Uma Arquitetura Modular para Monitoramento de Ambientes IoT

Eduardo Souza[1], Beatriz Azevedo Alves[2], Adriana V Ribeiro[2], Leobino N Sampaio[2]

[1]Escola Politécnica - Universidade Federal da Bahia (UFBA)
Salvador – BA – Brasil

[2]Instituto de Computação – Universidade Federal da Bahia (UFBA)
Salvador – BA – Brasil

{eduardo.souza,adrianavr,leobino}@ufba.br, beaazevedo14@gmail.com

Abstract. *The use of Internet of Things (IoT) devices for monitoring individuals and environments has become common in several fields such as industry, agriculture, and healthcare. This paper proposes a modular, generalist and multi-scenario architecture for monitoring environments with IoT devices. To validate the proposed architecture, we developed a proof of concept that demonstrated the feasibility of implementing and using the architecture in a real environment.*

Resumo. *O uso de dispositivos de Internet das Coisas (do inglês, Internet of Things - IoT) para monitoramento de indivíduos e ambientes tem se tornado comum em diversos setores como indústria, agricultura e saúde. Neste artigo, é proposta uma arquitetura modular, generalista e multi-cenário para monitoramento de ambientes com dispositivos IoT. Para validar o funcionamento da arquitetura proposta, foi desenvolvida uma prova de conceito que demonstrou a viabilidade da implantação e uso da arquitetura em um ambiente real.*

1. Introdução

A popularização da Internet das Coisas (do inglês, *Internet of Things - IoT*) tem sido caracterizada pela heterogeneidade de dispositivos e diversidade de aplicações, principalmente relacionadas ao monitoramento de ambientes e indivíduos. Microcontroladores como Arduíno, ESP3 e *single board computers* mais robustos como Raspberry Pi são utilizados para monitorar ambientes em setores como agricultura inteligente (aumento de colheitas e redução de custos), cuidados médicos (continuidade de tratamento de pacientes distantes, isolados, em observação temporária ou com enfermidades crônicas), proteção ambiental (enfrentamento de mudanças climáticas e preservação de recursos naturais), cidades inteligentes (coleta de lixo, economia de energia e gerenciamento de tráfico urbano de pedestres e veículos), entre outros [Witczak and Szymoniak 2024].

Apesar de diferentes áreas e cenários de aplicação apresentarem particularidades, os sistemas de monitoramento baseado em IoT apresentam cinco passos principais: (i) coletar dados do ambiente; (ii) armazenar os dados em nuvem ou em servidor local; (iii) pré-processar os dados; iv) analisar os dados; e (v) criar mecanismos de visualização dos dados. Essas etapas aparecem nos trabalhos de monitoramento de ambientes na literatura com diferentes estratégias de implementação e características de *design* [Dahan et al. 2023, Chanchí G et al. 2021, Guimarães Jr et al. 2020,

Paudel and Neupane 2021]. Os autores em [Chanchí G et al. 2021] propuseram uma arquitetura para monitoramento ambiental (temperatura, umidade e luminosidade) de plantas interiores. A arquitetura é organizada em quatro camadas: captura dos dados ambientais, armazenamento dos dados, análise dos dados por meio de algoritmos e gráficos e uma camada para visualização dos resultados monitorados. Um diferencial desse trabalho são as diferentes formas de visualizar a arquitetura: *business view*, *functional view* e *implementation view*, que são na prática diferentes formas de interpretá-la com base nas suas características de negócios, funcionalidades e ferramentas de implementação possíveis.

Já em [Guimarães Jr et al. 2020], é proposta uma arquitetura em quatro camadas para monitoramento e interoperabilidade de nós industriais: i) *edge layer*, para dispositivos de chão de fábrica; ii) *fog layer*, para *gateways* e supervisores; iii) *cloud layer*, para lógicas de negócio; iv) *services layer*, para dispositivos e nós industriais. A arquitetura tem o objetivo de aumentar o tempo de vida útil dos dispositivos e a disponibilidade de produção, além de reduzir custos. Os autores também afirmam que a alta coesão e baixo acomplamento da arquitetura contribuem para a interoperabilidade de nós industriais. Arquiteturas modulares de monitoramento têm se tornado foco de pesquisa em IoT pois viabilizam a implementação de sistemas de monitoramento e controle que se adaptam a novos requisitos e permitem a integração de novas tecnologias [Witczak and Szymoniak 2024].

Este artigo apresenta uma arquitetura genérica para o monitoramento de ambientes com IoT. A arquitetura proposta é composta por quatro módulos (*broker* MQTT, interconexão, armazenamento e visualização) e é caracterizada por ser generalista, ou seja, seus módulos podem ser implementados por meio de diversas ferramentas. O principal diferencial dessa arquitetura em relação às mencionadas anteriormente é seu caráter modular, de fácil implementação e adaptabilidade e com uso de ferramentas gratuitas. Para validar a usabilidade da arquitetura, foi realizada uma prova de conceito utilizando dispositivos reais e a implementação dos módulos da arquitetura em contêineres. As características da arquitetura e a implementação da prova de conceito estão descritas nas próximas seções.

2. Uma Arquitetura Modular para Monitoramento de Ambientes IoT

O monitoramento de ambientes IoT baseia-se em algumas etapas que incluem desde a coleta de dados ao seu armazenamento e visualização. Neste trabalho, foi proposta uma arquitetura modular para monitoramento de ambientes IoT a fim de possibilitar maior flexibilidade e adaptabilidade a diferentes cenários, visto que o uso da estrutura modular possibilita a substituição de ferramentas e implementações em um módulo sem afetar as demais partes da arquiteturaos demais. A Figura 1 ilustra a arquitetura composta por sensores e quatro módulos: i) um *broker*, responsável por receber os dados publicados pelos sensores; ii) um módulo de interconexão que é responsável por definir os fluxos de processamento dos dados coletados; iii) o módulo de armazenamento, no qual os dados coletados são armazenados em estruturas de dados; e iv) o módulo de visualização, que serve para exibir os dados dos sensores, indicando as condições do ambiente.

Os sensores se conectam com a arquitetura de monitoramento usando uma comunicação do tipo *publish/subscribe*, na qual um conjunto de nós publica dados em um tópico e os assinantes daquele tópico podem consumi-los à medida que os dados são produzidos. Na arquitetura proposta, essa comunicação é realizada com o *Message Queuing Telemetry Transport* (MQTT), um protocolo leve para comunicação *publish/subscribe*

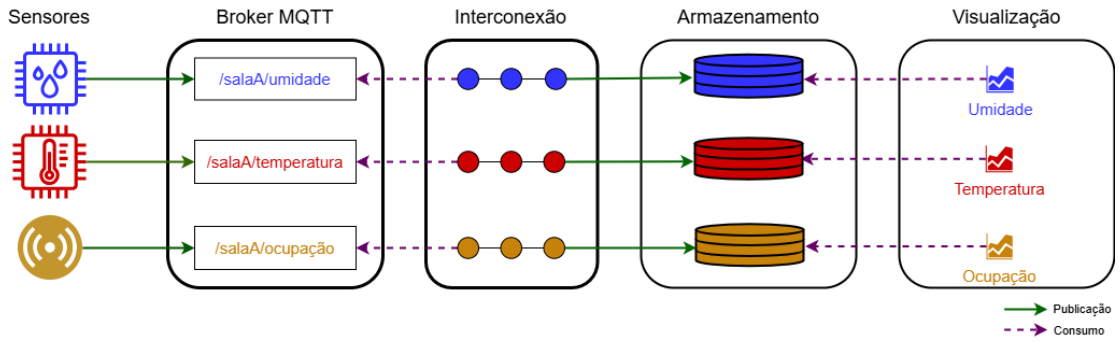


Figura 1. Arquitetura modular para monitoramento de ambientes IoT.

em IoT. O MQTT baseia-se na comunicação entre duas partes principais: clientes MQTT, publicadores ou assinantes de dados; e *broker* MQTT, que é responsável por gerenciar a comunicação entre os clientes, incluindo regras de fluxo de informação e serviços de autenticação/autorização de nós. Na Figura 1, observa-se a comunicação de clientes MQTT com o *broker* através da publicação e assinatura dos dados organizados em tópicos.

Os dados coletados devem ser armazenados para possibilitar a visualização das informações, criação de *baselines* e detecção de anomalias. Para que o módulo de armazenamento se comunique com o *broker* MQTT, é utilizado o módulo de interconexão. Este módulo é responsável por modelar os nós e os fluxos de processamento que serão aplicados aos dados para que eles fiquem em um formato adequado para armazenamento. Como as fontes de dados variam de acordo com as informações coletadas, formato, frequência de envio, volume, entre outras características, é importante que a estrutura de armazenamento seja adequada ao que se deseja observar no ambiente. O módulo de armazenamento é onde estão as estruturas necessárias para manter um histórico organizado dos dados de acordo com as informações enviadas. Nesses cenários, destaca-se o uso de bancos de dados de séries temporais, pois possibilitam a marcação de tempo nos dados e sua correlação histórica, bem como apresentam otimizações para armazenar e processar grandes volumes de dados. Os dados armazenados serão consultados pelo módulo de visualização e exibidos em *dashboards* que facilitem o monitoramento do ambiente. Esse módulo também é responsável pela geração de alertas e notificações aos administradores do sistema.

2.1. Prova de Conceito

Para validar o funcionamento da arquitetura, os módulos foram construídos em contêineres Docker, que facilitam a portabilidade e manutenibilidade. Os contêineres foram executados em um notebook Dell Inspiron 14-N4050 com processador Intel Core i3-2330M CPU @ 2.20GHz, memória RAM de 16 GB do tipo DDR3 @ 660MHz em *Dual-Channel* e memória de armazenamento SSD SATA-2 com 512GB. Além disso, foram utilizados Raspberry Pi 4 Model B com 4 GB de memória RAM e 64 GB de armazenamento para emular os sensores e alguns equipamentos de comunicação de rede. Adicionalmente, todos os dados e scripts utilizados na prova de conceito estão acessíveis no GitHub[1].

[1] `https://github.com/EduSanSou/iniciacao_cientifica_2024-2025`

2.1.1. Topologia de Rede

Na Figura 2, é possível observar os compomentes que foram utilizados na prova de conceito da arquitetura. Dois dispositivos Raspberry Pi representam os sensores e são responsáveis por publicar os dados do ambiente. Enquanto um terceiro Raspberry Pi atua como um ponto de acesso para a rede de sensores. Paralelamente, há um servidor que hospeda os contêineres que implementam os módulos da arquitetura. Tanto o servidor quanto o ponto de acesso estão conectados em um roteador, que funciona como *gateway* para as duas redes. O roteador utilizado, trata-se de um roteador doméstico Sagemcom de modelo F@st 5657 TIM LIVE com suporte a redes sem fio nos padrões IEEE 802.11 b/g/n (2.4 GHz) e IEEE 802.11 a/n/ac (5 GHz) com segurança WPA2/WPA e quatro portas Ethernet.

Os dispositivos foram configurados com endereçamento estático. O servidor, o roteador e a interface física (eth) do ponto de acesso pertencem à rede 192.168.1.0/24. Enquanto os sensores e a interface de rede sem fio do ponto de acesso têm IPs configurados na rede 10.42.0.0/24. As rotas necessárias para comunicação entre as redes foram adicionadas no ponto de acesso, garantindo a comunicação entre os dispositivos.

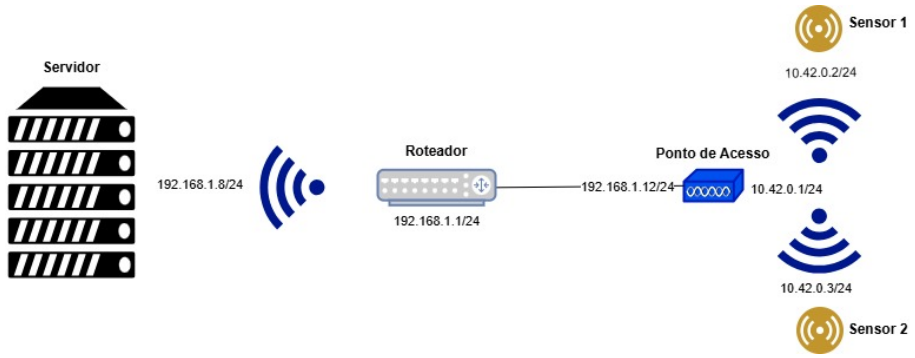


Figura 2. Topologia de rede utilizada na prova de conceito da arquitetura.

2.1.2. Caracterização de datasets e geração de traces

Para execução da prova de conceito, foram criados *traces* com base em um *dataset* de sensores IoT para emular a coleta de dados ambientais e a publicação dos dados no *broker* MQTT. *Datasets* são conjuntos de dados utilizados para testes de aplicações e serviços IoT. No entanto, grande parte dos *dataset* é voltada à aplicabilidade em *Machine Learning* e muitas vezes não apresentam todas as informações necessárias para gerar *traces* [Shiravi et al. 2012]. Assim, os *traces* utilizados na prova de conceito foram criados a partir do *dataset* público da Intel Berkeley Research Lab[2]. Este *dataset* foi construído com base em 54 sensores implantados no laboratório de pesquisa da Intel, em Berkeley, que coletaram dados ambientais, incluindo umidade, temperatura, luz e voltagem. Este *dataset* foi escolhido para a geração de *traces* por conter dados numéricos e *timestamps* detalhados (e.g. data, hora e época), possibilitando a geração de tráfego de rede realista em termos de volume, tamanho de pacotes e frequência de envio dos dados.

[2]`https://db.csail.mit.edu/labdata/labdata.html`

O *dataset* foi caracterizado com a utilização da ferramenta Jupyter Notebook[3]. A caracterização consistiu na identificação das informações do *dataset*, como quantidade de dados, média dos valores, intervalo entre registros, categorias, identificação de *outliers* e distribuição dos dados. Alguns dados dessa caracterização são listados a seguir:

- Total de registros: $2.313.682$
- 95.5% dos registros estavam completos, 4.1% estavam incompletos e 0.4% eram registros inconsistentes
- A proporção de *outliers* foi de 18.65% para dados de temperatura, 13.71% para umidade, 10.27% para a luminosidade e 0.27% para a voltagem
- Distribuição dos dados: identificação de valores médios, mediana, desvio padrão, valores máximos/mínimos

Após a caracterização do *dataset*, foi realizada a remoção de registros inconsistentes e incompletos. Como resultado desta etapa, obteve-se um novo *dataset* com um total de $2.210.083$ registros, que compreendia dados de 54 sensores. Os dados de cada sensor foram separados individualmente para geração de *traces* e apresentavam colunas como tempo, umidade, luminosidade, temperatura e voltagem.

2.1.3. Coleta e publicação de dados dos sensores

O fluxo de coleta e publicação dos dados dos sensores se deu da seguinte forma: os sensores enviam dados de temperatura, umidade, voltagem e luminosidade do *dataset* escolhido considerando a frequência definida pelo tempo de envio nos *traces*[4]. Essa comunicação entre dispositivos (publicador de dados) e o *broker* MQTT foi realizada por meio de *scripts* Python que lêem os *traces* e utilizam a biblioteca *Paho* (cliente MQTT) para publicar dados em diferentes tópicos definidos no *broker*. O *broker* MQTT foi implementado utilizando a ferramenta Mosquitto[5] em sua versão $2.0.18$, com MQTT na versão $3.1.1$. O Mosquitto foi escolhido pois é uma ferramenta de fácil uso que possui uma ampla documentação e tem sido utilizada na literatura. Além disso, atende aos propósitos de *design* da arquiterura, por tratar-se de uma ferramenta *open-source*, gratuita e leve.

Uma vez publicados no *broker* MQTT, os dados podem ser consumidos pelo módulo de armazenamento. Essa comunicação entre o *broker* MQTT e o módulo de armazenamento é realizada através do módulo de interconexão. O módulo de interconexão foi implementado utilizando a ferramenta Node-RED[6] na versão $4.0.2$. O Node-RED é uma ferramenta *low-code* e de programação visual que possui a funcionalidade de integrar dispositivos, interfaces, bancos de dados, serviços *on-line* e outros componentes de *software* e *hardware* de forma prática e segura. Ele tem sido bastante utilizado em projetos de engenharia e IoT. Nesta prova de conceito, o Node-RED foi utilizado para modelar os clientes MQTT via interface gráfica, bem como suas assinaturas aos tópicos do *broker*. Além disso, a interconexão desses clientes com o banco de dados definido no módulo de armazenamento também foi realizada com o apoio dessa ferramenta.

[3] `https://jupyter.org/`

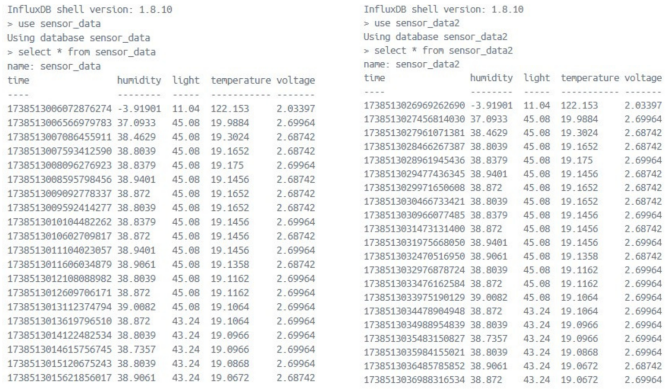
[4] De acordo com os dados coletados no *dataset*, os envios ocorriam aproximadamente a cada 30s. No entanto, também foram executados alguns experimentos utilizando o intervalo padrão de 0.5s entre os envios com o objetivo de facilitar a visualização gráfica das informações durante as simulações.

[5] `https://mosquitto.org/`

[6] `https://nodered.org/`

2.1.4. Armazenamento dos dados

Em sistemas de monitoramento IoT, o armazenamento de dados é essencial para correlação de eventos, visualização de dados históricos e detecção de anomalias. Assim, o módulo de armazenamento é o responsável por definir como e onde os dados serão armazenados. Para esse propósito, utilizou-se o InfluxDB[7], banco de dados voltado ao armazenamento e análise de séries temporais, ideal para monitoramento de sensores e dados em tempo real. Para a prova de conceito, as variáveis coletadas pelos sensores foram armazenadas em dois bancos de dados em formato de séries temporais, conforme observado na Figura 3.



```
InfluxDB shell version: 1.8.10
> use sensor_data
Using database sensor_data
> select * from sensor_data
name: sensor_data
time                humidity  light  temperature voltage
----                --------  -----  ----------- -------
1738513006072876274 -3.91901  11.04  122.153     2.03397
1738513006566979783 37.0933   45.08  19.9884     2.69964
1738513007086455911 38.4629   45.08  19.3024     2.68742
1738513007593412590 38.8039   45.08  19.1652     2.68742
1738513008096276923 38.8379   45.08  19.175      2.69964
1738513008595798456 38.9401   45.08  19.1456     2.68742
1738513009092778337 38.872    45.08  19.1652     2.68742
1738513009592414277 38.8039   45.08  19.1652     2.68742
1738513010104482262 38.8379   45.08  19.1456     2.69964
1738513010602709817 38.872    45.08  19.1456     2.68742
1738513011104023057 38.9401   45.08  19.1456     2.69964
1738513011606034879 38.9061   45.08  19.1358     2.68742
1738513012108088982 38.8039   45.08  19.1162     2.69964
1738513012609706171 38.872    45.08  19.1162     2.69964
1738513013112374794 39.0082   45.08  19.1064     2.69964
1738513013619796510 38.872    43.24  19.1064     2.69964
1738513014122482534 38.8039   43.24  19.0966     2.69964
1738513014615756745 38.7357   43.24  19.0966     2.69964
1738513015120675243 38.8039   43.24  19.0868     2.69964
1738513015621856017 38.9061   43.24  19.0672     2.68742
```

```
InfluxDB shell version: 1.8.10
> use sensor_data2
Using database sensor_data2
> select * from sensor_data2
name: sensor_data2
time                humidity  light  temperature voltage
----                --------  -----  ----------- -------
1738513026969262690 -3.91901  11.04  122.153     2.03397
1738513027456814030 37.0933   45.08  19.9884     2.69964
1738513027961071381 38.4629   45.08  19.3024     2.68742
1738513028466267387 38.8039   45.08  19.1652     2.68742
1738513028961945436 38.8379   45.08  19.175      2.69964
1738513029477436345 38.9401   45.08  19.1456     2.68742
1738513029971650608 38.872    45.08  19.1652     2.68742
1738513030466733421 38.8039   45.08  19.1652     2.68742
1738513030966077485 38.8379   45.08  19.1456     2.69964
1738513031473131400 38.872    45.08  19.1456     2.68742
1738513031975668050 38.9401   45.08  19.1456     2.69964
1738513032470516950 38.9061   45.08  19.1358     2.68742
1738513032976878724 38.8039   45.08  19.1162     2.69964
1738513033476162584 38.872    45.08  19.1162     2.69964
1738513033975190129 39.0082   45.08  19.1064     2.69964
1738513034478904948 38.872    43.24  19.1064     2.69964
1738513034988954839 38.8039   43.24  19.0966     2.69964
1738513035483150827 38.7357   43.24  19.0966     2.69964
1738513035984155021 38.8039   43.24  19.0868     2.69964
1738513036485785852 38.9061   43.24  19.0672     2.68742
1738513036988316534 38.872    43.24  19.0672     2.69964
```

Figura 3. Exemplo de armazenamento de informações de monitoramento em um banco de dados de séries temporais.

2.1.5. Plataforma de visualização

O módulo de visualização é o responsável pela confecção de gráficos e alertas que possibilitam o monitoramento do ambiente. Apesar do InfluxDB possuir uma funcionalidade de monitoramento e visualização de dados, a ferramenta Grafana[8] foi utilizada. Esta possui visualização mais robusta que permite criar *dashboards* mais completos e elaborados. É facilmente integrada a diversos bancos de dados, customizável e permite configurar alertas e notificações.

Na prova de conceito desenvolvida, foi criado um *dashboard* por dispositivo. No entanto, esses *dashboards* podem ser definidos de acordo com a semântica ou o contexto da aplicação. Por exemplo: pode-se definir *dashboards* de acordo com a localização física, possibilitando a verificação de dados de inúmeros sensores localizados em uma mesma sala. Independente de como o *dashboard* é definido, é preciso que ele tenha acesso ao banco de dados para que realize requisições e colete as informações necessárias para plotagem gráfica. Comumente, esse processo é realizado através de requisições $HTTP$ utilizando o método GET. A Figura 4 apresenta alguns exemplos dos gráficos plotados a partir do monitoramento de um sensor. No *dashboard* é possível observar os gráficos de temperatura, umidade, luminosidade e voltagem. Esses gráficos são atualizados em

[7] https://www.influxdata.com/
[8] https://grafana.com/

tempo real, à medida que os dados são publicados pelos dispositivos e armazenados no InfluxDB. A visualização gráfica dessas informações em tempo real permite correlacionar eventos e observar o comportamento das variáveis ao longo do tempo. Por exemplo: é possível visualizar um aumento de temperatura, de 18°C para 20°C, enquanto a umidade inicia com um valor mais alto (39) e decresce gradualmente até 36.

Figura 4. Exemplo de dashboard do Grafana para ilustrar os dados de temperatura, umidade, luminosidade e voltagem coletados por um sensor.

Além da visualização dos dados em tempo real, é possível configurar a emissão de alertas e notificações no Grafana, essenciais em um sistema de monitoramento de ambiente. Na prova de conceito, foram definidos alguns limiares por variável e, caso a média dos dados coletados em um período de tempo predefinido ultrapassasse esse limiar, um alerta seria enviado por e-mail. A Figura 5 ilustra o uso dessa funcionalidade através de uma advertência sobre o valor de temperatura medido por um dos sensores.

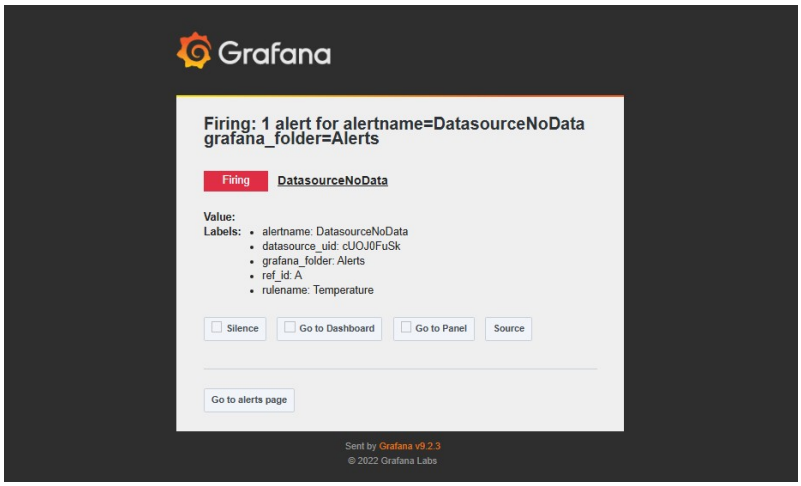


Figura 5. Exemplo de alerta de temperatura do Grafana.

3. Conclusão

O trabalho apresentou uma arquitetura modular para monitoramento de ambientes com dispositivos IoT, caracterizada por ser generalista, escalável e aplicada a vários cenários,

possibilitando seu uso em sistemas de monitoramento de saúde, ambiente, indústria, entre outros. A arquitetura é composta por módulos que se comunicam para possibilitar a coleta, o armazenamento e a visualização dos dados. Com a prova de conceito desenvolvida, foi possível observar a viabilidade da implementação da arquitetura com dispositivos reais. Durante a implementação da arquitetura, os principais desafios enfrentados estiveram relacionados à curva de aprendizado dos fundamentos teóricos e das implementações práticas da PoC, incluindo a escolha das ferramentas adequadas para cada módulo. Além disso, assegurar uma conexão estável entre os Raspberry Pis e à rede ligada a eles foi um processo desafiador. Essas condições exigiram flexibilidade e adaptação constantes para garantir a estabilidade do sistema durante os testes e simulações.

Como perspectivas futuras para a evolução da arquitetura, pretende-se utilizar sensores reais para realizar a captação dos dados e a experimentação próximo de um ambiente real. Ademais, utilizar diferentes estruturas de armazenamento para outros tipos de dados (e.g. dados de câmeras) e avaliar como as ferramentas lidam com dados faltantes ou incompletos. Adicionalmente, pretende-se avaliar o comportamento da arquitetura em cenários com conectividade intermitente ou perda de pacotes. Também espera-se aplicar algumas abordagens de segurança ao sistema para garantir que apenas indivíduos e dispositivos autorizados tenham acesso aos módulos da arquitetura de forma granular. Além disso, pretende-se realizar estudos relacionados à segurança dos dados no monitoramento de ambientes, principalmente em redes sem fio.

Agradecimentos

O projeto é resultado do Programa Institucional de Bolsas de Iniciação Científica (PIBIC) da UFBA com fomento da Fundação de Amparo à Pesquisa do Estado da Bahia (FAPESB) e do Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq).

Referências

Chanchí G, G. E., Ospina A, M. A., and Campo M, W. Y. (2021). Iot architecture for monitoring variables of interest in indoor plants. *Computación y Sistemas*, 25(4):695–705.

Dahan, F., Alroobaea, R., Alghamdi, W. Y., Mohammed, M. K., Hajjej, F., and Raahemifar, K. (2023). A smart iomt based architecture for e-healthcare patient monitoring system using artificial intelligence algorithms. *Frontiers in Physiology*, 14:1125952.

Guimarães Jr, C. S. S., de Andrade, M., De Avila, F. R., Gomes, V. E. D. O., and Nardelli, V. C. (2020). Iot architecture for interoperability and monitoring of industrial nodes. *Procedia Manufacturing*, 52:313–318.

Paudel, N. and Neupane, R. C. (2021). A general architecture for a real-time monitoring system based on the internet of things. *Internet of Things*, 14:100367.

Shiravi, A., Shiravi, H., Tavallaee, M., and Ghorbani, A. A. (2012). Toward developing a systematic approach to generate benchmark datasets for intrusion detection. *computers & security*, 31(3):357–374.

Witczak, D. and Szymoniak, S. (2024). Review of monitoring and control systems based on internet of things. *Applied Sciences*, 14(19).