

# FERRAMENTAS DE SEGURANÇA DE DADOS PARA SISTEMAS IOT

RHUAN MARTINS DE SOUZA<sup>1</sup>, ESDRAS NICOLETTO DA CUNHA<sup>2</sup>

<sup>1</sup> Graduando em Engenharia de Controle e Automação, IFSP, Câmpus São João da Boa Vista, rhuam.m@aluno.ifsp.edu.br.

<sup>2</sup> Professor orientador, IFSP, Câmpus São João da Boa Vista, esdras.nicoletto@ifsp.edu.br.

**RESUMO:** A automação residencial é uma área da tecnologia que estuda a automatização de uma residência ou habitação para oferecer conforto, praticidade e segurança. Nas últimas décadas, muitos profissionais se concentraram em conectar equipamentos, sensores e atuadores à Internet, para permitir que os usuários controlem e monitorem as coisas a qualquer hora e em qualquer lugar. Diversos equipamentos IoT (Internet of things) podem ter problemas em relação a segurança de informações, por estarem diretamente conectados à rede doméstica, além de toda a comunicação facilitada, o equipamento também fica suscetível a diversos ataques, que comprometem as suas funcionalidades, criando caminhos, onde usuários não autorizados podem aproveitar as brechas de segurança e roubar dados pessoais diretamente da rede doméstica. Este trabalho comprova uma metodologia de aplicação de IOT, utilizando um Raspberry Pi, onde os principais sistemas de proteção e comunicação estão configurados, o microcontrolador ESP32, que é responsável por receber os dados do sensor de gás (MQ-02) e os envia-los ao Raspberry Pi com segurança e praticidade. Assim como qualquer sistema IoT, que traz os benefícios da integração de equipamentos digitais e gera automatização de tarefas, mas com foco principal na proteção dos dados do sistema e dos dispositivos conectados à rede, através de ferramentas de segurança e boas práticas de design de rede, visando diminuir as fragilidades possíveis nos dispositivos conectados à rede.

**PALAVRAS-CHAVE:** IoT. Automação Residencial. Segurança.

## DATA SECURITY TOOLS FOR IOT SYSTEMS

**ABSTRACT:** Home automation is an area of technology that studies the automation of a home or dwelling to provide comfort, convenience, and security. In recent decades, many professionals have focused on connecting equipment, sensors, and actuators to the Internet to allow users to control and monitor things anytime, anywhere. Several IoT (Internet of things) devices may have problems regarding information security, because they are directly connected to the home network. Besides all the facilitated communication, the equipment is also susceptible to several attacks that compromise its functionality, creating paths where unauthorized users can take advantage of security holes and steal personal data directly from the home network. This work proves a methodology of IoT application, using a Raspberry Pi, where the main protection and communication systems are configured, the ESP32 microcontroller, which is responsible for receiving the data from the gas sensor (MQ-02) and sending them to the Raspberry Pi safely and conveniently, just like any IoT system, bringing all the benefits of the integration of digital equipment and generating the automation of tasks, but with the main intention of focusing on the protection of the system data and the devices connected to the network, through security tools and good practices of network design, aiming to reduce the possible weaknesses in the devices connected to the network.

**KEYWORDS:** IoT. Home automation. Security.

## 1. INTRODUÇÃO

A automação residencial é uma tecnologia que estuda a automatização de uma residência ou habitação para oferecer conforto, praticidade e segurança. Trata-se de um mercado em forte expansão e consolidado a vários anos, desenvolvendo uma oferta articulada e bem recebida pelos clientes. Nas últimas décadas, muitos profissionais se concentraram em conectar objetos, incluindo aparelhos, sensores e atuadores a Internet, para permitir que os usuários controlem e monitorem as coisas a qualquer hora e em qualquer lugar. Essa revolução inventou o termo IoT (*Internet of Things*) (ASHTON, 2009).

Desta maneira é inevitável que o aumento desta ideia e a implementação em diversos campos, atrai profissionais para o atuar nesta área. Pesquisadores estudam formas de aperfeiçoar as implementações dos sistemas, tanto em *hardware* como em *software*. Portanto, para contribuição deste trabalho, com a comunidade acadêmica, que busca conhecimento sobre aplicações e melhorias constantes na segurança e proteção de dados que trafegam em rede doméstica ou industrial em sistemas digitais e automáticos, vendo que independentemente do local onde a automação será aplicada e do seu uso final, devemos ter a segurança como prioridade, através de táticas de design de redes e segurança de dados.

Todos os sistemas IoT são conectados entre si, logo, existe alta passagem de dados entre os dispositivos. Alguns desses dados, em um sistema ideal, devem se manter privados, apenas para os usuários autorizados do sistema. Porém, através de ataques indesejados como D/DoS<sup>1</sup> (*Denial-of-service*) ou MITM<sup>2</sup> (*Man-in-the-middle*), pode-se ter os dados vazados, o que comprometeria o sistema e os usuários, principalmente quando se trata de uma residência, já que os dados poderiam infringir a privacidade pessoal de todos os moradores.

Diante deste cenário, esse trabalho tem como objetivo auxiliar no funcionamento ideal do sistema, tanto no controle dos sensores/atuadores da residência e na segurança dos dados que trafegam sobre os equipamentos de comunicação. Procura-se também verificar como os mecanismos e o design de rede melhoram a segurança dos sistemas IoT, através do sistema operacional Linux. Além disso, pretende-se implementar uma interface de controle (*Framework OpenHABian*) em web server, e testar a eficiência dos sistemas, verificando a privacidade dos dados e os protocolos utilizados por todo o sistema de acordo com a utilização das táticas de encriptação dos dados em navegação, protocolos de comunicação e da DMZ's (*Demilitarized zone* ou zona desmilitarizada).

## 2 FUNDAMENTAÇÃO TEÓRICA

A fundamentação é uma breve revisão sobre as principais teorias e conceitos utilizados ao decorrer do artigo, em suma, são conceitos utilizados em artigos, dissertações e livros científicos com alta relevância em meio ao público acadêmico. A fundamentação se divide em alguns tópicos, onde vemos uma breve teoria sobre automação residencial, teorias sobre IOTs e segurança de dados, buscando validar o tema de pesquisa experimental, para a aplicação e desenvolvimento de uma metodologia segura, para implementação de IOTs com foco em automação residencial.

### 2.1 Automação Residencial

Domótica é a adoção de um sistema para controlar e monitorar a iluminação, condições atmosféricas, sistemas de entretenimento, sistemas de vigilância e eletrodomésticos. Ele permite que dispositivos em residências sejam conectados a uma rede, possibilitando o controle remoto desses equipamentos, tal como a ligação de lâmpadas ou qualquer outro equipamento elétrico/eletrônico. Essa tecnologia facilita a vida do usuário e economiza energia utilizando dispositivos com maior prioridade e importância, utilizando dispositivos como smartphones e computadores, podemos acessar, verificar, enviar comandos e receber respostas dos dispositivos coligados a uma rede de automação residencial (SOWAH et al, 2020, p. 2). A tabela 1 mostra o ranking de países com o número de casas inteligentes (em milhões) e sua porcentagem. Os dados demonstram que o mercado é grande, e ainda pode ser

---

<sup>1</sup> Ataque cibernético, em que a máquina alvo tem seus recursos interrompidos, através de muitas solicitações de serviço, ocasionando uma sobrecarga de dados, comprometendo o sistema.

<sup>2</sup> Ataque cibernético, em que um invasor fica no meio da comunicação do sistema, captando os dados enviados pelo usuário, possivelmente modifica os dados e altera a comunicação entre duas partes do sistema.

explorado, portanto, é necessário que haja sempre o crescimento do setor, visando melhorias intrínsecas no *hardware* e *software* do sistema.

Tabela 1 – Dados sobre *smart houses*.

Ranque (Número de casas inteligentes)	País	Número de casas inteligentes (Milhões)	Porcentagem de casas inteligentes (% do total de casas)
1	USA	40,3	32
2	China	19,3	4,9
3	Japão	7	15
4	Alemanha	6,1	15,7
5	Reino Unido	5,3	19,7
6	Coreia do Sul	4	20
7	Índia	2,2	<1
8	Austrália	1,8	19,12
9	Brasil	1,2	1,9
10	Rússia	0,9	1,7

Fonte: The Ambient Team (2018).

O artigo "*Around the world in smart home stats*", nos permite analisar a Tabela 1 e traz um panorama da adoção de dispositivos e tecnologias de casa inteligente em diversos países. De acordo com o artigo, o mercado global de casas inteligentes deve atingir um valor de US\$ 138,9 bilhões em 2023, com uma taxa de crescimento anual composta de 13,6% entre 2018 e 2023.

Nos Estados Unidos, a adoção de dispositivos de casa inteligente tem sido liderada por termostatos inteligentes, com uma porcentagem no mercado de 18,5%, seguida por sistemas de segurança (15,3%) e assistentes de voz (13,4%). Além disso, 70% dos usuários de dispositivos inteligentes dizem que a segurança é a principal razão para a adoção.

Na Europa, o Reino Unido lidera em termos de adoção de dispositivos de casa inteligente, com um mercado avaliado em £ 900 milhões em 2020. Termostatos inteligentes são os dispositivos mais populares no Reino Unido, seguidos por assistentes de voz e sistemas de segurança.

Na Ásia, a China é o maior mercado de casas inteligentes, com um valor estimado em US\$ 22,8 bilhões em 2019. A adoção de dispositivos de casa inteligente na China é impulsionada pelo governo, que está incentivando a adoção de tecnologias de internet das coisas em todo o país. Os dispositivos mais populares na China incluem câmeras de segurança, assistentes de voz e robôs de limpeza.

O artigo também observa que a privacidade é uma grande preocupação para muitos consumidores quando se trata de dispositivos de casa inteligente, com 60% dos usuários relatando preocupações sobre como seus dados são coletados e usados.

Em resumo, o mercado global de casa inteligente está em ascensão, com dispositivos como termostatos inteligentes, sistemas de segurança e assistentes de voz liderando a adoção. A segurança é uma preocupação importante para muitos consumidores e a privacidade continua a ser uma preocupação crescente.

## 2.2 Fragilidades encontradas em dispositivos IoT (*Application Layer, Processing Layer, Transport Layer e Perception Layer*)

Nas últimas duas décadas, os hackers ganharam mais publicidade e notoriedade do que nunca. Portanto, o termo "Hacking" ficou popularmente conhecido como um conceito intimamente relacionado à internet e a maioria dos hackers apareceram nos últimos 20 anos. Mas isso é apenas parte da verdade. Embora o número de hackers tenha aumentado no mundo com o surgimento da *World Wide Web*(WWW), os *hackers* surgiram desde meados do século 20. Muitos especialistas argumentam que a segunda década do século 20 marcou as verdadeiras origens da modernidade *hacker*, como alguns dos principais eventos do início de 1900 mostram semelhanças impressionantes com o *hacking* atual. Por exemplo, existem certos incidentes isolados que podem ser considerados

“hacks”, nas décadas de 1910 e 1920, muito disso envolvia a manipulação do envio de código Morse em transmissores e receptores, ou interferindo na transmissão de ondas de rádio (HOFFMAN, 2020).

A implementação de quatro camadas para o IoT em questão, sendo essas camadas, de aplicação, de processamento, de transporte e de percepção, respectivamente responsáveis pela interface IHM, web-server, transmissão e recebimento de dados e por último, a camada responsável pelos sensores e atuadores. Algumas dessas camadas possuem diversas possibilidades de falhas, detalhadas a seguir e descritas segundo Hassija et al (2019).

### **Possíveis fragilidades na camada de Aplicação**

*Access Control Attacks:* Ataques de controle de acesso referem-se a uma classe de ataques que visam comprometer os mecanismos de autenticação e autorização em um sistema, permitindo que um invasor acesse informações ou recursos que normalmente não estariam disponíveis. No contexto de IoT, um ataque de controle de acesso pode ser crítico porque pode permitir que um invasor acesse dispositivos ou dados sensíveis, comprometendo a integridade e confidencialidade dos dados. Além disso, se um dispositivo de IoT comprometido for usado como um ponto de entrada em uma rede, um ataque de controle de acesso pode permitir que um invasor acesse outros dispositivos na rede, causando ainda mais danos.

*Malicious code injection attacks:* Os invasores geralmente optam pelo método mais fácil ou simples que podem usar para invadir um sistema ou rede. Se o sistema estiver vulnerável a scripts maliciosos e direcionamentos incorretos devido a verificações de código insuficientes, esse seria o primeiro ponto de entrada que um invasor escolheria. Geralmente, os invasores usam XSS (*cross-site scripting*), Ataques de injeção de SQL (*SQL injection attacks*), de injeção de comandos (*command injection attacks*) e de injeção de XML (*XML injection attacks*) para injetar algum script malicioso em um site confiável..

*Reprogram attacks:* tipo de ataque que envolve alterar o firmware ou software em dispositivos IoT para obter controle sobre eles. Isso pode ser feito remotamente se o dispositivo estiver conectado à Internet e for vulnerável a ataques. O objetivo dos invasores pode ser alterar a funcionalidade do dispositivo, desabilitá-lo ou usá-lo como parte de um ataque maior, (AKRAM; KONSTANTAS; MAHYOUB, 2018).

### **Possíveis fragilidades na camada de Processamento**

*SQL Injection attack:* A camada de processamento também é suscetível a ataques de *SQL Injection* (SQLi). Nesses ataques, o invasor pode incorporar instruções SQL maliciosas em um programa. Assim, os invasores podem obter dados privados de qualquer usuário e podem até alterar registros no banco de dados (RAZZAQUE et al., 2016).

*Man-in-the-Middle Attack:* O protocolo MQTT(*Message Queue Telemetry Transport*) usa o modelo de publicação-assinatura de comunicação entre clientes e assinantes usando o broker MQTT(Servidor que gere as informações), que atua efetivamente como um proxy. Isso ajuda a desacoplar a publicação e os clientes assinantes uns dos outros e as mensagens podem ser enviadas sem o conhecimento do destino. Se o invasor puder controlar o corretor e se tornar um *man-in-the-middle*, ele poderá obter controle total de toda a comunicação sem nenhum conhecimento dos clientes.

### **Possíveis fragilidades na camada de Transmissão**

*Phishing Site attack:* Tipo de ataque que tenta enganar usuários para que revelem informações pessoais e confidenciais, como nome de usuário, senha e detalhes financeiros, por meio de um site falso ou fraudulento que se parece com um site legítimo. Esse tipo de ataque pode ser direcionado a dispositivos IoT, como câmeras e dispositivos de segurança doméstica, para obter acesso a credenciais de login ou informações sensíveis. No entanto, a camada de rede em si não é vulnerável a ataques de

phishing; são os usuários que estão em risco de serem enganados por sites falsos. Além disso, uma vez que as informações de login são comprometidas, o ambiente de IoT do usuário pode se tornar vulnerável a ataques cibernéticos (“APWG”, 2019).

*DDoS/DOS Attack:* Nesse tipo de ataque, o invasor inunda os servidores de destino com um grande número de solicitações indesejadas. Isso incapacita o servidor de destino, interrompendo assim os serviços para usuários genuínos. Se houver várias fontes usadas pelo invasor para inundar o servidor de destino, esse ataque será denominado DDoS ou ataque de negação de serviço. Tais ataques não são específicos para aplicações IoT, mas devido à heterogeneidade e complexidade das redes IoT, a camada de rede da IoT é propensa a tais ataques. Muitos dispositivos IoT em aplicativos IoT não são fortemente configurados e, portanto, tornam-se gateways fáceis para os invasores lançarem ataques DDoS nos servidores de destino (KOLIAS et al., 2017).

*Data Transit Attacks:* Os aplicativos de IoT lidam com muito armazenamento e troca de dados. Os dados são valiosos e, portanto, são sempre alvo de hackers. Os dados armazenados nos servidores locais ou na nuvem apresentam um risco de segurança, mas os dados que estão em trânsito ou em movimento de um local para outro são ainda mais vulneráveis a ataques cibernéticos. Em aplicativos de IoT, há muita movimentação de dados entre sensores, atuadores, nuvem, etc. Diferentes tecnologias de conexão são usadas em tais movimentações de dados e, portanto, os aplicativos de IoT são suscetíveis a violações de dados.

### **Possíveis fragilidades na camada de Percepção**

*Malicious code injection attacks:* Os invasores geralmente optam pelo método mais fácil ou simples que podem usar para invadir um sistema ou rede. Se o sistema estiver vulnerável a scripts maliciosos e direcionamentos incorretos devido a verificações de código insuficientes, esse seria o primeiro ponto de entrada que um invasor escolheria. Geralmente, os invasores usam XSS (cross-site scripting), Ataques de injeção de SQL (SQL injection attacks), de injeção de comandos (command injection attacks) e de injeção de XML (XML injection attacks) para injetar algum script malicioso em um site confiável.

*Sleep Deprivation Attacks:* Uma forma de ataque cibernético que visa esgotar a bateria dos dispositivos IoT de baixa potência, como sensores, atuadores e outros dispositivos de borda (dispositivos IoT que estão localizados na borda de uma rede, ou seja, dispositivos que estão mais próximos do local onde ocorre a coleta ou geração de dados). Isso é feito por meio da execução de loops infinitos no dispositivo, aumento artificial do consumo de energia ou uso de código malicioso que força o dispositivo a realizar tarefas desnecessárias e consumir mais energia do que o necessário. O objetivo desse tipo de ataque é negar o serviço do aplicativo IoT, tornando o dispositivo inutilizável ou reduzindo significativamente sua capacidade de operar.

*Bootling Vulnerabilities:* Os dispositivos de borda são vulneráveis a vários ataques durante o processo de inicialização. Isso ocorre porque os processos de segurança integrados não estão habilitados nesse ponto. Os invasores podem aproveitar essa vulnerabilidade e tentar atacar os dispositivos quando eles estiverem sendo reiniciados. Como os dispositivos de borda geralmente são de baixa potência e às vezes passam por ciclos de suspensão, é essencial proteger o processo de inicialização nesses dispositivos.

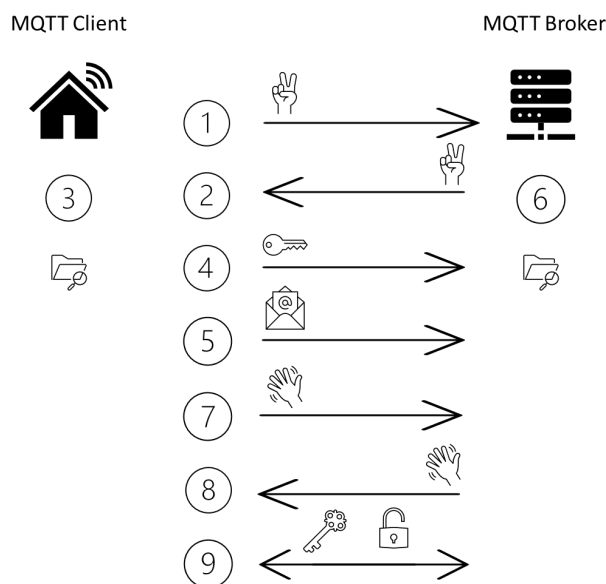
### **2.3 Táticas de solução aplicáveis**

Visando aumentar a segurança de sistemas conectados à internet, é necessário que haja a criação de um plano de segurança, que busque analisar os riscos (perda de dados e acessos não autorizados) em que o projeto está envolvido, de certa forma dispondo-se de parâmetros iniciais para as políticas de segurança do sistema, considerando também os custos, conhecimento e treinamento da equipe e o hardware/plataformas que são ou serão utilizadas.

O protocolo *Transport Layer Security*(TLS) é usado por dispositivos e aplicações *endpoint* (dispositivos que representam fisicamente, um ponto final de uma rede), visando autenticar e

criptografar dados com segurança quando transferidos por uma rede. O protocolo TLS é um padrão amplamente aceito e utilizado por dispositivos como computadores, telefones, *IoT*s, medidores e sensores. Mais precisamente, o protocolo é encontrado em aplicações quando os usuários acessam informações usando um *cliente web* (*software* que permite aos usuários acessarem e interagirem com conteúdos na *web*), assim como roteamento de rede. A comunicação deste protocolo é feita através de sinalizações, conduzindo o cliente ao servidor, ambos sinalizam as opções de criptografia e autenticação da sessão TLS, o cliente envia uma sinalização desejando iniciar a conexão com o servidor, o servidor, por sua vez, está programado para utilizar uma determinada encriptação para autenticar a sessão, o servidor retorna essas preferências para o cliente, envia o certificado digital ao cliente, o certificado possui a chave pública e outras informações do servidor. O cliente verifica o certificado do servidor, confirma sua identidade e utiliza a chave pública do servidor para estabelecer um canal criptografado entre ambos, enquanto essa conexão estiver ligada, existe troca de mensagens entre eles. Durante a comunicação, o TLS utiliza algoritmos de criptografia para proteger a confidencialidade e a integridade das informações transmitidas. O TLS também oferece opções para autenticação do servidor e do cliente, a fim de garantir que apenas as partes autorizadas tenham acesso às informações transmitidas (DIERKS; RESCORLA, 2008).

Figura 1. Sequência do *handshake*



Fonte: Elaboração Própria.

O TLS possui um passo muito importante, denominado de *handshake* (aperto de mão), é como o protocolo inicia, cria e mantém a comunicação com o cliente e o *webserver* de forma segura. consiste nos seguintes passos seguindo a figura 1:

1. O cliente envia uma solicitação de conexão (Olá).
2. O servidor responde com o certificado digital que contém a chave pública.
3. O cliente verifica o certificado (APÊNDICE A, contida da linha treze até a linha trinta e dois)
4. O cliente gera uma chave de sessão usando a chave pública do servidor.
5. O cliente envia o certificado.
6. O servidor verifica o certificado do cliente.
7. O cliente termina o reconhecimento
8. O servidor termina o reconhecimento.
9. Inicia-se a troca de mensagens criptografada pela própria chave da sessão.

O algoritmo RSA(Rivest-Shamir-Adleman), sistema criptográfico de chave pública, permite a troca segura de informações entre duas partes, sem que seja necessário compartilhar a chave secreta utilizada para criptografar e descriptografar as mensagens. O algoritmo RSA é baseado em dois números primos grandes e aleatórios, “p” e “q”. O processo de criptografia e descriptografia começa com a escolha desses dois números primos e o produto deles igual a  $N$  ( $pq = N$ ). O valor de  $N$  é utilizado para gerar duas chaves diferentes, a chave pública e a chave privada (RIVEST; SHAMIR; ADLEMAN, 1978).

O algoritmo 3DES(*Triple Data Encryption Standard*) utiliza uma técnica chamada cifra de bloco, que criptografa os dados em blocos de 64 bits. O algoritmo executa três rodadas do algoritmo DES (*Data Encryption Standard*) em cada bloco de dados, usando três chaves diferentes. As três chaves são geradas a partir da chave de 168 bits usando um processo chamado "derivação de chave" (*key derivation*). O algoritmo DES é executado em modo CBC (*Cipher Block Chaining*), o que significa que cada bloco de dados é cifrado usando o bloco anterior como vetor de inicialização. O processo de cifragem funciona dividindo a chave de 168 bits em três chaves de 56 bits cada, executa-se o algoritmo DES no bloco de dados usando a primeira chave, executa-se o algoritmo DES novamente no resultado da primeira operação, usando a segunda chave e por fim, executa-se novamente o algoritmo no resultado da segunda operação usando a terceira chave, o processo de decifragem é feito de forma reversa. O DES é um algoritmo de criptografia simétrica de bloco de 64 bits que usa uma chave de 56 bits para cifrar e decifrar os dados. Ele divide o bloco de 64 bits em blocos menores e aplica a cifra a cada bloco. O algoritmo requer uma série de etapas que envolvem substituições e permutações dos bits do bloco, juntamente com a aplicação de uma subchave gerada a partir da chave principal (BROWN; THOMSON, 2018).

O conceito da DMZ fornece um recurso de segurança em multicamadas, em contrapartida, tem-se o aumento de custos, uma vez que o mesmo contém componentes e administração adicionais, pela maior complexidade do sistema. Uma DMZ é um método que fornece a segregação de redes e serviços que precisam ser fornecidos a usuários, visitantes ou parceiros por meio do uso de firewalls e múltiplas camadas de filtragem e controle para proteger sistemas internos. Em redes de computadores, uma zona desmilitarizada é um computador *host* ou uma pequena rede, inserida como uma “zona neutra” entre a rede privada de uma empresa (neste caso, seria o sistema de automação residencial) e o público externo à rede. A DMZ impede que usuários externos recebam acesso a um servidor que possui dados da empresa. Uma DMZ é uma opção segura para a abordagem de um firewall e atua efetivamente como um servidor *proxy* (SHIMONSKI, 2003).

### 3 MATERIAL E MÉTODOS

Neste tópico são descritas as etapas e a metodologia proposta. Apresentam-se também os materiais ou equipamentos utilizados e os procedimentos experimentais adotados para a realização desta pesquisa. Composto por três tópicos principais: (1) Equipamentos utilizados, onde é detalhado o *hardware/software* utilizado; (2) Implementação do *OpenHABian*, MQTT(*Message Queuing Telemetry Transport*) com protocolo TLS 1.2 e DMZ;

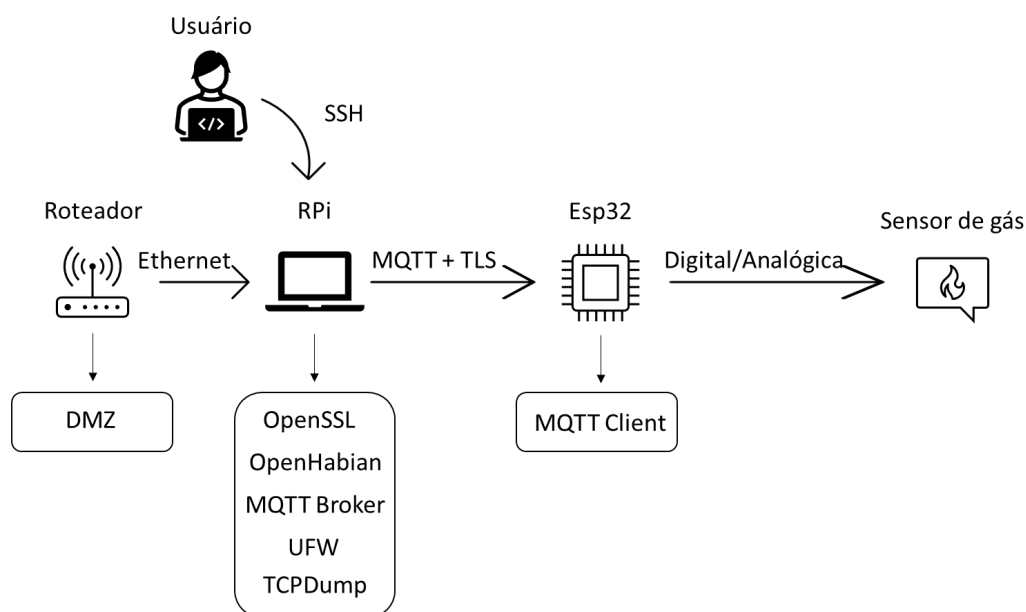
#### 3.1 Equipamentos utilizados

*Hardware*: Será utilizado um *Raspberry pi 3 model b*, sendo mencionado como “RPi”, é um micro-computador, com um processador *Quad Core* 1.2GHz, 1 GB de memória RAM, e *wireless*(rede sem fio), responsável por receber os dados enviados pelo ESP32, e disponibilizá-los através da interface *OpenHABian*; um microcontrolador ESP32; um módulo de sensor de gás(MQ-02); um roteador TL-WR949N(Onde será aplicado a DMZ na porta onde o RPi fica conectado) e também um computador com o sistema *Windows*(servirá para acessar o *webserver* gerado pelo *OpenHABian* e configuração do roteador).

*Software e protocolos*: Será utilizado o *OpenHABian*(OPENHAB FOUNDATION E.V. et al, 2022), que junta o sistema operacional Linux(RASPBERRY PI FOUNDATION, 2019) e o *software* de automação residencial *openhav*, o mesmo será responsável por gerar a interface homem-máquina e permitir/negar o envio de comandos por dispositivos autorizados dentro da rede; PUTTY(TATHAM,

2022), terminal de simulação open source, que cria conexões seguras com canais criptografados; protocolo MQTT, para comunicação entre o RPi e o ESP32, o MQTT *Broker* utilizado será o Eclipse Mosquitto(ECLIPSE FOUNDATION, 2022); Openssl, uma biblioteca de criptografia que suporta protocolos como o TLS(OPENSSL, 2022); TPCDUMP, ferramenta utilizada para visualizar e examinar pacotes de rede em tempo real, verificando a segurança da rede e o seu desempenho(LABORATÓRIO NACIONAL LAWRENCE BERKELEY, 2022); UFW(*The Uncomplicated Firewall*) estrutura para gerenciar o filtro da internet, bem como uma interface de linha de comando para manipular o firewall(PITT, 2007).

Figura 2. Visualização da montagem



Fonte: Elaboração Própria.

### 3.2 Implementação do OpenHab, MQTT com protocolo TLS e DMZ

O OpenHABian, por ser uma ferramenta de livre acesso, com propósito de facilitar automações residenciais, será implementada no RPi. Esta ferramenta une os princípios do OpenHAB, junto ao Debian Linux. O sistema irá operar sem interface gráfica, dessa forma, podemos utilizar todo o processamento do RPi dentro dos softwares que serão implementados, dessa forma, todos os comandos serão feitos através de SSH (*Secure Shell*), com o software PuTTY(responsável por simular o terminal de comandos), que está instalado em uma máquina com o sistema *Windows*. *Promovendo* maior aproveitamento dos recursos do equipamento, visto que o sistema SSH também promove um envio de dados encriptados através de um sistema de chaves públicas para autenticação.

Os dados vindos do sensor, serão enviados ao ESP32, que atua também como o *MQTT Client*, que por sua vez, faz as publicações ao *MQTT Broker*, que organiza as informações em tópicos e cria um canal onde podemos acessar a informação por meio de publicações(enviar informação) ou inscrições(receber informação) no *webserver* criado pelo Openhab, toda a configuração será programada na linguagem de programação C, disponível no APÊNDICE A.

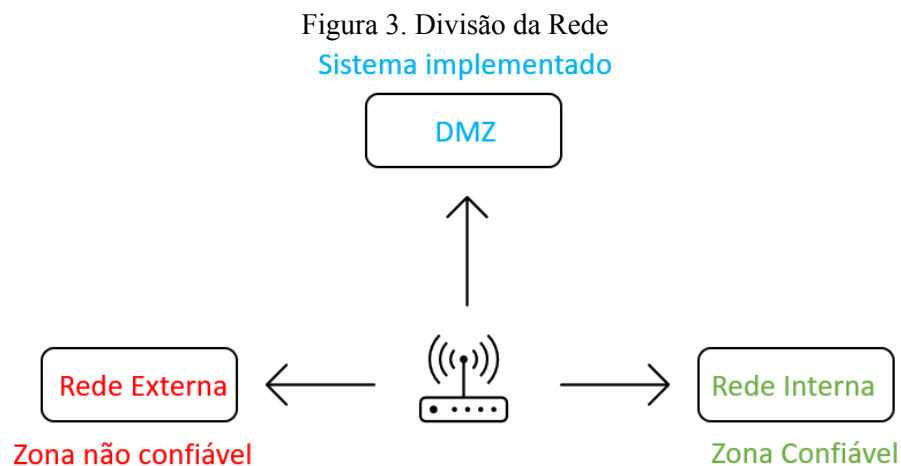
O OpenSSL é uma biblioteca de criptografia de código aberto que fornece suporte para SSL e TLS. O uso do OpenSSL em uma implementação de sistema MQTT com TLS oferece várias vantagens, incluindo: Segurança: O OpenSSL fornece criptografia forte para comunicações MQTT, protegendo os dados transmitidos de interceptação e manipulação; Confiabilidade: A biblioteca OpenSSL é amplamente utilizada e testada em uma variedade de sistemas operacionais e plataformas, o que a torna uma escolha confiável para implementações MQTT; Flexibilidade: O OpenSSL oferece suporte a uma variedade de algoritmos de criptografia e protocolos TLS, permitindo que as implementações MQTT escolham a combinação certa para atender às suas necessidades específicas; Compatibilidade: O uso do OpenSSL em uma implementação MQTT com TLS garante



compatibilidade com outros sistemas que também usam OpenSSL. Em geral, a implementação do TLS com OpenSSL em um sistema MQTT fornece um alto nível de segurança e confiabilidade na comunicação de dados, além de ser uma opção flexível e compatível com outras plataformas.

Instalando o OpenSSL no RPi, pode-se criar um par de chaves pública e privada, criar um pedido de assinatura de certificado para a chave pública do servidor MQTT, enviar esse certificado para uma autoridade de certificação, configurar o servidor para usar esses certificados e a chave privada e configurar os clientes MQTT para confiar no certificado do servidor.

A DMZ é facilmente aplicada dentro de um roteador acessando o painel de controle do próprio dispositivo roteador, implementando a configuração das portas, isolando uma das portas de conexão de internet o equipamento utilizando o seu endereço *Internet Protocol(IP)*, desta forma, ele protege os equipamentos na rede interna, como demonstrado na Figura 3.



Fonte: Elaboração Própria.

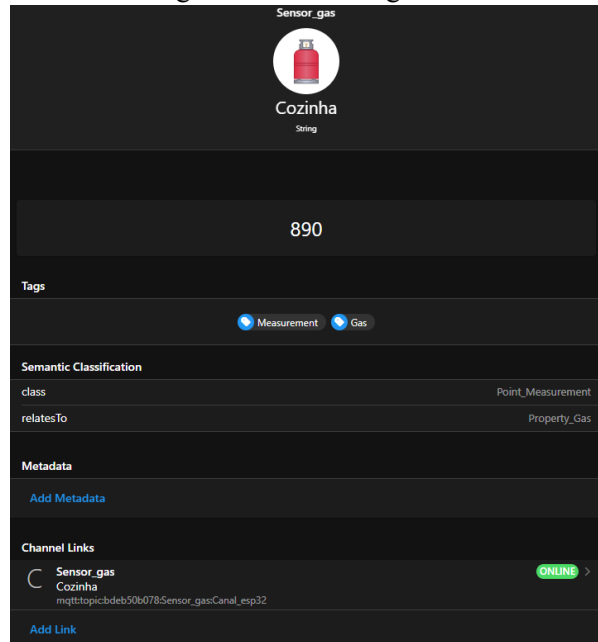
Configuração do sistema:

O primeiro passo é utilizar um software como o Etcher, para escrever a imagem do OpenHABian no cartão SD do RPi. Com o microcontrolador já programado e com suas ligações eletrônicas concluídas, basta iniciar o PuTTY, adicionar o IP referente ao RPi, inserir usuário e senha, para identificação do login(inicialmente o usuário e senha serão “openhabian”, é recomendado fazer a modificação de todos os usuários e senhas padrões para maior segurança). Ao acessar o terminal, os seguintes comandos serão utilizados:

1. `sudo apt-get update`
2. `sudo apt-get install mosquitto`
3. `sudo apt-get install mosquitto-clients`
4. `sudo apt clean`
5. `mosquitto -v`
6. `netstat -at`

Utilizando qualquer navegador web, pode-se acessar a URL(*Uniform Resource Locator*): “<http://openhabian:8080/>”, em seu primeiro acesso, a autenticação do administrador será criada. A opção *Bindings*(Vinculadores de informações), dá acesso aos aplicativos, e as diversas possibilidades de comunicação/serviços disponíveis para implementação que permitirá que os dispositivos IoT sejam conectados. Acessando a aba *Things*(referência a equipamentos), é possível configurar a conexão o Openhab com o MQTT *Broker* e através dele, criar um “*Generic MQTT Thing*”, que será o responsável por buscar nos tópicos do MQTT e retornar a publicação do ESP32.

Figura 4. Item configurado



Fonte: Elaboração Própria.

A figura 4 mostra o item configurado de maneira correta, com o canal fazendo as requisições do tópico de publicação programado no microcontrolador, o valor no momento seria de 890 ppm(partes por milhão).

Utilizando o TCPDUMP, é possível visualizar os pacotes que estão trafegando na porta 1883(MQTT Broker), utilizando os comandos abaixo:

```
7. sudo apt-get install tcpdump
8. sudo tcpdump -A -s0 port 1883
```

Para implementação dos algoritmos RSA e 3DES, juntamente com o protocolo de comunicação TLS, seguiremos em conexão SSH com o RPi e seguiremos os seguintes comandos:

```
9. sudo apt-get install ufw
10.sudo systemctl start ufw
11.sudo ufw allow 8883/tls
12.sudo ufw reload
13.sudo ufw status
```

Os comandos acima, tem o objetivo de instalar e liberar o tráfego de dados da porta 8883 utilizando o UFW, o próximo passo é criar dentro da pasta “/etc/mosquitto”, o diretório “certs”, onde serão alocados todas as chaves e certificados. Para gerar os certificados, utiliza-se os seguintes comandos:

```
14.sudo apt-get install openssl
15.openssl genrsa -des3 -out ca.key 2048
16.openssl req -new -x509 -days 1833 -key ca.key -out ca.crt
17.openssl genrsa -out server.key 2048
18.openssl req -new -out server.csr -key server.key
19.openssl x509 -req -in server.csr -CA ca.crt -CAkey ca.key
    -CAcreateserial -out server.crt -days 360
```

Esses comandos, têm a função de, respectivamente, instalar o openssl; Gerar uma chave privada com algoritmo RSA é criptografada por algoritmo 3DES, salvar essa chave no arquivo

ca.key(chave do cliente) com o tamanho de 2048 bits; Gerar a “entidade” responsável por emitir e gerenciar certificado digitais CA(*Certificate Authority*); Gerar a chave do servidor; Cria um certificado de requisição CSR(*Certificate Signing Request*), que contém informações sobre o servidor, como nome de domínio, e é usada para solicitar o certificado assinado pela autoridade de certificação CA, sendo responsável por estabelecer a identidade do servidor para os clientes que se conectam ao serviço; Cria o certificado do MQTT *Broker*, utilizando os certificados e chaves já criados, juntamente com um número de série único do certificado.

Também é necessário criar uma arquivo de texto, no diretório “/etc/mosquitto/conf.d”:

```
20.sudo nano acesso.conf
```

Com o seguinte conteúdo nele:

```
"listener 8883
cafile /etc/mosquitto/certs/ca.crt
certfile /etc/mosquitto/certs/server.crt
keyfile /etc/mosquitto/certs/server.key
tls_version tlsv1.2"
```

Esse arquivo será carregado e iniciado no MQTT *Broker*, ditando as regras de funcionamento dele, identificando os certificados e a chave do servidor, além da preferência de comunicação pelo protocolo TLS 1.2. A configuração em relação ao MQTT Broker diretamente no openhan, acessada, com qualquer navegador web pela porta 8080 tem poucas diferenças da anterior, basta adicionar a porta 8883, manter o usuário e senha escolhidos, e adicionar a *hash*(resumo) do CA nas configurações avançadas.

Utilizando o TCPDUMP, é possível visualizar os pacotes que estão trafegando na porta 8883(MQTT *Broker*), utilizando os comandos 21 e 22:

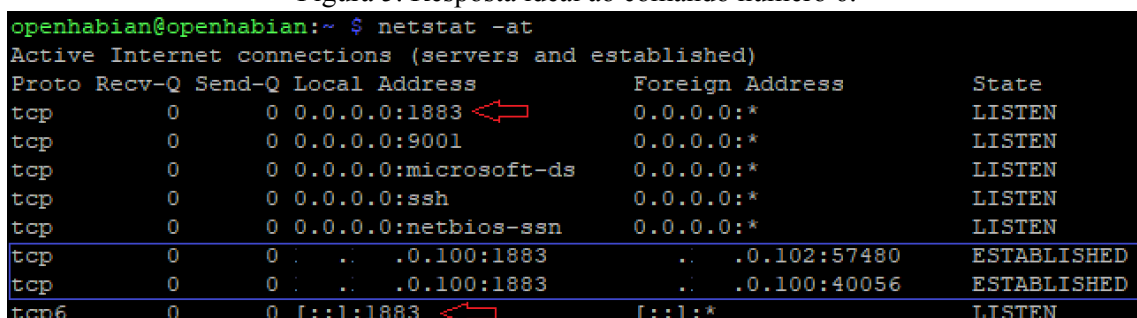
```
21.sudo apt-get install tcpdump
22.sudo tcpdump -A -s0 port 8883
```

## 4 RESULTADOS E DISCUSSÃO

### 4.1 Resultados Obtidos

A figura 5 mostra uma resposta ideal e apresenta as conexões estabelecidas, observa-se em vermelho, as conexões do MQTT *Broker*, e em azul, as conexões estabelecidas entre o MQTT *Broker* e o MQTT *Client*.

Figura 5. Resposta ideal ao comando número 6.

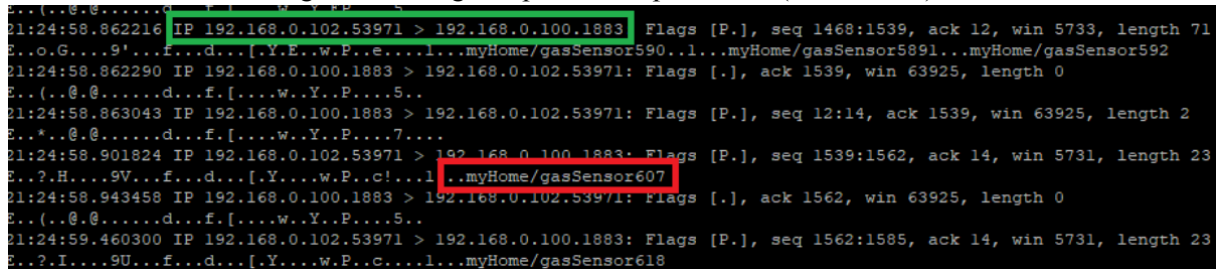


Proto	Recv-Q	Send-Q	Local Address	Foreign Address	State
tcp	0	0	0.0.0.0:1883	0.0.0.0:*	LISTEN
tcp	0	0	0.0.0.0:9001	0.0.0.0:*	LISTEN
tcp	0	0	0.0.0.0:microsoft-ds	0.0.0.0:*	LISTEN
tcp	0	0	0.0.0.0:ssh	0.0.0.0:*	LISTEN
tcp	0	0	0.0.0.0:netbios-ssn	0.0.0.0:*	LISTEN
tcp	0	0	:::0.100:1883	:::0.102:57480	ESTABLISHED
tcp	0	0	:::0.100:1883	:::0.100:40056	ESTABLISHED
tcp6	0	0	:::1883	:::*	LISTEN

Fonte: Elaboração Própria.

Na figura 6, observamos os dados dentro do retângulo verde, os equipamentos que estão se comunicando, IP 192.168.0.100(RPi) está recebendo informações do IP 192.168.0.102 (ESP32), os dados estão sendo publicados no tópico “myHome/gasSensor” (dentro do quadrado vermelho), o número ao lado do tópico corresponde ao valor do sensor. A conexão entre o microcontrolador e o servidor MQTT foi estabelecida com sucesso mediante a utilização da porta 1883, mas de acordo com a figura 6, nota-se que o sistema está vulnerável em relação aos dados dos pacotes enviados, e recebidos(caso seja da implementação do IoT).

Figura 6. Tráfego de pacotes na porta 1883(Comando 8).



```

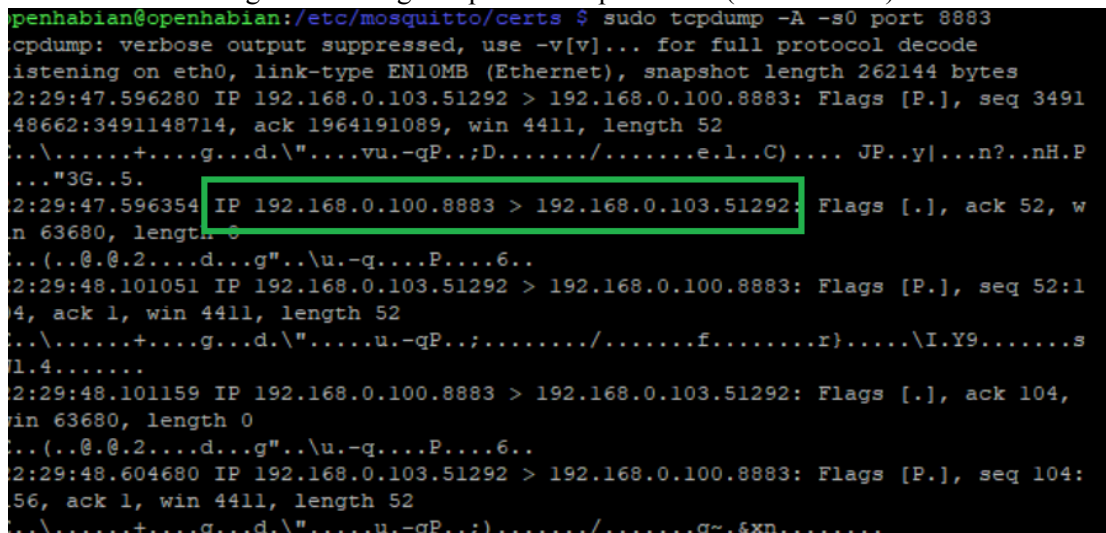
21:24:58.862216 IP 192.168.0.102.53971 > 192.168.0.100.1883: Flags [P.], seq 1468:1539, ack 12, win 5733, length 71
21:24:58.862290 IP 192.168.0.100.1883 > 192.168.0.102.53971: Flags [.], ack 1539, win 63925, length 0
21:24:58.863043 IP 192.168.0.100.1883 > 192.168.0.102.53971: Flags [P.], seq 12:14, ack 1539, win 63925, length 2
21:24:58.901824 IP 192.168.0.102.53971 > 192.168.0.100.1883: Flags [P.], seq 1539:1562, ack 14, win 5731, length 23
21:24:58.943458 IP 192.168.0.100.1883 > 192.168.0.102.53971: Flags [.], ack 1562, win 63925, length 0
21:24:59.460300 IP 192.168.0.102.53971 > 192.168.0.100.1883: Flags [P.], seq 1562:1585, ack 14, win 5731, length 23

```

Fonte: Elaboração Própria.

As figuras 6 e 7 mostram a mesma forma de visualização do tráfego de dados nas portas de comunicação do RPi, mas na figura 7, vemos os dados de forma diferente, pois se trata do tráfego após a implementação de todos os algoritmos(RSA, 3DES e protocolo de comunicação TLS), dentro do quadrado verde, os equipamentos que estão se comunicando, IP 192.168.0.100(RPi) está recebendo informações do IP 192.168.0.103(ESP32), os dados estão sendo publicados no tópico myHome/gasSensor mas, de forma diferente da figura 6, não é possível identificar o tópico de publicação e nem o conteúdo dos pacotes.

Figura 7. Tráfego de pacotes na porta 8883(Comando 22).



```

penhajian@openhabian:/etc/mosquitto/certs $ sudo tcpdump -A -s0 port 8883
tcpdump: verbose output suppressed, use -v[v]... for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), snapshot length 262144 bytes
2:29:47.596280 IP 192.168.0.103.51292 > 192.168.0.100.8883: Flags [P.], seq 3491
48662:3491148714, ack 1964191089, win 4411, length 52
... \.....+....g...d.\".....vu.-qP.;D...../.....e.l..C).... JP..y|...n?..nH.P
... "3G..5.
2:29:47.596354 IP 192.168.0.100.8883 > 192.168.0.103.51292: Flags [.], ack 52, w
n 63680, length 0
... (..@.@.2....d...g\".. \u.-q....P....6..
2:29:48.101051 IP 192.168.0.103.51292 > 192.168.0.100.8883: Flags [P.], seq 52:1
4, ack 1, win 4411, length 52
... \.....+....g...d.\".....u.-qP.;...../.....f.....r}..... \I.Y9.....s
1.4.....
2:29:48.101159 IP 192.168.0.100.8883 > 192.168.0.103.51292: Flags [.], ack 104,
in 63680, length 0
... (..@.@.2....d...g\".. \u.-q....P....6..
2:29:48.604680 IP 192.168.0.103.51292 > 192.168.0.100.8883: Flags [P.], seq 104:
56, ack 1, win 4411, length 52
... \.....+....g...d.\".....u.-qP.;...../.....g~.~xn.....

```

Fonte: Elaboração Própria.

Com base na sequência do *handshake* e na metodologia implementada, é nítido que as táticas foram implementadas de forma correta e tiveram a funcionalidade explorada. A figura 6, representa o tráfego dos dados na porta 1883 sem nenhum tipo de segurança, após gerar todos os certificados e chaves necessárias, assim como liberar as portas por meio do UFW e configurar o MQTT Broker no Openhab, obtemos os resultados da figura 7, mostrando o tráfego dos dados na porta 8883. Ao utilizar um protocolo TLS com chaves e certificados gerados pelo OpenSSL, os dados trocados entre o MQTT broker e o MQTT client são criptografados, o que reduz a possibilidade da interceptação e manipulação por parte de terceiros mal intencionados, garantindo a segurança, confidencialidade e integridade das informações transmitidas, além disso o protocolo TLS melhora a eficiência da

comunicação, pois a compressão de dados reduz o tamanho das mensagens transmitidas e, consequentemente, o tempo de transmissão. A utilização do tcpdump permite o monitoramento do tráfego de pacotes na rede, permitindo identificar possíveis falhas, vulnerabilidades ou problemas de comunicação, além de verificar se a criptografia está sendo aplicada corretamente.

O OpenHABian permite criar regras para automatizar tarefas, tornando o sistema mais eficiente e prático. Nesse caso, é possível criar regras para acionar dispositivos de acordo com as informações coletadas pelo sensor de gás, ao utilizar um protocolo padrão como o MQTT, é possível integrar diferentes dispositivos e sistemas em uma rede, permitindo que eles troquem informações e realizem ações em conjunto.

Outros benefícios desse IoT são a redução de custos, já que a utilização de sistemas de código aberto, como o Mosquitto e o OpenSSL, e a possibilidade de utilizar dispositivos de baixo custo, como o Raspberry Pi e o ESP32, podem reduzir significativamente os custos de implementação e manutenção do sistema.

## 5 CONSIDERAÇÕES FINAIS

Considerando o sucesso do projeto implementado, é possível concluir que a utilização de um protocolo seguro de comunicação como o TLS pode trazer diversos benefícios em termos de segurança e privacidade para a transmissão de dados em sistemas IoT. A utilização do MQTT como protocolo de comunicação também mostrou-se eficiente, permitindo a troca de mensagens de forma assíncrona. Além disso, a utilização do Raspberry Pi como centralizador do sistema, juntamente com o uso do OpenHABian como sistema operacional, permitiu uma fácil integração entre os diferentes componentes do projeto, bem como a disponibilidade de ferramentas de gerenciamento e monitoramento. A utilização do UFW para gerenciamento de portas, bem como a geração e uso de chaves e certificados criptográficos com o OpenSSL, confirmou a segurança da comunicação entre os dispositivos, evitando possíveis ataques externos, tais como:

- Interceptação de dados: como a comunicação entre o MQTT *broker* e o MQTT *client* é criptografada, um atacante não pode interceptar ou ler os dados transmitidos sem a chave de criptografia.
- Modificação de dados: uma vez que os dados são criptografados durante a transmissão, um atacante não pode modificar o conteúdo da mensagem sem ser detectado. Qualquer alteração na mensagem criptografada fará com que a mensagem não seja reconhecida.
- *Spoofing*: TLS pode ajudar a prevenir ataques de spoofing, onde um atacante tenta se passar por um dispositivo legítimo. Como os certificados digitais são usados na autenticação, o MQTT broker pode verificar se o MQTT *client* é realmente quem ele diz ser antes de permitir a conexão.
- Ataques de negação de serviço (DoS): o uso de TLS também pode ajudar a prevenir ataques DoS que tentam inundar o MQTT broker com solicitações falsas ou excessivas, pois o protocolo TLS tem recursos para limitar o número de conexões por segundo.
- *Sniffing*: o uso de TLS também ajuda a prevenir ataques de *sniffing*, onde um atacante tenta capturar informações sensíveis, como senhas, enquanto elas são transmitidas pela rede. Com a criptografia, mesmo que o atacante capture o tráfego de rede, ele não pode ler as informações criptografadas.

Por fim, a implementação do sensor de gás MQ2 no ESP32 permitiu uma coleta de dados eficiente e confiável, possibilitando o monitoramento da qualidade do ar em um determinado ambiente.

Assim, conclui-se que a utilização da arquitetura proposta, com a combinação de diferentes tecnologias e ferramentas, permitiu a criação de um sistema de monitoramento (IoT) da qualidade do ar de forma segura, eficiente e com baixo consumo de recursos.

## 6 LIMITAÇÕES

O sistema descrito, que utiliza MQTT com TLS, OpenSSL e UFW em um Raspberry Pi, juntamente com um ESP32 e um sensor de gás MQ2, oferece uma camada adicional de segurança para a transmissão de dados. No entanto, é importante reconhecer que, assim como qualquer sistema de segurança, existem limitações a serem consideradas.

Uma limitação desse sistema é a falta de um firewall físico na DMZ. Embora o firewall do sistema operacional (como o UFW neste caso) possa ajudar a bloquear tráfego indesejado, um firewall físico dedicado pode oferecer uma proteção mais robusta. Um firewall físico pode inspecionar o tráfego de rede em um nível mais profundo e tomar decisões de bloqueio com base em políticas específicas de segurança. No entanto, a implementação de um firewall físico também adiciona complexidade ao sistema e pode aumentar os custos de infraestrutura.

Outra limitação é que o sistema não oferece proteção contra ataques de força bruta. Se um atacante conseguir acesso às chaves de criptografia ou senhas, ele poderá tentar adivinhar a chave ou senha através de tentativa e erro. Para minimizar esse risco, é importante escolher senhas seguras e utilizar medidas adicionais de autenticação, como a autenticação em dois fatores.

Além disso, o sistema não oferece proteção contra ataques de phishing, que são aqueles em que um atacante tenta enganar o usuário final para obter informações confidenciais, como senhas ou chaves de criptografia. É importante que os usuários finais sejam treinados para reconhecer esse tipo de ataque e evitar clicar em links suspeitos ou fornecer informações confidenciais.

Por fim, deve-se ter em mente que a segurança de um sistema é uma questão contínua e em constante evolução. Embora o sistema descrito ofereça uma camada adicional de segurança para a transmissão de dados, é importante continuar monitorando e atualizando o sistema para garantir que ele esteja sempre protegido contra as mais recentes ameaças de segurança.

## REFERÊNCIAS

AKRAM, H.; KONSTANTAS, D.; MAHYOUB, M. A Comprehensive IoT Attacks Survey based on a Building-blocked Reference Model. **International Journal of Advanced Computer Science and Applications**, v. 9, n. 3, 2018.

APWG: Phishing Activity Trends Report Q4 2018. **Computer Fraud & Security**, v. 2019, n. 3, p. 4–4, jan. 2019.

ASHTON, KEVIN. That ‘Internet of Things’ Thing: In the real world, things matter more than ideas.. **RFID Journal** , [S. l.], p. 0-1, 11 maio 2009. Disponível em: <https://www.rfidjournal.com/that-internet-of-things-thing>. Acesso em: 8 abr. 2022.

BROWN, N; THOMSON, M. Triple DES (3DES) Cipher Suites for Transport Layer Security (TLS) Protocol. Internet Engineering Task Force (IETF), p. 1-34, 1 ago. 2018. Disponível em: <https://www.rfc-editor.org/rfc/rfc8422>. Acesso em: 2 abr. 2023.

DIERKS, T; RESCORLA , E. The Transport Layer Security (TLS) Protocol Version 1.2. Internet Engineering Task Force (IETF), [S. l.], p. 1-101, 1 ago. 2008. Disponível em: <https://tools.ietf.org/html/rfc5246>. Acesso em: 1 abr. 2023.

ECLIPSE FOUNDATION (org.). Eclipse Mosquitto: An open source MQTT broker. 2.0.14. [S. l.], 13 jan. 2022. Disponível em: <https://mosquitto.org/>. Acesso em: 2 abr. 2023.

GUALBERTO, Eder Souza et al. The Answer Is in the Text: Multi-Stage Methods for Phishing Detection Based on Feature Engineering. **IEEE**. Brasília, p. 0-18. 28 dez. 2020.

HASSIJA, Vikas; CHAMOLA, Vinay; SAXENA, Vikas; et al. **A Survey on IoT Security: Application Areas, Security Threats, and Solution Architectures**. IEEE Access, v. 7, p. 82721–82743, 2019. Disponível em: <<https://ieeexplore.ieee.org/document/8742551/>>. Acesso em: 7 maio 2022.

HOFFMAN, Andrew. **Web Application Security: Exploitation and Countermeasures for Modern Web Applications**. [S. l.: s. n.], 2020. 298 p.

KOLIAS, Constantinos; KAMBOURAKIS, Georgios; STAVROU, Angelos; et al. DDoS in the IoT: Mirai and Other Botnets. **Computer**, v. 50, n. 7, p. 80–84, 2017. Disponível em: <<http://ieeexplore.ieee.org/document/7971869/>>. Acesso em: 7 maio 2022.

LABORATÓRIO NACIONAL LAWRENCE BERKELEY (org.). TPCDUMP. 4.99.1. [S. l.], 16 mar. 2022. Disponível em: <https://www.tcpdump.org/>. Acesso em: 2 abr. 2023.

OPENHAB. 3.2.0. [S. l.], 2022. Disponível em: <https://www.openhab.org/>. Acesso em: 11 jun. 2022.

OPENSSL (org.). OpenSSL. 1.1.1. [S. l.], 23 mar. 2022. Disponível em: <https://www.openssl.org/>. Acesso em: 2 abr. 2023.

RASPBERRY PI FOUNDATION (org.). Raspberry Pi Debian. 4.99.1. [S. l.], 2019. Disponível em: <https://raspi.debian.net/>. Acesso em: 2 abr. 2023.

RAZZAQUE, M. A. et al. Middleware for Internet of Things: A Survey. **IEEE Internet of Things Journal**, v. 3, n. 1, p. 70–95, fev. 2016.

RIVEST, R; SHAMIR, A; ADLEMAN, L. A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. Communications of the ACM. [S. l.], p. 120-126, 1 fev. 1978. Disponível em: <https://people.csail.mit.edu/rivest/Rsapaper.pdf>. Acesso em: 2 abr. 2023.

SHIMONSKI, Robert. **Building DMZs for Enterprise Networks**: Design, Deploy, and Maintain a Secure DMZ on Your Network. [S. l.]: Syngress, 2003. 744 p.

SOWAH, R. A. et al. Design of a Secure Wireless Home Automation System with an Open Home Automation Bus (OpenHAB 2) Framework. **Journal of Sensors**, v. 2020, p. 1–22, 30 out. 2020.

TATHAM, Simon. PuTTY. 0.77. [S. l.], 2022. Disponível em: <https://www.putty.org/>. Acesso em: 11 jun. 2022.

THE AMBIENT TEAM. **Around the world in smart home trends**: Are there more smart homes in India or Australia? Find out here. The Ambient, 31 jul. 2018. Disponível em: <https://www.the-ambient.com/features/around-the-world-smart-home-stats-832>. Acesso em: 6 maio 2022.

UNCOMPLICATED Firewall: UFW. 0.36. [S. l.], 13 dez. 2007. Disponível em: <https://wiki.ubuntu.com/UncomplicatedFirewall>. Acesso em: 2 abr. 2023.

WIRESHARK. 3.6.5. [S. l.], 2022. Disponível em: <https://www.wireshark.org/>. Acesso em: 11 jun. 2022.



## APÊNDICE A - CÓDIGO IMPLEMENTADO NO MICROCONTROLADOR.

```
1 #include <WiFi.h>
2 #include <PubSubClient.h>
3 #include <Wire.h>
4 #include <WiFiClientSecure.h>
5
6 const char* ssid = "nome_da_rede";
7 const char* password = "senha_da_rede";
8 const char* mqtt_server = "dominio_do_RPi";
9 const int mqttPort = 8883;
10 const char* mqttUser = "openhabian";
11 const char* mqttPassword = "openhabian";
12 // inicio do certificado
13 const char* cert = "-----BEGIN CERTIFICATE-----\n"
14 "MIIDVTCCAj2gAwIBAgIUBxvuDeL5L9scIMHBCrbfZJeIuGAwDQYJKoZIhvcNAQEL"
15 "BQAwOjELMAkGA1UEBhMCQlIxDDAKBgNVBAoMA0lPVDEPMA0GA1UECwwGY2xpZW50"
16 "MQwwCgYDVQQDDAN0Y2MwHhcNMjMwMzI4MTg1NjA3WhcNMjgwNDZjMTg1NjA3WjA6"
17 "MQswCQYDVQQGEwJCUjEMMAoGA1UECgwDSU9UMQ8wDQYDVQQLEDAZjbGllbnQxDDAK"
18 "BgNVBAMMA3RjYzCCASIwDQYJKoZIhvcNAQEBBQADggEPADCCAQoCggEBAKaS70v3"
19 "ABnocMqEV3TBEJrcnko+L1qyhGhNkMKBSTTGMne5+OA/c3w/fjPKhKT8TjMidkOI"
20 "rledrM8CAmYlHgdDJT69dbqED0Wt6AzT5NYldroYlriINJLsArt+aX7BJoGRaY8N"
21 "PWFZf4bAC9JWF2oLU5Z6GNQq/BnzG3u9EcPSSulfIvQXcd4kiuoDJiq8FUjBkXmh"
22 "ul9s+TIMmC8aaxyfOJQz3DtqOEftttlNmMKgCloPaz99gz/LLrDlNx6J9KWrHn5V"
23 "ON3OpLbLBTbUdMTW7cDiJH3xZsRUxDxME7Mu3ENeKnNEq74Bh1CxxUvpAawr6neZ"
24 "MvEhbKALtUyb8H0CAwEAaANTMFewHQYDVR0OBBYEF0iyhY83eaoLXetsjTvtXpvS"
25 "e8skMB8GA1UdIwQYMBaAF0iyhY83eaoLXetsjTvtXpvSe8skMA8GA1UdEwEB/wQF"
26 "MAMBAf8wDQYJKoZIhvcNAQELBQADggEBAKL4rZ2S/fwoULvZiH1QEYqYJIOqzQoqc"
27 "tQWDgzGsz9oLgjs0ZW5Jt9cve2yM1JpmBk0kQBga3wNhjnSLQusLRgGBtwTEdePl"
28 "MeaV6FPRsQRas5wD2B9qjWZiwyjFZUI3MXhp2tPLbMnYTwtRQ90nBva4cYQnSNZ"
29 "bgw6wEHc+ZKSyFluC4f/dKShcYaTAlceyJ7D1ooj5JBDNmzY1Dg+Jc0bLHgqmYOA"
30 "sx16MgMZQRWwSfgp7AHOCAeQUy0sxqL8/fsZkENJ5ikkfifQjHsyPH2pujqVxVH1"
31 "wKUy9qWL0ojdM1AJEwQCW7RSmvXzLhAfi4mEWlDV68JKbPa0wMPWZok="
32 "-----END CERTIFICATE-----\n";
33 // final do certificado
34 WiFiClientSecure espClient;
35 PubSubClient client(espClient);
36 long lastMsg = 0;
37 char msg[50];
38 int value = 0;
39
40 int gas = 34;
41
42 void setup() {
43   Serial.begin(115200);
44
45   setup_wifi();
46   client.setServer(mqtt_server, mqttPort);
47   espClient.setCACert(cert);
48 }
49
50
51 void setup_wifi() {
52   delay(10);
53
54   Serial.println();
55   Serial.print("Conectando a rede ");
56   Serial.println(ssid);
57
```

```

58  WiFi.begin(ssid, password);
59
60  while (WiFi.status() != WL_CONNECTED) {
61      delay(500);
62      Serial.print(".");
63  }
64
65  Serial.println("");
66  Serial.println("WiFi conectado");
67  Serial.println("Endereço IP: ");
68  Serial.println(WiFi.localIP());
69 }
70
71 void reconnect() {
72     // Conexão ao MQTT
73     while (!client.connected()) {
74         Serial.print("Conectando ao MQTT Broker...");
75         // Attempt to connect
76         if (client.connect("ESP32",mqttUser,mqttPassword)) {
77             Serial.println("Conectado");
78             // Subscribe
79             client.subscribe("esp32/output");
80         } else {
81             Serial.print("Erro, rc=");
82             Serial.print(client.state());
83             Serial.println("Tentando novamente em 5 segundos...");
84             delay(5000);
85         }
86     }
87 }
88 void loop() {
89     if (!client.connected()) {
90         reconnect();
91     }
92     client.loop();
93
94     int gasValue = analogRead(gas);
95     // Publica o valor do sensor
96     String gasValueStr = String(gasValue);
97     client.publish("myHome/gasSensor", gasValueStr.c_str(),
98 gasValueStr.length());
99     Serial.println(gasValueStr);
100    Serial.println(analogRead(gas));
101    delay(500);
102 }

```