RDEV - UMA SOLUÇÃO BASEADA EM IoT PARA COLETA, MONITORAMENTO E VISUALIZAÇÃO DE TEMPERATURA E UMIDADE DE AMBIENTES EM TEMPO REAL

NOME DO AUTOR
ARTUR STEFAN MOURA DE FREITAS
NOME DO ORIENTADOR
PROF. DR. EUGÊNIO SPER DE ALMEIDA

Resumo
Com a democratização da tecnologia, a quantidade de dispositivos sensíveis às variações das condições de um ambiente aumentou drasticamente. Para reduzir os possíveis danos causados por oscilações de umidade relativa do ar e temperatura se faz necessário o monitoramento em tempo real das condições do ambiente. Pensando nessa necessidade de monitoramento foi desenvolvido o RDEV que por meio de um sensor de temperatura e umidade e um microcontrolador acoplados faz a leitura desses dados ambientais e envia para um *broker* via protocolo MQTT. Em seguida um agente coletor busca os dados no *broker* e os armazena no banco de dados. Uma ferramenta, que permite a criação de *Dashboard,* realiza a leitura e visualização desses dados. O RDEV se mostrou uma boa solução monitoramento de ambientes tendo como vantagem a fácil expansão e o baixo custo de implementação.

Palavras-chave: IoT. Sensores. Monitoramento em Tempo Real. Dados Ambientais. Representações Gráficas.

Abstract
With the democratization of technology, the number of devices sensitive to changes in the environmental conditions has increased dramatically. In order to reduce possible damage caused by fluctuations in relative humidity and temperature, real-time monitoring of environmental conditions is necessary. Thinking about this need for monitoring we have developed the RDEV, which uses a temperature and humidity sensor and a connected microcontroller to read these environmental data and send them to a broker via the MQTT protocol. Then a collecting agent fetches the data from the broker and stores it in the database. A tool, which allows the creation of a Dashboard, performs the reading and visualization of this data. The RDEV proved to be a good solution for monitoring environments with the advantage of easy expansion and low cost of implementation.

Keywords: *IoT. Sensors. Real Time Monitoring. Environmental Data. Graphic Representations.*

Introdução

Aparelhos eletrônicos são muito sensíveis às variações de temperatura e umidade do ambiente onde estão instalados, mesmo sendo comum a existência de aparelhos de ar condicionado nesses locais, não podemos descartar a possibilidade de um mau funcionamento do mesmo, o que poderia resultar em danos, mau funcionamento e até mesmo na diminuição do tempo de vida dos equipamentos instalados.

Para evitar que problemas como esses ocorram, é necessário o controle contínuo das condições do ambiente, sendo que, este controle pode ser melhorado com o monitoramento

contínuo das condições do ambiente, visando a tomada de medidas preventivas ou reativas em tempo hábil.

Para que seja possível esse monitoramento foi desenvolvido uma solução utilizando tecnologias IoT (*Internet of Things*) que Madakam et al (2015) descreve como uma rede capaz de se organizar e compartilhar dados, recursos e serviços conforme as situações ou mudanças em seu ambiente. Essa tecnologia será utilizada para coleta dos dados. Para o armazenamento será utilizado um banco de dados NoSQL que segundo Kang et al (2016) possui características adequadas para IoT, pois possuem bases de dados distribuídos e processamento paralelo nativo; e para a visualização dos mesmos será adotada a ferramenta Grafana, que é um suíte de análise utilizado para a visualização de dados e métricas em tempo real.

O desenvolvimento dessa ferramenta permite o monitoramento das condições ambientais em tempo real, contribuindo para a tomada de decisões de forma rápida, efetiva e assertiva, reduzindo possíveis riscos, prejuízos e problemas advindos das grandes variações de temperatura.

1. Fundamentação Teórica

Esta seção resume os aspectos das bases teóricas que envolvem o RDEV, a saber: Tecnologias IoT, Sensores e Tecnologias de Armazenamento de Dados (Banco de Dados).

1.1. Internet of Things (IoT)

De uma forma mais abrangente, "A internet das Coisas representa uma visão na qual a Internet se estende ao mundo real abraçando objetos do cotidiano" (MATTERN; FLOERKEMEIR, 2010, p.1, tradução nossa).

Dessa forma, pode-se considerar que qualquer objeto que possa ser identificado e conectado à internet é um dispositivo IoT, sendo que esse conceito pode ser aplicado e utilizado em várias áreas distintas como indústrias, escolas e hospitais. Porém, Perumal, Sulaiman e Leong (2016) destacam que "Uma das grandes contribuições da Iot está no campo do monitoramento ambiental, especialmente no gerenciamento de desastres, sistemas de alerta precoce e análise de dados ambientais".

Nesse sentido, pode-se dizer que tecnologia IoT é composta por diversas tecnologias como redes de sensores sem fio (WSNs), RFID, código de barras, NFC, computação em nuvem, protocolos de internet (IP), etc (LI et al, 2015). Deve-se considerar, também, que existem "seis elementos principais necessários para fornecer as funcionalidades da IoT"

(Al-FUQAHA et al., 2015 p.5, tradução nossa): a identificação, detecção, comunicação, computação (processamento), serviços e semântica.

1.1.1. Identificação

Para funcionar de forma correta, uma rede de dispositivos IoT deve permitir que cada sensor/atuador seja identificado de uma forma que não seja possível confundir os dispositivos, pois "A identificação é crucial para IoT nomear e corresponder os serviços com a sua demanda" (AL-FUQAHA et al., 2015 p.5, tradução nossa), pois sem a correta identificação dos dispositivos seria inviável o uso destes dispositivos.

1.1.2. Detecção

Existem diversos tipos de sensores IoT, que vão desde sensores inteligentes a atuadores, Al-Fuqaha et al. (2015) explica que um sensor Iot deve ser capaz de realizar coletas de dados do ambiente ou objeto que está relacionado com a sua rede e enviar essas informações para que sejam processadas e armazenadas.

1.1.3. Comunicação

Comunicação é um requisito essencial para o funcionamento de tecnologias IoT, sabendo que "As tecnologias de comunicação IoT conectam objetos heterogêneos para fornecer informações inteligentes e serviços." (AL-FUQAHA et al., 2015 p.5, tradução nossa), para que isso ocorra de forma ordenada e consistente é necessário o uso de protocolos de comunicação.

Dos diversos protocolos utilizados para criar a conexão entre essas "coisas", Čolaković et al. (2018) destaca que os protocolos mais utilizados para isso são o CoAP (Protocolo de Aplicação Restrita), MQTT (MQTT-SN – MQTT para redes de sensores), AMQP (Protocolo de Enfileiramento de Mensagens Avançado), XMPP (Protocolo de Presença de Mensagem Extensível) e DDS (Serviço de Distribuição de Dados), sendo que a conexão com a internet é um requisito indispensável para os dispositivos conseguirem trabalhar entre si com serviços rodando no *back end*.

1.1.3.1. Message Queue Telemetry Transport (MQTT)

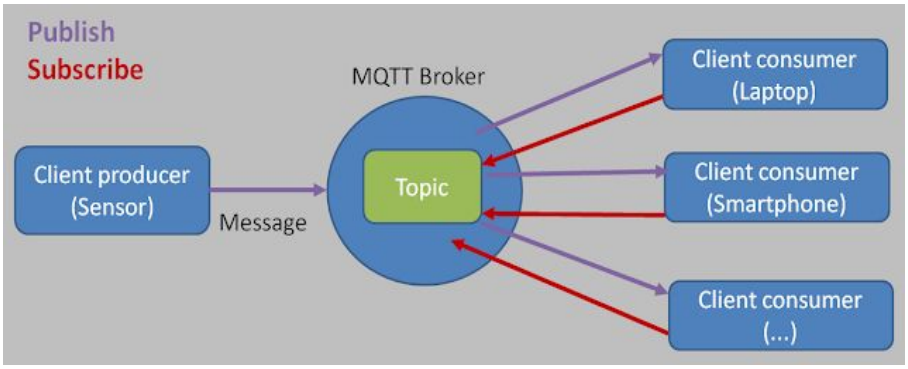
O MQTT tornou-se padrão para a comunicação IoT por ter sido projetado para conexões remotas com pouca largura de banda disponível.

Podemos descrevê-lo como um protocolo de mensagens com suporte a comunicação assíncrona, que faz uso do protocolo TCP/IP (padrão da internet) e utiliza o "padrão de

publish/subscribe para fornecer flexibilidade de transição e simplicidade de implementação" (AL-FUQAHA et al., 2015 p.9, tradução nossa).

Al-Fuqaha et al. (2015) explica que o MQTT é composto basicamente por três elementos: um *publisher* que transmite informações para um tópico do broker*, o subscriber* que se inscreve em um tópico de seu interesse e o *broker* que é o servidor responsável pelo roteamento das mensagens trocadas entres os *publishers* e os *subscribers.* A Figura 1 ilustra a arquitetura MQTT.

Figura 1: Arquitetura MQTT



Fonte: Fernando K (2019)

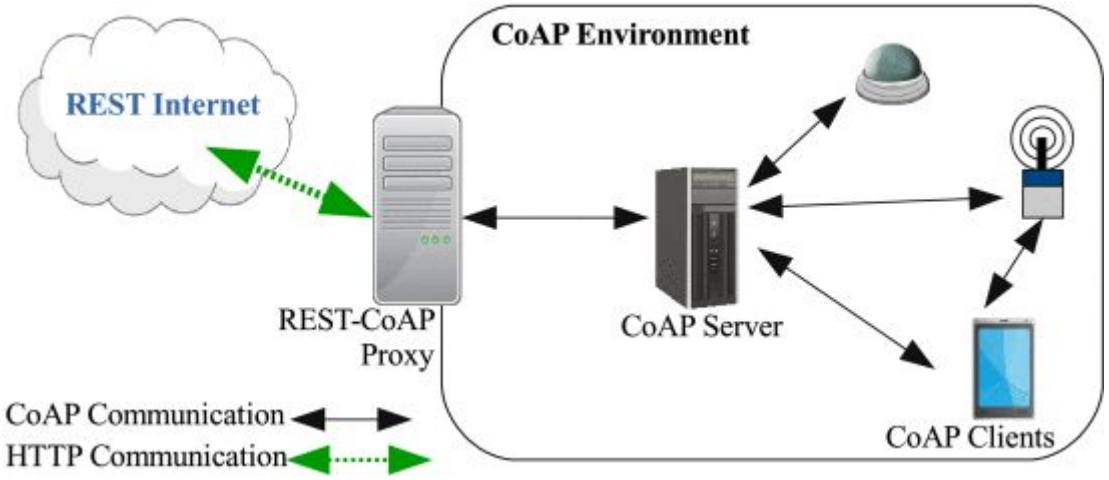
1.1.3.2. Constrained Application Protocol (CoAP)

O protocolo de comunicação CoAp é um modelo que "permite que a interação *Machine-to-Machine* seja de uma forma semelhante ao modelo cliente/servidor, porém onde ambas as máquinas fazem as duas funções"(CASSINELLI, MIRANDA, VALE, 2018, p.2), em outras palavras, ele é um protocolo web utilizado para redes de Internet das Coisas e comunicação *Machine-to-Machine* (M2M) visando "habilitar pequenos dispositivos com baixo consumo de energia e recursos computacionais" (AL-FUQAHA et al. 2015, p.8, tradução nossa), pois ele foi projetado para usar o mínimo de recursos de rede e dos dispositivos.

Como na utilização desse protocolo a mesma máquina pode trabalhar como cliente e servidor e as mensagens chegam de forma assíncrona, utiliza-se "para definir se a mensagem enviada é uma solicitação ou uma resposta o protocolo insere alguns códigos de método e de respostas nas próprias mensagens" (CASSINELLI, MIRANDA, VALE, 2018, p.2).

Mesmo o CoAP sendo utilizado na camada de aplicação, Cassinelli, Miranda, Vale (2018) consideram que podemos abstrair esse protocolo em duas camadas, onde a UDP e as mensagens são tratadas na camada inferior e as solicitações e respostas tratadas em nível de aplicação. A Figura 2 ilustra a arquitetura CoAP.

Figura 2: Arquitetura CoAP



Fonte: AL-FUQAHA et al. (2015)

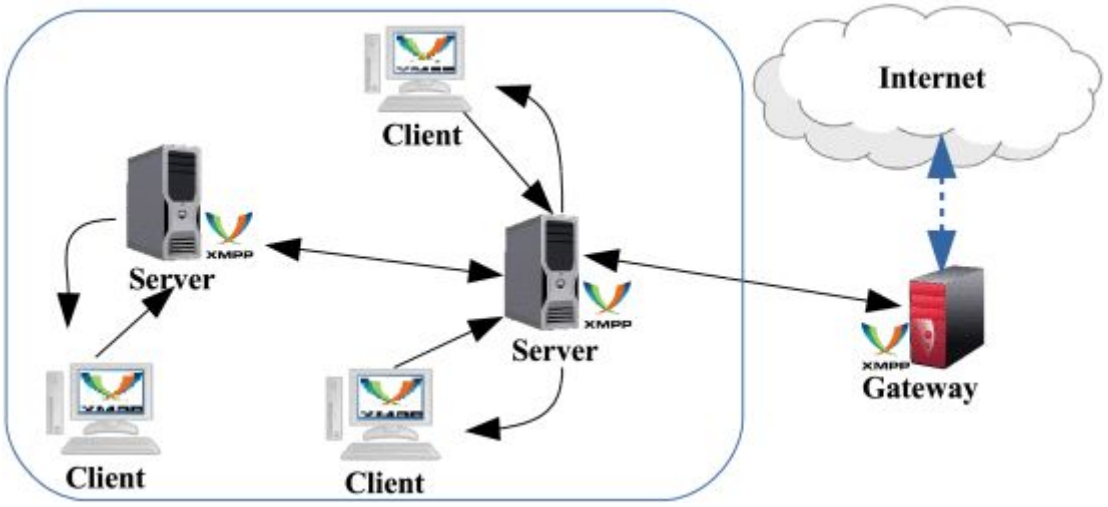
1.1.3.3. Extensible Messaging and Presence Protocol (XMPP)

O protocolo XMPP por usar o padrão IETF que é largamente utilizado para salas de bate papo, chamadas de voz e vídeo, oferece suporte para comunicação segura, descentralizada e livre de spam.

Como esse protocolo é bem versátil ele “permite que os usuários se comuniquem enviando mensagens instantâneas pela internet, independente do sistema operacional que esteja usando” (AL-FUQAHA et al. 2015, p.10, tradução nossa), além disso ele possibilita o total controle de acesso, privacidade, criptografia de ponta a ponta e é compatível com diversos outros protocolos de comunicação.

Como o protocolo fornece um ambiente “seguro e permite a adição de novos aplicativos nos protocolos principais” (AL-FUQAHA et al. 2015, p.10, tradução nossa) ele tem se tornando um protocolo relevante dentro do ambiente IoT. A Figura 3 ilustra a arquitetura XMPP.

Figura 3: Arquitetura XMPP



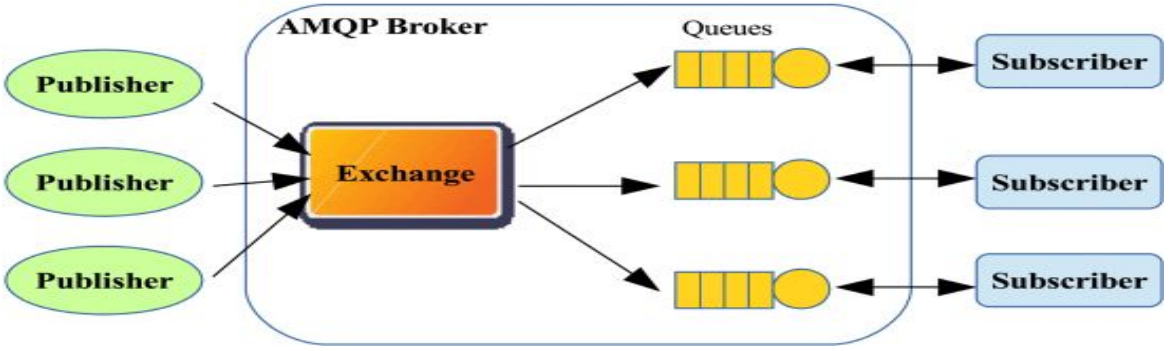
Fonte: AL-FUQAHA et al. (2015)

1.1.3.4. Advanced Message Queuing Protocol (AMQP)

O AMQP é um protocolo de comunicação orientado a mensagens que utiliza o protocolo TCP para uma comunicação confiável. Ele "define uma camada de mensagens sobre sua camada de transporte. Os recursos de mensagem são tratados nesta camada." (AL-FUQAHA et al. 2015, p.10, tradução nossa), garantindo que as mensagens são entregues "no máximo uma vez, pelo menos uma vez e exatamente uma vez" (AL-FUQAHA et al. 2015, p.10, tradução nossa).

Conforme ocorre a troca de mensagens, a *Exchange* faz o roteamento delas para suas *Queues* apropriadas e depois elas são enviadas aos seus respectivos destinatários. Além desse padrão de comunicação, o AMQP suporta o modelo *publish*/*subscribe*. A Figura 4 ilustra a arquitetura AMQP.

Figura 4: Arquitetura AMQP



Fonte: AL-FUQAHA et al. (2015 **)**

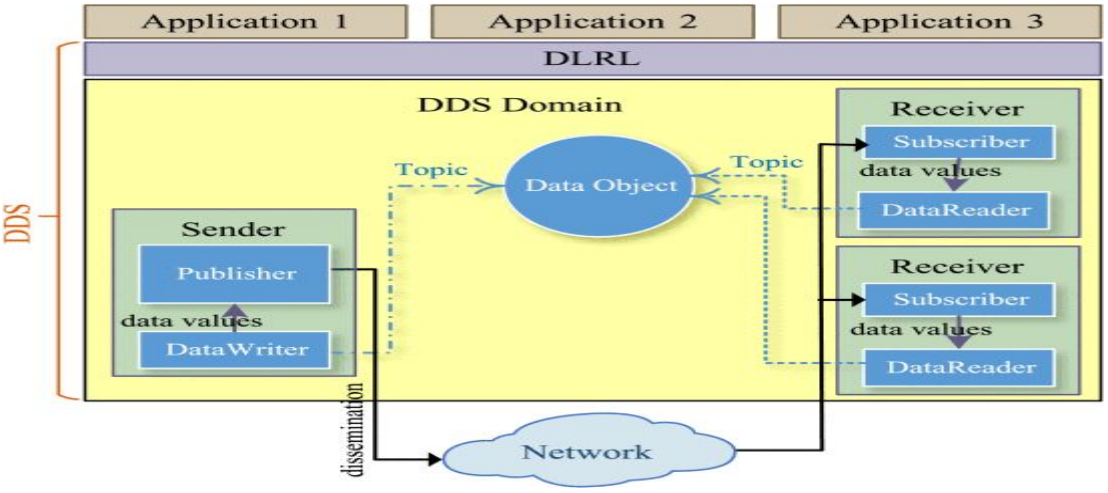
1.1.3.5. Data Distribution Service (DDS)

DDS é um "protocolo de *publish*/*subscribe* para comunicações M2M em tempo real" (AL-FUQAHA et al. 2015, p.11, tradução nossa), esse protocolo ao contrário do padrão do MQTT e AMQP não utiliza um intermediário, o DDS usa um *multicasting* para oferecer seus serviços de forma eficiente pois seu formato de arquitetura se adapta bem às restrições de comunicação em tempo real tanto do IoT quanto do M2M, garantindo uma alta confiabilidade para suas aplicações.

Al-Fuqaha et al (2015) separa a arquitetura DDS em duas camadas, a *Date*-Centric *Publish-Subscribe* (DCPS) e a *Data-Local Reconstruction Layer* (DLRL), sendo, a camada DCPS responsável pelo fornecimento das informações para os assinantes e a DLRL uma camada opcional que atua como uma interface para funcionalidades DCPS.

Dentro do fluxo de dados na camada DCPS, Al-Fuqaha et al. (2015) elenca cinco entidades envolvidas nesse fluxo, um *publisher* que é responsável por publicar os dados, um *DataWriter* que é o aplicativo que utiliza a interação com o editor para definir os valores e alterações nos dados, essa associação indica que os dados serão publicados em um determinado contexto, também temos um *subscriber* que irá receber os dados publicados e entregar eles para o aplicativo, temos também um *DataReader* que é utilizado pelo *subscriber* para acessar os dados recebidos e por último temos um tópico que é identificado por um tipo de dado e um nome. A Figura 5 ilustra a arquitetura DDS.

Figura 5: Arquitetura DDS



Fonte: AL-FUQAHA et al. (2015)

1.1.4. Computação (Processamento)

Um fator importante para os dispositivos IoT é a capacidade de processar as informações, podemos dizer que "Unidades de processamento (por exemplo

microcontroladores, microprocessadores, SOCs, FPGAs) e aplicativos de software representam o cérebro e a capacidade computacional da IoT." (AL-FUQAHA et al. 2015, p.6, tradução nossa). As plataformas de computação em nuvem também são relevantes para essa rede de dispositivos pois permite que as informações processadas estejam disponíveis a partir de qualquer lugar.

1.1.5. Serviços

Os "serviços relacionados à identidade, serviços de agregação de informações, serviços colaborativos, e serviços ubíquos" (AL-FUQAHA et al. 2015, p.6, tradução nossa) são os principais tipos de serviços que uma rede IoT pode prover.

Um serviço de identificação é a capacidade de uma aplicação "transportar" objetos do mundo real para o mundo virtual e dentro desse novo contexto identificar o objeto. Esse tipo de serviço é considerado básico, porém, como é utilizado por outros ele é de suma importância.

O serviço de agregação é responsável pela coleta, processamento dos dados brutos que são coletados pelos sensores; o serviço colaborativo trabalha em cima de serviços de agregação e utilizando os dados recebidos ele é capaz de reagir e tomar decisões em conformidade com a situação.

Um serviço ubíquo ou onipresente, tem o objetivo de "fornecer serviços de reconhecimento e de colaboração a qualquer momento para quem precisar deles em qualquer lugar" (AL-FUQAHA et al. 2015, p.6, tradução nossa) facilitando assim o acesso a qualquer serviço disponível.

1.1.6. Semântica

Entende-se que a "Semântica na IoT refere-se à capacidade de extrair conhecimento por diferentes máquinas para fornecer aos serviços necessários." (AL-FUQAHA et al. 2015, p.7, tradução nossa). Entende-se que esse processo de extração inclui a capacidade de descobrir, usar recursos ou informações e também fazer o reconhecimento e a análise dos dados buscando assim o fornecimento de serviços adequados a cada situação. Baseado nisso podemos dizer que a semântica equivale ao cérebro da IoT, enviando comandos para que sejam utilizados serviços e recursos (Al-Fuqaha et al., 2015).

1.2. Sensores

Sensor é um "Termo empregado para designar dispositivos sensíveis a alguma forma de energia do ambiente que pode ser luminosa, térmica, cinética, relacionando informações

sobre uma grandeza que precisa ser medida" (THOMAZINI, ALBUQUERQUE, 2011, p.17). Existem diversos tipos de sensores que respondem a diferentes estímulos como "temperatura, pressão, velocidade, corrente, aceleração, posição, etc" (THOMAZINI, ALBUQUERQUE, 2011, p.17).

Thomazini e Albuquerque (2011) destacam que os sensores possuem várias características que são importantes para definir qual é o mais indicado para cada situação. Essas características são as saídas de dados que podem ser analógicas onde o valor do seu sinal de saída pode assumir qualquer valor que esteja dentro da sua faixa de operação e saídas digitais onde o sinal pode assumir apenas dois valores no seu sinal de saída, que podem ser interpretados como zero ou um; sensibilidade que "é a razão entre o sinal de saída e de entrada para um dado sensor" (THOMAZINI, ALBUQUERQUE, 2011, p.22).

Exatidão é "a aptidão de um instrumento de medição para dar respostas próximas a um valor verdadeiro" (THOMAZINI, ALBUQUERQUE, 2011, p.22). Precisão descreve a "característica relativa ao grau de repetitividade do valor medido" (THOMAZINI, ALBUQUERQUE, 2011, p.22); alcance ou range que "Representa toda a faixa de valores de entrada" (THOMAZINI, ALBUQUERQUE, 2011, p.23) que um sensor suporta. Estabilidade está diretamente "relacionada com a flutuação da saída do sensor" (THOMAZINI, ALBUQUERQUE, 2011, p.22). Velocidade de resposta é o "tempo com que a medida fornecida pelo sensor alcança o valor real do processo" (THOMAZINI, ALBUQUERQUE, 2011, p.22).

Existem outras diversas características dos sensores, porém Thomazini e Albuquerque (2011) tratam os citados acima como sendo os principais a serem levados em consideração durante a escolha.

1.3. Armazenamento de Dados

Por volta dos anos 60 surgem os primeiros Sistemas Gerenciadores de Banco de Dados (SGBD) , baseado nos antigos sistemas de arquivos, que tinham como propósito melhorar o sistema de armazenamento que era usado na época.

Para ser considerado um SGBD é necessário possuir algumas características como "controle de concorrência, segurança, recuperação de falhas, gerenciamento do mecanismo de armazenamento de dados e controle das restrições de integridade do BD" (LÓSCIO, 2019, p.3), além de ser capaz de cuidar do gerenciamento das transações. Lóscio (2019) define uma transação como uma coleção de operações dentro de um banco de dados que desempenha de forma lógica operações de escrita ou leitura.

Podemos destacar dois tipos de banco de dados, os bancos relacionais(SQL) e os não relacionais(NoSQL), embora bancos relacionais sejam usados em larga escala nos mais diversos tipos de aplicações, eles ainda possuem “muitas limitações no processamento de grandes quantidades de dados não estruturados como o *Big Data* gerado pela IoT” (KANG et al, 2016, p.1, tradução nossa), devido a essas limitações, o uso dos bancos SQL não seria a opção mais viável para aplicações IoT.

Lóscio (2019) define que bancos de dados NoSQL apresentam características que os tornam diferentes dos bancos SQL. A escalabilidade horizontal permite o aumento do poder de processamento através do aumento no número de máquinas disponíveis para o armazenamento e processamento dos dados. Essa característica se opõe à escalabilidade vertical , que se dá pelo aumento do processamento e armazenamento do servidor.

Este esquema flexível faz com que a forma de armazenar não fique presa a um único padrão, algo que facilita quando é necessário salvar dados não estruturados e favorece a escalabilidade, porém, não garante a integridade dos dados. Uma API simples para acesso aos dados pois a ideia dos bancos NoSQL é promover alta disponibilidade e acesso aos dados de forma simples e eficiente. Existem diversos modelos de bancos NoSQL, os mais utilizados são: chave-valor, orientado a documentos, orientado a colunas e orientado a grafos.

1.3.1. Chave-Valor (Key-Value)

Este modelo de banco “é considerado bastante simples e permite a visualização de dados como uma grande tabela *hash*” (LÓSCIO, 2019, p.7), sua implementação se dá de forma simples e seus dados são facilmente manipulados usando operações *get*() e *set*() através do esquema chave-valor.

1.3.2. Orientado a Colunas

Bancos orientados a colunas são um pouco mais complexos pois “Neste modelo os dados são indexados por uma tripla (linha, coluna e *timestamp*), onde linhas e colunas são identificados por chave e *timestamp*” (LÓSCIO, 2019, p.7), sendo o *timestamp* o responsável por fazer a diferenciação das versões de um dado.

1.3.3. Orientado a Documentos

Este modelo leva este nome pois ele armazena uma coleção de documentos, onde um documento ”é um objeto com um identificador único e um conjunto de campos que podem ser *strings*, listas ou documentos aninhados” (LÓSCIO, 2019, p.8), sendo assim esse modelo é muito flexível e não se prende a um esquema rígido “ou seja, não exige uma estrutura fixa

como ocorre nos bancos relacionais" (LÓSCIO, 2019, p.8) permitindo que novos campos possam ser adicionados de forma simples.

1.3.4. Orientado a Grafos

Um banco orientado a grafos "possui três componentes básicos: os nós(são os vértices do grafo), os relacionamentos (são as arestas) e as propriedades (ou atributos) dos nós" (LÓSCIO, 2019, p.9). Esse modelo é ideal para ser usado para consultas complexas, levando vantagem sobre bancos SQL.

2. Materiais e Métodos

Essa seção descreve os materiais e métodos utilizados para o desenvolvimento do projeto RDEV.

2.1. Materiais

Neste trabalho foram utilizadas ferramentas *Open Source* e componentes de baixo custo a fim de reduzir as despesas da produção e distribuição.

Para a leitura dos dados foi utilizado um sensor de umidade e temperatura DHT11 acoplado a um microcontrolador Wemos D1 mini. Ambos são alimentados por uma fonte de 5 volts através de porta USB.

Para a programação do microcontrolador foi utilizada a linguagem C++. O protocolo MQTT foi utilizado para a comunicação entre o microcontrolador/broker e *broker*/*coletor*. Utilizou-se o *broker* Mosquitto e o Telegraf para fazer a coleta dos dados no *broker* e salvá-los no banco de dados. Para o armazenamento das informações foi adotado o banco de dados InfluxDB e para a visualização gráfica dos dados a plataforma Grafana.

2.2. Métodos

A arquitetura utilizada é apresentada na Figura 6.

Figura 6: Arquitetura RDEV

Fonte: Elaborado pelo autor

Um sensor DHT11 é conectado ao microcontrolador Wemos através de uma porta GPIO, após estabelecer as conexões entre o sensor e o controlador, ambos são colocados dentro de um case de acrílico e na parte posterior do case é fixada uma fonte de 5 volts que será responsável pela alimentação do microcontrolador através de uma porta USB.

O kit contendo o sensor, microcontrolador e fonte podem ser instalados em salas de aula, laboratórios ou ambientes similares para coletar os dados do ambiente.

Ao conectar o kit a uma tomada, o sensor começa a coletar os dados do ambiente e enviar para o microcontrolador que por sua vez se conecta a uma rede WiFi pré-determinada. A aplicação embarcada no microcontrolador se inscreve como *publisher* em um tópico do *broker* e começa a enviar os dados coletados pelo sensor utilizando o protocolo MQTT.

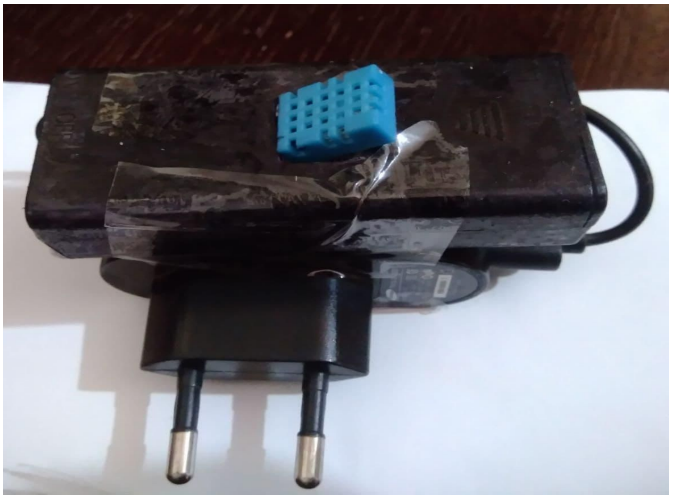
Ao receber os dados o *broker* os armazena temporariamente e notifica os *subscribers* que estão inscritos neste tópico. Para fazer o papel de *subscriber* foi utilizado o agente de coletas Telegraf que se inscreve nos tópicos do *broker* e começa a consumir os dados do *broker*. De posse desses dados o Telegraf os salva em uma *measurement* (equivalente as tabelas de bancos SQL) que é composta por quatro *fields* (equivalente aos campos dos bancos SQL) sendo eles o *timestamp, host, topic e value.*

Com os dados salvos no banco é necessário configurar o Grafana. Essa configuração é realizada em duas etapas, na primeira etapa é especificado o SGBD, a *measurement* e o *field* que será exibido. Depois de finalizar a primeira etapa é necessário designar quais gráficos serão utilizados, as unidades de medida utilizadas, o tempo de intervalo entre as leituras no banco, feito isso basta salvar o Dashboard.

3. Resultados

O RDEV foi construído seguindo a arquitetura especificada, na qual o sensor acoplado ao microcontrolador, mostrado na Figura 7, capta os dados de temperatura e umidade e envia para o *broker* utilizando o protocolo MQTT.

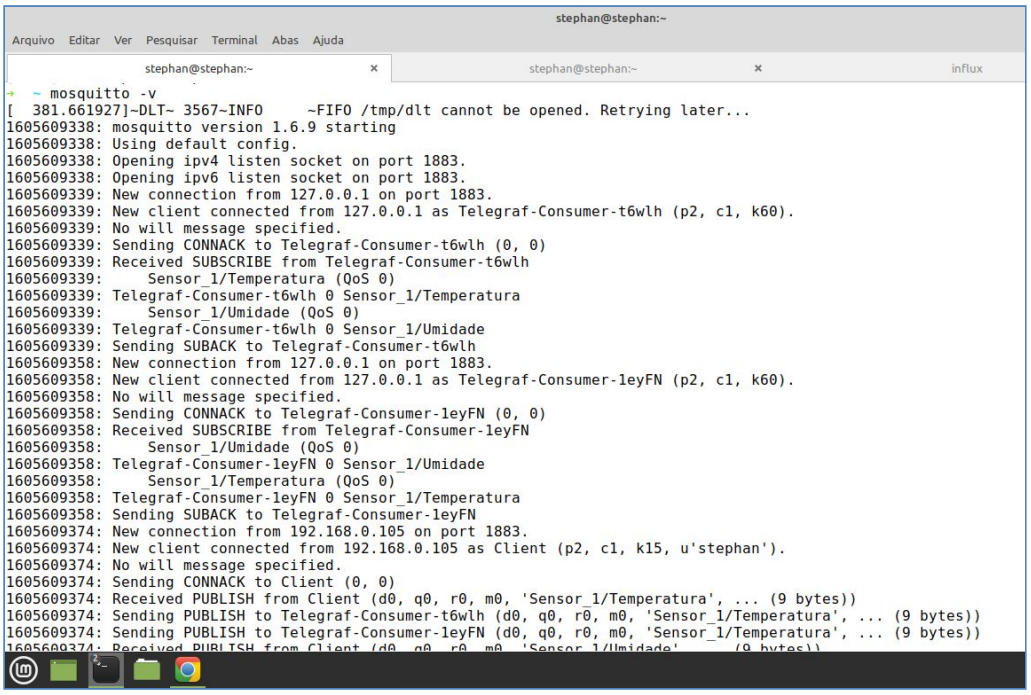
Figura 7: Protótipo



Fonte: Elaborado pelo autor

Com os dados temporariamente salvos no *broker,* como é mostrado na Figura 8 que mostra o *broker,* suas configurações e conexões com o Telegraf e o sensor e também mostra os dados chegando do sensor e sendo coletados pelo Telegraf, o agente coletor é notificado e busca os dados para então salvar no banco de dados.

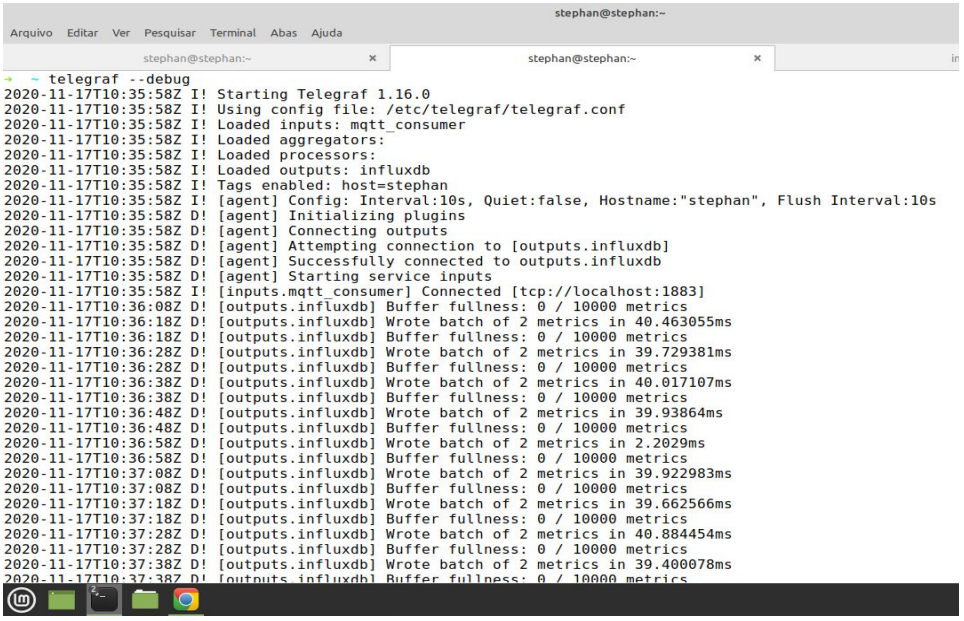
Figura 8: Broker Mosquitto



Fonte: Elaborado pelo autor

Com os dados em posse do *broker*, o Telegraf é notificado e inicia a busca dos dados como é mostrado na Figura 9. Aqui é possível ver a conexão do Telegraf com o InfluxDB e com o *broker,* a Figura também mostra o Telegraf salvando os dados no InfluxDB.

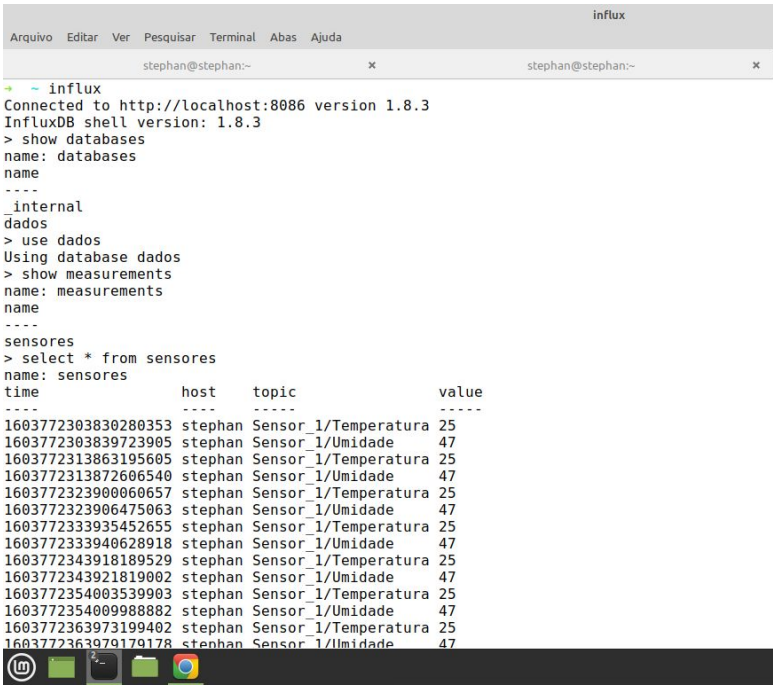
Figura 9: Telegraf



Fonte: Elaborado pelo autor

Na Figura 10 podemos observar a estrutura do banco de dados e os dados que já foram salvos nele.

Figura 10: InfluxDB



Fonte: Elaborado pelo autor

Com os dados já disponíveis no banco de dados é realizada a conexão entre o InfluxDB e o Grafana, que os exibe em quatro gráficos: dois de linha e dois de *gauges*. Cada par de gráfico *gauge*/linhas exibe valores de temperatura e umidade (Figura 11) na tela do Grafana. Os quatro gráficos serão gerados através de consultas no banco de dados que ocorrem em um período pré-determinado pelo usuário.

Figura 11: Grafana

Fonte: Elaborado pelo autor

Os gráficos da Figura 11 exibem a temperatura em graus Celsius e a umidade em umidade relativa (%), as cores azul, verde, amarela, laranja e vermelha representam os estados. A cor azul é para quando a temperatura ou a umidade estão abaixo do normal. O verde é para quando estão no nível ideal. O amarelo é para quando os valores estão um pouco acima do normal. Laranja é um sinal de alerta que indica que os valores estão bem acima da média aceitável. O vermelho representa valores muito altos em relação aos valores ideais.

Ainda é possível ver uma linha cortando os gráficos de linha, essa linha indica o ponto de alarme, quando o valor da umidade ou da temperatura ultrapasse, ela envia um *e-mail* de alerta é enviado para o *e-mail* configurado no sistema de alertas.

4. Conclusão

A ferramenta RDEV foi testada em diferentes ambientes e mostrou um desempenho satisfatório, conseguindo fazer leituras em ambientes de condições amenas e ambientes mais frios e secos.

O Grafana se mostrou uma ferramenta potente e de fácil utilização, permitindo a construção desde *Dashboards* mais simples até mais complexos. Além disso, ele oferece um sistema de alarmes com mais de 15 canais de alerta.

O banco de dados Influx se mostrou uma ótima escolha por facilitar o processo de integração com o Grafana, especialmente por prover um carimbo automático de data e hora.

O Telegraf por ser da mesma empresa que o Influx facilitou a integração e provendo um funcionamento satisfatório como agente intermediador, coletando os dados do broker e salvando no Influx.

O desempenho do Mosquitto como *broker* também foi satisfatório atendendo perfeitamente as expectativas, o microcontrolador foi facilmente configurado e ofereceu um bom poder de processamento o que permitiria a criação de novas funcionalidades.

O sensor de temperatura e umidade DHT11 apresentou boas leituras, para validar suas leituras a qualidade das leituras o sensor foi exposto a diferentes tipos de ambientes e condições, as leituras se mantiverem condizentes com os ambientes testados, apresentando queda de temperatura quando colocado em um ambiente mais frio e aumento da temperatura ao ser deslocado para um ambiente mais quente.

Mesmo apresentando boas leituras, o uso de outro sensor como o DHT22 poderíamos ter medições ainda mais precisas.

No contexto geral, podemos concluir que a ferramenta alcançou as expectativas, sendo capaz de monitorar diferentes ambientes.

Referências

AL-FUQAHA, Ala; GUIZANI, Mohsen; MOHAMMADI, Mehdi ; MOHAMMED, Aledhari; AYYASH, Moussa (2015). Internet of Things: A Survey on Enabling Technologies, Protocols and Applications.

CASSINELLI, Fernanda; MIRANDA, Lucas; VALE, Lucas (2018). Constrained Application Protocol.

ČOLAKOVIC, A., & HADZIALIC, M. (2018). Internet of Things (IoT): A review of enabling technologies, challenges, and open research issues. Computer Networks.

KANG, Y.S., PARK, I.H.; RHEE, J.; LEE, Y.H. (2016). MongoDB-based repository design for IoT-generated RFID/sensor big data. IEEE Sensors Journal.

LI, S., DA XU, L., ZHAO, S. (2015). The internet of things: a survey. Information Systems Frontiers.

LÓSCIO, Bernadette. (2019). NoSQL no desenvolvimento de aplicações Web colaborativas.

MADAKAM, S.; RAMASWAMY, R.; TRIPATHI, S. (2015). Internet of Things (IoT): A literature review. Journal of Computer and Communications.

MATTERN, F; FLOERKEMEIER, C. (2010) From the internet of computers to the internet of things, From active data management to event-based systems and more: papers in honor of Alejandro Buchmann on the occasion of his 60th birthday.

PERUMAL, T; SULAIMAN, M.N.; LEONG, C. (2015). Internet of Things (IoT) enabled water monitoring system.

THOMAZINI, Daniel; ALBUQUERQUE, Pedro U. Braga. Título: Sensores Industriais Fundamentos e Aplicações. Ed.4°. Erica, 2011.