

Code Dominó - Linguagem de programação tangível e expansível para Educação STEM

Daniel Chagas

prof.daniel.chagas@gmail.com
Universidade de Fortaleza
Fortaleza, Brazil

Elizabeth S. Furtado

Universidade de Fortaleza
Fortaleza, Brazil
elizabethsfur@gmail.com

ABSTRACT

Tangible programming in the educational environment has benefits compared to traditional virtual programming, such as promoting collaboration and less abstraction in algorithm presentation. The LOGO programming language is the basis for tangible programming initiatives, however its modification to the tangible form brings questions who impact on classroom practices, about command complexities and solution cost of interactive objects. Code Domino's tangible programming platform was designed to minimize this issues, using IoT (Internet of Things) technologies to better adapt LOGO language to a tangible format, presenting itself as a accessible and scalable solution to teach programming to kids and adolescents.

CCS CONCEPTS

• **Applied computing** → **Interactive learning environments.**

KEYWORDS

stem, education, tangible user interface, logo, human-computer interaction, tangible programming

1 RESUMO

A programação tangível no ambiente educativo traz benefícios sobre a programação virtual, como o incentivo à colaboração e a apresentação de algoritmos de forma menos abstrata. A linguagem LOGO é a base para iniciativas de programação tangível. Porém seu uso adaptado à forma tangível traz questões que impactam na prática em sala, sobre complexidade dos comandos e de custos das peças e objetos interativos. A plataforma Code Dominó de programação tangível foi desenvolvida buscando minimizar estas questões, utilizando tecnologia IoT (Internet das Coisas) para adaptar a linguagem LOGO à tangibilidade, apresentando-se assim como uma solução acessível e escalável para o ensino de programação para crianças e adolescentes.

2 INTRODUÇÃO

Acompanha-se toda uma tendência mundial do chamado ensino STEM - Ciência (Science), Tecnologia, Engenharia e Matemática. Essa tendência visa tratar a educação de forma

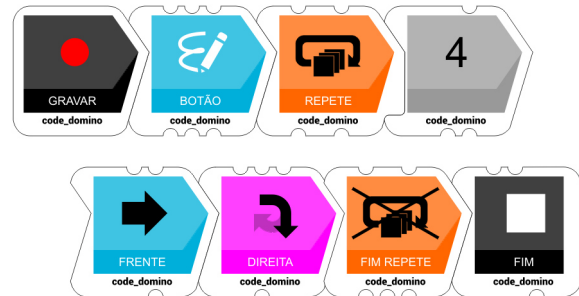


Figure 1: Code Dominó Tangible Programming Interface.

mais prática e aplicada, desde o ensino fundamental até o superior. O documento criado pela Casa Branca dos EUA, A Nation of Makers[29], declara a importância do ensino tecnológico para fomento da inovação e pesquisa nos Estados Unidos da América, e coleta várias iniciativas de instituições de ensino americanas para o assunto. Dentre elas, está o ensino de programação para idades cada vez menores, a aplicação de robótica e automação para ensino, e a criação de ferramentas específicas para auxiliar o ensino.

Seymour Papert, guiado pelos trabalhos de Vygotsky e Piaget sobre construtivismo, cunhou o termo que embasa muitos trabalhos de ensino tecnológico: Construcionismo[5, 22]. Na perspectiva construcionista, a metacognição da criança (o pensar sobre o pensar) no ato de criar algo é chave do entendimento particular, já que sua criação deriva de seu próprio mundo [5]. Papert discorre que o ato de programar, com sua sequência de passos orquestrados para um fim, porém particulares de cada criança, favorecia esta metacognição. O construcionismo segue inspirando projetos, através de sua linguagem de programação LOGO[5, 22], onde crianças digitam comandos em um computador que movimentam uma tartaruga (virtual ou robótica).

Uma dificuldade inerente ao ato de ensinar programação é o caráter abstrato da lógica. Uma solução é a adoção de práticas para 'materializar' os conceitos de programação, como programação em blocos e robótica[16, 28]. Tais abordagens

têm mostrado dados promissores[5, 7, 12, 14, 20], principalmente quando adotados em idades menores. A esta materialização chamamos de interfaces de usuário tangíveis, ou TUIs¹. Seu uso no ensino de programação para crianças, em geral adaptando a abordagem de movimentar um artefato interativo através de comandos da linguagem LOGO, desponta como uma série de soluções acadêmicas e comerciais para ensino.

Porém essas abordagens apresentam problemas de limitação no uso e custo. A limitação no uso se dá pela opção dos desenvolvedores de aplicar a linguagem LOGO de forma bastante simplificada[26], sem comandos mais complexos, variáveis, passagem de parâmetros ou uso de condicionais. Os projetos de TUIs para programação limitam-se em geral aos comandos de movimentação. Esta solução não condiz com uma das premissas de Papert sobre o Construtivismo[5], de que a programação não deve ser limitante ou tratada como entretenimento. Para Papert o Construcionismo está no entendimento e apropriação da programação pelo aluno, e a compreensão dele de que ele pode ampliar suas capacidades de solucionar problemas.

A questão de custo está diretamente ligada à tecnologia de materialização dos comandos. Peças com eletrônica embutida fazem com que o custo por peça torne a solução inviável para grupos de alunos. Soluções baseadas em símbolos impressos nas peças exigem equipamentos com câmera (como celulares ou tablets) para a leitura, execução e/ou transmissão dos comandos, o que pode inserir um grau de complexidade a mais na interação tangível.

Diante desta perspectiva, o presente artigo visa apresentar a programação tangível no ambiente educacional, os problemas em implementações passadas e uma solução baseada em tecnologias de Internet das Coisas (IoT), chamada Code Dominó.

O artigo se divide em cinco partes: Introdução; o Estado da arte do tema interações tangíveis, sua aplicação no ensino de programação e problemáticas; a apresentação da solução Code Dominó, seu funcionamento e diferenciação de outras plataformas; a avaliação da solução; e por fim, as conclusões obtidas a partir de todo o desenvolvimento deste estudo.

3 INTERAÇÃO TANGÍVEL

Durante toda a sua história o ser humano ampliou suas capacidades manipulando ferramentas. Através do agarrar e manipular de objetos, usuários do passado desenvolveram ricas linguagens e culturas, que valoram a **interação háptica** com objetos reais. Estes artefatos físicos e seus *affordances* (capacidade que um objeto tem de ser reconhecido e utilizado sem a necessidade de uma explicação prévia) serviram de inspiração para várias formas de interação modernas[16],

que aos poucos foram migrando para a interação virtual. Em 1981, no Xerox PARC, a metáfora usada para a criação do primeiro sistema de computador com apontador (mouse) foi a de uma mesa de trabalho (*desktop*), ou seja um ambiente do mundo real, definindo importantes princípios da interação familiares como o ver-e-apontar e o WYSIWYG (o que você vê é o que você tem)[16].

A interação com o mundo virtual (ou *ciberespaço*[16]) consolida-se com a criação e massificação das interfaces WIMP (Windows ou janelas, ícones, menus e ponteiro) nos anos 1990, reforçada em seguida pelas interações táteis no final dos anos 2000, com a inserção de telas de toque e os movimentos de pinçar, ampliar, arrastar e segurar[24]. A manipulação passou a ser intermediada por "uma carreira de bits que vazam do ciberespaço através de uma miríade de telas retangulares na forma de feixes de fótons no mundo real"[16]. Neste cenário Ishii e Ullmer (1997) levantam a necessidade de retornar a riqueza do mundo real na Interação Humano-Computador (IHC). A esta forma de interação chamaram de **interfaces de usuário tangíveis** (TUI - *Tangible User Interfaces*).

As TUIs ampliam o mundo real ao acoplar informação digital aos objetos físicos e ambientes do dia a dia. Trata-se de um novo paradigma de interação com informação digital onde usuários se relacionam com a informação pela manipulação dos dados com as próprias mãos[28]. Ishii e Ullmer previram ainda em 1997 a computação ubíqua através da Internet das Coisas (IoT)[3, 24], onde seus conceitos chave eram as superfícies interativas (pontes entre o virtual e o real), o acoplamento entre bits e átomos, e a mídia ambiente (uso da percepção periférica humana como mídia)[16]. A maior vantagem de uma TUI é que a **interação é mais natural** comparada a interfaces convencionais, além deste tipo de interface permitir a interação de múltiplos usuários e múltiplos sentidos do usuário[28].

As interfaces tangíveis compartilham três características destacáveis[26]:

- Não há um local único de controle ou interação: Diferente de uma interface WIMP (onde o ponteiro é o *locus* da interação) as possibilidades de interação estão distribuídas e podem ser coordenadas ou combinadas;
- Não há uma sequência restrita da interação no tempo, e o usuário pode iniciar onde quiser;
- O projeto de interação usa intencionalmente *affordances* dos objetos para guiar como a interação deve ocorrer (Ex.: peças de formato que só se conectam de uma dada maneira, criando uma restrição física).

Na figura 2, Ishii e Ullmer[16] ilustram exemplos da instanciação física entre uma GUI (interface de usuário gráfica)

¹Tangible User Interfaces

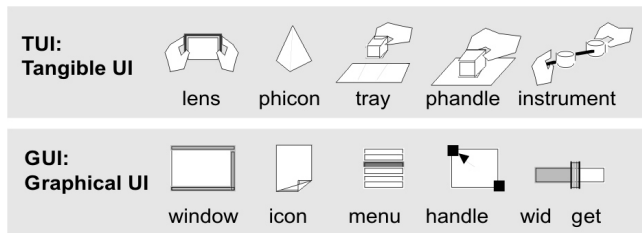


Figure 2: Estanciação física dos elementos de uma GUI em uma TUI [16].

e uma TUI, trazendo a tona termos novos para indicar essa correlação, como o ficone² (*phicon*) ou ícone físico.

Em um sentido mais amplo, a área de estudo das TUI é um componente do framework unificado **interação baseada na realidade**, proposto por Jacob et al.[17, 26] que combina uma série de estilos de interação e visões emergentes como uma nova geração (IHC), incluindo realidade virtual, realidade aumentada, computação ubíqua e pervasiva, interação manual e interação tangível.

Ao projetar a interação sobre conhecimentos e habilidades preexistentes, estes estilos emergentes de interação reduzem o chamado *golfo de execução* da engenharia cognitiva[24, 26], o "espaço" entre o objetivo do usuário na interação e os meios para executar esse objetivo.

TUIs na Educação

O contexto educacional tem muito a ganhar com o uso de TUIs[28], dado suas características intuitivas. Neste contexto, as TUIs são classificadas como ferramentas computacionais de apoio ao ensino e se apresentam comumente como brinquedos com características "aumentadas" que amplificam sua atratividade e funcionalidade [26]. Para pequenos aprendizes, o aspecto cinestésico da aprendizagem é importante, já que estes aprendem através do toque, visão, audição, tato e olfato [7, 12].

Uma aplicação chamativa das TUIs na educação infantil se dá no desenvolvimento do *storytelling* ou narrativa[5, 26]; a capacidade do sujeito de se expressar, contar uma história, utilizando recursos disponíveis (fala, mímica, objetos). Outras aplicações são o ensino por manipulação (blocos de construção, materiais montessorianos), simulações, e ensino de crianças com necessidades especiais[26].

TUIs para programação

A programação tangível nasce com o trabalho de Perlman em 1976 com uma interface de ranhuras[5, 26] onde crianças programavam sua tartaruga virtual em LOGO[22], usando cartões dispostos em sequência em espaços de uma mesa. A utilização de objetos físicos para programação permite a

²Tradução do autor.

criação de uma sintaxe física que adere à sintaxe lógica dos comandos, através de restrições físicas como encaixes, uso de ímãs, rotação, etc. O projeto TERN[14, 26] por exemplo, usa cubos de madeira com furos e pinos que definem a forma de encaixe dos comandos para a movimentação de um robô (figura 3).



Figure 3: TUI de madeira do projeto TERN [14].

Outra forma de programação tangível é através de demonstração ou ensaio[26], onde o usuário manipula o próprio objeto de saída do resultado (robô, braço robótico, etc.) "mostrando" como este deve atuar no tempo. Trata-se de uma programação direta, onde a interface de entrada e saída estão no mesmo objeto, porém restrita a movimentos (sem uso de matemática ou lógica booleana).

O entretenimento e a livre exploração é muitas vezes apresentado como o fator principal para a popularidade da programação tangível. O estudo comparando programação visual com programação tangível no Museu de Ciências de Boston[14] traz evidências que crianças são mais abertas e tem mais engajamento com a abordagem tangível (com um impacto substancial no público feminino) do que com interfaces virtuais. Esta evidência mostra que programação tangível cuidadosamente desenhada pode de fato oferecer benefícios educacionais concretos[26].

Problemas em Implementações de Programação Tangível

A programação tangível contudo tem problemas com o avanço da complexidade do que é programado: Algoritmos complexos demandam mais "comandos" (peças físicas) e mais espaço físico para a programação. Esta limitação é deixada de lado por projetistas[26] que usam a programação tangível somente no nível introdutório, sem a maioria das estruturas que compõem uma linguagem de programação atual (condicionais, operações lógicas, operações aritméticas, variáveis, funções, etc.).

Outra questão é o custo por peça da interface tangível de programação. Peças que contenham componentes eletrônicos no seu interior tornam-se muito caras, e ter em quantidade suficiente para um grupo de crianças pode ser oneroso para escolas e governos[21]. Soluções como os P-Bricks e Crickets[21], blocos de lego contendo circuitos programáveis e baterias próprias, mostram-se versáteis ao se conectar diretamente com pelas de Lego, porém inviáveis de se ter em sala de aula em quantidades viáveis.

Soluções usando marcações (código de barras, QR Codes, cores, etc.) são acessíveis, porém jogam o custo para a leitura, já que demandam equipamentos com câmeras ou sensores mais caros. A praticidade da leitura dessas marcações também é questionável:

- No projeto Kibo[5] os estudantes precisam segurar o robô sobre cada uma das peças para efetuar uma leitura de código de barras com um sensor exclusivo.
- No projeto Tern[14] um código especial chamado Top-Code (*Tangible Object Placement Code*) foi criado para simplificar a leitura, porém este precisa ser lido por um dispositivo com câmera para depois ser executado ou transferido a um robô.
- Soluções que rodam em dispositivos móveis como o Strawbies[15] e Osmo[27] usam espelhos na câmera frontal do dispositivo (em geral um iPad) para que o dispositivo identifique as peças. Porém o ângulo de visão permite poucos comandos no campo de visão.
- O projeto Cubetto[20] traz peças simples, porém que são usadas com uma mesa com encaixes, que faz a leitura. A limitação se dá com o número de peças possíveis e o número de encaixes na mesa (16 espaços).

As limitações desses projetos vão contra a um dos pensamentos principais do Construcionismo de Papert [5, 22], de que a linguagem não deve impor limites. Para Papert, a fluência tecnológica só é alcançada com o ato de programar de fato. Para ele, abordagens ligadas a puzzles e desafios (como o de instruir um personagem a sair de um labirinto com uma sequência de movimentos) não dão a liberdade à criança de fato criar seus artefatos. "É como oferecer uma aula de pontuação e gramática, e não permitir que os estudantes criem suas próprias histórias" [5, 22]. Papert comparava o ensino de lógica de programação com a alfabetização, já que ambos permitem a criação de artefatos que estão destacados de seu criador: Textos no caso da alfabetização, e algoritmos no caso da programação [5]. Ou seja, para Papert a programação é uma ferramenta cognitiva [1, 10].

Bers[5] faz um paralelo entre Papert e Paulo Freire, e sua pedagogia do oprimido. Freire traz a alfabetização como uma ferramenta para a libertação, e não somente uma habilidade prática. Assim como a alfabetização, a programação dá ao usuário as ferramentas intelectuais para ter voz e um papel na

sociedade civil. A valorização da programação, equiparável à alfabetização, é condizente com os rumos do ensino de Pensamento Computacional, consolidado com Wing[30, 31], que o definiu como uma série de atitudes e habilidades que uma pessoa deve ter para, com confiança, persistir na identificação, absorção e solução de problemas. Educadores levantam que a importância do estudo de pensamento computacional não mais está ligado à áreas de exatas, e que seus componentes principais (decomposição, busca de padrões, abstração e automação) devem ser perseguidos em qualquer profissão [5, 11, 19].

Outro ponto importante e muitas vezes não abordado por pesquisas e produtos para educação STEM é o apoio ao professor. Em um artigo sobre barreiras da integração de tecnologias no ensino, Bingimlas[6] cita a falta de treinamento do professor com a tecnologia, a falta de confiança do professor no uso, e a resistência do professor em mudar. Esses pontos devem ser colocados como prioritários no desenho de soluções para educação STEM.

4 SOLUÇÃO TANGÍVEL

Dois objetos interativos foram desenvolvidos para atender as necessidades de programação tangível para educação: uma linguagem de programação tangível chamada Code Dominó (Fig. 1), e um robô que lê e executa a linguagem criada.

O desenvolvimento de ambas as soluções seguiu requisitos para ensino de lógica de programação extraídos por revisão sistemática, de Chagas et. Al. [8]. O requisitos apresentados são:

- Apresentação de comandos não como texto mas como blocos de comandos reais ou virtuais;
- Execução dos códigos devem gerar divertimento (movimento, sons, luzes, etc.) usando dispositivos físicos;
- Encorajar a colaboração entre estudantes na resolução de desafios;
- Acessibilidade e dos dispositivos para diferentes perfis de estudantes e professores;
- Custo acessível para diferentes contextos escolares.

Os objetos interativos também seguiram os guias de projeto de linguagens de Bers e McNerney[5, 21].

Linguagem de Programação Code Dominó

A linguagem de programação Code Dominó é apresentada como um conjunto de peças cortadas a laser, em material MDF ou acrílico, acrescidas de imagens alusivas aos comandos que executa. O armazenamento dos dados é feito através de etiquetas adesivas de RFID (identificação por radiofrequência). A etiqueta RFID contém uma antena e um chip de silício. A leitura se dá através de uma placa de leitura de baixo custo (<US\$ 4). A alimentação é recebida por ondas de rádio, que energizam o chip e retornam as informações

armazenadas. Cada adesivo possui um número identificador único.

Esta união de materiais e tecnologia IoT permite um conjunto de peças de baixo custo porém extremamente versáteis eletronicamente. A tecnologia RFID para programação tangível foi utilizada com sucesso por Carbajal e Branauskas[7] trazendo excelente confiabilidade de leitura e baixo custo (tanto por peça quanto o custo do leitor), porém servindo apenas como identificador das peças. O adesivo RFID tem entre 300Kb e 900Kb de armazenamento disponível para o usuário, que é usado no Code Dominó para armazenar o conjunto de comandos da peça, e está acessível aos usuários.

O formato das peças buscou affordances que facilitem a iniciantes a programar. Seu encaixe serve como guia do fluxo do algoritmo, e permite que as peças sejam manipuladas e alinhadas formando o código. Marcações nas laterais em braille garantem a acessibilidade para alunos cegos (Figura 4).

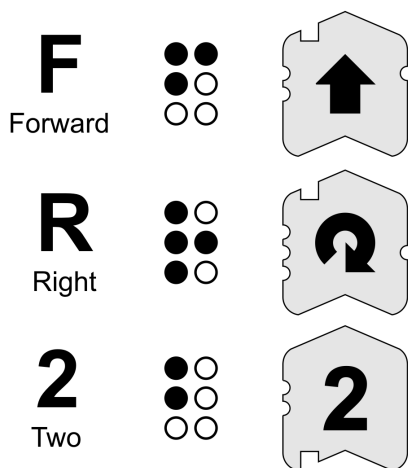


Figure 4: Formato das peças Code Dominó, com destaque as marcações em Braille (Fonte do autor).

As peças são divididas em dois conjuntos: comandos e parâmetros. No conjunto de comandos temos os básicos de movimentação do Logo (frente, direita, esquerda, inicia desenho, para desenho), operações aritméticas, comparações (<, >, =, !=), repetições e condicionais, além de peças especiais representando os sensores e atuadores do robô. Os parâmetros são números ou variáveis que, associados à comandos, mudam sua forma de atuar. A figura 5 mostra um algoritmo montado com peças de Code Dominó.

Programação com Code Dominó. A programação através do Code Dominó ocorre posicionando as peças em uma fila, com os comandos a serem executados da esquerda para a



Figure 5: Algoritmo montado com a linguagem de programação tangível Code Dominó (fonte: do autor).

direita. O código abaixo faz o robô desenhar dois vértices de um quadrado com 10cm de lado:

```
[inicio][desenha][frente][direita][frente][fim]
```

Os parâmetros são opcionais, e são entradas numéricas colocadas a direita dos comandos, passando informações extras de como o comando deve se comportar. O exemplo abaixo faz o robô desenhar duas linhas de 5cm em um ângulo de 45°:

```
[frente][5][esquerda][4][5][frente][5]
```

Comandos de repetição da computação também estão presentes, permitindo que ciclos complexos possam ser construídos com poucas peças. O ciclo abaixo desenha um quadrado:

```
[repita][4][frente][direita][fim-repita]
```

Variáveis globais são representadas por peças com letras. Elas podem substituir parâmetros quando posicionadas à direita de um comando:

```
[frente][X]
```

O valor de uma variável é mudado com a peça VAR seguida da variável. O código a seguir atribui o valor 45 à variável X:

```
[var][X][4][5]
```

Existem blocos sensores, que retornam parâmetros para os comandos (similar à uma variável). O robô pode ser dotado de diversos sensores (luz, distância, presença, som, etc.) e cada um deles tem um bloco associado. O código abaixo fará o robô andar a quantidade de centímetros que o sensor de distância medir (se o sensor medir 12cm, ele irá caminhar 12cm):

```
[frente][sensor-distância]
```

Código expansível. Duas características do Code Dominó dão versatilidade à interação, permitindo que algoritmos complexos sejam programados através de TUIs. São a criação de funções/procedimentos, e a criação de novas peças pelos próprios usuários.

Funções são blocos de código reutilizáveis pelo programa. São fruto da atividade de decomposição de um problema em problemas menores, da busca de padrões e da abstração. Ao construir um programa/algoritmo em Code Dominó, o usuário inicia construindo cada uma das "reações" do seu robô ao ambiente. Ex:

```
[aprenda] que [meia-volta] é [direita][180]º
[aprenda] que [obstáculo] é [sensor dist.][<][5]cm
[se] [obstáculo] [então] dê [meia-volta]
```

O usuário inicia a programação de seu robô lhe "ensinando" conceitos básicos, como dar meia-volta, ou dizer o que é um obstáculo. Esta abordagem é inspirada no efeito Protégé[9], onde um estudante tem um melhor desempenho e engajamento de seus estudos quando se torna o tutor ou professor, no chamando "aprendizado por ensino" (*learning by teaching*)[23, 25]. O usuário torna-se então tutor de seu agente aprendiz (no caso seu robô), que aos poucos vai armazenando em sua memória EEPROM³ uma série de "conceitos" (funções) que podem ser chamados por seus programas.

A segunda característica que permite a expansão do código é chamada de **code craft** (inspirada no jogo Minecraft). Trata-se da possibilidade do usuário de unir um conjunto de peças e "moldar" uma nova, inexistente antes no conjunto. Isso permite a reutilização de peças, bem como a colaboração entre usuários, emprestando conjunto de códigos facilmente. Esta função é possível pela capacidade da peça RFID de armazenar dados. Ex⁴:

³Memória não volátil, que se mantém gravada mesmo após o equipamento ser desligado.

⁴Os códigos que desenhavam as letras foram omitidos para ajudar no entendimento da programação.

```
//desenha a letra A
(...) [salvar em >>>] [nova peça A]
//copia a peça A
[A] [salvar em >>>] [nova peça A]
//desenha a letra N
(...) [salvar em] [nova peça N]
[desenha][A][N][A][fim]
```

No código acima o usuário usou uma série de peças para desenhar a letra A. Ele salvou o conjunto dessas peças em uma nova peça, que ele chamou de 'A'. Em seguida ele copiou uma nova peça 'A', agora tendo em mãos duas peças 'A'. Após criar uma nova peça da letra 'N', ele pode então pedir para o robô escrever a palavra ANA com sua caneta embutida. Diferentes formas de se desenhar a letra 'A' podem ser trocadas e experimentadas pelos usuários, bastando emprestar suas peças com sua "caligrafia" única. O adesivo RFID pode então ser marcado pelo usuário com a função da nova peça criada (Fig. 6).



Figure 6: Peça que desenha uma casa, personalizada pela criança (Fonte do autor).

A linguagem de programação tangível Code Dominó apresenta-se então como uma adaptação da linguagem LOGO ao formato tangível. Suas características e capacidades permitem que usuários façam desde comandos simples de movimentos, até programas complexos com chamadas de funções e recursividade. Sua capacidade de trabalhar em blocos separados permite que mesmo com um jogo de poucas peças (52 peças ao total) se construa algoritmos com alta complexidade e funcionalidade.

Robô Decabot

O robô decabot (Fig. 7) é feito de MDF ou acrílico cortado a laser, com eletrônica baseada em Arduino[2], buscando ser o mais acessível possível para professores e alunos. Seu chassi forma um cubo perfeito de 10cm de lado, e seu corpo está preparado para a inclusão de diversos sensores, atuadores e motores, de acordo com a disponibilidade de peças do usuário.



Figure 7: Robô decabot (fonte: do autor).

Na parte inferior do robô, entre as rodas, existe um leitor e gravador de chips RFID. É por ele que o robô lê o programa feito pelo usuário, caminhando por cima das peças. O usuário (1) monta o código, (2) posiciona o robô para gravação, (3) executa a gravação, e depois (4) usuário clica no botão correspondente ao bloco de memória onde foi gravado, e o robô o executa (figura 8). No centro do robô é posicionada uma caneta, para que o robô possa desenhar enquanto caminha por superfícies lisas, permitindo que o usuário possa verificar a resposta ao seu programa.

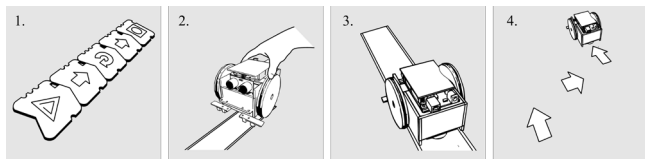


Figure 8: Gravando um código e executando no robô (fonte: do autor).

5 AVALIAÇÃO

Crianças de 7 a 9 anos

Em sessões de experimentação com 10 crianças entre 7 e 9 anos, os usuários puderam interagir com as peças inicialmente, e em seguida também com o robô (Fig. 9). O instrutor condutor do experimento fazia pequenos desafios para que o usuário respondesse através do dominó (inicialmente somente com os comandos de movimento). Quando o usuário já demonstrava perícia com a sintaxe básica do dominó, lhe era apresentado o robô e sua forma de leitura e execução. O usuário então tinha a possibilidade de experimentar as peças e ver as respostas. O experimento também era acompanhado de um avaliador para fazer registro da interação, e de uma entrevista pós-teste.



Figure 9: Usuário experimentando o robô e as peças do Code Dominó (fonte: do autor).

No teste os instrutores apresentavam os comandos de movimento com o objetivo das crianças desenharem um quadrado. Em seguida com a introdução de parâmetros, que mudassem o tamanho do quadrado. A atividade seguia com a introdução do comando repetir, e o desafio de fazer com que o usuário definisse um código para que o robô desenhasse um círculo com as peças disponíveis.

As experimentações ocorreram individualmente (4 usuários) e coletivamente (com 6 usuários juntos em sala, e dois robôs).

Professores

A avaliação com os professores se deu através de entrevistas sobre metodologias STEM adotadas, suas dificuldades de adoção e a apresentação dos objetos interativos desenvolvidos (dominó e robô). Foram entrevistados dois professores do ensino público do Brasil.

Em entrevista, os professores relatam que na realidade brasileira a adoção de tecnologias na sala de aula muitas vezes é um trabalho árduo e pontual de um ou poucos professores por escola. A aquisição, manutenção e replicabilidade da tecnologia muitas vezes barra na questão custo, por isso soluções virtuais, via software ou sites, costumam ser mais adotadas.

Além das dificuldades de acesso, a falta de apoio em materiais didáticos foi levantada em entrevista. Tal reclamação já foi evidenciada por Kalelioglu et al.[18] onde em revisão sistemática de 125 artigos sobre ensino de pensamento computacional, somente 7 deles traziam uma base teórica pedagógica: 5 deles abordavam construcionismo de Papert, e apenas dois o construtivismo de Vygotsky.

6 DISCUSSÃO

Sem tela, sem distração

Devido a natureza da interação ser totalmente tangível, sem o uso de dispositivos com tela como tablets e celulares, a atenção dos alunos foi constante ao experimento. Nas experimentações individuais os usuários mostravam-se comprometidos nos desafios dados pelo avaliador. Na experimentação coletiva houve uma tendência maior à tentativa e erro (e uma agitação natural de muitas crianças em conjunto).

Tangibilidade, um convite a colaboração

A facilidade de montagem das peças no chão ou na mesa, e a simplicidade na manipulação dos comandos mostraram-se positivos para a colaboração entre crianças. Nenhum dos usuários se mostrou acanhado ou relutante em experimentar a montagem dos códigos via peças. Nas experimentações coletivas a correção e experimentação em grupo do código seguiram sem brigas ou problemas.

A tangibilidade então mostra-se um fator importante contra o *sequestro* da operação da máquina por um dos alunos num grupo (geralmente o que já teve experiências no assunto ou no equipamento). Alunos tímidos ou meninas subjugadas por meninos podem ceder a operação do computador em uma aula, pelo temor de errar. Soluções convidativas e colaborativas podem ser uma solução para o problema[28].

Mais programação, menos puzzles

A exploração livre dos códigos e do robô permitiu que os usuários pudessem prever usos para além dos apresentados pelos avaliadores. Por exemplo, ao notarem que um sensor pode disparar uma ação programada, logo despontam ideias com a de um robô cão de guarda, onde um sensor de presença dispara um código de rastreamento que segue em perseguição a um intruso. Ideias sobre competições de dança, batalhas ou corridas em labirintos também despontam com as possibilidades da programação dos robôs para outras tarefas. Este

resultado condiz com as premissas de Papert[5, 22] sobre a necessidade da programação ser vista pela criança como uma ferramenta que expande suas capacidades, e não simplesmente um entretenimento.

7 TRABALHOS FUTUROS

A entrevista com os professores mostrou que a proposta está incompleta se não contemplar planos de aula com metodologias pedagógicas definidas para apoio ao professor em sua condução na sala de aula. Novas experimentações, agora com o professor no papel de avaliador, serão conduzidas para uma completa análise dos objetos da pesquisa.

Ficou latente a necessidade de avaliar o impacto da afetividade dos alunos no ensino através de objetos tangíveis. Em experimentos com usuários crianças, existe um encantamento extra quando usamos robôs com rosto, olhos e respostas. Esta resposta afetiva e empática pode ser amplificada com a personalização dos robôs através de arte (onde a educação STEM ganha mais a letra A de arte: STEAM[5]). Os objetos interativos desta pesquisa permitem esta abordagem, ainda a ser avaliada.

Outra abordagem é a de transformar robôs em uma mídia para ensino de outras matérias, questão já levantada por Grover e Pea[13]. A abordagem de usar robôs como aprendizes dos alunos, aproveitando-se do chamado efeito Protégé[9], pode beneficiar outras áreas, e não somente a programação. A plataforma Code Dominó mostra-se simples o suficiente, porém poderosa, para experimentar estes desafios.

A equipe também acredita no impacto de soluções como o Code Dominó nas questões de gênero na educação tecnológica[4], e sua influência deve ser estudada.

8 CONCLUSÃO

Para ampliar o uso e possibilidades das interfaces tangíveis na educação tecnológica, é necessário que estas ultrapassem a categoria de entretenimento e solução de desafios, para se tornarem reais ferramentas de expansão das capacidades dos alunos. A linguagem de programação Code Dominó, ainda que baseada em peças físicas, é versátil para a construção de complexos algoritmos, empoderando alunos e professores. Feita utilizando-se de materiais baratos e acessíveis como MDF e etiquetas adesivas RFID, a linguagem é expansível pelos próprios usuários que dividem seus programas em blocos de execução. O robô apresentado também foi desenvolvido com uma perspectiva pragmática de ter um custo baixo para professores e alunos, aproveitando peças comuns e plataformas livres, como a Arduino.

ACKNOWLEDGMENTS

A equipe agradece a Funcap pelo financiamento do projeto através do edital Inovafit II 2017, e a Diretoria de Pesquisa e Inovação da Unifor pelo apoio na execução do projeto.

REFERENCES

- [1] Pedro Ferreira Alves. 2014. Vygotsky and Piaget: Scientific concepts. *Psychology in Russia: State of the Art* (2014). <https://doi.org/10.11621/pir.2014.0303>
- [2] Arduino. 2019. Arduino - Home.
- [3] Massimo Banzi. 2008. *Getting Started with Arduino (Make: Projects)* (ill ed.). Make: Projects, Vol. 11. Make Books. 118 pages. <http://www.amazon.com/exec/obidos/redirect?tag=citeulike07-20&path=ASIN/0596155514http://www.amazon.com/Getting-Started-Arduino-Massimo-Banzi/dp/1449309879>
- [4] David N. Beede, Tiffany A. Julian, David Langdon, George McKittrick, Beethika Khan, and Mark E. Doms. 2012. Women in STEM: A Gender Gap to Innovation. *SSRN Electronic Journal* (2012). <https://doi.org/10.2139/ssrn.1964782>
- [5] Marina Umaschi Bers. 2018. *Coding as a Playground* (1 ed.). Routledge, New York, NY, United States. <https://doi.org/10.4324/9781315398945>
- [6] Khalid Abdullah Bingimlas. 2009. Barriers to the successful integration of ICT in teaching and learning environments: A review of the literature. <https://doi.org/10.12973/ejmste/75275>
- [7] Marleny Luque Carbajal and M Cecília C Baranauskas. 2015. TaPrEC: Desenvolvendo um ambiente de programação tangível de baixo custo para crianças. *Anais do XX Congresso Internacional de Informática Educativa - TISE* (2015).
- [8] Daniel Almeida Chagas and Elizabeth Sucupira Furtado. 2019. Computational Thinking in Basic Education in a Developing Country Perspective. In *Research & Innovation Forum 2019* (1 ed.), Anna Visvizi and Miltiadis Lytras (Eds.). Springer International Publishing, Rome, 340. <https://doi.org/10.1007/978-3-030-30809-4>
- [9] Catherine C. Chase, Doris B. Chin, Marily A. Oppezzo, and Daniel L. Schwartz. 2009. Teachable Agents and the Protégé Effect: Increasing the Effort Towards Learning. *Journal of Science Education and Technology* 18, 4 (01 Aug 2009), 334–352. <https://doi.org/10.1007/s10956-009-9180-4>
- [10] Crístia Rosineiri Gonçalves Lopes Correa. 2017. A relação entre desenvolvimento humano e aprendizagem: perspectivas teóricas. *Psicologia Escolar e Educacional* 21 (12 2017), 379 – 386. http://www.scielo.br/scielo.php?script=sci_arttext&pid=S1413-85572017000300379&nrm=iso
- [11] Peter Denning and Matti Tedre. 2019. *Computational Thinking*. Cambridge, MA, USA. 264 pages.
- [12] Gerald Futschek and Julia Moschitz. 2011. Learning algorithmic thinking with tangible objects eases transition to computer programming. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. https://doi.org/10.1007/978-3-642-24722-4_14
- [13] Shuchi Grover and Roy Pea. 2013. Computational Thinking in K–12. *Educational Researcher* (2013). <https://doi.org/10.3102/0013189x12463051>
- [14] Michael S. Horn, Erin Treacy Solovey, R. Jordan Crouser, and Robert J.K. Jacob. 2009. Comparing the use of tangible and graphical programming languages for informal science education. <https://doi.org/10.1145/1518701.1518851>
- [15] Anning Hu and Nicholas Vargas. 2015. Economic consequences of horizontal stratification in postsecondary education: evidence from urban China. *Higher Education* 70, 3 (sep 2015), 337–358. <https://doi.org/10.1007/s10734-014-9833-y>
- [16] Hiroshi Ishii and Brygg Ullmer. 1997. Tangible Bits: Towards Seamless Interfaces Between People, Bits and Atoms. In *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems (CHI '97)*. ACM, New York, NY, USA, 234–241. <https://doi.org/10.1145/258549.258715>
- [17] Robert JK Jacob, Audrey Girouard, Leanne M. Hirshfield, Michael S. Horn, Orit Shaer, Erin Treacy Solovey, and Jamie Zigelbaum. 2008. Reality-Based Interaction : A Framework for Post-WIMP Interfaces. In *Proceedings of the SIGCHI conference on Human factors in computing systems*.
- [18] Filiz Kalelioglu, Yasemin Gulbahar, and Volkan Kukul. 2016. A Framework for Computational Thinking Based on a Systematic Research Review. *Baltic Journal of Modern Computing* 4 (05 2016), 583–596.
- [19] Jane Krauss and Kiki Prottzman. 2017. *Computational Thinking and Coding for Every Student*. Corwin, Corwin. 184 pages.
- [20] Eva Marinus, Zoe Powell, Rosalind Thornton, Genevieve McArthur, and Stephen Crain. 2018. Unravelling the Cognition of Coding in 3-to-6-year Olds. <https://doi.org/10.1145/3230977.3230984>
- [21] Timothy S. McNerney. 2004. From turtles to Tangible Programming Bricks: Explorations in physical language design. *Personal and Ubiquitous Computing* (2004). <https://doi.org/10.1007/s00779-004-0295-6>
- [22] Seymour Papert. 1993. *The Children’s Machine: Rethinking School in the Age of the Computer*. Basic Books, Inc., New York, NY, USA.
- [23] Rolf Ploetzner, Pierre Dillenbourg, Michael Preier, and David Traum. 1999. Learning by Explaining to Oneself and to Others.
- [24] Yvone Rogers, Helen Sharp, and Jennifer Preece. 2013. O que é Design de Interação? In *Design de Interação* (3ª ed. ed.). Bookman, Porto Alegre, Brazil, Chapter 1 - O que, 25–29.
- [25] Rod D. Roscoe and Michelene T. H. Chi. 2007. Understanding Tutor Learning: Knowledge-Building and Knowledge-Telling in Peer Tutors’ Explanations and Questions. *Review of Educational Research* 77, 4 (2007), 534–574. <https://doi.org/10.3102/0034654307309920> arXiv:<https://doi.org/10.3102/0034654307309920>
- [26] Orit Shaer and Eva Hornecker. 2010. Tangible User Interfaces: Past, Present, and Future Directions. *Found. Trends Hum.-Comput. Interact.* 3, 1–2 (Jan. 2010), 1–137. <https://doi.org/10.1561/1100000026>
- [27] Tangible Play Inc. 2013. Osmo. (2013). www.playosmo.com
- [28] Javier Mauricio Vera, Paola Rodriguez, and Ivette Muñoz. 2017. Tangible User Interfaces: concepts and practice. *Nuevas Ideas en Informática Educativa* 13 (2017). <http://www.tise.cl/volumen13/III.html>
- [29] White House. 2015. A Nation of Makers. <https://www.whitehouse.gov/nation-of-makers>
- [30] Jeannette M. Wing. 2006. COMPUTATIONAL THINKING BENEFITS SOCIETY. *Commun. ACM* (2006). <https://doi.org/10.1145/1118178.1118215>
- [31] Jeannette M. Wing. 2011. Computational thinking - What and why? *The Link Magazine* (2011).

A ONLINE RESOURCES

Os arquivos-fonte e materiais para construção dos objetos interativos estão disponíveis em <https://github.com/codedomino/> distribuídos através da licença Creative Commons.