Emulador de *Publisher* MQTT para Redes de Sensores Sem Fio

Edineide das Dores Sousa[1], **Rodolfo Francisco de Oliveira**[2]

[1]Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas
Instituto Federal de São Paulo (IFSP) – Campus Hortolândia, SP – Brasil

[2]Área de Informática - Câmpus Hortolândia - Instituto Federal de São Paulo (IFSP)

edineidedsousa@gmail.com, rodolfo_foliveira@hotmail.com

Abstract. *The concept of the Internet of Things (IoT), refers to the fact that various everyday objects such as bracelets, watches, televisions, cars, among others, have an Internet connection. IoT is formed from the combination of several technologies, among them the Wireless Sensor Networks (WSN). However, there is a need to improve the quality of applications developed for IoT in a faster and more efficient way. And it is in this intention that the present work proposes to emulate the MQTT protocol for an RSSF from the point of view of the generation of data collected. The main goal is to publish this data to an MQTT broker server and make it available transparently for use in initial application testing without the need for real physical equipment, thus gaining more productivity and efficiency. Through this work the possibility of emulation of elements of a WSN and MQTT was verified, as well as the feasibility for future implementations.*

Resumo. *O conceito de Internet das Coisas (IoT), se refere ao fato de diversos objetos do cotidiano como pulseiras, relógios, TVs, carros, entre outros possuírem conexão com a Internet. IoT é formada da combinação de várias tecnologias, dentre elas as Redes de Sensores Sem Fio (RSSF). No entanto, existe a necessidade de aprimoramento da qualidade das aplicações desenvolvidas para IoT de maneira mais rápida e eficiente. E é nesse intuito que o presente trabalho propôe fazer a emulação do protocolo MQTT para uma RSSF do ponto de vista da geração de dados coletados. O objetivo principal é publicar esses dados para um servidor broker MQTT e disponibilizá-los de forma transparente para serem usados nos testes iniciais de aplicações, sem a necessidade de equipamentos físicos reais, ganhando assim mais produtividade e eficiência. Através desse trabalho foi constatada a possibilidade de emulação de elementos de uma RSSF e do MQTT, e ainda a viabilidade para futuras implementações.*

1. Introdução

É notável no cenário atual o surgimento de novos conceitos voltados para a área de tecnologia. Um exemplo disso é a evolução da Internet das Coisas (em inglês: *Internet of Things* – IoT) que se refere à conexão de dispositivos do cotidiano, tais como geladeiras, pulseiras e carros à Internet. Segundo o Cisco Systems, em 2008 já havia mais coisas conectadas à Internet no planeta do que seres humanos. A estimativa é que em 2020,

50 bilhões de coisas devem estar conectadas por meio das Redes de Sensores Sem Fio (RSSF) [SAS Insights 2018].

As RSSF são uma das tecnologias implementadas na IoT que consistem em coletar dados do ambiente em tempo real através de seus componentes (Nó Sensores e Sensores), e disponibilizá-los para aplicações diversas [Souza 2017].

Outra tecnologia que vem ganhando destaque no cenário de IoT é o protocolo *Message Queue Telemetry Transport* (MQTT), desenvolvido pela IBM no ano de 1999 e redescoberto atualmente para atender às demandas de IoT. O MQTT tem se tornado um padrão para comunicações relacionadas a IoT, devido o fato dele ser um protocolo leve e fléxivel que permite a sua implementação em redes com alta restrição e banda limitada, como no caso das RSSF [IBM 2017].

Diante disso, há também a necessidade de aprimoramento da qualidade das aplicações desenvolvidas para IoT de maneira mais rápida e eficiente. No entanto, isso muitas vezes se torna difícil para os desenvolvedores devido à ausência de dispositivos físicos (*hardwares*) tais como Nó Sensores, Sensores, Atuadores e outros equipamentos para a implementação de RSSF. Tais equipamentos podem não estar disponíveis na fase inicial de testes devido ao desenvolvimento paralelo dos mesmos e/ou alto custo, seja para a aquisição de protótipos ou produtos finais. É nesse cenário que os métodos de emulação de ferramentas e dispositivos podem ser aplicados.

Emulação é o processo em que ferramentas ou dispositivos reais como por exemplo, Sensores e/ou Atuadores de redes, são reproduzidos de forma virtualizada a ponto de realizar a mesma função que ferramentas ou dispositivos reais são capazes de realizar [Silva 2017]. As ferramentas emuladas possuêm um alto nível de abstração, sendo assim é possível que os resultados possam ser diferentes dos resultados obtidos do mundo real, mas ainda tais ferramentas são úteis para serem usadas nos testes das aplicações de redes [Font et al. 2011]. Além disso, é importante que a interação da ferramenta de emulação com aplicações reais seja realizada de forma transparente, de forma a evitar alterações e adaptações nos códigos das mesmas.

É nesse contexto que o presente trabalho apresenta o protótipo de um sistema denominado Emulador de *Publisher* MQTT, propondo fazer a emulação de uma RSSF do ponto de vista da geração de dados coletados, e publicar os mesmos para um *broker* MQTT, disponibilizando os mesmos para aplicações *subscribers* (assinantes) interessadas.

O trabalho é organizado da seguinte maneira: no capítulo 2, será apresentado o conceito de IoT, seus componentes e suas aplicações. No capítulo 3, será abordado o tema de RSSF com seus principais componentes. No capítulo 4, será abordado o Protocolo MQQT e sua utilização dentro da IoT. No capítulo 5, serão apresentados alguns trabalhos correlatos como sistemas similares. No capítulo 6, será apresentada a arquitetura do sistema bem como os respectivos diagramas. No capítulo 7, será mostrado o processo de desenvolvimento da plataforma bem como as ferramentas, tecnologias e o Sistema de Gerenciamento de Banco de Dados (SGBD) utilizado. No capítulo 8, serão mostrados os testes e resultados encontrados com demonstração do gráfico do teste de performance e o resultado do teste de sistema realizado e, por último, no capítulo 9 será apresentada a conclusão final e perspectivas de trabalhos futuros.

2. Internet das Coisas

Internet das Coisas (IoT) é o nome dado atualmente ao conceito relacionado a conexão de diversos objetos do cotidiano à Internet [Souza 2017]. Há poucos anos atrás, o equipamento que possuía capacidade de se conectar à Internet era o que é conhecido tradicionalmente como computador pessoal (PC – *Personal Computer*), mas esse cenário mudou. Além dos dispositivos móveis (*smartphones* e *tablets*, por exemplo), uma série de objetos como geladeiras, relógios, TVs, pulseiras, carros, casas e plantas possuem atualmente a possibilidade de conversar entre si, e entre outros sistemas computacionais através da Internet [Loureiro and Nogueira 2016]. E além disso, tais objetos realizam diversas funções especiais, como por exemplo, uma determinada geladeira detecta que faltam alimentos e avisa o proprietário que é hora de fazer novas compras; a pulseira que detecta comportamentos corporais e envia os dados para um servidor na nuvem, para que estes possam ser lidos por interessados, como por exemplo um médico. De acordo com as estimativas, até 2020, de 50 a 100 bilhões de dispositivos estarão conectados a internet, e a previsão é de que a IoT movimente em 2020, 1,534 bilhões [Oliveira 2016].

No entanto, IoT não é uma tecnologia, e sim um conceito formado da combinação de várias tecnologias. "Há, essencialmente, três componentes que precisam ser combinados para termos uma aplicação de IoT: dispositivos, redes de comunicação e sistemas de controle"[Info Wester 2017]. Os dispositivos físicos que implementam o conceito de IoT são conhecidos como RSSF. Já os sistemas de controle são implementados em *softwares*, nas mais diversas plataformas, tais como *mobile*, aplicações em *Cloud* (nuvem), sistemas de gerenciamento, etc. As implementações tecnológicas do âmbito de IoT pode ser aplicadas em várias áreas, por exemplo na área da sáude, agropecuária, energia elétrica, casas, cidades entre outras [Oliveira 2016].

3. Redes de Sensores Sem Fio

Conforme já mencionado, as Redes de Sensores Sem Fio (RSSF) é uma das principais tecnologias que implementam a IoT, e ela é composta por estações de rádios infra-estruturada para realizar a comunicação entre os elementos da rede. A Figura 1 exemplifica uma RSSF.

Figura 1. Representação da Redes de Senssores Sem Fio
Fonte: Adaptada de
https://www.inovacaotecnologica.com.br/noticias/noticia.php?artigo=kit-codigo-aberto-implementacao-internet-coisas&id=010150131028#.W8UpqWhKjIU

Uma RSSF tem como função coletar e disponibilizar dados das grandezas do meio ambiente, tais como temperatura, pressão, luz, umidade etc. Estes dados serão aprovei-

tados por uma ou mais aplicações finais localizadas, por exemplo, em uma rede *Ethernet* [Souza 2017].

As RSSF se divide em homogêneas e heterogêneas. Isso se dá pelos tipos e funcionalidades dos elementos utilizados nas aplicações. Caso os elementos sejam do mesmo tipo e características físicas, se constitue uma RSSF homogênea. Senão, é uma rede heterogênea [Loureiro and Nogueira 2016].

3.1. Componentes de Redes de Sensores Sem Fio

A RSSF é composta de três principais elementos: Nó Sensor (NS), Nó Sorvedouro e Gateway. O NS tem a função de coletar os dados (grandezas) do meio ambiente e enviar para outro NS, nesse caso é denominado roteamento, ou enviar para um Nó Sorvedouro. O Nó Sorvedouro, através da conexão com *gateway*, envia os dados coletados para uma aplicação localizada na rede externa, como por exemplo, uma rede TCP/IP [Oliveira 2016]. A Figura 2 exemplifica a arquitetura de um NS.

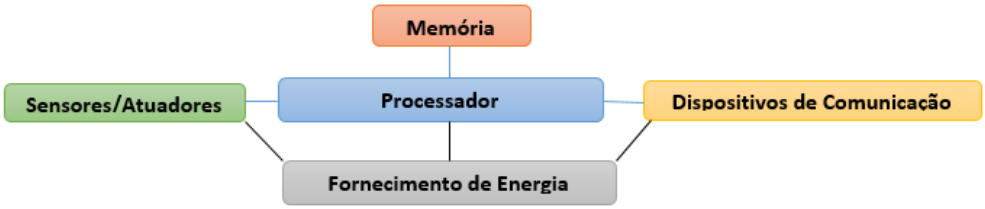


Figura 2. Arquitetura de um Nó Sensor
Fonte: Adaptada de [Oliveira 2016]

A seguir será descrito cada um dos elementos que compõem o NS.

O processador é o componente responsável por processar os dados, atuar sobre os atuadores, sensores e dispositivos de comunicação, e ainda por escrever e recuperar os dados na memória do NS. Geralmente os processadores consumem pouca energia devido ao fato de terem poder computacional baixo [Oliveira 2016].

O fornecimento de energia no NS é fornecido através de baterias, painéis solares ou fontes de energia fixa, como tomadas. Dentre elas, as mais utilizadas são as baterias e a escolha delas depende do volume e temperatura que desejam ser alcançados, podendo ser do tipo linear simples, lítio NR e lítio Coin Cell. [Loureiro and Nogueira 2016].

Os dispositivos de comunicação utilizados no NS são transceptor de Rádio Frequência, que possuem capacidade de transmitir e receber dados [Cano 2016]. Outras opções menos utilizadas são o infravermelho e micro-ondas.

A memória do NS tem a responsabilidade de armazenar os dados coletados pelos sensores [Oliveira 2016]. Há três tipos de memórias que podem ser usadas: Memórias de Armazenamento de dados persiStentes (EPROM- *Electrically Erasable Programmable Read Only Memory* e a memória *Flash*); Armazenamento de programas e dados constantes (memória *ROM* e a memória *Flash*); e as memórias de acesso rápido (memória *RAM*) [Cano 2016].

Os sensores são dispositivos físicos com capacidade de ler grandezas do meio ambiente como temperatura, calor entre outras e convertê-las através de um conversor analógico-digital para o meio virtual, nas quais poderão ser utilizadas. Eles são classificados em três categorias: Sensores Passivos Omnidirecionais, Sensores Passivos de Feixe Estreito e Sensores Ativos. Dentro da primeira categoria são englobados os sensores que captam as grandezas do ambiente sem manipulá-lo, por exemplo, sensor de luz e umidade. Na segunda categoria estão os sensores que também captam as grandezas do ambiente sem manipulá-lo, porém a direção em que o mesmo está localizado influencia nos resultados finais, por exemplo, um sensor de câmera. Na última categoria, ficam os sensores que captam uma série de eventos no ambiente em que está localizado, sendo esses eventos ocasionados por ondas de choques que causam pequenas explosões, por exemplo, sensor de radar [Karl and Willig 2006]. A escolha deles e da quantidade dependem do objetivo da aplicação [Loureiro and Nogueira 2016]. A Figura 3 apresenta alguns exemplos de sensores que podem ser utilizados.

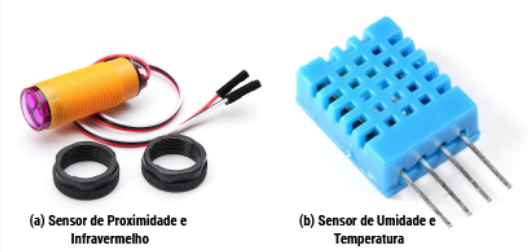


Figura 3. Exemplos de alguns tipos de sensores
Fonte: https://www.filipeflop.com/categoria/sensores/

Os atuadores assim como os sensores, são de diversos tipos. Eles possuem a função de alterar valores do ambiente para corrigir falhas e controle do objeto monitorado. Quando o NS possui atuador conectado em si, ele fica responsável por abrir ou fechar um *switch* ou relé para permitir a passagem do valor captado pelo atuador e acionar o mecanismo desejado, como por exemplo, um motor ou uma lâmpada [Karl and Willig 2006].

Esse trabalho foca apenas na emulação do elemento Nó Sensor e os Sensores aclopados ao mesmo como, por exemplo, sensores de Temperatura e Umidade.

4. Protocolo MQTT

O protocolo *Message Queue Telemetry Transport* (MQTT) é um protocolo de rede baseado em TCP/IP, desenvolvido inicialmente pela IBM no final dos anos 90 com o objetivo de vincular Sensores em *pipelines* de petróleo a satélites [IBM 2017]. Ele é caracterizado no modelo *publish/subcribe*, no qual alguma aplicação publica mensagens em algum lugar, por exemplo, um servidor e outra(s) aplicação(ões) subscreve e recebe tais mensagens, não havendo assim comunicação direta entre a aplicação que envia e entre a aplicação que recebe as mensagens [Correa et al. 2016].

Conforme é apresentado na Figura 4, o MQTT é dividido nas instâncias: *publisher*, servidor *broker* e *subscriber*. O *publisher* é qualquer aplicação que comunica com o servidor *broker*, por exemplo, sensor de IoT ou um aplicativo que envia dados em forma de mensagem, sendo que essas mensagens são organizadas por tópicos. O servidor recebe esse tópico e o roteia para o *subscriber*.

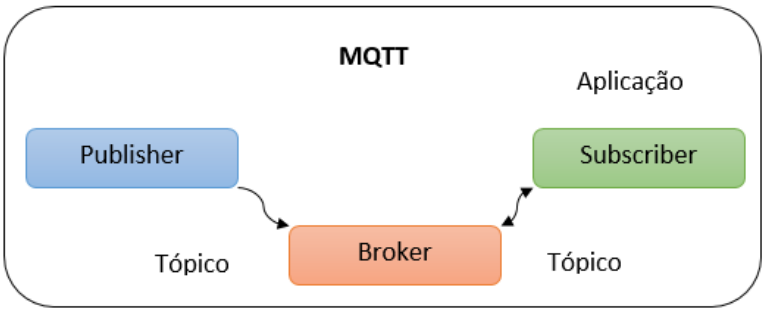


Figura 4. Representação da comunicação do protocolo MQTT
Fonte: Adaptada de [IBM 2017]

MQTT é um protocolo leve, simples e assim como outros protocolos de mensagens, ele desacopla o publicador e o consumidor de dados e ainda permite sua implementação em aplicações restringidas e serviços de IoT, nos quais as redes possuem alta latência e largura de banda limitada [Correa et al. 2016], [IBM 2017].

5. Trabalhos Correlatos

Em [Correa et al. 2016] é apresentada a utilização do protocolo de comunicação MQTT na emulação de aplicações em ambientes industriais. Nesse trabalho foi utilizado aplicações desenvolvidas na linguagem de programação *Python* junto com o módulo *paho-mqtt* para realizarem comunicação entre si, conforme ocorre nas comunicações IoT. Essas aplicações emulam sensores/atuadores presentes em indústrias e possuem a função de mandar informações para o *broker* MQTT. Tais informações são coletadas em intervalos regulares e são alocadas em tópicos no *broker* do MQTT. O *broker* escolhido para implementar esse trabalho foi o *Mosquitto*. E além disso, foi implementada uma pequena aplicação de teste para monitorar todas as atividades presentes no broker, como tempo online, número de clientes conectados, quantidade de bytes recebidos e enviados, entre outros.

Em [Font et al. 2011] é apresentada uma plataforma de testes na qual os elementos do mundo real (Nó Sensores e Nó Sorvedouro) de uma RSSF podem ser substituídos por elementos simulados/emulados de forma transparente para o aplicativo. Ela permitirá a comunicação entre os elementos da RSSF e as ferramentas de simulação. Nessa integração o Nó Sorvedouro é executado em uma plataforma física real, sendo um computador pessoal x86 padrão e a comunicação entre ele e a RSSF é feita através de um canal emulado. Apesar da complexidade e funcionalidade desse projeto, ele necessita de uma plataforma de física de *hardware* poderosa e suficiente para executar simultaneamente o *software* do coletor e a simulação de rede.

6. Arquitetura do Sistema

O sistema Emulador de *Publisher* MQTT é dividido nos seguintes módulos: Módulo Banco de Dados, Módulo Gerador de Dados, Módulo *Publisher* MQTT e Módulo Interface *Web*, conforme é apresentado na Figura 5.

6.1. Módulo Banco de Dados

O Módulo Banco de Dados é o responsável por o armazenamento dos dados do sistema. Esses dados chegam até ele através do Módulo Interface *Web*, no qual são cadastrados os

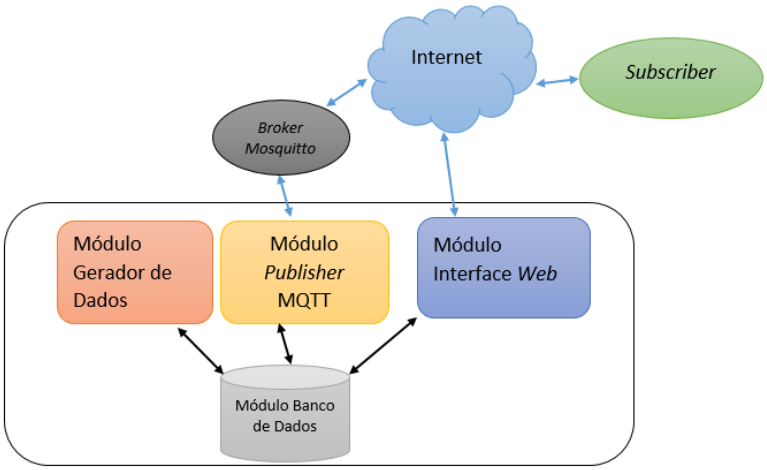


Figura 5. Representação da Arquitetura do sistema

dados dos Sensores e Nó Sensores que serão emulados. Na Figura 6 é mostrado o Modelo Entidade-Relacionamento (MER), que consiste em representar os objetos de dados e seus relacionamentos [Sommerville 2011], do banco de dados em questão. Ele possui três entidades: sensor, dados-sensor e nósensor.

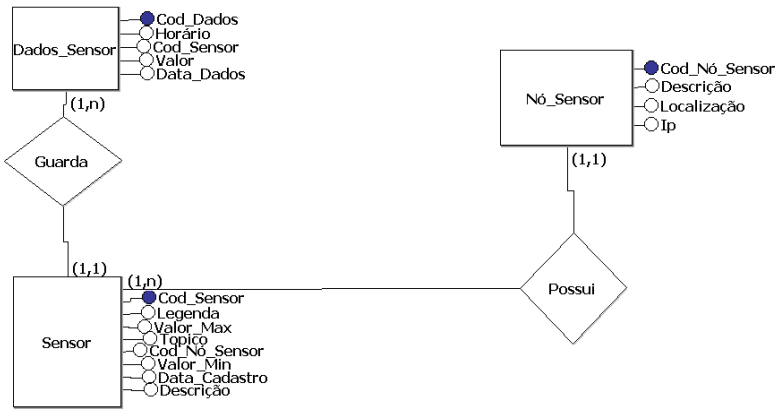


Figura 6. Modelo Entidade-Relacionamento do banco de dados

A entidade “sensor” tem como função armazenar os dados dos Sensores aclopados aos Nó Sensores. A mesma é composta de sete atributos, sendo eles os atributos “codsensor” e “codnosensor” ambos tipo inteiro, que respectivamente, é a chave primária dessa entidade e a chave estrangeira que se relacionará com a entidade nosensor. Os atributos “descrição”, “legenda” e “topico” são do tipo varchar e respectivamente, representam as informações do tipo do Sensor cadastrado, bem como uma pequena descrição da função do mesmo, e o atributo “topico” é o representam o tópico da mensagem a ser enviada para os *subscribes*. Os atributos “valormin” e “valormax”, são do tipo *float*, e representam os valores mínimos e máximos do sensor em questão. Já o atributo “datacadastro” é do tipo

date e representam a data em que o cadastro de Sensor é realizado.

A entidade "dadossensor" tem como função armazenar os dados gerados pelos Sensores através do processo de emulação. A mesma possui cinco atributos, sendo os atributos "coddados" e "codsensor" ambos do tipo inteiro. O primeiro é a chave primária dessa entidade e o segundo é uma chave estrangeira que corresponde ao relacionamento com a entidade Sensor, indicando qual sensor em questão se origina. Os atributos "data-dados" e "horário" são do tipo date, e representam a data e hora do cadastro dos dados dos Sensores. Já o atributo "valor", é do tipo float e representa o valor do Sensor gerado pela aplicação.

A entidade "nosensor" tem como função armazenar os dados referentes aos Nó Sensores do sistema. A mesma possui quatro atributos, sendo o atributo "codnosensor" do tipo inteiro e é a chave primária dessa entidade. O atributo "ip" é do tipo varchar, e representa o endereço IP do Nó Sensor em questão. Os atributos "descrição" e "localização" são do tipo varchar e respectivamente, representam uma pequena descrição e a localização do Nó Sensor.

6.2. Módulo Gerador de Dados

O Módulo Gerador de Dados é formado por um *script* responsável por selecionar os dados mínimo e máximo dos sensores cadastrados no sistema, e através de uma função gerar dados aletórios a partir deles em um intervalo de tempo determinado. Os dados gerados são armazenados no Módulo Banco de Dados na entidade dados_sensor para serem usados em testes e comparações e também serão publicados no Módulo *Publisher* MQTT.

6.3. Módulo *Publisher* MQTT

O Módulo *Publisher* MQTT é formado por um *script* que ao ser executado possui a função de buscar na entidade "sensor" os dados do Sensor (mínimo e máximo) e o nome do tópico que fica armazenado no campo "topico" e, através da mesma função do Módulo Gerador de Dados é gerado um valor aleatório entre o valor mínimo e máximo toda vez em que ocorre a execução desse *script*. Esse valor aletório gerado é inserido no campo "valor" da entidade "dados-sensor" do mesmo banco de dados e juntamente com o nome do "topico", são enviados através de uma função específica do MQTT que passa como parâmetro tais valores, para o servidor *Broker Mosquitto* e partir desse, serem enviados para o *Subscriber*.

6.4. Módulo Interface *Web*

O Módulo Interface *Web* é aplicação responsável por realizar a interação do usuário final, como cadastrar os dados dos Sensores, Nó Sensores no Módulo Banco de Dados, e exibir tais cadastros permitindo excluí-los ou alterá-los. Esses dados cadastrados serão aproveitados pelos Módulos *Publisher* MQTT e Gerador de Dados para cumprir seus objetivos.

6.5. Caso de Uso

A Figura 7 apresenta o Caso de Uso que descreve a interação do sistema com o ambiente [Sommerville 2011] a partir do Módulo Interface *Web*. Os usuários físicos, tempo e *broker* podem realizar respectivamente, as seguintes atividades:

- Manter Nó Sensor: compreende cadastrar, consultar, alterar e excluir os dados dos Nó Sensores;
- Manter Sensor: compreende cadastrar, consultar, alterar e excluir os dados dos Sensores;
- Gerar Dados: compreende o tempo em que o Módulo Gerar Dados leva para gerar dados aletórios a partir dos valores mínimos e máximos.
- Publicar Dados: compreende o envio de dados para o *Subscriber*.

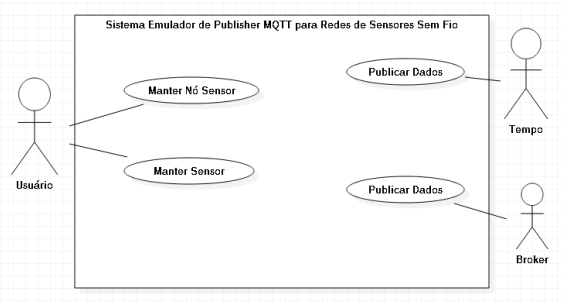


Figura 7. Diagrama de Caso de Uso do sistema

6.6. Diagrama de Classe

Para o desenvolvimento do sistema Emulador *Publisher* MQTT, foi elaborado o Diagrama de Classe que representa os objetos que o sistema irá manipular e as operações que serão aplicadas a esses objetos, bem como seus atributos e métodos [Sommerville 2011]. A Figura 8 representa as classes Sensor, Nó Sensor e Dados_Sensor com seus respectivos atributos e métodos.

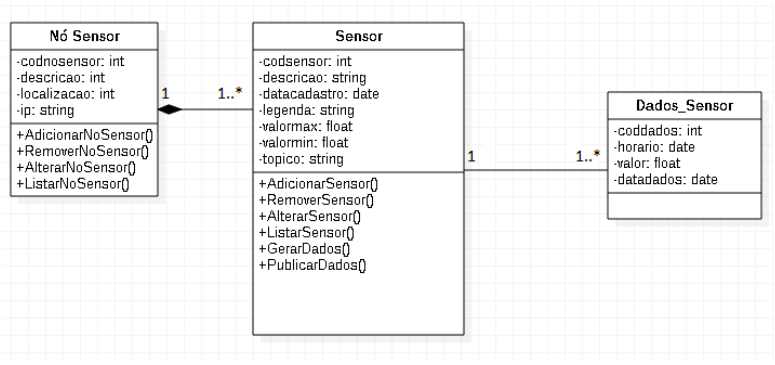


Figura 8. Diagrama de Classe do Emulador

As classes Nó Sensor e Sensor são compostas dos métodos adicionar(), remover(), alterar() e aistar(), pois são métodos de operações básicas do sistema como adicionar, remover, alterar e listar cadastros de Sensores, Nó Sensores no Módulo Banco de Dados. Os métodos GerarDados() e PublicarDados() são exclusivos da classe Sensor. O primeiro é responsável por gerar dados entre o valor máximo e mínimo de cada cadastro de sensor inserido no Módulo Banco de Dados. O segundo é responsável por publicar os dados gerados pelo método GerarDados() e enviar esse valor para o *Subscriber*. A classe Dados_Sensor possui os atributos necessários para armazenar os dados gerados pelo método GerarDados().

7. Desenvolvimento

Para a implementação do sistema Emulador *Publisher* MQTT, foi utilizada uma máquina com as seguintes configurações: Sistema Operacional *Windows* 10 64 bits, Processador AMD C-50 Processor 1.00GHz, e Memória 6,00 GB. Foi instalado o pacote XAMPP sendo ele uma distribuição do servidor *Apache* contendo a linguagem de programação *Hypertext Preprocessor* (PHP) e *MySQL*, sendo de fácil utilização e de instalação nos Sistemas Operacionais *Windows, Linux* e OS X [PHP: Hypertext Preprocessor 2018], [Apache Friends 2018]. Através dele foi implementado o Módulo Banco de Dados do sistema e os *softwares* necessários. Nos tópicos a seguir, serão descritos a implementação de cada módulo do sistema.

7.1. Módulo Banco de Dados

Para a implementação do Módulo Banco de Dados foi utilizado como Sistema de Gerenciamento do Banco de Dados (SGBD) o *MySQL* na versão 4.7.7 [Apache Friends 2018].

7.2. Módulo Gerador de Dados

Para implementação do Módulo Gerador de Dados foi utilizado a linguagem de programação PHP na versão 7.2.2.0, que é uma linguagem *web*, *open source* de uso geral, focada na execução de *scripts* [PHP: Hypertext Preprocessor 2018].

7.3. Módulo *Publisher* MQTT

Para implementação do Módulo *Publisher* MQTT foi utilizada também a linguagem de programação PHP na versão 7.2.2.0 [PHP: Hypertext Preprocessor 2018], focada criação desse *script* e na publicação dos dados para o *Broker* MQTT através da biblioteca *phpMQTT*.

7.4. Módulo Interface *Web*

Para o desenvolvimento do Módulo Interface *Web* também foi utilizada a linguagem de programação PHP. E para a criação da parte gráfica da interface, foram utilizados recursos da linguagem de estilo *Cascading Style Sheets* (CSS3) e *HyperText Markup Language* 5.2 (HTML). Ambas, *open web* e de fácil utilização [MDN web docs 2018].

7.5. *Broker Mosquitto* MQTT

Dentre os diversos *Brokers* MQTT disponíveis para uso, foi escolhido o *Mosquitto*, devido ser leve e adequado para ser utilizado em todos os dispositivos, desde computadores até servidores mais completos [Eclipse Mosquitto 2018].

Para a instalação do *Mosquitto*, foi utilizada uma máquina virtual com o Sistema Operacional *Linux*, e após foi executado o comando *sudo apt-get install mosquitto mosquitto-clients*.

Para um simples teste afim de verificar o envio e o recebimento de uma mensagem usando o *Broker Mosquitto*, em uma janela do terminal foi executado o comando *$ mosquitto -d*, e em seguida para conectar-se ao *Mosquitto* e assinar um tópico, em outra janela do terminal foi executado o comando *$ mosquitto_sub -t "dw/demo"*. E para conectar-se ao *broker* local e depois publicar uma mensagem em um tópico, foi aberto outra janela do terminal e executado o comando *$ mosquitto_pub -t "dw/demo" -m "hello world!"*. No final a janela que executa o *mosquitto_sub* exibiu a mensagem *"hello world!"* [IBM 2017].

8. Testes e Resultados

8.1. Módulo Interface *Web*

Neste subtópico serão apresentados os resultados da construção do Módulo Interface *Web* construída, que fará a interação com os usuários do sistema.

8.1.1. Tela Principal e Menu de Opções

A tela principal do sistema possui uma barra de menu na qual é possível o usuário cadastrar, consultar, alterar e excluir os dados referentes aos Nó Sensores e Sensores. A Figura 9 apresenta essa tela com um resumo sobre o sistema em questão.

Figura 9. Tela principal

8.1.2. Telas de cadastros de Nó Sensores e Sensores

As telas de cadastros de Nó Sensores e Sensores possuem os campos necessários para cada cadastro permitindo salvar os dados e enviá-los para o Módulo Banco de Dados. A Figura 10 (cadastro de Sensores) apresenta um exemplo desse processo.

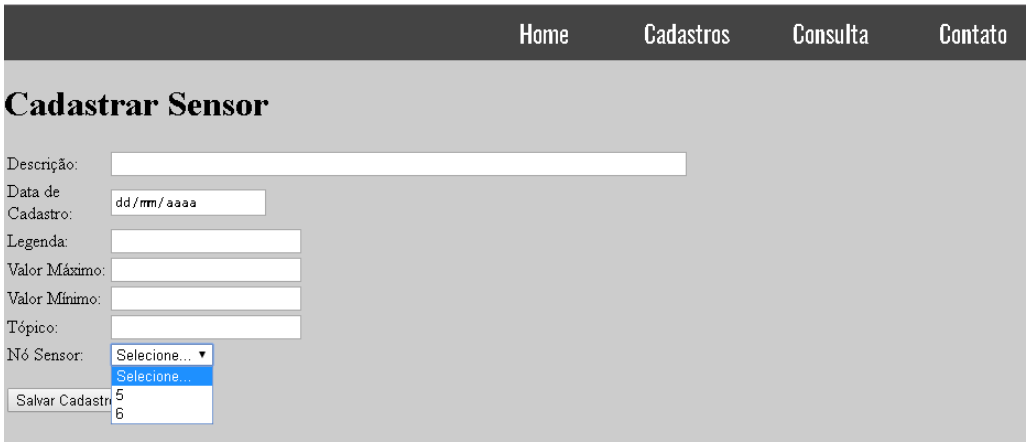


Figura 10. Tela de cadastro de Sensor

8.1.3. Telas de Consulta Nó Sensores e Sensores

As telas de consultas de Nó Sensores e Sensores permite consultar os cadastros salvos no Módulo Banco de Dados do sistema. Nessa tela possui dois botões: Alterar e Excluir, respectivamente para alteração e exclusão dos cadastros desejados. A Figura 11 (consulta de Nó Sensores) apresenta um exemplo desse processo.

Figura 11. Tela de consulta de Nó Sensores

8.1.4. Telas de Alteração de Nó Sensores e Sensores

As telas de alteração de Nó Sensores e Sensores permite alterar os cadastros salvos no Módulo Banco de Dados do sistema. A Figura 12 (alteração de Nó Sensores) apresenta um exemplo desse processo.

Figura 12. Tela de alteração de Nó Sensores

8.2. Performance do Sistema

Afim de obter dados relacionados à performance do sistema (tempo que o Módulo Gerador de Dados leva para gerar dados para N sensores), foi desenvolvido um *script* para

registrar o tempo de processamento desse Módulo. Tal análise é importante, visto que, conforme a quantidade de Nó Sensores e, respectivamente, os Sensores aumentam no sistema, o tempo de processamento dedicado a geração dos dados também tende a aumentar. Após a mensuração do tempo para N sensores, o *script* em questão cadastrava um novo Sensor. Sendo assim, o Módulo Gerador de Dados terá na sua próxima execução N+1 Sensores para gerar valores, aumentando a complexidade do mesmo conforme o decorrer do tempo. A Figura 13 ilustra tal ideia.

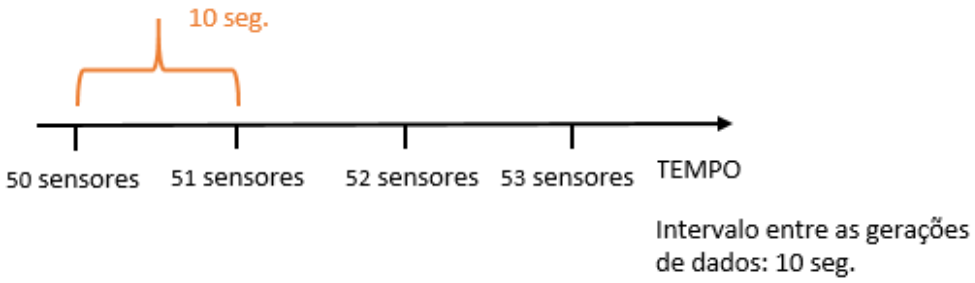


Figura 13. Ilustração do tempo entre as gerações de dados

Todos os Sensores do teste em questão correspondem a um único Nó Sensor. Do ponto de vista do desempenho do Emulador, tal característica não interfere no desempenho do sistema, já que o Módulo Gerador de Dados irá gerar valores para todos os Sensores cadastrados, independente do Nó Sensor ao qual os mesmos estão relacionados.

O teste foi realizado em uma máquina com as seguintes configurações: Sistema Operacional Windows 10 Pro, Memória: 8,00 GB, Processador: AMD FX (tm)-3600 Six-Core Processor 3.50 GHz. O Módulo Gerador de Dados foi executado 5 vezes, sendo o tempo total de execução de cada vez aproximadamente 30 minutos. Durante este período, foram cadastrados (gradativamente) 1000 Sensores. Ao término de cada vez, o tempo de execução dedicado a geração dos dados de cada Sensor cadastrado foi registrado em um banco de dados paralelo ao do Emulador, e depois exportado para uma planilha, limpando assim o banco de dados para a próxima execução. Ao final do teste, foi criado uma média do tempo de processamento armazenado na planilha.

A Figura 14 apresenta os resultados em um gráfico de linha, onde o eixo X representa a quantidade de Sensores ao longo do tempo e o eixo Y o tempo de processamento (em segundos).

Através do gráfico é possível observar que o tempo (em segundos) de processamento do *script* aumenta conforme a quantidade de Sensores existente no banco de dados. Existem oscilações no decorrer desse tempo devido ao processamento da máquina utilizada, caso seja uma máquina com o processamento mais lento, essas oscilações tendem a serem maiores. Com esse teste foi possível obter o tempo de resposta do sistema com aproximadamente 1000 sensores cadastrados no banco de dados, caso a quantidade seja maior, o tempo de processamento tende a aumentar.

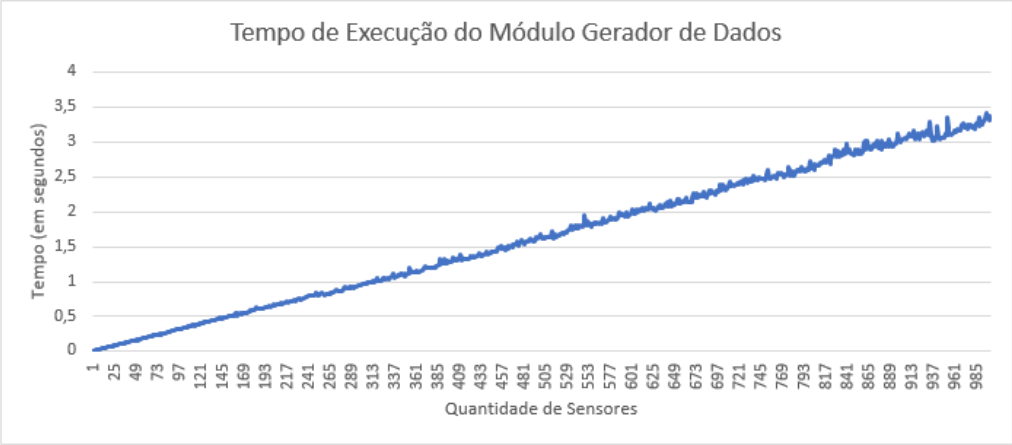


Figura 14. Gráfico gerado a partir da média do tempo de processamento

8.3. Teste de Sistema

Afim de verificar a correta funcionalidade do Módulo *Publisher* MQTT de publicar dados para o servidor *Broker* através do protocolo MQTT, e este enviar para um *Subscriber* (assinante), foi realizado um teste de sistema que consiste em testar o sistema por completo frisando nas partes mais importantes, afim de encontrar comportamentos incorretos ou falta de implementação e corrigi-las [Sommerville 2011].

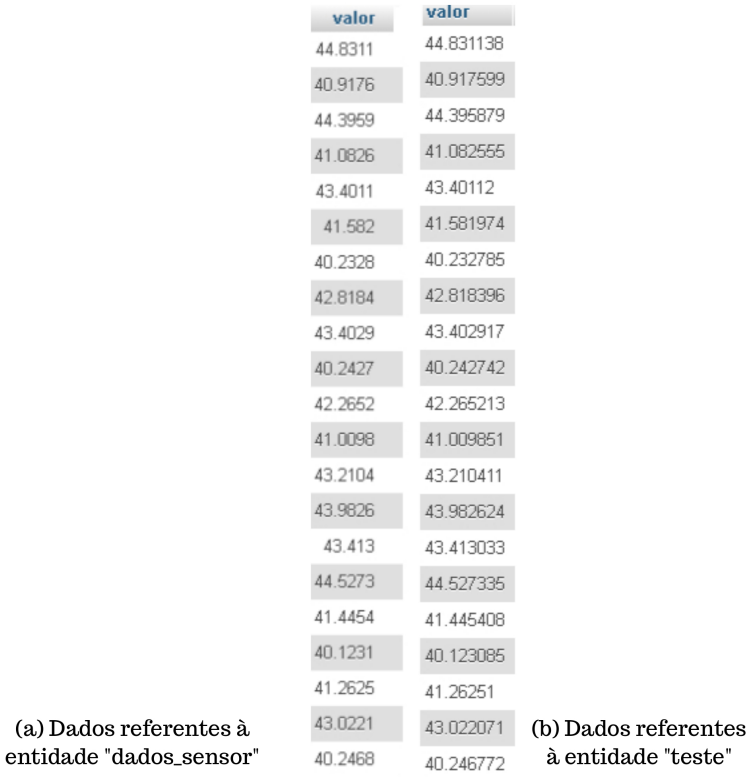
Com o *Broker Mosquitto* executando na máquina virtual com o Sistema Operacional *Linux*, foi utilizado uma máquina física com o Sistema Operacional *Windows* para executar o Módulo *Publisher* MQTT e o *SubscribeR*. Tal teste foi realizado contendo apenas um cadastro de Sensor do tipo Temperatura com o valor máximo de $45\,^{\circ}\mathrm{C}$ e mínimo de $40\,^{\circ}\mathrm{C}$ inserido no Módulo Banco de Dados precisamente na entidade "sensor".

Para a implementação do *Subscriber* de testes, foi utilizada a linguagem PHP em conjunto com a biblioteca *phpMQTT*. E quando *Subscriber* é executado o mesmo fica em alerta para receber as publicações do *Mosquitto*, nesse caso ele possui a função de receber o nome do “topico” e do campo “valor” e enviar esse “valor” para uma entidade denominada “teste” localizada em outro banco de dados implementado usando como Sistema de Gerenciamento do Banco de Dados (SGBD) o *MySQL* [Apache Friends 2018].

O tempo de duração do teste foi de aproximadente uma hora, e durante esse tempo foram feitas 1000 publicações do *Publisher* para o *Mosquitto* e desse para o *Subscriber*, e ao final foi utilizado um *script* na linguagem PHP para percorrer os banco de dados e comparar os valores das entidades “valor” e “dados-sensor”. A Figura 15 apresenta alguns dados coletados em ambos os bancos de dados (do Emulador e do paralelo a ele criado para esse teste). Através deste teste, foi possível verificar que os dados foram publicados e lidos corretamente pela aplicação teste.

9. Conclusão e Trabalhos Futuros

Considerando os benefícios das ferramentas emuladas que podem ser usadas em testes iniciais nos ambientes de desenvolvimentos de aplicações IoT, este trabalho apresentou o desenvolvimento do Emulador de *Publisher* MQTT para RSSF, com o objetivo de emular o protocolo MQTT e através dele, publicar os dados gerados pela RSSF para aplicações



valor
44.8311
40.9176
44.3959
41.0826
43.4011
41.582
40.2328
42.8184
43.4029
40.2427
42.2652
41.0098
43.2104
43.9826
43.413
44.5273
41.4454
40.1231
41.2625
43.0221
40.2468

(a) Dados referentes à entidade "dados_sensor"

valor
44.831138
40.917599
44.395879
41.082555
43.40112
41.581974
40.232785
42.818396
43.402917
40.242742
42.265213
41.009851
43.210411
43.982624
43.413033
44.527335
41.445408
40.123085
41.26251
43.022071
40.246772

(b) Dados referentes à entidade "teste"

Figura 15. Dados coletados do banco de dados do Emulador e do banco de dados de teste

de forma transparente.

Através do desenvolvimento deste trabalho, foram colocados em prática alguns conhecimentos adquiridos durante o curso de Análise e Desenvolvimento de Sistemas, tais como Banco de Dados (MYSQL e MER), Desenvolvimento *Web* (CSS e PHP), Análise Orientada à Objetos (Diagrama de Caso de Uso e Diagrama de Classe), Redes de Computadores. Além disso, foram adquiridos novos conhecimentos, tais como o conceito de Internet das Coisas englobando Redes de Sensores Sem Fio, emulação de ferramentas e protocolo MQTT.

Ainda no Emulador de *Publisher* MQTT para RSSF, é possível a implementação de novas funcionalidades, como por exemplo: criar um método para gerar a média dos valores máximos e mínimos dos Sensores inseridos no Módulo Banco de Dados, capacidade de lidar com variações mais realistas das grandezas físicas como por exemplo: temperaturas, umidades entre outras, e ainda emular perda de pacotes do *Publisher* para a aplicação assinante, além de características voltadas para a qualidade da infraestrutura de redes. E ainda novos testes podem ser realizados em outras configurações de *Hardware* e *Software*, como por exemplo: testar o sistema com desenvolvedores afim de mensurar o grau de aceitação e desempenho do mesmo em tais configurações.

Referências

Apache Friends (2018). Apache Friends Foruns. https://www.apachefriends.org/pt_br/download.html. [Online; acessado em 21

de Maio de 2018].

Cano, A. C. (2016). Redes de Sensores Sem Fio Redes Virtuais de Sensores – uma análise literária. In *Universidade Federal do Rio de Janeiro Instituto Tércio Pacitti de Aplicações e Pesquisas Computacionais, Pós Graduação em Gerência de Redes de Computadores*. Rio de Janeiro, Rio de Janeiro. (Monografia).

Correa, R., Cunha, M., Almeida, M., and Moraes, J. (2016). Simulação de aplicações utilizando o protocolo de comunicação MQTT com aplicações em ambientes industriais. In *Faculdade de Engenharia Elétrica - FEELT, Universidade Federal de Uberlândia – UFU*. Uberlândia, Minas Gerais. (Trabalho de Conclusão de Curso).

Eclipse Mosquitto (2018). An open source MQTT broker. https://mosquitto.org/. [Online; acessado em 25 de Setembro de 2018].

Font, J., Inigo, P., Domingues, M., Sevillano, L., and Cascado, D. (2011). Application-Transparent Integration of Simulation Tools in a WSN Development Environment. In *University of Seville, Department of Computer Technology and Architecture*. Seville, Spain. (Dissertação).

IBM (2017). Conhecendo o MQTT. https://www.ibm.com/developerworks/br/library/iot-mqtt-why-good-for-iot/index.html. [Online; acessado em 14 de Maio de 2018].

Info Wester (2017). O que é Internet das Coisas (Internet of Things)? https://www.infowester.com/iot.php. [Online; acessado em 11 de Setembro de 2018].

Karl, H. and Willig, A. (2006). *Protocols and Architectures for Wireless Sensor Networks*. John Wiley e Sons Ltd, The Atrium, Southern Gate, Chichester, West Sussex PO19 8SQ, England, 1th edition.

Loureiro, A. and Nogueira, J. (2016). Redes de Sensores Sem Fio. In *Departamento de Ciência da Computação, Universidade Federal de Minas Gerais*. Belo Horizonte, Minas Gerais. (Dissertação).

MDN web docs (2018). https://developer.mozilla.org/pt-BR/docs/Web/CSS. [Online; acessado em 21 de Maio de 2018].

Oliveira, R. (2016). Proposta de um Proxy Manager para Internet das Coisas. In *Pontifícia Universidade Católica de Campinas, Programa de Pós-Graduação em Engenharia Elétrica*. Campinas, São Paulo. (Dissertação).

PHP: Hypertext Preprocessor (2018). http://php.net/docs.php. [Online; acessado em 21 de Maio de 2018].

SAS Insights (2018). Evolução Tecnológica: Internet das Coisas (IoT) e processamento em tempo real (ESP). https://www.infowester.com/iot.php. [Online; acessado em 21 de Maio de 2018].

Silva, F. (2017). Avaliação de Simuladores para o ensino de IoT em Redes de Computadores. In *Universidade Federal do Ceará Campus Quixadá Tecnólogo em Redes de Computadores*. Quixadá, Ceará. (Monografia).

Sommerville, I. (2011). *Engenharia de Software*. Pearson Education, 9th edition.

Souza, I. (2017). Sistema para Monitoramento de Reservatórios de Água - Aquameasure. In *Instituto Federal de São Paulo (IFSP),Curso Superior de Tecnologia em Análise*

e Desenvolvimento de Sistemas. Hortolândia, São Paulo. (Trabalho de Conclusão de Curso).