

Jogos com LEDs Endereçáveis: Uma Metodologia Usando LLMs Integrando JavaScript, Arduino, Wifi e NotebookLM

Ricardo Ferreira¹ , Talles Costa¹ , Isabela de Castro Freitas¹ 
Pedro Coura Pereira¹ , Alice Faria¹ , Racyus D. G. Pacífico² 

¹Departamento de Informática, Universidade Federal de Viçosa, Brasil
email: {ricardo,talles.costa,isabela.castro.freitas,pedro.coura,alice.s.faria}@ufv.br

² Universidade Federal de Ouro Preto (UFOP) – João Monlevade, MG – Brasil
racyus.pacifico@ufop.edu.br

Abstract. *Digital games have established themselves as pedagogical tools for active learning. However, their practical implementation still presents challenges. This paper presents a methodology for developing Arcade games using Large Language Models (LLMs). The games are initially generated in JavaScript with the assistance of LLMs and later adapted for microcontrollers with LED matrices. Interactivity is expanded through client-server architectures or the MQTT protocol, enabling control via mobile devices. To promote reproducibility and educational use, both the source code and the prompts used are available in a public repository. Additionally, supplementary project documentation was created with the support of NotebookLM.*

Resumo. *Jogos digitais consolidaram-se como ferramentas pedagógicas de aprendizagem ativa, porém, ainda apresentam desafios de implementação. Este trabalho apresenta uma metodologia para o desenvolvimento de jogos de Arcade utilizando LLMs. Os jogos são inicialmente gerados em JavaScript com o auxílio de LLMs e, posteriormente, adaptados para microcontroladores com matrizes de LEDs endereçáveis WS2812b. A interação é ampliada por meio de comunicação cliente-servidor ou MQTT com o controle via dispositivos móveis. Visando favorecer a reprodutibilidade e o uso educacional, os códigos e os prompts utilizados estão disponíveis em um repositório público. Adicionalmente, uma documentação suplementar foi elaborada com o NotebookLM.*

1. Introdução

Jogos digitais consolidaram-se como ferramentas pedagógicas por promoverem aprendizagem ativa e desenvolvimento de competências cognitivas, favorecem a transferência de conhecimento para contextos reais, alinhados aos princípios cognitivos que potencializam a aplicação do conteúdo em diferentes áreas, promovendo o raciocínio lógico [Yulia et al. 2025]. Apesar de seus benefícios, a implementação de jogos digitais na educação enfrenta desafios, como limitações de infraestrutura e barreiras no aprendizado inicial para a integração entre tecnologias. Os resultados positivos dependem do alinhamento entre tecnologias utilizadas e objetivos educacionais para que o aprendizado no desenvolvimento das atividades produza efeitos duradouros [Sambo et al. 2025].

Este trabalho apresenta uma abordagem integrada para o desenvolvimento de jogos de Arcade utilizando JavaScript, C/C++, painéis de LEDs endereçáveis e dispositivos móveis, apoiada por serviços Web, conectividade Wi-Fi, protocolos MQTT e modelos de linguagem de grande porte (LLMs). A metodologia adotada baseia-se no uso

de códigos existentes ou gerados automaticamente por LLMs, iniciando pela prototipagem dos jogos, o que facilita a validação rápida da lógica e da interação. Em seguida, esses códigos são convertidos para execução em microcontroladores, possibilitando a implementação em matrizes de LEDs endereçáveis com desempenho em tempo real. Todo o trabalho está disponível na plataforma Google Colab [Ferreira et al. 2024a] que possibilita interatividade [Ferreira et al. 2024b] e suporte para documentação de prompts e demonstrações [Freitas et al. 2025, Araújo et al. 2026].

Além disso, o trabalho considera duas plataformas-alvo distintas. A primeira utiliza uma fita de LEDs endereçáveis com o jogo *OpenLED Race*, uma corrida de LEDs amplamente difundida e já empregada em diversos projetos educacionais e feiras de ciência em diferentes países. A segunda plataforma baseia-se em matrizes de LEDs de baixa resolução, tipicamente 12×12 ou 16×16 , para as quais foram gerados, com auxílio de LLMs, diversos jogos clássicos de Arcade, como *Tetris*, *Pong*, entre outros. Devido à ampla disponibilidade de projetos envolvendo Arduino e saídas matriciais de baixa resolução, as LLMs demonstram facilidade para esse contexto. As matrizes de LEDs tornam os jogos visualmente atrativos, enquanto a popularidade e a simplicidade desses jogos clássicos dispensam apresentações formais e para motivar os estudantes a explorar, compreender a lógica e funcionamento, além de poder modificar o código.

Este trabalho está organizado: A Seção 2 introduz os LEDs endereçáveis e os protocolos de comunicação para contextualizar tecnologias alvo. A Seção 3 apresenta a metodologia de desenvolvimento com uso das LLMs. A Seção 4 apresenta os jogos desenvolvidos. A Seção 5 apresenta uma avaliação quantitativa para os códigos gerados e a documentação. As Seções 6 e 7 apresentam os trabalhos relacionados e as conclusões.

2. Fundamentos

2.1. Comunicação

Para controlar jogos de Arcade, como *Tetris* e *Pong*, utilizando o celular sem a necessidade de desenvolver um aplicativo dedicado, pode-se recorrer ao navegador Web como interface de controle. Nesse contexto, existem diferentes possibilidades de comunicação Wi-Fi entre o celular e o microcontrolador ESP. A abordagem mais direta consiste no modelo cliente-servidor em rede local ou ponto de acesso, no qual o ESP atua como servidor e gerencia as requisições dos clientes, representados pelos jogadores acessando a interface via navegador. Essa comunicação pode ser realizada por meio de HTTP ou *sockets*, menos eficiente para jogos que exigem respostas rápidas e interação contínua.

Como alternativa, o protocolo MQTT introduz o paradigma publicador-assinante, permitindo comunicação desacoplada por meio de um *broker* e facilitando a integração com diferentes plataformas. Embora seja amplamente utilizado em aplicações IoT, o uso de MQTT pode introduzir latência adicional, o que tende a prejudicar a experiência em jogos de Arcade que dependem de comandos em tempo real. Para jogos que demandam baixa latência e troca contínua de dados, como *Pong*, o uso de *WebSockets* se mostra mais adequado, pois permite comunicação bidirecional persistente sobre uma única conexão TCP, reduzindo a sobrecarga associada a múltiplas requisições HTTP. A biblioteca `Socket.io` complementa essa abordagem ao abstrair a complexidade dos *WebSockets*.

Por fim, quando a interação se limita ao envio contínuo de atualizações do servidor para o cliente, eventos *Server-Sent* podem ser utilizados como uma solução mais

simples, baseada em conexões HTTP unidirecionais. No entanto, essa abordagem é menos indicada para jogos interativos, pois não oferece suporte nativo à troca bidirecional de comandos em tempo real.

2.2. Leds endereçáveis

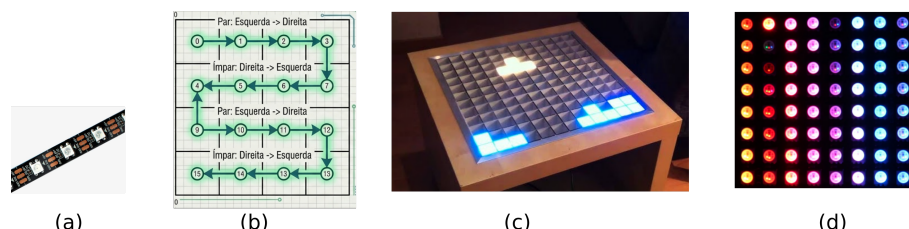


Figura 1. LEDs endereçáveis: (a) Fita; (b) Zigue-Zague; (c) Mesa; (d) Matriz.

Os LEDs endereçáveis simplificaram as conexões ao incorporar um protocolo serial de fio único (Figura 1(a)), permitindo acender e apagar individualmente os LEDs em um vetor. Apesar das limitações inerentes à comunicação serial, cada LED opera com um tempo de transmissão de aproximadamente $2 \mu s$ por bit, sendo necessários 24 bits por LED (8 bits por cor). O tempo total de atualização depende diretamente do comprimento do vetor: por exemplo, um conjunto com 300 LEDs apresenta um tempo de atualização de cerca de 15 ms, o que corresponde a 66 quadros por segundo. Além disso, o controle das cores e a conexão simples com microcontroladores tornam esses dispositivos adequados para jogos interativos. Nesse contexto, o uso de fitas e matrizes de LEDs endereçáveis viabiliza o desenvolvimento de plataformas educacionais e recreativas que aliam baixo custo, apelo visual e flexibilidade. As matrizes de LEDs endereçáveis segue o padrão de zigue-zague como ilustrado na Figura 1(b) e podem ser montadas com uma fita. Uma fita com 30 LEDs por metro pode decorar uma mesa como ilustrado na Figura 1(c). Existem matriz já prontas, também seguem a lógica zigue-zaque como ilustrado na Figura 1(d).

3. Metodologia

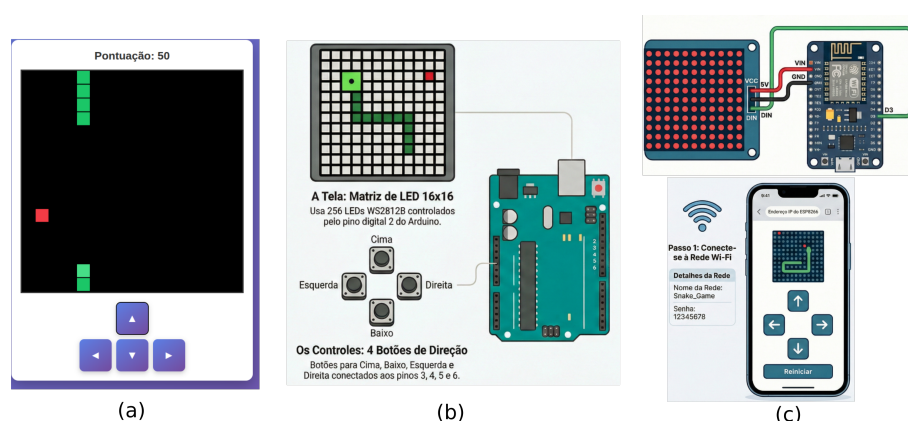


Figura 2. Versões do jogo Snake: (a) Javascript no Google Colab; (b) Arduíno com chaves; (c) Controle com celular e painel conectado no ESP.

A metodologia fundamenta-se em um fluxo de desenvolvimento híbrido que integra a agilidade da Web para validação inicial do jogo para depois fazer a portabilidade

para um microcontrolador. Inicialmente, realiza-se a geração em JavaScript dentro do ambiente Google Colab, utilizando apenas as LLMs. Esta etapa permite a validação imediata da mecânica do jogo e da interface sem a necessidade de hardware físico, reduzindo o ciclo de depuração lógica. Para jogos conhecidos, a geração é simples e direta devido a grande disponibilidade de implementações, mesmo sendo em outras plataformas e linguagens. A Figura 2(a) mostra a interface para o jogo *snake* em JavaScript. O JavaScript é convertido para microcontroladores, como Arduino e NodeMCU/ESP32. As LLMs adaptam as saídas visuais para as matrizes de LEDs, validada no simulador Wokwi online, onde o comportamento e a temporização dos LEDs são testados em um ambiente virtual, ainda sem a necessidade do hardware. A Figura 2(b) mostra o Arduino em ambiente simulado com 4 botões para movimentação.

A etapa final testa no hardware físico com integração da conectividade sem fio. Utiliza-se as implementações cliente-servidor e o protocolo MQTT para a comunicação com microcontrolador, permitindo que múltiplos participantes interajam simultaneamente através de *dashboards* ou páginas Web hospedadas no próprio ESP8266 ou ESP32. A Figura 2(c) mostra a versão final controlada via navegador usando um celular. Todo o processo de desenvolvimento, desde os *prompts* iniciais até as versões finais dos códigos, é documentado com auxílio do NotebookLM para gerar uma base de conhecimento que facilite a reprodutibilidade e o uso pedagógico do projeto.

4. Jogos

4.1. OpenLed Race - Código Original sem LLMs

O jogo *OpenLED Race* surgiu como uma atividade educacional desenvolvida para o evento Arduino Day Sevilla 2019. Criado antes da popularização das LLM, o projeto evoluiu como uma iniciativa independente de código aberto com ampla difusão na comunidade maker global. A lógica do jogo é propositalmente simples: cada carrinho é representado por uma sequência contígua de LEDs que se desloca ao longo da fita de LEDs. A interação do usuário ocorre por meio de um único botão, no qual acionamentos mais rápidos e frequentes aumentam a velocidade do carrinho, enquanto a ausência de interação provoca sua desaceleração gradual, simulando a aceleração e a inércia.

A Figura 3(a) mostra duas montagens. A primeira destaca os controles do jogo original. A segunda mostra o potencial marker para uso de impressora 3D para criar uma pista com loops e curvas. A figura 3(b) ilustra duas versões com MQTT. Uma com 5 a 15 jogadores e o Google Colab, onde cada um escolhe sua cor, que já foi validada em vários eventos. O segundo exemplo faz uso de um aplicativo MQTT com atrativos visuais.

Neste trabalho, o código-fonte original do *OpenLED Race* foi utilizado como base e, inicialmente, estendido sem o uso de LLMs. As extensões incluem a implementação de duas pistas paralelas com possibilidade de ultrapassagem, o acesso e controle remoto por meio do protocolo MQTT e a visualização em tempo real do estado do jogo utilizando painéis em aplicativos do tipo *MQTT dashboard*. Adicionalmente, a integração com o Google Colab via MQTT com participação simultânea de múltiplos jogadores. Nessa etapa, o NotebookLM foi empregado para avaliar sua capacidade de compreender o código e apresentar explicações sobre sua estrutura e lógica. Avaliamos também outros códigos com duas pistas e efeito de fogo, tendo resultados satisfatórios com ilustrações didáticas (Fig. 3(b) e Fig. 4(c)).

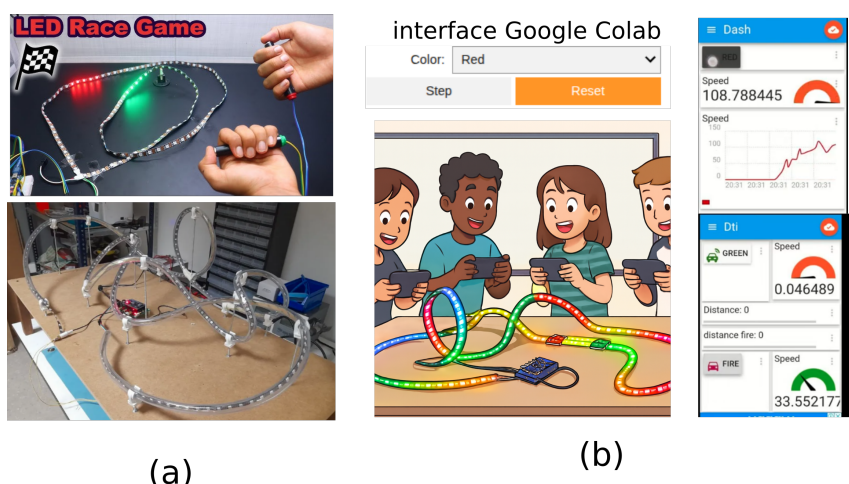


Figura 3. (a) Jogo original *OpenLed Race* com controle usando botões e pistas em loops; (b) Controle por celular e painel em aplicativo MQTT.

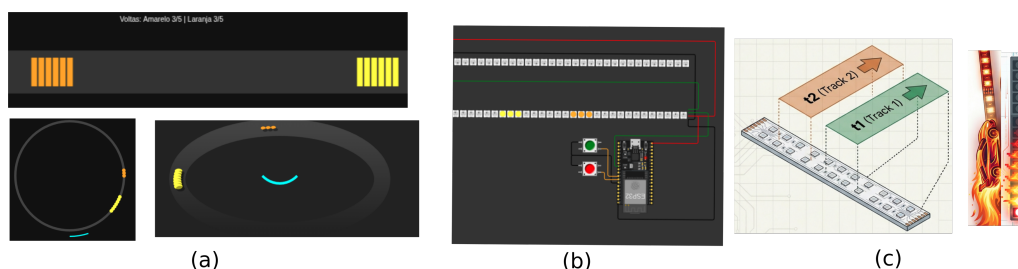


Figura 4. (a) Três versões em JavaScript; (b) Mapeamento no ESP/Wokwi; (c) Extensão NotebookLM com duas pistas e efeito de fogo.

4.2. OpenLedRace gerado pela LLM

Seguimos também o outro caminho. Criar o jogo do zero, que não é tão popular como os de Arcade. Utilizou-se a LLM Copilot, com os comandos apresentados a seguir, para gerar o jogo de corrida sem acesso prévio ao código-fonte original. Inicialmente, o modelo gerou uma versão 1D, conforme ilustrado na Figura 4(a) bem limitada mas funcional; na sequência, foram solicitadas uma pista circular e uma versão com perspectiva. Posteriormente, foi solicitada a conversão do código da animação simples para um microcontrolador com a fita de LED, obtendo-se sucesso com apenas um *prompt* direto, como demonstrado na Figura 4(b). O código gerado é mais simples e funcional que o código original do projeto *OpenLed Race* que não usou LLMs.

Quadro 1: Exemplo de *prompt* para corrida de LED

Shot 1: Fazer o jogo em JavaScript para rodar no Colab com interface via teclado, usando o botão A para acelerar o carro amarelo e L para o carro laranja. Os carros começam com três pixels e, a cada volta, o tamanho aumenta em 1 pixel. A corrida tem 5 voltas. Quanto mais rápido apertamos A ou L, o carro de LED se desloca mais rápido; ao parar de pressionar, o carro desacelera até parar.

Shot 2: Melhorar para uma pista circular com um loop.

Shot 3: Aplicar efeitos "3D" com profundidade, simulando uma pista curva de autorama com loop.

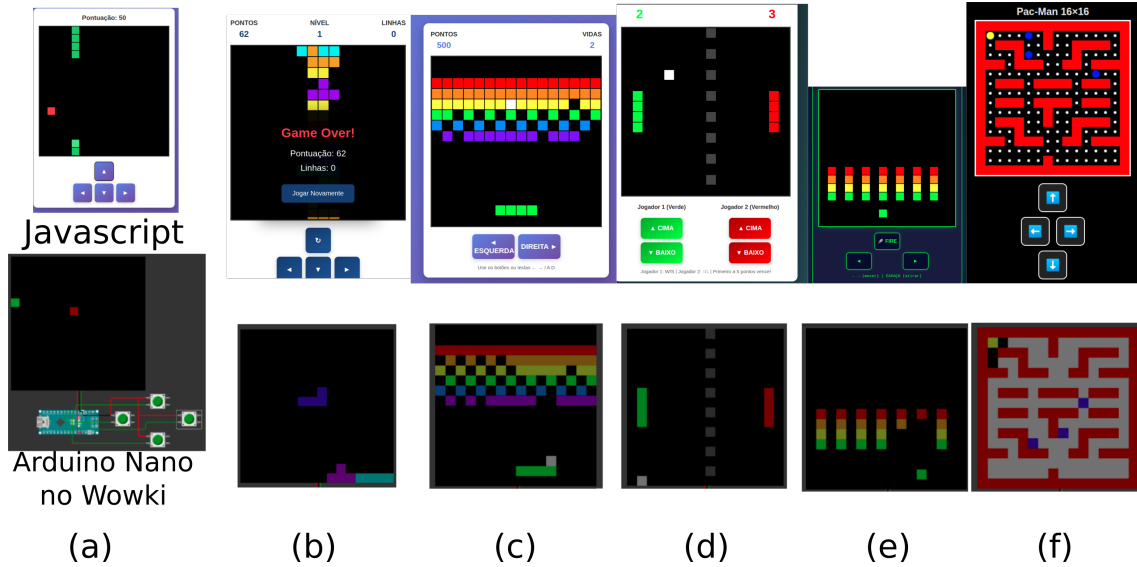


Figura 5. Jogos: (a) Snake; (b) Tetris; (c) Breakout; (d) Pong; (e) Space Invaders; (f) Pacman simples.

4.3. Arcade

Na parte superior da Figura 5 são apresentadas as telas correspondentes aos jogos gerados pela LLM Claude Opus 4.7 em JavaScript e executados no ambiente Google Colab. Por se tratarem de aplicações em JavaScript, não é necessária a conexão com os servidores da Google e os jogos são executados localmente no navegador, permitindo a validação da lógica e da dinâmica visual. Essa etapa facilita a prototipagem e a verificação inicial do comportamento dos jogos, eventual correção ou modificação com auxílio ou não das LLMS, antes de sua conversão para microcontroladores. O primeiro jogo gerado foi o *Snake* com um *prompt* com apenas 26 palavras. Para os demais jogos foi usado um *prompt* ainda mais curto, "agora modifique para o jogo X e gere o código em javascript".

A parte inferior da Figura 5 mostra as telas do simulador de Arduino Wokwi, onde os jogos traduzidos pela LLM Claude de JavaScript para C/C++ com o requisito de usar a biblioteca *fastled* e botões em cruz para o controle do jogo. Os códigos foram executados em um Arduino no Wokwi, onde todo o código e sua lógica foram validados em condições próximas às reais, incluindo temporização, controle de entradas e atualização dos LEDs. Essa abordagem em duas etapas garante consistência entre as versões em JavaScript e embarcada, além de reduzir erros durante a transição para o hardware final.

Os quatro jogos *Snake*, *Tetris*, *Breakout* e *Pong* são amplamente populares em versões pixeladas, com versões em matrizes de LEDs de baixa resolução. Já os jogos *Space Invaders* e *Pac-Man* exigiram a solicitação de implementações minimalistas, adequadas às limitações de resolução. No caso do *Pac-Man*, o processo foi invertido: inicialmente foi gerada a versão para Arduino, a partir da adaptação do código de *Space Invaders*, e, em seguida, solicitou-se a versão em JavaScript para avaliar a tradução no sentido inverso. Nesse experimento, utilizou-se o ChatGPT 5.2.

4.4. Interface em Dispositivos Móveis

Uma vez validado o código no Wokwi, fizemos a conversão para 5 variações de controle com Wifi: cliente-servidor como ponto de acesso, cliente-servidor com rede local, *Websocket*, *Server-event* e MQTT. A Figura 6 mostra a matriz de LEDs com um gabarito impresso em uma folha A4 para suavizar o brilho e pixelar o painel. Ao lado vemos a interface Web no celular. Pode-se incentivar oficinas maker para se criar caixas, pistas de corridas e outros efeitos para a montagem final.

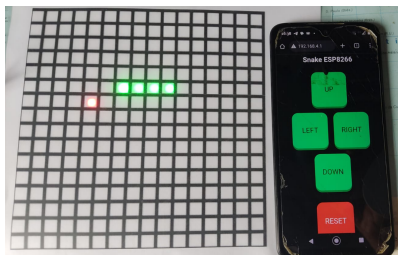


Figura 6. NodeMCU com cliente-servidor e interface no celular para *Snake*.

5. Resultados experimentais

A Tabela 1 mostra alguns resultados quantitativos. A coluna *Prompt* mostra que todos os códigos só precisaram de uma requisição. A coluna *Cód. exemplo* mostra que apenas três códigos foram gerados adicionando um exemplo de código ao *prompt*. A coluna *Tam. Total* mostra o número de palavras de cada *prompt*, cuja a maioria é curto, no estilo "agora gere a versão Arduino" ou "gere o jogo Tetris". Apenas os primeiros *prompts* são mais longos, depois a LLM faz uso do contexto e *prompts* curtos são suficientes. A coluna *Linhas (LOC)* mostra o tamanho dos códigos incluindo comentários. Em geral, a versão JavaScript é maior por conter a parte do desenho em HTML. Pode-se observar que os códigos são pequenos entre 100 a 515 linhas no máximo.

Finalmente, a coluna *tempo* mostra o tempo para gerar e realizar o teste piloto. Em geral, todos os códigos foram validados rapidamente entre 5 a 10 minutos nas versões JavaScript e Wokwi. Após digitar o *prompt*, em menos de 1 a 2 minutos, já se tem o resultado, que copiado e executado. Três minutos adicionais para testar. Todos executaram na primeira tentativa. Apenas para o teste no NodeMCU que foi necessário um tempo maior, pois é preciso compilar na IDE do Arduino, carregar o código no NodeMCU, depois conectar com o celular e testar o jogo.

Qual é o número total de LEDs na matriz utilizada para o jogo <i>Breakout</i> ? Marque a alternativa correta.	Qual biblioteca do Arduino é responsável pelo controle dos LEDs WS2812B? Marque a alternativa correta.	Como o código trata a conversão de (x, y) para zig-zag? Marque a alternativa correta.
A. 256	A. LedControl	A. Fórmula linear: $y \times largura + x$.
B. 128	B. Adafruit_NeoPixel	B. Tabela de consulta.
C. 512	C. FastLED	C. Inverte x p/ y ímpar.

Figura 7. Exemplo de três questões.

Para documentar, se fez uso do NotebookLM, que a partir do código-fonte, gerou automaticamente mapas mentais, relatórios técnicos simplificados e tabelas detalhadas

Tabela 1. Métricas de desenvolvimento e implementação dos jogos e protocolos.

Projeto / Projeto / Plataforma	Dados dos Prompts			Código Final Linhas (LOC)	Tempo em minutos
Prompt	Cód. ex.	Tam. total			
<i>Snake JS</i>	1	-	26	279	10
<i>Snake Wokwi</i>	1	100	15	249	8
<i>Tetris JS</i>	1	-	10	465	5
<i>Tetris Wokwi</i>	1	-	5	403	5
<i>Breakout JS</i>	1	-	10	451	5
<i>Breakout Wokwi</i>	1	-	5	355	5
<i>Pong JS</i>	1	-	5	472	5
<i>Pong Wokwi</i>	1	-	4	320	5
<i>Space Invaders JS</i>	1	-	5	515	5
<i>Space Invaders Wokwi</i>	1	-	4	436	5
<i>Pacman Wokwi</i>	1	-	35	158	5
<i>Pacman JS</i>	1	158	12	144	5
<i>OpenLedRace JS (1D)</i>	1	-	67	97	5
<i>OpenLedRace JS (Circ)</i>	1	-	7	123	5
<i>OpenLedRace JS (3D)</i>	1	-	19	136	5
<i>OpenLedRace Wokwi</i>	1	141	22	71	10
<i>Snake (Ponto Acesso)</i>	1	-	38	194	20
<i>Snake (Rede Local)</i>	1	-	6	191	20
<i>Snake (Webserver)</i>	1	-	6	185	20
<i>Snake (Server-Event)</i>	1	-	8	198	20

contendo as variáveis e suas respectivas funções. Adicionalmente, a criação de baterias de testes e *flashcards* personalizados pode motivar estudantes a se questionarem sobre a lógica do jogo e seu mapeamento em código como ilustrado na Figura 7. Outro fator é a capacidade de síntese de conteúdo em formato de vídeo explicativo de forma multimodal. Essa estratégia visa combater o uso passivo de modelos generativos, onde o estudante poderia apenas solicitar o código pronto sem compreender seu funcionamento. A Figura 8 mostra parcialmente a documentação gerada para o jogo *Breakout*.

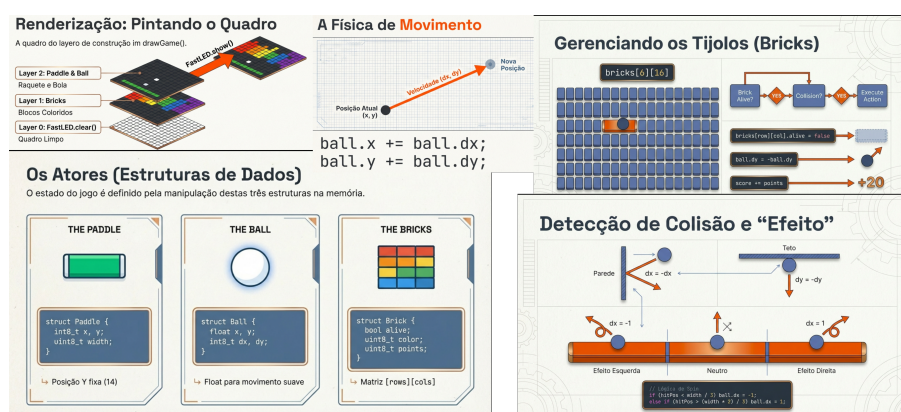


Figura 8. Documentação para o *Breakout* gerada pelo NotebookLM

6. Trabalhos relacionados

O uso de LLM para geração de códigos está amplamente difundido [Li et al. 2024], onde alguns trabalhos avaliam a capacidade específica para microcontrolado-

res [Petrović et al. 2023, Rocha et al. 2024, Englhardt et al. 2024, Johnson et al. 2024, Martikkala et al. 2024, Johnson et al. 2025], mostrando o ganho de produtividade no desenvolvimento para embarcados que envolvem questões de sensoriamento e temporização.

Sem uso de LLMs, iniciativas como a de [Pereira and Oyamada 2023] utilizam o *OpenLED Race* e jogos como *Connect-4* em matrizes de LED com programação tradicional para atrair o público em feiras de ciências. Este trabalho expande essa premissa ao integrar várias ideias: uso de LLMs para auxiliar na programação e motivar estudantes a explorar as explicações, conectividade remota, proposta de jogos, uso do celular com interface com Wifi, LEDs para efeito visual atrativo. Além disso, este trabalho é uma extensão de uma proposta inicial de trabalho apresentada como resumo [Freitas et al. 2025].

7. Conclusão

Este trabalho apresentou uma metodologia com LLMs para o desenvolvimento de jogos de Arcade utilizando LEDs endereçáveis, microcontroladores e interfaces móveis. A abordagem inicia com a validação em JavaScript no ambiente Google Colab e evoluindo para a implementação em dispositivos como o NodeMCU/ESP32/Arduino. Os resultados experimentais indicam que ferramentas como Gemini Pro, Claude, Copilot e ChatGPT possuem eficácia na geração e tradução de códigos para este domínio específico. A utilização de matrizes de LEDs mostrou-se uma solução de baixo custo e impacto visual de forma minimalista. A integração de protocolos de comunicação permitiu transformar o celular em um controle universal, eliminando barreiras e aumentando a interatividade em ambientes educacionais. O uso do NotebookLM complementou o ciclo de desenvolvimento, provendo uma camada de documentação e análise pedagógica. Como trabalhos futuros, planeja-se realizar um estudo de caso pedagógico formal para avaliar o ganho de engajamento e aprendizado em turmas de graduação e ensino médio.

8. Agradecimentos e Declaração de Uso de Inteligência Artificial

O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Código de Financiamento 001, CNPq Bolsa Produtividade, da Fapemig APQ-01577-22, programa institucional de bolsas IC FAPEMIG e CNPq, Funarbe e do Curso de Especialização Lato Sensu em Robótica da Universidade Federal de Viçosa.

Ferramentas de Inteligência Artificial Generativa baseadas em LLM foram utilizadas neste artigo e avaliadas na geração dos códigos, todos disponibilizados no material suplementar [UFV 2026], que contém os *prompts*, os códigos em JavaScript, os links para execução na plataforma Wokwi e os códigos para os microcontroladores ESP com Wifi. Além disso, as LLMs foram usadas para auxiliar na estruturação do texto e revisão gramatical/sintática do conteúdo. Todos os códigos foram validados pelos autores. Usamos o NotebookLM para documentação dos códigos e geração de ilustrações.

Referências

Araújo, H. C., Lisboa, M. O., Pereira, P. H. C., de Castro Freitas, I., Villela, M. L. B., and Ferreira, R. (2026). Desenvolvimento de ferramentas educacionais para computação com apoio de modelos generativos de linguagem. *Revista Brasileira de Informática na Educação*, 34:279–313.

- Englhardt, Z., Li, R., Nissanka, D., Zhang, Z., Narayanswamy, G., Breda, J., Liu, X., Patel, S., and Iyer, V. (2024). Exploring and characterizing large language models for embedded system development and debugging. In *Extended Abstracts of the CHI Conference on Human Factors in Computing Systems*, pages 1–9.
- Ferreira, R., Canesche, M., Jamieson, P., Neto, O. P. V., and Nacif, J. A. (2024a). Examples and tutorials on using google colab and gradio to create online interactive student-learning modules. *Computer Applications in Engineering Education*, page e22729.
- Ferreira, R., Sabino, C., Canesche, M., Neto, O. P. V., and Nacif, J. A. (2024b). Aiot tool integration for enriching teaching resources and monitoring student engagement. *Internet of Things*, 26:101045.
- Freitas, I., Coura, P., Nacif, J. A., and Ferreira, R. (2025). Programação de jogos com google colab e chatgpt. In *Lab. de Ideias - Simpósio Brasileiro de Educação em Computação (EDUCOMP)*, pages 22–24. SBC.
- Johnson, D., Estepp, C., and Doss, W. (2025). Using generative ai for microcontroller programming: Effects on experienced and inexperienced programmers. *Journal of Agricultural Systems, Technology, and Management*, 36(1):27–41.
- Johnson, D. M., Doss, W., and Estepp, C. M. (2024). Agriculture students’ use of generative artificial intelligence for microcontroller programming. *Natural Sciences Education*, 53(2):e20155.
- Li, Y., Shi, J., and Zhang, Z. (2024). An approach for rapid source code development based on chatgpt and prompt engineering. *IEEE Access*, 12:53074–53087.
- Martikkala, A., David, J., and Ituarte, I. F. (2024). Chatgpt+ arduino-a powerful approach for unique use-case tailored sensors. In *IET Conference Proceedings CP885*, volume 2024, pages 124–131. IET.
- Pereira, M. F. and Oyamada, M. S. (2023). Applying arduino and individually addressable led strips to the development of interactive materials to promote computer science undergraduate course. In *Congresso Latino-Americano de Software Livre e Tecnologias Abertas (Latinoware)*, pages 97–101. SBC.
- Petrović, N., Koničanin, S., and Suljović, S. (2023). Chatgpt in iot systems: Arduino case studies. In *IEEE Int. Conference on Microelectronics (MIEL)*.
- Rocha, A., Sousa, L., Alves, M., and Sousa, A. (2024). The underlying potential of nlp for microcontroller programming education. *Computer Applications in Engineering Education*, 32(6):e22778.
- Sambo, K. U. et al. (2025). Barriers to gamification adoption in education: A systematic review of cultural, technological, and institutional challenges in developing countries. *Information Development*.
- UFV (2026). Repositório com Prompt e Demonstrações. <https://github.com/arduinoufv/ideia26/>. [Online].
- Yulia, N. M. et al. (2025). Use of game-based learning media education as an effort to increase interest elementary school students learning. *JERIT: Journal of Educational Research and Innovation Technology*, 2(1):38–45.