PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO GRANDE DO SUL
ESCOLA POLITÉCNICA
ENGENHARIA DE COMPUTAÇÃO

WILLIAM MATHEUS DE OLIVEIRA RODRIGUES

PROJETO E IMPLEMENTAÇÃO INTEGRADA DE HARDWARE E APLICATIVO MOBILE PARA GERENCIAMENTO E CONTROLE SEGURO DE DISPOSITIVOS DE IOT

Desenvolvimento de uma Plataforma IoT com Camadas de Proteção e Criptografia Visando a Segurança da Informação

Porto Alegre, Rio Grande do Sul, Brasil
2024

WILLIAM MATHEUS DE OLIVEIRA RODRIGUES

PROJETO E IMPLEMENTAÇÃO INTEGRADA DE HARDWARE E APLICATIVO MOBILE PARA GERENCIAMENTO E CONTROLE SEGURO DE DISPOSITIVOS DE IOT: DESENVOLVIMENTO DE UMA PLATAFORMA IOT COM CAMADAS DE PROTEÇÃO E CRIPTOGRAFIA VISANDO A SEGURANÇA DA INFORMAÇÃO

Trabalho de conclusão de curso de graduação apresentado na Escola Politécnica da Pontifícia Universidade Católica do Rio Grande do Sul, como requisito parcial para obtenção do grau de Engenheiro de Computação.

Orientador: Prof. Msc. Anderson Royes Terroso

Porto Alegre, Rio Grande do Sul, Brasil
2024

WILLIAM MATHEUS DE OLIVEIRA RODRIGUES

PROJETO E IMPLEMENTAÇÃO INTEGRADA DE HARDWARE E APLICATIVO MOBILE PARA GERENCIAMENTO E CONTROLE SEGURO DE DISPOSITIVOS DE IOT: DESENVOLVIMENTO DE UMA PLATAFORMA IOT COM CAMADAS DE PROTEÇÃO E CRIPTOGRAFIA VISANDO A SEGURANÇA DA INFORMAÇÃO

Trabalho de conclusão de curso de graduação apresentado na Escola Politécnica da Pontifícia Universidade Católica do Rio Grande do Sul, como requisito parcial para obtenção do grau de Engenheiro de Computação.

Aprovada em _____ de _____ de _____.

BANCA EXAMINADORA:

Nome do Professor

_____

Nome do Professor

_____

Nome do Professor

_____

Dedico este trabalho, como forma de homenagem, à minha mãe, Neida, minha dinda, Neusa e minha namorada, Eduarda, que estiveram comigo e me apoiaram ao longo de toda a jornada de construção deste Trabalho de Conclusão de Curso.

RESUMO

O avanço da Internet das Coisas (IoT) trouxe consigo inovações tecnológicas significativas, mas também ampliou os desafios relacionados à segurança da informação em dispositivos conectados. Este trabalho de conclusão de curso apresenta o desenvolvimento de uma plataforma integrada de hardware e software para o gerenciamento e controle seguro de dispositivos IoT, com foco em automação residencial. A solução foi projetada utilizando microcontroladores ESP32 e protocolos de comunicação MQTT, aliados a práticas robustas de segurança da informação, como criptografia AES-256 no modo CBC e autenticação segura. A aplicação mobile desenvolvida permite o monitoramento e o controle de dispositivos, como sensores de temperatura e umidade e fechaduras magnéticas, garantindo a proteção dos dados sensíveis transmitidos entre os dispositivos. O sistema adota o conceito de *Security by Design*, implementando camadas de proteção que asseguram a confidencialidade, integridade e autenticidade das comunicações. Os resultados confirmaram a viabilidade da solução proposta, com desempenho satisfatório nas simulações de comunicação e resistência a ataques cibernéticos simulados, como interceptação de dados. Os principais desafios enfrentados incluíram a sincronização dos algoritmos criptográficos entre plataformas e a integração de diferentes tecnologias, como hardware IoT e aplicativos móveis. Conclui-se que a abordagem utilizada proporciona uma solução eficiente e segura para sistemas IoT em ambientes residenciais, com potencial para ser expandida a outros cenários de automação. Como trabalhos futuros, sugere-se a implementação de suporte a múltiplos dispositivos simultâneos e a integração com outras tecnologias de comunicação IoT.

Palavras-chave: IoT. Segurança da Informação. Criptografia AES. MQTT. ESP32.

ABSTRACT

The advancement of the Internet of Things (IoT) has brought with it significant technological innovations, but it has also increased the challenges related to information security in connected devices. This final project presents the development of an integrated hardware and software platform for the secure management and control of IoT devices, with a focus on home automation. The solution was designed using ESP32 microcontrollers and MQTT communication protocols, combined with robust information security practices, such as AES-256 encryption in CBC mode and secure authentication. The mobile application developed allows the monitoring and control of devices, such as temperature and humidity sensors and magnetic locks, ensuring the protection of sensitive data transmitted between devices. The system adopts the concept of Security by Design, implementing layers of protection that ensure the confidentiality, integrity and authenticity of communications. The results confirmed the viability of the proposed solution, with satisfactory performance in communication simulations and resistance to simulated cyber attacks, such as data interception. The main challenges faced included the synchronization of cryptographic algorithms between platforms and the integration of different technologies, such as IoT hardware and mobile applications. It is concluded that the approach used provides an efficient and secure solution for IoT systems in residential environments, with potential to be expanded to other automation scenarios. As future work, it is suggested to implement support for multiple simultaneous devices and integration with other IoT communication technologies.

Keywords: IoT. Information Security. AES Encryption. MQTT. ESP32.

LISTA DE ILUSTRAÇÕES

SUMÁRIO

1 INTRODUÇÃO

A Internet da Coisas (do termo em inglês *Internet of Things,* ou simplesmente IoT), tem revolucionado a forma como interagimos com o ambiente ao nosso redor ao trazer conectividade a dispositivos antes isolados, como eletrodomésticos, sensores, controles de iluminação e fechaduras. Essa conectividade proporciona diversos benefícios, como um melhor e mais prático controle, uma maior automação e conveniência para as pessoas que os utilizam, especialmente em ambientes domésticos e industriais.

Segundo estatísticas levantadas pela agência global de dados *Statista (*2022*)* [5]*,* no Brasil, o número de dispositivos IoT's vem aumentando constantemente ao longo dos anos, como comparação, em 2018 esse número era de 138.2 milhões e em 2023 o número de dispositivos chegou a 415.7 milhões, um aumento de mais de 200%. Outras estimativas indicam que, até 2029, a receita anual gerada pelo mercado de IoT no Brasil atinja um pico de 26.54 bilhões de dólares, valor que também vêm em uma crescente anual desde 2021, onde a receita no ano foi de 10.04 bilhões de dólares; essa tendência de aumento é global, pois o número de usuários de IoT domésticos têm crescido constantemente em todo o mundo, em 2022 esse valor era de aproximadamente 307.8 milhões de usuários - um aumento de 88% em relação a 2018 – e as previsões apontam para um expressivo aumento para 672.6 milhões de usuários até o ano de 2027 (acréscimo de 119% em relação a 2022) [5]. No entanto, junto com esses avanços e com a proliferação e aumento do uso desses dispositivos, surgem também preocupações crescentes com a segurança da informação envolvida nesses sistemas, pois os dispositivos IoT são alvos atrativos para ataques cibernéticos devido à diversos fatores [2], tais como:

a. Sua superfície de ataque ampliada, pois, como os dispositivos IoT estão conectados à internet e muitas vezes conectados a outros dispositivos, o número de pontos vulneráveis aumenta [6];
b. Podem apresentar autenticação fraca e possuírem senha padrão de fábrica, onde muitos *hardwares* IoT são lançados no mercado com credenciais padrão fáceis de serem obtidas por *hackers,* pois muitos usuários não mudam essas configurações [7];

c. A coleta de quantidades significativas de dados pessoais ou sensíveis, pois quando a criptografia ou outros mecanismos de proteção dos dados são inadequados, essas informações sensíveis podem ser expostas a invasores [2];
d. Sistemas no contexto IoT são fixos e autônomos, diferentemente de ambientes tradicionais de TI onde os sistemas são separados uns dos outros por segurança física adequada, tornando os sistemas IoT mais propensos e vulneráveis a ataques [2].

As ameaças à segurança na IoT são diversas e variam desde invasões à privacidade dos usuários, com o acesso a dados sensíveis gerados ou coletados, até ataques mais críticos, como a manipulação de sistemas de controle de infraestrutura. Essas vulnerabilidades decorrem de alguns fatores, entre eles estão a falta de padronização e fiscalização na segurança dos dispositivos e a crescente complexidade e proliferação exponencial dos ecossistemas IoT. Como discutido no artigo por Hubert; Kaledio (2024) [3], garantir a privacidade e a segurança em dispositivos IoT é um grande desafio que exige soluções robustas, envolvendo tanto criptografia quanto o gerenciamento seguro de dispositivos e dados.

1.1 OBJETIVOS

Neste contexto, o presente trabalho desenvolveu uma plataforma integrada de hardwares IoT, a partir de microcontrolador comercial, simulando principalmente dispositivos residenciais, e um aplicativo *mobile* para o gerenciamento e controle seguro dessa rede de dispositivos IoT, aplicando conceitos e boas práticas de segurança. A plataforma foi projetada e desenvolvida com uma arquitetura de segurança em múltiplas camadas, incluindo uma criptografia de ponta a ponta entre dados sensíveis e autenticação robusta, como o uso de biometria na aplicação. Essa abordagem é alinhada com os desafios discutidos por Satyanarayana et al. (2023) [1], que destacam a importância de algoritmos de criptografia otimizados para IoT, como o AES (*Advanced Encryption Standard*) modificado, que oferecem maior proteção sem comprometer o desempenho do sistema.

Importante mencionar que, alinhado com o objetivo principal do trabalho, o controle de acesso seguro, a transmissão segura de informações entre dispositivos e o monitoramento contínuo foram implementados para prevenir ataques de

intermediários e a interceptação de dados. Assim, o objetivo do trabalho consiste em desenvolver um sistema que busca atender à crescente demanda por soluções de IoT que combinem funcionalidade e segurança, oferecendo uma alternativa viável para cenários de automação residencial e industrial. O desenvolvimento deste trabalho tem como enfoque este tópico tão relevante e atual que é a Internet das Coisas, e não apenas a integração hardware e software para IoT, mas também a relevância da sua proteção contra ameaças cibernéticas, contribuindo para o avanço da segurança na área de IoT e seguindo os preceitos da Segurança da Informação, principalmente no que diz respeito à Confidencialidade, Integridade e Autenticidade dos dados.

2 REFERENCIAL TEÓRICO

Este capítulo visa apresentar os conceitos teóricos que fundamentam e embasam o desenvolvimento do trabalho, divididos em tópicos essenciais. Inicialmente, são descritas as principais tecnologias que sustentam a Internet das Coisas, abordando sensores, protocolos de comunicação e sistemas embarcados. Em seguida, discute-se a segurança em IoT, com destaque para a norma internacional ISO/IEC 27400:2022 [10] e os protocolos seguros utilizados para proteger os dispositivos e as comunicações.

Também são explorados os fundamentos de criptografia e autenticação, com ênfase no uso do algoritmo AES-256 [14] aplicado à dispositivos embarcados, e os desafios relacionados à implementação de medidas de proteção. Ademais, é destacado o processo de desenvolvimento de hardware IoT e o desenvolvimento de aplicativo mobile para sistemas IoT.

Este referencial organiza os elementos necessários para o entendimento e desenvolvimento do trabalho, proporcionando uma base técnica para as etapas seguintes.

2.1 INTERNET DAS COISAS (IOT)

A Internet das Coisas refere-se à interconexão de dispositivos físicos capazes de coletar, processar e transmitir dados pela internet. Desde a sua concepção na década de 1990, a IoT evoluiu de simples sensores conectados à uma vasta rede de dispositivos inteligentes, presentes em setores como saúde, agricultura, transportes e automação residencial. Essa evolução, dentre outros fatores, está ligada à crescente miniaturização da eletrônica, ao aprimoramento de tecnologias de comunicação e à ampla disponibilidade de conectividade de alta velocidade, como 4G e 5G [4].

2.1.1 Principais Tecnologias Envolvidas

A IoT é construída e integrada com várias tecnologias-chave para o seu funcionamento, dentre as principais para este projeto se encontram:

a) Microcontroladores: Dispositivos de baixo custo e baixa potência, amplamente utilizados na IoT devido à sua capacidade de comunicação sem fio (*Wi-Fi* e *Bluetooth*) e processamento integrado [1].
b) Protocolos de Comunicação: Um dos protocolos amplamente utilizados e que contribuiu para a expansão e disseminação da IoT foi o protocolo MQTT (*Message Queuing Telemetry Transport*), que fornece uma comunicação eficiente entre dispositivos IoT e servidores, ou *brokers*, essencial para a transmissão e captação de dados entre dispositivos [20].
c) Sensores e Atuadores: Sensores que coletam informações e dados do ambiente, enquanto atuadores realizam ações físicas com base em comandos recebidos [4].

2.2 SEGURANÇA DA INFORMAÇÃO EM IOT

Os desafios de segurança em dispositivos IoT são amplos, devido à conectividade constante e à falta de atualizações regulares, também pela ausência de normas de segurança que regulamentam o setor, bem como de suas fiscalizações ou não obrigatoriedade em alguns países. Muitos dispositivos, por exemplo, são lançados com senhas padrão, facilitando ataques bem-sucedidos aos mesmos. Outros dispositivos são limitados em termos de processamento, o que os torna inadequados para o uso de algoritmos de criptografia mais robustos [1]. Além disso, as vulnerabilidades em firmware e a ausência de práticas seguras de desenvolvimento tornam a proteção de alguns dispositivos IoT uma tarefa complexa [2].

2.2.1 Norma Internacional de Segurança e Privacidade em IoT

Recentemente, mais especificamente em junho de 2022, a ISO (*International Organization for Standardization*), juntamente com a IEC (*International Electrotechnical Commission*), publicou uma nova norma internacional voltada a segurança cibernética para dispositivos IoT, a ISO/IEC 27400:2022 [10]. Devido ao rápido crescimento da Internet das Coisas, surgiram desafios significativos relacionados à segurança e à privacidade dos dispositivos conectados, por isso, a fim

de mitigar esses riscos, a norma ISO/IEC 27400:2022 foi publicada. Nela são fornecidas diretrizes abrangentes para a segurança cibernética e privacidade no contexto de soluções IoT. Esta norma tem como objetivo ajudar as partes interessadas — desenvolvedores, fornecedores de serviços e usuários finais — a implementar práticas de segurança robustas em todo o ciclo de vida dos dispositivos IoT [8].

Alguns dos principais pontos abordados na norma são:

a. Política de Segurança IoT: A norma enfatiza a necessidade de definir políticas claras para proteger dispositivos IoT, estabelecendo requisitos e procedimentos que garantam a segurança desde o desenvolvimento até o uso final.
b. Gestão de Ativos: Para assegurar a proteção dos dispositivos conectados, a norma trata da gestão dos ativos IoT, incluindo desde a configuração segura até o descarte apropriado dos dispositivos. Ela também sugere o aprendizado com incidentes de segurança para melhorar continuamente as práticas de proteção.
c. Engenharia de Sistemas Seguros: O documento orienta as práticas de engenharia de sistemas para IoT, abordando o design e o desenvolvimento de dispositivos e seus softwares associados de forma a garantir uma segurança sólida. Isso inclui a necessidade de considerar atualizações e correções contínuas para mitigar novas ameaças.
d. Autenticação e Controle de Acesso: Um dos pilares da segurança em IoT é a autenticação adequada de dispositivos e usuários. A ISO/IEC 27400:2022 sugere mecanismos robustos para prevenir o acesso não autorizado, como autenticação multifator (MFA) e biometria.
e. Proteção à Privacidade: A norma segue o princípio da "privacidade por design", ou seja, a proteção de dados é incorporada desde as fases iniciais de desenvolvimento do dispositivo. Esse ponto é crucial para garantir que informações sensíveis dos usuários sejam tratadas com confidencialidade durante todo o ciclo de vida do produto.
f. Criptografia: Um aspecto essencial abordado pela ISO/IEC 27400:2022 é a utilização de criptografia para proteger tanto a comunicação entre dispositivos IoT quanto os dados transmitidos. A norma sugere o uso de algoritmos robustos, como o AES (Advanced Encryption Standard), para garantir que as informações trocadas estejam seguras.

g. Gerenciamento de Vulnerabilidades: A gestão proativa de vulnerabilidades é outro ponto central. A norma recomenda processos de monitoramento contínuo, identificação e mitigação de vulnerabilidades em dispositivos IoT, de modo a responder rapidamente a possíveis ameaças.
h. Proteção de Logs e Monitoramento Contínuo: A ISO/IEC 27400:2022 orienta sobre a proteção de logs gerados pelos sistemas IoT, que podem conter informações valiosas para a identificação de incidentes. Além disso, o monitoramento contínuo do comportamento dos dispositivos é necessário para garantir que ameaças sejam detectadas e neutralizadas em tempo hábil.

Essas diretrizes são aplicáveis tanto para dispositivos em ambientes domésticos quanto para sistemas mais complexos em contextos industriais, assegurando que a confiança dos usuários seja mantida e que as soluções IoT continuem a crescer de maneira segura. Importante mencionar que a ISO/IEC 27400:2022 também se alinha com outras normas internacionais de segurança da informação, como a ISO/IEC 27000 (Gestão de Segurança da Informação) e a ISO/IEC 27701 (Gerenciamento de Informações de Privacidade), complementando a abordagem e importância para segurança cibernética [11].

Essa norma serve como uma estrutura sólida para mitigar os riscos cada vez mais sofisticados que ameaçam o ecossistema de IoT, oferecendo diretrizes que ajudam a proteger tanto os dados quanto a privacidade dos usuários [9], porém, principalmente por ela ser consideravelmente recente, muitas empresas fabricantes de dispositivos IoT não seguem à risca, de fato, suas recomendações, devido à ausência de uma fiscalização efetiva em todos os dispositivos utilizados por consumidores finais.

2.2.2 Técnicas de Proteção e Protocolos de Comunicação Segura

Para proteger dispositivos IoT, seus dados e sua comunicação, uma abordagem multicamadas é necessária, combinando:

a. Criptografia de Dados: Para garantir que as informações transmitidas entre dispositivos e servidores permaneçam seguras.

b. Autenticação Forte: Como utilizando-se de MFA (Autenticação Multifator) e biometria, que são essenciais para prevenir acessos não autorizados [2].
c. Monitoramento Contínuo e Detecção de Intrusões: Implementação de sistemas de monitoramento de tráfego para identificar comportamentos anômalos e ataques.

O uso de protocolos de comunicação seguros, como TLS (*Transport Layer Security*), é essencial para proteger a troca de informações [13]. O protocolo MQTT é amplamente utilizado em IoT devido à sua eficiência e confiabilidade, e quando combinado com TLS, proporciona uma comunicação segura.

2.3 CRIPTOGRAFIA E AUTENTICAÇÃO

A criptografia é um dos pilares da segurança da informação. No contexto de IoT, algoritmos de criptografia precisam ser leves e eficientes, devido às limitações de hardware dos dispositivos. O AES - *Advanced Encryption Standard* -, é um dos algoritmos mais utilizados, porém, a necessidade de adaptação para dispositivos IoT deu origem a variações mais leves, como o AES modificado, que oferece melhor desempenho e segurança, conforme apresentado por Satyanarayana et al. (2023) [1].

Além da criptografia, a autenticação segura é fundamental em IoT. A MFA (Autenticação Multifator), que combina pelo menos dois fatores de verificação (como senha e biometria), é amplamente utilizada para evitar acessos não autorizados [12]. Métodos biométricos, como o reconhecimento de impressões digitais e faciais, são considerados seguros e eficazes, tendo o uso recomendado em dispositivos móveis que controlam dispositivos IoT.

2.3.1 Contextualização sobre Aspectos da Criptografia AES

O *Advanced Encryption Standard* (AES) é, atualmente, o cifrador simétrico mais amplamente utilizado em sistemas de segurança. Embora o termo "*Standard*" em seu nome se refira inicialmente às aplicações do governo dos Estados Unidos, o AES se tornou obrigatório em diversos padrões industriais e é amplamente empregado em sistemas comerciais [13]. Entre os padrões que incluem o AES estão o IPsec (padrão

de segurança para Internet), o TLS (protocolo de segurança para comunicações na web), o padrão de criptografia para Wi-Fi IEEE 802.11i, o protocolo de rede SSH (Secure Shell), a tecnologia de comunicação Skype, além de inúmeros produtos de segurança ao redor do mundo. Até o momento, não foram identificados ataques práticos contra o AES que superem a força bruta, o que reforça sua robustez e confiabilidade em proteger dados sensíveis em uma variedade de aplicações [13].

Para compreender o processo de criptografia e descriptografia utilizando o algoritmo AES, especialmente na implementação da extensão utilizada na plataforma MIT App Inventor para construção do aplicativo *mobile* [22], é necessário abordar os seguintes conceitos fundamentais:

a. Geração de *Salt:* O *Salt* é um valor aleatório gerado para tornar a criptografia mais segura, adicionando entropia ao processo de derivação da chave. Ele impede que ataques baseados em tabelas de *hash* pré-calculadas sejam eficazes, mesmo que a senha utilizada seja fraca ou repetida. Na extensão *MareshaAES* [22], o Salt é incorporado durante a derivação da chave, utilizando técnicas como PBKDF2 com múltiplas iterações para aumentar a resistência a ataques [16].
b. *Padding*: Os algoritmos de criptografia como o AES operam em blocos de tamanho fixo. Quando os dados a serem criptografados não têm o tamanho exato de múltiplos do bloco (16 bytes para o AES), é necessário preenchê-los. Este processo é conhecido como Padding, e o método mais utilizado é o PKCS5Padding [16]. O Padding adiciona bytes específicos ao final do texto para completar o bloco, permitindo que o algoritmo funcione corretamente. Durante a descriptografia, o Padding é identificado e removido para recuperar os dados originais.
c. Derivação de Chave: A derivação de chave é o processo de transformar uma senha ou frase de segurança em uma chave criptográfica segura. Neste projeto utiliza-se o método *PBKDF2WithHmacSHA256* [16], que aplica a função *hash* SHA-256 iterativamente para gerar uma chave de 256 bits a partir da senha e do Salt. Esse processo melhora a segurança, dificultando

ataques de força bruta devido às milhares de iterações necessárias para gerar a chave.

d. Base64 *Coding:* O Base64 é um esquema de codificação que converte dados binários em texto ASCII [23]. Após a criptografia com o AES, os dados criptografados são representados em Base64 para facilitar o armazenamento ou transmissão em sistemas que lidam apenas com texto. Codificação (Encoding): Transforma os dados binários criptografados em uma string Base64. Decodificação (Decoding): Converte a string Base64 de volta para o formato binário antes da descriptografia.
e. *Hashing*: É um processo que transforma uma entrada, um dado qualquer, em uma sequência de caracteres de tamanho fixo, conhecida como *hash*. Este processo utiliza uma função matemática (função *hash)*, que é determinística e unidirecional. Qualquer mínima alteração na entrada, resulta em um *hash* completamente diferente. Como exemplo, o algoritmo PBKDF2 utiliza HMAC-SHA-256 para gerar chaves criptográficas seguras a partir de senhas, adicionando um nível extra de segurança ao sistema [16].

2.4 DESENVOLVIMENTO DE HARDWARE IOT

O desenvolvimento de hardware IoT envolve a escolha de microcontroladores eficientes e confiáveis. O ESP32, por exemplo, é amplamente utilizado devido à sua capacidade de processamento integrado e conectividade Wi-Fi/Bluetooth e por ser altamente versátil, além de suportar criptografia, tornando-o uma excelente escolha para projetos IoT voltados à segurança [18].

A integração de hardware com software em IoT exige um design modular, onde o firmware do dispositivo é capaz de se comunicar com a aplicação de controle, muitas vezes utilizando APIs específicas ou o protocolo de mensagens MQTT. A combinação de firmware robusto, com algoritmos de segurança embutidos - e softwares mobile seguros - garante que os dados trocados entre dispositivos e aplicativos sejam confiáveis e protegidos [3].

2.5 DESENVOLVIMENTO DE APLICATIVO MOBILE

O desenvolvimento de aplicativos móveis para IoT requer o uso de ferramentas que permitam uma fácil integração com dispositivos. Uma das plataformas mais acessíveis para esse fim é o MIT App Inventor [24], que facilita a criação de aplicativos com funcionalidades de controle e monitoramento de dispositivos IoT. Ele permite a implementação de recursos de segurança, como autenticação de usuários e controle de acesso, com uma interface gráfica amigável.

A integração entre aplicativos móveis e dispositivos IoT apresenta desafios, como a latência de rede e a necessidade de garantir a segurança na comunicação entre o aplicativo e os dispositivos. Proteger essa comunicação de ataques de intermediários, como o *man-in-the-middle* (MitM), exige o uso de protocolos seguros e criptografia ponta a ponta [3].

3 METODOLOGIA

A metodologia adotada neste trabalho foi estruturada de forma a garantir uma abordagem integrada entre hardware e software, com foco na segurança da informação. O projeto foi planejado e executado em etapas bem definidas, abrangendo o estudo inicial das tecnologias necessárias, o desenvolvimento de cada componente do sistema de forma modular e a validação de sua integração. Cada etapa seguiu um cronograma pré-estabelecido, no qual práticas de segurança, como criptografia e autenticação por biometria, foram incorporadas desde a concepção inicial.

A escolha das ferramentas e tecnologias foi guiada pelos requisitos de desempenho, compatibilidade e disponibilidade. Durante esta etapa, foram realizados estudos aprofundados sobre o protocolo MQTT, a plataforma App Inventor, a criptografia AES-256 com modo de operação CBC [15], e as funcionalidades do microcontrolador ESP32. A aplicação mobile foi projetada para ser o ponto central de controle e monitoramento, integrando todos os dispositivos IoT do sistema. Esta metodologia buscou assegurar que cada etapa contribuísse para um sistema eficiente, seguro e funcional, capaz de atender aos objetivos propostos pelo trabalho.

3.1 PLANEJAMENTO DO PROJETO

O projeto foi estruturado em fases, desde o estudo e entendimento inicial das tecnologias que serão utilizadas até a implementação prática e testes de hardware e software para verificar a viabilidade do projeto como um todo, em módulos, seguindo um cronograma de atividades que incluiu a integração de práticas de segurança da informação em todo o escopo. Adotou-se uma abordagem de desenvolvimento para sistemas embarcados IoT, utilizando-se como hardware principal o microcontrolador ESP32 e como software um aplicativo mobile para controle e monitoramento do sistema IoT. Desde a concepção do projeto, a segurança foi tratada como um aspecto central, aplicando-se o conceito de *Security by Design* para garantir que todos os componentes do sistema possuíssem camadas de proteção adequadas [2].

Inicialmente, foi realizado um estudo aprofundado das tecnologias necessárias e suficientes para o desenvolvimento do projeto, incluindo a plataforma web MIT App Inventor para o desenvolvimento completo do aplicativo *mobile*, o protocolo MQTT [19] para a comunicação entre os dispositivos, a documentação e utilização das bibliotecas de criptografia utilizadas no microcontrolador, e o funcionamento do microcontrolador ESP32 com os sensores e atuadores utilizados no TCC. A escolha do protocolo MQTT foi feita devido à sua praticidade, leveza e eficiência para comunicação em sistemas de IoT, considerando as limitações de banda e recursos de processamento nos dispositivos. Além disso, o ponto principal de pesquisa ao longo da seção foi a investigação do funcionamento e aplicação do algoritmo criptográfico AES-256 com o modo de operação CBC [15] especificamente para a ESP32, vital para o sucesso projeto, podendo ser utilizado e processado em tempo real pelo microcontrolador de maneira eficiente e viável, o que, conforme proposto pelos autores Satyanarayana et al. (2023) [1], destacam a importância de otimizar a segurança sem comprometer o desempenho do sistema.

3.1.1 Ferramentas e Tecnologias Utilizadas

a. Hardware IoT: O ESP32 foi escolhido como o microcontrolador principal por diversos motivos, entre eles devido à sua boa capacidade de processamento, suporte nativo a protocolos de comunicação sem fio (Wi-Fi, Bluetooth), facilidade e versatilidade de uso e programação para vários propósitos, e por possuir funcionalidades de segurança já embutidas de fábrica, como criptografia de dados e armazenamento seguro do código em memória *flash* [18]. Ao longo do planejamento do projeto, definiu-se que seriam utilizados dois microcontroladores ESP32, um atuando como *Publisher* e outro atuando como *Subscriber* no contexto de funcionamento do protocolo de comunicação MQTT [21], além do aplicativo móvel instalado em um smartphone. Os ESP32 serão configurados, então, para enviar e receber dados via MQTT, utilizando o *broker* EMQX e acessando ou enviando os dados através de tópicos específicos para cada função (por exemplo, abertura de fechadura e monitoramento de temperatura).

b. Software Mobile: O MIT App Inventor [24] foi selecionado como plataforma de desenvolvimento do aplicativo *mobile* devido à sua interface gráfica simples e intuitiva para construção da aplicação completa, tanto em *front-end* quanto em *back-end,* além de oferecer extensões de uso livres com suporte a MQTT e ao padrão de criptografia AES. Essa escolha permitiu o rápido desenvolvimento do aplicativo de controle e gerenciamento do sistema IoT, responsável por enviar comandos e receber dados do ESP32 em tempo real. A interface foi projetada para ser fácil de utilizar, intuitiva, facilitando o monitoramento de dados em tempo real e o controle remoto dos dispositivos conectados.

c. Segurança: Para garantir a confidencialidade dos dados transmitidos entre o aplicativo e o ESP32, principalmente no que diz respeito ao acionamento de hardware sensível e crítico – como a abertura da fechadura, por exemplo – foi selecionado ao longo da pesquisa, como método criptográfico ideal e muito seguro, a criptografia AES de 256 bits (ou somente AES-256) em modo de operação Cipher Block Chaining (CBC) [15]. O algoritmo foi escolhido devido à sua robustez e uso consolidado no mercado como um todo e em dispositivos IoT [1]. A biblioteca *mbedTLS* [17] foi utilizada no ESP32 para realizar a criptografia e descriptografia dos dados, enquanto o aplicativo no MIT App Inventor empregou a extensão *MareshaAES* [22] para criptografar os dados enviados. Um fator extra de autenticação também foi implementado no aplicativo, utilizando biometria para validar o usuário do aplicativo antes de enviar comandos sensíveis, como o de desbloqueio de uma fechadura magnética.

3.2 DESENVOLVIMENTO DO HARDWARE IOT

a. Seleção de Componentes: O projeto utilizou o sensor DHT11, amplamente utilizado no meio acadêmico, para coletar os dados de temperatura e umidade, um LED dimerizável, e uma fechadura magnética 12V como hardwares controlados pelas ESP32. A escolha dos componentes foi baseada na disponibilidade e compatibilidade com o microcontrolador, no intuito de demonstração de componentes residenciais do projeto, bem como no baixo consumo de energia, característica fundamental em sistemas IoT.

b. Projeto e Montagem do Circuito: O circuito foi projetado considerando a integração de todos os componentes conectados à dois microcontroladores ESP32. A montagem incluiu o sensor DHT11, o controle de um LED dimerizável via PWM e a fechadura magnética acionável por 12 volts. Uma ESP32 foi configurada para atuar como S*ubscriber* recebendo comandos de dimerização e acionamento da fechadura via MQTT, e a outra foi configurada para atuar como *Publisher* no MQTT, enviando os dados de monitoramento dos sensores ao aplicativo mobile periodicamente.

3.3 DESENVOLVIMENTO DO APLICATIVO MOBILE

a. Escolha de Plataformas: O MIT App Inventor foi escolhido pela facilidade de desenvolvimento e integração com o protocolo MQTT. O aplicativo foi projetado com uma interface gráfica simples e intuitiva para permitir o monitoramento das condições ambientais de temperatura e umidade, via sensor DHT11, e o controle de dispositivos, como o LED dimerizável e a fechadura magnética do tipo solenoide.
b. Processo de Codificação e Integração: O aplicativo foi programado para se conectar ao servidor MQTT e se comunicar com a ESP32 por meio de tópicos específicos. Além disso, as funcionalidades de controle de acesso e segurança, como autenticação via biometria, foram integradas para evitar que usuários não autorizados pudessem enviar comandos críticos, como a abertura da fechadura.

3.4 IMPLEMENTAÇÃO DE CAMADAS DE SEGURANÇA

Em concordância com os termos relacionados à criptografia contextualizados na seção anterior, ao longo do estudo e desenvolvimento do projeto, foi priorizado e posto em prática preceitos essenciais para a garantia da Segurança da Informação, tanto do aplicativo *mobile*, quanto do *hardware* IoT, que foram traduzidas em implementações de criptografia ponta-a-ponta entre dispositivos, via protocolo de comunicação, e em uma autenticação por biometria digital, conforme segue:

a. Criptografia e Autenticação: A criptografia AES-256 foi implementada tanto no aplicativo quanto na ESP32, utilizando o algoritmo em modo CBC com padding PKCS5 para proteger os dados transmitidos. A chave de criptografia foi derivada usando PBKDF2 com SHA-256 e um salt específico, gerado e armazenado na ESP32. A autenticação do usuário foi realizada via biometria no aplicativo, garantindo que apenas usuários autorizados pudessem enviar comandos, e a comunicação entre o aplicativo e a ESP32 foi criptografada para impedir interceptação de dados sensíveis.

b. Protocolos de Comunicação: O protocolo MQTT foi utilizado para a troca de mensagens entre o aplicativo e a ESP32, com as camadas de segurança adicionadas ao protocolo para garantir a confidencialidade, integridade e autenticidade das mensagens. Além da criptografia, foi implementado um sistema de verificação de integridade dos dados utilizando algoritmos de *hashing*, para garantir que os dados não fossem adulterados durante a transmissão.

3.5 CRITÉRIOS DE AVALIAÇÃO

O êxito do projeto foi determinado com base nos seguintes critérios: (1) a funcionalidade correta do sistema IoT, incluindo a coleta de dados dos sensores e o controle dos atuadores, bem como a integração do *software mobile* com o hardware; (2) a eficiência da criptografia AES-256, garantindo a confidencialidade das mensagens trocadas entre a ESP32 e o aplicativo; (3) a robustez da autenticação por biometria, validando a identidade do usuário antes de comandos críticos; e (4) a resistência do sistema a ataques cibernéticos, testada por meio de simulações de interceptação de dados.

4 DESENVOLVIMENTO

Esta seção descreve as etapas práticas realizadas para concretizar o sistema pretendido, abrangendo o desenvolvimento e a integração dos componentes de hardware e software, bem como a implementação das medidas de segurança necessárias. O projeto foi desenvolvido de forma modular, garantindo que cada parte do sistema pudesse ser testada e validada individualmente antes da integração total.

As subseções a seguir detalham o design e implementação do hardware IoT, o desenvolvimento do aplicativo mobile, e a aplicação das medidas de segurança, destacando os desafios enfrentados e as soluções adotadas. Diagramas e fluxogramas complementam as explicações, fornecendo uma visão clara das interações entre os elementos do projeto.

4.1 PROJETO DO HARDWARE IOT

O projeto do hardware IoT foi desenvolvido com foco em atender às necessidades funcionais e de segurança do sistema proposto. Para isso, foram utilizados componentes cuidadosamente selecionados, como o microcontrolador ESP32, que oferece suporte nativo a protocolos de comunicação sem fio, além de funcionalidades integradas de segurança.

As subseções seguintes abordam em detalhes a escolha dos sensores e atuadores, como o sensor DHT11, o LED dimerizável e a fechadura magnética, além de apresentar os esquemas e diagramas que representam a arquitetura do hardware e suas conexões. Essa estrutura possibilita a coleta de dados, como temperatura e umidade, e o controle remoto de dispositivos críticos, garantindo a funcionalidade do sistema e sua integração com o aplicativo mobile.

4.1.1 Descrição Detalhada do Design e Implementação do Hardware

O projeto de hardware se baseou na escolha do ESP32 como microcontrolador programável principal, que possui Wi-Fi e Bluetooth integrados, sendo adequado para aplicações de Internet das Coisas (IoT) devido à sua eficiência de energia, robustez de *design*, conectividade e suporte à criptografia embutida [18]. A escolha do ESP32

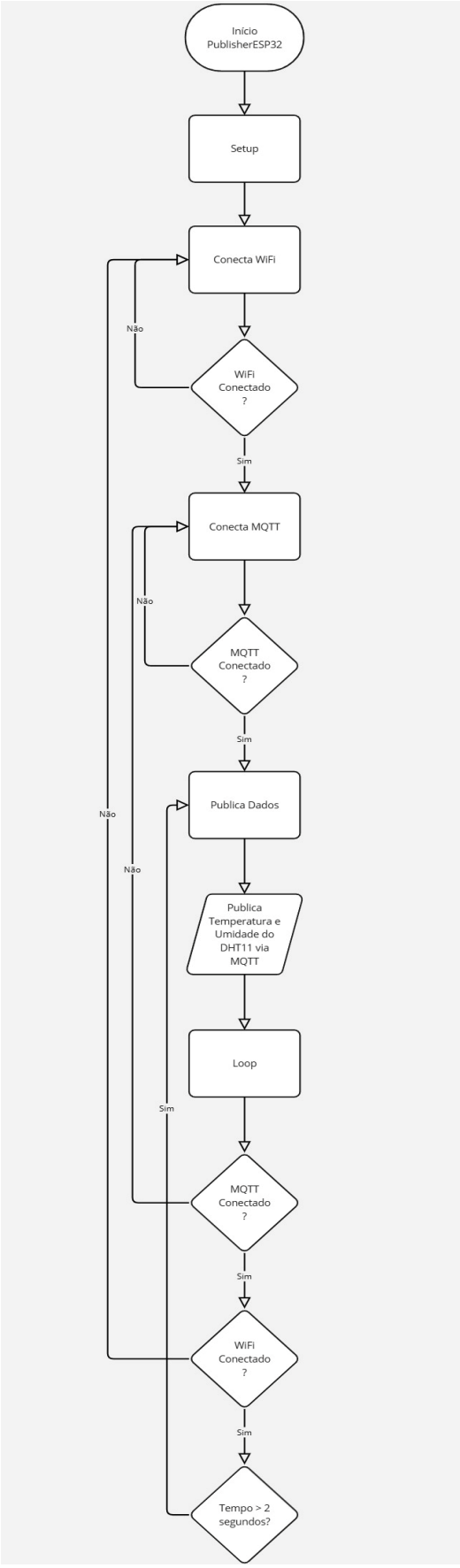
também foi guiada por sua versatilidade e compatibilidade com protocolos como o MQTT para comunicação eficiente e segura entre dispositivos IoT, bem como pela facilidade e flexibilidade em programá-lo, sendo compatível com o *software* Arduino IDE, por exemplo.

A arquitetura do hardware incluiu:

a. Utilização de bibliotecas que viabilizam a comunicação do microcontrolador via WiFi e a comunicação via protocolo MQTT, como a *WiFi.h* e a *PubSubClient.h*.
b. Módulos de controle e gerenciamento de LEDs, de fechadura e de sensor DHT11, com estados dos tópicos MQTT pré-definidos para a dimerização do LED ("led0" a "led100", por exemplo), que representam a intensidade, além de tópicos para captar a temperatura, umidade e controle da fechadura, abrangendo todos os dispositivos conectados ao sistema.
c. Desenvolvimento de uma biblioteca, chamada *DecryptAES* que implementa a descriptografia dos dados, realizando o algoritmo reverso da criptografia inicial, e utilizando o padrão de criptografia AES-256 com modo de criptografia CBC (*Cipher Block Chaining*) [15].

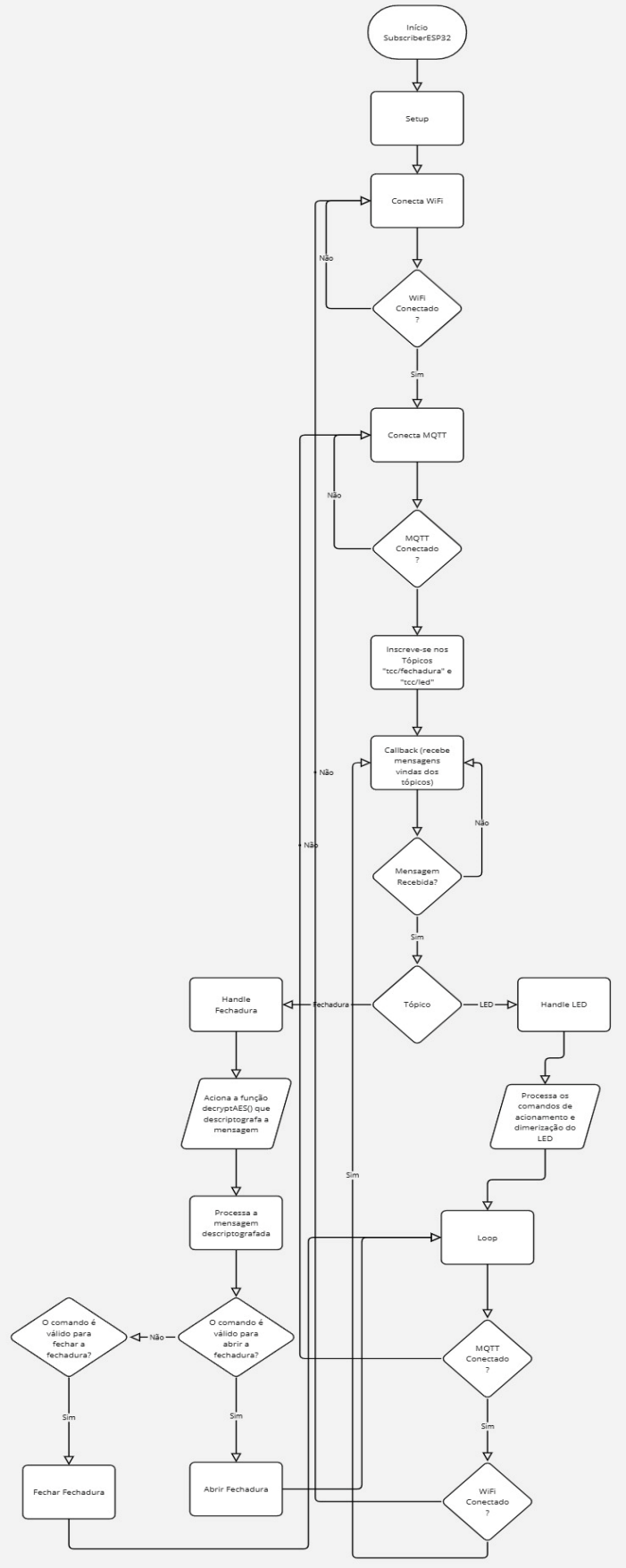
No projeto em si foram utilizados dois ESP32: um atuando apenas como *Publisher* no contexto do MQTT [21], enviando continuamente os dados de temperatura e umidade do ambiente a partir do sensor DHT11 para o broker, e outro atuando como *Subscriber*, acionando os comandos de controle e dimerização do LED e abrindo ou fechando a fechadura caso a descriptografia do dado coincida com o comando correto. Abaixo, seguem os fluxogramas do algoritmo de programação de cada ESP32 utilizada no projeto (Figuras 1 e 2):

Figura 1 - Fluxograma do algoritmo da ESP32 *Publisher*



Fonte: Autor

Figura 2 - Fluxograma do algoritmo da ESP32 *Subscriber*



Fonte: Autor

Atrelado ao projeto do hardware do ESP32 Subscriber, especificamente ao correto controle e acionamento da fechadura magnética do tipo solenoide, viu-se a necessidade de analisar, testar o dispositivo e pesquisar algumas informações de componentes utilizados no projeto para calcular as corretas resistências a se utilizar para o acionamento desse tipo específico de dispositivo pelo ESP32, sem danificar nenhum componente e tendo um controle assertivo. Para esse componente, o transistor NPN escolhido foi o TIP120, juntamente com um diodo de roda-livre 1N4007 em paralelo aos terminais da fechadura, para proteger o circuito contra picos de tensão induzidos ao desligue do solenoide, como pode ser observado no diagrama esquemático da Figura 6. Aplicando-se as Leis de *Kirchhoff* e observando-se o gráfico do base-emissor “On” voltagem (Figura 3) que fornece o V_{BE}, foi possível determinar o valor correto do resistor R_B, conforme a equação:

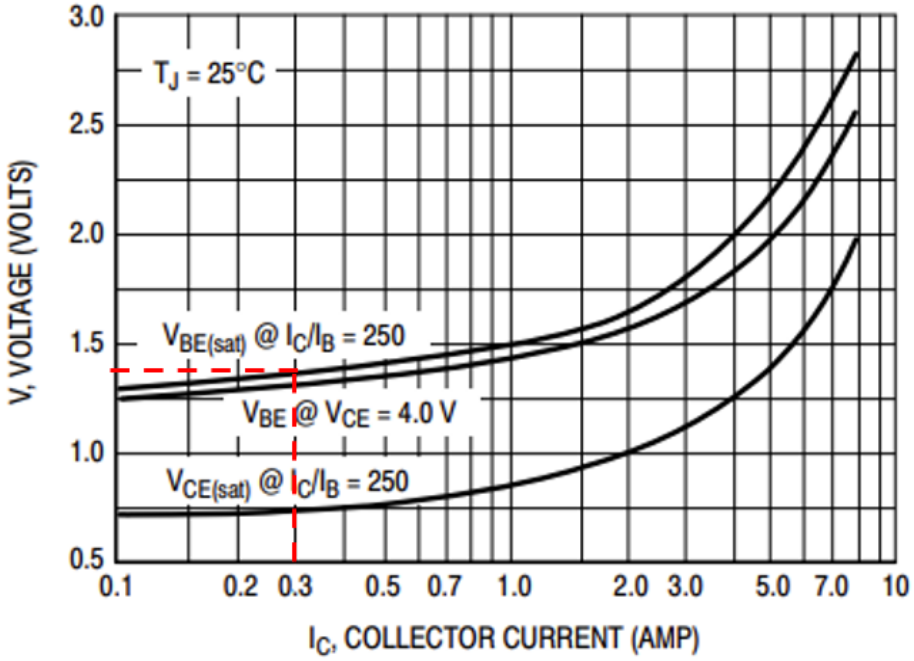
$$Ic = 300mA;\ \beta = 250;\ Ib = \frac{300mA}{250} = 1.2mA;$$

$$-3.3 + Rb * Ib + Vbe = 0$$

$$-3.3 + Rb * 1.2m + 1.4 = 0$$

$$Rb = 1k5\Omega$$

Figura 3 - ”On” Voltages TIP120

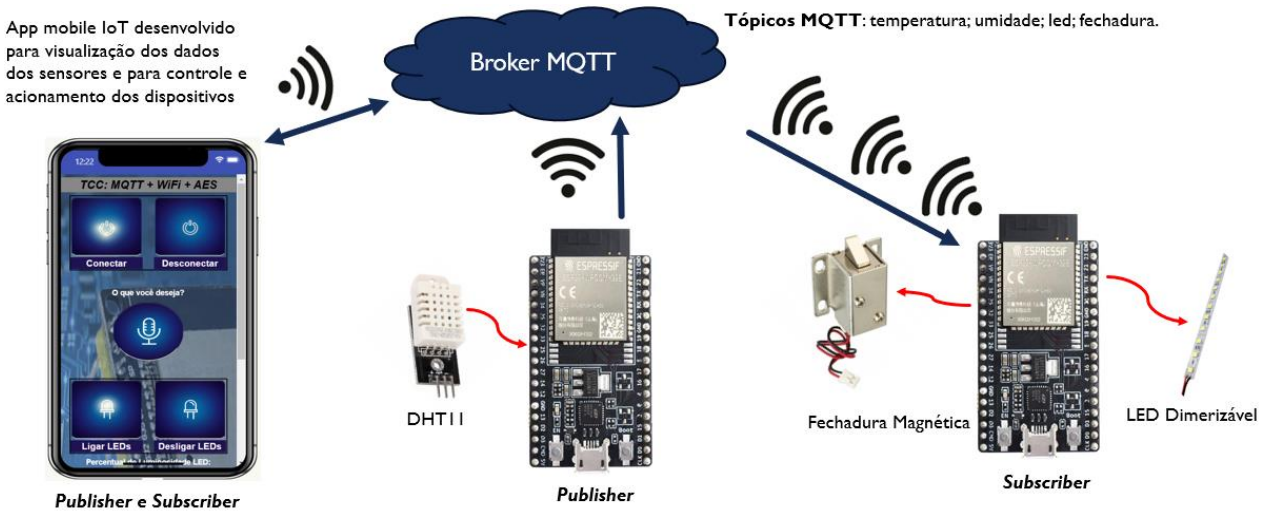


Fonte: Plastic Medium-Power Complementary Silicon Transistors, TIP120 (NPN). Disponível em <https://www.onsemi.com/pdf/datasheet/tip120-d.pdf>. Acesso em 5 nov. 2024.

4.1.2 Diagramas do Projeto

A arquitetura do projeto possui quatro principais componentes independentes: 1) O aplicativo *mobile* desenvolvido, que atua tanto como *Subscriber* quanto *Publisher,* responsável por enviar os comandos de controle para os atuadores e por receber e apresentar os dados dos sensores para o usuário; 2) a ESP32 que atua apenas como *Publisher*, responsável por enviar as informações em tempo real do sensor DHT11 ao *broker* MQTT; 3) a ESP32 *Subscriber* que recebe os comandos vindos do *app,* via *broker,* e atua na da fechadura e no LED, conforme os comandos, caso sejam válidos; 4) e o *Broker* MQTT, que é um servidor de comunicação intermediário das informações, responsável por gerir as publicações e as subscrições do protocolo MQTT, ou seja, receber os dados dos *Publishers* e entrega-los aos *Subscribers*. O diagrama simplificado da arquitetura do projeto (Figura 4), demonstrando as integrações descritas, permite uma visualização geral *macro* de todos os componentes do projeto e as interações entre si.

Figura 4 - Diagrama de Arquitetura do projeto

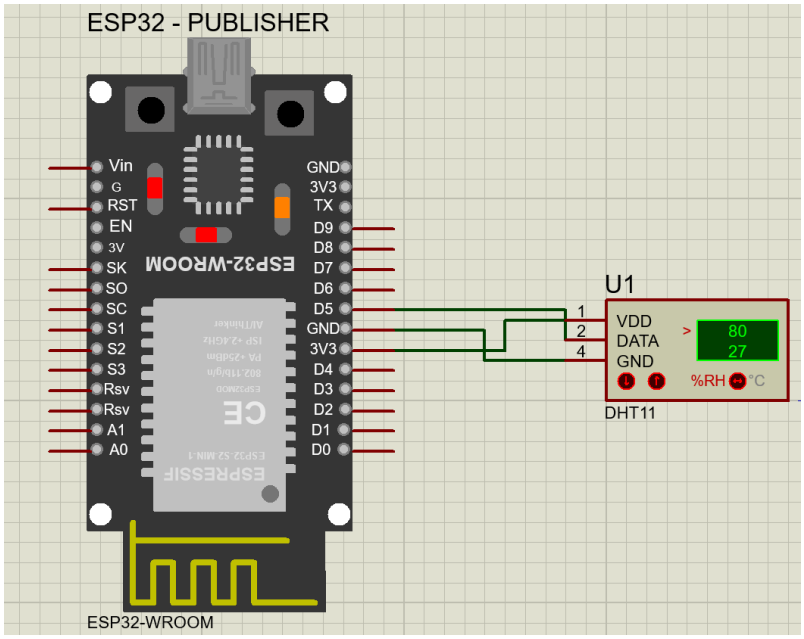


Fonte: Autor

Os diagramas esquemáticos dos circuitos desenvolvidos no projeto foram construídos via software *Proteus Design Suite*, detalhando as conexões dos ESP32 *Publisher* (Figura 5) e *Subscriber* (Figura 6), aos componentes utilizados no TCC. Cada componente foi ligado a uma porta GPIO do ESP32, facilitando o controle direto por comandos MQTT, conectados com resistores, diodos e transistores apropriados para garantir o correto funcionamento do circuito e evitar sobrecarga. Essa estrutura

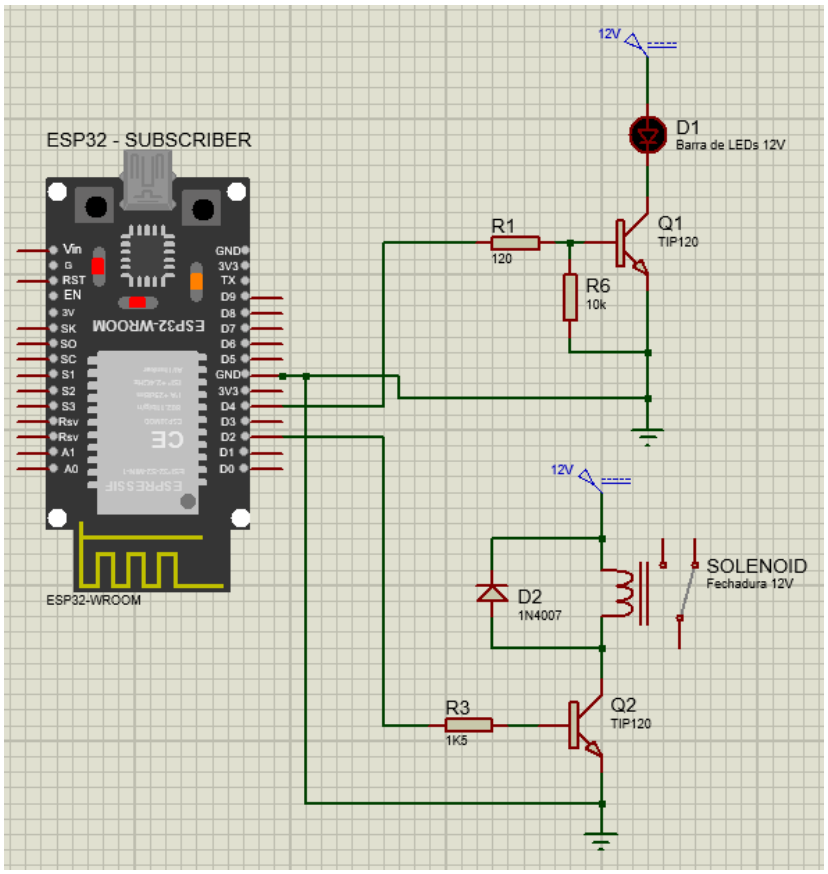
foi fundamental para garantir eficiência e confiabilidade do sistema no controle dos dispositivos IoT, a partir do aplicativo mobile.

Figura 5 - Diagrama Esquemático da ESP32 Publisher



Fonte: Autor

Figura 6 - Diagrama Esquemático da ESP32 Subscriber



Fonte: Autor

4.2 DESENVOLVIMENTO DO APLICATIVO MOBILE

O aplicativo mobile desenvolvido para este projeto foi concebido com o objetivo de gerenciar e controlar, de forma segura, os dispositivos IoT integrados ao sistema. Construído utilizando a plataforma MIT App Inventor, o aplicativo oferece uma interface amigável e intuitiva, permitindo ao usuário realizar todas as ações de forma prática, como conectar-se ao broker MQTT, acionar dispositivos como LEDs e fechadura magnética, e monitorar em tempo real os valores de temperatura e umidade do ambiente obtidos por sensores.

Entre as funcionalidades implementadas, destacam-se os botões de "Conectar" e "Desconectar", que possibilitam a interação direta com o broker MQTT. O botão "Conectar" é complementado por uma camada adicional de segurança, exigindo a autenticação biométrica por impressão digital, garantindo que apenas usuários autorizados possam acessar e controlar os dispositivos. Importante mencionar que, caso o número de tentativas mal-sucedidas seja superior a cinco, o aplicativo bloqueia temporariamente as próximas tentativas do usuário. Essa abordagem reforça os preceitos de segurança da informação, especialmente no que diz respeito à autenticidade, e agregando valor ao projeto.

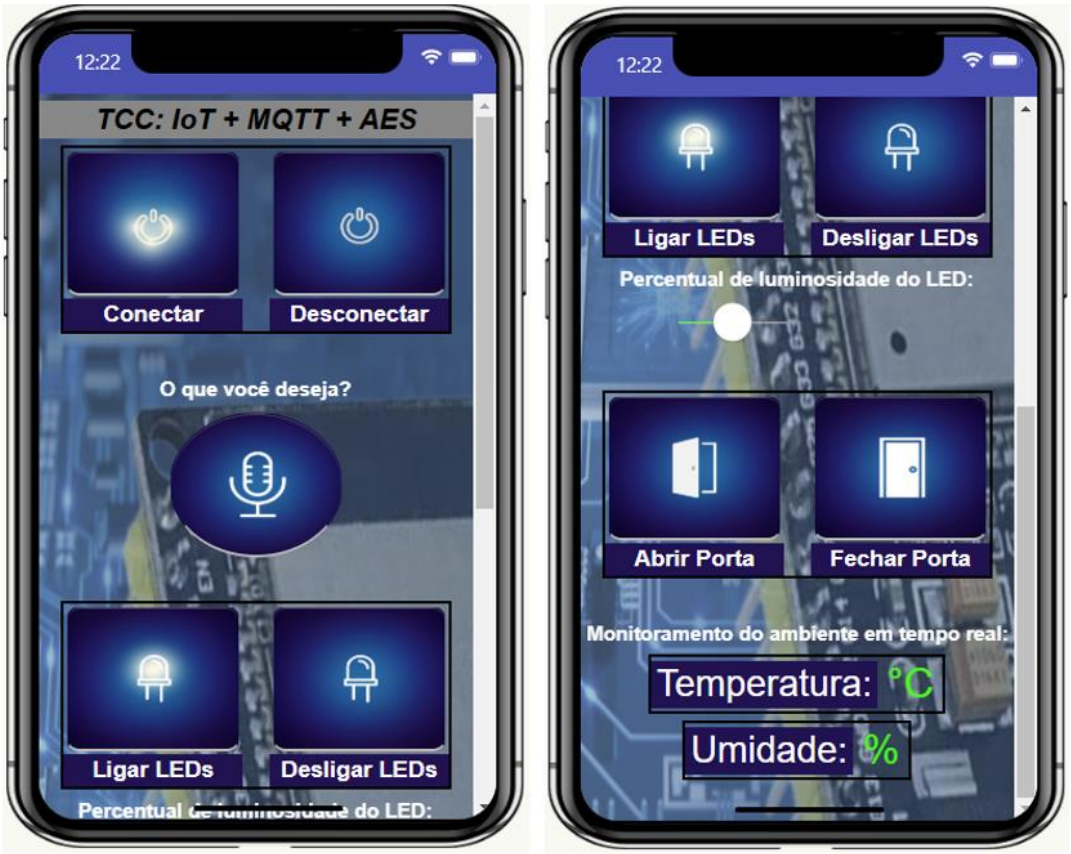
4.2.1 Apresentação da Interface do Usuário (UI)

O aplicativo *mobile* foi desenvolvido no MIT App Inventor, que oferece uma plataforma *web* visual simples, intuitiva e completa para programação de aplicativos móveis, tanto o *desing* funcional quanto o *back-end* programável em blocos, com capacidade de instalação e uso de extensões que permitem a comunicação com dispositivos IoT através de MQTT, além da criptografia AES também utilizada no projeto. A UI foi projetada para ser intuitiva, simples e de fácil utilização, permitindo aos usuários controlarem os dispositivos IoT rapidamente e visualizar os dados dos sensores em tempo real.

A Interface do Usuário possui botões de controle para os dispositivos IoT conectados à rede (como abrir a fechadura e controlar a intensidade do LED, por exemplo), também foi implementada a funcionalidade de reconhecimento dos comandos do usuário por voz, através do acionamento de um botão com o símbolo

de um microfone, e indicadores de status que mostram o feedback dos dispositivos em tempo real, garantindo que o usuário receba uma confirmação visual ao interagir com o hardware, além disso, também foi reservado um espaço no *app* onde é mostrado e atualizado em tempo real os dados de temperatura e umidade vindos dos sensores. Abaixo (Figura 7), é possível visualizar a Interface de Usuário completa do aplicativo:

Figura 7 - Interface do Usuário do Aplicativo Mobile Desenvolvido



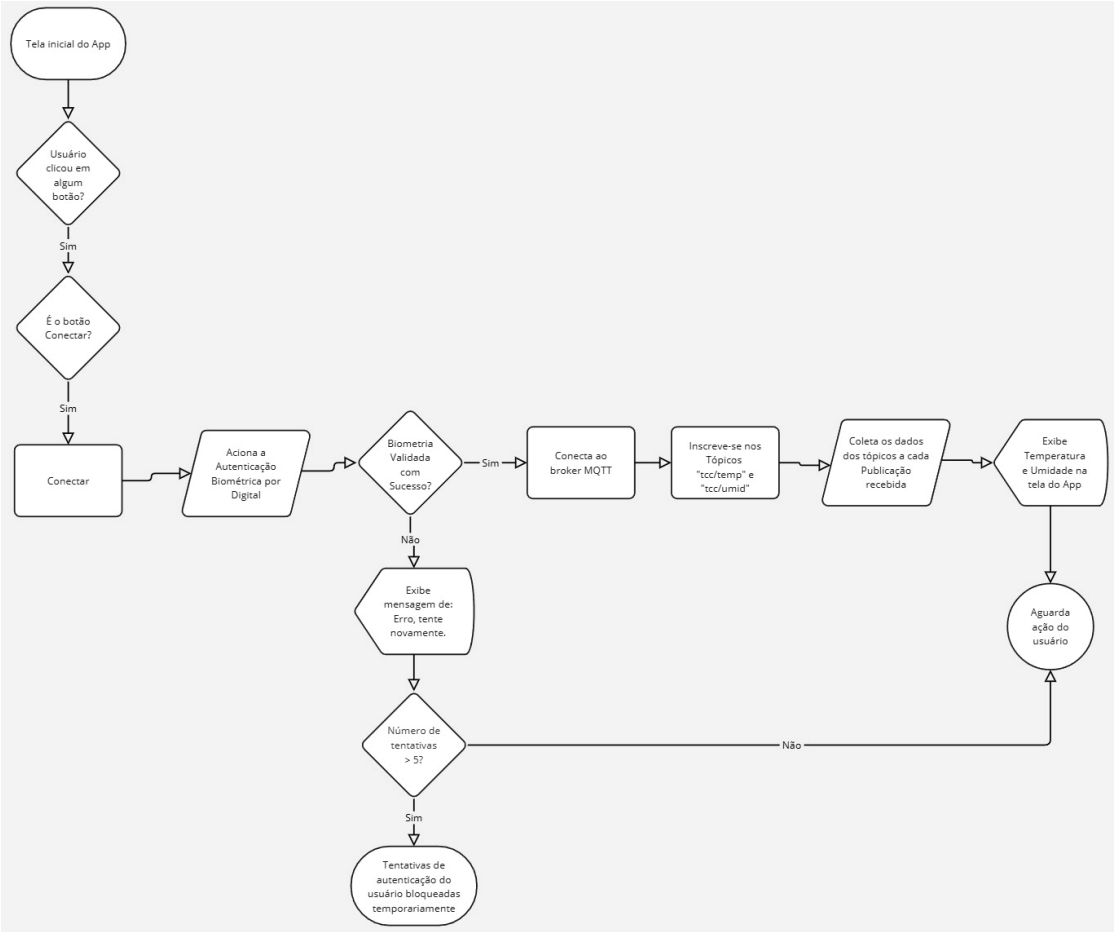
Fonte: Autor

4.2.2 Fluxogramas de Funcionamento do App

Basicamente, o aplicativo apresenta botões específicos para o controle dos LEDs e da fechadura magnética. Por meio de comandos simples e intuitivos, o usuário pode ligar, desligar ou ajustar o brilho dos LEDs, bem como abrir ou trancar a fechadura. Outra funcionalidade essencial do aplicativo é o monitoramento em tempo real da temperatura e umidade do ambiente. Os dados são transmitidos pelo sensor DHT11 via protocolo MQTT e exibidos diretamente na interface do aplicativo, proporcionando uma visualização clara e atualizada das condições ambientais.

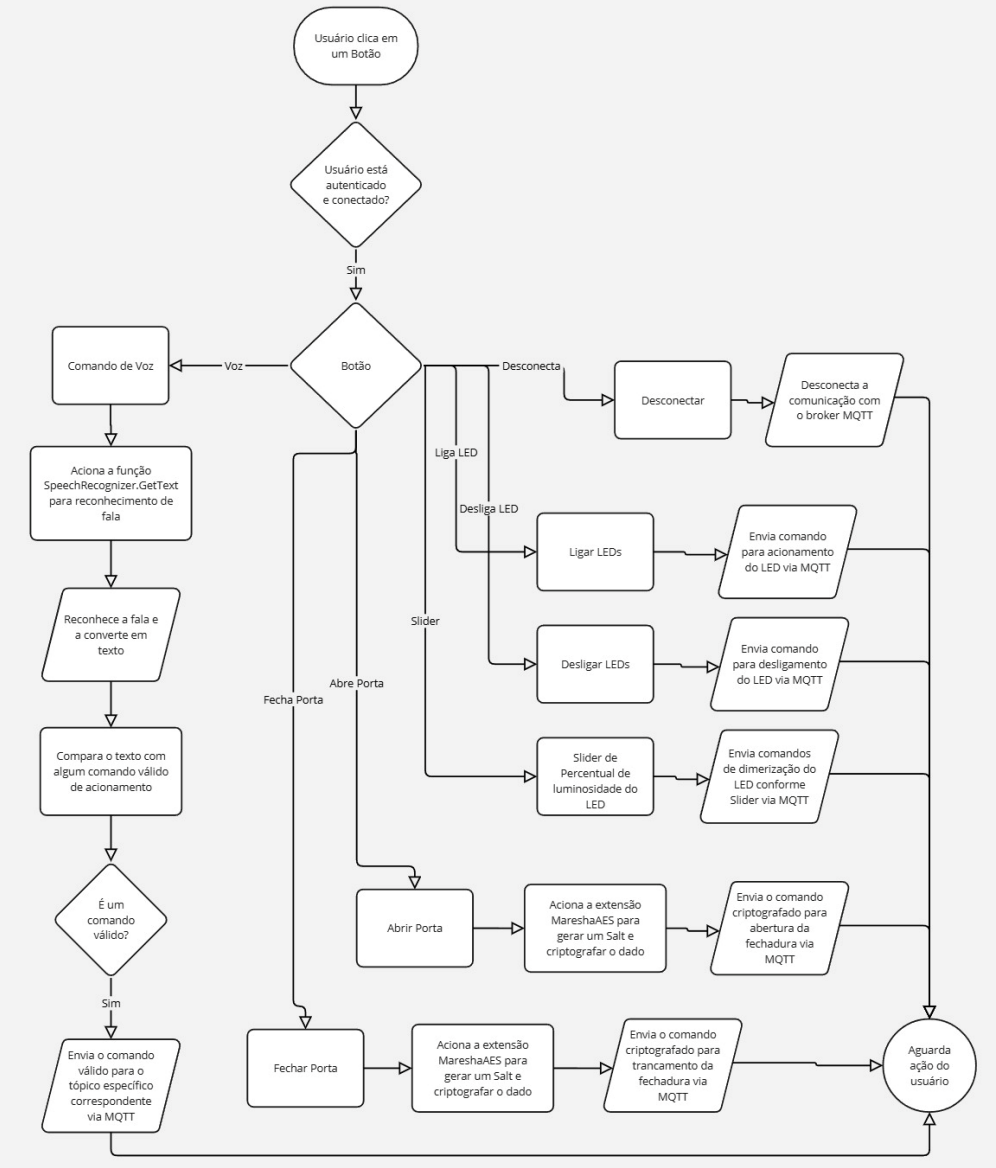
Para ilustrar a lógica do funcionamento do aplicativo, foram desenvolvidos fluxogramas detalhando as interações e decisões que ocorrem no código, desde a autenticação biométrica inicial até o envio e recepção de mensagens via protocolo MQTT. O fluxograma com a lógica inicial de conexão e autenticação no App pode ser visualizado na Figura 8. O fluxograma demonstrando o algoritmo relacionado ao funcionamento dos botões de acionamento clicados pelo usuário pós autenticação bem-sucedida está disponível na Figura 9. Os fluxogramas foram desenvolvidos pois permitem uma compreensão fácil e visual do funcionamento do sistema como um todo.

Figura 8 - Fluxograma da Autenticação Inicial do App



Fonte: Autor

Figura 9 - Fluxograma de acionamento dos Botões pós autenticação realizada



Fonte: Autor

4.2.3 Detalhes sobre a Integração com o Hardware IoT

Como já dito anteriormente, a comunicação entre o aplicativo móvel e os ESP32 é realizada por meio do protocolo MQTT, que viabiliza o envio prático de mensagens seguras em tempo real entre dispositivos conectados. No App Inventor, além da extensão que implementa o MQTT na plataforma, também foi utilizada a extensão não-nativa da plataforma *MareshaAES* [22] para criptografar as mensagens de controle crítico antes de enviá-las ao *broker*, garantindo que apenas o hardware com a chave correta possa descriptografar, interpretar a mensagem e responder a

esses comandos para abrir a fechadura, por exemplo. Implementadas todas essas funcionalidades e extensões, atreladas ao uso do MQTT, torna-se possível que a plataforma *mobile* gerencie dispositivos IoT residenciais de forma confiável, segura e à distância, independentemente da localização do usuário, bastando apenas uma conexão à internet.

4.3 IMPLEMENTAÇÃO DAS MEDIDAS DE SEGURANÇA

Nesta seção, são detalhadas as estratégias de segurança adotadas para proteger os dados e a comunicação no sistema IoT. A implementação focou em garantir a confidencialidade, integridade e autenticidade das informações transmitidas entre os dispositivos IoT e o aplicativo mobile, utilizando técnicas de criptografia avançadas e mecanismos de autenticação robustos.

As subseções a seguir exploram a descrição da implementação de criptografia e autenticação, que envolveu o uso do algoritmo AES-256 em modo CBC e autenticação biométrica no aplicativo, além de discutir os desafios enfrentados e as soluções adotadas durante o desenvolvimento do projeto. Essas medidas foram fundamentais para mitigar riscos como interceptação de dados, acessos não autorizados e adulteração de informações críticas.

4.3.1 Descrição da Implementação de Criptografia e Autenticação

Para garantir a segurança das comunicações entre o aplicativo mobile e o microcontrolador ESP32, foi implementada uma solução de criptografia robusta baseada no algoritmo AES-256 em modo CBC. O processo de criptografia no aplicativo utiliza a extensão *MareshaAES* [22], disponível para o MIT App Inventor, enquanto a descriptografia no ESP32 foi desenvolvida utilizando funções específicas da biblioteca mbedTLS [17], amplamente reconhecida por sua eficiência e compatibilidade com dispositivos embarcados.

O foco principal da implementação foi replicar no ESP32 as técnicas utilizadas pela extensão no aplicativo, assegurando a interoperabilidade entre os dois ambientes. Isso incluiu a aplicação do algoritmo de derivação de chave PBKDF2 com o *hash* HMAC-SHA256, o uso de codificação *Base64* e a remoção do *padding* PKCS5

durante a descriptografia [16]. A partir do estudo detalhado das funções utilizadas no *MareshaAES*, foi possível construir e validar a implementação no microcontrolador.

Para a ESP32, foi desenvolvida a biblioteca DecryptAES, projetada para realizar a descriptografia de dados criptografados recebidos via MQTT. Esta biblioteca encapsula três funções principais:

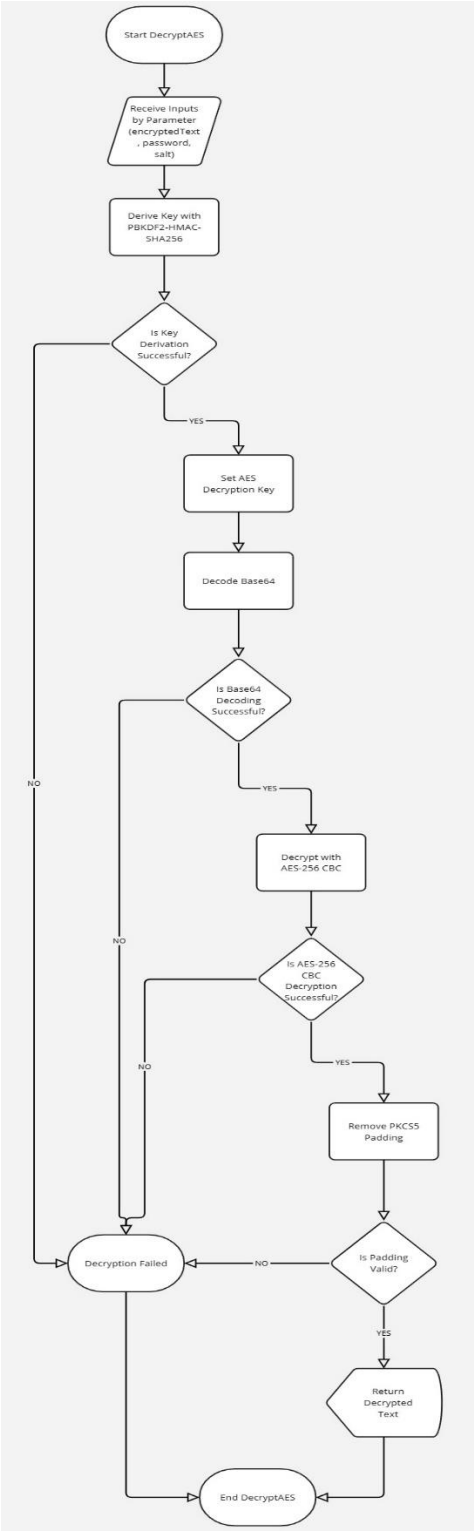
a. removePKCSPadding: Remove o preenchimento PKCS5 aplicado durante o processo de criptografia para ajustar o tamanho do dado ao bloco.
b. deriveKey: Deriva uma chave segura a partir de uma senha e um Salt utilizando PBKDF2, garantindo maior proteção contra ataques de força bruta.
c. decryptAES: Realiza a descriptografia do dado recebido utilizando a chave derivada e o vetor de inicialização (IV), retornando o conteúdo original.

O uso da extensão no App Inventor e a construção do algoritmo de descriptografia em linguagem C para o microcontrolador, permitiram que as mensagens fossem criptografadas antes de serem transmitidas ao *broker* MQTT, trafegassem dessa maneira pelo servidor até chegar ao microcontrolador, para somente ser descriptografado internamente no dispositivo em tempo real, garantindo assim uma criptografia de ponta-a-ponta. Os principais parâmetros de segurança da criptografia incluem [16]:

a. SALT e SECRET_KEY: A senha e o *salt* são utilizados para derivar uma chave única, garantindo que apenas o dispositivo específico com a chave correta possa descriptografar as mensagens.
b. IV (Vetor de Inicialização): Nesse caso inicializado com zeros, de acordo com o implementado na extensão *MareshaAES*, para garantir compatibilidade entre o aplicativo e o dispositivo.

Abaixo (Figura 10), é possível visualizar o fluxograma detalhado do algoritmo de implementação da biblioteca *DecryptAES*, desenvolvida ao longo deste trabalho, utilizado para descriptografar os dados vindos via MQTT.

Figura 10 - Fluxograma do Algoritmo DecryptAES Desenvolvido



Fonte: Autor

É importante ressaltar que a implementação de criptografia foi adotada para prevenir possíveis ataques de interceptação de dados, como o *Man-in-the-Middle*, uma vulnerabilidade comum em dispositivos IoT expostos na internet, e para manter sigilosas e seguras informações críticas e sensíveis.

4.3.2 Discussão sobre os Desafios e Soluções Adotadas

Durante a implementação, surgiram desafios principalmente relacionados à sincronização e replicação dos resultados de ambos os algoritmos com sucesso, devido principalmente a incompatibilidade inicial dos algoritmos criptográficos entre o ESP32 e o MIT App Inventor, visto que um foi codificado e desenvolvido em Java e o outro em C++, linguagens bem distintas e que apresentam suas particularidades, possuindo bibliotecas diferentes entre si e com modos de utilização de funções também específicos. Quaisquer passos da criptografia executados de forma errada, ou fora da ordem correta de implementação, gerariam resultados finais diferentes e, portanto, incompatíveis entre si. O ajuste correto dos parâmetros de *Padding* (PKCS5) e da quantidade de iterações para a derivação da chave (65536 iterações no PBKDF2) foram essenciais para garantir que a criptografia e a descriptografia ocorressem de forma eficiente, equivalente e sem erros.

5 RESULTADOS

Esta seção apresenta os resultados obtidos com o desenvolvimento do projeto, destacando as funcionalidades implementadas, o desempenho do sistema e as análises realizadas durante os testes. Foram avaliados critérios como o funcionamento correto do hardware e software, a eficiência da comunicação via MQTT, e a robustez das camadas de segurança, incluindo a criptografia e a autenticação biométrica.

Os resultados são detalhados a partir da integração entre hardware e software, considerando o processo de configuração e os ajustes necessários para o pleno funcionamento do sistema. Além disso, os testes realizados para validar a segurança e a funcionalidade do sistema são apresentados, oferecendo uma análise prática do desempenho do projeto.

5.1 INTEGRAÇÃO E TESTES

A integração entre os componentes do hardware e do software foi um ponto central no desenvolvimento do sistema IoT. Esta etapa envolveu a conexão física e lógica entre os dispositivos, como os microcontroladores ESP32, sensores, e atuadores, além da interface com o aplicativo mobile.

Nas subseções a seguir, detalham-se o processo de integração entre hardware e software, que garantiu a comunicação eficiente entre os componentes, e os resultados dos testes de funcionamento e segurança, que avaliaram a capacidade do sistema de operar corretamente em cenários simulados, assegurando proteção contra acessos não autorizados e adulterações de dados.

5.1.1 Processo de Integração entre Hardware e Software

A integração e os testes do sistema como um todo foram realizados em etapas, começando pela comunicação básica entre o aplicativo, o *broker* e os microcontroladores via protocolo MQTT, seguida pela aplicação das camadas de segurança e autenticação. Cada componente (ESP32 e aplicativo *mobile*) foi testado

de forma isolada antes de ser integrado, garantindo que o sistema de controle e os protocolos de segurança funcionassem corretamente em conjunto.

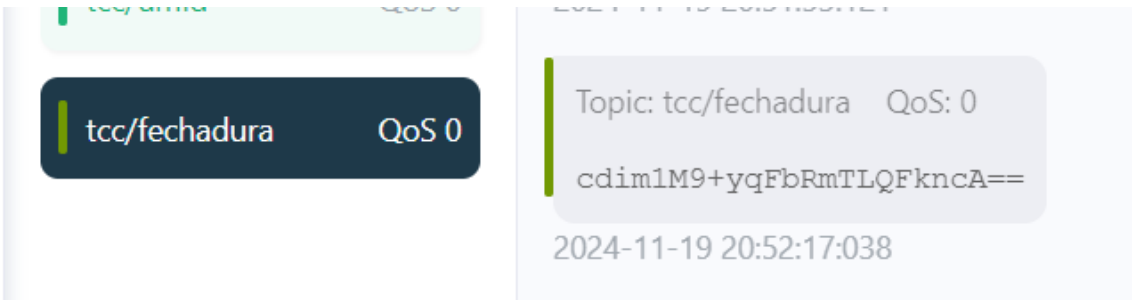
5.1.2 Resultados dos Testes de Funcionamento e Segurança

Os testes de funcionamento envolveram a verificação de resposta e controle em tempo real dos dispositivos via MQTT, enquanto os testes de segurança foram focados na integridade e eficácia da criptografia. As mensagens enviadas e recebidas foram comparadas para verificar a precisão da descriptografia, e os comandos MQTT foram testados para garantir o pleno funcionamento dos comandos, e também que apenas dispositivos autenticados pudessem modificar o estado dos dispositivos controlados.

Além disso, foram realizadas simulações de ataques comuns, como tentativas de interceptação de dados e pequenas modificações no texto criptografado via MQTT, para validar de fato a eficácia da segurança e criptografia implementada. Esse procedimento garantiu que o sistema possa operar de forma segura em redes potencialmente não confiáveis.

Abaixo, seguem algumas imagens (Figuras 11 a 22) de mensagens retornadas em tempo real pelo monitor serial da ESP32 e da mensagem correlacionada enviada no tópico MQTT tcc/fechadura, demonstrando alguns testes realizados relacionados ao acionamento da fechadura, tanto os testes de sucesso, aqueles cuja criptografia está correta e o acionamento da fechadura ocorreu conforme o esperado, quanto os de falha, simulando a uso de tentativa com chave incorreta apresentando pequenas mudanças apenas no texto cifrado.

Figura 11 - Código criptografado correto utilizado para abrir a porta, vindo do App



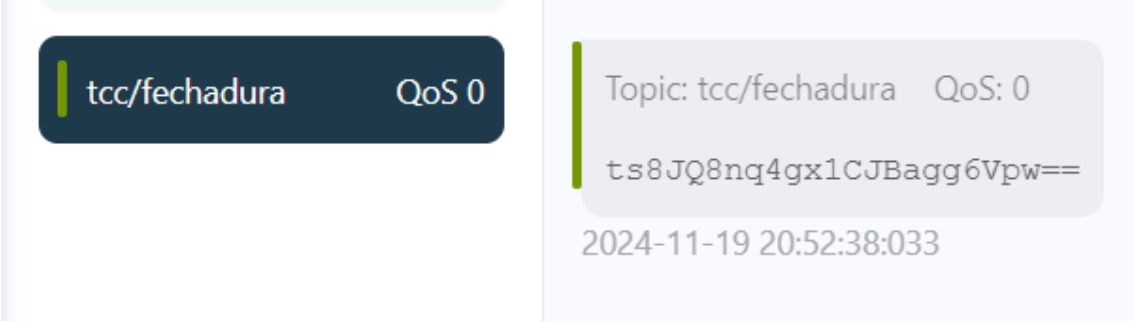
Fonte: Autor

Figura 12 - Resultado da ESP32 impresso na Serial, pós descriptografia: Sucesso em abrir fechadura

```
20:52:17.044 -> Mensagem recebida do topico tcc/fechadura
20:52:17.044 -> Mensagem: cdim1M9+yqFbRmTLQFkncA==
20:52:17.044 -> ----------------------
20:52:17.044 -> Starting decryption...
20:52:17.044 -> Deriving key with PBKDF2 (SHA-256, 65536 iterations)...
20:52:24.273 -> Key derivation complete.
20:52:24.273 -> AES decryption key set.
20:52:24.273 -> Base64 decoding complete.
20:52:24.273 -> AES decryption complete.
20:52:24.273 -> PKCS5Padding removed. Original length: 14
20:52:24.339 -> PKCS5 padding removed.
20:52:24.339 -> Decrypted text: AbrirFechadura
20:52:24.339 -> Porta Aberta!
```

Fonte: Autor

Figura 13 - Código criptografado usado para trancar a porta, vindo do App



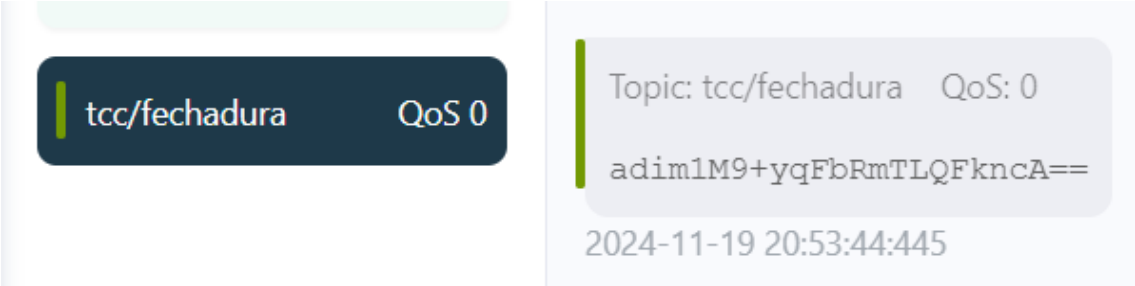
Fonte: Autor

Figura 14 - Resultado da ESP32 impresso na Serial, pós descriptografia: Sucesso em trancar a fechadura

```
20:52:38.136 -> Mensagem recebida do topico tcc/fechadura
20:52:38.137 -> Mensagem: ts8JQ8nq4gx1CJBagg6Vpw==
20:52:38.137 -> ----------------------
20:52:38.137 -> Starting decryption...
20:52:38.137 -> Deriving key with PBKDF2 (SHA-256, 65536 iterations)...
20:52:45.358 -> Key derivation complete.
20:52:45.421 -> AES decryption key set.
20:52:45.421 -> Base64 decoding complete.
20:52:45.421 -> AES decryption complete.
20:52:45.421 -> PKCS5Padding removed. Original length: 15
20:52:45.421 -> PKCS5 padding removed.
20:52:45.421 -> Decrypted text: FecharFechadura
20:52:45.421 -> Porta Trancada!
```

Fonte: Autor

Figura 15 - Envio de um dado criptografado diferente do esperado para abrir a fechadura



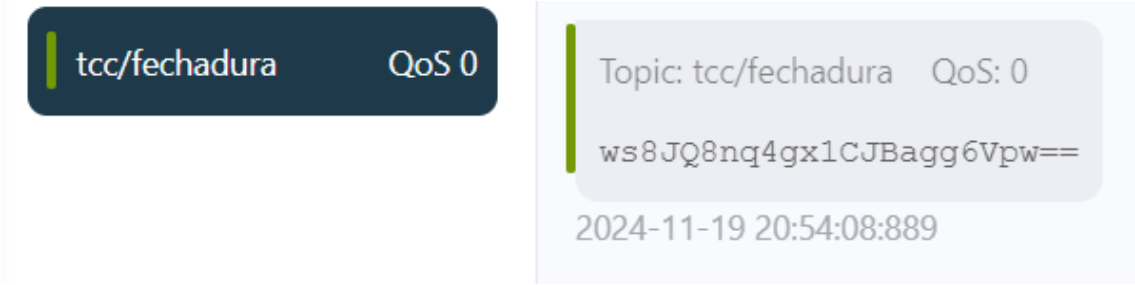
Fonte: Autor

Figura 16 - Resultado da ESP32 impresso na Serial, pós descriptografia: Falha em abrir a fechadura, Chave inválida!

```
20:53:44.564 -> Mensagem recebida do topico tcc/fechadura
20:53:44.624 -> Mensagem: adim1M9+yqFbRmTLQFkncA==
20:53:44.624 ->
20:53:44.625 -> ----------------------
20:53:44.625 -> Starting decryption...
20:53:44.625 -> Deriving key with PBKDF2 (SHA-256, 65536 iterations)...
20:53:51.857 -> Key derivation complete.
20:53:51.857 -> AES decryption key set.
20:53:51.857 -> Base64 decoding complete.
20:53:51.857 -> AES decryption complete.
20:53:51.857 -> Invalid padding detected.
20:53:51.857 -> PKCS5 padding removed.
20:53:51.857 -> Decrypted text: □gF□    ��1�$�/□□��
20:53:51.857 -> Chave inválida!
```

Fonte: Autor

Figura 17 - Envio de um dado criptografado diferente do esperado para fechar a fechadura, com pequenas diferenças de caracteres



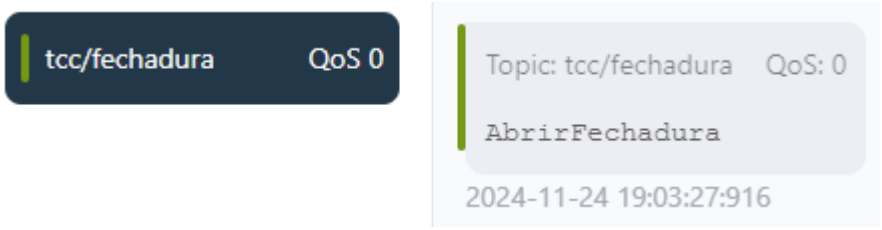
Fonte: Autor

Figura 18 - Resultado da ESP32 impresso na Serial, pós descriptografia: Falha em fechar a fechadura, Chave inválida!

```
20:54:09.072 -> Mensagem recebida do topico tcc/fechadura
20:54:09.072 -> Mensagem: ws8JQ8nq4gx1CJBagg6Vpw==
20:54:09.072 ->
20:54:09.072 -> ----------------------
20:54:09.072 -> Starting decryption...
20:54:09.072 -> Deriving key with PBKDF2 (SHA-256, 65536 iterations)...
20:54:16.288 -> Key derivation complete.
20:54:16.347 -> AES decryption key set.
20:54:16.347 -> Base64 decoding complete.
20:54:16.347 -> AES decryption complete.
20:54:16.347 -> Invalid padding detected.
20:54:16.347 -> PKCS5 padding removed.
20:54:16.347 -> Decrypted text: 5J��_□h=q��T���
20:54:16.347 -> Chave inválida!
```

Fonte: Autor

Figura 19 - Envio de um comando em texto puro, sem criptografia, para abrir a fechadura



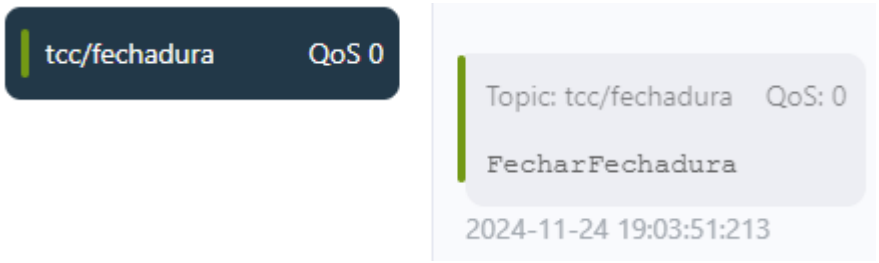
Fonte: Autor

Figura 20 – Resultado da ESP32 impresso na Serial, pós descriptografia: Falha em abrir a fechadura, Chave inválida!

```
19:03:27.987 -> Mensagem recebida do topico tcc/fechadura
19:03:27.987 -> Mensagem: AbrirFechadura
19:03:27.987 ->
19:03:27.987 -> ----------------------
19:03:27.987 -> Starting decryption...
19:03:27.987 -> Deriving key with PBKDF2 (SHA-256, 65536 iterations)...
19:03:35.240 -> Key derivation complete.
19:03:35.240 -> AES decryption key set.
19:03:35.240 -> Base64 decoding complete.
19:03:35.240 -> AES decryption complete.
19:03:35.240 -> Invalid padding detected.
19:03:35.240 -> PKCS5 padding removed.
19:03:35.240 -> Decrypted text: ,0!?(
19:03:35.240 -> Chave inválida!
```

Fonte: Autor

Figura 21 - Envio de um comando em texto puro, sem criptografia, para fechar a fechadura



Fonte: Autor

Figura 22 - Resultado da ESP32 impresso na Serial, pós descriptografia: Falha em fechar a fechadura, Chave inválida!

```
19:03:52.180 -> Mensagem recebida do topico tcc/fechadura
19:03:52.180 -> Mensagem: FecharFechadura
19:03:52.180 ->
19:03:52.180 -> ----------------------
19:03:52.180 -> Starting decryption...
19:03:52.180 -> Deriving key with PBKDF2 (SHA-256, 65536 iterations)...
19:03:59.446 -> Key derivation complete.
19:03:59.446 -> AES decryption key set.
19:03:59.446 -> Base64 decoding complete.
19:03:59.446 -> AES decryption complete.
19:03:59.446 -> Invalid padding detected.
19:03:59.446 -> PKCS5 padding removed.
19:03:59.446 -> Decrypted text: ,0!?(
19:03:59.446 -> Chave inválida!
```

Fonte: Autor

Conforme pôde-se observar nas imagens dos resultados da descriptografia no ESP32 acima, é valido mencionar que existe um *delay* de aproximadamente 7 segundos para que a descriptografia seja concluída, ou melhor, para que a porta seja efetivamente aberta em caso de sucesso. Isso está diretamente ligado à dois fatores: 1) a complexidade e robustez do algoritmo de derivação de chave, que utiliza o SHA-256 e realiza 65536 iterações para produzir o resultado e, 2) à velocidade de processamento do microcontrolador utilizado. Este último ponto, inclusive, foi observado testando-se duas versões diferentes do ESP32: uma mais atual, com mais recursos, e outra mais antiga e com menos poder de processamento. Comparando-se as duas, percebeu-se que o microcontrolador com menos recursos levou cerca de 45 segundos para realizar a descriptografia, e o ESP32 mais atual foi o utilizado neste projeto. Ou seja, de posse de um microcontrolador mais veloz, o processo de descriptografia será mais rápido.

Além dos testes de validação da eficácia da criptografia implementada no sistema IoT, via simulações de envio de comandos via MQTT, também foram realizados testes de eficácia da autenticação biométrica implementada na aplicação *mobile* que controla os dispositivos. A autenticação biométrica foi projetada como uma camada adicional de proteção para o sistema, impedindo acessos não autorizados e garantindo que apenas usuários validados pudessem se conectar ao App e executar os comandos no sistema.

Durante os testes, diversas situações foram simuladas para validar o comportamento do sistema em diferentes cenários. Foram capturadas telas do aplicativo *mobile* para ilustrar os resultados, incluindo:

a. Notificação para validação biométrica do usuário (Figura 23);
b. Sucesso na validação biométrica, confirmando que a impressão digital do usuário autorizado permitiu o acesso (Figura 24);
c. Falha na autenticação biométrica, quando a digital apresentada não correspondeu ao usuário registrado (Figura 25);
d. Erro na validação biométrica, simulando problemas técnicos, como a leitura incompleta da digital (Figura 26);
e. Bloqueio do usuário por excesso de tentativas, em que o sistema negou o acesso após cinco falhas consecutivas, o que reforça as medidas de proteção à ataques de força bruta (Figura 27);

As capturas de tela a seguir ilustram o comportamento do aplicativo em cada uma dessas situações, demonstrando que o sistema reage conforme projetado, garantindo tanto a funcionalidade quanto a segurança da aplicação.

Figura 23 - Notificação para validação biométrica digital no sensor do *smarphone*

Fonte: Autor

Figura 24 - Sucesso na autenticação biométrica, MQTT conectado

Fonte: Autor

Figura 25 - Falha na autenticação biométrica, tente novamente

Fonte: Autor

Figura 26 - Erro na validação biométrica, operação cancelada

Fonte: Autor

Figura 27 - Bloqueio temporário do usuário por excesso de tentativas falhas

Fonte: Autor

6 CONCLUSÃO

O presente trabalho apresentou o desenvolvimento de uma plataforma integrada de hardware e software IoT voltada para o controle e gerenciamento seguro de dispositivos conectados, demonstrando a aplicabilidade de práticas modernas de segurança da informação no contexto de automação residencial. A implementação de algoritmos de criptografia avançados, como o AES-256 em modo CBC, foi fundamental para garantir a confidencialidade e integridade das mensagens transmitidas, atendendo aos preceitos da segurança da informação, especialmente em um ambiente IoT.

Os resultados obtidos confirmaram a viabilidade e eficiência das soluções propostas. O sistema construído foi capaz de executar comandos críticos em cenários reais, como a abertura de fechadura de uma porta, com alto grau de confiabilidade, além de oferecer proteção, através de criptografia ponta-a-ponta, contra interceptações de dados via ataques simulados, e autenticação biométrica, para validação de uso e controle do aplicativo contra o acesso indevido. A integração entre o aplicativo mobile, o protocolo MQTT e os microcontroladores ESP32 provou ser uma solução robusta e eficiente, alcançando os objetivos definidos inicialmente.

Durante o desenvolvimento, enfrentaram-se desafios relacionados à sincronização de algoritmos criptográficos entre diferentes plataformas e tecnologias, o que exigiu pesquisa aprofundada sobre o tema e ajustes específicos. Esses desafios destacaram a importância de um planejamento detalhado e da adoção do conceito de *Security by Design* implementado em projetos de IoT.

Como desdobramento futuro, sugere-se a ampliação do sistema para suportar múltiplos dispositivos e usuários simultaneamente, com a implementação de um gerenciamento centralizado mais robusto. Além disso, a integração com outros padrões de comunicação pode aumentar a abrangência e a eficiência do sistema em diferentes cenários.

Dessa forma, este trabalho contribui para o avanço da segurança aplicada à área de Internet das Coisas, oferecendo uma solução prática, utilizando ferramentas acessíveis e que aliam funcionalidade e proteção, mostrando que é possível desenvolver sistemas embarcados, de automação e de IoT que atendam às crescentes demandas por segurança no cenário cibernético atual.

REFERÊNCIAS

1. SATYANARAYANA, P *et al*. Enhancement of Security in IoT Using Modified AES Algorithm for IoT Applications. **International Conference on Sustainable Communication Networks and Application**, Theni, p. 380-386, 2023, DOI: 10.1109/ICSCNA58489.2023.10370606. Disponível em: <https://ieeexplore.ieee.org/document/10370606>. Acesso em: 22 nov. 2024.

2. JURCUT, A. et al. Security considerations for internet of things: A survey. **SN computer science**, v. 1, n. 4, 2020. Disponível em: <https://doi.org/10.1007/s42979-020-00201-3>. Acesso em: 28 ago. 2024

3. HUBERT, K., KALEDIO, P. **Security and Privacy in IoT***:* Considerations for securing IoT devices*. [S. l.],* 2024. Em PDF.

4. Wolf, M. **Computers as components: Principles of embedded computing system design**. 3. ed. [S.l.] Elsevier, 2012.

5. STATISTA. **Internet of Things – Brazil**. Disponível em: <https://www.statista.com/outlook/tmo/internet-of-things/brazil>. Acesso em: 20 set. 2024.

6. TELENOR. **What is IoT Security?.** Disponível em: <https://iot.telenor.com/iot-insights/what-is-iot-security/>. Acesso em: 27 set. 2024.

7. PEERBITS. **Biggest IoT Security Challenges**. Disponível em: <https://www.peerbits.com/blog/biggest-iot-security-challenges.html>. Acesso em: 27 set. 2024.

8. INTERNATIONAL ELECTRONIC COMMISSION. **Cyber security for the internet of things**. Disponível em: <https://iec.ch/blog/cyber-security-internet-things>. Acesso em: 08 set. 2024.

9. CRUZ, D. M. **O que diz norma de segurança internacional para dispositivos IoT**. Disponível em: <https://www.fiee.com.br/pt-br/blog/tecnologia/o-que-diz-norma-de-seguranca-internacional-para-dispositivos-iot.html>. Acesso em: 14 set. 2024.

10. ISO/IEC 27400. INTERNATIONAL ELECTROTECHNICAL COMMISSION. **ISO/IEC 27400: Cybersecurity - IoT security and privacy - Guidelines**. [s.l: s.n.], 2022. Em PDF.

11. ALL ABOUT TESTING. **Brief Overview of ISO/IEC 27400: Comprehensive Standard on IoT Security and Privacy**. Disponível em: <https://allabouttesting.org/brief-overview-of-iso-iec-27400-comprehensive-standard-on-iot-security-and-privacy/>. Acesso em: 29 set. 2024.

12. STALLINGS, W. **Cryptography and network security: Principles and practice**. 7 ed. [s.l.] Pearson, 2017.

13. PAAR, C.; PELZL, J. **Understanding cryptography: A textbook for students and practitioners**. 2010. ed. [s.l.] Springer, 2010.

14. NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY (US). **Advanced Encryption Standard (AES)**. Washington, D.C.: National Institute of Standards and Technology (U.S.), 2023. DOI: 10.6028/NIST.FIPS.197-upd1.

15. DWORKIN, M. J. **Recommendation for block cipher modes of operation: Methods and techniques**. Gaithersburg, MD: National Institute of Standards and Technology, 2001. DOI:10.6028/NIST.SP.800-38A.

16. INTERNET ENGINEERING TASK FORCE (IETF). **RFC 2898: PKCS #5: Password-Based Cryptography Specification Version 2.0**. 2000. Disponível em: <https://datatracker.ietf.org/doc/html/rfc2898>. Acesso em: 17 set. 2024.

17. ESPRESSIF SYSTEMS. **ESP-IDF API Reference for mbedTLS**. Disponível em: <https://docs.espressif.com/projects/esp-idf/en/stable/esp32/api-reference/protocols/mbedtls.html>. Acesso em: 20 set. 2024.

18. ESPRESSIF SYSTEMS. **ESP32 Datasheet**. Disponível em: <https://www.espressif.com/en/products/socs/esp32>. Acesso em: 20 set. 2024.

19. MQTT. **MQTT: The Standard for IoT Messaging**. Disponível em: <https://mqtt.org/>. Acesso em: 18 out. 2024.

20. AMAZON. **What is MQTT?**. Disponível em: <https://aws.amazon.com/pt/what-is/mqtt/>. Acesso em: 18 out. 2024.

21. UNIVERSIDADE FEDERAL DO RIO DE JANEIRO (UFRJ). **MQTT: Protocolo de Mensageria para IoT**. Disponível em: <https://www.gta.ufrj.br/ensino/eel878/redes1-2019-1/vf/mqtt/>. Acesso em: 19 out. 2024.

22. SHARIFI, A. **[FREE] AES Encrypt/Decrypt**. Disponível em: <https://community.appinventor.mit.edu/t/free-aes-encrypt-decrypt/63415>. Acesso em: 15 set. 2024.

23. JOSEFSSON, S. **RFC 4648: The Base16, Base32, and Base64 data encodings**. Disponível em: <https://datatracker.ietf.org/doc/html/rfc4648>. Acesso em: 12 out. 2024.

24. MIT APP INVENTOR. **About Us**. Disponível em: <https://appinventor.mit.edu/about-us />. Acesso em: 26 ago. 2024.