IMPLEMENTAÇÃO DE APRENDIZADO POR REFORÇO EM HARDWARE PARA APLICAÇÕES DE COMPUTAÇÃO DE BORDA ("EDGE COMPUTING")[1]

MARQUES, Pedro dos Prazeres[2]; SOUZA, Olívia Furlani Camargo de[3]; LIMA, Felipe Neves de Sousa[4]; PIRES, Ricardo[5]; SOUSA, Miguel Angelo de Abreu de[6];

RESUMO

Este trabalho apresenta uma pesquisa de natureza aplicada, com enfoque exploratório e bibliográfico, cujo objetivo é desenvolver uma implementação do algoritmo de Aprendizado por Reforço (Reinforcement Learning) em um sistema de computação de borda, utilizando, em sua fase inicial, o microcontrolador ESP32. O aprendizado por reforço é uma subárea da Inteligência Artificial que permite capacitar agentes autônomos a tomar decisões em ambientes dinâmicos com base em interações sucessivas com o meio. Apesar de seu potencial em diversas aplicações, sua execução ainda é amplamente dependente de plataformas baseadas em software e computação em nuvem, o que pode acarretar instabilidades de conexão, aumento da latência e vulnerabilidades na segurança dos dados. A proposta deste estudo é investigar alternativas para viabilizar a execução local e embarcada do algoritmo, contribuindo com soluções que aliam portabilidade, desempenho e segurança. A escolha do ESP32 como plataforma inicial se justifica por seu bom desempenho computacional, dimensões reduzidas, ampla conectividade e custo acessível, características compatíveis com os requisitos de aplicações embarcadas. Essa abordagem visa validar a viabilidade do uso de hardware leve e de baixo custo como etapa preliminar à futura implementação em FPGA, conforme previsto no projeto.

Palavras-chave: Inteligência Artificial, Aprendizado por Reforço, Computação de borda, microcontroladores, ESP32.

INTRODUÇÃO

Atualmente, a Inteligência Artificial (IA) tem avançado de forma significativa, ganhando notoriedade e protagonismo em diversas áreas do conhecimento, como as ciências e engenharia (Silva;Spatti;Flauzino, 2010;Russell;Norvig, 2010). Uma de suas subáreas, o Reinforcement Learning (RL), ou simplesmente Aprendizado por Reforço, destaca-se por permitir que agentes autônomos aprendam a tomar suas decisões em ambientes dinâmicos, por meio de interação contínua e recompensas cumulativas (Sutton;Barto, 2018).

Diferentemente das técnicas de aprendizado supervisionado e não supervisionado, utilizadas comumente nas redes neurais, o RL fundamenta-se na experimentação, onde num ambiente dinâmico um agente escolhe ações e, com base nas consequências, isto é, recompensas, ajusta sua política de decisão. A aplicação do RL tem permitido avanços

[1] Projeto de Iniciação Científica (PIBIFSP)

[2]Estudante Engenharia de Controle e Automação;IFSP; São Paulo; SP;pedro.prazeres@aluno.ifsp.edu.br
[3] Estudante Engenharia de Controle e Automação;IFSP; São Paulo; SP;f.olivia@aluno.ifsp.edu.br
[4] Estudante Engenharia de Controle e Automação;IFSP; São Paulo; SP;neves.felipe@aluno.ifsp.edu.br
[5] Doutor em Sistemas Automáticos e Microeletrônicos; Instituto Federal de Educação, Ciência e Tecnologia de São Paulo - IFSP; ricardo_pires@ifsp.edu.br
[6] Doutor em Ciências; Instituto Federal de Educação, Ciência e Tecnologia de São Paulo - IFSP; angelo@ifsp.edu.br

significativos em jogos complexos, sistemas robóticos e na otimização de processos industriais(Sutton;Barto, 2018).

Um dos algoritmos mais utilizados nesse contexto é o Q-Learning, que estima uma função de valor de ação Q(s,a), que indica uma expectativa de recompensa futura ao executar uma ação “a” no estado “s”. Essa função é atualizada de forma iterativa, com base na equação de Bellman, o que permite um aprendizado sem um modelo prévio de ambiente (Sutton; Barto, 2018).

Equação de Bellman:

$$Q(s, a) = Q(s, a) + \alpha \times [r + \gamma \times \max Q(s', a') - Q(s, a)]$$

Onde:

- Q(s, a): valor da ação *a* no estado *s*
- α: taxa de aprendizado (learning rate)
- r: recompensa imediata recebida
- γ: fator de desconto (discount factor)
- s′: próximo estado alcançado
- a′: ação possível no próximo estado
- max Q(s′, a′): maior valor de ação no próximo estado

Os algoritmos de RL são comumente executados em software, em servidores remotos ou ambientes em nuvem (Sutton;Barto, 2018). Apesar do poder computacional dessa abordagem, ela impõe desafios como instabilidade de conexão e riscos de vazamento de dados. Surge assim a necessidade de soluções que propiciem a execução local dos algoritmos, conceito central da computação de borda, processando os dados diretamente nos dispositivos, com maior confiabilidade, segurança e autonomia. Nesse contexto, a implementação por meio de microcontroladores ou circuitos reconfiguráveis como os FPGAs (Field-Programmable Gate Arrays) são vantajosos, pois possibilitam a criação que arquiteturas dedicadas e paralelizáveis, possibilitando adaptar a lógica do Q-Learning para operações em hardware (Spano,2019).

Assim, este trabalho tem por finalidade a integração entre os avanços teóricos do aprendizado por reforço e as possibilidades práticas oferecidas por plataformas de computação de borda. Essa integração contribui para o desenvolvimento de sistemas inteligentes mais portáteis, ampliando o alcance da IA para dispositivos em hardware.

OBJETIVOS

Desenvolver um sistema embarcado de computação de borda, utilizando o microcontrolador ESP32, capaz de executar o algoritmo de aprendizado por reforço Q-Learning de forma autônoma e eficiente, visando aplicações portáteis e de baixo consumo energético.

Para chegar aos fins propostos no objetivo geral acima disposto, são sugeridos os seguintes objetivos específicos:

- Investigar os fundamentos teóricos do aprendizado por reforço, com foco no algoritmo Q-Learning e suas aplicações embarcadas;
- Analisar abordagens de implementação de algoritmos de Inteligência Artificial em sistemas de hardware, com ênfase em microcontroladores e FPGA;
- Estudar as principais arquiteturas de computação de borda e suas implicações em sistemas portáteis;
- Projetar e configurar um ambiente de testes baseado no microcontrolador ESP32, considerando limitações computacionais e requisitos de conectividade;
- Realizar testes de desempenho e viabilidade, considerando tempo de resposta, estabilidade e consumo energético.

METODOLOGIA

Para os dois primeiros objetivos, foi conduzida uma revisão bibliográfica em bases científicas como IEEE Xplore e Google Scholar, a fim de consolidar os fundamentos teóricos do aprendizado por reforço e suas aplicações embarcadas.

A seguir, foi projetado um ambiente de testes baseado no microcontrolador ESP32, escolhido por sua robustez computacional, conectividade Wi-Fi/Bluetooth integrada, baixo custo e tamanho reduzido. O algoritmo Q-Learning foi inicialmente desenvolvido em linguagem Python, e sua integração com o microcontrolador se deu por meio do protocolo MQTT, devido à sua leveza e praticidade de uso.

A programação do ESP32 foi realizada em C++, com o auxílio da IDE Arduino, permitindo a leitura de comandos enviados pelo código em Python e a execução de respostas controladas pelo algoritmo. A estrutura do código foi adaptada considerando as restrições de memória e tempo de processamento da plataforma utilizada.

Para os testes, serão avaliados aspectos como estabilidade de conexão, tempo de execução, confiabilidade da comunicação e comportamento do sistema frente às variações do ambiente. Os resultados obtidos servirão de base para a análise da viabilidade de migração para plataformas mais robustas, como o FPGA, cujos estudos preliminares serão conduzidos como parte do escopo final da pesquisa.

O projeto é desenvolvido no contexto de um grupo de estudos vinculado ao IFSP, que já dispõe de placas FPGA e acesso a laboratórios com infraestrutura compatível. Os softwares utilizados são de acesso gratuito e compatíveis com os equipamentos disponíveis no campus e com os dispositivos pessoais dos membros da pesquisa. Os custos eventuais foram arcados pelo orientador, mediante orçamento prévio e planejamento.

RESULTADOS E DISCUSSÃO

A implementação inicial do sistema de aprendizado por reforço (Q-Learning) foi conduzida com sucesso utilizando o microcontrolador ESP32, cuja arquitetura dual-core e conectividade integrada (Wi-Fi e Bluetooth) demonstraram-se adequadas para o processamento embarcado e a comunicação assíncrona entre sistemas. Essa escolha se baseia em sua robustez computacional e ampla compatibilidade com sensores e atuadores, atributos já reconhecidos em estudos voltados à automação e IoT (Espressif, s.d; Banzi; Shiloh, 2014; Circuitstate, 2022).

Figura 1 - Pinagem ESP32 (a) e diagrama de funcionamento MQTT (b).

Fonte(a) Playelek, s.d; (b) Quincozes; Tubino; Kazienko, 2019.

.

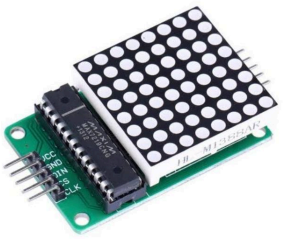
O ambiente de desenvolvimento, baseado na IDE Arduino e na linguagem C++, possibilitou o controle eficiente do hardware e a integração com bibliotecas específicas para o protocolo MQTT, optando por um Broker em nuvem, devido a facilidade e baixo custo. O protocolo MQTT, estruturado no paradigma publish-subscribe, demonstrou-se uma alternativa leve e eficaz para aplicações embarcadas, permitindo o envio e recebimento de mensagens com baixa latência, conforme discutido por Quincozes, Tubino e Kazienko (2019) e validado em aplicações recentes (Random Nerd Tutorials,s.d;Emqx, 2024).

Para a validação da comunicação, foi utilizada uma matriz de LEDs 8x8 com controlador MAX7219, que respondeu adequadamente aos comandos recebidos via MQTT. Essa etapa experimental confirmou a viabilidade de utilizar o ESP32 como elo entre algoritmos de IA desenvolvidos em Python e saídas físicas, como indicadores visuais, reforçando sua aplicação em sistemas autônomos embarcados (Embarcados, 2015). O sucesso dessa etapa permitiu estabelecer uma base sólida para a futura integração do algoritmo Q-Learning em ambientes com recursos limitados, sinalizando a possibilidade real de migração para FPGA, com ganhos em paralelismo e desempenho computacional (Spano,2019).

Figura 2 - Código de *reinforcement learning* (a) e Matriz de LEDs 8x8 (b).

```
stTeste = np.zeros( [ 25, 2 ] )
indTeste = 0
for indX in range( 5 ):
    for indY in range( 5 ):
        # print( 'indX:', indX, ' indY:', indY )
        stTeste[ indTeste ] = [ indX, indY ]
        indTeste += 1
acoesPreditas = Q.predict( stTeste )
print( '\n-------------------------------------------------------' )
print( '\npredições\n' )
for indTeste in range( 25 ):
    print( '--------------------' )
    print( 'stTeste:', stTeste[ indTeste ] )
    print( 'parado  :', acoesPreditas[ indTeste ][ 0 ] )
    print( 'cima    :', acoesPreditas[ indTeste ][ 1 ] )
    print( 'direita :', acoesPreditas[ indTeste ][ 2 ] )
    print( 'baixo:   ', acoesPreditas[ indTeste ][ 3 ] )
    print( 'esquerda:', acoesPreditas[ indTeste ][ 4 ] )
```

```
cima    : 2.767128887425802
direita : 1.88570562387233
baixo:    0.79901714413281
esquerda: 0.909613660415541
```

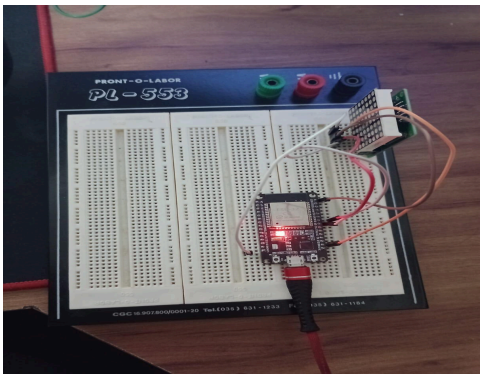


Fonte (a) Elaborado pelos autores; (b) Embarcados, 2015.

Na atual fase de desenvolvimento, a comunicação entre o microcontrolador ESP32 e uma aplicação externa em Python foi plenamente estabelecida por meio do protocolo MQTT. Esse marco representa um passo crítico na consolidação da arquitetura de comunicação do sistema embarcado, permitindo que mensagens sejam enviadas e interpretadas corretamente entre as camadas de software e hardware. O código desenvolvido na IDE Arduino permite que comandos enviados via computador sejam processados e revertidos em saídas físicas controladas — como a ativação seletiva de LEDs em uma matriz 8x8 MAX7219 — simulando, de forma visual, as respostas de um agente inteligente (Quincozes; Tubino; Kazienko, 2019; Espressif, s.d).

A arquitetura embarcada demonstrou comportamento estável em testes de conectividade, tempo de resposta e sincronização, validando o protótipo como plataforma viável para aplicações baseadas em aprendizado por reforço embarcado. Embora a aplicação atual esteja estruturada com comandos simples, o sistema já se encontra funcional como base prática para a posterior integração do algoritmo Q-Learning. Essa estrutura modular permite que as camadas de software sejam adaptadas de forma incremental ao aumento da complexidade algorítmica, respeitando os limites computacionais do ESP32 e promovendo escalabilidade gradual (Spano,2019; Embarcados, 2015).

Figura 3 - Implementação do circuito



Fonte: Elaborado pelos autores

Com a etapa de comunicação consolidada, os próximos avanços do projeto visam a incorporação progressiva do algoritmo Q-Learning no microcontrolador, utilizando como referência a lógica de atualização da equação de Bellman. A saída física da

aprendizagem será representada pela ativação lógica dos LEDs, funcionando como um feedback visual direto das decisões tomadas pelo agente com base nas recompensas acumuladas. Essa fase busca validar a capacidade do sistema de operar autonomamente em ambientes com restrições de hardware, mantendo coerência com os princípios da computação de borda (Sutton; Barto, 2018; Espressif, s.d).

CONSIDERAÇÕES FINAIS

O desenvolvimento deste projeto demonstrou a viabilidade de utilizar o microcontrolador ESP32 como plataforma de execução local para algoritmos de aprendizado por reforço, representando um avanço no uso de inteligência artificial embarcada. A comunicação bem-sucedida entre os comandos e o ESP32 via protocolo MQTT validou a proposta de integração entre software e hardware de baixo custo, com potencial para aplicações móveis e autônomas. A continuidade do trabalho visa otimizar a implementação do algoritmo Q-Learning e explorar sua migração para FPGAs, ampliando o desempenho e a eficiência do sistema em ambientes com recursos limitados.

REFERÊNCIAS

BANZI, Massimo; SHILOH, Michael. *Arduino: an open-source electronics prototyping platform*. 2. ed. Sebastopol: Maker Media, 2014.

CIRCUITSTATE. *DOIT ESP32 DevKit V1 Wi-Fi Development Board - Pinout Diagram & Reference*. [S.l.], 2022. Disponível em: https://www.circuitstate.com/pinouts/doit-esp32-devkit-v1-wifi-development-board-pinout-diagram-and-reference/. Acesso em: 27 jul. 2025.

EMBARCADOS. *Módulo matriz de LEDs com MAX7219*. [S.l.], 2015. Disponível em: https://embarcados.com.br/modulo-matriz-de-leds-com-max7219. Acesso em: 27 jul. 2025.

EMQX. *ESP32 connects to the free public MQTT broker publish & subscribe demo with Arduino IDE*. [S.l.], 2024. Disponível em: https://www.emqx.com/en/blog/esp32-connects-to-the-free-public-mqtt-broker. Acesso em: 27 jul. 2025.

ESPRESSIF SYSTEMS. *ESP32 Overview*. [S.l.], s.d. Disponível em: https://www.espressif.com/en/products/socs/esp32. Acesso em: 29 jul. 2025.

RANDOM NERD TUTORIALS. *ESP32 MQTT Publish and Subscribe with Arduino IDE*. [S.l.], s.d. Disponível em: https://randomnerdtutorials.com/esp32-mqtt-publish-subscribe-arduino-ide/. Acesso em: 27 jul. 2025.

PLAYELEK. *pinout-doit-32devkitv1*. [S.l.]: GitHub, s.d. Disponível em: https://github.com/playelek/pinout-doit-32devkitv1. Acesso em: 27 jul. 2025.

QUINCOZES, S. E.; TUBINO, E. R.; KAZIENKO, J. F. MQTT Protocol: Fundamentals, Tools and Future Directions. *IEEE Latin America Transactions*, v. 17, n. 9, p. 1439-1447, set. 2019. Disponível em: https://www.researchgate.net/publication/345356734. Acesso em: 31 jul. 2025.

RUSSELL, S. J.; NORVIG, P. *Artificial intelligence: a modern approach*. [S.l.]: Pearson Education, Inc., 2010.

SILVA, I. N.; SPATTI, D. H.; FLAUZINO, R. A. *Redes Neurais Artificiais para engenharia e ciências aplicadas*. São Paulo: Artliber, 2010.

SPANO, Sergio et al. An efficient hardware implementation of reinforcement learning: The q-learning algorithm. *IEEE Access*, v. 7, p. 186340-186351, 2019.

SUTTON, R. S.; BARTO, A. G. *Reinforcement learning: An introduction*. [S.l.]: MIT press, 2018.