https://doi.org/10.47734/regrasp.v10.04.p40-46

INSTITUTO FEDERAL
São Paulo
Campus São Paulo
REGRASP

Implementação de aprendizado por reforço em hardware para aplicações de computação de borda (“edge computing”)

Hardware reinforcement learning implementation for edge computing applications

Pedro dos Prazeres Marques
Graduando em Engenharia de Controle e Automação
IFSP / Campus São Paulo
https://orcid.org/0009-0001-1539-3551
pedro.prazeres@aluno.ifsp.edu.br

Olívia Furlani Camargo de Souza
Graduando em Engenharia de Controle e Automação
IFSP / Campus São Paulo
https://orcid.org/0009-0009-6131-8534
f.olivia@aluno.ifsp.edu.br

Felipe Neves de Sousa Lima
Graduando em Engenharia de Controle e Automação
IFSP / Campus São Paulo
https://orcid.org/0009-0006-9837-233X
neves.felipe@aluno.ifsp.edu.br

Ricardo Pires
Doutor em Sistemas Automáticos e Microeletrônicos pela Université Montpellier 2 - Sciences et Techniques,UM2, França
Docente do IFSP / Campus São Paulo
https://orcid.org/0000-0003-4677-8435
ricardo_pires@ifsp.edu.br

Miguel Angelo de Abreu de Sousa
Doutor em Engenharia Elétrica pela Escola Politécnica da Universidade de São Paulo (POLI-USP)
Docente do IFSP / Campus São Paulo
https://orcid.org/0000-0001-9056-1095
angelo@ifsp.edu.br

EDIÇÃO ESPECIAL - X ENCONTRO DE INICIAÇÃO CIENTÍFICA E PÓS-GRADUAÇÃO DO IFSP - CAMPUS SÃO PAULO (EICPOG)

Resumo

Este estudo apresenta uma pesquisa de caráter aplicado, com ênfase exploratória e bibliográfica, que visa desenvolver a implementação do algoritmo de Aprendizado por Reforço em um sistema de computação de borda, começando, em sua fase inicial, pelo microcontrolador ESP32. O aprendizado por reforço é um ramo da Inteligência Artificial que possibilita que agentes autônomos façam escolhas em ambientes dinâmicos a partir de interações contínuas com o ambiente. Embora tenha potencial em várias aplicações, sua implementação ainda é bastante dependente de plataformas de software e computação em nuvem, o que pode resultar em instabilidades de conexão, aumento da latência e vulnerabilidades na segurança das informações. O objetivo deste estudo é analisar opções para permitir a implementação local e integrada do algoritmo, oferecendo soluções que combinam portabilidade, eficiência e segurança. A seleção do ESP32 como plataforma inicial é válida devido ao seu elevado desempenho computacional, tamanhos compactos, ampla conectividade e preço acessível, atributos que se alinham aos requisitos de aplicações embarcadas. Essa estratégia tem como objetivo confirmar a viabilidade da utilização de hardware simples e econômico como fase inicial para a futura aplicação em chips do tipo FPGA, conforme estipulado no projeto.

Palavras-chave: inteligência artificial, aprendizado por reforço, computação de borda, microcontroladores, ESP32.

Abstract
This study presents an applied research project with exploratory and bibliographic emphasis, aimed at developing the implementation of a Reinforcement Learning algorithm in an edge computing system, beginning in its initial phase with the ESP32 microcontroller. Reinforcement Learning is a branch of Artificial Intelligence that enables autonomous agents to make decisions in dynamic environments through continuous interaction with their surroundings. Although it holds significant potential across various applications, its implementation still largely depends on software platforms and cloud computing, which can lead to connection instabilities, increased latency, and security vulnerabilities. The objective of this study is to analyze alternatives that enable local and integrated execution of the algorithm, offering solutions that combine portability, efficiency, and data security. The selection of the ESP32 as the initial platform is justified by its strong computational performance, compact form factor, extensive connectivity, and low cost, making it suitable for embedded applications. This strategy aims to confirm the feasibility of using simple and inexpensive hardware as an initial stage for future implementation on FPGA, as established in the project.
Keywords: artificial intelligence, reinforcement learning, edge computing, microcontrollers, ESP32.

Introdução

A Inteligência Artificial (IA) se estabeleceu como um dos principais motores de inovação do século XXI, ampliando sua abrangência para setores como saúde, indústria, transporte, comunicação e várias áreas científicas (Russell & Norvig, 2010). Nesse vasto domínio, sobressai o Reinforcement Learning (RL), ou Aprendizado por Reforço, uma estratégia que possibilita a agentes autônomos adquirirem habilidades para atuar em contextos dinâmicos e imprevisíveis. Esse conhecimento se dá por meio de interações constantes com o ambiente, pela acumulação de recompensas e pelo aperfeiçoamento gradual de seu comportamento (Sutton & Barto, 2020).

O RL se distingue de métodos supervisionados e não supervisionados por não necessitar de dados rotulados ou da identificação de padrões já existentes. Seu funcionamento se fundamenta essencialmente na experimentação: o agente escolhe ações em um estado específico, observa as consequências – positivas ou negativas – e ajusta sua política para otimizar a soma das recompensas ao longo do tempo. Esse método de aprendizado tem permitido progressos significativos em domínios como jogos sofisticados, robótica autônoma e otimização de processos industriais (Sutton & Barto, 2020).

Entre os métodos mais populares no RL, o Q-Learning se tornou um dos mais relevantes devido à sua utilização prática. O algoritmo faz estimativas da função de valor de ação Q(s, a), que indica a expectativa de retorno futuro ao executar uma ação “a” no estado “s”. Essa função é continuamente aprimorada pela equação de Bellman, possibilitando que o agente aprenda sem precisar de um modelo anterior do ambiente (Sutton & Barto, 2020; Souza & Braga, 2009).

Equação de Bellman:

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

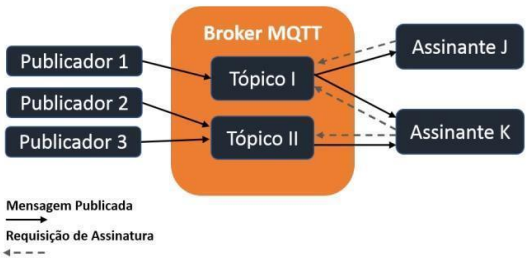
Onde:

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

De modo geral, algoritmos de RL são implementados em plataformas de software, frequentemente em servidores distantes ou serviços na nuvem (Sutton & Barto, 2020). Embora proporcionem significativa capacidade computacional, essas soluções enfrentam desafios como a dependência da conectividade e riscos associados à segurança das informações. Por isso, aumenta o interesse por soluções que possibilitem executar os algoritmos localmente, um princípio essencial da computação de borda, proporcionando mais confiabilidade, privacidade e autonomia operacional. Nesse cenário, a utilização de circuitos dedicados, como microcontroladores ou FPGAs (Field-Programmable Gate Arrays), se mostra particularmente interessante (Sousa, Pires & Del-Moral-Hernandez, 2020), uma vez que permite criar arquiteturas específicas, paralelas e ajustadas à lógica do Q-Learning em hardware (Spano et al., 2019).

Neste estudo, a integração entre algoritmos de RL e plataformas de computação em borda foi analisada, com ênfase na implementação de Q-Learning e na comunicação entre módulos utilizando o protocolo MQTT (Message Queuing Telemetry Transport), que é amplamente adotado em sistemas embarcados e na Internet das Coisas devido à sua leveza e arquitetura baseada no modelo Publish-Subscribe (Quincozes, Tubino, & Kazienko, 2019), como mostrado na Figura 1. Essa junção possibilitou examinar a viabilidade de implementar RL em dispositivos locais, avaliando desempenho, estabilidade, autonomia e segurança da solução sugerida.

Figura 1 - Diagrama da estrutura de funcionamento do MQTT



Fonte: Quincozes, Tubino e Kazienko, 2019.

Assim, este trabalho expõe os resultados agregados da investigação, detalhando a base teórica escolhida, a metodologia aplicada, os testes executados e as análises obtidas. Igualmente, são abordadas as restrições identificadas e possíveis desdobramentos futuros, destacando a importância da utilização de aprendizado por reforço em arquiteturas integradas e sua contribuição para o progresso da computação de borda aplicada a sistemas inteligentes.

Metodologia

O presente estudo objetivou criar um sistema de computação de borda embarcado, empregando o microcontrolador ESP32, que possa implementar o algoritmo de aprendizado por reforço Q-Learning de maneira autônoma e eficaz, focando em aplicações móveis e com baixo consumo de energia.

Para atingir tais metas, executaram-se os seguintes passos metodológicos:

i. Pesquisaram-se os princípios teóricos do aprendizado por reforço, com ênfase no algoritmo Q-Learning e suas implementações integradas, assim como os métodos de aplicação de algoritmos de Inteligência Artificial em plataformas de *hardware*;

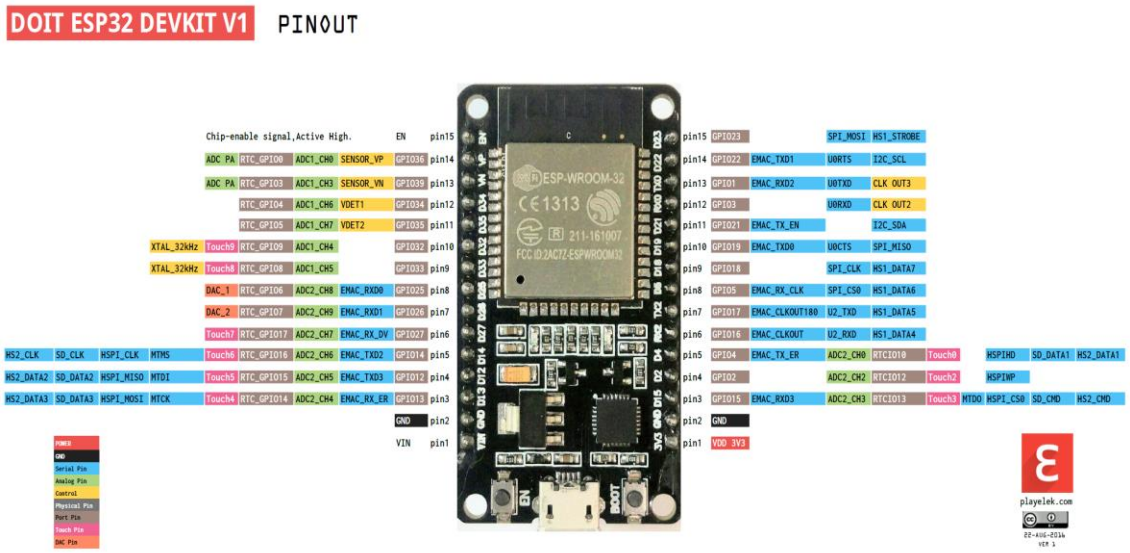
ii. Analisaram-se as principais arquiteturas de computação de borda para que fosse possível elaborar e ajustar um ambiente de testes fundamentado no microcontrolador ESP32, levando em conta restrições computacionais e exigências de conectividade;

iii. Executaram-se testes de desempenho e viabilidade, levando em conta o tempo de resposta, estabilidade e consumo de energia. Desta forma, foi possível avaliar a viabilidade do projeto em contextos reais, além das situações simuladas ideais.

As pesquisas foram feitas por uma revisão da literatura em bases científicas como IEEE Xplore e Google Scholar, com o intuito de consolidar os princípios teóricos do aprendizado por reforço e suas aplicações em sistemas embarcados.

A implementação utilizou um ambiente de testes desenvolvido com o microcontrolador ESP32, selecionado por sua capacidade de processamento, conectividade Wi-Fi/Bluetooth embutida, custo acessível e dimensões compactas. O Q-Learning foi primeiramente criado em Python, e sua conexão com o microcontrolador ocorreu através do protocolo MQTT, por causa de sua facilidade de utilização. Na Figura 2, é possível ver a função dos pinos do microcontrolador utilizado.

Figura 2 - Pinagem ESP32



Fonte: Playelek, s.d.

A programação do ESP32 foi feita em C++, utilizando a IDE Arduino, possibilitando a captura de comandos enviados pelo código em Python e a execução de respostas geradas pelo algoritmo. A organização do código foi ajustada levando em conta as limitações de memória e o tempo de execução da plataforma empregada. Na Figura 3 é possível ver um trecho do código em python, onde as ações do agente são mapeadas de acordo com cinco movimentos e sua respectiva saída.

Figura 3 - Imagem do código em Python

```
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
```

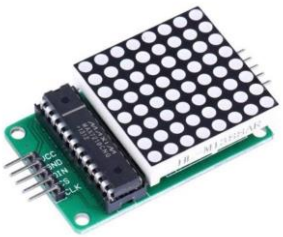
Fonte: Elaborado pelos autores.

Nos testes, foram considerados fatores como estabilidade da conexão, duração da execução, confiabilidade da comunicação e reações do sistema às mudanças do ambiente. Os resultados obtidos fornecerão suporte para a avaliação da viabilidade de transição para plataformas mais avançadas, como o FPGA, cujos estudos iniciais serão realizados como parte do escopo final da investigação.

A comunicação entre o código em Python — onde o algoritmo Q-Learning foi inicialmente estruturado — e o ESP32 foi estabelecido por meio do protocolo MQTT, utilizando um *broker* em nuvem. Essa arquitetura demonstrou baixa latência, estabilidade na transmissão de mensagens e capacidade de operar por longos períodos sem perda de sincronização. Esse resultado confirma o MQTT como uma solução leve e adequada para sistemas embarcados que demandam troca contínua de estados e ações, alinhado ao observado na literatura especializada (Quincozes, Tubino & Kazienko, 2019; Random Nerd Tutorials, s.d.; EMQX, 2024).

Para validar o fluxo de comunicação e os comandos de controle, foi utilizada uma matriz de LEDs 8×8 controlada pelo módulo MAX7219 - como a apresentada na Figura 2. Esse componente foi mapeado internamente como um vetor de 64 posições, possibilitando a representação visual do ambiente e o deslocamento do agente de forma direta e eficiente. Os testes mostraram que o ESP32 respondeu adequadamente às mensagens recebidas via MQTT, acionando padrões visuais que simulam movimentos e decisões de um agente inteligente. Essa etapa experimental confirmou que o microcontrolador pode funcionar como elo entre algoritmos de IA e saídas físicas, reforçando sua aplicabilidade em sistemas autônomos embarcados (Embarcados, 2015).

Figura 4 - Matriz de LEDs 8x8



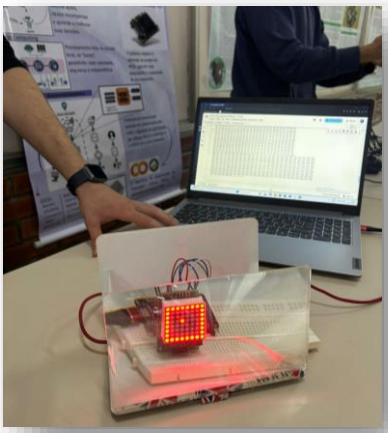
Fonte*:* Embarcados, 2015.

Resultados

Testes de funcionamento de todo o sistema foram realizados em ambiente físico real, incluindo cenários com obstáculos definidos na matriz, isto é, dentre as posições possíveis de movimentação do agente dentro da matriz alguns pontos foram definidos como obstáculos, para que o agente precisasse desviar dos mesmos para alcançar o objetivo definido[1]. Nessas situações, o sistema manteve a estabilidade tanto na comunicação quanto na execução dos comandos, demonstrando que a solução proposta suporta condições externas não controladas, como variações de alimentação, interferências ambientais e mudanças de conectividade. Os resultados obtidos indicam que a arquitetura é robusta o suficiente para operar fora de ambientes totalmente idealizados, um aspecto fundamental para aplicações reais de computação de borda[1].

A análise integrada dos experimentos indica que o sistema é funcional como base modular e escalável para a futura incorporação do algoritmo Q-Learning diretamente no microcontrolador, o sistema montado pode ser visto na Figura 5. A arquitetura programada em C++ foi estruturada de modo a permitir expansão gradual da complexidade algorítmica, respeitando os limites computacionais do ESP32 e garantindo que a lógica de atualização da equação de Bellman possa ser integrada sem necessidade de reformulação completa do sistema (Spano, 2019; Sutton & Barto, 2018).

Figura 5 - Divulgação por meio de demonstração prática no ambiente do campus da instituição



Fonte: Elaborado pelos autores.

[1] Um vídeo com uma sequência de testes do sistema completo pode ser encontrado no link: https://drive.google.com/file/d/1Nnn5NU0EVTc07_ghkp787s8sbRjqRTGm

A implementação inicial do sistema embarcado demonstrou que o microcontrolador ESP32 é capaz de operar como plataforma viável para o desenvolvimento de aplicações baseadas em Aprendizado por Reforço em ambiente de computação de borda. A arquitetura dual-core do dispositivo, aliada à integração nativa de Wi-Fi e Bluetooth, possibilitou a execução eficiente de tarefas de comunicação assíncrona e controle em tempo real, características essenciais para a construção de agentes inteligentes em hardware (Banzi & Shiloh, 2014; Circuitstate, 2022; Espressif, s.d.).

Esses resultados também apontam para a viabilidade de uma futura migração para plataformas FPGA, que podem fornecer maior paralelismo, desempenho e eficiência energética, conforme previsto no planejamento do projeto. Assim, o conjunto de testes realizados valida a proposta de utilizar hardware leve, de baixo custo e alta disponibilidade como etapa inicial para aplicações avançadas de Aprendizado por Reforço embarcado.

Conclusão

A execução deste projeto evidenciou a possibilidade de empregar o microcontrolador ESP32 como uma plataforma local para algoritmos de aprendizado por reforço, representando um progresso na aplicação de inteligência artificial de forma embarcada. A comunicação eficaz entre os comandos e o ESP32 através do protocolo MQTT comprovou a viabilidade da integração entre software e hardware acessível, com possibilidades para utilizações móveis e autônomas. Trabalhos futuros terão como objetivo otimizar a execução de algoritmos como Q-Learning e Deep Reinforcement Learning e investigar suas transições para FPGAs, aumentando tanto o desempenho quanto a eficiência do sistema em contextos com recursos restritos.

Referências

Banzi, M., & Shiloh, M. (2014). *Arduino: An open-source electronics prototyping platform* (2ª ed.). Maker Media.

Circuitstate. (2022). *DOIT ESP32 DevKit V1 Wi-Fi development board: Pinout diagram & reference*. https://www.circuitstate.com/pinouts/doit-esp32-devkit-v1-wifi-development-board-pinout-diagram-and-reference/

Embarcados. (2015). *Módulo matriz de LEDs com MAX7219*. https://embarcados.com.br/modulo-matriz-de-leds-com-max7219

EMQX. (2024). *ESP32 connects to the free public MQTT broker: Publish & subscribe demo with Arduino IDE*. https://www.emqx.com/en/blog/esp32-connects-to-the-free-public-mqtt-broker

Espressif Systems. (s.d.). *ESP32 overview*. https://www.espressif.com/en/products/socs/esp32

Playelek. (s.d.). *Pinout DOIT 32 DevKit V1* [Repositório GitHub]. https://github.com/playelek/pinout-doit-32devkitv1

Quincozes, S. E., Tubino, E. R., & Kazienko, J. F. (2019). MQTT protocol: Fundamentals, tools and future directions. *IEEE Latin America Transactions, 17*(9), 1439–1447. https://doi.org/10.1109/TLA.2019.8991277

Random Nerd Tutorials. (s.d.). *ESP32 MQTT: Publish and subscribe with Arduino IDE*. https://randomnerdtutorials.com/esp32-mqtt-publish-subscribe-arduino-ide/

Russell, S. J., & Norvig, P. (2010). *Artificial intelligence: A modern approach* (3rd ed.). Pearson.

Silva, I. N., Spatti, D. H., & Flauzino, R. A. (2010). *Redes neurais artificiais para engenharia e ciências aplicadas*. Artliber.

Sousa, M. A. A., Pires, R., & Del-Moral-Hernandez, E. (2020). Somprocessor: A high-throughput FPGA-based architecture for implementing self-organizing maps and its application to video processing. *Neural Networks, 125*, 349–362.

Souza, E. S., & Braga, A. P. (2009). *Aprendizado por reforço aplicado ao controle*. Revista Controle & Automação, 20(3), 284–295.

Spano, S., Fanni, A., Marras, M., Massidda, L., Pani, D., Raffo, L., & Tuveri, G. (2019). An efficient hardware implementation of reinforcement learning: The Q-learning algorithm. *IEEE Access, 7*, 186340–186351. https://doi.org/10.1109/ACCESS.2019.2959466

Sutton, R. S., & Barto, A. G. (2018). *Reinforcement learning: An introduction* (2nd ed.). MIT Press.