

Conectividade CoAP para Anti-Colisão em Sistema de Estacionamento Inteligente

Elizabeth Juliana Murara e Janine Kniess

Resumo— Este trabalho apresenta um sistema IoT de estacionamento inteligente que através de microcontroladores orquestra via protocolo CoAP a comunicação entre um *gateway* gerenciador e robôs móveis. A arquitetura implementa a alocação automática de vagas, e propõe um algoritmo anti-colisão baseado em zonas. O uso do protocolo CoAP garante a baixa latência para a tomada de decisão em tempo real. O protocolo MQTT provê monitoramento e notificações. Os resultados validam a efetividade de ambos os protocolos, na automação de manobras.

Palavras-Chave— IoT, Estacionamento Inteligente, CoAP.

Abstract— This paper presents an IoT smart parking system that, through microcontrollers, orchestrates communication between a management gateway and mobile robots via the CoAP protocol. The architecture implements automatic parking space allocation and proposes a zone-based anti-collision algorithm. The use of the CoAP protocol ensures low latency for real-time decision-making. The MQTT protocol provides monitoring and notifications. The results demonstrate the effectiveness of both protocols in automating maneuvers.

Keywords— IoT, Smart Parking, CoAP.

I. INTRODUÇÃO

A IoT oferece diversas tecnologias e protocolos que permitem automatizar estacionamentos, como o uso de sensores para monitorar as vagas, e protocolos de comunicação que podem transmitir em tempo real informações sobre as vagas disponíveis e mecanismos de cobrança [1].

Essa pesquisa introduz um sistema IoT para estacionamento inteligente com um algoritmo anti-colisão, para monitorar as trajetórias e garantir a integridade física dos veículos. Um estudo de caso foi realizado com robôs móveis para simular o comportamento dos veículos. O protocolo CoAP (*Constrained Application Protocol*) [2] permite que os robôs atuem como clientes, trocando mensagens curtas e sobre UDP, garantindo que as instruções do algoritmo consigam prevenir impactos de maneira ágil. Um *gateway* agregador é responsável por realizar a interoperabilidade entre as mensagens CoAP e MQTT. O protocolo *Message Queuing Telemetry Transport* (MQTT) [3] foi integrado para atuar na transmissão de dados, e no envio de alertas para a interface web e notificações por e-mail.

II. TRABALHOS RELACIONADOS

Soluções recentes de estacionamento inteligente exploram diversas tecnologias IoT para o monitoramento de vagas. A detecção baseada em sensores ultrassônicos é amplamente adotada na literatura, podendo ser integrada a placas *Raspberry Pi* via MQTT [4], ou a microcontroladores *Arduino* equipados

com módulos Wi-Fi para a transmissão direta de dados à nuvem [5], [6]. Para cenários que demandam conectividade de longo alcance, destaca-se o uso do protocolo LoRaWAN em conjunto com a plataforma *Arduino* [7]. Ampliando o escopo de sensoriamento e comunicação, o trabalho de [8] emprega sensores infravermelhos (IR) conectados a microcontroladores ESP8266, combinando as redes Wi-Fi e Bluetooth para o gerenciamento da ocupação das vagas via aplicativo celular.

Diante da literatura, justifica-se uma arquitetura híbrida: enquanto o CoAP confere eficiência de rede, o MQTT garante baixa latência [9]. Diferente das abordagens limitadas à detecção de ocupação, esta proposta inova ao integrar o controle de movimentação autônoma via CoAP para comunicação M2M (*Machine-To-Machine*) à supervisão assíncrona via MQTT.

III. ARQUITETURA DO SISTEMA

A arquitetura de comunicação do sistema na Figura 1, detalha o fluxo de dados entre os componentes. Na arquitetura, um *gateway* CoAP agregador, recebe as mensagens do tipo *CoAP Confirmable* (CON), o receptor confirma que recebeu a mensagem dos microcontroladores nos robôs móveis, e interage com o módulo gerenciador. O gerenciador toma as decisões para evitar colisões através do algoritmo de alocação automática dos robôs para as vagas livres, e do algoritmo anti-colisão para segurança de navegação entre os robôs. As decisões para a anti-colisão e ocupação de vagas são retornadas aos robôs móveis via mensagens CoAP CON. O *gateway* traduz as mensagens CoAP para MQTT, e publica dados e alertas em tempo real com o protocolo MQTT. Considera-se estacionamentos cobertos ou subterrâneos (*indoor*), onde o sinal de GPS é frequentemente degradado ou inexistente.

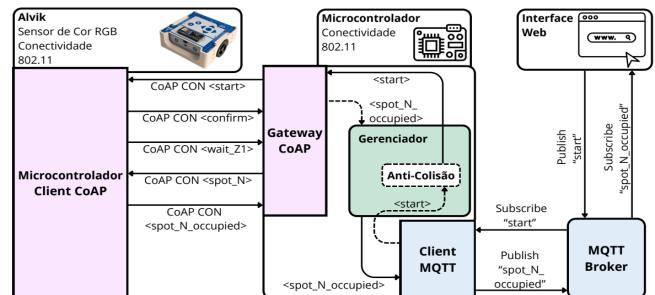


Fig. 1. Arquitetura de Comunicação dos Componentes do Sistema.

Na metodologia de desenvolvimento, o ambiente do estacionamento foi discretizado como uma malha (*grid*), conforme apresentado na Figura 2. Cada célula do cenário corresponde a uma coordenada (x, y) no sistema de navegação. Os sensores ópticos de cor nos robôs (*Arduino Alvik*) [10] identificam

a marcação visual do solo que foi planejada para viabilizar a leitura pelos sensores. Por exemplo, conforme o mapa, o vermelho representa o fim de uma trajetória.

	Alvik 1 (0, 0)	Alvik 2 (0, 1)	Vaga 1 (0, 2)	Vaga 2 (0, 3)	Vaga 3 (0, 4)	Vaga 4 (0, 5)
	(1, 0)	(1, 1)	(1, 2)	(1, 3)	(1, 4)	(1, 5)
	(2, 0)	(2, 1)	(2, 2)	(2, 3)	(2, 4)	(2, 5)

← 14 cm →

↑ 14 cm ↑

Fig. 2. Mapa do Sistema de Estacionamento.

A lógica do algoritmo anti-colisão (Algoritmo 1) foi separada por zonas: A Zona 1 compreende todo o espaço móvel fora do estacionamento: (1–2, 0–1) e a Zona 2, o espaço móvel dentro do estacionamento (1–2, 2–5).

Dentro das zonas de movimentação, o fluxo é unidirecional por faixa: a linha $y = 1$ é utilizada para o trajeto de ida (alocação), e a linha $y = 2$ para o trajeto de volta, garantindo um ciclo de navegação organizado.

Algorithm 1 Anti-Colisão dos Robôs.

Require: ip_{alvik} (Endereço IP do robô), cmd (Comando recebido do Supervisor via MQTT)

Ensure: Atualização do estado do robô $Status(alvik)$ e coordenação de acesso seguro às zonas

Definições Iniciais:
 Z_1, Z_2 : Zonas críticas de trânsito (Rua e Estacionamento, respectivamente)
 $Livre(Z)$: Função booleana que retorna **verdadeiro** se a zona Z estiver vazia
 $vaga_r$: Vaga reservada para robô aguardando em Z_1
 $LiberarVaga(ip_{alvik})$: Função que mapeia a vaga ocupada pelo robô e atualiza seu estado para $Status(vaga) \leftarrow LIVRE$

```

1: function REQUISITARACessoESTACIONAMENTO( $ip_{alvik}$ )    ▷ Solicitar Zona 2
2:    $vaga \leftarrow ENCONTRARVAGALIVRE()$ 
3:    $Status(vaga) \leftarrow RESERVADA$     ▷ Garante exclusão mútua da vaga
4:   if  $Livre(Z_2)$  then
5:      $RESPOSTACOAP(ip_{alvik}, vaga)$ 
6:      $Status(alvik) \leftarrow EM\_ZONA\_2$ 
7:   else
8:      $Status(alvik) \leftarrow AGUARDANDO\_EM\_ZONA\_1$ 
9:      $vaga_r \leftarrow vaga$     ▷ Aloca a vaga reservada na variável
10:  end if
11: end function
12: function REQUISITARSAIDARUA( $ip_{alvik}$ )    ▷ Solicitar Zona 1
13:   if  $Livre(Z_1)$  then
14:      $RESPOSTACOAP(ip_{alvik}, "saída")$ 
15:      $Status(alvik) \leftarrow EM\_ZONA\_1$ 
16:   else
17:      $Status(alvik) \leftarrow AGUARDANDO\_EM\_ZONA\_2$ 
18:   end if
19: end function
20: function PROCESSARCOMANDO( $ip_{alvik}, cmd$ )    ▷ Comandos recebidos via MQTT
21:   if  $cmd == "iniciar"$  then
22:     if  $Livre(Z_1)$  then
23:        $ENVIARREQUISICAOCOAP(ip_{alvik}, "iniciar")$ 
24:        $Status(alvik) \leftarrow EM\_ZONA\_1$ 
25:     else
26:        $Status(alvik) \leftarrow AGUARDANDO\_INICIO$ 
27:     end if
28:   else if  $cmd == "parar"$  then
29:     if  $Livre(Z_2)$  then
30:        $ENVIARREQUISICAOCOAP(ip_{alvik}, "parar")$ 
31:        $Status(alvik) \leftarrow EM\_ZONA\_2$ 
32:        $LIBERARVAGA(ip_{alvik})$ 
33:     else
34:        $Status(alvik) \leftarrow AGUARDANDO\_PARADA$ 
35:     end if
36:   end if
37: end function

```

Antes de enviar um comando de movimento via CoAP, o gateway realiza uma consulta sob o status de ocupação da zona pretendida. No Algoritmo 1, linhas 21 a 27, caso a zona esteja ocupada por outro robô, o sistema retém o veículo atual no estado *aguardando início* ou *aguardando parada* (linha 34). Após a liberação da zona, procedimento definido no Algoritmo 2 (linhas 15 a 18), o comando de movimentação é enviado através de uma mensagem CoAP CON. Essa estratégia de exclusão mútua por zonas previne colisões físicas e garante a integridade dos veículos autônomos no cenário. No Algoritmo 2 (linhas 21 a 22), trata-se o caso de bloqueio mútuo, quando um robô quer entrar no estacionamento e o outro sair. Ambos são liberados, e o algoritmo anti-colisão, evita que os robôs colidam.

Algorithm 2 Gerenciador de Robôs em Espera.

Require: ip_{alvik} (Endereço IP do robô) e $Status(alvik)$ (Estado do robô) para todos os robôs do sistema

Ensure: Transição segura de estados pendentes e evasão de bloqueios críticos

Definições Iniciais:

Z_1, Z_2 : Zonas críticas de trânsito

$Livre(Z)$: Função booleana que retorna **verdadeiro** se a zona Z estiver vazia

$vaga_r$: Vaga reservada para robô aguardando em Z_1

```

1: procedure ATUALIZARFILADEESPERA()    ▷ Loop para atualizar e avançar os robôs retidos
2:   if  $Livre(Z_1)$  then
3:     if Robô retido em  $AGUARDANDO\_EM\_ZONA\_2$  then
4:        $Status(alvik) \leftarrow EM\_ZONA\_1$ 
5:        $ENVIARREQUISICAOCOAP(ip_{alvik}, "atualizar", "saída")$ 
6:     else if  $Status(alvik)$  em  $AGUARDANDO\_INICIO$  then
7:        $Status(alvik) \leftarrow EM\_ZONA\_1$ 
8:        $ENVIARREQUISICAOCOAP(ip_{alvik}, "comando", "iniciar")$ 
9:     end if
10:  end if
11:  if  $Livre(Z_2)$  then
12:    if Robô retido em  $AGUARDANDO\_EM\_ZONA\_1$  then
13:       $Status(alvik) \leftarrow EM\_ZONA\_2$ 
14:       $ENVIARREQUISICAOCOAP(ip_{alvik}, "atualizar", vaga_r)$ 
15:    else if Robô retido em  $AGUARDANDO\_PARADA$  then
16:       $Status(alvik) \leftarrow EM\_ZONA\_2$ 
17:       $ENVIARREQUISICAOCOAP(ip_{alvik}, "comando", "parar")$ 
18:       $LIBERARVAGAATUAL(alvik)$ 
19:    end if
20:  end if
21:  if (Robô retido em  $AGUARDANDO\_EM\_ZONA\_1$ ) and (Robô retido em  $AGUARDANDO\_EM\_ZONA\_2$ ) then
22:     $TROCARZONASSIMULTANEAMENTE(alvik', alvik'')$     ▷ Realiza a manobra de passagem cruzada
23:  end if
24: end procedure

```

IV. METODOLOGIA E RESULTADOS

A efetividade da arquitetura foi avaliada por meio de uma simulação física utilizando dois robôs *Arduino Alvik*, cada um equipado com um microcontrolador ESP32 (WiFi 802.11) atuando como cliente CoAP. Um terceiro ESP32 foi empregado centralmente como gateway agregador, responsável por executar o gerenciador de interoperabilidade entre as mensagens CoAP e MQTT. O cenário foi sinalizado por fitas LED RGB (verde para disponível, amarela para aguardando e vermelha para ocupada). Na implementação do *firmware*, priorizou-se uma rotina não bloqueante para evitar latência excessiva durante a troca de mensagens.

Como resultado, tem-se na Figura 3 a interface do sistema, equipada com um indicador de conectividade MQTT. O gerenciamento das vagas ocorre via menu de seleção, permitindo alternar o estado (aberto/fechado) das posições com atualização em tempo real no painel. A interface desenvolvida atualiza

o status do sistema e dispara notificações por e-mail ao usuário a cada comando de alteração de estado dos robôs (Figura 4).

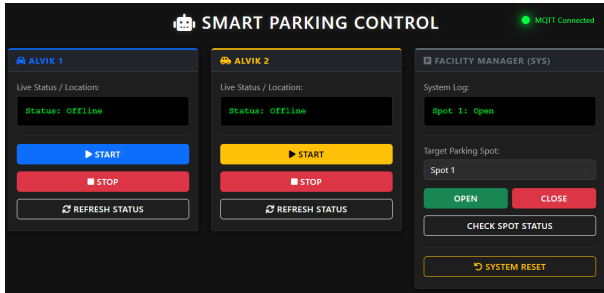


Fig. 3. Interface Web do Sistema.

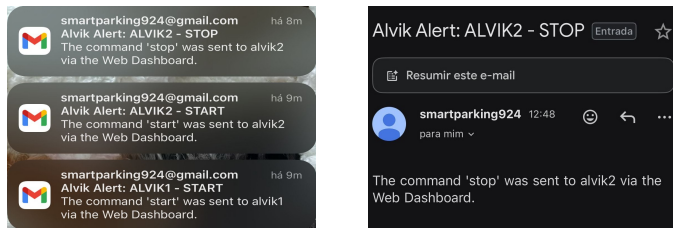


Fig. 4. Notificação de Estado do Alvik.

Na Figura 5, apresenta-se o modelo físico do estacionamento com os Alviks. Inicialmente, todas as vagas estão disponíveis (lado esquerdo). Após o envio dos comandos de inicialização para o Alvik 1 e Alvik 2, observa-se a dinâmica de ocupação (lado direito): o Alvik 1 encontra-se estacionado na primeira vaga, acionando a sinalização visual vermelha (ocupado), enquanto o Alvik 2 realiza a manobra de aproximação para a segunda vaga, cujo status de reserva é indicado pela cor amarela (aguardando). Verifica-se que a abordagem direcionou corretamente os robôs para as vagas. Para reduzir erros quanto à intermitência na leitura dos sensores ópticos sobre as faixas demarcatórias, o que compromete a localização absoluta do robô, o algoritmo de navegação foi refinado para operar em passos discretos de 1 cm com leitura estática. Ao detectar um marco visual (cor branca ou vermelha), o robô executa um deslocamento contínuo de 7 cm (*blind move*) para evitar a leitura duplicada da mesma faixa. A técnica resultou em uma melhora notável na navegação do robô, entretanto, não mitigou totalmente esse desvio.

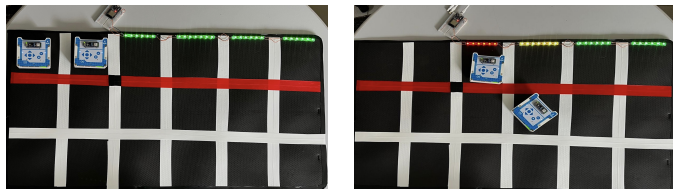


Fig. 5. Modelo físico do Sistema de Estacionamento.

V. CONCLUSÃO

Este trabalho demonstrou uma arquitetura IoT para estacionamentos autônomos baseada na interoperabilidade entre CoAP e MQTT. O algoritmo anti-colisão desenvolvido viabilizou manobras seguras de alocação de vagas por robôs móveis,

dispensando a alta sobrecarga computacional típica de técnicas avançadas de *Machine Learning* [8]. A abordagem provou-se uma solução efetiva para ambientes *indoor*, superando a frequente degradação de sinais de GPS e de conectividades de longo alcance [7].

Diferenciando-se da literatura focada em orientar motoristas humanos por meio de aplicativos [5], [6], o foco do sistema na comunicação M2M via CoAP elimina a necessidade de intervenção humana nas manobras. Por fim, a automação integral foi apoiada por interfaces visuais de supervisão e notificações, garantindo a confiabilidade e a segurança necessárias para a operação do sistema.

Como trabalhos futuros, propõe-se a realização de ensaios quantitativos para aferir métricas de eficiência (latência, tempo de resposta e vazão do *gateway*). No escopo de *software*, vislumbra-se a evolução da arquitetura de *software* embarcada nos robôs *Arduino Alvik*. A incorporação de um *micro-ROS* no microcontrolador dos robôs permitiria integrar os agentes móveis nativamente ao ecossistema *ROS 2 (Robot Operating System)*, facilitando a aplicação de algoritmos de navegação mais sofisticados (como *SLAM* [11]) e padronizando a comunicação de dados em redes de robótica distribuída.

VI. AGRADECIMENTOS

Este trabalho recebeu apoio financeiro da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - CAPES - Brasil (PROAP/AUXPE).

REFERÊNCIAS

- [1] T. N. Pham, M.-F. Tsai, D. B. Nguyen, C.-R. Dow, and D.-J. Deng, "A cloud-based smart-parking system based on internet-of-things technologies," *IEEE Access*, vol. 3, pp. 1581–1591, 2015.
- [2] Z. Shelby, K. Hartke, and C. Bormann, "The Constrained Application Protocol (CoAP)," RFC 7252, Jun. 2014. [Online]. Available: <https://www.rfc-editor.org/info/rfc7252>
- [3] A. Banks, E. Briggs, K. Borgendale, and R. Gupta, "MQTT version 5.0," March 2019. [Online]. Available: <https://docs.oasis-open.org/mqtt/mqtt/v5.0/os/mqtt-v5.0-os.html>
- [4] M. S. Sayeed, H. Abdulrahman, S. F. A. Razak, U. A. Bakar, and S. Yogarayan, "IoT Raspberry Pi based smart parking system with weighted k-nearest neighbours approach," *Civil Engineering Journal*, vol. 9, no. 8, pp. 1991–2011, 2023. [Online]. Available: <https://doi.org/10.28991/CEJ-2023-09-08-012>
- [5] H. Albatati, S. Meer, F. Alharbi, M. Aljohani, and R. Albehise, "Quick park: Integrated sensor-based smart parking system with chatbot for enhanced urban mobility and traffic management," in *2025 2nd International Conference on Advanced Innovations in Smart Cities (ICAISC)*, 2025, pp. 1–6.
- [6] Y. M. Dalal, D. R. K. Raja, and U. Ashwin Kumar, "Eco park: A low-cost and sustainable approach for smart parking systems," in *2023 World Conference on Communication Computing (WCONF)*, 2023, pp. 1–6.
- [7] W. A. Jabbar, L. Y. Tiew, and N. Y. Ali Shah, "Internet of things enabled parking management system using long range wide area network for smart city," *Internet of Things and Cyber-Physical Systems*, vol. 4, pp. 82–98, 2024. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S2667345223000494>
- [8] M. M. Abo-Zahhad and M. Abo-Zahhad, "Smart parking system in smart cities based on the internet of things and machine learning," in *2024 International Telecommunications Conference (ITC-Egypt)*, 2024, pp. 190–195.
- [9] M. Živić, D. Nemec, and Bojović, "Mqtt protocol in iot environment: Comparison with coap and zeromq protocols," in *2023 31st Telecommunications Forum (TELFOR)*, 2023, pp. 1–4.
- [10] Arduino, *Arduino Alvik - Official Documentation*, 2024, acesso em: 14 mar. 2026. [Online]. Available: <https://docs.arduino.cc/hardware/alvik/>
- [11] H. Durrant-Whyte and T. Bailey, "Simultaneous localization and mapping: part i," *IEEE Robotics & Automation Magazine*, vol. 13, no. 2, pp. 99–110, 2006.