Perspectiva do IoT e softwares de gerenciamento de redes

Gabriel Baia Da Silva

Orientador: Dionne Cavalcante Monteiro

Instituto de Ciências Exatas e Naturais - Universidade Federal Do Pará - Faculdade de computação - Campus Universitário Do Guamá

GabrielKadran@hotmail.com
Dionne@ufpa.br

Abstract. *This work will study the functioning of computer network management systems, from the perspective of an IoT application that will collect ambient temperature and humidity data and pass it on to a pipeline of applications that will distribute data, starting with the MQTT protocol that will using a public broker, then going to a Node-RED hub, and then PRTG, and finally being passed on via alarms to Telegram.*

Resumo. *Este trabalho estuda o funcionamento de sistemas de gerenciamento de de redes de computador, a partir da perspectiva de uma aplicação IoT que coleta dados de temperatura ambiente e umidade e repassar para uma pipeline de aplicações que distribuem estes dados, começando pelo protocolo MQTT que usa um broker público, depois indo para um hub Node-RED, e então o PRTG, e por fim sendo repassado por meio de alarmes para o Telegram.*

1. Introdução

Internet das coisas, ou *Internet of Things* (IoT), é um conceito que abrange objetos, eletrônicos ou eletrodomésticos que se interconectam a uma rede via internet. A IoT está cada vez mais presente no cotidiano das pessoas, com um número estimado de 16,8 bilhões de dispositivos conectados no final de 2023 (Satyajit, 2024). Com este número crescente é importante o estudo do uso desses dispositivos e como os conectar a uma rede para trabalho.

A IoT é a junção de diversas tecnologias tais como: sistemas embarcados, meios de comunicação e protocolos específicos para o tráfego de poucas informações oriundos de dispositivos de baixa capacidade de processamento.

Os sistemas embarcados em IoT são compostos por: unidade de processamento, sensor para coleta de dados e/ou atuadores para controlar equipamentos, sistemas de comunicação de dados, e bibliotecas que disponibilizam os protocolos. As unidades de processamento de dispositivos IoT são de capacidade reduzida, podendo processar informações de 8 a 32 bits. Os sensores ou atuadores são conectados às unidades de processamento de dados para capturar dados do entorno do sensor ou para acionar equipamentos. A comunicação dos dados é realizada através de conexões físicas com ou sem fio (Ethernet, Wi-Fi, ZigBee) e sua principal função é enviar dados coletados dos

sensores, ou receber dados para os atuadores, que serão processados por servidores com maior capacidade de processamento. Os protocolos, tal como TCP/IP (*Transmission Control Protocol*/*Internet Protocol*), UDP/IP(*User Datagram Protocol/Internet Protocol/Internet Protocol*), MQTT(*Message Queuing Telemetry Transport*), são usados para empacotar e disponibilizar as informações para todos os equipamentos interessados nos dados coletados dos sensores.

Softwares específicos para IoT são utilizados para o processamento e visualização de dados nos servidores. Dentre os diversos existentes na literatura pode-se destacar o Node-RED, que é uma ferramenta de programação baseada em fluxos, que executa Node.js em *runtime*, em nós programáveis. O Node-RED é útil para a integração IoT, com o fluxo separado em caixas que executam ações e as passam para o próximo nó do fluxo, tornando a configuração e programação de IoT mais prática. Outro conjunto de softwares que podem fornecer a visualização de dados são os softwares de gerenciamento de redes de computadores, que já possuem uma estrutura de banco de dados necessária para tratamento de dados e emissão de alertas, tais como o OpenNMS (*Open Network Monitoring Software*), Zabbix e PRTG (*Paessler Router Traffic Grapher*).

O trabalho foi desenvolvido usando um sensor de temperatura DHT11, conectado a uma placa D1 mini ESP8266. Esta placa possui capacidade de conexão Wi-Fi e usará um gerenciador Wi-Fi configurado em um trabalho anterior para providenciar a conexão com a internet. O código usado neste projeto estará disponível nos anexos deste trabalho.

Depois de conectado o sensor envia os dados coletados via protocolo MQTT para um *broker* MQTT em nuvem, este *broker* público, providenciado por meio do endereço “broker.emqx.io” fornece uma estrutura de multicast, ou seja, é capaz de enviar dados a todos os interessados da rede ao mesmo tempo. Estes dados serão acessados por meio do protocolo MQTT configurado no Node-RED.

2 - Revisão bibliográfica

Na seção seguinte iremos discutir a pilha de protocolo de redes, tecnologias e softwares usados na área de gerenciamento de redes e seus protocolos de comunicação, com ênfase aos que serão usados neste trabalho.

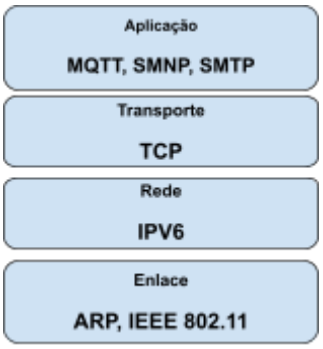
2.1 - Pilha de protocolos de redes de computadores

Protocolos de redes de computadores são usados em conjunto em uma “pilha” de protocolos dividida em camadas. Existem modelos de pilhas de protocolos como o OSI de 7 camadas ou TCP/IP de 4 camadas, estas camadas trabalham em conjunto e usam protocolos diferentes para situações diferentes.

Para a compreensão dos protocolos de rede é necessária a ilustração do modelo de pilha de protocolos **TCP/IP**, adotado como padrão para redes de internet em 1983(KUROSE, ROSS, 2013, p. 46), conforme mostrado na figura 1. O modelo é composto por 4 camadas ilustradas na imagem abaixo. As camadas são **aplicação, transporte, rede e enlace** (IBM, 2024). Estas camadas funcionam em conjunto para transmitir com segurança uma mensagem via internet ou conexão local (LAN).

A camada mais próxima do usuário final é a camada de aplicação, enquanto que a mais próxima da rede física de comunicação, responsável pela transmissão de dados, e a camada de enlace, realizando uma série de operações, transmissões, encapsulamento de dados e tradução de endereços para o envio da mensagem ao destinatário final.

Figura 1 - Protocolos de rede no modelo TCP/IP



Fonte: Elaborado pelo autor

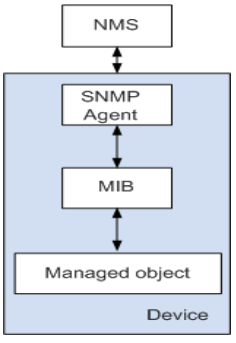
Esta pilha ilustra apenas os protocolos mais relevantes para o trabalho realizado aqui, existe uma gama enorme de protocolos de rede que se encaixam na pilha TCP/IP que não foram mostrados aqui. Para o trabalho iremos nos focar na camada de aplicação que contém o protocolo MQTT, usado na metodologia do trabalho.

2.2 Camada de Aplicação

Na camada de aplicação há uma quantidade enorme de protocolos que podem ser utilizados para a coleta dos datagramas sendo repassados pelo TCP, dentre eles iremos destacar 2 protocolos: SNMP(*Simple Network Manager Protocol*) e MQTT.

O protocolo SNMP foi criado para o controle sobre os ativos de redes de computadores de forma a minimizar os requisitos que um participante da rede terá para se integrar a ela. Este protocolo é feito de forma que a grande parte do trabalho será realizado por um nó mestre, com agentes desse mestre apenas repassando informações e comandos para os computadores conectados, o protocolo é ilustrado na figura 2:

Figura 2 - Estrutura do SNMP



Fonte: "what is SNMP?", Huawei Technical Guides (2024)

O protocolo SNMP separa a rede em 4 componentes: NMS, Agent, MIB e Managed Object. O NMS (Network Management System) é o componente gerente da rede, ele funciona a partir de um servidor central que recebe e envia *queries* para os vários agentes (HUAWEI, 2024).

O agente é um programa que está instalado em um ponto final que é responsável por responder e repassar os queries recebidos do NMS por meio do que são chamadas de SNMP traps, mensagens que não esperam resposta ou por meio de Inform Requests, que esperam resposta do NMS. O MIB é um componente do dispositivo gerenciado que possui todos os valores e variáveis que são relevantes para o gerenciamento do sistema, como nome, atributos, acesso e tipos de dados que estão presentes no objeto. Por fim o *Managed Object* ou "Objeto Gerenciado" em si é o componente que possui de fato os dados e atributos que o sistema está gerenciando.

No SNMP a quantidade de dados gerada é considerada adequada para aplicações em desktop, com tamanho de pacote de até 1500 bytes (Hewlett-Packard, 2016), é um pacote mais pesado para aplicações IoT, portanto temos que procurar um protocolo mais leve como o MQTT.

O protocolo MQTT é um protocolo de transporte que usa o modelo *publish/subscribe* para transmitir informações para todos os interessados. É um protocolo leve e simples, com um foco em mensagens pequenas, ideais para IoT e sistemas embarcados com pouco poder de processamento, como os sistemas embarcados e sensores de temperatura usados neste trabalho. É uma alternativa ao SNMP que é um protocolo mais robusto, porém possui um *overhead* maior, que aumenta o volume de dados no transporte da informação.

O MQTT se originou em 1999 com a intenção de economizar bateria na conexão de Oleodutos via satélite (STEVE, 2018), os seus requerimentos eram: implementação simples, QoS(*Quality of service*), leveza e pouco consumo de banda, *data agnosticism*, ou seja, a capacidade de passar qualquer tipo de dado, no tamanho adequado, sessão contínua de duração longa, leveza e eficiência, comunicação bidirecional, escalabilidade, capacidade de QoS, sessões persistentes e a segurança (HiveMQ, [s.d]).

A estrutura de uma mensagem MQTT é formada por um valor de controle de tipo de pacote como CONNECT, PUBLISH ou SUBSCRIBE, e flags para serem usadas na operação da mensagem, por exemplo, as flags usadas para a operação PUBLISH, conforme observado na figura 3.

Figura 3 - Pacote MQTT

Bit	7	6	5	4	3	2	1	0
byte 1	MQTT Control Packet type				Flags specific to each MQTT Control Packet type			
byte 2…	Remaining Length							

Fonte: MQTT 3.1.1 Oasis standard

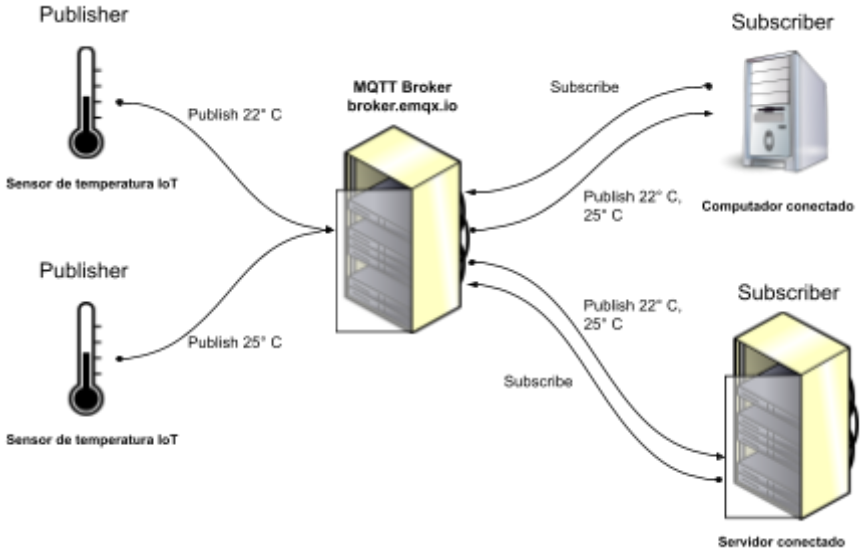
O modelo *publish/subscribe*, visto na figura 4, é uma alternativa ao modelo cliente-servidor tradicional, no qual há *publishers* que enviam mensagens e *subscribers* que as recebem. Esses membros são conectados por meio de um servidor (*broker*) que distribui essas mensagens as separando em tópicos.

O modelo publish/subscribe é feito de forma que *subscribers* e *publishers* não se comunicam diretamente, cortando a conexão direta entre distribuidor e receptor das mensagens. O *broker* será responsável por realizar essa conexão.

Neste modelo as mensagens são todas enviadas para o *broker*, que as filtra e envia apenas para os *subscribers* que estão interessados naquele tópico de mensagens. O

broker é uma aplicação separada dos *subscribers* e publishers e podem ser acessados de forma online em nuvem, ou hospedados localmente com aplicações como o MQTT Mosquito.

Figura 4 - Modelo Publish/Subscribe



Fonte: Elaborado pelo autor

O MQTT possui *Quality of Service* (QoS), que é responsável pela garantia da entrega de mensagens. O modelo de QoS se divide em 3 níveis de garantia de entrega: a) "QoS 0" no não existe confirmação da entrega da mensagem; b) "QoS 1" garante confirmação de "apenas uma vez" para as mensagens, onde o *publisher* envia a mensagem e espera resposta, se o *broker* não responder, a mensagem é reenviada, independente desta ser duplicada, ou não; e c) "QoS 2" é usado onde a perda de mensagens é crítica, porém a duplicação de mensagens também não é aceitável, e a mensagem faz parte de um sistema no qual o *publisher* e o *broker* precisam confirmar o recebimento da mensagem.

O MQTT possui também sistema de persistência de mensagens, as quais serão guardadas no servidor (*broker*) até a confirmação de recebimento da mensagem, que é útil no caso de falha de comunicação. Por padrão, o MQTT usa mensagens sem persistência, então no caso de queda da rede a mensagem é perdida.

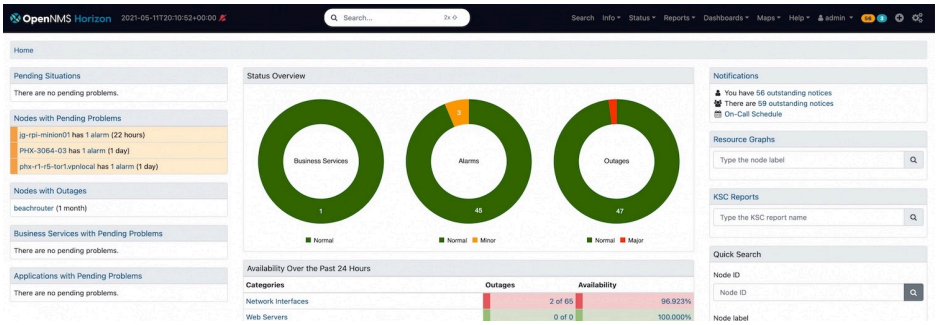
O MQTT é adequado para IoT por ser leve em comparação ao SNMP. O seu pacote possui comprimento de apenas 8 bits, o pacote SNMP pode ter até 1500 bytes de tamanho. Um aumento de carga inviável para aplicações do tipo.

2.3 - Softwares de gerenciamento de redes de computadores

Softwares de gerenciamento são aplicativos os quais implementam e utilizam os protocolos discutidos anteriormente. Dentre os mais utilizados destacam-se o OpenNMS, Zabbix e o PRTG, sendo este último o software utilizado na prática do trabalho.

O OpenNMS providencia uma interface que centraliza o gerenciamento de redes. A interface é um *dashboard* customizável com a opção de interagir com os sistemas conectados ao OpenNMS, vista na figura 5.

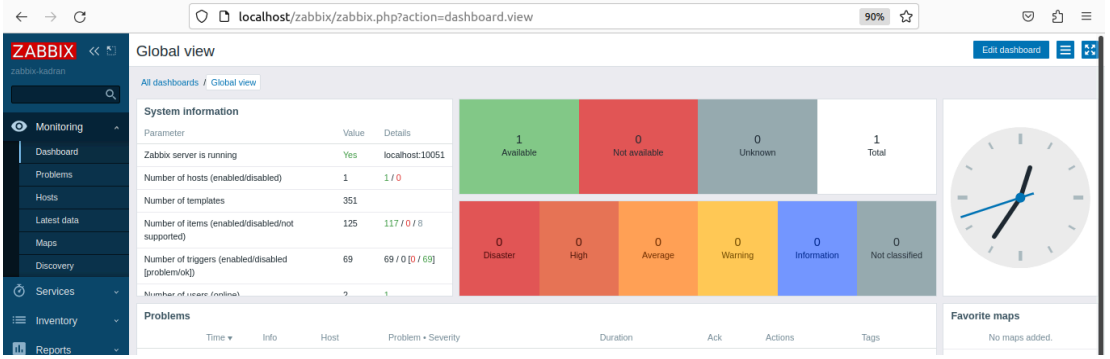
Figura 5 - Interface OpenNMS



Fonte: OpenNMS Horizon , ([s.d.])

Outro software de gerenciamento de rede é o Zabbix (ZABBIX, [s.d.]), com ênfase em customização e modularidade de funções que podem ser instaladas. Devido a modularidade, o Zabbix é compatível com uma grande quantidade de casos de uso que sejam necessários para cada administrador de rede, porém por ser customizável ele é dependente de documentação e experiência dos administradores. O Zabbix pode ser visto na figura 6.

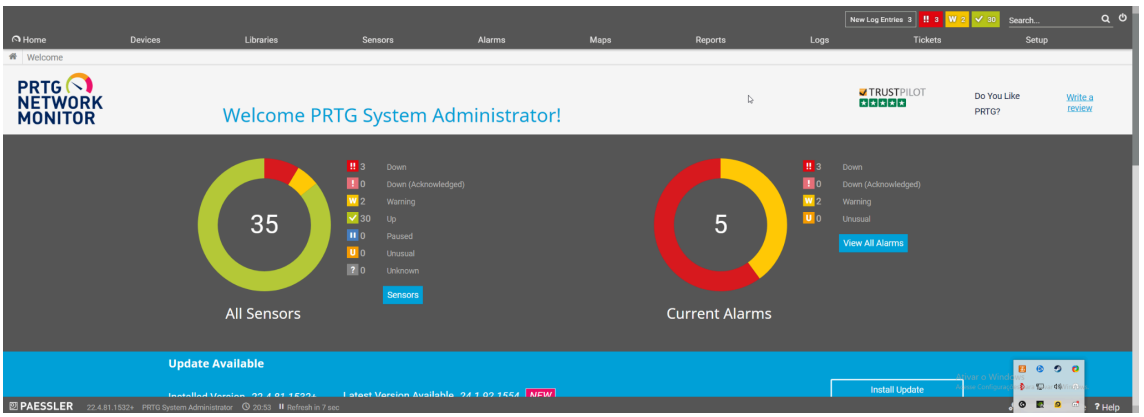
Figura 6 - Interface do dashboard Zabbix



Fonte: Elaborado pelo autor

O PRTG é um software de gerenciamento de rede que é mais facilmente instalado e mais robusto por padrão do que os outros mostrados anteriormente. O PRTG, assim como os outros, possui uma interface baseada em web, vista na figura 7, ou um aplicativo para desktops. Na qual é feita a configuração dos dispositivos e sensores.

Figura 7 - Interface Web do PRTG

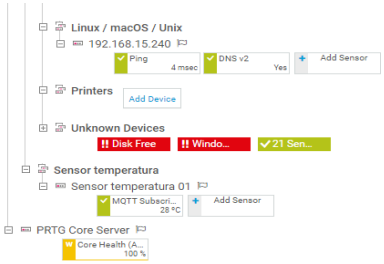


Fonte: Elaborado pelo autor

O PRTG é dividido em 3 partes (PAESSLER, 2024 p. 124): **Sistema,** que inclui o *PRTG core server* e as *probes* de coleta de dados; **Interfaces,** que possui a *interface web* que é usada para a visualização dos dados; e a **ferramenta de administração** do *PRTG core server*.

Uma instalação de core PRTG virá com *probes* que monitoram o sistema local, com relatórios de performance do sistema. Dentre estes sensores conectados estarão os necessários para os propósitos deste trabalho: a conexão com o MQTT, que é feita por meio de um sensor chamado de *MQTT Subscribe Custom sensor*, o qual recebe os dados dos sensores remotos de temperatura, a interface de *probes* é ilustrada na figura 8:

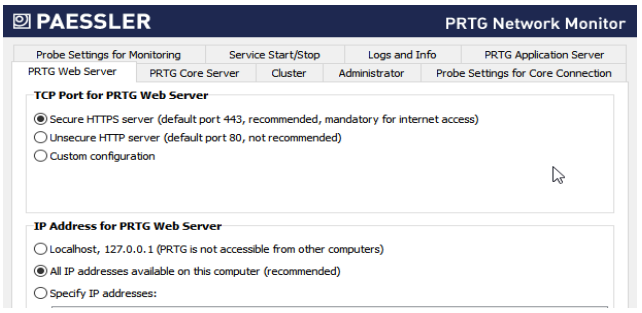
Figura 8 - Interface de probes do PRTG



Fonte: Elaborado pelo autor

Por fim, o PRTG possui um aplicativo de administração chamado de *PRTG Administration Tool,* visto na figura 9, o qual será usado para configurar acesso a interface web, configurações locais de IP e porta, modos de conexão para as *probes*, clusters e linguagem do sistema.

Figura 9 - PRTG Administration Tool



Fonte: Elaborado pelo autor

3 - trabalhos relacionados

Em trabalhos similares procuramos aplicações do MQTT em IoT. No trabalho de R. A. Atmoko foi realizada a comparação entre o HTTP e o MQTT, no qual foram mostradas as vantagens do MQTT, tanto para velocidade quanto para tempo de vida de baterias em sistemas embarcados. Para a coleta de informações usaram uma placa ESP8266 e DHT11, da mesma forma que neste trabalho, com o objetivo de coleta de dados de performance do protocolo MQTT.

No trabalho foi feito um experimento de transferência de dados para um SGBD (Sistema Gerenciador de Banco de Dados), que durou 60 segundos por teste. Foi encontrado que na média de 5 testes o HTTP transferiu dados com sucesso 934.4 vezes e o MQTT transferiu dados 6502.2 vezes, mostrando a velocidade maior do protocolo MQTT.

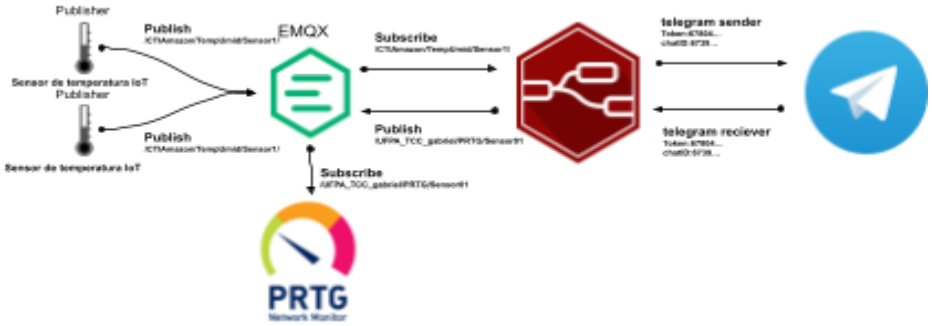
O trabalho de Megha Q. (2019) aplicou o MQTT em um ambiente de *smart-home*, com uso de tecnologia baseada em servidor cloud para acesso das informações coletadas pelos sensores da casa. A aplicação do trabalho foi o monitoramento de temperatura remoto por meio das tecnologias MQTTBox, responsável por criar clientes MQTT de forma virtual. Desta forma, testes com os dispositivos conectados à rede foram feitos, induzindo uma carga nos dispositivos por meio do software.

Um ambiente cloud foi utilizado no trabalho que ficou responsável por armazenamento de dados, interface web e configuração da aplicação. Foi medido no trabalho o tempo de resposta da aplicação, com valores de publish e de resposta de QoS sendo consideradas. No trabalho foi encontrado que conforme o aumento de requisitos de publish, aumenta também o tempo de respostas para cada resposta de QoS em valor "1", no publish de 10 mensagens o tempo de resposta foi de 4,52 segundos e para 20 mensagens o tempo foi de 5,42 segundos.

4 - Metodologia

Nesta seção é mostrada a metodologia do trabalho que foi realizado e detalhes sobre as tecnologias presentes, com especificações técnicas e explicações sobre a implementação do software de gerenciamento de redes, o protocolo MQTT, as unidades de processamento ESP8266, sensores de temperatura e umidade, e o *hub* Node-RED usado para processar os dados e enviá-los para o PRTG e para alarmes. A figura 10 ilustra as tecnologias usadas.

Figura 10 - Tecnologias usadas na rede



Fonte: Elaborado pelo autor

No trabalho será usado o *broker* MQTT público **emqx.io**, que possui capacidade limitada de conexões. Caso seja necessário um *broker* com maior capacidade, pode-se usar o Eclipse Mosquito MQTT *broker* para um trabalho em grande escala.

Para a coleta de dados usamos o sensor DHT11 que possui grande integridade e confiabilidade para a coleta de dados por um período longo (MOUSER, [s.d.]). Este sensor é ligado a uma placa de processamento ESP8266EX, em um SoC (*System on a chip*) que possui capacidade de conexão Wi-Fi (Espressif, 2015), usada para o envio de dados para o monitor de rede via o protocolo MQTT.

O Node-RED é uma ferramenta de conexão entre os dispositivos de rede, a qual fornece uma interface gráfica baseada em fluxogramas, que facilita a programação da comunicação entre os dispositivos usando JavaScript com Node.js (OpenJS,[s.d.]). Na implementação do Node-RED, incluímos um nó programável que se comunica com o Telegram, enviando mensagens de texto automáticas como alarmes.

Para monitorar a rede usamos o PRTG, um software de gerenciamento de redes,

responsável por mostrar os dados coletados pelos sensores usados no trabalho.

Os dados coletados nos sensores DHT11 são coletados pelo ESP8266 que os envia através de Wi-Fi para o *broker* MQTT, o qual publica as informações de forma bruta. Estas informações são então coletadas no ambiente Node-RED, e processadas para que possam ser lidas pelo software de redes PRTG ou enviadas para o *chatbot*.

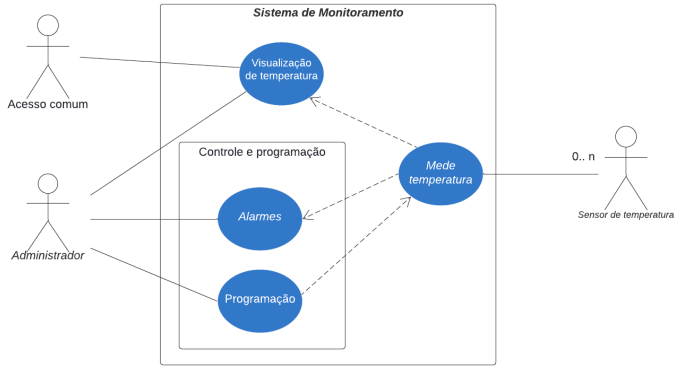
Para alarmes de estado do sensor foi adicionado uma conexão a um *chatbot* do Telegram que envia mensagens automáticas para os administradores. Este *chatbot* é uma alternativa a uma implementação que usa notificações diretamente no PRTG por meio de emails.

4.1 Diagramas do sistema desenvolvido

A seguir demonstramos os aspectos do sistemas através de dois diagramas UML: Casos de uso e Diagrama de Sequência.

O diagrama de casos de uso**,** ilustrado na figura 11, mostra que o sistema é usado por dois tipos de usuários com dois acessos possíveis, um acesso comum que irá apenas visualizar a temperatura dos sensores e um acesso de administrador que é responsável pelo controle do sistema.

Figura 11 - Diagrama de Casos de uso



Fonte: Elaborado pelo autor

O administrador possui o controle e programação, e pode visualizar os dados do sistema por meio dos vários pontos finais como o PRTG, *Dashboard* Node-RED e programação do firmware da placa ESP8266.

O diagrama de sequência, figura 12, ilustra todos os estados esperados do sistema de sensoriamento desenvolvido. O diagrama é dividido em duas porções: uma porção online que considera o funcionamento esperado do sensor e uma porção de sequência offline que trata de quando a conexão com o sensor estiver perdida.

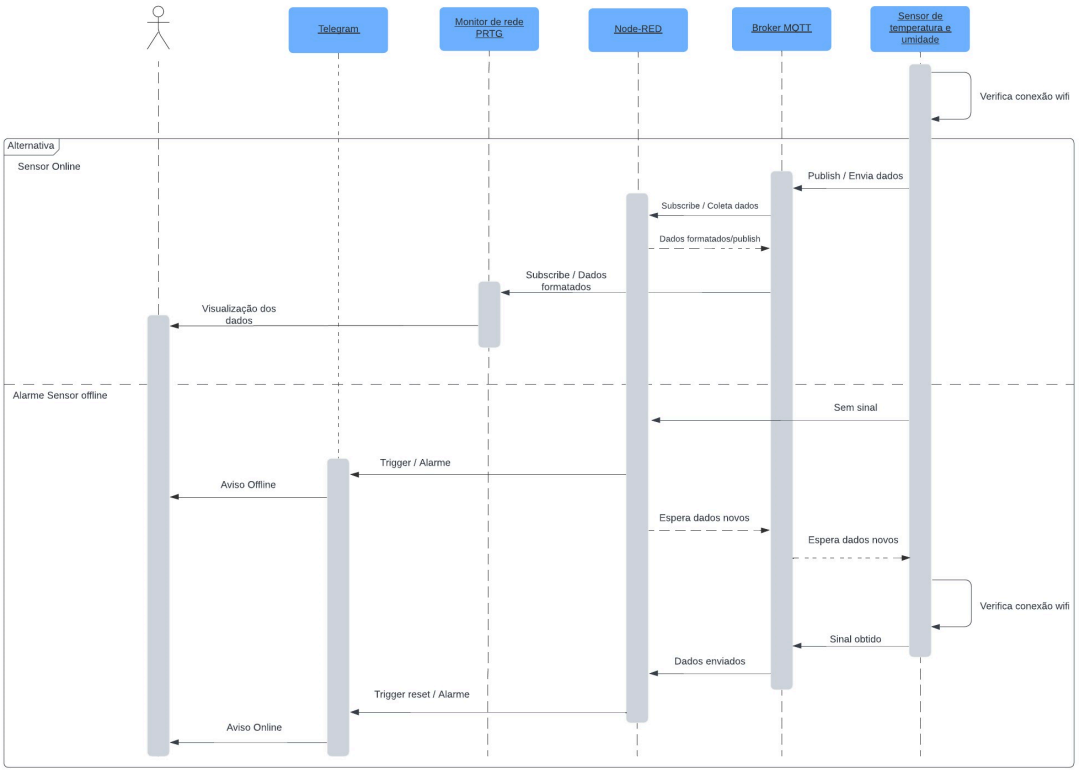
A sequência *online* parte da ativação do sensor, que quando ligado a rede elétrica automaticamente efetua conexão com o *broker* através do Wi-Fi e inicia a coleta, tratamento e envio de dados de temperatura e umidade para o *broker* utilizando conexão do tipo *publish*.

O *broker* por sua vez coleta estes dados e os envia para os *subscribers* que estão inscritos no tópico, então os nós do Node-RED. os formata em JSON corretamente e os enviará para o PRTG, que pode exibir os dados coletados em seu dashboard.

A sequência *offline* inicia a partir da ausência de sinal do sensor que ativa um nó trigger configurado no Node-RED. Este trigger envia uma mensagem de alarme ao

chatbot Telegram que mostra o estado offline do sensor para o administrador. Este aviso permanece enquanto o sensor não enviar dados novos ao *broker*.

Figura 12 - Diagrama de Sequência

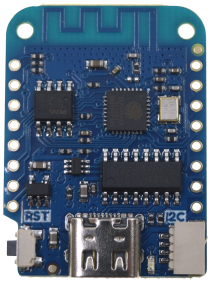


Fonte: Elaborado pelo autor

4.2 Sistema de sensoriamento

O SoC D1 Mini é uma placa Wi-Fi com memória e um processador embutidos feito pela Wemos, ilustrado nas figuras 13 e 14. Foi elaborada para dar capacidade de comunicação de dados para aplicações IoT, possuindo 4 MB de memória flash, clock de 80/160 MHz, pinos configuráveis, com capacidade de processar dados simples, e pouco consumo de energia (WEMOS, 2021). Na data de publicação deste trabalho o preço deste dispositivo está em uma média de R$30,00.

Figura 13 - Frente do SoC D1 Mini



(a)

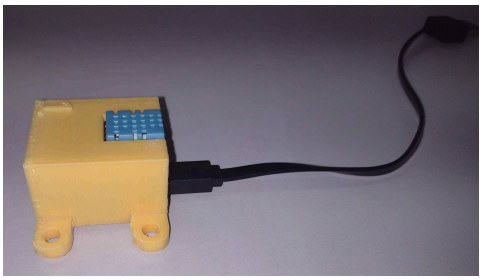
Figura 14 - Verso do SoC D1 Mini

(b)

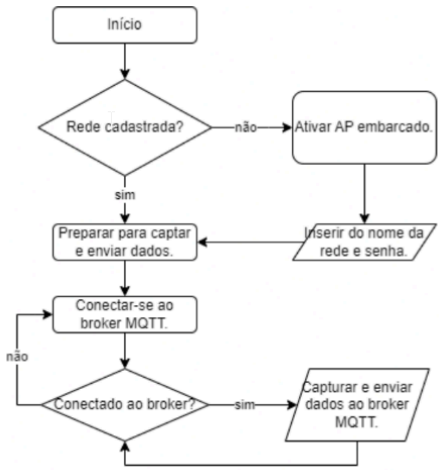
Fonte: Wemos, 2021

A placa possui um firmware desenvolvido por (SAMUEL, 2023), no fluxograma da figura 15, que mostra em detalhes mais informações sobre as suas possíveis configurações e conexão com sistemas gerenciadores de banco de dados.

Figura 15 - a) Placa ESP8266 com DHT11; b) Fluxograma de conexão do firmware



(a)



(b)

Fonte: a) Elaborado pelo autor; b) Samuel, 2023.

O sensor DHT11 é conectado à placa D1 Mini e captura a temperatura e umidade ambiente. A temperatura será medida com resolução de 1°C, com precisão de ± 2°C, dentro da faixa de 0 a 50 °C. A umidade será medida em umidade relativa do ar (UR), com resolução de 1% e precisão de 5%, dentro da faixa de 20-90% UR (MOUSER, [s.d.]). Assim como a placa, o sensor é barato, custando apenas R$13,00 na data deste trabalho.

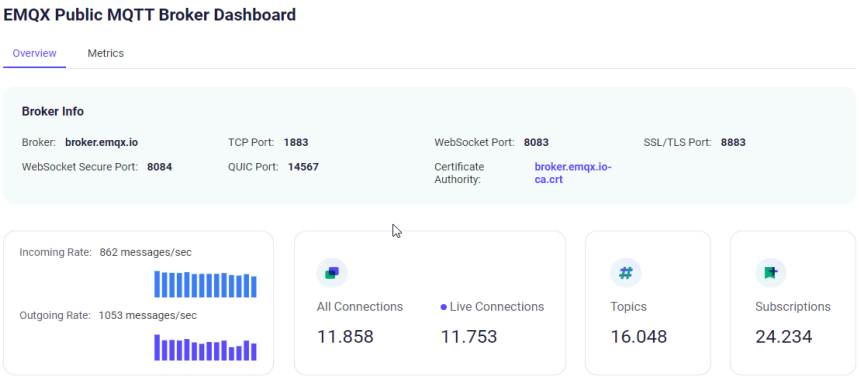
4.3 MQTT broker

O *broker* MQTT escolhido para o trabalho foi o EMQX (EMQX, [s.d.]), que fornece um *broker* público para testes no endereço broker.emqx.io. Para usá-lo é necessário apenas que o firmware do sensor faça referência ao endereço e que o tópico publicado seja acessado no software de monitoramento, fechando o modelo *publish/subscribe.* O EMQX também pode ser instalado localmente por meio de um software livre disponível para download, este software é configurável e escalável, compatível com IoT e mensagens instantâneas. Na figura 16 podemos ver o *dashboard* de todas as conexões ativas no broker público.

O broker foi escolhido por não ser necessária a implementação de um broker local, como o Mosquitto MQTT, que teria uma capacidade maior de host de dispositivos de rede. O broker público EMQX irá suprir a necessidade de apenas um dispositivo IoT, que foi usado para o estudo de caso que foi realizado neste trabalho.

Os limites do broker público, segundo a EMQX(2024) são: Dependência do provedor de serviços, que pode vir a cair e interromper a coleta de dados; Customização limitada, na qual apenas o essencial do MQTT é oferecido; Possíveis problemas na privacidade dos dados, os quais são devido a natureza pública do broker.

Figura 16 - Dashboard Broker MQTT público



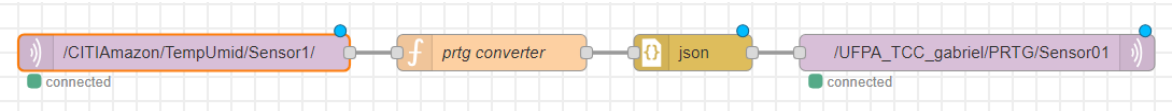
Fonte: EMQX (2024)

4.4 Node.js e Node-RED

Node.js é um ambiente de execução de javascript usado para executar javascript em diferentes plataformas, no nosso caso IoT, este ambiente é assíncrono, evitando deadlocks na rede. O Node.js será encapsulado no Node-RED, um ambiente de programação baseada em fluxo, no qual iremos controlar o fluxo de dados entre o *broker* MQTT e os sistemas de monitoramento de redes.

O Node-RED funciona com **nós programáveis** (OpenJS, [s.d.]) que se organizam em um fluxo de dados. Aqui usamos nós *mqtt in* e *mqtt out*, com esses nós tendo um tópico de subscribe para *mqtt in*, **/CITIAmazon/TempUmid/Sensor1/** e um tópico de publish **/UFPA_TCC_gabriel/PRTG/Sensor01** para *mqtt out*. Na figura 17 vemos como a mensagem passa pelo fluxo do sensor ao PRTG, na forma de um JSON com os dados de temperatura e umidade e os retransmitimos para o PRTG com a formatação correta:

Figura 17 - Fluxo Node-RED



Fonte: elaborado pelo autor

O fluxo mostrado recebe dados do *broker* sem formatação correta para o PRTG, então a função *prtg converter* converte estes dados e os retransmitir para o *broker*, que será dado em um tópico novo para o PRTG (veja o esquema da rede no início desta seção).

Os dados são recebidos no formato JSON `{ Sensor: 1, temperatura: 35, umidade: 30 }` e são retransmitidos no formato `{ Sensor1: {temperatura: 35, umidade: 30}}`. Sem esta formatação o PRTG não será capaz de receber a informação de forma correta.

4.5 Telegram receiver/sender

O Node-RED possui capacidade de conexão com o Telegram, por meio do template de nó feito por (WIND K. 2024), um programa de comunicação por chat, na qual podemos

programar um *chatbot* que envia para o Telegram conectado mensagens de estado do sensor.

Para a configuração do *chatbot* usamos o nós *Telegram receiver* e *Telegram sender,* a configuração do Telegram requer um ID de chat recebido pelo. Na figura 18, recebemos uma mensagem pelo *Telegram receiver*, no qual o *chatbot* envia uma mensagem para o Node-RED com as informações necessárias:

Figura 18 - Payload recebido no Telegram Receiver.

Fonte: elaborado pelo autor

Com a informação do *chatId*, conseguimos nos comunicar com o Telegram a partir do *Telegram sender* e formatamos a mensagem na função *Formata mensagem* que realiza uma operação de anexar os dados necessários para a conexão com o Telegram.

Usamos a comunicação com o Telegram para construir um sistema de alarmes que indica o estado do sensor, com a verificação por meio do nó *Trigger 200s*. Este nó verifica a chegada de mensagens a cada 200 segundos, na ausência de uma mensagem recebida da comunicação mqtt, será enviada uma mensagem para o telegram indicando que o sensor está offline.

Os nós de *Status check, Filter e trigger 200s* determinam se houve alguma mudança que necessita de alarme para o administrador da rede. A função *Status check* verifica a temperatura do sensor, adicionando ao payload da mensagem um texto relevante a temperatura encontrada, na figura 19 vemos como a função adiciona informação extra a mensagem de acordo com a temperatura.

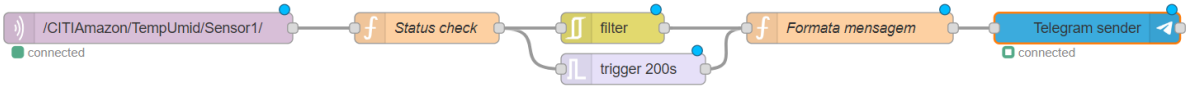
Figura 19 - Função status check

```
if (msg.payload.Sensor==1)
{msg.payload.Sensor="Sensor 1"};
if (msg.payload.temperatura>30)
    {msg.payload.status="temperatura ACIMA de 30!"}
else if (msg.payload.temperatura<15)
    {msg.payload.status = "temperatura ABAIXO de 15!";}
else
    {msg.payload.status = "temperatura NORMAL!";}
return msg;
```

Fonte: Elaborado pelo autor

A mensagem passa então por uma de duas possíveis entradas para o telegram: *Filter ou trigger 200s*. A mensagem será filtrada por *filter*, caso o valor de *status*, obtido na função *Status check* seja diferente ao anterior. Na figura 20 adicionamos os nós *filter* e *trigger* ao fluxo, construindo uma forma de enviar o alarme.

Figura 20 Fluxo Node-RED com trigger



Fonte: Elaborado pelo autor

Na função status check, a mensagem se mantem no mesmo estado até uma mensagem diferente ser lida do status, por exemplo, se o status “Temperatura ABAIXO de 15!” não for alterado, a mensagem será mantida e não enviada. Caso o valor seja alterado, uma nova mensagem é enviada para o Telegram, com o status novo, avisando aos administradores que houve uma mudança na temperatura da rede.

Estes valores de temperatura podem ser configurados para qualquer valor. Neste exemplo usamos os intervalos entre 15 e 30 °C, a função também pode ser configurada para verificar valores de umidade.

A função *formata mensagem*, mostrada na figura 21, processa a mensagem para ser enviada ao Telegram, separando as mensagens em status de sensor *online* ou *offline* e mensagens de dados coletadas pelo sensor.

Figura 21 - Função de formatação do payload para o Telegram

```
var status = '';
if (msg.payload == ("Sensor 1 online!" || "Sensor 1 offline!"))
{
    status = JSON.stringify(msg.payload);
}
else{
    status = 'temperatura: ' + msg.payload.temperatura +'°C, umidade: '+msg.payload.umidade+'% ... '+msg.payload.status
}
msg.payload = {};
msg.payload.chatId = 6739442109;
msg.payload.type = 'message';
msg.payload.content = status;
return msg;
```

Fonte: Elaborado pelo autor

Esta função formata a mensagem com as informações necessárias para o envio ao Telegram, com as informações de tipo de mensagem ‘message’ e *ChatID* da conversa com o *chatbot* Sensor01.

4.6 Dashboard Node-RED

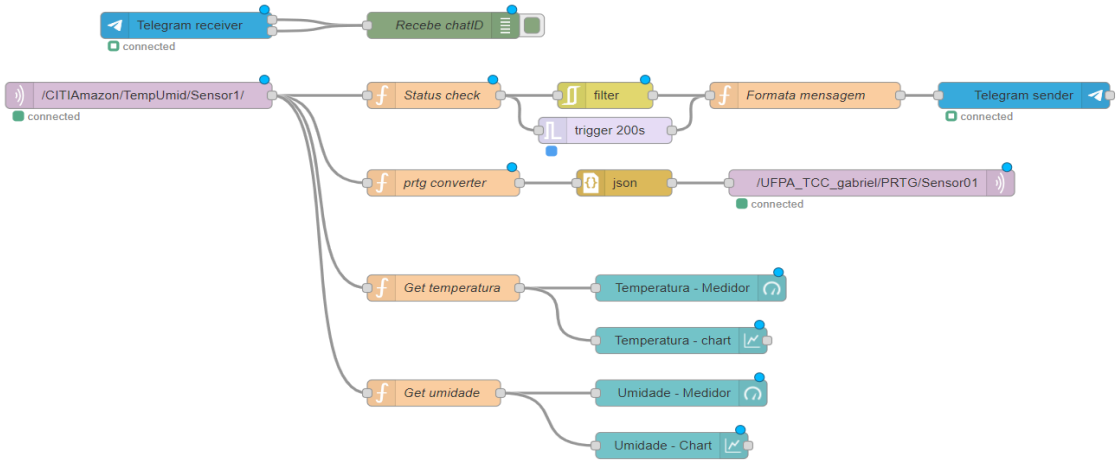
O *dashboard* Node-RED, visto na figura é construído usando os nós *dashboard* do Node-RED, os usados aqui serão o nó *gauge* e o nó *chart*. O nó *gauge* mostra a temperatura e umidade atual do sensor, o nó *chart* mostra dados ao longo do tempo, de forma configurável.

O nó gauge pode ser configurado de forma a ter um valor máximo e mínimo na opção *range*, a unidade mostrada é configurada em *unit*, o *label* mostra o nome do sensor. O nó pode ser configurado em cores com gradientes de forma opcional com *colour gradient* e *sectors*.

O nó *chart* é configurável por meio das opções *label* que contém seu nome, *type*, com o tipo de gráfico mostrado, um eixo X que contém o tempo de gravação dos dados, um eixo Y que contém valores mínimos e máximos de temperatura e umidade.

Depois de configurado por completo o Node-RED estará configurado para fazer 3 tarefas: Alarmes por meio de um *chatbot* do Telegram, Enviar dados para o PRTG e exibir dados em um *dashboard* próprio. Na figura 22 podemos ver o fluxo Node-RED completo.

Figura 22 - Fluxo Node-RED completo



Fonte: elaborado pelo autor

4.7 Software de monitoramento de redes PRTG

O PRTG foi o software escolhido para a visualização dos dados de uma forma adicional ao *dashboard* do Node-RED. Para a visualização dos dados coletados pelos sensores no trabalho, este software foi instalado em uma máquina Windows 10 que possui conexão com o broker MQTT por meio do Node-RED, pelos nós *mqtt in* e *mqtt out*. Para realizar a conexão o sensor é configurado como *MQTT Subscribe Custom Sensor* (PAESSLER, 2024), como visto nas figuras 22 e 23. Este template de sensor, fornecido no PRTG é capaz de receber até 10 valores em formato de JSON. O formato deve ser o visto anteriormente quando mostramos a conversão em *formata mensagem* na seção 4.4.

Figura 22 -Dashboard de sensores PRTG

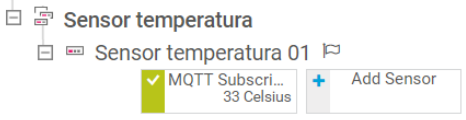
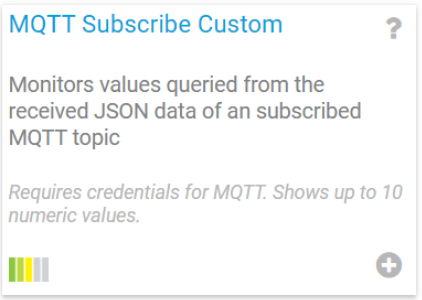


Figura 23 - Adicionando sensores MQTT



Fonte: elaborado pelo autor

Para configurar o sensor, iremos inserir um nome, canais de dados e unidades de medida para os dados que forem coletados. Após configurado o sensor MQTT podemos decidir quais dados serão mostrados nos gráficos do *dashboard* PRTG. A configuração pode ser vista nas figuras 24 e 25.

Figura 24 - Configuração do Sensor MQTT

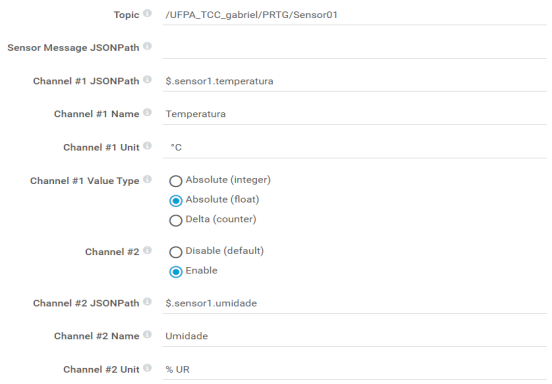
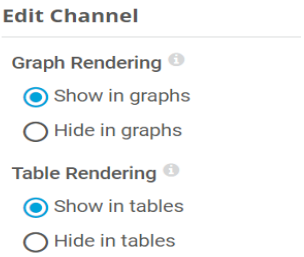


Figura 25 - Mostrar ou esconder dados no dashboard PRTG



Fonte: Elaborado pelo autor

Depois de configuradas as opções, podemos configurar alarmes para ocorrências. O PRTG tem acesso a sistemas de alarme que podem ser configurados por meio de triggers, estes *triggers* respondem a eventuais quedas na rede e problemas com o monitoramento dos dados. Neste trabalho não foram configurados *triggers* por já estarmos com uma solução via Telegram configurada a partir do Node-RED. Outra razão é a necessidade de uma conta para acesso a API de e-mails que seriam usados para notificações *push*.

5. Resultados

Nesta seção iremos mostrar os dados coletados pelos métodos descritos anteriormente. Os dados foram coletados a partir do acesso do tópico hosteado no EMQX público "**/CITIAmazon/TempUmid/Sensor1/"**, e repassados para o PRTG pelo publish "**/UFPA_TCC_gabriel/PRTG/Sensor01".** Os dados serão mostrados para ilustrar as diferenças nas capacidades das ferramentas para mostrar estes dados, começando pela ferramenta mais simples que é o *dashboard* Node-RED.

Os dados mostrados aqui foram coletados em um ambiente de servidor, no CTIC da UFPA, com o intuito de ajudar no monitoramento da temperatura ambiente para a operação segura de uma sala de servidor. Os alarmes escolhidos para serem enviados automaticamente foram baseados em temperaturas que são consideradas seguras para a operação de uma sala de servidor.

Segundo a Cisco (2016), a temperatura ambiente para operação segura de uma sala de servidores é entre 10 e 35 ºC e a umidade deve ficar entre 10 e 90% de umidade relativa. Estes valores de temperatura e umidade irão manter os computadores funcionando de forma a não superaquecer nem gerar condensação de umidade por frio.

Os dados coletados mostrados no *dashboard* são mostrados ao longo de 1 hora de coleta de dados, conforme configurado em *Umidade - chart* e *Temperatura - chart.* A temperatura neste caso ultrapassou 30°C, visto na figura 26 . O valor configurado para o alarme, gerando a mensagem vista na figura 27.

Com esta mensagem um administrador poderia tomar as medidas cautelares para controlar a temperatura ambiente do servidor e assim evitar problemas para a operação.

Também foi realizado o monitoramento dos dados por meio do PRTG, com o *dashboard* ilustrado na figura 28. Esta ferramenta é mais desenvolvida que o *dashboard* do Node-RED. o PRTG já fornece por padrão uma grande capacidade de armazenamento de dados ao longo de um tempo, assim como a capacidade de dar "zoom" nos dados de um tempo mais curto.

Figura 26 - Dashboard Node-RED

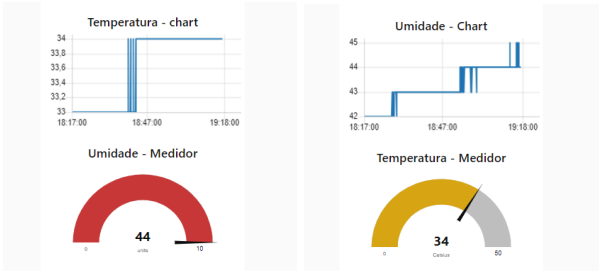
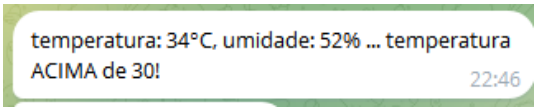
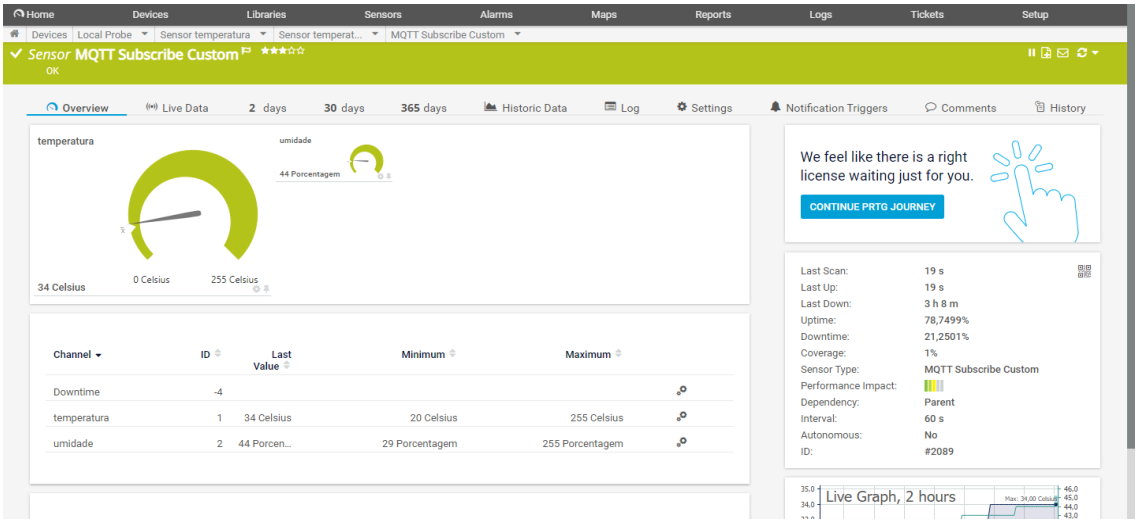


Figura 27 - Alarme de temperatura



Fonte: elaborado pelo autor

Figura 28 - Dashboard PRTG, MQTT Subscribe Custom

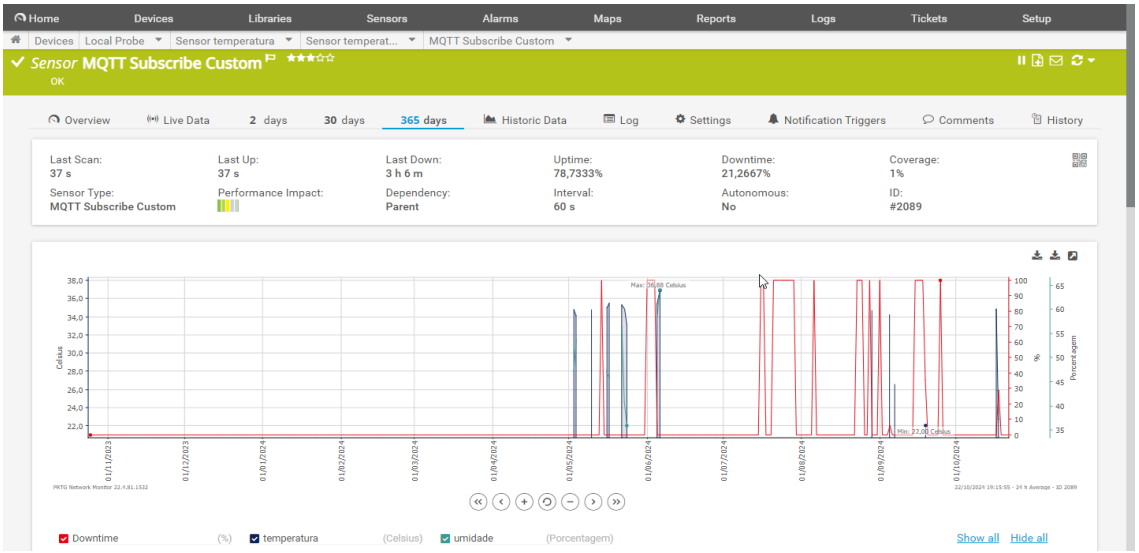


Fonte: Elaborado pelo autor

O *dashboard* PRTG mostra opções de configuração e gráficos que mostram os dados em tempo real e estatísticas históricas da coleta de dados ao longo do tempo.

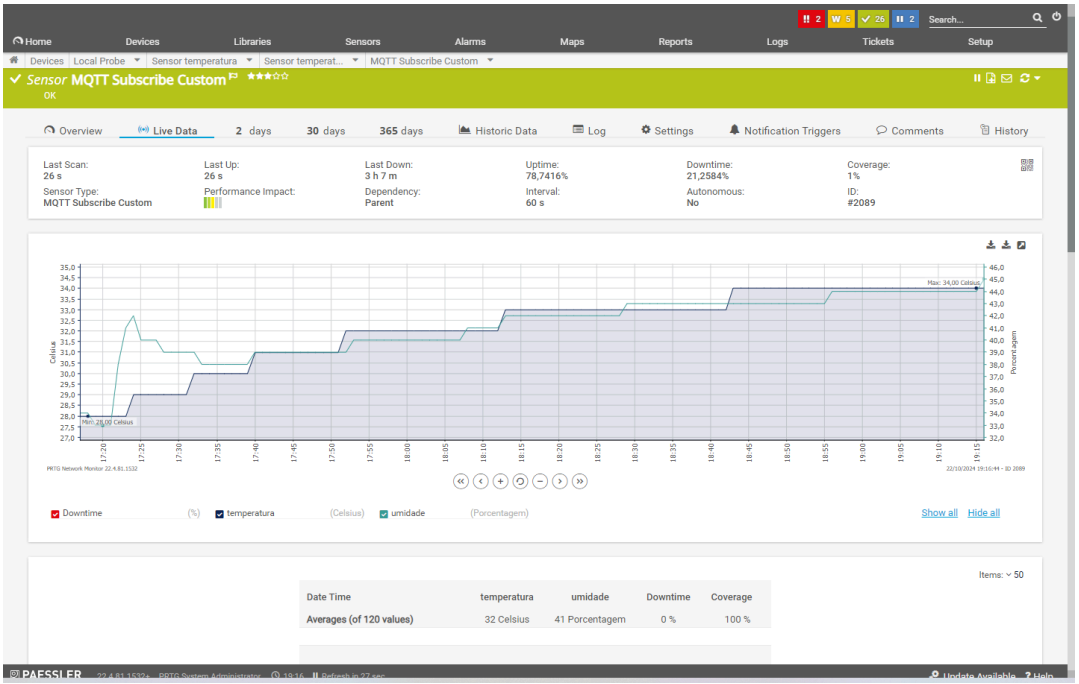
A figura 29 mostra a coleta de dados ao longo de 365 dias, com várias informações extras como: downtime, tempo de operação, cobertura do tempo e intervalo na coleta dos dados. Em toda a coleta de dados nenhuma queda na coleta de dados se deu devido ao sensor desconectado, mostrando a sua confiabilidade. O teste no ambiente do CTIC durou apenas as duas semanas finais desta coleta de dados, portanto no gráfico de 365 dias apenas o final dele pode ser considerado como parte do teste. Na opção de *Live Data*, figura 30, mostra os dados ao vivo com um zoom de 2 horas, ou seja a variação de temperatura e umidade dentro de uma faixa de 2 horas. Estes dados foram coletados do data center CTIC UFPA.

Figura 29 - Coleta de dados PRTG ao longo de 365 dias



Fonte: Elaborado pelo autor

Figura 30 - Live Data PRTG

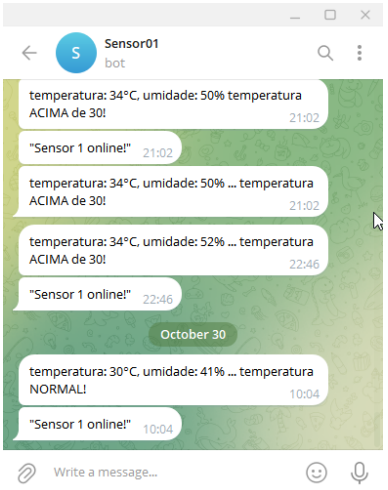


Fonte: Elaborado pelo autor

O alarme configurado para o envio de mensagens de alerta de temperatura e downtime do servidor envia mensagens via um *chatbot* do Telegram configurado no Node-RED, mostrado na figura 31.

Esta coleta de dados foi feita a partir de um único sensor de temperatura DHT11 em uma sala de servidor do CTIC UFPA. É possível a expansão do monitoramento por meio de mais sensores e seu cadastro de forma similar no *broker* MQTT.

Figura 31 - Chatbot do Telegram “Sensor01” enviando mensagens de alarme para o administrador



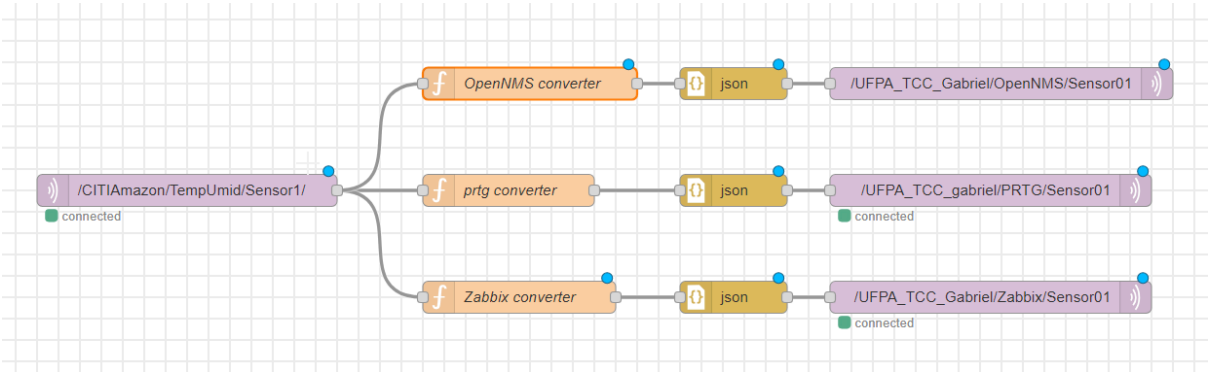
Fonte: Elaborado pelo autor

6. Conclusão

Em suma, este trabalho foi um estudo sobre como redes de computadores foram desenvolvidas e como podem ser monitoradas, com um estudo de caso em um sensor de temperatura em uma board ESP8266, ligado a uma rede Wi-Fi, e por meio dela se conectando a um *broker* via protocolo MQTT. Este *broker* transmite os dados para qualquer computador que se conecte ao tópico com a informação sendo postada. Os softwares ligados ao MQTT podem ser diversos, cada um com o seu método de acesso aos dados. Neste trabalho foi usado o Node-RED para servir de um hub de conexão para vários tipos de softwares, como foi discutido, o PRTG e Telegram foram usados para estudar como este hub pode funcionar para a distribuição destes dados coletados.

Embora não tenha sido implementada por razões discutidas mais a frente, é possível construir um hub de softwares de gerenciamento de rede conforme mostrado na figura 32.

Figura 32 - Hub Node-RED



Fonte: Elaborado pelo autor

Com este fluxo seria possível enviar dados de um único sensor para vários sistemas de gerenciamento de software, tornando a coleta de dados mais flexível para o uso em qualquer ambiente de trabalho.

Porém houve problemas de implementação com o Zabbix, devido a falta de documentação clara sobre como realizar a conexão do mesmo com um sensor via o protocolo MQTT. Para trabalhos futuros este problema poderia ser resolvido para completar a ideia do trabalho, que seria um hub de retransmissão de dados coletados por protocolo MQTT para vários softwares de gerenciamento de redes.

O custo do sensor de temperatura ambiente usado neste trabalho é pequeno, apenas R$48,00 quando somados os custos do sensor DHT11 e a placa ESP 8266, que se justifica quando comparado ao alto custo possível em um problema de controle de temperatura em sala de *data center*.

Com o trabalho atual, a sua contribuição é a possível implementação de sensores de coleta de dados IoT via MQTT em qualquer ambiente que precise ser monitorado. Estes sensores ativam alarmes, que enviam mensagens instantâneas, caso a temperatura ou umidade passem de um certo valor configurado, o que vai permitir aos administradores uma resposta rápida ao problema.

6. Referências

SINHA, Satyajit, Number of connected IoT devices growing 13% to 18.8 billion globally, **IoT Analytics**, Disponível em: <https://iot-analytics.com/number-connected-iot-devices/>. Acesso em 25/10/2024.

IBM, TCP/IP Protocols, **AIX 7.3 Documentation**, disponível em: <https://www.ibm.com/docs-/en/aix/7.1?topic=proocol-tcpip-protocols>. Acesso em 28/10/2024.

KUROSE, James F, KEITH W. Ross.(2013) **"Redes de computadores e a Internet: uma abordagem top-down 6ª ed."**, tradução Daniel Vieira, revisão técnica Wagner Luiz Zucchi. Pearson Education do Brasil. São Paulo.

HUAWEI, **Technical Guides: What is SNMP?**. Disponível em: <https://support.huawei.com/enterprise/en/doc/EDOC1100086963>. Acesso em: 25/10/2024.

STEVE, Beginners Guide To The MQTT Protocol (2018), **Steve's Internet guide**, disponível em: <http://www.steves-internet-guide.com/mqtt/>. Acesso em 25/10/2024.

OASIS, MQTT Version 3.1.1 (2014), **Oasis Standard**. Disponível em: <https://docs.oasis-open.org/mqtt/mqtt/v3.1.1/os/mqtt-v3.1.1-os.html>. Acesso em 25/10/2024.

HIVEMQ, **MQTT Essentials**. Disponível em: <https://www.hivemq.com/mqtt/>. Acesso em 25/10/2024.

EMQX, **Free Public MQTT broker**. Disponível em: <https://www.emqx.com/en/mqtt/public- -mqtt5-broker, acessado em 26/10/2024.

EMQX, **EMQX Public Broker Dashboard**. Disponível em:

<https://www.emqx.com/en/mqtt-dashboard#/dashboard/overview>. Acesso em 26/10/2024.

EMQX Team, Free MQTT Broker: Exploring Options and Choosing the Right Solution (2024), **EMQX blog**. Disponível em : https://www.emqx.com/en/blog/free-mqtt-broker. Acesso em 08/11/2024.

HORIZON, **OpenNMS.** Disponível em: <https://www.opennms.com/horizon/>. Acesso em 25/10/2024.

ZABBIX, **Zabbix Manual**. Disponível em: <https://www.zabbix.com/documentation/6.0/en/manual>. Acesso em 25/10/2024.

PAESSLER,(2024), **PRTG Manual,** disponível em: <https://manuals.paessler.com/prtgmanual.pdf>. Acesso em 26/10/2024.

PAESSLER, PRTG Manual: Subscribe custom sensor, **PRTG Manual.** Disponível em: <https://www.paessler.com/manuals/prtg/mqtt_subscribe_custom_sensor>. Acesso em 26/10/2024.

OpenJS Foundation & Contributors, **Node-RED**. Disponível em: <https://nodered.org>. Acesso em 26/10/2024.

WIND, K.H. et al. node-red-contrib-telegrambot, **flows.nodered.org**. Disponível em: <https://github.com/windkh/node-red-contrib-telegrambot>. Acesso em 26/10/2024.

MOUSER, **DHT11 humidity & temperature sensor**. Disponível em: <https://www.mouser.com/datasheet/2/758/DHT11-Technical-Data-Sheet-Translated-Version-1143054.pdf>. Acesso em 26/10/2024.

Espressif IoT Team,(2015), **ESP8266EX Hardware User Guide**. Disponível em: <https://s3-ap-northeast-1.amazonaws.com/cerevo-share/ESP8266/ESP8266EX_Hardware_User_Guide_EN_v1.1.pdf>. Acesso em 26/10/2024.

WEMOS,(2021), **LOLIN D1 mini**. Disponível em: <https://www.wemos.cc/en/latest/d1/d1_mini.html>. Acesso em 26/10/2024.

SAMUEL F. (2023) **“Infraestrutura IoT para Monitoramento de Conforto Ambiental”**, Universidade Federal do Pará. Belém.

Cisco, Preparing the Site (2016), **Cisco UCS Site Preparation Guide**. Disponível em: <https://www.cisco.com/c/en/us/td/docs/unified_computing/ucs/hw/site-prep-guide/ucs_site_prep/ucs_siteprep_chapter_010.html#:~:text=The%20Cisco%20UCS%20equipment%20should,F%20(35%C2%B0C)>. Acesso em: 26/10/2024.

Hewlett-Packard, snmp-agent packet max-size (2016), **Documentação Hewlett-Packard**. Disponível em:

<https://support.hpe.com/techhub/eginfolib/networking/docs/switches/3100-48/5998-7644r_nmm_cr/content/442453782.htm>. Acesso em 08/11/2024.

R. A. Atmoko et al. IoT real time data acquisition using MQTT protocol. In: **International Conference on Physical Instrumentation and Advanced Materials**, 853 012003, 2017, Indonesia.

Megha Q., B. B. Gupta. Shingo Y. MQTT-driven Remote Temperature Monitoring System for IoT-based Smart Homes, In: **IEEE 8th Global Conference on Consumer Electronics**, 2019, Osaka, Japão.

EMQX Team, Free MQTT Broker: Exploring Options and Choosing the Right Solution (2024), **EMQX blog**. Disponível em : https://www.emqx.com/en/blog/free-mqtt-broker. Acesso em 08/11/2024.