Universidade Estadual de Feira de Santana

Programa de Pós-Graduação em Ciência da Computação

Modelagem e desenvolvimento de uma arquitetura para reconfiguração de unidades de detecção de emergências em cidades inteligentes

Gustavo Falcão Paim da Silva

Feira de Santana

2024

Universidade Estadual de Feira de Santana

Programa de Pós-Graduação em Ciência da Computação

Gustavo Falcão Paim da Silva

Modelagem e desenvolvimento de uma arquitetura para reconfiguração de unidades de detecção de emergências em cidades inteligentes

Dissertação apresentada à Universidade Estadual de Feira de Santana como parte dos requisitos para a obtenção do título de Mestre em Ciência da Computação.

Orientador: Prof. Dr. Thiago Cerqueira de Jesus

Coorientador: Prof. Dr. Daniel Gouveia Costa

Feira de Santana

2024

Ficha Catalográfica – Biblioteca Central Julieta Carteado

Silva, Gustavo Falcão Paim da
S58m Modelagem e desenvolvimento de uma arquitetura para reconfiguração de unidades de detecção de emergências em cidades inteligentes./ Gustavo Falcão Paim da Silva. – 2024.
132 f.: il.

Orientador: Thiago Cerqueira de Jesus
Coorientador: Daniel Gouveia Costa
Dissertação (mestrado) – Universidade Estadual de Feira de Santana, Programa de Pós-Graduação em Ciência da Computação, 2024.

1.Sistemas de Detecção de Emergências. 2.Arquitetura baseada em sensores reconfiguráveis. 3.Cidades inteligentes. 4.Internet das coisas. I.Jesus, Thiago Cerqueira de, orient. II.Costa, Daniel Gouveia, coorient. III.Universidade Estadual de Feira de Santana. IV.Titulo.

CDU : 004.72:681.586

Maria de Fátima de Jesus Moreira – Bibliotecária – CRB5/1120

Gustavo Falcão Paim da Silva

Modelagem e desenvolvimento de uma arquitetura para reconfiguração de unidades de detecção de emergências em cidades inteligentes

Dissertação apresentada à Universidade Estadual de Feira de Santana como parte dos requisitos para a obtenção do título de Mestre em Ciência da Computação.

Feira de Santana, 01 de Fevereiro de 2024

BANCA EXAMINADORA

Thiago Cerqueira de Jesus (Orientador(a))
Universidade Estadual de Feira de Santana

Anfranserai Morais Dias
Universidade Estadual de Feira de Santana

Matheus Giovanni Pires
Universidade Estadual de Feira de Santana

Abstract

Emergency detection systems based on sensors play a crucial role in the early identification of critical situations and the activation of appropriate responses, preventing potential disasters. When these systems are built in smart cities, the wide coverage, heterogeneity, and dynamic nature of monitored environments may require the reconfiguration of sensor units to meet new requirements, enabling more effective decisions aligned with the monitoring context. In this context, this dissertation proposes to model and develop an architecture capable of allowing the reconfiguration of sensor-based emergency detection units securely, through remote requests via a wireless communication infrastructure. This enables the inclusion of new monitoring requirements at runtime, ensuring more agile and effective responses in emergency situations. The architecture is described by static and dynamic models that clearly and concisely map the communication structure and the associated hardware and software elements. Additionally, algorithms have been developed to effectively and efficiently describe the process of aggregating new monitoring requirements to emergency detection units. Security procedures and authentication mechanisms have been incorporated into the architecture to promote the integrity and continuous performance of components in the face of cyber threats inherent to IoT and communication systems. It is expected that this work will significantly contribute to the development of emergency detection systems and critical event management in smart cities, considering the dynamics and heterogeneity of urban environments, and providing proactive mechanisms for decision-making.

Keywords: Emergency Detection Systems, Architecture based on Reconfigurable Sensors, Smart Cities, Internet of Things.

Resumo

Sistemas de detecção de emergências baseados em sensores desempenham um papel crucial na identificação precoce de situações críticas e na ativação de respostas apropriadas, prevenindo possíveis desastres. Quando esses sistemas são construídos em cidades inteligentes, a ampla abrangência, a heterogeneidade e a dinamicidade dos ambientes monitorados podem demandar a reconfiguração das unidades de sensores para atender a novos requisitos, possibilitando decisões mais eficazes e alinhadas com o contexto de monitoramento. Nesse contexto, esta dissertação propõe modelar e desenvolver uma arquitetura capaz de permitir a reconfiguração de unidades de detecção de emergências baseadas em sensores, de forma segura, através de solicitações remotas por meio de uma infraestrutura de comunicação sem fio. Isso possibilita a inclusão de novos requisitos de monitoramento em tempo de execução, garantindo respostas mais ágeis e efetivas em situações de emergência. A arquitetura é descrita por modelos estáticos e dinâmicos que mapeiam de forma clara e concisa a estrutura de comunicação e os elementos de *hardware* e *software* associados. Além disso, foram desenvolvidos algoritmos para descrever o processo de agregação de novos requisitos de monitoramento de maneira eficaz e eficiente às unidades de detecção de emergências. Procedimentos de segurança e mecanismos de autenticação foram incorporados à arquitetura com o objetivo de promover a integridade e o desempenho contínuo dos componentes diante das ameaças cibernéticas inerentes à IoT e aos sistemas de comunicação. Espera-se que este trabalho contribua significativamente para o desenvolvimento de sistemas de detecção de emergências e eventos críticos em cidades inteligentes, levando em consideração a dinâmica e heterogeneidade dos ambientes urbanos, e fornecendo mecanismos proativos para tomada de decisões.

Palavras-chave: Sistemas de Detecção de Emergências, Arquitetura baseada em Sensores Reconfiguráveis, Cidades Inteligentes, Internet das Coisas.

Prefácio

Esta dissertação de mestrado foi submetida à Universidade Estadual de Feira de Santana (UEFS) como requisito parcial para obtenção do grau de Mestre em Ciência da Computação.

A dissertação foi desenvolvida no Programa de Pós-Graduação em Ciência da Computação (PGCC), tendo como orientador o **Prof. Dr. Thiago Cerqueira de Jesus**. O **Prof. Dr. Daniel Gouveia Costa** foi coorientador(a) deste trabalho.

Agradecimentos

Primeiramente agradeço a Deus por me guiar nessa trajetória, me dando força e coragem para superar todos os desafios e obstáculos que foram impostos no decorrer do Mestrado.

Agradeço ao meu pai, Edivaldo Oliveira e a minha mãe, Marli Falcão, pela educação que me foi dada para que eu pudesse estar trilhando a minha trajetória e realizando mais uma conquista em minha vida. Agradeço também a minha namorada, Ianka Cézar, pelo apoio que me foi dado durante os momentos mais difíceis que passei no decorrer da escrita deste trabalho.

Agradeço também aos meus orientadores, o Professor Dr. Thiago Cerqueira de Jesus e o Professor Dr. Daniel Gouveia Costa, por todo o apoio, compreensão, orientação, motivação e puxões de orelha que foram dados ao longo de todo este processo acadêmico.

Também deixo meus agradecimentos aos colegas do Instituto Federal da Bahia - Campus Irecê pelos conselhos, apoio, compreensão e suporte durante esse processo de jornada dupla, algumas vezes tripla, onde precisei conciliar a docência, a coordenação do Curso Técnico em Informática e as atividades de estudante de pós-graduação durante um momento pandêmico tão conturbado.

Para finalizar, deixo aqui minha eterna gratidão e grande admiração a Leonard Kleinrock, Paul Baran e Donald Davies, pois sem eles não faria sentido o que está sendo proposto nesta dissertação. Agradeço também a Tim Berners-Lee e a Alexandra Elbakyan pelos seus trabalhos e contribuições. Sem estes, seria extremamente difícil e complexo obter todo o volume de informações para o desenvolvimento desta dissertação.

“Dedico este trabalho ao meu pai,
à minha mãe e ao meu irmão”

Sumário

Alinhamento com a Linha de Pesquisa

Linha de Pesquisa: Software e Sistemas Computacionais

O presente trabalho enquadra-se na linha de pesquisa “Software e Sistemas Computacionais” e busca contribuir para o campo de pesquisa em Sistemas de Gerenciamento de Emergências em Cidades Inteligentes por meio da utilização de tecnologias associadas ao paradigma da Internet das Coisas.

O objetivo principal deste trabalho é modelar e desenvolver uma arquitetura capaz de permitir a reconfiguração de unidades de detecção de emergências em cidades inteligentes, de forma segura, por meio de solicitações remotas através de uma infraestrutura de comunicação sem fio. A arquitetura aqui descrita permitirá a agregação de novos requisitos de monitoramento de eventos de interesse, alcançando um alto grau de implementação, automação e escalabilidade em tempo de execução.

Dessa forma, espera-se que este trabalho possa contribuir para o monitoramento e detecção de eventos de emergência de forma mais fidedigna à realidade dos ambientes urbanos sensorizados, proporcionando um mecanismo proativo em relação à tomada de decisões e à execução de ações frente a problemas inerentes a esse contexto. O alinhamento de tecnologias digitais e disruptivas com o progresso social e ambiental poderá contribuir para uma sociedade mais conectada, eficiente, sustentável e segura, trazendo inúmeros benefícios para todas as esferas institucionais.

Lista de Publicações

G. F. Silva, D. G. Costa and T. C. Jesus, “A Framework for the Development of Reconfigurable Sensors-based Emergencies Detection Units in Smart Cities” 2022 IEEE International Smart Cities Conference (ISC2), Pafos, Cyprus, 2022, pp. 1-4, doi: 10.1109/ISC255366.2022.9922506.

G. F. P. Da Silva, D. G. Costa and T. C. De Jesus, “A Secure OTA Approach For Flexible Operation of Emergency Detection Units in Smart Cities,” 2023 IEEE International Smart Cities Conference (ISC2), Bucharest, Romania, 2023, pp. 01-07, doi: 10.1109/ISC257844.2023.10293637.

Lista de Tabelas

Lista de Figuras

Lista de Abreviações

Abreviação	**Descrição**
3G	*Third Generation Cellular Networks* (Terceira Geração de Telefonia Móvel)
4G	*Fourth Generation Cellular Networks* (Quarta Geração de Telefonia Móvel)
5G	*Fifth Generation Cellular Networks* (Quinta Geração de Telefonia Móvel)
6LoWPAN	*IPv6 over Low-Power Wireless Personal Area Networks* (IPv6 sobre Redes Pessoais sem Fio de Baixo Consumo de Energia)
AAA	*Authentication, Authorization and Accounting* (Autenticação, Autorização e Auditoria)
AMQP	*Advanced Message Queuing Protocol* (Protocolo Avançado de Enfileiramento de Mensagens)
EDU	*Emergency Detection Units* (Unidade de Detecção de Emergência)
HTTPS	*Hypertext Transfer Protocol Secure* (Protocolo de Transferência de Hipertexto Seguro)
I2C	*Inter-Integrated Circuit* (Circuito Interintegrado)
IaaS	*Infrastructure as a Service* (Infraestrutura como Serviço)
IoT	*Internet of Things* (Internet das Coisas)
IP	*Internet Protocol* (Protocolo de Internet)
LED	*Light-Emiting Diode* (Diodo Emissor de Luz)
JSON	*JavaScript Object Notation* (Notação de Objeto em JavaScript)
LPWAN	*Low-Power Wide-Area Network* (Rede de Área Ampla de Baixa Potência)
LTE-Advanced	*Long Term Evolution Advanced* (Evolução Avançada de Longo Prazo)
M2M	*Machine-to-Machine* (Máquina a Máquina)
MQTT	*Message Queuing Telemetry Transport* (Transporte de Filas de Mensagem de Telemetria)
NB-IoT	*Narrowband Internet of Things* (Internet das Coisas de Banda Estreita)

OTA	*Over-The-Air* (Pelo ar)
PaaS	*Platform as a Service* (Plataforma como Serviço)
QoS	*Quality of Service* (Qualidade de Serviço)
RFC	*Request for Comments* (Pedido de Comentários)
RFID	*Radio Frequency Identification* (Identificação por Radiofrequência)
SaaS	*Software as a Service* (Software como Serviço)
SHA-256	*Secure Hash Algorithm 256* (Algoritmo de Hash Seguro 256)
SMS	*Short Message Service* (Serviço de Mensagens Curtas)
SOA	*Service-Oriented Architecture* (Arquitetura Orientada a Serviços)
SPI	*Serial Peripheral Interface* (Interface Periférica Serial)
SSL	*Secure Sockets Layer* (Camada de Soquetes Seguros)
STaaS	*Storage as a Service* (Armazenamento como Serviço)
TCP	*Transmission Control Protocol* (Protocolo de Controle de Transmissão)
TLS	*Transport Layer Security* (Segurança da Camada de Transporte)
UART	*Universal Asynchronous Receiver/Transmitter* (Receptor/-Transmissor Assíncrono Universal)
WI-FI	*Wireless Fidelity* (Fidelidade sem Fio)
WSN	*Wireless Sensor Networks* (Redes de Sensores sem Fio)
XMPP	*Extensible Messaging and Presence Protocol* (Protocolo de Presença e Mensagens Extensível)

Capítulo 1

Introdução

A Revolução Industrial foi definida como um período de grande desenvolvimento tecnológico, transformações sociais e econômicas que se espalharam pelo mundo, promovendo o surgimento de novos processos fabris que barateaaram os custos de produção de mercadorias através da produção mecanizada em larga escala.

Entretanto, esse processo de industrialização intensificou a urbanização maciça e desordenada dos núcleos urbanos, uma vez que áreas industrializadas trazem consigo novas oportunidades e melhores condições de vida para a população. Assim, as cidades careciam de infraestrutura básica para suportar toda essa massa humana convergente. As ocupações inadequadas e irregulares dos centros urbanos acarretaram o surgimento de diversos problemas inerentes a esse contexto, destacando-se problemas ambientais como deslizamentos de terra, enchentes, ilhas de calor, entre outros.

A heterogeneidade e dinamicidade dos centros urbanos representam grandes desafios que requerem a promoção de estratégias tecnológicas para o gerenciamento eficiente de diferentes cenários, desde situações cotidianas, até o gerenciamento de emergências. Este último caso se caracteriza por situações muitas vezes inesperadas e potencialmente perigosas, nas quais o tempo de resposta aos eventos causadores se torna crítico. Dessa forma, a gestão de emergências e a prevenção de desastres tem ganhado destaque à medida que o processo de urbanização se intensifica e os impactos negativos deste fenômeno se tornam mais evidentes (Costa et al., 2020b).

Diversas iniciativas vêm sendo desenvolvidas em todo o mundo como uma forma eficaz de enfrentar alguns dos principais desafios das áreas urbanas, abordando vários aspectos da detecção, alerta e mitigação de emergências (Rangra e Sehgal, 2022; Alazawi et al., 2014; Shah et al., 2019). Aplicações baseadas no paradigma de internet das coisas (do inglês, *internet of things* -

IoT) têm sido exploradas como uma tecnologia acessível para recuperar dados importantes através do uso de dispositivos sensores em diferentes áreas urbanas e processá-los, possibilitando tomadas de decisões mais eficazes e coerentes com a realidade correspondente (Costa et al., 2022a).

Nesse cenário favorável à utilização de tecnologias para apoiar e melhorar a eficiência de serviços prestados aos cidadãos urbanos, as cidades inteligentes surgem com uma importante aplicação de IoT, trazendo desafios complexos em diferentes aspectos relacionados à estrutura das cidades e ao comportamento de seus habitantes, abrindo muitas oportunidades comerciais, industriais, educacionais e culturais (Costa e Duran-Faundez, 2018), proporcionando assim, um ambiente e qualidade de vida melhor para sua população.

Ao criar soluções de detecção de emergências baseadas no paradigma de IoT para cidades inteligentes, os dispositivos de monitoramento necessários podem variar consideravelmente de acordo com os parâmetros a serem detectados, frequências de amostragem, estratégias de priorização e outros algoritmos de processamento empregados. Além disso, os mesmos sistemas de detecção de emergências definidos para uma determinada cidade podem ter que atender a requisitos diferentes que só serão conhecidos após sua implantação inicial. A Figura 1.1 ilustra diferentes cenários de eventos críticos presentes em um mesmo contexto monitorado.

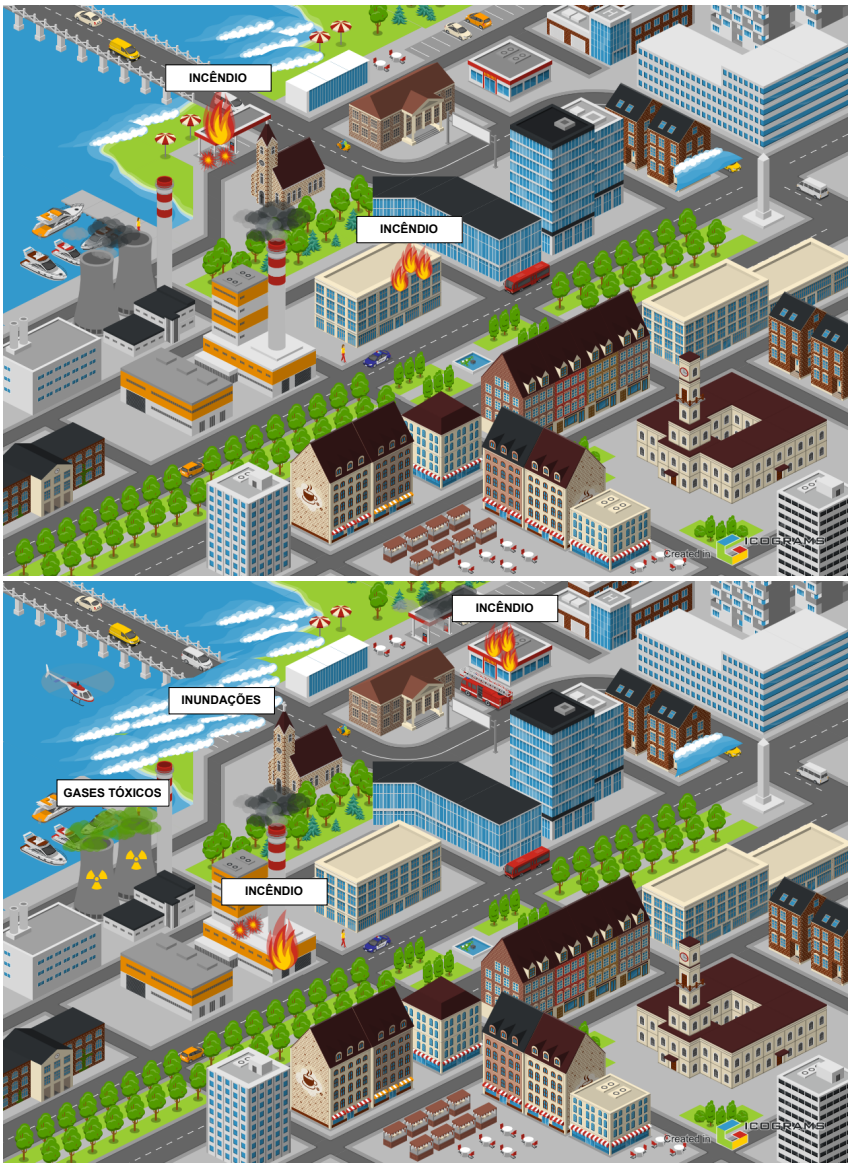
As soluções de sensoriamento existentes geralmente se concentram em um único fenômeno de monitoramento, como é o caso de aplicações para detecção de incêndios (Mahgoub et al., 2020; Sheth et al., 2020) e inundações (Chen et al., 2022), tornando-as limitadas no sentido de não se adaptarem a diferentes cenários ou ambientes com grande variabilidade de eventos (Costa et al., 2020a).

Nesses casos, Unidades de Detecção de Emergências (EDUs[1]) podem ser empregadas para o recolhimento de informações precisas que irão subsidiar a geração de alertas e a mitigação de emergências (Costa et al., 2022b). Estas unidades são definidas como sistemas computacionais multi-sensores e são implantados em uma região de interesse com a finalidade de coletar, processar e detectar mudanças nos dados ambientais e urbanos, transmitindo-os por redes sem fio para um centro de operações de emergência. Isso inicia uma resposta imediata ao evento ou, em alguns casos, realiza processamento local para aumentar a eficiência (Du et al., 2018; Peixoto et al., 2024).

No entanto, a utilização de EDUs em ambientes urbanos, onde a diversidade de eventos de emergência exige respostas imediatas, demanda sua capacidade

[1]De modo a manter a compatibilidade com a literatura e com os trabalhos publicados derivados desta dissertação (Silva et al., 2022; Da Silva et al., 2023), será mantida a sigla EDU (representação de *Emergency Detection Units*) para referenciar este componente.

Figura 1.1: Diferentes situações de emergências em um mesmo ambiente de monitoramento.



Fonte: O autor.

de adaptação às constantes mudanças. Isso enfatiza a necessidade de reconfiguração dessas unidades, visto que às transformações temporais do ambiente monitorado (mudanças sazonais, variações na densidade populacional ao longo do dia, flutuações nos padrões de tráfego, alterações nas condições climáticas, entre outros) influenciam a dinâmica do ambiente e, consequentemente, as exigências sobre o sistema de detecção de emergências. Além disso, o processo de reconfiguração promove a realização de aprimoramentos contínuos, abordando eventuais falhas (*bugs*) e refinando tanto o comporta-

mento de sensoriamento quanto as métricas, visando garantir um desempenho ótimo do sistema. Adicionalmente, é essencial considerar a possibilidade de incorporação de novos requisitos de monitoramento, mediante a adição de novos dispositivos sensores às EDUs, para garantir que o sistema permaneça alinhado com as demandas emergentes.

Para atender aos requisitos esperados de uma EDU, é crucial especificar uma infraestrutura de comunicação apropriada, implementar mecanismos de segurança robustos e definir componentes capazes de suportar e processar solicitações de reconfiguração de acordo com a mutabilidade do contexto monitorado.

Assim, o presente trabalho propõe a modelagem e a construção de uma arquitetura que oriente desenvolvimento, implantação e operação de EDUs reconfiguráveis para suportar o monitoramento e a detecção de eventos emergenciais em ambientes urbanos, contribuindo para a eficiência das aplicações de IoT no gerenciamento de emergências em cidades inteligentes. Além disso, serão delineadas características fundamentais de uma arquitetura de IoT segura, que possibilite a incorporação de novos requisitos de monitoramento por meio de solicitações remotas através de uma infraestrutura de comunicação sem fio. Isso garantirá a integração desses novos requisitos em tempo de execução, sem prejudicar a operação dos demais dispositivos, resultando em tempos de resposta cada vez menores diante de eventos críticos.

1.1 Objetivos

1.1.1 Objetivo geral

Este trabalho tem por objetivo modelar e desenvolver uma arquitetura capaz de permitir a reconfiguração de EDUs em cidades inteligentes, de forma segura, através de solicitações remotas por meio de uma infraestrutura de comunicação sem fio. A arquitetura aqui descrita permitirá a agregação de novos requisitos de monitoramento de eventos de interesse, alcançando um alto grau de implantação, automação e escalabilidade em tempo de execução.

1.1.2 Objetivos específicos

- Definir componentes de *hardware*, *software* e protocolos de comunicação associados de modo a projetar uma arquitetura capaz de orientar o desenvolvimento, implantação e operação de EDUs reconfiguráveis.
- Formalizar a comunicação e a integração entre os elementos pertencentes à arquitetura, permitindo que EDUs sejam reconfiguradas remota-

mente, conforme requisitos do ambiente monitorado.

- Especificar parâmetros de segurança fundamentais que deverão ser utilizados para garantir níveis adequados de segurança e operacionalização dos elementos pertencentes à arquitetura.
- Projetar uma EDU utilizando uma plataforma de *hardware* aberta e de baixo custo para integrá-la à arquitetura proposta, a fim de avaliar seu comportamento diante das solicitações de reconfiguração.
- Validar a viabilidade técnica da arquitetura proposta através da realização de prova de conceito.

1.2 Relevância

Segundo a Organização das Nações Unidas, atualmente 55% da população mundial vive em áreas urbanas e a expectativa é de que esta proporção aumente para 70% até 2050. No Brasil, por exemplo, a rapidez do processo de urbanização gerou o crescimento exponencial de cidades desordenadas, sobretudo nas regiões metropolitanas, marcadas por fortes políticas de incentivo que desencadearam a ampliação da ocupação de áreas de riscos, dentro dos espaços urbanos (Silva Júnior e Chaves, 2021).

O desenvolvimento de aplicações voltadas para o gerenciamento de emergências visa sobretudo auxiliar o poder público no processo de tomadas de decisões mais eficazes em relação a situações emergenciais inerentes ao contexto desordenado de centros urbanos, que se não observadas de imediato, podem resultar em danos físicos às pessoas e também causar perdas econômicas em magnitude (Costa et al., 2020a, 2022a).

Deste modo, a definição de uma arquitetura capaz de orientar o desenvolvimento e operação de EDUs adaptáveis aos contexto de monitoramento é um passo importante no que se refere à construção de aplicações baseadas no paradigma de IoT para o gerenciamento eficiente de emergências. A incorporação de novos requisitos de monitoramento em tempo de execução, sem degradar a operação vigente dos dispositivos, que promova tempos de interrupções ou de inatividade cada vez menores, é algo de extrema relevância uma vez que a natureza heterogênea e dinâmica das cidades representam grandes desafios com impacto direto no bem-estar e segurança da população.

1.3 Contribuições

As principais contribuições deste trabalho são:

- Proposição de uma arquitetura capaz de facilitar o desenvolvimento, implantação e operação de EDUs, baseadas em sensores, adaptáveis ao contexto por meio da incorporação segura de novos requisitos de monitoramento através de solicitações remotas em uma infraestrutura de comunicação sem fio.
- Formalização de elementos de comunicação e a integração destes de modo a permitir a agregação de novos requisitos de monitoramento às EDUs em tempo de execução, alcançando um alto grau de implantação, automação e escalabilidade para apoiar a detecção de eventos críticos em cidades de forma mais eficaz e coerente com a realidade monitorada.

1.4 Organização do Trabalho

No Capítulo 2, é apresentada a revisão bibliográfica dos temas e técnicas importantes que norteiam a elaboração deste trabalho. O Capítulo 3 apresenta a formalização do problema a ser solucionado, bem como a definição da arquitetura de comunicação e dos elementos de *hardware* e *software* associados através da especificação de um conjunto de regras, protocolos e padrões que irão governar a comunicação entre os dispositivos. Neste capítulo, também é apresentado o processo para a formalização dos modelos conceituais e os mecanismos adotados para o desenvolvimento, implantação, testes e validação da arquitetura proposta. O Capítulo 4 descreve detalhadamente a formalização dos elementos de comunicação e a integração destes em uma arquitetura, possibilitando que as EDUs sejam reconfiguradas de forma segura, implementando novos requisitos de monitoramento em tempo de execução através de solicitações remotas por meio de uma infraestrutura de comunicação sem fio, seguindo a abordagem *Over-the-air*. No Capítulo 5, são apresentadas as diferentes tecnologias e ferramentas utilizadas para implementar as funcionalidades propostas neste trabalho, permitindo uma avaliação inicial com base em uma abordagem de prova de conceito. Por fim, o Capítulo 6 discute as conclusões diante da prova de conceito desenvolvida e os resultados obtidos. Além disso, são apresentados trabalhos de pesquisas futuras que deverão ser produzidos tendo em vista cenários de aplicação real e implementações em larga escala da arquitetura proposta.

Capítulo 2

Revisão Bibliográfica

No século XX, período em que a Revolução Industrial se dissemina pelo mundo, observa-se um significativo crescimento populacional, inaugurando o modelo de urbanização conhecido atualmente. A população mundial teve um aumento considerável, passando de cerca de 2,5 bilhões em 1950 para 7 bilhões em 2011 (Amaral, 2019). Entretanto, o processo acelerado e, muitas vezes desordenado, de ocupação dos espaços urbanos, juntamente com o elevado crescimento populacional promovido por esse importante processo histórico, resultou em graves problemas ambientais e urbanos. Todos esses fatores contribuem significativamente para que as cidades se tornem altamente suscetíveis a catástrofes decorrentes de situações de emergência (Costa et al., 2022a).

Adicionalmente, segundo dados das Nações Unidas (Gerland et al., 2022), a população global pode chegar a cerca de 8,5 bilhões em 2030 e adicionar mais 1,18 bilhão nas duas décadas seguintes, atingindo 9,7 bilhões em 2050. Estima-se que 70% desse aumento populacional se concentre em ambientes urbanos, o que acarretará em uma série de desafios para o desenvolvimento de soluções e ações que minimizem os impactos negativos dessa nova configuração dos espaços urbanos.

Nesse cenário já complexo, com áreas urbanas cada vez mais superlotadas, diferentes tipos de situações de emergência podem ocorrer em qualquer lugar e a qualquer momento, com impactos incalculáveis nas vidas humanas, no meio ambiente e na economia (Costa et al., 2022a). A heterogeneidade e a dinamicidade dos ambientes urbanos, aliadas à sua grande extensão territorial, representam desafios significativos e exigem a adoção de estratégias tecnológicas eficientes para o gerenciamento de emergências. Situações emergenciais, muitas vezes inesperadas e potencialmente perigosas, demandam uma resposta rápida, onde o tempo de reação aos eventos se torna crucial.

A implementação de mecanismos capazes de responder prontamente a eventos críticos, de maneira coerente com o contexto monitorado, é de suma importância para o gerenciamento de riscos em ambientes urbanos. A falta de observação imediata e ação correta diante dessas situações podem resultar em consequências catastróficas.

Neste capítulo, será apresentada uma fundamentação teórica que aborda os principais conceitos e questões para uma melhor compreensão das ideias aqui discutidas. Dentre esses conceitos, destacam-se: o uso das Redes de Sensores sem Fio (RSSF) como tecnologia de conexão para dispositivos de nós sensores responsáveis pela medição e coleta de variáveis ambientais; o paradigma da Internet das Coisas (IoT) e seu arcabouço associado, permitindo compreender quais arquiteturas, elementos, protocolos e mecanismos de segurança podem ser empregados para a construção de aplicações; as cidades inteligentes e o gerenciamento de emergências em cidades inteligentes, que constituem o cenário de interesse do modelo proposto neste trabalho.

2.1 Urbanização e Planejamento Urbano

A urbanização é um fenômeno que remonta ao surgimento das primeiras civilizações, sendo caracterizada como uma tendência de desenvolvimento comum a países em todo o mundo, marcando o progresso da civilização humana. O processo de urbanização pode ser definido como a concentração populacional, manifestada de duas maneiras: pela multiplicação dos pontos de concentração e pelo aumento do tamanho das concentrações individuais (Tisdale, 1942).

Embora o movimento de urbanização seja transitório, este processo é culturalmente traumático, devido à total transformação espacial e à consequente transformação institucional e social envolvida (Henderson, 2005). A ocupação desordenada e acelerada dos espaços urbanos impacta diretamente a qualidade de vida da população, pois muitas vezes os aglomerados urbanos não apresentam condições satisfatórias para a instalação dessa grande massa humana, devido à falta de infraestrutura e equipamentos urbanos básicos (água, esgoto, gás, eletricidade e serviços urbanos como transporte, educação e saúde), além de condições precárias de moradia (Suriano e Reschilian, 2012; Silva et al., 2014).

Portanto, há uma necessidade por parte do setor público de propor e desenvolver programas e medidas públicas para melhorar e requalificar os espaços urbanos, sinalizando quais medidas devem ser tomadas para melhorar a qualidade de vida da população urbana. Isso inclui questões relacionadas aos equipamentos urbanos básicos e um modo de vida mais sustentável, promovendo o uso eficiente dos recursos ambientais. O planejamento urbano possibilita o

mapeamento da realidade dos ambientes urbanos, fornecendo um mecanismo proativo para a tomada de decisões e a execução de ações em resposta aos problemas inerentes a esse contexto.

2.1.1 Desafios recorrentes da urbanização

O processo de urbanização e toda a problemática associada a este movimento de ocupação desordenada em ambientes muitas vezes despreparados, tornou as cidades altamente suscetíveis a catástrofes decorrentes de situações de emergência.

De acordo com Jatobá (2011), a ocorrência de desastres em áreas urbanas, principalmente aqueles provocados por fenômenos naturais, têm se intensificado à medida que a própria urbanização se acelera. Eventos como terremotos, tsunamis, furações, chuvas intensas, invernos rigorosos e secas prolongadas parecem ter consequências cada vez mais extensas e graves à proporção que as cidades se expandem, se adensam e a população urbana cresce.

Este tema tem sido tratado com frequência, sendo motivo de interesse científico e de grande preocupação por parte de organismos internacionais, além de ser assunto recorrente na imprensa a cada nova tragédia que associa perdas humanas e materiais a eventos de desastres naturais. Alguns exemplos proeminentes são: a liberação de gases tóxicos pelas indústrias do polo petroquímico de Cubatão em 1980 (Klanovicz e Ferreira Filho, 2018); o incêndio na Vila de Socó em Cubatão no ano de 1984 (Porto, 2016); o rompimento da barragem do Fundão em Mariana em 2015 (Porto, 2016); e o rompimento da barragem Mina do Feijão em Brumadinho em 2019 (Freitas et al., 2019).

As cidades são ambientes altamente dinâmicos e heterogêneos, sujeitos a constantes mudanças causadas tanto pela intervenção humana quanto pelos fenômenos naturais. Portanto, o processo de urbanização traz consigo uma série de desafios para o setor público e para a população que depende dos serviços e da infraestrutura oferecida. Uma maneira de otimizar o funcionamento das infraestruturas e dos serviços públicos, visando proporcionar uma vida mais conveniente, segura e sustentável para os habitantes, é por meio do uso de Tecnologias da Informação e Comunicação (TICs).

Atualmente, considerando as novas tecnologias disponíveis, têm-se buscado soluções mais eficientes para minimizar os impactos negativos das emergências. A disponibilidade de novos recursos tecnológicos é um fator para o avanço para o desenvolvimento de cidades sustentáveis, resilientes e inteligentes (Costa et al., 2022a).

O uso de sensores eletrônicos, por exemplo, podem auxiliar na detecção e prevenção de situações de emergência em áreas urbanas. Além de dispositivos

sensores, estes eventos podem ser melhores detectados e gerenciados com o uso de *smartphones*, dispositivos pessoais, veículos, *drones*, robôs, redes de mídia social, servidores da web e serviços baseados na nuvem. A adoção destas soluções possibilita tomadas de decisões mais eficazes e coerentes com a realidade correspondente (Costa et al., 2022a).

2.2 Rede de Sensores Sem Fio - RSSF

Uma Rede de Sensores sem Fio (RSSF) caracteriza-se como uma tecnologia de redes de comunicação que interconecta, ao invés de computadores, um conjunto de diversos dispositivos embarcados, chamados nós sensores (Gonçalves, 2015). Esses nós geralmente possuem recursos computacionais e suprimentos de energia bastante limitados, mas apresentam amplo poder de sensoriamento, sendo capazes de detectar e reagir a ocorrências de mudanças, além de se comunicar com outros dispositivos em uma área geográfica específica, com o propósito de realizar uma função específica.

Os nós sensores podem ser equipados com diversos dispositivos de sensoriamento e são responsáveis pela medição e coleta de variáveis ambientais relevantes à aplicação em questão, como umidade do ar, temperatura, níveis de ruídos sonoros, presença de gases tóxicos no ar, velocidade do vento, entre outras características físico-químicas ambientais (Yick et al., 2008). O uso colaborativo de uma quantidade suficiente de dispositivos sensores permite que uma RSSF realize a aquisição simultânea de um grande volume de dados em vários pontos de interesse posicionados em áreas amplas, conforme os requisitos das aplicações (Kandris et al., 2020).

Diversas são as áreas de aplicabilidade de RSSF e seu uso tem sido bastante efetivo, uma vez que esta tecnologia permite coletar dados que seriam muito difíceis de serem obtidos, por exemplo, em áreas remotas ou em ambientes que mudam constantemente. Além disso, as redes de sensores sem fio permitem a coleta de dados em tempo real e podem ser escaladas facilmente, adicionando ou removendo sensores conforme necessário. Posto isto, esta tecnologia é de grande valia para a construção de aplicações voltadas para detecção de eventos de emergência em ambientes urbanos.

2.3 Internet das Coisas - IoT

O advento de novas tecnologias e a incorporação destas nas atividades cotidianas fomenta a ideia de que haverá sempre uma solução de tecnologia da informação e comunicação capaz de lidar de maneira satisfatória com novos

desafios sociais. Recentemente, a solução que é proposta quase sempre é a IoT (Atzori et al., 2017).

O conceito de IoT refere-se a um paradigma de interconexão avançado de diversos objetos do cotidiano (dotados de tecnologia embarcada, sensores e conexão com a rede), capazes de reunir e de transmitir dados através da internet, promovendo sistemas de comunicação inteligentes inter-objetos, máquinas, infraestruturas e também ambientes (Ashton, 2009).

Sendo considerado como uma extensão da internet que abarca uma ampla variedade de protocolos, domínios e aplicações (Kaur, 2018), a IoT tem recebido bastante atenção tanto no meio acadêmico, como um campo de estudo bastante extenso, quanto na indústria, devido ao seu potencial de mercado através da sua aplicação nos mais diversos contextos da sociedade.

Ao contrário dos sistemas cibernéticos tradicionais, que conectam computadores de uso geral, os sistemas IoT geralmente conectam dispositivos altamente especializados, projetados para fins específicos com apenas um grau limitado de programabilidade e personalização (Yoo, 2019). Sistemas IoT geralmente armazenam e processam dados de maneira distribuída, em contraste com a abordagem tradicional de centralização de armazenamento e processamento em grandes *data centers*. Além disso, os sistemas IoT às vezes são chamados de sistemas ciber físicos, porque, ao contrário dos sistemas puramente cibernéticos, eles também incluem sensores que coletam dados do mundo físico (Yoo, 2019).

Atualmente, é possível observar uma série de aplicações baseadas no paradigma da IoT, e isso se deve aos avanços tecnológico em diversas áreas do conhecimento, como eletrônica, programação, redes de computadores, inteligência artificial, visão computacional (Oliveira, 2021).

Ao longo dos anos, a IoT vivenciou três estágios evolutivos que foram marcados por tecnologias e objetivos distintos. De acordo com Atzori et al. (2017):

- A primeira geração foi caracterizada através da ideia de objetos marcados, arquiteturas de comunicação entre máquinas (*Machine-to-Machine - M2M*) e a integração entre tecnologias de Identificação por Radiofrequência (RFID) e o uso de Redes de Sensores sem Fio (RSSF).
- A segunda geração da IoT foi marcada pelo uso da arquitetura de comunicação TCP/IP e os seus respectivos protocolos de modo a promover o acesso, compartilhamento de dados e ações colaborativas entre objetos na internet além de descrever os recursos que cada objetivo deve possuir para promover a interoperabilidade entre os sistemas.
- A terceira geração aborda aspectos relacionados à internet do futuro, como por exemplo, o uso de comunicações orientadas à conteúdo ao in-

vés da orientação à dispositivos, o uso da infraestrutura de comunicação na Nuvem, possibilitando aos dispositivos realizarem a comunicação, armazenamento e processamento de dados em infraestruturas de serviços remotas além da integração avançada entre RFID e IoT em aplicações variadas.

A ideia de IoT, portanto, evoluiu ao longo do tempo e passou por transformações sucessivas que, previsivelmente, ainda continuarão nos próximos anos com o advento de novas tecnologias capacitadoras, como por exemplo, novos conceitos relacionados à computação em nuvem, redes centradas em informações, *big data*, redes sociais dentre outros paradigmas futuristas propensos aos impactos e uso da IoT (Atzori et al., 2017).

2.3.1 Arquiteturas de IoT

Diante da rápida evolução dos objetos conectados, torna-se essencial o desenvolvimento de arquiteturas adaptáveis e flexíveis, considerando os diferentes tipos e escopos, dada a diversidade de domínios de aplicações existentes. Entretanto, não há um consenso único sobre a arquitetura IoT que seja universalmente adotada (Elhadi et al., 2018).

A ausência de um padrão arquitetural de referência pode tornar o processo de construção e implantação de sistemas e aplicações de IoT difícil e demorado. Diversas arquiteturas têm sido propostas por diferentes pesquisadores (Sethi e Sarangi, 2017). Investimentos em estudos e pesquisas têm sido realizados para compreender a integração de diferentes arquiteturas existentes na resolução de problemas específicos.

Embora existam várias arquiteturas propostas para IoT, ainda não há convergência para formar um modelo de referência único (Kaur, 2018). Devido à sua abrangência, envolvendo tecnologias de redes de comunicação e computação, e à heterogeneidade de objetos interconectados, a arquitetura de IoT deve ser flexível por natureza. Assim, para atender a esses requisitos, foi desenvolvida uma arquitetura que pode ser considerada a mais utilizada no mundo da IoT: a arquitetura de três camadas (Sethi e Sarangi, 2017).

Este é um dos modelos mais usados até agora. É composto por três camadas básicas, definidas como camada de aplicação, camada de rede e camada de percepção. Cada camada é brevemente descrita na Figura 2.1.

A camada de percepção ou camada de sensores é implementada na parte inferior da Arquitetura. Esta camada possui diversos componentes físicos como sensores e atuadores para a detecção, coleta, medição e processamento de informações sobre o meio onde está inserida além de transmitir as informações obtidas para as camadas superiores (Kaur, 2018).

Figura 2.1: Arquitetura de Três Camadas - IoT.

Fonte: Adaptado de Sethi e Sarangi (2017).

Para que os dados obtidos através da camada de percepção possam ser fornecidos para usuários e aplicações, se faz necessário uma infraestrutura de comunicação que promova o seu transporte. Para isto, a arquitetura de três camadas define a camada de rede.

A camada de rede recebe a informação passada pela camada de percepção e, através de rotas de comunicação obtidas por protocolos de rede, a transporta entre os objetos através da rede integrada para um nó central, que podem ser *switchs*, roteadores, servidores, *hubs* dentre outros. Este nó central também fornece serviços de dados como agregação, análise e processamento de dados além de outros tarefas computacionais (Kaur, 2018).

A camada superior da arquitetura é a camada de aplicação. Esta camada recebe dados provenientes da camada de rede e os utiliza para fornecer os serviços necessários. Dentre os serviços fornecidos pela camada de aplicação, é possível destacar: serviços como armazenamento para fornecer *backup* e serviços de análise para avaliar informações e prever o estado futuro dos dispositivos conectados. A camada de aplicação define várias aplicações nas quais a IoT pode ser implementada, como casas inteligentes, cidades inteligentes, redes de transporte inteligente, saúde inteligente, entre outros (Sethi e Sarangi, 2017), (Kaur, 2018).

Vários estudos sugerem a adoção de novas camadas à arquitetura de três

camadas com o objetivo de facilitar a interoperabilidade de dispositivos heterogêneos de IoT (Muccini e Moghaddam, 2018). A arquitetura de quatro camadas, por exemplo, adiciona a camada de negócios na parte superior e é responsável pelo manuseio de todo o sistema IoT. A camada de negócios possibilita apoiar processos de tomada de decisão baseados na análise de *Big Data*. Além disso, o monitoramento e o gerenciamento das quatro camadas subjacentes são alcançados nessa camada através da análise de dados e produção de relatórios de alto nível (Al-Fuqaha et al., 2015).

A arquitetura de cinco camadas possui as mesmas três camadas do modelo tradicional e adiciona mais duas camadas, intituladas camada de *gateway* de acesso e camada de *middleware*. A camada de *gateway* de acesso é responsável por gerenciar a comunicação IoT no ambiente específico e trocar mensagens entre objetos e sistemas, enquanto a camada de *middleware* fornece uma associação mais flexível entre dispositivos de *hardware* e aplicativos (Al-Qaseemi et al., 2016).

Em geral, a arquitetura de três camadas é bastante utilizada em projetos e soluções de IoT (Oliveira, 2021). Alguns autores sugerem a arquitetura de cinco camadas como o modelo mais indicado para aplicações de IoT devido às funcionalidades adicionais provenientes das camadas de negócio, que agregam valor às aplicações, e à possibilidade de aplicativos de IoT trabalharem com objetos heterogêneos sem considerar uma plataforma de *hardware* específica (Al-Qaseemi et al., 2016).

Entretanto, na literatura recente, foram propostos alguns outros modelos arquiteturais, os quais adicionam mais abstração ao desenvolvimento de sistemas de IoT. Essas abstrações têm como objetivo solucionar problemas imediatos, adaptando sistemas de IoT aos contextos computacionais atuais. Por exemplo, destacam-se o uso de *big data*, interoperabilidade, escalabilidade, armazenamento, segurança e privacidade, além de se adequarem a cenários complexos e dinâmicos de monitoramento. Tais cenários exigem a reconfiguração desses sistemas sem que haja a degradação dos serviços prestados (Al-Fuqaha et al., 2015).

Com o objetivo de fornecer uma base sólida para classificar abordagens existentes e futuras para estilos de IoT benéficos para pesquisadores acadêmicos e industriais, Muccini e Moghaddam (2018) apresentam um estudo de mapeamento sistemático com o objetivo de classificar e identificar o estado da arte acerca do conjunto de arquiteturas de referência abstratas de IoT para serem aplicáveis no desenvolvimento de sistemas de IoT de vários estilos arquitetônicos. Neste referido estudo, além dos padrões arquiteturais convencionais (arquiteturas de três, quarto e cinco camadas), foram observados outros padrões de arquiteturas para sistemas de IoT bastante utilizadas, como é o caso

das arquiteturas baseada em nuvem, orientadas a serviços, microsserviços e publicar/assinar.

Arquitetura Baseada em Nuvem

A arquitetura baseada em nuvem provê capacidade de processar e armazenar grandes quantidades de dados e fornecer informações contextuais além de possibilitar maior flexibilidade e escalabilidade para sistemas de IoT. Esta arquitetura é caracterizada pelo fornecimento de vários serviços para sistemas de IoT, por exemplo, Infraestrutura como Serviço (IaaS) que fornece recursos computacionais virtualizados; Armazenamento de Dados como Serviço (STaaS), que fornece armazenamento de dados e recuperação de informações por um gerenciador de banco de dados; Plataforma como Serviço (PaaS), que fornece as ferramentas para trabalhar com dispositivos e máquinas na nuvem; e *Software* como Serviço (SaaS), que fornece recursos aos usuários para interpretação e visualização de dados na nuvem (Muccini e Moghaddam, 2018), (Sethi e Sarangi, 2017).

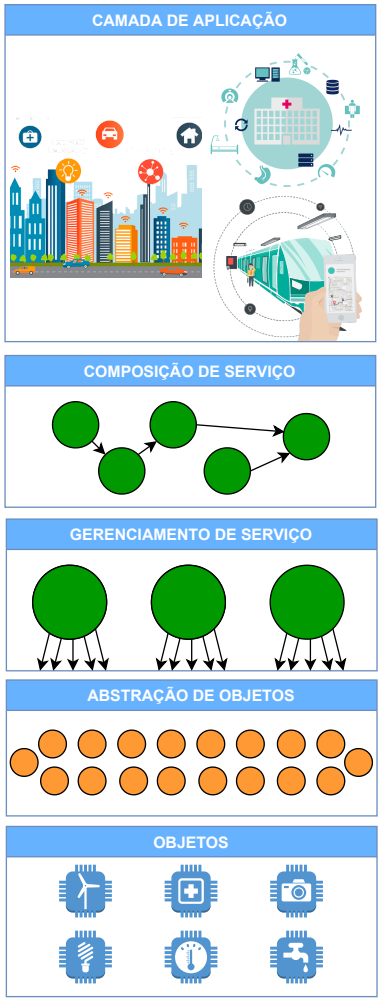
Arquitetura Orientada a Serviços

Na Arquitetura Orientada a Serviços (SOA), as camadas de composição de serviços, gerenciamento de serviços e abstração de objetos são introduzidas entre as camadas de aplicação e percepção (em referência à arquitetura de três camadas) para tornar a arquitetura mais flexível e fornecer os serviços de dados em IoT (Figura 2.2).

De acordo com Atzori et al. (2010), as funcionalidades destas camadas são descritas a seguir:

- **Composição do Serviço** - Esta camada proporciona funcionalidades para a composição de serviços únicos oferecidos por objetos em rede, visando a construção de aplicativos específicos. Nessa perspectiva, não há noção de dispositivos, sendo os únicos ativos visíveis os serviços. Uma perspectiva crucial no cenário de serviço é a existência de um repositório de todas as instâncias de serviço conectadas atualmente, as quais são executadas em tempo de execução para construir serviços compostos.
- **Gerenciamento de Serviços** - Esta camada oferece as funções principais esperadas para cada objeto e permite o seu gerenciamento no contexto da IoT. Um conjunto básico de serviços inclui: descoberta dinâmica de objetos, monitoramento de status e configuração do serviço. Nesse contexto, é possível obter serviços e funcionalidades relacionadas ao gerenciamento de Qualidade de Serviço (QoS) e gerenciamento de

Figura 2.2: Arquitetura Orientada a Serviços - IoT.



Fonte: Adaptado de Atzori et al. (2010).

bloqueio, bem como algumas funções semânticas e a capacidade de implantar novos serviços remotamente durante o tempo de execução, a fim de satisfazer as necessidades da aplicação.

- **Abstração do Objeto** - Essa camada permite que um amplo conjunto de objetos heterogêneos, cada um fornecendo funções específicas acessíveis através de seu próprio dialeto, sejam interoperáveis por meio da adoção de um mecanismo de abstração capaz de harmonizar o acesso aos diferentes dispositivos com uma linguagem e procedimento comum.

A abordagem SOA promove a redução de custos e esforços necessários para o reconhecimento e aplicabilidade de sistemas de IoT em novos domínios, uma vez que a adoção dos princípios SOA permite decompor sistemas complexos e monolíticos em aplicações que consistem em um ecossistema de componen-

tes mais simples e bem definidos e o suporte a comunicação aberta garante a interoperabilidade entre sistemas distintos (Atzori et al., 2010). Outro benefício promovido pelo uso desta arquitetura está relacionada a sua capacidade de reutilização dos componentes de software e hardware, o que melhora a viabilidade do seu uso (Kaur, 2018).

Arquitetura de Microsserviços

A arquitetura de microsserviços é um estilo arquitetural dominante na indústria de *software* orientado a serviço, que enfatiza a divisão do sistema em serviços pequenos e leves, cada um rodando em seu próprio processo, e são propositadamente construídos para desempenhar uma função de negócio muito coesa (Alshuqayran et al., 2016).

Basicamente, esta arquitetura provê uma disruptura de uma estrutura única de processos dependentes e altamente acoplados, como por exemplo, arquiteturas monolíticas onde processos são executados como um único serviço (Singh e Peddoju, 2017), em uma série de componentes independentes fracamente acoplados, que se comunicam por meio de uma interface bem definida (Chen et al., 2017b). Cada microsserviço pode ser implantado, atualizado, expandido e reiniciado independentemente de outros serviços da aplicação conforme demandas específicas (de Santana et al., 2021), diferente de aplicações monolíticas onde o funcionamento atípico de um processo compromete todo o conjunto.

Sistemas construídos com arquitetura de microsserviços são normalmente associados com a seguintes benefícios (Chen et al., 2017b; Krylovskiy et al., 2015; de Santana et al., 2021):

- **Resiliência e facilidade de implantação** - São habilitadas por decomposição via serviços que fornecem componentes com limites claros, permitindo isolar falhas gradualmente sem degradar as funcionalidades do sistema, bem como atualizar e implantar serviços individuais de forma independente.
- **Escalabilidade** - A construção de aplicações sob a perspectiva da arquitetura de microsserviços são decompostas em diversos componentes, sendo possível o aprimoramento e expansão de componentes específicos sem que haja a necessidade da modificação do sistema por completo.
- **Heterogeneidade tecnológica** - Pelo fato da arquitetura de microsserviços possuir características de fraco acoplamento, é possível permitir que cada parte da aplicação possa ser implementada com diferentes tipos de tecnologias, adequadas para cada tipo de trabalho.

Diante dos benefícios apresentados, a arquitetura de microsserviços provê a construção, gerenciamento e projeto de construção de pequenas unidades autônomas a fim de apoiar o desenvolvimento de sistemas de larga escala que sejam mais robustos, resilientes, escaláveis e melhor adaptados para atender as demandas atuais de aplicações IoT, possibilitando implantações continuadas por dezenas ou centenas de vezes ao dia, tornando-as mais dinâmicas conforme a utilização e contexto (Zeiner et al., 2016; Santana et al., 2018).

Propor uma linguagem arquitetural para microsserviços pode ajudar os arquitetos de IoT em muitas atividades como, por exemplo, no raciocínio sobre o sistema como um todo. Realizando análises sobre as qualidades do sistema, lidando com os aspectos dinâmicos e mutáveis de aplicações em tempo de execução (Francesco et al., 2017),

As vantagens promovidas por uma arquitetura de microsserviços a credencia como uma promissora solução diante do contexto expansivo de popularização dos centros urbanos e suas problemáticas associadas, onde a heterogeneidade e a dinamicidade destes ambientes requerem a promoção de estratégias tecnológicas eficientes para o gerenciamento de emergências. Entretanto, conforme Francesco et al. (2017), há poucos trabalhos presentes na literatura que investigam a adoção de padrões de *design* e linguagens de arquitetura para microsserviços, revelando lacunas a serem preenchidas pela comunidade de pesquisa.

Outras questões relacionadas à adoção da arquitetura de microsserviços em projetos de IoT que podem trazer grandes desafios para a sua implementação está relacionado a: complexidade, uma vez que implementar este tipo de arquitetura envolve dividir o sistema em vários serviços menores, tornando mais difícil testes e manutenções; sobrecarga, haja vista que os serviços precisam se comunicar uns com os outros, é possível que haja uma sobrecarga de tráfego, especialmente em sistemas com muitos dispositivos e aplicativos conectados; integração de vários serviços, que pode ser desafiador e pode introduzir erros e falhas no sistema.

Arquitetura Publicar/Assinar

De acordo com Hohpe e Woolf (2004), basicamente existem dois principais padrões de arquitetura para propagação de mensagens entre computadores conectados em rede, são eles: requisição/resposta e publicar/assinar.

O padrão requisição/resposta conta com dois componentes que interagem entre si, um que requisita uma informação e um que responde à esta requisição. Na extensa maioria dos casos este padrão é implementado de forma síncrona, ou seja, ao realizar uma requisição, o requisitante permanece esperando até

obter uma resposta do destinatário ou até o período máximo de espera ser atingido (Hohpe e Woolf, 2004).

A arquitetura publicar/assinar é um mecanismo ou paradigma de comunicação para sistemas distribuídos no qual, cada nó presente nesta rede se comunica entre si publicando dados e recebendo (assinando) dados anonimamente (Made Wirawan et al., 2018). Neste padrão de arquitetura, ao contrário do padrão requisição/resposta, a troca de mensagens entre o remetente e o destinatário não ocorre de forma direta, sendo composto de três componentes descritos como assinantes, publicadores e corretores (do inglês, *Broker*) (Made Wirawan et al., 2018).

O publicador envia uma mensagem sobre um tópico específico, independentemente do destinatário, e um assinante pode assinar e receber o mesmo tópico de forma assíncrona. O sistema é geralmente mediado por vários corretores que recebem mensagens publicadas de editores e as enviam para assinantes.

A arquitetura publicar/assinar oferece várias vantagens sobre outros padrões de comunicação, sendo elas (Li e Jacobsen, 2005; Oh et al., 2010; Kraijak e Tuwanut, 2015):

- **Escalabilidade** - A capacidade que os dispositivos e aplicativos tem em se registrarem para receber atualizações de tópicos específicos viabiliza a inclusão ou remoção de tais elementos sem a necessidade de reconfigurar diretamente a comunicação entre eles. Este processo otimiza a escalabilidade e a flexibilidade do sistema.
- **Desacoplamento** - A comunicação entre dispositivos e aplicativos por meio de tópicos, em contraposição a conexões diretas, permite o desacoplamento entre eles. Essa abordagem possibilita modificações ou substituições individuais de dispositivos e aplicativos sem afetar a integridade da comunicação entre os elementos.
- **Robustez** - A inscrição dos dispositivos e aplicativos para receberem atualizações de tópicos específicos assegura a continuidade das atualizações, mesmo na eventualidade de alguns dispositivos ou aplicativos temporariamente indisponíveis. Esse processo reforça a robustez do sistema.
- **Comunicação assíncrona** - Os dispositivos e aplicativos têm a capacidade de publicar atualizações sem a necessidade de aguardar por uma resposta imediata, viabilizando, assim, uma forma de comunicação assíncrona, o que significa que os dispositivos e aplicativos podem enviar e receber atualizações independentemente uns dos outros. Esta abordagem tem o potencial de otimizar a eficiência e reduzir a latência inerente à comunicação.

- **Comunicação *one-to-many*** - Como os dispositivos e aplicativos se inscrevem para receber atualizações de tópicos específicos, é possível ter uma comunicação *one-to-many*, o que significa que um dispositivo ou aplicativo pode publicar atualizações para vários outros dispositivos ou aplicativos ao mesmo tempo.
- **Comunicação *real-time*** - Considerando que dispositivos e aplicativos têm a capacidade de enviar e receber atualizações instantâneas, é possível permitir a comunicação em tempo real. Tal aspecto ganha relevância em contextos de aplicativos críticos e de natureza temporal, como no âmbito da IoT, automação industrial e monitoramento de dispositivos.

Entretanto, a arquitetura publicar/assinar também possui algumas desvantagens em comparação com outros padrões de comunicação, como exemplo, a latência e sobrecarga de tráfego, uma vez que as atualizações são enviadas para todos os dispositivos inscritos. A falta de garantia de entrega também é um fator que merece atenção, pois não há garantia de que as atualizações serão recebidas pelos dispositivos e aplicativos inscritos. Contudo, estes pontos podem ser contornados com a definição de um formato leve de mensagens a serem trocadas entre dos dispositivos comunicantes e adoção de algumas técnicas, como exemplo, implementação de mecanismos para registro dos eventos e definição de métodos de confirmação de recebimento de mensagens.

Para os propósitos específicos e, em observância a inexistência de uma arquitetura de referência, a arquitetura proposta neste trabalho levará em consideração a adoção de mecanismos presentes na arquitetura publicar/assinar, dado os benefícios supracitados.

2.3.2 Elementos de IoT

Sistemas de IoT são compostos por uma série de blocos funcionais com o objetivo de facilitar atividades do sistema, como por exemplo, sensoriamento, identificação, atuação, comunicação e gerenciamento (Ray, 2018). A compreensão destes blocos de construção da IoT ajuda a obter uma melhor percepção do seu real significado e funcionalidades esperadas.

Figura 2.3: Elementos da IoT.

Fonte: Adaptado de Al-Fuqaha et al. (2015).

- **Identificação** - Consiste na definição de padrões e métodos empregados para fornecer uma identidade clara a cada objeto dentro da rede. Conforme Al-Fuqaha et al. (2015), a identificação é crucial para a IoT nomear e combinar serviços com sua demanda. Objetos de IoT possuem um identificador associado ao seu nome, como por exemplo, "T1" para um sensor de temperatura específico, e um endereço do objeto, que refere-se ao seu endereço dentro de uma rede de comunicações.
- **Detecção** - Tem por objetivo coletar dados de objetos relacionados dentro da rede e enviá-los para servidores de processamento centralizados, banco de dados ou nuvem (Al-Fuqaha et al., 2015). Os dados coletados são analisados para tomar ações específicas com base nos serviços necessários.
- **Dispositivos** - Um sistema IoT é baseado em dispositivos como sensores inteligentes, atuadores ou dispositivos de detecção "vestivéis" que fornecem atividades de detecção, atuação, processamento, controle e monitoramento. De acordo com Ray (2018), dispositivos IoT podem trocam dados com outros dispositivos e aplicativos conectados ou coletar dados de outros dispositivos e processar os dados localmente ou enviar os dados para servidores centralizados ou aplicativos baseados em nuvem com base em restrições temporais e de espaço (ou seja, memória, recursos de processamento, latências de comunicação, e velocidades e prazos). Computadores de placa única (do inglês, *Single Board Computers* - SBCs) integrados com sensores, protocolos de comunicação embutidos e funcionalidades de segurança, por exemplo, *Arduino Yun*, *Raspberry PI*, *BeagleBone Black*, dentre outros, também são usados na construção de sistemas de IoT (Al-Fuqaha et al., 2015). Além destes dispositivos, muitas plataformas de *software* são utilizadas para fornecer funcionalidades esperadas para sistemas de IoT, dentre essas plataformas, os Sistemas Operacionais são vitais para o controle e gerenciamento do *hardware* de IoT. Existem vários sistemas operacionais para dispositivos de IoT, dos quais é possível destacar: *TinyOS*, *LiteOS* e *Riot OS*. Para o suporte à aplicações de IoT onde o tempo de resposta aos eventos se torna crítico, sistemas operacionais em tempo real (RTOS), como exemplo, o *Contiki* RTOS, tem sido amplamente utilizados (Al-Fuqaha et al., 2015).
- **Comunicação** - As tecnologias de comunicação IoT conectam objetos heterogêneos e servidores remotos para fornecer serviços inteligentes específicos (Al-Fuqaha et al., 2015), (Ray, 2018). Normalmente, nós de IoT devem operar usando baixa potência, promovendo uma melhor eficiência energética, haja visto as limitações de *hardware* destes dispositivos. Exemplos de protocolos de comunicação de baixo custo computacional usados para a IoT são *Wi-Fi*, *Bluetooth*, IEEE 802.15.4, *Z-wave*, *ZigBee* e

LTE-*Advanced* (Ferro e Potorti, 2005), (Rawat et al., 2013) e (Al-Fuqaha et al., 2015).

- **Serviços** - Um sistema de IoT promove vários tipos de serviços. Conforme Gigli et al. (2011), os serviços de IoT podem ser categorizados em quatro classes: serviços relacionados à identidade, serviços de agregação de informações, serviços colaborativos e serviços ubíquos. Os serviços relacionados à identidade são os serviços mais básicos e importantes que tem por objetivo trazer objetos do mundo real para o mundo virtual, identificando-os. Os serviços de agregação de informações coletam e resumem medições sensoriais brutas que precisam ser processadas e relatadas. serviços colaborativos atuam sobre os serviços de agregação de informações e usam os dados obtidos para tomar decisões. Os serviços ubíquos resumem a IoT, tornando estes sistemas onipresentes para todos, tudo, em todos os momentos. Para que a IoT alcance o nível de fornecer serviços onipresentes, se faz necessário superar a barreira das distinções de protocolo entre as tecnologias e unificar todas as aspecto da rede.
- **Semântica** - Refere-se à capacidade de extrair conhecimento de forma inteligente por diferentes máquinas para fornecer os serviços necessários, incluindo a descoberta e uso de de recursos e informações de modelagem (Al-Fuqaha et al., 2015).

2.3.3 Protocolos de IoT

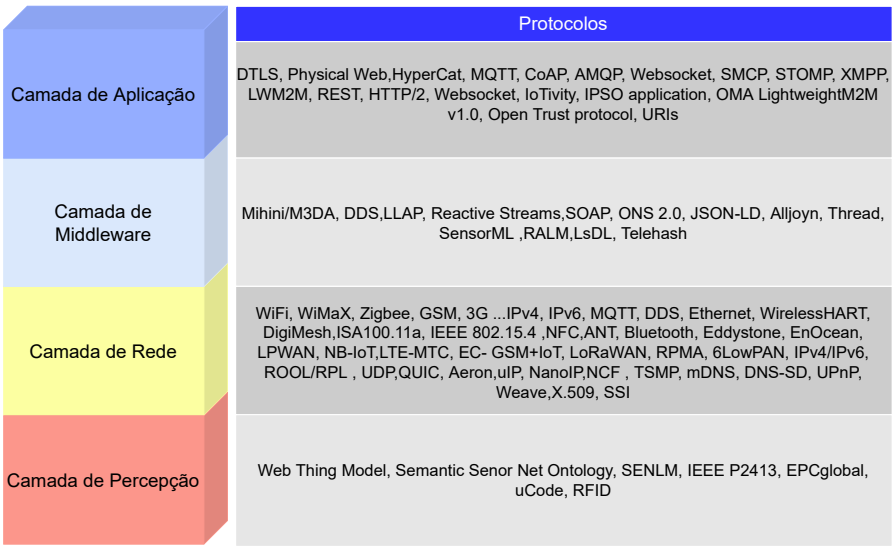
Os protocolos de comunicação em geral representam um conjunto de regras, sintaxe e semântica, podendo ser implementados por hardware, software ou uma combinação de ambos que permitem que duas ou mais entidades em um sistema de comunicação transmitam informações (Hofer-Schmitz e Stojanović, 2020). As regras e convenções estabelecidas pelos protocolos garantem que dispositivos se comuniquem de forma eficiente e precisa, independentemente de plataformas e ou fabricantes, uma vez que especificam como os dados deverão ser transmitidos, formatados e interpretados.

A IoT é caracterizada como uma tecnologia capaz interagir com vários dispositivos, coisas e objetos. Essas interações abrem diferentes direções de aprimoramento e desenvolvimento em muitos campos, como arquitetura, dependências, comunicações, protocolos, segurança, aplicativos e *big data* (Yassein et al., 2016). Dessa forma, de modo a garantir a interconectividade e interoperabilidade, bem como alcançar as mudanças e melhorias desejadas através do uso da tecnologia de IoT, são necessários diversos protocolos de comunicação.

A Figura 2.4 apresenta os principais protocolos de IoT separados por camadas

de abstrações, onde cada protocolo implementa uma funcionalidade específica a uma determinada camada.

Figura 2.4: Protocolos de IoT



Fonte: Adaptado de Elhadi et al. (2018).

Conforme observado, há uma diversidade enorme de protocolos que são utilizados no paradigma de IoT. A medida que os estudos e pesquisas em IoT avançam, surgem novos protocolos com características adaptadas às necessidades dos objetos conectados, como exemplo, baixo consumo de energia, grande alcance, baixo *throughput*, facilidade de implementação, dentre outros (Elhadi et al., 2018).

Entretanto, nos subtópicos que seguem, serão abordados alguns dos principais protocolos de aplicação para IoT baseados na Arquitetura Publicar/Assinar, tendo em vista a escolha deste padrão para a definição do modelo conceitual aqui proposto.

A camada de aplicação é responsável por prover serviços e determinar um conjunto de protocolos para passagem de mensagens no nível de aplicação (Yassein et al., 2016). Os protocolos da camada de aplicação interagem diretamente com o usuário final, usados para atualizar servidores online com os valores de dispositivos finais além de transportarem comandos de aplicativos para os atuadores destes dispositivos finais.

Message Queuing Telemetry Transport (MQTT)

Projetado pela IBM e, em 2013, padronizado pela OASIS, o *Message Queuing Telemetry Transport* (do inglês, Transporte de Telemetria do Enfileiramento

de Mensagens - MQTT) (Salman e Jain, 2017) é um protocolo de comunicação entre máquinas (*Machine to Machine* - M2M), baseado no conceito de publicação/assinatura, semelhante ao modelo cliente-servidor que como objetivo de reduzir a necessidade de largura de banda, além de garantir a confiabilidade da entrega de pacotes (Banks et al., 2020).

Sua simplicidade e código-fonte aberto tornam esse protocolo ideal para uso em muitas situações incluindo ambientes restritos, comunicação em contextos M2M e IoT (Banks et al., 2020).

Estas características tornam o MQTT um dos melhores protocolos candidatos para projetos de IoT (Costa et al., 2020b; Elemam et al., 2020), pois é um protocolo leve, projetado para encaminhar apenas dados solicitados, que se adéqua à dispositivos com capacidades de computação, memória e largura de banda limitadas, como exemplo, sensores de baixa potência, dispositivos móveis, como *smartphones*, sistemas embarcados como o *Raspberry PI* ou microcontroladores como o Arduíno (Yassein et al., 2016).

Além de garantir a confiabilidade da entrega de pacotes (Yassein et al., 2016), o MQTT fornece um conjunto de recursos que inclui (Banks et al., 2020):

- O protocolo é executado em TCP/IP ou em outros protocolos de rede que fornecem conexões ordenadas, sem perdas e bidirecionais;
- Suporte para redes não confiáveis através do uso de sessões persistentes;
- Segurança habilitada através da criptografia de mensagens usando TLS e autenticação de clientes usando protocolos de autenticação modernos;
- Uso do padrão de mensagens de publicação/assinatura, que fornece distribuição de mensagens um-para-muitos e desacoplamento de aplicativos;
- Transporte de mensagens independente do conteúdo da carga útil;
- Capacidade de estabelecer comunicações entre dispositivos remotos;
- Três níveis de Qualidade de Serviço (QoS) para a entrega de mensagens;
- Os dados podem conter arquivos binários ou de texto.

A importância do protocolo MQTT se deve à sua simplicidade além do suporte a uma ampla gama de diferentes dispositivos e plataformas móveis (Yassein et al., 2016).

Advanced Message Queuing Protocol (AMQP)

De acordo com Standard (2012), o *Advanced Message Queuing Protocol* (AMQP) é um padrão aberto para encaminhamento de mensagens de forma

confiável, amplamente utilizado em plataformas de negócios, comerciais, dispositivos e aplicativos de software. O AMQP usa diferentes garantias de entrega de mensagens para garantir a confiabilidade além de utilizar a camada de transporte TCP para garantir a confiabilidade.

A abordagem de publicação/assinatura do AMQP consiste em dois componentes: fila de troca e fila de mensagens, para a transferência de dados de forma confiável. Quando mensagens são publicadas ao *broker*, não são diretamente roteadas para o consumidor, como no caso do MQTT, em vez disso, eles serão transferidos para uma fila de troca, que será responsável por rotear mensagens para filas de mensagens com base nas chaves de roteamento (Uy e Nam, 2019; Yassein et al., 2016). Vale ressaltar que cada fila de mensagem é usada para um consumidor.

As mensagens ficarão armazenadas na fila até serem retiradas pelo consumidor. Esta é a principal diferença entre MQTT e AMQP, possibilitando armazenar a mensagem quando esta não foi recuperada no lado receptor (Uy e Nam, 2019).

O AMQP suporta comunicações características de heterogeneidade e interoperabilidade entre diferentes dispositivos, promovendo confiabilidade, segurança e desempenho. Entretanto, este protocolo apresenta um tamanho de pacote maior que os demais protocolos apresentados, não sendo adequado para ambientes restritos e aplicativos em tempo real (Elhadi et al., 2018; Yassein et al., 2016) e menor eficiência energética em relação ao MQTT (Sidna et al., 2020).

Extensible Messaging And Presence Protocol (XMPP)

O *Extensible Messaging and Presence Protocol* (XMPP) é um perfil de aplicativo da *Extensible Markup Language* (XML) que permite a troca quase em tempo real de dados estruturados, porém extensíveis, entre quaisquer duas ou mais entidades de rede, sendo padronizado pelo IETF sob as *Request for Comments* (RFC) - 6120, 6121, 7395, 7590, 7622 (Saint-Andre, 2011).

Este protocolo é bastante conhecido, suporta pequenas mensagens e baixa latência, o que o torna uma boa escolha para comunicações IoT. Os principais métodos de protocolo do XMPP: configuração e desmontagem de fluxos XML, criptografia de canal, autenticação, tratamento de erros e comunicação primitivas para mensagens, disponibilidade de rede, oferecendo suporte a modelos de solicitação/resposta e publicação/assinatura (Saint-Andre, 2011; Yassein et al., 2016).

O XMPP tem como características: a alta escalabilidade, que é obtida pelo uso de uma arquitetura descentralizada; simplicidade, podendo ser utilizada

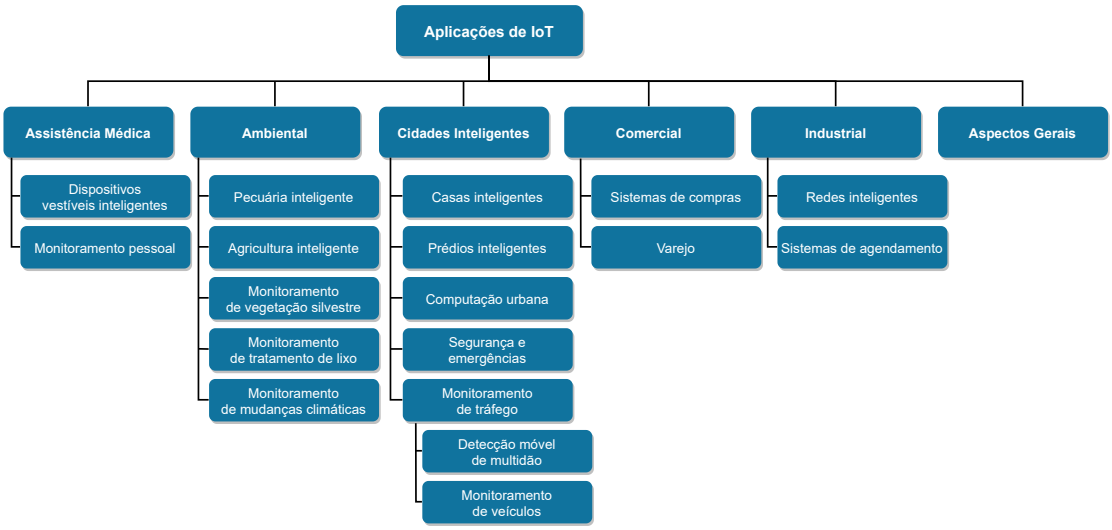
em projetos e aplicações heterogêneas; extensível e flexível, possuindo muitas extensões definidas. Por outro lado, tem alguns pontos fracos; uma vez que este protocolo necessita de alto consumo de largura de banda e alto uso de processamento, não há garantia de QoS e é restrito a tipos de dados simples (Elhadi et al., 2018; Yassein et al., 2016), além de apresentar um consumo de energia elevado em relação ao MQTT e ao AMQP (Sidna et al., 2020).

2.3.4 Aplicações de IoT

As potencialidades oferecidas pela IoT possibilitam o desenvolvimento de um grande número de aplicações e, é notório que, o uso correto da tecnologia nos mais diversos domínios sociais e ambientais possibilita melhorias na qualidade de vida dos cidadãos (Atzori et al., 2010).

Com a proliferação e aceitação do paradigma IoT, novos domínios de aplicativos continuam a surgir à medida que surgem novos requisitos do usuário. Embora o foco de cada domínio de aplicativo seja diferente, todos os aplicativos compartilham um objetivo comum que é provisionar serviços inteligentes para aumentar a qualidade de vida humana (Bello e Zeadally, 2019). De acordo com Asghari et al. (2019), os domínios de aplicações de IoT podem ser descritos através da Figura 2.5.

Figura 2.5: Domínios de aplicações de IoT.



Fonte: Adaptado de Asghari et al. (2019).

A diversidade de domínios de aplicativos no ambiente IoT resultou no desenvolvimento e implantação de diferentes padrões de tecnologia dentro do ecossistema IoT. No entanto, independente do domínio de aplicação, os processos

operacionais de aplicações de IoT tem preocupações em comum: satisfazer métricas de qualidade e requisitos de usuário (Asghari et al., 2019).

Dessa forma, os requisitos do usuário devem ser suportados por serviços inteligentes em aplicações IoT que cubram as métricas de QoS, como segurança, custo, tempo de serviço, consumo de energia, confiabilidade e disponibilidade.

A Tabela 2.1 sumariza alguns trabalhos práticos que abordam aplicações promissoras nos mais diversos domínios da IoT, desenvolvidos no período de 2018 à 2022. Estes trabalhos propõem soluções para diferentes áreas, como saúde, gerenciamento de resíduos, agricultura, tráfego e sustentabilidade ambiental, destacando a importância do uso de tecnologias avançadas, como *blockchain* e aprendizado de máquina para otimizar a funcionalidade e segurança dos sistemas IoT.

É possível perceber o potencial da tecnologia IoT para solucionar problemas concretos da sociedade, por meio da aplicação de recursos avançados que aprimoram tanto a funcionalidade quanto a segurança desses sistemas. Além disso, sua implementação pode gerar melhorias significativas em termos de eficiência, economia e qualidade de vida nas cidades.

Entretanto, de modo geral, observa-se que os trabalhos supracitados negligenciam aspectos éticos e sociais relacionados ao uso da tecnologia IoT, como a privacidade e a propriedade dos dados. Essa omissão representa um desafio significativo para a adoção e disseminação desse paradigma tecnológico, especialmente considerando que envolve a coleta de dados associados à monitoração de indivíduos.

Outro aspecto que merece destaque é a ausência de uma abordagem mais aprofundada sobre as dificuldades práticas relacionadas à implementação de sistemas IoT em larga escala, tais como os altos custos e a complexidade envolvida na implantação e manutenção de redes de sensores.

2.3.5 Segurança em IoT

Nos últimos anos, uma grande variedade de dispositivos tornaram-se interconectadas através do uso da internet. O crescente interesse de na IoT levou ao surgimento de vários sistemas e aplicações nas mais diversas esferas da sociedade, como agricultura (Goap et al., 2018), hospitais (Harun-Ar-Rashid et al., 2023; AlGhamdi et al., 2023), vigilância (Jain et al., 2008; Krull et al., 2018), segurança (Kodali et al., 2016; Majumder e Izaguirre, 2020), transportes (Sutar et al., 2016; Geetha e Cicilia, 2017) dentre outros.

Dispositivos IoT representam 50% de todos os dispositivos em rede no mundo (Cisco, 2020). Os avanços desta tecnologia são os principais impulsionadores

Tabela 2.1: Desenvolvimento de aplicações de IoT em diferentes domínios nos últimos 5 anos.

Trabalho	**Domínio**	**Descrição**
Harun-Ar-Rashid et al. (2023)	Assistência Médica	O estudo propõe a utilização de um painel IoT para a exibição de dados e imagens médicas de pacientes. Estas informações são obtidas através de dispositivos médicos, como máquinas de ultrassom, ressonância magnética ou tomografia computadorizada e encaminhadas para um *Raspberry Pi* para que este possa monitorar e extrair dados de imagem.
AlGhamdi et al. (2023)	Assistência Médica	Este estudo propõe o uso da tecnologia de blockchain combinada ao paradigma de IoT para melhorar a eficiência, a segurança e a transparência do setor de saúde através do processo de clínica médica inteligente não supervisionada e sem intervenções da equipe médica. O objetivo é de fornecer serviços seguros e rápidos para enfrentar a pandemia sem expor a equipe médica ao perigo além de propor um algoritmo de aprendizado profundo para imagens de raios X baseadas na detecção de COVID-19.
Veloz-Cherrez et al. (2020)	Ambiental	Propuseram um protótipo de um sistema inteligente para monitoramento e controle de parâmetros de decomposição dos resíduos sólidos presentes em lixeiras instaladas na cidade de Riobamba, Equador, através do uso de IoT.
Goap et al. (2018)	Ambiental	Propuseram um sistema de gerenciamento de irrigação inteligente baseado em IoT. Através da detecção de parâmetros do solo obtidos por uma RSSF, como por exemplo, umidade, temperatura, condições ambientais e dados de previsão do tempo obtidos na Internet, podendo prever os requisitos para a irrigação necessária do ambiente avaliado.
Costa et al. (2019)	Cidades Inteligentes	Propuseram o desenvolvimento de plataforma pública de uso geral para aquisição e visualização de velocidades veiculares através do *crowdsensing* de dados de *smartphones* com o objetivo de fornecer dados que podem dar melhor suporte ao planejamento e gerenciamento de tráfego em áreas urbanas.
Chaudhari e Bhole (2018)	Ambiental	Propuseram um sistema de gerenciamento de resíduos sólidos baseado em IoT que permite o monitoramento de lixeiras, programação dinâmica e roteirização de caminhões coletores de lixo em uma cidade inteligente.

do seu crescimento econômico. Entretanto, todos esses benefícios costumam estar associados a muitos riscos de segurança, uma vez que a origem e natu-

reza das aplicações de IoT geralmente estão fora do controle de organizações ou indivíduos que implantam esses sistemas (Nebbione e Calzarossa, 2020). Para agravar o problema, a natureza das ameaças está se tornando mais diversificada, especialmente com o desenvolvimento e a ampla adoção de aplicações IoT que lidam com informações sensíveis, por exemplo, informações pessoais, industriais, governamentais dentre outras (El-hajj et al., 2019).

A diversidade de domínios de aplicações e dispositivos de IoT além da integração destes com uma variedade de tecnologias capacitadoras, por exemplo, módulos de *software*, bibliotecas, *middlewares*, interfaces de programação de aplicativos, protocolos e redes móveis, exige uma maior consideração dos possíveis desafios de segurança (Nebbione e Calzarossa, 2020). A inteligência integrada em residências, carros e redes elétricas através do uso de aplicações de IoT, por exemplo, pode ser desviada para cenários prejudiciais quando explorada por indivíduos maliciosos (El-hajj et al., 2019).

Como os dispositivos *IoT* são parte integrante da vida cotidiana, é importante protegê-los, identificando adequadamente os possíveis riscos de segurança e criando medidas de mitigação adequadas.

À medida que o número de aplicações de *IoT* crescem, é necessário modificar ou aprimorar os protocolos da camada de aplicação de acordo com as especificidades de cada sistema, permitindo a abordagem de questões como adaptação dinâmica às condições de rede e interoperabilidade (Donta et al., 2022).

Para garantir a proteção e segurança da informação em ambientes interconectados de *IoT*, é preciso atingir três premissas básicas de segurança que compõem a tríade de segurança da informação:

- **Disponibilidade** - O processo de garantir que o serviço necessário esteja disponível em qualquer lugar e a qualquer momento para os usuários pretendidos (Hintzbergen et al., 2018). A disponibilidade na IoT leva em consideração também a disponibilidade dos próprios objetos.
- **Confidencialidade** - O processo de garantir que as informações sejam acessadas apenas por pessoas autorizadas (El-hajj et al., 2019).
- **Integridade** - A integridade garante a consistência, precisão e confiabilidade dos dados durante toda a sua vida útil. As preocupações com a integridade dos dados surgem devido à natureza autônoma dos dispositivos IoT (Saqib e Moon, 2023). Após a implantação, a maioria desses dispositivos se tornará autossuficiente, o que facilita a alteração de informações básicas ou mesmo a infusão de informações inválidas. Adicionalmente, há o risco de coleta de dados de baixa qualidade e corrupção

ambiental, seja por perda natural de calibração ou manipulação intencional por um adversário. Em resumo, os dados coletados pela IoT podem ser pouco confiáveis e facilmente alterados (El-hajj et al., 2019; Saqib e Moon, 2023).

Uma vez que os objetivos de segurança da informação são atingidos, de modo a possibilitar uma infraestrutura de comunicação capaz de garantir níveis de segurança aceitáveis, para uma implantação de IoT confiável e segura, existem vários requisitos de segurança que um ambiente de IoT deve atender, sendo eles:

- **Não-repúdio** - A forma de garantir a capacidade de demonstrar que uma tarefa ou evento ocorreu com o objetivo de que isso não possa ser negado posteriormente (Hintzbergen et al., 2018). Em outras palavras, o objeto não pode negar a autenticidade de um determinado dado transferido ou, que este não negue a autoria de uma determinada ação.
- **Privacidade** - O processo de garantir a não acessibilidade a informações privadas por objetos públicos ou maliciosos (El-hajj et al., 2019). O crescente número de dispositivos na tecnologia IoT aumenta as preocupações com a privacidade. É fundamental proteger os dados para aumentar a facilidade de uso e o conforto do usuário da IoT, ao mesmo tempo em que resolve problemas de propriedade relacionados à privacidade dos dados (Saqib e Moon, 2023).

Além dos requisitos de não-repúdio e privacidade, os requisitos de autenticação, autorização e contabilidade (*Authentication, Authorization and Accounting* - AAA) são cruciais para a IoT garantir a segurança da comunicação entre dois nós conectados e evitar o acesso não autorizado à rede, onde um único nó comprometido pode ser explorado por um adversário (Al-Naji e Zagrouba, 2020).

- **Autenticação** - Garante que apenas usuários autenticados tenham acesso a informações confidenciais e recursos de sistemas. Isso inclui a verificação de identidade, autorização de acesso e garantia de que as ações de um usuário são autorizadas. No contexto IoT, cada objeto deve ter a capacidade de identificar e autenticar todos os outros objetos no sistema (ou em uma determinada parte do sistema com o qual ele interage) (Nebbione e Calzarossa, 2020; El-hajj et al., 2019)
- **Autorização** - O processo de dar permissão a uma entidade para fazer ou ter algo. Garantia de que as ações de um usuário ou um nó são autorizadas (Nebbione e Calzarossa, 2020; El-hajj et al., 2019).
- **Contabilidade** - É o processo de auditar e rastrear o uso de recursos durante a sessão de acesso, incluindo, por exemplo, a quantidade de

tempo e/ou dados que um nó enviou e/ou recebeu durante uma sessão (Al-Naji e Zagrouba, 2020).

A complexidade dos ambientes urbanos, aliada às características tecnológicas da IoT, impõe desafios relevantes que devem ser enfrentados para a implementação eficaz de aplicações voltadas às cidades inteligentes. Essas aplicações devem estar alinhadas aos princípios da segurança da informação e contemplar os principais requisitos de segurança em seus diversos componentes.

Os desafios de segurança na IoT abrangem todas as suas camadas arquiteturais, desde as arquiteturas mais básicas — como as convencionais ou genéricas — até os modelos mais avançados e atuais. No entanto, para fins deste trabalho, os requisitos de segurança considerados concentram-se na camada de aplicação da IoT, com o objetivo de assegurar níveis adequados de proteção no processo de incorporação dinâmica e flexível de novos requisitos voltados ao monitoramento de eventos de interesse.

———— A camada de aplicação da IoT é responsável pela interação direta com o usuário. No entanto, os protocolos "tradicionais"dessa camada não se adaptam adequadamente às especificidades da IoT, e a ausência de padrões internacionais consolidados para esse paradigma contribui para o surgimento de diversos problemas de segurança (El-hajj et al., 2019). De acordo com Nebbione e Calzarossa (2020), os principais riscos de segurança na camada de aplicação da IoT estão associados a aplicações voltadas à troca de mensagens e à descoberta de serviços — como é o caso dos protocolos MQTT, AMQP e XMPP, abordados neste trabalho. As aplicações de troca de mensagens referem-se ao compartilhamento de dados entre dispositivos, enquanto as de descoberta de serviços dizem respeito à identificação de dispositivos e dos serviços por eles oferecidos.

A autenticação é um requisito fundamental para a proteção da privacidade dos usuários e das mensagens trocadas entre os dispositivos, sendo reconhecida como um dos princípios essenciais de segurança. Por meio da identificação e classificação dos usuários e dispositivos que acessam a rede, é possível impor restrições de acesso, limitando-o a entidades legítimas e não comprometidas, o que contribui para a proteção dos dados mesmo em situações de interceptação ou interrupção (El-hajj et al., 2019; Saqib e Moon, 2023).

Em um cenário marcado pela diversidade, complexidade, natureza aberta e restrições de recursos — características intrínsecas à IoT (Saqib e Moon, 2023) — os protocolos da camada de aplicação estão particularmente expostos a uma ampla variedade de ataques maliciosos que exploram suas vulnerabilidades específicas (Nebbione e Calzarossa, 2020). Por essa razão, a autenticação nessa camada tem se consolidado como um tema fundamental de pesquisa no campo da segurança aplicada à IoT.

Conforme Saqib e Moon (2023), inúmeras soluções de segurança foram propostas e desenvolvidas para a IoT nos últimos anos. Entretanto, os autores afirmam que devido à natureza única da IoT, algumas dessas soluções são incompetentes, inviáveis e inaplicáveis neste paradigma. Protocolos da camada de aplicação como o MQTT, AMQP e XMPP fornecem alguns serviços de segurança integrados, como por exemplo, os serviços de autenticação e autorização, além de serviços de criptografia, através da combinação de tecnologias, técnicas e mecanismos computacionais para a proteção e confidencialidade dos dados (Nebbione e Calzarossa, 2020).

Em aplicações de IoT para o gerenciamento de eventos de emergências se faz necessário garantir que as informações sejam provenientes de fontes confiáveis, tendo em vista a natureza crítica destas aplicações com relação a manipulação de dados sensíveis que podem por em risco vidas humanas e ou ocasionar danos materiais. A escolha de métodos eficientes de autenticação é um ponto crucial para o correto funcionamento de soluções de IoT e a sua implementação está associada aos perigos existentes da divulgação de dados nos canais de comunicação sem fio (Saqib e Moon, 2023).

Segundo Saqib e Moon (2023), diversas soluções de segurança foram propostas para o contexto da IoT nos últimos anos. No entanto, os autores ressaltam que, devido às características únicas da IoT, muitas dessas soluções revelam-se ineficazes, inviáveis ou inadequadas. Protocolos da camada de aplicação como MQTT, AMQP e XMPP oferecem, ainda que de forma limitada, serviços integrados de segurança, como autenticação, autorização e criptografia. Esses mecanismos visam garantir a confidencialidade e a integridade dos dados por meio da combinação de técnicas, tecnologias e recursos computacionais (Nebbione e Calzarossa, 2020).

No contexto de aplicações de IoT voltadas ao gerenciamento de eventos de emergência, é essencial assegurar que as informações transmitidas sejam provenientes de fontes confiáveis. Dada a criticidade desses cenários, que envolvem o tratamento de dados sensíveis com potencial de risco à vida humana e a danos materiais, a adoção de métodos eficientes de autenticação torna-se um elemento crucial. Sua implementação visa mitigar os riscos associados à exposição de dados em canais de comunicação sem fio (Saqib e Moon, 2023).

Serviços de segurança geralmente são opcionais e devem ser ativados e configurados explicitamente pelos desenvolvedores das aplicações. A falta de serviços de segurança adequados e ou configurações incorretas tornam os dispositivos IoT vulneráveis a diversas ameaças como acessos não autorizados ou divulgação de dados confidenciais (Nebbione e Calzarossa, 2020).

Diversos esquemas de autenticação podem ser utilizadas em aplicações de IoT para garantir que os dispositivos pertencentes à uma infraestrutura de

comunicação sejam autenticados antes de conceder recursos ou encaminhar informações. Em El-hajj et al. (2019), são discutidos alguns dos principais esquemas de autenticação propostos e desenvolvidos para ambientes de IoT que estão presentes na literatura. Estes esquemas levam em consideração a heterogeneidade dos dispositivos e a variação de tamanho, forma, armazenamento, poder computacional e capacidade da bateria e são apresentados a seguir:

- **Autenticação baseada em identidade** - Abordagem de autenticação de implementação simples na qual exige que uma parte apresente informações a outra parte para efetuar a autenticação. Utiliza credenciais como senhas e valores-chave que são armazenados na memória do dispositivo para validar a identidade.
- **Autenticação baseada em criptografia** - Envolve o uso de algoritmos criptográficos durante a fase de autenticação. Esses algoritmos são divididos em três tipos: simétricos, assimétricos e funções de *hash*. Alguns mecanismos de autenticação dependem exclusivamente de algoritmos simétricos, enquanto outros dependem exclusivamente de algoritmos assimétricos por conta de questões de sobrecarga e custo computacional. As funções de *hash* são usadas em protocolos criptográficos mais complexos como primitivas de integridade de dados.
- **Autenticação baseada em fichas (*tokens* e credenciais)** - A autenticação baseada em *token* usa um mecanismo de identificação gerado pelo servidor para autenticar um usuário ou dispositivo, enquanto a autenticação não baseada em *token* requer credenciais de usuário, como nome de usuário e senha, para autenticação sempre que os dados são trocados.
- **Autenticação baseada em *hardware*** - Usa as características físicas do *hardware* para autenticar. Existem duas soluções: implícitas, que usam *hardware* existente durante a autenticação, por exemplo, através de uma função física não clonável; e explícitas, que exigem um componente dedicado adicional para as operações de autenticação.
- **Autenticação baseada em contexto** - É o processo de usar dados físicos ou comportamentais para melhorar a verificação. Os dados podem ser biométricos, como impressões digitais, ou baseados em comportamento, como a dinâmica do pressionamento de tecla ou a autenticação de voz usando a impressão de voz do indivíduo.
- **Autenticação baseada em procedimento** - Inclui autenticação unidirecional: quando duas partes desejam se comunicar, apenas uma autentica a outra, deixando a outra sem autenticação; bidirecional (autenticação mútua): Corresponde a uma técnica de autenticação bidirecional,

onde ambas as partes da comunicação se autenticam dentro da rede. Ajuda no não repúdio e na responsabilidade autorizada (Hammi et al., 2018); e autenticação tridirecional (autenticação de três vias): neste caso, uma autoridade central autentica ambas as partes e as ajuda a se autenticar mutuamente.

O protocolo MQTT é capaz de suportar a autenticação mútua baseado em certificados de chave pública (certificados digitais ou de identidade) através da implementação protocolo TLS - *Transport Layer Security*, projetado para fornecer maior privacidade, integridade e segurança para a comunicação entre dispositivos.

2.4 Padrões de Tecnologias de Comunicação para Dispositivos de IoT

O sensoriamento remoto habilita a obtenção de informações sobre uma área monitorada, sem a necessidade de contato direto, agilizando processos de tratamento, armazenamento e monitoramento dos dados coletados. A IoT viabiliza a implantação dessas aplicações, permitindo a transmissão de dados entre dispositivos antes incapazes de se comunicarem. O uso de tecnologias de comunicação sem fio para aplicações de IoT desempenham um papel fundamental na coleta, transmissão e troca de dados entre dispositivos IoT e sistemas de gerenciamento. Existem várias tecnologias de comunicação disponíveis, cada uma com suas próprias características e casos de uso específicos. De acordo com (Ramathulasi e Rajasekhara Babu, 2020), é possível definir padrões de curto e longo alcance para tecnologias de comunicação sem fio empregadas em uma infraestrutura de IoT.

Os padrões de curto alcance promovem a comunicação de dispositivos IoT entre diferentes tipos de redes, onde o alcance é limitado, são muito fáceis e convenientes, com baixo consumo de energia. A Tabela 2.2 sumariza alguns dos protocolos de comunicação de rede sem fio de curto alcance empregados para a comunicação de dispositivos em aplcações de IoT.

Já os padrões de longo alcance são caracterizados como uma rede de baixa potência de longo alcance (*Low-Control Wide-Region Network* - LPWAN) é uma tecnologia de transmissão sem fio que está sendo vista como a solução definitiva, operando tanto nas faixas licenciadas quanto não licenciadas. A Tabela 2.3 sumariza alguns dos protocolos de comunicação de rede sem fio de longo alcance empregados para a comunicação de dispositivos em aplicações de IoT.

A escolha da tecnologia de comunicação para dispositivos IoT depende das

Tabela 2.2: Padrões de comunicação de curto alcance para aplicações de IoT.

Padrões de Curto Alcance	**Descrição**
6LoWPAN	Protocolo de *internetworking* que permite que pacotes IPv6 sejam transportados com requisitos de baixa potência em redes sem fio.. O 6LoWPAN representa o padrão pioneiro e amplamente adotado para a comunicação na IoT, com suporte para uma vasta gama de endereços IP. Foi concebido pelo grupo IETF e viabiliza a comunicação IP em redes sem fio, eliminando a necessidade de gateways ou proxies. Ele se integra facilmente com outras redes baseadas em IP e oferece suporte para diversos tipos de topologias (Mulligan, 2007; Hossen et al., 2010).
ZigBee	Um padrão de rede sem fio com baixo consumo de energia, baixa taxa de dados, longa vida útil da bateriade alimentação dos componentes, segurança de dispositivo e padrão de rede sem fio IEEE 802.15.4. Foi desenvolvido pela ZigBee Alliance e é adequado para redes de área pessoal e aplicações de alto nível de baixo custo. Ele suporta diferentes tipos de topologias (Ergen, 2004).
Bluetooth de Baixo Consumo de Energia (BLE)	Também conhecido como Bluetooth inteligente, representa um protocolo significativo para aplicações na IoT. Foi desenvolvido e aprimorado para comunicações de curta distância, com baixa largura de banda e latência reduzida, atendendo às necessidades da IoT. Suas vantagens em relação ao Bluetooth clássico incluem menor consumo de energia, configuração mais rápida e suporte para topologia de rede em estrela com um número ilimitado de nós (Heydon e Hunn, 2012).
Wi-Fi	Tecnologia de comunicação sem fio que utiliza o padrão 802.11 para transmitir dados em curtas distâncias (geralmente em torno de 100 metros em ambientes internos). É amplamente utilizado em dispositivos IoT quando uma conexão de alta velocidade e largura de banda é necessária, como câmeras de segurança doméstica e dispositivos de automação residencial.

necessidades específicas de cada aplicação, incluindo alcance, consumo de energia, largura de banda e custo. Muitas vezes, uma combinação de várias tecnologias pode ser usada em uma rede IoT para atender a diferentes requisitos de dispositivos e cenários de uso.

Tabela 2.3: Padrões de comunicação de longo alcance para aplicações de IoT.

Padrões de Longo Alcance	**Descrição**
Long Range Wide Area Network (LoRaWAN)	LoRaWAN é uma tecnologia de comunicação de longo alcance que permite a conectividade de dispositivos IoT em grandes áreas geográficas (Haxhibeqiri et al., 2018). É frequentemente usado em aplicações como agricultura inteligente e cidades inteligentes. Possui baixa largura de banda, adequado para transmissão de pequenas quantidades de dados (Khutsoane et al., 2017).
Narrowband IoT (NB-IoT)	O NB-IoT é uma tecnologia de comunicação de baixa potência projetada especificamente para dispositivos IoT. É uma opção econômica para aplicações que exigem transmissão de pequenas quantidades de dados, como medidores inteligentes e rastreamento de ativos, possuindo baixo consumo de energia e longo alcance (Chen et al., 2017a). Oferece suporte a diversos tipos de equipamentos móveis e à coexistência entre eles, eliminando a necessidade de investimentos em torres de comunicação adicionais e outros requisitos.
Padrões Celulares	As redes celulares, como 3G, 4G e 5G, oferecem conectividade ampla e de alta velocidade para dispositivos IoT. São adequadas para aplicações que exigem alta largura de banda, como veículos conectados e sistemas de monitoramento de ativos. Esses padrões garantem alta largura de banda, cobertura global e são adequados para mobilidade. Entretanto, possuem consumo de energia relativamente alto, custo de assinatura, não sendo ideais para dispositivos de baixo consumo.

2.5 Cidades Inteligentes

O termo cidade inteligente surge do paradigma da Internet das Coisas (IoT), representando uma cidade que opera de forma sustentável e inteligente, integrando suas infraestruturas e serviços de forma coesa, e utilizando dispositivos inteligentes para monitoramento e controle, visando a sustentabilidade e a eficiência (Giffinger et al., 2007). Este conceito ganhou maior atenção nos círculos acadêmicos, industriais e governamentais devido ao crescimento populacional urbano, que muitas vezes sobrecarrega a infraestrutura e os recursos disponíveis, resultando em serviços públicos deficientes (Santana et al., 2017). Nesse sentido, tornando-se de suma importância o incentivo para o desenvolvimento de soluções inovadoras orientadas por Tecnologias de Informação e Comunicação (TIC) para os contextos urbanos.

As áreas urbanas, em particular, assumem a maior parcela do consumo de recursos, impulsionando uma necessidade crescente de desenvolver infraestruturas mais inteligentes. O objetivo é aprimorar a qualidade dos serviços oferecidos aos cidadãos, fomentando práticas urbanas mais sustentáveis e energeticamente eficientes (Schaffers et al., 2011; Hancke et al., 2013). Isso visa promover a utilização sustentável dos recursos e serviços urbanos, contribuindo para uma melhor qualidade de vida dos habitantes das cidades (Santana et al., 2017).

Tornar as cidades mais inteligentes pode ajudar a otimizar a utilização de recursos e infraestrutura para aumentar sustentabilidade. Uma abordagem envolve combinar criativamente as vastas quantidades de dados gerados por várias fontes da cidade (como redes de sensores, sistemas de tráfego, dispositivos de usuários e redes sociais) para criar serviços e aplicativos integrados, melhorando assim os serviços da cidade e fazendo melhor uso dos seus recursos (Santana et al., 2017).

Entretanto, cabe ressaltar que o conceito de cidades inteligentes possui diversas definições e isso ocorre pela generalidade de significados que este termo pode representar. De uma forma geral, todos os termos referem-se ao uso das Tecnologias de Informação e Comunicação (TICs) para melhorar as capacidades e o desempenho de uma cidade moderna. Contudo, de acordo com Yin et al. (2015), alguns termos são sutilmente diferentes sendo necessário apresentá-los:

- ***Digital City*** - Refere-se à digitalização de uma cidade através da combinação de infraestruturas de comunicação e computação para atender às necessidades do governo, cidadãos e empresas, através da promoção do acesso e compartilhamento de recursos, dados econômicos, sociais, ambientais, populacionais dentre outros.
- ***Intelligent City*** - É uma cidade digital que possui uma camada de inteligência que pode tomar decisões de alto nível com base em um nível de inteligência artificial.
- ***Smart City*** - Consiste em uma cidade inteligente onde a aplicação é focada no uso prático e na experiência do usuário, capaz de se autoadaptar e fornecer interfaces e serviços personalizados. Implica em ser ágil e responsivo ao feedback.

Explorar essas definições é de grande importância, pois esclarece o fato de que incorporar as TICs para realizar as operações da cidade não interpreta totalmente uma cidade inteligente (Hollands, 2008), mas sim, define aspectos relativos às cidades digitais.

Em suma, é possível definir cidades inteligentes como ambientes urbanos que

utilizam TICs e outras tecnologias relacionadas para melhorar a eficiência do desempenho das operações regulares da cidade e a Qualidade dos Serviços (QoS) fornecidos aos cidadãos urbanos, possibilitando uma melhoria na qualidade de vida destes e garantindo a disponibilidade de recursos para as gerações presentes e futuras nos aspectos sociais, econômicos e ambientais (Yin et al., 2015; Silva et al., 2018).

Nesse cenário de serviços urbanos automatizados, aplicações baseadas em IoT para cidades inteligentes têm recebido bastante destaque, tornando os dados associados a essas aplicações em importantes ativos que devem ser adequadamente coletados e processados. Dentre os recursos possíveis para a implementação dessas tarefas, unidades multifuncionais baseadas em sensores têm sido exploradas como um mecanismo acessível para recuperar informações importantes de diferentes áreas das cidades.

Quando tecnologias adequadas são empregadas, dados de múltiplas fontes podem ser processados de forma que serviços públicos como gestão de recursos, mobilidade urbana, assistência à saúde e até segurança pública possam ser aprimorados, tornando as cidades mais inteligentes (Allam e Dhunny, 2019).

Uma variedade de aplicações podem ser apresentadas para o contexto de cidades inteligentes, mantendo as premissas referentes ao aumento da qualidade e aprimoramento dos serviços oferecidos aos cidadãos. A Tabela 2.4 apresenta alguns trabalhos na literatura com contribuições interessantes nesta área além dos que já foram apresentados.

Conforme observado na A Tabela 2.4, os trabalhos apresentados destacam a proposição de aplicações para a melhoria da qualidade de vida nos centros urbanos através da prestação de serviços públicos de qualidade com foco em áreas específicas, como gerenciamento de emergências, poluição do ar, iluminação pública e gerenciamento de vagas de estacionamento. Outra questão importante abordada é com relação à necessidade de uma infraestrutura robusta de rede e segurança para garantir que estas soluções funcionem de forma confiável e segura, sugerindo a adoção de tecnologias de telecomunicações avançadas para a melhorar a eficiência destes sistemas.

Além dos serviços descritos na Tabela 2.4, a gestão de emergências e a prevenção de desastres em cidades inteligentes têm ganhado destaque à medida que o processo de urbanização se intensifica e os impactos negativos das mudanças climáticas se tornam mais evidentes (Costa et al., 2020b).

Tabela 2.4: Desenvolvimento de aplicações para o contexto de Cidades Inteligentes.

Trabalho	Descrição
Mahgoub et al. (2020)	Propuseram o desenvolvimento de um sistema de alarme de incêndio, em contraponto aos alarmes de incêndio convencionais, baseado em IoT e computação de borda para cidades inteligentes. Quando um nó detecta um incêndio, ele sinaliza ao nó centralizado para alertar o usuário e o corpo de bombeiros através do envio de SMS pela rede 4G.
Duangsuwan et al. (2018)	O trabalho apresenta o desenvolvimento de sensores inteligentes de poluição do ar para monitorar a qualidade do ar em cidades inteligentes, promovendo aferições da qualidade do ar em tempo real utilizando IoT como serviço.
Prasad (2020)	Apresenta um estudo de caso de sistema de iluminação inteligente na cidade inteligente de Nagpur - Índia, onde um dos objetivos era reduzir a pegada de carbono reduzindo o consumo de energia. Isso foi alcançado substituindo as 320 luminárias de rua desatualizadas e integrando 63 luzes LED adicionais com sistema de iluminação inteligente de detecção de movimento.
Vakula e Kolli (2017)	Apresentam um sistema de estacionamento baseado na IoT para cidades inteligentes. O sistema de estacionamento proposto contém um módulo IoT implantado no local para gerenciar as vagas de estacionamento disponíveis. Uma plataforma disponibilizada em forma de portal para reserva de lugares de estacionamento.

2.6 Gerenciamento de Emergências em Cidades Inteligentes

Entende-se por emergência uma situação crítica associada a ocorrência de perigo em que a vida, saúde, propriedade ou o meio ambiente possa enfrentar. Quando uma emergência é identificada e medidas de intervenção imediatas são tomadas para que danos e prejuízos sejam evitados ou minimizados, tem-se a definição de um conceito mais amplo, denominado gerenciamento de emergências ou gestão de desastres.

O gerenciamento de emergências é a organização de recursos e responsabilidades que visam abordar todos os aspectos humanitários das emergências, direcionando ações com o objetivo de prevenir e reduzir os efeitos nocivos de todos os perigos, incluindo desastres. Conforme descrito por Costa et al. (2022a), um desastre é observado como a pior consequência de uma emer-

gência em termos de danos infligidos. Dessa forma, buscam-se soluções para evitar ou retardar a ocorrência de desastres, idealmente diminuindo seus impactos quando eles se tornarem inevitáveis.

Diante de toda a problemática relacionada aos ambientes urbanos, provenientes do processo de urbanização acelerado e desordenado, estes espaços se tornaram propensos à diferentes situações de emergências. Diferentes abordagens são esperadas na promoção do enfrentamento à emergências e redução dos seus impactos.

A adoção de tecnologias, arquiteturas, padrões e técnicas são utilizadas na construção de sistemas ciber-físicos para o gerenciamento de emergências em cidades. Estes sistemas processam grandes volumes de dados urbanos provenientes de dispositivos sensores eletrônicos, por exemplo, para fornecer serviços de alerta e mitigação de emergências. Com o amadurecimento da IoT e dos conceitos relacionados às cidades inteligentes, os sensores agora são percebidos em uma perspectiva mais ampla que compreende praticamente qualquer coisa em um ambiente urbano, podendo ser também *smartphones*, drones, veículos, redes de mídias sociais dentre outros (Costa et al., 2022a).

A Tabela 2.5 sumariza alguns trabalhos desenvolvidos para o domínio do gerenciamento de emergências.

Os trabalhos apresentados na Tabela 2.5 abordam a utilização de tecnologias de sensores para a coleta de dados ambientais em tempo real. Baseados no paradigma de IoT, estas soluções visam a detecção, alerta e mitigação de eventos de emergências para os domínios de cidades inteligentes e ambiental. Embora utilizem técnicas de sensoriamento distintos, como por exemplo, o uso de sensores visuais e escalares e ou a combinação entre eles, estas soluções visam melhorar a eficiência, precisão e a rapidez na detecção e mitigação de emergências ambientais, ajudando a salvar vidas e a preservar o meio ambiente.

Entretanto, cabe destacar que algumas dessas soluções propostas são limitadas no que se refere à eventos de monitoramento, não levando em consideração a natureza dinâmica e heterogênea dos ambientes monitorados. Como por exemplo, em (Grover et al., 2020), um incêndio pode não ser o único evento de emergência a ser detectado em uma floresta, o que levaria a necessidade de adotação de outras soluções para o monitoramento e detecção de emergências a serem utilizadas de forma paralela às existentes. Uma solução flexível de monitoramento poderia ser adotada, fazendo com que, diante da necessidade, fossem reajustados os parâmetros de monitoramento para atender a diferentes contextos de emergências. No trabalho de Dragulinescu et al. (2019), por exemplo, não há informações sobre a escala da implantação da solução proposta ou sobre como esta seria implementada para além dos ambientes re-

Tabela 2.5: Desenvolvimento de aplicações para o domínio do Gerenciamento de Emergências.

Trabalho	Descrição
Grover et al. (2020)	Este estudo apresenta um sistema baseado em redes de sensores sem fio para a detecção e mitigação de incêndios florestais. Os autores descrevem a estrutura geral do sistema de detecção, algoritmos de detecção e previsão, topologia da RSSF, técnicas de localização e uma visão de um sistema de mitigação baseado em drone sugerido para localizar o incêndio.
Dragulinescu et al. (2019)	Propõem uma solução eficiente baseada em uma plataforma IoT colaborativa que monitora bairros e alerta moradores e equipes de resposta quando ocorre um perigo em suas residências ou em outros edifícios da região. A solução proposta leva em consideração a utilização de tecnologias de comunicação leves, porém com ampla cobertura, para reduzir os custos relacionados ao consumo de energia.
Costa et al. (2020b)	É proposto um sistema de detecção e alerta de emergência baseado no uso de sensores visuais e escalares para a detecção de eventos críticos em um sistema de alerta de emergência. A utilização de sensores visuais e escalares permite a identificação de dois grupos diferentes de eventos: eventos de instância, onde as variáveis coletadas podem ser representadas diretamente por um único tipo de valor numérico e eventos complexos, que podem ser percebidos processando dados visuais, aumentando a precisão na detecção dos eventos.

sidenciais descritos, o que pode ser um limitador ao pensar na aplicação desta tecnologia em um contexto urbano mais amplo.

Outro ponto a ser destacado é com relação a segurança na comunicação entre os elementos presentes nas arquiteturas das soluções propostas. No trabalho de Costa et al. (2020b), por exemplo, não são mencionados quais mecanismos foram empregados para garantir a autenticidade dos elementos publicadores e assinantes de mensagens ao *broker* bem como técnicas para verificar a autoria de transações e comunicações realizadas no sistema. A falta destes mecanismos impossibilita verificar se as informações obtidas são de fontes confiáveis e quais elementos as produziram. Cabe destacar também que, tanto no trabalho de Dragulinescu et al. (2019) quanto no trabalho de Costa et al. (2020b), não são mencionados questões acerca da privacidade dos dados sensoreados, o que pode ser um problema significativo, especialmente em relação ao monitoramento de dados pessoais e geração de imagens da população.

Em Rangra e Sehgal (2022), através do uso da Internet Social das Coisas

(SIoT), é proposta uma abordagem para o gerenciamento de desastres naturais, onde a pré-identificação de comunidades que podem se expor a desastres naturais e a identificação do melhor nó onde o sistema de transmissão pode ser colocado é obtido através de um novo modelo de computação distribuída, que processa com eficiência informações de condições climáticas obtidas através da análise de mídias sociais. Outra aplicação do uso de elementos diversos para o gerenciamento de emergência em cidades está presente em Khan e Neustaedter (2019), onde os autores avaliam a viabilidade do uso de drones, por parte do Corpo de Bombeiros do Canadá, como tecnologia para apoiar o trabalho de primeira resposta à emergências cotidianas.

Diversas soluções promissoras para lidar com a detecção e o gerenciamento de emergências têm sido exploradas, automatizando os processos de identificação, alerta e mitigação de incidentes em potencial de maneira rápida e distribuída (Shah et al., 2019). Entretanto, embora a diversidade tecnológica empregada no desenvolvimento dessas aplicações traga benefícios para a sociedade que faz usufruto destas, isso acarreta em uma série de desafios para pesquisadores acadêmicos e industriais, uma vez que há a necessidade, por exemplo, da adaptabilidade das tecnologias aos contextos dinâmicos dos ambientes de monitoramento e da interoperabilidade destas soluções entre os diversos componentes e tecnologias já existentes.

2.7 Trabalhos Relacionados

Dentre o escopo de aplicações típicas para o domínio das cidades inteligentes, aquelas associadas à detecção, alerta e resposta a emergências têm recebido bastante atenção, uma vez que o alerta de emergências é considerado como uma tarefa bastante complexa, principalmente em cenários de grande escala, como as cidades (Costa et al., 2020a).

A adoção de arquiteturas, metodologias, técnicas e *frameworks* de referência para auxiliar na construção de aplicações de IoT vem sendo discutida ao longo dos últimos anos por diversos pesquisadores. Os resultados alcançados são fundamentais para a construção de sistemas de gerenciamento de emergência baseados em IoT que sejam dinâmicos, adaptáveis, escaláveis, personalizáveis, fáceis de manter e interoperáveis, tornando-os mais eficientes e eficazes em diferentes contextos e cenários.

No estudo conduzido por Costa e de Oliveira (2020), é discutida a configuração dinâmica de redes de sensores sem fio, apresentando uma nova metodologia para melhorar a resposta destas infraestruturas diante de eventos capturados. Esta abordagem leva em conta diversos fatores urbanos que influenciam diretamente nos resultados, como clima, tráfego, temperatura e dia da se-

mana. Os autores propõem que essa metodologia oriente o desenvolvimento de aplicativos baseados em sensores para diferentes cenários urbanos, considerando objetivos de monitoramento, parâmetros de priorização contextual e a detecção de eventos críticos.

Além disso, Costa e de Oliveira (2020) destaca que a configuração dinâmica permite que os nós sensores se adaptem às mudanças urbanas em tempo real. Uma abordagem comum para avaliar a prioridade de eventos é atribuir a cada tipo de evento um nível de prioridade. Com base nesses conceitos, é possível pré-configurar os nós para que percebam e se adaptem a essas mudanças, gerando respostas diferentes sem a necessidade de intervenção humana a cada mudança de parâmetro. Os nós capturam o evento e, com base na prioridade associada, executam uma ação correspondente.

No entanto, este estudo não aborda um aspecto relevante, que é a segurança do canal de comunicação durante a transmissão de mensagens. Embora seja mencionada a necessidade de segurança de ponta a ponta, por meio do uso de criptografia assimétrica e certificados digitais, não foram definidos os procedimentos para realizar esse processo, incluindo os mecanismos de criptografia e autenticação a serem empregados. Isso é especialmente crucial considerando a limitação de recursos computacionais dos nós sensores.

O desenvolvimento de soluções de sensoreamento de baixo custo também vem sendo explorado por pesquisadores. de unidades móveis de sensores multiflexíveis baseadas em plataformas de *hardware* de código aberto e um *framework* de referência é descrito por Oliveira et al. (2021). O objetivo é fornecer uma solução acessível e fácil de usar para coleta de dados em uma variedade de aplicações de IoT. Uma vez que a coleta de dados de sensores é uma parte importante de atividades de pesquisa e monitoramento, esta não pode ser limitada e inacessível, muitas vezes por restrições de *hardware* e ou de *softwares* proprietários. Além da solução baseada em código aberto, os autores propõem um *framework* de referência capaz de fornecer uma estrutura para a aquisição, armazenamento e análise de dados de sensores, independente do tipo de sensor utilizado, promovendo assim flexibilidade para a construção de aplicações de um amplo escopo, com especial atenção para aplicações como detecção de emergências, agricultura inteligente e monitoramento urbano (Oliveira et al., 2021).

O desenvolvimento de soluções de baixo custo voltadas para a detecção e resposta a eventos críticos tem sido explorado por diversos pesquisadores. Um exemplo é a descrição de unidades móveis de sensores multiflexíveis, baseadas em plataformas de *hardware* de código aberto, conforme descrito por Oliveira et al. (2021). O objetivo é oferecer uma solução acessível e de fácil utilização para a coleta de dados em uma variedade de aplicações de IoT.

Como a coleta de dados de sensores é crucial para atividades de pesquisa e monitoramento, ela não deve ser limitada ou inacessível, muitas vezes devido a restrições de *hardware* e/ou *software* proprietários.

Além da solução baseada em código aberto, os autores propõem um *framework* de referência capaz de fornecer uma estrutura para a aquisição, armazenamento e análise de dados de sensores, independentemente do tipo de sensor utilizado. Isso promove flexibilidade na construção de aplicações de amplo escopo, com especial atenção para áreas como detecção de emergências, agricultura inteligente e monitoramento urbano (Oliveira et al., 2021).

As implementações, experimentos e análises realizados por Oliveira et al. (2021) são contribuições importantes para dar suporte a construção de soluções em IoT para cenários dinâmicos e heterogêneos. Entretanto, conforme observado pelos autores, unidades de monitoramento configuráveis de forma flexível são projetadas para operar e reagir através de requisições advindas da internet ou mesmo através de uma conexão sem fio. Como consequência, tais sistemas podem enfrentar diferentes ameaças à segurança, o que pode comprometer a eficácia dos sistemas de maneiras imprevisíveis. Assim, há uma necessidade na observância dos aspectos de segurança associados à implementação de soluções que promovam flexibilidade e adaptabilidade aos cenários através de requisições remotas.

No trabalho de Costa et al. (2020a), é apresentado um sistema de alerta de emergência multicamada distribuído, denominado *CityAlarm*, construído em torno de unidades de detecção de eventos baseadas em sensores. Este sistema fornece informações georreferenciadas em tempo real sobre eventos críticos, utilizando um conjunto configurável de diferentes sensores escalares e dados de GPS como entrada. Os autores delineiam o funcionamento do sistema, descrevendo o fluxo teórico de mensagens e a comunicação entre as unidades dentro da arquitetura. Além disso, sugerem protocolos de comunicação para a camada de aplicação da IoT.

Conforme observado por Costa et al. (2020a), os atuais sistemas de emergência baseados em IoT têm limitações ao considerar o escopo mais amplo de cidades inteligentes, explorando uma ou apenas algumas variáveis de monitoramento ou mesmo alocando alta carga computacional para nós de sensores regulares. Um sistema de alerta de emergência para ambientes de cidades inteligentes em larga escala deve ser flexível o suficiente para suportar diferentes abordagens quando os alarmes de emergência são disparados, permitindo qualquer processamento dos alarmes recebidos, garantindo uma melhor percepção do ambiente monitorado. A modularização do sistema possibilita uma melhor adaptação a diferentes escopos de monitoramento, facilitando a definição de eventos de interesse, alarmes de emergência e zonas de risco.

Entretanto, alguns pontos que merecem destaque não foram mencionados, em especial, pontos relacionados à segurança. É possível observar que na proposta do *CityAlarm* é retratado apenas que o uso de criptografia é a ferramenta mais eficaz para proteger dados e autenticar os seus elementos. Não são definidas técnicas para garantir premissas básicas de segurança, como autenticidade e não-repúdio, capazes de garantir que quaisquer adaptações e ou reconfigurações do sistema para uma nova abordagem de monitoramento possam ser provenientes de determinada fonte, assegurando que foram criados, expedidos ou alterados por certo órgão, entidade ou sistema autenticado e que este não negue a autoria de uma ação específica.

Diante do panorama de tecnologias IoT para cidades inteligentes apresentadas no decorrer deste capítulo, observa-se através dos trabalhos de (Costa e de Oliveira, 2020; Oliveira et al., 2021; Costa et al., 2020a) esforços para a proposição de soluções para o domínio de aplicações associadas à detecção, alerta e resposta à emergências que se adaptem ao contexto dinâmico e heterogêneo dos ambientes urbanos. Embora estes trabalhos promovam considerações valiosas relacionadas a aplicações adaptáveis e dinâmicas para o contexto de cidades inteligentes, a falta de uma arquitetura de referência para este domínio em especial, dificulta a promoção de outras abordagens do mesmo campo aplicacional.

Nessa perspectiva, o trabalho proposto visa preencher lacunas relacionadas à segurança nos processos de comunicação e no funcionamento reconfigurável de unidades de detecção de emergências, baseadas em sensores, por meio da modelagem e desenvolvimento de uma arquitetura de referência para a construção de aplicações destinadas ao gerenciamento de emergências em cidades inteligentes. O objetivo é garantir que as unidades de detecção de emergências possam ser reconfiguradas de forma remota, de maneira íntegra e autêntica, contribuindo significativamente para a eficácia, eficiência e qualidade das aplicações no domínio do gerenciamento de emergências em cidades inteligentes.

Capítulo 3

Metodologia

O desenvolvimento deste trabalho foi fundamentado nos conceitos relativos ao emprego de técnicas e metodologias pertencentes ao contexto de aplicações de cidades inteligentes, em especial, para aquelas relacionados ao gerenciamento de eventos de emergências em cidades inteligentes. Partindo deste pressuposto, este trabalho propõe uma arquitetura capaz de orientar o desenvolvimento, implantação e operação de EDUs, baseadas em sensores, que podem ser atualizadas e ou reconfiguradas de forma segura através de solicitações remotas por meio de uma infraestrutura de comunicação sem fio. A agregação de novos requisitos de monitoramento possibilita às EDUs, adaptar-se aos ambientes dinâmicos e heterogêneos de eventos críticos existentes em centros urbanos, possibilitando tomadas de decisões mais eficazes e coerentes à medida que situações emergenciais são detectadas.

Quando consideramos que uma cidade é “inteligente”, assumimos que um conjunto de sensores eventualmente implantados devem ser capazes de capturar eventos e realizar uma ou mais ações em prol da aplicação a qual ele foi designado (Oliveira, 2018).

Uma aplicação para cidade inteligente é um propósito específico que um conjunto de sensores possuem, visando solucionar ou prevenir algum tipo de problema. Dentre algumas aplicações, é possível destacar:

- Monitoramento de variáveis ambientais (temperatura, humidade do ar, pressão atmosférica, ruído, dentre outros);
- Alerta de terremotos e ou inundações;
- Identificação de indivíduos;
- Controle semafórico dentre outros.

Entretanto, é possível observar que as soluções de sensoriamento existentes geralmente se concentram em um único fenômeno de monitoramento,

tornando-as limitadas no sentido de não se adaptarem a diferentes cenários ou ambientes com grande variabilidade de eventos, com é o caso dos centros urbanos.

A reconfiguração de unidades de sensoriamento apresenta diversas questões relevantes para aplicações de IoT, especialmente aquelas relacionadas ao surgimento de novos requisitos de monitoramento diante da ocorrência de eventos críticos (Costa et al., 2020b). Além disso, a capacidade de resposta imediata e adequada, conforme o contexto e as circunstâncias do ambiente monitorado, é uma premissa a ser alcançada.

3.1 Formalização do Problema

Conforme apresentado na Seção 2.1, a urbanização e problemas associados a essa ocupação desordenada tornaram as cidades altamente suscetíveis a desastres decorrentes de situações de emergência. As cidades são ambientes dinâmicos e heterogêneos, com seus espaços em constante mudança devido à ação humana ou natural. Esse processo urbanístico cria desafios significativos para o setor público e para a população que depende dos serviços e infraestrutura oferecidos. Em meio a esse cenário complexo, um dos grandes desafios das cidades inteligentes é a promoção de serviços adequados, ou seja, serviços coerentes com a realidade correspondente. Ao desenvolver soluções de IoT para a detecção de emergências voltadas ao domínio das cidades inteligentes, os sensores requeridos podem apresentar diversidades significativas de acordo com os parâmetros a serem identificados, taxas de coleta de dados, técnicas de priorização e demais algoritmos de análise empregados. Adicionalmente, os sistemas de detecção de emergências implantados em uma cidade específica podem estar sujeitos a demandas distintas que só se tornarão evidentes após sua implantação inicial, podendo estas modificarem ao longo do tempo.

Esta situação pode gerar um risco à população em relação ao tempo de resposta e tomada de decisão mediante à situações catastróficas. Como garantir que uma aplicação para o gerenciamento de emergências em cidades inteligentes é eficaz e eficiente sendo que emergências podem ser desencadeadas por uma variedade de eventos e situações de diferentes graus e com diferentes magnitudes que dependem da natureza do incidente e do ambiente na qual ela foi gerada? Como princípio fundamental estabelecido neste estudo, um sistema voltado ao gerenciamento de emergências em cidades inteligentes deve ser reconfigurável diante do ambiente monitorado. Sistemas que se concentram em um único fenômeno e metodologia de monitoramento, ou seja, limitados, é ineficaz, pois, conforme mencionado, eventos de emergências em

uma cidade são imprevisíveis, seja na sua ocorrência, seja na combinação ou não de fatores e situações adversas que levaram a sua ocorrência.

O desacoplamento esperado entre *hardware* e *software* possibilita que componentes de entrada, como diferentes tipos de dados e dispositivos sensores, possam sofrer variações, assim como os mecanismos de decisão para a detecção de eventos de emergências. Isso permitirá que as EDUs, dotadas de um conjunto de sensores semelhantes ou não, possam monitorar eventos de emergências distintos, além de ser possível a modificação e adaptação de novos recursos conforme a necessidade do monitoramento de eventos de interesse. De forma mais simples, em uma cidade inteligente, é possível ter uma aplicação para a detecção de eventos de incêndio que utilize unidades compostas de sensores de temperatura, umidade e detecção de gases tóxicos. Uma vez que não haja mais necessidade do monitoramento desse evento, ou uma outra aplicação de monitoramento de eventos de emergência de maior importância e eficácia deva ser implantada, mantém-se os sensores descritos, alterando apenas os padrões que condicionam o funcionamento da operação desejada.

No entanto, o desenvolvimento de uma arquitetura de IoT que possibilite a reconfiguração de unidades de sensoriamento de acordo com o ambiente monitorado, embora extremamente necessária, traz consigo desafios de segurança intrínsecos à sua tecnologia de comunicação, em grande parte sem fio, devido à localização remota e, por vezes, de difícil acesso dos dispositivos sensores. Especificamente, questões relacionadas à autenticidade dos dispositivos comunicantes e à integridade das solicitações e mensagens trocadas entre eles. O objetivo é garantir que os componentes da aplicação possam autenticar a identidade uns dos outros, assegurando que apenas aqueles que realmente são quem afirmam ser possam disponibilizar informações ou recursos, enquanto a integridade garante a consistência, precisão e confiabilidade dos dados ao longo de sua vida útil. Assim, ao solicitar a incorporação de novos parâmetros, estratégias de priorização ou outros algoritmos para detecção e processamento de eventos de emergência nos dispositivos sensores, é possível atribuir um certo nível de confiabilidade à incorporação desses novos parâmetros, evitando a operação incorreta dos dispositivos sensores devido à corrupção das mensagens.

Nesse sentido, o presente trabalho estabelece uma arquitetura de IoT voltada para o desenvolvimento, implantação e operação de sistemas baseados em sensores mais eficazes e eficientes, direcionados ao gerenciamento de emergências em cidades inteligentes. Destaca-se a necessidade de reconfiguração das unidades de sensoriamento de acordo com o contexto monitorado, além de garantir a segurança, integridade e desempenho contínuo dos componentes contra ameaças cibernéticas em constante evolução, inerentes ao paradigma da IoT e aos sistemas de comunicação.

3.2 Metodologia de Desenvolvimento do Trabalho

Uma pesquisa pode ser definida como o processo formal e sistemático de desenvolvimento do método científico, sendo que esta possui como objetivo fundamental descobrir respostas para problemas mediante o emprego de procedimentos científicos (Gil, 2008). De modo a garantir a qualidade e validade dos resultados obtidos, permitindo que esta pesquisa seja replicada e os resultados possam ser confiáveis e generalizáveis, foram utilizados alguns métodos científicos que definiram o conjunto de etapas a serem realizadas.

Sendo assim, para o desenvolvimento deste trabalho, foram definidas etapas em ordem cronológica descritas nos subtópicos a seguir.

3.2.1 Definição da arquitetura de comunicação e elementos de *hardware* e *software* associados

A arquitetura de comunicação é um conjunto de regras, protocolos, dispositivos e padrões que governam a comunicação entre entre elementos em uma rede. No caso da arquitetura de comunicação para aplicações de IoT, esta define como os sensores coletam e transmitem dados, como os dispositivos de gerenciamento de rede acessam e processam esses dados, bem como a rede é escalonada e gerenciada. Outra característica inerente às arquiteturas de comunicação é que estas geralmente incluem mecanismos de segurança para proteger os dados transmitidos e garantir a confiabilidade da rede.

Nesta etapa foram levantados quais elementos de *hardware* e *software* devem ser considerados para o desenvolvimento do modelo conceitual a partir dos parâmetros observados e definidos no levantamento bibliográfico. Este modelo foi evoluindo durante o desenvolvimento do projeto a medida que novas necessidades eram percebidas.

Espera-se que a definição de protocolos de segurança, comunicação e elementos associados possa permitir que os dispositivos sensores interconectados possam detectar eventos críticos relativos a potenciais emergências de forma rápida e distribuída bem como serem capazes de lidar com falhas, imprevistos diversos e solicitações de reconfigurações para atendimento de novos requisitos de sensoriamento com um menor tempo de interrupção dos seus serviços sem que hajam perdas substanciais dos dados sensoriados.

É importante ressaltar que, para o tipo de aplicação em questão, foi levado em consideração aspectos como a complexidade e tamanho das mensagens para processamento em dispositivos baseados em sensores de baixo consumo de energético, garantindo a eficiência e eficácia desta proposta.

3.2.2 Definição da modelagem conceitual para o desenvolvimento da arquitetura proposta

Nesta fase, foram tomadas as principais decisões em relação à proposição deste projeto como forma de melhor resolver o problema relacionado à construção de aplicações de IoT para o gerenciamento de emergências em ambientes altamente heterogêneos e dinâmicos e com vasta área de abrangência, características relativas aos centros urbanos.

Dessa forma, foi definido a modelagem conceitual, representação abstrata da arquitetura proposta, a partir de parâmetros bem definidos, como por exemplo, evitar o processamento de dados complexos diante das restrições de *hardware* de dispositivos de IoT, utilizar protocolos e padrões de comunicação abertos e mecanismos de segurança que possam garantir a operação eficaz e eficiente de aplicações que se baseiam na arquitetura apresentada.

O modelo conceitual teve como objetivo ajudar na compreensão dos seus requisitos, bem como, os processos relacionados à comunicação entre os seus elementos e suas relações com o domínio de aplicações para o gerenciamento de emergências em cidades inteligentes. Para apresentar o modelo conceitual, foram utilizados diagramas de sequência (modelos arquiteturais) para a representação dos componentes relacionados e a construção de algoritmos como forma de descrever o *middleware* responsável pelo processo de agregação de novos requisitos de monitoramento de eventos de interesse de maneira dinâmica e flexível através da arquitetura proposta.

3.2.3 Implementação e validação da arquitetura proposta através de uma prova de conceito

Nesta etapa foi desenvolvida uma prova de conceito, modelo prático para provar conceitos teóricos e validar as funcionalidades esperadas da arquitetura proposta. Diferentes tecnologias e ferramentas foram utilizadas para a sua implementação, permitindo uma avaliação inicial com base em uma abordagem de prova de conceito. As funcionalidades descritas no Solicitante de Atualização e Servidor de Atualização foram desenvolvidas usando a linguagem *Python* 3. Para o controle, gerenciamento e promoção da comunicação entre os elementos pertencentes à arquitetura proposta, foi utilizado o MQTT *Broker Eclipse Mosquitto* versão 2.0.15, que implementa várias versões do protocolo MQTT 5.0 (Light, 2017).

Uma placa ESP32 com microcontrolador integrado, equipada com capacidade de comunicação *Wi-Fi* e *Bluetooth*, foi utilizada para implementar uma EDU. A programação responsável por permitir que o microcontrolador atualize a

lógica de operação de seus sensores foi escrita em *MicroPython*, que consiste em uma implementação de *software* de uma linguagem de programação amplamente compatível com o *Python* 3, otimizada para ser executada em microcontroladores.

Para validar as funcionalidades esperadas, diferentes versões da lógica de operação dos sensores foram enviadas à EDU. Em cada processo de implementação, era verificado nesta unidade qual a lógica de operação vigente, comprovando assim o funcionamento da arquitetura proposta e sua capacidade de implementar, de forma segura, novos requisitos de monitoramento mediante solicitações de atualização remotas por meio da abordagem OTA.

As etapas metodológicas descrita nesse capítulo visam assegurar não apenas a fundamentação teórica sólida, mas também a aplicação prática e a validação concreta da arquitetura proposta. Este capítulo fornece, assim, um arcabouço metodológico robusto que orientará as fases subsequentes deste trabalho, contribuindo para a construção de uma base sólida e confiável para o desenvolvimento e avaliação da arquitetura de detecção de emergências em cidades inteligentes.

Capítulo 4

Arquitetura Proposta

Este capítulo descreve, em detalhes, os componentes fundamentais para a concepção de uma arquitetura de IoT que viabilize o desenvolvimento, a implementação e a operação de Unidades de Detecção de Emergências (EDUs) capazes de suportarem e processarem solicitações de reconfiguração remotas, através de uma infraestrutura de comunicação sem fio, levando em consideração algumas premissas básicas de segurança.

A reconfiguração de unidades de sensoriamento é um assunto de extrema relevância. A dinamicidade, heterogeneidade e complexidade dos ambientes urbanos influenciam diretamente o contexto no qual um sistema de monitoramento e detecção de emergências está operando. Nesse sentido, as demandas sobre esse sistema são suscetíveis a mudanças, exigindo a implementação de novas abordagens e percepções em relação aos eventos observados. Além disso, são necessários aprimoramentos e correções nos algoritmos utilizados para garantir tomadas de decisões mais eficazes e coerentes com a realidade monitorada.

O cenário apresentado na Figura 4.1 ilustra a arquitetura proposta, enfatizando seus componentes fundamentais e suas interações para permitir que as EDUs processem solicitações de atualizações remotas de maneira segura, através de uma infraestrutura de comunicação sem fio.

Para alcançar os resultados definidos na Figura 4.1, é necessário especificar os conceitos teóricos subjacentes. A Seção 4.1 discute a tecnologia de comunicação e o método de envio de solicitações de reconfiguração. A Seção 4.2, define os componentes e apresenta o modelo de domínio da arquitetura, destacando os procedimentos inerentes ao processo de encaminhamento de solicitações de reconfiguração de EDUs. O processo de conexão e autenticação do servidor de atualização e das EDUs ao *Broker* MQTT é descrito na Seção 4.3, enquanto a publicação e assinatura de tópicos serão abordados na

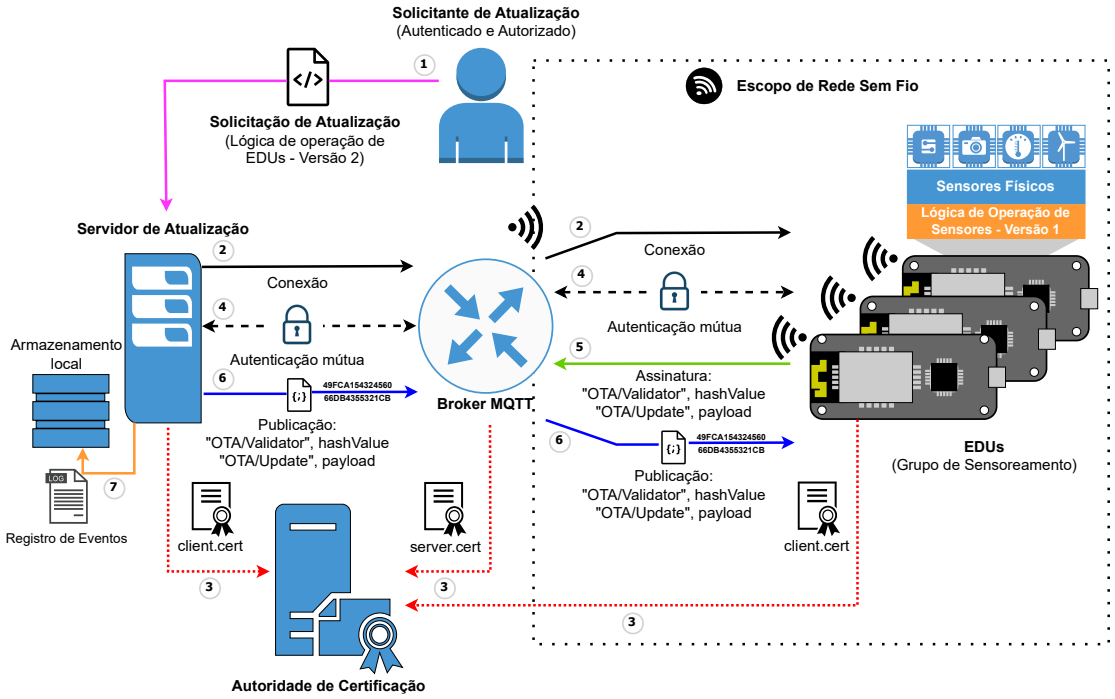


Figura 4.1: Arquitetura proposta, componentes fundamentais e interações.

Seção 4.4. Por fim, a Seção 4.5, apresenta a modelagem final da arquitetura, facilitando a compreensão das interações entre os componentes e garantindo a confiabilidade do sistema em ambientes de monitoramento dinâmicos e heterogêneos.

4.1 Definição da Tecnologia de Comunicação e Método de Envio de Atualizações

Aplicações de IoT tendem a incorporar pequenos componentes de rádio sem fio, baratos e eficientes, nos mais diversos objetos do cotidiano (Bauwens et al., 2020). Definir a tecnologia de comunicação depende de vários fatores, como requisitos da aplicação, escala do projeto, eficiência energética e cobertura necessária. Na Seção 2.4, várias tecnologias foram descritas, considerando esses requisitos. Para aplicações de IoT do domínio das cidades inteligentes, é sugerido pela literatura a utilização de padrões e tecnologias de comunicação sem fio de longa distância, como por exemplo, *Narrowband IoT* (NB-IoT) e *Long Range Wide Area Network* (LoRaWAN) (Chen et al., 2017a; Haxhibeqiri et al., 2018; Khutsoane et al., 2017). Contudo, para este trabalho, e em especial para a construção da prova de conceito apresentada no Capítulo 5, foi adotada a tecnologia de comunicação sem fio WI-FI, padrão 802.11N.

Os requisitos específicos da arquitetura proposta influenciam tanto na escolha da tecnologia de comunicação sem fio quanto nos métodos para o envio de atualizações aos dispositivos comunicantes. Ao projetar aplicações de sensoriamento para cidades inteligentes, além de considerar os aspectos como heterogeneidade e dinamicidade dos ambientes urbanos, é crucial levar em consideração a magnitude territorial, que frequentemente resultam em um grande número de dispositivos implantados em diversos locais, muitas vezes de de difícil acesso. A distribuição desses dispositivos em locais remotos torna o processo de manutenção demorado e custoso. Para contornar esse desafio, a solução é utilizar atualizações *Over-The-Air* (OTA), que podem ser realizadas remotamente, aproveitando a rede de comunicação para distribuir atualização de *software* ou *firmware* aos dispositivos (de Sousa et al., 2022).

As atualizações via OTA referem-se a um conjunto de tecnologias e métodos que permitem a atualizar dispositivos, como smartphones, dispositivos IoT e veículos conectados, por meio de uma conexão sem fio, sem intervenção manual intermediária, sendo particularmente vantajosa. Essa abordagem é cada vez mais importante devido à proliferação de dispositivos conectados e a necessidade de mantê-los atualizados com correções de segurança, novos recursos e melhorias de desempenho, permitindo que atendam demandas constantes.

O processo de atualização via OTA e seus componentes são descritos na Figura 4.2. Ele se inicia com o desenvolvimento de uma nova versão de *firmware* ou *software*, que é mantida em um servidor remoto. Os dispositivos de IoT verificam periodicamente a disponibilidade de atualizações por meio de uma conexão com o servidor. Se houver atualizações disponíveis, eles as baixam e, após a conclusão do *download*, a atualização é instalada e o dispositivo é reiniciado para ativá-la.

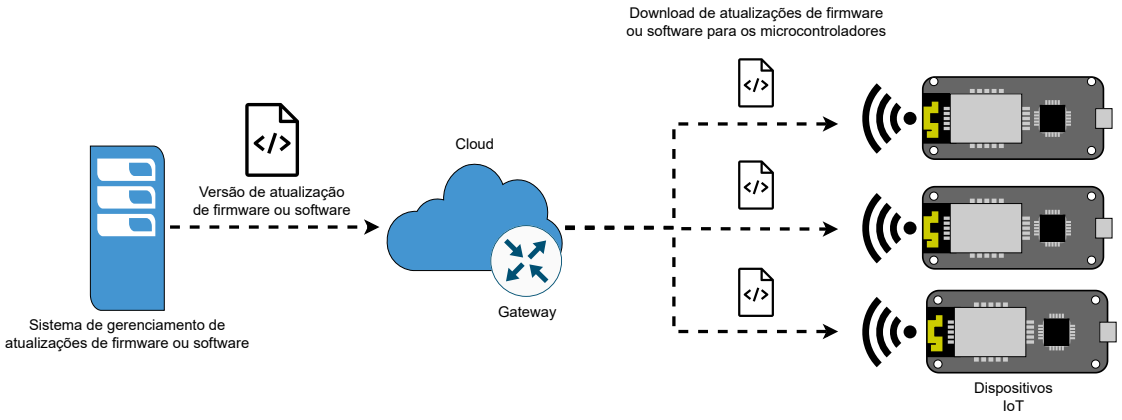


Figura 4.2: Processo de atualização OTA para dispositivos IoT.

A abordagem de atualização OTA é essencial para a manutenção das EDUs por meio da arquitetura proposta. Ela garante a distribuição das solicitações de atualização contendo os parâmetros de reconfiguração necessários para a operação eficaz e coerente dos dispositivos.

4.2 Definição de Componentes e Modelo de Domínio

Os conceitos relativos aos componentes e procedimentos essenciais da arquitetura proposta são expressos a seguir. Cabe ressaltar que as especificações de implementação, além dos requisitos inerentes à execução destes, serão abordadas no Capítulo 5.

- **Unidades de Detecção de Emergências (EDUs)** - Como já mencionado, as EDUs são os componentes responsáveis por detectar condições emergenciais. Equipadas com uma variedade de sensores, esses dispositivos monitoram e detectam eventos críticos em ambientes urbanos. Em uma cidade inteligente, caracterizada pela dinamicidade e heterogeneidade, é essencial que a maneira de lidar com esses sensores se adapte ao longo do tempo, exigindo a reconfiguração das EDUs. Esse processo deve ser realizado mediante solicitação de atualização, seguindo um conjunto de regras e especificações dos métodos de monitoramento e detecção de eventos necessários. Uma tabela com a sugestão do conjunto de sensores para construir uma EDU é apresentada no trabalho de Costa et al. (2020b). Essas unidades devem ser capazes de detectar um ou mais tipos de eventos de interesse, dependendo do número e tipo de dispositivos sensores utilizados (Costa et al., 2020a).

  Para o desenvolvimento das EDUs, visando garantir um certo nível de acessibilidade (com baixo custo de implantação em larga escala) e facilidade de configuração (possibilidade de atualização e reconfiguração a qualquer momento após a implementação), sugere-se que essas unidades possuam recursos adequados de processamento, comunicação e armazenamento compatíveis com as plataformas de hardware de código aberto mais populares, como *Raspberry Pi*, *Cubieboard*, ESP32, *Beaglebone*, entre outras.

  A utilização de componentes capazes de gerenciar e promover eficientemente a comunicação permitirá a transmissão remota de mensagens de atualização e reconfiguração para as EDUs em um contexto específico de monitoramento. Aspectos como a complexidade e o tamanho das mensagens para processamento em dispositivos baseados em sensores de baixa potência também devem ser considerados.

- **Solicitante de Atualização** - Um solicitante de atualização é um componente da arquitetura de comunicação responsável por encaminhar código executável, dados, recursos e/ou parâmetros de reconfiguração para as EDUs existentes em uma área de monitoramento, visando aprimorar ou incorporar funcionalidades ao contexto operacional dessas unidades. Esse elemento inicia uma solicitação de atualização por meio de uma interface com o gerenciador de sistema utilizado pelo servidor de atualização. Mecanismos de autenticação de acesso do solicitante de atualização à interface do gerenciador do sistema devem ser implementados para garantir a segurança e a privacidade dos dados e recursos associados a uma conta ou perfil de usuário.
- **Solicitação de Atualização** - Uma solicitação de atualização é representada por um arquivo no formato JSON, contendo parâmetros operacionais usados para reconfigurar as EDUs. Esse arquivo é transmitido às EDUs por meio de publicações feitas em tópicos pelo servidor de atualização para o *Broker*. Ao receber a solicitação de atualização, as EDUs transferem o conteúdo do arquivo para sua memória, incorporando os novos requisitos operacionais. A reconfiguração das EDUs se torna necessária diante de questões como a correção de falhas, refinamento do comportamento de sensoriamento, ajuste de métricas, melhoria de desempenho e implementação de novos recursos.

  O uso do formato JSON é justificado pelas suas característica de ser leve, de fácil leitura e viabiliza o armazenamento e a transmissão de informações estruturadas. Baseado em texto e menos complexo, é intuitivo e amplamente suportado por várias linguagens de programação, incluindo *JavaScript*, *Python*, *Ruby* e *Java*, garantindo interoperabilidade entre diferentes sistemas operacionais e ambientes. É comumente utilizado em aplicações que demandam troca de dados em tempo real e integração com outros sistemas, como interfaces de programação de aplicações (*APIs*).
- ***Broker* MQTT** - Para o controle, gerenciamento e encaminhamento das solicitações de atualização entre o solicitante de atualização e as EDUs, é definida a implementação de um *Broker* MQTT. O *Broker* é um componente do protocolo MQTT cuja função é lidar com o recebimento e encaminhamento de mensagens entre elementos publicadores e elementos assinantes em uma infraestrutura de comunicação. Todo o processo de entrega de mensagens entre publicadores e unidades assinantes não ocorre diretamente, havendo a necessidade de um *Broker* para realizar essa intermediação e gerenciar novos assinantes do sistema (Costa et al., 2020b). Dependendo de sua configuração, o *Broker* pode gerenciar centenas a milhares de clientes simultâneos, garantindo aspectos de

segurança relacionados à autenticação e autorização.

Muitas soluções baseadas em MQTT têm sido implementadas recentemente devido à flexibilidade e facilidade de uso da arquitetura de publicação/assinatura, que pode operar sobre qualquer protocolo de rede (Costa et al., 2022a), permitindo a interoperabilidade entre os mais diversos sistemas existentes.

- **Servidor de Atualização** - O servidor de atualização é um serviço criado para receber solicitações de atualização e encaminhá-las ao *Broker* para publicação por meio de tópicos específicos. Além disso, este serviço tem a capacidade de armazenar informações em um diretório específico relacionado à ocorrência de eventos de solicitações de atualização das EDUs, garantindo a autenticidade das ações de encaminhamento de atualizações e possibilitando a criação de trilhas de auditoria para o sistema proposto.

- **Autoridade de Certificação** - Conforme apresentado na Subseção 2.3.5, a ampla adoção do paradigma da IoT trouxe novas oportunidades para ataques cibernéticos. Garantir a confiabilidade dos dados, a manutenção dos dispositivos e a auto-calibração são essenciais para um ecossistema de IoT confiável (Lukaj et al., 2023). A instalação de um dispositivo IoT comprometido ou danificado pode invalidar serviços e, em alguns casos, a capacidade dos sistemas de tomar decisões adequadas. A inserção de dispositivos não certificados em uma arquitetura de IoT pode expor o sistema a diversos ataques cibernéticos, como a interrupção dos serviços ou a produção e disseminação de dados não confiáveis.

  Uma autoridade de certificação (CA) é uma entidade confiável responsável por emitir e gerenciar certificados digitais usados em sistemas de infraestrutura de chave pública (PKI), a fim de verificar a identidade de indivíduos ou entidades no mundo digital e garantir a integridade e autenticidade das comunicações e transações digitais (Lukaj et al., 2023; Palmo et al., 2022). Um certificado digital é um documento digital que vincula a identidade de uma entidade (como uma pessoa, organização ou site) a uma chave pública. Nesta abordagem, a autoridade de certificação deve desempenhar funções específicas: certificar de maneira mútua a identidade de cada dispositivo IoT em um domínio isolado e a identidade do servidor de atualizações ao *Broker* MQTT.

  Vale ressaltar que o detalhamento da implementação de uma autoridade de certificação está fora do escopo deste trabalho. O que está sendo definido nesta abordagem é a utilização deste serviço para a verificação da identidade dos dispositivos mencionados.

A partir da especificação dos componentes e procedimentos apresentados, é possível extrair o modelo de domínio da solução para a formalização dos conceitos que serão apresentados nas sessões 4.3 (Processo de Conexão e Autenticação) e 4.4 (Publicação e Assinatura de Tópicos no Broker MQTT). O modelo de domínio é representado no diagrama da Figura 4.3, ilustrando a interação entre os elementos já descritos.

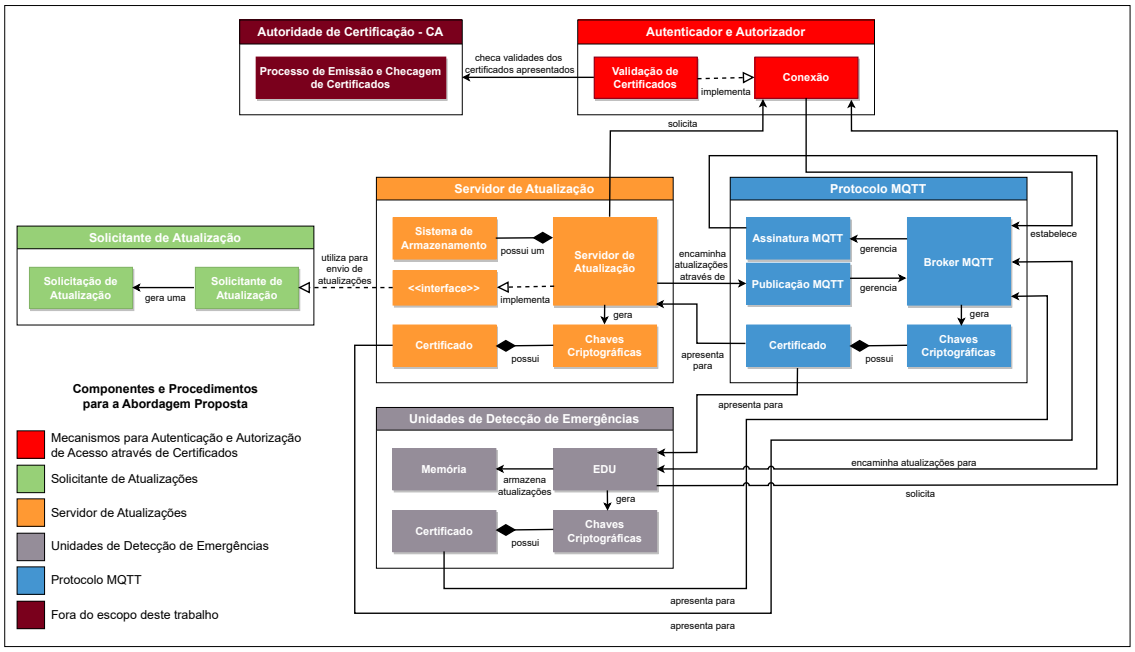


Figura 4.3: Modelo de domínio da arquitetura proposta.

4.3 Processo de Conexão e Autenticação

O *Broker* e o protocolo MQTT atuam como uma interface simples e comum, permitindo a conexão e comunicação de diversas tecnologias. Uma vez que todo o processo de atualização das EDUs proposto neste trabalho é realizado remotamente, por meio de solicitações OTA, torna-se necessário estabelecer fluxos de comunicação seguros e eficientes para a entrega confiável das solicitações de atualizações a esses dispositivos, bem como garantir que apenas dispositivos autorizadas possam publicar e assinar tópicos MQTT através de um *Broker*.

Possíveis ataques em sistemas OTA incluem ataques *Man-in-the-Middle*, injeção de código, repetição de pacotes, falsificação de identidade e negociação de *downgrade*. Esses ataques visam comprometer atualizações de software, capturar informações confidenciais e explorar vulnerabilidades. Portanto, é essencial implementar medidas de segurança, como autenticação, criptogra-

fia e mecanismos para validação da integridade de dados de modo a proteger o processo de atualizações e garantir a segurança dos componentes pertencentes à arquitetura.

Nesse sentido, é especificado o uso de autenticação mútua baseada em certificados para verificar a identidade dos elementos pertencentes à arquitetura. Isso envolve o uso de certificados digitais para garantir a autenticidade das partes envolvidas. A autenticação mútua baseada em certificado refere-se a duas partes que se autenticam por meio da verificação do certificado digital fornecido para que ambas as partes tenham certeza da identidade da outra (Siriwardena, 2014). Esse mecanismo é frequentemente utilizado em aplicações e sistemas de segurança para garantir que apenas usuários autorizados tenham acesso a dados confidenciais. No processo, cada entidade possui um certificado digital, que é uma estrutura de dados eletrônicos contendo informações sobre a identidade da entidade, como o nome e uma chave pública de criptografia. Esses certificados são emitidos por uma autoridade de certificação confiável.

O *Broker Eclipse Mosquitto* é sugerido nesta implementação devido a este fornecer suporte TLS/SSL para conexões de rede criptografadas e autenticação de clientes. O processo de conexão e autenticação mútua envolve etapas bem definidas para garantir a segurança na troca de informações entre os dispositivos clientes (servidor de atualização e EDUs) e o *Broker* MQTT, conforme ilustrado no diagrama de sequência da Figura 4.4.

As etapas deste processo são resumidas a seguir. Vale ressaltar que a autenticação segura só será concedida mediante a confirmação da validade dos certificados digitais, por meio de uma Autoridade de Certificação, de todos envolvidos no processo de conexão.

1. Os clientes iniciam uma conexão com o *Broker* MQTT.
2. Os clientes enviam solicitações de conexão segura TLS/SSL ao *Broker* MQTT.
3. O *Broker* MQTT envia informações de seu certificado para os clientes, que verificam sua validade por meio de uma Autoridade de Certificação. Da mesma forma, cada cliente encaminha suas informações de certificado para validação no *Broker* MQTT, garantindo comunicação segura por meio de autenticação mútua.
4. Uma conexão segura é estabelecida entre os clientes e o *Broker* MQTT, permitindo que eles troquem informações de forma segura.

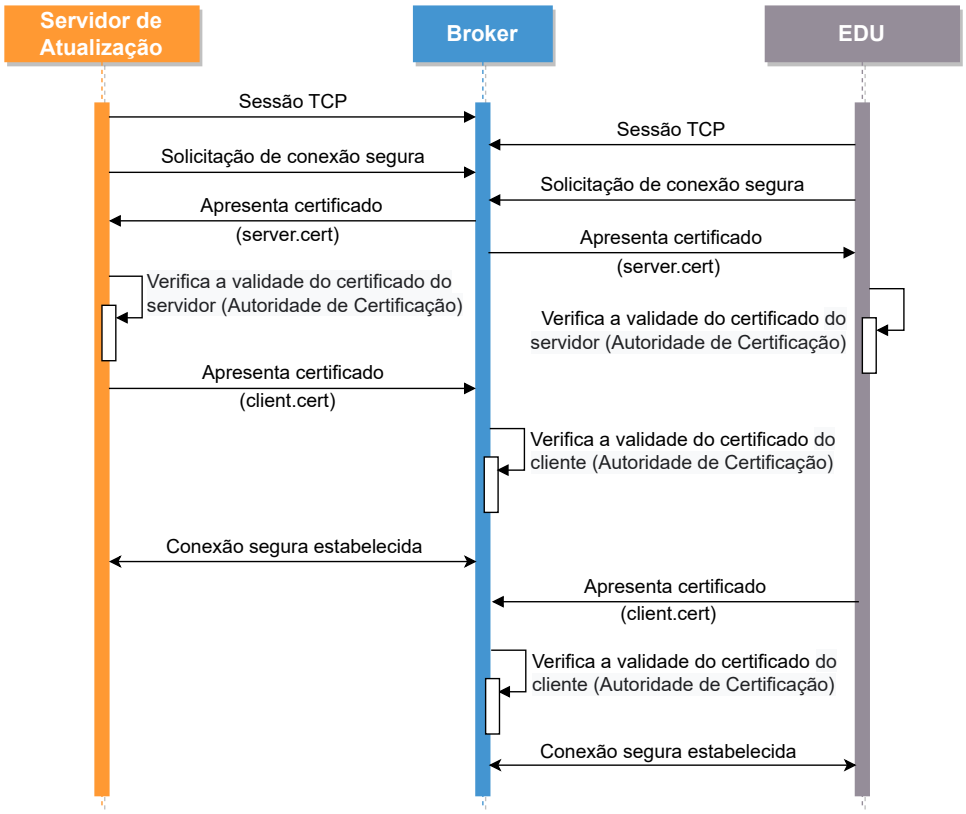


Figura 4.4: Diagrama do processo de estabelecimento de conexão e autenticação mútua entre o *Broker* MQTT e clientes servidor de atualização e EDU.

4.4 Publicação e Assinatura de Tópicos no *Broker* MQTT

O protocolo MQTT é baseado no princípio de publicação de mensagens e assinatura de tópicos, ou Arquitetura de Publicação/Assinatura, no qual vários clientes se conectam a um *Broker* e assinam tópicos de seu interesse. Os clientes também se conectam ao *Broker* e publicam mensagens em tópicos. Muitos clientes podem assinar os mesmos tópicos e fazer o que quiserem com as informações obtidas (Light, 2017; Banks et al., 2020).

Antes que um dispositivo possa publicar ou assinar um tópico, ele deve primeiro estabelecer uma conexão com o *Broker* MQTT, conforme Seção 4.3. Isso normalmente é feito usando a mensagem *CONNECT* do protocolo MQTT, que é enviada ao *Broker* MQTT junto com as credenciais do dispositivo e outras opções de conexão.

Uma vez que um dispositivo está conectado ao *Broker* MQTT, ele pode publicar mensagens em um tópico enviando uma mensagem *PUBLISH* para o *Broker* MQTT. Não há necessidade de configurar um tópico, basta publicar nele.

Os tópicos são tratados como uma hierarquia, usando uma barra (/) como separador. Isso permite a criação de um arranjo sensato de temas comuns, da mesma forma que um sistema de arquivos (Light, 2017). Neste trabalho, serão definidos os seguintes tópicos para publicações:

- ***OTA/Update*** - Inclui a carga útil (variável de nome *payload*), que é o conteúdo real da mensagem, neste caso, um arquivo JSON contendo a atualização ou reconfiguração da lógica de operação dos sensores das EDUs;
- ***OTA/Validator*** - Armazena um objeto do tipo *Secure Hash Algorithm 256 bits* (SHA-256) (variável de nome *hashValue*) calculado a partir do *payload* contido no tópico "*OTA/Update*". O SHA-256, é um dos algoritmos de *hash* criptográfico mais amplamente utilizados. Ele é projetado para produzir uma saída (*digest*) de 256 *bits* (32 *bytes*) a partir de dados de entrada de tamanho variável, tornando-o útil para várias aplicações de segurança, como verificação de integridade de dados, autenticação e criptografia. Desta forma, este procedimento visa garantir a integridade e autenticidade das mensagens de atualização e reconfiguração encaminhadas para as EDUs. Ao receber uma mensagem de atualização ou reconfiguração, a EDU deverá calcular o seu *hash* e comparar com o valor obtido no tópico "*OTA/Validator*", se os valores de *hash* coincidirem, a atualização é armazenada em um arquivo na memória interna do dispositivo microcontrolador da EDU.

Ao processar as mensagens contidas em tópicos de publicação, o *Broker* então realiza o encaminhamento destas para todos os dispositivos inscritos nos tópicos.

Para a assinatura de um tópico, o dispositivo deverá enviar uma mensagem do tipo *SUBSCRIBE* ao *Broker* identificando o tópico do qual deseja receber mensagens. A mensagem de assinatura inclui o nome do tópico e um ID exclusivo (conhecido como "ID do pacote") que é usado para identificar a assinatura. Uma vez que um dispositivo assinou um tópico, ele receberá todas as mensagens publicadas naquele tópico pelo *Broker*.

O protocolo MQTT também fornece a possibilidade do cancelamento de assinaturas caso o dispositivo não deseje mais receber mensagens relativas a um tópico específico, bastando enviar uma mensagem *UNSUBSCRIBE* ao *Broker*. A mensagem de cancelamento de assinatura deverá incluir o nome do tópico e o seu ID. Há também a possiblidade de desconexão do *Broker*, uma vez que um dispositivo termina de publicar ou se inscrever nos tópicos, ele pode se desconectar do *Broker* enviando uma mensagem *DISCONNECT*.

O diagrama de sequência da Figura 4.5 é usado para representar o processo de publicação de tópicos MQTT em um *Broker*, proposto nesta abordagem.

No exemplo, destacam-se o servidor de atualização como elemento publicante de mensagens nos tópicos "*OTA/Update*" e "*OTA/Validator*" e o *Broker* como elemento intermediário para fazer uma ponte de comunicação entre o servidor de atualização e as EDUs, se tornando responsável pelo recebimento, enfileiramento e envio das mensagens.

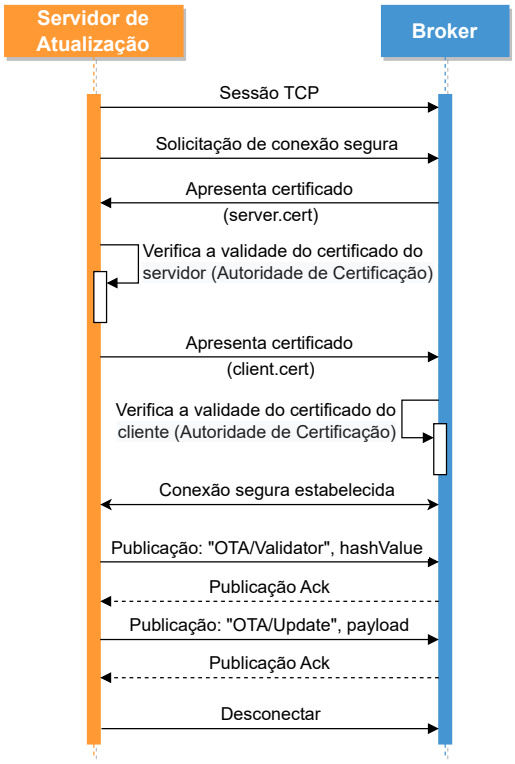


Figura 4.5: Diagrama do processo de publicação de mensagens em tópicos MQTT pelo servidor de atualização em um *Broker*.

As etapas deste processo são resumidas a seguir:

1. O servidor de atualização inicia uma conexão com o *Broker*.
2. O servidor de atualização envia uma solicitação de conexão segura TLS/SSL para o *Broker*.
3. O *Broker* envia suas informações de certificado para o servidor de atualização, que verifica sua validade por meio de uma Autoridade de Certificação. Da mesma forma, o servidor de atualização encaminha suas informações de certificado para validação no *Broker*, garantindo comunicação segura por meio de autenticação mútua.
4. O servidor de atualização publica o objeto do tipo *hash* (*hashValue*) calculado a partir do *payload* contido no arquivo JSON no tópico "*OTA/Validator*" e após esse procedimento, publica o *payload* da mensagem de

atualização ou reconfiguração da lógica de operação das EDUs através de um arquivo JSON no tópico "*OTA/Update*".

5. Após a publicação, o servidor de atualização solicita o término da conexão com o *Broker*.

O diagrama de sequência da Figura 4.6 representa o processo realizado pelas EDUs para assinatura em tópicos MQTT em um *Broker*, assim como as etapas que compreendem o processo de implementação do conteúdo das mensagens para a atualização ou reconfiguração da lógica de operação destes dispositivos. O processo de assinatura de tópicos é fundamental para garantir que as EDUs recebam informações relevantes e atualizações de forma confiável e segura, permitindo a integração eficaz das atualizações ou reconfigurações destinadas a operação dos dispositivos de detecção de emergências, garantindo que esses dispositivos funcionem de maneira consistente e segura em situações críticas.

Para esta abordagem, o processo de assinatura nos tópicos MQTT em um *Broker* é resumido da seguinte forma:

1. A EDU inicia uma conexão segura TLS/SSL com o *Broker* (etapas de negociação de parâmetros de segurança e apresentação mútua de certificados digitais apresentadas na Seção 4.3).
2. Uma conexão segura é estabelecida entre a EDU e o Broker MQTT, permitindo a troca de informações de forma segura.
3. A EDU realiza a assinatura dos tópicos MQTT "OTA/Validator" e "OTA/Update" através do *Broker*.
4. Na EDU, um *loop* é executado para verificar a chegada de mensagens através dos tópicos assinados.
5. Ao receber uma mensagem, a EDU verifica a sua integridade comparando o valor do objeto SHA-256 recebido através do tópico "OTA/Validator" com o valor do objeto SHA-256 calculado a partir do *payload* contido no tópico "OTA/Update". Este procedimento irá garantir a integridade das mensagens de atualização ou reconfiguração encaminhadas para as EDUs. Se estes valores coincidirem, o conteúdo da mensagem é armazenado em um arquivo na memória interna do dispositivo microcontrolador da EDU para ser implementado posteriormente.
6. O dispositivo é reiniciado para que este possa operar com base nos parâmetros de funcionamento mais atuais encaminhados pela última atualização ou reconfiguração válida.

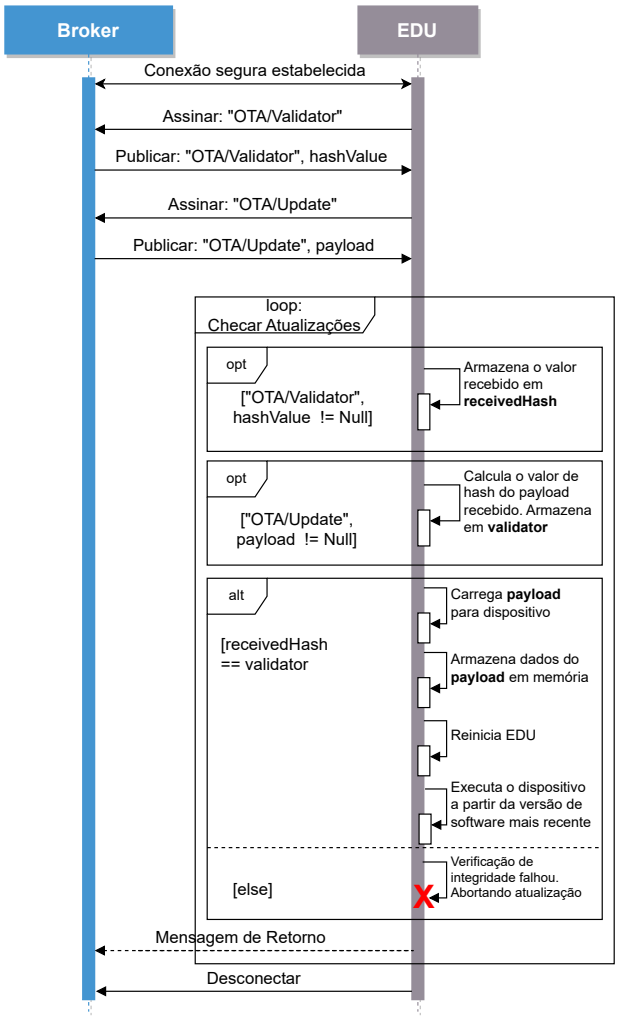


Figura 4.6: Diagrama do processo de assinatura de mensagens em tópicos MQTT por uma EDU em um *Broker* e atualização do contexto de operação da EDU.

7. Caso haja uma falta de correspondência entre os valores dos objetos SHA-256, é indicado uma falha na verificação da integridade do arquivo de atualização ou reconfiguração, e o processo é abortado.

4.5 Integração dos Processos

A modelagem final da arquitetura proposta neste trabalho é apresentada no diagrama de sequência da Figura 4.7. Esse diagrama oferece uma representação gráfica detalhada para a compreensão de todas as interações entre os componentes da arquitetura ao longo do tempo, detalhando procedimentos e métodos essenciais para o processo de reconfiguração segura das EDUs,

promovendo maior fidelidade ao processo de monitoramento de eventos críticos que afetam os centros urbanos. Esta modelagem detalhada não apenas fornece uma visão clara do funcionamento esperado da arquitetura, mas também serve como uma ferramenta valiosa para o seu desenvolvimento, testes e manutenções contínuas dos serviços.

A comunicação é estabelecida por meio do paradigma de publicação/assinatura com suporte a determinados níveis de segurança. Sugere-se a adoção do protocolo MQTT para essa finalidade, embora alternativas adicionais possam ser consideradas, contanto que estejam em conformidade com os requisitos fundamentais da arquitetura proposta. É importante destacar que, para a implementação das EDUs, há uma variedade de dispositivos microcontroladores compatíveis com *MicroPython* disponíveis. Esse modelo flexível e ajustável tem como objetivo não apenas fornecer orientações explícitas, mas também garantir adaptabilidade e desenvolvimento futuro, mitigando quaisquer restrições tecnológicas limitantes. Os detalhes relacionados à implementação serão abordados no Capítulo 5.

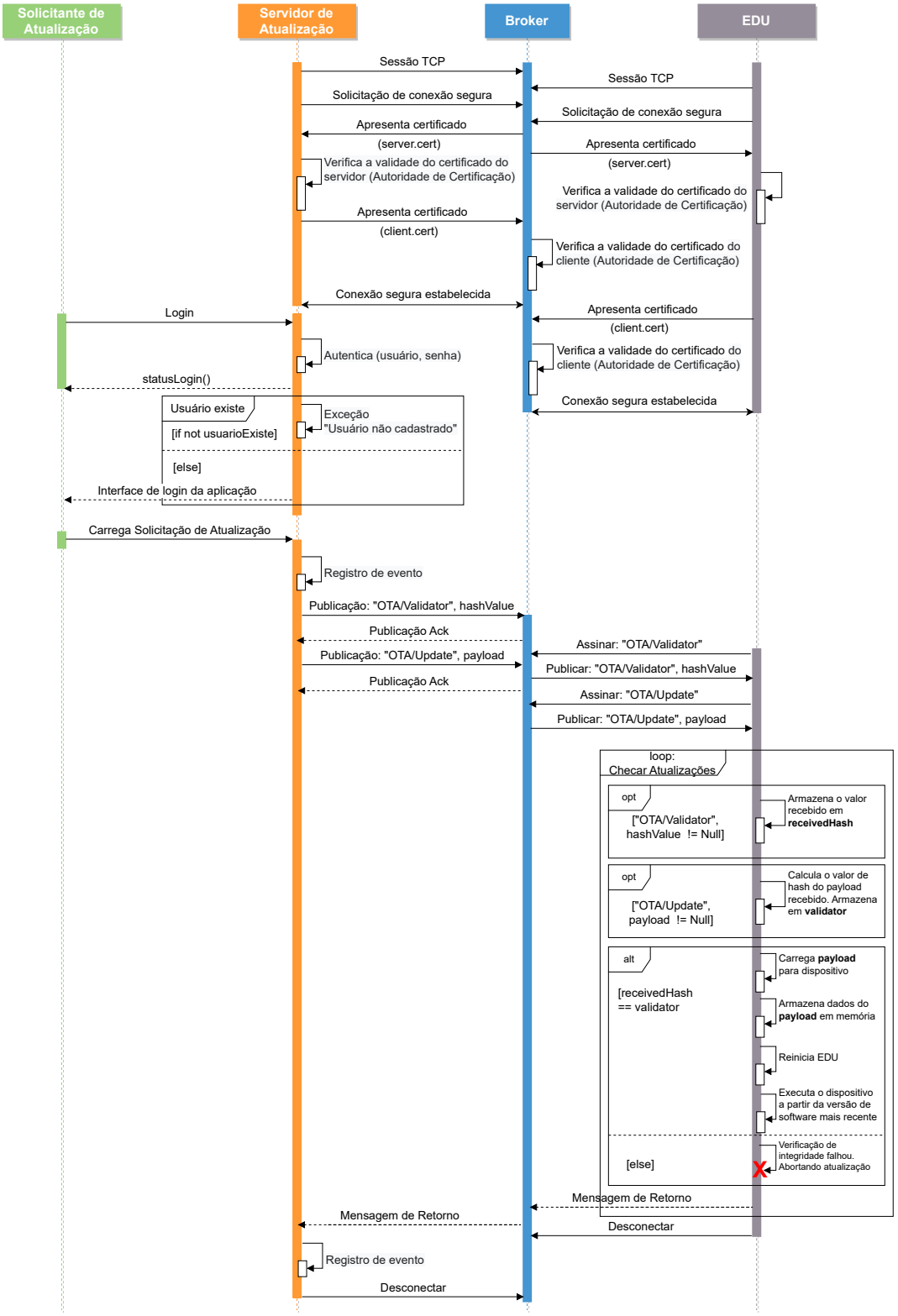


Figura 4.7: Diagrama completo da arquitetura proposta.

Capítulo 5

Implementação e Experimentos

Neste capítulo, serão apresentados detalhes acerca da implementação da arquitetura proposta nesta dissertação. A arquitetura foi concebida para permitir a reconfiguração *Over-the-Air* de Unidades de Detecção de Emergências (EDUs), e serão discutidos aspectos práticos relativos aos elementos fundamentais que possibilitam essa funcionalidade. Inicialmente, será abordada a dinâmica da solicitação de atualização, com a discussão dos procedimentos que viabilizam a reconfiguração das unidades em resposta a novos requisitos de operação. Em seguida, será explorado de maneira detalhada o papel desempenhado pelo servidor de atualização, delineando seu funcionamento e integração no contexto da arquitetura. Posteriormente, será examinada a configuração do Broker MQTT de acordo com a versão 5.0 do protocolo, assegurando uma comunicação eficiente entre as EDUs e o servidor de atualização. Ainda será destacado o uso do microcontrolador ESP32 na implementação prática da arquitetura. Por fim, será apresentada a implementação de uma EDU. Além da descrição detalhada da implementação, este capítulo abordará os experimentos projetados para validar a eficácia e robustez da arquitetura proposta, proporcionando percepções valiosas para a compreensão do desempenho do sistema em cenários reais.

5.1 Prova de Conceito

Diferentes tecnologias e ferramentas foram utilizadas para implementar as funcionalidades propostas neste trabalho, permitindo uma avaliação inicial com base em uma abordagem de prova de conceito. As informações aqui descritas são essenciais para permitir a replicação do experimento por outros pesquisadores e para garantir a validade e confiabilidade dos resultados apresentados.

Uma prova de conceito é um mecanismo usado para demonstrar a viabilidade e funcionalidade de uma ideia, conceito ou tecnologia em um ambiente prático e controlado. Neste trabalho, a abordagem de prova de conceito foi utilizada para a construção de um ambiente controlado e em menor escala com o objetivo de avaliar a viabilidade da arquitetura proposta, observando se este atende aos requisitos básicos aqui definidos. As implementações de código referentes às funcionalidades descritas na Seção 4 e utilizadas por esta prova de conceito podem ser encontradas no repositório acessível através da URL `https://github.com/daniel-gcosta/otaedu/`. Inclusive vale salientar que os trechos de códigos mostrados neste capítulo seguem aproximadamente a numeração de linhas dos códigos no referido repositório, a fim de auxiliar o leitor na eventual reprodução deste trabalho.

Para a realização da prova de conceito, foi estabelecido um cenário hipotético de monitoramento de temperatura ambiente visando à detecção de alertas de incêndio, onde uma EDU opera em uma área urbana específica. A Figura 5.1 apresenta a infraestrutura arquitetural utilizada neste cenário, destacando a integração de diversas tecnologias e ferramentas de *hardware* e *software*. Esses elementos, descritos a seguir, constituem o ambiente no qual o processo de reconfiguração da EDU será executado e são fundamentais para demonstrar com eficácia os processos definidos no Capítulo 4.

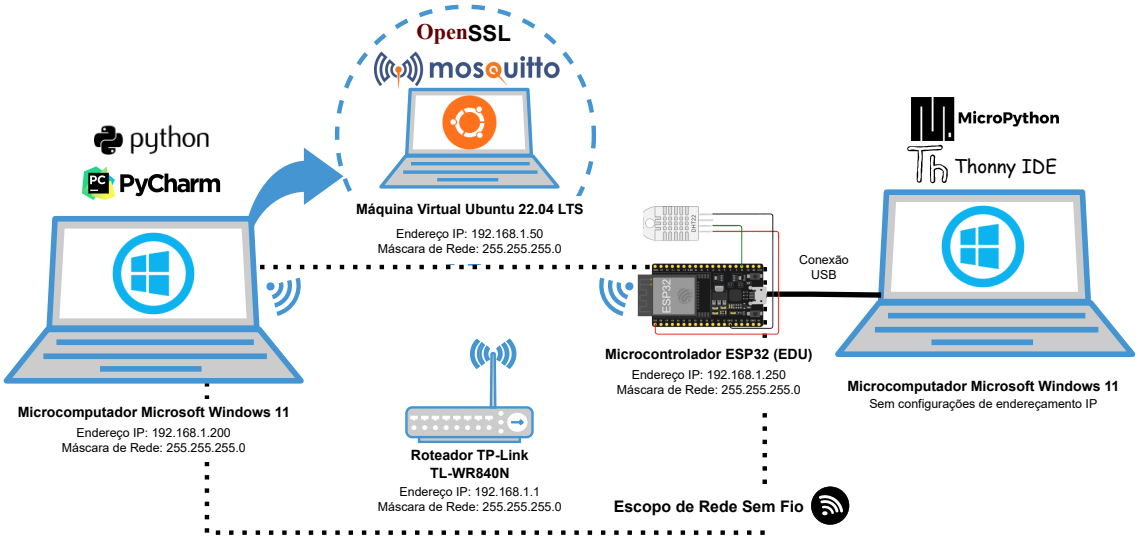


Figura 5.1: Infraestrutura projetada para a prova de conceito.

- Microcomputador com sistema operacional *Microsoft Windows* 11 como hospedeiro para a execução dos processos referentes à solicitação de atualização, ao solicitante de atualização e ao servidor de atualização. Foi utilizado o endereço IPv4 192.168.1.200 e máscara de sub-rede

255.255.255.0 para a identificação e promoção da conectividade deste dispositivo na infraestrutura de comunicação sem fio;

- Microcomputador com sistema operacional *Microsoft Windows* 11, sem configurações de endereçamento IP, servindo apenas para a implementação do algoritmo responsável pelas funcionalidades da EDU, bem como, a visualização de mensagens relativas ao processo de reconfiguração das unidades;
- Máquina Virtual com sistema Operacional Ubuntu 22.04 LTS como hospedeiro para a instalação e execução dos processos referentes ao *Eclipse Mosquitto Broker* MQTT e para geração de certificados digitais autoassinados através do *OpenSSL*. Foi utilizado o endereço IPv4 192.168.1.50 e máscara de sub-rede 255.255.255.0 para a identificação e promoção da conectividade desta máquina virtual na infraestrutura de comunicação sem fio (configuração da interface de rede em modo *bridge*);
- *Eclipse Mosquitto* como *Broker* de mensagens com suporte ao protocolo MQTT 5.0 instalado e configurado no microcomputador hospedeiro;
- Linguagem de programação *Python* 3.9 e ambiente de desenvolvimento *PyCharm* 2023.2.3 (*Community Edition*) para implementação das funcionalidades referentes à solicitação de atualização, ao solicitante de atualização e ao servidor de atualização;
- Microcontrolador ESP32 de baixo custo e baixo consumo de energia para o controle e gerenciamento dos processos referentes à EDU. Foi utilizado o endereço IPv4 192.168.1.250 e máscara de sub-rede 255.255.255.0 para a identificação e promoção da conectividade deste dispositivo na infraestrutura de comunicação sem fio;
- Linguagem de programação *MicroPython* v1.20.0, implementação baseada no *Python* 3, e ambiente de desenvolvimento *Thonny* 3.3.14 para a especificação em alto nível das funcionalidades que garantem a reconfiguração das EDUs de maneira segura.
- Roteador TP-Link TL-WR840N para a implementação de uma infraestrutura de comunicação sem fio, do tipo rede local (WLAN), capaz de permitir os processos de reconfiguração da EDU através da abordagem *Over-the-air*. A configuração IP para endereçamento da rede é definida pelo escopo 192.168.1.0 utilizando a máscara de sub-rede 255.255.255.0. Cabe ressaltar que a infraestrutura de comunicação sem fio utilizada para a prova de conceito não terá acesso à *internet*. Foi utilizado o endereço IPv4 192.168.1.1 e máscara de sub-rede 255.255.255.0 para a identificação e promoção da conectividade deste dispositivo aos demais presentes na infraestrutura de comunicação sem fio;

No cenário descrito, foi desenvolvida uma aplicação inicial para gerenciamento de emergências. Basicamente, se os valores de temperatura observados forem maior que um limiar previamente definido, serão encaminhadas mensagens de alerta para uma tomada de decisão por parte dos operadores do sistema. A EDU foi construída utilizando um microcontrolador do tipo ESP32 e um sensor de temperatura e umidade associado. O Código 5.1 apresenta o contexto atual de operação desta EDU.

Código Fonte 5.1: Representação em alto nível, através de linguagem de programação *Micropython*, da lógica de operação atual da EDU para o monitoramento de temperatura ambiente e geração de alertas de incêndio.

```
1  import utime
2  # Simulando valores de temperatura
3  temp_ambiente = 30 # Valor inicial da temperatura ambiente
4
5  num_iteracoes = 100
6  print("LÓGICA DE OPERAÇÃO DE EDU VERSÃO 1 - DESATUALIZADA")
7  for _ in range(num_iteracoes):
8      if temp_ambiente > 70:
9          print("Temperatura extremamente elevada, risco de incêndio! Valor atual:
           ↪ {}°C".format(temp_ambiente))
10     temp_ambiente += 1
11     utime.sleep(3)
```

Ao analisar o Código 5.1, percebe-se a definição, meramente ilustrativa, de temperatura ambiente à 30ºC (linha 3) e de um limiar configurado para 70°C (linha 8) no qual, medições acima deste valor irão disparar mensagens de alerta de risco de incêndio. Note-se que, neste caso, não está sendo dada nenhuma aplicação ao sensoriamento de valores de umidade, suportados pelo dispositivo apresentado anteriormente. Isso resulta em uma aplicação de monitoramento não condizente com o contexto monitorado, uma vez que é pouco confiável determinar a ocorrência de incêndios baseando-se apenas nos valores de temperatura. Na prática o que ocorre é a combinação desta informação com outras variáveis, como exemplo, a própria umidade, que pode influenciar as condições ideais para uma combustão. Esta questão reafirma a necessidade de reconfigurar a operação de sensoriamento da EDU.

A aplicação representada através do Código 5.1 é extremamente simples, pois a sua finalidade é demonstrar que a arquitetura proposta é capaz de permitir a reconfiguração do contexto de operação de uma EDU, em tempo de execução e de forma segura, através de solicitações remotas por meio de uma infraestrutura de comunicação sem fio.

5.2 Solicitação de atualização e solicitante de atualização

Uma solicitação de atualização consiste em um arquivo produzido em uma linguagem de programação voltada para os microcontroladores das EDUs, que contém parâmetros operacionais específicos destinados a reconfigurar as EDUs. Esse arquivo é produzido por um solicitante de atualização, que é o ator responsável por submeter ações de reconfiguração, encaminhando códigos executáveis, dados, recursos e/ou parâmetros de reconfiguração para as EDUs existentes em uma área de monitoramento.

Considerando as limitações observadas no método de monitoramento empregado na lógica de operação da EDU, conforme apresentado no Código 5.1, o solicitante de atualização pode desenvolver uma nova implementação desse algoritmo, tornando-o mais eficiente para o contexto monitorado. Um exemplo dessa nova abordagem de monitoramento desenvolvida pelo solicitante de atualização pode ser observado no Código 5.2.

Código Fonte 5.2: Representação em alto nível, através de linguagem de programação *Micropython*, da reconfiguração da lógica de operação para o monitoramento de temperatura ambiente, umidade relativa do ar e geração de alertas de incêndio a ser realizada pela EDU.

```
1  import utime
2  # Simulando valores de temperatura e umidade
3  temp_ambiente = 30 # Temperatura inicial
4  umidade_ambiente = 70 # Umidade inicial
5
6  num_iteracoes = 100
7  print("LÓGICA DE OPERAÇÃO DE EDU VERSÃO 2 - ATUALIZADA")
8  for _ in range(num_iteracoes):
9      # Verifique o valor da temperatura ambiente
10     if temp_ambiente > 70:
11         print ("Alerta: Temperatura elevada! Valor atual: {}°C".format(temp_ambiente))
12     if umidade_ambiente < 20:
13         print("Alerta: Baixa umidade! Valor atual: {}%".format(umidade_ambiente))
14     if temp_ambiente > 90 and umidade_ambiente < 15:
15         print("Alerta de Incêndio! Temperatura: {}°C, Umidade: {}%".format(temp_ambiente,
           ↪  umidade_ambiente))
16     temp_ambiente += 1
17     umidade_ambiente -= 1
18     utime.sleep(1)
```

Nesta nova abordagem, percebe-se a definição meramente ilustrativa, de temperatura ambiente à 30ºC (linha 3) e de umidade relativa do ambiente em 30% (linha 4). Entretanto, observa-se entre as linhas 8 e 18 a definição de uma lógica de operação mais aprimorada no que se refere a observação de diferentes faixas de temperaturas e condições de umidade associadas, que geram diferentes mensagens de alerta dependendo dos valores sensoriados. Novamente, o foco da discussão aqui não é a implementação em si. Esse é

apenas um exemplo hipotético de reconfiguração de uma EDU, visando a compreensão das atividades relativas à solicitação de atualização e solicitante de atualização. Reiterando, o escopo da presente análise não se destina à consideração da implementação prática em si. O exemplo de código fornecido representa meramente uma instância hipotética de reconfiguração de uma EDU, almejando facilitar a compreensão das atividades concernentes à solicitação de atualização e solicitante de atualização.

Essa nova proposição traz consigo uma melhoria em relação ao sensoriamento do ambiente, tornando o processo mais fidedigno à realidade observada. Nesse sentido, surge a necessidade de implementação dessa lógica de operação na EDU, o que caracteriza uma solicitação de atualização. Essa solicitação é expressa em um arquivo JSON que encapsula todo o código executável do novo contexto de operação dos sensores presentes na EDU e será encaminhado ao servidor de atualização por meio de uma interface de comunicação. O solicitante de atualização deve ter acesso controlado, o que requer a apresentação de mecanismos de autenticação.

O encapsulamento do código executável em um arquivo JSON poderá ser feito através de uma implementação de aplicação de código simples, no qual será definido o objeto JSON, delimitado por chaves e contém pares de chave-valor separados por vírgulas; uma chave: que consiste em uma *string* que representa o nome da propriedade; e um valor: podendo ser um número, *string*, booleano, *array*, objeto ou *null* que será atribuído à chave. Neste caso, o valor da chave será todo o código executável do novo contexto de operação das EDUs, conforme mostra o Código 5.3. Foi utilizado o diretório "D:/eduversions" no microcomputador hospedeiro para o armazenamento do arquivo, conforme descrito na linha 28 do código supracitado. A determinação do diretório destinado ao armazenamento das solicitações de atualização é deixada à discricionariedade. Recomenda-se a consideração de aspectos inerentes à segurança e ao controle desses artefatos, assegurando a preservação dos princípios de integridade, confidencialidade e autenticidade.

Código Fonte 5.3: Representação em alto nível, através de linguagem de programação *Python*, do processo de encapsulamento da nova abordagem de operação da EDU para o formato JSON.

```
1  import json
2
3  code = '''import utime
4  # Simulando valores de temperatura e umidade
5  temp_ambiente = 30  # Temperatura inicial
6  umidade_ambiente = 70  # Umidade inicial
7
8  num_iteracoes = 100
9  print ("LÓGICA DE OPERAÇÃO DE EDU VERSÃO 2 - ATUALIZADA")
10 for _ in range(num_iteracoes):
11     # Verifique o valor da temperatura ambiente
```

```
    if temp_ambiente > 70:
        print("Alerta: Temperatura elevada! Valor atual: {}°C".format(temp_ambiente))
    if umidade_ambiente < 20:
        print("Alerta: Baixa umidade! Valor atual: {}%".format(umidade_ambiente))
    if temp_ambiente > 90 and umidade_ambiente < 15:
        print("Alerta de Incêndio! Temperatura: {}°C, Umidade: {}%".format(temp_ambiente,
↪  umidade_ambiente))
    temp_ambiente += 1
    umidade_ambiente -= 1
    utime.sleep(1)'''
# Creating dictionary called "data". It will be used later to store the provided code in
↪  JSON format.
data = {
    "code": code
}
# Directory path to store the JSON file
file_path = "D:/edu-versions/swap_v2.json"

# Convert the object to a JSON representation
json_data = json.dumps(data, indent=4)  # indent=4 é opcional, apenas para formatar o JSON
↪  com indentação

# Write JSON to a file
with open(file_path, "w") as file:
    file.write(json_data)

print("JSON file created successfully!")
```

O resultado esperado é um arquivo JSON, apresentado em sua sintaxe formatada, contendo todo o código executável referente à nova abordagem de operação da EDU, conforme apresenta o Código 5.4.

Código Fonte 5.4: Arquivo em formato JSON contendo a reconfiguração da lógica de operação da EDU.

```
{
    "code": "import utime\n# Simulando valores de temperatura e umidade\ntemp_ambiente = 30 #
↪  Temperatura inicial\numidade_ambiente = 70 # Umidade inicial\n\nnum_iteracoes =
↪  100\nprint(\"L\u00d3GICA DE OPERA\u00c7\u00c3O DE EDU VERS\u00c3O 2 - ATUALIZADA\")\nfor _
↪  in range(num_iteracoes):\n    # Verifique o valor da temperatura ambiente\n    if
↪  temp_ambiente > 70:\n        print (\"Alerta: Temperatura elevada! Valor atual:
↪  {}\u00b0C\".format(temp_ambiente))\n    if umidade_ambiente < 20:\n        print(\"Alerta:
↪  Baixa umidade! Valor atual: {}%\".format(umidade_ambiente))\n    if temp_ambiente > 90 and
↪  umidade_ambiente < 15:\n        print(\"Alerta de Inc\u00eandio! Temperatura: {}\u00b0C,
↪  Umidade: {}%\".format(temp_ambiente, umidade_ambiente))\n    temp_ambiente += 1\n
↪  umidade_ambiente -= 1\n    utime.sleep(1)"
}
```

Este arquivo será enviado ao servidor de atualização por meio de uma interface de comunicação que pode ser personalizada de acordo com os requisitos específicos de cada implementação da arquitetura. Na prova de conceito, por exemplo, um diretório específico do sistema operacional do hospedeiro foi utilizado para que o arquivo pudesse ser processado pela aplicação do servidor de atualização.

Os próximos passos detalham as metodologias e procedimentos utilizados para o envio seguro desta reconfiguração da lógica de operação à EDU, levando em

consideração os aspectos relacionados à infraestrutura de comunicação utilizada, bem como as ameaças e riscos à segurança inerentes a essa infraestrutura.

5.3 Servidor de atualização

A implementação do servidor de atualização deve ser feita seguindo algumas boas práticas e diretrizes para garantir robustez e segurança na execução dos seus processos. O Algoritmo 1 a seguir, detalha aspectos importantes para essa implementação. Em linhas gerais, o Algoritmo 1 descreve procedimentos formais para estabelecer conexão e autenticação mútua entre o servidor de atualização e o *Broker* MQTT, utilizando certificados digitais para criar um canal seguro. Ao realizar o carregamento de um arquivo em formato JSON referente à solicitação de atualização por meio de um usuário com permissões administrativas para tal ação (solicitante de atualização), este algoritmo calcula o seu valor de *hash* e realiza a publicação tanto do arquivo JSON quanto do seu *hash* associado através de tópicos MQTT ao *Broker*. Adicionalmente, implementa mecanismos para registrar as atividades de publicação de solicitações de atualização em um arquivo de registro de eventos.

Uma vez definida a sequência de passos lógicos para o funcionamento esperado da EDU, por meio do Algoritmo 1, serão formalmente apresentados no decorrer desta Seção, as representações em alto nível, através da linguagem de programação *Python*, de todos os procedimentos mencionados, proporcionando uma compreensão abrangente das funcionalidades e operações fundamentais delineadas no referido algoritmo.

Inicialmente é necessário especificar na aplicação relativa ao servidor de atualização, informações referentes à conexão com o *Broker* MQTT, como endereço IP e informações de porta de comunicação, bem como, informações referentes à localização dos certificados digitais que serão utilizados no procedimento de autenticação mútua entre o servidor de atualização e o *Broker* MQTT.

As informações de certificados deverão ser mantidas em um diretório, sendo necessário especificar a localização destes certificados à aplicação, facilitando assim procedimentos de alteração e atualização de certificados do componente. Para uma segurança eficaz, é importante manter estes certificados e credenciais em um local seguro e protegido. O propósito dos certificados é de assegurar níveis aceitáveis de proteção e segurança no âmbito dos processos de comunicação entre os dispositivos. Uma vez que há diversas vulnerabilidades inerentes à infraestruturas de comunicação sem fio, torna-se indispensável a implementação de esquemas de autenticação mais eficazes, fundamenta-

Algorithm 1: Algoritmo servidor de atualização.

```
Set MQTT Broker connection information by specifying certificates path
connBroker ← MQTTClientValues
# Set user logon settings
userLogon ← ApplicationLogonValues
connBroker.connect()
Function UploadUpdate(filePath)
    fileContent.read(filePath)
    # Convert the content data update to string
    update.toString(json(fileContent))
    Function CalculateHash(update)
        return hash ← hashlib.sha256(update)
    Function LogActivity(userLogon, update)
        logFile
        logFile.write(timestamp, userLogon, update)
    # Record update request in log file
    LogActivity (userLogon, update)
    Publish Update and Validator to Broker
# Provide the file path manually
UploadUpdate (filePathinput(path))
# Events loop MQTT
connBroker.loop()
```

dos em procedimentos mútuos, utilizando o protocolos de segurança TLS/SSL, no qual ambas as entidades comunicantes procedem à sua autenticação na rede. Este procedimento se efetua mediante a apresentação de certificados digitais, que consistem em arquivos eletrônicos intrinsecamente ligados a pares de chaves criptográficas, conferindo autenticidade à identidade de cada origem, por parte dos dispositivos interconectados.

Os procedimentos supracitados podem ser visualizados através do trecho de Código 5.5.

Código Fonte 5.5: Representação em alto nível, através de linguagem de programação *Python*, dos processos de conexão e autenticação mútua entre o servidor de atualização e o *Broker* MQTT descritos no Algoritmo 1.

```python
import paho.mqtt.client as mqtt
import time
import hashlib
import json
```

```
5  import ssl
6
7  # Set MQTT Broker connection information
8  MQTT_BROKER = "192.168.1.50"  # Set MQTT Broker IP
9  MQTT_PORT = 8883  # Using 8883 port for TLS/SSL
10 MQTT_CLIENT_ID = "1"  # MQTT client identification
11 KEEPALIVE = 15  # Interval in seconds to maintain active connection via control messages
12
13 # User authentication and registration settings in the Update Server application
14 UPDATE_REQUESTER = "administrador"
15 UPDATE_REQUESTER_PASS = "12345"
16
17 # Certificates path Client Server Update
18 CA_CERT = "mqtt_ca.pem" # Certificate authority certificate
19 CLIENT_CERT = "mqtt_updServer.pem" # Client (device) certificate
20 CLIENT_KEY = "mqtt_updServerKey.pem" # Client (device) private key
21
22 # Configuring MQTT connection with TLS
23 mqtt_client = mqtt.Client(client_id=MQTT_CLIENT_ID)
24 mqtt_client.tls_set(ca_certs=CA_CERT, certfile=CLIENT_CERT, keyfile=CLIENT_KEY,
   ↪  cert_reqs=ssl.CERT_REQUIRED, tls_version=ssl.PROTOCOL_TLSv1_2)
25 # Disable server IP address checking (NOT recommended for production)
26 mqtt_client.tls_insecure_set(True)
27
28 # Connect to Broker MQTT
29 mqtt_client.connect(MQTT_BROKER, MQTT_PORT, KEEPALIVE)
30 print("MQTT connection established successfully.")
```

Os parâmetros apresentados entre as linhas 8 a 11 são relativos a conexão do servidor de atualização ao *Broker* MQTT. Os parâmetros de usuário definidos entre as linhas 14 e 15 dizem respeito a definição de perfis de acesso à aplicação do servidor de atualização, mais especificamente, aos usuários que poderão encaminhar solicitações de atualização ao servidor de atualização, sendo que estes terão suas ações registradas em um arquivo de eventos. As informações contidas entre as linhas 18 a 20 fazem referência aos certificados necessários para estabelecer uma conexão segura (TLS/SSL) entre o servidor de atualização e o *Broker* MQTT ao qual pretende-se conectar. A seguir, são apresentados os detalhes das variáveis "*CA_CERT*", "*CLIENT_CERT*" e "*CLIENT_KEY*" e o tipo de conteúdo que cada uma delas armazena conforme a definição da arquitetura.

- ***CA_CERT*** - Contém o caminho para o arquivo de certificado da Autoridade Certificadora (CA). A CA é uma entidade confiável que emite certificados para autenticar servidores e clientes em uma conexão segura. Este arquivo contém as informações do certificado da CA que o cliente utilizará para verificar a autenticidade do servidor *Broker* MQTT. Ou seja, ele verifica se o servidor é realmente quem ele diz ser.
- ***CLIENT_CERT*** - Contém o caminho para o arquivo de certificado do cliente. Este certificado é usado para autenticar o próprio dispositivo cliente perante o servidor MQTT. Este arquivo contém as informações do certificado do cliente, que é emitido pela mesma autoridade certificadora

(CA) ou por uma autoridade confiável.

- ***CLIENT_KEY*** - Contém o caminho para o arquivo da chave privada do cliente. A chave privada é usada em conjunto com o certificado do cliente para estabelecer a autenticação mútua, onde tanto o cliente quanto o servidor se autenticam. Este arquivo de chave privada corresponde ao certificado do cliente e é mantida em segredo pelo próprio cliente.

Estes certificados e arquivos de chaves são essenciais para garantir a segurança da comunicação entre o cliente MQTT e o servidor MQTT, evitando a interceptação não autorizada e garantindo a autenticidade de ambas as partes. Entretanto, os certificados produzidos nesta prova de conceito foram autoassinados, ou seja, foram gerados e assinados por uma única entidade, geralmente o próprio proprietário do certificado, em vez de serem emitidos por uma autoridade de certificação (AC) confiável. Esses certificados são úteis em ambientes de desenvolvimento, teste ou em redes internas, mas não são recomendados para uso em cenários de produção ou em redes públicas.

Os parâmetros descritos nas linhas 23 a 24 do trecho de Código 5.5 configuram o servidor de atualização para estabelecer uma conexão segura com o *Broker* MQTT usando o protocolo TLS/SSL, autenticam o cliente e o servidor e especificam as credenciais de autenticação. Os seguintes argumentos são necessário para garantir que a conexão MQTT seja segura e autenticada:

- ***cert_reqs=ssl.CERT_REQUIRED*** - Define que a verificação do certificado do *Broker* MQTT é obrigatória, devendo este fornecer um certificado válido.
- ***tls_version=ssl.PROTOCOL_TLS*** - Especifica a versão do protocolo TLS/SSL a ser usada.

É importante ressaltar alguns detalhes acerca do parâmetro *“mqtt_client.tls_insecure_set(True)”*, apresentado na linha 26 do trecho de Código 5.5. A prova de conceito aqui definida foi construída sob uma infraestrutura de rede local sem fio, sem conexão à internet, permitindo a conectividade dos dispositivos apenas por endereços IPv4. Certificados TLS/SSL são normalmente emitidos para um nome de domínio registrado, como "mqtt.example.com", que está associado a um endereço IP público, na maioria das vezes, e não para um endereço IPv4. Ao utilizar endereços IPv4 para a emissão de certificados digitais é provável a existência de falhas na verificação destes certificados pelas partes que desejam autenticar-se.

Em virtude da não proposição de um serviço de resolução de nomes de domínio (DNS) para esta prova de conceito, o parâmetro *“mqtt_client.tls_insecure_set(True)”* foi utilizado para desativar temporariamente a verificação do endereço IP do servidor, permitindo os testes de au-

tenticação mútua sem a verificação de IP. Esta abordagem é útil para testes locais, mas não deve ser usada em cenários de produção.

Definido as especificações dos parâmetros necessários ao processo de conexão e autenticação, é preciso implementar mecanismos capazes de receber como entrada, um arquivo ou o caminho do arquivo de solicitação de atualização para que este possa ser processado e encaminhado através de tópicos de publicação. É importante verificar a existência do arquivo e se este pode ser lido, além de verificar se o formato é adequado para a transmissão. Para esta prova de conceito, o processo de encaminhamento da solicitação de atualização ao servidor de atualização será feito manualmente, informando à aplicação em qual diretório do microcomputador hospedeiro se encontra o arquivo. Uma vez realizado o processo de carregamento manual do arquivo, a próxima etapa a ser realizada pelo servidor de atualização, antes do seu envio ao *Broker*, diz respeito ao emprego de técnicas para garantir a integridade das informações.

A implementação de mecanismos para a verificação da integridade das atualizações é de extrema relevância. No servidor de atualização foi implementado uma função responsável por calcular o valor do objeto SHA-256 dos arquivos de atualizações. O valor obtido é encaminhado à EDU através do tópico de publicação "*OTA/Validator*", que irá utilizar como valor de referência de modo a verificar se o arquivo recebido pelo tópico "*OTA/Update*" possui o mesmo valor de objeto SHA-256 informado pelo servidor de atualização. Se os valores coincidirem, isso indica que o arquivo contendo a nova versão de operação da EDU está íntegro. O trecho do Código 5.6 ilustra os processos de carregamento do arquivo de atualização para o servidor de atualização (linhas 33 a 37), bem como o procedimento de cálculo do objeto SHA-256 aplicado ao arquivo JSON (linhas 44 a 47) que será utilizado pela EDU na verificação da integridade do arquivo de atualização recebido.

Código Fonte 5.6: Representação em alto nível, através de linguagem de programação *Python*, dos mecanismos de carregamento de arquivos de atualização e cálculo de valor dos objetos SHA-256 descritos no Algoritmo 1.

```
33  def upload_update(filepath):
34      # Read the firmware file
35      with open(filepath, "r") as file:
36          # Read the content of the file
37          file_content = file.read()
38
39      # Convert the data to a JSON object
40      softwareUpdate = json.loads(file_content)
41      # Convert the JSON object to a JSON string
42      update = json.dumps(softwareUpdate)
43
44      def calculate_hash(data):
45          hash_obj = hashlib.sha256()
46          hash_obj.update(data)
47          return hash_obj.digest()
```

De modo a garantir o não-repúdio das ações de atualização e ou reconfiguração das EDUs, o servidor de atualização implementa mecanismos para a construção de registros eficazes de atividades. Este procedimento foi pode ser visualizado no trecho do Código 5.7 e utiliza um arquivo de texto, armazenado em um diretório do computador hospedeiro, como forma de armazenamento dos eventos. Uma solução mais avançada poderá ser implementada, dependendo dos requisitos do sistema proposto. Informações relevantes, como o usuário gerador da ação e as informações referentes à atualização e ou reconfiguração disparadas às unidades são associadas a uma marca temporal que registra a hora e data que o evento ocorreu. Este procedimento busca a criação de trilhas de auditoria que tem por objetivo, auxiliar na gestão e no monitoramento das atividades, detectando quaisquer eventos e ações que uma pessoa ou sistema realizou ao manipular os dados.

Código Fonte 5.7: Representação em alto nível, através de linguagem de programação *Python*, do mecanismo para o registro de eventos de envio de atualizações e ou reconfigurações para as EDUs descrito no Algoritmo 1.

```
49      def log_activity(user, versionUpdate):
50          current_time = time.time()
51          log_file_path = "activity_log.txt"
52          with open(log_file_path, "a") as log_file:
53              log_file.write(f"{current_time}:{user} published EDU software
                ↪  update-{versionUpdate}\n")
```

Após a implementação das funções anteriormente mencionadas, torna-se necessário realizar a publicação desses artefatos para o *Broker*. Recomenda-se a adoção de mecanismos de confirmação a fim de assegurar que as atualizações sejam recebidas e validadas pelos dispositivos clientes. Alternativas viáveis incluem a utilização de tópicos de notificação, a subscrição por parte do servidor de atualização, ou mecanismos análogos com o propósito de atingir esta finalidade

O funcionamento esperado do servidor de atualização é apresentado na Figura 5.2 que apresenta a saída do terminal do ambiente de desenvolvimento *PyCharm*. Este resultado informa que, tanto a atualização da lógica de operação da EDU, quanto o valor do objeto SHA-256 associado ao conteúdo da atualização, foram encaminhados respectivamente através dos tópicos de publicação “*OTA/Update*” e “*OTA/Validator*"” ao *Broker* MQTT.

A Figura 5.3 apresenta o terminal do *Broker Eclipse Mosquitto*, no qual é possível verificar a confirmação do recebimento das publicações efetuadas pelo servidor de atualização.

Os registros referentes aos eventos de publicação de atualizações ao *Broker* são passíveis de análise mediante consulta ao arquivo “*activity_log.txt*”, apresentada através da Figura 5.4. Este arquivo encontra-se armazenado em

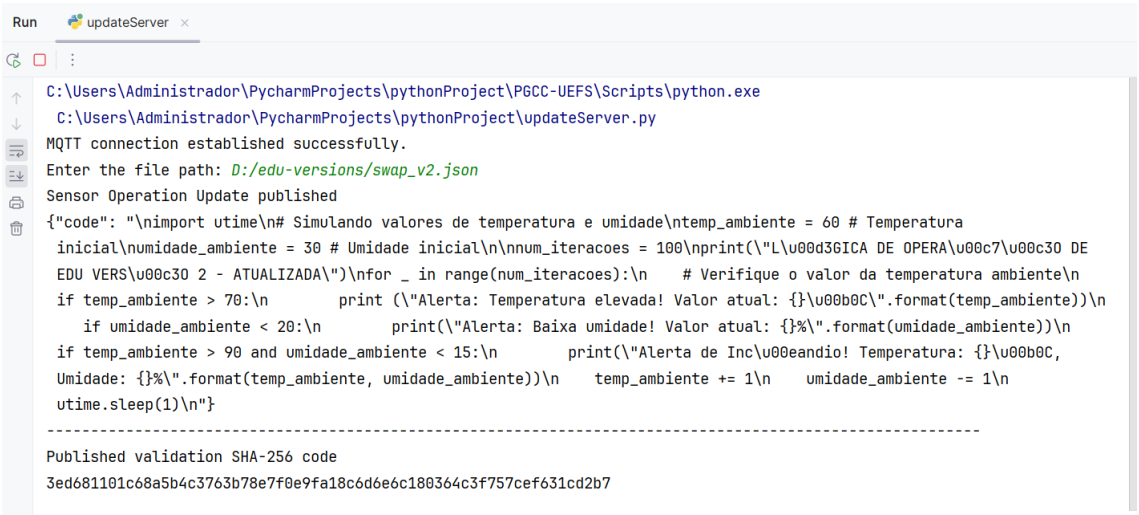


Figura 5.2: Publicação ao *Broker* das mensagens referentes ao novo contexto de operação da EDU, no formato JSON, e valor de objeto SHA-256 associado.

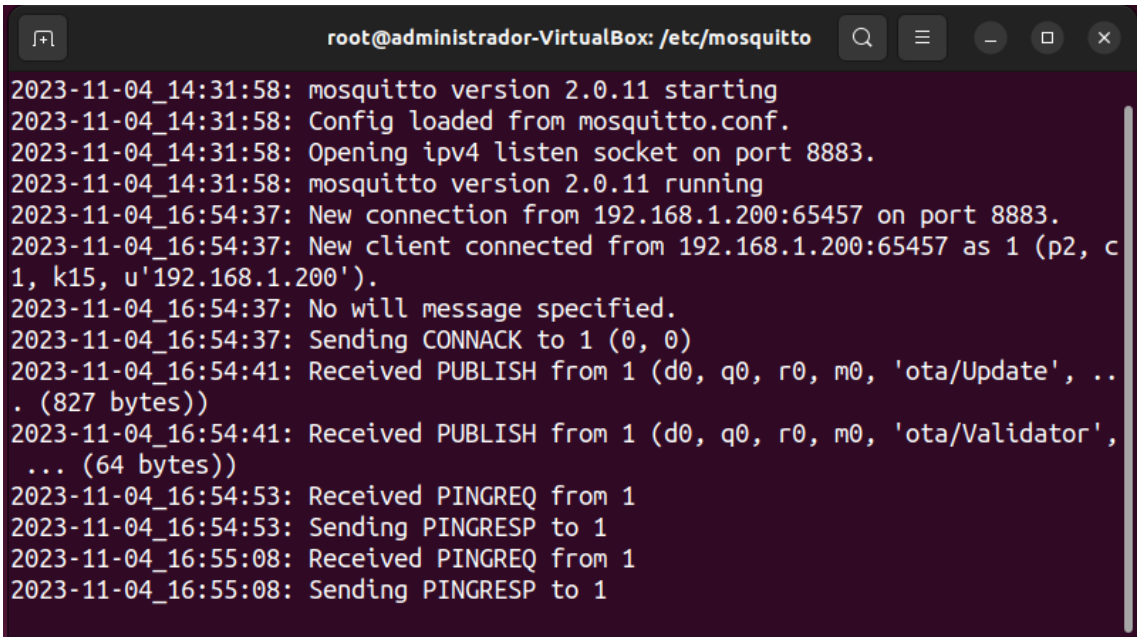


Figura 5.3: Detalhamento de informações verbais exibidas durante o processamento de publicações, por parte do *Broker* MQTT, advindas do servidor de atualização.

um diretório específico do computador hospedeiro do servidor de atualização, conforme definido nesta prova de conceito.

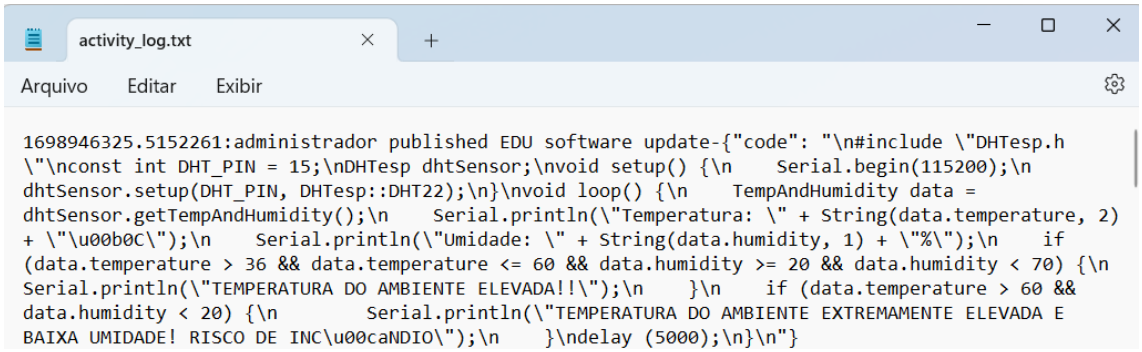


Figura 5.4: Arquivo contendo o registro dos eventos relativos à publicação de mensagens ao *Broker* MQTT por parte do servidor de atualização.

5.4 *Broker* MQTT e Protocolo MQTT 5.0

Para o controle, gerenciamento, armazenamento e distribuição das mensagens trocadas entre o servidor de atualização e as EDUs, é necessário a implementação de um corretor de mensagens (*broker*). Um *Broker* é um servidor intermediário que facilita a troca de mensagens em um sistema, atuando como um *hub* para dispositivos conectados. Eles atuam como intermediários entre outros aplicativos, permitindo que os remetentes emitam mensagens sem saber onde estão os destinatários, se eles estão ativos ou não ou quantos deles existem, facilitando o desacoplamento de processos e serviços dentro de sistemas (IBM, 2023). Ainda de acordo com IBM (2023), estes sistemas podem ser de dois tipos:

- Sistemas de mensagens ponto a ponto: Padrão de distribuição que estabelece uma correspondência individual entre remetente e destinatário, assegurando que cada mensagem seja enviada e utilizada apenas uma vez. É apropriado para cenários em que a ação deve ser executada de forma única, como em folhas de pagamento e transações financeiras, onde é crucial garantir que os pagamentos sejam processados com precisão e sem duplicações.
- Sistemas de mensagens de publicação/assinatura: Operam em um modelo de transmissão, onde os produtores publicam mensagens em tópicos e vários consumidores se inscrevem nesses tópicos para receber as mensagens. Essa abordagem permite a distribuição de mensagens para vários aplicativos inscritos, estabelecendo um relacionamento de um para muitos entre o produtor e os consumidores. É útil quando a disseminação de informações para um público amplo é necessária.

Dado o exposto e, levando em consideração o contexto de aplicação deste trabalho, fica evidente a implementação de um *Broker* de mensagens em seu

modelo Publicação/Assinatura.

Uma vez que o modelo de funcionamento do *Broker* é definido como Publicação/Assinatura, há a necessidade também do uso de protocolos de comunicação capazes de operar neste modelo de arquitetura, garantindo o encaminhamento adequado das mensagens ao dispositivos da infraestrutura. O protocolo MQTT é um protocolo de comunicação leve e eficiente projetado para sistemas de mensagens em redes com recursos limitados, como a IoT. Ele é baseado no modelo de Publicação/Assinatura, sendo conhecido por sua simplicidade e baixo consumo de largura de banda, o que o torna ideal para dispositivos com recursos limitados, como sensores e microcontroladores. Ele também oferece mecanismos de Qualidade de Serviço (QoS) para garantir a entrega confiável de mensagens, tornando-o adequado para cenários em que a confiabilidade da comunicação é crucial. O MQTT é amplamente utilizado em aplicações de IoT, automação residencial, monitoramento remoto e em qualquer contexto onde a troca de informações eficiente e confiável entre dispositivos conectados é necessária.

Para esta prova de conceito, foi utilizado o *Eclipse Mosquitto* como *Broker* de mensagens instalado e configurado em uma máquina virtual com Ubuntu 22.04 LTS. Os procedimentos para a instalação do *Eclipse Mosquitto* podem ser obtidos através de sua página oficial na internet em Mosquitto (2018).

Esta implementação de *Broker* possui suporte ao MQTT em sua versão 5.0 e, de acordo com Standard (2019), desenvolvedor e mantenedor do protocolo, esta versão representa uma melhoria substancial em relação às versões anteriores, introduzindo aprimoramentos consideráveis em termos de funcionalidade e eficiência. Destacam-se características como a comunicação bidirecional, que permite que tanto clientes quanto servidores enviem mensagens reciprocamente, o gerenciamento de sessão otimizado, oferecendo maior confiabilidade, controle de fluxo dinâmico para adaptação à capacidade de processamento do cliente e suporte a metadados mais abrangentes, possibilitando a inclusão de informações adicionais nas mensagens. Além disso, houve melhorias no tratamento de erros e na retenção de mensagens por tempo limitado, assegurando que os clientes recebam informações pertinentes e recentes. A versão 5.0 também permite que um cliente mantenha várias sessões simultâneas com configurações específicas e é projetada para ser compatível com versões anteriores, facilitando a transição. Essas melhorias fazem do MQTT 5.0 uma escolha robusta e versátil, particularmente vantajosa em aplicações de IoT e sistemas de mensagens.

Além destas, o protocolo MQTT versão 5.0 introduz melhorias substanciais na segurança em comparação com versões anteriores. Essas melhorias envolvem uma autenticação mais robusta para clientes, maior controle de acesso aos

tópicos, suporte ao TLS/SSL 1.3 para criptografia mais forte, sinalização de erros aprimorada, capacidade de incluir metadados nas mensagens e compatibilidade com certificados X.509. Esses aprimoramentos tornam o protocolo MQTT 5.0 uma opção mais segura para ambientes que exigem comunicações confiáveis e protegidas, contribuindo para mitigar riscos de segurança e garantir a integridade das mensagens transmitidas (Standard, 2019).

As configurações necessárias para que o *Broker* possa implementar mecanismos para autenticação mútua entre os dispositivos comunicantes, conforme apresentado no Capítulo 4, são apresentadas na Figura 5.5. Estas configurações devem ser escritas no arquivo de configuração do *Eclipse Mosquitto*, presente no diretório da aplicação e é identificado como *mosquitto.conf*.

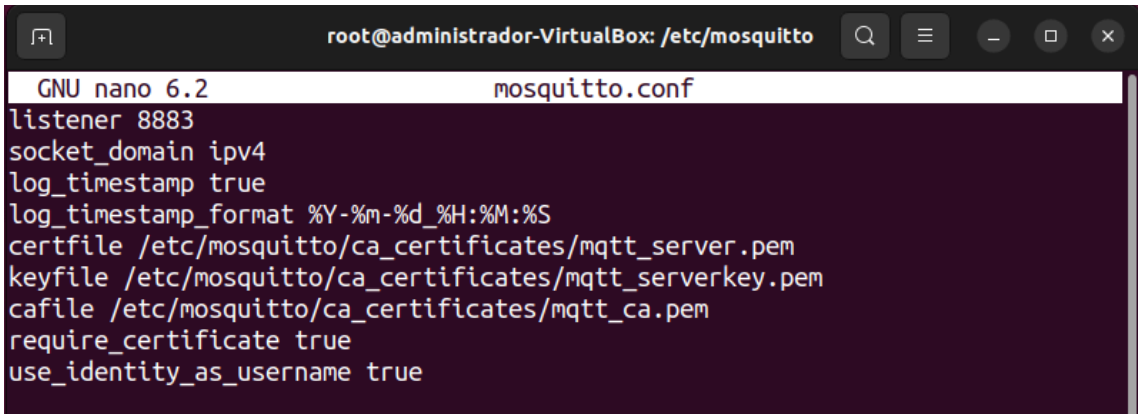


Figura 5.5: Parâmetros de configuração presentes no arquivo *mosquitto.conf* que dão suporte à comunicação segura através de TLS/SSL.

O detalhamento das funcionalidades descritas na Figura 5.5 é apresentado a seguir:

- ***listener 8883*** - Define que o servidor MQTT está ouvindo na porta 8883. A porta 8883 é frequentemente usada para comunicação MQTT segura com TLS/SSL.
- ***socket_domain ipv4*** - Refere-se ao domínio do soquete que o *Broker* Mosquitto deve usar para criar soquetes, sendo eles no formato IPv4 ou IPv6. O domínio do soquete indica o tipo de comunicação que o soquete permitirá. Será definido o valor IPv4 para o domínio do soquete, uma vez que, para esta prova de conceito, o protocolo IPv6 não foi levado em consideração. Caso sera necessário o suporte de ambas as versões, a documentação do Eclipse Mosquitto sugere que a opção *socket domain* não seja utilizada.
- ***log_timestamp*** e ***log_timestamp_format*** - O parâmetro *log_timestamp* determina se um valor de data e hora (*timestamp*)

será adicionado a cada entrada de log no sistema. Se estiver definido como *true*, um registro de log conterá um carimbo de data e hora. Se estiver definido como *false*, os registros de log não incluirão essa informação de data e hora. Já o parâmetro *log_timestamp_format* determina o formato da data e hora que é registrado nos logs do *Broker* Mosquitto. Ele permite personalizar como as datas e horas são formatadas nos registros de log.

- ***certfile /etc/mosquitto/ca_certificates/mqtt_server.pem*** - Especifica o caminho para o arquivo de certificado do servidor MQTT. O arquivo mqtt_server.pem contém o certificado do servidor que é usado para estabelecer a comunicação segura com os clientes MQTT. Ele deve ser emitido por uma autoridade certificadora confiável.
- ***keyfile /etc/mosquitto/ca_certificates/mqtt_serverkey.pem*** - Especifica o caminho para o arquivo de chave privada do servidor MQTT. O arquivo mqtt_serverkey.pem contém a chave privada correspondente ao certificado do servidor e é necessário para a negociação de TLS/SSL.
- ***cafile /etc/mosquitto/ca_certificates/mqtt_ca.pem*** - Especifica o caminho para o arquivo de certificado da autoridade certificadora (CA) raiz. O arquivo mqtt_ca.pem contém o certificado da CA raiz, que é usado pelo servidor para verificar os certificados apresentados pelos clientes durante a negociação de TLS/SSL.
- ***require_certificate true*** - Configura o servidor MQTT para exigir que os clientes apresentem um certificado durante o processo de autenticação por TLS/SSL. Isso aumenta a segurança, garantindo que apenas clientes com certificados válidos se conectem ao servidor.
- ***use_identity_as_username true*** - Indica que o servidor MQTT deve usar a identidade do cliente (geralmente derivada do certificado do cliente) como nome de usuário. Isso é útil para autenticar os clientes com base em seus certificados.

O detalhamento dos recursos fornecidos pelo *Eclipse Mosquitto* para a implementação de um *Broker* MQTT com suporte para conexões de rede criptografadas através de TLS/SSL e autenticação por meio de certificados digitais pode ser consultado em Mosquitto (2018).

Uma vez que o arquivo *mosquitto.conf* é configurado com os parâmetros descritos, é possível inicializar as funcionalidades previstas do *Broker* MQTT, garantindo níveis de segurança adequados ao processo de comunicação, por meio da autenticação mútua. O funcionamento do *Broker* MQTT usando as configurações especificadas pode ser observado na Figura 5.6.

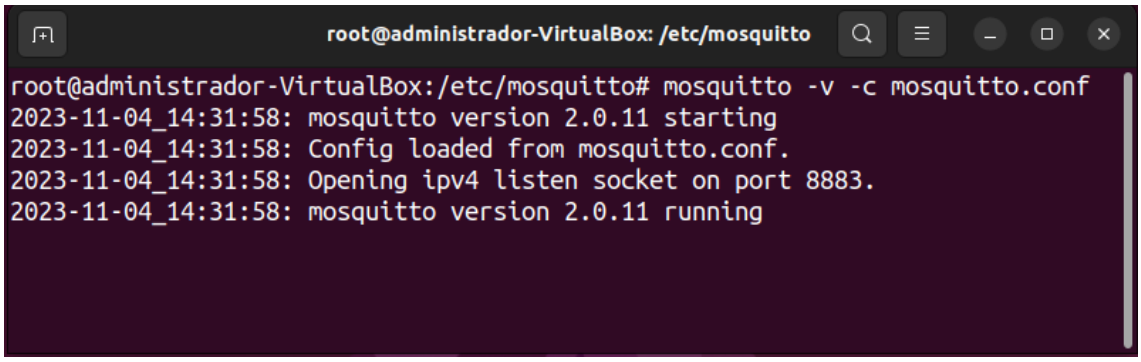


Figura 5.6: *Broker* MQTT em funcionamento escutando requisições na porta 8883 (referente aos protocolos TLS/SSL).

Por fim, vale ressaltar algumas características e detalhes importantes, além das que já foram apresentadas, sobre o Eclipse Mosquitto fazem dele uma excelente opção para implementação de *Broker* MQTT, esta incluem: o baixo *overhead*, permitindo uma maior eficiência em termos de uso de largura de banda e recursos do sistema, minimizando a sobrecarga na rede e no servidor; multiplataforma, podendo este ser executado em sistemas operacionais como *Linux*, *Windows* e *MacOS*; e flexibilidade de configuração, uma vez que possibilita aos administradores configurá-lo de acordo com suas necessidades específicas, controlando aspectos como política de segurança, retenção de mensagens e regras de acesso.

5.5 Microcontrolador ESP32

O ESP32, desenvolvido pela *Espressif Systems* como uma evolução do ESP8266, é um microcontrolador amplamente reconhecido por suas capacidades avançadas de conectividade sem fio, desempenho, recursos e flexibilidade. Baseado na arquitetura *Xtensa* LX6 da *Tensilica*, apresenta dois núcleos de CPU, Wi-Fi integrado, *Bluetooth*, GPIO, e periféricos como UART, SPI e I2C. Sua programação flexível suporta diversas linguagens por meio do Arduino IDE e do *framework* ESP-IDF da *Espressif Systems* (Kurniawan, 2019).

Devido a essa abrangência funcional, o ESP32 é amplamente empregado em projetos de IoT devido à sua conectividade sem fio integrada, ao consumo de energia ultrabaixo e a capacidades de processamento avançadas, beneficiando-se de uma combinação de diversos *softwares* proprietários. Este microcontrolador é apropriado para uma diversidade de aplicações, abrangendo desde o monitoramento remoto, controle de dispositivos, automação residencial, sensores sem fio até a comunicação M2M (*machine-to-machine*), dentre outras. Sua capacidade de processamento o posiciona como uma esco-

lha adequada para tarefas mais complexas em comparação com microcontroladores de natureza mais simplificada.

Diante das características apresentadas, foi utilizada uma placa ESP32 com um microcontrolador integrado e com capacidades de comunicação *Wi-Fi* e *Bluetooth*, para implementação de uma EDU. A programação responsável por permitir que o microcontrolador atualize a lógica de operação de seus sensores foi escrita em *MicroPython*. O *MicroPython* é uma implementação de *software* de uma linguagem de programação amplamente compatível com o *Python* 3, otimizada para funcionar em um microcontrolador, consistindo de um compilador *Python* para *bytecode* e um interpretador em tempo de execução para esse *bytecode* (Parab et al., 2023).

Entretanto, o microcontrolador ESP32, inicialmente, não é configurado para a execução de códigos em *MicroPython*, demandando, portanto, a execução do processo de gravação do *firmware* do *MicroPython* no dispositivo, conforme apresentado em MicroPython (2023). Nesse contexto, recomenda-se a preparação de um ambiente computacional apto a possibilitar a gravação do arquivo de imagem binária do *MicroPython*, especificamente projetado para o ESP32. A instalação de ferramentas de desenvolvimento, como *Python* e *esptool.py*, no ambiente computacional, viabilizará a transferência da imagem do *MicroPython* para a memória *flash* do ESP32.

Após a instalação da imagem do *MicroPython* no ESP32, será possível acessar o terminal serial para interagir com o microcontrolador. Essa interação possibilita a programação responsável pela comunicação e interação entre as EDUs e a arquitetura proposta, permitindo que essas unidades sejam reconfiguradas remotamente, de acordo com os requisitos do ambiente monitorado.

5.6 Unidade de Detecção de Emergência

O procedimento de atualização e ou reconfiguração de uma EDU se inicia por meio do solicitante de atualização, o qual encaminha um arquivo em formato JSON que encapsula o novo contexto operacional da unidade. Este arquivo é submetido ao servidor de atualização para ser publicado no *Broker* MQTT, sendo, em seguida, transmitido à EDU por meio de uma infraestrutura de comunicação sem fio, utilizando a abordagem *Over-the-Air* (OTA), a fim de possibilitar a sua implementação subsequente.

Deste modo, torna-se imprescindível adotar medidas mais rigorosas de segurança ao processo de reconfiguração da EDU. Esta necessidade emerge da potencial ameaça de ataques cibernéticos e vulnerabilidades que poderiam comprometer a autenticidade e a integridade dos dados. A salvaguarda da

infraestrutura OTA requer a implementação de robustos protocolos de autenticação, criptografia de dados, monitoramento incessante, e a capacidade de realizar atualizações de forma segura. Assegurar tais elementos é de suma importância para resguardar a confiabilidade e a integridade das operações essenciais previstas neste trabalho.

A partir dos aspectos básicos e inerentes à operação das EDUs, é definido o Algoritmo 2. Este algoritmo detalha todas estas especificidades, através uma abstração clara e lógica que irá nortear o seu desenvolvimento e implementação. Em linhas gerais, o Algoritmo 2 descreve procedimentos formais para estabelecer conexão e autenticação mútua entre a EDU e o *Broker* MQTT, utilizando certificados digitais para criar um canal seguro. Suas funções incluem o cálculo de *hashes* para garantir integridade, operações de *swap* para verificar e armazenar a versão mais recente do *software* na memória do dispositivo, e a execução da lógica de operação mais recente. Adicionalmente, implementa *thread* para verificação contínua das mensagens do *Broker* sem impactar a operação de monitoramento em curso.

Uma vez definida a sequência de passos lógicos para o funcionamento esperado da EDU, por meio do Algoritmo 2, serão formalmente apresentados no decorrer desta Seção, as representações em alto nível através da linguagem de programação *Micropython*, de todos os procedimentos mencionados, proporcionando uma compreensão abrangente das funcionalidades e operações fundamentais delineadas no referido algoritmo.

Os parâmetros delineados nas linhas 14 a 22, conforme evidenciado no Código 5.8, descrevem de maneira específica os requisitos cruciais relativos à integração do microcontrolador ESP32 da EDU com a rede local sem fio. Esta abordagem é de significativa importância para a abordagem OTA, contribuindo fundamentalmente para a conectividade remota e as operações contínuas do dispositivo.

Código Fonte 5.8: Representação em alto nível, através de linguagem de programação *MicroPython*, do processo de conexão da EDU à rede de comunicação sem fio descrito no Algoritmo 2.

```
13 # Set the details of the WLAN network
14 WLAN_SSID = "TP-LINK"
15 WLAN_PASS = "tplink12345"
16
17 # Connect to the WLAN network
18 connect_wlan = network.WLAN(network.STA_IF)
19 connect_wlan.active(True)
20 connect_wlan.connect(WLAN_SSID, WLAN_PASS)
21 while not connect_wlan.isconnected():
22     pass
23
24 print("Connected to the Wi-Fi Network")
25 print("IP Address:", connect_wlan.ifconfig()[0])
```

Algorithm 2: Algoritmo EDU.

```
1  Set WLAN and MQTT Broker connection information by specifying
     certificates path
2  initialize receivedHash, validator, update
3  #Load and execute files from device memory
4  Function executeSwap()
5      latestSwap ← device.swapVersion()
6      #Run the latest version in the device's memory
7      if latestSwap is not empty then
8          Execute latestSwap
9  Function CalculateHash(update)
10     return hash ← hashlib.sha256(update)
11 #Write JSON file in device memory
12 Function UpdateSwap(content)
13     file ← createFile(maxVersion + 1)
14     swapData ← json (content)
15     json.dump(swapData, file)
16     edu.reset
17 #Callback function to process incoming msgs
18 Function Callback(topic, msg)
19     #Check msgs received from MQTT Broker
20     if topic == OTA_TOPIC then
21         update ← msg
22         validator ← CalculateHash(update)
23     else if topic == OTA_TOPIC_HASH then
24         receivedHash ← msg
25         if receivedHash == validator then
26             UpdateSwap (update)
27         else
28             print ("Check Failed. Aborting.")
29 Subscribe Update and Validator topics
30 connBroker.setCallback(Callback)
31 #Create a new thread to run the message checking loop
32 Function CheckMsg()
33     while true do
34         connBroker.checkMessage(), time.sleep(5)
35 thread.new(CheckMsg, ())
36 executeSwap()
```

```
26 print("Network Mask:", connect_wlan.ifconfig()[1])
```

No trecho de Código 5.9, entre as linhas 29 a 32, são definidos os parâmetros relativos à conexão com o *Broker* MQTT, já o conteúdo das linhas 35 a 37 especificam a identificação dos tópicos, tanto de assinaturas, quanto de publicação que serão implementados na EDU.

Código Fonte 5.9: Representação em alto nível, através de linguagem de programação *MicroPython*, da implementação dos parâmetros de conexão com o *Broker* MQTT realizados pela EDU e descritos no Algoritmo 2.

```
28 # Configure MQTT Broker connection information
29 MQTT_BROKER = "192.168.1.50"
30 MQTT_PORT = 8883
31 MQTT_CLIENT_ID = ubinascii.hexlify(machine.unique_id()).decode("utf-8")
32 KEEPALIVE = 60
33
34 # Configure MQTT Topic Information
35 OTA_TOPIC = "ota/Update"
36 OTA_TOPIC_HASH = "ota/Validator"
37 OTA_TOPIC_CONFIRM = "ota/ConfirmUpdate"
```

Os parâmetros apresentados no trecho de Código 5.10 detalham a implementação dos requisitos necessários para o processo de autenticação mútua entre o *Broker* MQTT e a EDU. Os parâmetros definidos entre as linhas 40 a 45 fazem referência à leitura e armazenamento, em variáveis, do conteúdo dos certificados necessários para estabelecer uma conexão segura (TLS/SSL) entre a EDU e o *Broker* MQTT ao qual ele está se conectando. Os certificados estão armazenados na memória do microcontrolador da EDU. As funcionalidades atribuídas às variáveis "*mqtt_ca*", "*mqtt_edu*" e "*mqtt_key*" são similares as que já foram definidas na Seção 5.3, o que difere apenas é que as informações presentes em "*mqtt_edu*" e "*mqtt_key*" são exclusivas de cada EDU.

Código Fonte 5.10: Representação em alto nível, através de linguagem de programação *MicroPython*, dos processos de leitura do conteúdo dos certificados digitais e, em seguida, configuração parâmetros SSL para uma conexão segura com o servidor *Broker* realizados pela EDU e descritos no Algoritmo 2.

```
39 # Reading the contents of the certificates
40 with open('mqtt_ca.pem', 'rb') as f:
41     mqtt_ca = f.read()
42 with open('mqtt_educrt.pem', 'rb') as f:
43     mqtt_edu = f.read()
44 with open('mqtt_edu.pem', 'rb') as f:
45     mqtt_key = f.read()
46
47 # Parameters for TLS/SSL connection
48 SSL_PARAMS = {
49     'key': mqtt_key,
50     'cert': mqtt_edu,
51     'cadata': mqtt_ca,
52     'server_hostname': "192.168.1.50",
53     'server_side': False,
```

```
54      'cert_reqs': ssl.CERT_REQUIRED,
55      'do_handshake': True
56  }
```

Os parâmetros do dicionário descrito entre as linhas 48 a 56 do referido trecho de código estão relacionados à configuração e estabelecimento de uma conexão segura com o *Broker* MQTT através da apresentação e verificação de certificados digitais.

A especificação dos mecanismos relativos à conexão segura da EDU com o *Broker* MQTT, por meio da apresentação e verificação de certificados digitais, pode ser observada no trecho de Código 5.11, na linha 138. A definição de uma função de retorno de chamada para processar mensagens recebidas em tempo real é apresentada na linha 141. O processo de conexão pode ser visto na linha 143, por meio da função "*connect()*" atribuída ao objeto "*mqtt_client*".

Código Fonte 5.11: Representação em alto nível, através de linguagem de programação *MicroPython*, dos parâmetros de conexão e processamento de mensagens MQTT realizados pela EDU e descritos no Algoritmo 2.

```
137 # Configuring MQTT connections with TLS/SSL
138 mqtt_client = MQTTClient(client_id=MQTT_CLIENT_ID, server=MQTT_BROKER, port=MQTT_PORT,
    ↪  keepalive=KEEPALIVE, ssl=True, ssl_params=SSL_PARAMS)
139
140 # Process and perform specific actions based on MQTT message content in real time
141 mqtt_client.set_callback(mqtt_callback)
142
143 mqtt_client.connect()
144 print("MQTT connection successfully established.")
```

Após o estabelecimento da conexão, é feito o processo de assinatura dos tópicos dos quais, as mensagens recebidas através destes, permitirão que a EDU execute as suas funcionalidades esperadas. O arquivo JSON contendo a nova lógica de operação da EDU e as informações do objeto SHA-256 associadas a este arquivo JSON serão obtidos através dos tópicos descritos no trecho de Código 5.12.

Código Fonte 5.12: Representação em alto nível, através de linguagem de programação *MicroPython*, dos tópicos de mensagens a serem assinados pela EDU e descritos no Algoritmo 2.

```
146 # Topic Subscription
147 mqtt_client.subscribe(OTA_TOPIC)
148 print("Topic subscription:", OTA_TOPIC)
149 mqtt_client.subscribe(OTA_TOPIC_HASH)
150 print("Topic subscription:", OTA_TOPIC_HASH)
```

A implementação dos procedimentos para verificação da integridade dos arquivos de solicitação de atualizações, recebidos pela EDU por meio das assinaturas dos tópicos ao *Broker* MQTT, é apresentada no trecho de Código 5.13.

Código Fonte 5.13: Representação em alto nível, através de linguagem de programação *MicroPython*, do processo de verificação de integridade do arquivo de atualização realizado pela EDU e descrito no Algoritmo 2.

```
118 # Callback function to process incoming messages
119 def mqtt_callback(topic, msg):
120     global received_hash, validator_hex, update
121     if topic.decode() == OTA_TOPIC:
122         update = msg.decode()
123         print("Update file received in JSON format:", update)
124         validator = calculate_hash(update)
125         validator_hex = validator.hex()
126         print("SHA-256 object value of the received file:", validator_hex)
127     elif topic.decode() == OTA_TOPIC_HASH:
128         received_hash = msg.decode()
129         print("Value of the SHA-256 object forwarded by the Update Server:", received_hash)
130         if received_hash == validator_hex:
131             print ("SHA-256 object values are similar")
132             print ("Wait for the device to restart.")
133             update_swap(update)
134         else:
135             mqtt_client.publish(OTA_TOPIC_CONFIRM, "EDU ID: " + MQTT_CLIENT_ID + "
    ↪  Application integrity check failed. Aborting update.")
```

A EDU calcula o valor do objeto SHA-256 da atualização recebida através do tópico "*OTA/Update*", comparando-o com o valor de objeto SHA-256 informado pelo servidor de atualização através do tópico "*OTA/Validator*". Havendo uma correlação entre os valores, será executado o armazenamento conteúdo da atualização na memória do dispositivo através da função "*update_swap(update)*" presente na linha 133 do trecho de Código 5.13. Caso os valores não coincidam, é definido um procedimento para abortar o processo de atualização, descartando o arquivo recebido.

Após a definição dos processos para a verificação da integridade dos arquivos recebidos, são implementados os procedimentos para a escrita das informações contidas no arquivo de atualização, em formato JSON, para a memória do dispositivo e posteriormente a sua reinicialização. O trecho de Código 5.14 ilustra estes procedimentos.

Código Fonte 5.14: Representação em alto nível, através de linguagem de programação *MicroPython*, dos processos de criação de uma nova versão de *software* a partir do conteúdo do arquivo JSON, gravação em memória do microcontrolador e reinicialização do dispositivo realizados pela EDU e descritos no Algoritmo 2.

```
86 # Function to write a JSON file containing the new EDU operation logic ti the device's
   ↪  memory. The get_next_version() function works as a versioning mechanism
87 def update_swap(content):
88     next_version = get_next_version()
89     print (next_version)
90     filename = "swap_v{}.json".format(next_version)
91
92     swap_data = {
93         "code": content
94     }
```

```
95      with open(filename, "w") as file:
96          ujson.dump(swap_data, file)
97      machine.reset()
```

A função “*update_swap()*” recebe como parâmetro o arquivo JSON após o processo de verificação de integridade. No entanto, antes de armazenar o arquivo na memória do dispositivo, é necessário identificar o próximo valor numérico disponível para nomear a versão da atualização. Para isso, na linha 88, a função “*get_next_version()*” é chamada. Esta função foi definida com o propósito de encontrar e determinar o número da próxima versão de um arquivo de atualização na memória do dispositivo. Para realizar essa tarefa, ela verifica todos os arquivos na memória, procurando por aqueles que seguem um padrão de nomenclatura. A função identifica a versão mais alta existente, adiciona o valor 1 a esse número e retorna o valor como a próxima versão disponível para um novo arquivo de atualização, permitindo que o dispositivo saiba qual versão deverá ser implementada. Uma vez que o valor numérico da nova versão de atualização da EDU é definido, este é adicionado ao padrão de nomenclatura “*swap_v.json*”, onde as chaves “{}” recebem esse valor numérico, observado na linha 90 do trecho de Código 5.14. Após esse processo, o conteúdo do JSON é atribuído à variável “*swap_data*”, observado na linha 92. Por fim, esse conteúdo é armazenado em um novo arquivo diretamente na memória do dispositivo, como visto nas linhas 95 a 96 do trecho de Código 5.14.

Após a definição desses procedimentos, torna-se necessário reiniciar o dispositivo. Na linha 97 do trecho de Código 5.14, é implementado o procedimento de reinicialização, garantindo a remoção de quaisquer configurações ou estados anteriores relacionados à lógica de operação do dispositivo. Essa ação previne conflitos ou comportamentos indesejados decorrentes da combinação de dados ou configurações antigas com as novas.

Após o processo de reinicialização, o dispositivo é capaz de verificar a existência da versão mais recente da lógica de operação disponível na memória por meio da função “*execute_swap(update)*”, conforme ilustrado no trecho de Código 5.15.

Código Fonte 5.15: Representação em alto nível, através de linguagem de programação *MicroPython*, do processo de busca pelo arquivo de atualização mais recente na memória do microcontrolador da EDU após a reinicialização do dispositivo.

```
58  # Function to load and execute files from the device's memory
59  def execute_swap():
60      max_version = 0
61      latest_swap = None
62
63      # Search the device memory for software with the highest version number
```

```
64      for filename in os.listdir():
65          if filename.startswith("swap_v") and filename.endswith(".json"):
66              try:
67                  version = int(filename.split("_v")[1].split(".json")[0])
68                  if version > max_version:
69                      max_version = version
70                      latest_swap = filename
71              except:
72                  pass
73
74      # Opens the most recent file in the device's memory and executes its instructions
75      if latest_swap:
76          with open(latest_swap) as file:
77              swap_data = ujson.load(file)
78              code = ujson.loads(swap_data["code"])
79              exec(code["code"])
```

Esta função tem como objetivo buscar o arquivo de atualização mais recente na memória do dispositivo, carregar o código contido neste arquivo e executá-lo. Para alcançar esse propósito, a função examina a memória em busca de arquivos que sigam um padrão de nomenclatura definido. Ao identificar o arquivo com a versão mais alta, procede-se à abertura do mesmo, extração do código nele contido e subsequente execução.

Inicialmente, a função percorre o diretório raiz da memória do dispositivo em busca de arquivos de atualização, os quais são nomeados seguindo a nomenclatura definida, conforme ilustrado na linha 64 do trecho de código 5.15. Uma vez identificada a versão mais recente disponível na memória do microcontrolador da EDU, é realizado o processo de abertura do seu conteúdo e extração das instruções a serem executadas para a variável "*code*", conforme observado na linha 78. Após esses procedimentos, é feita a execução do conteúdo desta variável, que corresponde à nova lógica de operação da EDU. Isso permite ao dispositivo incorporar as alterações e melhorias contidas no arquivo de atualização.

Para assegurar a comunicação em tempo real entre a EDU e o *Broker* via protocolo MQTT, garantindo o processamento eficiente e contínuo das mensagens trocadas entre eles e mantendo o dispositivo atualizado com as informações da rede, foi fundamental estabelecer funções para a verificação constante de recebimento de mensagens nos tópicos de assinaturas. Essas funções são detalhadas nas linhas 153 a 160 do Código 5.16.

Código Fonte 5.16: Representação em alto nível, através de linguagem de programação *MicroPython*, do processo de checagem constante de mensagens encaminhadas pelo *Broker* à EDU.

```
152 # Function with loop to check for updates
153 def check_msg_execution():
154     try:
155         while True:
156                 mqtt_client.check_msg()
157                 time.sleep(5)
```

```
158         except Exception as e:
159             print("Error in check_msg_execution function:", str(e))
160     _thread.start_new_thread(check_msg_execution, ())
161
162     execute_swap()
```

Nesta etapa do processo, é preciso observar que, para o funcionamento esperado das funcionalidades descritas no Capítulo 4 à EDU, esta deve ser capaz de executar dois processos simultâneos. O primeiro processo relativo ao sensoriamento e um segundo processo é relativo a a checagem de mensagens MQTT e, posteriormente, o seu processamento.

Sendo assim, a função "*check_msg_execution()*", apresentada entre as linhas 153 a 160 do trecho de Código 5.16 e que tem por finalidade a verificação periódica da chegada de mensagens MQTT através de um *loop* contínuo, é gerenciada por uma *thread*, uma vez que este processo abrange a leitura de mensagens e a realização de operações de entrada e saída, incluindo a leitura e gravação do conteúdo das mensagens de atualização em arquivos situados no diretório do microcontrolador. Isto garante a independência de execução deste processo em relação ao programa principal, a saber, relativo a lógica de operação da EDU contida no arquivo de atualização. Deste modo, a implementação desta *thread* se torna crucial para evitar potenciais bloqueios na execução do programa principal, garantindo a continuidade ininterrupta das operações da EDU durante a verificação de mensagens.

Com todas as funcionalidades básicas definidas para o funcionamento esperado da EDU, conforme descrita pela arquitetura, é iniciada a simulação da prova de conceito para atestar a capacidade de atualização do contexto de funcionamento da EDU. Isso é realizado por meio do encaminhamento de arquivos contendo informações de reconfiguração, de modo a atender aos requisitos de monitoramento de eventos de interesse em um ambiente específico. A Figura 5.7 ilustra a EDU em funcionamento conforme os parâmetros de sensoriamento apresentados no Código 5.1.

Observa-se na saída do terminal de execução, apresentado na Figura 5.7, durante a execução da EDU, que o dispositivo se conecta à rede sem fio e adquire informações de endereçamento IPv4. Logo em seguida, realiza o processo de conexão e autenticação ao *Broker* MQTT com sucesso, efetua a assinatura dos tópicos "*OTA/Update*" e "*OTA/Validator*", e começa a executar as funcionalidades definidas no Código 5.1, referentes à lógica de operação atual da EDU para o monitoramento de temperatura ambiente e geração de alertas de incêndio.

Em determinado momento, é possível observar, na Figura 5.7, a chegada de algumas mensagens. A primeira mensagem apresenta a seguinte descrição: "*Update file received in JSON format:*", associada a uma *string* em formato

Figura 5.7: Funcionamento da EDU com base no contexto de operação definido no Código 5.1.

JSON. A segunda mensagem exibe a seguinte descrição: "*SHA-256 object value of the received file:*", acompanhada de uma *string* no formato hexadecimal. Isso indica que a atualização do contexto de operação da EDU, como definido pelo Código 5.2, foi encaminhada pelo solicitante de atualização ao servidor de atualização. Este, por sua vez, realizou a publicação dessas mensagens no *Broker* MQTT, que as direcionou imediatamente à EDU através dos tópicos de publicação.

Por fim, observa-se que a EDU executa os mecanismos para validação da integridade da atualização recebida, através da visualização da mensagem "*SHA-256 object values are similar*" e a execução do processo de reinicialização do dispositivo, através da mensagem "*Wait for the device to restart*", observada na Figura 5.7.

Após a reinicialização do microcontrolador da EDU, verifica-se uma mudança no seu contexto de operação. Agora, a EDU implementa e executa o seu processo de sensoriamento com base na nova lógica de operação definida pelo algoritmo proposto no Código 5.2. O processo de atualização do contexto de funcionamento da EDU pode ser observado através Figura 5.8.

Inicialmente, é perceptível uma série de informações registradas no terminal, conforme apresentado na Figura 5.8, que corresponde ao processo de inicialização do ESP32. Essas mensagens fornecem detalhes cruciais, como

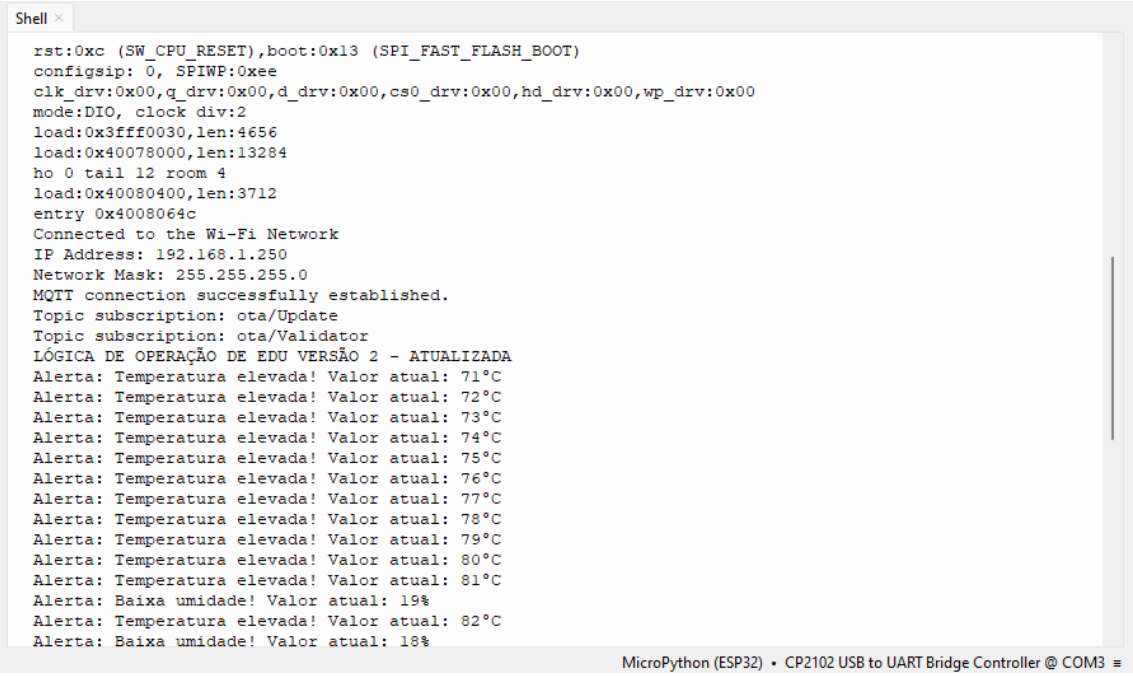


Figura 5.8: Funcionamento da EDU com base na atualização do seu contexto de operação definido no Código 5.2.

a ocorrência de uma reinicialização do dispositivo devido a razão de *software (SW_CPU_RESET)*, a definição do modo de inicialização rápido a partir da flash SPI *(SPI_FAST_FLASH_BOOT)*, e configurações específicas de *hardware*. A riqueza de informações oferecidas por essas mensagens não apenas descreve minuciosamente o processo de inicialização do ESP32, mas também serve como uma ferramenta diagnóstica valiosa para identificar possíveis problemas durante essa fase.

É pertinente ressaltar que estas informações estão relacionadas ao *status* e configuração do microcontrolador, as quais são exibidas nativamente durante sua reinicialização. Este registro adiciona uma camada de compreensão mais profunda sobre o estado e as configurações do dispositivo, contribuindo para a análise abrangente do seu ambiente operacional.

Logo após este detalhamento, observa-se, ainda na Figura 5.8, a referência da atualização da lógica de operação da EDU, onde se lê “LÓGICA DE OPERAÇÃO DE EDU VERSÃO 2 - ATUALIZADA”. Complementar a esta mensagens, observamos também que a lógica do monitoramento também é alterada, agora é possível verificar o sensoriamento das variáveis de “Temperatura” e “Umidade”, conforme descrito no Código 5.2.

5.7 Testes de Validade dos Mecanismos de Segurança

Com o propósito de validar o mecanismo de segurança baseado na verificação de objeto *hash* dos arquivos de solicitação de atualização, foram conduzidos uma série de procedimentos com o intuito de assegurar a integridade dos dados e a confiabilidade das atualizações. Inicialmente, uma solicitação de atualização contendo parâmetros de reconfiguração da EDU foi publicado no tópico "*OTA/Update*" ao *Broker* MQTT e em seguida, no tópico "*OTA/Validator*" foi publicado um valor de objeto *hash* inválido associado ao arquivo, conforme mostra a Figura 5.9.

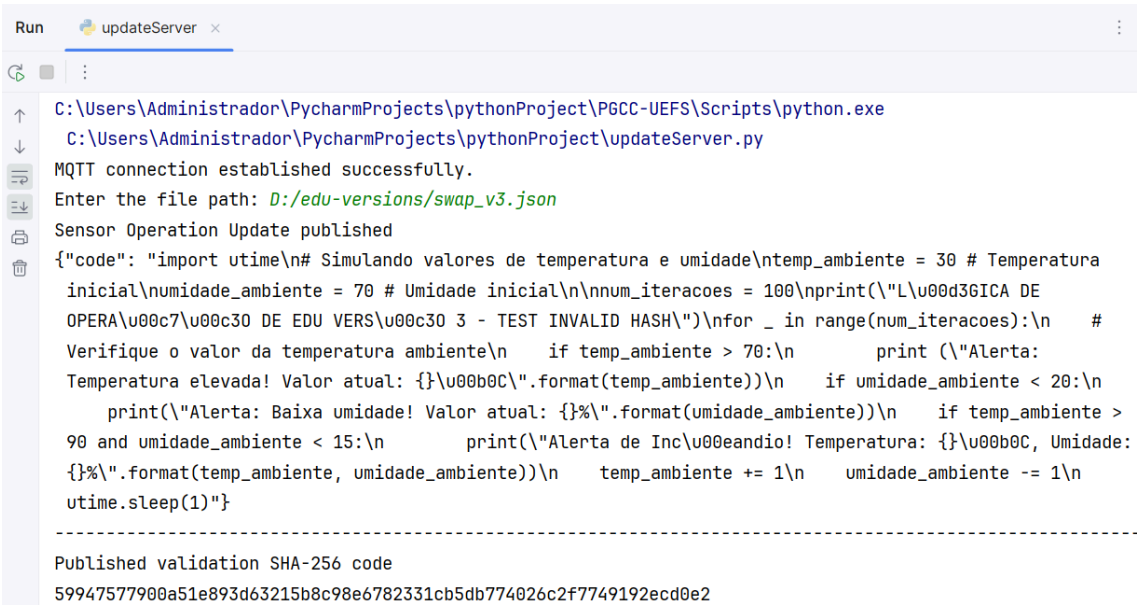


Figura 5.9: Publicação ao *Broker* das mensagens referentes ao novo contexto de operação da EDU, no formato JSON, e valor de objeto SHA-256 inválido.

Ao receber os arquivos publicados nos tópicos mencionados, a EDU inicia o processo de verificação da integridade do arquivo recebido. Antes de implementar os parâmetros de reconfiguração contidos na solicitação de atualização, é realizado o cálculo do objeto *hash* do arquivo recebido o resultado obtido é confrontado com valor de objeto *hash* encaminhado pelo servidor de atualização pelo tópico "*OTA/Validator*". A Figura 5.10 ilustra esse procedimento.

Durante a realização dos testes, a EDU constatou que o valor do objeto *hash* calculado e o valor do objeto *hash* informado eram distintos, conforme ilustrado na Figura 5.10, indicando uma possível modificação ou corrupção do arquivo de atualização. Diante desse cenário, o processo de atualização foi

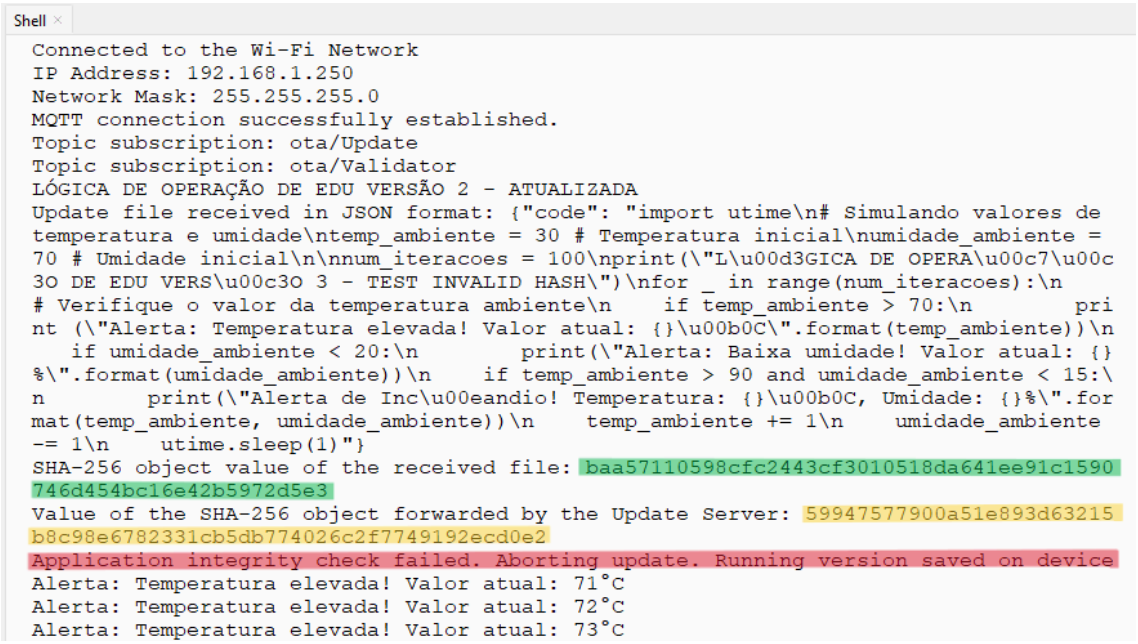


Figura 5.10: Processo de validação de integridade de objeto *hash* realizado pela EDU. Destacado na cor verde, objeto *hash* calculado a partir do arquivo recebido pelo tópico "*OTA/Update*". Destacado na cor amarela, valor de objeto *hash* inválido recebido pelo tópico "*OTA/Validator*". Destacado em vermelho, falha no processo de verificação de integridade e o cancelamento da atualização.

automaticamente abortado pela EDU. Como resultado, a operação da EDU permaneceu baseada no contexto de operação previamente salvo em memória, garantindo assim a continuidade do monitoramento.

Posteriormente, foram realizados testes destinados a validar o mecanismo de segurança de autenticação mútua baseado em certificados, foi conduzido um experimento que consistiu na geração deliberada de um certificado de cliente fraudulento para uma EDU. A Figura 5.11 apresenta a unidade de armazenamento interna da EDU contendo o certificado falso, nomeado como "*mqtt_edu_false.pem*".

Para fins comparativos, a Figura 5.12 ilustra as disparidades entre o certificado de cliente legítimo e o certificado de cliente fraudulento.

O certificado de cliente fraudulento foi submetido à análise pelo Algoritmo 2, com o propósito de viabilizar sua incorporação no procedimento de autenticação mútua, e subsequente estabelecimento de uma comunicação segura entre a EDU e o *Broker* MQTT. Detalhes deste processo são apresentados através da linha 44 do trecho de Código 5.17.

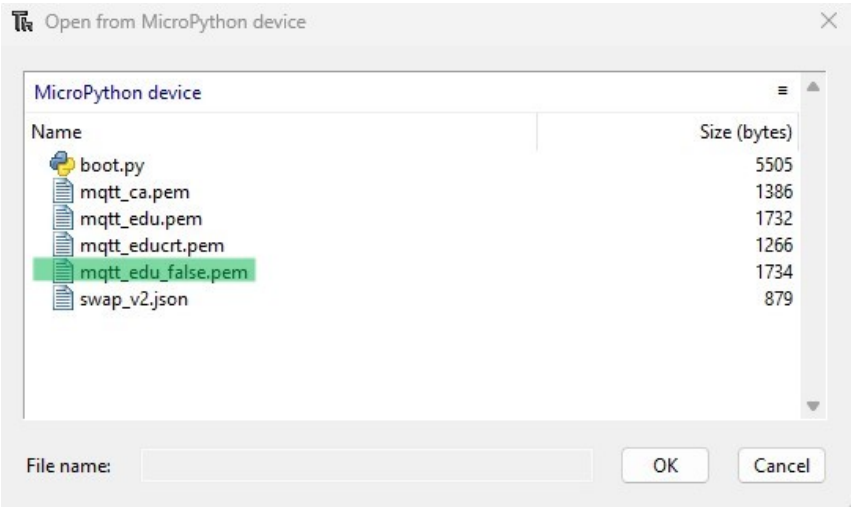


Figura 5.11: Armazenamento interno da EDU contendo o arquivo de inicialização *boot.py*, os certificados utilizados para o processo de autenticação mútua e a lógica de operação vigente do dispositivo.

[mqtt_edu.pem]

```
-----BEGIN PRIVATE KEY-----
MIIEvQIBADANBgkqhkiG9w0BAQEFAASCBKcwggSjAgEAAoIBAQCi8tJfqPfanwj3
cfhDKt2C7hrYlsMx/ppxrHPuhXRHW8HW5T/yFXRUo2hiP60M6Q81/JUNBXXQuEQ+
IhHulQ8NNfclwFKLlhQwT1w+Vm6G59mXoRXvVE8vURz5gUVUsOau7sV92XG4qHP7
J2FQwiNiggE8pRm0NEbgpC/bNsB8basypEnpNvJRNEJNzMXm3GdtCcX96jIcowig
ajE7LjRBz61rA109Fxvfk2NAl8awFPkd0VShbjSJxCe5nPbfTJv3wfi7d4apZ9n4
hbaM39QcLy1u6TB1xGO55RAw3f0SneqD0g8bNAu/9G6/0uyuydtsN23vuEpuDeXO
YH20DW5LAgMBAAECggEARplTOXwBSvjNjzwTVrKbP49/FSkY0C/h5W9U2Vp72zDv
pnY0AuR1vvf/kVXKWcYeP4M8ZuRfO7uiYZMsVsTBPTWM64/Mk8rF23p+2kljYNLS
DSaPVN6f4j3QAnAWRE/S2qb77P9JjLp7aa0RtkYuw47VD1vu64sczFaGn4E025ON
YcxxGcDoJ8jYzDf9UzBQiLj4DxL66Wb4RZc0aqXJ4vc7WBMl4mAWnC4wK92tAgKt
D0Yi429XA3dr6VMmCmHN5zIYBitfaJqu8Sb/1QQOM1MQwkT31b1FXZtPxILotiyn
JBgJhC8ohe54DIxwfOp/cBbHM7x7wiD9NXgXtytu9QKBgQDScp07IpvanqymEjBa
CPyfvbtCff9+RUAi2CLm40bggb0Hga6+YjSwFqPkSAHWuqirUt1qEy99GTROaKTg
HWXiBEC7tTd3zqc51ggZzH8SZF/LcPynDAfBcNAdYPvEpYeGtsghWmEmnOaq6uFB
3xlELjbx/rBZO7Z5cXxsfjdZ7QKBgQDGOCtrYniwYPV/vgNua8urDydp+t/VhDwA
cQzvdMQVfmHnSXdnap11hiqGIjVqPRxcIN80lYAsBUh7DbXgutNDXIjnhBJTE9cA
nVdFyyzowxvptERgh2S6h7e8JWpOz8obyzT7mqK8MfrDp+WCUWe7DWL3SE/Fdt+D
ozmk0dKCFwKBgHVTMrc6NzutxMEdWNmJMxX3mAOc7Si/gN8sMmEYinY1vmYF9Z7s
b0VQifSF9Ih7Lfe9zfUKVypeN9wS2QCdQyS7jD9Z61+e1XoYChIgASQG06U/C4wx
180PkOcgH5zlT+ZZ1r9JfOojBTJ6mjS4JYTuYvGNNVG6SaPxEhYGU+ONAoGBAJyJ
x2Bqx70jFT3opXUSlClnMk0z3okUgP4oSXrhz+jZgh/RmQaG4ArsBbEgWuanlDn9
i8hPOIWLNo5XaYcsg6Ly9PtBBeeuB2Fo4H8x0SmQy/5oaoeA5T+Z5rQjKC1NU+NX
O2Rn3pSZM9xIP6wG9dB25e07oiPjmOiwgtlzCrfJAoGALwLPDf+dvfMXsOh7Prg9
MVpR6cNbW08n7TJe8lY7MpYmHcGLPpsgPuSMXHbsiHlnPiYEmmxOwdh9tLD5OKcD
X3CjbI++GlbooPyyCuMfCnSLevkbkWs//smJu7jRQMrFWd1oa2/Sjjs113jwhmXf
ZAGt+6YyywaVKrfLJs13Xkg=
-----END PRIVATE KEY-----
```

[mqtt_edu_false.pem]

```
-----BEGIN PRIVATE KEY-----
MIIDaTCCAlECFFVGI5YpZ5JfSMtBL/3uU9h7h69EMA0GCSqGSIb3DQEBCwUAMHAx
CzAJBgNVBAYTAkJSMQ4wDAYDVQQIDAVCYWhpYTEZMBcGA1UEBwwQRmVpcmEgZGUg
U2FudGFuYTENMAsGA1UECgwEVUVGUzEQMA4GA1UECwwHUEdDQ19DQTEVMBMGA1UE
AwwMMTkyLjE2OC4xLjUwMB4XDTIzMTEwMjE5MDMwMloXDTI0MTEwMTE5MDMwMlow
cjELMAkGA1UEBhMCQlIxDjAMBgNVBAgMBUJhaGlhMRkwFwYDVQQHDBBGZWlyYSBk
ZSBTYW50YW5hMQ0wCwYDVQQKDARVRUZTMREwDwYDVQQLDAhQR0NDX0VEVTEWMBQG
A1UEAwwNMTkyLjE2OC4xLjI1MDCCASIwDQYJKoZIhvcNAQEBBQADggEPADCCAQoC
ggEBAKLy01+o99qfCPdx+EMq3YLuGtiWwzH+mnGsc+6FdEdbwdblP/IVdFSjaGI/
YH20DW5LAgMBAAECggEARplTOXwBSvjNjzwTVrKbP49/FSkY0C/h5W9U2Vp72zDv
pnY0AuR1vvf/kVXKWcYeP4M8ZuRfO7uiYZMsVsTBPTWM64/Mk8rF23p+2kljYNLS
DSaPVN6f4j3QAnAWRE/S2qb77P9JjLp7aa0RtkYuw47VD1vu64sczFaGn4E025ON
YcxxGcDoJ8jYzDf9UzBQiLj4DxL66Wb4RZc0aqXJ4vc7WBMl4mAWnC4wK92tAgKt
D0Yi429XA3dr6VMmCmHN5zIYBitfaJqu8Sb/1QQOM1MQwkT31b1FXZtPxILotiyn
JBgJhC8ohe54DIxwfOp/cBbHM7x7wiD9NXgXtytu9QKBgQDScp07IpvanqymEjBa
9t9Mm/fB+Lt3hqln2fiFtozf1BwvLW7pMHXEY7nlEDDd/RKd6oPSDxs0C7/0br/S
7K7J22w3be+4Sm4N5c5gfbQNbksCAwEAATANBgkqhkiG9w0BAQsFAAOCAQEAmHvX
X4g9s6SnbTlpRCCwb4LaRQPBP/HEQ0gRXz4xFLADBl+fWAV8KwF5wZrtYqCF2o0i
aC0U5CjrcK7m/pyf5tkda73ByxZPJRwZO/B1waBB04LEkqzRaGmjm0AMVHTNKpb3
m72scih2f4GQRHDtfT0I0miuMFqRI3ty0+gmsYTRCqbKaT1/2WhY9RetrRdYZthY
R7gMBMGmdrYBuX1g9ZRDXlyahZlDgU/Yi0XCfOivRPjYlvfchu2D30uTmrVsae1F
UshyS6EyEKbbwf7To+2Zir7wk6mWpWbPQq5uVIu0E2Al66tHiC3R8cr0hA4C6uAx
B58gUDu4U4POcmEZlg==8x0SmQy/5oaoeA5T+ZDSGGRsewwsfhyg5rQjKC1NU+NX
O2Rn3pSZM9xIP6wG9dB25e07oiPjmOiwgtlzCrfJAoGALwLPDf+dvfMXsOh7Prg9
MVpR6cNbW08n7TJe8lY7MpYmHcGLPpsgPuSMXHbsiHlnPiYEmmxOwdh9tLD5OKcD
X3CjbI++GlbooPyyCuMfCnSLevkbkWs//smJu7jRQMrFWd1oa2/Sjjs113jwhmXf
ZAGt+6YyywaVKrfLJs13Xkg=
-----END PRIVATE KEY-----
```

Figura 5.12: Certificados de cliente pertencentes à EDU utilizados no processo de autenticação mútua ao *Broker* MQTT. À esquerda, certificado cliente autêntico, à direita, certificado cliente fraudado.

Código Fonte 5.17: Representação em alto nível, através de linguagem de programação *MicroPython*, dos processos de leitura do conteúdo dos certificados digitais para uma conexão segura com o servidor *Broker* realizados pela EDU e descritos no Algoritmo 2.

```
39  # Reading the contents of the certificates
40  with open('mqtt_ca.pem', 'rb') as f:
41      mqtt_ca = f.read()
42  with open('mqtt_educrt.pem', 'rb') as f:
```

```
43      mqtt_edu = f.read()
44  with open('mqtt_edu_false.pem', 'rb') as f:
45      mqtt_key = f.read()
```

Posteriormente, foram realizadas tentativas de conexões ao *Broker* MQTT. Contudo, ao prosseguir com o processo de autenticação, o *Broker* MQTT executa uma verificação da integridade do certificado apresentado pela EDU e prontamente detecta a sua autenticidade duvidosa. Como resultado, o *Broker* nega imediatamente o acesso, impedindo qualquer estabelecimento de comunicação com a EDU portadora do certificado falso. Detalhes deste processo são apresentados através da Figura 5.13.

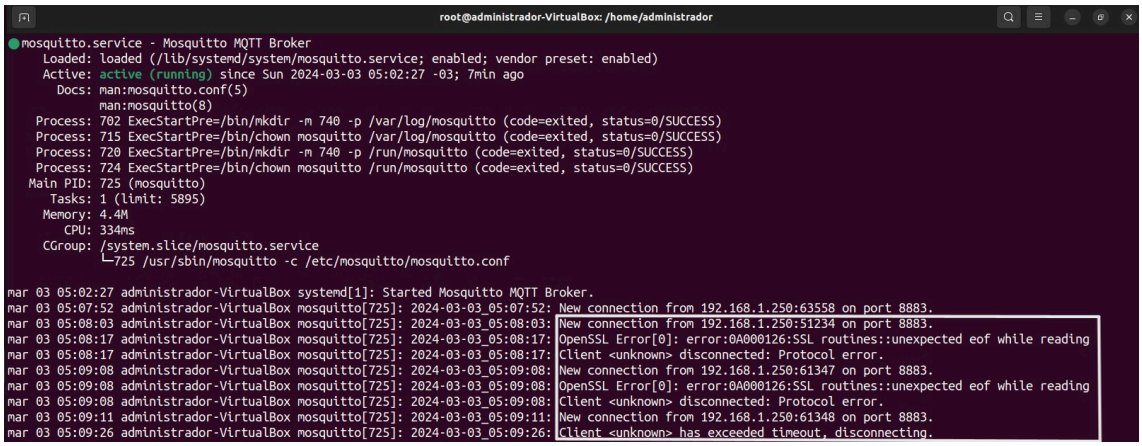


Figura 5.13: Detalhamento de informações verbais exibidas durante a tentativa de autenticação da EDU ao *Broker* MQTT utilizando um certificado de cliente fraudado.

Por fim, ao analisar os resultados desta prova de conceito, torna-se evidente que a arquitetura delineada no Capítulo 4 não apenas demonstra sua funcionalidade, mas também estabelece um paradigma eficaz para a integração de requisitos de monitoramento em EDUs por meio de solicitações remotas, utilizando uma infraestrutura de comunicação sem fio conforme preconizado pela abordagem OTA. Destaca-se que a implementação desse processo não se limita à mera oferta de funcionalidade; ela também confere um nível substancial de confiabilidade e segurança, atributos cruciais em cenários que exigem prontidão e resposta efetiva a situações de emergência. Os experimentos realizados corroboram a robustez da arquitetura, revelando sua capacidade de promover a reconfiguração de EDUs quando devidamente solicitada. Essa evidência fortalece a convicção de que a abordagem adotada neste trabalho não só atende aos requisitos propostos, mas também representa um avanço significativo na construção de sistemas de detecção de emergências, proporcionando uma base sólida para futuras inovações e aprimoramentos na área.

Capítulo 6

Conclusões

Esta dissertação propôs uma arquitetura destinada a guiar o desenvolvimento, implantação e operação de Unidades de Detecção de Emergências baseadas em sensores. Essas unidades têm a capacidade de serem reconfiguradas de maneira segura por meio de solicitações remotas, utilizando uma infraestrutura de comunicação sem fio. A arquitetura permite a incorporação de novos requisitos de monitoramento de eventos de interesse, alcançando assim um alto grau de implantação, automação e escalabilidade durante a execução. Para atingir esse objetivo, foi necessário especificar uma série de componentes com funções bem definidas, capazes de suportar os procedimentos relacionados.

Na literatura de aplicações de IoT para o domínio das cidades inteligentes, diversas abordagens dedicadas ao monitoramento e detecção de emergências já foram exploradas. Entretanto, a maioria dessas aplicações focam em um único fenômeno de monitoramento, o que as tornam limitadas diante da complexidade de ambientes altamente dinâmicos e heterogêneos, como os centros urbanos. Tais ambientes apresentam demandas distintas que evoluem ao longo do tempo. Para superar essas limitações, este estudo apresentou uma abordagem mais abrangente, capaz de permitir a reconfiguração de unidades de sensoriamento, adaptando as aplicações aos diversos contextos emergenciais em constante mudança em ambientes urbanos.

Este trabalho apresenta uma série de contribuições relevantes para o desenvolvimento de aplicações de IoT, especialmente aquelas direcionadas ao gerenciamento de emergências em cidades inteligentes. Os resultados alcançados apresentam o potencial de aprimorar significativamente a eficiência e a consistência no processo de tomada de decisões diante de situações emergenciais inerentes ao contexto dos centros urbanos.

Desvincular o *hardware* dos dispositivos sensores do *software* de sensoria-

mento possibilita variações e adaptações dos componentes de entrada de dados, bem como dos algoritmos e métodos para a detecção de eventos de emergência. Essa abordagem viabiliza que as EDUs possam monitorar uma variedade de eventos de emergência, garantindo flexibilidade de operação conforme a necessidade de monitoramento de eventos de interesse. No entanto, é importante ressaltar que a flexibilidade proposta é garantida pela disponibilidade de *hardwares* de sensoriamento específicos presentes nas EDUs. Em outras palavras, caso haja a necessidade de mudança ou reconfiguração do método de detecção de eventos, levando em consideração a presença de novas variáveis ambientais no contexto monitorado, é fundamental que as EDUs possuam os sensores adequados. Contudo, a definição dos elementos de *hardware* para o sensoriamento não faz parte deste trabalho.

A adoção desta arquitetura proporciona benefícios adicionais, como escalabilidade, redução de custos, agilidade, integração e segurança. A escalabilidade pode ser observada pela capacidade da arquitetura proposta em adicionar ou remover EDUs e reconfigurar métodos de detecção, proporcionando flexibilidade às aplicações de gerenciamento de emergências para lidarem com diferentes demandas. A redução de custos está atrelada ao desenvolvimento e implementação de novas aplicações de monitoramento, uma vez que é suficiente reconfigurar as EDUs existentes para atender aos novos requisitos.

O aspecto da agilidade, por sua vez, está relacionado à capacidade de implementar novos requisitos de monitoramento por meio de solicitações remotas em tempo de execução, sem impactar a infraestrutura como um todo. Isso possibilita respostas mais rápidas a eventos e situações de emergência. A integração proporcionada pela arquitetura permite que as unidades de detecção de emergências sejam incorporadas a outras aplicações de IoT e sistemas de informação já existentes, conferindo à solução uma abordagem mais completa e eficiente, uma vez que o *hardware* não está vinculado a um *software* de monitoramento específico. Por fim, a segurança é garantida por meio da implementação de mecanismos de autenticação dos elementos comunicantes e de integridade das solicitações e mensagens trocadas entre os dispositivos, garantindo um maior nível de confiabilidade durante os processos de reconfiguração através da prevenção de solicitações não autorizadas, bem como a corrupção das mensagens de reconfiguração às EDUs.

Os experimentos iniciais, baseados na prova de conceito, demonstraram que a arquitetura proposta é capaz de permitir a reconfiguração de EDUs de forma segura através de uma infraestrutura de comunicação sem fio. Por meio do desenvolvimento dos procedimentos especificados na arquitetura, torna-se possível encaminhar requisições de alteração de algoritmos ou mecanismos de decisão para detecção de eventos de maneira segura a essas unidades, permitindo a incorporação de novos requisitos de monitoramento. Cabe ressaltar

que aspectos práticos da implementação da arquitetura podem ser alterados de acordo com as peculiaridades e necessidades de cada aplicação.

Dessa forma, considerando os resultados obtidos através da prova de conceito, bem como a publicação de artigos específicos sobre a temática em questão, torna-se evidente que a arquitetura proposta emerge como um modelo de referência valioso a ser adotado e utilizado como solução para sistemas de gerenciamento de emergências. Entretanto, é reconhecida a necessidade de validações adicionais, especialmente em cenários de monitoramento mais abrangentes e com a presença de várias unidades de detecção de emergências, a fim de aprimorar a robustez e generalização dos resultados.

Sugere-se, portanto, que este trabalho seja estendido no futuro para implantação em um cenário real de cidade inteligente, fazendo uso de um grande número de unidades de detecção de emergências compatíveis, distribuídas em diversas áreas do ambiente urbano. A intenção é avaliar a performance relativa ao processo de reconfiguração, bem como identificar, por meio de estudos mais aprofundados, mecanismos de segurança mais eficientes para aplicações de IoT voltadas ao domínio do gerenciamento de emergências em cidades inteligentes.

Referências

Al-Fuqaha, A., Guizani, M., Mohammadi, M., Aledhari, M., e Ayyash, M. (2015). Internet of things: A survey on enabling technologies, protocols, and applications. *IEEE Communications Surveys & Tutorials*, 17(4):2347–2376.

Al-Naji, F. H. e Zagrouba, R. (2020). A survey on continuous authentication methods in internet of things environment. *Computer Communications*, 163:109–133.

Al-Qaseemi, S. A., Almulhim, H. A., Almulhim, M. F., e Chaudhry, S. R. (2016). Iot architecture challenges and issues: Lack of standardization. In *2016 Future Technologies Conference (FTC)*, páginas 731–738.

Alazawi, Z., Alani, O. Y., Abdljabar, M. B., Altowaijri, S. M., e Mehmood, R. (2014). A smart disaster management system for future cities. In *WiMobCity '14*.

AlGhamdi, R., Alassafi, M. O., Alshdadi, A. A., Dessouky, M. M., Ramdan, R. A., e Aboshosha, B. W. (2023). Developing trusted iot healthcare information-based ai and blockchain. *Processes*, 11(1).

Allam, Z. e Dhunny, Z. A. (2019). On big data, artificial intelligence and smart cities. *Cities*, 89:80–91.

Alshuqayran, N., Ali, N., e Evans, R. (2016). A systematic mapping study in microservice architecture. In *2016 IEEE 9th International Conference on Service-Oriented Computing and Applications (SOCA)*, páginas 44–51.

Amaral, M. d. (2019). *O papel do direito urbanístico na sociedade potencializadora de desastres*. Coleção Direito das catástrofes e dos desastres ambientais. Editora Prismas.

Asghari, P., Rahmani, A. M., e Javadi, H. H. S. (2019). Internet of things applications: A systematic review. *Computer Networks*, 148:241–261.

Ashton, K. (2009). That internet of things thing. rfid journal (2009). *URL: http://www. rfidjournal. com/articles/view*, 4986.

Atzori, L., Iera, A., e Morabito, G. (2010). The internet of things: A survey. *Computer Networks*, 54(15):2787–2805.

Atzori, L., Iera, A., e Morabito, G. (2017). Understanding the internet of things: definition, potentials, and societal role of a fast evolving paradigm. *Ad Hoc Networks*, 56:122–140.

Banks, A., Briggs, E., Borgendale, K., e Gupta, R. (2020). Mqtt version 5.0, oasis standard (2019).

Bauwens, J., Ruckebusch, P., Giannoulis, S., Moerman, I., e De Poorter, E. (2020). Over-the-air software updates in the internet of things: An overview of key principles. *IEEE Communications Magazine*, 58(2):35–41.

Bello, O. e Zeadally, S. (2019). Toward efficient smartification of the internet of things (iot) services. *Future Generation Computer Systems*, 92:663–673.

Chaudhari, S. S. e Bhole, V. Y. (2018). Solid waste collection as a service using iot-solution for smart cities. In *2018 International Conference on Smart City and Emerging Technology (ICSCET)*, páginas 1–5.

Chen, C., Jiang, J., Zhou, Y., Lv, N., Liang, X., e Wan, S. (2022). An edge intelligence empowered flooding process prediction using internet of things in smart city. *Journal of Parallel and Distributed Computing*, 165:66–78.

Chen, M., Miao, Y., Hao, Y., e Hwang, K. (2017a). Narrow band internet of things. *IEEE access*, 5:20557–20577.

Chen, R., Li, S., e Li, Z. (2017b). From monolith to microservices: A dataflow-driven approach. In *2017 24th Asia-Pacific Software Engineering Conference (APSEC)*, páginas 466–475.

Cisco, U. (2020). Cisco annual internet report (2018–2023) white paper. *Cisco: San Jose, CA, USA*, 10(1):1–35.

Costa, D. G., Damasceno, A., e Silva, I. (2019). Cityspeed: A crowdsensing-based integrated platform for general-purpose monitoring of vehicular speeds in smart cities. *Smart Cities*, 2(1):46–65.

Costa, D. G. e de Oliveira, F. P. (2020). A prioritization approach for optimization of multiple concurrent sensing applications in smart cities. *Future Generation Computer Systems*, 108:228–243.

Costa, D. G. e Duran-Faundez, C. (2018). Open-source electronics platforms as enabling technologies for smart cities: Recent developments and perspectives. *Electronics*, 7(12).

Costa, D. G., Peixoto, J. P. J., Jesus, T. C., Portugal, P., Vasques, F., Rangel, E., e Peixoto, M. (2022a). A survey of emergencies management systems in smart cities. *IEEE Access*, páginas 1–1.

Costa, D. G., Peixoto, J. P. J., Jesus, T. C., Portugal, P., Vasques, F., Rangel, E., e Peixoto, M. (2022b). A survey of emergencies management systems in smart cities. *IEEE Access*, 10:61843–61872.

Costa, D. G., Vasques, F., Portugal, P., e Aguiar, A. (2020a). A distributed multi-tier emergency alerting system exploiting sensors-based event detection to support smart city applications. *Sensors*, 20(1).

Costa, D. G., Vasques, F., Portugal, P., e Aguiar, A. (2020b). On the use of cameras for the detection of critical events in sensors-based emergency alerting systems. *Journal of Sensor and Actuator Networks*, 9(4).

Da Silva, G. F. P., Costa, D. G., e De Jesus, T. C. (2023). A secure ota approach for flexible operation of emergency detection units in smart cities. In *2023 IEEE International Smart Cities Conference (ISC2)*, páginas 01–07.

de Santana, C. J. L., Andrade, L. J. S., Delicato, F. C., e Prazeres, C. V. S. (2021). Increasing the availability of iot applications with reactive microservices. *Serv. Oriented Comput. Appl.*, 15:109–126.

de Sousa, M. J. B., Gonzalez, L. F. G., Ferdinando, E. M., e Borin, J. F. (2022). Over-the-air firmware update for iot devices on the wild. *Internet of Things*, 19:100578.

Donta, P. K., Srirama, S. N., Amgoth, T., e Annavarapu, C. S. R. (2022). Survey on recent advances in iot application layer protocols and machine learning scope for research directions. *Digital Communications and Networks*, 8(5):727–744.

Dragulinescu, A.-M., Dragulinescu, A., Zamfirescu, C., Halunga, S., e Suciu, G. (2019). Smart neighbourhood: Lora-based environmental monitoring and emergency management collaborative iot platform. In *2019 22nd International Symposium on Wireless Personal Multimedia Communications (WPMC)*, páginas 1–6.

Du, R., Santi, P., Xiao, M., Vasilakos, A. V., e Fischione, C. (2018). The sensable city: A survey on the deployment and management for smart city monitoring. *IEEE Communications Surveys & Tutorials*, 21(2):1533–1560.

Duangsuwan, S., Takarn, A., Nujankaew, R., e Jamjareegulgarn, P. (2018). A study of air pollution smart sensors lpwan via nb-iot for thailand smart cities 4.0. In *2018 10th International Conference on Knowledge and Smart Technology (KST)*, páginas 206–209.

El-hajj, M., Fadlallah, A., Chamoun, M., e Serhrouchni, A. (2019). A survey of internet of things (iot) authentication schemes. *Sensors*, 19(5).

Elemam, E., Bahaa-Eldin, A. M., Shaker, N. H., e Sobh, M. (2020). Formal verification for a pmqtt protocol. *Egyptian Informatics Journal*, 21(3):169–182.

Elhadi, S., Marzak, A., Sael, N., e Merzouk, S. (2018). Comparative study of iot protocols. *Smart Application and Data Analysis for Smart Cities (SADASC'18)*.

Ergen, S. C. (2004). Zigbee/ieee 802.15. 4 summary. *UC Berkeley, September*, 10(17):11.

Ferro, E. e Potorti, F. (2005). Bluetooth and wi-fi wireless protocols: a survey and a comparison. *IEEE Wireless Communications*, 12(1):12–26.

Francesco, P. D., Malavolta, I., e Lago, P. (2017). Research on architecting microservices: Trends, focus, and potential for industrial adoption. In *2017 IEEE International Conference on Software Architecture (ICSA)*, páginas 21–30.

Freitas, C. M. d., Barcellos, C., Asmus, C. I. R. F., Silva, M. A. d., e Xavier, D. R. (2019). Da samarco em mariana à vale em brumadinho: desastres em barragens de mineração e saúde coletiva. *Cadernos de Saúde Pública*, 35.

Geetha, S. e Cicilia, D. (2017). Iot enabled intelligent bus transportation system. In *2017 2nd International Conference on Communication and Electronics Systems (ICCES)*, páginas 7–11. IEEE.

Gerland, P., Hertog, S., Wheldon, M., Kantorova, V., Gu, D., Gonnella, G., Williams, I., Zeifman, L., Bay, G., Castanheira, H., Kamiya, Y., Bassarsky, L., Gaigbe-Togbe, V., e Spoorenberg, T. (2022). *World Population Prospects 2022: Summary of results*.

Giffinger, R., Fertner, C., Kramar, H., Meijers, E., et al. (2007). City-ranking of european medium-sized cities. *Cent. Reg. Sci. Vienna UT*, 9(1):1–12.

Gigli, M., Koo, S. G., et al. (2011). Internet of things: services and applications categorization. *Adv. Internet Things*, 1(2):27–31.

Gil, A. C. (2008). *Métodos e técnicas de pesquisa social*. 6. ed. Ediitora Atlas SA.

Goap, A., Sharma, D., Shukla, A. K., e Krishna, C. R. (2018). An iot based smart irrigation management system using machine learning and open source technologies. *Comput. Electron. Agric.*, 155:41–49.

Gonçalves, D. d. O. (2015). Criptografia adaptativa em redes de sensores visuais sem fio. Dissertação de Mestrado, Mestrado em Computaçãoo Aplicada. DEPARTAMENTO DE TECNOLOGIA.

Grover, K., Kahali, D., Verma, S., e Subramanian, B. (2020). Wsn-based system for forest fire detection and mitigation. In Subramanian, B., Chen, S.-S., e Reddy, K. R., editores, *Emerging Technologies for Agriculture and Environment*, páginas 249–260, Singapore. Springer Singapore.

Hammi, M. T., Livolant, E., Bellot, P., Serhrouchni, A., e Minet, P. (2018). A lightweight mutual authentication protocol for the iot. In *Mobile and Wireless Technologies 2017: ICMWT 2017 4*, páginas 3–12. Springer.

Hancke, G. P., Silva, B. D. C. e., e Hancke, Jr., G. P. (2013). The role of advanced sensing in smart cities. *Sensors*, 13(1):393–425.

Harun-Ar-Rashid, M., Chowdhury, O., Hossain, M. M., Rahman, M. M., Muhammad, G., AlQahtani, S. A., Alrashoud, M., Yassine, A., e Hossain, M. S. (2023). Iot-based medical image monitoring system using hl7 in a hospital database. *Healthcare*, 11(1).

Haxhibeqiri, J., De Poorter, E., Moerman, I., e Hoebeke, J. (2018). A survey of lorawan for iot: From technology to application. *Sensors*, 18(11):3995.

Henderson, J. V. (2005). Chapter 24 - urbanization and growth. In Aghion, P. e Durlauf, S. N., editores, *Handbook of Economic Growth*, volume 1 of *Handbook of Economic Growth*, páginas 1543–1591. Elsevier.

Heydon, R. e Hunn, N. (2012). Bluetooth low energy. *CSR Presentation, Bluetooth SIG https://www. bluetooth. org/DocMan/handlers/DownloadDoc. ashx*.

Hintzbergen, J., Hintzbergen, K., Smulders, A., e Baars, H. (2018). *Fundamentos de Segurança da Informação: Com base na ISO 27001 e na ISO 27002*. Brasport.

Hofer-Schmitz, K. e Stojanović, B. (2020). Towards formal verification of iot protocols: A review. *Computer Networks*, 174:107233.

Hohpe, G. e Woolf, B. (2004). *Enterprise integration patterns: Designing, building, and deploying messaging solutions*. Addison-Wesley Professional.

Hollands, R. (2008). Will the real smart city please stand up? *City*, 12:303–320.

Hossen, M. S., Kabir, A. F. M. S., Khan, R. H., e Azfar, A. (2010). Interconnection between 802.15.4 devices and ipv6: Implications and existing approaches.

IBM (2023). O que são message brokers?

Jain, V. R., Bagree, R., Kumar, A., e Ranjan, P. (2008). wildcense: Gps based animal tracking system. In *2008 International Conference on Intelligent Sensors, Sensor Networks and Information Processing*, páginas 617–622.

Jatobá, S. U. S. (2011). Urbanização, meio ambiente e vulnerabilidade social. *Boletim Regional, Urbano e Ambiental*.

Kandris, D., Nakas, C., Vomvas, D., e Koulouras, G. (2020). Applications of wireless sensor networks: An up-to-date survey. *Applied System Innovation*, 3(1).

Kaur, K. (2018). A survey on internet of things – architecture, applications, and future trends. In *2018 First International Conference on Secure Cyber Computing and Communication (ICSCCC)*, páginas 581–583.

Khan, M. N. H. e Neustaedter, C. (2019). An exploratory study of the use of drones for assisting firefighters during emergency situations. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, CHI '19, página 1–14, New York, NY, USA. Association for Computing Machinery.

Khutsoane, O., Isong, B., e Abu-Mahfouz, A. M. (2017). Iot devices and applications based on lora/lorawan. In *IECON 2017-43rd Annual Conference of the IEEE Industrial Electronics Society*, páginas 6107–6112. IEEE.

Klanovicz, J. e Ferreira Filho, C. B. (2018). A fabricação de uma cidade tóxica: A tribuna de santos e os desastres tecnológicos de cubatão (brasil) na década de 1980. *Revista Cadernos do Ceom*, 31(48):10–20.

Kodali, R. K., Jain, V., Bose, S., e Boppana, L. (2016). Iot based smart security and home automation system. In *2016 international conference on computing, communication and automation (ICCCA)*, páginas 1286–1289. IEEE.

Kraijak, S. e Tuwanut, P. (2015). A survey on internet of things architecture, protocols, possible applications, security, privacy, real-world implementation and future trends. In *2015 IEEE 16th International Conference on Communication Technology (ICCT)*, páginas 26–31.

Krull, C. R., McMillan, L. F., Fewster, R. M., van der Ree, R., Pech, R., Dennis, T., e Stanley, M. C. (2018). Testing the feasibility of wireless sensor networks and the use of radio signal strength indicator to track the movements of wild animals. *Wildlife Research*, 45(8):659.

Krylovskiy, A., Jahn, M., e Patti, E. (2015). Designing a smart city internet of things platform with microservice architecture. In *2015 3rd International Conference on Future Internet of Things and Cloud*, páginas 25–30.

Kurniawan, A. (2019). *Internet of Things Projects with ESP32: Build exciting and powerful IoT projects using the all-new Espressif ESP32*. Packt Publishing Ltd.

Li, G. e Jacobsen, H.-A. (2005). Composite subscriptions in content-based publish/subscribe systems. In *ACM/IFIP/USENIX International Conference on Distributed Systems Platforms and Open Distributed Processing*, páginas 249–269. Springer.

Light, R. A. (2017). Mosquitto: server and client implementation of the mqtt protocol. *Journal of Open Source Software*, 2(13):265.

Lukaj, V., Martella, F., Fazio, M., Celesti, A., e Villari, M. (2023). Establishment of a trusted environment for iot service provisioning based on x3dh-based brokering and federated blockchain. *Internet of Things*, 21:100686.

Made Wirawan, I., Dwi Wahyono, I., Idfi, G., e Radityo Kusumo, G. (2018). Iot communication system using publish-subscribe. In *2018 International Seminar on Application for Technology of Information and Communication*, páginas 61–65.

Mahgoub, A., Tarrad, N., Elsherif, R., Ismail, L., e Al-Ali, A. (2020). Fire alarm system for smart cities using edge computing. In *2020 IEEE International Conference on Informatics, IoT, and Enabling Technologies (ICIoT)*, páginas 597–602.

Majumder, A. J. e Izaguirre, J. A. (2020). A smart iot security system for smart-home using motion detection and facial recognition. In *2020 IEEE 44th Annual Computers, Software, and Applications Conference (COMPSAC)*, páginas 1065–1071. IEEE.

MicroPython (2023). Micropython tutorial for esp32. https://docs.micropython.org/en/latest/esp32/tutorial/intro.html.

Mosquitto (2018). Eclipse mosquitto. https://mosquitto.org/.

Muccini, H. e Moghaddam, M. T. (2018). Iot architectural styles. In Cuesta, C. E., Garlan, D., e Pérez, J., editores, *Software Architecture*, páginas 68–85, Cham. Springer International Publishing.

Mulligan, G. (2007). The 6lowpan architecture. In *Proceedings of the 4th workshop on Embedded networked sensors*, páginas 78–82.

Nebbione, G. e Calzarossa, M. C. (2020). Security of iot application layer protocols: Challenges and findings. *Future Internet*, 12(3).

Oh, S., Kim, J.-H., e Fox, G. (2010). Real-time performance analysis for publish/subscribe systems. *Future Generation Computer Systems*, 26(3):318–323.

Oliveira, F., Costa, D. G., e Silva, I. (2021). On the development of flexible mobile multi-sensor units based on open-source hardware platforms and a reference framework. *HardwareX*, 10:e00243.

Oliveira, F. L. S. d. (2021). Monitoramento e avaliação de condições adversas em ciclovias através de multi-sensoriamento por bicicletas e processamento de dados urbanos. Dissertação de Mestrado, Mestrado em Computação Aplicada. Departamento de Tecnologia.

Oliveira, F. P. d. (2018). Uma proposta para configuração dinâmica de sensores em cidades inteligentes. Dissertação de Mestrado, Mestrado em Computação Aplicada. Departamento de Ciências Exatas.

Palmo, Y., Tanimoto, S., Sato, H., e Kanai, A. (2022). Optimal federation method for embedding internet of things in software-defined perimeter. *IEEE Consumer Electronics Magazine*, páginas 1–7.

Parab, J., Lanjewar, M., Sequeira, M., Naik, G., e Shaikh, A. (2023). *Python Programming Recipes for IoT Applications*. Transactions on Computer Systems and Networks. Springer Nature Singapore.

Peixoto, J. P. J., Bittencourt, J. C. N., Jesus, T. C., Costa, D. G., Portugal, P., e Vasques, F. (2024). Exploiting geospatial data of connectivity and urban infrastructure for efficient positioning of emergency detection units in smart cities. *Computers, Environment and Urban Systems*, 107:102054.

Porto, M. F. d. S. (2016). A tragédia da mineração e do desenvolvimento no brasil: desafios para a saúde coletiva.

Prasad, R. (2020). Energy efficient smart street lighting system in nagpur smart city using iot-a case study. In *2020 Fifth International Conference on Fog and Mobile Edge Computing (FMEC)*, páginas 100–103.

Ramathulasi, T. e Rajasekhara Babu, M. (2020). Comprehensive survey of iot communication technologies. In Venkata Krishna, P. e Obaidat, M. S., editores, *Emerging Research in Data Engineering Systems and Computer Communications*, páginas 303–311, Singapore. Springer Singapore.

Rangra, A. e Sehgal, V. (2022). Natural disasters management using social internet of things. *Multimedia Tools and Applications*, páginas 1–15.

Rawat, P., Singh, K., Chaouchi, H., e Bonnin, J.-M. (2013). Wireless sensor networks: A survey on recent developments and potential synergies. *The Journal of Supercomputing*, 68.

Ray, P. (2018). A survey on internet of things architectures. *Journal of King Saud University - Computer and Information Sciences*, 30(3):291–319.

Saint-Andre, P. (2011). Extensible Messaging and Presence Protocol (XMPP): Core. RFC 6120.

Salman, T. e Jain, R. (2017). Networking protocols and standards for internet of things. *Internet of things and data analytics handbook*, páginas 215–238.

Santana, C., Alencar, B., e Prazeres, C. (2018). Microservices: A mapping study for internet of things solutions. In *2018 IEEE 17th International Symposium on Network Computing and Applications (NCA)*, páginas 1–4.

Santana, E. F. Z., Chaves, A. P., Gerosa, M. A., Kon, F., e Milojicic, D. S. (2017). Software platforms for smart cities: Concepts, requirements, challenges, and a unified reference architecture. *ACM Comput. Surv.*, 50(6).

Saqib, M. e Moon, A. H. (2023). A systematic security assessment and review of internet of things in the context of authentication. *Computers & Security*, 125:103053.

Schaffers, H., Komninos, N., Pallot, M., Trousse, B., Nilsson, M., e de Oliveira, Á. (2011). Smart cities and the future internet: Towards cooperation frameworks for open innovation. In *Future Internet Assembly*.

Sethi, P. e Sarangi, S. (2017). Internet of things: Architectures, protocols, and applications. *Journal of Electrical and Computer Engineering*, 2017:1–25.

Shah, S. A., Seker, D. Z., Rathore, M. M., Hameed, S., Ben Yahia, S., e Draheim, D. (2019). Towards disaster resilient smart cities: Can internet of things and big data analytics be the game changers? *IEEE Access*, 7:91885–91903.

Sheth, M., Trivedi, A., Suchak, K., Parmar, K., e Jetpariya, D. (2020). Inventive fire detection utilizing raspberry pi for new age home of smart cities. In *2020 Third International Conference on Smart Systems and Inventive Technology (ICSSIT)*, páginas 724–728.

Sidna, J., Amine, B., Abdallah, N., e El Alami, H. (2020). Analysis and evaluation of communication protocols for iot applications. In *Proceedings of the 13th International Conference on Intelligent Systems: Theories and Applications*, SITA'20, New York, NY, USA. Association for Computing Machinery.

Silva, B. N., Khan, M., e Han, K. (2018). Towards sustainable smart cities: A review of trends, architectures, components, and open challenges in smart cities. *Sustainable Cities and Society*, 38:697–713.

Silva, G. F., Costa, D. G., e Jesus, T. C. (2022). A framework for the development of reconfigurable sensors-based emergencies detection units in smart cities. In *2022 IEEE International Smart Cities Conference (ISC2)*, páginas 1–4.

Silva, J. A. B., Barroso, R. d. C. A., Rodrigues, A. J., Costa, S. S., e Fontana, R. L. M. (2014). Á urbanização no mundo contemporâneo e os problemas ambientais. *Caderno de Graduação - Ciências Humanas e Sociais - UNIT - SERGIPE*, 2(2):197–207.

Silva Júnior, F. P. d. e Chaves, S. V. V. (2021). Desastres naturais no brasil: um estudo acerca dos extremos climáticos nas cidades brasileiras. *Revista da Academia de Ciências do Piauí*, 2(2).

Singh, V. e Peddoju, S. K. (2017). Container-based microservice architecture for cloud applications. In *2017 International Conference on Computing, Communication and Automation (ICCCA)*, páginas 847–852.

Siriwardena, P. (2014). Mutual authentication with tls. In *Advanced API Security*, páginas 47–58. Springer.

Standard, O. (2012). Oasis advanced message queuing protocol (amqp) version 1.0. *International Journal of Aerospace Engineering Hindawi www. hindawi. com*, 2018.

Standard, O. (2019). Mqtt version 5.0. *Retrieved June*, 22:2020.

Suriano, A. L. C. e Reschilian, P. R. (2012). Urbanização, habitação e segregação socioespacial. *Revista Univap*, 18(32):190–202.

Sutar, S. H., Koul, R., e Suryavanshi, R. (2016). Integration of smart phone and iot for development of smart public transportation system. In *2016 international conference on internet of things and applications (IOTA)*, páginas 73–78. IEEE.

Tisdale, H. (1942). The process of urbanization. *Social Forces*, 20(3):311–316.

Uy, N. Q. e Nam, V. H. (2019). A comparison of amqp and mqtt protocols for internet of things. In *2019 6th NAFOSTED Conference on Information and Computer Science (NICS)*, páginas 292–297.

Vakula, D. e Kolli, Y. K. (2017). Low cost smart parking system for smart cities. In *2017 International Conference on Intelligent Sustainable Systems (ICISS)*, páginas 280–284.

Veloz-Cherrez, D., Lozada-Yanez, R., Rodríguez, J., Mayorga, P., e Panchi, J. (2020). Smart waste monitoring system as an initiative to develop a digital territory in riobamba city. *Information*, 11(4).

Yassein, M. B., Shatnawi, M. Q., e Al-zoubi, D. (2016). Application layer protocols for the internet of things: A survey. In *2016 International Conference on Engineering & MIS (ICEMIS)*, páginas 1–4.

Yick, J., Mukherjee, B., e Ghosal, D. (2008). Wireless sensor network survey. *Computer Networks*, 52(12):2292–2330.

Yin, C., Xiong, Z., Chen, H., Wang, J., Cooper, D., e David, B. (2015). A literature survey on smart cities. *Science China Information Sciences*, 58.

Yoo, C. S. (2019). The emerging internet of things.

Zeiner, H., Goller, M., Expósito Jiménez, V., Salmhofer, F., e Haas, W. (2016). Secos: Web of things platform based on a microservices architecture and support of time-awareness. *e & i Elektrotechnik und Informationstechnik*, 133.