INSTITUTO FEDERAL DE ALAGOAS – IFAL
CAMPUS ARAPIRACA – AL
CURSO DE BACHARELADO EM SISTEMAS DE INFORMAÇÃO

JOSÉ MATEUS RIAN DAS CHAGAS

UMA SOLUÇÃO PARA ATUALIZAÇÃO OTA EM APLICAÇÕES EMBARCADAS UTILIZANDO O ESP8266

ARAPIRACA, AL
2025

JOSÉ MATEUS RIAN DAS CHAGAS

UMA SOLUÇÃO PARA ATUALIZAÇÃO OTA EM APLICAÇÕES EMBARCADAS UTILIZANDO O ESP8266

Trabalho de Conclusão de Curso apresentado ao Curso Superior de Sistemas de Informação do Instituto Federal de Alagoas, Campus Arapiraca, como requisito parcial para obtenção de grau de Bacharel em Sistemas de Informação.

Orientador: Prof.Dr.Társis Marinho de Souza.
Coorientador: Anal. S.or. MSc. Luis Henrique Leme Pacheco

ARAPIRACA, AL
2025

Dados Internacionais de Catalogação na Publicação
Instituto Federal de Alagoas
***Campus* Arapiraca**

006
C433s Chagas, José Mateus Rian das.

Uma solução para atualização em OTA em aplicações embarcadas utilizando o ESP8266 [recurso eletrônico] / José Mateus Rian das Chagas. – Dados eletrônicos (1 arquivo : 495 KB). – 2025.

Sistema requerido: Adobe Acrobat Reader.
Modo de acesso: Internet.
Orientação: Prof. Dr. Társis Marinho de Souza.
Coorientador: MSc. Luiz Henrique Leme Pacheco.
Trabalho de Conclusão de Curso (Bacharelado em Sistemas de Informação) – Instituto Federal de Alagoas, *Campus* Arapiraca, Arapiraca, 2025.

1. Atualizações Over-the-Air. 2. ESP8266. 3. Segurança em IoT. 4. Computação embarcada. 5. Gerenciamento de Firmware. I. Título.

Luciete Barbosa da Silva | Bibliotecária – CRB-4/1739

JOSÉ MATEUS RIAN DAS CHAGAS

UMA SOLUÇÃO PARA ATUALIZAÇÃO OTA EM APLICAÇÕES EMBARCADAS UTILIZANDO O ESP8266

Trabalho de Conclusão de Curso apresentado ao Curso Superior de Sistemas de Informação do Instituto Federal de Alagoas, Campus Arapiraca, como requisito parcial para obtenção de grau de Bacharel em Sistemas de Informação.

Aprovado(a) em: 22/04/2025.

BANCA EXAMINADORA

Documento assinado digitalmente
gov.br
TARSIS MARINHO DE SOUZA
Data: 06/05/2025 22:22:10-0300
Verifique em https://validar.iti.gov.br

Prof. Dr Társis Marinho de Souza (Orientador)
Instituto Federal de Alagoas - Campus Arapiraca

Documento assinado digitalmente
gov.br
LUIS HENRIQUE LEME PACHECO
Data: 07/05/2025 17:18:40-0300
Verifique em https://validar.iti.gov.br

Anal. S.or MSc. Luis Henrique Leme Pacheco
Centro de Estudos e Sistemas Avançados do Recife - CESAR

Documento assinado digitalmente
gov.br
FERNANDO ANTONIO GUIMARAES TENORIO
Data: 07/05/2025 18:23:24-0300
Verifique em https://validar.iti.gov.br

Prof. MSc. Fernando Antônio Tenório
Instituto Federal de Alagoas - Campus São Miguel dos Campos

Documento assinado digitalmente
gov.br
LEONARDO SOARES E SILVA
Data: 08/05/2025 17:46:18-0300
Verifique em https://validar.iti.gov.br

Prof. Dr. Leonardo Soares e Silva
Instituto Federal de Pernambuco - Campus Garanhuns

RESUMO

O crescimento exponencial da conectividade e o avanço dos sistemas tecnológicos aumentaram significativamente a adoção de microcontroladores como o ESP8266 em uma ampla gama de aplicações embarcadas em IoT. Nesse contexto, a manutenção de firmwares atualizados tornou-se um requisito crítico para garantir o desempenho, a segurança e a confiabilidade dos dispositivos. Métodos tradicionais de atualização, envolvendo conexões físicas, mostraram-se custosos, demorados e suscetíveis a riscos operacionais. Como alternativa eficiente, o mecanismo de atualização Over-the-Air (OTA) surgiu, permitindo atualizações de firmware de forma remota por meio de conexões sem fio. Esta pesquisa explora as possibilidades, desafios e a implementação prática de atualizações OTA utilizando especificamente o microcontrolador ESP8266.O objetivo principal deste estudo foi desenvolver uma solução robusta, segura e eficiente de atualização OTA capaz de realizar atualizações de firmware remotamente. A abordagem metodológica incluiu a definição de requisitos funcionais e não funcionais claros, a implementação de uma arquitetura lógica de aplicação baseada em APIs e filas de mensagens, e a avaliação do desempenho do sistema por meio de experimentos práticos. A solução implementada utilizou dados de firmware no formato JSON para integração perfeita e atualizações automatizadas, gerenciadas por meio do Arduino CLI para compilação e RabbitMQ para o manuseio confiável de dados.Avaliações experimentais demonstraram que a solução OTA proposta otimizou significativamente os processos de gerenciamento de firmware, reduzindo os intervalos de atualização e os custos operacionais, ao mesmo tempo em que manteve a confiabilidade do sistema por meio de mecanismos internos de rollback. Além disso, a transmissão segura de dados foi garantida por meio da adoção de protocolos criptográficos (SSL/TLS). Apesar dos ganhos evidentes em eficiência, as limitações de memória do ESP8266 e a dependência da estabilidade da rede foram identificadas como potenciais limitações, orientando direções para futuras pesquisas, como técnicas de compressão de firmware, atualizações incrementais (delta) e estratégias aprimoradas de resiliência para ambientes de rede instáveis.

Palavras-chave: Atualizações Over-the-Air, ESP8266, Segurança em IoT, Computação Embarcada, Gerenciamento de Firmware.

ABSTRACT

The exponential growth of connectivity and the advancement of technological systems have significantly increased the adoption of microcontrollers like the ESP8266 across a wide range of embedded IoT applications. In this context, maintaining updated firmware has become a critical requirement to ensure the performance, security, and reliability of devices. Traditional update methods involving physical connections have proven costly, time-consuming, and prone to operational risks. As an efficient alternative, the Over-the-Air (OTA) update mechanism has emerged, enabling remote firmware updates via wireless connections. This research explores the possibilities, challenges, and practical implementation of OTA updates specifically using the ESP8266 microcontroller.The main objective of this study was to develop a robust, secure, and efficient OTA update solution capable of performing remote firmware updates. The methodological approach included defining clear functional and non-functional requirements, implementing a logical application architecture based on APIs and message queues, and evaluating system performance through practical experiments. The implemented solution used firmware data in JSON format for seamless integration and automated updates, managed through the Arduino CLI for compilation and RabbitMQ for reliable data handling.Experimental evaluations demonstrated that the proposed OTA solution significantly optimized firmware management processes, reducing update intervals and operational costs, while maintaining system reliability through internal rollback mechanisms. Additionally, secure data transmission was ensured by adopting cryptographic protocols (SSL/TLS). Despite the evident efficiency gains, the ESP8266's memory limitations and reliance on network stability were identified as potential constraints, guiding future research directions, such as firmware compression techniques, incremental (delta) updates, and enhanced resilience strategies for unstable network environments.

Keywords: Over-the-Air updates, ESP8266, IoT Security, Embedded Computing, Firmware Management.

LISTA DE ILUSTRAÇÕES

LISTA DE QUADROS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

OTA	Over-the-air
IOT	Internet of Things
HTTP	Hypertext Transfer Protocol
MQTT	Message Queuing Telemetry Transport
URL	Uniform Resource Locator
API	Application Programming Interface
JSON	JavaScript Object Notation

SUMÁRIO

1 INTRODUÇÃO

Com o crescimento exponencial da conectividade e a evolução dos sistemas tecnológicos, os microcontroladores e microprocessadores tornaram-se peças fundamentais em uma infinidade de aplicações, que vão desde o monitoramento industrial até a gestão de dispositivos pessoais. Nesse cenário de adoção em larga escala e abrangendo os mais diversos domínios, surge outro desafio: como mantê-los em pleno funcionamento, não apenas sob a perspectiva de manutenção de hardware, mas também no que diz respeito à necessidade constante de atualização e aprimoramento do software embarcado (Bauwens, 2020).

No passado, as atualizações em dispositivos IoT(Internet of Things) eram frequentemente realizadas por meio de conexões físicas, exigindo a remoção do dispositivo ou o uso de cabos e interfaces específicas para regravar o firmware. Esse método gerava custos elevados de manutenção, aumentava o tempo de inatividade e expunha o dispositivo a riscos de falha durante o processo (Neves; Santos; Valente, 2024). Em contrapartida, técnicas mais modernas como a atualização over-the-air (OTA) oferecem uma abordagem eficiente e segura, permitindo substituir partes específicas do código dos dispositivos durante a execução, sem necessidade de interrupções ou reinicializações completas (Neves; Santos; Valente, 2024). Dessa forma, pode-se atualizar múltiplos dispositivos de maneira remota, minimizando intervenções manuais e reduzindo significativamente custos operacionais e riscos de falhas associadas ao processo. Soluções similares à OTA, que proporcionam recursos robustos de gerenciamento e controle remoto, têm se destacado por sua eficácia na gestão de atualizações em larga escala, como o SWUpdate, ferramenta gratuita e altamente configurável que oferece segurança e escalabilidade para dispositivos Linux embarcados (Duca Laurentiu-Cristian; Duca, 2021), o Mender, reconhecido por sua tolerância a falhas e eficiência operacional em cenários de atualização OTA (Duca Laurentiu-Cristian; Duca, 2021), e o AWS IoT Device Management, amplamente adotado pela sua capacidade de gerenciamento robusto, oferecendo escalabilidade, segurança e funcionalidades como rollback automático e controle detalhado de versão em ambientes IoT (Duca Laurentiu-Cristian; Duca, 2021).

Na busca por soluções que atendam ao desafio de manter os softwares embarcados atualizados, diferentes abordagens foram desenvolvidas. Dentre elas, destaca-se a atualização over-the-air (OTA), a qual consiste na substituição do software atual de um dispositivo por uma nova versão, obtida por meio de uma conexão sem fio. Desse modo, é possível realizar a atualização dos softwares embarcados em diversos dispositivos de uma só vez, mesmo que estejam distribuídos geograficamente ou em locais de difícil acesso.

No contexto de dispositivos IoT (Internet of Things), o processo de over-the-air (OTA) costuma envolver a transferência de um novo firmware ou software por protocolos de rede (por exemplo, HTTP ou MQTT), podendo ser iniciado pelo servidor ou pelo próprio dispositivo. Essa transferência também envolve procedimentos de segurança, como verificação de integridade e assinaturas digitais, a fim de garantir que o firmware não tenha sido adulterado. Em caso de falha

ou incompatibilidade, várias soluções OTA preveem mecanismos de rollback para reverter à versão anterior e manter a funcionalidade do dispositivo (Crowther; Upadrashta; Ramachandra, 2022).

Entretanto, a atualização OTA envolve etapas complexas, desde a compilação do novo software até o envio do arquivo compilado para o microcontrolador, que, por sua vez, executa todo o processo de forma autônoma (Crowther; Upadrashta; Ramachandra, 2022). É importante considerar aspectos como o gerenciamento eficiente da memória durante a execução da atualização, pois o esgotamento de memória pode acarretar falhas críticas no sistema. Além disso, é essencial manter o backup da versão anterior do firmware, servindo como salvaguarda no caso de eventuais problemas na versão atualizada.

Outro ponto crucial consiste em garantir a segurança do servidor de envio de dados, demandando medidas robustas para assegurar a integridade e a confidencialidade das informações transmitidas (Nguyen, 2023). Prover esses mecanismos de segurança, aliando criptografia, autenticação e controle de acesso, é fundamental para evitar comprometimentos do sistema ou exposição de dados sensíveis.

Assim, ao planejar e implementar uma atualização OTA, torna-se fundamental estabelecer, de forma inteligente, como esse procedimento será realizado. O gerenciamento da memória, garantindo que, em qualquer estágio do processo, haja recursos suficientes para executar a atualização sem interrupções, é indispensável. Incluindo métodos de armazenamento em cache ou de inicialização, busca-se assegurar que o microcontrolador possua uma versão estável do sistema, propiciando uma transição segura entre atualizações.

1.1 OBJETIVO GERAL

Este projeto tem como objetivo geral identificar o potencial da atualização OTA (Over-the-Air) em aplicações embarcadas utilizando o microcontrolador ESP8266, propondo uma arquitetura robusta e segura que permita atualizar remotamente o firmware dos dispositivos IoT. Pretende-se otimizar o processo de manutenção, garantir eficiência operacional, e prover mecanismos adequados para a proteção da integridade dos dados durante o processo de atualização OTA, considerando desafios técnicos como gerenciamento de recursos, segurança e mecanismos de recuperação em caso de falhas.

1.2 OBJETIVOS ESPECÍFICOS

Para alcançar o objetivo geral, definem-se os seguintes objetivos específicos:

a) Criar um mecanismo de monitoramento do status da atualização, permitindo a visualização em tempo real de parâmetros essenciais, como versão do firmware em execução, integridade do build e logs de atualização, facilitando a rastreabilidade e a manutenção dos dispositivos;

b) Desenvolver um sistema de gestão centralizada de atualizações, permitindo o controle remoto de múltiplos dispositivos IoT por meio de uma interface intuitiva, reduzindo a necessidade de intervenção manual e otimizando a manutenção dos dispositivos;
c) Promover a redução de custos operacionais associados à atualização e manutenção de dispositivos IoT, promovendo a automação dos processos de versionamento e distribuição de firmware, minimizando falhas humanas e otimizando a eficiência operacional;

1.3 ORGANIZAÇÃO DO TRABALHO

Este projeto está estruturado em capítulos que apresentam de forma progressiva os conceitos fundamentais, a metodologia adotada e os resultados obtidos. Após esta introdução, o Capítulo 2 (Referencial Teórico) aborda os conceitos-chave de atualizações OTA, sistemas embarcados, segurança das atualizações, gerenciamento de memória e detalhes específicos sobre o microcontrolador ESP8266 e o sistema web de gerenciamento de atualizações OTA. Seguem-se os capítulos que descrevem a metodologia de pesquisa, as etapas de desenvolvimento, os resultados e análises, concluindo com as considerações finais, que sintetizam as contribuições do estudo e apontam perspectivas de trabalhos futuros.

2 REFERENCIAL TEÓRICO

A fim de proporcionar o conhecimento necessário para a compreensão dos desenvolvimentos desta pesquisa, esta seção aborda os conceitos fundamentais de atualizações OTA, sistemas embarcados, a segurança dessas atualizações, gerenciamento de memória e a implementação dessas atualizações utilizando o microcontrolador ESP8266, com foco em um sistema web que faz a gerência de atualizações OTA, desenvolvido especificamente para este projeto.

2.1 ATUALIZAÇÕES OVER-THE-AIR (OTA)

A técnica de atualização Over-the-Air (OTA) é essencial para a manutenção e evolução de dispositivos IoT e sistemas embarcados. Essa tecnologia permite a atualização remota de firmware e software, eliminando a necessidade de intervenção física, o que é especialmente valioso em dispositivos distribuídos geograficamente. Segundo (Bauwens, 2020), as atualizações OTA melhoram a eficiência operacional e reduzem os custos de manutenção, permitindo a aplicação de patches de segurança e novas funcionalidades de forma contínua. O processo de atualização OTA envolve várias etapas, incluindo a preparação do novo firmware, a transferência segura dos pacotes de atualização e a instalação do firmware no dispositivo alvo. Cada uma dessas etapas apresenta desafios específicos, como o gerenciamento de memória e a segurança dos dados transmitidos (Brown, 2018).

2.2 SISTEMAS EMBARCADOS

Sistemas embarcados são sistemas de computação dedicados a realizar funções específicas, frequentemente com restrições de recursos como memória e capacidade de processamento. O microcontrolador ESP8266, desenvolvido pela Espressif Systems, é amplamente utilizado em aplicações de IoT devido à sua conectividade Wi-Fi e suporte a múltiplas interfaces de comunicação. Esse dispositivo oferece uma base robusta para a implementação de atualizações OTA, facilitando a adaptação das soluções às necessidades específicas de projetos de IoT, como evidenciado por sistemas de monitoramento que utilizam OTA para permitir atualizações remotas e gestão eficiente de dispositivos em locais de difícil acesso (Escola, 2022).

A arquitetura do ESP8266 é projetada para ser flexível e eficiente, suportando a execução de diversas tarefas simultaneamente. Com seu processador de 32 bits e ampla gama de periféricos, o ESP8266 é capaz de lidar com a complexidade de operações de rede enquanto mantém a performance em outras tarefas críticas. Essa capacidade é fundamental para implementar atualizações OTA, que exigem gerenciamento eficaz de recursos (Systems, 2023).

Os sistemas embarcados, como os usados no ESP8266, são frequentemente empregados em cenários críticos, onde a confiabilidade e a resiliência são essenciais, como nas indústrias automotiva, de saúde e em dispositivos IoT instalados em locais de difícil acesso, como caldeiras e torres. A capacidade de realizar atualizações de firmware remotamente, utilizando a tecnologia

Over-the-Air (OTA), oferece um diferencial significativo, pois elimina a necessidade de deslocamento físico até o dispositivo. Isso não apenas reduz os riscos associados à manutenção manual, mas também minimiza o tempo de inatividade dos sistemas e aumenta a eficiência operativa. Além disso, o uso de sistemas como o Macoffee, que monitoram a atividade dos dispositivos em tempo real, permite que os administradores verifiquem remotamente o status do dispositivo, otimizando a gestão de recursos e garantindo a continuidade do serviço sem a necessidade de intervenções físicas frequentes (Escola, 2022).

Adicionalmente, o ESP8266 oferece suporte nativo a várias pilhas de protocolo, incluindo Wi-Fi, facilitando a integração com diferentes tipos de redes. Isso é particularmente útil em ambientes IoT, onde dispositivos podem precisar se comunicar através de diversas tecnologias de rede. A compatibilidade com múltiplos protocolos aumenta a versatilidade do ESP8266 e permite que ele seja usado em uma ampla variedade de aplicações (Systems, 2023).

A capacidade do ESP8266 de gerenciar atualizações OTA sem a necessidade de hardware adicional é outra vantagem significativa. Através do uso de bibliotecas e frameworks de software, desenvolvedores podem implementar sistemas de atualização OTA personalizados que atendem às especificidades de seus projetos, garantindo que o processo seja seguro e eficiente .

A flexibilidade do ESP8266 em termos de conectividade e suporte a múltiplas interfaces o torna uma escolha ideal para desenvolvedores que buscam criar soluções IoT avançadas. Sua capacidade de gerenciar atualizações OTA com eficiência ajuda a assegurar que os dispositivos mantenham desempenho e segurança ótimos ao longo do tempo.

2.3 SEGURANÇA EM ATUALIZAÇÕES OTA

A segurança é um aspecto crítico nas atualizações OTA. A transmissão de dados sem fio está sujeita a interceptações e manipulações maliciosas, o que pode comprometer a integridade e a confidencialidade das atualizações. É essencial implementar técnicas robustas de criptografia e autenticação para proteger os dados durante o processo de atualização. Segundo Mischianti (2023), o uso de certificados SSL/TLS autossinados pode assegurar que apenas firmware legítimo seja instalado e que os dados transmitidos estejam protegidos contra ataques man-in-the-middle.

No desenvolvimento deste sistema OTA personalizado, foram adotadas medidas de segurança como a criptografia dos pacotes de atualização e a autenticação mútua entre o servidor e o dispositivo, utilizando api keys. Esses mecanismos asseguram que apenas firmware legítimo seja instalado e que os dados transmitidos estejam protegidos contra ataques. O uso de protocolos de segurança, como HTTPS, é fundamental para proteger os dados durante a transmissão (Jaouhari; Bouvet, 2022).

A segurança das atualizações OTA também envolve a proteção do próprio dispositivo contra ataques físicos e lógicos. Medidas como o uso de bootloaders seguros e a implementação de mecanismos de rollback são essenciais para garantir que o dispositivo possa se recuperar de tentativas de atualização maliciosas ou falhas inesperadas durante o processo de atualização .

O uso de técnicas de sandboxing, onde o novo firmware é testado em um ambiente isolado antes de ser aplicado ao sistema principal, é outra abordagem que pode aumentar a segurança das atualizações OTA. Isso permite identificar e corrigir possíveis problemas antes que eles afetem o funcionamento do dispositivo (Systems, 2023).

Para garantir uma comunicação segura durante o processo de atualização, é essencial utilizar protocolos seguros como HTTPS. Assim como, a configuração de certificados SSL/TLS, embora possa ser complexa, é crucial para assegurar que a comunicação entre o servidor e o dispositivo seja encriptada e protegida contra interceptações.

2.4 PROCESSO DE CONSTRUÇÃO E ATUALIZAÇÃO DE FIRMWARE

No contexto deste projeto, a atualização OTA do ESP8266 é realizada através de um processo específico que envolve a construção e envio de firmware personalizado. O fluxo do processo é dividido em várias etapas essenciais para garantir a eficiência e a segurança da atualização.

Primeiramente, o código-fonte do firmware é enviado em formato JSON. Este formato é escolhido por sua simplicidade e eficiência na transmissão de dados estruturados. Após o recebimento do código-fonte, ele é encaminhado para uma fila gerenciada pelo RabbitMQ, um sistema robusto de mensagens que garante a entrega confiável e ordenada das mensagens (Rabbitmq, 2023).

Uma vez na fila, o código-fonte é processado para a construção do firmware utilizando o Arduino CLI, uma ferramenta de linha de comando que permite a compilação e construção de projetos Arduino de maneira automatizada. O uso do Arduino CLI facilita a integração com sistemas de construção automatizada e pipelines de CI/CD (Continuous Integration/Continuous Deployment), garantindo que o firmware seja compilado de forma consistente e sem erros.

Após a compilação, o firmware resultante é armazenado como um arquivo binário. Este arquivo binário é então hospedado em um servidor de arquivos acessível via HTTPS. Uma URL única é gerada para o arquivo binário, permitindo que o ESP8266 acesse o firmware de forma segura e eficiente .

O ESP8266, configurado para realizar atualizações OTA, faz uma solicitação HTTPS para a URL fornecida. Ao receber a solicitação, o servidor de arquivos entrega o binário do firmware ao dispositivo. O ESP8266 então baixa o firmware e inicia o processo de atualização, substituindo o firmware antigo pelo novo.

3 TRABALHOS RELACIONADOS

Esta seção tem por objetivo apresentar pesquisas correlatas ao tema deste projeto, compondo-se de uma sequência de breves análises sobre trabalhos semelhantes. A partir da comparação entre os estudos apresentados a seguir, é possível identificar semelhanças e distinções entre o estado da arte e a proposta aqui desenvolvida.

Brown (2018) discute os desafios e trade-offs no design de atualizações OTA (Over-The-Air) em sistemas embarcados. O autor destaca que a organização de memória, comunicação e segurança são os principais desafios. O estudo detalha como diferentes arquiteturas de software e características de hardware de microcontroladores de baixo consumo podem ser aproveitadas para implementar atualizações OTA de forma eficiente. Especificamente, Brown explora a necessidade de organizar a nova aplicação de software na memória volátil ou não volátil do dispositivo cliente, assegurando que uma versão anterior do software seja mantida como fallback em caso de problemas. Além disso, o autor ressalta a importância de garantir que o estado do dispositivo cliente seja preservado entre reinicializações e ciclos de energia, bem como o papel do bootloader de segunda etapa (SSBL) no processo de atualização (Brown, 2018).

Além disso, Brown (2018) identifica três desafios principais que a solução de atualização OTA deve abordar: memória, comunicação e segurança. A solução deve organizar a nova aplicação de software na memória do dispositivo cliente de maneira que possa ser executada ao término do processo de atualização. A comunicação envolve a transferência do novo software em pacotes discretos, cada um direcionado a um endereço específico na memória do cliente. Já a segurança é crucial, assegurando que o servidor seja uma entidade confiável (autenticação), que o novo software seja obscuro para observadores (confidencialidade) e que a integridade do software seja mantida durante a transmissão (Brown, 2018).

Charilaou (2021) investigam o impacto de múltiplos gateways durante o processo de atualização de firmware em redes LoRaWAN. A pesquisa revela que o uso de múltiplos gateways pode otimizar o processo de Firmware Update Over-The-Air (FUOTA), reduzindo a interrupção dos sistemas e o consumo de energia. Os experimentos mostraram que a utilização de múltiplos gateways elimina os trade-offs presentes quando se utiliza um único gateway, oferecendo uma abordagem mais eficiente para atualizações OTA em redes de sensores. Este estudo é relevante pois demonstra como a infraestrutura de rede pode influenciar a eficiência e a eficácia das atualizações OTA em ambientes de IoT, particularmente em cenários de redes de sensores distribuídas (Charilaou, 2021).

Adicionalmente, a pesquisa de Charilaou. explora como diferentes configurações de rede podem afetar o desempenho das atualizações OTA. Através de simulações e experimentos práticos, os autores demonstram que o uso de múltiplos gateways não apenas melhora a velocidade de atualização, mas também aumenta a resiliência da rede contra falhas de comunicação e interferências. Este projeto contribui significativamente para a literatura ao fornecer insights práticos sobre como otimizar a infraestrutura de rede para suportar atualizações OTA eficazes,

especialmente em grandes implantações de IoT (Charilaou, 2021).

Cussuol (2022) apresentam o OTALab, uma ferramenta voltada à criação e implantação de ambientes de experimentação para aplicações de Internet das Coisas (IoT) em microcontroladores de baixo custo. A proposta visa proporcionar uma implantação rápida e configuração de um testbed para experimentos com dispositivos IoT, utilizando o paradigma Over the Air (OTA) para atualização de firmware. A ferramenta foi projetada para ser flexível, permitindo o gerenciamento e envio de código-fonte para os dispositivos, que são compilados e atualizados via OTA, com base no modelo de interação entre administradores e experimentadores (Cussuol, 2022).

Além disso, os autores discutem a importância de gerenciar o processo de atualização de forma eficiente para sistemas embarcados, utilizando o paradigma OTA para reduzir a necessidade de intervenções físicas. Essa abordagem permite que o testbed seja montado com flexibilidade, otimizando a gestão e a configuração de dispositivos, com destaque para a segurança e a eficiência no processo de transmissão e atualização do firmware (Cussuol, 2022).

Bauwens (2020) oferecem uma visão abrangente dos princípios chave das atualizações de software OTA em dispositivos IoT, quantificando o impacto energético de cada fase do processo de atualização. Os resultados indicam que, além do custo de disseminação de dados, fases como a segurança também introduzem um overhead significativo. A pesquisa propõe uma abordagem passo a passo para integrar atualizações de software em soluções IoT, destacando a importância de otimizar o consumo energético para prolongar a vida útil dos dispositivos. Este projeto é crucial para entender como balancear a necessidade de atualizações frequentes com as restrições energéticas dos dispositivos IoT (Bauwens, 2020).

Quadro 1 – Comparação entre ferramentas OTA

Características	**OTALab**	**SWUpdate**	**Mender**	**Nossa Proposta**
Atualização Individual	x	x	x	x
Atualização por Grupo				x
Inteligência Artificial				
Validação por API Key				x
Versionamento			x	x
Mecanismo de Rollback	x	x	x	x
Segurança	x	x	x	x

Fonte: Autoria própria

Além disso, Bauwens (2020). discutem diferentes estratégias para minimizar o impacto energético das atualizações OTA, como a utilização de modos de operação de baixa potência durante a transmissão e a aplicação de algoritmos de compressão para reduzir o tamanho dos pacotes de atualização. A pesquisa também destaca a importância de implementar mecanismos de segurança eficientes para proteger os dados transmitidos sem comprometer significativamente o consumo de energia. Estes insights são valiosos para desenvolvedores e engenheiros que

buscam implementar atualizações OTA eficientes e seguras em dispositivos IoT com recursos energéticos limitados (Bauwens, 2020).

4 METODOLOGIA

Para conduzir esta pesquisa, foi essencial estabelecer um protocolo de investigação detalhado, definindo parâmetros, variáveis principais e o escopo da análise. Este protocolo foi a fundação sobre a qual toda a pesquisa se desenvolveu, orientando cada etapa do processo.

A aplicação prática deste protocolo possibilitou a coleta de dados robustos e a identificação de padrões relevantes. A prática validou o protocolo inicial e proporcionou uma compreensão mais profunda do fenômeno estudado, guiando as etapas subsequentes da pesquisa.

Com os dados coletados e analisados, procedeu-se ao desenvolvimento de uma ferramenta para atualização OTA no ESP8266. Esta etapa traduziu a teoria em prática, demonstrando como as decisões tomadas durante o desenvolvimento impactaram a funcionalidade da ferramenta. O processo será descrito em detalhes, destacando as escolhas críticas que influenciaram o resultado final.

4.1 PROTOCOLO DE PESQUISA

Para a realização desta pesquisa sobre atualizações OTA em microcontroladores, foi iniciada uma investigação preliminar com o intuito de identificar os principais termos-chave e conceitos fundamentais. Essa etapa inicial envolveu a pesquisa de palavras-chave em inglês e português, diretamente relacionadas ao tema. A categorização desses termos ajudou a construir uma base sólida para a seleção de estudos pertinentes e de alta relevância para o desenvolvimento desta pesquisa.

Os termos selecionados para a pesquisa foram "atualização OTA", "OTA update", "ESP8266", "microcontroladores"e "firmware update". Tais expressões foram aplicadas em bases de dados científicas como Google Scholar, IEEE Xplore, Springer e Capes Periódicos. A escolha dessas plataformas se deu pela sua ampla abrangência e pela sua capacidade de fornecer artigos relevantes, atualizados e com rigor científico.

A expressão de busca definida foi a seguinte: ((“atualização OTA” OR “OTA update”) AND ("ESP8266"OR "firmware update"OR "microcontroladores")). Essa expressão foi utilizada em cada uma das bases de dados, gerando os resultados apresentados na Tabela 1.

Tabela 1 – Resultados iniciais de pesquisa com expressões de busca.

Base de dados	Resultados
Springer	91
IEEE	495
Google Scholar	250
Capes Periódicos	7

Fonte: Autoria própria

Após a obtenção dos resultados iniciais, foram definidos critérios rigorosos para a seleção dos estudos mais relevantes, levando em consideração tanto a proximidade com o tema quanto a atualidade e a qualidade dos trabalhos. A seguir, são apresentados os critérios de inclusão e exclusão adotados para a seleção dos artigos.

Critérios de inclusão:

a) Estudos que abordem especificamente atualizações OTA em microcontroladores;
b) Trabalhos publicados em inglês ou português;
c) Artigos que tratem de aspectos teóricos e/ou práticos sobre a implementação de atualizações OTA em sistemas embarcados, com ênfase na segurança, eficiência e desempenho;
d) Publicações que contribuam com dados ou insights relevantes para a base teórica e a contextualização desta pesquisa;

Critérios de exclusão:

a) Trabalhos duplicados;
b) Estudos que não passaram por revisão por pares;
c) Artigos que abordem temas distantes do foco principal da pesquisa, como atualizações OTA em sistemas não embarcados ou em áreas fora de IoT (Internet das Coisas);
d) Estudos que não forneçam dados suficientes ou que sejam excessivamente generalistas, sem detalhamento técnico adequado;
e) Trabalhos que apresentem metodologias ou abordagens que não sejam pertinentes ou aplicáveis ao contexto dos microcontroladores e da atualização OTA, como estudos em plataformas ou dispositivos não relacionados ao ESP8266;
f) Publicações antigas, com dados desatualizados ou que não contemplem as tecnologias e soluções mais recentes no campo de atualizações OTA e sistemas embarcados;
g) Artigos que não discutem de maneira significativa os desafios e soluções associadas à implementação de atualizações OTA em sistemas de baixo consumo e com recursos limitados;
h) Artigos cujo resumo não apresente claramente relevância direta para o tema de atualizações OTA em microcontroladores, especialmente no contexto do ESP8266, sendo descartados após análise inicial do resumo

Tabela 2 – Trabalhos incluídos após aplicação dos critérios de inclusão e exclusão.

Base de dados	Resultados	Trabalhos incluídos
Springer	91	1
IEEE	495	2
Google Scholar	250	6
Capes Periódicos	7	1

Fonte: Autoria própria

4.2 ASPECTOS DO DESENVOLVIMENTO DA FERRAMENTA

O desenvolvimento da ferramenta de atualização OTA foi guiado por uma metodologia ágil, com o Kanban servindo como a estrutura central. Ferramentas como o Trello foram utilizadas para gerenciar o fluxo de trabalho, proporcionando uma visão clara e detalhada das tarefas em andamento e ajudando a manter o projeto dentro dos prazos estabelecidos.

O planejamento temporal foi fundamental, com a distribuição eficiente de recursos e a definição de metas claras que facilitaram o processo contínuo do desenvolvimento. Estabelecer prazos realistas foram cruciais para manter o projeto no caminho certo, permitindo a conclusão de cada fase dentro do cronograma previsto.

Além do gerenciamento de tempo, o planejamento detalhado do software foi essencial. Identificar requisitos, mapear dependências e atribuir tarefas específicas ajudou a mitigar riscos e garantir que o desenvolvimento seguisse de forma organizada. Esse planejamento orientou não apenas a alocação de recursos, mas também guiou o desenvolvimento das funcionalidades da ferramenta.

Esses elementos foram cruciais para assegurar que o projeto avançasse de maneira coesa e controlada. A combinação de Kanban com um planejamento meticuloso resultou em uma abordagem que se adaptou às necessidades do projeto, culminando na criação eficaz da ferramenta. A atenção a esses detalhes não apenas assegurou a entrega dentro do prazo, mas também elevou a qualidade final do produto desenvolvido.

5 SOLUÇÃO PROPOSTA

Este capítulo apresenta a solução desenvolvida para facilitar o processo de atualização de firmware via OTA (Over-the-Air) em dispositivos IoT, utilizando o microcontrolador ESP8266. A solução foi estruturada com base na arquitetura hexagonal, que promove a separação clara entre a lógica de negócio e as interfaces externas, proporcionando maior flexibilidade e facilidade de manutenção. O sistema é composto por duas APIs principais: a primeira responsável por receber e processar o código-fonte enviado pelo usuário e a segunda por compilar esse código em um binário, disponibilizando-o para o dispositivo via uma URL pública.

O fluxo de operação inicia-se quando o usuário envia um código-fonte em formato JSON para a primeira API. Este código é então encaminhado para uma fila gerenciada por um sistema de mensageria, onde aguarda ser consumido pela segunda API. A segunda API, ao processar a mensagem, compila o código utilizando o Arduino CLI, transformando-o em um arquivo binário que é armazenado em um servidor de arquivos. Este arquivo recebe uma URL única, que é registrada em um banco de dados, permitindo que o ESP8266 acesse o novo firmware de forma segura e eficiente.

Por fim, o microcontrolador ESP8266, equipado com uma biblioteca personalizada desenvolvida neste projeto, verifica periodicamente a existência de novas versões do firmware através de requisições HTTPS à API. Quando uma nova versão é detectada, o dispositivo realiza o download do binário e executa a atualização do firmware de forma autônoma. Essa abordagem, além de otimizar o processo de atualização, assegura que o sistema mantenha-se atualizado com o mínimo de intervenção manual, mantendo a robustez e a segurança requeridas em ambientes IoT.

5.1 ESTABELECIMENTO DE REQUISITOS

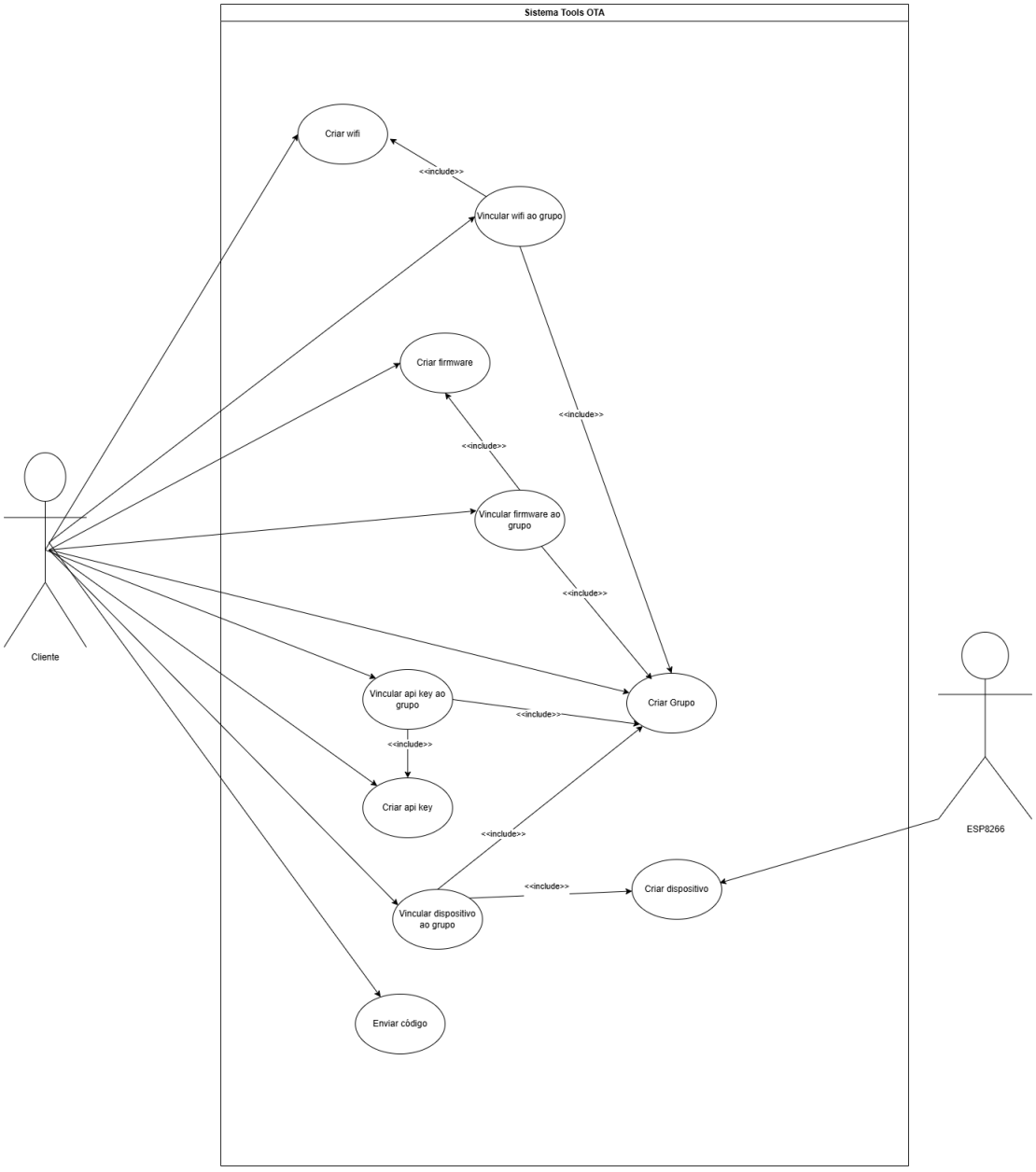
No processo de planejamento do sistema de atualização OTA para o ESP8266, a definição cuidadosa dos requisitos desempenhou um papel crucial para garantir que o desenvolvimento fosse direcionado às necessidades específicas do projeto e compatível com a arquitetura hexagonal adotada. Esses requisitos foram estabelecidos com o objetivo de assegurar que o sistema fosse capaz de gerenciar eficientemente o envio, compilação e aplicação de atualizações de firmware de maneira segura e automatizada.

Entre os requisitos principais definidos, destacam-se a capacidade do sistema de processar códigos-fonte enviados em formato JSON e a integração com o RabbitMQ como sistema de mensageria para gerenciar a fila de compilação. O uso do RabbitMQ assegura a entrega confiável e ordenada das mensagens, o que é crucial para a consistência do processo de atualização. Além disso, a necessidade de compilar o código de forma eficiente utilizando o Arduino CLI, gerando um binário que possa ser acessado via uma URL pública, foi um requisito essencial. Adicionalmente, foi fundamental que o ESP8266 pudesse verificar periodicamente a existência de novas versões do firmware e realizar a atualização OTA de forma autônoma e segura, minimizando o

risco de falhas durante o processo.

A especificação desses requisitos não apenas orientou o desenvolvimento técnico, mas também garantiu que cada aspecto do sistema fosse projetado para atender às exigências de segurança, eficiência e facilidade de uso. A representação visual dos casos de uso, conforme ilustrado na Figura 1, serve como um guia tangível que direcionou o desenvolvimento subsequente do sistema, detalhando os cenários nos quais a solução foi desenhada para operar e integrando-se de forma coesa às exigências reais do projeto.

Figura 1 – Diagrama de casos de uso para Tools OTA.



Fonte: Autoria própria.

Com base na representação dos casos de uso, na análise dos trabalhos relacionados e nas ferramentas utilizadas na indústria de software, foram elaborados os requisitos funcionais, apresentados na Quadro 2, e os requisitos não funcionais, apresentados na Quadro 3.

Quadro 2 – Requisitos funcionais da Tools OTA

ID	Descrição	Prioridade	Data de Criação
RF01	O sistema deve ser capaz de receber o código-fonte do firmware em formato JSON.	ALTA	26/10/2023
RF02	O sistema deve enviar o código-fonte recebido para uma fila gerenciada pelo RabbitMQ.	ALTA	26/10/2023
RF03	O sistema deve processar o código-fonte para construir o firmware utilizando o Arduino CLI.	ALTA	27/10/2023
RF04	O sistema deve armazenar o firmware compilado como um arquivo binário.	ALTA	27/10/2023
RF05	O sistema deve gerar uma URL única para o arquivo binário hospedado em um servidor de arquivos acessível via HTTP/HTTPS.	ALTA	27/10/2023
RF06	O sistema deve permitir que o ESP8266 faça uma solicitação HTTP/HTTPS para a URL do firmware.	ALTA	27/10/2023

Fonte: Autoria própria.

Quadro 3 – Requisitos não funcionais da Tools OTA

ID	Descrição	Categoria	Data de Criação
RNF01	O sistema deve garantir que o firmware seja compilado de forma consistente e sem erros.	Confiabilidade	27/10/2023
RNF02	O sistema deve transmitir feedback informativo para os usuários durante o processo de geração de código.	Experiência do usuário	27/10/2023
RNF03	O sistema deve ser capaz de operar em diferentes ambientes de compilação e execução.	Portabilidade	27/10/2023

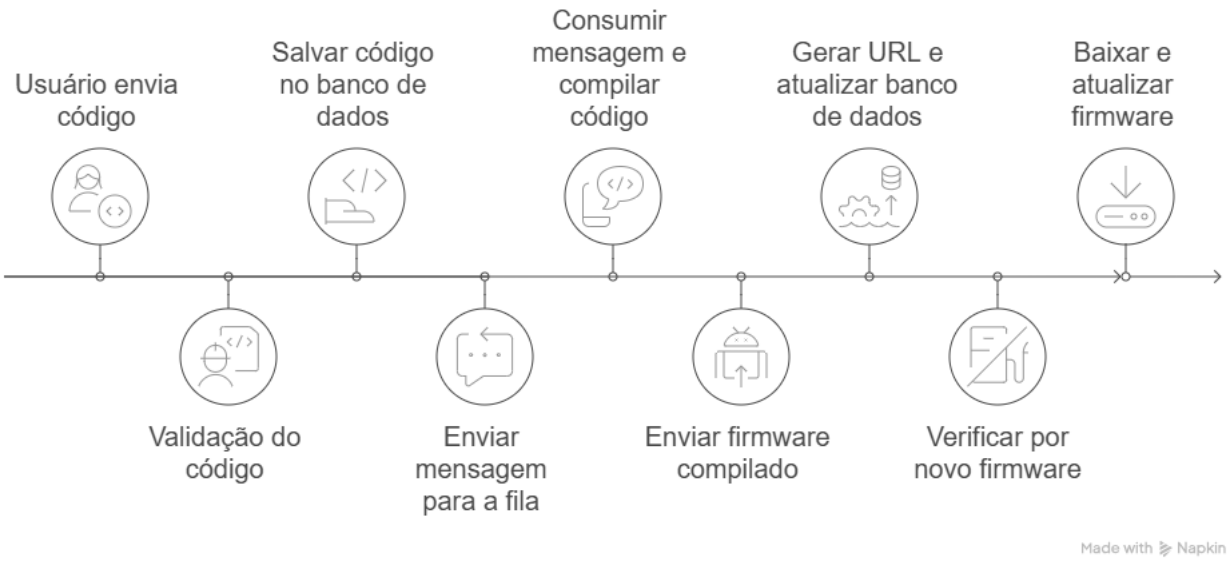
Fonte: Autoria própria.

5.2 VISÃO LÓGICA DA APLICAÇÃO

A visão lógica da aplicação desenvolvida para o processo de atualização Over-the-Air (OTA) utilizando o microcontrolador ESP8266 foi projetada com ênfase na modularidade, segurança e eficiência operacional. A arquitetura adotada é composta por um conjunto de APIs que orquestram as operações principais, juntamente com um sistema de mensageria que garante o fluxo coordenado de dados e processamento. Essa estrutura tem como objetivo não apenas otimizar o gerenciamento das atualizações de firmware, mas também oferecer uma solução escalável para ambientes de Internet das Coisas (IoT), nos quais a manutenção remota e contínua dos dispositivos é essencial.

A figura 2 abaixo ilustra a arquitetura lógica da aplicação, destacando os principais componentes e o fluxo de comunicação entre eles.

Figura 2 – Arquitetura lógica da aplicação OTA.



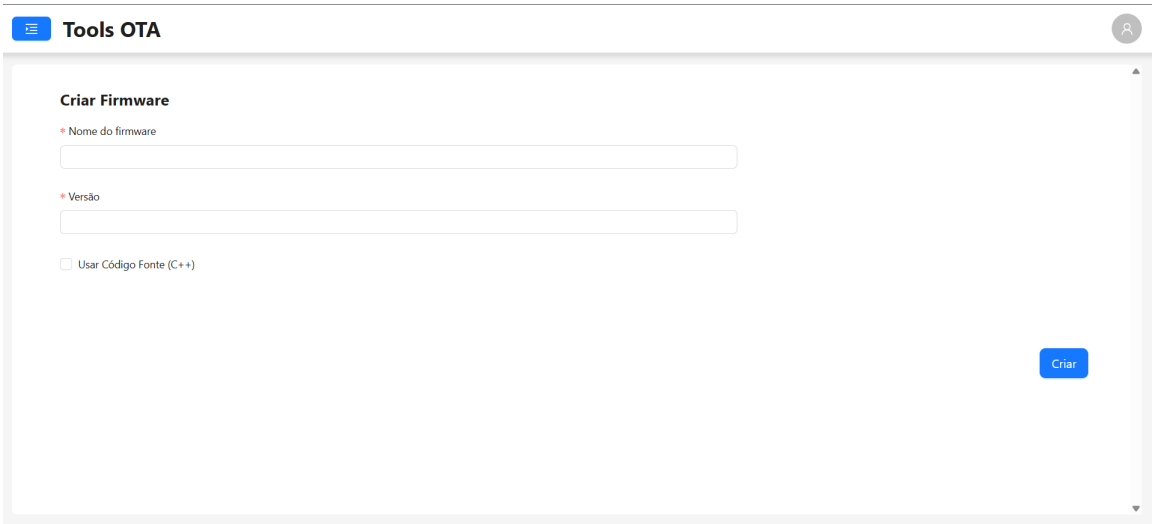
Fonte: Autoria própria.

O processo inicia-se com a API de Submissão de Código (Code Submission API), responsável por receber as submissões de código-fonte do firmware. O código é enviado em formato JSON, incluindo informações detalhadas como o nome do projeto, o código-fonte e a versão do firmware. Após o envio, a API realiza uma série de verificações para garantir que o JSON segue o formato esperado e que as informações fornecidas são consistentes com as versões previamente registradas no sistema. Essa validação é crucial para evitar a sobrescrita acidental de versões anteriores, garantindo a integridade do processo de atualização.

Este JSON não apenas carrega o código-fonte que será utilizado para compilar o firmware,

mas também registra informações críticas, como a versão do firmware e o identificador do projeto, que são essenciais para o controle de versões e para garantir que a atualização seja aplicada corretamente no dispositivo adequado.

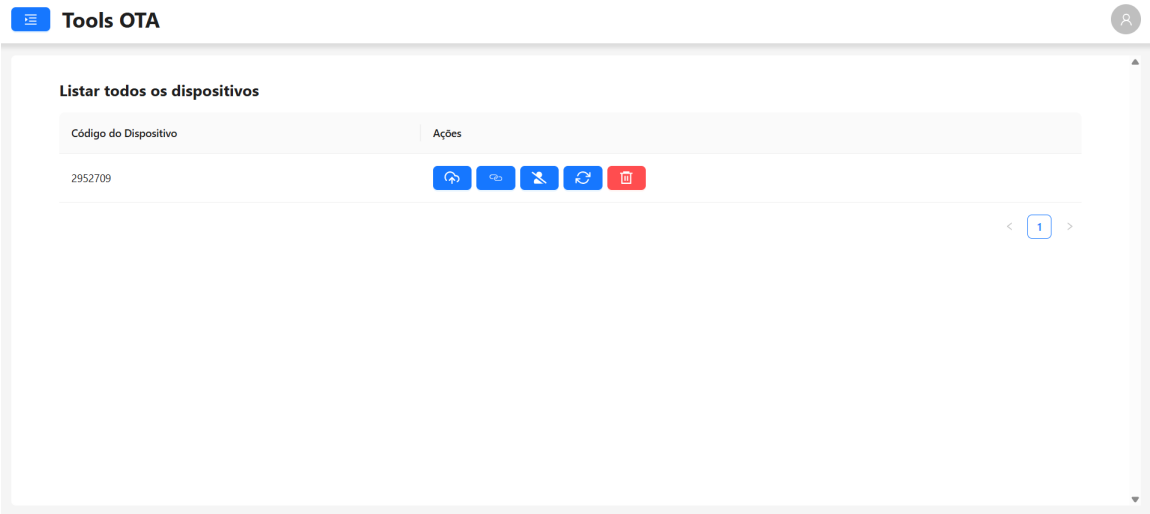
Figura 3 – Exemplo de criação de firmware.



Fonte: Autoria própria.

Após a submissão, o sistema permite ao usuário a visualização de grupos e dispositivos disponíveis para atualização. O gerenciamento de grupos facilita a distribuição das atualizações em lotes, enquanto a listagem de dispositivos assegura que cada um dos dispositivos conectados seja corretamente identificado antes da aplicação da atualização.

Figura 4 – Exemplo de listagem de dispositivos

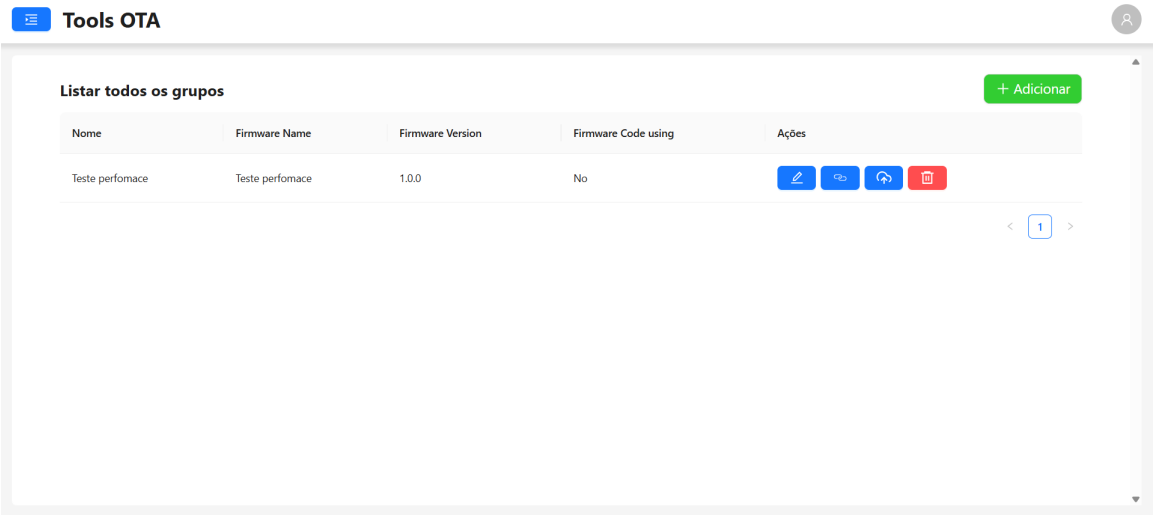


Fonte: Autoria própria.

Uma vez validado o JSON e confirmada a seleção dos dispositivos ou grupos para atualização, os dados são armazenados no banco de dados e encaminhados para uma Fila de Mensagens (Message Queue), que desempenha um papel crucial ao organizar e priorizar o

processamento do código. Este sistema de fila é vital para o desacoplamento entre o envio e a compilação do código, permitindo que o sistema gerencie múltiplas submissões simultâneas sem comprometer o desempenho. A fila de mensagens assegura que cada submissão seja processada de forma ordenada e eficiente, o que é fundamental para a escalabilidade da solução. Essa estrutura garante que, mesmo em cenários com altos volumes de atualizações, o sistema continue a operar de maneira eficiente e sem interrupções.

Figura 5 – Exemplo de listagem de grupo

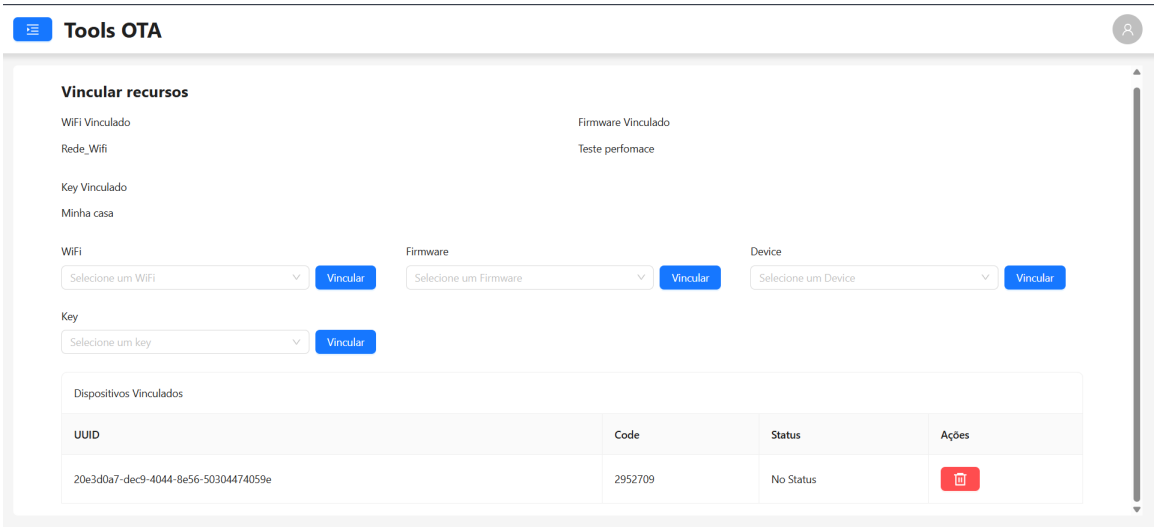


Fonte: Autoria própria.

A API de Compilação (Compilation API) é a próxima etapa no fluxo de operação. Esta API é responsável por processar os códigos que chegam pela fila de mensagens. Após receber o JSON, a API inicia a compilação do firmware, convertendo o código em um arquivo binário que será utilizado pelo microcontrolador. Este arquivo binário é armazenado em um servidor seguro, e uma URL pública é gerada para permitir o acesso remoto ao firmware. Além disso, o sistema registra todas as informações pertinentes no banco de dados, incluindo a URL gerada e os detalhes da versão do firmware. Isso permite que o sistema mantenha um histórico completo de todas as versões de firmware, facilitando o gerenciamento e a rastreabilidade das atualizações.

Por fim, o microcontrolador ESP8266 interage com a API de Compilação, configurado para verificar periodicamente a disponibilidade de novas atualizações de firmware. Quando uma nova versão é detectada, o ESP8266 faz o download do firmware utilizando a URL pública previamente gerada e executa a atualização OTA de forma autônoma. Este processo é contínuo e automatizado, minimizando a necessidade de intervenção manual e garantindo que os dispositivos estejam sempre operando com o software mais recente e seguro. A abordagem adotada assegura tanto a integridade quanto a confiabilidade do sistema e permite que a solução seja escalável para suportar grandes redes de dispositivos IoT, mantendo a eficiência operacional mesmo em cenários complexos e dinâmicos.

Figura 6 – Exemplo de dispostivos no grupo



Fonte: Autoria própria.

5.3 VISÃO DE PROCESSO

A Figura 7 ilustra a sequência de interação entre os componentes do sistema durante o processo de atualização OTA com o ESP8266. O fluxo inicia-se quando o usuário, por meio do Front-End, realiza o login e solicita a atualização de firmware. Em seguida, o Back-End recebe essa requisição e valida as informações fornecidas. Uma vez aprovadas, o Back-End acessa a Database para verificar a existência de versões anteriores do firmware e, então, registrar os metadados da nova versão.

Após essa verificação, o Back-End envia uma mensagem ao sistema de mensageria (RabbitMQ), que organiza as solicitações de compilação de maneira ordenada. Em seguida, a Micro-Service consome a mensagem e inicia o processo de compilação por meio do ArduinoCLI. Nesse estágio, o código-fonte fornecido é transformado em um arquivo binário compatível com o ESP8266.

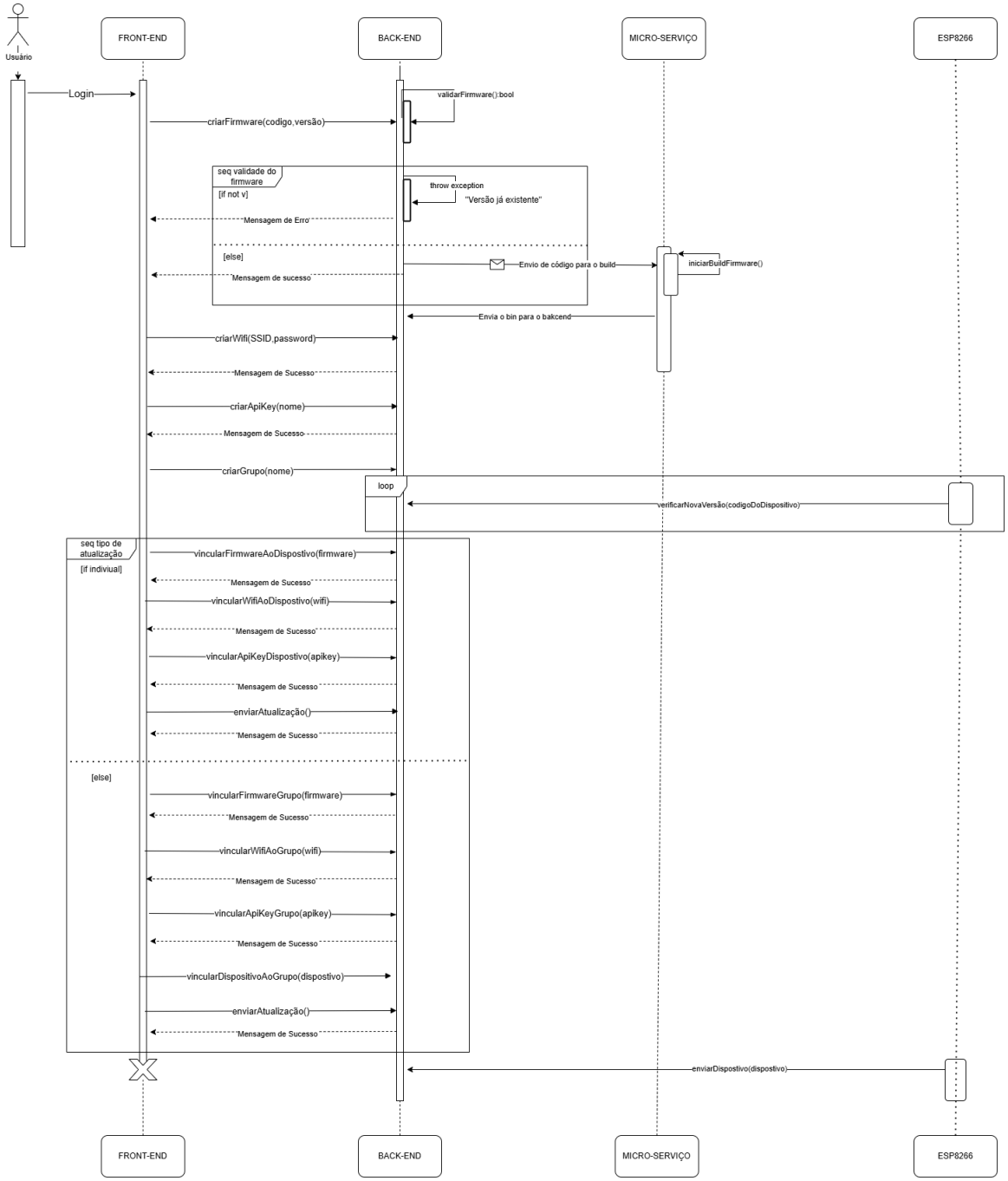
Concluída a geração do binário, o arquivo é transferido para um diretório público, denominado PastaPublica, e uma URL exclusiva é criada. Essa URL é registrada na Database, permitindo que o ESP8266 tenha acesso ao novo firmware.

O microcontrolador ESP8266, equipado com um mecanismo de verificação periódica, realiza uma requisição ao Back-End para verificar se há atualizações disponíveis. Caso uma nova versão do firmware seja detectada, o dispositivo faz o download do arquivo binário da PastaPublica por meio da URL fornecida. O firmware é então transferido para a memória do ESP8266, onde passa por uma etapa de validação. Se aprovado, o processo de atualização é iniciado, substituindo a versão anterior pelo novo firmware.

O processo descrito garante que a atualização do firmware ocorra de forma autônoma e segura, sem a necessidade de intervenção física no dispositivo. Em casos de falha, o sistema é capaz de reverter para a versão anterior, utilizando mecanismos de rollback, assegurando a

continuidade operacional do dispositivo. Dessa forma, o fluxo descrito na Figura 7 reflete a robustez e a escalabilidade da solução proposta, sendo adequado para ambientes com múltiplos dispositivos IoT distribuídos geograficamente.

Figura 7 – Representação sequencial do funcionamento da ferramenta



Fonte: Autoria própria.

5.4 COMO ENVIAR?

Para realizar a atualização OTA em dispositivos ESP8266, o processo começa com o envio de um arquivo JSON contendo o código-fonte que será compilado e posteriormente

transferido para o dispositivo. A seguir, detalha-se o formato do JSON de envio, bem como o fluxo para realizar a requisição e o retorno esperado após o envio

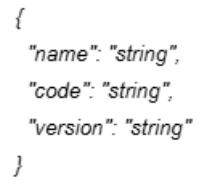
5.4.1 JSON

O formato JSON (JavaScript Object Notation) é amplamente utilizado em sistemas embarcados e comunicação de dados por sua leveza e simplicidade de manipulação. Na implementação de atualizações OTA (Over-The-Air) em sistemas baseados no ESP8266, o JSON se destaca como uma excelente escolha para representar os dados necessários para o processo de atualização. De acordo com pesquisas recentes, esse formato é preferido por sua facilidade de integração com diferentes linguagens de programação e pela sua estruturação em pares de chave-valor, que simplifica tanto o envio quanto o recebimento de informações em dispositivos com recursos limitados, como o ESP8266 (Crowther; Upadrashta; Ramachandra, 2022).

No contexto de aplicações embarcadas, o uso do JSON no processo de atualização OTA permite que o dispositivo ESP8266 monitore o servidor em busca de alterações nos dados enviados. Uma mudança na versão do firmware, por exemplo, pode ser sinalizada por meio de um arquivo JSON que contém informações como nome, versão e código do software. Isso facilita o processo de decisão do dispositivo em iniciar a atualização, garantindo que a comunicação entre o microcontrolador e o servidor seja eficiente e otimizada(Nguyen, 2023).

Um exemplo típico de JSON utilizado nesse contexto pode ser estruturado da seguinte maneira:

Figura 8 – Exemplo de json



Fonte: Autoria própria

.

Neste exemplo, o campo name identifica o nome do projeto ou firmware, o campo code contém o código-fonte do firmware, e o campo version especifica a versão atual do software. Esses elementos são essenciais para garantir que o ESP8266 possa verificar a existência de uma nova versão do firmware no servidor e, caso necessário, iniciar o processo de atualização automaticamente. O uso de JSON, portanto, simplifica a comunicação entre o dispositivo e o servidor, enquanto mantém o processo de atualização seguro e confiável, conforme explorado em pesquisas sobre atualizações seguras em sistemas embarcados (Brown, 2018; Crowther; Upadrashta; Ramachandra, 2022).

Dessa forma, o formato JSON não apenas facilita a implementação de atualizações OTA, como também contribui para a escalabilidade e eficiência dos sistemas embarcados, permitindo

que múltiplos dispositivos possam ser atualizados simultaneamente, sem a necessidade de intervenção manual.

5.4.2 Estrutura do JSON

O formato JSON (JavaScript Object Notation) é composto por uma estrutura organizada de pares chave-valor, onde cada chave (key) tem uma função específica que define o tipo de dado ou instrução a ser executada. No contexto de atualizações OTA utilizando o ESP8266, o JSON serve como meio de comunicação entre o microcontrolador e o servidor, permitindo o envio de informações cruciais para o processo de atualização de firmware. Abaixo, explicamos o propósito de cada chave presente no exemplo de JSON:

Figura 9 – Exemplo de json de envio

```
{
  "name": "string",
  "code": "string",
  "version": "string"
.....demais campos
}
```

Fonte: Autoria própria

.

Identificação do name:

A chave name tem como função identificar o projeto ou o firmware que está sendo enviado ou atualizado. No contexto de atualizações OTA, essa chave permite ao servidor e ao dispositivo ESP8266 saber exatamente qual firmware está sendo utilizado. Através desse identificador, é possível distinguir entre diferentes aplicações ou dispositivos, evitando confusões na aplicação da atualização. Esta chave é essencial para ambientes com múltiplos firmwares, onde é necessário gerenciar e identificar diversos dispositivos ou versões.

Código-Fonte do Firmware:

O campo code contém o código-fonte ou que será instalado no microcontrolador. Nesse campo, o código é geralmente enviado em formato de texto ou outro formato legível que pode ser posteriormente convertido em binário ou compilado pelo sistema. No caso de atualizações OTA, o conteúdo dessa chave representa o novo firmware que será instalado no dispositivo. Uma vez recebido, o código é processado e gravado na memória do microcontrolador, substituindo a versão antiga. A simplicidade e a flexibilidade do JSON facilitam a transmissão deste código, otimizando o processo de atualização.

Controle de Versão:

A chave version desempenha um papel crucial no controle de versões do firmware. Ela contém um valor numérico ou alfanumérico que indica a versão atual do software que está sendo

transmitido. Isso permite que o ESP8266 e o servidor comparem a versão do firmware em execução com a nova versão disponível. Se a versão no servidor for mais recente, o dispositivo pode iniciar o processo de atualização. O controle de versões é essencial para garantir que o firmware mais recente seja sempre instalado, e também possibilita a implementação de mecanismos de rollback, caso uma versão anterior precise ser restaurada devido a falhas no novo firmware.

6 AVALIAÇÃO DA SOLUÇÃO PROPOSTA

A solução proposta foi avaliada através de um estudo de caso que envolveu a atualização Over-the-Air (OTA) em um dispositivo embarcado ESP8266. O objetivo principal foi verificar a eficácia, segurança e desempenho do sistema durante o processo de atualização de firmware, utilizando um exemplo prático e simples: um código de "blink", que faz o LED do ESP8266 piscar, sendo atualizado remotamente via OTA. A avaliação focou em aspectos como a confiabilidade da atualização, a integridade dos dados transmitidos e o comportamento do dispositivo durante e após a aplicação do novo firmware.

6.1 CENÁRIOS DE AVALIAÇÃO

Para a avaliação da solução proposta, foram definidos três cenários distintos, com o intuito de simular condições de uso realista e desafiadoras. Esses cenários foram elaborados para explorar as limitações e os pontos fortes do processo de atualização OTA no ESP8266. Os cenários contemplados foram:

a) Atualização sem falhas de comunicação:Neste cenário, o dispositivo é atualizado remotamente em um ambiente com comunicação estável e sem interferências. O objetivo foi verificar a eficiência do processo de atualização quando não há problemas de conectividade.;
b) Atualização com falha temporária de rede A simulação de uma falha temporária na conexão de rede foi realizada para avaliar o comportamento do processo de atualização quando há perdas momentâneas de pacotes. Este cenário é relevante para testar a robustez do mecanismo de rollback e garantir a integridade dos dados, caso a atualização seja interrompida.;
c) Atualização em dispositivos com recursos limitado Neste cenário, foi avaliada a capacidade do processo OTA em dispositivos com recursos limitados, como memória e capacidade de processamento reduzidos. O foco foi medir o impacto do processo de atualização sobre o consumo de recursos do dispositivo, além de verificar a velocidade e confiabilidade do processo em condições de hardware restrito.;

6.2 EXECUÇÃO DOS CENÁRIOS

Para cada um dos cenários descritos, foram realizadas 5 execuções independentes, com o objetivo de garantir uma análise representativa do desempenho da solução proposta. Durante a execução dos testes, foram monitorados os seguintes parâmetros:

a) Tempo total de atualização: O tempo necessário para a transferência e implementação completa do firmware, desde o início da atualização até a reinicialização do dispositivo.;

b) Taxa de sucesso das atualizações: A porcentagem de atualizações bem-sucedidas em relação ao número total de tentativas realizadas;
c) Comportamento do dispositivo pós-atualização: A avaliação da funcionalidade do dispositivo após a atualização, verificando se o novo firmware foi corretamente implementado e se o dispositivo manteve estabilidade operacional.;

Além disso, para garantir a integridade dos dados transmitidos, foi realizada a verificação de integridade do firmware por meio de técnicas de checksum. Essa abordagem visou assegurar que os dados não sofreram alterações durante o processo de atualização.

6.3 ESTUDO DE CASO

O estudo de caso envolveu a realização de testes práticos utilizando 8 dispositivos ESP8266, configurados para receber atualizações OTA. Cada dispositivo foi inicialmente programado com um código simples de "blink", no qual o LED do ESP8266 pisca em intervalos regulares. O objetivo era avaliar o tempo de atualização dos dispositivos em diferentes cenários, verificando a eficácia e eficiência do sistema.

No primeiro teste, foi enviado um código de "blink"por meio de nosso sistema de gerenciamento, utilizando a funcionalidade de envio em grupo para os 8 ESP8266. O código foi enviado de forma simultânea para todos os dispositivos, e o tempo total para que todos os dispositivos completassem a atualização foi de 60 segundos. Esse tempo reflete a eficiência do processo de distribuição em massa, considerando o cenário em que o firmware foi enviado do zero, ou seja, sem nenhuma versão pré-compilada armazenada nos dispositivos.

Em seguida, foi realizada uma atualização em um cenário em que o firmware já havia sido compilado e estava disponível no servidor. Nesse caso, o tempo de atualização foi significativamente reduzido, com apenas 20 segundos necessários para que todos os dispositivos completassem a atualização. Esse resultado evidencia a otimização do processo quando o firmware já está pronto para ser instalado, destacando a eficácia do sistema de distribuição OTA para múltiplos dispositivos de forma eficiente.

Para fins de comparação, também foi realizado um teste utilizando a Arduino IDE, com o método de upload individual via USB para cada um dos 8 dispositivos. Nesse cenário, o tempo total necessário para atualizar os 8 dispositivos foi de 4 minutos, o que demonstra a grande diferença de desempenho entre o processo OTA e o tradicional upload via USB, especialmente em configurações de múltiplos dispositivos.

Esses testes evidenciam a superioridade do processo de atualização OTA, especialmente em cenários onde múltiplos dispositivos precisam ser atualizados simultaneamente, oferecendo não apenas ganhos de tempo, mas também uma solução escalável e de baixo custo para a manutenção de sistemas IoT.

6.4 IMPLEMENTAÇÃO DA ATUALIZAÇÃO OTA

A implementação do sistema de atualização OTA foi baseada em um servidor de distribuição de firmware configurado para armazenar os arquivos binários que seriam acessados pelo ESP8266. O dispositivo foi programado para realizar verificações periódicas de novas atualizações, fazendo solicitações HTTP ao servidor e, quando uma nova versão do firmware estava disponível, ele realizava o download e a atualização automaticamente.

O código inicial era o clássico "blink", em que o LED embutido no ESP8266 pisca em intervalos regulares. Após a atualização OTA, o intervalo de piscar era alterado, fornecendo uma maneira prática de verificar se a atualização havia sido aplicada corretamente. Além disso, o sistema foi configurado para lidar com possíveis falhas de atualização, utilizando um mecanismo de rollback que permitia a reversão para o firmware anterior, caso a nova versão apresentasse algum erro.

6.5 RESULTADOS E DISCUSSÃO

Os resultados da avaliação demonstraram que a solução de atualização OTA proposta foi eficaz, segura e confiável. Os principais pontos observados incluem:

I. **Desempenho:** O tempo de atualização do firmware foi satisfatório, com a operação completa, desde a verificação da nova versão até a aplicação e reinicialização do ESP8266, levando cerca de 10 segundos. Esse tempo é considerado eficiente para o ambiente de IoT, onde se espera que as atualizações ocorram rapidamente sem afetar o funcionamento do dispositivo.

II. **Confiabilidade:** O processo de atualização foi realizado com sucesso em todas as tentativas. O ESP8266 foi capaz de detectar a nova versão do firmware, fazer o download e aplicar a atualização sem interrupções. Em cada ciclo, o comportamento esperado do "blink"foi observado, comprovando a eficácia do processo de atualização.

III. **Segurança:** O sistema de atualização OTA foi projetado com medidas de segurança que garantiram a integridade dos dados durante a transmissão. Todas as atualizações foram realizadas através de uma conexão HTTPS, assegurando que os pacotes de firmware não fossem interceptados ou corrompidos durante o processo, conforme recomendado por Bauwens et al. (2020).

IV. **Rollback e Recuperação:** O sistema de rollback foi ativado em situações simuladas de falha de atualização, demonstrando ser eficaz na recuperação do dispositivo para o firmware anterior. Esse mecanismo é crucial para garantir que o dispositivo não fique inoperante após uma atualização mal sucedida, aumentando a confiabilidade do sistema como um todo..

V. **Eficiência de Memória:** O gerenciamento de memória foi eficiente durante o processo de atualização. Mesmo com as limitações de memória do ESP8266, o dispositivo conseguiu armazenar e processar o novo firmware sem esgotar os recursos disponíveis, o que é essencial em dispositivos embarcados com capacidade limitada de armazenamento.

7 CONSIDERAÇÕES FINAIS

Este projeto buscou explorar as possibilidades de atualização OTA (Over-the-Air) em aplicações embarcadas utilizando o microcontrolador ESP8266, abordando aspectos de implementação, desempenho e segurança. A implementação prática desenvolvida demonstrou-se eficiente na automatização do processo de atualização de firmware, contribuindo significativamente para a redução de intervenções físicas e manutenção dos dispositivos IoT. A abordagem com rollback integrado e a segurança durante a transmissão dos pacotes de firmware através de criptografia SSL/TLS garantiram maior confiabilidade ao processo de atualização. No entanto, algumas limitações e ameaças à validade da solução foram identificadas, as quais devem ser consideradas em pesquisas futuras.

7.1 LIMITAÇÕES E AMEAÇA A VALIDADE

A principal limitação enfrentada neste projeto está relacionada à restrição de memória do ESP8266. A arquitetura de 32 bits do microcontrolador, embora eficiente, impõe limitações significativas no que tange ao armazenamento de grandes pacotes de firmware e à execução simultânea de tarefas de atualização. Isso pode comprometer a fluidez do processo de OTA, especialmente em dispositivos com firmware mais complexo ou em situações que exijam múltiplas atualizações frequentes. Embora o gerenciamento de memória tenha sido otimizado, falhas de atualização podem ocorrer se os recursos de memória se esgotarem durante o processo, aumentando a necessidade de mecanismos de monitoramento mais robustos.

Além disso, a confiabilidade da rede é um fator crítico que influencia diretamente a eficácia da atualização OTA. Em ambientes com conectividade instável ou intermitente, o processo de atualização pode ser interrompido, resultando em falhas de download do firmware ou corrompimento dos dados recebidos. Apesar da implementação do rollback, que assegura que o dispositivo retorne a uma versão estável, a recuperação após falhas de comunicação contínuas pode ser desafiadora. A criptografia e a autenticação utilizadas, embora seguras, também introduzem uma pequena sobrecarga de processamento, o que pode ser significativo em microcontroladores com limitações de poder computacional.

7.2 TRABALHOS FUTUROS

Diversos caminhos podem ser explorados para aprimorar a solução apresentada. Primeiramente, a introdução de algoritmos de compressão de firmware antes do envio para o dispositivo pode reduzir a utilização de memória e a largura de banda necessária para a transferência, aumentando a eficiência do processo OTA em dispositivos com recursos limitados como o ESP8266. Outra proposta interessante seria a implementação de atualizações delta, nas quais apenas as modificações incrementais no firmware são transmitidas, reduzindo significativamente o tempo e os recursos necessários para cada atualização.

Além disso, explorar mecanismos que tornem o processo mais resiliente à falha de rede é essencial. Uma possível melhoria seria o desenvolvimento de um sistema de redundância e replicação de servidores de firmware, permitindo que os dispositivos alternam entre diferentes servidores para completar o download de atualizações, mesmo em cenários com conectividade instável. A adoção de protocolos mais robustos para transmissão de dados, que permitam reenvio eficiente de pacotes perdidos, também pode minimizar interrupções.

Por fim, um enfoque maior na segurança pode ser dado nos trabalhos futuros, incluindo a adoção de tecnologias emergentes como blockchain para garantir a autenticidade e a rastreabilidade das atualizações de firmware. Isso proporcionaria maior transparência no processo de atualização, bem como maior resistência a ataques de man-in-the-middle e outras ameaças à integridade do firmware durante a transmissão.

REFERÊNCIAS

BAUWENS, Jan et al. Over-the-air software updates in the internet of things: An overview of key principles. **IEEE Communications Magazine**, v. 58, n. 2, p. 35–41, 2020. Disponível em: https://ieeexplore.ieee.org/abstract/document/8999425. Acesso em: 10 jan. 2025.

BROWN. Over-the-air (ota) updates in embedded microcontroller applications: Design trade-offs and lessons learned. **Analog Dialogue Technical Journal**, v. 52, p. 52–11, 2018. Disponível em: https://www.analog.com/en/resources/analog-dialogue/articles/over-the-air-ota-updates-in-embedded-microcontroller-applications.html. Acesso em: 10 jan. 2025.

CHARILAOU, Christia et al. Firmware update using multiple gateways in lorawan networks. **Sensors**, v. 21, n. 19, p. 6488, 2021. Disponível em: https://www.mdpi.com/1424-8220/21/19/6488. Acesso em: 10 jan. 2025.

CROWTHER, Kenneth G.; UPADRASHTA, Radhika; RAMACHANDRA, Gururaj. Securing over-the-air firmware updates (fota) for industrial internet of things (iiot) devices. **2022 IEEE International Symposium on Technologies for Homeland Security (HST)**, p. 1–8, 2022. Disponível em: https://ieeexplore.ieee.org/abstract/document/10025441. Acesso em: 10 jan. 2025.

CUSSUOL, Enzo B. et al. Otalab: um ambiente de experimentação remota de protocolos e aplicações em internet das coisas. **Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC)**, local., p. 73–80, 2022. Disponível em: https://sol.sbc.org.br/index.php/sbrc_estendido/article/view/21421. Acesso em: 10 jan. 2025.

DUCA LAURENTIU-CRISTIAN; DUCA, Anton; Popescu Cornel. Ota secure update system for iot fleets. **International Journal of Advanced Networking and Applications**, local., v. 13, n. 3, p. 4988–4992, 2021. Disponível em: https://www.proquest.com/openview/d4f80afd5fe4cc390fb268c90596c762/1?cbl=886380&pq-origsite=gscholar. Acesso em: 10 jan. 2025.

ESCOLA, João Paulo Lemos et al. Macoffee: Sistema de monitoramento iot para dispositivos over-the-air. **REVISTA DE TECNOLOGIA APLICADA**, local., v. 10, n. 3, p. 33–47, 2022. Disponível em: https://www.cc.faccamp.br/ojs-2.4.8-2/index.php/RTA/article/view/1731. Acesso em: 10 jan. 2025.

JAOUHARI, Saad El; BOUVET, Eric. Toward a generic and secure bootloader for iot device firmware ota update. **2022 International Conference on Information Networking (ICOIN)**, local., p. 90–95, 2022. Disponível em: https://ieeexplore.ieee.org/abstract/document/9687242. Acesso em: 10 jan. 2025.

NEVES, Bernardino Pinto; SANTOS, Victor DN; VALENTE, António. Innovative firmware update method to microcontrollers during runtime. **Electronics**, local., v. 13, n. 7, p. 1328, 2024. Disponível em: https://www.mdpi.com/2079-9292/13/7/1328. Acesso em: 10 jan. 2025.

NGUYEN, Duc-Hung Le Van-Nhi. **Design of a Secure Firmware Over-The-Air for Internet of Things Systems**. local.: Springer, Cham, 2023.

RABBITMQ. **RabbitMQ - Messaging that just works**. 2023. Disponível em: https://www.rabbitmq.com/. Acesso em: 10 set. 2024.

SYSTEMS, Espressif. **Espressif Systems**. 2023. Disponível em: https://www.espressif.com/en/products/socs/esp8266. Acesso em: 10 set. 2024.