UNIVERSIDADE DE PASSO FUNDO

INSTITUTO DE CIÊNCIAS EXATAS E GEOCIÊNCIAS

PROGRAMA DE PÓS-GRADUAÇÃO EM

COMPUTAÇÃO APLICADA

UMA PLATAFORMA DE MIDDLEWARE PARA INTERNET DAS COISAS

Roger Luis Hoff Lavarda

Passo Fundo

2018

UNIVERSIDADE DE PASSO FUNDO

INSTITUTO DE CIÊNCIAS EXATAS E GEOCIÊNCIAS

PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO APLICADA

UMA PLATAFORMA DE MIDDLEWARE PARA INTERNET DAS COISAS

Roger Luis Hoff Lavarda

Dissertação apresentada como requisito parcial à obtenção do grau de Mestre em Computação Aplicada na Universidade de Passo Fundo.

Orientador: Prof. Dr. Marco Antônio Sandini Trentin

Passo Fundo

2018

CIP – Catalogação na Publicação

L396u Lavarda, Roger Luis Hoff

Uma plataforma de middleware para internet das coisas / Roger Luis Hoff Lavarda. – 2018.

50 f. : il. color. ; 30 cm.

Orientador: Dr. Marco Antônio Sandini Trentin.

Dissertação (Mestrado em Computação Aplicada) – Universidade de Passo Fundo, 2018.

1. Internet - Programas de computador. 2. Middleware. 3. Arquitetura de software. 4. Interface de programas aplicativos (Software). I. Trentin, Marco Antônio Sandini, orientador. II. Título.

CDU: 004.4

Catalogação: Bibliotecário Luís Diego Dias de S. da Silva – CRB 10/2241

UPF Universidade de Passo Fundo | PPGCA
Programa de Pós-Graduação em Computação Aplicada
Instituto de Ciências Exatas e Geociências - ICEG

ATA DE DEFESA DO TRABALHO DE CONCLUSÃO DE CURSO DO ACADÊMICO

ROGER LUIS HOFF LAVARDA

Aos vinte e seis dias do mês de março do ano de dois mil e dezoito, às 9 horas, realizou-se, no Instituto de Ciências Exatas e Geociências, prédio B5, da Universidade de Passo Fundo, a sessão pública de defesa do Trabalho de Conclusão de Curso **"Uma plataforma de Middleware para Internet das Coisas"**, de autoria de Roger Luis Hoff Lavarda, acadêmico do Curso de Mestrado em Computação Aplicada do Programa de Pós-Graduação em Computação Aplicada – PPGCA/UPF. Segundo as informações prestadas pelo Conselho de Pós-Graduação e constantes nos arquivos da Secretaria do PPGCA, o aluno preencheu os requisitos necessários para submeter seu trabalho à avaliação. A banca examinadora foi composta pelos doutores Marco Antônio Sandini Trentin, Luiz Eduardo Schardong Spalding e Amilton Rodrigo de Quadros Martins. Concluídos os trabalhos de apresentação e arguição, a banca examinadora considerou o candidato APROVADO. Foi concedido o prazo de até quarenta e cinco (45) dias, conforme Regimento do PPGCA, para o acadêmico apresentar ao Conselho de Pós-Graduação o trabalho em sua redação definitiva, a fim de que sejam feitos os encaminhamentos necessários à emissão do Diploma de Mestre em Computação Aplicada. Para constar, foi lavrada a presente ata, que vai assinada pelos membros da banca examinadora e pela Coordenação do PPGCA.

Prof. Dr. Marco Antônio Sandini Trentin - UPF
Presidente da Banca Examinadora
(Orientador)

Prof. Dr. Luiz Eduardo Schardong Spalding - UPF
(Avaliador Interno)

Prof. Dr. Amilton Rodrigo de Quadros Martins - IMED
(Avaliador Externo)

Prof. Dr. Rafael Rieder
Coordenador do PPGCA

AGRADECIMENTOS

Agradeço primeiramente à minha família, que me incentivou e deu suporte durante a realização do mestrado. Agradeço especialmente à minha esposa Camila, pela compreensão e apoio.

Agradeço ao meu orientador, Prof. Dr. Marco Antônio Sandini Trentin, por todos os ensinamentos, conselhos e oportunidades oferecidos.

Aos demais professores do PPGCA da UPF pelo conhecimento recebido.

Por fim, e não menos importante, ao IFRS, pelo seu programa de incentivo à capacitação de seus servidores, que permitiu a minha dedicação parcial ao mestrado e à realização desse trabalho.

UMA PLATAFORMA DE MIDDLEWARE PARA INTERNET DAS COISAS

RESUMO

O avanço das redes sem fio, a miniaturização e popularização de sensores e atuadores e o surgimento das plataformas de prototipagem eletrônica de hardware livre, estão proporcionando uma grande mudança no modo de como os dispositivos eletrônicos coletam e disseminam dados, e, consequentemente, no modo como as pessoas consomem informação. Esse novo cenário tecnológico tem levado ao surgimento de uma chamada Internet das Coisas. Este termo refere-se à integração de objetos físicos e virtuais em redes conectadas à Internet, permitindo que "coisas" coletem, troquem e armazenem uma enorme quantidade de dados numa nuvem. Além dos dispositivos físicos que fazem parte da Internet das Coisas, também são necessários serviços para realizar a comunicação entre estes dispositivos e a nuvem. Um dos agentes na Internet das Coisas são os middlewares, um elemento capaz de fornecer uma abstração do sistema para as aplicações e para os desenvolvedores de aplicações. Tendo em vista a relevância das plataformas de middleware para Internet das Coisas, bem como os desafios e oportunidades de pesquisa existentes, este trabalho propõe o desenvolvimento de uma plataforma de middleware para Internet das Coisas, de código aberto, com o objetivo de conectar dispositivos da Internet das Coisas, gerenciá-los através da nuvem e prover armazenamento e visualização de dados. A plataforma foi desenvolvida utilizando as tecnologias PHP e MySQL, integrada ao broker de mensagens Eclipse Mosquitto. A fim de avaliar a eficácia da plataforma, um estudo de caso foi desenvolvido, criando situações de comunicação na nuvem entre um aplicativo Android e um dispositivo Arduino conectado à internet via WiFi Shield.

Palavras-Chave: internet das coisas, middleware, plataforma IoT.

A MIDDLEWARE PLATFORM FOR INTERNET OF THINGS

ABSTRACT

The advancement of wireless networks, the miniaturization and popularization of sensors and actuators and the emergence of free hardware electronic prototyping platforms, are providing a major change in how electronic devices collect and dissipate data, and consequently in the way people consume information. This new technological scenario has led to the eminent emergence of an Internet of Things. This term refers to the integration of physical and virtual objects into Internet connected networks, allowing "things" to collect, exchange, and store an enormous amount of data in a cloud. In addition to the physical devices that are part of the Internet of Things, we also need services to carry out communication between these devices and the cloud. One of the agents in Internet of Things is the middleware, an element capable of providing an abstraction of the system for applications and developers. In view of the relevance of middleware platforms for Internet of Things, as well as existing challenges and research opportunities, this work proposes the development of a middleware platform for Internet of Things, open source, with the aim of connect IoT devices, manage them, and provide data storage and visualization. The platform was developed using PHP and MySQL technologies, integrated with the Eclipse Mosquitto messaging broker. In order to evaluate the effectiveness of the platform, a case study was developed, creating communication situations in the cloud between an Android application and an Arduino device connected to the internet with WiFi Shield.

Keywords: internet of things, middleware, IoT platform.

LISTA DE FIGURAS

Figura 1. Modelo de publicação e assinatura do MQTT para sensores de IoT. 26
Figura 2. Arquitetura da plataforma da Elipse Mobile 27
Figura 3. Tela do sistema da Elipse Mobile. 28
Figura 4. Arquitetura da plataforma WoT Enabler. 29
Figura 5. Exemplo de hierarquia das entidades da plataforma Carriots. 30
Figura 6. Arquitetura da plataforma Yaler 31
Figura 7. Mensagem enviada por dispositivo na plataforma Yaler. 31
Figura 8. Arquitetura da Plataforma de Middleware para IoT. 33
Figura 9. Arquitetura detalhada da Plataforma de Middleware para IoT. 34
Figura 10. Modelo relacional do banco de dados. 36
Figura 11. Plataforma Web - Tela dos dispositivos. 37
Figura 12. Plataforma Web - Cadastros. 38
Figura 13. Plataforma Web - Meus locais. 38
Figura 14. Plataforma Web - Enviar comando/valor. 39
Figura 15. Plataforma Web - Informações recebidas do dispositivo. 39
Figura 16. Código para publicar mensagem no broker via PHP. 40
Figura 17. Hardware empregado no estudo de caso. 41
Figura 18. Código Arduino para publicar no tópico. 42
Figura 19. Aplicativo Android para leitura do sensor. 43
Figura 20. Código Arduino - liga e desliga LED. 44
Figura 21. Código para inscrever Arduino no tópico. 44
Figura 22. Enviar comando via plataforma Web. 45
Figura 23. Visualizar valores do dispositivo via plataforma Web. 45

LISTA DE TABELAS

Tabela 1. Comparação entre as plataformas de middlewares analisadas quanto as características. 32

LISTA DE SIGLAS

API – Application Programming Interface

CSV – Comma-Separated Values

HTTP – Hypertext Transfer Protocol

IOT – Internet of Things

JSON – JavaScript Object Notation

LED – Light Emitting Diode

M2M – Machine-to-Machine

MQTT – Message Queuing Telemetry Transport

PHP – PHP: Hypertext Preprocessor

REST – Representational State Transfer

TI – Tecnologia da Informação

URI – Uniform Resource Identifier

XML – eXtensible Markup Language

SUMÁRIO

1 INTRODUÇÃO 19

2 REVISÃO DA LITERATURA 21

2.1 INTERNET DAS COISAS 21

2.2 MIDDLEWARE PARA IOT 22

2.3 PROTOCOLO E BROKER MQTT 24

2.4 TRABALHOS RELACIONADOS 27

3 A PLATAFORMA DE MIDDLEWARE PARA IOT 33

3.1 ARQUITETURA E FUNCIONAMENTO DA PLATAFORMA 33

3.2 MODELO RELACIONAL DOS DADOS 35

3.3 PLATAFORMA WEB 36

4 ESTUDO DE CASO 41

5 CONCLUSÃO 47

REFERÊNCIAS 49

1. INTRODUÇÃO

Nas últimas décadas, ocorreu significativo aumento no poder computacional, na quantidade de dispositivos eletrônicos e nas tecnologias de comunicação de computadores disponíveis no mercado. Associado às reduções no custo de aquisição e no consumo de energia de dispositivos eletrônicos, foi habilitado um cenário nunca antes possível, o da computação ubíqua.

Internet das Coisas, em inglês, *Internet of Things* (IoT), é um novo paradigma tecnológico concebido como uma rede global de dispositivos conectados à Internet e capazes de interagir entre si [1]. O avanço das redes sem fio, a miniaturização e popularização de sensores e, principalmente, a popularização das plataformas de prototipagem eletrônica de hardware livre, colaboraram com o avanço da IoT.

A IoT está aumentando a conexão entre pessoas e coisas em proporções jamais imaginadas, superando em 1.5 para 1 o número de dispositivos conectados em relação a população mundial [2]. Apontada como uma revolução tecnológica iminente e com mercado mundial estimado em 1,7 trilhão de dólares em 2020 [3], a IoT gera impacto em todas as áreas, incluindo indústria, eletrônica de consumo, saúde e, principalmente, na forma como a sociedade consome informação.

A empolgação atual com IoT é fruto da convergência de diversas tecnologias. Em primeiro lugar, a miniaturização e popularização de sensores viabilizam a coleta e transmissão de dados, com estimativa de mais de 40 bilhões de dispositivos conectados em 2020 [4]. Tal conectividade é viabilizada pelo avanço das redes sem fio, tornando onipresente o acesso e a transmissão dos dados para a Internet.

A popularização das plataformas de prototipagem eletrônica de hardware livre também tem um papel importantíssimo na expansão da IoT. Com o surgimento desse tipo de plataforma, que teve início com o lançamento do Arduino[1], é possível desenvolver experimentos, soluções e produtos microcontrolados que interagem remotamente com ambientes diversos investindo muito pouco.

A interoperabilidade entre dispositivos de diferentes fabricantes, o reaproveitamento de dispositivos já existentes (estes sem capacidade de comunicação), a escalabilidade, a adaptabilidade, a confiabilidade, a velocidade e a segurança de estruturas para comportar esses dispositivos também apresentam desafios que demandam esforços acadêmicos e da indústria para habilitar o real potencial da IoT e da ubiquidade.

Um dos agentes na IoT são os *middlewares*, um elemento capaz de fornecer uma abstração do sistema para as aplicações e para os desenvolvedores de aplicações, abstraindo a complexidade dos mecanismos de hardware, software e interfaces de comunicação. Dessa forma, a padronização de uma camada de middleware permite a construção de aplicações independentes do hardware e do sistema operacional, executáveis em qualquer plataforma de qualquer fabricante [6].

[1] Arduino é uma placa de prototipagem eletrônica de código aberto, lançada em 2005, inclui hardware e software livre e visa oferecer ferramentas adaptáveis e de baixo custo para a criação de projetos interativos de diversas ordens [5].

A heterogeneidade dos elementos de hardware e software da IoT torna o uso de *middlewares* essencial para prover a interoperabilidade entre os dispositivos e facilitar o desenvolvimento de aplicações [7].

Considerados tais desafios, as plataformas de *middleware* para habilitar a IoT são, portanto, estruturas físicas e lógicas complexas. Tais plataformas devem ser modularizadas e utilizar o máximo de tecnologias já existentes para assim permitir seu fácil aperfeiçoamento e reposição por métodos mais atuais e/ou adequados.

Atualmente existem plataformas que disponibilizam esse tipo de serviço. No entanto, ainda existem muitas limitações. A fim de acertar a continuidade do trabalho proposto, destacam-se: (i) plataformas que apenas proveem o envio de dados dos sensores que compõem os dispositivos para a nuvem (servidor), e da nuvem para as aplicações virtuais; (ii) outros serviços permitem que sejam enviados dados das aplicações virtuais para a nuvem, e, desta, para os dispositivos físicos, permitindo o controle destes dispositivos, e não apenas o recebimento de dados coletados por sensores, mas sem suporte a aplicações virtuais de terceiros, ou seja, os desenvolvedores do produto/solução são obrigados a usar os aplicativos virtuais que apresentam as informações e controlam os dispositivos físicos disponíveis nas plataformas supracitadas e; (iii) plataformas pagas, inviabilizando muitos projetos acadêmicos, entusiastas ou projetos que ainda estão em fase se estudos e prototipação.

Diante disso, surge o seguinte problema de pesquisa: quais os processos para construção de uma plataforma de *middleware* para IoT utilizando tecnologias e ferramentas *open source*?

Nesse sentido, o presente trabalho objetiva desenvolver uma plataforma de *middleware* gratuita capaz de abstrair o processo de comunicação e controle de dispositivos na IoT, além de fornecer armazenamento e visualização de dados na nuvem. Acredita-se que a disponibilização deste serviço vai ao encontro com paradigma da IoT e contribuirá para o desenvolvimento de soluções para esta, facilitando o trabalho dos desenvolvedores de aplicações para a IoT, através da abstração propiciada pelo *middleware*, uma vez que parte do trabalho a ser realizado na comunicação entre os dispositivos já estaria pronto.

Este documento está estruturado da seguinte forma. O Capítulo 2 descreve a revisão da literatura e os trabalhos relacionados, apresentando cinco plataformas de *middleware* usadas para IoT. O Capítulo 3 descreve os elementos da plataforma desenvolvida, a arquitetura, funcionamento, modelo de banco de dados e detalhes de implementação. No Capítulo 4 é apresentado um estudo de caso, a fim de avaliar o funcionamento da plataforma. Por fim, o Capítulo 5 contém as considerações finais e trabalhos futuros.

2. REVISÃO DA LITERATURA

Esta seção apresenta os principais conceitos adotados na plataforma de *middleware* para IoT, bem como os trabalhos relacionados ao tema mais relavantes.

2.1 INTERNET DAS COISAS

A IoT, refere-se à integração de objetos físicos e virtuais em redes conectadas à Internet, permitindo que “coisas” coletem, troquem e armazenem uma enorme quantidade de dados numa nuvem [8].

Borgia [9] ressalta que a IoT tem muito potencial para o desenvolvimento de novas aplicações inteligentes em quase todos os campos. Isso será possível devido a sua capacidade de realizar a detecção do ambiente (permitindo, por exemplo, coletar informações sobre fenômenos naturais, parâmetros médicos ou informações sobre os usuários), e oferecer-lhes serviços personalizados. Independentemente do campo de aplicação, tais aplicações destinam-se a melhorar a qualidade de vida, e terão um profundo impacto sobre a economia e da sociedade.

Os vários aplicativos podem ser agrupados em três grandes domínios: (A) industrial; (B) cidades inteligentes e saúde; (C) bem-estar doméstico. Cada domínio não é isolado a partir dos outros, mas é parcialmente sobreposto uma vez que algumas aplicações são compartilhadas [9].

Segundo Ocampos [10], a IoT demanda Computação em Nuvem em diversos níveis de serviço, incluindo infraestrutura, plataforma, software e análise de dados. Em um mundo com bilhões de dispositivos gerando dados, serão necessários serviços escaláveis, robustos e de alta disponibilidade para armazenar, processar, personalizar e entregar informações de alto valor agregado para os clientes, a qualquer momento, em qualquer lugar.

Os campos de aplicação para as tecnologias IoT são tão numerosos quanto diversos, já que as soluções IoT estão se estendendo cada vez mais a praticamente todas as áreas do cotidiano. As áreas de aplicação mais importantes incluem, por exemplo, a indústria inteligente, onde o desenvolvimento de sistemas inteligentes e sites de produção conectados são frequentemente discutidos sob o título Indústria 4.0. Na área de casas inteligentes, os termostatos inteligentes e os sistemas de segurança estão recebendo muita atenção, enquanto as aplicações inteligentes de energia se concentram em medidores de eletricidade, gás e água. As soluções de transporte inteligentes incluem, por exemplo, o rastreamento da frota de veículos e a emissão de bilhetes móveis, enquanto que na área da saúde inteligente, temas como a vigilância dos pacientes e o gerenciamento de doenças crônicas estão sendo abordados. E no contexto de projetos de cidades inteligentes, soluções como o monitoramento em tempo real da disponibilidade do espaço de estacionamento e iluminação inteligente das ruas estão sendo exploradas [11].

No seu núcleo, a inovação na IoT é caracterizada pela combinação de componentes físicos e digitais para criar novos produtos e permitir novos modelos de negócios. Graças ao gerenciamento

de energia cada vez mais eficiente, comunicação de banda larga, memória confiável e avanços em tecnologias de microprocessadores, tornou-se possível digitalizar funções e recursos-chave de produtos de idade industrial. Consequentemente, uma série de oportunidades está se desenrolando para que as empresas gerem valor incremental na Internet das coisas. Isso demonstra que as soluções IoT normalmente combinam coisas físicas com TI (Tecnologia da Informação) na forma de hardware e software. Como resultado, as principais funções físicas de uma coisa podem ser aprimoradas com serviços digitais adicionais baseados em TI, que podem ser acessados, não só a nível local, mas a nível global [11].

Contudo, há ainda uma série de desafios a serem superados para alavancar a ampla disseminação desse paradigma, principalmente com relação ao desenvolvimento de aplicações e a alta heterogeneidade decorrente da inerente diversidade de tecnologias de hardware e software desse ambiente. O primeiro desafio diz respeito a heterogeneidade dos ambientes de IoT, a qual demanda soluções para permitir a interoperabilidade e integração dos diversos componentes que fazem parte desses ambientes. Nesse contexto, plataformas de *middleware* têm surgido como soluções promissoras para prover tal interoperabilidade e gerenciar a crescente variedade de dispositivos associados a aplicações, bem como o consumo de dados por parte dos usuários finais [12].

Tais plataformas são inseridas entre as aplicações e a infraestrutura (de comunicação, processamento e sensoriamento) subjacente, provendo um meio padronizado para o acesso aos dados e serviços fornecidos pelos objetos através de uma interface de alto nível [13].

A adoção de uma plataforma de *middleware* também pode contribuir para facilitar a construção de aplicações para IoT. Nesse contexto, o desafio reside no fato de que, a fim de permitir a criação de aplicações que combinem recursos do mundo físico disponibilizados via Web, são necessários modelos de alto nível que abstraiam os serviços e dispositivos físicos subjacentes. Com isso, usuários e aplicações consumidores dos dados originados dos dispositivos conectados não precisarão lidar com funcionalidades de baixo nível para a manipulação de tais objetos [14].

2.2 MIDDLEWARE PARA IOT

De acordo com Gubbi et al. [15], *middleware* é uma camada de software interposta entre softwares heterogêneos, com o objetivo de ocultar do programador diferenças de protocolos de comunicação, plataformas e dependências. Sua característica de abstrair os detalhes de diferentes tecnologias é fundamental para que desenvolvedores de aplicações para IoT criem suas soluções sem a necessidade de dominar todas as tecnologias que envolvem o desenvolvimento destes softwares. *Middleware* ganhou popularidade na década de 1980 devido ao seu papel de simplificar a integração de tecnologias legadas a novas tecnologias. Também facilitou o desenvolvimento de novos serviços no ambiente de computação distribuída. Uma infraestrutura distribuída complexa da IoT com vários dispositivos heterogêneos requer o desenvolvimento de novas aplicações e serviços, de modo que o uso do *middleware* é primordial para o desenvolvimento de aplicativos IoT.

Considerando que dispositivos e redes fornecem conectividade, as aplicações IoT permitem as interações entre dispositivos e interações de humano a dispositivo de forma confiável e robusta. Aplicações IoT em dispositivos precisam garantir que os dados/mensagens tenham sido recebidos e atuados corretamente em tempo hábil [15].

Por exemplo, as aplicações de transporte e logística monitoram o status dos bens transportados, como frutas, produtos frescos, carne e produtos lácteos. Durante o transporte, o estado de conservação (temperatura, umidade, vibração) é monitorada e ações apropriadas são tomadas automaticamente para evitar deterioração quando a conexão fica fora de alcance. A FedEx[2], por exemplo, usa sensores para manter temperatura, localização e outros sinais vitais de um pacote, inclusive se é aberto e se foi adulterado ao longo do caminho. Embora as aplicações para dispositivos não necessitem necessariamente de visualização de dados, cada vez mais as aplicações IoT centradas no ser humano fornecem visualização para apresentar informações aos usuários finais de forma intuitiva e fácil de entender e permitir interação com o meio ambiente. É importante que aplicações IoT sejam criadas com inteligência para que os dispositivos possam monitorar o meio ambiente, identificar problemas, comunicar-se entre si e potencialmente resolver problemas sem a necessidade de intervenção humana [15].

Do ponto de vista tecnológico, para Wortmann e Flüchter [11], a implementação de um produto conectado normalmente requer a combinação de vários componentes de software e hardware em uma pilha multicamada de tecnologias IoT, composta por três camadas principais, isto é, a camada de dispositivo, camada de comunicação ou conectividade e a camada de nuvem do IoT.

Na camada do dispositivo, o hardware IoT, como sensores e atuadores, pode ser adicionado aos componentes de hardware principais existentes, e o software embarcado pode ser modificado para gerenciar e operar o dispositivo. Na camada de conectividade, protocolos de comunicação como o MQTT (*Message Queuing Telemetry Transport*) permitem a comunicação entre o indivíduo e a nuvem. E na camada de nuvem IoT, o software de comunicação e gerenciamento de dispositivos é usado para se comunicar, fornecer e gerenciar as coisas conectadas, enquanto uma plataforma de aplicativos permite o desenvolvimento e a execução das aplicações IoT. Além disso, o software de análise e gerenciamento de dados é empregado para armazenar, processar e analisar os dados gerados pelas coisas conectadas [11].

No contexto de discussões sobre tecnologias IoT, um conceito frequentemente usado é o das plataformas IoT. Elas são essencialmente produtos de software, que oferecem conjuntos abrangentes de funcionalidades independentes de plataforma, que podem ser utilizados para criar aplicativos IoT. A natureza dessas plataformas podem variar consideravelmente, os provedores se concentram em diferentes aspectos da pilha de tecnologia IoT e, correspondentemente, incluem diversos conjuntos de funcionalidades em suas ofertas.

Entretanto, não há padrão de arquitetura de uma plataforma IoT. Enquanto algumas plataformas IoT, por exemplo, Eclipse IoT[3], são focadas em oferecer funcionalidades para operar aplicativos

[2]FedEx: https://www.fedex.com/br/index.html
[3]Eclipse IoT: https://iot.eclipse.org/

embutidos em dispositivos IoT, outros plataformas, como Xively[4], se concentram nas funcionalidades específicas da IoT para complementar as plataformas não IoT existentes [11].

Existem várias propostas de *middleware* para IoT, cada uma atendendo um subconjunto dos requisitos necessários para viabilizar tais ambientes, principalmente no que tange a comunicação entre dispositivos heterogêneos. O desenvolvimento de plataformas de *middleware* especificamente voltadas para ambientes de IoT é uma área de pesquisa recente que tem atraído a atenção da indústria e da comunidade acadêmica [16].

Pires et al. [16] cita os trabalhos de Qin et al. [17] e Gao et al. [18], colocando-os como exemplos de plataformas concebidas para endereçar alguns dos desafios anteriormente descritos, principalmente com relação a integração transparente de dispositivos heterogêneos e a provisão de mecanismos de alto nível para desenvolvimento de aplicações. Contudo, tais propostas ainda não atingiram um nível de maturidade, requerendo esforços de pesquisa adicionais, como: (i) a construção de infraestruturas robustas e tolerantes a falha para gerenciar e processar dados provenientes dos vários dispositivos integrados; (ii) o gerenciamento de incertezas e resolução de conflitos, e; (iii) suporte à adaptação de aplicações sob condições dinâmicas do ambiente.

Outros desafios concernem a enorme escalabilidade desses ambientes, em termos do número de dispositivos conectados, a necessidade de gerenciar tais dispositivos e ao grande volume de dados produzidos.

2.3 PROTOCOLO E BROKER MQTT

Na Internet 'tradicional' existem diversos protocolos de comunicação responsáveis por gerenciar a transferência de dados em uma rede que conecta dois ou mais computadores. Estes protocolos de mensagens são utilizados a partir do dispositivo até um *gateway* de IoT. o MQTT tornou-se o protocolo padrão para comunicações de IoT. Segundo Yuan [19], o MQTT é um protocolo de comunicação M2M (*Machine-to-Machine*) e foi projetado para ter baixo consumo de banda de rede e requisitos de hardware, sendo extremamente leve e simples. Ele pode ser utilizado em vários tipos de dispositivos em diferentes áreas de negócios, como: saúde, energia, entre outros.

Uma das principais vantagens em utilizar um protocolo específico para IoT, como o MQTT, é poder trocar mensagens entre os dispositivos de forma mais ágil e menos burocrática. Em contrapartida, o HTTP (*Hypertext Transfer Protocol*), que também é largamente utilizado em plataformas de *middleware*, é um protocolo síncrono, unidirecional, pesado, com muitos cabeçalhos e regras e, principalmente, é um protocolo de um para um.

O MQTT utiliza o paradigma *publish/subscribe* (pub/sub) para a troca de mensagens. Este paradigma implementa um *middleware* chamado de MQTT *Broker*. Ele é responsável por receber, enfileirar e retransmitir as mensagens recebidas dos *publisher* para os *subscribers* [20]. O *publisher* é responsável por se conectar ao MQTT *Broker* e publicar mensagens. Já o *subscriber* é responsável por se conectar ao MQTT *Broker* e receber as mensagens que ele tiver interesse. Ou seja, *publishers*

[4]Xively: https://www.xively.com/

enviam informação para o *Broker*, *subscribers* recebem informação do *Broker*, e este gerencia a troca de mensagens.

O protocolo MQTT define dois tipos de entidades na rede: um *message broker* e inúmeros clientes. O *broker* é um servidor que recebe todas as mensagens dos clientes e, em seguida, roteia essas mensagens para os clientes de destino relevantes. Um cliente é qualquer coisa que possa interagir com o broker e receber mensagens. Um cliente pode ser um sensor de IoT em campo ou um aplicativo em um data center que processa dados de IoT [21]. Esse cenário pode ser melhor compreendido na situação que segue:

- O cliente conecta-se ao broker. Ele pode assinar qualquer "tópico"de mensagem no broker. Essa conexão pode ser uma conexão TCP/IP simples ou uma conexão TLS criptografada para mensagens sensíveis;
- O cliente publica as mensagens em um tópico, enviando a mensagem e o tópico ao *broker*;
- Em seguida, o broker encaminha a mensagem a todos os clientes que assinam esse tópico.

O paradigma pub/sub utiliza o conceito de tópicos para processar as mensagens, em que cada mensagem é enviada para um determinado tópico. Diferente de outros protocolos de mensagem, o *publisher* não envia a mensagem diretamente ao subscriber, mas sim ao *broker*. O *publisher* envia a mensagem para o MQTT Broker em um determinado tópico. O *Broker* é responsável por receber a mensagem do publisher e fazer uma pré-filtragem das mensagens e enviá-las para os *subscribers* que estivem registrados em um determinado tópico [19]. O tópico lembra o conceito de URI (*Uniform Resource Identifier*), com níveis separados por barras (“/”). Elementos da rede podem enviar diversos tópicos para o *broker* e subscritores podem escolher os tópicos que desejam subscrever.

Barros [22] exemplifica a identificação das mensagens no MQTT na seguinte situação: em uma rede de sensores, existam vários sensores diferentes de temperatura e umidade, publicando o valor do sensor como o dado útil e identificando as mensagens com tópicos nos seguintes formatos:

```
area/ID_da_area/sensor/ID_do_sensor/temperatura
area/ID_da_area/sensor/ID_do_sensor/umidade
```

Neste esquema, possíveis publicações seriam:

```
area/10/sensor/5000/temperatura
area/10/sensor/5000/umidade
area/10/sensor/5001/temperatura
area/10/sensor/5001/umidade
area/20/sensor/4000/temperatura
area/20/sensor/4000/umidade
area/20/sensor/4001/temperatura
area/20/sensor/4001/umidade
```

Neste exemplo, temos informações provenientes de duas áreas diferentes (10 e 20) sendo geradas por quatro sensores (5000, 5001, 4000, 4001), onde cada sensor publica temperatura e umi-

dade. O valor da temperatura ou umidade faz parte dos dados da mensagem, sendo o formato algo dependente da aplicação, uma vez que o MQTT não impõe restrições sobre isso.

Um subscritor pode assinar estas mensagens, especificando exatamente o tópico que deseja. No entanto, é muito mais interessante receber informações agrupadas. Por exemplo, para assinar todas as informações de sensores de temperatura da área 10, o seguinte tópico poderia ser usado:

```
area/10/sensor/#/temperatura
```

O "#" tem função de curinga, aceitando qualquer qualquer coisa abaixo de determinado nível do tópico. Sendo assim, no exemplo supramencionado, o subscritor para tem acesso a qualquer valor de sensoriamento dentro da área 10.

Como as mensagens do MQTT são organizadas por tópicos, o desenvolvedor de aplicativos tem a flexibilidade de especificar que determinados clientes somente podem interagir com determinadas mensagens. Por exemplo, considerando a Figura 1, os sensores publicarão suas leituras no tópico "sensor_data"e assinarão o tópico "config_change". Os aplicativos de processamento de dados que salvam os dados do sensor em um banco de dados de backend assinarão o tópico "sensor_data". Um aplicativo de console administrativo poderia receber comandos do administrador do sistema para ajustar as configurações dos sensores, como a sensibilidade e a frequência de amostragem, e publicar essas mudanças no tópico "config_change"[21].

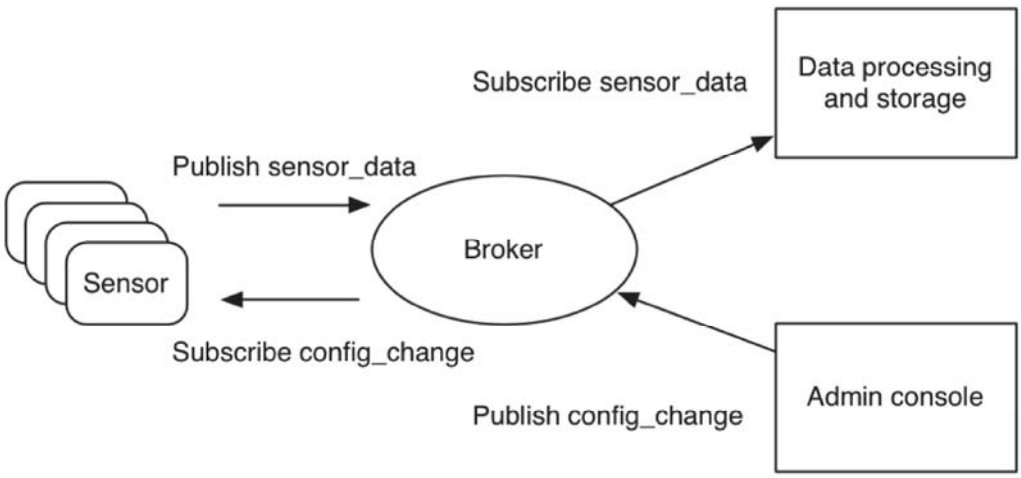


Figura 1. Modelo de publicação e assinatura do MQTT para sensores de IoT[21].

Contudo, o MQTT é apenas um protocolo, ou seja, um conjunto de regras e procedimentos a serem respeitados para emitir e receber dados numa rede. Para implementação do MQTT, é necessário que um *broker* seja instalado e configurado em um servidor. Nesse sentido, o Eclipse Mosquitto[5] é um *broker* de mensagens *open-source* que trabalha sob o protocolo MQTT para distribuir mensagens leves utilizando-se do protocolo publicador/subscritor [23]. Devido a seu protocolo de baixo consumo e a utilização de um protocolo leve, Mosquitto é um ótimo *broker* para se utilizar em ambientes de

[5]Aclipse Mosquitto: https://mosquitto.org/

IoT onde os componentes, normalmente, possuem baixa capacidade computacional. Ele pode ser baixado e instalado nos principais sistemas operacionais disponíveis do mercado.

2.4 TRABALHOS RELACIONADOS

Existem várias propostas de *middleware* para IoT, cada uma atendendo um subconjunto dos requisitos necessários para viabilizar tais ambientes, principalmente no que tange a comunicação entre dispositivos heterogêneos. O desenvolvimento de plataformas de middleware especificamente voltadas para ambientes de IoT é uma área de pesquisa recente que tem atraído a atenção da indústria e da comunidade acadêmica.

Na literatura são encontradas diversas propostas e sistemas voltados para o problema de integração entre dispositivos para IoT.

A seguir são apresentadas as arquiteturas de alguns sistemas: os mais diretamente relacionados com a presente proposta. Além destes, outros foram analisados, mas não serão aqui apresentados devido as semelhanças com outros sistemas ou por utilizarem outras abordagens. Pires et al. [16] apresenta um resumo de outras plataformas de *middleware* para Internet das Coisas.

A primeira plataforma, Elipse Mobile [24], é a plataforma da empresa Elipse para conectar seu sistema proprietário a equipamentos na internet. Utilizando um WebBrowser ou através do aplicativo do Elipse Mobile para celular é possível monitorar e controlar vários equipamentos conectados na internet, incluindo o Arduino.

A Elipse disponibiliza uma biblioteca para conectar o Arduino a um servidor online, chamado ArduinoCloudLink. O projeto contém a documentação e código para ser colocado no Arduino. A Figura 2 mostra a arquitetura da Elipse Mobile.

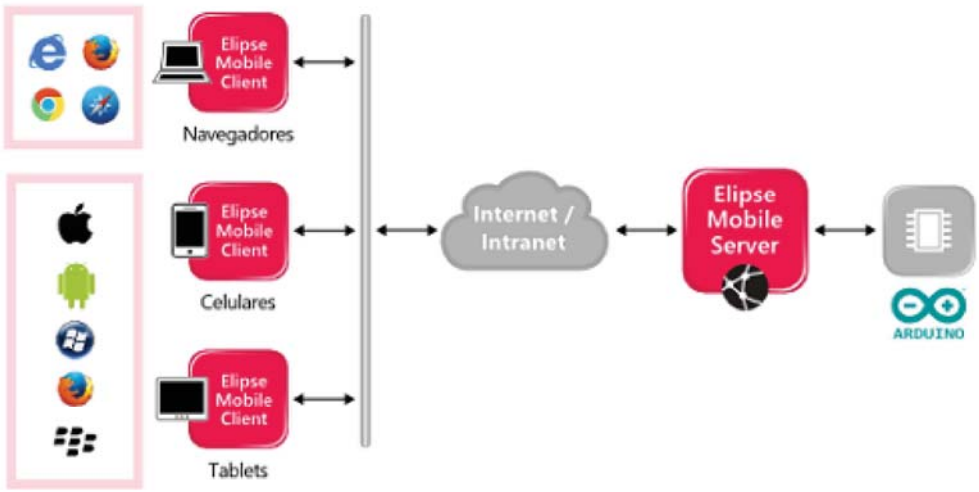


Figura 2. Arquitetura da plataforma da Elipse Mobile [24].

No entanto, a solução da Elipse Mobile não permite que aplicações de terceiros se comuniquem com a nuvem, forçando o desenvolvedor usar as aplicações virtuais disponibilizadas pela Elipse.

Na Figura 3 é mostrado uma tela do sistema online da referida plataforma, onde podem ser customizadas e programadas algumas funcionalidades, como apresentação dos dados recebidos de sensores, consumo e manutenção.

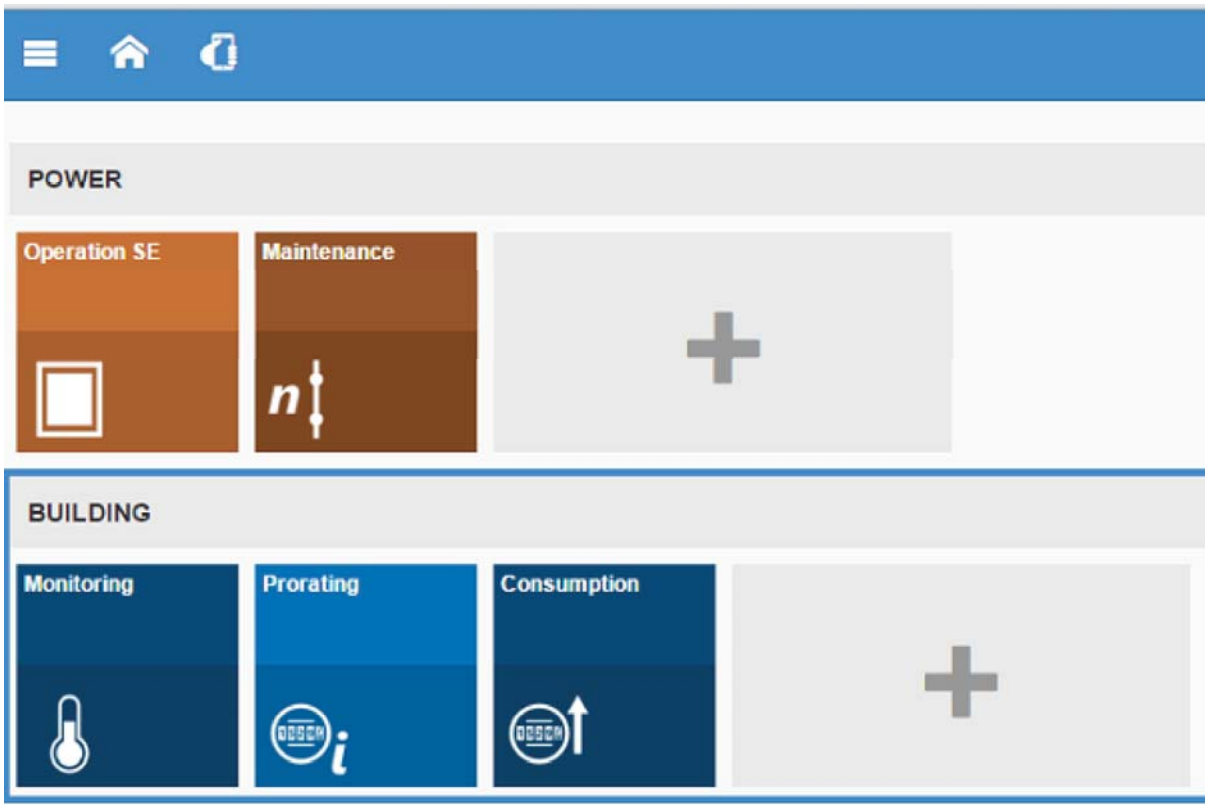


Figura 3. Tela do sistema da Elipse Mobile [24].

Já a plataforma de *middleware open source* OpenIoT [25] tem por objetivo ser uma camada de suporte para aplicações em IoT usando um modelo baseado em infraestrutura de nuvem. Os recursos IoT podem ser acessados por serviços sob demanda que seguem o modelo de computação em nuvem, e através do uso de serviços de nuvem para IoT, usuários podem configurar, implementar e usar a IoT.

A plataforma tem como proposta permitir a customização de ambientes de nuvens auto-organizáveis para aplicações IoT, habilitando a implantação por provedores de serviços de infraestruturas de nuvem que forneçam serviços IoT a usuários.

A OpenIoT conecta sensores com o ambiente de nuvem, de forma que os recursos da nuvem poderão ser usados para processamento e gerenciamento de dados, funções especialmente úteis para processamento intensivo de sinais que, em geral, não podem ser realizadas em infraestrutura de IoT devido aos recursos limitados dos dispositivos.

WoT Enabler [18] é uma plataforma gratuita baseada em REST (*Representational State Transfer*) para a integração de sensores e compartilhamento de seus dados. Implementada utilizando a linguagem de programação Ruby, seu servidor se comunica com uma base de dados conectada à arquitetura, responsável pelo armazenamento dos dados os quais são enviados diretamente pelos dispositivos conectados (para dispositivos dotados de conexão direta à Internet) ou por *gateways* (para dispositivos que não possuem tal capacidade). Assim, tais dados podem ser acessados por aplica-

ções e usuários através de requisições HTTP RESTful ao servidor WoT Enabler, que pode retornar representações de dados nos formatos XML, JSON ou CSV, como ilustra a Figura 4. A arquitetura usa a seguinte estruturação dos dados provenientes dos sensores, diferindo apenas na hierarquia de dados utilizada: (i) um system representa uma coleção de dados referentes a um determinado ambiente; (ii) um sensor representa um sensor individual no contexto de um system e; (iii) um data é um par <chave, valor> que se refere ao valor de uma medida aferida por um sensor em um dado instante de tempo.

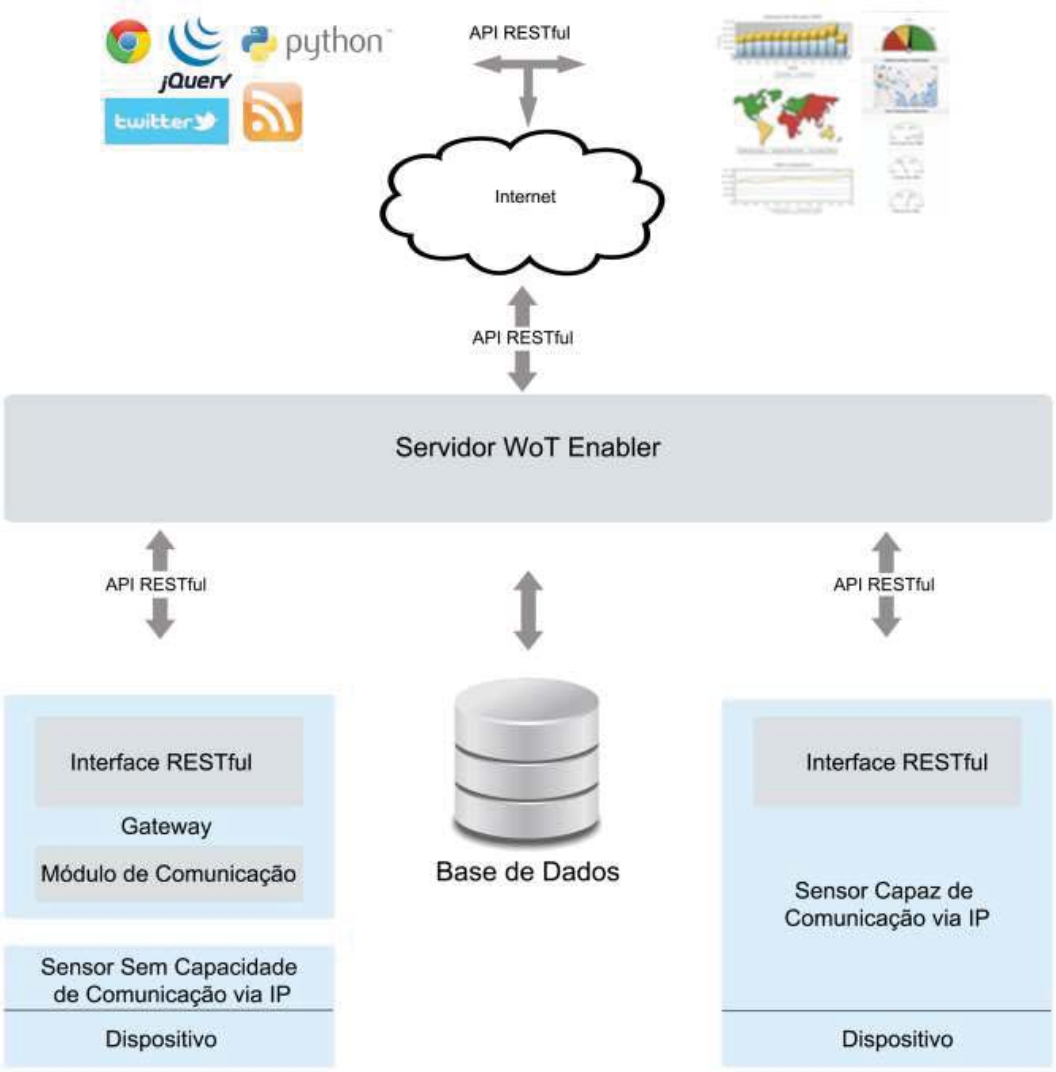


Figura 4. Arquitetura da plataforma WoT Enabler [18].

Outra plataforma encontrada, a Carriots[6] também é uma plataforma de *middleware* para IoT que utiliza serviços de nuvem para gerenciar dados providos por dispositivos, além de conectar dispositivos a dispositivos e a outros sistemas. Portanto, se um sistema for conectado a plataforma, ele também pode ser modelado como um dispositivo. A partir de sua API RESTful, a Carriots tem por objetivo coletar e armazenar qualquer dado originado dos mais diversos dispositivos, utilizá-lo em

[6]Carriots: http://www.carriots.com

seu motor de aplicações e disponibilizá-lo a seus usuários não importando o volume de dispositivos conectados.

Como ilustrado pela Figura 5, as entidades da plataforma possuem uma hierarquia bem definida. Dessa forma, todos os dispositivos conectados ao middleware são associados a serviços (e.g., dispositivos físicos ou algum outro recurso) e todos os serviços pertencem a um projeto. Por exemplo, sensores de estacionamento, independente do seu contexto de utilização (como localização geográfica distinta), pertencem ao mesmo serviço dentro de um projeto. Dessa forma, a primeira atividade de um usuário ao interagir com a plataforma é a criação de um projeto. Por conta dessa hierarquia, se um projeto for desabilitado, isso impactará na desativação de todas as entidades vinculadas a ele. Além disso, *triggers* nessa plataforma são associados a exportação dos dados dos serviços. Do ponto de vista dos dispositivos, eventos tais como o recebimento ou a persistência de dados podem ser monitorados por listeners, entidades responsáveis por executarem análise e processamento em caso de alguma condição pré-configurada ter sido atendida.

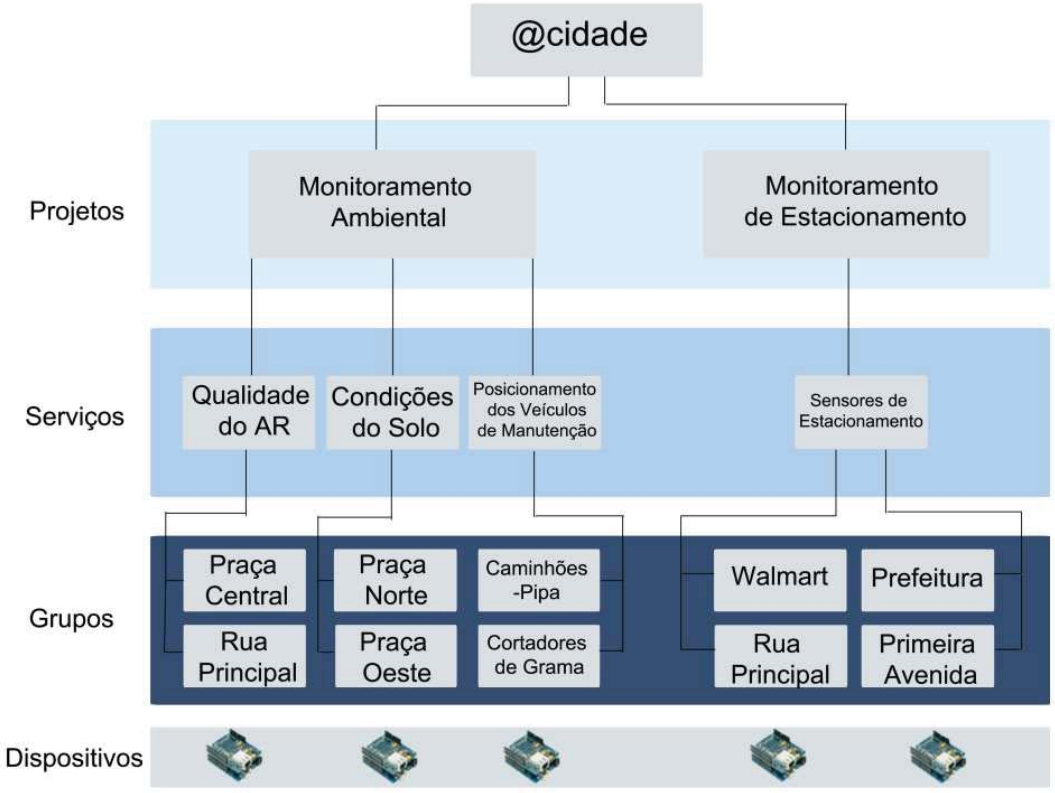


Figura 5. Exemplo de hierarquia das entidades da plataforma Carriots.

Os dados trocados entre os dispositivos, os sistemas conectados e a plataforma podem ser representados de duas formas distintas: (i) sensores enviam dados nos formatos JSON ou XML (em um formato particular da plataforma) usando a APIREST, ou (ii) através do protocolo de mensagens MQTT.

Por fim, a plataforma Yaler [26], oferece serviços de conexão entre as principais plataformas de prototipagem eletrônica (Arduino, Raspberry Pi, Arduino, Intel Edison, entre outras) e aplicações

cliente (Android, iOS, Google Chrome, Firefox). Na Figura 6 podemos observar como a plataforma realiza a conexão entre os dispositivos e o servidor.

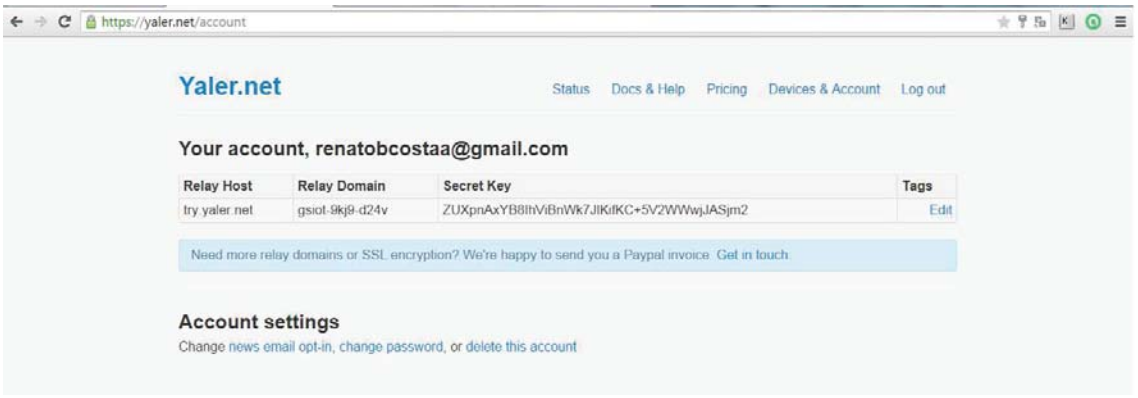


Figura 6. Arquitetura da plataforma Yaler [26].

Segundo a documentação disponível no portal [26], a plataforma Yaler disponibiliza a comunicação em duas vias entre os dispositivos virtuais e os dispositivos físicos através de protocolos alternativos, como Bluetooth Smart e ZigBee, e a conexão com a internet destina-se para o envio de dados dos dispositivos para a nuvem. A Figura 7 apresenta uma mensagem enviada por um dispositivo conectado à internet sendo impresso na tela do navegador.

Figura 7. Mensagem enviada por dispositivo na plataforma Yaler [26].

A Tabela 1 oferece uma visão das características/funcionalidades que são implementadas pelas plataformas apresentadas. As características e funcionalidades analisadas são: (i) protocolos dos dispositivos; (ii) visualização dos dados na nuvem; (iii) interface de alto nível; (iv) se a plataforma permite aplicativos de terceiros, e; (v) código aberto.

Pode-se observar que os protocolos utilizados comumente é o MQTT e o HTTP. Além disso, somente as plataformas Elipse Mobile e Carriots disponibilizam interface de alto nível e visualização de dados na nuvem. Porém, a plataforma da Elipse não permite que aplicativos de terceiros se conectem à

sua plataforma. Quanto a disponibilização do código, somente as plataformas OpenIoT e WoT Enabler são de código aberto.

Tabela 1. Comparação entre as plataformas de middlewares analisadas quanto as características.

Característica / funcionalidade	Elipse Mobile	OpenIoT	WoT Enabler	Carriots	Yaler
Protocolos dos dispositivos	HTTP	MQTT	HTTP	HTTP e MQTT	HTTP, SSH ou VNC
Visualização dos dados na Nuvem	Sim	Não	Não	Sim	Não
Interface de alto nível	Sim	Não	Sim	Sim	Não
Permite aplicativos de terceiros	Não	Sim	Sim	Sim	Sim
Código aberto	Não	Sim	Sim	Não	Não

3. A PLATAFORMA DE MIDDLEWARE PARA IOT

Este capítulo descreve o desenvolvimento da plataforma de *middleware* para IoT proposto, bem como os padrões, funcionamento e tecnologias utilizadas. Todo código desenvolvido neste trabalho é de domínio público e encontra-se disponível no GitHub, em <https://github.com/rogerlavarda/iot_platform>.

3.1 ARQUITETURA E FUNCIONAMENTO DA PLATAFORMA

Esta seção apresenta a arquitetura de *middleware* proposta para a IoT neste trabalho. A plataforma proposta neste trabalho é composta por uma aplicação em nuvem, responsável por permitir a visualização das informações dos dispositivos e permitir gerenciá-los, e um servidor com a função de intermediar as mensagens entre dispositivos conectados à IoT, denominado *broker*.

A Figura 8 traz uma visão de como o *middleware* proposto integra dispositivos e aplicações. As três funções fundamentais são a de controle e monitoramento de estado de dispositivos, armazenamento de dados e comunicação entre dispositivos e aplicações para IoT.

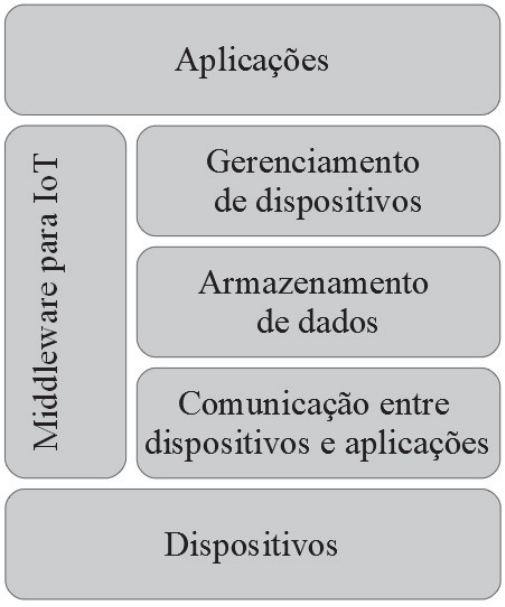


Figura 8. Arquitetura da Plataforma de Middleware para IoT.

A primeira função consiste em enviar requisições de controle de uma aplicação para um dispositivo. O *middleware* recebe as requisições de aplicações e transmite os comandos/dados para os dispositivos alvo. Para isso, o middleware utiliza suas abstrações lógicas de aplicações e repassa os comandos para os dispositivos, que executarão os comandos. Uma vez finalizado esse processo, o middleware retorna uma resposta para o dispositivo notificando a conclusão da operação.

A segunda função, responsável pela comunicação entre as entidades, consiste em monitorar o estado de um dispositivo e, quando perceber uma informação, enviá-las as aplicações e a plataforma

web desse novo estado. O fluxo contrário das informações também são de responsabilidade desta função.

Já a terceira função consiste em armazenar as informações enviadas pelos dispositivos e disponibilizá-la via interface web. Para isso, a plataforma guarda os dados enviados dos dispositivos para as aplicações e utiliza uma interface web para apresentá-las ao usuário.

A Figura 9 traz uma visão detalhada de como a plataforma de *middleware* para IoT desenvolvido funciona, englobando um conjunto de componentes representativo dos dispositivos, componentes, protocolos e sistemas que utilizam a solução proposta.

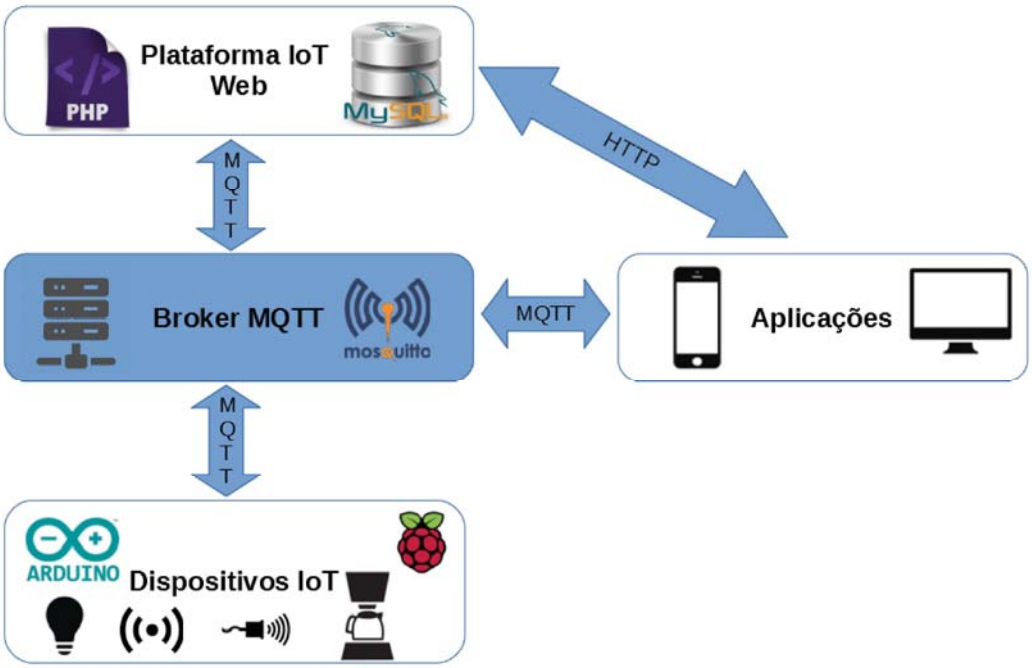


Figura 9. Arquitetura detalhada da Plataforma de Middleware para IoT.

Como pode ser visto na Figura 9, o cerne da plataforma de *middleware* é o *broker* MQTT, implementado com o servidor Mosquitto. O *broker* pode ser representado como um intermediário central de mensagens (servidor), destinado a gerenciar o intercâmbio de notificações e conectado a instâncias (clientes). Dentro o *middleware*, ele é o responsável por receber as mensagens enviadas pelos dispositivos e transmiti-las às aplicações e para a plataforma web.

A troca de mensagens entre os dispositivos, aplicações e plataforma web ocorre via protocolo MQTT. Os dispositivos publicam as mensagens (valor de sensores, comandos, etc.) em um tópico específico. Por exemplo, um sensor publica a temperatura coletada no tópico *ufe40qa345/cozinha/sensorTemp/28.90*, onde *ufe40qa345* é o ID do usuário na plataforma, *cozinha* é a localização do sensor, *sensorTemp* o nome do sensor e *28.90* é o valor coletado. O *Broker*, por sua vez, transmite estas mensagens para todos as aplicações que assinam o respectivo tópico, inclusive para a plataforma web. Esta, ao receber a mensagem contendo, além do conteúdo, o tópico alvo, armazena os valores enviados pelos dispositivos no banco de dados. Desta forma, a plataforma identifica quem enviou a mensagem, tornando possível armazenar o histórico de mensagens de todos os dispositivos acoplados a plataforma.

O mesmo acontece quando as aplicações enviam mensagens/comandos para os dispositivos ou quando um comando é disparado via plataforma web. A aplicação envia a mensagem utilizando o protocolo MQTT ao *broker*, que transmite aos respectivos dispositivos.

A especificação dos tópicos na plataforma segue o padrão *tópico_usuário/local_do_dispositivo/tópico_dispositivo/valor*, sendo que:

- tópico_usuário: cada usuário cadastrado na plataforma recebe uma chave única, utilizada como tópico do usuário, ou seja, cada usuário utiliza seu próprio tópico no broker;
- local_do_dispositivo: o local do dispositivo é usado para organizar os dispositivos (que podem ser placas de prototipagem eletrônica ou sensores e atuadores) em ambientes diferentes;
- tópico_dispositivo: cada dispositivo recebe uma chave única, responsável por identificar um dispositivo dos demais cadastrados no banco de dados da plataforma;
- valor: informação ou comando enviado pelos dispositivos ou aplicações.

Todos os dispositivos e aplicações conectadas estão relacionadas com um usuário da plataforma. Para fazer uso da plataforma de *middleware*, tanto os dispositivos IoT quanto as aplicações devem ser configurados para utilizar protocolo MQTT, indicando o endereço do servidor, usuário, chave de acesso, tópico que a mensagem será publicada e o seu conteúdo. A plataforma encarrega-se de entregar as mensagens aos destinatários e armazenar todas as informações no banco de dados.

Quanto a plataforma web, desenvolvida com a linguagem de programação PHP[7] e SGBD MySQL[8] para persistência dos dados, tem o objetivo de prover aos usuários do *middleware* as funções de gerenciamento dos dispositivos IoT e visualização de dados. Entende-se por gerenciamento dos dispositivos a capacidade da plataforma web de permitir o cadastro, edição e exclusão de placas para IoT e seus respectivos locais, além do envio de comandos aos dispositivos.

A visualização de dados na plataforma pode ser feito de duas maneiras, em tempo real e através de dados históricos. As informações que passam pelo *broker* (servidor MQTT), enviadas dos dispositivos as aplicações, também são enviadas a plataforma web. Esta, por sua vez, mostra as informações em um componente dinâmico no navegador e armazena-as no banco de dados. Uma vez que dados estejam armazenados, ficam disponíveis para futuras consultas. Para ter acesso a estes dados, o usuário deve realizar login na plataforma web e selecionar um dispositivo cadastrado. Esta funcionalidade é demonstrada no estudo de caso do presente trabalho, no Capítulo 4.

3.2 MODELO RELACIONAL DOS DADOS

O modelo gerado é baseado na arquitetura proposta e nas funções da plataforma. A Figura 10 traz o modelo relacional utilizado no protótipo. Nesta imagem, constam as entidades, seus campos, os tipos dos campos e o relacionamento entre elas.

[7] PHP: https://secure.php.net/
[8] MySQL: https://www.mysql.com/

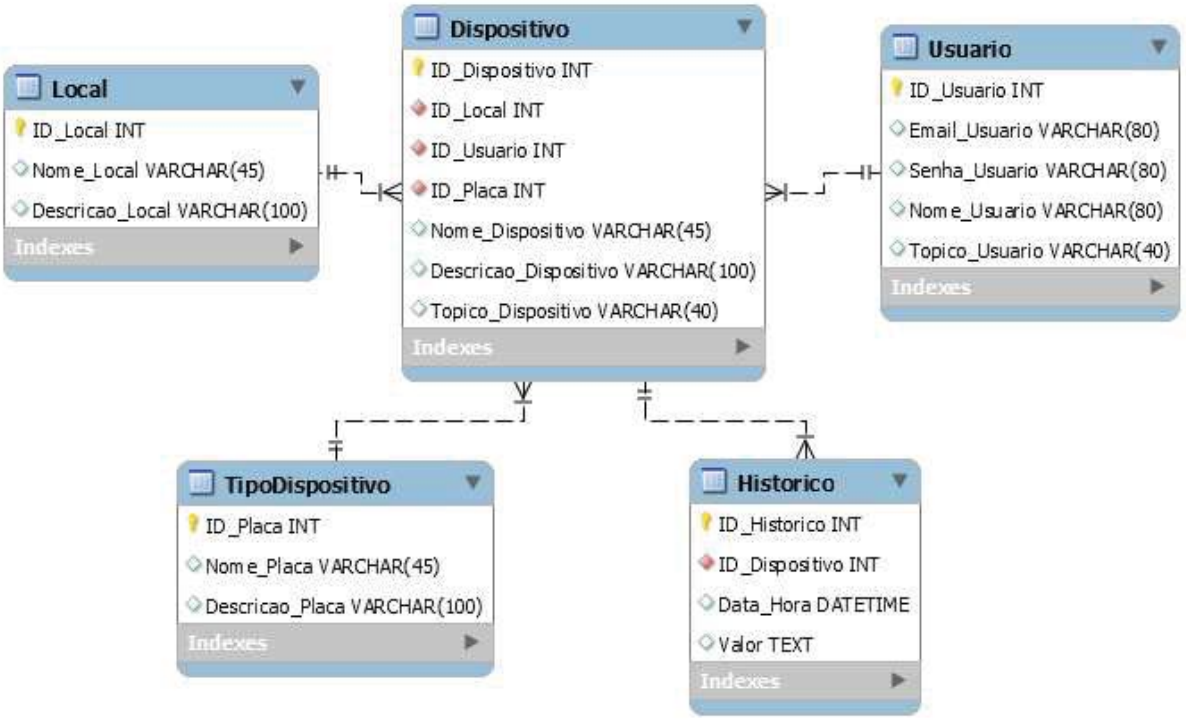


Figura 10. Modelo relacional do banco de dados.

Todo dispositivo cadastrado na plataforma está relacionado com um usuário, um local e tipo de dispositivo. A entidade *Historico* representa todas os dados enviados pelos dispositivos e armazenados no bando de dados.

A entidade *Usuario* representa os dados dos usuários na plataforma. Destaca-se o atributo *Topico_Usuario* desta entidade, responsável por armazenar um valor único na tabela para ser usado como tópico primário do *broker*.

Já na entidade *Dispositivo*, além dos atributos de identificação, nome e descrição, encontra-se o atributo *Topico_Dispositivo*, usado, assim como na entidade *Usuario*, para identificar um dispositivo no *broker*.

3.3 PLATAFORMA WEB

A plataforma desenvolvida pode ser vista por dois vieses. O primeiro é um *broker* MQTT, responsável por realizar a comunicação entre os dispositivos conectados e aplicações. O *broker* é formado pelo Eclipse Mosquitto configurado e rodando no mesmo servidor que a plataforma web e pode ser visto como o *middleware* de fato da plataforma. E o segundo é uma plataforma web, que possibilita a visualização, gerenciamento e envio de dados aos dispositivos.

Esta seção descreve a estrutura e funcionamento do sistema web desenvolvido.

Ao acessar o sistema utilizando um navegador web, a primeira funcionalidade a ser exibida é a interface de login, onde são exibidos os campos para validação de acesso. O usuário pode realizar o autocadastro no primeiro acesso ao sistema e, após, realizar login na plataforma.

Para realizar o cadastro na plataforma, o usuário deve informar o seu e-mail, nome e uma senha de acesso. Após a confirmação do cadastro, a plataforma gera uma chave única criptografada com o método SHA1, utilizada como tópico do usuário. Como toda mensagem enviada ao *broker* tem um tópico como destinatário, a criação de uma chave única criptografada utilizada como tópico do usuário ajuda a prevenir que mensagens indesejadas sejam publicadas.

Após a efetivação do login, o sistema direciona o usuário para a página principal, onde são exibidas todas as informações sobre o número de dispositivos cadastrados na plataforma, a quantidade de dados utilizados no banco de dados pelo usuário, se existem dispositivos conectados/disponíveis no momento e, principalmente, o ID do usuário criptografado, usado como tópico primário no *broker*.

No menu “Dispositivos” são exibidos todos os dispositivos cadastrados pelo usuário logado no sistema, como pode ser observado na Figura 11. Todos os dispositivos mostram o tipo de placa, nome, local instalado e status, que pode ser disponível ou indisponível. Ao acessar esta funcionalidade o servidor envia uma mensagem a cada dispositivo cadastrado e, se receber outra mensagem como resposta, informa que este está disponível, caso contrário, coloca o status como indisponível. Além disso, é possível interagir com cada um dos dispositivos cadastrados através as ações “Ver”, “Enviar”, “Editar” e “Deletar”.

Cabe ressaltar que é possível colocar vários sensores/atuadores em uma placa de prototipagem eletrônica (como Arduino e similares) e cadastrar cada sensor/atuador como um dispositivo na plataforma. Para cada um dos sensores/atuadores cadastrados é gerado um tópico a ser utilizado para enviar ou receber mensagens.

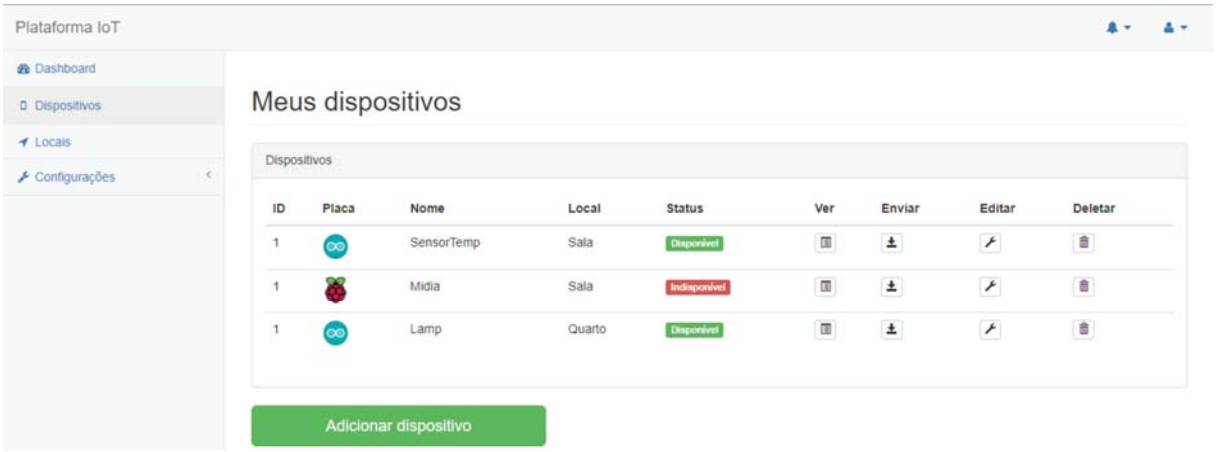
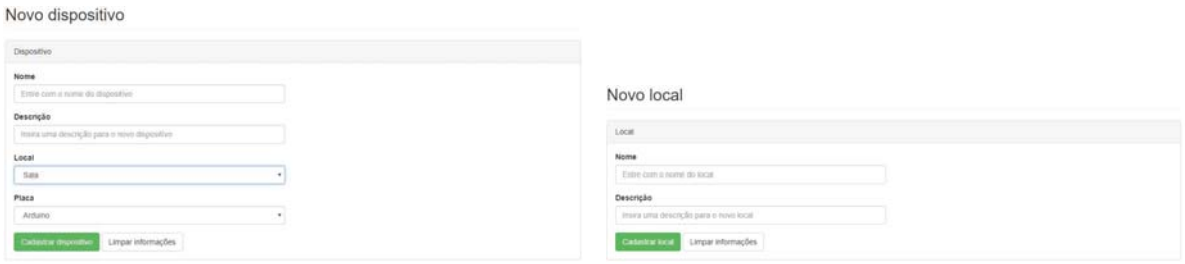


Figura 11. Plataforma Web - Tela dos dispositivos.

Para começar a utilizar a plataforma é necessário cadastrar um dispositivo (Figura 12(a)) e o seu respectivo local (Figura 12(b)). O local de um dispositivo indica a localização deste em um cômodo da casa, por exemplo. Para cadastrar o local basta informar o nome do local e uma descrição. Já para adicionar um dispositivo deve-se inserir seu nome e descrição, e selecionar o local e o tipo de dispositivo.



(a) Novo dispositivo. (b) Novo local.

Figura 12. Plataforma Web - Cadastros.

No processo de cadastro do dispositivo, o sistema gera uma chave única que será armazenada no banco de dados e utilizada como tópico no *broker*. Os desenvolvedores de aplicativos poderão utilizar esta chave como identificador do dispositivo para enviar e receber informações.

Para visualizar todos os locais cadastrados e seus respectivos atributos, basta selecionar o menu “Locais”. Além disso, é possível adicionar uma nova localização para os dispositivos, editar um local específico, ver todas suas informações e deletá-lo. A Figura 13 ilustra esta funcionalidade do sistema.

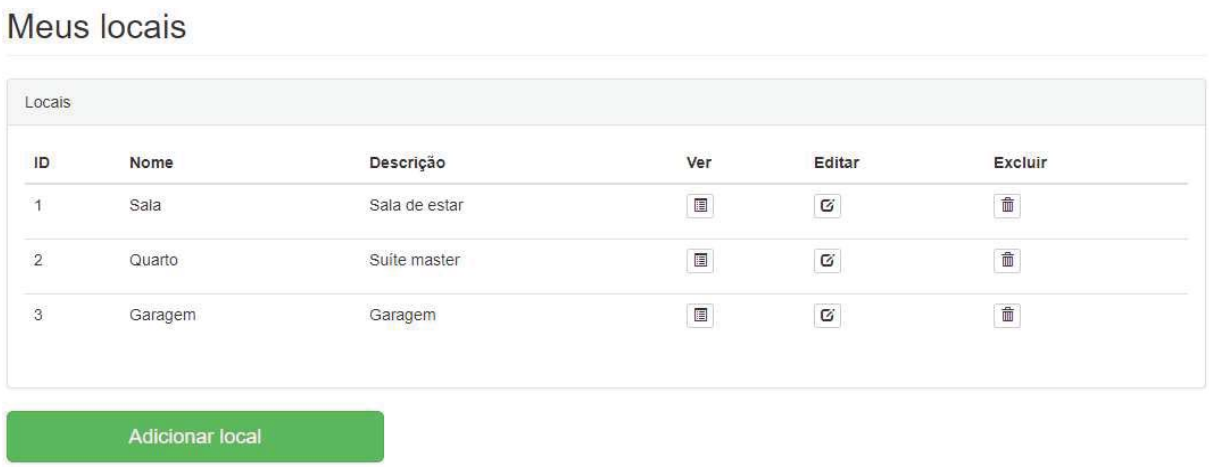


Figura 13. Plataforma Web - Meus locais.

Na funcionalidade “Enviar valor para dispositivo” (Figura 14), acessada no menu “Dispositivos” e selecionando a ação “Enviar”, o usuário da plataforma pode enviar um dado – do tipo texto – para um dispositivo. Este valor pode ser usado para gerenciar o dispositivo ou simplesmente como comando para realizar uma tarefa. Nesta interface também encontram-se todas as informações do dispositivo selecionado.

Por fim, na opção “Ver”, também acessada no menu “Dispositivos”, são apresentadas as informações sobre um dispositivo específico, como pode ser visto na Figura 15. Nesta tela são mostrados os dados enviados pelo dispositivo em tempo real e os dados históricos do mesmo. Podem ser visualizadas as 6 (seis) últimas mensagens enviadas pelo dispositivo ao servidor, com a data e hora que a informação foi recebida. Quanto aos dados históricos, são mostrados em uma tabela dinâmica,

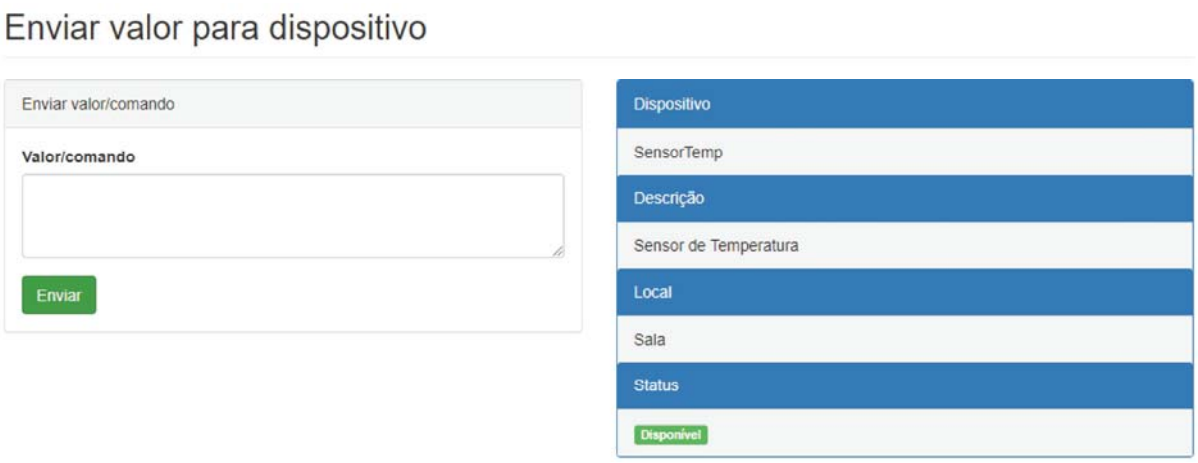


Figura 14. Plataforma Web - Enviar comando/valor.

podendo o usuário escolher a quantidade de registros por página e navegar pelas mesmas, além de poder filtrar o resultado da busca por data, hora ou valor.

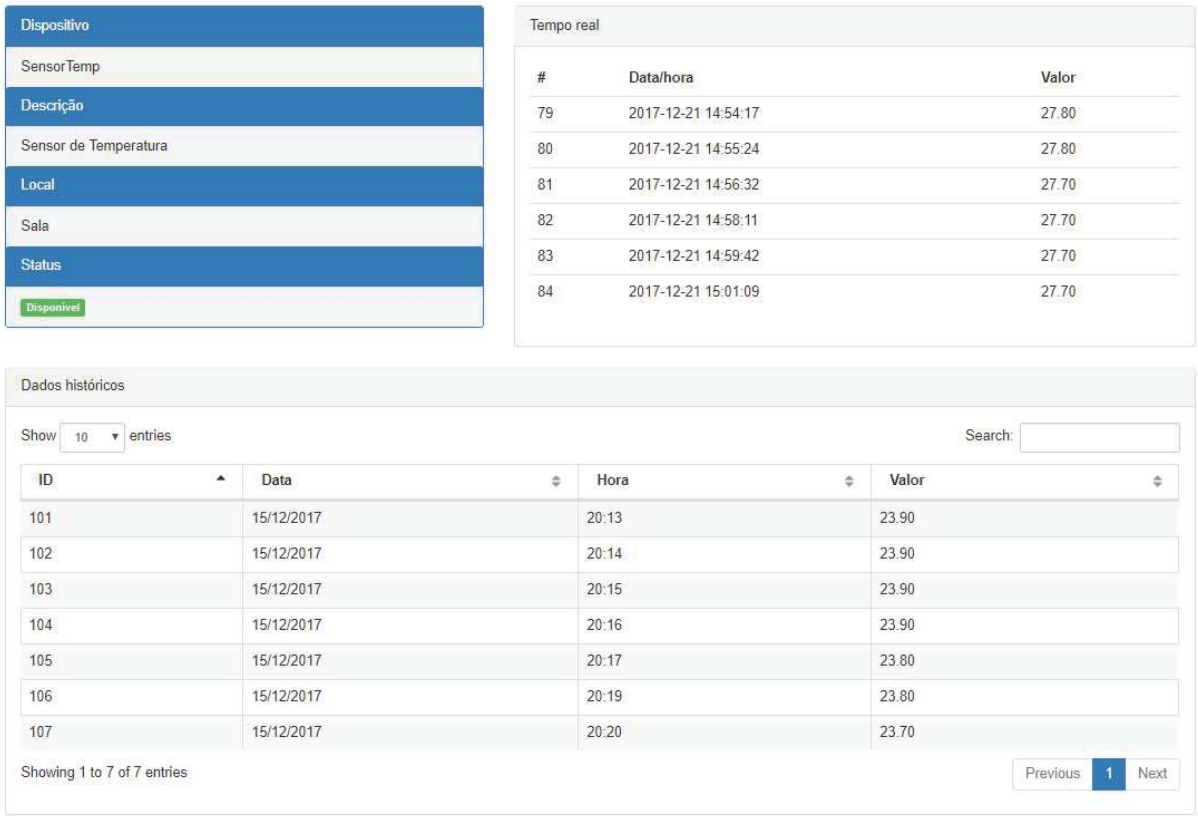


Figura 15. Plataforma Web - Informações recebidas do dispositivo.

A plataforma web utiliza a biblioteca PHP chamada phpMQTT[9] e Websockets[10], tornado-a capaz de monitorar e receber as mensagens enviadas pelo *broker* sem a necessidade de realizar requisições ao servidor, como ocorre normalmente com o modelo cliente-servidor que fazem uso do protocolo HTTP. Na Figura 16 pode-se observar o código em PHP responsável por conectar a plataforma

[9]phpMQTT: https://www.cloudmqtt.com/docs-php.html
[10]WebSocket: //developer.mozilla.org/pt-BR/docs/WebSockets

web no *broker* e publicar o comando 'L' no tópico "b911af807c2df88d671bd7004c54c1c2/sala/Led", sendo que "b911af807c2df88d671bd7004c54c1c2"é a identificação do usuário (única), "sala"o local e "Led"é o nome do dispositivo.

```
1.  $topico = "b911af807c2df88d671bd7004c54c1c2/sala/Led";
2.  $mensagem = 'L';
3.
4.  $c = new Mosquitto\Client;
5.  $c->onConnect(function() use ($c) {
6.    $c->publish($topico, $mensagem, 0);
7.    $c->disconnect();
8.  });
9.
10. $c->connect('192.168.5.1');
11. $c->loopForever();
12. echo '1';
```

Figura 16. Código para publicar mensagem no broker via PHP.

4. ESTUDO DE CASO

Este capítulo apresenta o estudo de caso desenvolvido com o objetivo de explorar funcionalidades propostas para a plataforma de *middleware*. As funcionalidades exploradas no estudo de caso contemplam tarefas relacionadas a comunicação entre as entidades da IoT, bem como a gerência de dispositivos e visualização de dados enviados por estes.

O estudo de caso foi desenvolvido utilizando a seguinte metodologia: foram conectados a plataforma um dispositivo IoT e um aplicativo móvel, a fim de demostrar que o *middleware* desenvolvido cumpre os requisitos mínimos para realizar a comunicação entre estes dois. Também são utilizadas as funcionalidades da plataforma web para enviar comandos ao dispositivo e mostrar informações enviados por este último.

O hardware utilizado no estudo de caso foi uma placa Arduino Uno, versão R3, conectado a uma placa Wifi Shield[11]. Foram utilizados 1 sensor de temperatura, modelo LM35[12], e um LED (*Light Emitting Diode*) padrão, como pode ser obervado na Figura 17. A escolha da placa Arduino foi decorrente da sua ampla utilização em projetos de IoT. Cabe ressaltar que proposta deste trabalho não inclui bibliotecas ou códigos para os dispositivos ou aplicativos, mas sim uma plataforma de *middleware*, ou seja, o *middleware* prove meios de conectar estas entidades da IoT de maneira simples, abstraindo detalhes de protocolos e configuração de *brokers*, e ainda disponibiliza uma plataforma para gerenciamento de dispositivos e visualizaçao de dados enviados por estes dispositivos.

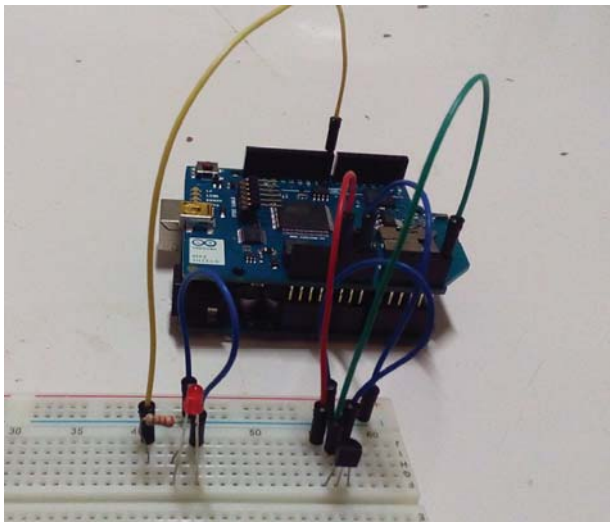


Figura 17. Hardware empregado no estudo de caso.

O estudo de caso foi executado em 4 cenários: (i) dispositivo envia mensagem para uma aplicação; (ii) uma aplicação envia um comando para o dispositivo; (iii) transmissão de um comando para um dispositivo via plataforma web; (iv) visualização de valores enviado pelo dispositivo na plataforma web.

[11] WifiShield Arduino: https://store.arduino.cc/usa/arduino-wifi-shield
[12] Sensor de temperatura LM35: https://www.arduinoecia.com.br/2013/02/lm35-sensor-de-temperatura.html

No primeiro cenário, o dispositivo (Arduino + Wifi Shield) envia a temperatura obtida do sensor LM35 para um aplicativo Android. A Figura 18 demonstra o código simplificado para conectar o Arduino ao servidor utilizando o protocolo MQTT. As informações necessárias para realizar a conexão são: endereço do servidor, tópico a ser publicado e o próprio valor (temperatura). O Arduino foi configurado para enviar a temperatura com frequência de 60 segundos.

```
#include<PubSubCliente.h>

#define TOPICO_PUBLISH "b911af807c2df88d671bd7004c54c1c2/sala/SensorTemp"

const char* BROKER_MQTT = "192.168.2.5";
int BROKER_PORT = 1883;

WiFiClient cliente;
PubSubClient mqtt(cliente); // instancia o mqtt

void setup() {
  mqtt.setServer(BROKER_MQTT, BROKER_PORT);
  mqtt.connect(BROKER_MQTT);
}

void loop() {
  float temperatura = (float(analogRead(A5))*5/(1023))/0.01;
  mqtt.publish(TOPICO_PUBLISH, temperatura);
  delay(60000);
}
```

Figura 18. Código Arduino para publicar no tópico.

A Figura 18 mostra a variável *TOPICO_PUBLISH*, responsável por armazenar o tópico em que as informações de temperatura serão publicados no broker MQTT. Percebe-se que ela é composta, na sequência e separados por um caractere '/', do tópico do usuário criptografado, localização e identificação do dispositivo. Também são configurados o endereço do servidor e porta. A instrução responsável por publicar a temperatura no servidor é:

mqtt.publish(TOPICO_PUBLISH, temperatura);

Já na Figura 19, percebe-se que os valores enviados pela placa Arduino são mostrados no aplicativo Android, desenvolvido com a plataforma App Inventor 2[13], mantida pelo MIT (*Massachusetts Institute of Technology*). O aplicativo também deve realizar a comunicação com servidor via protocolo MQTT e assinar o mesmo tópico do dispositivo.

Como ainda não dexiste um componente do App Inventor para trabalhar com o MQTT, foi necessário criar um mecanismo de interação entre o aplicativo e JavaScript, via componente WebViewString do App Inventor, ou seja, a comunicação com o *broker* MQTT é realizada por JavaScript. Para poder publicar ou receber as mensagens enviadas pelo *broker*, o aplicativo consulta o arquivo JavaScript armazenado no *smartphone* Android. Esta solução foi publicada pelo site InternetOfHomeThings [27]. Ora, o MQTT exige que uma conexão TCP permaneça aberta durante todo o tempo

[13]AppInventor: http://ai2.appinventor.mit.edu/

em que o aplicativo está sendo executado, enquanto o App Inventor fornece apenas uma interface HTTP.

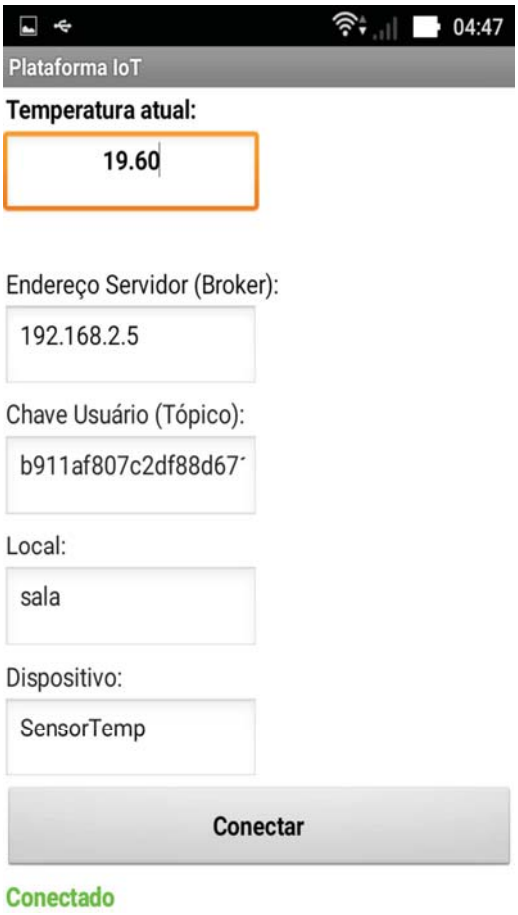


Figura 19. Aplicativo Android para leitura do sensor.

No segundo cenário, o sentido da comunicação é invertida, ou seja, o aplicativo Android envia um comando para o dispositivo Arduino ligar ou desligar o LED. A placa Arduino está programada para quando receber uma mensagem com o valor “L”, a mesma deve ligar o LED do experimento, e desligar quando o valor da mensagem for “D”. Este cenário é demostrado na Figura 20.

A Figura 21 mostra o código Arduino responsável por inscrever o dispositivo no *broker*, utilizando as mesmas credencias que o App Android. Nesta configuração, a constante responsável por guardar o tópico utilizado é *TOPICO_SUBSCRIBE*, com conteúdo idêntico a constante *TOPICO_PUBLISH*. Já a função *mqtt.subscribe(TOPICO_SUBSCRIBE)* inscreve o dispositivo no tópico definido de fato.

Da mesma forma que o primeiro modo, os dois atores desse cenário, aplicativo Android e placa Arduino, devem estar conectados a plataforma de middleware utilizando o protocolo MQTT.

Na terceira demonstração, uma informação/comando é enviada para o dispositivo Arduino utilizando a plataforma web. Este processo é semelhante ao caso anterior, com a diferença que o comando é enviado ao dispositivo Arduino partindo da plataforma web, e não do App Android. Neste caso, a plataforma já provê todos recursos para a conexão com o dispositivo, bastando cadastrá-lo na

```
void mqtt_callback(char* topic, byte* payload, unsigned int length)
{
    String msg;

    for(int i = 0; i < length; i++)
    {
       char c = (char)payload[i];
       msg += c;
    }

    if (msg.equals("L"))
    {
        digitalWrite(8, HIGH);
    }
    if (msg.equals("D"))
    {
        digitalWrite(8, LOW);
    }
}
```

Figura 20. Código Arduino - liga e desliga LED.

```
#include<PubSubCliente.h>
#define TOPICO_SUBSCRIBE "b911af807c2df88d671bd7004c54c1c2/sala/Led"

const char* BROKER_MQTT = "192.168.2.5";
int BROKER_PORT = 1883;

WiFiClient cliente;
PubSubClient mqtt(cliente); // instancia o mqtt

void setup() {
  mqtt.setServer(BROKER_MQTT, BROKER_PORT);
  mqtt.connect(BROKER_MQTT);
  mqtt.subscribe(TOPICO_SUBSCRIBE);
}
void loop() {
  mqtt.setCallback(mqtt_callback);
}
```

Figura 21. Código para inscrever Arduino no tópico.

platadorma. Sendo assim, não é necessáio programar ou configurar nada na platforma, tornando esta tarefa transparente para o usuário. A Figura 22 apresenta o valor sendo enviado na plataforma web.

Por fim, a Figura 23 apresenta, na página web, a temperatura enviada pelo Arduino. Na figura pode-se observar os valores em tempo real e os valores históricos armazenados no banco de dados, ordenados por data.

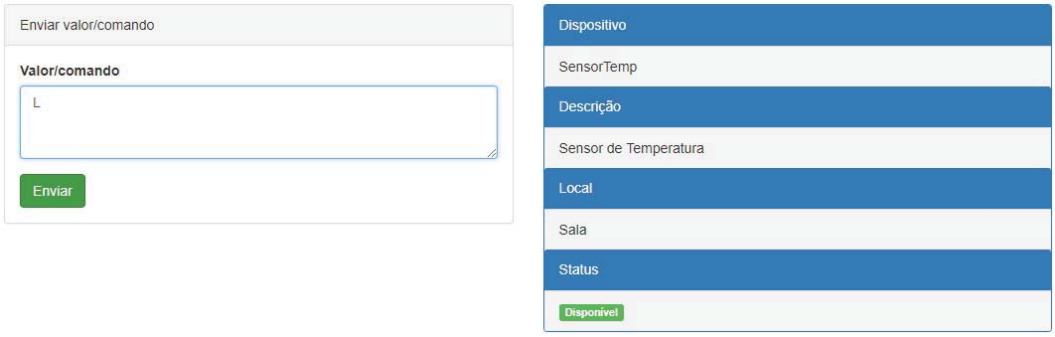


Figura 22. Enviar comando via plataforma Web.

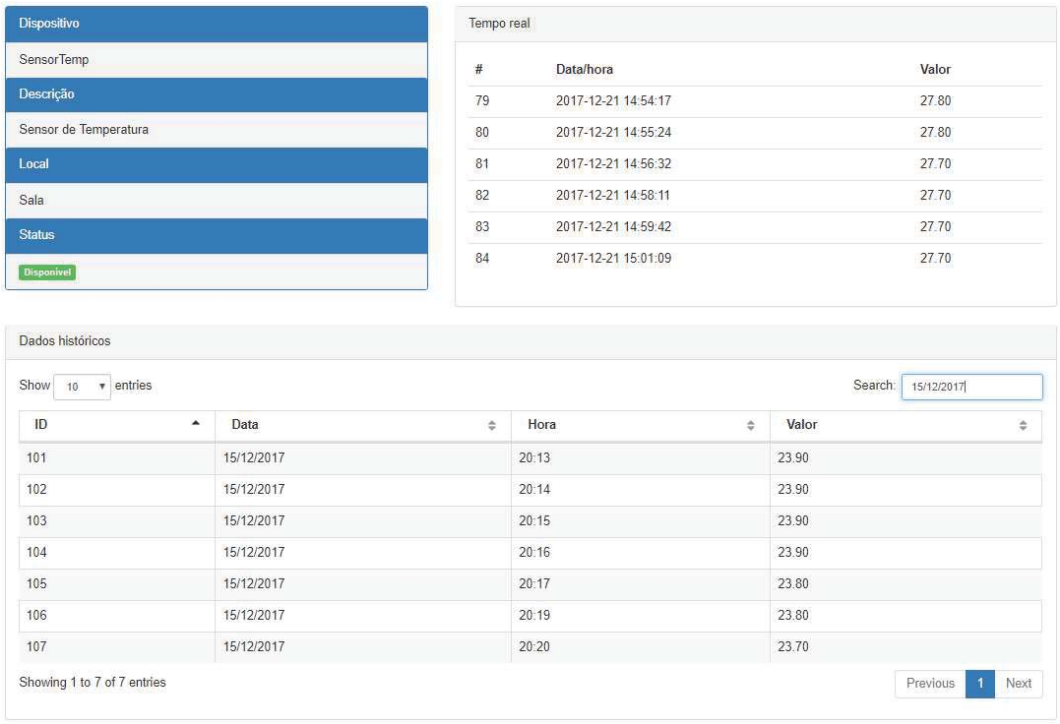


Figura 23. Visualizar valores do dispositivo via plataforma Web.

5. CONCLUSÃO

Este trabalho teve por objetivo demostrar o desenvolvimento de uma plataforma de *middleware* para Internet das Coisas *open source*, tema largamente explorado em pesquisas acadêmicas e corporativa. A plataforma é composta por um *broker* MQTT responsável por intermediar a troca de mensagens entre os dispositivos na IoT, e uma plataforma Web, com a função de gerenciar e visualizar informações trocadas por estes dispositivos. Um estudo de caso foi desenvolvido a fim de avaliar as funcionalidades da plataforma.

Nessa plataforma, foi proposto um padrão de comunicação simples e implementável, capaz de se adaptar a múltiplos cenários. Foi proposta uma solução que permite explorar, com experimentos reais, o ambiente da IoT e que pode ser adaptada, melhorada e estendida para assim prover um padrão que, de fato, se ajuste às necessidades que ainda serão descobertas para a IoT.

A metodologia utilizada neste trabalho, dividida em três fases, mostrou-se eficaz para o direcionamento da construção da arquitetura proposta e resolução do problema de pesquisa, que procurava conhecer quais os processos para construção de uma plataforma de *middleware* para IoT utilizando tecnologias *open source*. A revisão de literatura identifica o estado da arte sobre tecnologias envolvendo a IoT.

O *middleware* proposto, dividido em *broker* MQTT e plataforma Web, mostrou-se promissor e eficaz para controlar dispositivos e monitorar suas atividades. O estudo de cado demonstrou que a plataforma desenvolvida é funcional e de fácil entendimento, utilização e reconstrução. Com ele, foi possível averiguar e atestar a viabilidade de comunicação entre dispositivos da IoT utilizando um protocolo largamente utilizado na IoT.

Como proposta de trabalhos futuros são indicadas alguns pontos que podem ser evoluídos, testados e/ou implementados, destacam-se:

- Melhorias na interface da plataforma Web. A interface Web atual foi criada sem considerar critérios de usabilidade e acessibilidade;
- Criação de uma API (*Application Programming Interface*) para padronizar a troca de informações/mensagens entre os dispositivos e a plataforma;
- Adição de comandos via interface gráfica na plataforma Web para controle de dispositivos. Comandos podem ser adicionados na plataforma Web para realizarem ações específicas, como ligar ou desligar atuadores e ativar ou desativar um dispositivo;
- Gatilhos podem ser adicionados para disparar eventos automaticamente ao perceber mudanças de comportamento nos dispositivos conectados.

REFERÊNCIAS BIBLIOGRÁFICAS

[1] LEE, I.; LEE, K. The internet of things (iot): Applications, investments, and challenges for enterprises. *Business Horizons*, Elsevier Science Publishers B. V., v. 58, n. 4, p. 431–440, 2015.

[2] CISCO. *Internet das Coisas (IoT)*. 2016. Disponível em: <http://www.cisco.com/c/pt_br/solutions/internet-of-things/overview.html>. Acesso em: Set. 2017.

[3] CORPORATION, I. D. *Explosive Internet of Things Spending to Reach $1.7 Trillion in 2020*. 2015. Disponível em: <http://www.idc.com/getdoc.jsp?containerId=prUS25658015>. Acesso em: Set. 2017.

[4] RESEARCH, A. *The Internet of Things Will Drive Wireless Connected Devices to 40.9 Billion in 2020*. 2014. Disponível em: <https://www.abiresearch.com/press/the-internet-of-things-will-drive-wireless-connect/>. Acesso em: Set. 2017.

[5] ARDUINO. *What is Arduino?* 2017. Disponível em: <https://www.arduino.cc/en/Guide/Introduction>. Acesso em: Nov. 2017.

[6] LEITE, L. E. e. a. C. Flextv — uma proposta de arquitetura de middleware para o sistema brasileiro de tv digital. *Revista de Engenharia de Computação e Sistemas Digitais*, USP, São Paulo, 2006.

[7] RAZZAQUE, M. A. e. a. Middleware for internet of things: a survey. *IEEE Internet o f Things Journal*, IEEE, v. 3, n. 1, p. 70–95, 2016.

[8] ALMEIDA, H. Internet das coisas: tudo conectado. *Computação Brasil: Revista da Sociedade Brasileira de Computação*, n. 29, p. 6–8, 2015.

[9] BORGIA, E. The internet of things vision: Key features, applications and open issues. *Computer Communications*, Elsevier Science Publishers B. V., v. 54, n. 6, p. 1–31, 2014.

[10] OCAMPOS, T. Internet das coisas nas nuvens. *Computação Brasil: Revista da Sociedade Brasileira de Computação*, n. 29, p. 19–22, 2015.

[11] WORTMANN, F.; FLüCHTER, K. Internet of things: Technology and value added. *Business & Information Systems Engineering*, v. 57, n. 3, p. 221–224, 2015.

[12] TEIXEIRA, T. et al. Service oriented middleware for the internet of things: A perspective. In: PROCEEDINGS OF THE 4TH EUROPEAN CONFERENCE ON TOWARDS A SERVICE-BASED INTERNET. Berlin: Springer, 2011. v. 6994, p. 220–229.

[13] BANDYOPADHYAY, S. et al. Role of middleware for internet of things: A study. *International Journal of Computer Science & Engineering Survey*, v. 2, n. 3, p. 94–105, 2011.

[14] DELICATO, F. C.; PIRES, P.; BATISTA, T. *Middleware solutions for the Internet of Things*. London: Springer, 2013. 78 p.

[15] GUBBI, J. et al. Internet of things (iot): A vision, architectural elements, and future directions. *Future Generation Computer Systems*, v. 29, n. 8, p. 1645–1660, 2013.

[16] PIRES, P. F. et al. Plataformas para a internet das coisas. In: MARTINELLO, M.; ROBEIRO, M. R. N.; ROCHA, A. A. de A. (Ed.). *XXXIII Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*. Porto Alegre, RS, Brasil: SBC, 2015. p. 110–169.

[17] QIN, W. et al. Restthing: A restful web service infrastructure for mash-up physical and web resources. In: PROCEEDINGS OF THE 9TH IFIP INTERNATIONAL CONFERENCE ON EMBEDDED AND UBIQUITOUS COMPUTING. USA: IEEE, 2011. p. 197–204.

[18] GAO, L.; ZHANG, C.; SUN, L. Restful web of things api in sharing sensor data. In: PROCEEDINGS OF THE 2011 INTERNATIONAL CONFERENCE ON INTERNET TECHNOLOGY AND APPLICATIONS. USA: IEEE, 2011. p. 1–4.

[19] MQTT. *MQTT Documentation*. 2017. Disponível em: <http://mqtt.org/documentation>. Acesso em: Dez. 2017.

[20] PERERA, C. e. a. Sensing as a service model for smart cities supported by internet of things. *Transactions on Emerging Telecommunications Technologies*, Wiley Online Library, v. 25, n. 1, p. 81–93, 2014.

[21] YUAN, M. *Conhecendo o MQTT*. 2017. Disponível em: <https://www.ibm.com/developerworks/br/library/iot-mqtt-why-good-for-iot/index.html>. Acesso em: Dez. 2017.

[22] BARROS, M. *MQTT - Protocolos para IoT*. 2015. Disponível em: <https://www.embarcados.com.br/mqtt-protocolos-para-iot/>. Acesso em: Jan. 2018.

[23] LIGHT, R. *Mosquitto Documentation*. 2017. Disponível em: <https://mosquitto.org/man/mosquitto-8.html>. Acesso em: Dez. 2017.

[24] MOBILE, E. *Elipse Mobile e ArduinoCloudLink*. 2016. Disponível em: <http://www.elipsemobile.com/>. Acesso em: Abr. 2016.

[25] SOLDATOS, J.; SERRANO, M.; HAUSWIRTH, M. Convergence of utility computing with the internet-of-things. In: PROCEEDINGS OF THE 6TH INTERNATIONAL CONFERENCE ON INNOVATIVE MOBILE AND INTERNET SERVICES IN UBIQUITOUS COMPUTING. USA: IEEE, 2012. p. 874–879.

[26] YALER. *Yaler*. 2016. Disponível em: <https://yaler.net/>. Acesso em: Abr. 2016.

[27] INTERNETOFHOMETHINGS. *MQTT For App Inventor*. 2016. Disponível em: <https://internetofhomethings.com/homethings/?p=1317>. Acesso em: Jan. 2018.