GodotDriver: Conectando Jogos e Robótica em um Mundo Ciber-Físico

GodotDriver: Connecting Games and Robotics in a Cyber-Physical World

Fernando N. S. Almeida[1], André Koscianski[1]

[1] Universidade Tecnológica Federal do Paraná (UTFPR)
Rua Washington Subtil Chueire, 330, Ponta Grossa – PR – Brazil

fernandoalmeida@alunos.utfpr.edu.br, koscianski@utfpr.edu.br

Abstract. Introduction*: Toys that use mechanisms and electronics have existed for almost a century, and received a boost from computers. Smaller components, new sensors, wireless networks, are a few elements that have led to novel interface types. All such items become more affordable over time.* ***Objective****: This work develops a low-cost platform for cyber-physical toys and games, focusing on a remote-controlled car.* ***Methodology****: The prototype combines microcontrollers, WiFi, and the open-source Godot game engine.* ***Results****: The prototype features low energy consumption and reduced network usage, making it a functional and cost effective concept. The platform serves as a basis for games that are different from the more traditional ones, and paves the way for research and adaptations.*
Keywords: *cyber-physical toys, games, microcontrollers, Godot.*

Resumo. Introdução*: Brinquedos combinando mecanismos e eletrônica existem há quase cem anos, e receberam um impulso da computação. Componentes menores, novos sensores, redes sem fio, são alguns elementos que levaram a novas formas de interface. Todas essas tecnologias são hoje mais acessíveis.* ***Objetivo****: Este trabalho desenvolve uma plataforma de baixo custo para brinquedos e jogos ciber-físicos, focando um carrinho de controle remoto.* ***Metodologia****: A proposta combina microcontroladores, WiFi e o motor de jogos de código aberto Godot.* ***Resultados****: Características como consumo energético, uso de rede e custo resultaram em um produto funcional e de fácil acesso. A plataforma serve de base para jogos diferentes dos tradicionais, e abre caminho para pesquisas e adaptações.*
Palavras-chave: *brinquedos ciber-físicos, jogos, microcontroladores, Godot.*

1. Introdução

Dispositivos eletromecânicos são encontrados em uma grande variedade de brinquedos e jogos para todas as idades, e fazem parte da história do entretenimento ao lado de computadores. Alguns exemplos famosos no início da história dessa fusão são o fliperama, com quase um século de idade, e depois os jogos da SEGA como "Periscópio" de 1968, em que o usuário mirava alvos móveis, e "Grand prix", de 1969, que simulava uma corrida de automóveis e tinha um volante como controle.

Evoluções de software e hardware trouxeram mais formas de interação de usuários com computadores e, em decorrência, novas ideias de jogos. Dois dispositivos importantes nesse sentido foram o Wii Remote, lançado em 2006, e o Microsoft Kinect,

comercializado em 2010. Os dois periféricos permitiram trazer a participação física do usuário para dentro dos jogos de computador. Outro exemplo marcante surgiu da exploração do conjunto de sensores disponíveis em telefones celulares, no jogo Pokemon Go.

Apesar do interesse, a criação e comercialização de jogos que cruzam a fronteira entre físico e virtual esbarra tanto no custo de dispositivos quanto no escopo. Os periféricos dos exemplos citados foram criados em plataformas específicas e não foram projetados com intenção de permitir modificações: criar novos brinquedos depende então de produzir novo hardware específico. Uma alternativa que pode ser explorada para o desenvolvimento de produtos vem de microcontroladores como Arduíno e ESP32, que tem custo baixo e contam com um conjunto variado de sensores e atuadores.

Este trabalho apresenta o desenvolvimento de uma plataforma para brinquedos ciber-físicos, partindo da ideia de um carrinho de controle remoto. Foi desenvolvido um protótipo baseado no microcontrolador ESP32 e equipado com câmera. O carrinho é controlado por Wi-fi a partir de um aplicativo desenvolvido em um motor de jogos, que pode ser executado em um computador comum, ou um smartphone. O artigo discute características do projeto e alternativas de modificações e extensões. São apresentadas ideais para desenvolvimento de jogos com a plataforma. Apresentam-se também limitações do projeto, e por fim indicativos de custos de produção com vistas a empreendedores.

2. Brinquedos ciber-físicos

Brinquedos empregando mecanismos móveis são relativamente antigos, com exemplos iniciais como caixas de música e carrinhos de corda. Dispositivos eletromecânicos começaram a aparecer mais recentemente; o primeiro exemplo é possivelmente o Pinball dos anos 1930 [DeMaria e Wilson 2002]. A partir daí um dos grandes fabricantes no setor foi a SEGA, produzindo arcades como "Periscope" e "Missile" nos anos 1960 [Horowitz 2018]. Outros nomes importantes nessa indústria foram a japonesa Namco, que criaria o famoso Pac-Man, e a americana Midway, do Mortal Kombat. Ao aumentar a complexidade e capacidade dos brinquedos, os fabricantes dessa época geralmente utilizavam o processador de 8 bits Zilog Z80 [Koyama 2023].

Ao longo da década de 2000, uma série de eventos daria impulso ao conceito de jogos ciber-físicos. Em 2003 a Playstation lançou o acessório EyeToy, contendo uma câmera de resolução 320x240 que permitia explorar comandos gestuais em jogos. Em paralelo, a Nintendo fazia pesquisas e registrava patentes, tendo interesse em interfaces mais naturais para os usuários[1]; seus consoles de jogos aceitavam periféricos indo desde diferentes tipos de joystick até uma luva (Power Glove lançada em 1989) e um tapete com sensores de pressão. Em 2006, foi comercializado o Wii Remote. A utilização de acelerômetros para comandar jogos com movimentos corporais abriu uma direção nova de design, e ampliou a imersão dos usuários [Lee 2008]. Outro fabricante que explorou essas ideias foi a Microsoft, que em 2010 lançaria o periférico Kinect, com câmeras e algoritmos de visão computacional mais potentes do que o EyeToy. Além de novas possibilidades para o projeto de jogos, o Kinect permitiu mais engajamento físico e interação entre usuários [Zhang 2012, Isbister e Mueller 2015].

---

[1] Um relato está disponível nesta URL: https://iwataasks.nintendo.com/interviews/wii/wii_remote/0/0/

Cada vez mais usados em jogos, os telefones celulares também tiveram um papel importante ao longo dessa história. A combinação de vários sensores embarcados, acesso remoto à Internet e poder de processamento, levou em 2016 ao lançamento do Pokemon Go, um jogo legitimamente revolucionário. Surgia a ideia de misturar o ambiente urbano com personagens virtuais, promovendo mais mobilidade, exploração de novos ambientes e mesmo promovendo atividade física e interação social [Rauschnabel et al. 2017, LeBlanc e Chaput 2017].

No mundo de código e projetos abertos, plataformas como Arduino e ESP32 têm sido exploradas no desenvolvimento de jogos e brinquedos. Uma das aplicações vantajosas desses microcontroladores está na prototipagem rápida, diminuindo a distância entre concepção - ou design - e engenharia [Mellis et al. 2007]. Um caminho possível é a criação de brinquedos que tratem informações do ambiente e reajam à interação física de um usuário [Hartmann et al. 2008]. A capacidade de conexões sem fio e a boa capacidade de processamento desses dispositivos, dá margem à possibilidades de interação entre grupos de usuários (Huang e Menozzi 2016).

Outros campos interessantes de exploração incluem o universo de interfaces com retorno háptico [Tsetserukou et al. 2010], as possibilidades de dispositivos ciber-físicos em educação [Resnick et al. 2009], e as repercussões no comportamento ou sociabilização de jogadores [Voida e Greenberg 2009, Mueller et al. 2014]. Finalmente, vale mencionar que projetos com hardware aberto (OSH, open-source hardware) vão além do escopo de hobby e se encaixam efetivamente com modelos de negócio rentável [Tomas et al. 2023].

No contexto deste projeto, o veículo robótico desenvolvido representa uma ponte entre o mundo virtual dos jogos digitais e o ambiente físico do usuário, criando possibilidades únicas de interação e entretenimento.

3. Projeto e Desenvolvimento

O projeto de uma plataforma de desenvolvimento de jogos ciber-físicos foi concebido com hardware de baixo custo e soluções de código aberto. A implementação baseou-se nos seguintes componentes:

- uma placa de desenvolvimento ESP32-CAM, dotada de um microcontrolador ESP32 e uma câmera embutida;
- um Arduino Uno;
- um chassi, motores e rodas de um kit genérico de brinquedo robótico;
- um aplicativo desenvolvido no motor Godot, responsável pela interface do usuário;
- uma solução ou protocolo simples de comunicação em rede.

Esses componentes e as relações entre eles estão representados na Figura 1.

Conforme mostra a Figura 1 o usuário interage com o robô por meio de um aplicativo desenvolvido no motor Godot, que pode ser disponibilizado para desktop (Windows/Linux) ou dispositivos móveis (Android). Isso permite a desenvolvedores utilizar todo o potencial do motor para criação de cenários combinando objetos virtuais

com imagens capturadas pela câmera. A comunicação entre aplicativo e robô é realizada via Wifi, utilizando o roteador local do usuário.

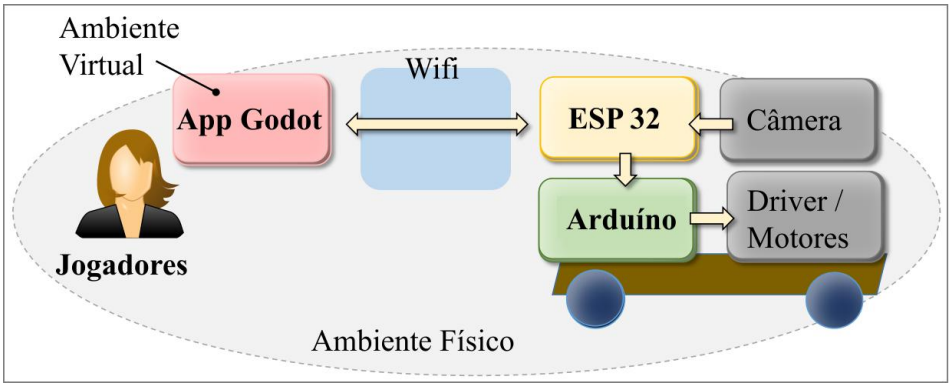

Figura 1. Diagrama de blocos do projeto.

O coração do projeto é uma placa ESP32-CAM de baixo consumo de energia, dotada de um microcontrolador ESP32 com conectividade WiFi e bluetooth, além de uma câmera [Dietz et al 2022]. A placa pode ser programada no ambiente Arduíno IDE e é equipada de um sistema operacional de tempo real (normalmente o FreeRTOS), com várias bibliotecas. Todas essas características aliadas a um baixo custo tornam a placa uma solução interessante para aplicações de IoT (Internet of Things) e prototipagem em geral.

O Arduino Uno é responsável pela modulação de largura de pulso (Pulse Width Modulation, PWM) para ajuste da velocidade variável de rotação dos motores do carro, alimentados por uma ponte H.

Essa configuração inicial do projeto atendia o requisito inicial de custo baixo, visando a produção de brinquedos com potencial comercial; além disso, a escolha de componentes também era compatível com as possibilidades do laboratório. Algumas variações a partir do projeto inicial serão discutidas no texto.

3.1. Robô

Todos os componentes foram acoplados no chassi do carro, e a alimentação das placas Arduino e ESP32-CAM foi obtida por uma bateria de 5V x 2A. Essa bateria alimenta o Arduino pela porta USB e a partir daí chega também ao ESP32-CAM. Os motores são excitados por um driver L298, cuja alimentação vem de quatro pilhas comuns de 1,5V. O ESP32 se comunica com o Arduíno por interface serial.

Após uma etapa inicial de configuração da câmera e de pinos de I/O, o ESP32 executa um laço contendo três tarefas:

- comandos: o software verifica chegada de pacotes com comandos para motores e pedidos de informações de depuração;
- imagens: o ESP32 recebe imagens da câmera com frequência de 40Hz, e as transmite via Wifi;
- sincronia: um sinal de sincronização (handshake) é enviado em broadcast com frequência de 1 Hz. O objetivo é permitir recuperação de erros por queda de conexão ou reinicialização de software.

Quatro comandos representados em strings são enviados do aplicativo Godot para o ESP32. O comando "fps" solicita o número de imagens por segundo processadas pelo robô; "ping" é usado para calcular o tempo de comunicação. O comando "myip" informa ao robô o endereço IP do dispositivo que executa o aplicativo Godot. Por fim, "power" contém parâmetros inteiros para comandar os motores do robô.

Uma vez recebido o comando de movimentação, o ESP32 transfere a informação à placa Arduíno pela interface serial. O comando contém parâmetros em formato texto para os motores do lado esquerdo e do lado direito do robô. O Arduino executa um algoritmo simples para converter valores do intervalo [-100, 100] para [-255, 255] e em seguida traduzir números positivos e negativos em saídas analógicas que são enviadas aos motores. As informações são ecoadas na saída serial para depuração.

O fluxo de imagens da câmera é transmitido via Wifi em uma taxa fixa de 40 quadros por segundo. A captura de imagens é tratada por uma biblioteca do ESP32, que executa um sistema operacional FreeRTOS. O formato foi configurado como JPEG, com uma resolução de 240 x 240; esse valor pode aumentar até 1600 x 1200, mas foi mantido baixo para os propósitos de construção e teste do protótipo.

A comunicação é feita com o protocolo UDP, partindo do princípio que uma rede Wifi local estaria menos sujeita a falhas do que comunicações em WAN (Wide Area Network), e que toda redução de complexidade de processamento é positiva quanto a gasto energético [Yüksel 2020]. Ainda assim, foram tomados cuidados para recuperação de erros, como um controle simples de pacotes de imagem.

Decidiu-se fragmentar os quadros de imagem em blocos de 1460 bytes, respeitado um tamanho máximo de unidade de transmissão (Maximum Transmission Unit - MTU), tipicamente de 1500 bytes, e deixando espaço para cabeçalhos de protocolo (UDP) e 5 Bytes com informação do robô. O formato dos pacotes é detalhado na Figura 2.

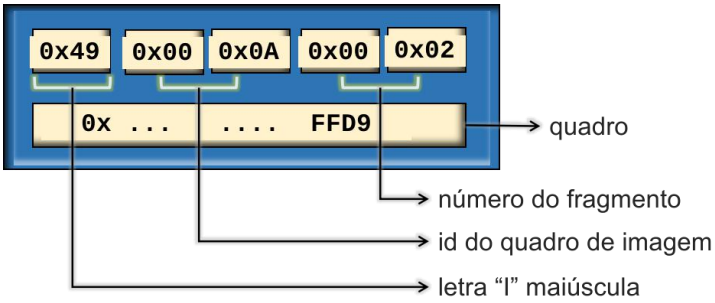

Figura 2. Pacote de imagem.

Os pacotes de imagem, conforme a Figura 2, contêm:

- um caractere “I” maiúsculo que distingue pacotes de imagem e de texto com informações de depuração;
- dois bytes com número de identificação da imagem;
- dois bytes com número de fragmento;
- os bytes restantes contêm dados.

3.2. Aplicativo Godot

O aplicativo desenvolvido teve o objetivo de validar o conceito e verificar o funcionamento do sistema. Foi composto de uma interface básica que apresenta o fluxo de vídeo recebido do robô, recebe comandos do usuário e os transmite ao carro.

Em testes iniciais o carro se comunicava via broadcast com o aplicativo, permitindo conexão automática sem intervenção do usuário. Apesar do conforto, a ideia acarreta sobrecarga da rede, podendo interferir com outros equipamentos e dispositivos IoT como alarmes e lâmpadas que são cada vez mais comuns. Para manter a configuração automática, o broadcast é usado apenas na inicialização: uma vez que o endereço IP do robô tenha sido descoberto, o aplicativo Godot retorna um comando “ myip ” , informando ao ESP32 o endereço do dispositivo em que o aplicativo é executado. Visando manter a configuração automática em caso de falhas de comunicação, manteve-se uma mensagem de broadcast periódico; no protótipo adotou-se uma frequência relativamente elevada, de 1 Hz.

Ao receber um pacote de imagem, o aplicativo Godot identifica o número da imagem e do fragmento. Essas informações são armazenadas juntamente com cada fragmento em uma estrutura de dicionário. À medida em que são recebidos, os fragmentos são armazenados em uma lista; uma vez a imagem completamente restaurada, é exibida na tela. Caso faltem fragmentos, ou uma imagem com um novo identificador seja transmitido, o aplicativo Godot descarta o quadro com falha e mantém a apresentação da última imagem corretamente recebida.

A Figura 3 mostra uma fotografia do protótipo em funcionamento.

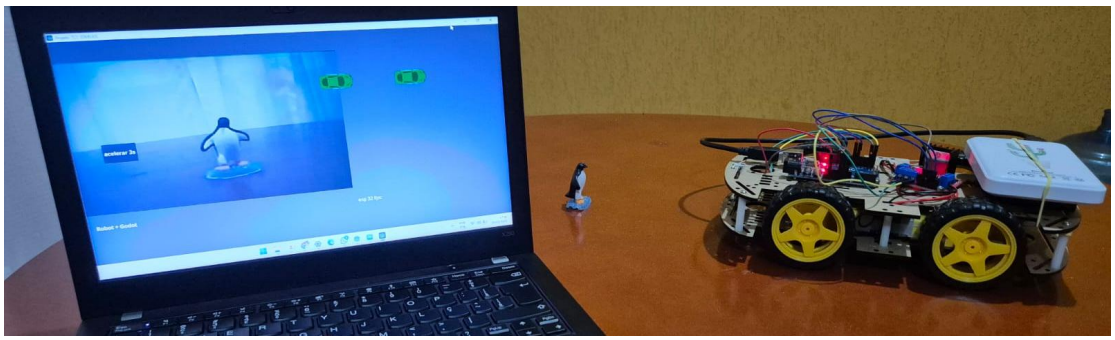

Figura 3. A plataforma completa, com aplicativo Desktop e o robô.

Na Figura 3 é possível observar a montagem do protótipo, e o tamanho comparado a um notebook de 14 polegadas. O chassi utilizado é bastante simples, composto por chapas finas de MDF, e não há carenagem, deixando a fiação exposta. A tela do notebook mostra um quadro de imagem capturado pelo robô; o tom azulado da tela é resultado da iluminação do ambiente.

4. Características e variações do projeto

A implementação do protótipo permitiu observar aspectos de funcionamento e características relevantes para, por exemplo, a produção em escala de um brinquedo. Pode-se pensar em um produto de prateleira acabado, ou kits para montagem e customização por usuários finais.

4.1. Características quantificáveis

Três pontos que dizem respeito ao desempenho do sistema requerem atenção.

O primeiro item a considerar é a capacidade de processamento. O Arduino é responsável por controlar via PWM os motores para realizar o movimento do robô. Esse microcontrolador opera com muita folga, e o gasto de energia mais significativo vem dos motores. Seria possível substituí-lo por um componente de menor capacidade e ainda mais barato, como eventualmente um microcontrolador PIC; ou empregar um ESP32-C2 (não disponível no momento da montagem) com capacidade de saída PWM. A placa ESP32-CAM coordena toda a lógica do robô, tratando a interface Wifi, comandos para o Arduíno, captura e envio de imagens e outras informações ao aplicativo Godot. A placa possui dois núcleos de processamento e executa um sistema operacional FreeRTOS, sendo capaz de executar tarefas paralelas. Esse recurso não foi utilizado, significando que existe margem para mais carga de processamento no robô.

Outro ponto que requer atenção é a autonomia de bateria. Segundo o datasheet da placa o consumo mínimo é de 180 mA. O manual de referência técnica recomenda uma fonte de alimentação de 5V com capacidade de 2A para garantir estabilidade de sinal e prevenir artefatos de imagem ao usar a câmera. A diferença entre o consumo nominal e a capacidade de alimentação recomendada sugere a necessidade de uma fonte de energia robusta para uma operação consistente.

O terceiro item é a comunicação em rede. Foram realizados testes com algumas configurações; os resultados estão na Tabela 1. Cada configuração foi testada durante 3 minutos. Todos os testes foram repetidos três vezes, sendo contados os quadros descartados no aplicativo Godot.

Tabela 1. Testes de configuração de câmera

Qualidade JPEG	Resolução	Hertz	Quadros descartados
máxima	1600 x 1200	60	19 %
alta	1600 x 1200	40	14 %
média	1280 x 720	40	8 %
baixa	800 x 600	30	12 %
mínima	240 x 240	30	5 %

O ESP32 foi configurado para enviar imagens comprimidas, mas retendo bastante qualidade. Considerando um tamanho médio de 15Kib por quadro, atinge-se um fluxo de cerca de 0.6 Mbps por segundo na configuração utilizada. Qualitativamente não se observaram problemas, havendo uma boa fluidez e qualidade de vídeo. Para referência, serviços de streaming de vídeo operam com valores típicos na casa de 3 Mbps, para uma resolução 720p e taxa de 24 quadros por segundo.

4.2. Alternativas e adaptações

O uso de duas fontes de alimentação veio de restrições para montar o protótipo; para usuários finais seria mais conveniente uma única fonte de alimentação, direcionada a ambos circuitos de potência e controle. Isto pode ser feito com uso de um powerbank

recarregável, que pode ou ter saídas isoladas, ou ser complementado por um regulador de tensão para obter voltagens diferentes para os motores e os microcontroladores.

Considerando quatro motores DC com corrente média de 0,7A cada e um ESP32 consumindo 0,18A, a corrente total do sistema é de aproximadamente 3,00A. Com um powerbank de 10.000 mAh, a estimativa de autonomia é de 3,36 horas.

Uma possibilidade de simplificar é substituir a placa ESP32 por um modelo com saídas analógicas que seriam usadas em PWM, eliminando assim a necessidade de um microcontrolador Arduíno dedicado exclusivamente a essa função. Os benefícios são reduzir custos diretos de hardware, evitar a programação de um dispositivo adicional, e diminuir possibilidades de falhas e necessidades de manutenção.

A montagem mecânica simples e variações de características dos motores acarretam desvios cumulativos de posição ao longo do tempo. O uso de encoders de ângulo nas rodas permitira monitorar os deslocamentos. Outro caminho seria substituir motores DC por motores de passo, permitindo um controle preciso da posição angular. Isto também permitiria eliminar o microcontrolador Arduíno, porém seria preciso equacionar o custo dos motores.

Por fim, a lógica de broadcast com o propósito de zero-configuration networking poderia ser substituída por uma implementação de mDNS (multicast DNS); uma implementação é disponível em código aberto pela empresa Expressif.

4.3. Jogos

É pertinente apontar algumas possibilidades de desenvolvimento de jogos para utilização com o robô apresentado, considerando o potencial da plataforma, ainda que o tema extrapole os objetivos centrais deste artigo.

Objetos virtuais podem ser apresentados sobrepostos ao ambiente real. Esses objetos podem representar por exemplo power-ups ou prêmios, ou então armadilhas ou mesmo inimigos (NPC, non-playable characters). O posicionamento e movimentação desses objetos virtuais pode ser feito em referência aos comandos do usuário: por exemplo ao avançar o robô, um objeto virtual em frente ao carro irá se aproximar do mesmo. O tratamento de trajetórias curvas não apresenta grandes complicações, e a saída de um objeto do campo de visão da câmera pode ser tratado simplesmente com a eliminação do mesmo. O uso de algoritmos de visão computacional para detecção de objetos ou marcadores fiduciários é uma linha de trabalho que pode ser explorada para enriquecer a interação entre o robô e os elementos virtuais.

Um exemplo possível seria um jogo de ‘defesa de torre’, no qual o robô localiza e coleta recursos representados virtualmente para uma base, enquanto a defende de hordas de inimigos, com obstáculos físicos e interações baseadas em sensores.

Os jogos podem explorar outras ideias como captura de bandeira, áreas de combate, áreas que alterem o comportamento do carro (como reduzir sua velocidade). Outra possibilidade é a integração de múltiplos veículos, permitindo casos de uso mais ricos bem como a interação entre jogadores.

Por fim, existem várias possibilidades de extensão com impacto em projetos de jogos. Novos elementos podem ser incorporados ao robô; sensores, como por exemplo de ultrassom, podem fazer o robô reagir dinamicamente a obstáculos; um atuador como

uma garra permitiria manipular objetos, abrindo espaço para jogos envolvendo coleta ou organização de itens.

4.4. Produção

O robô pode ser explorado de diferentes maneiras, como produto de prateleira ou como um kit de montagem. Do ponto de vista de empreendedorismo e implementação em escala industrial, é possível fazer uma estimativa conservadora de custos; isto é apresentado na Tabela 2.

Tabela 2. Estimativa de custos

Componente	Custo Aproximado (R$)
Chassi, rodas e motores	R$ 40
ESP32 e eletrônicos	R$ 40
Powerbank (10.000 mAh)	R$ 80
Total por unidade	R$ **160**

A Tabela 2 apresenta valores correspondentes à aquisição de 100 componentes no mercado comum, apenas para referência. A produção real poderia se valer de fabricantes nacionais de placas PCB sob encomenda para a parte eletrônica, e para a parte física existe um número expressivo de fabricantes de moldes e peças plásticas customizadas.

5. Conclusões

A plataforma desenvolvida e apresentada neste artigo apresenta características distintivas de jogos de computador tradicionais, incorporando elementos de robótica e realidade aumentada em uma experiência única de entretenimento interativo. Do ponto de vista da experiência do usuário, o sistema permite incentivar maior engajamento tanto na fase de preparação do ambiente quanto durante a execução do jogo, criando uma experiência mais imersiva e tangível em comparação com jogos puramente virtuais.

Além disso, a viabilidade técnica e econômica do robô, demonstrada por sua simplicidade construtiva e pelo custo relativamente baixo de produção em pequena escala, reforça o potencial da plataforma como produto educacional ou comercial.

Outras linhas de pesquisa a partir da plataforma incluem o desenvolvimento de diferentes tipos de robô, tais como réplicas de animais ou então figuras humanóides, e também drones. Além da integração de algoritmos de visão computacional, seria possível adicionar áudio por meio de microfones e auto-falantes, o que levaria a ideia inicial a um outro patamar no universo de jogos e brinquedos eletrônicos.

Referências

DeMaria, R., & Wilson, J. L. (2002) "High Score!: The Illustrated History of Electronic Games." McGraw-Hill Osborne Media.

Dietz, H., Abney, D., Eberhart, P., Santini, N., Davis, W., Wilson, E., & McKenzie, M. (2022) “ESP32-Cam as a programmable camera research platform.” Imaging, 232(2), 10-2352.

Hartmann, B., Klemmer, S. R., Bernstein, M., Abdulla, L., Burr, B., Robinson-Mosher, A., & Gee, J. (2008) "Reflective Physical Prototyping through Integrated Design, Test, and Analysis." Proceedings of the 19th Annual ACM Symposium on User Interface Software and Technology (UIST '06), 299-308.

Horowitz, Ken (2018), The Sega arcade revolution: A history in 62 games. McFarland.

Huang, C. Y., & Menozzi, M. (2016) "A Low-cost Wireless Sensor Network System Using ESP32 and Mobile Devices for Interactive Gaming Applications." Proceedings of the 2016 Annual Symposium on Computer-Human Interaction in Play (CHI PLAY '16), 103-107.

Isbister, K., & Mueller, F. F. (2015) "Guidelines for the Design of Movement-Based Games and Their Relevance to HCI." Human-Computer Interaction, 30(3-4), 366-399.

Koyama, Y. (2023) Arcade Games (1): From Elemecha to Video Games: The Birth of Space Invaders and the Establishment of the Arcade Game Industry. In History of the Japanese Video Game Industry (pp. 15-25). Singapore: Springer Nature Singapore.

LeBlanc, A. G., & Chaput, J. P. (2017) "Pokémon Go: A Game Changer for the Physical Inactivity Crisis?" Preventive Medicine, 101, 235-237.

Lee, J. C. (2008). "Hacking the Nintendo Wii Remote." IEEE Pervasive Computing, 7(3), 39-45.

Mellis, D. A., Banzi, M., Cuartielles, D., & Igoe, T. (2007) "Arduino: An Open Electronic Prototyping Platform." Extended Abstracts on Human Factors in Computing Systems (CHI EA '07), 1395-1400.

Rauschnabel, P. A., Rossmann, A., & tom Dieck, M. C. (2017). "An Adoption Framework for Mobile Augmented Reality Games: The Case of Pokémon Go." Computers in Human Behavior, 76, 276-286.

Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K., & Kafai, Y. (2009) "Scratch: Programming for All." Communications of the ACM, 52(11), 60-67.

Thomas, Laetitia Marie, Evrard-Samuel, Karine, and Troxler, Peter (2023) "Building open source hardware business models." In Business Models and Strategies for Open Source Projects, pp. 50-79. IGI Global.

Tsetserukou, D., Neviarouskaya, A., Prendinger, H., Kawakami, N., & Tachi, S. (2010) "Affective Haptics in Emotional Communication." In *Proceedings of the 3rd International Conference on Affective Computing and Intelligent Interaction and Workshops* (ACII '09), 1-6.

Yüksel, M. E. (2020). Power consumption analysis of a Wi-Fi-based IoT device. Electrica, 20(1), 62-70.

Zhang, Z. (2012) "Microsoft Kinect Sensor and Its Effect." IEEE Multimedia, 19(2), 4-10.