# OpenAPFinder: Solução ZeroConf para Conexão de Dispositivos ESP-8266 em Redes Wi-Fi

**Jonathan B. Thiengo**[1], **Raphael Abreu**[1,2]

[1]Unilasalle-RJ
Niterói - RJ, 24240-030

jonathan.thiengo, prof.raphael.abreu{@soulasalle.com.br}

[2]Universidade Federal Fluminense - Laboratório Midiacom
Niterói - RJ, 24210-240

***Abstract.*** *IoT devices, including smart bulbs, motion sensors, and health monitoring systems, are already present in many homes and promise to enhance our daily lives. However, for individuals without technology experience, connecting these devices to the home network can be a challenge. Many of these devices require internal configuration or the use of external devices, such as a phone, for setup. This article introduces OpenAPFinder, a system designed to find and connect ESP8266 devices in home networks using Wi-Fi exclusively. OpenAPFinder allows the SSID and Password of a network to be securely transmitted to new ESP8266 clients. This approach reduces the need for technical knowledge when adding new devices to the network. Experimental results show that the system is capable of integrating new devices into a Wi-Fi network. Future work includes improving temporal efficiency during the recognition, user validation, and data transmission stages.*

***Resumo.*** *Dispositivos IoT, incluindo lâmpadas inteligentes, sensores de movimento e sistemas de monitoramento de saúde, já estão presentes em muitos lares e prometem aprimorar nosso cotidiano. No entanto, para indivíduos sem experiência com tecnologia, a conexão desses dispositivos à rede doméstica pode ser um desafio. Muitos desses dispositivos exigem configuração interna ou o uso de dispositivos externos, como um telefone, para a configuração. Este artigo apresenta o OpenAPFinder, um sistema projetado para encontrar e conectar dispositivos ESP8266 em redes domésticas usando exclusivamente o Wi-Fi. O OpenAPFinder permite transmitir o SSID e Senha de uma rede de formar segura para novos clientes ESP8266. Essa abordagem reduz a necessidade de conhecimento técnico ao adicionar novos dispositivos à rede. Os resultados experimentais mostram que o sistema é capaz de integrar novos dispositivos em uma rede Wi-fi. Futuros trabalhos incluem aperfeiçoar a eficiência temporal durante as etapas de reconhecimento, validação do usuário e transmissão de dados.*

## 1. Introdução

A Internet das Coisas (IoT) promete desempenhar um papel crucial na melhoria de nossas vidas diárias, oferecendo conveniência, eficiência e automação. Contudo, para que

esses dispositivos IoT possam funcionar adequadamente, eles precisam estabelecer conexões em uma rede doméstica. Isso exige uma troca de informações, normalmente o SSID e a senha de uma rede. Para ilustrar esse desafio, este artigo motiva-se na necessidade de sistema IoT para atendimento domiciliar a descobrir, conectar e adicionar novos dispositivos na rede automaticamente, sem a intervenção humana. Na assistência domiciliar, sensores residenciais podem detectar padrões de movimento e comportamento, identificando potenciais situações de risco, como quedas e alterações comportamentais [Oliveira et al. 2020]. Um exemplo de sistema é o SmartCareIoT que utiliza sensores IoT para análise e intervenção imediata, transmitindo dados automaticamente a familiares, cuidadores ou profissionais de saúde em emergências [Oliveira et al. 2022]. O uso de Raspberry Pi e ESP8266 torna o monitoramento acessível e eficiente, ampliando a participação na pesquisa e facilitando implantações em larga escala.

Embora muitos sistemas tais como o SmartCareIoT permitam o monitoramento automático e distribuído, a instalação, configuração e manutenção da rede de sensores e plataforma de comunicação são frequentemente demoradas. Como consequência, a configuração manual de pontos de acesso Wi-Fi em dispositivos IoT requer conhecimentos técnicos em TI. Este artigo propõe uma metodologia de ZeroConf, denominada OpenAPFinder, projetada para conectar dispositivos em redes domésticas usando exclusivamente Wi-Fi. A inovação reside no uso do protocolo ZeroConf apenas com Wi-Fi para adição e comunicação entre dispositivos ESP8266 e Raspberry Pi em redes Wi-Fi. Projetado para ambientes com várias novas conexões, o método inclui verificações para evitar conexões não autorizadas e simplificar a adição de dispositivos em redes locais, minimizando a necessidade de configuração manual pelos usuários. Optar por não utilizar dispositivos intermediários complementa essa abordagem, simplificando ainda mais a interação entre os dispositivos. Isso não apenas reduz a complexidade e evita potenciais pontos de falha, mas também promove uma experiência mais direta e eficiente para os usuários. Eliminar intermediários reduz a dependência de múltiplos dispositivos e aplicativos, simplificando a configuração e operação. Essa abordagem pode ser integrada em uma arquitetura para implementação em hosts e clientes, permitindo conexões sem a necessidade de Bluetooth e NFC, facilitando ainda mais a interação entre dispositivos.

O resto do artigo é organizado da seguinte forma. A Seção 2 revisa a literatura existente e destaca lacunas. A Seção 3 descreve a estrutura teórica adotada, enquanto Seção 4 detalha como foi aplicada na prática. A seção 5 apresenta resultados com análise crítica. Por fim, a Seção 6 resume descobertas e sugere futuras pesquisas.

## 2. Trabalhos relacionados

Imagine um ecossistema em que os dispositivos possam se autoconfigurar e se comunicar sem a intervenção manual de um usuário. Isso é precisamente o que propostas de ZeroConf [Steinberg and Cheshire 2005], tais como Bonjour[Lee et al. 2007] e mDNS [Siddiqui et al. 2012] buscam proporcionar. Essas tecnologias permitem que os dispositivos conectados a uma rede local se identifiquem e se comuniquem automaticamente, sem a necessidade de configuração manual. Enquanto encontrar dispositivos na rede pode ser relativamente simples, a adição desses dispositivos à rede Wi-Fi ainda apresenta obstáculos significativos. Outra abordagem para comunicação IoT se chama Wi-Fi Direct [Khan et al. 2017], um método que permite dispositivos Wi-Fi se conectarem diretamente entre si, sem a necessidade de um ponto de acesso intermediário. É importante notar que

o Wi-FI Direct por si só não permite que um novo dispositivo se conecte a uma rede, invés disso ele cria uma nova rede local temporária que será usada por 2 dispositivos para se comunicarem.

Existem várias soluções proprietárias que utilizam diferentes abordagens para facilitar a conexão de dispositivos IoT a uma rede Wi-Fi. Algumas dessas soluções aproveitam os smartphones como intermediários, permitindo que eles transmitam as credenciais de conexão (como o SSID e a senha) para os dispositivos IoT por meio de tecnologias como Bluetooth [Ikasamo ] ou NFC [Wagner 2008]. Essas abordagens diversificadas refletem a busca contínua por métodos mais simples e eficientes de integração de dispositivos IoT em redes Wi-Fi.

Neste contexto, com foco no cuidado e monitoramento de pacientes fora do ambiente hospitalar, [Wagner 2008] propôs um projeto baseado em ZeroConf, eliminando a maior parte da configuração técnica na residência do paciente. A solução técnica adotada utiliza a tecnologia NFC habilitada em celulares para resolver problemas relacionados à configuração e instalação, oferecendo uma interface de usuário adicional a cada sensor e paciente. No entanto, alguns problemas surgem com essa abordagem, pois os pacientes são obrigados a usar tags na forma de pulseiras, braceletes ou cartões de identificação.

[Dijkstra et al. 2006] exploram a aplicação da tecnologia ZeroConf para endereçamento IP em redes ópticas, focando em configuração automática com ênfase em endereços IP link-local e descoberta de serviços DNS-SD (Service Discovery) usando multicast DNS (mDNS). Os autores sugerem que as tecnologias são adequadas para redes ópticas ad hoc, mas enfrentam limitações em ambientes complexos devido à dependência do Ethernet e restrições do protocolo IPv4 link-local. No trabalho de [Dijkstra et al. 2006], o mDNS é destacado por simplificar a descoberta dinâmica de serviços e promover a interação entre dispositivos em redes que priorizam a simplicidade e eficiência. No entanto, surge uma limitação quando um dispositivo está desconectado da rede, o que o impede de participar do processo de descoberta mDNS.

Uma solução para conectar novos dispositivos é usar a biblioteca Arduino AutoConnect for ESP8266. Desenvolvida por [Ikasamo ], ela simplifica a implementação da interface web que constitui a WLAN para conexão Wi-Fi do ESP8266. Com essa biblioteca, é possível estabelecer uma conexão com o ESP8266 usando o modo SoftAP, criando um ponto de acesso em tempo de execução através da interface web, sem precisar incluir diretamente no código o SSID e senha. Ao tentar conectar-se ao ESP8266, a tela de controle do AutoConnect automaticamente exibirá um menu que facilita ao usuário a gestão de novas conexões. No entanto, é importante notar que esta biblioteca não permite que um ESP8266 se conecte automaticamente ao ponto de acesso, exigindo o uso de um smartphone, o que limita sua adoção.

O conceito de ZeroConf para a integração de novos dispositivos em redes, apresentado neste artigo, embora tenha suas vantagens, também carrega consigo desafios e possíveis complicações. Um desafio é a ocorrência de conflitos de endereços IP, pois os dispositivos podem atribuir endereços automaticamente, levando a conflitos se dois dispositivos tentarem usar o mesmo endereço IP. Em relação à segurança, a confiança mútua entre dispositivos na descoberta e comunicação via ZeroConf pode aumentar a vulnerabilidade da rede a invasões, pois os atacantes podem se passar por dispositivos legítimos.

## 3. Proposta

Esta seção aborda a resolução dos desafios na conexão automática entre dispositivos em redes Wi-Fi. Para ilustrar o sistema OpenAPFinder, consideramos o seguinte cenário: dois dispositivos, A e B, e duas redes disponíveis, R1 e R2. Enquanto o dispositivo A está conectado à rede R1, o dispositivo B, recém-ligado, deve se unir automaticamente a essa rede. A questão principal é: como o dispositivo A pode transferir os dados necessários para permitir que B se conecte à R1 de forma automatizada, sem usar dispositivos externos (e.g., celular). Isso traz desafios na transmissão eficiente de dados entre dispositivos A e B e na garantia de uma transição suave de A para a rede desejada R1. Além disso, a solução deve considerar a integração e coexistência de dispositivos em ambas as redes R1 e R2, para evitar que o dispositivo B se conecte a R2. Para resolver esse problema, o sistema OpenAPFinder utiliza uma implementação de cliente-servidor compartilhado entre os dispositivos A e B. Isso permite que os dispositivos já conectados na rede busquem por novos entrantes, e um novo dispositivo recém-ligado possa agir como um servidor para se comunicar com eles.

A Figura 1 mostra o processo do sistema OpenApFinder. Ao iniciar o ESP8266, o dispositivo busca em arquivos de configuração autorizados o SSID e senha da rede local. Em seguida, tenta conectar-se Wi-Fi com base nessas informações. Após três tentativas malsucedidas, o ESP8266 entra em modo Access Point (AP), no qual ele gera um SSID aleatório e aguarda a conexão de outro dispositivo, como um Raspberry Pi.Quando o Raspberry Pi é iniciado, ele procura a rede gerada pelo ESP8266 e solicita a conexão e aguarda a validação do usuário. Após a conexão ser estabelecida com sucesso, o Raspberry Pi recebe os dados essenciais, como o SSID e a senha da rede Wi-Fi do usuário, e encerra a conexão, adicionando o SSID do ESP8266 a uma *blacklist* de dispositivos.

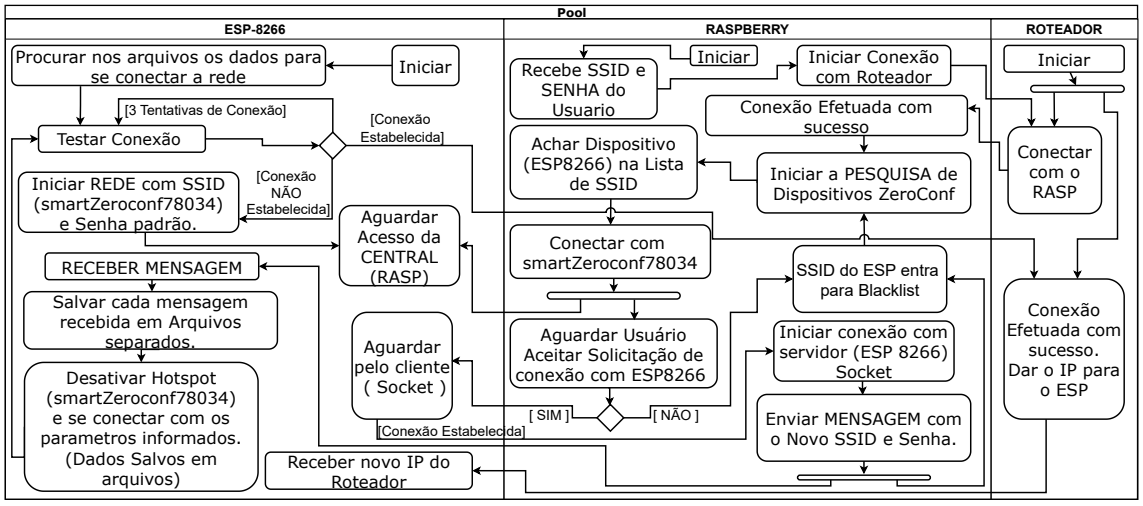


**Figura 1. Diagrama de Atividades dos dispositivos**

O programa é reiniciado para buscar novos dispositivos. Após o encerramento da conexão pelo Raspberry Pi, o ESP8266 retorna ao modo Wi-Fi e tenta novamente conectar-se usando as informações dos arquivos armazenados.

## 4. Implementação

Nesta seção, foi explorada a configuração dos hardwares empregados, seguida pela descrição da conexão automática entre os dispositivos e a transferência de dados entre eles. Também foi abordado o modo como o script Python identifica os dispositivos ESP, realizando varredura por novos SSIDs e gerando senhas Wi-Fi exclusivas com base nos nomes dos pontos de acesso gerados aleatoriamente pelo ESP. Dois códigos foram elaborados para este projeto. O primeiro, implementado em Python, foi desenvolvido através do software Visual Studio Code. No âmbito do hardware, o segundo código foi criado utilizando a linguagem C e foi destinado a um microcontrolador ESP8266 LOLIN (WeMos) D1 R1. O desenvolvimento ocorreu no ambiente Arduino IDE versão 2.2.1, utilizando as bibliotecas essenciais, tais como Arduino.h, ESP8266WiFi.h, SPI.h, FileOperations.h, Crypto.h e SHA256.h. Os códigos podem ser visualizados acessando o link https://github.com/jonathanthiengo/OpenAPFinder

Conforme ilustrado na figura 2, o processo inicial do modo Access Point no ESP8266, no qual o SSID gerado combina uma parte fixa, como ”smartZeroconf”, com uma série de números aleatórios, resultando em ”smartZeroconf78034”. Adicionalmente, para reforçar a segurança, implementou-se uma criptografia para gerar uma senha de 64 caracteres a partir do SSID. Um número SEED é acrescentado ao final deste valor, definido pelos desenvolvedores do sistema durante a compilação do código, dificultando a identificação e o acesso à rede por potenciais invasores, fortalecendo assim a segurança do sistema.

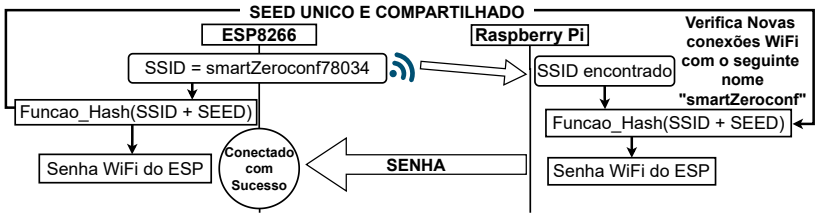


**Figura 2. Diagrama de conexão segura do OpenAPFinder**

No Raspberry Pi, um programa inicia armazenando o SSID e a senha da rede Wi-Fi à qual está conectado. Em seguida, é executado o modo de busca por um conjunto específico de caracteres na rede Wi-Fi, especificamente ”smartZeroconf”. Se uma rede correspondente for detectada, o Raspberry Pi salva o nome da rede encontrada e aguarda a aceitação do usuário para a conexão. O usuário tem 3 tentativas de 30 segundos cada. Se exceder o tempo ou não aceitar a conexão, o SSID do dispositivo é adicionado à *blacklist*, evitando novas tentativas de conexão (Figura 3).

Se o usuário aceitar a conexão, é iniciada uma função para gerar a senha da rede encontrada, seguindo o mesmo procedimento utilizado no ESP8266. Após esses passos, o Raspberry Pi utiliza o valor gerado como a senha Wi-Fi do dispositivo no qual está tentando se conectar, que no caso é o ESP8266.

Uma vez conectado com sucesso ao ESP8266 no modo AP, este entra no modo ”servidor”e o Raspberry Pi no modo ”cliente”por meio de um socket de rede. Os dados essenciais, como SSID e senha da rede Wi-Fi do usuário, são transferidos pelo cliente para o servidor, e armazenados em arquivos separados. Após a transferência bem-sucedida

dos dados, a conexão é encerrada e o SSID do ESP8266 é adicionado à *blacklist*, que é redefinida a cada cinco minutos. O programa é então reiniciado para buscar novos dispositivos (Figura 4).

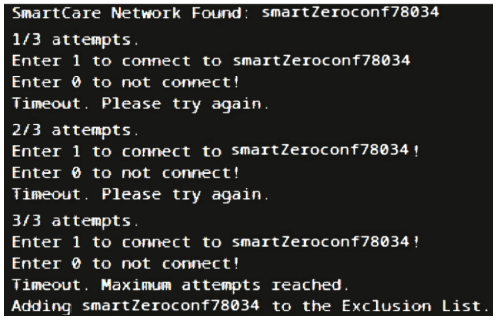


**Figura 3. Visão do ESP8266 ao reconhecer uma rede e adicioná-la a *blacklist***


**Figura 4. Transferência e Finalização da Conexão**

O ESP8266 inicia o processo de conexão Wi-Fi com a rede local por meio dos dados recebidos pelo Raspberry Pi conforme apresentado na Figura 5. Ao estabelecer a conexão com sucesso, o ESP8266 recebe do roteador um endereço IP, evitando conflito de endereçamento de IP, uma vez que a alocação dinâmica é administrada pelo roteador por meio do protocolo DHCP (Dynamic Host Configuration Protocol). Essa segregação de tarefas otimiza a eficiência e confiabilidade da rede, promovendo uma gestão mais eficaz dos recursos de IP.

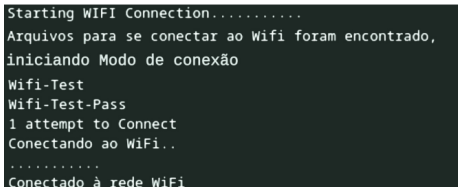


**Figura 5. Conexão Wi-Fi do ESP8266**

## 5. Experimentos e Discussão

O objetivo do teste é compreender o comportamento do sistema em diferentes fases, analisando o Tempo de Reconhecimento, o Tempo de Resposta do Usuário e o Tempo de Conexão e Transferência de Dados. Os testes foram realizados em um MacBook White 2009, equipado com um processador Intel Core 2 Duo de 2,26GHz, 3MB de cache L2 e 5 GiB de RAM. O dispositivo conta com uma rede sem fio Wi-Fi interna AirPort Extreme e opera com o Sistema Operacional Kali Linux.

Na Figura 6, o tempo total desde a ativação do ESP8266 até a conclusão da conexão em cada teste é representado. Esse intervalo engloba todas as etapas, desde a inicialização até o encerramento da interação com o ESP8266. O tempo de reconhecimento é o intervalo desde a ativação do ESP8266 na rede elétrica até o momento em que o computador o identifica. O Tempo de Resposta do Usuário refere-se ao tempo necessário para que o usuário responda à solicitação de conexão(Figura 3), com um tempo alvo de

cerca de 15 segundos para cada teste. O Tempo de Conexão e Transferência de Dados engloba o tempo desde a solicitação até a efetiva conexão com o ESP8266, incluindo também o tempo necessário para a transferência dos dados.

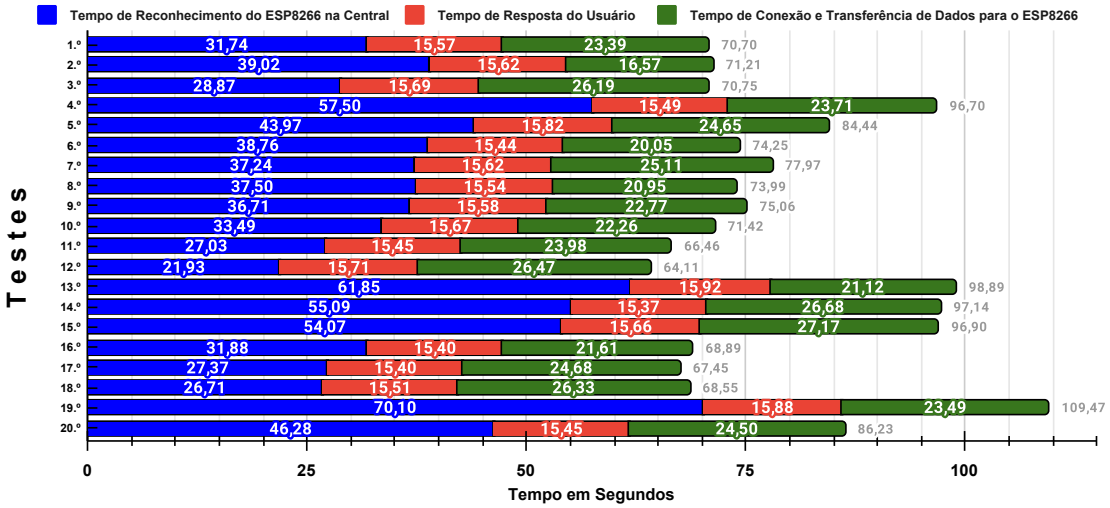


**Figura 6. Tempo Total em Cada Teste**

Após os testes, foi calculada a média dos resultados dos 20 testes realizados, abrangendo a identificação do dispositivo, a validação do usuário e a conexão até o encerramento da transferência dos dados. Como mostrado na Figura 7, a média foi de aproximadamente 40,36 segundos.

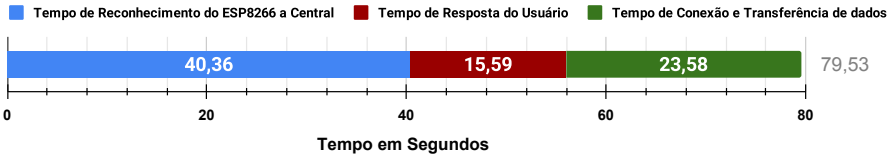


**Figura 7. Média do Tempo de Conexão**

Após os testes, a validação do usuário durou em média 15,59 segundos, enquanto a conexão e transferência de dados para o ESP8266 consumiram cerca de 23,58 segundos. A média total do processo, desde a ativação do ESP8266 até a conclusão da transferência dos dados, foi de aproximadamente 79,53 segundos.

## 6. Conclusão

Neste artigo, foi apresentada a proposta do sistema OpenAPFinder, que visa conectar dispositivos ESP8266 e Raspberry Pi usando o protocolo Zero Configuration Networking. O sistema utiliza scripts em ambos os dispositivos: o ESP8266 ativa o modo SoftAP, enquanto o Raspberry Pi busca e identifica a rede criada pelo ESP8266. Após identificação, uma conexão entre os dispositivos é solicitada automaticamente, permitindo a transferência de dados da rede do Raspberry Pi. Embora o objetivo inicial deste trabalho tenha sido alcançado, alguns problemas foram encontrados.

Um dos problemas observados é o tempo de reconhecimento do ESP8266, conforme demonstrado na Figura 6. Em alguns casos o tempo de reconhecimento da rede

e de transferência de dados ultrapassou os 50 segundos. Ao conectar-se ao ESP8266, o Raspberry Pi fica desconectado da rede principal por uma média de 23,58 segundos, conforme indicado na Figura 7, isto irá suspender a comunicação do dispositivo com outros sensores conectados a ele. É necessário que todos os dispositivos tenham mecanismos de cache de requisições para poder retomar a conexão com a central.

Para futuras implementações, estão planejados testes utilizando uma conexão via cabo para o Raspberry Pi, juntamente com a conexão Wi-Fi existente. Isso permitirá que o Raspberry Pi mantenha uma conexão constante enquanto continua recebendo dados dos ESPs já conectados. Também serão realizados testes em outros dispositivos para avaliar eventuais limitações de hardware nos testes anteriores, com foco na resolução e melhoria dos tempos de reconhecimento dos dispositivos, visando reduzir a média total de conexão com o ESP8266.

Diante disso, o sistema OpenAPFinder será continuamente aprimorado para superar essas dificuldades e otimizar sua eficácia. Com esses objetivos, espera-se que as principais contribuições resultem em um método eficaz para integrar a tecnologia de detecção e conexão de dispositivos em uma rede Wi-fi.

## Referências


Dijkstra, F., Van der Ham, J. J., and de Laat, C. T. (2006). Using zero configuration technology for ip addressing in optical networks. *Future Generation Computer Systems*, 22(8):908–914.

Ikasamo, H. AutoConnect for ESP8266/ESP32 — hieromon.github.io. `https://hieromon.github.io/AutoConnect/index.html`. [Accessed 01-12-2023].

Khan, M. A., Cherif, W., Filali, F., and Hamila, R. (2017). Wi-fi direct research-current status and future perspectives. *Journal of Network and Computer Applications*, 93:245–258.

Lee, J. W., Schulzrinne, H., Kellerer, W., and Despotovic, Z. (2007). z2z: Discovering zeroconf services beyond local link. In *2007 IEEE Globecom Workshops*, pages 1–7. IEEE.

Oliveira, R., Feres, R., Barreto, F., and Abreu, R. (2022). Cnn for elderly wandering prediction in indoor scenarios. *SN Computer Science*, 3(3):230.

Oliveira, R. F., Barreto, F., and Abreu, R. (2020). Convolutional neural network for elderly wandering prediction in indoor scenarios. *arXiv preprint arXiv:2012.12987*.

Siddiqui, F., Zeadally, S., Kacem, T., and Fowler, S. (2012). Zero configuration networking: Implementation, performance, and security. *Computers & electrical engineering*, 38(5):1129–1145.

Steinberg, D. H. and Cheshire, S. (2005). *Zero Configuration Networking: The Definitive Guide: The Definitive Guide*. "O'Reilly Media, Inc.".

Wagner, S. (2008). Zero-configuration of pervasive healthcare sensor networks. In *2008 Third International Conference on Pervasive Computing and Applications*, volume 1, pages 405–408. IEEE.