, a oportunidade de experimentar com seu protótipo de circuito, montando e desmontando quantas vezes for necessário, até ser bem-sucedido em sua empreitada, construindo pouco a pouco seu entendimento no processo (Figura 4.1).

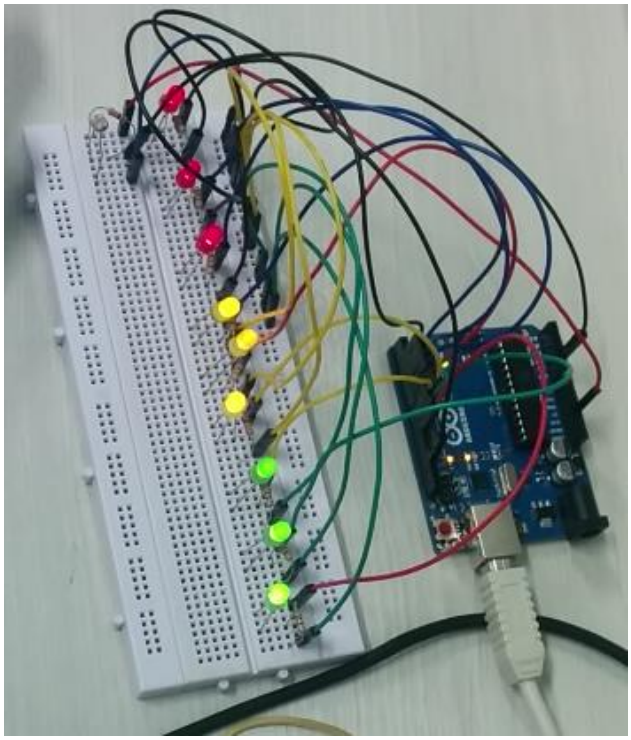


Figura 4.2: Exemplo de circuito montado em sala, utilizando uma protoboard

Ao longo da aplicação da Oficina diferentes dados foram coletados para a construção do Curso deste trabalho. Aspectos como duração e dinâmica de aula foram levados em conta, bem como a ordem de apresentação dos tópicos, escolha de exercícios e atividades em grupo.

Alguns pontos específicos receberam atenção especial, como a forma utilizada para explicar certos conceitos de programação e robótica. Por se tratar de assuntos que necessitam de abstração e compreensão raramente exigidas pelo sistema de ensino regular, em alguns momentos da Oficina percebeu-se que os alunos exibiam dificuldades em absorver tais assuntos.

As provas aplicadas no término da Oficina e do Curso foram construídas contendo questões relevantes ao tema - programação e robótica, ao mesmo tempo em que manteve a estrutura padrão dos exames do curso de Computação I da UFRJ. As provas da Oficina e do Curso estão disponíveis nos Apêndices B e C deste trabalho, respectivamente.

Um questionário online de opinião voltado aos alunos das duas aplicações foi desenvolvido com o objetivo de coletar impressões sobre as aulas ministradas, o conteúdo apresentado e outros aspectos importantes das aplicações.

Mann e Stewart (2002) afirmam:

> Um questionário online [...] tem a vantagem de parecer igual (dependendo do visualizador da internet usado) para todos os respondentes. O questionário pode ter uma aparência atrativa, utilizando formatação de texto, cores e gráficos. Ele também é de fácil preenchimento por parte dos respondentes, selecionando, geralmente, respostas de listas pré-definidas ou inserindo textos em caixas de mensagem e, em seguida, apenas pressionando o botão “Enviar” quando terminar. Os dados recebidos pelo pesquisador apresentam um formato previsível e consistente, podendo-se realizar a análise automática de dados, sem a edição que seria necessária com e-mail (Mann e Stewart, 2002, p. 70).

O questionário continha 20 questões no total: 3 perguntas sobre o aluno (nome, idade, turma) e 17 perguntas sobre as aulas, sendo duas delas incluídas a pedido da escola (sobre horários e ambiência). Para encorajar respostas verdadeiras, ficou estabelecido que a identificação dos alunos ao preenchê-lo seria opcional.

A Tabela 4.1 mostra a matriz de referência construída para o questionário de opinião. As perguntas sobre o Curso envolveram cinco dimensões, sendo 3 perguntas para as dimensões aprendizado de programação, aprendizado de robótica, as ferramentas utilizadas, o uso de um espaço não-formal de ensino para as aulas e 5 para a dimensão impressões sobre o Curso em geral, totalizando 17 perguntas. A análise das respostas dos alunos para este instrumento foi obtida através de uma planilha eletrônica online com o objetivo de verificar os índices percentuais obtidos em cada questão. Estes resultados são discutidos no Capítulo 5.

Tabela 4.1: Dimensões do questionário de opinião

Escopo	Dimensão	Perguntas objetivas	Perguntas dissertativas	Total de perguntas
Pessoal	Conhecimento pessoal	0	3	3
Curso	Aprendizado de programação	2	1	3
	Aprendizado de robótica	2	1	3
	Programação no ambiente DuinoBlocks vs ambiente Arduino	3	0	3
	Preferências sobre aprendizado em espaços formais vs não-formais	3	0	3
	Opiniões sobre o Curso em geral	4	1	5
	Total	14	6	20

Os questionários de opinião da Oficina e do Curso estão disponíveis no Apêndice J deste trabalho.

4.3.2 Postura do professor

Segundo Reeves (1998), o ensino de programação pode ser projetado para suportar diferentes papéis pedagógicos dos professores. Alguns cursos de programação são projetados para colocar os professores no papel de "facilitador". Outros programas são projetados para suportar o papel didático mais tradicional de um instrutor como "o professor".

Ainda segundo o autor:

> Os papéis didáticos dos professores estão bem estabelecidos. Um quarto de século atrás, Carroll (1968) nos disse que "de longe a maior quantidade de atividade docente em ambientes educacionais envolve dizer coisas para os alunos..." (p. 4). Análises mais recentes do ensino indicam que pouco mudou desde então (cf., Goodlad, 1984; Kidder, 1989; Perelman, 1992). Quando a exposição de conteúdo pelo professor for uma estratégia instrucional adequada, a tecnologia pode ser usada para apoiar, reforçar e estender as apresentações dos professores. (Reeves, 1998)

Diferentes autores (Chiaro e Leitão, 2005; Moran, 1994; Giordan, 1999; Reeves, 2006) têm discutido o papel do professor, que vem sendo redimensionado e cada vez mais ele se torna um supervisor ou facilitador. O Grupo de Cognição e Tecnologia de Vanderbilt (CTGV, 1992) descreve uma mudança no papel do professor "de provedor autoritário de conhecimento a um recurso que, por vezes, é consultado por alunos e em outras vezes pode tornar-se o estudante a quem os outros ensinam" (p. 73).

Segundo Santana (2013), há uma percepção crescente de que o professor precisa investir nas relações interativas, em prol da construção do conhecimento, desenvolvendo atividades que favoreçam as interações e a aprendizagem.

Para Reeves (2006), os instrutores acostumados a uma abordagem de ensino didático em que eles entregam a informação pré-embalada para os alunos, sob a forma de palestras e leituras, podem ter dificuldades com a necessidade de permitir que seus alunos lidem com as complexidades inevitáveis de tarefas autênticas ou aprendizagem de serviço. Bain (2004) descreve como os melhores professores entregam parte de seu “poder” como especialistas e tornam-se co-aprendizes com seus alunos, aprendendo ao mesmo tempo com eles.

A escolha de postura, portanto, foi a de um professor facilitador, que encoraja e orienta os alunos de forma a oferecer-lhes meios de concluírem suas tarefas e projetos, sem dar as respostas diretamente. Neste trabalho, ao assumir a postura de um professor facilitador, a ideia é permitir que a construção do conhecimento fosse feita pelo aluno, cabendo ao professor apenas servir de apoio durante o processo.

4.3.3 Abordagem qualitativa

Devido à sua natureza exploratória, este trabalho utilizará como instrumento de pesquisa o Estudo de Caso que, segundo Yin (2001), é um método que proporciona ao pesquisador a possibilidade de analisar as características significativas dos eventos da vida real. Podemos aplicar o método de Estudo de Caso para observar o comportamento de grupos, processos administrativos e organizacionais, entre outros, e é muito utilizado em pesquisas da área de psicologia, sociologia, administração e educação.

Segundo Bogdan e Bliken (1994), o Estudo de Caso "[...] consiste na observação detalhada de um contexto, ou indivíduo, de uma fonte de documentos ou acontecimento específico". Nesse método, o pesquisador começa pela análise de dados coletados ou com entrevistas a pessoas que possam ser objeto de estudo, servindo-lhe também de fonte de dados. Em seguida, as informações são organizadas e suas fontes, necessárias aos objetivos da pesquisa, avaliadas.

Dentro da metodologia do Estudo de Caso, existem vários instrumentos para a coleta de dados. Segundo Leffa (2006), podem ser utilizados questionários, entrevistas, gravações e testes, cujo objetivo é a investigação do comportamento de um indivíduo ou grupo, para ter-se acesso às informações relevantes à pesquisa.

Dentro dessa metodologia, podemos ainda analisar os dados de dois modos: quantitativa ou qualitativamente. Moreira e Caleffe (2008) distinguem essas maneiras da seguinte forma:

> [...] a pesquisa qualitativa explora as características dos indivíduos que não podem ser facilmente descritos de forma numérica. O dado é coletado por observação. Já a pesquisa quantitativa explora as características e situações em que dados numéricos são obtidos [...]. Ambas podem ser utilizadas no mesmo trabalho de pesquisa (Moreira e Caleffe, 2008).

Guribye e Wasson (2002) comentam que a observação é o meio mais indicado para entender o comportamento dos indivíduos participantes da pesquisa. A observação pode ser realizada a partir de entrevistas, conversas informais e diários do pesquisador e do participante.

Para esse trabalho, portanto, foram adotadas as metodologias de Estudo de Caso, com análise qualitativa dos dados coletados por meio de questionário de opinião, observação do comportamento individual e em grupo na forma de um diário de aula, entrevistas, resultados do projeto final e de um teste de conhecimentos ao final das aplicações. Gravações de áudio e vídeo não foram realizadas devido a questões de permissão.

4.4 Comparação com um curso universitário

Durante o levantamento de literatura de apoio do trabalho, uma lacuna foi descoberta nas pesquisas que almejam a melhoria do ensino de programação. Essa lacuna se dá em relação à estratégia da abordagem de categorias da Taxonomia, tanto na elaboração do conteúdo didático das aplicações como nos seus métodos de mensuração de ganhos cognitivos (Balogh, 2011; Goh e Aris, 2007; Berretta et al., 2011; Bini e Koscianki, 2009; Chiou, 2004; Val e Pastor 2012; Malec, 2001; Nourbakhsh, Ramirez e Sosa, 2013; Friedrich et al., 2012; Aureliano e Tedesco, 2012).

De acordo com os autores, os trabalhos analisados se limitam apenas às categorias inferiores da Taxonomia, com destaque à categoria *Aplicar*, terceiro nível na hierarquia. Isso se dá, provavelmente, devido ao caráter prático da programação (Scott, 2003). Uma barreira, porém, é inadvertidamente estabelecida, impedindo a avaliação de ganhos cognitivos nas categorias superiores.

As pesquisas analisadas também apontam a ausência de uma ferramenta adequada e padronizada para avaliação dos ganhos dos alunos durante a aplicação – cada trabalho realiza a análise dos resultados de uma forma diferente das outras.

Portanto, durante a criação das apresentações de tópicos de programação e robótica, das tarefas, das provas e dos projetos finais, um grande cuidado foi tomado para abordar todas as categorias da Taxonomia, seguindo sua ordem hierárquica, para garantir o aprendizado mais completo dos alunos. Para alcançar esse objetivo, a ementa da disciplina de Computação I do curso de Ciência da Computação, oferecido na UFRJ, foi completamente examinada.

Abaixo estão seus tópicos abordados, necessários para alcançar o domínio em todo o conteúdo visto (Quadro 4.2).

Quadro 4.2 Ementa da disciplina de Computação I do curso de Ciência da Computação – UFRJ

Tópicos	Subtópicos
Introdução ao ambiente de programação	
Funções	Declaração
	Parâmetros
	Valor de retorno
	Chamada de funções
Manipulação de dados na programação	Tipos de dados básicos
	Variáveis, atribuição, escopo de variáveis
	Tipos compostos
Estruturas de controle	Estrutura condicional (if / else)
	Laço com qualquer condição de parada (while)
	Laço com número pré-determinado de repetições (for)
Matrizes	
Programa principal (main)	
Entrada e saída simples	
Funções recursivas	

Podemos também considerar a seguinte lista de competências e habilidades, a partir do quadro de tópicos da ementa de um curso típico de Computação[17]. A disciplina foi reduzida para englobar apenas a disciplina de programação. O

[17] Curso de Ciência da Computação da Universidade Estadual do Sudoeste da Bahia. O curso, assim como o da UFRJ,

cumprimento dos itens da lista abaixo foi observado superficialmente durante as intervenções, uma vez que este não é o objetivo do trabalho.

Habilidades:

- Capacidade de liderar e ser liderado;
- Comunicação oral e escrita, com destaque para o uso correto da língua portuguesa;
- Desenvolvimento de soluções criativas e inovadoras;
- Aprendizagem e transmissão de conhecimentos;
- Conciliação entre teoria e prática.

Competências:

- Conhecimento de aspectos teóricos, científicos e tecnológicos da área de computação;
- Capacidade para pesquisar e viabilizar soluções criativas e inovadoras para várias áreas de conhecimento e aplicação;
- Capacidade de administrar grupos de pesquisa e recursos;
- Eficiência na operação de equipamentos computacionais e sistemas de software;
- Capacidade de iniciar, projetar, desenvolver, implementar, validar e gerenciar projetos computacionais, bem como promover pesquisas científicas e tecnológicas dentro e fora do ambiente acadêmico;
- Competência para analisar e documentar oportunidades, problemas e necessidades passíveis de solução via computação, e para empreender na concretização dessa solução;
- Compreensão da importância de valorizar o usuário no processo de interação com sistemas computacionais e competência na utilização de técnicas de interação homem e máquina neste processo;
- Aplicação eficiente dos princípios de gerenciamento, organização e busca de informações.

Uma análise da disciplina de Computação I da UFRJ, em relação aos seus conteúdos, aulas, exercícios e provas mostra que ela também se enquadra no

problema discutido por Scott (2003), anteriormente mencionado, conforme apresentado no Quadro 4.3. As provas aplicadas nos últimos três anos, disponíveis no Apêndice L deste trabalho, contemplaram apenas as três categorias inferiores da Taxonomia de Bloom revisada, deixando inexploradas as três categorias superiores. Embora cursos de computação exibam a tendência de progredir apenas até a categoria *Aplicar* (Scott, 2003), essa característica deixa de lado não só o desenvolvimento de metade da Taxonomia, como ignora justamente os níveis mais complexos em termos cognitivos (Anderson, 2001). Essas categorias superiores são responsáveis pela ampliação do pensamento holístico do estudante em relação aos tópicos apresentados como, por exemplo, o desenvolvimento de opiniões, avaliação crítica e construção de padrões. Sem elas, o aluno fica limitado às atividades cognitivas mais básicas.

Quadro 4.3 Avaliação das categorias da Taxonomia usadas em seis provas da disciplina Computação I do Curso de Ciência da Computação - UFRJ, nos últimos dois anos

Ano	Prova	Questão	Objetivo	Conhecimentos necessários	Taxonomia
2012	1	1	Escrever código	Função, laço de repetição, condicional	*Aplicar*
		2	Executar código	Operador aritmético, operador lógico	*Entender*
		3	Escrever código	Função, condicional, variável, operador aritmético	*Aplicar*
	2	1	Escrever código	Função, condicional, laço de repetição, operador aritmético, variável	*Aplicar*
		2	Escrever código	Laço de repetição, condicional, operador aritmético, operador relacional	*Aplicar*

		3	Escrever código	Função, laço de repetição, condicional, operador aritmético, operador relacional	*Aplicar*
	3	1	Escrever código	Função, laço de repetição, condicional	*Aplicar*
		2	Escrever código	Laço de repetição, condicional, variável, operador aritmético, operador lógico	*Aplicar*
		3	Escrever código	Função, laço de repetição, variável, operador aritmético, operador lógico, operador relacional	*Aplicar*
2013	1	1	Escrever código	Função, laço de repetição, condicional	*Aplicar*
		2a	Executar código	Operador aritmético, operador lógico, operador relacional	*Entender*
		2b	Executar código	Laço de repetição, condicional, operador lógico	*Entender*
		2c	Executar código	Função, variável, operador aritmético	*Entender*
		3	Escrever código	Função, laço de repetição, condicional, variável, operação aritmética	*Aplicar*
	2	1	Escrever código	Laço de repetição, condicional, operador relacional	*Aplicar*

		2	Escrever código	Laço de repetição, condicional, função, variável, operador aritmético	*Aplicar*
		3	Escrever código	Função, laço de repetição, condicional	*Aplicar*
	3	1	Escrever código	Função, laço de repetição, condicional	*Aplicar*
		2	Escrever código	Função, laço de repetição, variável	*Aplicar*
		3a	Executar código	Operador lógico, operador relacional, operador aritmético	*Entender*
		3b	Executar código	Operador lógico, operador relacional, operador aritmético	*Entender*

Um exemplo de questão de prova é analisado abaixo conforme as diretrizes da Taxonomia de Bloom revisada.

Questão 2: Respostas rápidas (1.0 ponto)

(2a) O que será impresso pelos programas a seguir?

a)

```
var1 = True
var2 = True
var3 = False
var4 = not not var1 and var2
var5 = var3 and var1 or (not var3 and not var4)
print var5
```

R. False

b)

```
resp = 1 * 2 % 3 + 4 / 5
print resp
```

R. 2

c)

```
z = float(5) / int(2.5)
print z
```

R. 2.5

Figura 4.3: Primeiro exemplo de questão de prova da UFRJ

Trata-se da questão 2 da prova 1 do ano de 2013. A questão solicita ao aluno as saídas de cada trecho de código apresentado. O aluno deve, portanto, executar o código mentalmente ou escrevendo em papel. Para tal, é necessário ter o conhecimento dos componentes utilizados na questão e como eles interagem entre si: variáveis, operadores aritméticos, lógicos e relacionais. Segundo Fuller et al. (2006), ao predizer valores de variáveis depois da execução de um trecho de código, um aluno utiliza a segunda categoria da Taxonomia, *Entender*. Não podemos afirmar que a categoria *Aplicar* é alcançada ainda, conforme sua definição por Anderson (2001):

> *Aplicar* é o uso do conhecimento adquirido para resolver problemas em situações novas, aplicando esse conhecimento para lidar com fatos, técnicas e regras de uma maneira diferente. Nesta categoria é preciso, além de compreender determinados conceitos, generalizá-los a fim de aplicá-los a novas situações. (Anderson et al., 2001)

Cabe lembrar que a dificuldade de realização de uma questão não está diretamente associada à complexidade na Taxonomia (Tan e Othman, 2013). Um exercício que aborda apenas a primeira categoria, *Lembrar*, pode ser

construído de forma a ser mais difícil que um que aborda uma superior, como por exemplo *Analisar*.

Segundo Souza (2006), complexidade descreve o processo de pensamento que o cérebro usa para lidar com a informação. Na revisão da Taxonomia de Bloom, isso pode ser descrito por qualquer uma das seis palavras que representam os seis níveis. Por exemplo, a pergunta: "Qual é a capital do Maranhão?" está no nível *Lembrar*, enquanto a pergunta: "Diga-me com suas próprias palavras o que se entende por uma capital de estado" está no nível *Entender*. A segunda questão é mais complexa do que a primeira, porque é em um nível superior na Taxonomia.

Dificuldade, por outro lado, refere-se à quantidade de esforço que o aluno tem de despender dentro de um nível de complexidade para realizar um objetivo de aprendizagem. É possível para uma atividade de aprendizagem se tornar cada vez mais difícil sem se tornar mais complexa. Por exemplo, a tarefa "Nomeie os estados do país" está no nível *Lembrar*-se de complexidade, pois envolve recordação simples (memória semântica) para a maioria dos estudantes. Da mesma forma, a tarefa "Nomeie os estados e suas capitais" também está no nível *Lembrar*, mas é mais difícil do que a questão prévia porque envolve maior esforço para recordar a informação adicional. Da mesma forma, a tarefa "Nomeie os estados e suas capitais na ordem de sua criação" ainda está no nível *Lembrar*, mas é consideravelmente mais difícil do que as duas primeiras. Ela exige reunir mais informações e, em seguida, sequenciá-las em ordem cronológica.

Um erro comum que vemos, portanto, é o de que os professores são mais suscetíveis a aumentar a dificuldade das questões, em vez de complexidade, quando se tenta aumentar o nível de raciocínio e pensamento do aluno.

Certos aspectos da disciplina de Computação I foram incorporados à proposta deste trabalho, como o conteúdo programático, a carga horária e a metodologia de avaliação de exercícios e testes. Alguns pontos da proposta, propositalmente, não estão alinhados com as características da disciplina oferecida na UFRJ, como a quantidade de alunos que fazem parte de cada turma, a disposição desses alunos em grupos e, principalmente, a presença da robótica educacional durante as aulas.

O objetivo desta comparação não é replicar um curso de computação universitário, e sim utilizá-lo como base para verificar se os alunos, com parâmetros ligeiramente modificados, são capazes de alcançar níveis superiores da Taxonomia de Bloom revisada com sucesso.

4.5 Ferramentas utilizadas

Podemos encontrar nas pesquisas apresentadas na literatura diferentes ferramentas e formas de utiliza-las ao aplicar um curso com a presença da robótica educativa. Entre as opções mais populares de hardware, existem o LEGO MindStorms, o Arduino e o Raspberry Pi. Este trabalho utiliza kits com a placa Arduino, disponibilizadas à equipe por meio de outros projetos do Grupo de Informática Aplicada à Educação (Ginape) da UFRJ. A escolha de uso dos kits Arduino deu-se devido ao fato de serem de baixo custo, podendo ser facilmente adquiridos futuramente por instituições de ensino públicas que desejam aplicar cursos de robótica educacional ou incorporar a robótica em outros cursos já oferecidos.

A escolha do ambiente de programação recaiu sobre o ambiente visual de programação DuinoBlocks, proposta por Alves (2013) como facilitadora no uso do Arduino (Figura 4.4).

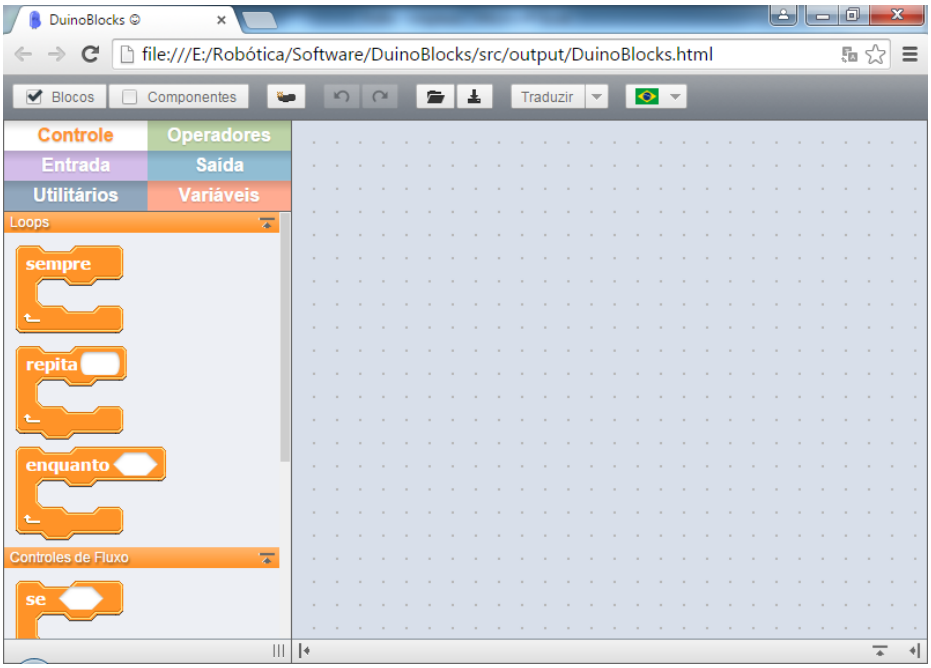


Figura 4.4: Tela principal do DuinoBlocks

O DuinoBlocks é um ambiente visual de programação desenvolvido para placas Arduino. Sua interface simples, baseada em blocos que podem se conectar, ignora a sintaxe da linguagem de programação, estimulando os alunos a construir seus algoritmos focando apenas em sua semântica. A codificação das categorias de comandos em cores também contribui com a curva de aprendizado dos alunos e sua performance durante a programação. Essas características resultam em um ambiente fácil de ser dominado, consequentemente facilitando também o aprendizado da montagem de algoritmos de programação.

O DuinoBlocks é dividido em duas interfaces distintas: Blocos e Componentes. Na interface de Blocos, o algoritmo do programa é montado, utilizando os blocos disponíveis na aba esquerda da tela. Blocos também podem ser criados pelo usuário, assim como funções, compostas de um ou mais blocos.

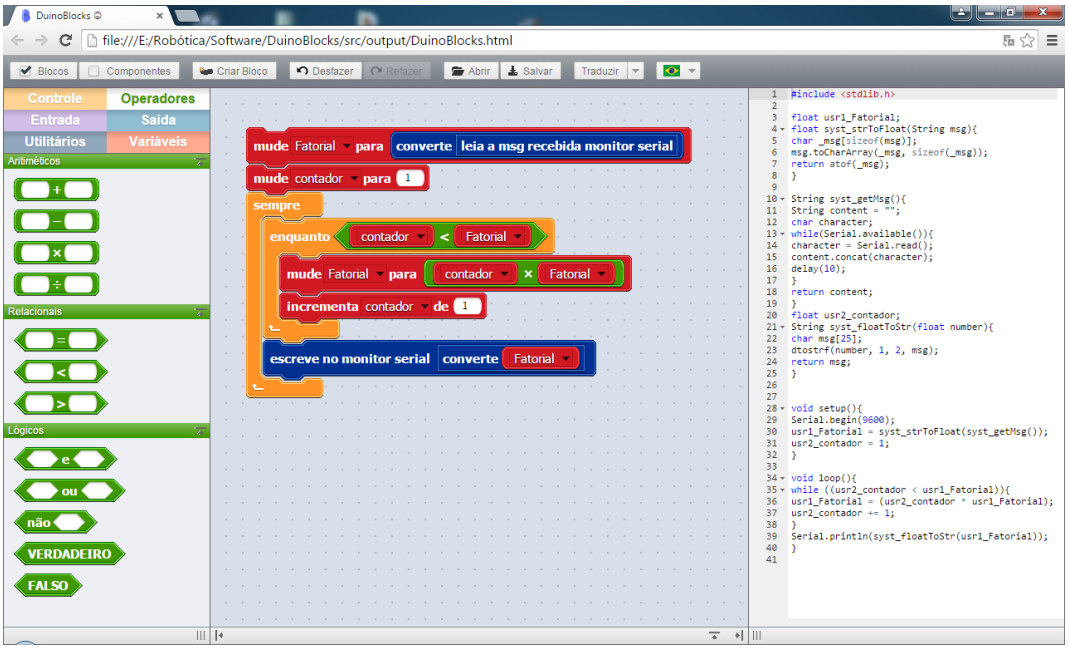


Figura 4.5: Tradução dos blocos montados na área de programação

Os blocos se encaixam como um quebra-cabeça, e é possível entender as condições de encaixe observando o formato de cada peça (Figura 4.5). Por exemplo, operadores relacionais são representados por um hexágono, e suas peças são da cor verde. Essa padronização ajuda os usuários a identificar mais

facilmente as peças disponíveis no ambiente e os seus possíveis locais de encaixe.

O código Arduino correspondente à estrutura de blocos montada pelo usuário é gerado em uma janela no lado direito do ambiente. O código gerado é uma tradução fiel do algoritmo no formato de blocos para o formato de textos[18]. Essa funcionalidade age como uma Pedra de Roseta[19], permitindo que o usuário possa não só verificar seu código, mas também aprender como construí-lo, utilizando a linguagem escrita.

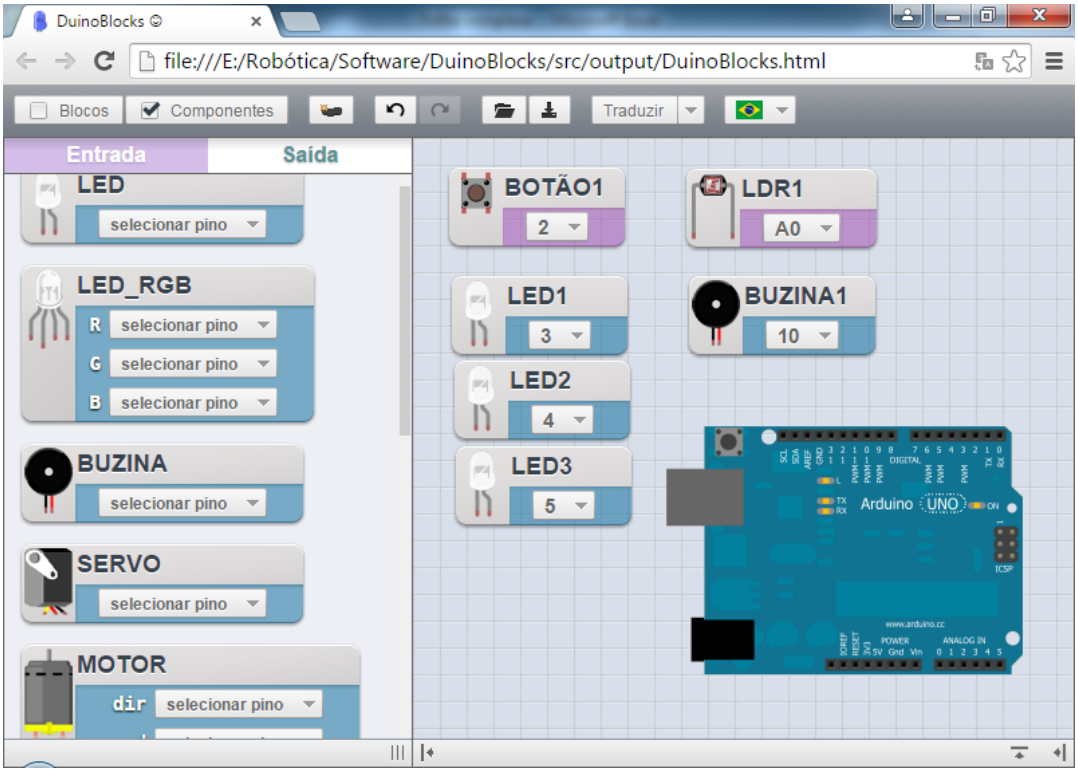


Figura 4.6: Tela de gerenciamento de componentes

A utilização da segunda interface do DuinoBlocks, Componentes (Figura 4.6), é opcional na construção de um programa Arduino. Nessa interface, são escolhidos os componentes virtuais desejados para simular os componentes

[18] Embora o contrário não seja possível na atual versão do ambiente.
[19] A Pedra de Roseta é um fragmento de uma estela de granodiorito do Egito Antigo, cujo texto foi crucial para a compreensão moderna dos hieróglifos egípcios (Ray, 2007).

reais utilizados na construção do circuito. A inclusão dos componentes é recomendada para administrar, no próprio ambiente, quais serão os pinos usados por cada um ao longo do programa. O pino pode ser alterado com facilidade, enquanto que no ambiente nativo do Arduino seria necessário alterar algumas linhas de código para alcançar o mesmo efeito. Não é possível, no entanto, criar novos Componentes.

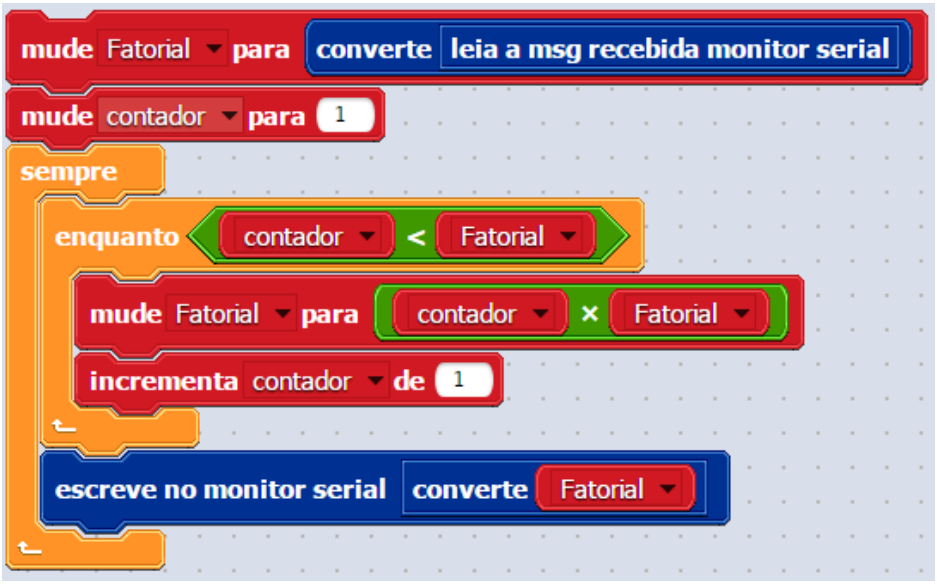


Figura 4.7: Algoritmo do cálculo do fatorial de um número

O algoritmo comentado na Seção 2.3.2 deste trabalho pode ser revisto aqui, reconstruído no ambiente DuinoBlocks. Os formatos e cores dos blocos do produto final tornam mais fácil o aprendizado e entendimento de um código que seria desafiador, caso estivesse apenas disponível um ambiente de programação textual. Com isso, espera-se anular os problemas levantados em relação ao aprendizado da sintaxe e da semântica de uma nova linguagem de programação simultaneamente.

5. ANÁLISE

Este Capítulo apresenta a análise e discussão dos dados coletados durante as duas intervenções, através dos diários do professor, provas e questionários de opinião, discutidos à luz do referencial teórico apresentado na revisão de literatura realizada no Capítulo 2 e da metodologia apresentada no Capítulo 4. As Seções deste Capítulo são divididos entre as ferramentas de coleta de dados, e uma síntese dos resultados é apresentada na Seção 5.7.

5.1 Estudos de caso

As duas intervenções deste trabalho – a Oficina e o Curso – foram aplicadas em duas escolas públicas do Rio de Janeiro no ano letivo de 2014. Seu propósito foi verificar se alunos sem conhecimento prévio de programação seriam capazes alcançar todas as categorias do domínio cognitivo da Taxonomia de Bloom revisada para esta área de conhecimento.

A melhor opção, portanto, seria utilizar alunos pré-universidade, pois estes ainda não foram apresentados à disciplina de programação. Nas escolas onde foram feitas as intervenções, os alunos do terceiro ano estavam ocupados com os estudos para o vestibular, que prestariam no final do ano, e o currículo do segundo ano já contempla essa matéria. Os alunos do primeiro ano, por sua vez, ainda não haviam aprendido a programar e nem estariam ocupados com outras agendas, como o vestibular, tornando-se então os participantes ideais da pesquisa.

Durante a fase de planejamento da proposta deste trabalho foi feita uma pesquisa cuidadosa do calendário acadêmico, considerando recessos, provas e outras questões que poderiam influenciar no cumprimento do cronograma. Entretanto, fatores como problemas técnicos nas salas de aula, alterações da agenda escolar e outros imprevistos resultaram em ajustes no referido cronograma, alterando a frequência das aulas. Além disso, o deslocamento da agenda fez com que as últimas aulas do Curso entrassem no período de provas finais de ano, afetando a constância de alguns alunos. Estes fatores podem ter alterado os resultados apresentados, referentes aos instrumentos de avaliação aplicados.

5.1.1 A Oficina

Durante o mês de maio e início de junho de 2014 a Oficina foi aplicada no Colégio Estadual José Leite Lopes, localizado na Zona Norte da cidade do Rio de Janeiro. O conteúdo da Oficina abordou os conceitos e construções lógicas mais relevantes na elaboração de programas computacionais simples, em acordo com a ementa do curso unificado de Computação I da Universidade Federal do Rio de Janeiro. A Oficina foi estruturada em um total de nove horas, divididas em

quatro aulas (Quadro 5.1). A turma experimental foi composta por dez alunos do 1º ano do Ensino Médio da referida escola, dentre eles seis meninos e quatro meninas.

A inscrição dos alunos da Oficina foi feita a partir do preenchimento de uma lista de interesse, exposta nas quatro turmas de 1º ano da escola, tendo cada uma em torno de 30 alunos. De um total de aproximadamente 120 alunos, 56 manifestaram interesse na Oficina. A primeira seleção foi feita a partir do conhecimento prévio de programação dos interessados – passavam para a próxima fase os alunos que não tinham qualquer conhecimento nesta área. A partir deste ponto, um sorteio foi realizado para definir os alunos da Oficina. As aulas, de uma hora e meia de duração (das 15:30 às 17:00), foram realizadas com reciprocidade semanal às quintas-feiras, em um horário compatível com a agenda dos alunos.

Quadro 5.1: Os tópicos abordados em cada aula da Oficina

Aula	Tópicos abordados
Aula 1	Apresentação da Oficina
	O que é robótica?
	Entendendo a eletricidade
	Seu primeiro circuito elétrico
	Principais componentes elétricos
	O microcontrolador Arduino
	O ambiente DuinoBlocks
	Sofisticando o seu primeiro circuito

Aula 2	Constantes e variáveis
	Operadores lógicos
	Operadores relacionais
	Operadores aritméticos
	Atribuição de valores
	Entrada e saída de dados
	Condicionais – if / else
	Laços de repetição – for/while
Aula 3	Programação de sensores
	Sensor de luminosidade
	Sensor de temperatura
Aula 4	Projeto final – construção e apresentações

Os alunos foram organizados em duplas, e à cada uma foram alocados um computador e um kit Arduino (Figura 5.1). A cada novo tópico, havia uma breve explicação apresentada pelo professor e o lançamento de um desafio. Cada dupla deveria trabalhar para resolvê-lo da forma que achasse melhor, com acompanhamento e orientação do professor. Dessa forma, cada aluno exerceu o comando de seu próprio aprendizado, construindo as suas soluções a partir do conhecimento já adquirido e tentativa e erro (Papert, 1991).

Figura 5.1: Aluno trabalhando com o Arduino e DuinoBlocks durante a Oficina

Ao final da Oficina, os alunos foram apresentados a uma prova moldada a partir de uma prova unificada de Computação I da UFRJ (prova modelo) - o mesmo curso cujo conteúdo foi utilizado para a construção da Oficina.

A prova modelo foi primeiramente analisada e cada questão classificada de acordo com as categorias da Taxonomia de Bloom revisada e as competências necessárias, conforme apresentado na Seção 4.4 desta pesquisa. A partir deste trabalho inicial foi então construída uma nova prova com questões que abordavam problemas envolvendo a robótica sem, no entanto, alterar as categorias e competências trabalhadas na prova da UFRJ.

Além da prova, também foi solicitado aos alunos o preenchimento de um questionário de opinião, envolvendo perguntas sobre o aprendizado de programação e robótica; as ferramentas utilizadas; o uso de um espaço não-formal de ensino para as aulas; e impressões sobre a Oficina em geral. A análise dos resultados da prova e do questionário de opinião serão discutidos mais adiante neste Capítulo.

As categorias da Taxonomia de Bloom revisada utilizadas ao longo das aulas da Oficina são apresentadas na Figura 5.2.

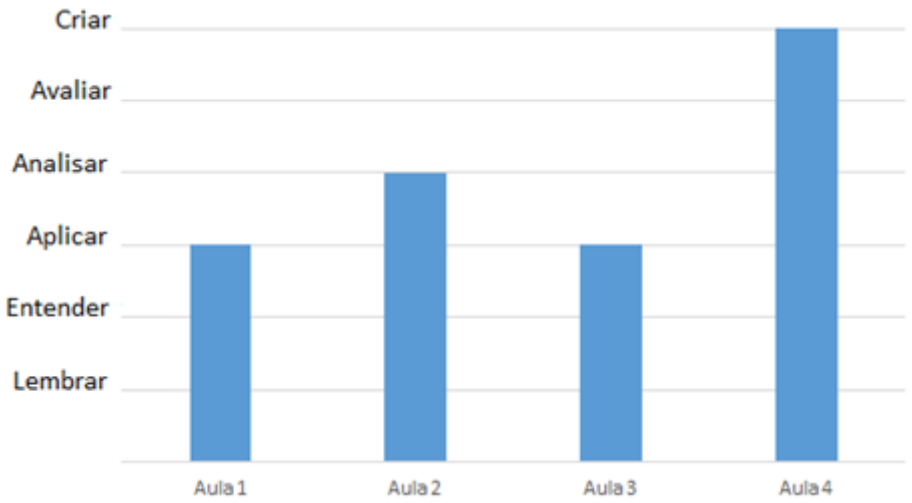


Figura 5.2: Categorias da Taxonomia de Bloom revisada utilizadas em cada aula da Oficina

5.1.2 O Curso

O Curso foi implantado no Colégio Pedro II, escola pública carioca localizada no bairro da Tijuca, com início no mês de agosto e término no mês de novembro de 2014. Embora o conteúdo do Curso tenha se mantido o mesmo do apresentado na Oficina, cada tópico pode ser mais bem explorado e debatido, uma vez que sua carga horária foi mais extensa, conforme mostrado no Quadro 5.2.

Quadro 5.2: Os tópicos abordados em cada aula do Curso

Aula	Tópicos abordados
Aula 1	Apresentação do Curso
	O que é a robótica?
	Exemplos - a robótica nas nossas vidas
	Entendendo a eletricidade
	A protoboard
	Seu primeiro circuito elétrico
Aula 2	O Microcontrolador Arduino

	DuinoBlocks
	Como programar o Arduino para piscar um led
	Sofisticando o seu primeiro circuito
	Como inserir e controlar múltiplos componentes
Aula 3	Desafio 1
Aula 4	Entrada e saída de dados
	Componentes no DuinoBlocks
	Constantes e variáveis
	Operadores lógicos
	Operadores aritméticos
	Operadores relacionais
	Atribuição de valores
	Condicionais – if / else
Aula 5	Funções
Aula 6	Sensores
	Estruturas de repetição
	Laços de repetição – for / while
Aula 7	Desafio 2
Aula 8	Um pouco mais de eletrônica
	Potenciômetro

	Display de sete segmentos
Aula 9	Comunicação serial
Aula 10	Projeto final – início
Aula 11	Projeto final – apresentações
	Teste de conhecimentos

Enquanto a duração de cada aula da Oficina foi de uma hora e meia (considerada curta pelos próprios alunos, segundo o questionário de opinião), o curso foi composto de 11 aulas de três horas de duração (das 13:00 às 16:00), com intervalo de quinze minutos na marca de uma hora e meia de aula. Esse aspecto do Curso foi alterado, em relação à Oficina, devido ao feedback dos alunos, coletado durante a primeira aplicação. As aulas aconteceram semanalmente, às terças-feiras, após o horário escolar.

As categorias da Taxonomia de Bloom revisada utilizadas ao longo das aulas do Curso podem ser analisadas conforme a Figura 5.3. É possível notar o uso de todas as categorias nas aulas de desafio e projeto final (aulas 3, 7, 10 e 11), onde os alunos criaram algoritmos e montaram circuitos baseados nos conhecimentos adquiridos.

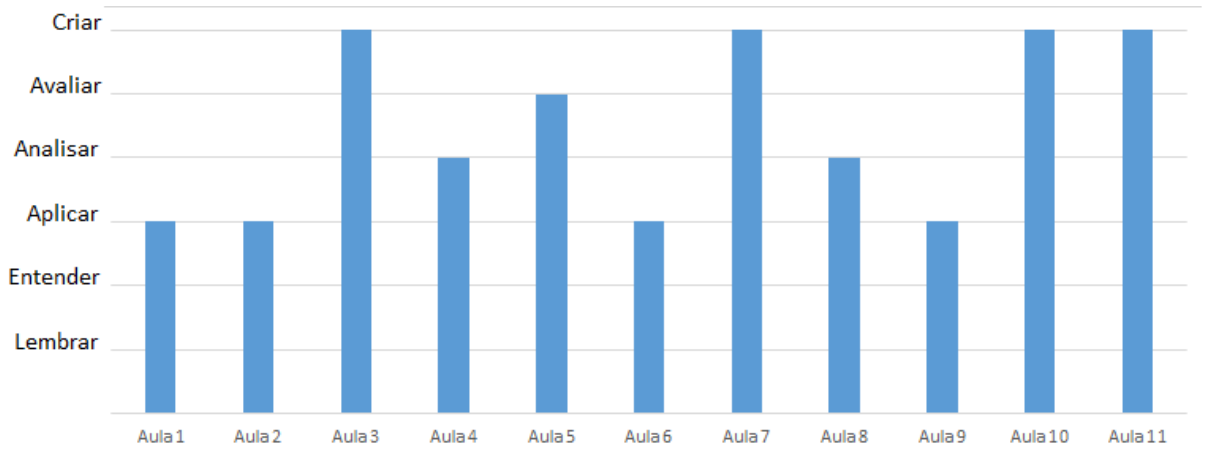


Figura 5.3: Categorias da Taxonomia de Bloom revisada utilizadas em cada aula do Curso

Ao contrário da Oficina, não houveram queixas durante o Curso indicando que as aulas não eram mais consideradas curtas. Da mesma forma, os alunos não indicaram que foram muito longas. Essa duração, portanto, pode aproximar-se do ideal para esse tipo de atividade, onde os alunos estão engajados e motivados (Figura 5.4).

A divulgação do Curso não foi feita de forma análoga à Oficina. A ficha de inscrição oferecida nas salas, na Oficina, foi substituída por um formulário online cujo endereço foi divulgado através de visitas às salas de aula e por um cartaz fixado no quadro de avisos da escola (disponível no Apêndice G deste trabalho). A necessidade de acessar um link de internet e inscrever-se, opondo-se a apenas escrever seu nome em uma lista, pode ter sido uma das razões da diminuição da quantidade de inscrições no Curso, em relação à Oficina. Não obstante, os mesmos critérios de seleção foram utilizados e quatorze alunos foram convidados a participar do Curso.

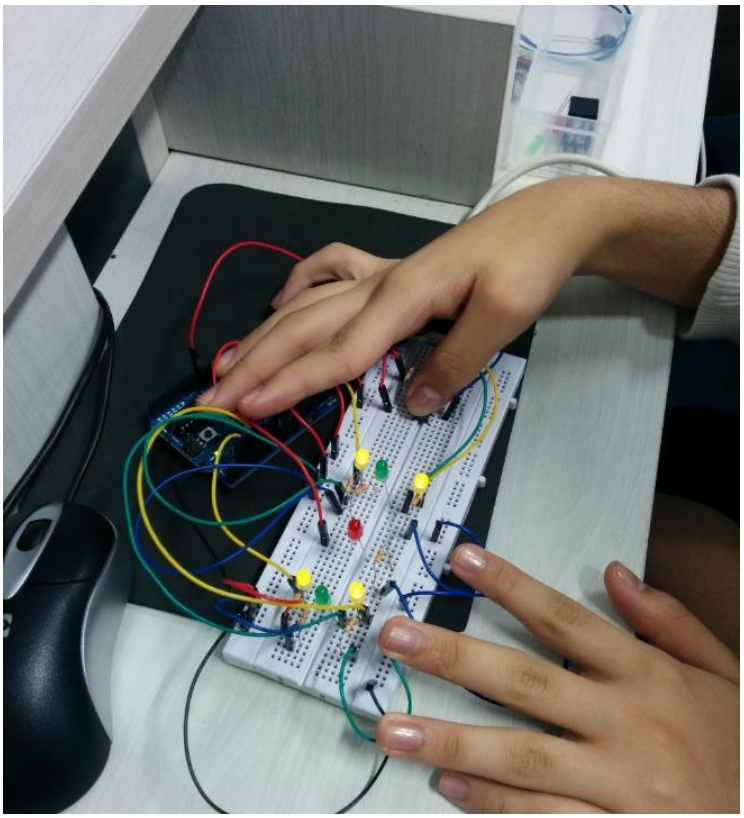


Figura 5.4: Aluno examinando circuito montado, no Curso

A inclusão dos desafios no Curso surgiu a partir de observações feitas durante a Oficina. Percebeu-se que os alunos, no processo de construção do projeto final de curso, agregaram maior valor ao seu produto por ser de natureza muito mais complexa que as tarefas propostas em aula. Essa situação gerou grande estima dos alunos pelos seus projetos, motivação para concluí-lo de forma mais correta possível e aumento do interesse nos cursos de computação e automação, manifestado através de perguntas sobre carreiras e universidades.

Os desafios, portanto, agiram como projetos intermediários durante o Curso: problemas complexos que demandam maior tempo e empenho dos alunos para chegar à solução, em comparação aos exercícios aplicados em aula. Desta forma foi possível manter sempre alta a motivação dos alunos, ao estarem constantemente sendo desafiados com novos problemas. Os desafios, portanto, agiram como uma espécie de avaliadores do conhecimento cumulativo dos alunos ao longo do Curso, uma vez que, em cada desafio, era necessário o domínio de todo o conteúdo já visto em sala.

O conceito de desafios também originou outro aspecto particular do Curso: as “versões difíceis” dos exercícios. Segundo Raabe e Silva (2005) cada aluno tem um ritmo próprio de aprendizado, uns de forma mais rápida e outros mais lenta. Um exercício passado na sala de aula, por exemplo, pode ser concluído pelos alunos mais rápidos em dez minutos e em trinta minutos pelos mais lentos. Esta característica torna-se um dilema para o educador: deve-se avançar a aula assim que o tempo dado alcançar um valor médio ou aguardar até que todos tenham terminado o exercício? O avanço da aula beneficia os alunos mais rápidos, pois eles receberão estímulo constante, e prejudica os mais lentos que, ao serem forçados a acompanhar a turma no próximo tópico, sem completar seu raciocínio e entendimento do tópico anterior, podem não conseguir prosseguir no conteúdo, afetando seu aprendizado e motivação. Por outro lado, se o educador decide esperar que todos os alunos consigam executar a tarefa e entender perfeitamente o tópico passado, os alunos mais rápidos ficarão entediados rapidamente após terminarem, esperando os alunos mais lentos. A solução encontrada foi a criação de uma “versão difícil” para cada exercício do curso, tendo como objetivo desafiar e manter a atenção dos alunos mais rápidos na

aula, enquanto os mais lentos utilizam esse tempo para resolver o problema. Desta forma, foi possível conservar a motivação dos alunos rápidos enquanto o educador auxiliava os alunos mais lentos, garantindo o andamento uniforme da turma.

Assim como na Oficina, ao final do Curso os alunos realizaram uma prova para medição parcial dos ganhos cognitivos e apropriação das competências e habilidades esperadas, vistas na seção 4.4.

O questionário de opinião, que mais uma vez foi apresentado aos alunos, também sofreu alterações baseadas nas impressões apuradas durante a Oficina. As cinco dimensões apresentadas na Seção 4.3.1 foram mantidas, no entanto algumas modificações nos enunciados das questões foram realizadas para evitar possíveis redundâncias e ambiguidades no entendimento por parte dos alunos.

5.2 Diário do Professor

Conforme mencionado na Seção 4.3.3, Guribye e Wasson (2002) afirmam que uma das ferramentas de suporte para a coleta de dados relevantes ao Estudo de Caso é o Diário do Professor, composto basicamente de anotações feitas durante o andamento do estudo. As anotações devem mostrar o dia a dia do estudo, evoluções e dificuldades de alunos, padrões de comportamento e expectativas, tanto na visão dos alunos quanto na do professor.

5.2.1 Oficina

O Diário da Oficina, por ter um período de duração muito curto, foi construído ao longo da aplicação, ao invés de utilizar o padrão de uma entrada por dia. Dessa forma as informações ficaram mais concentradas e foram mais facilmente analisadas.

Embora uma condição para inscrição tenha sido a ignorância em relação à programação, os alunos não mostraram grandes dificuldades em construir um algoritmo. Todos absorveram rapidamente as noções básicas de programação e o único empecilho na construção do programa no DuinoBlocks foi o aprendizado da ferramenta em si. O tempo de aprendizado de uma nova

ferramenta é um aspecto esperado em um curso, e os alunos não mostraram grandes dificuldades em alcançar um nível satisfatório de proficiência em relação ao referido ambiente.

Uma vez entendido o funcionamento do DuinoBlocks, os alunos começaram quase que imediatamente a experimentar com a ferramenta. Esse fato torna explícita a necessidade de adaptar a aula à rápida curva de aprendizado da ferramenta, para evitar a dispersão, como a utilização de questões extras ou desafios, para manter motivados os alunos mais velozes enquanto a classe espera os mais lentos.

Nenhum dos alunos apresentou conhecimento prévio em montagem de circuitos eletrônicos utilizando protoboard. Um pequeno tutorial foi utilizado no início da Oficina, e eventualmente todos os alunos tornaram-se aptos a construir seu próprio circuito.

Um dos desafios da Oficina foi gerenciar a atenção do professor entre todas as duplas, principalmente durante os momentos de montagem de circuito. Percebeu-se que a presença de um monitor é muito importante no sentido de auxiliar a execução das aulas, e para turmas maiores essa ajuda torna-se indispensável.

Os alunos mostraram claras manifestações de alegria quando seu programa/circuito funcionava, e em várias ocasiões chamavam o professor para mostrar o produto final olhando ansiosos, esperando o próximo desafio.

Houve troca entre os membros das duplas e entre duplas distintas em todas as aulas da Oficina. Em diversas ocasiões foi possível notar observações entre alunos, como "*o led está invertido, olha*" (Aluno 3, exercício em sala, aula 1), ou "*não esquece do resistor, ou o led pode queimar*" (Aluno 6, exercício em sala, aula 3). Esta prática sugere o desenvolvimento de um comportamento cooperativo, onde o objetivo final era construir um programa e um circuito eletrônico totalmente funcionais. Embora não esteja no escopo desta pesquisa, é possível assumir que também houve algum desenvolvimento dos alunos no domínio afetivo da Taxonomia de Bloom.

Nos términos das aulas, foi comum notar o interesse dos alunos em permanecer na sala realizando atividades e fazendo perguntas sobre programação e robótica, até os pais ligarem para buscá-los. Em uma ocasião, após ter ficado uma hora além do horário discutindo sobre os tópicos vistos em sala, o professor e alunos foram alertados pelo inspetor da escola que esta já deveria estar vazia para ser fechada. Vários alunos, durante tais conversas, mostraram interesse em aulas mais longas e aos sábados. Foi notado, também, um aumento – pelo menos temporário – no interesse em cursos universitários e carreiras do segmento STEM.

Considerando os resultados dos exercícios e da prova e as observações feitas durante as aulas da Oficina, é possível afirmar que os alunos não só absorveram todos os conteúdos de programação transmitidos como também alcançaram todas as categorias da Taxonomia de Bloom revisada. As análises que apoiam esses resultados são discutidas com maior profundidade nos tópicos adiante.

5.2.2 Curso

No Curso, ao contrário de na Oficina, por tratar-se de uma aplicação mais longa, houve a necessidade de individualizar as anotações de cada dia de aula, para aperfeiçoar a organização das informações.

O Curso ocorreu no Laboratório de Informática do Colégio Pedro II. Uma preocupação ao escolher o local das aulas foi garantir que houvesse um computador por aluno e que as mesas disponíveis na sala fossem bem espaçosas para serem utilizadas na fase de montagem dos circuitos. Na sala escolhida havia computadores com conexão à internet para todos os alunos, um projetor para apresentações e espaço para a realização das atividades práticas propostas durante o Curso. Embora algumas dificuldades técnicas – melhor explicadas adiante nesta seção – tenham ocorrido, o Laboratório de Informática mostrou-se um local satisfatório para a execução do trabalho.

Durante todas as aulas a professora de informática do Colégio esteve presente como suporte institucional ao Curso. Quaisquer problemas de infraestrutura, de logística ou com os alunos foram resolvidos por ela, tornando-a essencial para o sucesso da aplicação. A ponderação observada durante a Oficina, de que um

monitor tem a capacidade de auxiliar o andamento das aulas, foi confirmada no Curso.

A organização e execução de um curso extracurricular provou não ser uma tarefa trivial. Ao longo do processo alguns problemas de infraestrutura interferiram com o cronograma previsto, como por exemplo a conexão de internet. Esse problema, vivenciado recorrentemente na escola, tornou-se o principal empecilho na execução do Curso. A natureza online do DuinoBlocks, planejada como ponto positivo ao eliminar a necessidade de instalação nos computadores, converteu-se em um problema que ameaçou a continuidade do Curso. A solução encontrada foi utilizar uma versão off-line do ambiente, desenvolvida pelo autor do ambiente, de forma a não depender mais das limitações de rede da escola. Essa versão precisou ser transmitida manualmente a todos os computadores do laboratório, passando a ser adotada até o final do Curso. Questões sobre alterações de calendário da escola também afetaram o cronograma planejado.

As ferramentas Arduino e DuinoBlocks foram apresentadas no início do Curso e, como na Oficina, absorvidas rapidamente. Os alunos mostraram um retorno muito positivo em relação ao ambiente de programação, e não foi necessário utilizar muito tempo de aula para sua exposição. Os próprios alunos, por iniciativa pessoal, exploraram todas as funcionalidades do DuinoBlocks, comportamento previsto na Seção 2.3.2.

A partir da segunda aula a maioria dos grupos já havia entendido perfeitamente como uma protoboard funcionava, e salvo pequenos erros de falta de atenção, não apresentaram maiores dificuldades. Desta forma, assim que completavam um exercício, começavam quase que instantaneamente a experimentar com outros componentes eletrônicos, como botões e leds, que existiam nos kits. Alguns tentaram utilizar os displays de sete segmentos, também. A maioria, por experimentação, já aprendeu a diferença entre a potência dos resistores, e alguns grupos acessaram a tabela de cores dos resistores para referência.

As duplas formadas superaram expectativas em relação à complexidade dos circuitos montados por eles. Ao final da quarta aula, algumas duplas estavam programando circuitos com oito componentes independentes, com comportamentos e posições diferentes (até então o exercício mais complexo

continha apenas dois componentes). As perguntas feitas pelos alunos mostraram-se inteligentes e significativas, como por exemplo:

> "Por que a protoboard é construída dessa forma? " (Aluno A, exercício em sala, aula 2)
>
> “Por que eu preciso usar uma resistência para ligar um led? ” (Aluno B, exercício em sala, aula 3)
>
> “Os componentes precisam sempre ser colocados em uma protoboard? ” (Aluno G, exercício em sala, aula 3)
>
> "Por que o led tem uma direção certa para ser colocado na protoboard? “ (Aluno L, exercício em sala, aula 4)

A pergunta mais comum feita em sala é, genericamente, "*o que acontece se eu fizer X?*". Agindo de acordo com a metodologia construcionista, o professor sempre responde "*vamos programar/montar o circuito assim para ver o que acontece, e aí você me diz o que acontece*". Essa postura encoraja os alunos a explorarem os materiais disponíveis e novas possibilidades de interação entre eles.

A partir da sexta aula, os alunos das duplas já estavam divididos claramente entre programadores (aquele que escreve o programa) e engenheiros (aquele que monta o circuito), mas em todos os grupos houve troca de sugestões entre os membros dessas duas funções. No entanto, os alunos precisaram sincronizar informações essenciais, tanto de programação quanto de engenharia, para o funcionamento do conjunto, como planejar que pinos seriam utilizados para cada componente. Quando um integrante da dupla mostrava dúvida sobre algum tópico, o outro parava o que estava fazendo para tentar ajudá-lo. Isso aconteceu dentro e fora do grupo: foi possível notar que quando um grupo terminava a atividade, seus integrantes procuravam colegas com dificuldade para ajudar. Esse comportamento também foi visto na Oficina.

Na última aula do Curso, todos os projetos finais foram apresentados para a turma, um a um. Os alunos explicaram seus algoritmos e foi possível perceber pelo linguajar e forma de apresentação que eles entenderam e se apossaram do pensamento computacional durante o processo. Por exemplo, um aluno comentou, durante a apresentação do seu projeto “*nessa parte do algoritmo precisamos usar um loop com um contador decremental, já que resolvemos usar*

a variável contador como indicador de uma contagem regressiva". Nenhum projeto falhou, e todos foram filmados e fotografados, inclusive pelos alunos, que mostraram grande satisfação em apresenta-los para a turma, embora os alunos que utilizaram os carrinhos e a maquete foram os que ficaram mais orgulhosos com os seus projetos. Uma hipótese para esse comportamento é que carros e cidades são objetos concretos e familiares do dia a dia dos alunos, portanto animá-los gera maior sensação de conquista.

Alguns alunos se destacaram durante o Curso. Tais alunos que afirmaram gostar de física ou matemática e têm como objetivo ingressar em cursos universitários de ciência ou engenharia da computação. Perguntas frequentes eram feitas por eles sobre universidades e o mercado de trabalho, em relação a esse campo. É possível afirmar que o Curso influenciou em certo grau, pelo menos temporariamente, o interesse desses alunos em seguir carreiras relacionadas com o segmento STEM, da mesma forma que ocorreu durante a Oficina.

Um ponto observado foi que, a partir da metade do Curso, a demanda por atenção por parte dos alunos caiu sensivelmente em comparação com as aulas anteriores. Isso ocorreu devido a dois fatores: a) os alunos já estavam, nessa altura da aplicação, mais familiarizados com o software e hardware utilizados, e b) a constante postura cooperativa entre os alunos, onde um ajudava outros espontaneamente.

5.3 Projeto final

A última aula da Oficina e as duas últimas do Curso foram dedicadas ao projeto final. As etapas necessárias foram: formar uma ideia de projeto, elaborar o algoritmo para sua execução, desenvolver o programa, montar o circuito, corrigir possíveis erros e, finalmente, apresentar os produtos gerados para a turma.

O processo como um todo utiliza todas as categorias da Taxonomia de Bloom revisada, uma vez que é necessário *lembrar* dos trechos de código disponíveis na linguagem utilizada, *entender* como eles funcionam individualmente e em grupo, saber como *aplicar* esses conhecimentos em conjunto, *criar* um algoritmo necessário para realizar a operação desejada, *analisar* seu funcionamento e *avaliar* de forma crítica os resultados. Através desta atividade foi possível,

portanto, verificar se os alunos absorveram com sucesso os tópicos apresentados durante as aplicações, bem como o desenvolvimento das competências e habilidades exercitadas nesse período.

O tema do projeto final foi livre, contudo algumas ideias de projetos com certo nível de desafio foram apresentadas para estimular os alunos. Alguns adaptaram essas ideias, tornando-as ainda mais complexas, enquanto outros desenvolveram ideias próprias. Também como incentivo, dois carrinhos pré-montados com capacidade de programação via Arduino (Figura 5.5) e uma maquete de uma pequena cidade (Figura 5.6), também controlável via Arduino, foram disponibilizados para os alunos.

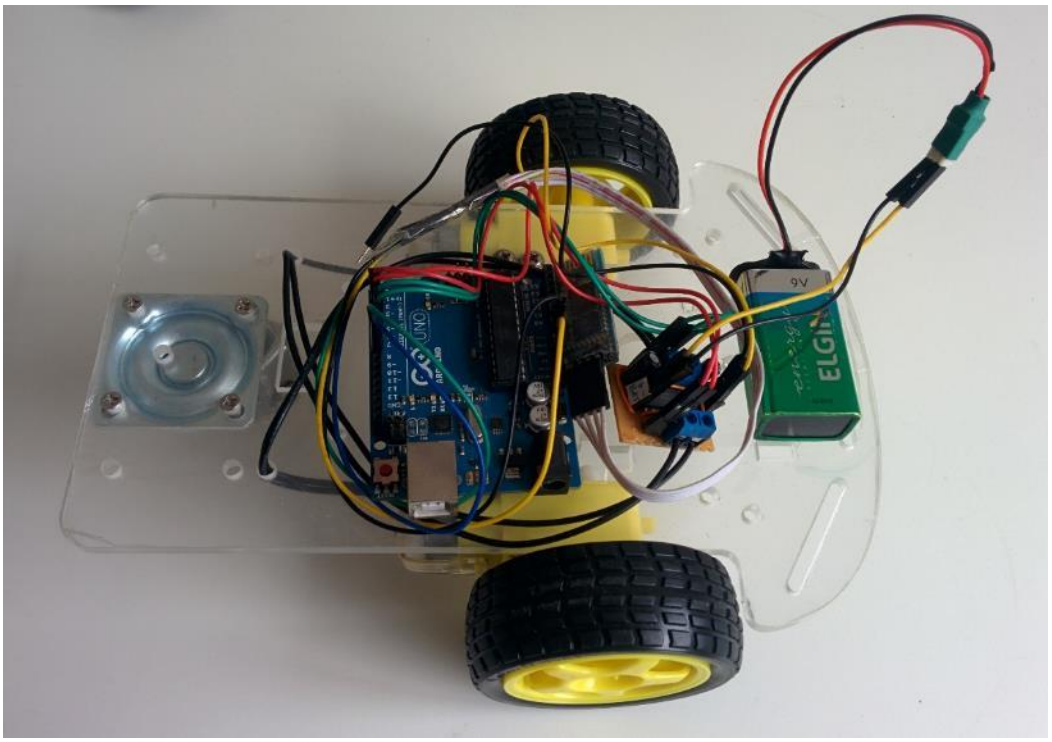


Figura 5.5: Carrinho de controle remoto criado como projeto final

Para cada projeto, no entanto, foi estipulada uma complexidade mínima para o algoritmo, cuja avaliação da evolução por parte dos alunos é o ponto principal deste trabalho. Em cada projeto seria necessário utilizar todos os conceitos básicos da programação, sendo eles variáveis, operadores (lógicos, relacionais

e aritméticos), atribuição de valores, entrada e saída de dados, condicionais, laços de repetição e funções.

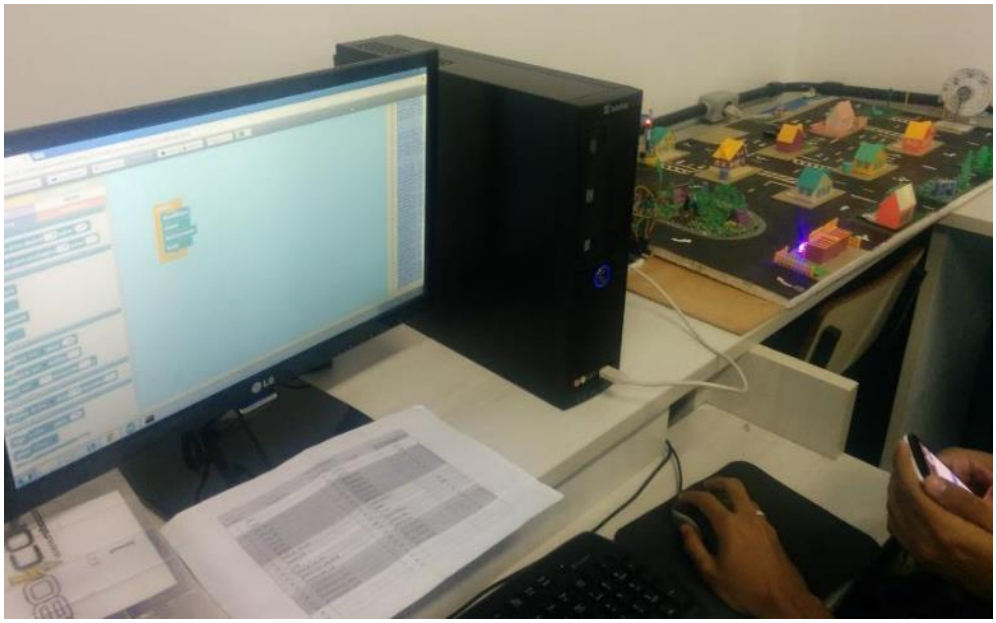


Figura 5.6: Maquete de cidade programada como projeto final

Devido à limitação de tempo, os projetos gerados na Oficina foram notoriamente mais simples que os do Curso. Isso não tira seu mérito, portanto, pois ao utilizar os mesmos conceitos de programação, os alunos da Oficina demostraram ganhos semelhantes aos do Curso.

Os resultados alcançaram os objetivos propostos, e em alguns casos ultrapassaram expectativas estabelecidas para as turmas. A capacidade de pensar, elaborar e construir algo necessita, como colocado anteriormente, do entendimento de todas as categorias do domínio cognitivo da Taxonomia de Bloom revisada. A demonstração dessa compreensão sugere que o aluno alcançou o nível esperado de conhecimento de programação e robótica.

5.4 Provas

Provas foram aplicadas no último dia de aula da Oficina e do Curso, com o objetivo de avaliar os ganhos cognitivos dos alunos durante as aplicações. A mídia escolhida foi a digital, por questões de praticidade, e os resultados

submetidos foram imediatamente inseridos, via internet, em uma planilha eletrônica.

Devido à curta duração dos encontros da Oficina, sua prova conteve apenas cinco questões, e os circuitos eletrônicos não foram disponibilizados para os alunos. No Curso, em contrapartida, houve maior disponibilidade de tempo. Portanto, foi dada aos alunos a escolha de utilizar os kits. Uma análise dessa escolha é apresentada mais adiante neste Capítulo.

Para sua construção foram utilizados como modelo testes aplicados nos últimos três anos no curso de Computação Unificado da UFRJ, analisados conforme a Taxonomia de Bloom revisada, como mostrado no Quadro 4.3 da Seção 4.4.

Durante a análise das provas modelo, viu-se que apenas três categorias da Taxonomia eram utilizadas: *Entender*, *Aplicar e Analisar*, as mais básicas. O uso dessas três categorias é muito comum em provas de computação, onde a maioria das questões solicita a construção de um programa, a correção de erros de um código ou a apresentação de quais são as saídas esperadas de um programa (Scott, 2003). Johnson e Fuller (2007) colocam que, de forma geral, cursos de programação tendem a considerar a categoria *Aplicar* como a mais importante a ser desenvolvida e inclinam-se a ignorar as categorias superiores a esta.

Esse cenário põe em evidência uma aparente deficiência no ensino de computação, pois o aprendizado se limita a apenas algumas categorias da Taxonomia. Segundo Scott (2003), cursos que englobam todas as seis categorias podem prover melhores resultados de aprendizado e melhores formas de se medir esses resultados.

Portanto, ao longo das aplicações deste trabalho, houve o cuidado de apresentar cada tópico abordando todas as categorias da Taxonomia, de forma a tornar o aprendizado cognitivamente mais completo. Da mesma forma, a prova aplicada foi construída de forma a englobar todas as categorias da Taxonomia, mantendo, ao mesmo tempo, as competências e habilidades necessárias existentes em cada questão da prova modelo.

As respostas dos alunos, de ambas as provas, estão disponíveis no Apêndice I deste trabalho.

5.4.1 Oficina

A prova conteve cinco questões e foi realizada por dez alunos no espaço de uma hora. Todos os alunos executaram as questões dentro do tempo concedido. Algumas dúvidas foram mencionadas durante o andamento da prova, mas todas elas tratavam do método de preenchimento do formulário de respostas. Os alunos não tinham certeza se a maneira correta de responder as questões era colar o código escrito traduzido a partir do algoritmo montado no DuinoBlocks ou utilizar uma imagem copiada do algoritmo em formato de blocos, a partir do ambiente.

O resultado médio foi: 90% de acertos, 2% de respostas parcialmente corretas e 8% de respostas inválidas. Ficou evidente, após a correção, que a maior dificuldade dos alunos foi no tópico laços de repetição. Apenas sete alunos de dez acertaram essa questão. Todos compreenderam com facilidade os demais tópicos, e no geral os resultados foram muito favoráveis.

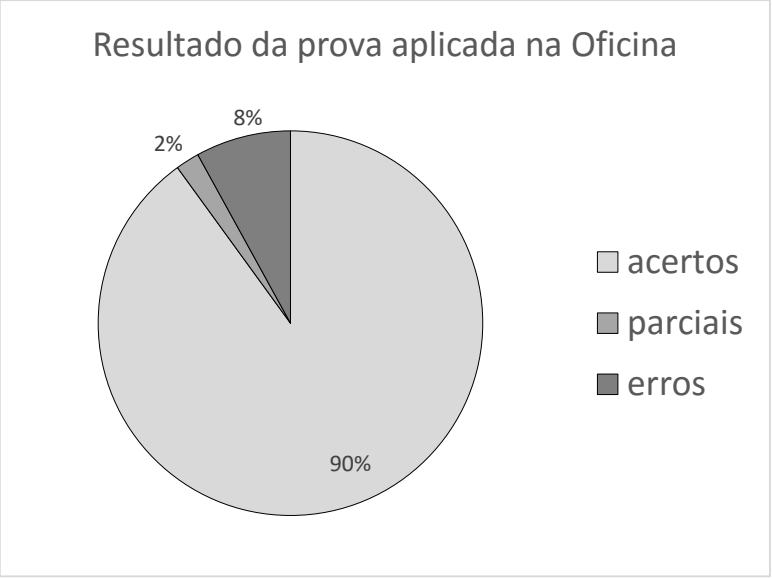


Figura 5.7: Resultados da prova aplicada na Oficina

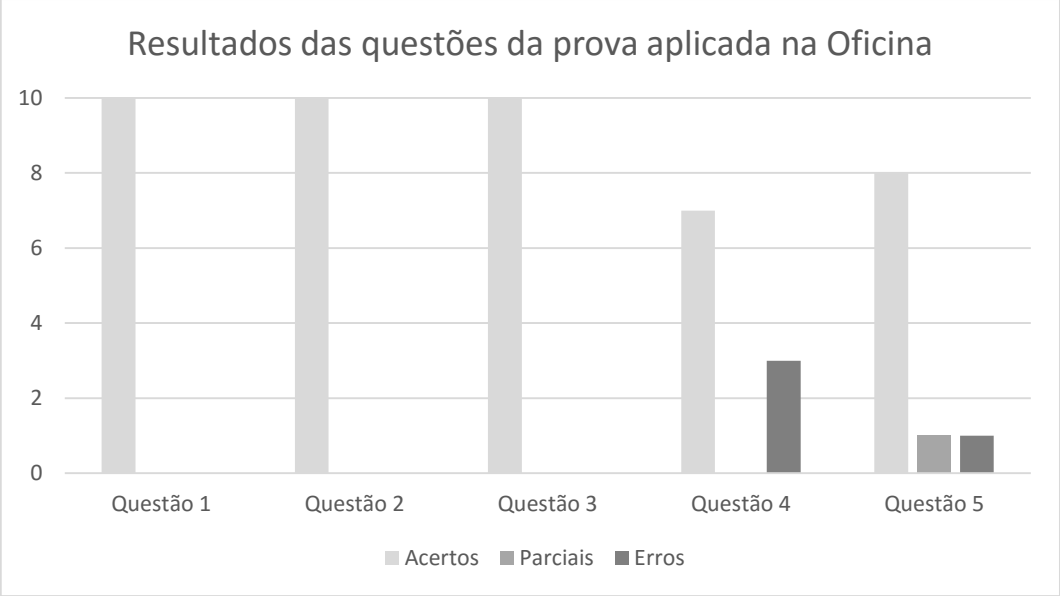


Figura 5.8: Resultados das questões da prova aplicada na Oficina

A Tabela 5.1 descreve as questões da prova aplicada na Oficina, considerando seus objetivos, categorias da Taxonomia e tópicos da ementa de programação abordados. A discussão completa das questões está disponível no Apêndice D.

Tabela 5.1: Descrição das questões da prova aplicada na Oficina.

Questão	Objetivo	Taxonomia	Tópicos
1	escrever código	*Aplicar*	constantes e variáveis, atribuição de valores, operações aritméticas, condicionais
2	corrigir código	*Analisar*	constantes e variáveis, atribuição de valores, operações aritméticas, condicionais
3	modificar código	*Aplicar*	constantes e variáveis, atribuição de valores, operações aritméticas
4	escrever código	Criar	constantes e variáveis, atribuição de valores, laços

5	descrever código	Entender	constantes e variáveis, atribuição de valores, operações aritméticas, operações relacionais, operações lógicas, condicionais

5.4.2 Curso

Após a aplicação na Oficina e coleta de feedback, o modelo de prova utilizado sofreu algumas modificações para deixar seus enunciados mais claros. Além disso, foram adicionadas mais três questões, uma vez que o Curso, mais extenso que a Oficina, permitiu maior aprofundamento no conteúdo.

O resultado médio da prova aplicada no curso foi de: 90,63% de acertos, 7,81% de respostas parcialmente corretas e 1,56% de respostas inválidas. Ficou evidente, após a correção, que a maior dificuldade dos alunos foi no tópico laços de repetição: em uma das questões que implicava no uso de laços, todos os alunos apresentaram algoritmos que não continham laços de repetição, embora também resultassem em um programa correto. Podemos inferir que a solução alternativa, além de não ser custosa de se construir, era cognitivamente muito mais simples, e, portanto, foi utilizada. Todos compreenderam com facilidade os demais tópicos, e os resultados foram ainda mais favoráveis que os obtidos na Oficina.

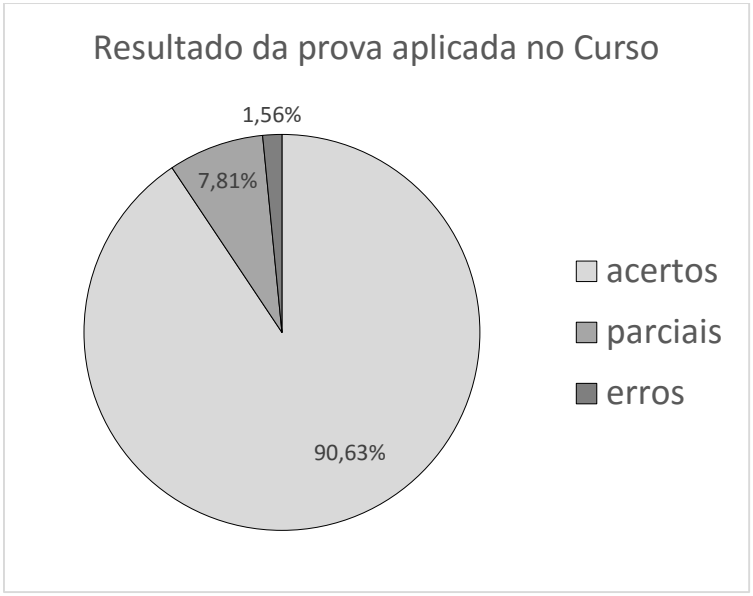


Figura 5.9: Resultado da prova aplicada no Curso

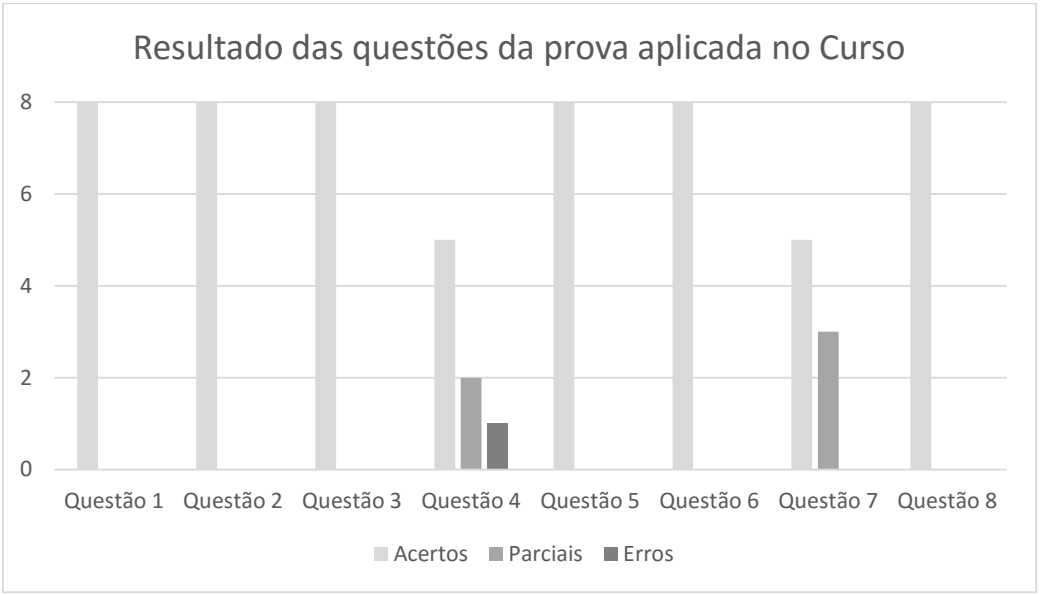


Figura 5.10: Resultados das questões da prova aplicada no Curso

A Tabela 5.2 descreve as questões da prova aplicada no Curso, considerando seus objetivos, categorias da Taxonomia e tópicos da ementa de programação abordados. A discussão completa das questões está disponível no Apêndice E.

Tabela 5.2: Descrição das questões da prova aplicada no Curso.

Questão	Objetivo	Taxonomia	Tópicos
1	escrever código	C*riar*	funções, condicionais
2	corrigir código	Analisar	constantes e variáveis, atribuição de valores, operações aritméticas
3	modificar código	Aplicar	constantes e variáveis, atribuição de valores, operações aritméticas
4	escrever código	Criar	constantes e variáveis, atribuição de valores, operações aritméticas, operações relacionais, laços
5	executar código e dar respostas	Aplicar	constantes e variáveis, atribuição de valores, operações aritméticas, operações relacionais, condicionais
6	escrever código	Criar	constantes e variáveis, atribuição de valores, laços
7	escrever código	Aplicar	constantes e variáveis, atribuição de valores, operações aritméticas, operações relacionais, condicionais
8	descrever código	Entender	condicionais

5.5 Questionário de opinião

Os alunos tiveram a opção de preencher um questionário online ao final do Curso contendo questões de opinião sobre a experiência realizada. Esse meio foi escolhido por tornar sua divulgação entre os entrevistados e recuperação de resultados muito mais simples, por estar dentro de um serviço online facilmente acessado de qualquer computador com acesso à internet.

Os questionários da oficina e do curso principal receberam juntos 19 respostas, ou 90% de adesão. Seu preenchimento foi opcional e anônimo, para encorajar os alunos a compartilharem suas impressões sobre as aplicações e opiniões sinceras sobre a metodologia, o conhecimento adquirido e os exercícios propostos.

Os resultados em relação ao ambiente de programação com a linguagem visual DuinoBlocks foram muito satisfatórios: a grande maioria dos alunos (89%) preferiu programar com esse ambiente, 5,3% disseram preferir o ambiente Arduino e 5,3% são indiferentes ao ambiente de programação.

O uso de circuitos físicos para a construção dos projetos também foi muito apreciado. 79% dos alunos preferiram testar seus programas em um circuito, mesmo implicando no trabalho da montagem do mesmo, e 89% disseram já se sentir confiantes para montar um circuito sozinho em uma protoboard.

Na pergunta "*O que você mais gostou no curso? ",* diferentes opções foram disponibilizadas para os alunos escolherem quantas quisessem. Questões sobre a dinâmica das aulas, o conteúdo visto, o ambiente de programação utilizado, a postura do professor, etc., foram apresentadas aos alunos. Praticamente todas as opções disponíveis foram escolhidas, com exceção de "trabalho em equipe". Em contrapartida, na pergunta "o que você menos gostou no Curso? ", também com diferentes opções, a única resposta escolhida por todos os alunos foi a duração das aulas/do curso – os alunos contestaram a "curta" duração da Oficina, afirmando que gostariam de aulas mais longas e por mais dias:

> "A duração das aulas" (Aluno F da Oficina)
>
> "A duração das aulas. Poderiam durar mais tempo" (Aluno A da Oficina)
>
> "A duração das aulas, poderia ter mais tempo semanal" (Aluno C da Oficina)
>
> "A duração das aulas, porque só houveram 5 aulas" (Aluno D da Oficina)

Alguns deles chegaram a afirmar que aceitariam aulas aos sábados para continuar com a experiência. Quanto ao Curso, os alunos ficaram satisfeitos com a duração das aulas, mas também argumentaram que gostariam de assistir mais aulas por semana.

O entusiasmo e motivação dos alunos durante o Curso pode ser verificado através de algumas frases escritas por eles no questionário de opinião, ao responder à pergunta "*O que você mais gostou no Curso?* ":

> "[…] o professor, os exercícios, a matéria vista, todas as coisas do universo, este curso foi uma benção na minha vida, amei tudo! ". (Aluno I da Oficina)
>
> "Conseguir fazer com que alguma coisa que eu pensasse conseguisse exercer sua função pensada, na vida real. " (Aluno E do Curso)
>
> "Esperava programar um robô, e gostei muito de descobrir que existem robôs por toda parte, e que um robô pode ter muitas aparências diferentes. "
>
> "Programar Robôs (bonecos) a se movimentarem! " (Aluno A do Curso)
>
> "Ter aprendido mais do que esperava..." (Aluno C da Oficina)

5.6 Evasão e reprovação

O índice de evasão na primeira aplicação - a Oficina - foi nulo. Algumas faltas foram anotadas ao longo das cinco semanas, no entanto nenhum aluno faltou com consistência ou comprometeu seu aprendizado com ausências excessivas. Entretanto, como a Oficina baseou-se em apenas cinco dias de aula, o conteúdo dado em cada uma consistia em grandes quantidades de teoria e prática dos tópicos abordados. Uma falta teria então a capacidade de afetar o desempenho dos alunos faltantes, e tais eventos foram combatidos com micro sessões de apoio a esses alunos na aula seguinte. Dessa forma, os alunos que haviam perdido a matéria estariam sempre atualizados logo após o início da próxima aula.

Alguns fatores contribuíram para faltas ao longo do Curso (Figura 5.18). Devido a sua longa duração, os alunos do curso passaram por duas sequências de provas escolares, em setembro e em novembro. As provas de setembro, que aconteceram por volta da aula 2, não mostraram influências graves, enquanto que as de novembro apresentaram profundas consequências visíveis nas taxas de presença das aulas 6, 7, 8 e 9. Segundo os próprios alunos, a bateria de provas de novembro é a última do ano, portanto não só demanda maior empenho dos alunos, por contemplar todo o conteúdo passado durante o ano letivo, para

todas as disciplinas, como para muitos é a última chance de alcançar notas suficientes para a aprovação.

A aula 10 mostra a volta dos alunos após a segunda sequência de provas, enquanto que a aula 11 mais uma vez apresenta um alto nível de faltas, cujo principal motivo foi a proximidade da aplicação da prova ENEM, a qual alguns alunos do Curso pretendiam fazer e, portanto, deixando de vir para estudar em grupos ou em casa.

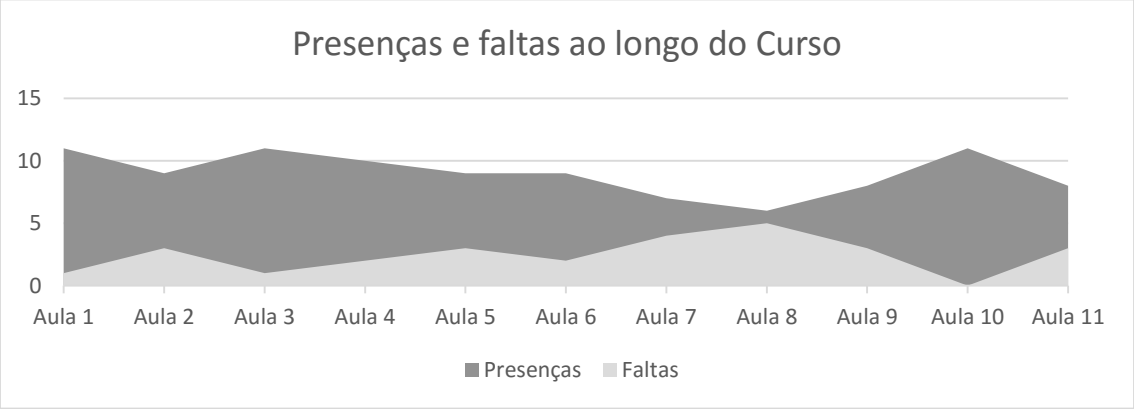


Figura 5.11: Gráfico de evolução de presenças e faltas ao longo do Curso

O número de reprovados, tanto na Oficina quanto no Curso, foi nulo. O critério de aprovação foi elaborado através de dois índices: apresentar entendimento dos tópicos de programação introduzidos durante as aulas e alcançar os níveis superiores da Taxonomia de Bloom revisada. Ambos os critérios puderam ser mensurados através das observações dos Diários do Professor e resultados das Provas, Questionários e Projetos Finais.

A análise e comparação dos números relativos às aprovações, reprovações e desistências de alunos nos cursos de programação aplicados nesta pesquisa, e oferecidos na UFRJ, foi compilada a partir dos dados dos cursos oferecidos no ano de 2014. Três turmas de Computação I, nomeadamente Engenharia (alunos do curso de Engenharia de Nanotecnologia), Ciência da Computação Turma A e Ciência da Computação Turma B (alunos de duas turmas distintas do curso de Ciência da Computação) serviram como representantes desta disciplina oferecida na UFRJ.

5.7 Análise das aplicações

Ambas as aplicações foram realizadas como atividades extracurriculares, oferecidas após os horários de aula. Segundo Hacker (2003), a estrutura de oficina extracurricular oferece diversos benefícios em termos de planejamento educacional. O currículo pode ser completamente construído pela equipe de pesquisa, oferecendo grande liberdade em termos de planejamento e execução do curso. Não houveram diretrizes externas sobre o que e como os alunos devem aprender, e não havia a obrigação, por parte dos alunos, de acumular pontos para alcançar aprovação, ao final da disciplina. Tais fatores auxiliaram na execução das aplicações, e talvez aplicações como as apresentadas neste trabalho possam ser uma forma de incorporar o ensino de programação e/ou robótica no currículo oficial nas escolas públicas. Pesquisas mais extensas, no entanto, são necessárias para determinar a melhor forma de inserir cursos como o proposto neste trabalho às escolas.

Apesar da curta duração da Oficina e do Curso, foi possível registrar uma rica quantidade de informações relevantes e importantes por meio das gravações, diário do professor, listas de exercícios, a prova e o questionário de opinião aplicados.

Os alunos, de uma forma geral, não estavam acostumados a programar ou trabalhar com circuitos elétricos. Assim, a atenção do professor foi muito necessária no sentido de orientá-los durante as tarefas, principalmente no início do curso. Eles, no entanto, não sentiram grandes dificuldades no aprendizado da programação, como pode ser visto nas respostas à questão “Você achou que aprender programação foi fácil ou difícil? ”, apresentadas abaixo no formato de um gráfico:

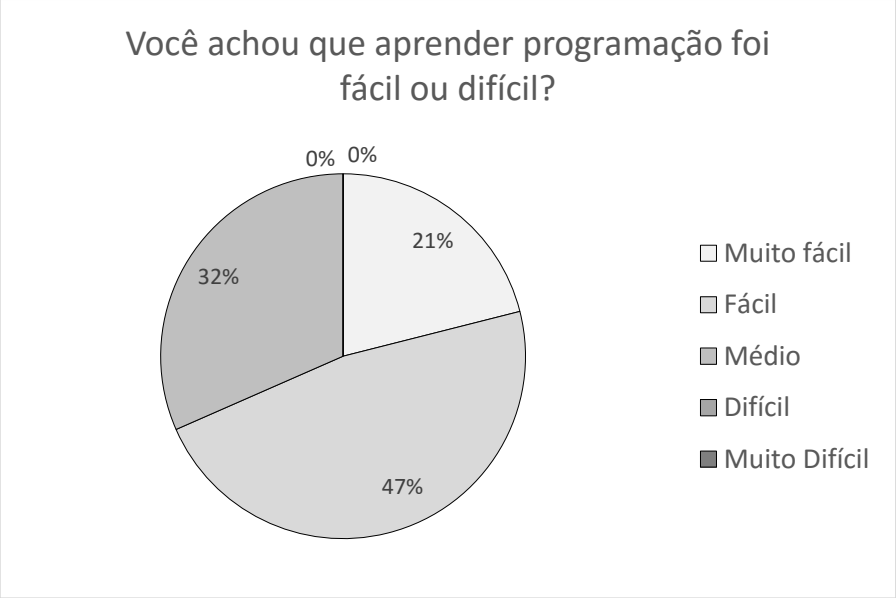


Figura 5.12: Respostas do questionário de opinião à pergunta acima.

A mesma estrutura de aula se manteve em ambas as aplicações: um novo tópico de programação era apresentado para a turma pelo professor, seu mecanismo explicado e desafios e exercícios lançados. Após a apresentação de soluções para o problema inicial proposto, as duplas eram muitas vezes estimuladas a pensar sobre possíveis modificações nos programas a fim de implementar novas funcionalidades (aumento de complexidade do problema).

Durante as aplicações, os passos dados pelos alunos na resolução dos problemas foram gravados com auxílio de um software de captura de tela e posteriormente analisados para avaliar o processo de desenvolvimento dos projetos criados por eles. Foi percebido um avanço rápido na construção de programas, uma vez que os estudantes não tiveram problemas no domínio do ambiente DuinoBlocks. Ao final do curso os alunos eram capazes de montar seus programas sem dificuldade aparente, pausando apenas em dúvidas de lógica ou de performance do algoritmo em questão. Os alunos não encontraram sérias dificuldades em trabalhar com o referido ambiente, conforme gráfico abaixo, montado a partir das respostas do questionário de opinião para a pergunta “Você achou que aprender a programar no DuinoBlocks foi fácil ou Difícil? “:

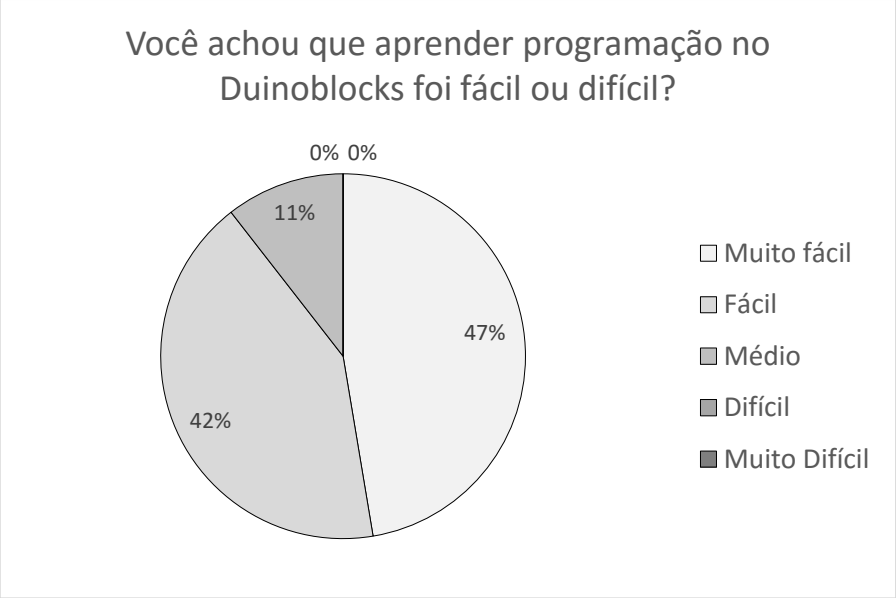


Figura 5.13: Respostas do questionário de opinião à pergunta acima

Um Diário do Professor foi mantido, narrando as diferentes atividades apresentadas em cada aula, pontos positivos e negativos, lista de chamada e observações gerais, conforme apresentado na Seção 5.2. Durante o desenvolvimento da Oficina e do Curso percebeu-se que tal documento foi de grande valia para feedback gerando pequenos ajustes nos exercícios e na apresentação dos tópicos dos encontros posteriores.

Uma preocupação inicial foi identificar se algum aluno teria dificuldades com o manuseio das ferramentas – o software DuinoBlocks e o hardware Arduino. Os alunos não apresentaram nenhum bloqueio visível com as questões computacionais (software e hardware), e não demoraram a se acostumar com a terminologia eletrônica e elétrica, bem como com a montagem de circuitos eletrônicos na protoboard.

Durante as aplicações, em diferentes momentos, foi possível perceber e anotar demonstrações de interesse e envolvimento por parte dos alunos, como por exemplo:

- Houve uma divisão, em todos os grupos, de papéis entre os membros. Essa divisão será discutida adiante, nesta seção.

- Os alunos mostraram claras manifestações de alegria quando seu programa/circuito funcionava. Chamavam o professor e outros alunos para mostrar o que produziram e aguardavam ansiosos, esperando o próximo desafio.
- Ao montar um programa ou um circuito particularmente difícil, os alunos frequentemente atribuíam valor sentimental ao projeto, hesitando em desmontá-lo para realizar a próxima tarefa. Em alguns casos os projetos chegaram a receber nomes e foram muitas vezes fotografados para compartilhamento em redes sociais (Figura 5.21).

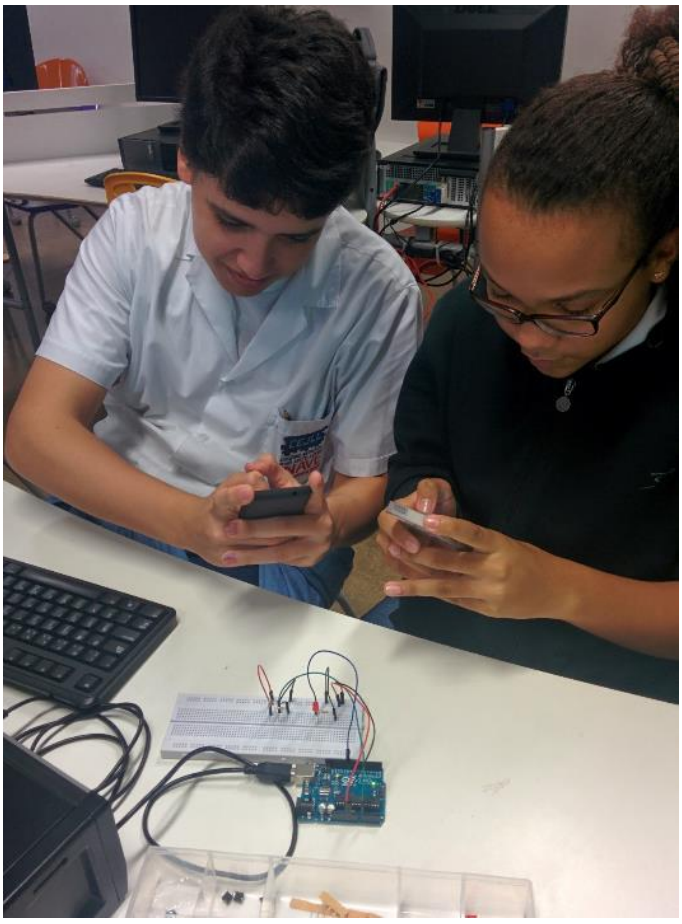


Figura 5.14: Alunos fotografando seu projeto

Foi possível constatar, ao longo da Oficina e do Curso, o desenvolvimento do entendimento de programação e robótica dos alunos através de interações entre

os membros das duplas. Múltiplas vezes foram observadas correções dos colegas como:

> "O led está invertido, olha" (Aluno D, desafio, aula 4 da Oficina);
>
> “Se você usar um loop aqui (apontando para o algoritmo do colega) dá para resolver o problema” (Aluno E, exercício em sala, aula 3 do Curso);
>
> "Não esquece do resistor, ou o led pode queimar" (Aluno A, exercício em sala, aula 5 do Curso).
>
> “Você precisa de uma condição verdadeira para continuar no while” (Aluno I, exercício em sala, aula 7 do Curso);

Essas observações indicam um crescimento claro do entendimento do conteúdo programático por parte dos alunos, uma vez que tais demonstrações requerem o domínio da categoria *Analisar* da Taxonomia de Bloom revisada ao olhar, de maneira crítica, os projetos dos colegas, e sugerir novas formas de abordar o problema em questão. Além disso, tais comentários mostram que os alunos não só aprenderam sobre questões de programação e robótica como estavam também ensinando seus colegas a respeito.

Durante um desses comentários, foi possível observar o aluno explicar uma questão relacionada a um algoritmo ao colega enquanto demonstrava cada passo apontando na tela do computador:

> “Para fazer um loop enquanto funcionar você precisa de uma operação relacional, olha. É a mesma coisa que você faz quando compara dois números, tipo se um é maior ou menor que o outro. Enquanto essa operação for verdadeira, o loop continua, e assim que for falsa, o programa sai dela. Você precisa criar uma variável, tipo um contador, e pensar quantas vezes o loop precisa rodar, aí você dá esse valor ao contador. Só não esquece de mudar o valor do contador toda vez que o loop rodar” (Aluno G, exercício em sala, aula 8 do Curso)

O passo a passo indicado pelo aluno demonstra seu entendimento do assunto, através do domínio das categorias *Lembrar*, *Entender* e *Aplicar* da Taxonomia em relação à programação.

Um exemplo de resolução correta de uma questão de prova pode ser analisado abaixo.

```
float usr1_repetir;
int syst1;
const int LED1 = 7;
void usr2_blink(){
digitalWrite(LED1, HIGH);
delay(1000);
digitalWrite(LED1, LOW);
delay(1000);
}

const int BUZZER1 = 8;

void setup(){
pinMode(BUZZER1, INPUT);
pinMode(LED1, OUTPUT);
}

void loop(){
usr1_repetir = 1000;
for (syst1 = 0; syst1 < (usr1_repetir); ++syst1){
usr2_blink();
}
tone(BUZZER1, 33, 2);
}
```

Figura 5.15: Solução do aluno 1 para a questão 4 da prova do Curso

O aluno, de acordo com o seu código, empregou corretamente a estrutura de laço de repetição, demostrando seu alcance da categoria *Criar* na Taxonomia de Bloom revisada. Ele, ademais, foi além do esperado ao construir e utilizar também uma função no programa (*usr2_blink*), mostrando também competência deste conteúdo. Este fato, à luz da Taxonomia, sugere domínio da categoria *Aplicar*, uma vez que o aluno lembrou da existência desta estrutura, entendeu como ela deve funcionar e soube como aplicá-la corretamente em um algoritmo.

Em contrapartida, um exemplo de dificuldade encontrada pode ser visto na resposta de outro aluno para a mesma questão:

```
const int LED1 = 3;
float usr1_Repetir;
const int BUZZER1 = 2;
void setup(){
pinMode(BUZZER1, INPUT);
pinMode(LED1, OUTPUT);
}
void loop(){
digitalWrite(LED1, HIGH);
usr1_Repetir = 1;
if((usr1_Repetir > 1000)){
tone(BUZZER1, 33, 2);
}
}
```

Figura 5.16: Solução do aluno 2 para a questão 4 da prova do Curso

A questão exigia que o aluno escrevesse um programa que faria um led piscar mil vezes em sequência, com intervalo de um segundo entre cada piscada. Após as mil piscadas, uma buzina deveria tocar por 2 segundos. O aluno, no entanto, mostrou não ter compreendido bem como funciona um laço de repetição para (for), uma vez que ele utilizou, em seu lugar, uma estrutura condicional. Esse erro é preocupante, pois está localizado na categoria *Entender* da Taxonomia de Bloom revisada. Isso significa que esse aluno não conseguiu desenvolver com sucesso a compreensão do que é e como utilizar uma estrutura de laço. Ao analisar as demais respostas desse aluno na prova, é possível confirmar que em seus códigos o conceito de laço de repetição não é empregado nenhuma vez com sucesso.

Os resultados das provas foram muito satisfatórios, em ambas as intervenções. As correções indicam uma taxa de acerto global de 90%, contra 30,4% dos alunos da disciplina de Computação I, do curso de Ciência da Computação, oferecido na UFRJ.

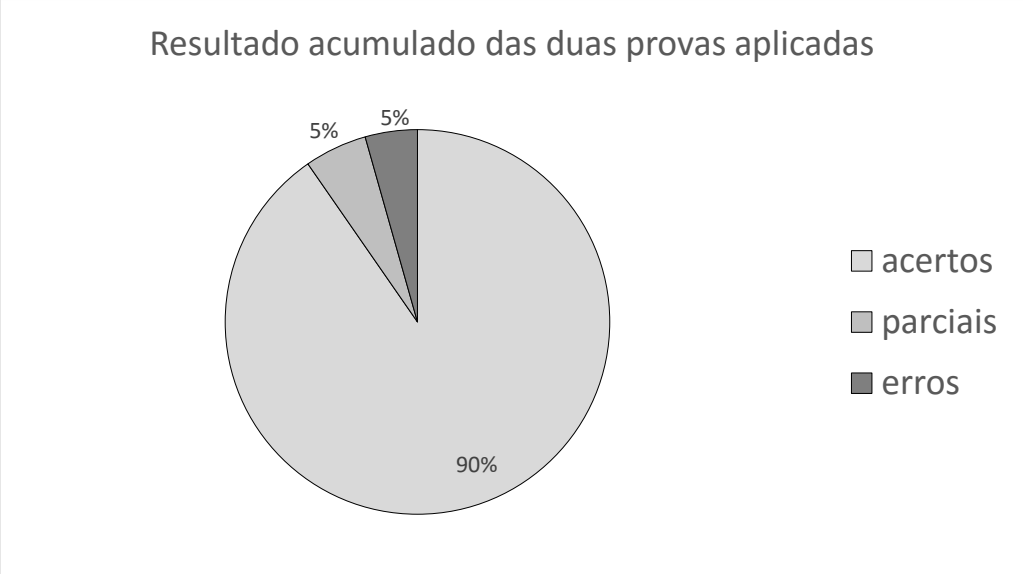


Figura 5.17: Resultado acumulado das duas provas – Oficina e Curso – aplicadas durante este trabalho

Devido à falta de detalhamento em relação às notas dos alunos da disciplina de Computação I, não foi possível construir um gráfico análogo ao da figura 5.24.

Os gráficos das figuras 5.25 e 5.26 expõem o uso da Taxonomia de Bloom revisada nas provas deste estudo e da UFRJ, respectivamente, em relação ao uso de suas categorias na construção das questões de prova.

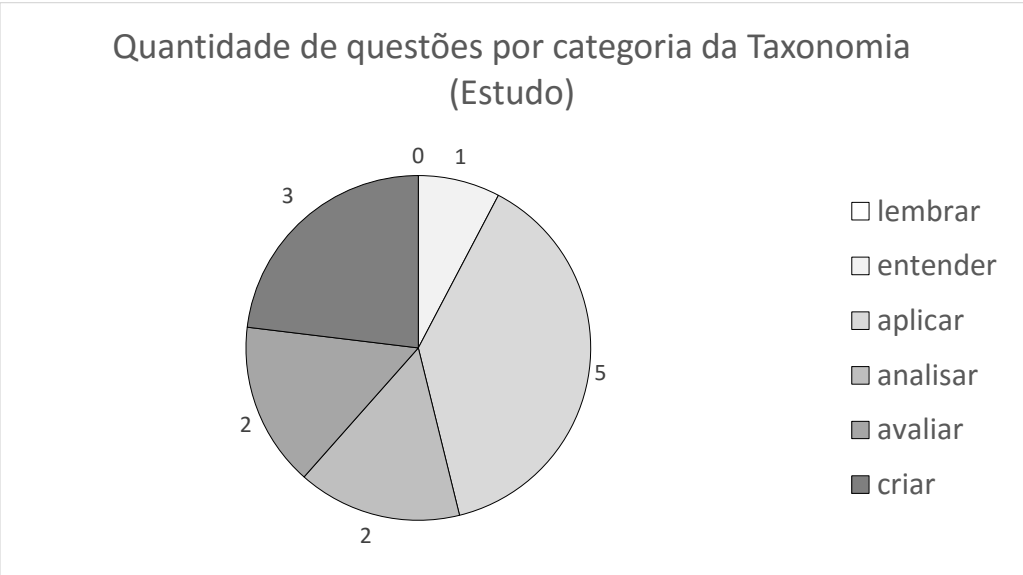


Figura 5.18: Quantidade de questões por categoria da Taxonomia de Bloom revisada, nas provas da Oficina e Curso, aplicadas neste estudo

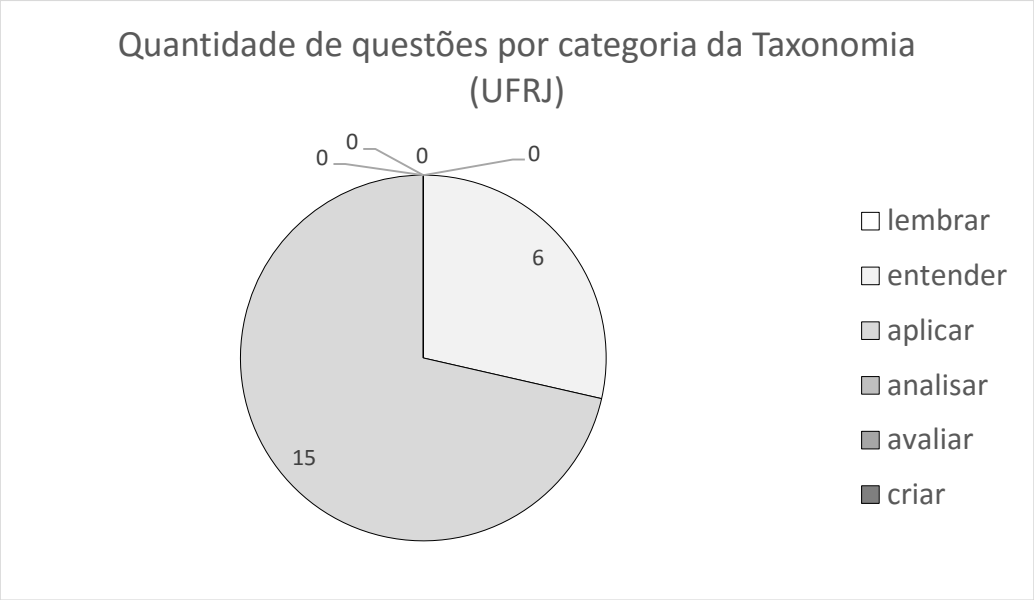


Figura 5.19: Quantidade de questões por categoria da Taxonomia de Bloom revisada, nas provas da Oficina e Curso, aplicadas na disciplina de Computação I, UFRJ

Os gráficos indicam que as provas deste estudo contemplaram todas as categorias da Taxonomia, enquanto que as provas da UFRJ focaram na terceira categoria, Aplicar. Esta característica foi apontada por Scott (2003) como sendo muito marcante em cursos de programação. O autor afirma ainda que provas construídas desta forma são deficientes no sentido que não exploram as possíveis habilidades que os alunos têm nas demais categorias da Taxonomia.

O aproveitamento dos alunos em aula foi sempre muito maior quando estavam em dupla, em comparação aos trabalhos desenvolvidos de forma solitária. A atenção do professor foi solicitada mais frequentemente por alunos que trabalharam sozinhos, sugerindo que em duplas os alunos conseguem chegar à solução mais facilmente. Uma organização informal se formou em quase todas as duplas, onde um dos integrantes assumia o papel de programador e o outro de engenheiro. Enquanto o programador desenvolvia o código, cabia ao engenheiro montar o circuito eletrônico na protoboard. Um ponto interessante observado dessa estrutura é que, embora os papéis e responsabilidades tenham sido definidos, ambos os alunos ainda pensavam e desenvolviam juntos o

algoritmo, base de todo o processo construtivo. Somente com o algoritmo pronto os alunos assumiam seus papéis no projeto.

A reação dos alunos quando percebiam um erro no seu código também foi notada. Ao executar os programas, os alunos não sabiam ao certo, inicialmente, onde estava o erro - no algoritmo ou na montagem do circuito. Era preciso então verificar ambos para encontrar o equívoco, o que demandava certo tempo e algumas vezes a atenção do professor ou colegas. Novamente, foi necessário o domínio das categorias superiores da Taxonomia de Bloom revisada para chegar à solução correta. No entanto, isso não inibiu de forma alguma o interesse – inclusive, muitas vezes erros eram considerados pelos alunos desafios a serem conquistados – e a vontade deles de montar um novo circuito a cada novo programa, independentemente de sua complexidade.

O fornecimento imediato de feedback presente na robótica foi de grande valia na busca por erros de algoritmo e montagem de circuitos. Os alunos recebiam, rapidamente, mensagens de erro ou resultados incompatíveis com o esperado, possibilitando-os a buscar correções e melhorias de performance. Esse aspecto foi observado por diversos alunos ao longo das aplicações. A correta interpretação do feedback, no entanto, só é possível após o aluno alcançar a categoria *Aplicar* da Taxonomia de Bloom revisada, uma vez que é necessário primeiro saber quais comandos foram utilizados, entender como cada um deles funciona individualmente e em integração com os outros comandos, e aplicá-los de forma correta para formar um algoritmo funcional.

Embora o ambiente visual baseado em blocos DuinoBlocks tenha sido utilizado, os alunos tiveram, a todo tempo, contato com o algoritmo textual gerado pelos blocos, a partir da funcionalidade “traduzir”. Embora não tenha sido requisitado, muitos deles aventuraram-se em tentar compreender o código textual, desistindo logo em seguida. Os alunos, em seguida, comentaram sobre a facilidade e sua preferência pelo ambiente visual contra o ambiente de programação nativo do Arduino, confirmada através das respostas coletadas nos questionários de opinião, apresentadas abaixo:

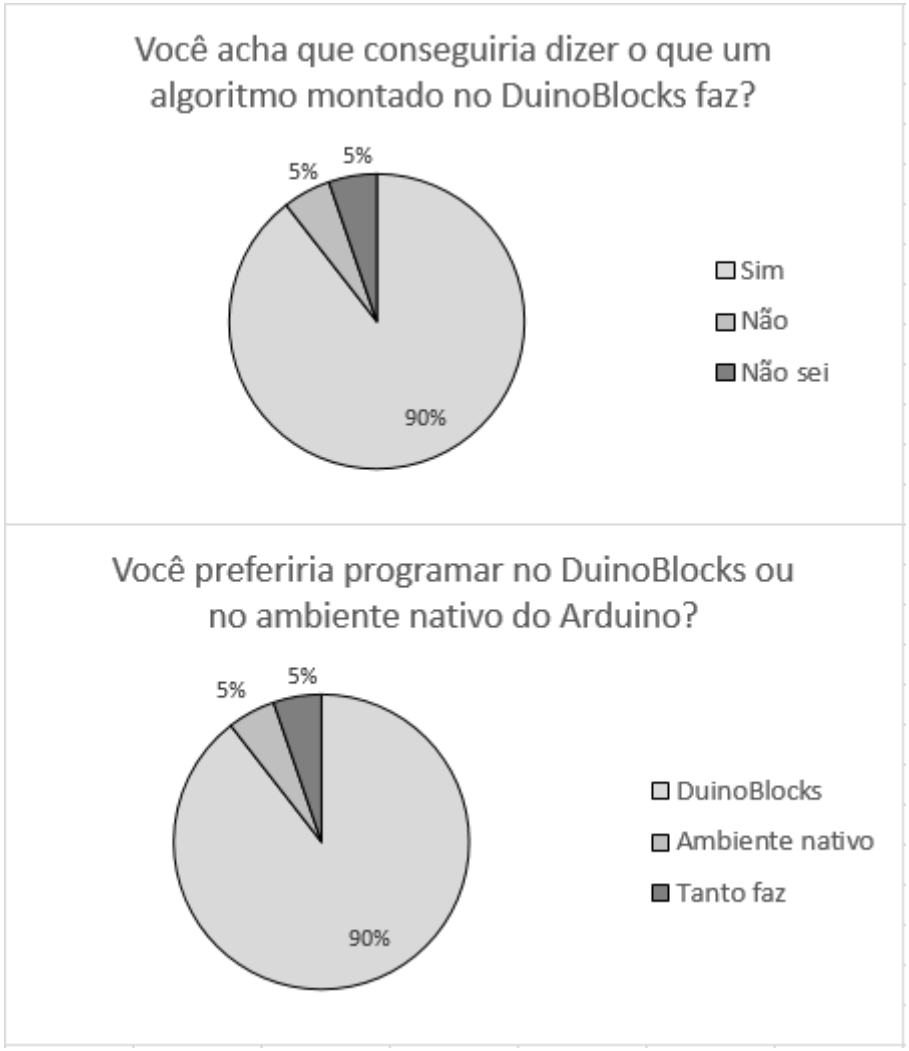


Figura 5.20: Respostas do questionário de opinião às perguntas acima

Os alunos frequentemente perguntavam acerca dos componentes eletrônicos utilizados na montagem dos circuitos que, embora apresentados apenas superficialmente no início do curso, (por não ser o objetivo maior da Oficina), eram empregados em todos os projetos. Eles buscaram mais informações também na internet e uns com os outros. Esse comportamento evidencia a curiosidade e uma predisposição deles no aprendizado baseado em projetos.

Um exemplo desse interesse foi visto durante o Curso: um aluno comentou que, em casa, assistiu diversos vídeos tutoriais sobre Arduino e montagem de circuitos na protoboard em um popular portal de vídeos online. Essa atividade foi de iniciativa própria do aluno, conforme mensagem de celular enviada pelo aluno ao professor do Curso:

> “Encontrei um site onde posso ver problemas para resolver e respostas de outros problemas, para estudar. Já consegui fazer dois [problemas], e vou tentar fazer mais um até semana que vem. ” (Aluno I, aula 7 do Curso)

As atividades e exercícios propostos eram sempre apresentados como provocações, explorando a predisposição natural dos alunos em aprender através de desafios. As duplas sempre reagiram bem a esta estratégia, tanto na Oficina quanto no Curso, algumas vezes chegando a estabelecer competições entre elas na busca de códigos e circuitos mais elegantes/eficientes. Entretanto, era comum observar alunos que já haviam terminado o exercício ajudavam os colegas que ainda estavam tentando resolvê-los.

Problemas de conexão com a internet e questões de configuração de computadores configuraram contratempos recorrentes no início do Curso. Esses obstáculos evidenciam a dificuldade que professores enfrentam nas escolas ao pensar e aplicar atividades que estejam fora do escopo tradicional da educação. A inflexibilidade inerente da instituição de ensino, portanto, atua como uma barreira que impede a realização de propostas inovadoras e, por fim, a própria evolução do sistema de ensino. Pedroza (2011) discute essa questão em profundidade, apontando causas, consequências e possíveis soluções.

Tanto na Oficina quanto no Curso, a atenção demandada pelos alunos foi imensa. A necessidade de incorporar um monitor à equipe foi detectada durante a Oficina, porém devido a questões de tempo, orçamento e logística, tal adição não foi possível para o Curso. Mesmo com os alunos divididos em duplas, o professor viu-se constantemente sendo requisitado pelos grupos. A intensidade da demanda, no entanto, dependeu da fase da aplicação. No início, quando os alunos ainda não estavam familiarizados com o software ou com o hardware, as solicitações foram muito mais frequentes. A medida que as aulas foram progredindo, os alunos se tornaram mais proficientes com os materiais e a demanda por atenção caiu. Outro fator que contribuiu para essa queda foi o comportamento cooperativo entre alunos da mesma dupla e entre duplas distintas, discutido anteriormente neste tópico.

Os resultados gerais da análise apontam para uma melhoria no entendimento de conceitos de tecnologia e engenharia através de suas descrições e explicações

acerca dos projetos. Isso pode ser afirmado através de observações feitas durante as aulas, dos resultados das provas e das respostas dos questionários. Todos os alunos atingiram níveis satisfatórios de compreensão em relação aos tópicos abordados de programação introdutória, alcançando inclusive as categorias superiores da Taxonomia de Bloom revisada.

A pesquisa mostra também que problemas no aprendizado de computação não estão necessariamente ligados à idade dos alunos. Nas duas aplicações deste trabalho, alunos do ensino médio exibiram plena capacidade de utilizar conhecimentos aprendidos de programação e eletrônica para criar projetos complexos unindo computação e robótica. Consequentemente, é possível afirmar que tais alunos foram capazes de dominar todas as categorias da Taxonomia de Bloom revisada com sucesso.

Podemos inferir, através da Tabela 5.2 da Seção 5.6 que em turmas menores os alunos têm apresentam menor tendência à evasão. Não é possível, entretanto, assumir esse fato como regra, pois existem parâmetros diferentes entre as turmas avaliadas na Tabela. A pesquisa também não pode comprovar qualquer relação entre o número de alunos em uma turma e a taxa de reprovação/evasão.

5.8 Considerações finais

Terminada a análise e discussão dos dados produzidos pelos Diários do Professor, provas, questionários e observações feitas durante as aplicações, é possível concluir que:

- Os alunos alcançaram as categorias superiores da Taxonomia de Bloom revisada com sucesso;
- Os alunos preferiram aprender programação e realizar exercícios com auxílio da robótica educacional, mesmo que isso implicasse em maior trabalho por parte dos próprios alunos;
- Os alunos demonstraram preferência a utilizar um ambiente visual de programação ao ambiente textual nativo do Arduino;
- O trabalho sugere uma ligação entre as taxas de aprovação, evasão e reprovação e a quantidade de alunos da turma: quanto mais alunos

inscritos, menor a taxa de aprovação e maiores as taxas de reprovação e evasão. Essa sugestão, no entanto, não foi estudada a fundo aqui;

- A organização dos alunos em grupos favorece a cooperação não só entre os alunos da dupla, mas entre duplas distintas, consequentemente aprimorando o aprendizado em geral.

6. CONCLUSÕES E TRABALHOS FUTUROS

O Capítulo 6 conclui o trabalho com considerações sobre a pesquisa, relata as dificuldades encontradas durante a execução do trabalho e propõe aprofundamentos para pesquisas futuras.

6.1 Trabalho realizado

Este trabalho propõe uma avaliação dos possíveis ganhos do uso da robótica educativa no ensino de programação introdutória para alunos do ensino médio, sem conhecimento prévio da disciplina.

Através de uma extensa revisão de literatura, pesquisas sobre o uso da robótica no ensino de programação foram levantadas, observando-se que uma vasta gama de iniciativas sugere uma ligação positiva entre o ensino de programação e a robótica educacional. No entanto, percebeu-se nos estudos contemplados a ausência de uma ferramenta que permitisse a avaliação padronizada e tornasse possível a comparação de resultados, alteração e reutilização de projetos em outros cenários. Devido à extensa aceitação e utilização da Taxonomia de Bloom apresentada nos Capítulos 2 e 3, sua versão revisada (Bloom et al., 1956; Anderson et al., 2001) foi escolhida como tal ferramenta neste trabalho.

Para realização do estudo, foram realizadas duas intervenções em escolas públicas cariocas: a primeira, de curta duração, chamada de Oficina, e a segunda, mais longa, chamada de Curso. Fez-se necessário utilizar alunos pré-universidade, pois estes ainda não foram apresentados à disciplina de programação. Nas escolas onde foram feitas as intervenções, os alunos do terceiro ano estavam ocupados com os estudos para o vestibular, que prestariam no final do ano, e o currículo do segundo ano já contempla essa matéria. Os alunos do primeiro ano, por sua vez, ainda não haviam aprendido a programar e nem estariam ocupados com outras agendas, como o vestibular, tornando-se então os participantes ideais da pesquisa.

As inscrições foram feitas a partir de um formulário online para alunos do primeiro ano e, após um primeiro filtro para retirada de interessados que já possuíam conhecimento de programação, os participantes foram selecionados a partir de um sorteio. A amostra, portanto, foi de 10 alunos na Oficina e de 12 no Curso, totalizando 22 adolescentes entre 15 e 17 anos. A construção dessas intervenções e a avaliação dos resultados foram feitas à luz da Taxonomia de Bloom revisada.

O trabalho utiliza como instrumento de pesquisa o Estudo de Caso, uma vez que este é mais indicado para estudos exploratórios (Yin, 2001). A análise qualitativa dos dados foi realizada a partir de passagens de diálogos entre alunos, dos resultados de exercícios e provas (Apêndice I) e das opiniões dadas pelos participantes (Apêndice K). A coleta desses dados foi feita por meio de questionários de opinião, observações do comportamento e individual dos alunos na forma de um diário de aula, resultados dos projetos apresentados e testes de conhecimentos.

Através da análise de dados, é possível afirmar que o trabalho proposto nesta pesquisa colaborou de forma significativa no entendimento de conceitos sobre lógica de programação, construção de algoritmos, estruturas de programação e montagem de circuitos eletrônicos. Isto pode ser ilustrado no comentário de um dos alunos:

> “Eu não sabia bem o que esperar quando entrei no curso, mas gostei muito de poder dar vida aos circuitos que eu montava, do jeito que eu queria. No início foi difícil, mas depois de um tempo passei a perceber os algoritmos por trás das coisas do dia a dia, como em um sinal de trânsito na rua. ” (Aluno B, Questionário de Opinião, Oficina)

Reações positivas, como a apresentada acima, motivaram a direção das escolas em que as intervenções foram realizadas a buscar uma forma de dar continuidade à oferta de um curso semelhante. O Colégio Estadual José Leite Lopes, palco da Oficina, passou a oferecer aos alunos, alguns meses após a intervenção, um curso de programação com robótica utilizando o Arduino, na mesma configuração da proposta deste trabalho.

6.2 Conclusões

As respostas das questões de pesquisa, apresentadas no Capítulo 4, podem ser vistas como uma síntese do que já foi detalhado no Capítulo 5, resumidas nas seções a seguir.

6.2.1 Alunos do ensino médio são capazes de aprender corretamente a ementa de um curso universitário de programação, utilizando robótica?

Segundo Fuller e seus colegas (2007), taxonomias educacionais são ferramentas importantes na avaliação da realização de alunos em cursos e na construção de objetivos de aprendizado. A principal característica da Taxonomia de Bloom revisada reside em classificar os objetivos de aprendizagem em ordem crescente de complexidade, com base nas operações mentais que eles requerem, independente do domínio do conhecimento. Através deles, é possível que os professores definam claramente o nível de conhecimento que eles esperam que os alunos atinjam em um determinado conteúdo, podendo, assim, preparar aulas e avaliações compatíveis com este nível.

A Taxonomia, portanto, indicou de forma sistemática os avanços cognitivos dos alunos ao longo das aulas e exercícios propostos nas intervenções. Além dos comentários observados, apresentados na seção 5.2, os resultados das provas (Figura 5.24) mostram que os alunos foram capazes de alcançar todas as categorias da Taxonomia com sucesso, em relação a todos os tópicos da ementa da disciplina de Computação I do curso de Ciência da Computação, oferecido pela UFRJ.

Os resultados apresentados ao longo da seção 5.7 e, principalmente, na figura que mostra o resultado acumulado das duas provas aplicadas, indicam que os alunos foram capazes de dominar toda a ementa do curso de Computação I. A nota média foi de 90% e nenhum dos alunos ficou reprovado.

6.2.2 Esses alunos são capazes de alcançar os níveis superiores da Taxonomia de Bloom revisada?

A resposta desta pergunta é consequência direta do cumprimento da anterior: afirmar que um indivíduo é capaz de aprender, corretamente, um determinado conteúdo, significa dizer que ele conseguiu, com sucesso, alcançar todas as categorias da Taxonomia, em relação a esse conteúdo.

A partir dos resultados analisados podemos afirmar que os alunos foram capazes de dominar todos os tópicos apresentados durante o Curso, ao alcançarem com sucesso todas as categorias da Taxonomia de Bloom revisada.

6.2.3 A disponibilidade de utilizar materiais físicos influencia no desempenho dos alunos?

A programação, tradicionalmente, é ensinada utilizando computadores para atividades práticas ou, em alguns casos, apenas lápis e papel. Os alunos devem ser capazes de observar seus algoritmos em execução através da tela do computador ou por meio de um passo a passo no papel, determinar se este está funcionando ou não e efetuar as devidas correções, caso seja necessário.

Diferentes estudos, abordados na seção 2.3 da revisão de literatura, indicam que a utilização da robótica educativa é interessante para o ensino de programação. Eles apontam que a robótica é uma facilitadora de aprendizagem de princípios científicos e matemáticos através da experimentação de materiais concretos (Rogers e Portsmore, 2004), incentivadora de classes baseadas em resolução de problemas (Rogers e Portsmore, 2004; Nourbakhsh et al., 2005; Robinson, 2005) e promotora de aprendizagem cooperativa (Nourbakhsh et al., 2005; Beer et al., 1999).

Um argumento para a preferência por materiais físicos no ensino com robôs é que os alunos veem os robôs como brinquedos e diversão (Mauch, 2001). De fato, um kit de robótica amplamente usado não só pela comunidade científica, mas também em escolas é desenvolvido pela Lego, um conhecido fabricante de brinquedos de blocos de construção para crianças. Alunos que usam este kit podem construir e programar robôs usando os mesmos materiais que eles têm em casa. Isso faz com que tudo o que aprendam com os kits pareça divertido também (Barker e Ansorge, 2007).

Durante as intervenções foi possível verificar que os alunos preferiram aprender programação e realizar exercícios com auxílio da robótica educacional, mesmo que isso implicasse em maior trabalho por parte deles ao precisar planejar e construir os circuitos necessários para executar os programas. Além disso, os alunos demonstraram preferência a utilizar um ambiente visual de programação

ao ambiente textual nativo do Arduino. Essas informações podem ser apoiadas pelos seguintes gráficos, construídos a partir das respostas do questionário de opinião:

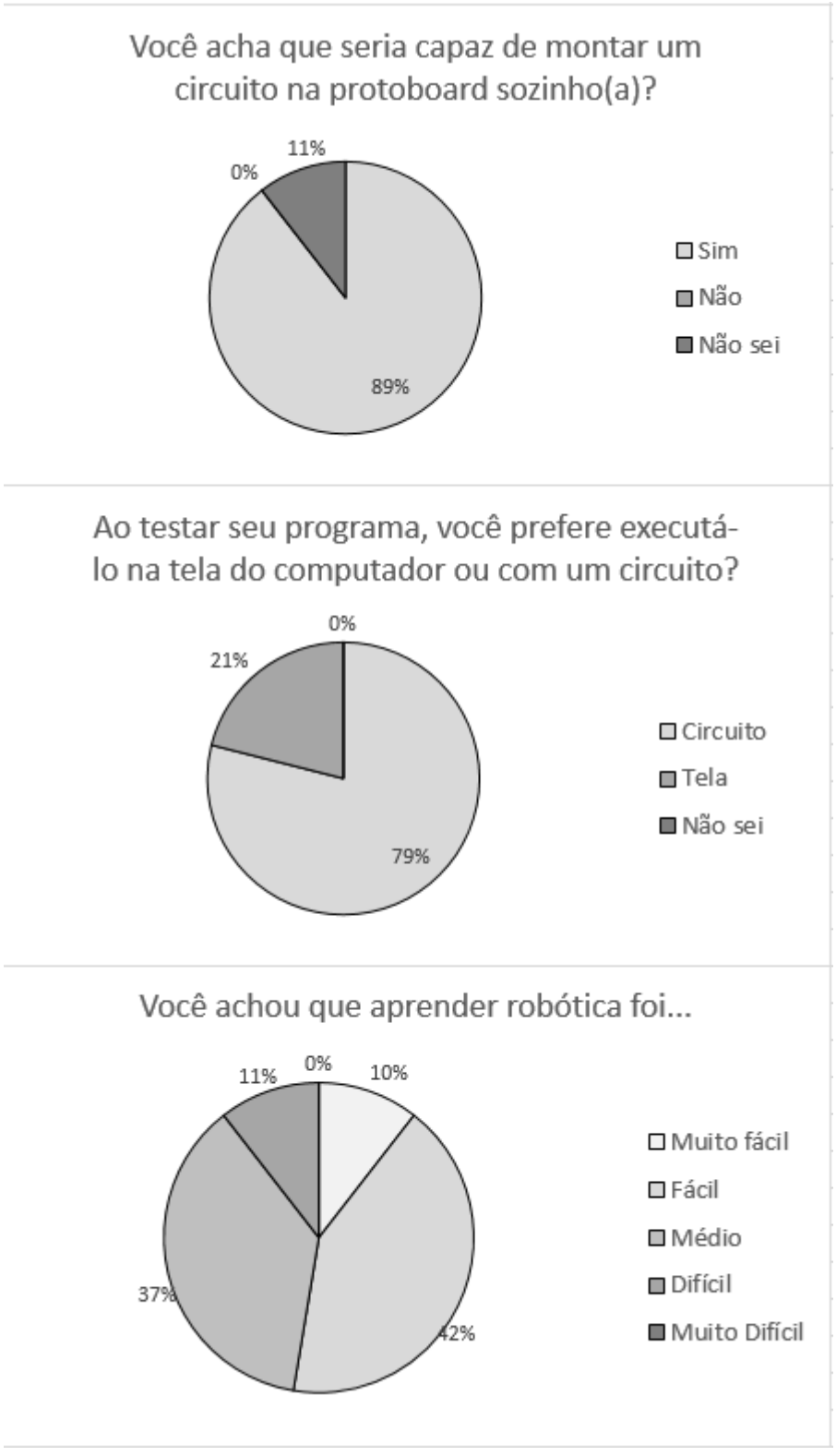


Figura 6.1: Gráficos construídos a partir de respostas dos questionários de opinião

O fato dos alunos terem sido capazes de aprender de forma satisfatória a ementa da disciplina de Computação I da UFRJ sugere que a disponibilidade de material físico, no formato de kits de robótica, foi benéfica para o seu desenvolvimento cognitivo.

6.3 Principais dificuldades encontradas

No desenvolvimento deste trabalho, algumas dificuldades foram observadas. A primeira, mencionada no Capítulo 5, foram as mudanças imprevistas nos calendários das escolas, gerando alterações no cronograma proposto para o decorrer das intervenções.

Problemas de conexão com a internet e infraestrutura do laboratório geraram contratempos recorrentes que impactaram não só na execução da aula, como também na motivação dos alunos – muitos ficavam entediados enquanto esperavam que os problemas fossem resolvidos. Como mencionado no Capítulo 5, a terceira aula do Curso precisou ser cancelada devido a problemas de infraestrutura. Esses obstáculos evidenciam a dificuldade que professores enfrentam nas escolas ao pensar em aplicar atividades que fogem do modelo "quadro negro e giz", sendo, portanto, uma barreira à realização de propostas inovadoras e, por fim, à própria evolução do sistema de ensino. Pedroza (2011) discute essa questão em profundidade, apontando causas, consequências e possíveis soluções.

A última adversidade mencionada é a demanda por atenção solicitada pelos alunos, principalmente no início das aulas. Por se tratar de um curso fundamentalmente prático e com componentes até então desconhecidos pelos alunos, estes se viam muitas vezes com dificuldades para utilizar o DuinoBlocks ou, principalmente, para montar circuitos eletrônicos. Mesmo com os alunos separados em duplas e sendo encorajados a adotarem uma postura exploratória, o professor era frequentemente solicitado a ajudar múltiplas duplas ao mesmo tempo, resultando em atrasos no cronograma de aula e diminuição na motivação dos alunos, por precisarem esperar sua vez de serem atendidos. Após essa experiência, ficou claro que para este tipo de atividade um segundo membro da

equipe se faz necessário para apoio, principalmente nos momentos de atividades.

6.4 Considerações finais

Cabe aqui mencionar que o uso da robótica não foi irrefletido. As aulas foram pensadas de forma a considerar os interesses e ambiente dos estudantes. Os exercícios e desafios propostos durante as aulas foram sempre, de alguma forma, construídos de maneira a abordar temas que despertassem a curiosidade e encanto da turma, ou referindo-se a assuntos do dia a dia dos alunos. Eles, portanto, mantiveram-se constantemente motivados e interessados, o que contribuiu para o seu aprendizado. Ou seja, houve uma preocupação em adaptar o uso da robótica educacional, por retroalimentação de dados coletados ao longo das aulas, aos estudantes que participaram desta pesquisa. Não podemos, portanto, afirmar que a simples adoção da robótica em sala sem um planejamento maior replicará os resultados aqui apresentados.

Desta maneira, é preciso atentar à forma como uma determinada ferramenta é utilizada em sala de aula, a fim de produzir resultados satisfatórios tanto para os professores quanto para os alunos.

Da mesma forma, durante todo o processo, tanto professor quanto alunos mantiveram-se estimulados, devido ao caráter construcionista das aulas. Um fator que contribuiu para a manutenção da motivação foi a quantidade de alunos na turma: 10 na Oficina e 12 no Curso. Desta maneira, foi possível atender todos eles durante os exercícios de maneira satisfatória. Turmas de ensino médio e de universidade, no entanto, frequentemente beiram – e algumas vezes ultrapassam – 40 alunos, afetando em demasiado a porção de atenção que o professor pode dispor a cada aluno, mesmo dividindo-os em grupos. Mencionado na seção anterior, o problema de distribuição de atenção a todos os alunos nas primeiras aulas da Oficina e do Curso seria exponencialmente maior à medida que a quantidade de alunos aumentasse, afetando seriamente o desempenho de todos.

6.5 Trabalhos futuros

O principal objetivo deste trabalho foi explorar o uso da robótica educacional no ensino de programação introdutória. Os resultados sugerem que os alunos do primeiro ano do ensino médio que participaram das intervenções são capazes de elaborar e entender algoritmos computacionais a partir da ferramenta de programação visual DuinoBlocks.

Entretanto, os resultados indicam também que outras questões de pesquisa podem ser estabelecidas, encorajando o desenvolvimento de novos estudos para que possam ser mais bem explorados.

Ambas as intervenções trataram de cursos extracurriculares, onde a inscrição foi voluntária. Os participantes estavam, por definição, altamente motivados por se tratar de um interesse pessoal. Além disso, as turmas de cada intervenção foram compostas de aproximadamente um terço da quantidade de alunos em uma sala de aula tradicional de ensino médio. Um aspecto que poderia ser aprofundado é a avaliação da participação de alunos pouco interessados em programação ou robótica, através da aplicação de uma variante do Curso em uma turma completa de ensino médio, durante todo um ano letivo.

Outro ponto interessante seria o acompanhamento e avaliação do desempenho de um aluno nas matérias associadas à educação STEM, durante sua participação no Curso de programação com robótica. Estudos indicam que disciplinas como matemática, física e química são favorecidas através do aprendizado de programação (Walck, 2014; Poore, 2011), embora não tenha sido encontrada nenhuma pesquisa onde a Taxonomia de Bloom revisada foi utilizada para sustentar as afirmações.

A última questão a ser sugerida é a avaliação dos resultados do Curso proposto considerando todos os três domínios da Taxonomia de Bloom Revisada para uma melhor avaliação do desenvolvimento intelectual dos alunos. Segundo Reeves (2006), todos os três domínios da Taxonomia são importantes para a avaliação completa do crescimento de um indivíduo. No entanto, devido à complexidade de utilizar todos os domínios e o foco no aprendizado cognitivo dos alunos de programação, a escolha de não contemplar os domínios afetivo e

psicomotor foi tomada neste trabalho. Uma análise mais completa dos resultados, à luz das três categorias da Taxonomia, traria um melhor entendimento sobre o ensino de programação introdutória com robótica educacional nas escolas. Desta forma, fazem-se necessárias investigações mais aprofundadas no sentido de desenvolver propostas pedagógicas para introdução de tal disciplina em sala de aula.

BIBLIOGRAFIA

Ackerman, E. (2001) “Piaget’s Constructivism, Papert’s Constructionism: What’s the difference?” Em: Constructivism: uses and perspectives in education, Vol 1 and 2, Conference Proceedings, Geneva, Research Center in Education (2001) 85-94

Alves, R. M.; Sampaio, F. F.; Elia, M. F. (2013) DuinoBlocks: Um Ambiente de Programação Visual para Robótica Educacional. In: XL Seminário Integrado de Software e Hardware (SEMISH) / XXXIII Congresso da Sociedade Brasileira de Computação (CSBC), 2013, Maceió, AL. Anais do XXXIII Congresso da Sociedade Brasileira de Computação. Porto Alegre, RS: SBC, 2013.

Anderson, L.W.; Krathwohl, D.R.; Airasian, P.W.; Cruikshank, K.A.; Mayer, R.E.; Pintrich, P.; Raths, J.; Wittrock, M.C. (2001) A taxonomy for learning, teaching, and assessing: A revision of Bloom’s Taxonomy of Educational Objectives. New York: Longman, 2001.

Araújo, A.; Portugal. D; Couceiro, M.; Rocha, R. (2013) Integrating Arduino-based Educational Mobile Robots in ROS. Proceedings of the 13th International Conference on Mobile Robots and Competitions April 24, 2013. Lisbon, Portugal.

Arendt, R. (2003) Construtivismo ou construcionismo? Contribuições deste debate para a Psicologia Social. Estudos de Psicologia 2003, 8(1), 5-13

Aureliano, V.; Tedesco, P. (2012) Ensino-aprendizagem de programação para Iniciantes: uma Revisão Sistemática da Literatura focada no SBIE e WIE. In Anais do Simpósio Brasileiro de Informática na Educação (Vol. 23, No. 1).

Balogh, R. (2011) Robotics course with acrob robot. Robotics in Education, 2011.

Barker, B. S., & Ansorge, J. (2007). Robotics as means to increase achievement scores in an informal learning environment. Journal of Research on Technology in Education, 39 (3), 229–243.

Barnes, D.J. (2002) Teaching Introductory Java through LEGO MINDSTORMS models. In SIGCSE '02: Proceedings of the 33rd SIGCSE Technical Symposium on Computer Science Education, New York, NY, USA, pp. 147–151. ACM.

Beer, R.; Chiel, H.; Drushel, R. (1999). Autonomous Robotics to Teach Science and Engineering. Communications of The ACM, 42 (6), 85-92.

Bini, E.; Koscianski, A. (2009) O ensino de programação de computadores em um ambiente criativo e motivador. Encontro Nacional de Pesquisa em Educação em Ciências. Florianópolis, 2009.

Bloom, B. S.; Engelhart, M. D.; Furst, E. J.; Hill, W. H.; Krathwohl, D. R. (1956). Taxonomy of educational objectives: The classification of educational goals. Handbook I: Cognitive domain. New York: David McKay Company.

Bogdan, R.; Biklen, S. (1994) Investigação qualitativa em educação uma introdução à teoria e aos métodos. Porto (Portugal): Porto, 1994.

Brown, T.; Kimura, T. (1994) Completeness of a visual computation model. Software - Concepts and Tools, v. 15, p. 34-48, 1994.

Carr, A.; Jonassen, D.; Litzinger, M.; Marra, R. (1998). Good ideas to foment educational revolution: The role of systematic change in advancing situated learning, constructivism, and feminist pedagogy. Educational Technology, 38(1), 5-14.

Chiaro, S.; Leitão (2005) O papel do professor na construção discursiva da argumentação em sala de aula. Psicologia: Reflexão e Crítica, 2005, 18(3), pp.350-357.

Chiou, A. (2004) Teaching Technology Using Educational Robotics. UniServe Science Scholarly Inquiry Symposium Proceedings.

Cognition and Technology Group at Vanderbilt. (1992). The Jasper experiment: An exploration of issues in learning and instructional design. Educational Technology Research and Development, 40(1), 65-80.

Coura, D. (2006) Produzindo animações através da programação por demonstração. Dissertação de Mestrado. Universidade de Viçosa, MG.

Cukierman, D.; McGee Thompson, D. (2007)Learning Strategies Sessions within the Classroom in Computing Science University CoursesProceedings of WCCCE 2007, 12th Western Canadian Conference on Computing Education, May 2007.

da Rocha, S. (2008) A Escola e os espaços não-formais: possibilidades para o ensino de ciências nos anos iniciais do ensino fundamental. Universidade do Estado do Amazonas. Manaus, 2008.

de Jesus E.; Raabe, A (2009) Interpretações da Taxonomia de Bloom no Contexto da Programação Introdutória. XX Simpósio Brasileiro de Informática na Educação.

de Souza, C. (2009) VisuAlg – Ferramenta de Apoio ao Ensino de Programação. Revista TECCEN, volume 2, número 2, setembro de 2009. ISSN 1984-0993.

Delgado, J.; Güell, J.; García, J.; Conde, M.; Casado, V. (2013) Aprendizaje de la programación en el Citilab. Revista CTS, nº23, vol. 8, Maio de 2013 (pág. 123 a 133).

diSessa, A. (2000) Changing Minds: Computers, Learning, and Literacy. MIT Press, 2000

Evangelista, S. (2001) Modelo para Programação Visual de Matrizes (MVM): Uma Nova Abordagem para Visualização, Manipulação e Programação de Algoritmos Matriciais. Relatório Técnico 14. Embrapa, ISSN 1517-0330. Dezembro, 2001.

Evans, B (2011) Beginning Arduino Programming. ISBN13: 978-1-4302-3777-8. Publicado em outubro, 2011.

Fagin, B.; Merkle, L. (2003) "Measuring the Effectiveness of Robots in Teaching Computer Science." ACM SIGCSE Bulletin 35.1 (2003): 307. Print.

Falkembach, G.; Tarouco, L.; Amoretti, M.; Viero, F. (2003). “Aprendizagem de Algoritmos: Uso da Estratégia Ascendente de Resolução de Problemas”. In 8º Taller Internacional de Software Educativo. Santiago, Chile.

Ferreira, C.; Gonzaga, F.; Santos, R. (2010) Um Estudo sobre a Aprendizagem de Lógica de Programação Utilizando Programação por Demonstração. In: Anais do XVIII Workshop sobre Educação em Computação, XXX CSBC, Belo Horizonte, MG, Brasil, pp. 981-990, 2010.

Ferreira, C.; Gonzaga, F.; Santos, R. (2010) Um Estudo sobre a Aprendizagem de Lógica de Programação Utilizando Programação por Demonstração. In: Anais do XVIII Workshop sobre Educação em Computação, XXX CSBC, Belo Horizonte, MG, Brasil, pp. 981-990, 2010.

Ferreira, M. (2005). “Proposta de Uma Metodologia Para Ensino-Aprendizagem de Algoritmos”. Trabalho de Conclusão de Curso: Curso de Informática (Bacharelado), Departamento de Ciências Exatas e Tecnológicas, Universidade do Planalto Catarinense. Lages.

Figueiredo, A.; Afonso, A. (2006) Managing Learning in Virtual Settings: the Role of Context. Information Science Publishing

Filatro, A. (2009) As teorias pedagógicas fundamentais em EAD. In: Litto, Fredric Michael; Formiga, Manuel Marcos Maciel. Educação a Distância: o estado da arte. São Paulo: Pearson Education do Brasil, p.96-104.

Flannery, L.; Silverman, B; Kazakoff, E; Bers, M; Bontá, P; Resnick, M. (2013) Designing scratchjr: Support for early childhood learning through computer programming. In Proceedings of the 12th International Conference on Interaction Design and Children, pages 1–10, 2013.

Friedrich, R.; dos Santos, D.; Keller, R.; Puntel, Márcio D.; Biasoli, D. (2012) Proposta Metodológica para a Inserção ao Ensino de Lógica de Programação com Logo e Lego Mindstorms. Anais do SBIE 2012

Fuller, U.; Johnson, C.; Ahoniemi, T.; Cukierman, D.; Hernán-Losada, I.; Jackova, J.; Lahtinen, E.; Lewis, T.; Thompson, D.; Riedesel, C.; Thompson, E.

(2007) Developing a computer science-specific learning taxonomy, Working group reports on ITiCSE on Innovation and technology in computer science education, December 01-01, 2007, Dundee, Scotland

Gaspar, A. (1993) Museus e centros de ciências: conceituação e proposta de um referencial teórico. Faculdade de Educação da Universidade de São Paulo. São Paulo, 1993

Giordan, M. (1999) O papel da experimentação no ensino de ciências. II Encontro Nacional De Pesquisa Em Educação Em Ciências.

Goh, H.; Aris, B. (2007) Using robotics in education: lessons learned and learning experiences. Smart Teaching & Learning: Re-engineering ID, Utilization and Innovation of Technology, 2. ISSN 983-42733-2-3

Gomes, T.; Melo, J. (2013) App Inventor for Android: Uma Nova Possibilidade para o Ensino de Lógica de Programação. Anais do II Congresso Brasileiro de Informática na Educação, p.620-629, 2013.

Guribye, F.; Wasson, B. (2002) The ethnography of distributed collaborative learning. Computer Supported Collaborative Learning 2002.

Hacker, L. (2003) Robotics in Education: ROBOLAB and robotic technology as tools for learning science and engineering. Tufts University, Department of Child Development.

Hall, C. e Johnson, A. (1994) Module A5: Planning a Test or Examination. In B. Imrie & C. Hall, Assessment of Student Performance. Wellington, New Zealand: University Teaching Development Centre, Victoria University of Wellington

Heinzen, T. (1994) Situational affect: proactive and reactive creativity. In: Creativity and Affect.

Hernán-Losada, I.; Lázaro-Carrascosa, C.; Velázquez-Iturbide, J. Á. (2004) On the use of Bloom's taxonomy as a basis to design educational software on programming. Proceedings of World Conference on Engineering and Technology Education, WCETE 2004, COPEC, Brasil, 2004, 351-355.

Hmelo, C.; Gotterer, G.; Bransford, J. (1997) Theory-driven approach to assessing the cognitive effects of PBL. Instructional Science, 25, 387–408.

Howard, R.; Craver, C.; Lane, W. (1996) Felder's Learning Styles, Bloom's Taxonomy, and the Kold Learning Cycle: Tying it All Together in the CS2 Course. SIGCSE '96.

Hundhausen, C. D.; Farley, S.; Brown, J. L. (2006) Can Direct Manipulation Lower the Barriers to Programming and Promote Positive Transfer to Textual Programming? An Experimental Study. Visual Languages and Human-Centric Computing, 2006.

Johnson, C.; Fuller, U. (2006) Is Bloom's taxonomy appropriate for computer science?, Proceedings of the 6th Baltic Sea conference on Computing education research: Koli Calling 2006, February 01-01, 2006, Uppsala, Sweden

Kelleher, C.; Pausch, R. (2005) Lowering the Barriers to Programming: a survey of programming environments and languages for novice programmers. ACM Computing Surveys 37, 2 (junho de 2005), 83–137.

Krathwohl, D., Bloom, B.; Masia, B. (1964) Taxonomy of educational objectives: the classification of educational goals. Handbook Volume 2: Affective domain. McKay, New York.

Krathwohl, D.; Bloom, B.; Masia, B. (1973) Taxonomy of educational objectives, the Classification of educational goals. Handbook II: Affective domain. New York: David McKay Co., Inc.

Lahtinen, E.; Ahoniemi, T. (2005) Visualizations to Support Programming on Different Levels of Cognitive Development. Proceedings of The Fifth Koli Calling Conference on Computer Science Education, 2005, 87-94.

Lahtinen, E.; Mutka, K.A.; Jarvinen, H.M. (2005) A Study of the difficulties of novice programmers. In: Proceedings of the 10th annual SIGSCE conference on Innovation and technology in computer science education (ITICSE 2005), Monte da Caparica, Portugal, June 27-29, 2005, pp. 14–18. ACM Press, New York

Langsch, C. (1999) Avaliação no Ensino à Distância via Web. Porto Alegre: Programa de Pós-Graduação em Computação da UFRGS.

Leffa, V. (2006) A aprendizagem de línguas mediada por computador. Pesquisa em Linguística Aplicada: temas e métodos. Pelotas: Educat, 2006. P. 11-36

LEGO (2015). Mindstorms. Visitado em 8 de março de 2015, em http://mindstorms.lego.com

Lister, R.; Adams, E.; Fitzgerald, S.; Fone, W.; Hamer, J.; Lindholm, M.; McCartney, R.; Moström, J.; Sanders, K.; Seppälä, O.; Simon, B.; Thomas, L. (2004) A multi-national study of reading and tracing skills in novice programmers, ACM SIGCSE Bulletin, v.36 n.4, Dezembro de 2004.

Lister, R.; Leaney, J. (2003) Introductory programming, criterion-referencing, and Bloom. Proceedings of the 34th SIGCSE technical symposium on Computer science education, Reno, Nevada, USA, ACM Press, 2003.

Major, L.; Kyriacou, T.; Brerenton, O.P. (2011) Systematic Literature Review: Teaching Novices Programming Using Robots. Proceedings of EASE 2011.

Maloney, J.; Resnick, M.; Rusk, N.; Silverman, B.; Eastmond, E. (2010) The Scratch Programming Language and Environment, ACM Transactions on Computing Education (TOCE), v.10 n.4, p.1-15, November 2010.

Mann, C.; Stewart, F. (2002) Internet communication and qualitative research: a handbook for researching online. Londres: Sage, 2002).

Mauch, E. (2001) Using technological innovation to improve the problem solving skills of middle school students. The Clearing House, 75 (4), 211–213.

McCracken, M; Almstrum, V.; Diaz, D.; Guzdial, M.; Hagan, D; Kolikant, Y.; Laxer, C.; Thomas, L.; Utting, I.; Wilusz, T. (2001) A multi-national, multi-institutional study of assessment of programming skills of first-year CS students, Working group reports from ITiCSE on Innovation and technology in computer science education, 1 de Dezembro de 2001, Canterbury, Inglaterra.

Mellis, A.; Banzi, M.; Cuartielles, D.; Igoe, T.; (2007) Arduino: an open electronics prototyping platform. CHI 2007, April 28 – May 3, 2007, San Jose, USA.

Miliszewska, I.; Tan, G. (2007) Befriending Computer Programming: A Proposed Approach to Teaching Introductory Programming. Journal of Issues in Informing Science & Information Technology 4, 277–289

Moon, J. (2002) How to use level descriptors. Southern England Consortium for Credit Accumulation and Transfer, 2002.

Moran, J. (1994) Novos caminhos do ensino à distância. Informe CEAD - Centro de Educação à Distância. SENAI. Rio de Janeiro, Ano 1, n. 5, out/nov/dez 1994, p. 1-3.

Moreira, H.; Caleffe, L. (2008) Metodologia da pesquisa para professor pesquisador. 2ed Rio de Janeiro: Lamparina.

Neto, W.; Cechinel, C. (2006) Uma Análise dos Problemas Enfrentados no Ensino-Aprendizagem de Fundamentos de Programação à Luz da Taxionomia de Bloom. In Anais do XXVI Congresso da Sociedade Brasileira de Computação: XIV Workshop sobre Educação em Computação. Campo Grande: Brasil, p. 244-253.

Norman, G.; Schmidt, H. (1992) The psychological basis of problem based learning: a review of the evidence. Academic Medicine, 67, 557–565.

Nourbakhsh, I. (2000). Robots and Education in the Classroom and in the Museum. IEEE Transaction on Robotics and Automation. Workshop for Personal Robotics for Education, IEEE ICRA 2000.

Nourbakhsh, I.; Crowley, K; Bhave, A.; Hamner, E.; Hsiu, T.; Perez-Bergquist, A.; Richards, S.; Wilkinson, K; (2005) "The Robotic Autonomy Mobile Robotics Course: Robot Design, Curriculum Design and Educational Assessment." Autonomous Robots 18.1 (2005): 103-27. Print.

Olsen, A. (2005). “Using Pseudocode to Teach Problem Solving”. In Journal of Computing Sciences in Colleges, 21(2). Consortium for Computing Sciences in Colleges, USA.

Orey, M. (2010) Emerging Perspectives on Learning, Teaching and Technology. Disponível em: http://www.textbookequity.org/oct/Textbooks/Orey_Emergin_Perspectives_Learning.pdf. Acessado em 29 de março de 2015.

Papert, S. (1991). Situating constructionism. In S. Papert and Is. Harel (Eds.), Constructionism (pp. 1-11). Norwood, NJ: Ablex.

Papert, S. (1994) A máquina das crianças: repensando a escola na era da informática. Porto Alegre: Artmed, 1994.

Papert, S. (2008) “A máquina das crianças: repensando a escola na era da informática.” Edição revisada. Porto Alegre: Artmed, 2008.

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

Pedroza, S. (2011) a evolução da educação: necessidade de uma nova gestão escolar. 2º Congresso Ibero-Americano de Política e Administração da Educação.

Perrenoud, P. (1999) Avaliação: da excelência à regulamentação das aprendizagens entre duas lógicas. Porto Alegre: Artes Médicas.

Petty, G (2002) Constructivist Teaching. Disponível em ***geoffpetty****.com/wp-content/uploads/2012/12/****constructivism****32.doc.* Acessado em 28/03/2015.

Poore, G. (2011) Python as a Tool for Squeezing More Learning into Mathematical Physics: Powerful, Versatile, and Free. Disponível em https://www.uu.edu/centers/faculty/programs/innovative/2011_GeoffreyPoore.pdf

Prensky, M. (2001) "Digital Natives, Digital Immigrants Part 1", On the Horizon, Vol. 9 Iss: 5, pp.1 – 6

Prietch, S.; Pazeto, T. (2010) Estudo sobre a Evasão em um Curso de Licenciatura em Informática e Considerações para Melhorias. WEIBASE, Maceió/AL.

Raabe, A.; Silva, J. (2005). “Um Ambiente para Atendimento às Dificuldades de Aprendizagem de Algoritmos”. In: Anais do XXV Congresso da Sociedade Brasileira de Computação: XIII Workshop sobre Educação em Computação. São Leopoldo: Brasil, pages. 2326-2337.

Ramirez, P.; Sosa, H. (2013) Aprendizaje de y con robótica, algunas experiencias. Revista Educatión 37(1), 43-63, ISSN: 2215-2644, enero-junio, 2013.

Ray, J. (2007) The Rosetta Stone and the rebirth of Ancient Egypt. Disponível em http://resolutereader.blogspot.com.br/2007/05/john-ray-rosetta-stone-and-rebirth-of.html. Acessado em 23 e maio de 2015.

Reeves, T. (1998) Evaluating What Really Matters in Computer-Based Education. Disponível em http://www.educationau.edu.au/archives/cp/reeves.html

Reeves, T. (2006) How do you know they are learning?: the importance of alignment in higher education. Int. J. Learning Technology, Vol 2, No. 4, 2006.

Resnick, M. (2007) Sowing the Seeds for a More Creative Society. Learning and Leading with Technology (Dec. 2007), 18-22

Robinson, M. (2005). Robotics-driven activities: Can they improve middle school science learning? Bulletin of Science, Technology & Society, 25(1), 73-84.

Rogers, C.; Portsmore, M. (2004) Bringing Engineering to Elementary School. Journal of STEM Education, Vol 5, No 3, 2004

Rubio, M.; Hierro, C.; Madrid y Pablo, A. (2013) Using Arduino to enhance computer programming courses in science and engineering. Proceedings of EDULEARN13 Conference 1st-3rd July 2013, Barcelona, Spain

Rubio, M.; Hierro, C.; Pablo, A. (2012) Using Arduino to enhance computer programming courses in Science and engineering. Proceedings of EDULEARN13 Conference 1st-3rd July 2013, Barcelona, Spain.

Santana, A. (2013) A utilização da plataforma Moodle para o apoio ao ensino presencial: um estudo exploratório em uma disciplina em nível de pós-graduação. Dissertação (Mestrado em Educação) Universidade Federal do Espírito Santo. Vitória, 2013.

Scaico, P.; Corlett, E.; Paiva, L.; Raposo, E.; Alencar, Y. (2012) “Relato da Utilização de uma Metodologia de Trabalho para o Ensino de Ciência da Computação no Ensino Médio”. In: XVIII Workshop de Informática na Escola, Rio de Janeiro. Anais do XVIII WIE.

Scott, T. (2003) Bloom’s Taxonomy Applied to Testing in Computer Science Classes. Journal of Computing Sciences in Colleges, v.19 n.1, p.267-274, Outubro 2003

Scott, T. (2003) Bloom’s Taxonomy Applied to Testing in Computer Science Classes. The Journal of Computing in Small Colleges, 19, 1 (October 2003), 267-274.

Simpson, E. (1966). "The classification of educational objectives: Psychomotor domain". Illinois Journal of Home Economics 10 (4): 110–144.

Slavin, R. (2000) Educational Psychology: Theory and practice (6th ed.). Needham Heights, MA.

Smith, D. (2000) “Building Personal Tools by Programming”. Communications of the ACM, v. 43, n. 8 (Aug), 92-95.

Smith, J. (1981). “A Method for Teaching Programming”. In: Proceedings of the Twelfth SIGCSE Technical Symposium on Computer Science Education: Technical Symposium on Computer Science Education. St. Louis, Missouri: United States, pages. 252-255, 1981. ACM Press New York, NY, USA.

SOFTEX. A indústria brasileira em perspectiva, v.1, n.1, nov. 2009. Disponível em: http://publicacao.observatorio.softex.br/_publicacoes/arquivos/completa/Software_e_Servicos_de_TI_2009.pdf. Acesso em: 14 jun. 2010.

Souza, D. (2006) How the Brain Learns. Disponível em http://ncbtp.org/docs/critical_difference.pdf

Stager, G. (2009) A Constructionist Approach to Teaching with Robotics. 9th IFIP World Conference on Computers in Education.

Svec, S. (2005) Taxonomy for Teaching: A System for Teaching Objectives, Learning Activities and Assessment Tasks (Revision of Bloom’s Taxonomy of the Cognitive Domain). In Pedagogicka revue 57, 453-476, 2005.

Tan, Y., Othman, A. (2013) The Relationship between Complexity (Taxonomy) and Difficulty. AIP conference proceedings.

Thorndike, E. (1913). Educational psychology: The psychology of learning (Vol. 2). New York: Teachers College Press.

Trautman, E. (2015) http://www.vikingcodeschool.com/posts/why-learning-to-code-is-so-damn-hard

Val, S.; Pastor, J. (2012) Robotics in Education. Advanced Research in Scientific Areas.

Valente, J. (1993) Por quê o computador na educação. Computadores e Conhecimento: repensando a educação. Campinas: Gráfica da UNICAMP.

Valentim, M. (2000) Atuação e perspectivas profissionais para o profissional da informação. O profissional da informação: formação, perfil e atuação profissional. São Paulo: Polis, 2000. cap. 7, p. 135-152.

Walck, S. (2014) Learn Physics by Programming in Haskell. The 3rd International Workshop on Trends in Functional Programming in Education, TFPIE 2014.

Whalley, J.; Lister, R.; Thompson, E.; Clear, T.; Robbins, P.; Kumar, P.; Prasad, C. (2006) “An Australasian Study of Reading and Comprehension Skills in

Novice Programmers, using the Bloom and SOLO Taxonomies", In: VIII Australasian Computing Education Conference (ACE2006), Computer Society, p. 243-252.

Wing, J. (2006) Computacional Thinking. Communications of the ACM, 49 (3), 33-35.

Yin, R. (2001) Estudo de caso: planejamento e métodos. 3ed Porto Alegre: Bookman.

Zhao, Y. (2003). What teachers should know about technology: Perspectives and practices. Greenwich, CT: Information Age Publishing.

Zilli, S. (2004) Robótica Educacional no Ensino Fundamental: Perspectivas e Prática. Dissertação de Mestrado, Universidade Federal de Santa Catarina, 2004.

APÊNDICES

Os apêndices deste trabalho encontram-se anexados em CD.

APÊNDICE A

Diário do Professor – Curso

Aula 1 – 02/09

Deu-se início ao Curso de Programação com o uso da robótica no Colégio Pedro II. De um total de quatorze alunos inscritos no Curso, onze estiveram presentes na primeira aula. Segundo os colegas, os três falantes estavam estudando em casa para um teste da escola que foi aplicado no dia seguinte. Esse tipo de ocorrência foi comum durante o Curso, como poderá ser observado ao longo do Diário. Além dos onze alunos inscritos, um aluno que se interessou pelo Curso, mas não havia se inscrito pelo formulário online, esteve presente com intenção de fazer parte da turma. O curso teve início então com um total de 15 alunos.

Houveram algumas questões técnicas no início da aula: o projetor e a conexão de internet não estavam funcionando, mas ambos foram resolvidos rapidamente pela professora de Informática, durante a introdução do Curso aos alunos. Esses problemas, como mostrado adiante, serão recorrentes até a terceira aula.

Um carrinho de controle remoto construído com uma placa Arduino e controlado por um celular foi apresentado à turma com fins de motivação, e os alunos ficaram muito animados com a ideia de criar seu próprio carrinho, no final do curso. Todos pareceram estar muito animados com o Curso, e quatro alunos abordaram o professor ao final da aula sobre onde poderiam adquirir um kit Arduino como o utilizado em sala.

Todo o conteúdo programado para a aula foi transmitido, com alguma folga. Isso deu-se à mudança de carga horária entre a Oficina e o Curso. O material didático já estava preparado para aulas de uma hora e meia, da Oficina, e precisou ser adaptado para aulas de três horas, do Curso. Os tópicos abordados foram a introdução à eletrônica, para entendimento de como um circuito funciona, contextualização da robótica com exemplos de aplicações no dia a dia, apresentação da protoboard e montagem do primeiro circuito elétrico, ainda sem o uso do Arduino. Nesses últimos tópicos o feedback coletado durante a Oficina

foi essencial para o refinamento do material de apresentação. Ao contrário dos alunos da primeira aplicação, os alunos do Curso entenderam os princípios do funcionamento da protoboard sem grandes dificuldades.

Aula 2 – 09/09

Os mesmos problemas técnicos vivenciados no primeiro dia de aula surgiram no início da segunda aula. A internet e o leitor de PDF (formato do material didático) não estavam funcionando no computador do professor, resultando no empréstimo do laptop pessoal da professora do Colégio. Além disso, foi necessário instalar manualmente o driver do Arduino em todas as máquinas, pois foi necessário utilizar a senha de administrador. A combinação desses problemas atrasou o início da aula em aproximadamente 30 minutos.

Outro ponto preocupante foi a necessidade de alguns alunos se ausentarem para estudar ou fazer trabalhos de escola. Eles foram alertados que embora o curso não valha nota no boletim, as faltas são contadas e caso um aluno falte mais de duas aulas ele perderá a vaga.

As ferramentas Arduino e DuinoBlocks foram apresentadas nessa aula, e como na Oficina, absorvidas rapidamente. Os alunos mostraram um retorno muito positivo em relação ao ambiente de programação, e não foi necessário utilizar muito tempo de aula para sua exposição. Os próprios alunos, por iniciativa pessoal, exploraram todas as funcionalidades do DuinoBlocks, comportamento previsto na Seção 2.3.2.

Na segunda aula maioria dos grupos já havia entendido perfeitamente como uma protoboard funciona, e salvo pequenos erros de falta de atenção, não apresentaram maiores dificuldades. Isso significa que os alunos, assim que completam um exercício, começam quase que instantaneamente a experimentar com outros componentes. Foi observado que quase todos os grupos tentam usar botões e todos colocaram mais leds em seus circuitos, inicialmente com apenas um. Alguns tentaram utilizar os displays de sete segmentos, também. A maioria, por experimentação, já aprendeu a diferença entre a potência dos resistores, e alguns grupos acessaram a tabela de cores dos resistores para referência.

Aula 3 – 16/09

No terceiro dia de Curso, a equipe apresentou-se mais cedo para garantir o funcionamento da infraestrutura da sala. Os problemas técnicos, no entanto, persistiram: o driver do Arduino precisou ser reinstalado e a internet estava muito lenta. Depois de resolvido o problema do driver (a senha de administrador dos computadores havia sido alterada), a equipe foi comunicada que um dos cabos de internet da escola estava rompido, e que um reserva de 2mb para toda a escola estava sendo usado - por isso a lentidão.

Como o funcionamento do DuinoBlocks é limitado à internet e a conexão estava impraticável, a turma precisou ser dispensada. Para compensar, duas aulas extras foram marcadas, nos dias 2 e 9 de outubro. Caso seja necessário, uma terceira aula será marcada no final de outubro para que o Curso seja concluído dentro do cronograma planejado.

Os problemas de conexão, vivenciados recorrentemente, tornaram-se o principal empecilho na execução do Curso. A natureza online do DuinoBlocks, planejada como ponto positivo ao eliminar a necessidade de instalação nos computadores, converteu-se em um problema que ameaçou a continuidade do Curso. A solução encontrada foi utilizar uma versão off-line do ambiente, desenvolvida pelo autor do ambiente, de forma a não depender mais das limitações de rede da escola.

Aula 4 – 30/09

Assim como na aula 3, a equipe mais uma vez chegou antes do horário da aula para preparação do ambiente. No entanto, por estar utilizando a versão off-line do DuinoBlocks, não houve absolutamente nenhum problema técnico. Isso permitiu que a aula fluísse a passos largos, e foi possível revisar todo o conteúdo das aulas 1 e 2, além de apresentar Desafio planejado para a Aula 3 como exercício final de aula, uma vez que os alunos já haviam adquirido o conhecimento necessário para resolvê-lo.

Os alunos apresentaram grande capacidade de retenção (todos lembravam como funciona a protoboard, e nenhuma dupla mostrou dificuldade em montar seus circuitos). Além disso, todas as duplas superaram expectativas em relação à velocidade de crescimento da complexidade dos circuitos montados. Ao final da aula, algumas duplas estavam programando circuitos com oito componentes independentes, com comportamentos e posições diferentes (até então o

exercício mais complexo continha apenas dois componentes). As perguntas feitas pelos alunos mostraram-se inteligentes e significativas, como por exemplo "*porque a protoboard é construída dessa forma?*", e "*por que o led tem uma direção certa para ser colocado na protoboard?*". A pergunta mais comum feita em sala é, genericamente, "*o que acontece se eu fizer X?*". Agindo de acordo com a metodologia construcionista, o professor sempre responde "*vamos programar/montar o circuito assim para ver o que acontece, e aí você me diz o que acontece*". Essa postura encoraja os alunos a explorarem os materiais disponíveis e novas possibilidades de interação entre eles.

Aula 5 - 07/10

O planejamento da Aula 5 conteve uma revisão de estruturas condicionais, visto na aula anterior, e a introdução do uso de funções em algoritmos. O tópico funções foi o que mais gerou dificuldades até então no Curso. Essa dificuldade se deu ao caráter abstrato de uma função que, embora muito utilizado de forma subconsciente ao realizar tarefas do dia a dia, não é facilmente compreendido por alunos cujo pensamento abstrato ainda não está completamente desenvolvido (Fuller e Johnson, 2007). Portanto, seguindo o planejamento feito durante a etapa de construção do Curso, o tópico funções foi o único apresentado durante a Aula 5.

Tentativas iniciais de esclarecimento falharam, devido ao caráter abstrato utilizado. Uma vez notada a falha pedagógica, uma demonstração por analogia foi empregada com sucesso.

Um ponto observado foi que, a partir dessa aula, a demanda por atenção por parte dos alunos caiu sensivelmente em comparação com as aulas anteriores. Isso ocorreu devido a dois fatores: a) os alunos já estavam, nessa altura da aplicação, mais familiarizados com o software e hardware utilizados, e b) a constante postura cooperativa entre os alunos, onde um ajudava outros espontaneamente.

Durante a aula, um erro no ambiente de programação DuinoBlocks foi descoberto acidentalmente por um aluno e reportado ao desenvolvedor, para correção.

Aula 6 - 14/10

A Aula 6 iniciou no horário planejado, e não houveram quaisquer problemas técnicos. O erro do ambiente de programação descoberto na aula anterior foi corrigido, sendo possível consolidar o aprendizado de funções através de um exercício de fixação.

O conceito de laços de repetição (while e for) foi introduzido na Aula 6. Para auxiliar o entendimento de laços de repetição, foi também apresentado o sensor de luminosidade incluso no kit (LDR). Com esse novo material, os alunos foram capazes de aplicar laços para funcionalidades do dia a dia, como em postes de luz, luzes de emergência em prédios, etc.

Foi possível observar que os alunos não apresentavam mais qualquer dificuldade em montagem de circuitos (salvo raras exceções, principalmente na utilização de componentes botões). A partir desta aula, os alunos das duplas já estão divididos claramente entre programadores (escreve o programa) e engenheiros (montam o circuito), mas em todos os grupos há troca de sugestões entre essas duas funções. A montagem inicial do algoritmo, no entanto, é realizada pelos dois papéis – programador e engenheiro. Além disso, os alunos precisam sincronizar informações essenciais para o funcionamento do conjunto, como alinhar que pinos serão utilizados para cada componente. Quando um integrante da dupla mostra dúvida sobre algum tópico, o outro para o que está fazendo para tentar ajudá-lo. Isso acontece dentro e fora do grupo: é possível notar que quando um grupo termina a atividade, seus integrantes procuram colegas com dificuldade para ajudar. Esse comportamento também foi visto na Oficina.

A partir dessa aula, o conceito de versão difícil – comentado na Seção 5.1.2 – foi integrado aos exercícios normais, para serem executados pelas duplas que já haviam terminado a versão normal. Eventualmente, todos os alunos quiseram fazer as versões difíceis dos exercícios, e foi possível notar a utilidade da natureza de feedback imediato da robótica: os alunos aprendem com seus erros e consertam o algoritmo e/ou o circuito de acordo com a necessidade. Notou-se também os primeiros casos de frustração pelo sistema não funcionar, mas todos

conseguiram programar e montar com sucesso os programas e circuitos pedidos nos exercícios.

Aula 7 - 21/10

A Aula 7 foi a segunda aula de desafio. Foram apresentados cinco projetos de programação e montagem de circuitos que englobam todos os tópicos estudados até o momento. Exercícios cujo enunciado solicitam a criação de um programa (e seu circuito) pertencem à categoria *Criar* da Taxonomia de Bloom revisada. Para explorar outros níveis, foram também inseridas questões na qual um programa precisava ser analisado e alterado para chegar ao resultado desejado, incluindo também as categorias *Analisar* e *Avaliar*. Vale ressaltar que as três categorias mencionadas são as três mais altas da Taxonomia. A correta execução dos desafios sugere, então, entendimento completo de acordo com a Taxonomia dos tópicos vistos até então pelos alunos.

Os alunos apresentaram alguma dificuldade em trabalhar com variáveis, o tópico menos explorado durante as aulas. No entanto, não tiveram dificuldade alguma de usar condicionais, laços de repetição, operadores relacionais e funções. Também não houve dificuldade na montagem dos circuitos. Os papéis de programador e engenheiro já estão claramente definidos em todas as duplas, e cada membro já começa a trabalhar no seu campo de expertise assim que o exercício é iniciado e o algoritmo definido pela dupla, em conjunto, mantendo sempre a comunicação de requisitos do sistema. Um exemplo dessa comunicação pode ser visto em uma ocasião, quando o engenheiro de uma dupla, ao observar as faixas de valores que o sensor de luminosidade trabalhava através do feedback coletado, orientou o programador a usar os números coletados na construção do programa.

Aulas 8 e 9 - 04 e 06/11

As Aulas 8 e 9 foram planejadas para exploração dos componentes eletrônicos disponíveis no kit e revisar o conteúdo de programação já visto. Portanto foram apresentados o potenciômetro, o display de sete segmentos, motor, led rgb e a buzina como detector sísmico.

Em ambas aulas a taxa de presença foi muito baixa. Isso se deu à concomitância das aulas do Curso com as datas da prova do Enem e as provas finais da escola. Tais fatores foram levados em conta na fase de planejamento e montagem do cronograma, porém, como posto anteriormente, certas questões interferiram na agenda de aulas resultando na entrada do período de provas. Foi solicitado aos alunos que fizessem um esforço e viessem para as aulas da semana seguinte, as últimas do Curso. Além disso, mensagens de telefone foram disparadas para todos os alunos, solicitando sua presença.

Em relação aos alunos que vieram nas Aulas 8 e 9, é possível afirmar que até então estão apresentando uma performance excepcional. São alunos que gostam de física ou matemática e tem como objetivo ingressar em cursos universitários de ciência ou engenharia da computação. Tais alunos fazem perguntas frequentes sobre universidades e o mercado de trabalho, em relação a esse campo. É possível afirmar que o Curso influenciou em certo grau, pelo menos temporariamente, o interesse desses alunos em seguir carreiras relacionadas com o segmento STEM, da mesma forma que ocorreu durante a Oficina.

Aulas 10 e 11 - 11 e 18/11

Conforme planejado, nas Aulas 10 e 11 ocorreram a apresentação, planejamento, desenvolvimento e apresentação dos projetos finais de curso. Assim como na Oficina, algumas ideias de projeto final foram oferecidas para os alunos se inspirarem, embora tenha ficado claro que o tema era livre, desde que estivesse de acordo com o nível de complexidade que já havíamos chegado.

Alguns adaptaram essas ideias, tornando-as ainda mais complexas, enquanto outros desenvolveram ideias próprias. Dois chassis, compostos de uma placa de plástico, dois motores e duas rodas, foram disponibilizados, juntamente com a maquete de uma pequena cidade, onde era possível programar os postes de iluminação, semáforos e outros aspectos.

Todos os projetos foram programados, os circuitos montados e apresentados para a turma, um a um. Os alunos explicaram seus algoritmos e foi possível perceber pelo linguajar e forma de apresentação que eles entenderam e se apossaram do pensamento computacional durante o processo. Por exemplo, um

aluno comentou, durante a apresentação do seu projeto "*nessa parte do algoritmo precisamos usar um loop com um contador decremental, já que resolvemos usar a variável contador como indicador de uma contagem regressiva*". Nenhum projeto falhou, e todos foram filmados e fotografados, inclusive pelos alunos, que mostraram grande satisfação em apresenta-los para a turma, embora os alunos que utilizaram os carrinhos e a maquete foram os que ficaram mais orgulhosos com os seus projetos. Uma hipótese para esse comportamento é que carros e cidades são objetos concretos e familiares do dia a dia dos alunos, portanto recriá-los gera maior sensação de conquista.

A prova foi aplicada após as apresentações dos projetos, na Aula 11. Durante a prova, foi possível observar que alguns alunos apresentaram dificuldades com temas específicos, não da programação, mas da robótica. Ficou claro que os tópicos deficientes foram exatamente os apresentados nas Aulas 8 e 9, quando grande parte dos alunos faltou. Os alunos presentes nessas aulas não apresentaram dificuldades ao realizar a prova.

Ao final do Curso, um grupo de alunos mais uma vez indagou sobre questões da faculdade de computação, especializações, oportunidades e escolhas de carreiras, mostrando interesse genuíno em seguir nesta área. Esses alunos, muitos satisfeitos, colocaram que o Curso foi "*muito interessante ao mostrar que a programação e a robótica são áreas fascinantes e que não são tão difíceis quanto pensávamos*".

APÊNDICE B

Prova da Oficina

Prova

*Required

1 - Escreva um programa que compara dois números escolhidos por você. Se o primeiro for maior que o segundo, acende o LED1. Se o segundo for maior que o primeiro, acende o LED2. Se a soma dos dois números for maior que 10, acende o LED3. *

escrever código - aplicar - constantes e variáveis, atribuição de valores, operações aritméticas

2 - Conserte o programa acima para que o LED1 ligue. *

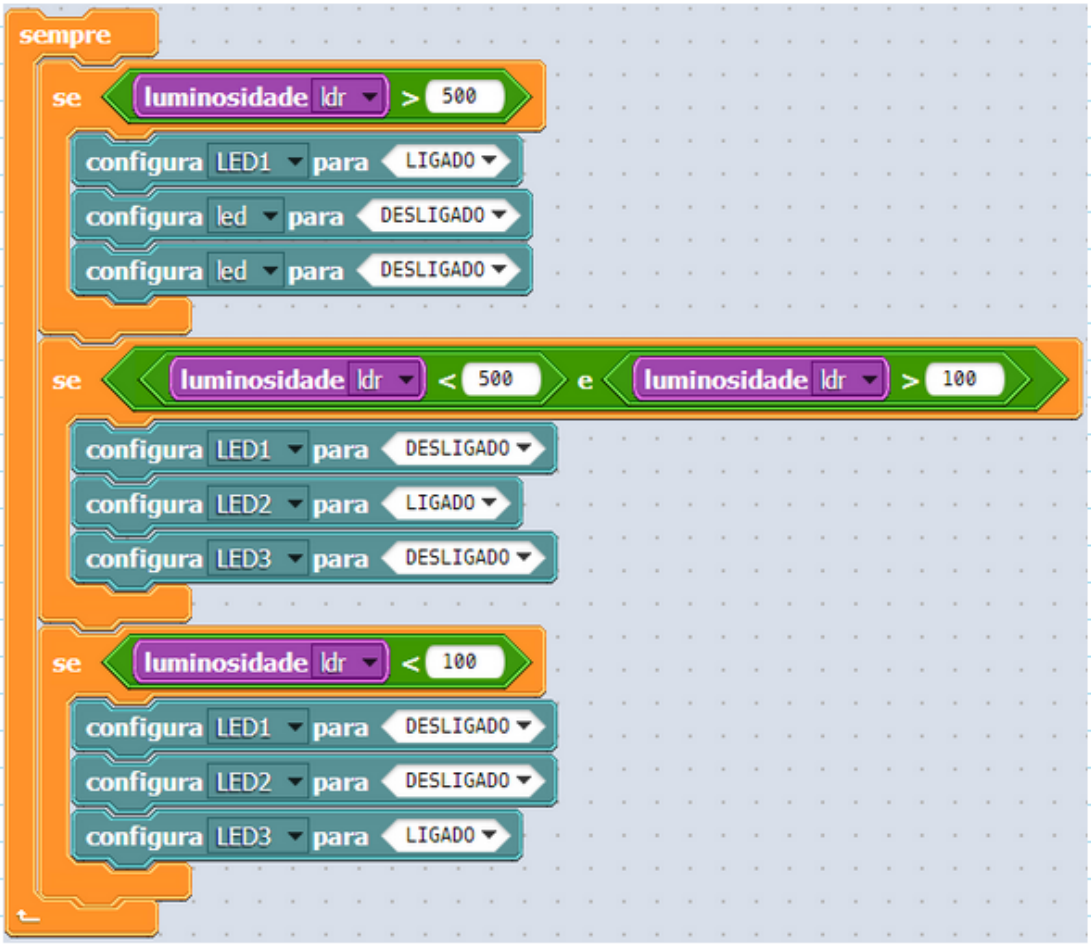
corrigir código - analisar + aplicar - constantes e variáveis, atribuição de valores, operações aritméticas

3 - Altere o programa acima para que a primeira espera seja a soma das variáveis Tempo1 e Tempo2, e a segunda espera seja a diferença entre as variáveis Tempo1 e Tempo2. *

modificar código - aplicar - constantes e variáveis, atribuição de valores, operações aritméticas

4 - Escreva um programa que faça um LED piscar mil vezes em sequência, com intervalo de um segundo entre cada piscada. *

escrever código - aplicar - constantes e variáveis, atribuição de valores, operações aritméticas, operações relacionais, condicionais



5 - Considerando o programa acima, em que casos cada LED vai acender? *

rodar código e dar respostas - validar - constantes e variáveis, atribuição de valores, operações aritméticas, operações relacionais, condicionais

Submit

Never submit passwords through Google Forms.

APÊNDICE C

Prova do Curso

Prova

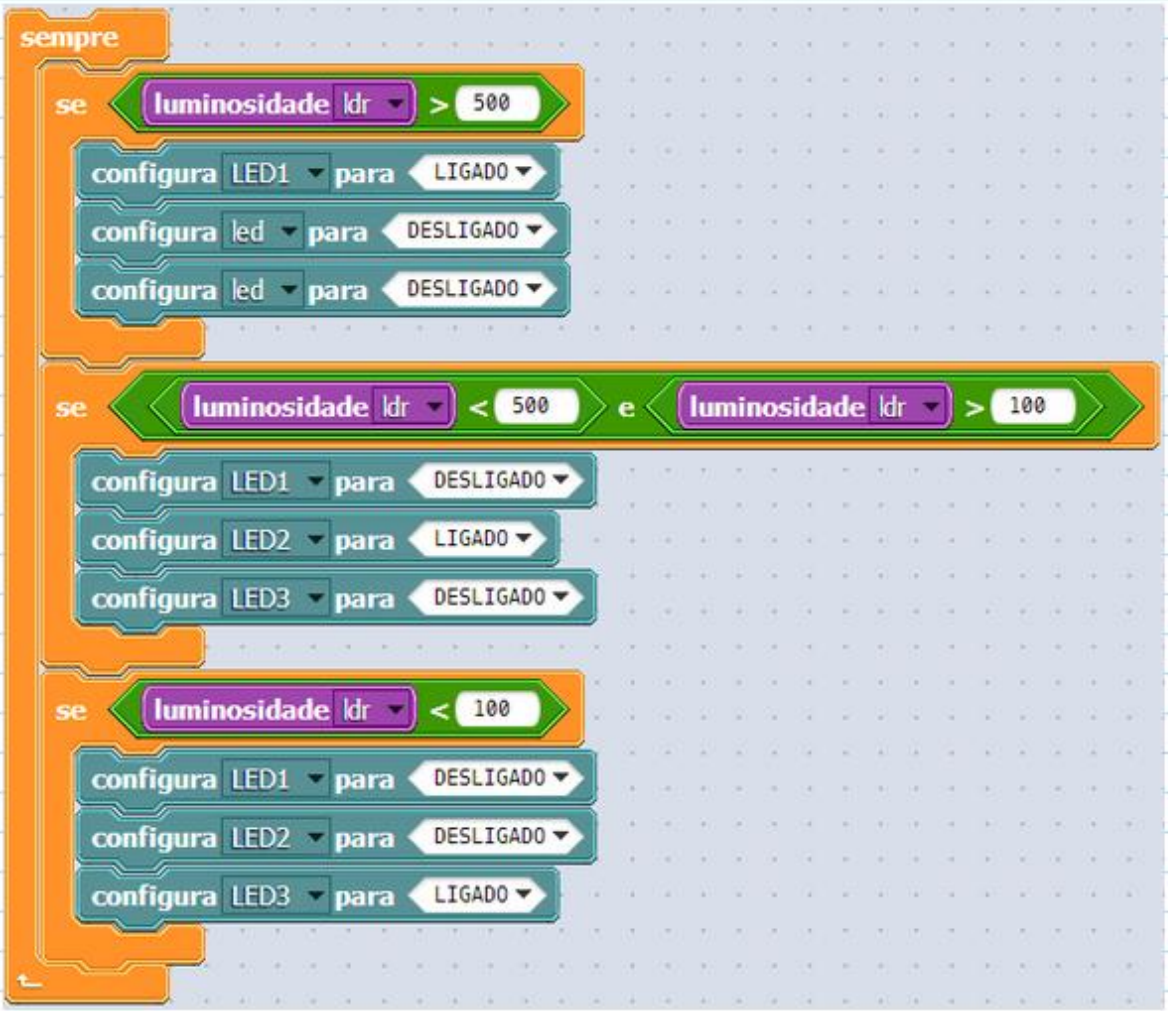
*Required

1 - Escreva um programa que contém cinco botões. Quando o primeiro botão é apertado, um led pisca uma vez com intervalo de 1 segundo entre cada piscada. Quando o segundo botão é apertado, o led pisca duas vezes com intervalo de 1 segundo entre cada piscada, e daí em diante, até o led piscar cinco vezes, quando o quinto botão for pressionado. *

2 - Conserte o programa acima para que o LED1 ligue. *

3 - Altere o programa acima para que a primeira espera seja a soma das variáveis Tempo1 e Tempo2, e a segunda espera seja a diferença entre as variáveis Tempo1 e Tempo2. *

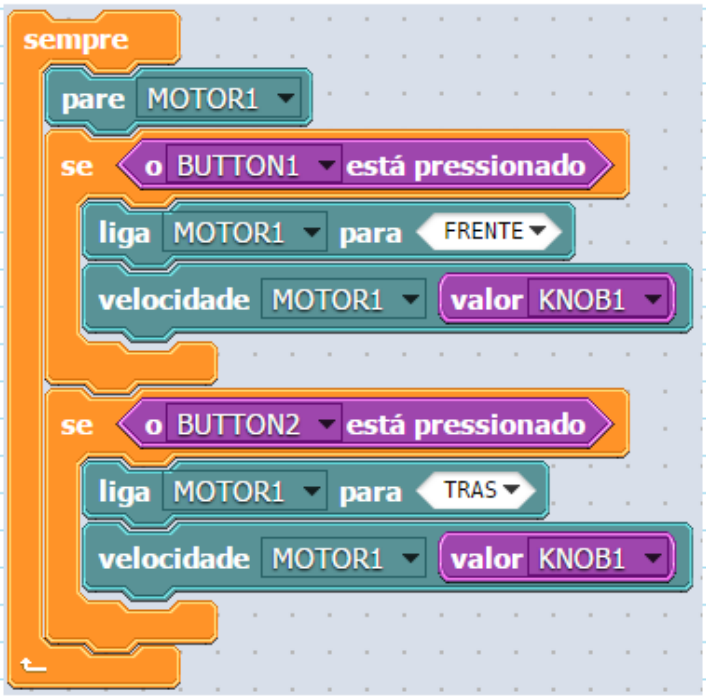
4 - Escreva um programa que faça um LED piscar mil vezes em sequência, com intervalo de um segundo entre cada piscada. Após as mil piscadas, uma buzina toca por 2 segundos e o programa reinicia. *



5 - Considerando o programa acima, em que casos cada LED vai acender? *

6 - Escreva um programa que mostra uma contagem regressiva de 9 a 0 no display de 7 segmentos. *

7 - Escreva um programa que só acende um led se o usuário acertar a senha (escolhida por você), solicitada via Monitor Serial. Se o usuário errar a senha três vezes, o programa trava (e xinga o invasor). *



8 - Explique o que o programa acima faz. *

Submit

Never submit passwords through Google Forms.

APÊNDICE D

Questões e comentários da prova da Oficina.

Questão 1

Enunciado: Escreva um programa que compara dois números escolhidos por você. Se o primeiro for maior que o segundo, acende o led1. Se o segundo for maior que o primeiro, acende o led2. Se a soma dos dois números for maior que 10, acende o led3.

Objetivo: escrever código.

Taxonomia: *Aplicar*.

Tópicos abordados: constantes e variáveis, atribuição de valores, operações aritméticas, condicionais.

Resultado e observações: a primeira questão da prova solicita a montagem de um algoritmo focado em uma estrutura de decisão condicional, baseado nos valores de dois parâmetros escolhidos arbitrariamente pelo aluno. Nesta questão a categoria da Taxonomia utilizada é *Aplicar*, pois tal algoritmo já havia sido visto durante as aulas. Portanto, embora seja necessário desenvolver um algoritmo, como tal programa já havia sido apresentado aos alunos, a questão não pode ser configurada como presente na categoria *Criar*. Nesta questão o domínio de operadores aritméticos e relacionais foi necessário, assim como o uso correto do bloco de decisão condicional. Conhecimento básico de montagem de circuitos também é necessário, uma vez que é preciso utilizar leds durante o programa. Os resultados da questão foram satisfatórios, uma vez que todos os alunos a responderam corretamente.

Questão 2

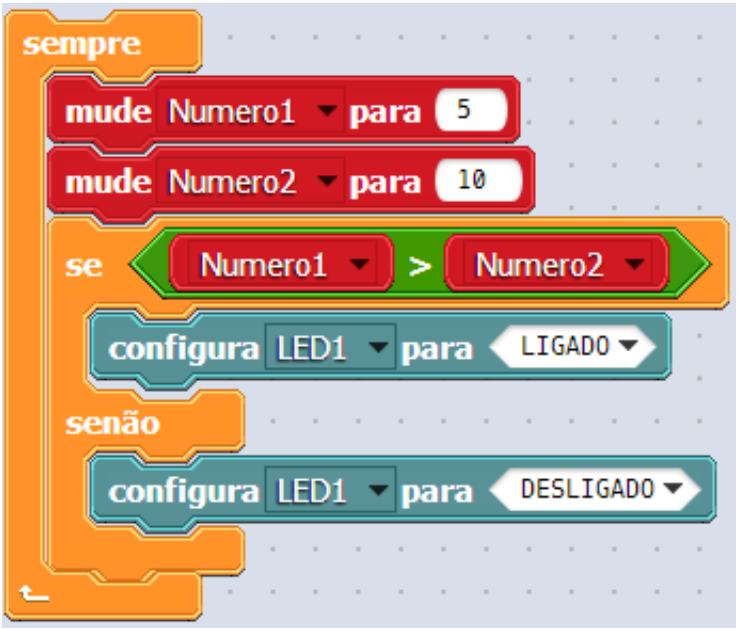


Figura 0.1: Imagem da questão 2

Enunciado: Conserte o programa acima para que o led1 ligue.

Objetivo: corrigir código.

Taxonomia: *Analisar* e *Aplicar*.

Tópicos abordados: constantes e variáveis, atribuição de valores, operações aritméticas, condicionais.

Resultado e observações: a questão apresenta um programa já construído e pede ao aluno para que realize uma correção, de modo que a condição no enunciado (o led1 ligar) seja satisfeita. Para isso, é necessário, primeiro, empregar a categoria analisar da taxonomia, para verificar como o programa funciona e onde a alteração deve ser feita. Após isso a modificação é efetuada, de acordo com o enunciado, utilizando a categoria aplicar. O algoritmo foi desenvolvido especificamente para oferecer três soluções distintas, todas corretas. Essa estratégia visa explorar os conhecimentos adquiridos pelos alunos e sua capacidade de análise crítica, já que as diferentes opções de resolução do problema os obriga a verificar se a solução encontrada realmente é a certa, uma vez que existem outras. Dois alunos solicitaram a atenção do professor, perguntando se a questão realmente continha mais de uma opção de

resposta, pois haviam achado duas ou mais. Todos os alunos acertaram essa questão, e as três soluções foram usadas nas repostas.

Questão 3

Figura 0.2: Imagem da questão 3

Enunciado: Altere o programa acima para que a primeira espera seja a soma das variáveis Tempo1 e Tempo2, e a segunda espera seja a diferença entre as variáveis Tempo1 e Tempo2.

Objetivo: modificar código.

Taxonomia: *Entender* e *Aplicar*.

Tópicos abordados: constantes e variáveis, atribuição de valores, operações aritméticas.

Resultado e observações: da mesma forma que a questão 2, a questão 3 apresenta um programa já construído, mas dessa vez é solicitada a alteração do escopo do programa. Essa questão foi mantida propositalmente simples, uma vez que o conceito de variáveis e constantes, embora seja fundamental e elementar no aprendizado de programação, requer uma abstração pouco familiar aos estudantes do primeiro ano do ensino médio (apenas mais tarde no ensino médio eles exercitarão o conceito de variáveis em equações matemáticas). Não

houveram problemas nessa questão, em que para chegar à solução correta o aluno deveria usar operadores aritméticos.

Questão 4

Enunciado: Escreva um programa que faça um led piscar mil vezes em sequência, com intervalo de um segundo entre cada piscada.

Objetivo: escrever código.

Taxonomia: *Criar*.

Tópicos abordados: constantes e variáveis, atribuição de valores, laços.

Resultado e observações: ao solicitar um programa que repete uma instrução um número determinado de vezes, ou seja, uma estrutura composta de um laço finito, a questão coagiu os alunos a usarem um laço de repetição *while* ou *for* para conseguir construir o programa corretamente. Como mencionado anteriormente, o conceito de laço de repetição foi o principal problema de aprendizado da turma, embora o próprio Arduino funcione baseado em um eterno laço (a função *loop()*), e isso tenha sido bem captado por todos. Nessa questão, sete dos dez alunos responderam corretamente, isto é, a resposta exata. Um aluno utilizou a função *blink* para simular o objetivo da questão: fazer um led piscar mil vezes. Embora essa solução trouxesse o efeito desejado, o conceito de laço de repetição não foi utilizado. Um aluno pareceu não ter entendido o objetivo da questão, e outro não escreveu nada e não adquiriu pontos nesta questão.

Questão 5

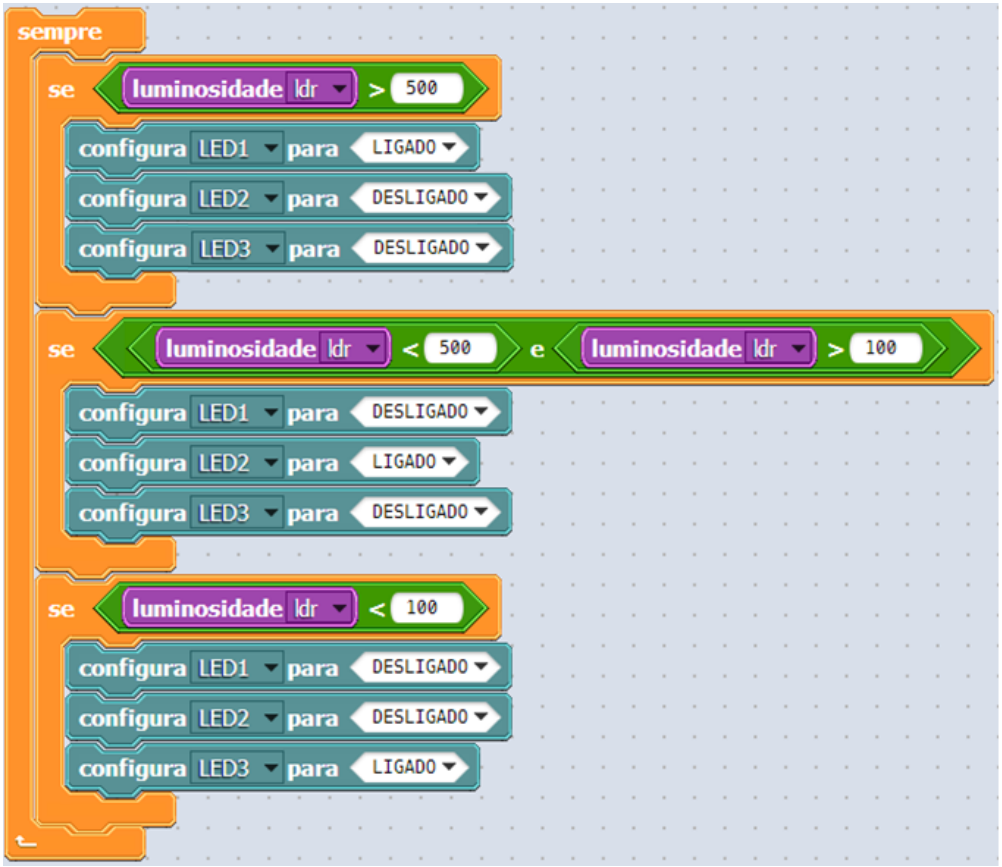


Figura 0.3: Imagem da questão 5

Enunciado: Considerando o programa acima, em que casos cada led vai acender?

Objetivo: descrever código.

Taxonomia: *Entender*.

Tópicos abordados: constantes e variáveis, atribuição de valores, operações aritméticas, operações relacionais, operações lógicas, condicionais.

Resultado e observações: o programa apresentado, embora extenso, mostra um algoritmo relativamente simples, cujo conhecimento sobre condicionais e operadores relacionais e lógicos é essencial para o seu entendimento. A questão pede a leitura e compreensão do seu funcionamento ao solicitar em quais casos, em função do valor da variável luminosidade, cada led que faz parte do circuito deverá acender. Neste caso, o aluno precisa primeiro detectar as estruturas condicionais, colocadas visualmente no exterior do código, e então agregar as

condições dadas em cada grupo de instruções, até montar todas as possibilidades do programa. Oito dos dez alunos responderam com êxito essa questão. Um dos alunos confundiu a resposta da segunda condição (“o led 1 desligado, led 2 desligado assim como o led 3”, quando o led 2 deveria estar ligado) e outro mostrou falta de compreensão com o funcionamento do componente eletrônico, afetando negativamente sua resposta.

APÊNDICE E

Questões e comentários da prova do Curso.

Questão 1

Enunciado: Escreva um programa que contém cinco botões. Quando o primeiro botão é apertado, um led pisca uma vez com intervalo de 1 segundo entre cada piscada. Quando o segundo botão é apertado, o led pisca duas vezes, e daí em diante, até o led piscar cinco vezes, quando o quinto botão for pressionado.

Objetivo: escrever código.

Taxonomia abordada: C*riar*.

Tópicos abordados: funções, condicionais.

Resultado e observações: a primeira questão tinha como objetivo principal instigar os alunos a usar funções em seus programas, uma vez que seu uso tornaria a construção muito mais simples. Ao precisar programar as ações idênticas dos cinco botões, com alteração apenas no número de vezes que o led pisca, os alunos perceberam sua utilidade e empregaram funções em seus programas. O conceito de condicional também foi necessário para administrar o que acontecia quando cada botão era pressionado. Todos os alunos projetaram seus programas com êxito, e não houveram problemas aqui.

Questão 2

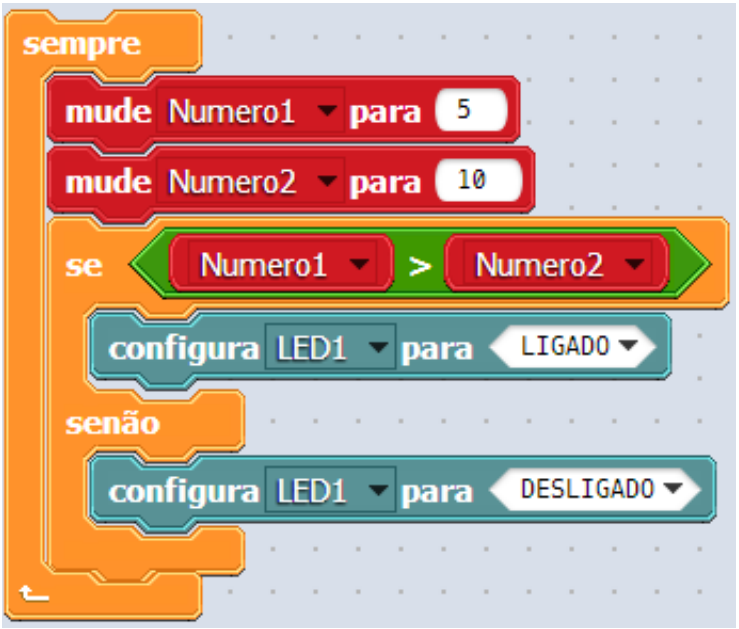


Figura 0.1: Imagem da questão 2

Enunciado: Conserte o programa acima para que o led1 ligue.

Objetivo: corrigir código.

Taxonomia abordada: *Analisar*.

Tópicos abordados: constantes e variáveis, atribuição de valores, operações aritméticas.

Resultado e observações: a questão apresenta um programa já construído e pede ao aluno que realize uma correção, de modo que a condição no enunciado (o led1 ligar) seja satisfeita. Para isso, é necessário, primeiro, empregar a categoria analisar da taxonomia, para verificar como o programa funciona e onde a alteração deve ser feita. Após isso a modificação é efetuada, de acordo com o enunciado, utilizando a categoria aplicar. O algoritmo foi desenvolvido especificamente para oferecer três soluções distintas, todas corretas. Essa estratégia visa explorar os conhecimentos adquiridos pelos alunos e sua capacidade de análise crítica, já que as diferentes opções de resolução do problema os obriga a verificar se a solução encontrada realmente é a certa, uma vez que existem outras. Três alunos solicitaram a atenção do professor, perguntando se a questão realmente continha mais de uma opção de resposta, pois haviam achado duas ou mais. Todos os alunos acertaram essa questão, e as três soluções foram usadas nas repostas.

Questão 3

Figura 0.2: Imagem da questão 3

Enunciado: Altere o programa acima para que a primeira espera seja a soma das variáveis Tempo1 e Tempo2, e a segunda espera seja a diferença entre as variáveis Tempo1 e Tempo2.

Objetivo: modificar código.

Taxonomia abordada: *Aplicar*.

Tópicos abordados: constantes e variáveis, atribuição de valores, operações aritméticas.

Resultado e observações: da mesma forma que a questão 2, a questão 3 apresenta um programa já construído, mas dessa vez é solicitada a alteração do escopo do programa. Essa questão foi mantida propositalmente simples, uma vez que o conceito de variáveis e constantes, embora seja fundamental e elementar no aprendizado de programação, requer uma abstração pouco familiar aos estudantes do primeiro ano do ensino médio (apenas mais tarde no ensino médio eles exercitarão o conceito de variáveis em equações matemáticas). Não houveram problemas nessa questão, em que para chegar à solução correta o aluno deveria usar operadores aritméticos.

Questão 4

Enunciado: Escreva um programa que faça um led piscar mil vezes em sequência, com intervalo de um segundo entre cada piscada. Após as mil piscadas, uma buzina toca por 2 segundos.

Objetivo: escrever código.

Taxonomia abordada: C*riar*.

Tópicos abordados: constantes e variáveis, atribuição de valores, operações aritméticas, operações relacionais, laços.

Resultado e observações: ao solicitar um programa que repete uma instrução um número determinado de vezes, ou seja, uma estrutura composta de um laço finito, a intenção da questão era que os alunos trabalhassem com um laço de repetição *while* ou *for* para conseguir construir o programa corretamente. Como mencionado anteriormente, o conceito de laço de repetição foi o principal problema de aprendizado da turma, embora o próprio Arduino funcione baseado em um eterno laço (a função *loop()*), e isso tenha sido bem captado por todos. Nessa questão, apenas cinco dos oito alunos responderam corretamente, isto é, a resposta exata. Dois alunos construíram o laço acertadamente, porém realizaram 100 iterações, e não 1000, conforme solicitado no enunciado da questão. Essas duas respostas receberam meio certo na correção, pois demonstraram conhecer o funcionamento de um laço de repetição, e seu erro pode ser atribuído à falta de atenção, pois segundo eles mesmos “só faltou um zero”. Um aluno falhou em demonstrar conhecimento na definição de laços de repetição e não adquiriu pontos nesta questão.

Questão 5

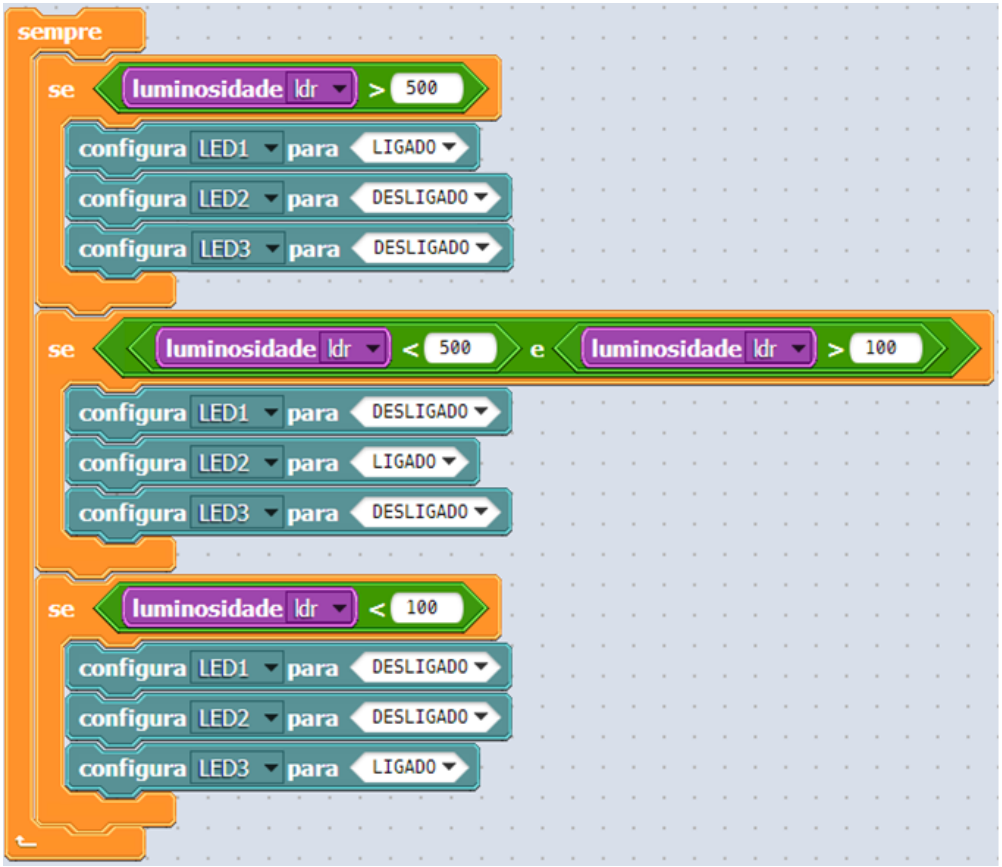


Figura 0.3: Imagem da questão 5

Enunciado: Considerando o programa acima, em que casos cada led vai acender?

Objetivo: executar código e dar respostas.

Taxonomia abordada: *Aplicar*.

Tópicos abordados: constantes e variáveis, atribuição de valores, operações aritméticas, operações relacionais, condicionais.

Resultado e observações: o programa apresentado, embora extenso, mostra um algoritmo relativamente simples, cujo conhecimento sobre condicionais e operadores relacionais e lógicos é essencial para o seu entendimento. A questão pede a leitura e compreensão do seu funcionamento ao solicitar em quais casos, em função do valor da variável luminosidade, cada led que faz parte do circuito deverá acender. Neste caso, o aluno precisa primeiro detectar as estruturas condicionais, colocadas visualmente no exterior do código, e então agregar as

condições dadas em cada grupo de instruções, até montar todas as possibilidades do programa. Todos os alunos responderam com êxito essa questão.

Questão 6

Enunciado: Escreva um programa que mostra uma contagem regressiva de 9 a 0 no display de 7 segmentos.

Objetivo: escrever código.

Taxonomia abordada: *Criar*.

Tópicos abordados: constantes e variáveis, atribuição de valores, laços.

Resultado e observações: essa questão obteve resultados muito curiosos, pois embora todos os alunos tenham conseguido escrever programas cujo comportamento demonstrasse o esperado, uma vez executado em um circuito eletrônico, nenhuma resposta foi considerada plenamente certa, uma vez que todos falharam em utilizar a competência necessária para tal. O problema pedia o desenvolvimento de um código que gerasse uma contagem regressiva, mostrada em um display de sete segmentos, começando do número nove e terminando no número zero. Programadores com certo grau de experiência reconhecem o padrão aqui exibido, e usam de estruturas de repetição para resolver rapidamente o problema: um laço decrescente contando de nove a zero, cujo próprio contador poderia servir de variável a ser mostrada no display. No entanto, os alunos unanimemente escreveram seus códigos ignorando qualquer tipo de solução com o uso de laços de repetição, e escreveram, linha a linha, todas as iterações do possível laço, começando em nove e terminando em zero. Talvez isso tenha ocorrido devido à natureza visual do ambiente de programação por blocos e sua facilidade de montar códigos repetitivos. Se os alunos estivessem programando no ambiente nativo do Arduino, de forma textual, é possível que outras opções - entre elas o laço de repetição - fossem exploradas antes de eleger uma como a mais apta a ser usada. É possível inferir que a solução alternativa, além de não ser custosa de se construir, era cognitivamente muito mais simples, e, portanto, foi utilizada.

Questão 7

Enunciado: Escreva um programa que só acende um led se o usuário acertar a senha (escolhida por você), solicitada via Monitor Serial. Se o usuário errar a senha três vezes, o programa trava.

Objetivo: escrever código.

Taxonomia abordada: *Aplicar*.

Tópicos abordados: constantes e variáveis, atribuição de valores, operações aritméticas, operações relacionais, condicionais.

Resultado e observações: essa questão é considerada pelo autor a mais difícil do teste pois para chegar ao algoritmo correto o aluno precisa demonstrar não só conhecimentos de programação e robótica, como também saber como o próprio Arduino e o ambiente DuinoBlocks funcionam. Ao pedir aos alunos para que criem programa com comunicação via monitor serial, uma série de blocos essenciais, com ordem de montagem específica, precisam ser incorporados nos lugares certos do código. Qualquer montagem equivocada resulta em um programa incorreto. O aluno precisa então primeiro lembrar (primeira categoria da taxonomia) quais blocos precisam ser usados e em que ordem, para então poder desenvolver um algoritmo que resolva corretamente o problema (categoria *Aplicar*, da Taxonomia). Nessa questão, cinco alunos elaboraram respostas perfeitas e três respostas parcialmente corretas. Os três alunos que responderam incorretamente mostraram certo grau de confusão ao usar os blocos destinados à montagem da comunicação serial do Arduino. Isso pode significar uma deficiência na categoria *Lembrar*, a primeira da taxonomia, e que, por definição, é o alicerce do conhecimento sobre um tópico específico. Ao demonstrar essa deficiência, não poderíamos esperar que o restante do algoritmo fosse montado corretamente. Após a correção do teste, conversas com esses alunos explicitaram a origem do problema: eles haviam faltado a aula em que esse material foi apresentado, minando então seu conhecimento acerca da questão.

Questão 8

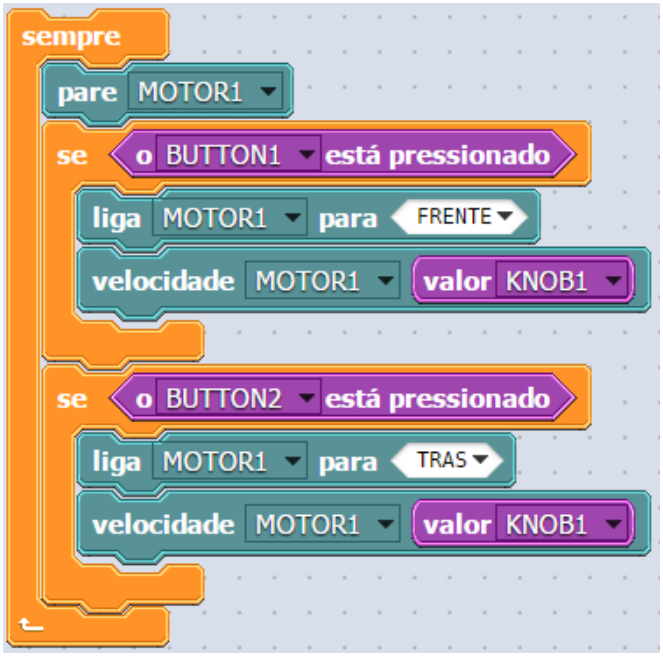


Figura 0.4: Imagem da questão 8

Enunciado: Explique o que o programa acima faz.

Objetivo: descrever código.

Taxonomia abordada: *Entender*.

Tópicos abordados: condicionais.

Resultado e observações: a última questão da prova pede uma leitura completa pelo programa mostrado na imagem. Embora ela gere uma quantidade relativamente grande de código escrito, essa questão é um bom exemplo de como a linguagem de programação por blocos pode ser simples, quando o objetivo é ater-se à semântica e à construção do algoritmo. O tipo de questão induz o uso da categoria *Entender*, uma vez que é preciso primeiro lembrar qual é a função dos blocos individualmente e, posteriormente, como eles se comportam em conjunto para gerar um programa coerente. Esse programa remete aos experimentos feitos com um carrinho de controle remoto, durante o curso, e mostra a tentativa do autor de constantemente buscar exercícios e atividades lúdicas ou que cativem os alunos. Nessa questão, todos os alunos conseguiram responder corretamente.

APÊNDICE F

Formulário de inscrição no Curso de Robótica, disponibilizado para os alunos do Colégio Pedro II.

Curso de Programação e Robótica - Colégio Pedro II / Tijuca

Este é um curso VOLUNTÁRIO oferecido pelo Colégio Pedro II, que faz parte de uma pesquisa da UFRJ, com a supervisão da profª. Viviane Rodrigues Silva.

*Required

Nome *

Data de nascimento *

dd/mm/aaaa

Telefone de contato *

E-mail de contato *

Escola *

Série escolar *

Quantas vezes por semana você usa um computador? *

- Nunca
- Menos de uma vez por semana
- Uma vez por semana
- De 2 a 4 vezes por semana
- Praticamente todos os dias

Você sabe programar? *

- ○ Sim
- ○ Não
- ○ Estou aprendendo

Se você já sabe programar, onde aprendeu?

- ○ Em casa
- ○ Na escola
- ○ Em um curso fora da escola
- ○ Other:

Você já usou, montou ou programou um robô antes? *

- ○ Não
- ○ Sim

Se já usou/montou/programou um robô antes, conte onde e como foi

Você acha que tecnologia é... *

- ○ Muito fácil
- ○ Fácil
- ○ Nem fácil e nem difícil
- ○ Difícil
- ○ Muito difícil

Você acha que programar computadores é... *

- ○ Muito fácil
- ○ Fácil
- ○ Nem fácil e nem difícil
- ○ Difícil
- ○ Muito difícil

Você acha que programar um robô é... *

- ○ Muito fácil
- ○ Fácil
- ○ Nem fácil e nem difícil
- ○ Difícil
- ○ Muito difícil

Você acha que construir um robô é... *

- ○ Muito fácil
- ○ Fácil
- ○ Nem fácil e nem difícil
- ○ Difícil
- ○ Muito difícil

O que você espera aprender no curso de programação com robótica? *

Escreva quais são as habilidades que você espera aprender ou melhorar

Submit

Never submit passwords through Google Forms.

APÊNDICE G

Cartaz de divulgação do Curso de Robótica no Colégio Pedro II

APÊNDICE H

Componentes do Kit Arduino usado nas aplicações:

- 01 Arduino UNO R3
- 01 Protoboard 840
- 01 Cabo USB
- 01 Sensor de Temperatura (LM35)
- 01 Sensor de Luminosidade (LDR 5mm)
- 01 Potenciômetro 10k
- 01 Barra Gráfica de LEDs
- 01 Display de 7 Segmentos
- 01 Circuito integrado 4511
- 01 Pastilha Piezoelétrica
- 04 Chave Momentânea (PushButton)
- 05 LEDs Amarelos
- 05 LEDs Verdes
- 05 LEDs Vermelhos
- 01 LED alto brilho
- 15 Resistores 300Ω
- 05 Resistores 10kΩ
- 05 Resistores 1MΩ
- 01 Buzzer
- 01 Display de LCD 16x2 com backlight
- 20 fios Jumper de 20 cm
- 10 fios Jumper de 10 cm
- 01 Caixa Organizadora

APÊNDICE I

Enunciados e respostas da prova aplicada no Curso.

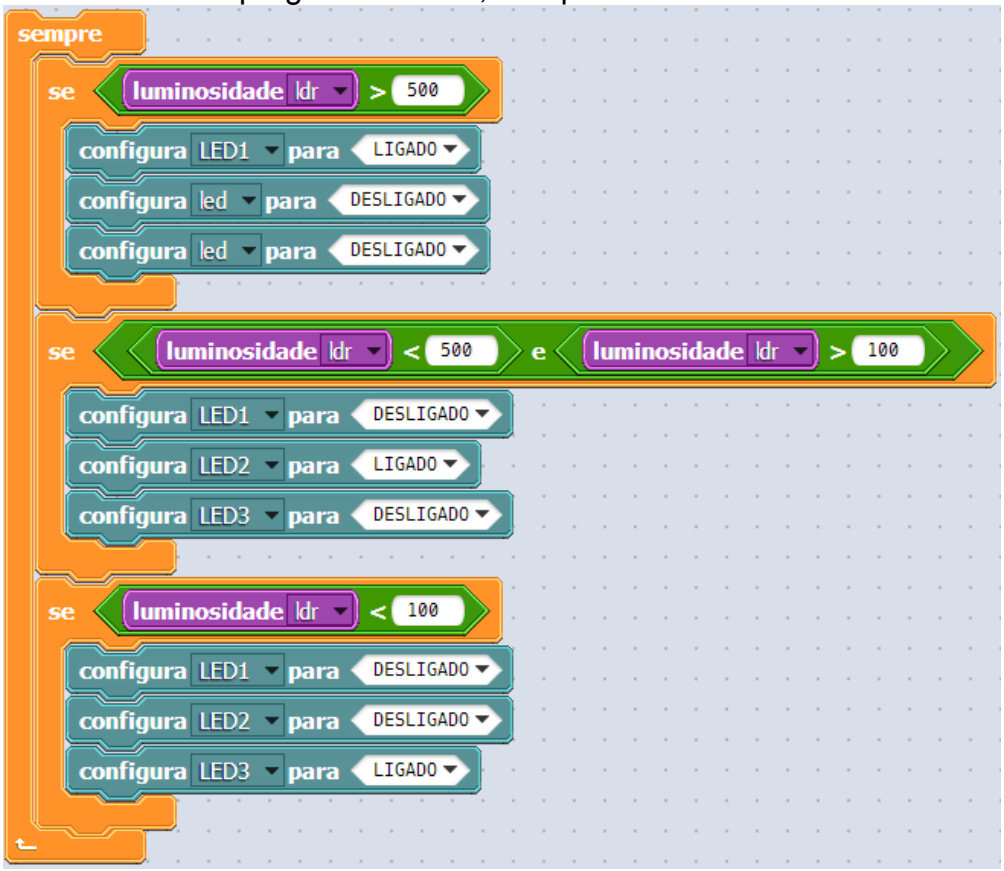
Enunciados
Escreva um programa que contém cinco botões. Quando o primeiro botão é apertado, um led pisca uma vez com intervalo de 1 segundo entre cada piscada. Quando o segundo botão é apertado, o led pisca duas vezes com intervalo de 1 segundo entre cada piscada, e daí em diante, até o led piscar cinco vezes, quando o quinto botão for pressionado.

Conserte o programa abaixo para que o LED1 ligue.

Altere o programa abaixo para que a primeira espera seja a soma das variáveis Tempo1 e Tempo2, e a segunda espera seja a diferença entre as variáveis Tempo1 e Tempo2.

Escreva um programa que faça um LED piscar mil vezes em sequência, com intervalo de um segundo entre cada piscada. Após as mil piscadas, uma buzina toca por 2 segundos e o programa reinicia.

Considerando o programa abaixo, em que casos cada LED vai acender?



Escreva um programa que mostra uma contagem regressiva de 9 a 0 no display de 7 segmentos.

Escreva um programa que só acende um led se o usuário acertar a senha (escolhida por você), solicitada via Monitor Serial. Se o usuário errar a senha três vezes, o programa trava (e xinga o invasor).

Explique o que o programa abaixo faz.

Aluno 1
Questão 1

```
const int BUTTON1 = 2;
const int LED1 = 7;
void blinkLed(int ledPin, int qty, int time){
for (int i = 0; i < qty; i++) {
digitalWrite(ledPin, HIGH);
delay((500*time)/qty);
digitalWrite(ledPin, LOW);
delay((500*time)/qty);
}
}
const int BUTTON2 = 3;
const int BUTTON3 = 4;
const int BUTTON4 = 5;
const int BUTTON5 = 6;

void setup(){
pinMode(BUTTON1, INPUT);
pinMode(BUTTON2, INPUT);
pinMode(BUTTON3, INPUT);
pinMode(BUTTON4, INPUT);
pinMode(BUTTON5, INPUT);
pinMode(LED1, OUTPUT);
}

void loop(){
```

```
if(digitalRead(BUTTON1) == LOW){
blinkLed(LED1, 1, 1);
}
if(digitalRead(BUTTON2) == LOW){
blinkLed(LED1, 2, 2);
}
if(digitalRead(BUTTON3) == LOW){
blinkLed(LED1, 3, 3);
}
if(digitalRead(BUTTON4) == LOW){
blinkLed(LED1, 4, 4);
}
if(digitalRead(BUTTON5) == LOW){
blinkLed(LED1, 5, 5);
}
}
```

Questão 2

```
float usr1_numero1;
float usr2_numero2;
const int LED1 = 7;

void setup(){
pinMode(LED1, OUTPUT);
}

void loop(){
usr1_numero1 = 5;
usr2_numero2 = 10;
if((usr1_numero1 < usr2_numero2)){
digitalWrite(LED1, HIGH);
}
else{
digitalWrite(LED1, LOW);
}
}
```

Questão 3

```
float usr1_tempo1;
float usr2_tempo2;
const int LED1 = 7;

void setup(){
pinMode(LED1, OUTPUT);
}

void loop(){
usr1_tempo1 = 1000;
usr2_tempo2 = 2000;
digitalWrite(LED1, HIGH);
delay((usr1_tempo1 + usr2_tempo2));
```

```
digitalWrite(LED1, LOW);
delay((usr2_tempo2 - usr1_tempo1));
}
```

Questão 4

```
float usr1_repetir;
int syst1;
const int LED1 = 7;
void usr2_blink(){
digitalWrite(LED1, HIGH);
delay(1000);
digitalWrite(LED1, LOW);
delay(1000);
}

const int BUZZER1 = 8;

void setup(){
pinMode(BUZZER1, INPUT);
pinMode(LED1, OUTPUT);
}

void loop(){
usr1_repetir = 1000;
for (syst1 = 0; syst1 < (usr1_repetir); ++syst1){
usr2_blink();
}
tone(BUZZER1, 33, 2);
}
```

Questão 5

Se a luminosidade do sensor de luz for mais que 500 o led 1 irá acender. Se a luminosidade for menor que 500 e maior que 100 o led dois irá acender. Já o led 3 só irá caso a luminosidade for menor que 100.

Questão 6

```
byte DISPLAY7SEG1[7] = {2, 3, 4, 5, 6, 7, 8};

byte seven_seg_digits[10][7] = {{1,1,1,1,1,1,0},
{0,1,1,0,0,0,0}, {1,1,0,1,1,0,1}, {1,1,1,1,0,0,1},
{0,1,1,0,0,1,1}, {1,0,1,1,0,1,1}, {1,0,1,1,1,1,1},
{1,1,1,0,0,0,0}, {1,1,1,1,1,1,1}, {1,1,1,0,0,1,1}};

void sevenSegWrite(byte digit, byte array[]) {
for (byte i = 0; i < 7; i++)
digitalWrite(array[i], seven_seg_digits[digit][i]);
}

void sevenSegClear(byte array[]) {
for (byte i = 0; i < 7; i++)
digitalWrite(array[i], 0);
}
```

```
void setup(){
for (byte i = 0; i < 7; i++)
pinMode(DISPLAY7SEG1[i], OUTPUT);
}

void loop(){
sevenSegWrite(9, DISPLAY7SEG1);
delay(1000);
sevenSegWrite(8, DISPLAY7SEG1);
delay(1000);
sevenSegWrite(7, DISPLAY7SEG1);
delay(1000);
sevenSegWrite(6, DISPLAY7SEG1);
delay(1000);
sevenSegWrite(5, DISPLAY7SEG1);
delay(1000);
sevenSegWrite(4, DISPLAY7SEG1);
delay(1000);
sevenSegWrite(3, DISPLAY7SEG1);
delay(1000);
sevenSegWrite(2, DISPLAY7SEG1);
delay(1000);
sevenSegWrite(1, DISPLAY7SEG1);
delay(1000);
sevenSegWrite(0, DISPLAY7SEG1);
}
Questão 7
float usr1_erros;
String syst_getMsg(){
String content = "";
char character;
while(Serial.available()){
character = Serial.read();
content.concat(character);
delay(10);
}
return content;
}
void syst_waitForTheMsg(String s){
do{ while(!Serial.available());
}
while(syst_getMsg() != s);
} const int LED1 = 9;
void setup(){
pinMode(LED1, OUTPUT);
usr1_erros = 3;
Serial.begin(9600);
}
```

```
void loop(){
syst_waitForTheMsg("cu");
if(Serial.available()){
digitalWrite(LED1, HIGH);
} else{ usr1_erros += 1; }
if((usr1_erros == 3)){
Serial.println("rioso");
while (!(false)){ }
} }
```

Questão 8

Dependendo do botão que está pressionado o motor (carrinho) se movimenta para frente ou pra trás em x velocidade. E se nenhum botão está pressionado o motor (carrinho) não se movimenta.

Aluno 2

Questão 1

```
const int BUTTON1 = 2;
const int LED1 = 7;
void blinkLed(int ledPin, int qty, int time){
for (int i = 0; i < qty; i++) {
digitalWrite(ledPin, HIGH);
delay((500*time)/qty);
digitalWrite(ledPin, LOW);
delay((500*time)/qty);
}
}
const int BUTTON2 = 3;
const int BUTTON4 = 5;
const int BUTTON5 = 6;

void setup(){
pinMode(BUTTON1, INPUT);
pinMode(BUTTON2, INPUT);
pinMode(BUTTON4, INPUT);
pinMode(BUTTON5, INPUT);
pinMode(LED1, OUTPUT);
}

void loop(){
if(digitalRead(BUTTON1) == LOW){
blinkLed(LED1, 1, 1);
}
if(digitalRead(BUTTON2) == LOW){
blinkLed(LED1, 2, 2);
}
if(digitalRead(BUTTON2) == LOW){
blinkLed(LED1, 3, 3);
}
if(digitalRead(BUTTON4) == LOW){
blinkLed(LED1, 4, 4);
```

```
}
if(digitalRead(BUTTON5) == LOW){
blinkLed(LED1, 5, 5);
}
}
Questão 2
float usr1_numero1;
float usr2_Numero2;
const int LED1 = 7;

void setup(){
pinMode(LED1, OUTPUT);
usr1_numero1 = 5;
usr2_Numero2 = 10;
}

void loop(){
if((usr1_numero1 < usr2_Numero2)){
digitalWrite(LED1, HIGH);
}
else{
digitalWrite(LED1, LOW);
}
}
Questão 3
float usr1_Tempo1;
float usr2_Tempo2;
const int LED1 = 7;

void setup(){
pinMode(LED1, OUTPUT);
}

void loop(){
usr1_Tempo1 = 1000;
usr2_Tempo2 = 2000;
digitalWrite(LED1, HIGH);
delay((usr1_Tempo1 + usr2_Tempo2));
digitalWrite(LED1, LOW);
delay((usr2_Tempo2 - usr1_Tempo1));
}
Questão 4
int syst1;
const int BUZZER1 = 2;

void setup(){
pinMode(BUZZER1, INPUT);
pinMode(7, OUTPUT);
}
```

```
void loop(){
for (syst1 = 0; syst1 < (100); ++syst1){
digitalWrite(7, HIGH);
}
tone(BUZZER1, 33, 2);
}
```

Questão 5

O led1 vai acender se a incidência de luz no ldr for maior que 500

O led2 vai acender se a incidência de luz no ldr for menor que 500 e maior que 100

O led3 vai acender se a incidência de luz no ldr for menor que 100

Questão 6

```
byte DISPLAY7SEG1[7] = {2, 3, 4, 5, 6, 7, 8};

byte seven_seg_digits[10][7] = {{1,1,1,1,1,1,0},
{0,1,1,0,0,0,0}, {1,1,0,1,1,0,1}, {1,1,1,1,0,0,1},
{0,1,1,0,0,1,1}, {1,0,1,1,0,1,1}, {1,0,1,1,1,1,1},
{1,1,1,0,0,0,0}, {1,1,1,1,1,1,1}, {1,1,1,0,0,1,1}};

void sevenSegWrite(byte digit, byte array[]) {
for (byte i = 0; i < 7; i++)
digitalWrite(array[i], seven_seg_digits[digit][i]);
}

void sevenSegClear(byte array[]) {
for (byte i = 0; i < 7; i++)
digitalWrite(array[i], 0);
}

void setup(){
for (byte i = 0; i < 7; i++)
pinMode(DISPLAY7SEG1[i], OUTPUT);
}

void loop(){
sevenSegWrite(9, DISPLAY7SEG1);
delay(1000);
sevenSegWrite(8, DISPLAY7SEG1);
delay(1000);
sevenSegWrite(7, DISPLAY7SEG1);
delay(1000);
sevenSegWrite(6, DISPLAY7SEG1);
delay(1000);
sevenSegWrite(5, DISPLAY7SEG1);
delay(1000);
sevenSegWrite(4, DISPLAY7SEG1);
delay(1000);
sevenSegWrite(3, DISPLAY7SEG1);
```

```
delay(1000);
sevenSegWrite(2, DISPLAY7SEG1);
delay(1000);
sevenSegWrite(1, DISPLAY7SEG1);
delay(1000);
sevenSegWrite(0, DISPLAY7SEG1);
}
```

Questão 7

```
String syst_getMsg(){ String content = ""; char character;
while(Serial.available()){ character = Serial.read(); content.concat(character);
delay(10); } return content; } float usr1_senha; void setup(){ pinMode(7,
OUTPUT); Serial.begin(9600); usr1_senha = 0; }
void loop(){
Serial.println("Digite a senha:");
if((syst_getMsg() == "100")){
digitalWrite(7, HIGH);
Serial.println("acertou"); }
else{ Serial.println("errou"); }
usr1_senha += 1;
if((syst_getMsg() == "100")){
digitalWrite(7, HIGH);
} else{ Serial.println("errou"); }
usr1_senha += 2;
if((syst_getMsg() == "100")){
digitalWrite(7, HIGH);
} else{ Serial.println("errou"); }
usr1_senha += 3;
if((usr1_senha == 3)){ while (!(false)){ } } }
```

Questão 8

Se o botão 1 for pressionado o motor gira para para frente
e se o botão 2 for pressionado o motor gira para trás

Aluno 3

Questão 1

```
const int BUTTON1 = 2;
const int LED1 = 7;
void blinkLed(int ledPin, int qty, int time){
for (int i = 0; i < qty; i++) {
digitalWrite(ledPin, HIGH);
delay((500*time)/qty);
digitalWrite(ledPin, LOW);
delay((500*time)/qty);
}
}
const int BUTTON2 = 3;
const int BUTTON3 = 4;
const int BUTTON4 = 5;
const int BUTTON5 = 6;

void setup(){
```

```
pinMode(BUTTON1, INPUT);
pinMode(BUTTON2, INPUT);
pinMode(BUTTON3, INPUT);
pinMode(BUTTON4, INPUT);
pinMode(BUTTON5, INPUT);
pinMode(LED1, OUTPUT);
}

void loop(){
if(digitalRead(BUTTON1) == LOW){
blinkLed(LED1, 1, 1);
}
if(digitalRead(BUTTON2) == LOW){
blinkLed(LED1, 2, 1);
}
if(digitalRead(BUTTON3) == LOW){
blinkLed(LED1, 3, 1);
}
if(digitalRead(BUTTON4) == LOW){
blinkLed(LED1, 4, 1);
}
if(digitalRead(BUTTON5) == LOW){
blinkLed(LED1, 5, 1);
}
}
Questão 2
float usr1_numero1;
float usr2_numero2;
const int LED1 = 7;

void setup(){
pinMode(LED1, OUTPUT);
}

void loop(){
usr1_numero1 = 5;
usr2_numero2 = 10;
if((usr1_numero1 < usr2_numero2)){
digitalWrite(LED1, HIGH);
}
else{
digitalWrite(LED1, LOW);
}
}
Questão 3
float usr1_tempo1;
float usr2_tempo2;
const int LED1 = 7;
```

```
void setup(){
pinMode(LED1, OUTPUT);
}

void loop(){
usr1_tempo1 = 1000;
usr2_tempo2 = 2000;
digitalWrite(LED1, HIGH);
delay((usr1_tempo1 + usr2_tempo2));
digitalWrite(LED1, LOW);
delay((usr1_tempo1 - usr2_tempo2));
}
```

Questão 4

```
int syst1;
const int LED1 = 3;
const int BUZZER1 = 2;

void setup(){
pinMode(BUZZER1, INPUT);
pinMode(LED1, OUTPUT);
}

void loop(){
for (syst1 = 0; syst1 < (1000); ++syst1){
digitalWrite(LED1, HIGH);
delay(1000);
digitalWrite(LED1, LOW);
delay(1000);
}
tone(BUZZER1, 131, 2000);
}
```

Questão 5

led1- quando a luminosidade for maior que 500
led2- quando a luminosidade for menor que 500 e maior que 100
led3- quando a luminosidade for menor que 100

Questão 6

```
float usr1_contagem;
byte DISPLAY7SEG1[7] = {2, 3, 4, 5, 6, 7, 8};

byte seven_seg_digits[10][7] = {{1,1,1,1,1,1,0},
{0,1,1,0,0,0,0}, {1,1,0,1,1,0,1}, {1,1,1,1,0,0,1},
{0,1,1,0,0,1,1}, {1,0,1,1,0,1,1}, {1,0,1,1,1,1,1},
{1,1,1,0,0,0,0}, {1,1,1,1,1,1,1}, {1,1,1,0,0,1,1}};

void sevenSegWrite(byte digit, byte array[]) {
for (byte i = 0; i < 7; i++)
digitalWrite(array[i], seven_seg_digits[digit][i]);
}
```

```
void sevenSegClear(byte array[]) {
for (byte i = 0; i < 7; i++)
digitalWrite(array[i], 0);
}

void setup(){
for (byte i = 0; i < 7; i++)
pinMode(DISPLAY7SEG1[i], OUTPUT);
}

void loop(){
usr1_contagem = 9;
delay(1000);
usr1_contagem -= 1;
if((usr1_contagem == 9)){
sevenSegWrite(0, DISPLAY7SEG1);
}
if((usr1_contagem == 8)){
sevenSegWrite(8, DISPLAY7SEG1);
}
if((usr1_contagem == 7)){
sevenSegWrite(7, DISPLAY7SEG1);
}
if((usr1_contagem == 6)){
sevenSegWrite(6, DISPLAY7SEG1);
}
if((usr1_contagem == 5)){
sevenSegWrite(5, DISPLAY7SEG1);
}
if((usr1_contagem == 4)){
sevenSegWrite(4, DISPLAY7SEG1);
}
if((usr1_contagem == 3)){
sevenSegWrite(3, DISPLAY7SEG1);
}
if((usr1_contagem == 2)){
sevenSegWrite(2, DISPLAY7SEG1);
}
if((usr1_contagem == 1)){
sevenSegWrite(1, DISPLAY7SEG1);
}
if((usr1_contagem == 0)){
sevenSegWrite(0, DISPLAY7SEG1);
}
}
```

Questão 7

```
float usr1_senha; String syst_getMsg(){ String content = ""; char character;
while(Serial.available()){ character = Serial.read(); content.concat(character);
delay(10); } return content; } const int LED1 = 3; void setup(){ pinMode(LED1,
```

```
OUTPUT); usr1_senha = 3; Serial.begin(9600); }
void loop(){
Serial.println("qual a senha?");
while (!(Serial.available())){ }
if(("123" == syst_getMsg())){
Serial.println("senha correta");
digitalWrite(LED1, HIGH); }
else{ Serial.println("senha incorreta");
digitalWrite(LED1, LOW);
usr1_senha -= 1;
if((usr1_senha == 0)){ while (!(false)){ } } } }
```

Questão 8

Se o botão 1 for pressionado, liga o motor para frente. Se o botão 2 for pressionado, liga o motor para trás

Aluno 4

Questão 1

```
const int BUTTON1 = 2;
const int LED1 = 7;
const int BUTTON2 = 3;
const int BUTTON3 = 4;
const int BUTTON4 = 5;
const int BUTTON5 = 6;

void setup(){
pinMode(BUTTON1, INPUT);
pinMode(BUTTON2, INPUT);
pinMode(BUTTON3, INPUT);
pinMode(BUTTON4, INPUT);
pinMode(BUTTON5, INPUT);
pinMode(LED1, OUTPUT);
}

void loop(){
if(digitalRead(BUTTON1) == LOW){
digitalWrite(LED1, HIGH);
blinkLed(LED1, 1, 2);
}
else{
}
if(digitalRead(BUTTON2) == LOW){
digitalWrite(LED1, HIGH);
blinkLed(LED1, 2, 4);
}
if(digitalRead(BUTTON3) == LOW){
digitalWrite(LED1, HIGH);
blinkLed(LED1, 3, 6);
}
if(digitalRead(BUTTON4) == LOW){
digitalWrite(LED1, HIGH);
```

```
blinkLed(LED1, 4, 8);
}
if(digitalRead(BUTTON5) == LOW){
digitalWrite(LED1, HIGH);
blinkLed(LED1, 5, 10);
}
}
Questão 2
float usr1_Numero1;
float usr2_Numero2;
const int LED1 = 2;

void setup(){
pinMode(LED1, OUTPUT);
}

void loop(){
usr1_Numero1 = 10;
usr2_Numero2 = 9;
if((usr1_Numero1 > usr2_Numero2)){
digitalWrite(LED1, HIGH);
}
else{
digitalWrite(LED1, LOW);
}
}
Questão 3
float usr1_tp1;
float usr2_tp2;
const int LED1 = 2;

void setup(){
pinMode(LED1, OUTPUT);
}

void loop(){
usr1_tp1 = 1000;
usr2_tp2 = 2000;
digitalWrite(LED1, HIGH);
delay((usr1_tp1 + usr2_tp2));
digitalWrite(LED1, LOW);
delay((usr1_tp1 - usr2_tp2));
}
Questão 4
int syst1;
const int LED1 = 2;
const int BUZZER1 = 3;

void setup(){
```

```
pinMode(BUZZER1, INPUT);
pinMode(LED1, OUTPUT);
}

void loop(){
for (syst1 = 0; syst1 < (1000); ++syst1){
digitalWrite(LED1, HIGH);
delay(1000);
digitalWrite(LED1, LOW);
delay(1000);
}
tone(BUZZER1, 33, 2001);
}
```

Questão 5

o Led 1 acende com luminosidade maior que 500, o led 2 acende com luminosidade maior que 100 e menor que 500 e o led 3 acende com luminosidade menor que 100

Questão 6

```
byte DISPLAY7SEG1[7] = {2, 3, 4, 5, 6, 7, 8};

byte seven_seg_digits[10][7] = {{1,1,1,1,1,1,0},
{0,1,1,0,0,0,0}, {1,1,0,1,1,0,1}, {1,1,1,1,0,0,1},
{0,1,1,0,0,1,1}, {1,0,1,1,0,1,1}, {1,0,1,1,1,1,1},
{1,1,1,0,0,0,0}, {1,1,1,1,1,1,1}, {1,1,1,0,0,1,1}};

void sevenSegWrite(byte digit, byte array[]) {
for (byte i = 0; i < 7; i++)
digitalWrite(array[i], seven_seg_digits[digit][i]);
}

void sevenSegClear(byte array[]) {
for (byte i = 0; i < 7; i++)
digitalWrite(array[i], 0);
}

void setup(){
for (byte i = 0; i < 7; i++)
pinMode(DISPLAY7SEG1[i], OUTPUT);
}

void loop(){
sevenSegWrite(9, DISPLAY7SEG1);
delay(1000);
sevenSegWrite(8, DISPLAY7SEG1);
delay(1000);
sevenSegWrite(7, DISPLAY7SEG1);
delay(1000);
sevenSegWrite(6, DISPLAY7SEG1);
delay(1000);
```

```
sevenSegWrite(5, DISPLAY7SEG1);
delay(1000);
sevenSegWrite(4, DISPLAY7SEG1);
delay(1000);
sevenSegWrite(3, DISPLAY7SEG1);
delay(1000);
sevenSegWrite(2, DISPLAY7SEG1);
delay(1000);
sevenSegWrite(1, DISPLAY7SEG1);
delay(1000);
sevenSegWrite(0, DISPLAY7SEG1);
}
```

Questão 7

```
String syst_getMsg(){ String content = ""; char character;
while(Serial.available()){ character = Serial.read(); content.concat(character);
delay(10); } return content; } const int LED1 = 2; float usr1_contador; void
setup(){ pinMode(LED1, OUTPUT); Serial.begin(9600); usr1_contador = 0; }
void loop(){
Serial.println("senha?");
if((syst_getMsg() == "12345")){
Serial.println("yeah");
digitalWrite(LED1, HIGH);
delay(1000);
digitalWrite(LED1, LOW); }
else{
Serial.println("tu caiu nas peripecias do pica pau");
usr1_contador += 1; }
if((usr1_contador == 3)){ while (!(false)){ } } }
```

Questão 8

Se o BOTTON1 está precionado o motor vai pra frente, Se o BOTTON2 está precionado o motor vai pra trás

Aluno 5

Questão 1

```
const int BUTTON1 = 2;
const int LED1 = 7;
const int BUTTON2 = 4;
const int BUTTON3 = 3;
const int BUTTON4 = 5;
const int BUTTON5 = 6;

void setup(){
pinMode(BUTTON1, INPUT);
pinMode(BUTTON2, INPUT);
pinMode(BUTTON3, INPUT);
pinMode(BUTTON4, INPUT);
pinMode(BUTTON5, INPUT);
pinMode(LED1, OUTPUT);
}
```

```
void loop(){
if(digitalRead(BUTTON1) == LOW){
digitalWrite(LED1, HIGH);
blinkLed(LED1, 1, 1);
}
else{
digitalWrite(LED1, LOW);
}
if(digitalRead(BUTTON2) == LOW){
digitalWrite(LED1, HIGH);
blinkLed(LED1, 2, 1);
}
else{
digitalWrite(LED1, LOW);
}
if(digitalRead(BUTTON3) == LOW){
digitalWrite(LED1, HIGH);
blinkLed(LED1, 3, 6);
}
else{
digitalWrite(LED1, LOW);
}
if(digitalRead(BUTTON4) == LOW){
digitalWrite(LED1, HIGH);
blinkLed(LED1, 4, 8);
}
else{
digitalWrite(LED1, LOW);
}
if(digitalRead(BUTTON5) == LOW){
digitalWrite(LED1, HIGH);
blinkLed(LED1, 5, 10);
}
else{
digitalWrite(LED1, LOW);
}
}
Questão 2
float usr1_Numero1;
float usr2_numero2;
const int LED1 = 2;

void setup(){
pinMode(LED1, OUTPUT);
}

void loop(){
usr1_Numero1 = 10;
usr2_numero2 = 5;
```

```
if((usr1_Numero1 > usr2_numero2)){
digitalWrite(LED1, HIGH);
}
else{
digitalWrite(LED1, LOW);
}
}
```

Questão 3

```
float usr1_Tempo1;
float usr2_Tempo2;
const int LED1 = 2;

void setup(){
pinMode(LED1, OUTPUT);
}

void loop(){
usr1_Tempo1 = 1000;
usr2_Tempo2 = 2000;
digitalWrite(LED1, HIGH);
delay((usr1_Tempo1 + usr2_Tempo2));
digitalWrite(LED1, LOW);
delay((usr1_Tempo1 - usr2_Tempo2));
}
```

Questão 4

```
int syst1;
const int LED1 = 2;
const int BUZZER1 = 3;

void setup(){
pinMode(BUZZER1, INPUT);
pinMode(LED1, OUTPUT);
}

void loop(){
for (syst1 = 0; syst1 < (1000); ++syst1){
digitalWrite(LED1, HIGH);
delay(1000);
digitalWrite(LED1, LOW);
delay(1000);
}
tone(BUZZER1, 196, 2000);
}
```

Questão 5

LED1- Se a luminosidade for maior que 500
LED2 - Se a luminosidade estiver inferior a 500 e superior a 100
LED3 - Se a luminosidade estiver inferior a 100

Questão 6

```
byte DISPLAY7SEG1[7] = {2, 3, 4, 5, 6, 7, 8};

byte seven_seg_digits[10][7] = {{1,1,1,1,1,1,0},
{0,1,1,0,0,0,0}, {1,1,0,1,1,0,1}, {1,1,1,1,0,0,1},
{0,1,1,0,0,1,1}, {1,0,1,1,0,1,1}, {1,0,1,1,1,1,1},
{1,1,1,0,0,0,0}, {1,1,1,1,1,1,1}, {1,1,1,0,0,1,1}};

void sevenSegWrite(byte digit, byte array[]) {
for (byte i = 0; i < 7; i++)
digitalWrite(array[i], seven_seg_digits[digit][i]);
}

void sevenSegClear(byte array[]) {
for (byte i = 0; i < 7; i++)
digitalWrite(array[i], 0);
}

void setup(){
for (byte i = 0; i < 7; i++)
pinMode(DISPLAY7SEG1[i], OUTPUT);
}

void loop(){
sevenSegWrite(9, DISPLAY7SEG1);
delay(1000);
sevenSegWrite(8, DISPLAY7SEG1);
delay(1000);
sevenSegWrite(7, DISPLAY7SEG1);
delay(1000);
sevenSegWrite(6, DISPLAY7SEG1);
delay(1000);
sevenSegWrite(5, DISPLAY7SEG1);
delay(1000);
sevenSegWrite(4, DISPLAY7SEG1);
delay(1000);
sevenSegWrite(3, DISPLAY7SEG1);
delay(1000);
sevenSegWrite(2, DISPLAY7SEG1);
delay(1000);
sevenSegWrite(1, DISPLAY7SEG1);
delay(1000);
sevenSegWrite(0, DISPLAY7SEG1);
delay(1000);
}
```

Questão 7

```
String syst_getMsg(){ String content = ""; char character;
while(Serial.available()){ character = Serial.read(); content.concat(character);
delay(10); } return content; } const int LED1 = 2; float usr1_contador; void
setup(){ pinMode(LED1, OUTPUT); Serial.begin(9600); usr1_contador = 0; }
```

```
void loop(){ Serial.println("Qual a senha ?"); if((syst_getMsg() == "12345")){
Serial.println("Correto "); digitalWrite(LED1, HIGH); delay(1000);
digitalWrite(LED1, LOW); } else{ Serial.println("Errado"); usr1_contador += 1; }
if((usr1_contador == 3)){ digitalWrite(LED1, LOW); } }
```

Questão 8

Num ciclo paro o motor se o botao esta pressionado ligue-o para frente numa determinada velocidade. Se outro botao esta pressionado ligue-o para tras numa determinada velocidade

Aluno 6

Questão 1

```
const int BUTTON1 = 2;
const int LED1 = 7;
void blinkLed(int ledPin, int qty, int time){
for (int i = 0; i < qty; i++) {
digitalWrite(ledPin, HIGH);
delay((500*time)/qty);
digitalWrite(ledPin, LOW);
delay((500*time)/qty);
}
}
const int BUTTON2 = 3;
const int LED2 = 8;
const int BUTTON3 = 4;
const int LED3 = 9;
const int BUTTON4 = 5;
const int LED4 = 10;
const int BUTTON5 = 6;
const int LED5 = 11;

void setup(){
pinMode(BUTTON1, INPUT);
pinMode(BUTTON2, INPUT);
pinMode(BUTTON3, INPUT);
pinMode(BUTTON4, INPUT);
pinMode(BUTTON5, INPUT);
pinMode(LED1, OUTPUT);
pinMode(LED2, OUTPUT);
pinMode(LED3, OUTPUT);
pinMode(LED4, OUTPUT);
pinMode(LED5, OUTPUT);
}

void loop(){
if(digitalRead(BUTTON1) == LOW){
blinkLed(LED1, 1, 1);
}
if(digitalRead(BUTTON2) == LOW){
blinkLed(LED2, 2, 2);
}
```

```
if(digitalRead(BUTTON3) == LOW){
blinkLed(LED3, 3, 3);
}
if(digitalRead(BUTTON4) == LOW){
blinkLed(LED4, 4, 4);
}
if(digitalRead(BUTTON5) == LOW){
blinkLed(LED5, 5, 5);
}
}
```

Questão 2

```
float usr1_Numero1;
float usr2_Numero2;
const int LED1 = 2;

void setup(){
pinMode(LED1, OUTPUT);
}

void loop(){
usr1_Numero1 = 10;
usr2_Numero2 = 5;
if((usr1_Numero1 > usr2_Numero2)){
digitalWrite(LED1, HIGH);
}
else{
}
}
```

Questão 3

```
float usr1_Tempo1;
float usr2_Tempo2;
const int LED1 = 2;

void setup(){
pinMode(LED1, OUTPUT);
}

void loop(){
usr1_Tempo1 = 1000;
usr2_Tempo2 = 2000;
digitalWrite(LED1, HIGH);
delay((usr1_Tempo1 + usr2_Tempo2));
digitalWrite(LED1, LOW);
delay((usr2_Tempo2 - usr1_Tempo1));
}
```

Questão 4

```
const int LED1 = 3;
float usr1_Repetir;
const int BUZZER1 = 2;
```

```
void setup(){
pinMode(BUZZER1, INPUT);
pinMode(LED1, OUTPUT);
}

void loop(){
digitalWrite(LED1, HIGH);
usr1_Repetir = 1;
if((usr1_Repetir > 1000)){
tone(BUZZER1, 33, 2);
}
}
```

Questão 5

Sempre que a luminosidade captada pelo LDR for maior que 500 o LED1 acende.
Sempre que a luminosidade captada pelo LDR for menor que 500 e maior que 100 o LED2 acende e LED1 e LED2 ficam apagados.
Sempre que a luminosidade captada pelo LDR for menor que 100 o LED3 acende e LED1 e LED2 ficam apagados.

Questão 6

```
byte DISPLAY7SEG1[7] = {5, 6, 7, 8, 9, 10, 11};

byte seven_seg_digits[10][7] = {{1,1,1,1,1,1,0},
{0,1,1,0,0,0,0}, {1,1,0,1,1,0,1}, {1,1,1,1,0,0,1},
{0,1,1,0,0,1,1}, {1,0,1,1,0,1,1}, {1,0,1,1,1,1,1},
{1,1,1,0,0,0,0}, {1,1,1,1,1,1,1}, {1,1,1,0,0,1,1}};

void sevenSegWrite(byte digit, byte array[]) {
for (byte i = 0; i < 7; i++)
digitalWrite(array[i], seven_seg_digits[digit][i]);
}

void sevenSegClear(byte array[]) {
for (byte i = 0; i < 7; i++)
digitalWrite(array[i], 0);
}

void setup(){
for (byte i = 0; i < 7; i++)
pinMode(DISPLAY7SEG1[i], OUTPUT);
}

void loop(){
sevenSegWrite(9, DISPLAY7SEG1);
delay(500);
sevenSegWrite(8, DISPLAY7SEG1);
delay(500);
sevenSegWrite(7, DISPLAY7SEG1);
```

```
delay(500);
sevenSegWrite(6, DISPLAY7SEG1);
delay(500);
sevenSegWrite(5, DISPLAY7SEG1);
delay(500);
sevenSegWrite(4, DISPLAY7SEG1);
delay(500);
sevenSegWrite(3, DISPLAY7SEG1);
delay(500);
sevenSegWrite(2, DISPLAY7SEG1);
delay(500);
sevenSegWrite(1, DISPLAY7SEG1);
delay(500);
sevenSegWrite(0, DISPLAY7SEG1);
delay(500);
}
```

Questão 7

```
String syst_getMsg(){ String content = ""; char character;
while(Serial.available()){ character = Serial.read(); content.concat(character);
delay(10); } return content; } void syst_waitForTheMsg(String s){ do{
while(!Serial.available()); }while(syst_getMsg() != s); } const int LED1 = 2; float
usr1_Erro; void setup(){ pinMode(LED1, OUTPUT); Serial.begin(9600);
usr1_Erro = 0; } void loop(){ syst_waitForTheMsg("123");if(Serial.available()){
digitalWrite(LED1, HIGH); } else{ usr1_Erro += 3; } if((usr1_Erro == 3)){
Serial.println("Bundão"); } }
```

Questão 8

Sempre que o botão 1 estiver pressionado o motor 1 liga para frente com a velocidade do valor do potenciômetro. Sempre que o botão 2 estiver pressionado o motor 1 liga para trás com a velocidade do valor do potenciômetro. Sempre que nenhum do dois botões for pressionado o motor 1 desliga.

Aluno 7

Questão 1

```
const int BUTTON1 = 2;
const int LED1 = 7;
void blinkLed(int ledPin, int qty, int time){
for (int i = 0; i < qty; i++) {
digitalWrite(ledPin, HIGH);
delay((500*time)/qty);
digitalWrite(ledPin, LOW);
delay((500*time)/qty);
}
}
const int BUTTON2 = 3;
const int BUTTON3 = 4;
const int BUTTON4 = 5;
const int BUTTON5 = 6;

void setup(){
```

```
pinMode(BUTTON1, INPUT);
pinMode(BUTTON2, INPUT);
pinMode(BUTTON3, INPUT);
pinMode(BUTTON4, INPUT);
pinMode(BUTTON5, INPUT);
pinMode(LED1, OUTPUT);
}

void loop(){
if(digitalRead(BUTTON1) == LOW){
blinkLed(LED1, 1, 1);
}
if(digitalRead(BUTTON2) == LOW){
blinkLed(LED1, 2, 2);
}
if(digitalRead(BUTTON3) == LOW){
blinkLed(LED1, 3, 3);
}
if(digitalRead(BUTTON4) == LOW){
blinkLed(LED1, 4, 4);
}
if(digitalRead(BUTTON5) == LOW){
blinkLed(LED1, 5, 5);
}
}
Questão 2
float usr1_Numero1;
float usr2_Numero2;
const int LED1 = 7;

void setup(){
pinMode(LED1, OUTPUT);
}

void loop(){
usr1_Numero1 = 5;
usr2_Numero2 = 10;
if((usr1_Numero1 < usr2_Numero2)){
digitalWrite(LED1, HIGH);
}
else{
digitalWrite(LED1, LOW);
}
}
Questão 3
float usr1_Tempo1;
float usr2_Tempo2;
const int LED1 = 7;
```

```
void setup(){
pinMode(LED1, OUTPUT);
}

void loop(){
usr1_Tempo1 = 3000;
usr2_Tempo2 = 1000;
digitalWrite(LED1, HIGH);
delay(usr1_Tempo1);
digitalWrite(LED1, LOW);
delay(usr2_Tempo2);
}
```

Questão 4

```
int syst1;
const int BUZZER1 = 2;

void setup(){
pinMode(BUZZER1, INPUT);
pinMode(9, OUTPUT);
}

void loop(){
for (syst1 = 0; syst1 < (100); ++syst1){
digitalWrite(9, HIGH);
}
tone(BUZZER1, 33, 2);
}
```

Questão 5

CASO 1: O LED ficará aceso quando a luminosidade for maior que 500.
CASO 2: O LED ficará aceso quando a luminosidade for menor que 500 e luminosidade for maior que 100.
CASO 3: O LED ficará aceso quando a luminosidade for menor que 100.

Questão 6

```
byte DISPLAY7SEG1[7] = {2, 3, 4, 5, 6, 7, 8};

byte seven_seg_digits[10][7] = {{1,1,1,1,1,1,0},
{0,1,1,0,0,0,0}, {1,1,0,1,1,0,1}, {1,1,1,1,0,0,1},
{0,1,1,0,0,1,1}, {1,0,1,1,0,1,1}, {1,0,1,1,1,1,1},
{1,1,1,0,0,0,0}, {1,1,1,1,1,1,1}, {1,1,1,0,0,1,1}};

void sevenSegWrite(byte digit, byte array[]) {
for (byte i = 0; i < 7; i++)
digitalWrite(array[i], seven_seg_digits[digit][i]);
}

void sevenSegClear(byte array[]) {
for (byte i = 0; i < 7; i++)
digitalWrite(array[i], 0);
}
```

```
void setup(){
for (byte i = 0; i < 7; i++)
pinMode(DISPLAY7SEG1[i], OUTPUT);
}

void loop(){
sevenSegWrite(9, DISPLAY7SEG1);
delay(1000);
sevenSegWrite(8, DISPLAY7SEG1);
delay(1000);
sevenSegWrite(7, DISPLAY7SEG1);
delay(1000);
sevenSegWrite(6, DISPLAY7SEG1);
delay(1000);
sevenSegWrite(5, DISPLAY7SEG1);
delay(1000);
sevenSegWrite(4, DISPLAY7SEG1);
delay(1000);
sevenSegWrite(3, DISPLAY7SEG1);
delay(1000);
sevenSegWrite(2, DISPLAY7SEG1);
delay(1000);
sevenSegWrite(1, DISPLAY7SEG1);
delay(1000);
sevenSegWrite(0, DISPLAY7SEG1);
}
```

Questão 7

```
String syst_getMsg(){
String content = "";
char character;
while(Serial.available()){
character = Serial.read();
content.concat(character);
delay(10);
}
return content;
}
float usr1_codigo;
void setup(){
pinMode(6, OUTPUT);
pinMode(7, OUTPUT);
pinMode(8, OUTPUT);
Serial.begin(9600);
usr1_codigo = 0;
}
void loop(){
Serial.println("Coloque o código");
if((syst_getMsg() == "200")){
```

```
digitalWrite(6, HIGH);
Serial.println("acertou");
}
else{
Serial.println("errou");
}
usr1_codigo += 2;
if((syst_getMsg() == "errou")){
digitalWrite(7, HIGH);
}
else{ Serial.println("200");
}
usr1_codigo += 2;
if((syst_getMsg() == "100")){
digitalWrite(8, HIGH);
}
else{
Serial.println("errou");
}
 usr1_codigo += 3;
}
```

Questão 8

Se o botão 1 estiver pressionado, liga o motor 1 para frente E SUA VELOCIDADE SERÁ KNOB1.

Se o botão 2 estiver pressionado, liga o motor 1 para trás E SUA VELOCIDADE SERÁ KNOB1.

Aluno 8

Questão 1

```
const int BUTTON1 = 3;
const int LED1 = 2;
void blinkLed(int ledPin, int qty, int time){
for (int i = 0; i < qty; i++) {
digitalWrite(ledPin, HIGH);
delay((500*time)/qty);
digitalWrite(ledPin, LOW);
delay((500*time)/qty);
}
}
const int BUTTON2 = 4;
const int BUTTON3 = 5;
const int BUTTON4 = 6;
const int BUTTON5 = 7;

void setup(){
pinMode(BUTTON1, INPUT);
pinMode(BUTTON2, INPUT);
pinMode(BUTTON3, INPUT);
pinMode(BUTTON4, INPUT);
pinMode(BUTTON5, INPUT);
```

```
pinMode(LED1, OUTPUT);
}

void loop(){
if(digitalRead(BUTTON1) == LOW){
blinkLed(LED1, 1, 1);
}
if(digitalRead(BUTTON2) == LOW){
blinkLed(LED1, 2, 2);
}
if(digitalRead(BUTTON3) == LOW){
blinkLed(LED1, 3, 3);
}
if(digitalRead(BUTTON4) == LOW){
blinkLed(LED1, 4, 4);
}
if(digitalRead(BUTTON5) == LOW){
blinkLed(LED1, 5, 5);
}
}
```

Questão 2

```
float usr1_VAR1;
float usr2_VAR2;
const int LED1 = 2;

void setup(){
pinMode(LED1, OUTPUT);
}

void loop(){
usr1_VAR1 = 5;
usr2_VAR2 = 10;
if((usr1_VAR1 < usr2_VAR2)){
digitalWrite(LED1, HIGH);
}
else{
digitalWrite(LED1, LOW);
}
}
```

Questão 3

```
float usr1_TEMPO1;
float usr2_TEMPO2;
const int LED1 = 2;

void setup(){
pinMode(LED1, OUTPUT);
}

void loop(){
```

```
usr1_TEMPO1 = 1000;
usr2_TEMPO2 = 2000;
digitalWrite(LED1, HIGH);
delay((usr1_TEMPO1 + usr2_TEMPO2));
digitalWrite(LED1, LOW);
delay((usr1_TEMPO1 - usr2_TEMPO2));
}
```

Questão 4

```
float usr1_Repetiao;
const int LED1 = 2;
void usr2_Blink(){
digitalWrite(LED1, HIGH);
delay(1000);
digitalWrite(LED1, LOW);
delay(1000);
}

const int BUZZER1 = 8;

void setup(){
pinMode(BUZZER1, INPUT);
pinMode(LED1, OUTPUT);
usr1_Repetiao = 1;
}

void loop(){
if((usr1_Repetiao < 1000)){
usr1_Repetiao += 1;
usr2_Blink();
if((usr1_Repetiao == 1000)){
tone(BUZZER1, 33, 2);
}
}
usr1_Repetiao = 1;
}
```

Questão 5

Sempre que luminosidade captada pelo LDR for maior que 500 acende o LED1 e deligue o LED2 e LED3. Se a luminosidade for de 101 e 499 apague LED1, acenda LED2 e apague o LED3.E quando a luminosidade captada pelo LDR menor que 100 apague LED1 e LED2 e acenda o LED3 após isso o programa irá recomeçar.

Questão 6

```
byte DISPLAY7SEG1[7] = {2, 3, 4, 5, 6, 7, 8};

byte seven_seg_digits[10][7] = {{1,1,1,1,1,1,0},
{0,1,1,0,0,0,0}, {1,1,0,1,1,0,1}, {1,1,1,1,0,0,1},
{0,1,1,0,0,1,1}, {1,0,1,1,0,1,1}, {1,0,1,1,1,1,1},
{1,1,1,0,0,0,0}, {1,1,1,1,1,1,1}, {1,1,1,0,0,1,1}};
```

```
void sevenSegWrite(byte digit, byte array[]) {
for (byte i = 0; i < 7; i++)
digitalWrite(array[i], seven_seg_digits[digit][i]);
}

void sevenSegClear(byte array[]) {
for (byte i = 0; i < 7; i++)
digitalWrite(array[i], 0);
}

void setup(){
for (byte i = 0; i < 7; i++)
pinMode(DISPLAY7SEG1[i], OUTPUT);
}

void loop(){
sevenSegWrite(9, DISPLAY7SEG1);
delay(1000);
sevenSegWrite(8, DISPLAY7SEG1);
delay(1000);
sevenSegWrite(7, DISPLAY7SEG1);
delay(1000);
sevenSegWrite(6, DISPLAY7SEG1);
delay(1000);
sevenSegWrite(5, DISPLAY7SEG1);
delay(1000);
sevenSegWrite(4, DISPLAY7SEG1);
delay(1000);
sevenSegWrite(3, DISPLAY7SEG1);
delay(1000);
sevenSegWrite(2, DISPLAY7SEG1);
delay(1000);
sevenSegWrite(1, DISPLAY7SEG1);
delay(1000);
sevenSegWrite(0, DISPLAY7SEG1);
delay(1000);
}
Questão 7
float usr1_Errox3;
String syst_getMsg(){ String content = "";
char character;
 while(Serial.available()){
character = Serial.read();
content.concat(character);
delay(10);
}
return content;
}
const int LED1 = 9;
```

```
void setup(){
pinMode(LED1, OUTPUT);
usr1_Errox3 = 1;
Serial.begin(9600);
}
void loop(){
if((syst_getMsg() == "Sonsofanarchyitsending666")){
digitalWrite(LED1, HIGH);
}
else{
usr1_Errox3 += 1;
}
if((usr1_Errox3 == 3)){
Serial.println("Vai se foder intrometido de merda!");
while (!(false)){ }
} }
```

Questão 8

Sempre q o botao 1 estiver ligado ele o motor liga para frente com a velocidade do valor do potenciometro.Pro botao dois a mesma coisa só q pra tras.se nehum dos dois esttiver pressionadop o motor fica parado !!!!onze1111!11onze

APÊNDICE J

Questionários de opinião

Oficina

Pesquisa de Opinião

*Required

Nome Completo

Turma

Idade

1 - Você achou que aprender programação (montagem e teste de programas) é fácil ou difícil? *

- Muito fácil
- Fácil
- Médio
- Difícil
- Muito difícil

2 - Cite três coisas sobre programação que você aprendeu na oficina. *

3 - Você achou que aprender robótica (montagem e teste de circuitos) é fácil ou difícil? *

- Muito fácil
- Fácil
- Médio
- Difícil
- Muito difícil

4 - Cite três coisas sobre robótica que você aprendeu na oficina. *

5 - Você acha que conseguiria montar um circuito elétrico na protoboard sozinho(a)? *

- Sim
- Não
- Não sei

6 - Você achou que programar no DuinoBlocks é fácil ou difícil? *

- Muito fácil
- Fácil
- Médio
- Difícil
- Muito difícil

7 - Você preferiria programar usando o DuinoBlocks ou o código de Arduino? *

- DuinoBlocks
- Código Arduino
- Tanto faz

8 - Se te pedissem para dizer o que um programa faz, você acha que seria mais fácil se o código estivesse em DuinoBlocks ou em código Arduino? *

- DuinoBlocks
- Código Arduino
- Tanto faz

9 - Na hora de testar se o seu programa funciona, o que você preferiria? *

- Testar virtualmente na tela do computador
- Montar um circuito para testar o programa
- Other:

10 - O que você achou da oficina ter sido em um laboratório da escola? *

- Gostei
- Não gostei
- Não tenho opinião sobre isso

11 - Você gostaria que a oficina de programação usando robótica se tornasse uma disciplina obrigatória na sua escola? *

- Sim
- Não
- Não sei

12 - Você gostou da oficina ter acontecido fora do horário normal de aula? *

- Gostei
- Não gostei, preferia que fosse em horário normal de aula
- Não tenho opinião sobre isso

13 - O que você gostou no curso? *

- [ ] Aprender a programar
- [ ] Aprender a montar os circuitos
- [ ] Ver os programas ganhando vida
- [ ] O ambiente de programação em blocos (DuinoBlocks)
- [ ] A duração das aulas
- [ ] O dinamismo da aula
- [ ] Aprender sobre os componentes eletrônicos
- [ ] A aula ser na própria escola
- [ ] O professor
- [ ] Os exercícios
- [ ] A matéria vista
- [ ] Trabalhar em duplas
- [ ] Other:

14 - O que você não gostou no curso? *

- [ ] Aprender a programar
- [ ] Aprender a montar os circuitos
- [ ] Ver os programas ganhando vida
- [ ] O ambiente de programação em blocos (DuinoBlocks)
- [ ] A duração das aulas
- [ ] O dinamismo da aula
- [ ] Aprender sobre os componentes eletrônicos
- [ ] A aula ser na própria escola
- [ ] O professor
- [ ] Os exercícios
- [ ] A matéria vista
- [ ] Trabalhar em duplas
- [ ] Other:

15 - O que você esperava aprender na oficina quando se inscreveu? *

16 - Você recomendaria esse curso para um amigo? *

- ( ) Sim
- ( ) Não

17 - Se a oficina fosse mais longa (por volta de três meses) você aceitaria vir aos sábados? *

- ( ) Sim
- ( ) Não
- ( ) Não sei

Submit

Never submit passwords through Google Forms.

Curso

Pesquisa de Opinião - Curso de Robótica

*Required

Nome Completo

Série

Idade

1 - Você achou que aprender programação (montagem e teste de programas) é fácil ou difícil? *

- Muito fácil
- Fácil
- Médio
- Difícil
- Muito difícil

2 - Cite três coisas sobre programação que você aprendeu na oficina. *

3 - Você achou que aprender robótica (montagem e teste de circuitos) é fácil ou difícil? *

- Muito fácil
- Fácil
- Médio
- Difícil
- Muito difícil

4 - Cite três coisas sobre robótica que você aprendeu na oficina. *

5 - Você acha que conseguiria montar um circuito elétrico na protoboard sozinho(a)? *

- Sim
- Não
- Não sei

6 - Você achou que programar no DuinoBlocks é fácil ou difícil? *

- Muito fácil
- Fácil
- Médio
- Difícil
- Muito difícil

7 - Você preferiria programar usando o DuinoBlocks ou o código de Arduino? *

- DuinoBlocks
- Código Arduino
- Tanto faz

8 - Se te pedissem para dizer o que um programa faz, você acha que seria mais fácil se o código estivesse em DuinoBlocks ou em código Arduino? *

- DuinoBlocks
- Código Arduino
- Tanto faz

9 - Na hora de testar se o seu programa funciona, o que você preferiria? *

- Testar virtualmente na tela do computador
- Montar um circuito para testar o programa

10 - O que você achou da oficina ter sido em um laboratório da escola? *

- Gostei
- Não gostei
- Não tenho opinião sobre isso

11 - Você gostaria que a oficina de programação usando robótica se tornasse uma disciplina obrigatória na sua escola? *

- Sim
- Não
- Não sei

12 - Você gostou da oficina ter acontecido fora do horário normal de aula? *

- Gostei
- Não gostei, preferia que fosse em horário normal de aula
- Não tenho opinião sobre isso

13 - O que você mais e menos gostou no curso? *

Marque apenas três opções do que mais gostou e três do que menos gostou

	Gostei	Não gostei
Aprender a programar	○	○
Aprender a montar os circuitos	○	○
Ver o circuito ganhando vida	○	○
O ambiente de programação em blocos (DuinoBlocks)	○	○
A duração das aulas	○	○
A dinamica da aula	○	○
Aprender sobre os componentes eletrônicos	○	○
A aula ser na própria escola	○	○
O professor	○	○
Os exercícios	○	○
A matéria estudada	○	○
Trabalhar em equipes	○	○

14 - O que você esperava aprender na oficina quando se inscreveu? *

15 - Você recomendaria esse curso para um amigo? *

○ Sim
○ Não

Submit

Never submit passwords through Google Forms.

APÊNDICE K

Resultados do questionário de opinião

Oficina

Turma	Idade	1 - Você achou que aprender programação (montagem e teste de programas) é fácil ou difícil?	2 - Cite três coisas sobre programação que você aprendeu na oficina.	3 - Você achou que aprender robótica (montagem e teste de circuitos) é fácil ou difícil?	4 - Cite três coisas sobre robótica que você aprendeu na oficina.
1004	15	Fácil	- Como usar certas condicionais - Usar melhor intervalos de tempo - Configurar melhor variáveis e booleanas	Médio	- Nomes dos elementos - Como funciona a protoboard - Montar circuitos elétricos
1002	16	Muito fácil	-> tive que desenvolver novas lógicas; -> pude ver várias lógicas para fazer a mesma função; -> aprendi melhor como usar as booleanas.	Fácil	-> como montar os circuitos; -> tentar ser menos desastrada; -> ver e prestar mais atenção nos detalhes.
1001	15	Muito fácil	Condicionais, While, Random, Booleanas etc...	Difícil	Ligação de circuitos, funcionamento do Arduino e um pouco dos muitos projetos possiveis com ele.
1002	15	Fácil	if e else, loop, boolean.	Médio	Circuitos, código de Arduino, resistor.
1002	15 anos	Fácil	Aprendi a compreender melhor a mecânica usada para programar, os recursos usados e entendi melhor o uso das funções.	Fácil	Eu aprendi a ligar o LED, a programar um sensor e a controlar a luminosidade pelo potenciômetro.

1003	16	Médio	aprendi como determinar variáveis ficou mais simples programar aprendi a trabalhar melhor com if else	Fácil	como ligar um led como usar um arduino como funciona a conxao de positivo e negativo
1004	17	Médio	nada	Fácil	eu relembrei dos meus conhecimentos do passado mais ganhei mais algumas coisas como fazer o led acender com o censor de luminosidade etc ....
1004	15	Médio	Piscar um led. Escolher ligar ou não o led de acordo com a luminosidade. Ligar a buzina. usar um botão para controlar:o ligamento de um led,o ligamento da buzina e outras coisas que podem ser ligadas com o botão.	Difícil	A usar as saídas certas com os volts certos. a usar as coisas sólidas nas entradas certas(buzina,botão,etc). Aprendi a saber as entradas certas na protoboard.

5 - Você acha que conseguiria montar um circuito elétrico na protoboard sozinho(a)?	6 - Você achou que programar no DuinoBlocks é fácil ou difícil?	7 - Você preferiria programar usando o DuinoBlocks ou o código de Arduino?	8 - Se te pedissem para dizer o que um programa faz, você acha que seria mais fácil se o código estivesse em DuinoBlocks ou em código Arduino?	9 - Na hora de testar se o seu programa funciona, o que você preferiria?	10 - O que você achou da oficina ter sido em um laboratório da escola?
Sim	Fácil	DuinoBlocks	DuinoBlocks	Montar um circuito para testar o programa	Gostei
Não sei	Muito fácil	DuinoBlocks	DuinoBlocks	Testar virtualmente na tela do computador	Gostei
Sim	Muito fácil	Tanto faz	Tanto faz	Montar um circuito para testar o programa	Gostei
Sim	Fácil	DuinoBlocks	DuinoBlocks	Montar um circuito para testar o programa	Gostei
Sim	Fácil	DuinoBlocks	Código Arduino	Montar um circuito para testar o programa	Gostei

Sim	Muito fácil	DuinoBlocks	DuinoBlocks	Montar um circuito para testar o programa	Gostei
Sim	Médio	DuinoBlocks	DuinoBlocks	Montar um circuito para testar o programa	Gostei
Sim	Fácil	DuinoBlocks	DuinoBlocks	Testar virtualmente na tela do computador	Gostei

11 - Você gostaria que a oficina de programação usando robótica se tornasse uma disciplina obrigatória na sua escola?	12 - Você gostou da oficina ter acontecido fora do horário normal de aula?	13 - O que você gostou no curso?	14 - O que você não gostou no curso?	15 - O que você esperava aprender na oficina quando se inscreveu?	16 - Você recomendaria esse curso para um amigo?	17 - Se a oficina fosse mais longa (por volta de três meses) você aceitaria vir aos sábados?
Não sei	Gostei	Aprender a montar os circuitos, Ver os programas ganhando vida, O ambiente de programação em blocos (DuinoBlocks), O dinamismo da aula, Aprender sobre os componentes eletrônicos, A aula ser na própria escola, O professor, Os exercícios, A matéria vista	A duração das aulas, Poderiam durar mais tempo	Honestamente, quando me explicaram o curso, achei que ia aprender a fazer tais coisas como abrir portas automaticamente e criar elevadores. Não que não seja possível fazer isso com o Arduino, mas como eu não o conhecia até vir ao curso, acho que não esperava muita coisa. Me surpreendi bastante.	Sim	Sim
Sim	Gostei	Aprender a programar, Aprender a montar os circuitos, Ver os programas ganhando vida, O ambiente de programação em blocos (DuinoBlocks), O dinamismo da aula, Aprender sobre os componentes eletrônicos, A aula ser na própria escola, O professor,	A duração das aulas, poderia ter mais tempo semanal.	Eu não sabia bem o que esperar quando entrei no curso, mas gostei muito de poder dar vida aos circuitos que eu montava, do jeito que eu queria. No início foi difícil, mas depois de um tempo passei a perceber os algoritmos por trás das	Sim	Sim

		Os exercícios, A matéria vista, Trabalhar em duplas		coisas do dia a dia, como um sinal de transito.		
Sim	Gostei	Aprender a programar, Aprender a montar os circuitos, Ver os programas ganhando vida, O ambiente de programação em blocos (DuinoBlocks), O dinamismo da aula, Aprender sobre os componentes eletrônicos, A aula ser na própria escola, O professor, Os exercícios, A matéria vista, Ter aprendido mais do que esperava...	A duração das aulas, Porque só houveram 5 aulas :'(	Automação, programação, controle de servos, o básico de robótica.	Sim	Sim
Sim	Gostei	Aprender a programar, Aprender a montar os circuitos, Ver os programas ganhando vida, O ambiente de programação em blocos (DuinoBlocks), A duração das aulas, O dinamismo da aula, Aprender sobre os componentes eletrônicos, A aula ser na própria escola, O professor, Os exercícios, A matéria vista	A duração das aulas, Mais tempo de aula	Aprender mais sobre eletrônica e circuitos.	Sim	Sim
Sim	Gostei	Aprender a programar, Aprender a montar os circuitos, Ver os programas ganhando vida, O ambiente de programação em blocos (DuinoBlocks), O dinamismo da aula, Aprender sobre os componentes eletrônicos, A aula ser na própria escola, O professor, Os exercícios, A matéria vista, Trabalhar em duplas	A duração das aulas	Eu não tinha exatamente uma expectativa, eu entrei por curiosidade e gostei muito.	Sim	Não sei
Sim	Gostei	Aprender a programar, Aprender a montar os circuitos, Ver os programas ganhando vida, O	nda.	certamente o que eu esperava.	Sim	Sim

		ambiente de programação em blocos (DuinoBlocks), A duração das aulas, O dinamismo da aula, Aprender sobre os componentes eletrônicos, A aula ser na própria escola, O professor, Os exercícios, A matéria vista, Trabalhar em duplas				
Sim	Não tenho opinião sobre isso	Aprender a montar os circuitos	A duração das aulas	Fazer objetos loucos com arduino	Sim	Não sei
Sim	Gostei	Aprender a programar, Aprender a montar os circuitos, Ver os programas ganhando vida, O ambiente de programação em blocos (DuinoBlocks), A duração das aulas, O dinamismo da aula, Aprender sobre os componentes eletrônicos, A aula ser na própria escola, O professor, Os exercícios, A matéria vista, todas as coisas do universo,este curso foi uma benção na minha vida,amei tudo(só que preferi faze sozinho que aí posso escolher melhor e fazer do jeito que quiser)	Trabalhar em duplas	criar robôs(como nós vemos na televisão)	Sim	Sim

Curso

Série	Idade	1 - Você achou que aprender programação (montagem e teste de programas) é fácil ou difícil?	2 - Cite três coisas sobre programação que você aprendeu na oficina.	3 - Você achou que aprender robótica (montagem e teste de circuitos) é fácil ou difícil?	4 - Cite três coisas sobre robótica que você aprendeu na oficina.	5 - Você acha que conseguiria montar um circuito elétrico na protoboard sozinho(a)?
1° ano	17	Médio	Manter uma organização O uso da condicionais corretamente Ter paciência	Médio	Programar uma mini cidade Como posso usar o LDR E usar o buzzer	Sim
1º ano	17	Fácil	- sequenciação dos comandos em um programa - programar pensando nas ações de algo físico, como um robô - programação em blocos	Médio	- chegar a um programa perfeito por tentativa e erro - solucionar problemas de circuitos - conhecimento sobre alguns componentes eletrônicos	Sim
1° ano	16	Médio	Variáveis, condicionais e funções.	Fácil	Montar um circuito, trabalhar com leds, buzinas e outros e o que de fato é um robô, pois antes da oficina eu nunca imaginaria que a máquina de lavar roupa que eu tenho em casa é um robô.	Sim
1° ano	15	Fácil	Programar uma função, criar e utilizar variáveis e loop.	Médio	As diferenças de pino analógico para pino digital, montar circuitos no Arduino e que resistências são necessárias.	Sim
1° ano	16	Médio	A trabalhar com variaveis, com blocos e a sempre colocar o sempre no começo!	Fácil	Que um circuito tem que estar fechado, que precisa de um fio negativo e positivo e como é o esquema de utilizar a protoboard	Sim
3º ano	17	Fácil	pensamento lógico trabalhar em equipe revisar os programas	Fácil	eletricidade componentes protoboard	Sim

1º ano	16	Fácil	condicionais, funções, loops	Fácil	resistores, buzinas, leds	Sim
1º ano	15	Fácil	programar funções, loops e condicionais, nomear variáveis corretamente e colocar tudo na ordem certa	Médio	montar o circuito na protoboard, ver a direção certa dos componentes (led e buzina) e sempre fechar o circuito elétrico	Sim
1º ano	15	Muito fácil	aprendi a programar loops, condicionais, funções, variáveis e outras coisas. aprendi tb a aperfeiçoar meu programa atraves de várias tentativas	Muito fácil	aprendi a sempre fechar o circuito para a eletricidade circular, a usar as cores certas de fios e a montar os componentes na protobord	Sim
1º ano	16	Fácil	blocos, variáveis e se-senão	Muito fácil	a organizar os fios para que eles não se embolem, a não ter medo de choque e a entender como a protobord funciona	Sim
1º ano	17	Muito fácil	organizar as ideias do programa, consertar os erros que aparecem e melhorar a experiencia do usuario que está usando o programra	Médio	colocar os componentes na protoboard, ligar tudo com os fios, testar pra ver se esta funcionando	Não sei

6 - Você achou que programar no DuinoBlocks é fácil ou difícil?	7 - Você preferiria programar usando o DuinoBlocks ou o código de Arduino?	8 - Se te pedissem para dizer o que um programa faz, você acha que seria mais fácil se o código estivesse em DuinoBlocks ou em código Arduino?	9 - Na hora de testar se o seu programa funciona, o que você preferiria?	10 - O que você achou da oficina ter sido em um laboratório da escola?	11 - Você gostaria que a oficina de programação usando robótica se tornasse uma disciplina obrigatória na sua escola?	12 - Você gostou da oficina ter acontecido fora do horário normal de aula?
Fácil	DuinoBlocks	DuinoBlocks	Testar virtualmente na tela do computador	Gostei	Não	Gostei
Muito fácil	Código Arduino	DuinoBlocks	Montar um circuito para testar o programa	Gostei	Não sei	Gostei

Médio	DuinoBlocks	DuinoBlocks	Montar um circuito para testar o programa	Gostei	Sim	Gostei
Muito fácil	DuinoBlocks	DuinoBlocks	Montar um circuito para testar o programa	Gostei	Não sei	Gostei
Fácil	DuinoBlocks	DuinoBlocks	Montar um circuito para testar o programa	Gostei	Não sei	Gostei
Muito fácil	DuinoBlocks	DuinoBlocks	Montar um circuito para testar o programa	Gostei	Não sei	Gostei
Fácil	DuinoBlocks	DuinoBlocks	Montar um circuito para testar o programa	Gostei	Sim	Não gostei, preferia que fosse em horário normal de aula
Muito fácil	DuinoBlocks	DuinoBlocks	Montar um circuito para testar o programa	Gostei	Não sei	Gostei
Muito fácil	DuinoBlocks	DuinoBlocks	Montar um circuito para testar o programa	Gostei	Sim	Não gostei, preferia que fosse em horário normal de aula
Fácil	DuinoBlocks	DuinoBlocks	Montar um circuito para testar o programa	Gostei	Não sei	Gostei
Muito fácil	DuinoBlocks	DuinoBlocks	Testar virtualmente na tela do computador	Gostei	Sim	Não gostei, preferia que fosse em horário normal de aula

13a - O que você mais gostou no curso?	13b - O que você menos gostou no curso?	14 - O que você esperava aprender na oficina quando se inscreveu?	15 - Você recomendaria esse curso para um amigo?
Aprender a montar os circuitos, Ver o circuito ganhando vida, O ambiente de programação em blocos (DuinoBlocks), A duração das aulas, A dinamica da aula, Aprender sobre os componentes eletrônicos, A aula ser na própria escola, O professor, Os exercícios, A matéria estudada, Trabalhar em equipes		Programar Robôs (bonecos) a se movimentarem!	Sim
Aprender a montar os circuitos, Ver o circuito ganhando vida, O ambiente de programação em blocos (DuinoBlocks), A dinamica da aula, Aprender sobre os componentes eletrônicos, A aula ser na própria escola, O professor, Os exercícios, A matéria estudada, Trabalhar em equipes	A duração das aulas	Esperava aprender o básico sobre programação com robótica e aprender algumas coisas sobre componentes eletrônicos e circuitos	Sim
Aprender a montar os circuitos, Ver o circuito ganhando vida, O ambiente de programação em blocos (DuinoBlocks), A dinamica da aula, A aula ser na própria escola, O professor, Os exercícios, A matéria estudada, Trabalhar em equipes	A duração das aulas, Aprender sobre os componentes eletrônicos	Esperava aprender a montar e programar um robô.	Sim
Aprender a montar os circuitos, Ver o circuito ganhando vida, O ambiente de programação em blocos (DuinoBlocks), A duração das aulas, A dinamica da aula, Aprender sobre os componentes eletrônicos, A aula ser na própria escola, O professor, Os exercícios, A matéria estudada, Trabalhar em equipes		Esperava aprender mais robótico pois não havia sido informado que iríamos aprender a programar com o uso da Robótica.(botei que não gostei da matéria estudada porquê marquei errado e teria que fazer tudo de novo hahah.)	Sim
Aprender a montar os circuitos, Ver o circuito ganhando vida, O ambiente de programação em blocos (DuinoBlocks), A duração das aulas, A dinamica da aula, Aprender sobre os componentes eletrônicos, A aula ser na própria escola, O professor, Os exercícios, A matéria estudada		Conseguir fazer com que alguma coisa que eu pensasse conseguisse exercer sua função pensada	Sim
Aprender a montar os circuitos, Ver o circuito ganhando vida, O ambiente de programação em blocos (DuinoBlocks), A dinamica da aula, Aprender sobre os componentes	A duração das aulas	esperava aprender a programar robos e como eles funcionam	Sim

eletrônicos, A aula ser na própria escola, O professor, Os exercícios, A matéria estudada, Trabalhar em equipes			
Aprender a montar os circuitos, Ver o circuito ganhando vida, O ambiente de programação em blocos (DuinoBlocks), A dinamica da aula, Aprender sobre os componentes eletrônicos, A aula ser na própria escola, O professor, Os exercícios, A matéria estudada	A duração das aulas	esperava poder criar meu próprio robo	Sim
Aprender a montar os circuitos, Ver o circuito ganhando vida, O ambiente de programação em blocos (DuinoBlocks), A duração das aulas, A dinamica da aula, Aprender sobre os componentes eletrônicos, A aula ser na própria escola, O professor, A matéria estudada, Trabalhar em equipes		esperava poder criar um robo e levar ele pra minha casa	Sim
Aprender a montar os circuitos, Ver o circuito ganhando vida, O ambiente de programação em blocos (DuinoBlocks), A duração das aulas, A dinamica da aula, Aprender sobre os componentes eletrônicos, A aula ser na própria escola, O professor, Os exercícios, A matéria estudada, Trabalhar em equipes		esperava programar um robo, mas gostei muito de descobrir que existem robos por toda parte, e que um robo pode ter muitas aparencias diferentes	Sim
Aprender a montar os circuitos, Ver o circuito ganhando vida, O ambiente de programação em blocos (DuinoBlocks), A duração das aulas, A dinamica da aula, Aprender sobre os componentes eletrônicos, A aula ser na própria escola, O professor, Os exercícios, A matéria estudada, Trabalhar em equipes		esperava poder aprender a mexer e montar circuitos elétricos, e aprendi muito nesse quesito	Sim
Aprender a montar os circuitos, Ver o circuito ganhando vida, O ambiente de programação em blocos (DuinoBlocks), A duração das aulas, A dinamica da aula, A aula ser na própria escola, O professor, Os exercícios, A matéria estudada, Trabalhar em equipes	Aprender sobre os componentes eletrônicos	esperava poder dar vida aos robos e outras coisas q eu montasse	Sim

APÊNDICE L

Provas analisadas da disciplina de Computação I Unificada, oferecida na Universidade Federal do Rio de Janeiro.