Universidade Estadual de Feira de Santana

Programa de Pós-Graduação em Ciência da Computação

Detecção de emergências urbanas utilizando redes de sensores sem fio hierárquicas, colaborativas e configuráveis

Gustavo Araujo Alvaro Coelho

Feira de Santana

2024

Universidade Estadual de Feira de Santana

Programa de Pós-Graduação em Ciência da Computação

Gustavo Araujo Alvaro Coelho

Detecção de emergências urbanas utilizando redes de sensores sem fio hierárquicas, colaborativas e configuráveis

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 - UEFS

Coelho, Gustavo Araujo Alvaro
C616d Detecção de emergências urbanas utilizando redes de sensores sem fio hierárquicas, colaborativas e configuráveis / Gustavo Araujo Alvaro Coelho. - 2024.
95f. : 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. Cidades inteligentes. 2. Detecção de emergência. 3. Internet das coisas. 4. Rede de sensores. 5. Sistema *Fuzzy*. I. Jesus, Thiago Cerqueira de, orient. II. Costa, Daniel Gouveia, coorient. III. Universidade Estadual de Feira de Santana. IV. Título.

CDU: 004.73:711.4

Rejane Maria Rosa Ribeiro – Bibliotecária CRB-5/695

Gustavo Araújo Álvaro Coelho

Detecção de emergências urbanas utilizando redes de sensores sem fio hierárquicas, colaborativas e configuráveis

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, 16 de Fevereiro de 2024

BANCA EXAMINADORA

gov.br Documento assinado digitalmente
THIAGO CERQUEIRA DE JESUS
Data: 16/02/2024 12:47:59-0300
Verifique em https://validar.iti.gov.br

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

gov.br Documento assinado digitalmente
VICTOR ARAUJO FERRAZ
Data: 16/02/2024 11:59:25-0300
Verifique em https://validar.iti.gov.br

Victor Araújo Ferraz
Universidade Federal do Rio Grande do Norte

gov.br Documento assinado digitalmente
MATHEUS GIOVANNI PIRES
Data: 16/02/2024 12:03:42-0300
Verifique em https://validar.iti.gov.br

Matheus Giovanni Pires
Universidade Estadual de Feira de Santana

Abstract

With the growth of the urban population in recent decades, the impacts of structural problems in cities have become increasingly serious and present, among them the occurrence of emergencies, such as fires, floods and earthquakes, is one of the most complex. The occurrence of emergencies is an old urban problem, which over the years has brought numerous negative impacts both on the lives of inhabitants and on the structure of cities. For this reason, it is extremely important to identify the occurrence of emergencies in their initial stages, allowing mitigation actions to be taken as quickly as possible, minimizing their damage. This work aims to develop a hierarchical, collaborative and configurable system, using wireless sensor networks and Fuzzy controller, which automates the detection of urban emergencies in an assertive, energetically efficient way, exploring the computing agility at the edge of the network. Simulations of emergencies in artificial scenarios prove the effectiveness of the system.

Keywords: Smart cities, emergency detection, Internet of Things, sensor network, Fuzzy system.

Resumo

Com o crescimento da população urbana nas últimas décadas, os impactos dos problemas estruturais das cidades vem se tornando cada vez mais graves e presentes, dentre eles a ocorrência de emergências, como incêndios, enchentes e terremotos, se apresenta como um dos mais complexos. A ocorrência de emergências é um antigo problema urbano, que ao longo dos anos vem trazendo inúmeros impactos negativos tanto na vida dos habitantes, quanto à estrutura das cidades. Por esse motivo, é de extrema importância que seja possível identificar a ocorrência de emergências em seus estágios iniciais, permitindo que ações de mitigação sejam tomadas o mais rapidamente possível, minimizando seus danos. Esse trabalho tem como objetivo o desenvolvimento de um sistema hierárquico, colaborativo e configurável, utilizando redes de sensores sem fio e controlador *Fuzzy*, que automatize a detecção de emergências urbanas de forma assertiva, explorando a agilidade de computação na borda da rede. Simulações de emergências em cenários artificiais comprovam a eficácia do sistema.

Palavras-chave: Cidade inteligentes, detecção de emergência, Internet das Coisas, rede de sensores, sistema *Fuzzy*.

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.

Sumário

Alinhamento com a Linha de Pesquisa

Linha de Pesquisa: Software e Sistemas Computacionais

O trabalho apresentado nesse documento consiste na elaboração de um sistema computacional, baseado na utilização de rede de sensores sem fio e na interação entre sistema e usuário. O *framework* em desenvolvimento consiste na implementação de um conjunto de funcionalidades flexíveis e configuráveis, que automatize a detecção de emergências urbanas de forma assertiva, explorando a agilidade de computação na borda da rede. O *framework* se baseia na utilização de uma rede de sensores sem fio heterogênea, hierárquica e colaborativa. Tais características, em conjunto com a alta configurabilidade, permitem que o *framework* seja genérico, capaz de detectar qualquer emergência, desde que esta seja devidamente configurada. Espera-se que a utilização de nós heterogêneos em termos de configuração de *hardware* e funcionalidades, e a definição de diferentes comportamentos para eles, seja capaz de garantir os resultados obtidos pelo modelo *Fuzzy*.

Produções Bibliográficas, Produções Técnicas e Premiações

G. A. A. Coelho, T. C. Jesus and D. G. Costa,"Urban emergency detection system using hierarchical, collaborative and configurable wireless sensor networks", 2023 XIII Brazilian Symposium on Computing Systems Engineering (SBESC), Porto Alegre, Brazil, 2023, pp. 1-6, doi: *10.1109/SBESC60926.2023.10324250*.

Lista de Tabelas

Lista de Figuras

Lista de Abreviações

Abreviação	**Descrição**
RSSF	Rede de Sensores Sem Fio
IoT	Internet das Coisas (*Internet of Things*)
UDE	Unidade de Detecção de Emergência
ZI	Zona de Interesse
BPC	Baixo Poder Computacional
MPC	Médio Poder Computacional
APC	Alto Poder Computacional

Capítulo 1

Introdução

O contexto demográfico urbano vem mudando significativamente nas últimas décadas. Os centros urbanos estão cada vez mais populosos, e a existência de megacidades já é uma realidade. O processo de urbanização traz consigo questões que precisam ser tratadas adequadamente, como a necessidade de fornecimento de energia elétrica, saneamento, água potável, gestão de resíduos, entre outros serviços importantes. À medida que cresce o número de pessoas que vivem nas cidades, aumenta também a dificuldade de lidar com essas questões. Alguns problemas surgem ou são agravados, como a poluição, o tráfego automotivo, os acidentes automobilísticos e o impacto de emergências.

Muitas medidas são tomadas para minimizar os impactos gerados pela urbanização, mas nem todas se mostram realmente eficazes (Pereira, 2014). Com a evolução da tecnologia, as novas soluções englobam diversos elementos tecnológicos, visando automatizar e melhorar a qualidade dos serviços. Esses tipos de soluções também são chamadas de soluções inteligentes e as cidades que as utilizam são comumente denominadas cidades inteligentes (Albino et al., 2015).

Desde que o termo "cidade inteligente" começou a ser utilizado, diferentes definições foram elaboradas. Dentre as várias definições, uma das mais robustas é a que defende que cidades inteligentes são cidades instrumentadas, interconectadas e, claro, inteligentes (Harrison et al., 2010). Nesse conceito, a instrumentação em uma cidade inteligente está diretamente ligada à capacidade de capturar características e elementos definidores do ambiente, processo esse que é normalmente realizado a partir da utilização de sensores. A inteligência associada às cidades se refere à capacidade de, a partir da utilização de dados captados pelos sensores, tomar decisões que guiem a automação das atividades. Essas decisões são tomadas a partir de algoritmos complexos, normalmente baseados em IA, capazes de encontrar padrões em conjunto de dados, e a partir desses padrões, realizar ações específicas. Por fim, a interconexão em uma cidade inteligente está ligada à sua capacidade de compartilhar dados e resultados provenientes do processo de tomada de decisão entre os elementos que

englobam a rede, além de trocar mensagens com outras cidades (Harrison et al., 2010).

O processo de monitoramento presente nas cidades inteligentes está inserido em um contexto com condições específicas. Os sensores precisam trocar mensagens a partir da utilização de políticas de comunicação sem fio, uma vez que é custoso e complexo conectar uma rede com, potencialmente, centenas de nós por meio de cabos. Além disso, os nós sensores precisam ter baixo consumo energético, de modo a aumentar sua vida útil, uma vez que podem ser implantados em locais de difícil acesso a fontes de energia ou mesmo para a realização de manutenção. Neste contexto, as redes de sensores sem fio (RSSF) apresentam-se como uma ótima alternativa a ser utilizada no desenvolvimento das cidades inteligentes, uma vez que provêm ambientes interconectados, instrumentados e passíveis de serem implementados de forma inteligente. Para isso, cada nó sensor pode ser equipado com uma unidade de processamento, um conjunto variado de sensores, um rádio de comunicação sem fio, eventualmente elementos de armazenamento de dados, possibilitando funcionamento autônomo (Abdelgawad e Bayoumi, 2012).

O cenário de cidades inteligentes, portanto, se correlaciona bastante com a utilização de Internet das Coisas (Internet of Things - IoT). O conceito de IoT define uma infraestrutura de objetos em geral que se comunicam entre si e funcionam, em diversas situações, de forma autônoma, sem a necessidade da interferência humana (Rajab e Cinkelr, 2018). A autonomia provida pela estrutura da IoT é determinada pela necessidade do sistema de obter informações do ambiente, e claro, isso é feito a partir da utilização de sensores. Diante disso, é possível afirmar que as características do paradigma IoT e das RSSF estão intimamente conectadas aos conceitos de instrumentação, interconectividade e inteligência, requisitos definidos por Harrison et al. (2010) para o projeto de cidades inteligentes.

Nesse cenário das cidades inteligentes, o tratamento de diversos problemas passa a ser menos complexo e custoso, como por exemplo a detecção de emergências. A utilização de tecnologias como as redes de sensores sem fio e a facilidade de troca de mensagens proporcionada pelo paradigma IoT permite que todo o processo de comunicação e obtenção dos sinais do ambiente sejam feitos de forma mais eficiente. É importante ressaltar que o processo de detecção de emergências é um problema complexo, difícil de ser solucionado a partir da análise isolada dos valores dos sensores. Por esse motivo, diversos trabalhos na literatura vem aplicando inteligência artificial na execução dessa tarefa.

Os trabalhos nessa área normalmente divergem entre si na escolha da técnica de detecção e no tipo de evento que eles se propõem a detectar. Korshikova e Trofimov (2019), por exemplo, utiliza a técnica de regressão linear para a identificação de potenciais problemas em plantas elétricas que podem resultar em incêndios. Já Hashi et al. (2021) implementa redes neurais para a detecção de enchentes. De forma mais geral, Sultan Mahmud et al. (2017) realiza um estudo comparando diversas técnicas de aprendizado de máquina para a detecção de incêndios já em suas fases iniciais.

Diversos trabalhos realizam a detecção de emergências/eventos a partir da utilização de sistemas Fuzzy, tendo como principal elemento diferenciador os tipos de eventos detectados, tais como: incêndios (Kapitanova et al. (2012) e Liang e Wang (2005)), objetos potencialmente perigosos (Liang e Wang (2005) e Khelifi et al. (2014)), fogo em cabos elétricos (Li e Zhou, 2004) e fogo em compartimentos secos ou motores e aeronaves (Foo, 2000). Além disso, há alguns trabalhos cujo diferencial não está no tipo de emergência, nem na técnica de detecção, mas sim na avaliação de diferentes formas de implementação do sistema em si, como é o caso de Dima et al. (2014) que avalia o desempenho de um identificador de eventos variando como os nós distribuem os dados e se comunicam entre si.

É possível identificar uma variedade de pesquisas com o foco na área de detecção de emergências em cidades inteligentes, mas, ao mesmo tempo, é possível verificar algumas limitações em comum entre elas, como a falta de trabalhos que proponham sistemas capazes de detectar múltiplas emergências. A detecção de emergências urbanas pode se caracterizar como uma tarefa de tempo crítico, ou seja, execuções que ultrapassem determinado tempo limite podem causar grandes prejuízos econômicos, danos ambientais, além de expor pessoas a situações de risco (Jesus et al., 2018, 2020). Além disso, para garantir uma detecção precisa, é necessária a coleta de várias informações a partir de diferentes localidades, de forma que elas possam ser correlacionadas. Por isso, é crucial a implementação de um sistema capaz de receber dados georreferenciados, analisá-los e emitir um diagnóstico confiável sobre a ocorrência de uma emergência, antes que ela escale para um desastre. O sucesso de tal sistema, além da qualidade dos sensores e da eficiência dos algoritmos implementados, depende da própria estrutura física e lógica da rede e de suas políticas de comunicação. Por exemplo, coletar e enviar dados brutos (sem pré-processamento) consome muita energia dos nós sensores, diminuindo o tempo de vida da rede, podendo ainda sobrecarregá-la, gerando perda de informação. Enviar todos os dados para um nó sincronizador ou para uma aplicação em nuvem realizar a detecção de emergência pode gerar um atraso significativo, inviabilizando a execução de ações de mitigação em tempo hábil. Assim, a definição das políticas de comunicação utilizadas para a troca de mensagens, a definição do funcionamento dos nós na rede, quais tarefas devem ser realizadas por cada tipo de nó no caso de redes heterogêneas, são elementos importantes para a definição do sistema, uma vez que podem afetar diretamente o seu desempenho.

Diante desse cenário, este trabalho propõe a modelagem e implementação de um sistema genérico (configurável) de detecção de emergências urbanas baseado em redes de sensores sem fios (RSSF). O sistema será capaz de detectar múltiplas emergências, que serão descritas pelo usuário em termos de sua natureza (incêndio, enchente, deslizamento, pandemia, atentado terrorista, etc), das variáveis e grandezas físicas associadas, das regras de correlação dessas variáveis e grandezas, além de aspectos específicos de configuração da RSSF. Essas redes poderão ser heterogêneas, hierárquicas e colaborativas. Ou seja:

- heterogêneas: as unidades de sensoriamento podem não ser necessariamente

idênticas, tanto em sua configuração de *hardware*, quanto em suas atribuições e funcionalidades;

- hierárquicas: as unidades de sensoriamento podem ser categorizadas em níveis, sendo que as unidades de um nível inferior deve atender às requisições das unidades de níveis superiores;
- colaborativas: decisões tomadas pela rede podem ser baseadas em informações globais, provenientes da troca de dados entre diversas unidades de sensoriamento.

O resultado esperado por esse projeto é a implementação de um conjunto de funcionalidades flexíveis e configuráveis, que automatize a detecção de emergências urbanas de forma assertiva. Além disso, assumindo que o processamento na borda da rede é menos custoso que a transferência de dados, tentamos criar um modelo energeticamente eficiente, dando prioridade a exploração da agilidade de computação na borda da rede em detrimento ao tráfego na rede. Essas funcionalidades implementadas deverão ser configuradas pelo usuário a fim de gerar um sistema de detecção de emergências urbanas apropriado para as peculiaridades da cidade ou região onde se deseja implantá-lo.

1.1 Objetivos

O objetivo desse trabalho é desenvolver um *framework*, altamente configurável, capaz de realizar a detecção de emergências. A configurabilidade deve permitir ao usuário definir parâmetros essenciais para a execução do processo, como as emergências a serem detectadas e as variáveis que as caracterizam, além dos valores de *threshold* e regras aplicadas por um sistema *Fuzzy*.

O *framework* se baseia na utilização de uma rede de sensores sem fio heterogênea, hierárquica e colaborativa. Tais características, em conjunto com a alta configurabilidade, permitem que o *framework* seja genérico, capaz de detectar qualquer emergência, desde que esta seja devidamente configurada. A detecção é implementada zelando pela agilidade e precisão da detecção, por meio da diminuição da sobrecarga da rede.

Visando atingir o objetivo geral, alguns objetivos específicos foram traçados:

- Configurabilidade de emergências em termos de sua natureza;
- Configurabilidade de emergências em termos das variáveis que as compõem;
- Configurabilidade das regras de correlação das variáveis (Sistema Fuzzy);
- Configurabilidade de aspectos específicos da RSSF;
- Definição das tarefas de cada nó a partir da abordagem hierárquica e colaborativa;

- Implementação do sistema Fuzzy a partir das regras de correlação definidas.

Espera-se que a utilização de nós heterogêneos em termos de configuração de *hardware* e funcionalidades, e a definição de diferentes comportamentos para eles, seja capaz de garantir os resultados obtidos pelo modelo *Fuzzy*, que deverá identificar eventos de difícil caracterização e alto grau de incerteza.

1.2 Contribuições

As principais contribuições deste trabalho são:

- Modelagem abstrata de emergências urbanas;
- Modelo hierárquico e colaborativo para detecção de emergências urbanas;
- Modelagem e implementação de um framework genérico para a detecção de múltiplas emergências.

1.3 Organização do Trabalho

No Capítulo 2, os trabalhos relacionados são apresentados, sendo apontados os pontos positivos de cada um deles, além de lacunas que serão preenchidas com a solução proposta. O Capítulo 3 aborda os principais elementos teóricos necessários para o entendimento e a elaboração do framework de detecção de emergências. No Capítulo 4 são descritos detalhadamente os passos tomados para a implementação do framework, como as políticas de comunicação, modo de configurabilidade do sistema e implementação do sistema *Fuzzy*. No Capítulo 5 são apresentados e discutidos os resultados obtidos a partir da execução dos testes modulares e de integração. Por fim, no Capítulo 6 são apresentadas as conclusões finais do trabalho.

Capítulo 2

Revisão Bibliográfica

Nessa seção serão detalhados os pontos positivos e negativos de trabalhos da literatura que abordam diferentes perspectivas para realizar a detecção de emergências, como detecção baseada em sensores, detecção colaborativa e uso de inteligência artificial. Esse é um campo muito amplo de pesquisa, por esse motivo diversos tipos de emergência e técnicas de detecção estão presentes na literatura, como é apresentado nesta seção.

Um dos trabalhos nessa área é o de Hashi et al. (2021). Nele o autor aborda o desenvolvimento de um sistema de detecção de eventos, focado especificamente em enchentes. O trabalho pode ser dividido em etapas, onde a primeira consiste na busca de informações a partir da observação do rio e da coleta de dados com a população ribeirinha sobre informações da região que será monitorada. Essas informações servem como guia para que seja possível definir o melhor posicionamento para a unidade de sensoriamento. A segunda etapa consiste na fase de implementação. Nessa fase, a unidade de sensoriamento, programada com algoritmos de inteligência artificial, é posicionada em um ponto estratégico de um rio. Visando comparar os resultados de diferentes algoritmos, foram definidos quatro implementações: rede neural convolucional, algoritmo *Random Forest*, *Naive Bayes* e J48. A partir da análise dos testes e observação dos resultados, o algoritmo que obteve maior precisão durante o processo de detecção de enchentes foi o *Random Forest*. Esse trabalho apresenta pontos interessantes, como a busca por informações iniciais para direcionar o posicionamento do sensor na área monitorada. Embora os algoritmos de detecção utilizados se mostrem eficientes, eles são específicos para um tipo de emergência, restringindo o sistema. Além disso, o fato de ter apenas uma unidade de sensoriamento faz com que a detecção seja local, portanto, limitada.

Outro trabalho relevante é o de Kapitanova et al. (2012), no qual é realizada uma análise da precisão de um sistema de detecção de incêndios a partir da utilização de redes de sensores sem fio e lógica *Fuzzy*. O modelo *Fuzzy* é desenvolvido para funcionar a partir da análise de quatro variáveis representando fatores de risco, que são a temperatura e a sua variação, além do nível de fumaça e sua variação. A

utilização da variação dos sinais como variável de entrada permite a análise do comportamento não somente a partir dos valores instantâneos, mas também a partir de comportamentos anteriores, trazendo maior robustez ao modelo. Além disso, os autores utilizam uma abordagem colaborativa, realizando o processamento considerando informações provenientes de nós vizinhos. Os nós mais próximos da fonte do sinal são considerados mais confiáveis, e por esse motivo, possuem maior peso no processamento de dados. A utilização da variação dos sinais no tempo e espaço (abordagem colaborativa), além da distinção de importância dos nós, culminou na obtenção de resultados mais precisos. No entanto, o artigo não aborda detalhes da implementação da rede de sensores sem fio, tampouco as políticas de comunicação e organização da rede, o que leva a crer que todas as medições de sensores são transmitidas na rede em *convergecast*, ou seja, direcionadas a um *gateway* (*sink*) específico. Esse modelo de comunicação pode não ser o ideal, uma vez que, com um alto volume de dados trafegados, pode gerar sobrecarga na rede.

Também utilizando lógica *Fuzzy*, Khelifi et al. (2014) desenvolveu um sistema capaz de identificar a existência de objetos possivelmente perigosos em uma determinada área. O processo de identificação é baseado na utilização de redes de sensores sem fio, onde cada nó possui um conjunto de sensores escalares e visuais. A partir da utilização de um modelo *Fuzzy* os nós identificam a presença ou não de um objeto na região monitorada. O sistema utiliza como variáveis de entrada dados obtidos de sensores infravermelho, ultra-sônico e acústico, além de imagens da área e informações sobre o estado dos nós sensores dentro da RSSF, como o estado do *buffer* de dados e de energia. Esses dois últimos elementos se apresentam como elementos de extrema importância, pois a depender dos seus valores os resultados obtidos pelos sensores podem ser considerados adulterados ou invalidados, devido a possível atraso na disponibilidade dos dados. Além disso, o sistema é capaz de identificar qual nó tem maior valor de confiabilidade, e faz com que esse nó ative a câmera acoplada e tire fotos para confirmar a detecção do objeto. No entanto, mesmo com diversos elementos bem estabelecidos, esse trabalho não aborda o efeito que a estrutura física da rede, assim como suas configurações e políticas de comunicação, exercem sobre a eficiência e acurácia do sistema de detecção de emergências. Outro aspecto a ser ressaltado é que o trabalho utiliza dados limitados a apenas um tipo de emergência.

Em Nazir et al. (2022) os autores apresentam um trabalho que tem como objetivo principal a descoberta dos melhores valores de limiar para detecção de incêndios. As variáveis utilizadas são os valores de temperatura, umidade, intensidade de fumaça, e quantidade de gases como CO^2 e nitrogênio. O autor afirma que a utilização de limiares mais precisos auxilia na precisão de sistemas baseados em inteligência artificial. O desenvolvimento do trabalho tem como etapa inicial a montagem de uma estrutura capaz de obter os valores em tempo real das variáveis monitoradas, além de adicionar um detector de incêndio comercial e avaliar a condição do ambiente no momento em que o incêndio era detectado, para que a partir daí decisões pudessem ser tomadas. Os autores apresentam um detalhamento interessante da forma como

os sensores se comunicam entre si e com a estrutura de controle. Os experimentos realizados consistem na criação de "mini-incêndios" em ambiente controlado, onde foram utilizados diferentes formas de originar o fogo, desde queimar roupas até queimar papel. Dentre os resultados deste trabalho foram observados comportamentos e correlações entre os dados que permitiram a escolha de limiares válidos para a detecção de incêndios. Entretanto, os resultados podem ter sido influenciados pelo pequeno número de experimentos realizados (apenas oito). Como o trabalho se propõe a validar a escolha dos valores de limiar para detecção de incêndios, ele ainda carece de uma maior quantidade de experimentos, além de experimentos mais realísticos. Esse é um problema comum em sistemas de detecção de emergências, já que é muito difícil e custoso simular uma emergência de forma realista e segura.

A pesquisa apresentada por Dima et al. (2014) aborda o desenvolvimento de um sistema de identificação de eventos focado no monitoramento de pessoas e suas questões de saúde. É utilizado um sistema baseado em lógica *Fuzzy* para analisar os dados e monitorá-los em busca de eventos como quedas, por exemplo. O ponto mais relevante do trabalho consiste na proposta de realizar a redistribuição das tarefas entre os nós, alterando o paradigma comum de definir um nó central que processa todos os dados. O autor propõe que determinados nós locais também façam o processamento de dados, culminando na menor quantidade de dados transmitidos pela rede, e consequentemente na diminuição da probabilidade de colisão de dados e, portanto, da perda de informações. Para a realização desse processo foi necessária a elaboração de técnicas de roteamento que distribuíssem as tarefas para otimizar o processo de troca de informações.

O modelo de detecção aqui proposto utiliza elementos positivos apresentados nos artigos anteriores como a utilização de modelos *Fuzzy* e a utilização de processamento na borda da rede, diminuindo a transferência de dados e, consequentemente, o consumo energético e as possíveis colisões de dados. Além disso, o modelo deste trabalho visa preencher algumas lacunas presentes nos trabalhos citados, como a falta de uma abordagem apropriada que considere o impacto da estrutura física da rede em relação à eficiência e acurácia do sistema de detecção de emergências. Nesse aspecto engloba-se desde a configuração heterogênea do *hardware* dos nós até a topologia da rede, assim como suas configurações e políticas de comunicação. Dessa forma, busca-se obter como resultado um modelo capaz de caracterizar todos os aspectos essenciais da rede, e também capaz de detectar múltiplas emergências a partir da configurabilidade de variáveis importantes do sistema como *thresholds*, regras *Fuzzy* e sensores ativos em cada nó.

Capítulo 3

Fundamentação Teórica

Nessa seção serão abordados detalhadamente os elementos teóricos que compõem a elaboração do modelo de detecção de emergências proposto neste trabalho. Primeiramente é feita uma revisão sobre o conceito de internet das coisas e rede de sensores sem fio, e como elas estão diretamente conectadas. Em seguida, é abordado o conceito de emergências e o seu impacto no ambiente urbano, além da importância do processo de detecção de emergências para a diminuição do impacto das mesmas. Por último, é feito um apanhado detalhado sobre inteligência artificial, mais especificamente de sistemas baseados em lógica *Fuzzy*.

3.1 Rede de sensores sem fio e internet das coisas

As redes sem fio são um tópico de pesquisa que vem recebendo atenção há décadas, sendo que apenas no final da década de 80 que o conceito rede de sensores sem fio (RSSF) começou a ser desenvolvido e utilizado. As RSSF conectam nós sensores a partir da utilização de políticas de comunicação sem fio, facilitando, a partir da proveniência de cobertura de largas áreas geográficas, o desenvolvimento de sistemas distribuídos para a coleta de dados (Manrique et al., 2016). Redes de sensores sem fio estão presentes nas mais diversas áreas do conhecimento, algumas das mais utilizadas são a de vigilância ambiental, construções inteligentes, monitoramento de saúde e transportes inteligentes (Wang et al., 2008).

A topologia e a política de comunicação das redes de sensores sem fio podem variar a depender da aplicação a ser desenvolvida, no entanto, é essencial que alguns requisitos básicos sejam cumpridos. Esses requisitos representam algumas características importantes, como a robustez, escalabilidade, qualidade de serviço (QoS), heterogeneidade, autonomia e gestão de consumo de energia. É interessante observar que essas mesmas características também se apresentam como elementos de imensa importância para a implantação da infraestrutura da internet das coisas (*Internet of Things* — IoT).

O conceito de internet das coisas foi proposto inicialmente por Ashton (1999), que o definiu como um sistema onde o mundo físico se comunica com a internet a partir da utilização de identificadores de rádio frequência. No entanto, a definição do termo ainda está em construção, tendo diversos autores contribuído com diferentes definições a partir de suas perspectivas individuais. Uma delas é a de que internet das coisas é uma infraestrutura de rede global e dinâmica, capaz de se autoconfigurar a partir de padrões e políticas de comunicação. Além disso, os dispositivos físicos e virtuais da rede seriam capazes de utilizar interfaces inteligentes para se integrarem como uma rede de informações (Li et al., 2012).

De forma geral, pode ser considerada uma infraestrutura capaz de interconectar diferentes dispositivos físicos a partir da internet, permitindo que eles acumulem e troquem dados entre si (Rajab e Cinkler, 2018). Esses dispositivos podem variar desde objetos comumente utilizados no dia-a-dia, como cafeteiras, ar-condicionados e televisões, até sensores inteligentes implantados em áreas isoladas, cujo objetivo é obter informações sensíveis do ambiente. A partir dessa infraestrutura, aplicações capazes de automatizar tarefas diárias e de infraestrutura nas cidades são possíveis de serem desenvolvidas.

Diante dessas definições, é possível perceber como a rede de sensores sem fio se apresenta como um elemento complementar à infraestrutura de IoT, pois é a partir de uma RSSF que os dispositivos se conectam entre si, permitindo a troca de informações e a realização de tarefas de forma autônoma pelos dispositivos. A utilização de IoT visando a melhoria de serviços e da qualidade de vida da população nos centros urbanos, são elementos definidores das populares cidades inteligentes.

3.2 Protocolo MQTT

O MQTT foi criado pela IBM como um protocolo de comunicação leve, com pouco consumo de rede e baseado no modelo de comunicação *publish-subscriber* (Dinculeană e Cheng, 2019). O baixo consumo de rede e de recursos por parte do usuário faz deste protocolo uma das melhores opções para ambientes com baixa latência de rede, como cenários de aplicações IoT (Nastase, 2017).

O modelo de comunicação *publish-subscriber* é um dos principais motivos do baixo consumo de rede. Esse modelo não precisa que o usuário solicite informações, tornando ele leve o suficiente para ser um dos protocolos mais indicados para a utilização em microcontroladores presentes em RSSF (Light, 2017). O MQTT se baseia na utilização de tópicos, onde cada mensagem deve estar vinculada a um tópico na hora da sua publicação. Nesse contexto, cada dispositivo pode se cadastrar como inscrito (*subscribers*) ou como publicador (*publishers*). Dispositivos que se cadastrem como publicadores podem publicar mensagens naquele tópico, enquanto os inscritos são notificados e possuem acesso a cada mensagem publicada naquele determinado tópico.

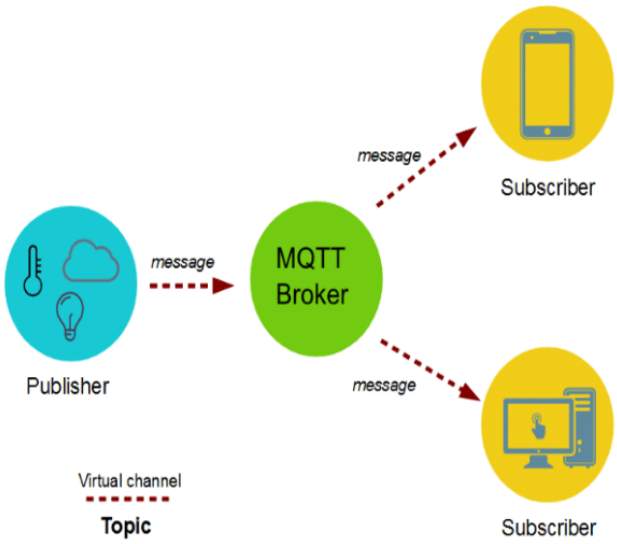


Figura 3.1: Representação gráfica da comunicação utilizando o protocolo MQTT (Ramelan et al., 2021).

A Figura 3.1 ilustra o processo de troca de mensagens baseado na utilização do protocolo MQTT. Ela ilustra a comunicação a partir da utilização de um tópico, onde dois dispositivos estão cadastrados como *subscribers* e um como *publisher*. O dispositivo cadastrado como *publisher* faz a publicação de uma mensagem por meio do tópico, essa mensagem é enviada para o *Broker* que a redireciona para os dispositivos inscritos. Importante dizer que os dispositivos podem variar desde computadores robustos até microcontroladores com baixo poder computacional, por exemplo.

3.3 Cidades Inteligentes

Os centros urbanos apresentam em sua essência grandes problemáticas no quesito de distribuição de recursos e serviços para a população. Transporte, água potável, energia elétrica e atendimento médico são alguns dos serviços e recursos que não estão disponíveis de forma igualitária para todos. Diante do crescimento contínuo da população urbana nos últimos anos, os problemas intrínsecos ao ambiente urbano passam a se intensificar, afetando cada vez mais negativamente a qualidade de vida da população. Diante desse cenário, a idealização de cidades inteligentes surge como uma alternativa para atenuar e, possivelmente, resolver essa gama de problemas.

O termo cidade inteligente ainda está em construção, tendo vários autores contribuído de formas diferentes em busca de sua definição final. Uma delas define cidade inteligente como as cidades que funcionam de forma sustentável e inteligente, enquanto incorporam uma gama de serviços que utilizam dispositivos inteligentes para garantir a eficiência, aumentar o conforto e qualidade de vida dos cidadãos (Hernafi

et al., 2017). Outra definição é a de Balakrishna (2012), que considera inteligentes as cidades onde os investimentos em capital humano e social, além de infraestrutura de comunicação, abastecem o desenvolvimento econômico sustentável e uma alta qualidade de vida.

O conceito de cidade abrange diversos microelementos que, em conjunto, formam o ambiente urbano e definem o seu funcionamento. Quando se fala em cidades inteligentes não é diferente. Sendo assim, cidades inteligentes podem ser divididas em dimensões menores para as quais as aplicações inteligentes podem ser desenvolvidas (Kirimtat et al., 2020; Balakrishna, 2012):

- Pessoas inteligentes: o termo inteligente não define aqui o grau de escolaridade ou capacidade intelectual dos cidadãos, mas sim a qualidade das interações sociais entre os indivíduos.
- Economia inteligente: engloba fatores sobre economia competitiva, como inovação, competitividade e flexibilização do trabalho.
- Governo inteligente: aborda aspectos da participação política e serviços aos cidadãos.
- Transporte inteligente: envolve aspectos de acessibilidade e disponibilidade de tecnologias de comunicação e sistemas de transporte sustentáveis.
- Gestão de recursos inteligente: referente à qualidade do ar, espaços verdes, manutenção da natureza e utilização de recursos naturais como mínimo desperdício.
- Vivência inteligente: abrange diversos elementos representativos da qualidade de vida, como cultura, saúde, segurança e moradia.

Uma vez definidas as dimensões de atuação das aplicações inteligentes, é importante definir os requisitos básicos que uma cidade inteligente precisa ter para ser possível a implementação de suas aplicações. Segundo Balakrishna (2012) existem três requisitos básicos. O primeiro deles é a necessidade de implantação de instrumentação em larga escala, envolvendo a distribuição de sensores, TAGs e outros elementos sensoriais pela região. As aplicações inteligentes se baseiam na utilização massiva desses dispositivos, uma vez que são os responsáveis por captar os sinais do ambiente, permitindo a análise do seu comportamento. É a partir dos sinais obtidos por esses dispositivos que os objetos inteligentes realizam o processamento e realizam suas ações. Essas ações podem variar desde ligar o ar-condicionado automaticamente uma vez que se detecte aumento de temperatura na residência, até enviar notificações para as autoridades em caso de detecção de elevação do nível de um rio.

Toda essa estrutura de sensores e dispositivos precisa ser conectada com uma infraestrutura de rede escalável e de alta velocidade. A escalabilidade é de extrema importância, porque os dispositivos envolvidos se organizam de forma dinâmica, onde, a qualquer momento, algum deles pode ser adicionado ou retirado da rede a depender da necessidade. Já a velocidade é essencial para que a imensa quantidade

de dados que flui pela rede não gere latência, congestionamento ou perda de dados. Esses dois elementos em conjunto compõem o segundo requisito básico para as cidades inteligentes.

O terceiro requisito envolve a gestão das informações obtidas pelos dispositivos sensores. O acesso organizado e bem estabelecido à quantidade massiva de dados disponibilizada pelas seis dimensões das cidades inteligentes permite que seja obtida inteligência. Essa inteligência permite que as aplicações e serviços *smart* sejam desenvolvidos e executados da forma mais eficiente e precisa possível. A Figura 3.2 ilustra as seis dimensões das cidades inteligentes e como as aplicações que atuam sobre elas necessitam da existência dos três requisitos básicos citados anteriormente.

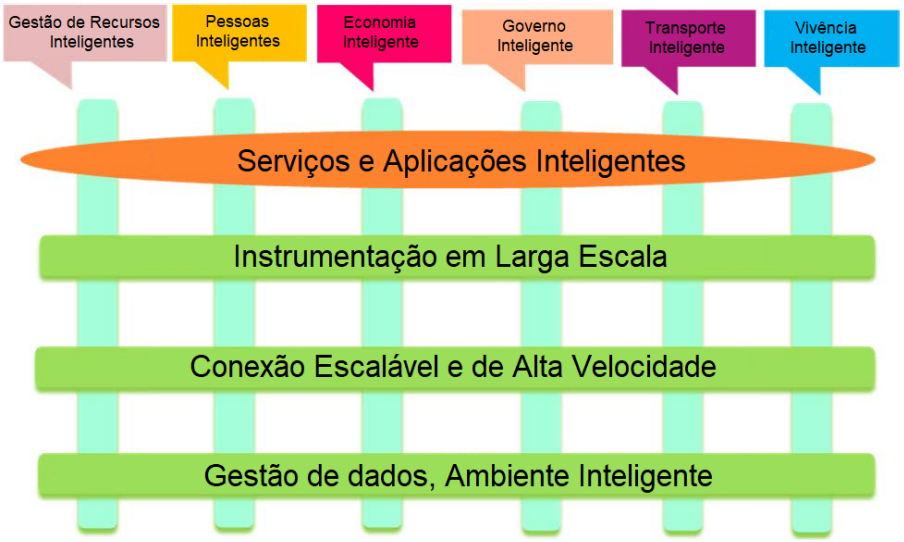


Figura 3.2: Representação gráfica da interconexão entre os requisitos básicos das cidades inteligentes e suas micro dimensões (Balakrishna, 2012)

Considerando todos os elementos que compõem as cidades inteligentes, tanto suas dimensões, quanto seus requisitos, é possível verificar como suas características remetem à utilização de IoT e redes de sensores sem fio. Nesse contexto, diversos pesquisadores vêm tentando definir boas práticas de desenvolvimento de serviços inteligentes visando melhorar problemas de desenvolvimento, como sustentabilidade, responsabilidade social e consumo de energia (Costa et al., 2022). Dessa forma, diversas iniciativas são propostas visando melhorar a qualidade de vida dos cidadãos, melhorando serviços como transporte público, fornecimento de água e energia elétrica, sistemas de gerenciamento de emergências, entre outros. A gestão de sistema de emergências é um dos principais tópicos de desenvolvimento de aplicações inteligentes, e é o foco deste trabalho.

3.4 Emergências

Dentre as diversas problemáticas intrínsecas ao desenvolvimento urbano, o impacto de emergências está dentre os mais complexos e importantes. Sua ocorrência é um problema antigo e recorrente nas cidades, e à medida que a densidade populacional aumenta, o impacto das emergências também aumenta (Kontokosta e Malik, 2018). As características da formação e desenvolvimentos das cidades ao longo dos séculos, como a distribuição espacial, falta de saneamento básico, dominância de prédios de madeira e a falta de equipes de resgate à emergência, propiciaram maior facilidade para a ocorrência de emergências (Huang et al., 2021). Esses elementos fizeram com que as emergências se tornassem um dos problemas mais importantes de se lidar.

Não existe um consenso sobre a definição exata do que são emergências, mas elas podem ser consideradas como o início de situações perigosas, que podem, se não forem aplicadas ações de retenção de danos eficientes, se tornar um desastre. O desastre é a pior consequência de uma emergência, quando já não é possível evitar as consequências negativas da mesma (Costa et al., 2022).

Com o desenvolvimento da tecnologia, da industrialização, redes de telefonia e veículos a motor, a gestão de emergências nos centros urbanos passou a ser mais eficiente. No entanto, para que ações de contenção de danos possam ser tomadas ainda é necessária a realização de chamadas de emergência para que então, a partir de um método não automatizado, o envio de equipes de emergência seja realizado (Tebeau, 2012). Quando se fala em tempo de resposta das ações para a mitigação dos impactos de emergências, cada segundo é valioso, influenciando diretamente no impacto e na quantidade de pessoas afetadas. Por esse motivo a detecção nos seus momentos iniciais (menor latência possível) é de extrema importância.

3.5 Detecção de emergências

Toda emergência possui como elemento gerador um ou mais fatores de risco. Os fatores de risco são definidos como qualquer elemento com potencial de gerar danos, uma vez que ele atinja o nível de emergência (Gasparini et al., 2014). Podem ser utilizados como exemplos temperaturas muito elevadas que podem evoluir para incêndios, nível da água muito alto, podendo gerar enchentes e ventos muito fortes que podem resultar em furacões. O processo de detecção de emergências se baseia em, além de analisar informações provenientes de fontes como *Big Data*, monitorar ao longo do tempo variáveis que caracterizam o ambiente. Dessa forma, é possível detectar variações de comportamento, permitindo a identificação dos fatores de risco que sinalizem a existência ou não de uma emergência naquela região (Zhou et al., 2018; Ortuño et al., 2013; Abdelgawad e Abdulhai, 2009).

Algumas emergências podem ser definidas a partir de um único fator de risco (emergências autônomas), como é caso de enchentes e alagamentos, já outras, possuem comportamento mais complexo (emergências baseadas em eventos), e por isso é pre-

ciso uma análise conjunta dos diversos fatores que a compõe para identificar sua existência. Diante de sua maior complexidade, as emergências baseadas em eventos são divididas em dois grupos: as emergências acionadas e as emergências agregadas (Costa et al., 2022).

- Emergências acionadas: são detectadas a partir da utilização de valores gatilhos, também chamados limiares. Esse método consiste na definição de valores específicos para cada fator de risco que compõe a emergência sendo analisada. Se os valores obtidos nos sinais forem maiores (ou menores) do que os limiares, então a detecção é dada como positiva. A detecção por meio dos limiares apresenta resultados rápidos, mas, ao mesmo tempo, costuma apresentar falsos positivos e/ou negativos como resultados.
- Emergências agregadas: são identificadas a partir de uma análise conjunta dos diversos fatores de risco que a envolvem. Para esse tipo de análise são normalmente utilizadas técnicas de inteligência artificial capazes de analisar diversas variáveis e quantificar a relação entre elas. A partir disso, é possível definir se aquele conjunto de variáveis representa a existência de uma emergência ou não. A utilização de algoritmos mais robustos traz maior confiabilidade nos resultados desse tipo de detecção.

Independente do tipo de emergência e dos fatores de risco analisados, o ciclo de detecção visa detectar emergências em seus estados iniciais, de modo que seja possível que ações de contenção de danos sejam executadas antes que a emergência se torne um desastre. A figura 3.3 ilustra esse ciclo de detecção de emergências citado.

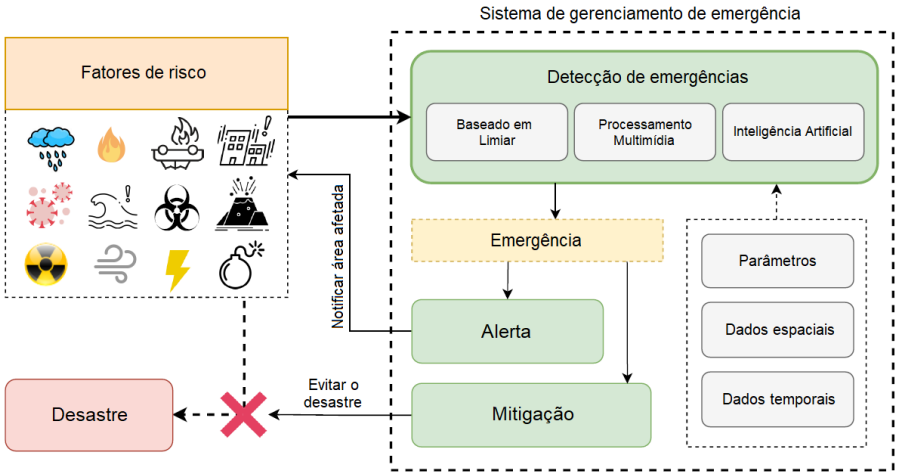


Figura 3.3: Esquema ilustrando os elementos que compõem o ciclo de detecção de emergências e suas relações (Costa et al., 2022)

Na estrutura definida no ciclo de detecção de emergência, as unidades de detecção de emergências (UDE) se apresentam como um de seus principais elementos. Essas unidades podem ser definidas na borda da rede ou mesmo em computadores centrais, mas independente disso, possuem uma série de comportamentos padrão, como a

capacidade de receber os dados de entrada, aplicar um ou mais algoritmos sobre eles e retornar o resultado desejado. As UDEs podem ser implementadas de formas diferentes a depender dos dados disponíveis e dos fatores de risco que estão sendo monitorados, podendo resultar em sistemas de diferentes custos e complexidades (Costa et al., 2022).

A detecção de elementos críticos em uma cidade requer um alto grau de acurácia nos resultados, uma vez que falsos positivos ou negativos podem gerar graves danos à integridade física da população, além de prejuízos financeiros e estruturais. As emergências estão dentro desse conjunto de elementos críticos, por esse motivo, por mais que tenha resultados rápidos, a quantidade de falsos positivos e/ou negativos provenientes da detecção baseada em limiares é um problema. Diante desse cenário surge uma forma de detecção de emergências mais robusta, os métodos utilizando inteligência artificial.

A utilização de inteligência artificial nesses sistemas retira a inflexibilidade provida pelos limiares, além de englobar cenários que necessitam de mais informações do que somente o fator de risco ser maior (ou menor) que determinado valor. Cenários de ambientes esfumaçados, acabam sendo imediatamente identificados como incêndio por sistemas de detecção usando limiares, no entanto, esse tipo de abordagem não é capaz de identificar se o alto grau de fumaça é de fato devido a um incêndio ou a fogos de artifícios acendidos na região, por exemplo. Diante desse cenário, tomar as decisões a partir de mais informações se apresenta como um elemento de grande importância para ser possível realizar essa diferenciação, e os algoritmos de inteligência artificial permitem essa abordagem (Wu et al., 2021).

3.6 Inteligência artificial — Lógica *Fuzzy*

Algoritmos tradicionais, de forma geral, executam exatamente o que eles estão programados para fazer. No entanto, com o passar do tempo e a evolução da tecnologia, passou-se a desejar que os sistemas computacionais conseguissem resolver problemas cada vez mais complexos. Problemas que não poderiam ser resolvidos a partir da utilização de algoritmos comuns. Diante disso, buscando resolver esses tipos de problemas, a modelagem de comportamentos biológicos em sistemas computacionais foi uma das áreas de estudo que mais cresceram, resultando nos chamados sistemas inteligentes (Engelbrecht, 2007).

A adição de lógica, raciocínio dedutivo, sistemas especialistas e sistemas simbólicos de aprendizado de máquina aos algoritmos inteligentes compõe a área da inteligência artificial. Esse campo permitiu o desenvolvimento de algoritmos capazes de resolver problemas cada vez mais complexos, a partir de processos de aprendizado atribuídos a eles. O aprendizado incluído na estrutura desses algoritmos deu flexibilidade para que não executassem somente o que estava diretamente programado, permitindo que fossem capazes de analisar dados e a partir deles tomar decisões. Dentre os

algoritmos inteligentes podem ser citadas as redes neurais, os algoritmos evolutivos e os modelos *Fuzzy* (Engelbrecht, 2007).

Modelos baseados em lógica *Fuzzy* representam um dentre os vários algoritmos inteligentes que compõem o amplo campo da inteligência artificial. Esse tipo de sistema possui uma característica peculiar quanto a teoria de pertencimento a conjuntos. A teoria tradicional dos conjuntos, utilizada pelos algoritmos comuns, define se um elemento faz parte de um conjunto ou não a partir de uma atribuição binário de 1 ou 0, pertence ou não pertence, sem meio-termo. A avaliação humana, no entanto, não trabalha com a ideia de pertencimento binário, e é esse tipo de comportamento que os modelos *Fuzzy* buscam representar. Essa característica faz dele um ótimo candidato para o processamento de dados em cenários de alta incerteza.(Driankov et al., 2013)

Na lógica *Fuzzy* um valor pode pertencer a mais de um conjunto simultaneamente, contendo diferentes graus de pertencimento (Zimmermann, 2010). O valor representando o quanto a variável de entrada pertence a cada conjunto é definido a partir do processo chamado de fuzzificação. Considerando que a temperatura pode ser definida a partir dos conjuntos *Baixa*, *Média* e *Alta*, cada uma delas possui uma função de pertencimento que define seu comportamento. Dessa forma, para cada valor de temperatura existe um valor correspondente nessas funções definindo o quanto aquele valor pertence ao conjunto em questão. Os valores de pertencimento estão sempre definidos no intervalo $[0, 1]$ (Tanscheit, 2004; Driankov et al., 2013).

A Figura 3.4 ilustra possíveis funções de pertencimento para os conjuntos citados anteriormente. Tomando como exemplo o valor de entrada de 27^{o}C, pode-se considerar que ele tem grau de pertencimento 0.85 para *Baixa*, 0.15 para *Média* e 0 para *Alta*. Diante dessa distribuição é possível definir que esse valor de temperatura é considerado como uma temperatura majoritariamente baixa.

As funções de pertencimento podem ser definidas a partir da utilização de diferentes funções-base, como, por exemplo, as funções triangulares e gaussianas (Figura 3.5).

Uma vez realizado o mapeamento dos valores de entrada para seus respectivos graus de pertencimento, o modelo *Fuzzy* utiliza esses valores para obter o resultado desejado por meio da inferência. Esse resultado é obtido a partir da análise de diversas regras pré-definidas. Considerando um modelo *Fuzzy* cujo objetivo é determinar, a partir da temperatura e da quantidade de fumaça no ambiente, a probabilidade de ocorrência de um incêndio (Muito Baixa, Baixa, Alta e Muito Alta), algumas das regras possíveis são:

- *Se temperatura é Alta e a quantidade de fumaça é Muita, então a probabilidade de incêndio é Muito Alta.*
- *Se temperatura é Média e a quantidade de fumaça é Muita, então a probabilidade de incêndio é Alta.*
- *Se temperatura é Baixa e a quantidade de fumaça é Pouca, então a probabilidade de incêndio é Muito Baixa.*

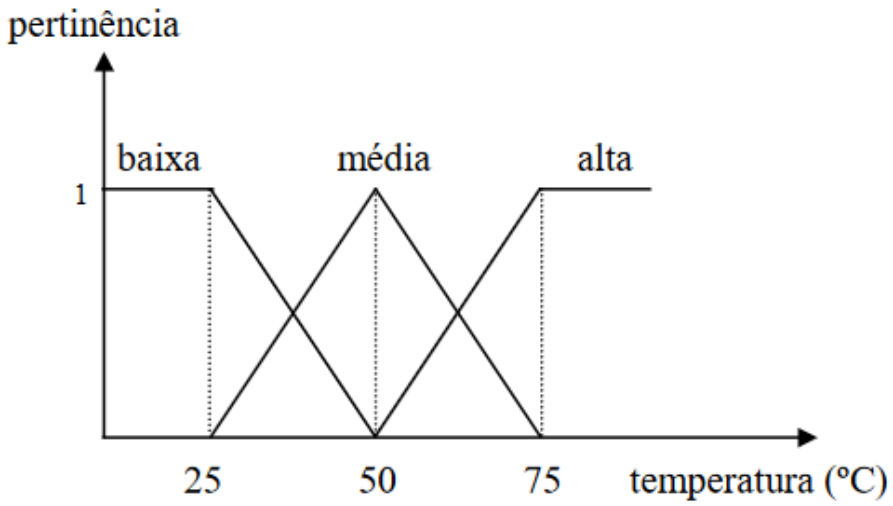
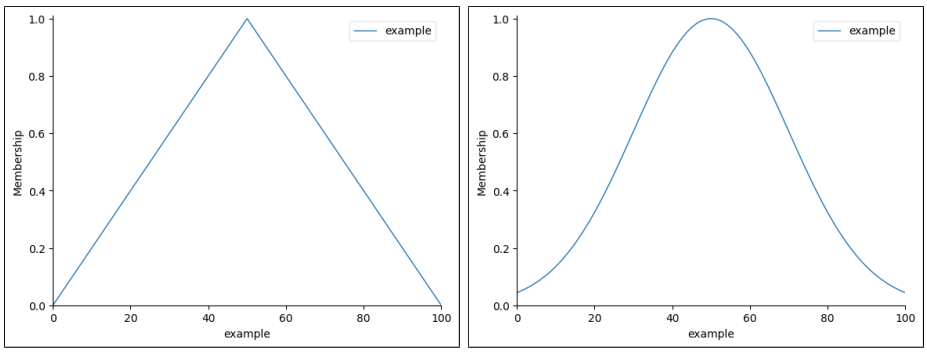


Figura 3.4: Representação de possíveis funções de pertencimento para valores de temperatura (Tanscheit, 2004).



(a) Triangular (b) Gaussiana

Figura 3.5: Exemplos de funções-base para a criação das funções de pertencimento: a) função triangular e b) função Gaussiana.

A medida que o número de variáveis cresce, a quantidade de regras necessárias para o funcionamento do sistema cresce proporcionalmente. Além disso, é importante ressaltar que a definição das regras necessita da participação de especialistas na área de aplicação do sistema. Pessoas capazes de relacionar os graus de intensidade das variáveis de entrada com os possíveis resultados de saída, permitindo que as regras sejam de fato representações da realidade.

Uma vez aplicadas as regras e obtido o resultado desejado, nesse caso exemplificado a probabilidade de ocorrência de incêndios naquela área, é necessário a transformação desse resultado para valores numéricos. Para seres humanos a resposta da probabilidade de ocorrência de um incêndio ser 'Alta' ou 'Baixa' faz sentido e é possível de ser analisada, no entanto, esses valores não fazem sentido quando inseridos em sistemas computacionais. Por esse motivo é necessária a realização da *deffuzificação* (Driankov et al., 2013), que consiste no processo de, a partir do resultado obtido, gerar um valor numérico que o represente de forma fidedigna.

A Figura 3.6 ilustra todos os passos da execução de um modelo *Fuzzy* citados nos parágrafos anteriores.

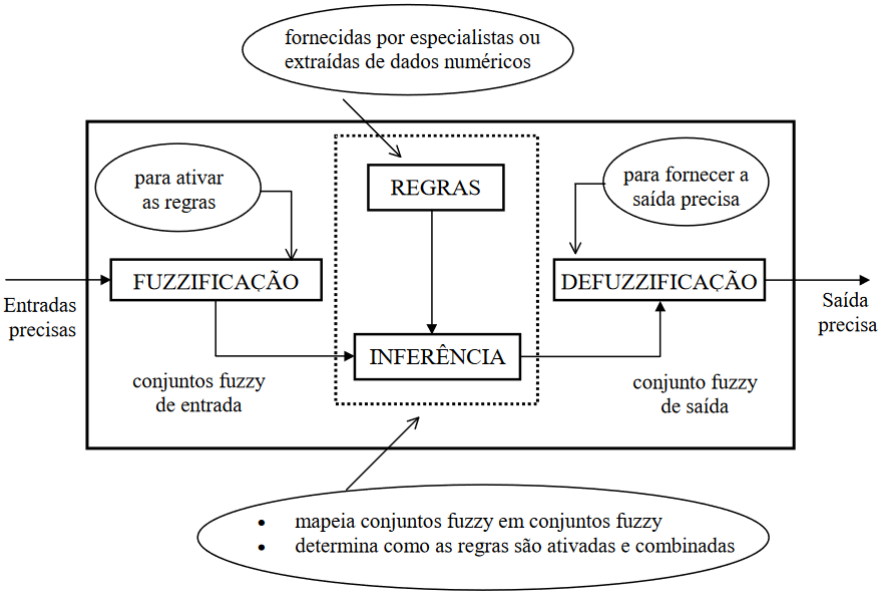


Figura 3.6: Representação gráfica das etapas que compõem um modelo baseado em lógica *Fuzzy* (Tanscheit, 2004)

3.6.1 Biblioteca *scikit-fuzzy*

Existem diversas ferramentas disponíveis para a implementação de um sistema baseado em lógica *Fuzzy*, dentre elas está a *scikit-fuzzy* (Pedregosa et al., 2011). Essa biblioteca foi desenvolvida pela comunidade *SciPy* com o objetivo de disponibilizar, ao ambiente Python, ferramentas para o desenvolvimento de algoritmos baseados em lógica *Fuzzy*.

Um controlador *Fuzzy* implementado a partir da biblioteca *scikit-fuzzy* possui elementos essenciais para o seu funcionamento, como a definição do universo, a definição das funções de pertinência dos antecedentes e consequentes e, por fim, a criação das regras. Nas próximas subseções serão detalhadas as funções de cada um desses elementos, além da forma como eles devem ser definidos dentro do *scikit-fuzzy*.

Criação do universo

O universo representa a caracterização das entradas e saídas do controlador *Fuzzy*. Essa definição consiste na definição do intervalo representando os valores válidos para aquela variável, além do seu nome. Considerando, por exemplo, que o controlador deseja detectar a ocorrência de incêndios (*fire*), pode-se definir um universo contendo a variável de saída representando o incêndio, e duas variáveis de entrada representando os sinais de fumaça (*smoke*) e temperatura (*temperature*).

Uma possível descrição desse universo pode ser definida pela Listagem 3.1.

Listagem 3.1: Criação do universo do controlador Fuzzy, contendo a definição da emergência e dos sinais de temperatura e fumaça.

```
smoke = ctrl.Antecedent(np.arange(0, 801, 1), "smoke")

temperature = ctrl.Antecedent(np.arange(0, 101, 1),
                              "temperature")

fire = ctrl.Consequent(np.arange(0, 101, 1), "fire")
```

No exemplo, é definido que os valores representando os sinais de fumaça e temperatura estão limitados aos intervalos $[0, 800]$ e $[0, 100]$, respectivamente. Por fim, outro intervalo de 0 a 100 é definido para a emergência, representando a probabilidade, em porcentagem, de ocorrência daquela emergência.

Uma vez definidos quais são as variáveis de entrada e de saída, e seus respectivos intervalos aceitos, é necessária a criação das funções de pertencimento caracterizando cada um deles.

Funções de pertencimento

As funções de pertencimento descrevem o grau de pertencimento, entre 0 e 1, de um valor numérico para um conjunto de uma determinada variável. Para a definição das funções de pertencimento, a biblioteca disponibiliza diversas opções, como *fuzz.trimf* para funções triangulares e *fuzz.gaussmf* para funções gaussianas. A Listagem 3.2 e a Figura 3.7 mostram a forma de criação da função de pertencimento para o conjunto *smoke* e sua representação gráfica, respectivamente.

Listagem 3.2: Definição das funções de pertencimento do conjunto representando o sinal de fumaça.

```
smoke["low"]=fuzz.trimf(smoke.universe, [-400,0,400])
smoke["medium"]=fuzz.trimf(smoke.universe, [0,400,800])
smoke["high"]=fuzz.trimf(smoke.universe, [400,800,1200])
```

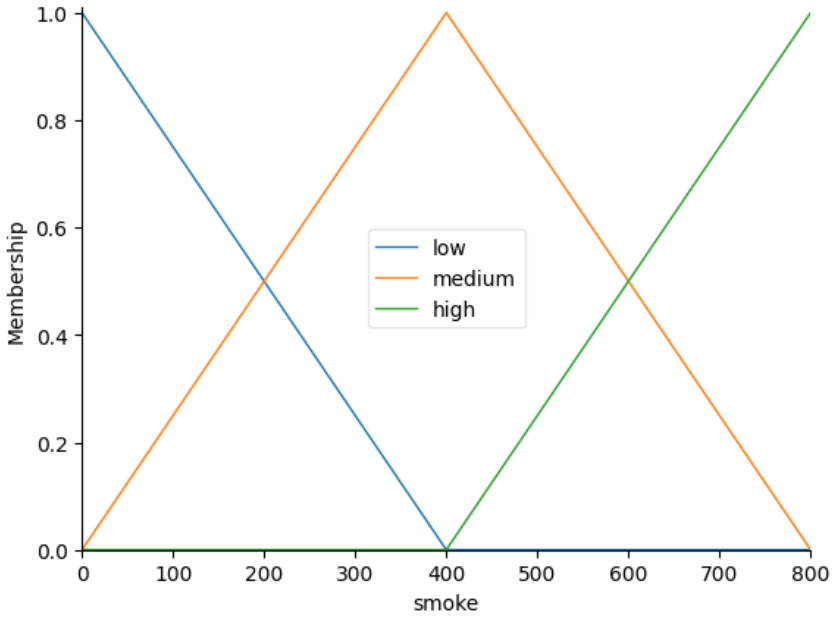


Figura 3.7: Representação gráfica da função de pertencimento do conjunto representando o sinal fumaça.

Essa função de pertencimento define três conjuntos representados por funções triangulares, *low*, *medium*, *high*, descrevendo a intensidade do sinal de fumaça. Outros formatos de função podem ser utilizados, como pode ser visto no exemplo de descrição do consequente incêndio (Listagem 3.3), onde foram utilizadas as funções gaussianas (*gaussmf*) e função de pertinência em forma de Pi (*pimf*).

Listagem 3.3: Definição das funções de pertencimento do conjunto representando a emergência incêndio (*fire*).

```
fire["very low"] = fuzz.pimf(fire.universe, -20, 0, 1,
                             30)
fire["low"]  = fuzz.gaussmf(fire.universe, 25, 8)
fire["medium"] = fuzz.gaussmf(fire.universe, 50, 8)
fire["high"] = fuzz.gaussmf(fire.universe, 70, 8)
fire["very high"] = fuzz.pimf(fire.universe, 65, 99,
                              101, 102)
```

As funções de pertencimento devem ser definidas para todas as variáveis presentes no universo. Dessa forma, valores escalares são mapeados com base nos conjuntos que caracterizam as variáveis, permitindo às regras do controlador *Fuzzy* criar uma relação de causa e consequência entre os valores de entrada e saída.

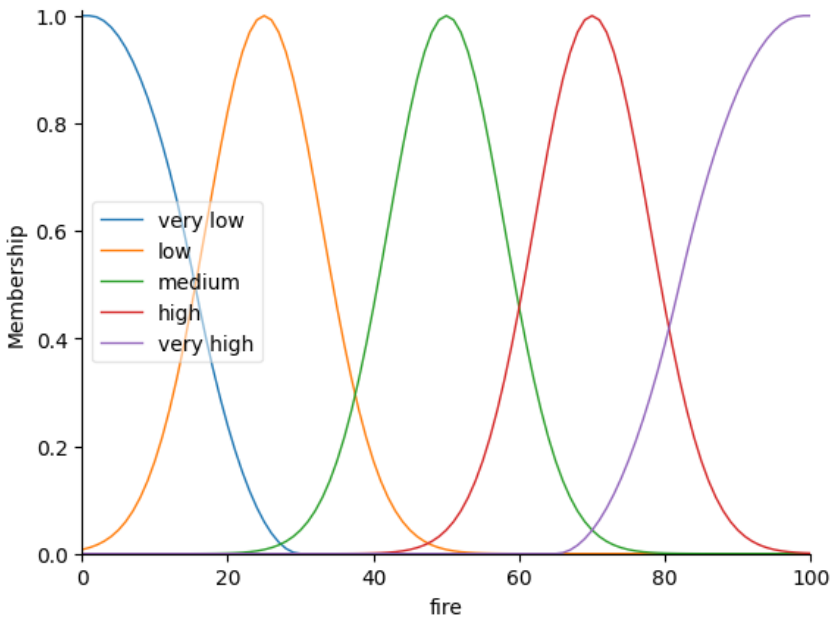


Figura 3.8: Representação gráfica da função de pertencimento do conjunto representando a emergência incêndio.

Regras

Um conjunto de regras compõe o núcleo do controlador *Fuzzy*, sendo cada regra composta por um ou mais antecedentes e um consequente. Dessa forma, é possível gerar a relação causa e consequência do tipo:

IF *antecedente*, **THEN** *consequente*

A Listagem 3.4 ilustra a construção de duas regras, visando exemplificar o seu funcionamento dentro do contexto da detecção de emergências. A classe *ctrl.Rule* define o tipo que as regras devem ter, recebendo como parâmetro os antecedentes e o consequente. No exemplo dado, a interpretação das regras é a seguinte: regra 1) ***SE*** a temperatura for baixa e a quantidade de fumaça for baixa, ***ENTÃO*** a probabilidade de ocorrência de um incêndio é muito baixa; regra 2) ***SE*** a temperatura for alta e a quantidade de fumaça for alta, ***ENTÃO*** a probabilidade de ocorrência de um incêndio é alta.

Listagem 3.4: Definição de duas regras utilizadas dentro do controlador

```
from skfuzzy import control as ctrl

rule1 = ctrl.Rule(temperatura["low"] & smoke["low"],
                  incendio["very low"])
rule2 = ctrl.Rule(temperatura["high"] & smoke["high"],
                  incendio["high"])
```

Com o conjunto de regras bem definido, o sistema é capaz de, a partir dos valores de entrada, identificar o comportamento do consequente, para então retornar um valor numérico correspondente.

Cada um dos elementos que compõe o sistema, universo, funções de pertencimento e regras, devem ser elaboradas por um especialista. Esse é um elemento de extrema importância, uma vez que é necessário que seus valores sejam definidos por alguém capaz de caracterizar, de forma mais precisa possível, o comportamento das emergências. Dessa forma, o sistema poderá retornar valores válidos e precisos como resposta. A definição desses parâmetros é realizada a partir de um arquivo configuração.

3.7 Computação hierárquica e colaborativa

Nas redes de sensores sem fio, os nós sensores podem ser representados por dispositivos com diferentes características de memória, consumo de energia e capacidade de processamento. Esse ambiente heterogêneo traz consigo a possibilidade de desenvolvimento de sistemas a partir de diferentes arquiteturas, como a definição de todos os nós realizarem as mesmas tarefas ou os nós serem divididos em grupos, por exemplo. A forma como é definida a arquitetura, a dinâmica de troca de informações e divisão de tarefas pode impactar diretamente no desempenho do sistema, tanto em consumo de recursos quanto em precisão de resultados. Dentre as diversas possibilidades de arquitetura do sistema existem duas que se apresentam como de grande importância: a computação hierárquica e a colaborativa.

A computação hierárquica se baseia na elaboração de níveis de atuação dentro de um sistema (La et al., 2019). Normalmente esses níveis são definidos a partir das características de processamento dos nós, então níveis mais baixos são compostos por dispositivos de menor capacidade computacional, consequentemente realizam processamentos mais simples (Ngo et al., 2020). A ideia é que os resultados do processamento nas camadas mais baixas sejam utilizados para a realização das tarefas dos níveis mais altos, até que se chegue ao último nível de processamento, onde o resultado desejado é obtido.

A computação colaborativa, por outro lado, não define sua estrutura a partir da utilização de níveis hierárquicos. O processamento é normalmente realizado de forma uniforme por todos os nós da rede. Além disso, sua característica principal é a de que os nós utilizam de forma direta as informações obtidas por outros nós (Nalini e Valarmathi, 2018). Diferentemente da computação hierárquica, essas informações não consistem em resultados de processamentos de outros nós, mas sim em informações brutas obtidas diretamente dos sensores, por exemplo.

É importante ressaltar que ambas as formas não são excludentes. É possível que um sistema seja dividido em níveis hierárquicos e, ao mesmo tempo, utilize o esquema colaborativo de troca de informações entre os nós. A utilização dessas abordagens

pode diminuir significativamente a quantidade de largura de banda utilizada, diminuindo possíveis colisões de dados, uma vez que processamentos mais leves são definidos na borda da rede (hierárquica), além de permitir que os resultados sejam obtidos a partir de um conjunto de dados maior (colaborativo), podendo influenciar diretamente na precisão do resultado obtido.

Capítulo 4

Metodologia

O modelo de detecção proposto visa, a partir da implementação de funcionalidades flexíveis e configuráveis, possibilitar a detecção de emergências com rapidez e precisão. Consequentemente, ações de mitigação e de alerta para a população ou autoridades poderão ser tomadas mais rapidamente, a fim de minimizar os possíveis danos à região e aos seus moradores. O modelo oferece uma detecção hierárquica em diferentes níveis da rede, uma vez que cada unidade de detecção possui autonomia para inferir sobre emergências. Ao mesmo tempo, oferece uma detecção colaborativa, na qual um sistema *Fuzzy* utiliza valores coletados por diversas unidades de detecção para realizar a detecção da emergência de forma mais precisa.

Para implementar esse modelo de detecção algumas características foram sobressaltadas para guiar a metodologia de desenvolvimento. O comportamento hierárquico da aplicação, no qual o funcionamento dos módulos mais complexos está diretamente conectado aos módulos mais simples, sugere a utilização de um modelo de desenvolvimento incremental, onde os módulos e subsistemas mais básicos serão desenvolvidos e testados primeiro, para que, então, sejam desenvolvidos os módulos mais complexos. Além disso, como todas as unidades de detecção possuem autonomia para, respeitando seu respectivo nível de complexidade e abrangência, inferir sobre emergências, é possível testar o sistema de detecção de emergências incrementalmente.

Outras características a serem consideradas são das emergências em si e das regiões onde se deseja realizar a detecção. O processo de monitoramento das cidades tem, em sua essência, uma característica complexa, uma vez que não é possível definir o comportamento global de uma região a partir de medições realizadas em localidades específicas. É coerente dizer, por exemplo, que a ocorrência de chuva em determinadas regiões não permite afirmar que também estará chovendo em toda a extensão da cidade. Isso mostra a dificuldade em generalizar as medições realizadas em grandes áreas. Por esse motivo, o modelo de detecção se baseará em dividir a cidade em áreas menores, denominadas zonas de interesse (ZI), que podem ser, por exemplo, um bairro ou uma freguesia. Pode-se definir ainda uma região total, R, como um

conjunto de r zonas de interesse. Assim, para cada zona de interesse $z_i \in R$, sendo $i = 1, \ldots, r$, é atribuído um subconjunto $E_i \subseteq E = \{e_1, \ldots, e_h\}$ contendo, dentre todas as emergências e_j $(j = 1, \ldots, h)$ que o sistema é capaz de detectar, as emergências que devem ser analisadas e detectadas naquela zona z_i. Cada ZI pode ter conjuntos de emergências que diferem entre si dependendo do contexto geográfico, social e histórico daquela região.

A aquisição e análise de sinais capazes de definir a existência ou não de uma emergência em uma ZI será realizada por redes de sensores sem fio. Uma das características importantes do modelo é que todos os sensores devem ser capazes de se comunicar entre si. Isso se dá por um motivo principal: um sensor s_1 pode necessitar de dados provenientes de outro sensor s_2 para determinar a ocorrência de uma emergência. No entanto, a depender da política de comunicação utilizada, pode acontecer desses dois sensores não conseguirem se comunicar diretamente, devido a uma grande distância física entre eles, por exemplo. Nesse caso seria necessário que a unidade s_1 enviasse uma requisição de dados para a rede, para que ela fizesse com que essa requisição chegasse até a unidade de detecção de destino. Para lidar com essa situação, foi utilizado o protocolo MQTT, que será abordado de forma mais detalhada na Seção 4.2.

A rede de sensores, composta por várias UDEs, deverá obter os sinais do ambiente, realizar o processamento e, se necessário, transmitir informações entre os nós da rede para que o processamento de dados possa ser realizado da melhor forma possível. Considere então uma unidade de detecção u_k composta por um conjunto de sensores ativos, Sk, $k = 1, \cdots, n$. É importante dizer que o conjunto S_k pode variar a depender do tipo de emergências que as unidades estão monitorando. As UDEs responsáveis pela detecção de incêndios, por exemplo, terão sensores ativos diferentes dos responsáveis pela detecção de enchentes.

O sistema aqui proposto foi implementado partindo do pressuposto que as unidades de detecção possuem o mesmo conjunto de sensores disponíveis para o uso. Dessa forma, uma vez que o conjunto de sensores disponíveis são iguais, o comportamento de cada unidade de detecção depende de sua configuração, que definirá quais destes sensores estarão ativos para o monitoramento.

A rede de sensores sem fio implementada é heterogênea, o que significa que as unidades não são necessariamente iguais, tanto em características de *hardware*, quanto em suas funcionalidades na rede. Inicialmente, as UDEs podem ser classificadas a partir da utilização de três categorias: unidades de baixo poder computacional (BPC), de médio poder computacional (MPC) e de alto poder computacional (APC). Cada categoria possui requisitos sugeridos, que devem, preferencialmente, ser atendidos de modo que seja possível realizar todas as suas atribuições da melhor forma possível. À medida que a capacidade de processamento e armazenamento de cada categoria cresce, a tendência é que se aumente também a complexidade das tarefas e atribuições daquela classe na rede. Entretanto, isso é apenas uma sugestão. Nada impede que as funcionalidades de uma unidade de detecção BPC sejam implementadas em

uma plataforma com um processador mais veloz ou com mais espaço de armazenamento, embora isso possa gerar sub-utilização de recursos. O caso contrário não é aceito, ou seja, implementar as funcionalidades de uma unidade de detecção APC em uma plataforma inferior, devido à falta de recursos mínimos.

A Tabela 4.1 mostra alguns desses requisitos, auxiliando no processo de decisão de qual plataforma de *hardware* utilizar para cada função da rede, e o Apêndice A apresenta a descrição de diversas plataformas de *hardware* que podem ser utilizadas como unidades de detecção. Lá são sugeridas as categorias onde cada *hardware* pode fazer parte, tomando como base as definições da Tabela 4.1.

Tabela 4.1: Tabela com sugestões de configuração de *hardware* para cada categoria.

Atributos	**BPC**	**MPC**	**APC**
Clock	$\leq$50 MHz	(50 MHz, 1 GHz)	$\geq$ 1 GHz
RAM	$\leq$ 130 KB	(130 KB, 100MB)	$\geq$ 100 MB
Memória Flash	$\leq$ 500 KB	(500 KB, 10 MB)	$\geq$ 10 MB
Memória de Dados	$\leq$ 64 KB	(64 KB, 500 MB)	$\geq$ 500 MB

4.1 Tarefas e atribuições

Nessa seção serão abordadas as atribuições principais de cada uma das três categorias definidas para as unidades de detecção. A implementação dessas funcionalidades foi realizada de forma incremental, uma vez que são complementares. A Figura 4.1 define o diagrama de classes representando cada uma dessas categorias. A classe base (UDE) define os funcionamentos comuns a todas as unidades de detecção e a medida que a complexidade da unidade aumenta, mais tarefas são atribuídas a ela, com respeito a cada categoria descrita: BPC, MPC e APC.

A partir do diagrama de classes, é possível observar o desenvolvimento com base em um modelo hierárquico, no qual os dispositivos herdam as funcionalidades da classe pai de forma inerente. Assim, classes mais robustas também podem executar as tarefas das classes menos complexas. A realização ou não dessas tarefas depende estritamente do modo como a UDE é configurada. Por exemplo, se um MPC for configurado com sensores ativos, ele também realizará o processo de sensoriamento, assim como um BPC.

4.1.1 Baixo Poder Computacional - BPC

A unidade de baixo poder computacional foi a primeira a ser implementada. As unidades de detecção dessa categoria possuem duas funções principais: obter os sinais do ambiente a partir dos seus sensores e realizar a primeira etapa de processamento dos dados. O conjunto de sensores S_k que faz parte de sua estrutura é definido a depender da lista de emergências que aquela unidade de detecção deve monitorar.

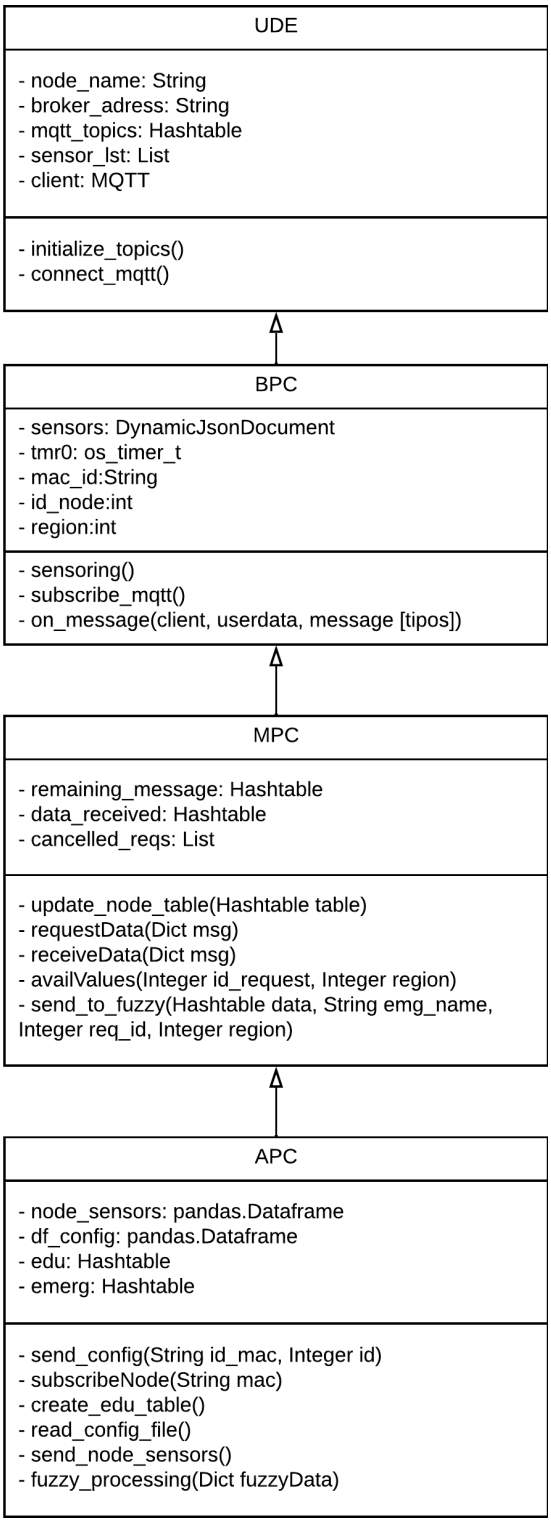


Figura 4.1: Diagrama de classes representando as três categorias de unidades de detecção: BPC, MPC e APC.

Diferentes emergências resultam em diferentes conjuntos de sensores. Uma vez definido o conjunto S_k, os sensores são configurados para estarem aptos a iniciar o processo de monitoramento.

Além da obtenção dos sinais do ambiente, as unidades de detecção de baixo poder computacional também são responsáveis por realizar a primeira etapa do processamento de dados. Este processamento consiste em verificar a existência de fatores de risco, analisando se os sinais obtidos apresentam valores que indiquem a existência de uma emergência. Essa etapa permite que o sistema não envie todos os valores medidos pelos BPCs, mas somente os dados que apresentam comportamento anômalo. Dessa forma, o fluxo de dados sendo enviados na rede diminui, assim como, a probabilidade de sobrecarga da rede e o gasto energético do sistema.

Essa detecção é realizada a partir da definição de valores gatilhos ou limiares. Se o valor do sinal for maior ou menor que o gatilho, a depender da configuração da emergência, significa que é um valor anormal. Portanto, considerando $x(t)$ o valor lido pelo sensor no instante t, pode-se dizer que:

$$H(x) = \begin{cases} 1, & x(t) \geq Th_x \\ 0, & x(t) < Th_x. \end{cases} \tag{4.1}$$

Na Equação 4.1 $H(x)$ representa uma função cujo valor define a existência ou não de valores incomuns para um dado sinal, e Th_x representa o limiar para o sinal x. Nesse exemplo, considera-se que uma medida é anormal quando seu valor for maior ou igual ao limite pré-determinado. Este processo de verificação de anomalias é o primeiro passo no processo de detecção de emergência. Quando um valor anormal é detectado, um alerta deverá ser enviado para as unidades de detecção de médio ou alto poder computacional (MPC ou APC), de modo que as medidas necessárias possam ser tomadas. Além disso, ações simples de alerta ou mitigação de emergências podem ser implementadas em uma unidade de detecção BPC, como alertas locais. Esse tipo de medida pode gerar mais agilidade para o início de procedimentos cruciais para reduzir os danos de uma emergência.

4.1.2 Médio Poder Computacional - MPC

Inicialmente, como uma unidade de detecção MPC possui maior poder de processamento e armazenamento do que um BPC, é possível associar a ele sensores mais complexos, como microfones e câmeras, podendo gerar dados com maior valor associado.

Além da possibilidade de realizar o monitoramento padrão, esse tipo de UDE recebe avisos de outras unidades quando o valor $H(x) = 1$. Diante desse alerta, ela entra em contato com as unidades de detecção presentes na zona de interesse onde a anomalia foi detectada e solicita o valor do sinal x e de valores que se correlacionem com ele. Dessa forma, a partir da obtenção do comportamento de outros sinais que, juntos,

podem caracterizar a existência ou não de uma determinada emergência, é possível verificar se a anomalia detectada foi apenas um valor isolado ou não. Por exemplo, se uma unidade de detecção indica que a temperatura de uma zona está elevada, provavelmente é interessante confirmar essa leitura com outras UDEs naquela região, além de averiguar com essas unidades se também há elevados níveis de fumaça, a fim de detectar incêndio.

Uma vez recebidos os valores do sinal $x(t)$ e os sinais a ele correlacionados, uma unidade de detecção MPC é capaz de decidir se aquela anomalia detectada inicialmente pode representar a existência de uma emergência ou não. Em caso afirmativo, os dados coletados pelo MCP são enviados às unidades de detecção de alto poder computacional (APC) para serem melhor avaliados. Assim como os BPCs, as unidades MPCs também têm autonomia para gerar ações de alerta ou mitigação de emergências, porém com mais recursos, como enviar mensagens com maior alcance ou controlar elementos na zona de interesse (semáforos, irrigadores, subestações de energia, etc).

4.1.3 Alto Poder Computacional - APC

Seguindo a lógica incremental de desenvolvimento, as unidades de detecção de alto poder computacional também podem realizar a medição do ambiente a partir dos seus sensores, além de receber notificações sobre possíveis anomalias detectadas nas unidades BPC e MPC. No entanto, sua principal função na rede é definir, com maior acurácia, o resultado do processo de detecção das emergências.

A partir da utilização dos dados obtidos pelas outras unidades de detecção, o APC tem informações suficientes para identificar a probabilidade de existência de determinadas emergências em uma determinada zona de interesse. Devido à intrínseca incerteza existente no processo de detecção de emergências, será utilizado um sistema baseado em lógica *Fuzzy*, que recebe os valores obtidos pelas outras unidades de detecção de forma pré-processada, além de informações importantes como sua localização geográfica (latitude e longitude). O resultado esperado é um valor definindo a probabilidade de existência ou não de uma emergência. Foi definido que cada zona de interesse teria uma e apenas uma unidade de detecção APC, sendo ela a responsável pelo processamento *Fuzzy* dos dados para a detecção de emergências daquela ZI.

Diante desse cenário, é essencial que as unidades APC sejam capazes de se comunicar com dispositivos fora da rede de sensores no caso da detecção de emergências. Dessa forma, será possível que, em caso de probabilidades consideravelmente altas de ocorrer uma emergência, possam ser enviados sinais de alerta tanto para a população, quanto para dispositivos monitorados pelas autoridades responsáveis.

A devida execução e coordenação das funcionalidades e troca de dados entre cada classe de unidade de detecção são garantidas pelas políticas de comunicação estabe-

lecidas pelo *framework* desenvolvido, conforme discutido na seção a seguir.

4.2 Políticas de Comunicação

Para a rede de sensores proposta funcionar da forma esperada é necessário que o processo de troca de mensagens seja bem estabelecido, com padrões e formatações de mensagens que permitam às UDEs funcionarem de forma colaborativa e precisa. A política de comunicação do *framework* define toda essa estrutura de comunicação durante o processo de detecção de emergência, como, por exemplo, a formatação das mensagens de inscrição, de aviso de valores anômalos e de requisição de dados, além de definir quais canais de comunicação serão utilizados.

Foi definida a utilização do protocolo MQTT para a troca de mensagens entre as unidades de detecção. O MQTT é um protocolo amplamente utilizado em aplicações IoT e de RSSF devido ao seu baixo consumo de memória e rede. A utilização de tópicos para a comunicação permite que mensagens sejam enviadas e recebidas somente pelas unidades de detecção cadastradas no tópico em questão, resultando em uma diminuição considerável no fluxo de dados na rede. Além disso, a comunicação via tópicos permite que, independentemente da distância física entre as UDEs, todas as unidades sejam capazes de se comunicar entre si, desde que estejam conectadas ao broker MQTT e cadastradas no mesmo tópico.

Para a política de comunicação aqui desenvolvida foram estabelecidos sete tópicos MQTT para a troca de mensagens: *subscribe*, *update_node_table*, *config*, *sensoring*, *request*, *required_values* e *hazard_data*. Toda mensagem enviada e recebida por esses tópicos deve estar configurada no padrão JSON. O diagrama de sequência, apresentado na Figura 4.2, ilustra cada um dos processos que requerem a troca de mensagem entre as unidades de detecção, considerando as seguintes funcionalidades para cada tópico:

- ***subscribe***: as unidades de detecção recém conectados ao sistema enviam por esse tópico mensagens solicitando a inscrição na rede.
- ***update_node_table***: toda vez que uma nova UDE é inscrita no sistema, a tabela de unidade ativas atualizada é enviada por esse tópico para os MPCs. Dessa forma eles se mantém com as informações atualizadas do sistema para uma possível realização de uma requisição.
- ***config***: assim que uma unidade de detecção é inscrita no sistema, é enviada uma mensagem de configuração por esse tópico. As mensagens de configuração definem quais sensores serão ativos e seus respectivos valores limite, além de definir sua ZI de atuação e seu ID na rede.
- ***sensoring***: assim que uma unidade detecta um valor anômalo durante o processo de monitoramento, ela, a partir do envio de mensagens por esse canal de comunicação, notifica as unidades de detecção de maior poder computacional.

- ***request***: ao receber a notificação de um valor anômalo, as UDEs de maior poder computacional enviam requisições para outras unidades, para obter mais amostras daquele sinal e dos que são relacionados a ele. Esse tópico é definido para o envio dessas requisições.
- ***required_values***: ao receber uma requisição, as unidades de detecção devem retornar o valor dos sinais solicitados. Esses valores são enviados a partir da utilização deste tópico.
- ***hazard_data***: tópico para o envio dos dados requisitados e que serão utilizados pelo controlador *Fuzzy*.

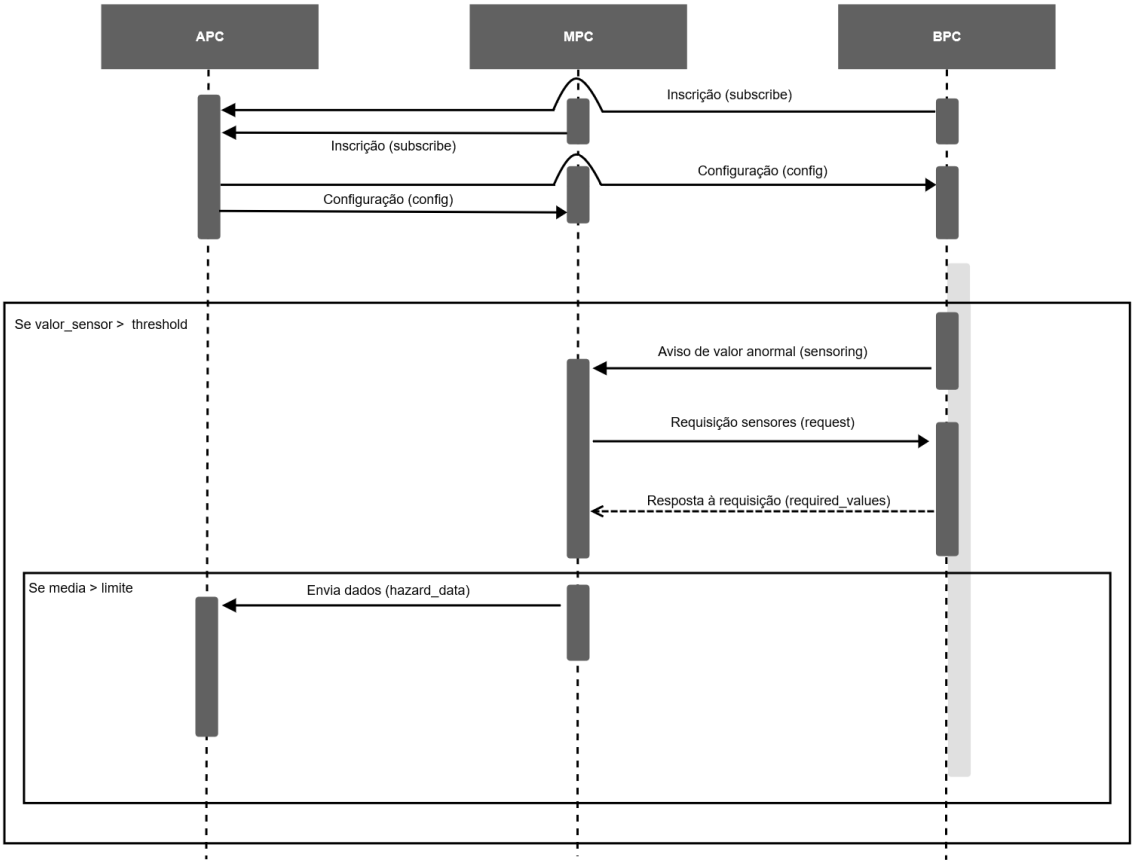


Figura 4.2: Diagrama de sequência caracterizando o processo de troca de mensagens entre as UDEs durante o processo de detecção de emergências.

A seguir serão detalhadas cada etapa do diagrama da Figura 4.2, abordando o seu papel no sistema e a formatação definida para suas respectivas mensagens.

4.2.1 Inscrição das unidades de detecção

Uma rede de sensores sem fio pode ter dezenas, até centenas de unidades de detecção, tornando inviável a inscrição manual de todas elas à medida que se conectam à rede. Dessa forma, foi definido um processo de inscrição automatizada. Cada unidade adicionada à rede será inscrita no sistema, para poder ser devidamente configurada, e então estar pronta para colaborar no processo de detecção de emergências.

Ao ser iniciado, o sistema deve contar com pelo menos uma unidade de detecção de alto poder computacional. Ela será responsável pelo recebimento das mensagens de inscrição das UDEs recém conectadas à rede e pelo processo de inscrição propriamente dito. A solicitação de inscrição é simples, e consiste no envio de uma mensagem de inscrição pela nova unidade de detecção. Essa mensagem é composta somente pelo seu ID MAC (ver Listagem 4.1), e é enviada pelo tópico MQTT *subscribe*.

Listagem 4.1: Exemplo de uma mensagem de inscrição enviada por uma unidade de detecção recém conectado à rede.

```
id_mac: {"A8:48:FA:DD:3D:D1"}
```

Ao identificar o recebimento de uma nova mensagem no tópico *subscribe*, a unidade de detecção APC inicia o processo de inscrição. Esse processo consiste em adicionar a nova unidade de detecção à tabela de unidades ativas, atribuindo a ela um ID na rede e especificando quais sensores serão utilizados e quais emergências serão monitoradas, além da zona de interesse que ela realizará o monitoramento.

Todas as informações sobre sensores, emergências, zona de atuação e categoria de cada unidade de detecção estão descritas em um arquivo de configuração previamente definido pelo usuário. A partir desse arquivo, é possível obter as informações necessárias para sua inscrição e, posteriormente, sua configuração.

Arquivo de configuração

O arquivo de configuração consiste em um arquivo binário previamente definido pelo usuário. Esse arquivo define duas estruturas que serão carregadas e utilizadas na execução do APC. Uma delas é a modelagem das emergências, descrevendo quais sensores a definem e seus respectivos valores gatilhos. A outra consiste nas informações de configuração de cada uma das UDEs: ID da zona de interesse, latitude, longitude, conjunto de sensores ativos e de emergências a serem monitoradas. A Listagem 4.2 exemplifica as informações armazenadas em um arquivo de configuração definindo dois tipos de emergência, incêndio e enchente, definindo quais sensores as definem, junto com seus valores gatilho; além de definir as informações de configuração de duas UDEs.

Listagem 4.2: Conteúdo armazenado no arquivo de configuração (*.bin*). Representa todos os elementos de configuração das unidades de detecção cadastradas e das emergências

```
emergency = {"fire":{"temperature":60,
                     "smoke":500,
                     "uv":9},
             "flood":{"humidity":90,
                      "vibration":1}}
             }

edu = {"A8:48:FA:DD:3D:D1":{
                    "class":"BPC",
                    "region":2,
                    "latitude":12.3456,
                    "longitude":-73.8765,
                    "sensors":["temperature", "smoke"],
                    "emergency":["fire"]},

       "A8:48:FA:DC:BC:60":{
                    "class":"BPC",
                    "region":2,
                    "latitude":12.0745,
                    "longitude":-71.3421,
                    "sensors":["temperature", "smoke",
                               "humidity"],
                    "emergency":["fire", "flood"]}
      }
```

Considerando o ID da unidade de detecção responsável por enviar a mensagem de inscrição definida na Listagem 4.1, o arquivo de configuração sinaliza que ela deverá atuar como BPC, realizando o monitoramento de incêndios na zona de interesse 2, a partir da utilização dos sinais de temperatura e fumaça.

As informações disponíveis no arquivo de configuração permitem que a unidade APC realize o cadastro da UDE na tabela de unidades ativas. Uma vez realizado esse cadastro, e considerando que não existia nenhuma unidade de detecção previamente inscrita na rede, a tabela de unidades inscritas é representada pela Tabela 4.2. Nessa tabela, estão listados os sensores disponíveis em cada UDE, onde o valor 1 atribuído a ele define um sensor ativo, e 0, um sensor inativo.

Uma vez que a tabela de UDEs ativas é atualizada, é necessário que ela também seja atualizada dentro de cada MPC, uma vez que ela é essencial no processo de obtenção de dados. Essa atualização é feita a partir do envio da tabela atualizada via tópico *update_node_table*. Ao receber a mensagem, os MPCs atualizam sua tabela com a

Tabela 4.2: Tabela contendo as unidades de detecção já inscritas no sistema

MAC	00:15:5D:4A:93:5F	A8:48:FA:DD:3D:D1
class	APC	BPC
id_node	1	2
region	2	2
latitude	12.0745	12.3456
longitude	-71.3421	-73.8765
temperature	0	1
smoke	0	1
uv	0	0
humidity	0	0
vibration	0	0

que está presente na mensagem.

Feito isso, uma vez que a unidade de detecção é inscrita no sistema, sendo adicionado a Tabela 4.2, falta apenas um passo para que ela esteja totalmente funcional: sua configuração.

4.2.2 Configuração das unidades de detecção

O processo de configuração visa definir os parâmetros essenciais para a execução do processo de sensoriamento por cada UDE. Essas informações são: o ID da unidade de detecção na rede, a zona de interesse que ela está monitorando, a categoria da UDE, uma lista contendo os sensores que devem ser utilizados para o monitoramento, além da descrição das emergências que devem ser monitoradas.

Dessa forma, assim que a unidade de detecção é inscrita no sistema, a unidade APC envia, pelo tópico *config*, uma mensagem contendo todas as informações necessárias para sua configuração. A Listagem 4.3 ilustra como seria a mensagem de configuração para a unidade BPC recém-adicionado utilizada como exemplo na seção 4.2.1.

Listagem 4.3: Exemplo de uma mensagem de configuração.

```
{"class": "BPC"
 "id_mac": "A8:48:FA:DD:3D:D1",
 "id_node": 2,
 "region": 2,
 "latitude": 12.3456,
 "longitude": -73.8765,
 "sensors":["temperature", "smoke"],
 "emergency": {"fire": {"temperature": 60, "smoke": 500}}
}
```

O *id_node* equivale ao ID daquela UDE na rede, atributo essencial para sua identificação tanto em cenários de requisição de dados, quanto de monitoramento e envio de sinais anômalos. O campo *region* representa o ID da zona de interesse sendo monitorada. O campo *sensors* apresenta o nome dos sensores que serão ativos naquela UDE, enquanto *emergency* contém o nome e a descrição das emergências a serem monitoradas. A descrição de uma emergência consiste na definição de sensores que a caracterizam e de seus valores de gatilhos.

A mensagem descrita na Listagem 4.3 configura a unidade de detecção de modo que ela possua o ID 2, esteja alocado à zona de interesse 2, com os sensores de temperatura e fumaça ativos, além de monitorar a ocorrência de incêndios. O incêndio está modelado a partir dos sensores de temperatura e fumaça, definidos com valores gatilhos de 60^oC e $500ppm$, respectivamente.

Analisando o arquivo de configuração e a mensagem de configuração dadas como exemplo, Listagens 4.2 e 4.3, respectivamente, é possível perceber diferenças na descrição da emergência *"fire"* (incêndio). No arquivo de configuração ela é definida a partir de três sinais (temperatura, fumaça e radiação UV), enquanto na mensagem de configuração é definida apenas por duas (temperatura e fumaça). Essa divergência acontece devido à decisão de passar a descrição da emergência na mensagem de configuração contendo apenas os sinais para os quais a UDE possui sensores ativos. Dessa forma, como a unidade para a qual essa mensagem de configuração foi gerada possui apenas os sensores de temperatura e fumaça ativos, a descrição da emergência é passada somente com os valores desses dois sinais. Caso a unidade de detecção possuísse somente o sensor de temperatura, o campo de descrição de incêndio apresentaria somente o sinal de temperatura.

Se a mensagem de configuração fosse gerada para um MPC, seria necessário um campo de informação a mais. Esse campo consiste na descrição completa de todas as emergências definidas no arquivo de configuração. Essa informação é essencial para que o MPC seja capaz de, em casos de alerta de sinais anômalos, realizar requisições para as UDEs. A Listagem 4.4 ilustra a mensagem de configuração gerada para uma unidade MPC.

Listagem 4.4: Exemplo de uma mensagem de configuração.

```
{"class": "MPC"
 "id_mac": "A8:48:FA:DD:3D:D1",
 "id_node": 2,
 "region": 2,
 "latitude": 12.3456,
 "longitude": -73.8765
 "sensors":["temperature", "smoke"],
 "emergency": {"fire": {"temperature": 60, "smoke": 500}
               },
 "emg_list": {"fire":{"temperature":60,
                      "smoke":500,
```

```
                        "uv":9},
              "flood":{"humidity":90,
                        "vibration":1}}
            }
}
```

Assim que uma unidade de detecção recebe a notificação de uma nova mensagem no tópico de configuração, ela deve verificar se o seu ID MAC é o mesmo do apresentado na mensagem. Uma vez reconhecida sua mensagem de configuração, ela utiliza os campos de dados para atualizar seus atributos de ID na rede, zona de interesse, latitude, longitude, além de passar pelo processo de inicialização dos sensores e das emergências.

A inicialização dos sensores consiste em ativar os sensores que devem ser utilizados para o sensoriamento. Cada unidade de detecção possui uma estrutura de dados que armazena, na forma {*chave-valor*}, as informações sobre os sensores. As chaves dessa estrutura representam os nomes de cada sensor disponível, enquanto o valor atribuído a elas funciona como uma *flag*, sinalizando a ativação ou não daquele sensor. Dessa forma, ao final do processo de inicialização, todos os sensores que possuem valores diferentes de zero são considerados ativos e funcionais para a realização do monitoramento.

A inicialização das emergências consiste simplesmente em salvar a descrição da emergência com seus sensores e respectivos valores gatilhos. De modo que, uma vez realizados esses processos, a unidade de detecção consiga saber qual sensor ela deve utilizar para fazer a leitura do ambiente; além de possuir a informação dos valores limite para cada emergência, permitindo que seja feita a identificação de valores anômalos.

4.2.3 Processo de sensoriamento

Uma vez configurada, a unidade de detecção está pronta para a realização do monitoramento da ZI em que ela está inserida. O processo de monitoramento consiste na utilização dos sensores ativos para a captação de sinais do ambiente, ao mesmo tempo que verifica se os valores obtidos são anômalos ou não (Equação 4.1). Se for detectado um valor anômalo, a unidade de detecção deve enviar uma mensagem de aviso para as unidades de maior poder computacional.

Durante o processo de monitoramento é realizada a verificação de quais sensores estão ativos, para que então se realize a captação do sinal. Essa captação é realizada periodicamente e de forma conjunta. Quando se inicia o sensoriamento, todos os sensores ativos irão realizar suas medições e verificações de valores anômalos juntos. Dito isso, uma vez que os sinais forem captados, eles são comparados com seus respectivos valores gatilhos presentes em cada emergência designada para aquela UDE. Dessa forma, considerando uma unidade com o sensor de temperatura ativo, o valor

obtido para esse sensor será comparado com todos os valores gatilhos de temperatura definidos na descrição das emergências. Caso o valor captado seja maior, é identificado um valor anômalo para aquele sinal, considerando aquela determinada emergência.

Nesses casos, onde é reconhecido um valor anômalo, deve ser enviada uma mensagem, via tópico *sensoring*, sinalizando a ocorrência para as unidades de maior poder computacional. No caso de nenhum sinal apresentar comportamento anormal o processo segue normalmente.

A estrutura dessa mensagem de aviso contém o ID da unidade de detecção na rede e o da ZI de atuação, o nome dos sinais que apresentaram comportamento anômalo, seus respectivos valores e o nome da emergência para o qual o comportamento anômalo foi identificado, além de conter o *timestamp* representando o momento que o sinal foi captado. Considerando que a unidade de ID 2 (exemplificada anteriormente), durante seu processo de monitoramento, identificou um sinal de temperatura com valor maior do que seu gatilho definido na descrição da emergência *fire*, a mensagem enviada por ela é ilustrada na Listagem 4.5.

Listagem 4.5: Exemplo de uma mensagem de sensoriamento reconhecendo o comportamento anômalo do sinal de temperatura.

```
{"id_node": 2,
 "region": 1,
 "timestamp": 1703800220,
 "emergency":{"fire":{"sensor":["temperature"],
                      "value":[70]}}}
```

No entanto, é possível que a mesma unidade detecte mais de um sinal com valor anômalo. Por exemplo, a unidade poderia ter detectado, também para a emergência *fire*, além da temperatura elevada, uma quantidade anormal de fumaça no ambiente. Nesse caso, ambos os alertas são enviados simultaneamente, como mostra a mensagem definida na Listagem 4.6. A estrutura da mensagem é a mesma, porém, neste exemplo, com dois nomes de sensores no campo *sensor* e dois valores no campo *value*. É importante ressaltar que os valores anômalos detectados e enviados em *value*, estão posicionados na mesma ordem que os nomes dos sensores.

Listagem 4.6: Exemplo de uma mensagem de sensoriamento reconhecendo o comportamento anômalo dos sinais de temperatura e fumaça.

```
{"id_node": 2,
 "region": 2,
 "timestamp": 1703800220,
 "emergency":{"fire":{"sensor":["temperature", "smoke"],
                      "value":[70, 550]}
             }
```

```
}
```

Dessa forma, considerando o exemplo, tem-se que a temperatura obtida foi de 70^oC, enquanto o sensor de fumaça captou o valor de $550ppm$, apresentando valores anômalos para a emergência *fire*.

4.2.4 Requisição dos Dados

As mensagens de sensoriamento enviadas pelo tópico *sensoring* representam avisos sobre valores que podem, ou não, representar a existência de uma emergência. De modo a definir se aquele valor anômalo não foi apenas um ruído de monitoramento ou uma perturbação, devem ser obtidas mais informações sobre o comportamento do ambiente naquela zona de interesse.

O processo de requisição tem como objetivo obter informações de sinais que auxiliem no processo de avaliação do estado da ZI, visando definir com maior acurácia se ela apresenta comportamento característico de uma emergência ou não. Os sinais necessários para essa avaliação são os sinais que descrevem a própria emergência que gerou o alerta. Sendo assim, o processo de geração das requisições consiste em verificar o nome da emergência para o qual o alerta foi gerado e, a partir da utilização da estrutura de dados contendo a descrição de todas as emergências, gerar mensagens de requisição para cada um dos sinais que a compõe.

As mensagens de requisição são enviadas pelo tópico *request_data* sendo compostas pelo ID da requisição, ID da UDE que a gerou e ID da ZI para qual está sendo realizada a requisição, além do nome do sinal que deve ser retornado. Considerando o arquivo de configuração descrito na Listagem 4.2, a emergência *fire* é descrita a partir da utilização dos sensores de temperatura, fumaça e radiação UV. Dessa forma, um alerta gerado para a emergência *fire*, resultará em requisições para cada um desses três sensores. A Listagem 4.7 mostra como seria a mensagem de requisição para o sensor de fumaça.

Listagem 4.7: Exemplo de uma mensagem de requisição.

```
{"id_request": 1,
 "id_node": 1
 "region": 2
 "sensor": "smoke"}
```

Ao identificar o recebimento de uma nova mensagem no tópico *request_data*, as unidades de detecção devem verificar se a ZI da requisição é a mesma que a sua. Se não for, a mensagem é ignorada e a unidade de detecção segue seu processo de sensoriamento.

Uma vez que a requisição é para sua ZI, a UDE deve obter o valor do sinal solicitado e enviar de volta para a unidade que fez a solicitação. Ao receber uma mensagem de

alerta, o MPC irá gerar o seu conjunto de mensagens de requisição. No entanto, é possível que o sistema possua mais de uma unidade atuando como MPC, nesse caso, existiriam múltiplas requisições redundantes para o mesmo alerta. Diante desse contexto, visando evitar esse problema, foi definido que as unidades iriam enviar mensagens de resposta apenas para a UDE cuja requisição chegou primeiro.

Para exemplificar essa situação, considere que o *framework* atua com dois MPCs ativos, com ID 4 e 5, e que ambos geraram requisições para determinado alerta. Se a requisição da UDE de ID 5 chegar primeiro ao BPC, então a resposta para a requisição será enviada apenas para o ID 5. Para a UDE de ID 4 será enviada uma mensagem sinalizando que ele não receberá resposta para aquela requisição, dessa forma poderá seguir o seu processamento. Essa abordagem evita respostas redundantes na rede, além de impedir que conjuntos de dados repetidos sejam enviados para o processamento *Fuzzy*.

Ao detectar uma requisição para sua ZI, ela analisa o sinal que está sendo solicitado, verificando se o sensor está ativo em sua estrutura. Se estiver, é realizada a captação daquele sinal e seu valor é retornado a partir de uma mensagem de resposta à requisição. Essa mensagem é enviada via tópico *required_values*, como apresentado na Listagem 4.8.

Listagem 4.8: Exemplo de uma mensagem de resposta à requisição, enviada via tópico *required_values*.

```
{"id_request": 1,
 "id_node": 2,
 "sensor": "smoke",
 "value": 451,
 "id_node_req": 1}
```

A mensagem de resposta contém o ID da requisição, de modo que seja possível identificar qual requisição está sendo respondida; o ID da unidade de detecção, identificando a unidade que enviou a resposta; o ID da unidade de requisição para a qual essa resposta está direcionada; além do nome do sensor e seu respectivo valor.

A UDE que realizou a requisição dos dados, ao receber as mensagens de todas as unidades de detecção com os sinais solicitados, deverá realizar um processamento nos dados para verificar se as informações obtidas sugerem a existência de uma emergência ou apenas um comportamento normal do ambiente. Para isso, é necessário que o sistema consiga identificar quando todas as mensagens forem recebidas. Dessa forma, foi definido um *HashMap* contendo todas as mensagens que devem ser recebidas. As chaves representam os *IDs* das requisições realizadas, e os valores representam as mensagens que a unidade de detecção espera receber como resposta àquela requisição.

A partir da utilização da tabela de unidades inscritos na rede, Tabela 4.2, é possível saber, para todas as unidades de detecção, quais sensores estão ativos, além,

claro, de suas ZIs de atuação. Diante disso, para cada sinal solicitado no processo na requisição, é realizado um pré-processamento para descobrir quais unidades de detecção estão na ZI solicitada e possuem aquele determinado sensor. A partir disso é criada uma tupla de formatação ($\mathrm{ID}_{u_k} - S$), sendo S o nome do sinal e ID_{u_k} o ID da unidade. Essa tupla define que é esperado que a unidade de detecção u_k retorne uma resposta à requisição do sinal S. Ao final da criação das tuplas de cada um dos sinais da requisição, a estrutura de dados está pronta (Listagem 4.9), contendo representações de todas as respostas esperadas para cada requisição.

Listagem 4.9: Estrutura de dados contendo tuplas representando as mensagens que devem ser recebidas pela requisição 1.

```
{1: [(2, "temperature"), (2, "smoke"), (2, "uv")] }
```

A Listagem 4.9 ilustra todas as respostas esperadas para a requisição 1. É esperado que a unidade de detecção de ID 1 responda às requisições dos sensores de temperatura, fumaça e radiação UV.

Ao receber uma mensagem de resposta, via *requested_data* , a unidade de detecção que realizou a requisição utilizará as informações do ID da unidade que enviou a mensagem e o nome do sinal enviado, para localizar a tupla referente àquela mensagem e retirá-la do *HashMap*. Dessa forma, considerando que a unidade de detecção de ID 1 respondeu à requisição de temperatura, a tupla (1, '*temperatura*') é excluída da estrutura de dados. Seguindo esse *modus operandi*, quando não existir mais tuplas referenciando determinada requisição, significa que todas as mensagens foram recebidas, e então a unidade de detecção pode seguir com o processamento desses dados.

Além disso, todos os dados obtidos pelas respostas às requisições são armazenados em uma estrutura, de modo que eles possam ser processados depois. Essa estrutura é definida por um *HashMap* cuja chave é o ID da requisição e o valor é outro *HashMap* contendo todas as informações obtidas. A Listagem 4.10 exemplifica o uso dessa estrutura de dados.

Listagem 4.10: Exemplo do *HashMap* contendo os dados enviados como resposta à requisição.

```
{1: {"fire": {"temperature":[85, 63], "smoke":[451, 650],
    "uv":[10]}} }
```

O exemplo definido pela Listagem 4.10 define os dados recebidos pela requisição de ID 1, para um alerta de incêndio (*fire*). Para essa requisição foram recebidos dois valores de resposta para o sinal de temperatura e de fumaça, e um valor para o sinal de radiação UV.

4.2.5 Fusão de dados e envio para Controlador *Fuzzy*

Uma vez recebidos todos os valores de resposta a uma requisição, é necessário que, a unidade de detecção atuando como MPC, faça um processo de avaliação, de modo a identificar se aquele conjunto de dados deve ou não ser enviado para o controlador *Fuzzy*. Esse processo de avaliação consiste na aplicação de métodos de fusão de dados, como uma análise estatística, por exemplo. O objetivo é avaliar se o comportamento daquela zona de interesse de modo geral indica uma emergência ou se aquele sinal de alerta foi apenas um elemento isolado.

A aplicação do processo de fusão de dados, ao invés do envio direto dos dados para o processamento *Fuzzy*, contribui para o aumento da eficiência energética do *framework* proposto. O envio de dados via rádio é mais custoso do que o processamento dos sinais dentro da UDE. Dessa forma, garantir que os dados sejam enviados somente quando necessário e em quantidade reduzida, diminui potencialmente o consumo de energia do sistema.

No desenvolvimento desse *framework*, optou-se pela utilização de uma análise estatística por média simples. A avaliação consistiu na aplicação da média em cada um dos conjuntos de dados recebidos para a requisição em questão. Considerando o exemplo definido na Listagem 4.10, será aplicada a média em cada um dos três conjuntos de dados definidos para a requisição de ID igual a 1.

Os valores de média são comparados aos valores gatilhos definidos na descrição da emergência. Caso um desses valores seja maior do que o seu respectivo valor gatilho, o sistema considera que aquele conjunto de dados deve ser enviado para o controlador *Fuzzy*.

Os dados são enviados para o controlador *Fuzzy* a partir de uma mensagem publicada no tópico *hazard_data*. A mensagem em questão possui o ID da requisição e da ZI, o nome da emergência, o nome dos sensores e suas respectivas médias. A Listagem 4.11 ilustra como é a mensagem.

Listagem 4.11: Estrutura de dados contendo tuplas representando as mensagens que devem ser recebidas pela requisição 1

```
{"data": {"temperature": 74, "smoke":550.5, "uv":10},
 "emergency":"fire", "req_id":1, "region":2}
```

Uma vez enviada a mensagem, o controlador *Fuzzy* pode processar os dados e definir, a partir disso, a probabilidade daquela emergência estar ocorrendo naquele momento.

4.3 Controlador *Fuzzy*

O controlador *Fuzzy* é a última e a mais elaborada camada de processamento do sistema de detecção de emergência. A partir dos dados de entrada, ele retorna a probabilidade daquela emergência estar ocorrendo. A depender do resultado, mensagens de alerta serão enviadas, de modo a desencadear a tomada de ações de mitigação.

Neste trabalho, o controlador *Fuzzy* foi implementado a partir da utilização da biblioteca *scikit-fuzzy*. Como citado na seção 3.6.1, um sistema baseado em lógica *Fuzzy*, desenvolvido a partir da utilização do *scikit-fuzzy*, possui elementos essenciais para o seu funcionamento: universo, funções de pertinência e regras. Nessa seção, é definido como o administrador do sistema é capaz de definir todos esses elementos a partir da utilização de um arquivo de configuração.

O arquivo de configuração do controlador *Fuzzy* é um arquivo Python que contém a definição dos antecedentes, consequentes e seus intervalos de valores aceitáveis, além das regras, com todos os parâmetros necessários para a sua utilização pela biblioteca *scikit-fuzzy*. Esses elementos são salvos em um arquivo binário a partir do uso da biblioteca Pickle, permitindo que objetos sejam salvos no arquivo e carregados em outro *script* com todos os campos de dados mantidos.

Quatro funções principais estão descritas nesse arquivo:

- $generate\_antecedent\_mf$: responsável pela definição dos antecedentes e suas funções de pertencimento.
- $generate\_emergency\_mf$: responsável pela definição dos consequentes e suas funções de pertencimento.
- $generate\_universe\_range$: responsável pela definição pelo intervalo aceitável que os valores de cada uma das variáveis podem ter, tanto antecedentes quanto consequentes.
- $generate\_rules$: responsável pela geração das regras.

Nas próximas partes do trabalho serão descritos, de forma detalhada, cada uma dessas funções.

Geração dos antecedentes

A estrutura de dados que armazena as informações dos antecedentes é uma *HashTable*. As chaves representam o nome da variável e o valor contém outra *HashTable* com todas as informações necessárias para a criação dos conjuntos das variáveis, como o nome, o tipo de função de pertencimento e os parâmetros que a definem.

A Listagem 4.12 mostra como é definido no arquivo de configuração a variável *smoke* descrita na Listagem 3.2. O primeiro parâmetro é o nome do conjunto que está sendo criado, nesse caso sendo *low*, *medium* e *high*. Em seguida é definido o nome da função que será utilizada para definir esse conjunto, $fuzz.trimf$, e em seguida o valor dos parâmetros que serão passados para a criação dessa função no *scikit-fuzzy*.

Listagem 4.12: Definição dos antecedentes: conjuntos, função de pertencimento e respectivos parâmetros.

```
antecedent = {}

smoke_mf = [["low", "trimf", [[-400, 0, 400]]],
```

```
            ["medium", "trimf", [[0, 400, 800]]],
            ["high", "trimf", [[400, 800, 1200]]]]

antecedent["smoke"] = smoke_mf
```

Essa definição deve ser feita para todos os sinais presentes no sistema, de modo que ao receber os sinais, o controlador saiba como modelar seus conjuntos e suas funções de pertinência.

Consequentes

A definição dos consequentes é idêntica ao dos antecedentes. A Listagem 4.13 mostra como é definido no arquivo de configuração a variável referente à emergência *fire* descrita na Listagem 3.3. O primeiro parâmetro é o nome do conjunto que está sendo criado, nesse caso: *very low*, *low*, *medium*, *high* e *very high*. Em seguida, é definido o nome da função que será utilizada para definir esse conjunto, e em seguida o valor dos parâmetros que serão passados para as funções $fuzz.pimf$ e $fuzz.gaussmf$.

Listagem 4.13: Definição dos consequentes: conjuntos, função de pertencimento e respectivos parâmetros.

```
emerg = {}

fire_mf = [["very low", "pimf", [-20, 0, 1, 30]],
           ["low", "gaussmf", [25, 8]],
           ["medium", "gaussmf", [50, 8]],
           ["high", "gaussmf", [70, 8]],
           ["very high", "pimf", [65, 99, 101, 102]]]

emerg["fire"] = fire_mf
```

Essa definição também deve ser feita para todas as emergências presentes no sistema, de modo que ao receber os sinais, o controlador saiba como modelar seus conjuntos e suas funções de pertinência.

Definição do *range* do universo

A criação do intervalo que será utilizado na criação do universo dentro do controlador é definido a partir da utilização da função *range* da biblioteca NumPy, que permite a definição de um limite inferior e superior, retornando uma lista com todos os valores inteiros nesse intervalo.

Esse intervalo é definido para todas as variáveis, tanto antecedentes quanto consequentes. E todos os intervalos são armazenadas em um *HashMap*, para depois serem utilizados dentro da estrutura do controlador. A Listagem 4.14 mostra como seria

criado essa estrutura de dados caso fossem definidos apenas as variáveis *smoke* e *fire*.

Listagem 4.14: Criação dos intervalos que serão utilizados para configurar o universo de cada variável.

```
    signals_range = {}
    signals_range["smoke"] = np.arange(0, 801, 1)
    signals_range["fire"] =np.arange(0, 101, 1)
```

Definição das regras

Para que o controlador *Fuzzy* carregue as informações definidas no arquivo de configuração e interprete cada elemento que deverá ser usado para criar o objeto *Rule*, é necessário que a estrutura contendo essas informações possua um padrão bem definido. Sendo assim o padrão escolhido está representado na Listagem 4.15.

Listagem 4.15: Formato padrão para a definição das regras dentro do arquivo de configuração *Fuzzy*

```
{antecedents:{"signal1":"classe1",
              "signal2":"classe2"},
 emergency:{"emergency":"class_emg"}
```

Para facilitar o entendimento, pode-se compreender que a regra descrita pelo padrão genérico acima pode ser descrita como "**SE** *signal*1 for *class*1 **E** *signal*2 for *class*2 **ENTÃO** a probabilidade de ocorrência de *emergency* é *class_emg*". Seguindo esse padrão, as regras exemplificadas na Listagem 3.4 são descritas no arquivo de configuração como é apresentado na Listagem 4.16.

Listagem 4.16: Formato padrão para a definição das regras dentro do arquivo de configuração *Fuzzy*.

```
rules = []
rules.append({"antecedents":{"temperature":"low",
                             "smoke":"low"},
              "emergency":{"fire":"very low"}})
rules.append({"antecedents":{"temperature":"high",
                             "smoke":"high"},
              "emergency":{"fire":"high"}})
```

Cada um desses elementos, antecedentes, consequentes, universo e regras, são adicionados a uma lista e então salvos em um arquivo binário. Dessa forma, quando for a hora de executar o controlador *Fuzzy*, esses dados podem ser carregados e utilizados para a definição dos elementos necessários para o funcionamento correto do controlador utilizando *scikit-fuzzy*.

É importante ressaltar que, embora neste trabalho tenha sido utilizado o controlador Fuzzy devido às suas vantagens em cenários de incerteza, o usuário do *framework* pode implementar diferentes métodos para a identificação de emergências. Este trabalho não tem como escopo a realização de testes de desempenho do controlador Fuzzy. Portanto, se o usuário entender que o processo de detecção pode ser mais eficiente com um método diferente, ele pode implementá-lo no *framework*.

A metodologia descrita neste capítulo foi aplicada durante o desenvolvimento de um *framework* que automatiza a implementação de um sistema de detecção de emergências urbanas. Os passos do desenvolvimento visaram garantir a criação de um sistema que automatize a detecção de emergências de forma assertiva e que explore a agilidade da computação na borda da rede. Os resultados obtidos a partir da implementação desse *framework* são apresentados no Capítulo 5 desta dissertação.

Capítulo 5

Resultados e Discussões

Neste capítulo serão abordados de forma detalhada os testes desenvolvidos e aplicados no *framework* desenvolvido a partir da metodologia detalhada no capítulo 4. Esses testes tem como objetivo avaliar o comportamento do sistema em diversos cenários, buscando garantir o seu bom funcionamento. Foram definidos dois tipos de testes: teste modular e teste de integração do sistema. O teste modular tem como objetivo avaliar cada elemento do sistema, garantindo que cada parte do *framework* está funcionando corretamente. Já o teste de integração tem em vista avaliar o comportamento do sistema na sua totalidade, utilizando diversos cenários pré-determinados, e avaliando se o comportamento apresentado é o esperado.

5.1 Teste Modular do Framework

Nesta sessão está descrito o teste desenvolvido e aplicado a cada um dos módulos do *framework*: inscrição dos dados, configuração, sensoriamento, requisição e resposta à requisição. Para a execução dos testes foram desenvolvidas três unidades de detecção baseadas em duas plataformas de *hardware* distintas. Uma unidade foi desenvolvida utilizando *Raspberry Pi 2*, e outras duas utilizando *ESP8266*. Segue a especificação de ambas as plataformas de *hardwares* utilizadas:

- Raspberry Pi 2:
  - Frequência Clock: 1 GHz Quad-core ARM Cortex-A7
  - GPU: Dual Core VideoCore IV® Multimedia Co-Processor
  - Memória RAM: 1 GB
  - Outros: 10/100 BaseT Ethernet socket, 4 portas USB 2.0, 27 pinos de E/S, interface serial para câmera (CSI-2), interface serial para Display (DSI), expansão com cartão MicroSD

- ESP 8266
  - Frequência Clock: 80 MHz Tensilica L106
  - Memória RAM: 160 KB
  - Memória de Dados: 10 KB
  - Memória Flash: 16 MB
  - Outros: Wi-Fi 802.11 b/g/n nativo, 17 pinos de E/S, conversor analógico-digital de 10 bits, 4 canais de PWM (*Pulse-Width Modulation*)

Ambos os *ESP8266* foram implementados para se comportarem como BPCs, realizando sensoriamento e respondendo às requisições das UDEs de maior poder computacional. Já o *Raspberry Pi* foi implementado para executar as atribuições tanto de APC, quanto de MPC, realizando a requisição de dados e avaliando-os, decidindo se deve ser aplicado o processamento *Fuzzy* sobre eles.

As seções a seguir detalham os testes realizados e os resultados obtidos. Os resultados são apresentados por *logs* gerados pelo APC ou pelo BPC, a depender do teste. Esses *logs* mostram elementos específicos do comportamento da unidade de detecção, permitindo avaliar se o comportamento apresentado é o mesmo comportamento esperado para aquela unidade de detecção.

5.1.1 Testes de Inscrição

Para a inicialização do sistema, é necessário que uma unidade de detecção APC esteja conectada à rede. Essa unidade será a responsável pela inscrição das outras UDEs que desejam se conectar à rede. Dito isso, foram definidos testes visando avaliar o funcionamento do processo de inscrição do sistema. Para a realização dos testes foi utilizado o mesmo arquivo de configuração mostrado na Listagem 4.2.

O primeiro teste realizado foi um teste de inscrição das unidades de detecção, onde ambas são conectadas à rede em momentos diferentes. Com o *Raspberry Pi* (APC) já conectado à rede, é conectado a primeira unidade de detecção BPC. Ao receber a mensagem de inscrição, o *Raspberry Pi* acessa o arquivo de configuração, obtém as informações daquela unidade e adiciona ela na tabela de unidades de detecção. Assim que a primeira unidade está inscrita, é conectado o segundo BPC à rede e o processo de inscrição é repetido. A Figura 5.1 mostra o *log* emitido pelo APC durante o processo de inscrição de ambas as UDEs.

O arquivo de *log* ilustrado pela Figura 5.1 apresenta o conteúdo da tabela de unidades de detecção, inicialmente contendo apenas o APC, em seguida sinaliza que foi recebida uma mensagem de inscrição e, assim que a inscrição é finalizada, apresenta o conteúdo da tabela atualizada, agora com as informações da nova unidade de detecção já inscrita na rede. Depois são exibidas mensagens sinalizando a conexão de outra unidade de detecção e a sua inscrição. Feito isso, é mostrada novamente a tabela atualizada, agora com as duas novas unidades de detecção adicionadas a

```
** Detection unit successfully enrolled **

Updated detection units table:
              id_mac class  id_node  region  latitude  longitude  temperature  smoke  humidity   uv  vibration
0  00:15:5d:b5:af:7d   APC      1.0     2.0   12.0745   -71.3421          0.0    0.0       0.0  0.0        0.0


*** Enrollment message received ***
{"id_mac":"A8:48:FA:DC:BC:60"}


** Detection unit successfully enrolled **

Updated detection units table:
              id_mac class  id_node  region  latitude  longitude  temperature  smoke  humidity   uv  vibration
0  00:15:5d:b5:af:7d   APC      1.0     2.0   12.0745   -71.3421          0.0    0.0       0.0  0.0        0.0
1  A8:48:FA:DC:BC:60   BPC      2.0     2.0   13.0987   -74.1234          1.0    1.0       1.0  0.0        0.0


*** Enrollment message received ***
{"id_mac":"A8:48:FA:DD:3D:D1"}


** Detection unit successfully enrolled **

Updated detection units table:
              id_mac class  id_node  region  latitude  longitude  temperature  smoke  humidity   uv  vibration
0  00:15:5d:b5:af:7d   APC      1.0     2.0   12.0745   -71.3421          0.0    0.0       0.0  0.0        0.0
1  A8:48:FA:DC:BC:60   BPC      2.0     2.0   13.0987   -74.1234          1.0    1.0       1.0  0.0        0.0
2  A8:48:FA:DD:3D:D1   BPC      3.0     2.0   12.3456   -73.8765          1.0    1.0       0.0  0.0        0.0
```

Figura 5.1: *Log* exibindo o comportamento do sistema durante o processo de inscrição de unidades de detecção de forma alternada.

ela. A partir disso é possível verificar que o processo de inscrição, nas circunstâncias propostas, está sendo realizado corretamente.

O protocolo MQTT, responsável pela gestão das trocas de mensagens na rede, utiliza o protocolo TCP para o envio e recebimento das mensagens. Ao receber múltiplas mensagens em um mesmo tópico, as mensagens que chegam depois ficam armazenadas em *buffers* até que as mensagens anteriores tenham sido processadas. Essa abordagem permite que múltiplas unidades de detecção sejam capazes de se conectar simultaneamente, sem impactar negativamente no funcionamento da rede. Visando verificar esse comportamento, foi realizado um teste onde duas unidades de detecção se conectam à rede simultaneamente.

De modo que nenhuma unidade de detecção permanecesse inscrita, o sistema foi reiniciado. Feito isso, o segundo teste consistiu em ligar ambas as unidades BPC ao mesmo tempo, de modo que suas conexões à rede fossem realizadas simultaneamente. Era esperado que a unidade de detecção cuja mensagem chegasse primeiro ao APC fosse a primeira a ser processada. Dessa forma, enquanto essa mensagem estivesse sendo processada, a outra estaria armazenada em *buffer*. Assim que a inscrição fosse finalizada, o APC acessaria a mensagem no *buffer* e realizaria a inscrição da unidade de detecção representada por ela. A Figura 5.2 mostra o *log* emitido pelo APC nessas circunstâncias.

O *log* apresentado na Figura 5.2 mostra que o processo de inscrição foi realizado da forma prevista. A inscrição da primeira unidade é efetuada com sucesso, e assim que esse processo finaliza, é iniciada a inscrição da segunda unidade de detecção. Dessa forma, foi possível verificar que a inscrição de unidades de detecção que se conectem

```
** Detection unit successfully enrolled **

Updated detection units table:
              id_mac class  id_node  region  latitude  longitude  temperature  smoke  humidity   uv  vibration
0  00:15:5d:b5:af:7d   APC      1.0     2.0   12.0745   -71.3421          0.0    0.0       0.0  0.0        0.0


*** Enrollment message received ***
{"id_mac":"A8:48:FA:DD:3D:D1"}


** Detection unit successfully enrolled **

Updated detection units table:
              id_mac class  id_node  region  latitude  longitude  temperature  smoke  humidity   uv  vibration
0  00:15:5d:b5:af:7d   APC      1.0     2.0   12.0745   -71.3421          0.0    0.0       0.0  0.0        0.0
1  A8:48:FA:DD:3D:D1   BPC      2.0     2.0   12.3456   -73.8765          1.0    1.0       0.0  0.0        0.0


*** Enrollment message received ***
{"id_mac":"A8:48:FA:DC:BC:60"}


** Detection unit successfully enrolled **

Updated detection units table:
              id_mac class  id_node  region  latitude  longitude  temperature  smoke  humidity   uv  vibration
0  00:15:5d:b5:af:7d   APC      1.0     2.0   12.0745   -71.3421          0.0    0.0       0.0  0.0        0.0
1  A8:48:FA:DD:3D:D1   BPC      2.0     2.0   12.3456   -73.8765          1.0    1.0       0.0  0.0        0.0
2  A8:48:FA:DC:BC:60   BPC      3.0     2.0   13.0987   -74.1234          1.0    1.0       1.0  0.0        0.0
```

Figura 5.2: *Log* exibindo o comportamento do sistema durante o processo de inscrição de unidades de detecção de forma simultânea.

simultaneamente à rede funciona corretamente.

5.1.2 Teste de Configuração das UDEs

Assim que é realizada a inscrição dos dados, o APC envia para a unidade de detecção recém inscrita uma mensagem contendo as informações necessárias para a sua configuração. Visando avaliar se a unidade de detecção foi configurada da forma correta, foram implementados testes capazes de verificar se os atributos e os sensores foram definidos corretamente.

O arquivo de configuração (Listagem 4.2) é utilizado como base de comparação para verificar se a configuração foi feita sem erros. Os BPCs foram implementados de modo que fosse retornado ao usuário um *log* contendo o nome e os respectivos valores dos atributos que foram setadas no processo de configuração. Comparando os valores apresentados pelo *log* e os apresentados no arquivo de configuração é possível verificar se o processo de configuração está correto ou não.

A Figura 5.3 mostra que a unidade de ID MAC A8:48:FA:DD:3D:D1 foi configurada com ID 2 na rede, monitorando a ZI 2, com os sensores de temperatura e fumaça ativos, além de possuir a descrição da emergência que irá monitorar: incêndio. Já a unidade com ID MAC A8:48:FA:DC:BC:60 foi configurada com ID 3 na rede, monitorando também a ZI 2, com os sensores de temperatura, fumaça e umidade ativos, além de possuir a descrição das duas emergência que irá monitorar: incêndio e enchente.

Avaliando os valores definidos no arquivo de configuração e os valores dos atributos

```
Configuration message identified

ID MAC: A8:48:FA:DD:3D:D1

Region: 2
Detection Unit ID: 2

Sensor temperature active.
Sensor smoke active.
Sensor humidity inactive.
Sensor uv inactive.
Sensor vibration inactive.

Emergency: fire
   Sensor: temperature
   Threshold: 60.00
   Sensor: smoke
   Threshold: 500.00
```

(a) Unidade de detecção 1.

```
Configuration message identified

ID MAC: A8:48:FA:DC:BC:60

Region: 2
Detection Unit ID: 3

Sensor temperature active.
Sensor smoke active.
Sensor humidity active.
Sensor uv inactive.
Sensor vibration inactive.

Emergency: fire
   Sensor: temperature
   Threshold: 60.00
   Sensor: smoke
   Threshold: 500.00
Emergency: flood
   Sensor: humidity
   Threshold: 90.00
```

(b) Unidade de detecção 2.

Figura 5.3: *Logs* gerados pelas unidades de detecção BPC durante o processo de configuração.

dos BPCs após a configuração é possível afirmar que o processo de configuração das unidades é realizado corretamente.

5.1.3 Teste de sensoriamento

O processo de sensoriamento das unidades BPC consiste em obter os valores de sinais do ambiente e verificar se ele é maior do que o valor gatilho configurado. Uma vez identificado um valor fora do intervalo normal, é enviada uma mensagem para o MPC para que ele possa realizar o processo de requisição de mais informações. Para testar o funcionamento do sensoriamento, foi desenvolvido um teste capaz de avaliar o processo de verificação dos sinais obtidos durante o processo de monitoramento do ambiente (Equação 4.1).

A premissa básica do teste foi confirmar se o sistema estava identificando corretamente os sinais que apresentavam comportamento anômalo, ou seja, cujos valores eram maiores ou iguais ao valor gatilho. Para isso, foram definidos diferentes conjuntos de dados, cada conjunto apresentando valores que simulam o valor obtido por cada sensor durante o monitoramento (Tabela 5.1). Dessa forma, com o conhecimento prévio dos valores dos sinais e dos valores gatilhos definidos para cada emergência é possível avaliar se o sistema está identificando corretamente os valores anômalos. A Listagem 5.1 mostra a descrição das duas emergências designadas para as unidades realizarem o monitoramento durante o teste de sensoriamento.

Listagem 5.1: Descrição das emergências atribuidas a cada uma das unidades no teste de sensoriamento.

```
emergency = {"fire":{"temperature":60,
                     "smoke":500,
                     "uv":10},
             "flood":{"humidity":90,
                      "vibration":1}}
             }
```

Os referidos conjuntos de dados foram definidos diretamente no código-fonte gravado em cada unidade de detecção. Assim, ao ler e atribuir os valores desses conjuntos às respectivas variáveis de cada sensor, tem-se a simulação de um cenário de monitoramento. Essa abordagem é um possível caminho para contornar as dificuldades de testar um sistema de detecção de emergências, já que não é trivial ou pode não ser seguro simular um incêndio ou uma enchente, por exemplo.

Tabela 5.1: Tabela apresentando os dados utilizados nos testes. Valores em vermelho representam valores que estão acima do valor gatilho.

Sensores	**Dados 1**	**Dados 2**	**Dados 3**
temperature	**71**	29	**62**
smoke	**650**	150	100
uv	5	**12**	4
humidity	30	**92**	50
vibration	0	**1**	0

Para esse teste, o arquivo de configuração foi alterado de modo que todos os sensores estivessem ativos. Na Tabela 5.1 todos os valores que estão acima do valor gatilho definidos pelas emergências monitoradas, portanto, os valores que o sensor deve identificar como anômalos, estão em vermelho. Para verificar se a unidade de detecção está se comportando da forma esperada, a mesma foi implementada de modo que durante o processo de sensoriamento seja gerado um *log* com as informações dos sinais que apresentaram comportamento anormal, além da mensagem de alerta enviada para o MPC (Figura 5.4).

```
*** Sensoring Process ***

Identified signs with anomalous behaviour:

- Emergency: fire
   - Sensor temperature: 71.00

   - Sensor smoke: 650.00

*** Sensoring Message ***

{"emergency":{"fire":{"sensor":["temperature","smoke"],"value":[71,650]}},
 "region":2,"id_node":2,"timestamp":1704419644}
```

(a) Conjunto de Dados 1.

```
*** Sensoring Process ***

Identified signs with anomalous behaviour:

- Emergency: fire
   - Sensor uv: 12.00

- Emergency: flood
   - Sensor humidity: 92.00
   - Sensor vibration: 1.00

*** Sensoring Message ***

{"emergency":{"flood":{"sensor":["humidity","vibration"],"value":[92,1],},
              "fire": {"sensor":["uv"],"value":[12]}},
 "region":2,"id_node":2,"timestamp":1704420867}
```

(b) Conjunto de Dados 2.

```
*** Sensoring Process ***

Identified signs with anomalous behaviour:

- Emergency: fire
   - Sensor temperature: 62.00

*** Sensoring Message ***

{"emergency":{"fire":{"sensor":["temperature"],"value":[62]}},
 "region":2,"id_node":2,"timestamp":1704419974}
```

(c) Conjunto de Dados 3.

Figura 5.4: *Log* de dados gerado pelo BPC durante o processo de sensoriamento. Cada uma dos *logs* foi gerado a partir da utilização de diferentes conjuntos de dados definidos na Tabela 5.1

A partir da análise dos *logs*, é possível verificar que todos os sinais que apresentaram

valores maiores ou iguais ao valor gatilho, foram reconhecidos como anômalos. Além disso, esses sinais, junto com as outras informações necessárias, foram adicionados à mensagem de alerta enviada para a unidade de detecção MPC via tópico *sensoring*.

5.1.4 Obtenção de dados

Uma vez recebida a mensagem de alerta sobre um valor anormal, a unidade MPC deve solicitar mais informações acerca da zona de interesse da qual veio a mensagem. Dessa forma, é possível identificar comportamentos que possam indicar a existência ou não de uma emergência. Para saber quais variáveis devem ser solicitadas às unidades de detecção é utilizada a descrição da emergência presente no arquivo de configuração. A descrição contém os sinais que a compõe e seus respectivos valores gatilhos. No caso de um alerta, devem ser solicitados todos os sinais que compõe a emergência para a qual o alerta se refere.

Cada mensagem de requisição de dados contém em sua estrutura o ID da zona de interesse para a qual ela está direcionada, o ID da requisição e o nome do sensor que está sendo solicitado. Cada unidade de detecção que esteja monitorando a zona de interesse sinalizada na mensagem e tenha aquele sensor ativo deve responder à requisição. A resposta consiste na unidade de detecção realizar a captação do sinal solicitado e enviá-lo para o MPC via tópico *request_data*.

Diante dessas informações, é possível dizer que o processo de obtenção de dados pode ser dividido em duas partes: requisição dos dados e respostas à requisição. Nessa seção serão detalhados os testes elaborados para avaliar o comportamento do sistema nesses dois momentos. Os testes foram elaborados com o intuito de avaliar o comportamento da realização da requisição e resposta a ela, utilizando diferentes cenários de configuração para as unidades de detecção. Os cenários são:

1. BPCs monitorando zonas de interesse diferentes.
2. BPCs monitorando zonas de interesse iguais, sem sensores em comum.
3. BPCs monitorando zonas de interesse iguais e com conjuntos de sensores iguais.

Para cada um dos cenários descritos é necessário que o sistema seja reconfigurado, e isso é feito a partir da utilização de diferentes arquivos de configuração. A Tabela 5.2 mostra o estado de configuração de cada unidade de detecção em cada um desses cenários. Para todos os cenários, as unidades monitoram a ocorrência de incêndios.

Tabela 5.2: Tabela apresentando como as unidades de detecção serão configuradas para satisfazer o esperado para cada cenário.

Cenário	ID	region	temperature	smoke	uv	humidity	vibration
1	2	1	√	√	X	√	X
	3	2	X	√	√	√	√
2	2	1	√	√	X	X	X
	3	1	X	X	X	√	√
3	2	1	√	√	X	√	X
	3	1	√	√	X	√	X

Para a avaliação de todos os cenários definidos na Tabela 5.2 as unidades de detecção foram implementadas de modo a simular um cenário de monitoramento em que fosse gerada a mensagem de alerta apresentada na Listagem 5.2. Essa mensagem indica que a unidade de detecção 2 identificou comportamento anormal nos sinais de temperatura e fumaça.

Listagem 5.2: Mensagem de sensoriamento tomada como base para a geração das mensagens de requisição

```
{ "id_node": 2,
 "region": 1,
 "timestamp": 1680888655},
 "emergency":{"fire":{"sensor":["temperature", "smoke"],
                      "value":[60, 500]}}
}
```

A partir dessas informações, as subseções subsequentes detalham os testes elaborados para validar o funcionamento do sistema durante a geração das mensagens de requisição e durante o processo de resposta a essa requisição.

Requisição de dados

A requisição de dados do *framework* é composta por duas partes principais: o envio da mensagem de requisição e a criação da estrutura de dados contendo todas as respostas que o MPC espera receber para aquela requisição. O teste detalhado nesta subseção foi elaborado para avaliar o funcionamento de ambas as partes da execução.

O MPC foi implementado de modo que fosse produzido um *log* sinalizando quais sensores serão requisitados e quais foram as mensagens enviadas, além de mostrar o conteúdo da *HashMap* contendo todas as respostas àquela requisição que a unidade espera receber. A partir do *log* foi possível comparar o comportamento do sistema com o que era esperado.

Uma vez que, para todas as execuções, o alerta recebido é da mesma região e possui os mesmos sinais, temperatura e fumaça, os sinais relacionados que serão requisitados também serão os mesmos, e consequentemente, as mensagens de requisição. A Figura 5.5 mostra a parte do *log* referente aos sinais requisitados e suas respectivas mensagens de requisição.

```
 Set of sensors to be requested :
    ->{'uv', 'smoke', 'temperature'}


Request message:
   -> {"id_request": 1, "region": 1, "sensor": "uv"}

Request message:
   -> {"id_request": 1, "region": 1, "sensor": "smoke"}

Request message:
   -> {"id_request": 1, "region": 1, "sensor": "temperature"}
```

Figura 5.5: *Log* exibindo os sinais que serão requisitados e suas respectivas mensagens de requisição.

A segunda parte do *log*, informando as mensagens que o MPC espera receber, deve variar a depender do cenário de configuração utilizado. Nos dois primeiros cenários, as mensagens esperadas são as mesmas, porque no Cenário 1, as unidades de detecção estão monitorando zonas de interesse diferentes, logo, com a requisição sendo feita para a zona 1, apenas a unidade 2 irá responder à requisição. Já no Cenário 2, mesmo com ambas as unidades monitorando a mesma zona de interesse, apenas a unidade 2 possui os sensores dos sinais requisitados. Por esse motivo, é novamente esperado que apenas a unidade 2 responda à requisição.

Já no Cenário 3 é esperado que sejam recebidas mensagens das duas unidades de detecção cadastradas. Isso porque nesse cenário, ambas estão monitorando a mesma zona, e possuem os sensores dos sinais requisitados. A Figura 5.6 mostra o *log* gerado pelo sistema durante sua execução em ambos os cenários, nele é possível verificar que em todos os cenários de configuração o *framework* se comporta da maneira esperada.

```
Set of expected messages:
  → {1: [('temperature', 2.0), ('smoke', 2.0)]}
```

(a) Respostas esperadas pelo MPC nos cenários 1 e 2.

```
Set of expected messages:
  → {1: [('temperature', 2.0), ('temperature', 3.0), ('smoke', 2.0), ('smoke', 3.0)]}
```

(b) Respostas esperadas pelo MPC no cenário 3.

Figura 5.6: *Log* exibindo as respostas à requisição esperadas para cada cenário.

Resposta à requisição

A segunda parte do processo de obtenção de dados consiste na resposta à requisição. Esse processo também pode ser dividido em duas partes, sendo elas o envio pelo BPC das respostas à requisição e o controle das mensagens já recebidas pelo MPC. Os testes elaborados nesta subseção tem como objetivo avaliar a gestão das respostas enviadas por cada unidade de detecção, verificando se estão se comportando da forma esperada.

Para avaliar o envio das respostas pelos BPCs e o controle das mensagens já recebidas, o MPC foi implementado de modo que durante sua execução fosse gerado um *log* apresentando informações importantes sobre o comportamento do *framework* durante o processo de resposta à requisição. Essas informações apresentadas nos logs equivalem a cada uma das mensagens que os BPCs enviam como resposta à requisição, e o conteúdo da estrutura de dados contendo as respostas que o sistema ainda espera receber.

Na subseção anterior, vimos que as requisições nos Cenários 1 e 2 resultam nas mesmas mensagens de resposta para o MPC. Em ambos os casos, esperávamos receber duas mensagens da unidade de detecção 2, uma contendo o valor do sensor de temperatura e outra do sensor de fumaça. A Figura 5.7 mostra que essas duas mensagens foram recebidas conforme o esperado. Quando cada mensagem é recebida, o MPC atualiza o estado das respostas recebidas e exibe as que ainda faltam chegar. É possível ver que, após receber a segunda mensagem, ambas estão no conjunto de mensagens recebidas e não há mais mensagens esperadas para aquela requisição. Isso indica que todas as mensagens foram recebidas com sucesso.

```
** Response to request received **
   {'sensor': 'temperature', 'value': 60, 'id_node': 2, 'region': 2, 'id_request': 1, 'id_node_req': 1}

Messages already received:
   → {1: {'emergency': 'fire', 'temperature': [60], 'smoke': [], 'uv': []}}

Set of expected messages:
   → {1: [('smoke', 2.0)]}


** Response to request received **
   {'sensor': 'smoke', 'value': 500, 'id_node': 2, 'region': 2, 'id_request': 1, 'id_node_req': 1}

Messages already received:
   → {1: {'emergency': 'fire', 'temperature': [60], 'smoke': [500], 'uv': []}}

Set of expected messages:
   → {1: []}

All responses to request 1 received
```

Figura 5.7: *Log* de dados gerado pelo MPC durante o processo de recebimento dos valores de resposta à requisição para os cenários 1 e 2.

Na execução utilizando o Cenário 3, é esperado que sejam recebidas quatro mensagens, duas da unidade de detecção 2 e duas da unidade de detecção 3. A Figura 5.8 mostra o recebimento de cada uma dessas mensagens e, assim como na utilização dos cenários anteriores, mostra os conjuntos de mensagens já recebidas e as que ainda devem ser recebidas. É importante notar que a cada vez que uma mensagem

é recebida, a tupla equivalente a ela no conjunto de mensagens a receber é retirada da lista. Dessa forma, assim que a última mensagem chegar, a lista estará vazia, indicando que todas as respostas àquela requisição já foram recebidas.

```
*** Response to request received ***
   {'sensor': 'temperature', 'value': 70, 'id_node': 2, 'region': 2, 'id_request': 1, 'id_node_req': 1}

Messages already received:
   → {1: {'emergency': 'fire', 'temperature': [70], 'smoke': [], 'uv': []}}

Set of expected messages:
   → {1: [('temperature', 3.0), ('smoke', 2.0), ('smoke', 3.0)]}


*** Response to request received ***
   {'sensor': 'temperature', 'value': 60, 'id_node': 3, 'region': 2, 'id_request': 1, 'id_node_req': 1}

Messages already received:
   → {1: {'emergency': 'fire', 'temperature': [70, 60], 'smoke': [], 'uv': []}}

Set of expected messages:
   → {1: [('smoke', 2.0), ('smoke', 3.0)]}


*** Response to request received ***
   {'sensor': 'smoke', 'value': 650, 'id_node': 2, 'region': 2, 'id_request': 1, 'id_node_req': 1}

Messages already received:
   → {1: {'emergency': 'fire', 'temperature': [70, 60], 'smoke': [650], 'uv': []}}

Set of expected messages:
   → {1: [('smoke', 3.0)]}


*** Response to request received ***
   {'sensor': 'smoke', 'value': 500, 'id_node': 3, 'region': 2, 'id_request': 1, 'id_node_req': 1}

Messages already received:
   → {1: {'emergency': 'fire', 'temperature': [70, 60], 'smoke': [650, 500], 'uv': []}}

Set of expected messages:
   → {1: []}

All responses to request 1 received
```

Figura 5.8: *Log* de dados gerado pelo MPC durante o processo de recebimento dos valores de resposta à requisição para o cenário 3.

A partir da avaliação de ambos os *logs*, é possível perceber que o sistema está se comportando do modo esperado, recebendo todas as respostas das requisições realizadas e identificando quando todas as mensagens foram recebidas. Desse ponto em diante o MPC poderá seguir seu processamento e avaliar os dados recebidos para saber se devem ser enviados ao controlador *Fuzzy* ou não.

5.2 Teste de integração do sistema

Uma vez realizado o teste modular e confirmado que os processos de inscrição, configuração, obtenção de dados e envio dos dados para o *Fuzzy* estão funcionando corretamente, foi elaborado um teste para avaliar o desempenho do sistema de forma geral.

Para esse teste foram utilizadas 4 unidade de detecção, uma atuando como APC e MPC, e as outras três atuando como BPCs. Para este propósito, foi criado um

conjunto de dados composto por valores pré-estabelecidos para cada sinal, onde cada etapa de sensoriamento resulta na obtenção e retorno de uma medida específica do sensor correspondente. O conjunto de dados foi modelado com o intuito de simular diversas situações, incluindo a captação de sinais normais, sinais anômalos que não evoluíram para o processamento *Fuzzy* e sinais anômalos que foram processados pelo controlador *Fuzzy*. Durante o teste, foi definido que apenas uma unidade de detecção realizaria o sensoriamento, as outras duas desempenhariam, exclusivamente, a função de responder possíveis requisições. A descrição das regras e funções de pertencimento utilizadas pelo controlador *Fuzzy* nesse experimento estão descritas no Apêndice B.

O teste foi realizado para dois arquivos de configuração diferentes. O primeiro define que as unidades irão monitorar apenas a emergência incêndio, tendo, cada uma, os sensores de temperatura, fumaça e radiação UV ativados. No segundo arquivo de configuração, as UDEs irão monitorar tanto incêndio quanto enchente, possuindo todos os cinco sensores ativados. A descrição das emergências é a mesma que a definida no arquivo de configuração descrito na Listagem 4.2.

Para avaliar o comportamento esperado com o que realmente foi apresentado pela execução do *framework*, o código-fonte dos APCs foi alterado de modo que durante a execução ele gere um log de dados apresentando elementos importantes como: os dados obtidos pela requisição, os valores resultantes da fusão de dados e sua comparação com os valores gatilhos da emergência e, por fim, se esses dados fossem enviados para o *Fuzzy*, o conjunto de dados utilizado como entrada e a resposta final do controlador. Uma vez que os dados foram pré-definidos, o comportamento esperado para cada momento da execução já é conhecido, dessa forma, é possível que o resultado real e o esperado sejam comparados, permitindo que seja feita a avaliação do comportamento do sistema nos cenários apresentados.

A Tabela 5.3 mostra o conjunto de dados pré-definido para o cenário onde o sistema foi configurado para monitorar apenas uma emergência. Nela são definidos os valores que serão captados nos momentos de sensoriamento, e os valores que serão retornados como resposta às requisições, caso tenha sido gerado um sinal de alerta. Os sinais que devem gerar sinais de alerta e, consequentemente, requisições, estão definidos na cor vermelha, enquanto as respostas à requisição, apresentadas nas linhas “Resposta”, estão em azul.

Tabela 5.3: Tabela apresentando os dados utilizados no teste com o arquivo de configuração com apenas uma emergência. Números em vermelho representam valores que estão acima do valor gatilho, e valores em azul representam as respostas à requisição.

	Sensors	**temperature**	**smoke**	**uv**
Medição 1	BPC 1	27	150	3
	BPC 2	-	-	-
	BPC 3	-	-	-
Medição 2	BPC 1	**70**	180	4
	BPC 2	-	-	-
	BPC 3	-	-	-
Resposta 1	BPC 1	**65**	**180**	**4**
	BPC 2	**15**	**90**	**2**
	BPC 3	**12**	**195**	**2**
Medição 3	BPC 1	22	150	4
	BPC 2	-	-	-
	BPC 3	-	-	-
Medição 4	BPC 1	32	**650**	6
	BPC 2	-	-	-
	BPC 3	-	-	-
Resposta 2	BPC 1	**50**	**650**	**2**
	BPC 2	**75**	**600**	**7**
	BPC 3	**72**	**710**	**8**

Na primeira medição, o BPC 1 não deve detectar nenhum valor medido maior do que os gatilhos da emergência. Já na Medição 2, deve ser emitido um alerta para o sinal de temperatura, gerando, como consequência, requisições para os outros sensores que definem aquela emergência. A linha da Resposta 1 mostra os valores que devem ser enviados como resposta à requisição por cada uma das unidades de detecção. Após a aplicação do processo de fusão de dados é esperado que nenhum sinal apresente valores acima do gatilho, não justificando o envio dos dados para o controlador *Fuzzy* (Figura 5.9).

A Medição 3, assim como a 1, também não apresenta valores anômalos. Já a Medição 4 apresenta valor anômalo para o sinal de fumaça, gerando requisições cujas respostas estão apresentadas em Resposta 2. Essas respostas apresentam valores significativamente acima dos valores gatilhos, por esse motivo eles devem ser enviados para o controlador *Fuzzy*. A Figura 5.9 mostra, além da avaliação dos valores pós-fusão de dados, os dados enviados para o *Fuzzy* e o resultado final do seu processamento.

Visando avaliar o comportamento do sistema quando o monitoramento da zona de interesse envolvesse mais de uma emergência, o conjunto de dados definido na

```
** Data obtained from requisition **
{'temperature': [12.0, 65.0, 15.0], 'smoke': [180.0, 195.0, 90.0], 'uv': [4.0, 2.0, 2.0]}


- Threshold for temperature: 60.0
  - Value obtained after data fusion: 30.7

- Threshold for smoke: 500.0
  - Value obtained after data fusion: 155.0

- Threshold for uv: 9.0
  - Value obtained after data fusion: 2.7

It was identified 0 signal(s) with value(s) higher than the threshold

No need to send this data to the Fuzzy Controller
```

Figura 5.9: *Log* sinalizando os dados recebidos como resposta à requisição gerada pelo alerta da Medição 2, além dos valores obtidos após a fusão de dados e informação sobre a execução ou não do controlador *Fuzzy*.

```
** Data obtained from requisition **
{'temperature': [72.0, 75.0, 50.0], 'smoke': [650.0, 600.0, 710.0], 'uv': [8.0, 7.0, 2.0]}


- Threshold for temperature: 60.0
  - Value obtained after data fusion: 65.7

- Threshold for smoke: 500.0
  - Value obtained after data fusion: 653.3

- Threshold for uv: 9.0
  - Value obtained after data fusion: 5.7

It was identified 2 signal(s) with value(s) higher than the threshold

Sending data to Fuzzy controller


**  Data received at Fuzzy Controller **


Input Data for Fuzzy Controller:
{'temperature': 65.7, 'smoke': 653.3, 'uv': 5.7}

There is an 86.26% probability of a fire occurring
```

Figura 5.10: *Log* sinalizando os dados recebidos como resposta à requisição gerada pelo alerta da Medição 4, além dos valores obtidos após a fusão de dados e informação sobre a execução ou não do controlador *Fuzzy*.

Tabela 5.3 foi estendido, englobando os sensores de umidade e vibração, que caracterizam a emergência "enchente".

Assim como no teste anterior, a Medição 1 não deve apresentar valores anômalos, enquanto a Medição 2 indica um valor anômalo para o sinal de temperatura, considerando a emergência incêndio, resultando na emissão de um alerta. Dessa forma, requisições devem ser geradas para os sinais caracterizadores da emergência, a linha da Resposta 1 mostra os dados que devem ser enviados pelas BPCs como resposta a essas requisições. A fusão dos dados não deve indicar comportamento suspeito, fazendo com que os dados não sejam enviados para o controlador *Fuzzy*. A Figura 5.11 mostra os valores enviados como resposta à requisição e o processo de fusão de dados.

Na Medição 3 não devem ser identificados valores anômalos. Já a Medição 4 deve acusar valores de umidade altos, quando comparados com o valor gatilho da emergên-

Tabela 5.4: Tabela apresentando os dados utilizados no teste com o arquivo de configuração com duas emergências. Valores em vermelho representam valores que estão acima do valor gatilho, e valores em azul representam as respostas à requisição.

	Sensors	**temperature**	**smoke**	**uv**	**humidity**	**vibration**
Medição 1	BPC 1	27	150	3	30	0
	BPC 2	-	-	-	-	-
	BPC 3	-	-	-	-	-
Medição 2	BPC 1	**70**	180	4	35	0
	BPC 2	-	-	-	-	-
	BPC 3	-	-	-	-	-
Resposta 1	BPC 1	**65**	**180**	**4**	-	-
	BPC 2	**15**	**90**	**2**	-	-
	BPC 3	**12**	**195**	**2**	-	-
Medição 3	BPC 1	30	200	6	40	0
	BPC 2	-	-	-	-	-
	BPC 3	-	-	-	-	-
Medição 4	BPC 1	22	140	4	**95**	1
	BPC 2	-	-	-	-	-
	BPC 3	-	-	-	-	-
Resposta 2	BPC 1	-	-	-	**95**	**1**
	BPC 2	-	-	-	**88**	**1**
	BPC 3	-	-	-	**92**	**1**
Medição 5	BPC 1	32	**650**	6	50	0
	BPC 2	-	-	-	-	-
	BPC 3	-	-	-	-	-
Resposta 3	BPC 1	**20**	**650**	**1**	-	-
	BPC 2	**35**	**600**	**2**	-	-
	BPC 3	**51**	**710**	**1**	-	-

cia enchente. Dessa forma, devem ser geradas requisições para os sinais de umidade e vibração. Os valores esperados de resposta (Resposta 2) devem resultar em valores, pós-fusão de dados, acima do limite aceitável, resultando no envio desses dados para o controlador *Fuzzy*. A Figura 5.12 mostra o comportamento do sistema nessa caso.

Por fim, os dados definidos para serem captados na Medição 5 são elaborados para a detecção de um valor anômalo de fumaça. Nesse caso, os valores de resposta às requisições geradas deveriam resultar no envio dos dados para o controlador *Fuzzy*. Para esse cenário é esperado que, após a fusão de dados, apenas o sinal de fumaça apresente comportamento acima do valor gatilho. Diante dos valores passados para o *Fuzzy*, prevê-se uma probabilidade baixa deles representarem a ocorrência de uma emergência A Figura 5.13 mostra o comportamento do *framework* para esse cenário.

Como foi citado anteriormente, foi decidido que apenas uma das três unidades BPCs

```
** Data obtained from requisition **
{'temperature': [12.0, 65.0, 15.0], 'smoke': [180.0, 195.0, 90.0], 'uv': [4.0, 2.0, 2.0]}


- Threshold for temperature: 60.0
  - Value obtained after data fusion: 30.7

- Threshold for smoke: 500.0
  - Value obtained after data fusion: 155.0

- Threshold for uv: 9.0
  - Value obtained after data fusion: 2.7

It was identified 0 signal(s) with value(s) higher than the threshold

No need to send this data to the Fuzzy Controller
```

Figura 5.11: *Log* sinalizando os dados recebidos como resposta à requisição gerada pelo alerta da Medição 2, além dos valores obtidos após a fusão de dados e informação sobre a execução ou não do controlador *Fuzzy*.

```
** Data obtained from requisition **
{'humidity': [92.0, 95.0, 88.0], 'vibration': [1.0, 1.0, 1.0]}


- Threshold for humidity: 90.0
  - Value obtained after data fusion: 91.7

- Threshold for vibration: 0.7
  - Value obtained after data fusion: 1.0

It was identified 2 signal(s) with value(s) higher than the threshold

Sending data to Fuzzy controller


**  Data received at Fuzzy Controller **


Input Data for Fuzzy Controller:
{'humidity': 91.7, 'vibration': 1.0}

There is an 89.66% probability of a flood occurring
```

Figura 5.12: *Log* sinalizando os dados recebidos como resposta à requisição gerada pelo alerta da Medição 4, além dos valores obtidos após a fusão de dados e informação sobre a execução ou não do controlador *Fuzzy*.

fariam o sensoriamento, enquanto que as outras apenas responderiam às requisições. Por esse motivo as tabelas não apresentam valores de medição para os BPCs 2 e 3.

Analisando os *logs* apresentados durante a execução do teste, pode-se verificar que, para todas as medições e respostas, o sistema se comportou da forma prevista. Dessa forma, é possível afirmar que o *framework*, desenvolvido a partir de uma abordagem hierárquica e colaborativa, é capaz de realizar o processo de monitoramento e detecção de múltiplas emergências, uma vez que elas estejam eficientemente modeladas no arquivo de configuração das UDEs e no arquivo de configuração do controlador *Fuzzy*. Dessa forma, o sistema proposto surge como uma importante ferramenta para detectar emergências urbanas de forma automática e confiável, diminuindo o tempo de resposta a situações de desastre e, consequentemente, aumentando as chances de diminuir os impactos catastróficos dessas emergências.

```
 *** Data obtained from requisition ***
 {'temperature': [35.0, 20.0, 51.0], 'smoke': [600.0, 710.0, 650.0], 'uv': [2.0, 1.0, 1.0]}


 - Threshold for temperature: 60.0
   - Value obtained after data fusion: 35.3

 - Threshold for smoke: 500.0
   - Value obtained after data fusion: 653.3

 - Threshold for uv: 9.0
   - Value obtained after data fusion: 1.3

 It was identified 1 signal(s) with value(s) higher than the threshold

 Sending data to Fuzzy controller


 ***  Data received at Fuzzy Controller ***


 Input Data for Fuzzy Controller:
{'temperature': 35.3, 'smoke': 653.3, 'uv': 1.3}

 There is an 38.23% probability of a fire occurring
```

Figura 5.13: *Log* sinalizando os dados recebidos como resposta à requisição gerada pelo alerta da Medição 5, além dos valores obtidos após a fusão de dados e informação sobre a execução ou não do controlador *Fuzzy*.

Capítulo 6

Conclusões

O crescimento acelerado da população urbana nas últimas décadas tem gerado uma série de emergências que desafiam a capacidade de gestão e resposta das cidades. Incêndios, enchentes, terremotos, deslizamentos, etc., são alguns exemplos de situações críticas que exigem estratégias eficientes de identificação e mitigação. Diante desse cenário, este trabalho teve como objetivo principal desenvolver um *framework* genérico e configurável para a detecção de emergências urbanas, baseado em redes de sensores sem fio (RSSF) heterogêneas, hierárquicas e colaborativas. O *framework* proposto é capaz de gerenciar múltiplas emergências simultaneamente, permitindo ao usuário definir as características das emergências a serem detectadas, bem como os parâmetros e regras para a sua detecção, e utiliza a computação na borda da rede para garantir a agilidade e a precisão do processo, além da eficiência energética. A implementação desenvolvida torna o sistema flexível e adaptável, podendo ser aplicado em diferentes contextos e regiões, contribuindo para a redução do tempo de resposta e dos impactos das emergências urbanas.

Para alcançar esse objetivo, foi realizada uma revisão bibliográfica sobre os conceitos e as técnicas relacionados à detecção de emergências urbanas, às redes de sensores sem fio e a inteligência artificial, especificamente Lógica *Fuzzy*. A partir dessa revisão, foi possível identificar as principais lacunas e desafios existentes na literatura, bem como as oportunidades e as potencialidades de cada abordagem. Em seguida, foi proposta a arquitetura do *framework*, detalhando os seus componentes, as suas funcionalidades e as suas interfaces. O *framework* foi modelado como um sistema hierárquico e colaborativo, composto por três tipos de unidades de sensoriamento. que são a unidades de baixo, médio e alto poder computacional. Cada tipo de unidade possui um papel específico no processo de detecção, sendo responsável por coletar, processar, transmitir e receber dados sobre as emergências, de acordo com as configurações definidas pelo usuário.

Para avaliar o desempenho do *framework*, foram realizadas simulações em cenários diversos. Primeiro foram aplicados testes modulares, que confirmaram a eficiência de cada parte do sistema: inscrição e configuração de UDEs, sensoriamento, obten-

ção de dados e processamento *Fuzzy*. Além disso, testes simulando situações reais, resultaram em comportamentos do sistema atendendo estritamente aos requisitos, e desempenhando como esperado a geração de sinais de alerta, obtenção dados e decisão de análise mais acurada pelo controlador *Fuzzy*. Os cenários de testes contemplaram diferentes tipos de emergências, como incêndios e enchentes, e diferentes configurações da rede de sensores, como o número de UDEs e sua configuração de *hardware*. Os cenários também incluíram situações onde múltiplas emergências eram monitoradas simultaneamente, a fim de testar a capacidade do *framework* de lidar com cenários complexos e dinâmicos.

Em síntese, esta dissertação contribui para a busca de soluções inovadoras no campo da detecção de emergências urbanas, introduzindo um *framework* eficaz, flexível e que, em busca de um comportamento energeticamente eficiente, explora a agilidade da computação na borda da rede. As simulações realizadas confirmam a viabilidade e eficácia do *framework* proposto, ressaltando seu potencial para melhorar a resiliência das cidades diante de eventos críticos. A implementação prática deste *framework* em ambientes urbanos reais representará um passo significativo em direção à construção de cidades mais seguras e preparadas para enfrentar os desafios emergentes do século XXI.

Como trabalhos futuros, sugere-se a realização de testes em campo a fim de validar o *framework* em cenários mais realistas e complexos. Para isso pretende-se utilizar dados reais, sejam eles dados históricos ou provenientes de emergências simuladas em ambientes reais. Também sugere-se a integração do *framework* com sistemas de alerta e de resposta a emergências, a fim de proporcionar uma solução completa e integrada para o gerenciamento de emergências urbanas. Além disso, sugere-se a ampliação do *framework* para incorporar novas técnicas e tecnologias, como outras técnicas de inteligência artificial, além da implementação do controlador *Fuzzy* para o processamento feito no nível do BPC, a fim de aprimorar o processo de detecção e de comunicação das emergências;

Referências

Abdelgawad, A. e Bayoumi, M. (2012). *Data Fusion in WSN*, páginas 17–35. Springer US, Boston, MA.

Abdelgawad, H. e Abdulhai, B. (2009). Emergency evacuation planning as a network design problem: a critical review. *Transportation Letters*, 1(1):41–58.

Al-Mimi, H., Al-Dahoud, A., Fezari, M., e Daoud, M. S. (2020). A study on new arduino nano board for wsn and iot applications. *International Journal of Advanced Science and Technology*, 29(4):10223–10230.

Albino, V., Berardi, U., e Dangelico, R. M. (2015). Smart cities: Definitions, dimensions, performance, and initiatives. *Journal of Urban Technology*, 22(1):3–21.

Ashton, K. (1999). That internet of things. *RFID journal*, 22.

Balakrishna, C. (2012). Enabling technologies for smart city services and applications. In *2012 Sixth International Conference on Next Generation Mobile Applications, Services and Technologies*, páginas 223–227.

Cheng, A. L., Georgoulas, C., e Bock, T. (2016). Fall detection and intervention based on wireless sensor network technologies. *Automation in Construction*, 71:116–136.

Cherifi, N., Boe, A., Vantroys, T., Herault, C., e Grimaud, G. (2018). A low-cost energy consumption measurement platform. In *Proceedings of the Workshop on INTelligent Embedded Systems Architectures and Applications*, páginas 7–12.

Cherifi, N., Grimaud, G., Boe, A., e Vantroys, T. (2016). Toward energy profiling of connected embedded systems. In *2016 8th IFIP International Conference on New Technologies, Mobility and Security (NTMS)*, páginas 1–4. IEEE.

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

Dima, S. M., Panagiotou, C., Tsitsipis, D., Antonopoulos, C., Gialelis, J., e Koubias, S. (2014). Performance evaluation of a wsn system for distributed event detection using fuzzy logic. *Ad Hoc Networks*, 23:87–108.

Dinculeană, D. e Cheng, X. (2019). Vulnerabilities and limitations of mqtt protocol used between iot devices. *Applied Sciences*, 9(5):848.

Driankov, D., Hellendoorn, H., e Reinfrank, M. (2013). *An introduction to fuzzy control*. Springer Science & Business Media.

Dutta, A. e Kant, S. (2021). Implementation of cyber threat intelligence platform on internet of things (iot) using tinyml approach for deceiving cyber invasion. In *2021 International Conference on Electrical, Computer, Communications and Mechatronics Engineering (ICECCME)*, páginas 1–6. IEEE.

Engelbrecht, A. P. (2007). *Computational intelligence: an introduction*. John Wiley & Sons.

Foo, S. (2000). A fuzzy logic approach to fire detection in aircraft dry bays and engine compartments. *IEEE Transactions on Industrial Electronics*, 47(5):1161–1171.

Gasparini, P., Di Ruocco, A., e Russo, R. (2014). Natural hazards impacting on future cities. *Resilience and Sustainability in Relation to Natural Disasters: A Challenge for Future Cities*, páginas 67–76.

Gatial, E., Balogh, Z., e Hluchỳ, L. (2020). Concept of energy efficient esp32 chip for industrial wireless sensor network. In *2020 IEEE 24th International Conference on Intelligent Engineering Systems (INES)*, páginas 179–184. IEEE.

Grigulo, J. e Becker, L. B. (2018). Experimenting sensor nodes localization in wsn with uav acting as mobile agent. In *2018 IEEE 23rd International Conference on Emerging Technologies and Factory Automation (ETFA)*, volume 1, páginas 808–815. IEEE.

Harrison, C., Eckman, B., Hamilton, R., Hartswick, P., Kalagnanam, J., Paraszczak, J., e Williams, P. (2010). Foundations for smarter cities. *IBM Journal of Research and Development*, 54(4):1–16.

Hashi, A., Abdirahman, A., Elmi, M., Hashi, S., e Rodriguez, O. (2021). A real-time flood detection system based on machine learning algorithms with emphasis on deep learning. *International Journal of Engineering Trends and Technology*, 69:249–256.

Hernafi, Y., Ahmed, M., e Bouhorma, M. (2017). Aco and pso algorithms for developing a new communication model for vanet applications in smart cities. *Wireless Personal Communications*, 96.

Hortelano, D., Olivares, T., e Ruiz, M. C. (2021). Providing interoperability in bluetooth mesh with an improved provisioning protocol. *Wireless Networks*, 27(2):1011–1033.

Huang, D., Wang, S., e Liu, Z. (2021). A systematic review of prediction methods for emergency management. *International Journal of Disaster Risk Reduction*, 62:102412.

Jesus, T. C., Portugal, P., Costa, D. G., e Vasques, F. (2020). A comprehensive dependability model for qom-aware industrial wsn when performing visual area coverage in occluded scenarios. *Sensors*, 20(22):6542.

Jesus, T. C., Portugal, P., Vasques, F., e Costa, D. G. (2018). Automated methodology for dependability evaluation of wireless visual sensor networks. *Sensors*, 18(8).

Kapitanova, K., Son, S. H., e Kang, K.-D. (2012). Using fuzzy logic for robust event detection in wireless sensor networks. *Ad Hoc Networks*, 10(4):709–722.

Khan, S., Won, J., Shin, J., Park, J., Park, J.-W., Kim, S.-E., Jang, Y., e Kim, D. J. (2021). Ssvm: An ultra-low-power strain sensing and visualization module for long-term structural health monitoring. *Sensors*, 21(6):2211.

Khelifi, F., Kaddachi, M. L., Bouallegue, B., e Soudani, A. (2014). Fuzzy logic-based hardware architecture for event detection in wireless sensor networks. In *2014 World Symposium on Computer Applications Research (WSCAR)*, páginas 1–4.

Kirimtat, A., Krejcar, O., Kertesz, A., e Tasgetiren, M. F. (2020). Future trends and current state of smart city concepts: A survey. *IEEE Access*, 8:86448–86467.

Kontokosta, C. E. e Malik, A. (2018). The resilience to emergencies and disasters index: Applying big data to benchmark and validate neighborhood resilience capacity. *Sustainable Cities and Society*, 36:272–285.

Korshikova, A. e Trofimov, A. (2019). Model for early detection of emergency conditions in power plant equipment based on machine learning methods. *Thermal Engineering*, 66(3):189–195.

Kruger, C. P. e Hancke, G. P. (2014). Benchmarking internet of things devices. In *2014 12th IEEE International Conference on Industrial Informatics (INDIN)*, páginas 611–616. IEEE.

Kurata, N., Spencer Jr, B., Ruiz-Sandoval, M., Miyamoto, Y., e Sako, Y. (2003). A study on building risk monitoring using wireless sensor network mica mote. *Strain*, 1:35.

La, Q. D., Ngo, M. V., Dinh, T. Q., Quek, T. Q., e Shin, H. (2019). Enabling intelligence in fog computing to achieve energy and latency reduction. *Digital Communications and Networks*, 5(1):3–9. Artificial Intelligence for Future Wireless Communications and Networking.

Lambrou, T. P., Anastasiou, C. C., Panayiotou, C. G., e Polycarpou, M. M. (2014). A low-cost sensor network for real-time monitoring and contamination detection in drinking water distribution systems. *IEEE sensors journal*, 14(8):2765–2772.

Li, Y., Hou, M., Liu, H., e Liu, Y. (2012). Towards a theoretical framework of strategic decision, supporting capability and information sharing under the context of internet of things. *Information Technology and Management*, 13.

Li, Z.-B. e Zhou, H. (2004). Research on the application of fuzzy data fusion to cable fire detecting system. In *Proceedings of 2004 International Conference on Machine Learning and Cybernetics (IEEE Cat. No.04EX826)*, volume 4, páginas 2083–2085 vol.4.

Liang, Q. e Wang, L. (2005). Event detection in wireless sensor networks using fuzzy logic system. In *CIHSPS 2005. Proceedings of the 2005 IEEE International Conference on Computational Intelligence for Homeland Security and Personal Safety, 2005.*, páginas 52–55.

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

Manrique, J. A., Rueda-Rueda, J. S., e Portocarrero, J. M. (2016). Contrasting internet of things and wireless sensor network from a conceptual overview. In *2016 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*, páginas 252–257.

Mikhaylov, K. e Tervonen, J. (2012). Evaluation of power efficiency for digital serial interfaces of microcontrollers. In *2012 5th International Conference on New Technologies, Mobility and Security (NTMS)*, páginas 1–5. IEEE.

Nalini, S. e Valarmathi, A. (2018). A collaborative composite event detection approach in wireless sensor network using fuzzy assisted decision system. *Applied Mathematics and Information Sciences*, 12(3):567–577.

Nastase, L. (2017). Security in the internet of things: A survey on application layer protocols. In *2017 21st international conference on control systems and computer science (CSCS)*, páginas 659–666. IEEE.

Nazir, A., Mosleh, H., Takruri, M., Jallad, A.-H., e Alhebsi, H. (2022). Early fire detection: A new indoor laboratory dataset and data distribution analysis. *Fire*, 5(1).

Ngo, M. V., Luo, T., Chaouchi, H., e Quek, T. Q. (2020). Contextual-bandit anomaly detection for iot data in distributed hierarchical edge computing. In *2020 IEEE 40th International Conference on Distributed Computing Systems (ICDCS)*, páginas 1227–1230.

Ortuño, M., Cristóbal, P., Ferrer, J., Martín-Campo, F., Muñoz, S., Tirado, G., e Vitoriano, B. (2013). Decision aid models and systems for humanitarian logistics. a survey. *Decision aid models for disaster management and emergencies*, páginas 17–44.

Parkinson, G., Crutchley, D., Green, P. M., Antoniou, M., Boon, M., Green, P. N., Green, P. R., Sloan, R., e York, T. (2010). Environmental monitoring in grain. In *2010 IEEE Instrumentation & Measurement Technology Conference Proceedings*, páginas 939–943. IEEE.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., e Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Pereira, E. M. (2014). Cidade, urbanismo e mobilidade urbana. *Revista do Departamento de Geociências - CFH/UFSC - Edição Especial: XXXV SEMAGEO*, 29.

Polastre, J., Szewczyk, R., e Culler, D. (2005). Telos: Enabling ultra-low power wireless research. In *IPSN 2005. Fourth International Symposium on Information Processing in Sensor Networks, 2005.*, páginas 364–369. IEEE.

Rajab, H. e Cinkelr, T. (2018). Iot based smart cities. In *2018 International Symposium on Networks, Computers and Communications (ISNCC)*, páginas 1–4.

Rajab, H. e Cinkler, T. (2018). Iot based smart cities. *2018 International Symposium on Networks, Computers and Communications (ISNCC)*, páginas 1–4.

Ramelan, A., Adriyanto, F., Hermanu, B., Ibrahim, M., Saputro, J., e Setiawan, O. (2021). Iot based building energy monitoring and controlling system using lora modulation and mqtt protocol. In *IOP Conference Series: Materials Science and Engineering*, volume 1096, página 012069. IOP Publishing.

Simitha, K. e Raj, S. (2019). Iot and wsn based water quality monitoring system. In *2019 3rd International conference on Electronics, Communication and Aerospace Technology (ICECA)*, páginas 205–210. IEEE.

Skeledzija, N., Cesic, J., Koco, E., Bachler, V., Vucemilo, H. N., e Džapo, H. (2014). Smart home automation system for energy efficient housing. In *2014 37th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, páginas 166–171. IEEE.

Sultan Mahmud, M., Islam, M., e Rahman, M. (2017). Smart fire detection system with early notifications using machine learning. *International Journal of Computational Intelligence and Applications*, 16(2). 00007.

Tanscheit, R. (2004). Sistemas fuzzy. *Departamento de Engenharia Elétrica, Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro*, páginas 338–353.

Tebeau, M. (2012). *Eating smoke: Fire in urban America, 1800–1950*. JHU Press.

Tran, M. A. T., Le, T. N., e Vo, T. P. (2018). Smart-config wifi technology using esp8266 for low-cost wireless sensor networks. In *2018 International Conference on Advanced Computing and Applications (ACOMP)*, páginas 22–28. IEEE.

Wang, M., Cao, J., Li, J., e Das, S. (2008). Middleware for wireless sensor networks: A survey. *J. Comput. Sci. Technol.*, 23:305–326.

Wu, X., Park, Y., Li, A., Huang, X., Xiao, F., e Usmani, A. (2021). Smart detection of fire source in tunnel based on the numerical database and artificial intelligence. *Fire Technology*, 57(2):657–682.

Zhou, L., Wu, X., Xu, Z., e Fujita, H. (2018). Emergency decision making for natural disasters: An overview. *International journal of disaster risk reduction*, 27:567–576.

Zimmermann, H.-J. (2010). Fuzzy set theory. *Wiley interdisciplinary reviews: computational statistics*, 2(3):317–332.

Apêndice A

Especificações de Unidades de *Hardware*

Nesse Apêndice, a Tabela A.1 apresenta descrições de diversas plataformas de *hardware* que podem ser utilizadas como unidades de detecção. Nela são sugeridas as categorias para as quais cada *hardware* pode ser utilizado, tomando como base as definições da Tabela 4.1.

As plataformas de *hardware* foram escolhidas a partir da análise de diversos artigos e trabalhos publicados na área de rede de sensores sem fio, a tabela apresenta as referências onde cada plataforma é utilizada. Não foram encontrados artigos ou publicações relevantes que abordassem a utilização de alguns *hardwares*, mas suas especificações os tornam uma boa alternativa para serem utilizados como unidades de detecção. Por esse motivo, eles também foram incluídos nesse contexto.

Tabela A.1: Lista de especificação de hardware.

Dispositivo	Categoria	Frequência de Processamento	Memória RAM	Memória Flash	Memória de Dados	Entradas Digitais	Entradas Analógicos	Conexão Wireless	Bluetooth	Micro SD	Referências
Mica Mote 2	BPC	4 MHz	128 KB	512 KB	-	8	-				[1]
Telos	BPC	8 MHz	10 KB	48 KB	1024 KB	48	8				[2]
Arduino Uno	BPC	16 MHz	2 KB	32 KB	1 KB	10	6				[5], [9]
Arduino Mega2560	BPC	16 MHz	8 KB	256 KB	4 KB	54	16				
PIC18F	BPC	48 MHz	16 KB	32 KB	2 MB	31	8				[12], [13]
Arduino Nano	BPC	16 MHz	2KB	32 KB	1 KB	14	8				[3]
PIC32	MPC	até 80 MHz	4 KB até 128 KB	32 KB até 512 KB	64 KB		16				[11], [14]
ESP32	MPC	até 240 MHz	520 KB	16 KB	512 B	25	15	√	√		[8], [9]
ESP8266	MPC	80 MHz	160 KB	16 MB	10 KB	17	1	√		√	[10]
Arduino Nano 33 BLE	MPC	64 MHz	256 KB	1 MB	-	14	8		√		[18], [3]
STM32F7 MCU	MPC	216 MHz	320 KB	1 MB	-	166	2			√	[15], [16]
Adafruit Edge-Badge	MPC	120 MHz (Boost 200 MHz)	192 KB	512 KB	-	19	16				[18]
Wio Terminal	MPC	120 MHz (Boost 200 MHz)	192 KB	512 KB	-	19	16	√	√	√	
Bluefruit - NRF52840	MPC	64 MHz	256 KB	1 MB	-	37	6				[17], [7]
BeagleBone Black	APC	1 GHz	512 MB	-	4 GB	69	7				[4], [5],[6]
BeagleBone Enhanced	APC	1 GHz	1 GB	-	4 GB	69	7	√	√		
Raspberry Pi	APC	até 1.5 GHz	1, 2 ou 4 GB	-	-	27	-				[6]

Apêndice B

Especificações do Controlador *Fuzzy*

Esse apêndice apresenta a forma como os parâmetros utilizados para implementação do controlador *Fuzzy*, utilizado no teste descrito na seção 5.2, foram definidos dentro do arquivo de configuração *Fuzzy*. Aqui serão descritas a parametrização dos antecedentes e consequentes, os seus intervalos de valores válidos e funções de pertinência, além, das regras que regem o processamento do controlador.

B.1 Intervalos válidos das variáveis

A Listagem B.1 mostra os intervalos de valores aceitáveis definidos para cada uma das variáveis envolvidas no processamento *Fuzzy*, tanto antecedentes, quanto consequentes, onde *np* é a biblioteca numPy.

Listagem B.1: Mensagem de sensoriamento tomada como base para a geração das mensagens de requisição

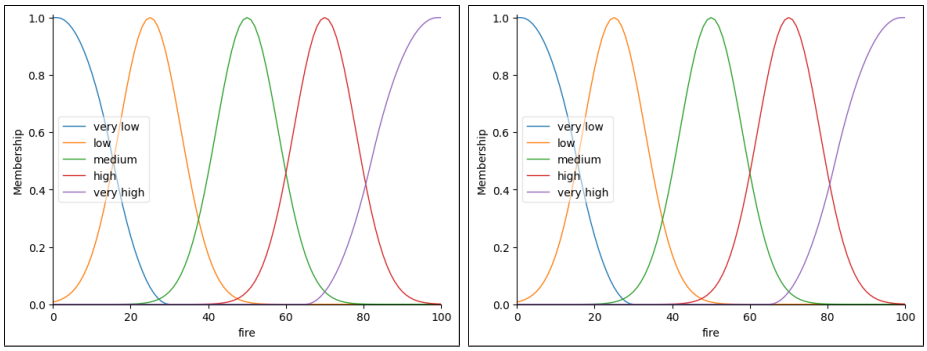
```
def generate_universe_range():
    signals_range = {}
    signals_range["temperature"] = np.arange(0, 101, 1)
    signals_range["smoke"] = np.arange(0, 801, 1)
    signals_range["uv"] = np.arange(0, 13, 1)
    signals_range["humidity"] = np.arange(0, 101, 1)
    signals_range["vibration"] = np.arange(0, 1.1, .1)
    signals_range["fire"] = np.arange(0, 101, 1)
    signals_range["flood"] = np.arange(0, 101, 1)

    return signals_range
```

B.2 Funções de pertencimento

Os consequentes são as emergências a serem monitoradas, nesse caso: incêndio (*fire*) e enchente (*flood*). A Listagem B.2 e a Figura B.1 apresentam o código do arquivo de configuração que definem os parâmetros das funções de pertencimento e suas representações gráficas, respectivamente.

Listagem B.2: Código presente no arquivo de configuração definindo os parâmetros que definem as funções de pertencimento dos consequentes

```
def generate_emergency_mf():
    emerg = {}
    fire_mf = [["very low", "pimf", [-20, 0, 1, 30]],
               ["low", "gaussmf", [25, 8]],
               ["medium", "gaussmf", [50, 8]],
               ["high", "gaussmf", [70, 8]],
               ["very high", "pimf", [65, 99, 101, 102]]]
    emerg["fire"] = fire_mf

    flood_mf = [["very low", "pimf", [-20, 0, 1, 30]],
               ["low", "gaussmf", [25, 8]],
               ["medium", "gaussmf", [50, 8]],
               ["high", "gaussmf", [70, 8]],
               ["very high", "pimf", [65, 99, 101, 102]]]
    emerg["flood"] = flood_mf
    return emerg
```



(a) Incêndio (b) Enchente

Figura B.1: Funções de pertencimento para as emergências incêndio e enchente.

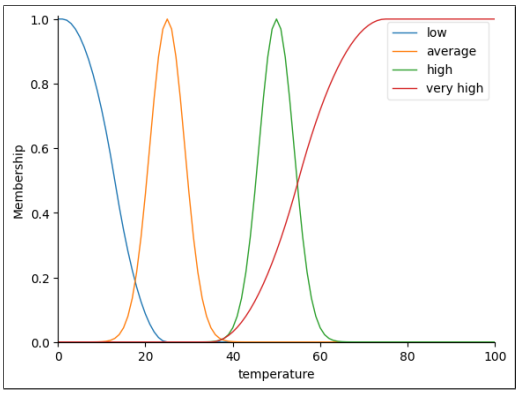
Os antecedentes representam os sinais que descrevem as emergências monitoradas pelo sistema, nesse caso: temperatura, fumaça e radiação UV (incêndio), e umidade e vibração (enchente). A Listagem B.3 e a Figura B.2 apresentam o código do arquivo de configuração que definem os parâmetros das funções de pertencimento e suas representações gráficas, respectivamente.

Listagem B.3: Código presente no arquivo de configuração definindo os parâmetros que definem as funções de pertencimento dos antecedentes

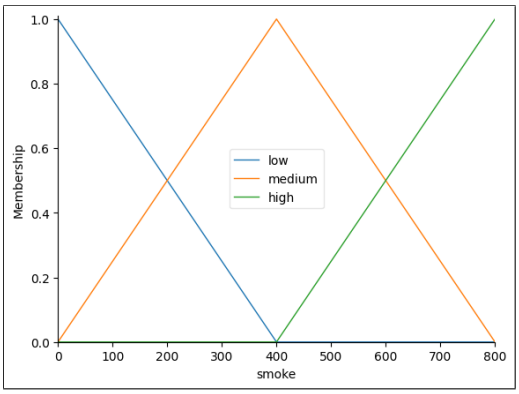
```
def generate_antecedent_mf():
    antecedent = {}

    temperature_mf =[["low", "pimf", [-20, 0, 1, 25]],
                    [ "average","gaussmf" ,[25, 4]],
                    [ "high","gaussmf", [50, 4]],
                    [ "very high","pimf" ,[35, 75, 101,
                       102]]]
    antecedent["temperature"] = temperature_mf

    smoke_mf = [["low", "trimf", [[-400, 0, 400]]],
                ["medium", "trimf", [[0, 400, 800]]],
                ["high", "trimf", [[400, 800, 1200]]]]
    antecedent["smoke"] = smoke_mf

    uv_mf = [["low", "trimf", [[-5, 0, 6]]],
             ["medium", "trimf", [[0, 6, 12]]],
             ["high", "trimf", [[6, 12, 17]]]]
    antecedent["uv"] = uv_mf

    humidity_mf =[["low", "pimf", [-20, 0, 1, 25]],
                  [ "average","gaussmf" ,[25, 4]],
                  [ "high","gaussmf", [50, 4]],
                  [ "very high","pimf" ,[35, 75, 101, 102
                    ]]]
    antecedent["humidity"] = humidity_mf

    vibration_mf = [["low", "trimf", [[-.5, 0, .5]]],
                    ["medium", "trimf", [[0, .5, 1.0]]],
                    ["high", "trimf", [[.5, 1.0, 1.5]]]]
    antecedent["vibration"] = vibration_mf
    return antecedent
```

B.3 Regras do controlador *Fuzzy*

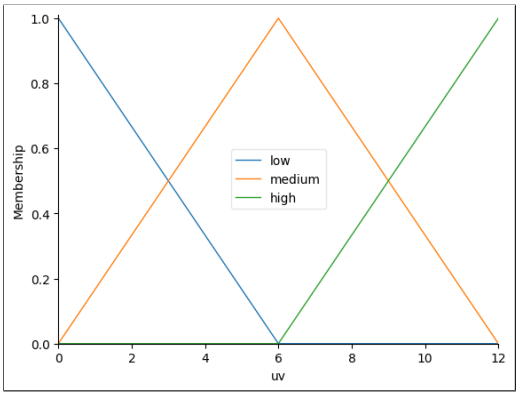
As regras definem o comportamento do controlador. Logo, elas foram elaboradas de modo que a relação entre antecedentes e consequentes se aproximassem o máximo possível da descrição real das emergências monitoradas. A Listagem B.4 mostra como as regras foram definidas no arquivo de configuração.



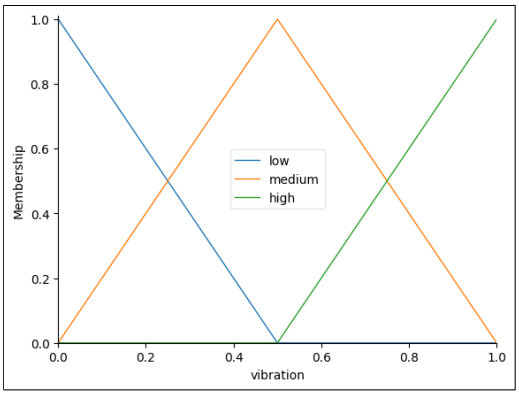
(a) Temperatura



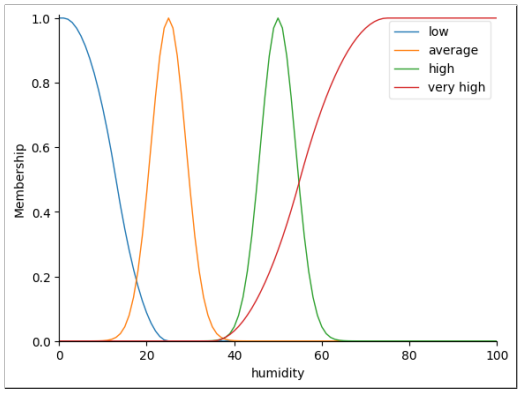
(b) Fumaça



(c) Radiação UV



(d) Vibration



(e) Umidade

Figura B.2: Funções de pertencimento dos antecedentes.

Listagem B.4: Código presente no arquivo de configuração definindo as regras a serem usadas pelo controlador *Fuzzy*.

```
def generate_rules():
    rules = []
    rules.append({"antecedents":{"temperature":"low",
    "smoke":"low", "uv":"low"}, "emergency":{"fire":"very
        low"}})
    rules.append({"antecedents":{"temperature":"low",
    "smoke":"low", "uv":"high"}, "emergency":{"fire":"low
       "}})
    rules.append({"antecedents":{"temperature":"average",
    "smoke":"medium", "uv":"low"}, "emergency":{"fire":
    "low"}})
    rules.append({"antecedents":{"temperature":"average",
    "smoke":"medium", "uv":"medium"}, "emergency":{"fire"
       :"medium"}})
    rules.append({"antecedents":{"temperature":"average",
    "smoke":"medium", "uv":"high"}, "emergency":{"fire":
    "medium"}})
    rules.append({"antecedents":{"temperature":"high", "
       smoke":"high", "uv":"low"}, "emergency":{"fire":
    "medium"}})
    rules.append({"antecedents":{"temperature":"high", "
       smoke":"high", "uv":"medium"}, "emergency":{"fire"
       :"high"}})
    rules.append({"antecedents":{"temperature":"high", "
       smoke":"high", "uv":"high"}, "emergency":{"fire":"
       high"}})
    rules.append({"antecedents":{"temperature":"very high
       ", "smoke":"high", "uv":"low"}, "emergency":{"fire
       ":"high"}})
    rules.append({"antecedents":{"temperature":"very high
       ", "smoke":"high", "uv":"medium"}, "emergency":{"
       fire":"very high"}})
    rules.append({"antecedents":{"temperature":"very high
       ", "smoke":"high", "uv":"high"}, "emergency":{"
       fire":"very high"}})
    rules.append({"antecedents":{"temperature":"high", "
       smoke":"medium", "uv":"low"}, "emergency":{"fire":
       "medium"}})
    rules.append({"antecedents":{"temperature":"high", "
       smoke":"medium", "uv":"medium"}, "emergency":{"
       fire":"high"}})
    rules.append({"antecedents":{"temperature":"high", "
```

```
        smoke":"medium", "uv":"high"}, "emergency":{"fire"
        :"high"}})
    rules.append({"antecedents":{"temperature":"average",
         "smoke":"low", "uv":"low"}, "emergency":{"fire":"
        very low"}})
    rules.append({"antecedents":{"temperature":"average",
         "smoke":"low", "uv":"medium"}, "emergency":{"fire
        ":"low"}})
    rules.append({"antecedents":{"temperature":"average",
         "smoke":"low", "uv":"high"}, "emergency":{"fire":
        "medium"}})
    rules.append({"antecedents":{"humidity":"low", "
        vibration":"low", "uv":"low"}, "emergency":{"flood
        ":"very low"}})
    rules.append({"antecedents":{"humidity":"average", "
        vibration":"medium"}, "emergency":{"flood":"medium
        "}})
    rules.append({"antecedents":{"humidity":"high", "
        vibration":"high"}, "emergency":{"flood":"high"}})
    rules.append({"antecedents":{"humidity":"very high",
        "vibration":"high"}, "emergency":{"flood":"very
        high"}})
    rules.append({"antecedents":{"humidity":"high", "
        vibration":"medium"}, "emergency":{"flood":"high"}
        })
    rules.append({"antecedents":{"humidity":"average", "
        vibration":"high"}, "emergency":{"flood":"medium"}
        })
    rules.append({"antecedents":{"humidity":"average", "
        vibration":"low"}, "emergency":{"flood":"very low"
        }})
    rules.append({"antecedents":{"humidity":"low", "
        vibration":"medium"}, "emergency":{"flood":"low"}}
        )

    return rules
```