Ensino de programação em robótica com Arduino para alunos do ensino fundamental: relato de experiência

Teaching Arduino robotics programming for elementary school students: experience report

Luciano Frontino de Medeiros[*]
Luana Priscila Wünsch[**]

Resumo

Este artigo relata práticas de ensino de programação para a plataforma Arduino, vivenciadas por meio de um curso elaborado para alunos do ensino fundamental II pertencentes a nove escolas públicas do município de Curitiba, abrangendo ao todo 117 alunos do 5º ao 9º ano. Esses alunos fazem parte de grupos que participam de competições de robótica de forma regular, entretanto, sem possuir familiaridade com a plataforma Arduino, de modo geral. No decorrer do curso, os alunos foram observados e constatou-se a capacidade de aprendizado em um tempo bastante breve, assim como as dificuldades na assimilação dos conceitos básicos de programação, os quais eram apresentados de forma incremental e tendo foco na proposição de desafios a partir de noções mais simples. O uso do simulador para as tarefas de programação facilitou a transição da forma de programação visual, da qual os alunos já possuíam certo conhecimento, para uma programação mais textual. Este relato busca mostrar, ainda, como o aprendizado de programação pode auxiliar na constituição do pensamento formal a partir do concreto. Ficou evidenciado o caráter motivador que as atividades de robótica proporcionam ao processo de aprendizagem e que podem, por sua vez, servir de facilitador para a introdução de conceitos mais complexos relativos a linguagens de programação.

Palavras-chave: Ensino de programação na educação básica. Pensamento computacional. Programação para Arduino. Robótica educacional.

Recebido em 30/09/2018 – Aprovado em 27/02/2019
http://dx.doi.org/10.5335/rep.v26i2.8701

[*] Doutor em Engenharia e Gestão do Conhecimento pela Universidade Federal de Santa Catarina, com pós-doutorado em Inteligência Artificial na Universidade Politécnica de Madri (2013). Professor permanente do Programa de Pós-Graduação Stricto Sensu em Educação – Mestrado Profissional: Educação e Novas Tecnologias do Centro Universitário Internacional (Uninter), São Paulo, Brasil. E-mail: luciano.me@uninter.com

[**] Doutora em Educação pela Universidade de Lisboa. Professora do Programa de Pós-Graduação Stricto Sensu em Educação – Mestrado Profissional: Educação e Novas Tecnologias do Centro Universitário Internacional (Uninter), São Paulo, Brasil. E-mail: luana.w@uninter.com

Abstract

The paper reports the experiences of teaching programming for the Arduino platform, through a course designed for Elementary School II students belonging to 9 (nine) public schools in the city of Curitiba, covering 117 students from the 5th to 9th grade . These students are part of groups that participate in robotics competitions on a regular basis, however without being familiar with the Arduino platform in general. During the course, the students were observed and the learning capacity of the students was verified in a very short time, as well as the difficulties in the assimilation of the basic concepts of programming, which were presented incrementally and focusing on the proposition of challenges from basic notions. The use of the simulator for the programming tasks facilitated the transition from the form of visual programming, of which the students already had certain knowledge, for a more textual programming. It is also tried to show how the programming learning can help in the constitution of the formal thought from the concrete. It was evidenced the motivating character that the robotic activities provide to the learning process and that, in turn, can serve as a facilitator for the introduction of more complex concepts related to programming languages.

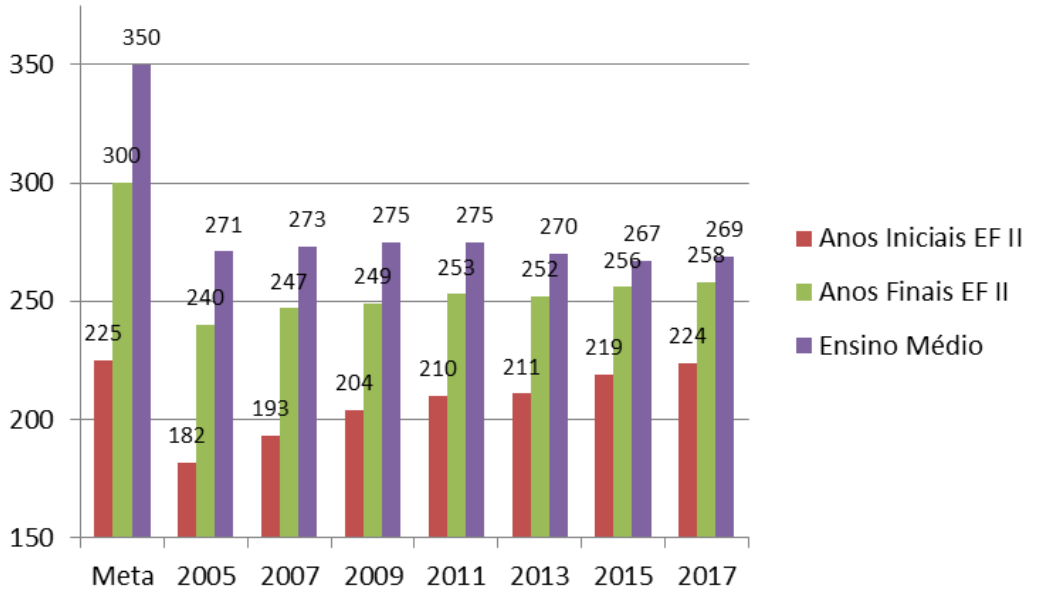
Keywords: Teaching programming in basic education. Computational thinking. Programming for Arduino. Educational robotics.

Introdução

Nos últimos anos, tem-se constatado de maneira mais frequente as dificuldades dos alunos brasileiros com conteúdos de matemática e ciências. Os resultados apontados pelo Programme for International Student Assessment (Pisa), em 2015, mostraram que, com relação à média dos países da Organização para a Cooperação e Desenvolvimento Económico (OCDE), na avaliação de alunos que têm em média 15 anos e estão próximos de terminar o ciclo da educação básica, o Brasil está bem abaixo, alcançando 401 pontos em ciências contra uma média de 493 pontos, e 377 pontos em matemática contra uma média de 490 pontos (OCDE, 2015).

Em contraponto, dados obtidos do Sistema de Avaliação da Educação Básica (Saeb), do Instituto Nacional de Estudos e Pesquisas Educacionais Anísio Teixeira (Inep) (2017), vinculado ao Ministério da Educação (MEC), têm mostrado que, para alunos dos anos iniciais e finais do ensino fundamental, a média em Matemática é crescente, enquanto fica estagnada no ensino médio. Na Figura 1, é possível comparar o desempenho desde 2005 até 2017. Entretanto, os índices ainda se configuram aquém se comparados com as metas do grupo 3, estabelecidas pelo movimento Todos pela Educação, exceto para os anos iniciais.[1] Esse cenário permite vislumbrar que os desafios e oportunidades não são poucos na adoção de iniciativas para melhorar o desempenho dos alunos. O ensino de tecnologias educacionais baseadas em robótica e programação pode se constituir num elemento impulsionador para a melhoria destes níveis, de maneira geral nas disciplinas de ciências, tecnologia, engenharia e matemática, denominadas de forma comum pela sigla STEM (*Science, Technology, Engineering and Mathematics*).

Figura 1 – Desempenho em matemática na educação básica de 2005 a 2017



* A meta nas primeiras colunas é a estipulada pelo movimento Todos pela Educação.

Fonte: Saeb – Inep/MEC, Todos pela Educação (2017).

Em nível curricular, ainda que a Base Nacional Comum Curricular (BNCC) esteja propondo o ensino de fluxogramas para representação de processos de resolução e problemas passo a passo em algumas habilidades do 6º ao 9º ano, carece-se da descrição de habilidades relativas ao pensamento computacional, bem como de conceitos de ensino de programação. Tomando como base iniciativas que acontecem em outros países, como nos Estados Unidos, onde a Computer Science Teacher Association (CSTA), uma associação de professores da área de Ciência da Computação, delineou um conjunto central de objetivos de aprendizagem desenhados para fornecer a base para um currículo completo em ciência da computação e sua implementação no nível K-12 (relativo aos níveis de ensino fundamental e médio). Apenas como ilustração, o primeiro nível (1-A), que abrange a faixa de 5 a 7 anos de idade, já prevê o desenvolvimento de programas com sequências e *loops* (laços de repetição) simples, para expressar ideias ou para mapear problemas (CSTA, 2017).

As bases teóricas do ensino da robótica e da programação têm como pano de fundo a abordagem do pensamento computacional. Wing (2006) defende que o pensamento computacional é uma habilidade a ser desenvolvida não somente para os

cientistas da computação, mas para qualquer pessoa, e que deveria ser já abordado no ensino fundamental, juntamente com aritmética, leitura e escrita. Pensar como um cientista da computação requer altos níveis de abstração. Porém, a noção por trás do pensamento computacional vai além de programar, sendo na verdade "conceituar". Como atividade típica, o pensamento computacional envolve a reformulação de um problema aparentemente difícil de uma forma que se saiba resolver. Não se trata de tentar fazer seres humanos pensarem como computadores.

Grover e Pea (2013, p. 39-40) listam uma série de elementos relativos ao pensamento computacional que deveria estar na base de qualquer currículo escolar: i) abstrações e generalização de padrões; ii) processamento sistemático da informação; iii) sistemas de símbolos e representações; iv) noções algorítmicas de controle de fluxo; v) decomposição estruturada de problemas; v) pensamento iterativo, recursivo e paralelo; vi) lógica condicional; vii) restrições de performance e eficiência; viii) depuração e detecção sistemática de erros.

A partir da problemática explicitada, este artigo detalha uma experiência de ensino de programação, utilizando em específico a plataforma Arduino. Buscou-se explorar as possibilidades do uso da robótica e da linguagem Scratch como facilitadores para o entendimento de conceitos mais complexos de programação, utilizando como recurso um simulador de Arduino. O curso foi planejado para alunos que já possuíam alguma experiência com robótica, o que de certa forma permitiu trabalhar aspectos mais avançados. Na sequência do artigo, são apresentados os temas relativos à programação, relacionando-os a alguns aspectos da aprendizagem e da programação com Arduino e Scratch. Depois, mostra-se o planejamento do curso de programação, seguido pelo relato da experiência e pelas considerações finais.

A atividade de programação e aspectos cognitivos

Programar significa, em suma, construir algoritmos. De acordo com Cormen et al. (2002), um algoritmo pode ser entendido como uma sequência de passos que transformam as entradas em saídas. Assim,

> [...] podemos visualizar um algoritmo como uma ferramenta para resolver um problema computacional bem especificado. O enunciado do problema especifica em termos gerais o relacionamento entre a entrada e a saída desejada. O algoritmo descreve um procedimento computacional específico para se alcançar esse relacionamento da entrada com a saída (CORMEN et al., 2002, p. 3).

Da concepção de algoritmo como uma sequência de passos para se atingir um objetivo, pode-se enquadrar tudo aquilo que se ensina para um aluno em termos de procedimentos que envolvem operações lógicas ou matemáticas, tais como a resolução de equações ou mesmo o problema de se encontrar o máximo divisor comum ou o mínimo múltiplo comum. Assim, os professores podem não se dar conta, mas ensinam algoritmos em vários momentos aos seus alunos.

Kazimoglu et al. (2012) sumarizam, a partir de uma série de estudos, um conjunto comum de habilidades que são trabalhadas na abordagem do pensamento computacional: a resolução de problemas, a construção de algoritmos, a depuração, a simulação e a socialização. A resolução de problemas refere-se ao raciocínio lógico feito por meio de diversos modelos computacionais. Isso inclui a aplicação da decomposição do problema em partes menores ou mesmo para gerar alternativas de representação. A construção de algoritmos envolve a elaboração de procedimentos passo a passo para a solução de um problema em particular. A depuração pressupõe a análise dos problemas quanto aos erros lógicos que podem acontecer, na qual o aluno trabalha com o *feedback* e deve rever as regras ou estratégias de abordagem do problema. A simulação envolve a implementação de modelos no computador. E, por fim, o aspecto da socialização envolve coordenação de esforços, cooperação e/ou competição durante os estágios de resolução do problema.

Lidar com o ensino e a aprendizagem de programação no nível do ensino fundamental pressupõe trabalhar com alunos que estão, à luz da teoria piagetiana, na transição da fase operatório-concreta para a operatório-formal. Este último estágio é identificado com o aparecimento do pensamento proposicional, que não fica mais restrito à consideração do concreto, mas começa a lidar também com o domínio daquilo que é hipotético (LEFRANÇOIS, 2013, p. 260). O sujeito não precisa mais do apoio no pensamento concreto e pode admitir possibilidades de explicar ou resolver uma situação antes de experimentá-la na realidade, ou seja, começa a elaborar hipóteses (MARTINELLI; MARTINELLI, 2016, p. 52). Assim, a tendência é que um indivíduo na fase operatório-formal faça algo mais do que testar proposições individuais.

> Ele raciocina sobre as relações lógicas que existem entre duas ou mais proposições, uma forma mais útil e abstrata de raciocínio que Piaget chamou de interproposicional. A mente menos madura olha somente para a relação factual entre uma proposição e a realidade empírica à qual ela se refere; a mente mais madura olha também ao contrário, para a relação lógica entre uma proposição e outra (FLAVELL; MILLER; MILLER, 1999, p. 118).

Na atividade de programação, um aluno irá desenvolver hipóteses sobre como resolver um problema, irá testar diferentes proposições na forma de comandos

e códigos expressos em uma linguagem de programação específica, com os quais irá elaborar os algoritmos, refinando o programa conforme o processo iterativo de depuração. Portanto, o aprendizado de programação pode auxiliar no desenvolvimento das estruturas cognitivas necessárias para que o aluno comece a lidar com o pensamento formal.

Além da teoria construtivista, a robótica e a programação também se fundamentam na abordagem do Construcionismo de Papert. Na crítica às apropriações equivocadas da teoria de Piaget quanto ao posicionamento do pensamento formal num patamar acima do pensamento concreto, no sentido estrito de uma sucessão de estágios, Papert busca resgatar a importância de se trabalhar a qualquer momento a inteligência concreta.

> O Construcionismo é construído sobre a suposição de que as crianças farão melhor descobrindo [...] por si mesmas o conhecimento específico de que precisam; [...]. O tipo de conhecimento que as crianças mais precisam é o que as ajudará a obter mais conhecimento (PAPERT, 2008, p. 135).

Resnick, um dos criadores do Scratch, acrescenta que alinhado ao conceito de programação está a fluência digital, que não deve significar apenas saber conversar em um *chat*, navegar em um *site* e interagir virtualmente, mas também adquirir habilidades de *design*, criar e inventar novas mídias. Para se alcançar este objetivo, é necessário o aprendizado de alguma forma de programação. As habilidades de programação ampliam consideravelmente a faixa do que pode ser criado e utilizado para se expressar com o computador. De forma particular, programação dá suporte para o pensamento computacional, ajudando no aprendizado de resolução de problemas e estratégias de *design* que se movem para domínios de não programação (RESNICK et al., 2009, p. 62).

Assim, a programação pode desempenhar um papel fundamental na transformação das "caixas pretas" em "caixas brancas". Campos (2017, p. 2116) ressalta a necessidade da mudança da metáfora da "caixa preta", assentada na ideia de que a programação de robôs seria uma tarefa muito complexa para uma criança. Blikstein (2013) afirma que as dificuldades percebidas pelos alunos estão muito mais assentadas nos problemas e nas deficiências de *design* das plataformas do que nas possibilidades cognitivas dos próprios alunos.

> Muito diferente desta perspectiva, as metodologias ativas requerem a transição para um design transparente, "caixa branca", dos robôs, onde os usuários podem construir e desconstruir objetos, podem programar robôs e ter acesso profundo às estruturas dos artefatos por eles mesmos ao invés de apenas consumir tecnologias prontas (CAMPOS, 2017, p. 2116).

Talvez esta, assim como o baixo custo, seja uma das razões da aceitação e disseminação da plataforma Arduino nas escolas. A possibilidade de se lidar com uma plataforma aberta, com um vasto conhecimento distribuído, permitindo construções de forma incremental e um acesso mais profundo às montagens, tem se tornado um atrativo formidável na consideração de tal plataforma para auxiliar no ensino e na aprendizagem.

Programação com Arduino

De forma sintética, Arduino é uma plataforma de microcontrolador que, devido à facilidade de uso e à sua natureza aberta, tem alcançado enorme popularidade entre os entusiastas da cultura *maker*. O Arduino permite a realização do que se denomina de "computação física", por meio da conexão de seus circuitos eletrônicos aos seus terminais, visando ao controle de dispositivos, tais como LED e motores, ou para a medição de variáveis, tais como temperatura e luminosidade (MONK, 2013, p. 5).

Ainda de acordo com Monk (2013, p. 6), um microcontrolador pode ser considerado um pequeno computador, contendo elementos que também existem nos computadores pessoais, como memória para guardar programas e dados. Entretanto, uma das diferenças básicas é a disponibilidade direta de terminais que permitem a conexão com outros dispositivos. Esses terminais são chamadas de "portas de entrada e saída" (ou, resumidamente, portas E/S). As portas permitem a leitura ou escrita de dados digitais ou lógicos (por exemplo, identificar se uma chave está ligada ou desligada) ou de dados analógicos (por exemplo, qual a voltagem presente em um pino, qual o nível de luz que está incidindo no sensor).

A origem do Arduino remonta ao seu desenvolvimento na Itália como recurso para auxiliar no ensino de estudantes. Apenas em 2005 foi lançado comercialmente, por Massimo Banzi e David Cuartielles, tornando-se um produto muito bem-sucedido entre fabricantes, estudantes e artistas, por causa da facilidade de utilização e da sua durabilidade. Um fator considerado chave no sucesso do Arduino é a disponibilidade das licenças de forma gratuita (conforme o licenciamento da Creative Commons), o que permitiu o aparecimento de placas alternativas com custo mais baixo (MONK, 2013, p. 6-7).

Quando comparado com um computador pessoal ou *notebook*, um Arduino é extremamente modesto em termos de recursos. Uma das placas mais populares da família é o Arduino Uno, que possui um *clock* de 16 MHz, que indica a veloci-

dade de processamento (comparando com um computador pessoal de 1 GHz, por exemplo, o Arduino é cerca de 60 vezes mais lento), possui memória RAM de 2 kilobytes, utilizada para processar os dados de um programa (cerca de 2 milhões de vezes menor que um computador com 4 gigabytes). Para armazenar os programas para execução, esta placa possui uma memória flash de 32 kilobytes (cerca de 32 milhões de vezes menor que um disco rígido de 1 terabyte). Para permitir a computação física, oferece 28 portas de E/S, possibilitando a conexão com vários dispositivos ou módulos externos.

Com o Arduino na condição de um controlador, é possível o desempenho de uma série de tarefas relacionadas com robótica. A atividade de um robô pressupõe autonomia de ação, para isso é necessário que ele "sinta" o ambiente em que está inserido e "aja" conforme os objetivos que foram estabelecidos para ele. Conforme Mataric (2014, p. 19), "[...] um robô é um sistema autônomo que existe no mundo, pode sentir o seu ambiente e pode agir sobre ele para alcançar alguns objetivos". Dessa forma, ele precisa ter "sensores" para perceber o que acontece no ambiente e coletar informações. Existem sensores de luz, proximidade, temperatura, som e infravermelho. Para agir sobre o mundo, é necessário que o robô seja conectado com "atuadores", tais como lâmpadas de LED, motores e braços articulados.

A modularização tem se constituído num conceito chave para a redução da complexidade nas plataformas de robótica. Por meio da modularização, o nível de conhecimento para o uso de dispositivos eletrônicos não mais requer estudo profundo de eletrônica ou automação. As plataformas oferecem, atualmente, módulos que encapsulam funções mais complexas, fazendo com que operem como "caixas-pretas". Por exemplo, na plataforma Arduino, há o conceito de *shield*: dispositivos com função específica que compartimentam a complexidade dos circuitos, "isolando-os da montagem de alto nível do kit" (MCROBERTS, 2011, p. 26). Portanto, é necessário apenas o conhecimento operacional das entradas e saídas para conectá-los ao sistema em montagem pelo usuário final e fazê-lo funcionar. Como exemplos de *shields*, tem-se os sensores de ultrassom, controladores para motores e conectividade por *bluetooth*.

Para o funcionamento dos circuitos utilizados com o Arduino, é necessário que seja gravado no seu microcontrolador um programa. Um programa é constituído de uma sequência ordenada de comandos para a obtenção de algum resultado. Ou seja, um programa é a implementação real de um algoritmo. Enquanto que um algoritmo tem uma característica mais abstrata e conceitual, um programa é uma realização concreta. Programar é, portanto, uma atividade de escrita de instruções

ou comandos em uma linguagem específica, para a realização de alguma tarefa ou obtenção de um resultado. De acordo com Monk (2013, p. 28), "[...] um programa representa uma lista de instruções, as quais devem ser executadas na ordem em que foram escritas".

A programação para o Arduino é feita utilizando-se um ambiente de desenvolvimento baseado em uma versão da linguagem C adaptada para a plataforma. O programador, após digitar os comandos de um programa específico, faz a compilação (ou seja, traduz o programa para a linguagem do microcontrolador) e depois carrega o programa no Arduino propriamente dito, ficando armazenado na memória *flash*. Esse processo acontece de forma iterativa, e por meio da reflexão o aluno pode comparar os resultados com aquilo que foi planejado.

Com relação à cognição e à aprendizagem, a atividade de programação com o Arduino permite, portanto, estabelecer uma ponte entre o pensamento concreto e o formal. Enquanto a montagem dos circuitos com o Arduino permite lidar com os aspectos de aprendizagem mais relacionados aos elementos físicos, pertencentes à realidade, a atividade de programação incentiva a criação de estruturas cognitivas que permitirão ao aluno lidar com as abstrações oriundas da escrita do código, em linguagem de programação. Essa constatação permite mostrar que a programação com Arduino pode muito bem se adaptar à fase de transição do pensamento concreto para o formal, próprio dos alunos do ensino fundamental II.

Programação em Arduino com Scratch

Seymour Papert teve um papel fundamental na introdução dos computadores na educação, originando a corrente do Construcionismo. A partir da influência dos anos em que esteve ao lado de Jean Piaget e também das pesquisas com inteligência artificial no Massachussetts Institute of Technology (MIT), Papert estabelece uma teoria que compartilha com a noção construtivista sobre o desenvolvimento cognitivo do aluno como um processo ativo de construção/desconstrução de estruturas mentais. No Construcionismo, o aprendizado acontece como um processo ativo no desenvolvimento de projetos, quebrando a visão tradicional de um professor que transmite conteúdos para os alunos (MALTEMPI, 2004, p. 288).

Entretanto, apenas as atividades em que os alunos "colocam a mão na massa" (*hands-on*) não são suficientes. É necessário o envolvimento do aprendiz com aquilo que está fazendo, comprometendo-se também com metas e resultados. Assim,

a abordagem construcionista proporciona maior controle do aluno na definição e resolução de problemas.

> A ideia é criar um ambiente no qual o aprendiz esteja conscientemente engajado em construir um artefato público e de interesse pessoal [...]. Portanto, ao conceito de que se aprende melhor fazendo, o Construcionismo acrescenta: aprende-se melhor ainda quando se gosta do que se faz, se pensa e se conversa sobre isso (MALTEMPI, 2004, p. 288).

Uma das ferramentas mais conhecidas de Papert foi a linguagem Logo, que mostrou como as noções construcionistas podiam alcançar resultados práticos (PAPERT, 2008). Por meio da interação com uma tartaruga virtual na forma de cursor, o aprendiz poderia digitar comandos que permitiam a construção de figuras geométricas. A tartaruga podia "aprender" comandos com complexidade crescente.

> Na utilização do Logo Gráfico, segundo as ideias construcionistas, o aprendiz assume uma postura ativa frente ao seu aprendizado e ao computador e vai, através do desenvolvimento de projetos pessoais, explorando novos conceitos e progredindo em seu próprio ritmo (MALTEMPI, 2004, p. 289).

Na esteira da proposta construcionista do Logo, o Scratch (MALONEY; RESNICK; RUSK, 2010, p. 2) surge como um ambiente de programação visual que permite aos usuários criar projetos interativos e ricos em mídias. Uma aplicação em Scratch é utilizada para criar projetos contendo mídias e roteiros (*scripts*). Imagens e sons podem ser importados ou mesmo criados em Scratch utilizando uma ferramenta de pintura embutida e um gravador de som. A programação é feita por meio do encaixe de blocos de comandos coloridos para controlar objetos gráficos bidimensionais (*sprites*), para movimentarem-se em um pano de fundo chamado de "palco". Os projetos em Scratch podem ser salvos para arquivos no sistema operacional ou compartilhados na página web do Scratch.

Ainda de acordo com a abordagem construcionista, para auxiliar os aprendizes no engajamento de seus projetos pessoais com motivação e significado, o Scratch torna fácil a importação ou criação de muitos tipos de mídias (imagens, sons, músicas), permitindo compartilhamento, recebimento de *feedback* e encorajamento dos seus pares, inclusive para aprender a partir de projetos de outros alunos. Uma característica chave do Scratch é introduzir programação para aqueles que não possuem experiência prévia (MALONEY; RESNICK; RUSK, 2010, p. 3).

Para auxiliar na programação com Arduino e torná-la mais visual, algumas iniciativas têm utilizado o modelo proporcionado pelo Scratch para permitir a escrita de programas em linguagem visual, além da programação textual com a linguagem C. Uma dessas iniciativas é o Tinkercad® – Circuits,[2] criado pela Au-

todesk®, que é um simulador de circuitos eletrônicos que permite a montagem virtual de circuitos com Arduino e uma ampla gama de dispositivos eletrônicos. Além dos componentes, oferece um ambiente de programação que permite manter o programa escrito em linguagem C, em linguagem Scratch, ou mesmo uma combinação das duas linguagens. A simulação reproduz, da forma mais fiel possível, o desempenho dos circuitos, constituindo-se numa ferramenta bastante útil para o aprendizado inicial de programação em Arduino.

Para ilustrar a facilidade de compreensão do Scratch, na Figura 2, encontra-se um pequeno programa, geralmente utilizado como um primeiro exemplo para programar em Arduino, que faz uma lâmpada LED piscar continuamente. Enquanto na esquerda está o programa escrito na linguagem C para o Arduino, na direita está o programa correlato em Scratch, mostrando o apelo visual na programação. No Scratch, o comando em cor azul modifica o estado do LED (aceso para "HIGH" e apagado para "LOW"), enquanto que o comando em cor laranja especifica uma espera de 1 segundo para cada ação. Já na linguagem C, apesar de intuitivo, é necessário o comando na forma de texto.

Figura 2 – Comparação de códigos em linguagem C do Arduino e em Scratch

```
1  void setup()
2  {
3    pinMode(13, OUTPUT);
4  }
5
6  void loop()
7  {
8    digitalWrite(13, HIGH);
9    delay(1000); // Wait for 1000 millisecond(s)
10   digitalWrite(13, LOW);
11   delay(1000); // Wait for 1000 millisecond(s)
12 }
```

Fonte: elaboração dos autores, 2018.

Planejamento do curso de programação

A elaboração do curso de programação foi feita de forma colaborativa, envolvendo o Grupo de Pesquisa Novas Tecnologias de Ensino e Aprendizagem do Programa de Pós-Graduação – Mestrado Profissional em Educação e Novas Tecnologias do Centro Internacional Uninter, juntamente com a Coordenação de Tecnologias da Secretaria de Educação de Curitiba. Após uma reunião preliminar e um curso visando à formação dos professores participantes, foi estabelecida a participação das

escolas que de forma regular estão presentes nas competições de robótica nacionais e mesmo internacionais. Dessa forma, o objetivo do curso seria o de dotar os alunos de mais conhecimento sobre a programação com Arduino, para a eventual ampliação de uso de plataformas prevista para as próximas edições das competições de robótica. Foram escolhidas 9 escolas, prevendo-se a aplicação do curso para 117 alunos, 53% meninos e 47% meninas, mediana de 12 anos de idade e do 7º ano do ensino fundamental. As escolas possuem um professor líder do grupo de robótica, o qual é responsável pelas atividades que o grupo precisa desenvolver, bem como por estabelecer os participantes e os horários de reuniões com os alunos. A maioria aproveitaria os horários de contraturno para o desenvolvimento das atividades de robótica.

A experiência de robótica desses grupos está relacionada com o uso de *kits* da plataforma Lego, a qual se constitui num ecossistema proprietário de peças, componentes e *software* de programação, que não permite a interação com outras plataformas mais abertas. Na intenção de dar oportunidade para o uso de outras plataformas, inclusive com o apelo do baixo custo, a Secretaria de Educação busca incentivar o uso de alternativas, tais como a plataforma Arduino, que cresceu muito nos últimos anos e está sendo adotada como referência para se trabalhar a robótica de baixo custo e a internet das coisas (MONK, 2013).

O curso foi programado para acontecer em cinco dias, envolvendo duas ou três escolas por dia, agendado para a manhã ou a tarde, de acordo com a disponibilidade da escola. Com a carga horária de 3 horas, o conteúdo previsto abordava:

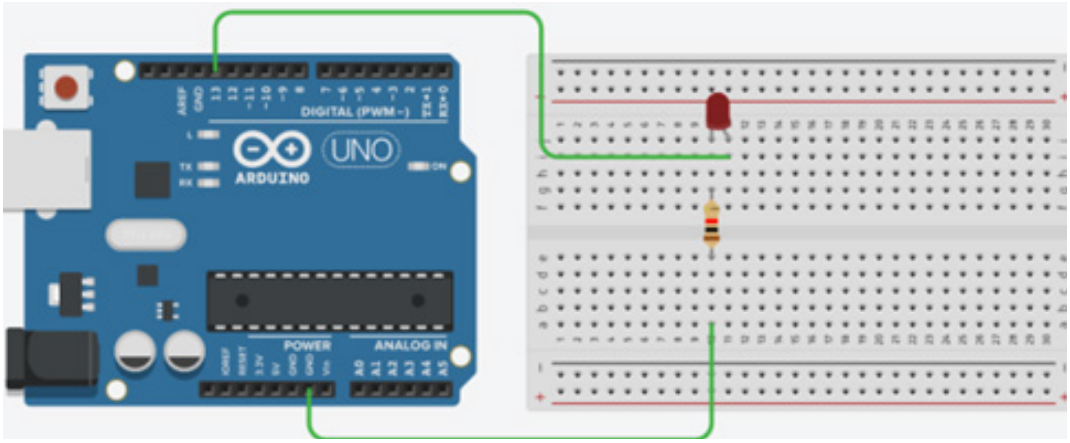
a) o que é programação;
b) a plataforma Arduino;
c) programação textual x visual;
d) tipos de sensores e atuadores;
e) simulador de Arduino: Tinkercad® – Circuits;
f) comandos básicos de programação em Scratch;
g) prática: montagem de LED piscante;
h) desafios: construção de semáforo, rotação de servomotor, sensor de temperatura;
i) montagem física e teste dos circuitos.

As atividades foram desenvolvidas nos laboratórios de informática preparados para o curso de programação. Também, foi fornecido um *kit* para que os alunos pudessem visualizar de forma concreta os dispositivos e os componentes que seriam abordados. O *kit* continha, dentre outros dispositivos:

a) uma placa de Arduino Uno;
b) uma *protoboard* (matriz de pinos ou ilhas para permitir as diversas conexões entre componentes em si e entre o Arduino);
c) lâmpadas LED de diversas cores;
d) sensores de luminosidade, ultrassom e temperatura;
e) servomotor (tipo de motor no qual se fornece um ângulo para girar);
f) resistores para ligar com as lâmpadas LED;
g) fios para conexão.

Após a explicação inicial do Arduino, dos sensores e atuadores, era introduzido o simulador Tinkercad®. Os alunos deveriam fazer o registro na ferramenta e logo depois estavam aptos a desenhar os circuitos. O primeiro circuito que deveriam montar era o LED piscante, feito a partir da demonstração do professor passo a passo, objetivando a familiarização com os aspectos operacionais do simulador, conforme mostrado na Figura 3. Durante a montagem no simulador, o professor explicava que não se podia ligar um LED diretamente na porta, sob pena de danificar (o que era mostrado no próprio simulador), devendo requerer uma ligação por meio de um resistor. Para melhor entendimento do papel de um resistor, o professor mostrava uma resistência de chuveiro, que possui uma função semelhante. Portanto, o aluno deveria primeiro localizar e colocar o Arduino na tela e depois a *protoboard* logo ao lado. Em seguida, deveria colocar a lâmpada LED (o aluno poderia escolher a cor, inclusive) e o resistor na *protoboard*. Ao final, deveria fazer as ligações com os "fios" virtuais nos terminais na ordem certa, para fazer o circuito funcionar. Neste primeiro exemplo, o Tinkercad® já traz embutido o programa visto na Figura 2, de forma que não há necessidade, neste momento, de desenvolver o programa. Os alunos poderiam então analisar o programa e fazer alterações, como variar o tempo de espera para acender ou apagar. Paralelamente à montagem no simulador, eles deveriam reproduzir também a montagem física, utilizando os *kits*.

Figura 3 – Primeira montagem proposta no simulador, um circuito de lâmpada LED piscante



Fonte: elaboração dos autores, 2018.

Na explicação dos sensores, o professor utilizava exemplos do cotidiano para auxiliar na compreensão da sua função. Em um dos sensores abordados, o de luminosidade, era solicitado aos alunos que identificassem na caixa do *kit* qual era o respectivo sensor. Esse sensor era do mesmo tipo daquele utilizado nos postes de iluminação pública, fazendo com que as lâmpadas acendam ao anoitecer e apaguem ao amanhecer. Para o funcionamento do sensor de ultrassom, era feita uma analogia com os sonares dos submarinos.

No segundo momento do curso, eram resumidos os comandos possíveis para utilizar no Scratch do simulador. Como a proposta do Scratch é bastante similar entre diferentes plataformas que oferecem o recurso e também devido à familiaridade dos alunos com o Scratch, tal fase consistia mais numa espécie de revisão. Em seguida, o professor propunha desafios para montar um semáforo com três lâmpadas LED, devendo cada um ficar ligado de acordo com os semáforos de rua. O professor colocava apenas um *slide* mostrando uma sugestão de conexões e os alunos deveriam montar o mesmo circuito, porém, devendo montar também o programa para fazê-lo funcionar.

Outro desafio proposto era a montagem de um servomotor. Esta montagem era mais simples. Entretanto, primeiro era proposto que o servomotor girasse 90 graus, o que é possível utilizando apenas um bloco de comando do Scratch. Na sequência, era proposto que o servomotor girasse 90 graus, porém de 10 em 10 graus. Após a atividade com o simulador, ao final, era explicado como eles deveriam fazer para programar o Arduino fisicamente.

A metodologia adotada para a coleta das informações sobre o desempenho dos alunos foi a observação, centrada na execução das atividades de programação propostas durante o curso. Dois integrantes do grupo de pesquisa, na condição de auxiliares, faziam as anotações relativas ao desempenho dos alunos. Poderia haver alguma orientação para a execução da tarefa, entretanto, sem fornecer os comandos prontos ou as respostas. O grupo de pesquisa contou também com dois alunos de iniciação científica do curso de Engenharia da Computação para suporte e orientação nas montagens.

Na prática com as escolas

Nos dias em que foram agendados os cursos, ao chegarem à instituição, os alunos eram direcionados ao laboratório no qual aconteceria a atividade. Foi solicitado que os alunos formassem pares ou trios para cada computador. Os kits de Arduino eram distribuídos e o professor projetava os *slides* para fazer as explicações sobre os conteúdos.

No primeiro momento do curso, acontecia a explanação sobre o conceito de programação, em que era perguntado sobre a familiaridade com programação e com robótica dos alunos. Foram explorados exemplos do cotidiano para embasar o conceito de algoritmo. Devido à especificidade dos grupos, a maioria era familiar com as montagens utilizando Lego, e apenas aqueles alunos mais recentes nos grupos não possuíam nenhuma familiaridade. Já com relação ao Arduino, a familiaridade foi muito pontual. Entretanto, uma das escolas já iria utilizar um robô construído com a plataforma Arduino para a competição de robótica, cujos alunos mostravam um bom conhecimento da plataforma.

Durante a explicação, o professor mostrava alguns exemplos já montados, como um carro robô seguidor de linha e um cubo de LED. O carro robô permitiu mostrar a utilidade do sensor de luminosidade, enquanto no cubo de LED foi explorado o cálculo da quantidade de lâmpadas LED existentes em cubos de diferentes tamanhos. O cubo mostrado no curso possuía 3x3x3, totalizando 27 LEDs. Também eram mostradas imagens de outros cubos maiores (4x4x4 ou 8x8x8), sendo solicitado que os alunos calculassem a quantidade de LEDs em cada um.

Na simulação com o Tinkercad®, uma das dificuldades encontradas foi o registro prévio dos alunos na ferramenta, pois era exigido que no cadastro fosse inserido o e-mail, sendo que alguns alunos já possuíam. Entretanto, para outros teve de ser feito o acesso à ferramenta a partir da conta dos professores ou auxiliares do

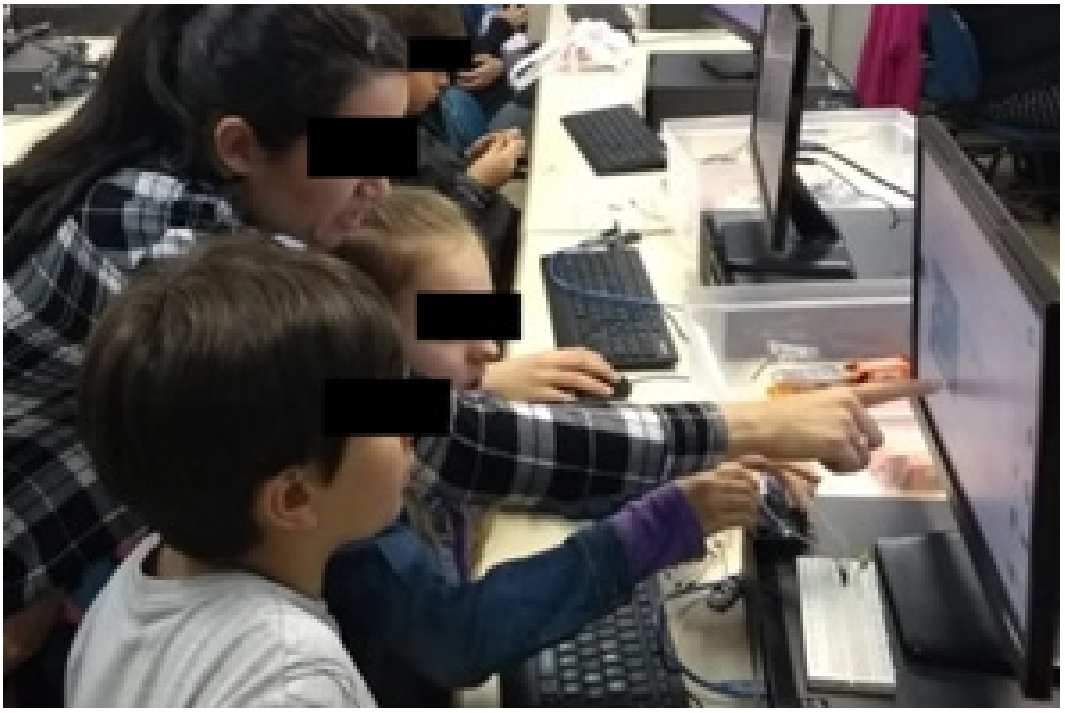
grupo de pesquisa presentes nos cursos. Após concluído o cadastro, boa parte já começava a montar por conta própria alguns circuitos, devido ao caráter intuitivo da ferramenta.

Figura 4 – Alunos criando circuito do semáforo no simulador

Fonte: elaboração dos autores, 2018.

Na primeira montagem, relativa ao LED piscante, era comum os alunos das primeiras turmas errarem a conexão dos fios nos componentes, ligando em ilhas ao lado do lugar em que deveriam ligar. Assim, a precisão de montagem foi um dos cuidados ressaltados nas edições subsequentes do curso. Os professores ou monitores deveriam solicitar apenas que os alunos revisassem onde estava o erro e descobrissem por si próprios (como, por exemplo, a situação na Figura 5). Após a montagem e a simulação de funcionamento, era mostrada aos alunos a aba na qual poderiam ver o programa que fazia o circuito funcionar, tanto no Scratch quanto na programação textual.

Figura 5 – Aluna de iniciação científica na orientação de montagem



Fonte: elaboração dos autores, 2018.

A partir da segunda montagem, o desafio do semáforo, deveria prevalecer a autonomia dos alunos nas montagens e na programação. De maneira geral, os alunos não tinham dificuldades para fazer a montagem, identificando que deveriam controlar três portas para o acendimento dos LEDs, ao invés de uma, do primeiro exemplo (como na situação da Figura 4). O programa a ser feito também seria construído de forma indutiva do exemplo do LED piscante. De forma geral, os programas construídos para o funcionamento do semáforo foram semelhantes ao que se encontra na Figura 6.

Figura 6 – Programa em Scratch para o semáforo

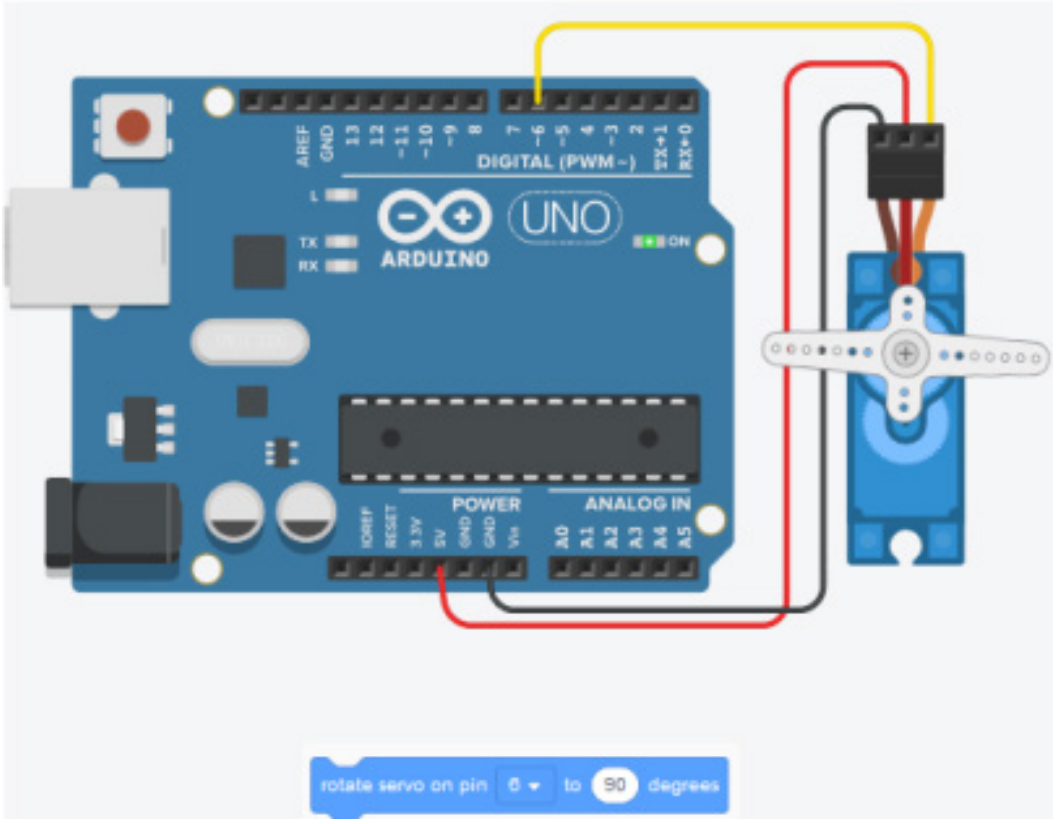
Fonte: elaboração dos autores, 2018.

No processo de depuração do programa, um dos erros mais comuns acontecia quando os alunos esqueciam de apagar um LED, quando se estava acendendo outro. No teste do semáforo, era explorado pelos alunos quanto tempo deveriam programar para os LEDs ficarem acesos na sequência correta. Foi comum a troca da ordem em que os LEDs deveriam acender (vermelho, amarelo, verde).

Na Figura 7, está ilustrado o circuito com o servomotor proposto como terceira montagem. Para o giro de 90 graus do eixo do servomotor, basta um comando em Scratch, que envia a mensagem para a rotação a partir do fornecimento de um ângulo entre 0 e 180 graus. Entretanto, o desafio de montagem para o giro de 90 graus a cada 10 graus decorreu de maneira bem diferente. A tendência dos alunos foi a de montar o programa para girar fazendo a repetição explícita dos blocos

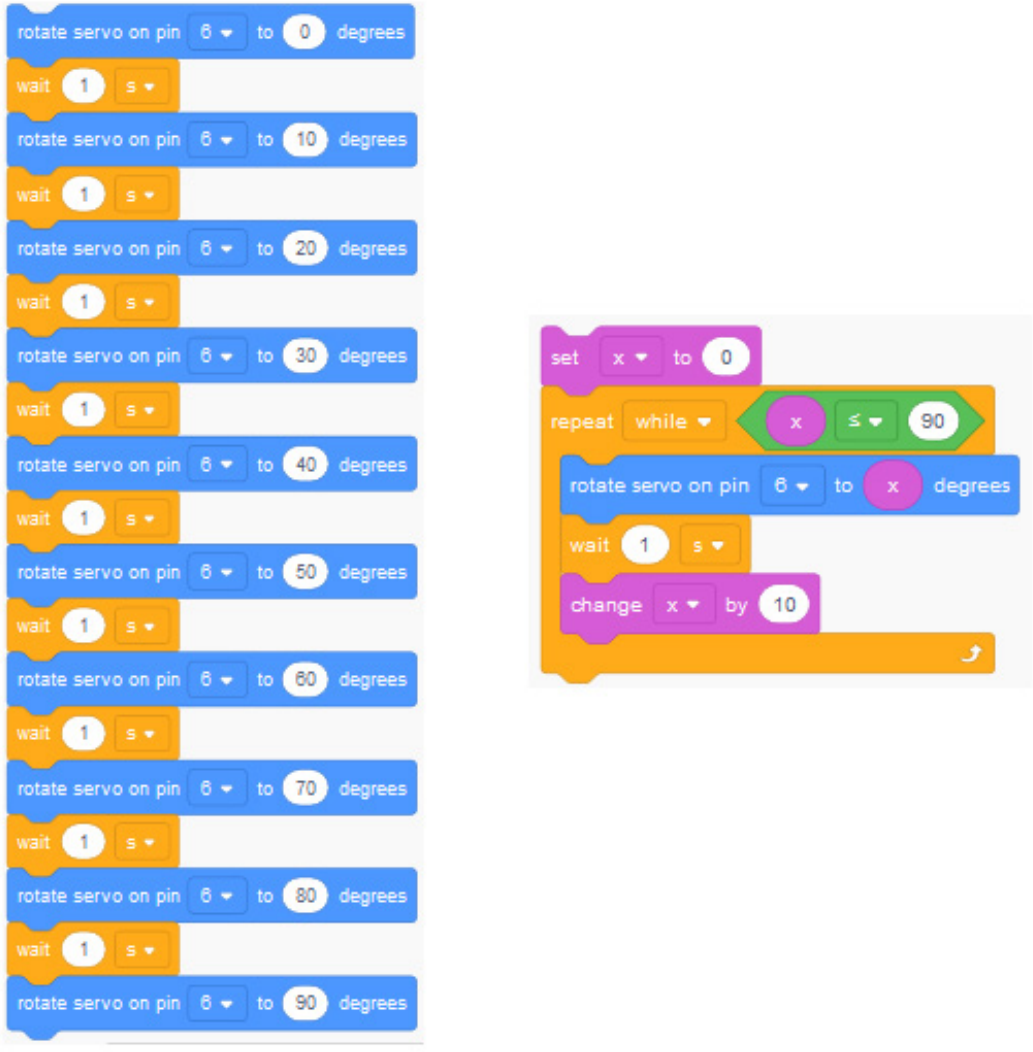
de comando para a execução do giro, como mostrado no programa à esquerda da Figura 8.

Figura 7 – Montagem do servomotor e o bloco em Scratch que executa o giro do eixo



Fonte: elaboração dos autores, 2018.

Figura 8 – Comparação dos dois programas em Scratch para movimentar o servomotor



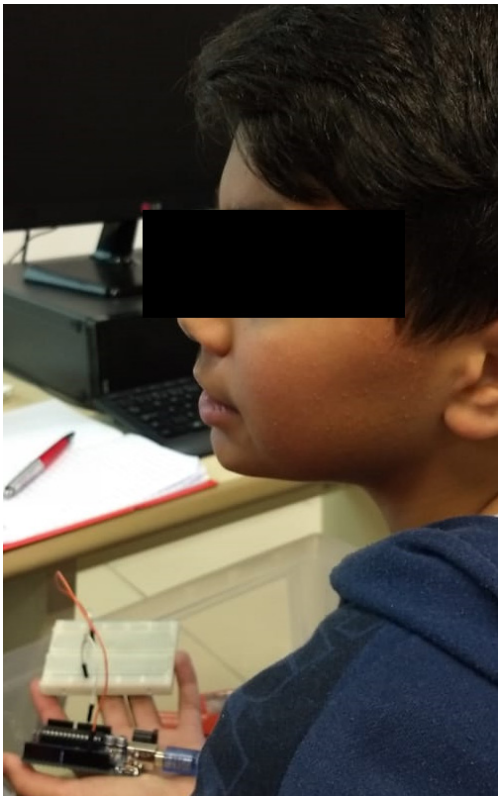
Fonte: elaboração dos autores, 2018.

Apesar disso, tal experimentação é válida, pois, logo em seguida, são demonstrados os comandos que abstraem as estruturas de controle e repetição. Como a quantidade de vezes em que os comandos de rotação se repetem, também é oportuna a explicação sobre o uso de variáveis no Scratch, além das expressões de comparação. O programa à direita da Figura 8 mostrou uma solução para o problema, permitindo também o aprendizado de elementos importantes para a execução de

programas. A explanação mostrava que era necessário criar uma variável para contar quantas vezes o comando seria enviado ao servomotor (no exemplo, a variável "x"). Esta variável, no início, deveria receber o primeiro valor (no caso, 0 grau). Depois, era necessário colocar uma estrutura de repetição "enquanto" (comumente chamado de *loop* "*while*"), na qual seria comparado o valor da variável (se chegasse a 90 graus). Esta estrutura permitiria a economia de código com relação ao programa anterior. Internamente à estrutura de repetição, figurava então o comando para a rotação propriamente dita, a espera de um segundo e o comando que alterava o valor (de 10 em 10 graus). Este exemplo ilustra bem o aspecto de transição do pensamento concreto para o formal, mencionado anteriormente. Os comandos para evitar a repetição sucessiva exigiam um novo nível de abstração por parte dos alunos, envolvendo o uso de variável numérica em substituição às constantes utilizadas no exemplo anterior; a comparação numérica para o controle do ângulo para girar; e a operação de incremento da variável a cada ciclo de repetição.

Apesar da receptividade com a fase de simulação do Arduino, de forma geral, restava pouco tempo para que os alunos pudessem aprender a utilizar a ferramenta para carregar os programas no Arduino físico. Assim, era solicitado que os alunos trouxessem a montagem para que os professores fizessem o carregamento do programa no Arduino para demonstrar o funcionamento (Figura 9). Apenas uma das turmas que fez o curso pela parte da manhã conseguiu fazer o experimento com o sensor de temperatura.

Figura 9 – Aluno com o circuito do LED piscante pronto para teste no Arduino



Fonte: elaboração dos autores, 2018.

Como ponto positivo, pode-se evidenciar a motivação dos alunos para aprenderem programação considerando algo concreto como o Arduino (dentre alguns comentários, destacam-se: "gostei de colocar a mão na massa"; "gostei do Arduino e do simulador"). Ainda que a simulação tenha preenchido a maior parte do tempo dos cursos, quando perguntado aos alunos no final sobre o que mais haviam gostado do curso, a maioria falou bem com relação ao simulador Tinkercad®. Como a maior parte dos alunos já estava familiarizada com a robótica em outra plataforma (em torno de 2/3 dos alunos já haviam montado robôs com Lego), o aprendizado foi bastante facilitado. Outro aspecto considerado positivo foi a colaboração entre os pares ou trios, em que se verificava que os próprios alunos faziam a distribuição das atividades a serem feitas (enquanto um se ocupava de montar no Tinkercad®, outro fazia a montagem física). Em média, 2 a 3 alunos por turma perguntavam,

ainda no final do curso, quanto custava um *kit* de Arduino, pois ficavam tão empolgados que pensaram em pedir para os pais comprarem.

Com relação aos pontos negativos, constatou-se uma maior dificuldade de alguns alunos. Quando perguntado aos alunos sobre o que havia sido mais difícil de aprender, alguns falaram da parte de programação propriamente dita ou da linguagem C (em torno de 27% dos alunos), enquanto que outros falaram mais da dificuldade na parte matemática (em torno de 9%), ainda que aproximadamente a metade dos alunos tenha como preferência a disciplina de Matemática. Uma constatação a respeito disso é que o aprendizado de linguagem nesta fase sempre deverá ser acompanhado da parte visual, a qual facilita muito a compreensão, se a intenção é, mais tarde, ensinar linguagens mais complexas ou textuais. O aspecto do tempo também foi um ponto a considerar, pois algumas turmas conseguiam ter um aproveitamento maior em relação a outras.

Considerações finais

Apesar da receptividade dos alunos e, de modo geral, da facilidade do aprendizado de programação com o Arduino, ficou evidenciada a necessidade de dar sequência ao curso, pois os alunos ficaram ansiosos para aprender mais e fazer mais montagens. Outro aspecto que poderia ser mais explorado consiste na associação das construções feitas com o Arduino e dos programas com conteúdos curriculares das disciplinas. Como assinala Campos (2017, p. 2119), a robótica é somente mais um recurso, pois o currículo é que deve determinar o resultado da aprendizagem e a sincronia da tecnologia com as teorias de aprendizagem.

Outra necessidade evidenciada refere-se à preparação prévia dos professores líderes dos grupos de robótica. Notou-se que alguns estavam mais envolvidos do que outros e, portanto, ficaram mais à vontade no laboratório para tirar dúvidas dos alunos ou para interagir com outros professores ou monitores. Na sequência do projeto, está prevista a formação com o curso de programação no Arduino para esses professores, no sentido de prepará-los melhor e também para que relacionem de forma adequada as práticas de construção dos circuitos com as necessidades curriculares, antes da próxima fase do curso de programação com os alunos.

O curso a ser feito na sequência contará com a programação direta no Arduino, com associação mais aprofundada entre os programas escritos em Scratch e em linguagem C, bem como o uso da plataforma S4A (Scratch for Arduino), a qual permite a gravação direta dos programas em Scratch no Arduino. No que tange ao

conteúdo de programação, além das estruturas de repetição, serão exploradas também as estruturas condicionais, o mapeamento para demonstrar funções e o uso mais detalhado de variáveis. Será adotada, ainda, uma metodologia de avaliação, visando analisar a forma como os programas foram escritos e também comparar os diferentes programas construídos entre os alunos.

Notas

[1] Disponível em: <https://www.todospelaeducacao.org.br/pag/dados-5-metas>. Acesso em: 20 ago. 2018.
[2] Disponível em: <www.Tinkercad®.com/circuits>.

Referências

BLIKSTEIN, P. Digital fabrication and "making" in education: the democratization of invention. In: WALTER-HERRMANN, J.; BOCHING, C. (Ed.). *FabLabs*: of machines, makers and inventors. Bielefeld: Transcript Publishers, 2013. p. 1-21.

CAMPOS, F. R. Robótica educacional no Brasil: questões em aberto, desafios e perspectivas futuras. *Revista Ibero-americana de Estudos em Educação*, Araraquara, v. 12, n. 4, p. 2108-2121, 2017.

COMPUTER SCIENCE TEACHER ASSOCIATION. *CSTA K–12 Computer Science Standards*. New York: Computer Science Teachers Association Association for Computing Machinery, 2017.

CORMEN, T. et al. *Algoritmos*: teoria e prática. Rio de Janeiro: Campus, 2002.

FLAVELL, J. H.; MILLER, P. H.; MILLER, S. A. *Desenvolvimento cognitivo*. Porto Alegre: Artmed, 1999.

GROVER, S.; PEA, R. Computational thinking in K–12. *Educational Researcher*, Thousand Oaks, v. 42, n. 1, p. 38-43, 2013.

INSTITUTO NACIONAL DE ESTUDOS E PESQUISAS EDUCACIONAIS ANÍSIO TEIXEIRA. *SAEB – Sistema de Avaliação da Educação Básica*. 2017. Disponível em: <http://portal.inep.gov.br/web/guest/educacao-basica/saeb/resultados>. Acesso em: 14 ago. 2018.

KAZIMOGLU, C. et al. Learning Programming at the Computational Thinking Level via Digital Game-Play. *Procedia Computer Science*, Treton, v. 9, n. Supplement C, p. 522-531, 2012.

LEFRANÇOIS, G. R. *Teorias da aprendizagem*. São Paulo: Cengage Learning, 2013.

MALONEY, J.; RESNICK, M.; RUSK, N. The Scratch programming language and environment. *ACM Transactions on Computing Education*, New York, v. 10, n. 4, p. 1-15, 2010.

MALTEMPI, M. V. Construcionismo: pano de fundo para pesquisas em informática aplicada à educação matemática. In: BICUDO, M. A. V.; BORBA, M. C. (Ed.). *Educação matemática*: pesquisa em movimento. São Paulo: Cortez, 2004. p. 264-282.

MARTINELLI, L. M. B.; MARTINELLI, P. *Materiais concretos para o ensino de matemática*. Curitiba: Intersaberes, 2016.

MATARIC, M. *Introdução à robótica*. São Paulo: Unesp; Blucher, 2014.

MCROBERTS, M. *Arduino básico*. São Paulo: Novatec, 2011.

MONK, S. *Programação com Arduino*. Porto Alegre: Bookman, 2013.

ORGANIZAÇÃO PARA A COOPERAÇÃO E DESENVOLVIMENTO ECONÔMICO. *Pisa Brazil*. 2015. Disponível em: <http://www.oecd.org/pisa/pisa-2015-Brazil-PRT.pdf>. Acesso em: 1 set. 2018.

PAPERT, S. *A máquina das crianças*: repensando a escola na era da informática. Porto Alegre: Artmed, 2008.

RESNICK, M. et al. Scratch: programming for all. *Communications of the ACM*, New York, v. 52, p. 60-67, 2009.

WING, J. M. Computational thinking. *Communications of the ACM*, New York, v. 49, n. 3, p. 33-35, 2006.