DAS Departamento de Automação e Sistemas
CTC Centro Tecnológico
UFSC Universidade Federal de Santa Catarina

Arquiteturas para Interoperabilidade em Verticais de IoT

Relatório submetido à Universidade Federal de Santa Catarina
como requisito para a aprovação da disciplina:
DAS 5511: Projeto de Fim de Curso

Henrique Peixe Maziero

Florianópolis, Fevereiro de 2019

Arquiteturas para Interoperabilidade em Verticais de IoT

Henrique Peixe Maziero

Esta monografia foi julgada no contexto da disciplina

DAS 5511: Projeto de Fim de Curso

e aprovada na sua forma final pelo

Curso de Engenharia de Controle e Automação

Prof. Leandro Buss Becker

Banca Examinadora:

Eduardo Schwinden Leal
Orientador na Empresa

Prof. Leandro Buss Becker
Orientador no Curso

Prof. Hector Bessa Silveira
Responsável pela disciplina

Prof. Felipe Gomes de Oliveira Cabral, Avaliador

Nilmar Luiz Guarda Junior, Debatedor

Luiz Arthur D'Ávila da Silva Prazeres, Debatedor

Dedico este trabalho à busca pelos meus sonhos, e às buscas de todos aqueles que sonham com um mundo melhor através da inovação.

Agradecimentos

Agradeço a meus pais, Carlos e Lúcia, por todos esses anos de carinho e suporte, por tudo o que me ensinaram e por terem me mostrado o mundo.

Agradeço a minha família, os Peixe e os Maziero, pelo amor e pelo refúgio que sempre encontro em sua presença.

Agradeço aos amigos, próximos e mesmo os não mais, pelos momentos de diversão e pela ajuda durante toda a faculdade.

Agradeço aos professores do curso de Engenharia de Controle e Automação, pelas lições, e também pela simpatia.

```
 ________________________________________
/ Writing is easy; all you do is sit     \
| staring at the blank sheet of paper    |
| until drops of blood form on your      |
| forehead.                              |
|                                        |
\ -- Gene Fowler                         /
 ----------------------------------------
        \   ^__^
         \  (oo)\_______
            (__)\       )\/\
                ||----w |
                ||     ||
```

fortune | cowsay

Resumo

O advento da Internet das Coisas (IoT) ainda não foi capaz de estabelecer um protocolo único que se adequasse a todos os casos de uso. O surgimento de diversas novas tecnologias de comunicação e aplicações impulsionou a adoção de diferentes protocolos nos vários setores da internet das coisas. O que se observa atualmente é uma grande fragmentação de protocolos entre aplicações pertencentes a diferentes verticais ainda que no mesmo setor, freando quaisquer tentativas de integração horizontal entre dispositivos e entre plataformas devido à não-padronização desses. A empresa Sensorweb tem interesse em realizar a integração de dispositivos de telemetria de diversos fabricantes para ampliar o seu catálogo de produtos oferecidos ao cliente. O objetivo deste trabalho consiste em uma análise dos protocolos mais utilizados para a comunicação entre dispositivos e plataformas em soluções atuais de IoT com ênfase em aplicações de telemetria, seguido de uma análise e implementação das soluções para a integração de uma gama seleta de dispositivos de empresas parceiras que utilizam diferentes protocolos. Dentre estas soluções foi realizado o desenvolvimento de uma camada da *middleware*, assim como o desenvolvimento de um módulo MQTT que foi incluído no *software SCADA* utilizado e mantido pela empresa.

Palavras-chave: Internet das coisas, Telemetria, Protocolos, Integração, Interoperabilidade, MQTT, SCADA.

Abstract

The advent of the Internet of Things (IoT) has not been able to establish a unified protocol that is suitable for all use cases just yet. The emergence of several new communication technologies and new applications pushed the adoption of different protocols on the several sectors of the IoT universe. What we see now is a tremendous fragmentation of the protocols found in applications even belonging to different verticals even in the same sector, slowing down any attempts of horizontal integration among devices and platforms due to the lack of standards. The company Sensorweb seeks to perform the integration of telemetry devices from several manufacturers, broadening the options it can provide to its clients. The main objective of this work consisted of an analysis of the most popular protocols used for communication between devices and platforms in modern solutions for IOT with emphasis on telemetry, followed by the analysis and implementation of the solutions needed for the integration of selected devices from partnered companies that used these protocols. Among these solutions, a middleware layer was implemented, alongside with the development of a MQTT module to be included in the SCADA software that is used and maintained by the company.

Keywords: Internet of Things, Telemetry, Protocols, Integration, Interoperability, MQTT, SCADA.

Lista de ilustrações

Lista de tabelas

Lista de Blocos de Código

Lista de abreviaturas e siglas

API – Application Programming Interface

AWS – Amazon Web Services

CLI – Command Line Interface

CoAP – Constrained Application Protocol

DWR – Direct Web Remoting

EIP – Enterprise Integration Patterns

HTTP – HyperText Transfer Protocol

HTTPS – HTTP Secure

IETF – Internet Engineering Task Force

IP – Internet Protocol

IoT – Internet of Things

JSON – JavaScript Object Notation

M2M – Machine to Machine

MOM – Message Oriented Middleware

MQTT – Message Queue Telemetry Transport

OEM – Original Equipment Manufacturer

REST – Representational State Transfer

RISC – Reduced Instruction Set Computer

RTOS – Real-Time Operating System

SCADA – Supervisory Control and Data Acquisition

SDK – Source Development Kit

TLS – Transport Layer Security

URL – Uniform Resource Locator

Sumário

1 Introdução

1.1 Linhas Gerais

O Projeto de Fim de Curso aqui apresentado foi realizado entre Agosto e Dezembro de 2018 na empresa Sensorweb em Florianópolis. A proposta de projeto escolhida consistia em atividades relacionadas à atualização tecnológica da plataforma de Internet das Coisas (em inglês *Internet of Things* ou IoT) da Sensorweb em vista da busca por interoperabilidade com mais fornecedores de hardware.

Para tal fim, realizou-se um estudo técnico dirigido da solução atualmente utilizada na empresa, seguido de um estudo comparativo de soluções de IoT semelhantes comerciais. Em seguida, foi realizada uma avaliação técnica de protocolos de comunicação utilizados em IoT, e a integração de dispositivos conectados via tais protocolos com a plataforma de monitoramento da empresa. Por fim, as soluções propostas e implementadas foram avaliadas em custo de funcionamento, performance e escalabilidade.

1.2 A Empresa Sensorweb

A empresa Sensorweb fornece um serviço de telemetria e registro de temperatura para instalações da área de saúde com refrigeradores, ambientes controlados ou outros sistemas de refrigeração, com o propósito de permitir o monitoramento da temperatura de armazenamento de medicamentos, vacinas, sangue etc.

Para tal fim, a empresa instala sondas, transmissores e *gateways* (aglomeradores) nos locais monitorados, que realizam o envio dos dados de telemetria para um serviço (*back-end*) em nuvem. Os dados monitorados chegam em servidores que executam um programa de monitoramento do tipo SCADA, onde os dados são recebidos, armazenados, geram alarmes e ficam disponíveis para a visualização em uma interface web acessível ao cliente. Uma visão dos componentes deste sistema é apresentada na Figura 1.

O serviço oferecido pela empresa engloba tanto a instalação e manutenção dos dispositivos em campo como a plataforma para o acesso aos dados coletados e o treinamento para a sua utilização. O valor agregado por este serviço se traduz na concordância com as normas para o monitoramento de armazenamento de materiais sensíveis, na detecção de falhas em sistemas de refrigeração, na substituição de métodos manuais mais propensos ao erro, e na prevenção da perda de insumos sensíveis a temperaturas incorretas de armazenagem.

Figura 1 – Transmissores, gateways e a aplicação da Sensorweb.

1.3 A Internet das Coisas

O universo da IoT engloba e traz inovação a inúmeros outros setores e aplicações, sendo um setor que atualmente movimenta bilhões de dólares anualmente [1]. A adoção de soluções de IoT tem permitido a muitas industrias aumentar suas margens de lucro devido a uma melhor gestão e eficiência graças aos dados coletados [1], tomando parte significativa na revolução conhecida como Indústria 4.0.

O que se entende por IoT constitui na verdade um conjunto bastante heterogêneo de aplicações e soluções, pertencente a diferentes setores, e com maiores ou menores graus de complexidade e inovação tecnológica, fazendo uso dos avanços em tecnologias de comunicação. Nestas soluções podem estar presentes dispositivos inteligentes ou não tão inteligentes, capazes de se comunicar entre si ou somente com seu sistema supervisório [2].

1.3.1 Redes de sensores

Uma das áreas de maior avanço recente na IoT são redes de sensores, graças aos avanços em microcontroladores de baixo custo e sistemas de comunicação de baixa energia. O uso de sensores comunicantes permite o monitoramento das mais diversas grandezas para as mais diversas aplicações [1].

O uso de redes de sensores constitui um caso moderno de aplicações de telemetria, onde o fluxo de informação é majoritariamente assimétrico, havendo a distinção clara entre produtores e consumidores de informação. Este caso de uso específico levou ao surgimento de diversos protocolos de comunicação de baixa energia classificados como *não-ip*, onde os dispositivos se conectam a uma rede local e não são diretamente endereçáveis a partir da

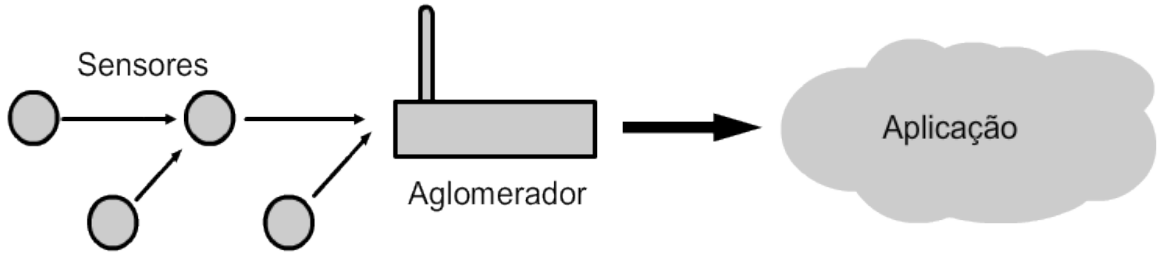

Figura 2 – Exemplo de topologia de uma rede de sensores.

internet [3], todas as suas comunicações passando por um mediador (gateway/agregador), como ilustrado em Figura 2.

1.3.2 Monitoramento na Cadeia do Frio e na Saúde

Na área da saúde, são vários os insumos que devem ter sua temperatura monitorada e controlada desde a produção até o seu consumo efetivo. Isto se aplica a remédios, vacinas, bolsas de sangue, materiais de pesquisa médica entre outros.

A RDC 17/2010 [4], artigo 117, define que: *As áreas de armazenamento devem ser projetadas ou adaptadas para assegurar as condições ideais de estocagem; devem ser limpas, secas, organizadas e mantidas dentro de limites de temperatura compatíveis com os materiais armazenados. Parágrafo único. Nos casos em que forem necessárias condições especiais de armazenamento, tais como temperatura e umidade, essas devem ser providenciadas, controladas, monitoradas e registradas.*

Tipicamente tal controle de temperatura é feito através de uma anotação das leituras de termômetros de máxima e mínima instalados nos ambientes de armazenamento, geladeiras, freezers etc. Esta atividade é feita periodicamente por funcionários e de maneira manual, sendo propensa a falhas, atrasos e falsificações, além de apresentar uma grande latência para qualquer possível detecção de problemas.

No transporte de medicamentos, diferentemente, é comum serem utilizados *data-loggers* portáteis alojados junto aos insumos para um registro contínuo da sua temperatura durante o trajeto e manuseio. Neste caso, os data-loggers devem ser posteriormente recuperados, seus dados extraídos, e por fim (nos casos mais comuns) descartados devido ao fim da sua vida útil.

Outro caso mais interessante são soluções de telemetria para o monitoramento. Nestes sistemas, são instalados sensores de temperatura e transmissores com ou sem fio para o monitoramento remoto em tempo real dos dado. O investimento para tais instalações depende da complexidade dos sensores e transmissores, assim como da necessidade de cabeamento para a alimentação e transmissão de dados.

1.3.3 Fragmentação de protocolos na IoT

O advento de novas tecnologias de comunicação de baixa energia e redes-não-ip fragmentou o universo IoT devido ao surgimento de protocolos específicos aos novos tipos de rede sendo implementados, como foi o caso para redes IEEE 802.15.4, nos quais protocolos como ZigBee e 6LoWPAN surgidos recentemente co-existiram de forma não interoperável.

A falta de uma escolha clara que pudesse atender simultaneamente a todas as necessidades de diversas aplicações de IoT levou à fragmentação e à utilização de diversos protocolos em diferentes aplicações. A não-interoperabilidade dos protocolos de comunicação se propagou na forma de protocolos de aplicação específicos e também não-interoperáveis.

Esta fragmentação de protocolos e isolamento de verticais por fim tem trazido dificuldades a uma integração horizontal entre soluções IoT devido à utilização de diferentes protocolos e modelos de aplicação que nem sempre podem ser diretamente traduzidos ou interfaceados [2].

Diversos dos protocolos M2M até então utilizados em sistemas distribuídos se provaram inadequados ao ambiente de pequenos dispositivos embarcados (things) com baixo poder de processamento, restrições de memória e energia [5], enquanto soluções M2M da área de telecomunicações obtiveram pouca popularidade devido às restrições impostas por suas patentes [6].

Esforços recentes de estandardização existem e propõem soluções relativamente versáteis e adequadas a diversos casos de uso presentes na IoT, como é o caso do conjunto de protocolos *CoAP/RPL/6LoWPAN/IEEE 802.15.4* proposto em [7]. Ha porém uma grande inércia para a sua adoção, devido ao uso dos demais protocolos já estabelecidos.

1.4 Problemática e Objetivos

1.4.1 Planejamento estratégico

A empresa busca expandir o seu catálogo de dispositivos que podem ser instalados *in situ* para o monitoramento de temperatura, por motivo do interesse de se ter múltiplos fornecedores possíveis, novas tecnologias de medição, menor preço, melhor qualidade etc.

A empresa atualmente está realizando diversas parcerias com produtores de hardware para a diversificação dos seus dispositivos instalados, buscando tanto soluções mais robustas para casos de uso já implantados, como protótipos para novos ramos do mercado da Cadeia do Frio e Hospitalar. No momento deste relatório, as parcerias são feitas tais que as especificações de funcionamento e de protocolo são estabelecidas pela Sensorweb

para serem efetuadas pela empresa parceira, de forma muito "próxima e íntima".

1.4.2 Integração

Para ser capaz de utilizar determinado dispositivo, deve ser feita a integração do mesmo com o sistema de supervisão e monitoramento da empresa, conhecido como Endrixx.

Para o uso de serviços em nuvem, como é o caso do sistema da Sensorweb, a única tecnologia de comunicação aceitável é a internet baseada em IP, portanto será considerada somente a integração de dispositivos capazes de se comunicar pela rede. Isto inclui dispositivos que se comuniquem diretamente pela internet, ou que se comuniquem através de agregadores/gateways conectados à mesma.

Para que um dado dispositivo possa ser integrado ao serviço de forma direta, ele deve se comunicar através de um dos protocolos disponíveis para a recepção de dados.

Outras arquiteturas possíveis são o uso de "*middleware*" capazes de interfacear e traduzir entre diferente protocolos, ou o uso de plataformas específicas do fabricante para a interface com os dispositivos e posterior coleta dos dados.

Uma análise mais aprofundada destas alternativas será apresentada no capítulo de fundamentação teórica.

1.4.3 Objetivo e Escopo

Como objetivo principal deste trabalho está a integração do maior número possível de dispositivos de telemetria de fabricantes parceiros e terceiros ou OEMs, através do desenvolvimento inteligente de arquiteturas e soluções repetíveis e escaláveis para a interoperabilidade.

Para tal fim, está previsto:

- Estudo de grandes serviços de Plataforma e Infraestrutura para a IoT com funcionalidade semelhante ao Endrixx, para melhor entendimento de outras soluções utilizadas no mercado.
- Estudo das propostas de arquiteturas de telemetria e sistemas supervisórios segundo a literatura, limitando-se à comunicação entre dispositivo e plataforma via internet.
- Estudo dos protocolos de comunicação utilizados por fornecedores de dispositivos de telemetria parceiros e suas arquiteturas propostas.
- Avaliação técnica do protocolo MQTT para a integração de sensores de propósito geral, e a sua abrangência e adoção dentre as arquiteturas estudadas e os dispositivos de empresas parceiras.

- Implementação do protocolo MQTT na plataforma Endrixx de forma a englobar o maior número de casos de uso possível com um esforço finito.
- Implementação de outras soluções de interoperabilidade consideradas relevantes para o objetivo da empresa.

2 Fundamentação e Pesquisa

Para os fins deste trabalho, o sistema de monitoramento Sensorweb foi cuidadosamente estudado em todos os seus níveis para entender em detalhe o funcionamento do serviço como um todo.

As etapas seguintes do trabalho realizado consistiram em um estudo de soluções já existentes para interoperabilidade, e a identificação de tendências observadas nas soluções mais utilizadas. Foi realizado o estudo de plataformas destinadas à integração de soluções IoT como candidatas para a substituição de parte do software atualmente utilizado pela empresa.

Foi em seguida estudado o conjunto de protocolos e soluções de comunicação que mais são utilizados no universo da IoT segundo a literatura, de um ponto de vista do caso de uso da telemetria, considerando sua flexibilidade e facilidade de implementação.

Foram então estudados os dispositivos de maior interesse a serem incluídos no catálogo da empresa, consistindo sobretudo dos dispositivos originais de empresas em parcerias atuais com a Sensorweb.

Por fim, dada as tendências observadas nas análises precedentes, realizou-se um estudo com ênfase no protocolo MQTT e em particular com payloads de tipo JSON, tendo em vista o âmbito da sua utilização em aplicações de Telemetria.

Boa parte dos estudos apresentados neste capítulo foram feitos em paralelo com as atividades de implementação, assim justificando, guiando e ocasionalmente redirecionando os esforços durante a realização das atividades.

2.1 Sistema de monitoramento Sensorweb

Como apresentado em 1.2, a empresa Sensorweb fornece um serviço de telemetria voltado para a área da saúde. Este serviço abrange vários aspectos de um sistema de monitoramento, como a instalação de sondas e transmissores de baica energia, o uso de aglomeradores, a transmissão e recepção dos dados em uma plataforma online, o processamento destes dados e disparo de alertas, e por fim a disponibilização das informações em uma interface web.

2.1.1 Dispositivos em campo

Atualmente a Sensorweb utiliza quase exclusivamente dispositivos em campo que foram projetados e produzidos em parceria com a empresa Traceback. Estes dispositivos

são alimentados a bateria e a ele são conectadas sondas instaladas dentro dos refrigeradores, ou pequenas sondas de temperatura e/ou umidade ambiente.

Estes dispositivos fazem a coleta de dados das sondas instaladas de forma periódica com longas pausas, produzindo tipicamente uma amostra a cada poucos minutos conforme sua configuração. Graças a este funcionamento periódico, e ao protocolo de rádio de baixa energia utilizado, a duração de vida das baterias é de cerca de dois anos.

Os dados coletados pelo dispositivo são enviados diretamente ou através de repetidores especializados até uma central/aglomerador/gateway (sinônimos). No caso de falha de comunicação, os dados são armazenados no dispositivo ou repetidor para serem posteriormente re-enviados.

2.1.2 Centrais aglomeradoras

As centrais são instaladas próximas à localização dos sensores, são alimentadas pela rede elétrica e conectadas à internet, e ficam à escuta de dados enviados pelos dispositivos pela rede sem fio de baixa energia. Dada a natureza não-ip e a não-bidirecionalidade dos dispositivos instalados, não ha comunicação direta entre os dispositivos e a internet. Todas as transações de informação com os servidores do serviço são efetuadas pela central.

As centrais contam com uma bateria para garantir o seu funcionamento por algumas horas no caso de uma queda de energia, e também com um modem GPRS (celular) para ter acesso à internet no caso de queda da rede local. As centrais recebem os dados de vários dispositivos e formam um pacote que é enviado para os servidores que executam o programa de Supervisão e Aquisição de Dados (SCADA). No caso de falha de envio, os dados são armazenados para serem enviados posteriormente.

2.1.3 Envio dos dados do monitoramento

Os dados enviados pelos dispositivos para as centrais são tipicamente as leituras das sondas, as leituras de qualquer sensor interno, a leitura de tensão da bateria, e um indicador da qualidade do sinal da interface de radio.

O envio de dados é feito periodicamente através de uma requisição HTTP-POST com um conteúdo em formato *application/x-www-form-urlencoded*, isto é, em pares chave-valor para cada amostra ou dado. Cada envio contém os valores lidos por cada um dos transmissores conectados a este aglomerador desde o ultimo envio corretamente executado.

2.1.4 Software de Supervisão e Aquisição de Dados

O processamento dos dados é feito em instâncias de máquinas virtuais hospedadas nos serviços de Nuvem da Amazon. As conexões são recebidas em um proxy reverso, e

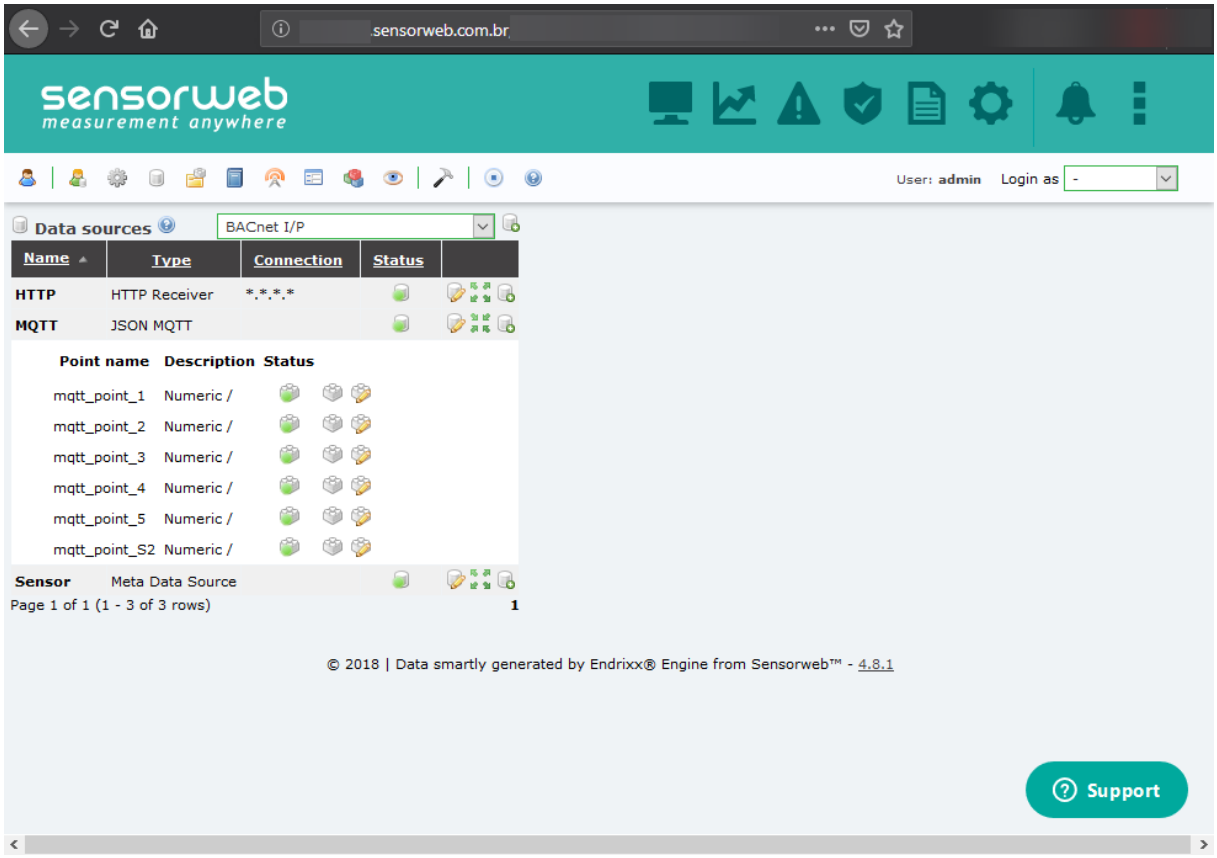

Figura 3 – Página para a configuração de Data Sources.

as requisições HTTP chegam em várias máquinas executando cada uma várias instâncias separadas do software supervisório, de acordo com o cliente ao qual corresponde o serviço.

O software de supervisão e monitoramento utilizado na empresa, apelidado Endrixx, é um sistema do tipo SCADA, construído majoritariamente sobre padrões abertos de interoperabilidade e bibliotecas open-source. Este software tem suporte a vários tipos de entradas de dados de dispositivos, como Modbus, Modbus/TCP, OneWire, RS232, Serial, HTTP-POST e HTTP-GET e outros.

Seus módulos de entrada de dados são denominados *Data Sources*, e podem ser instanciados múltiplas vezes para receber dados de diferentes dispositivos. Cada Data Source pode então ser populado com Data Points, que recebem os dados contidos na mensagem enviada pelo dispositivo, fazem o registro dos dados e disparam alarmes e eventos conforme o que for configurado. Um exemplo da página para a configuração de Data Sources é apresentado em Figura 3.

Por fim, os dados coletados são armazenados em bases de dados, disponíveis para consulta via web pelo cliente ou pela equipe de suporte. O Endrixx conta com uma interface web onde as leituras de cada sensor são exibidas, sendo possível visualizar a temperatura dos pontos monitorados através do navegador em uma interface amigável.

O sistema conta com alarmes configuráveis, disparo de emails e SMSs, gráficos, relatórios e funções de auditoria, tarefas típicas para um sistema SCADA.

2.2 Estudo de Plataformas para IoT

Uma das primeiras etapas deste projeto foi o estudo de plataformas destinadas ao IoT em comparação com o software SCADA atualmente utilizado na empresa.

O estudo destas plataformas serviu para identificar as funcionalidades em plataformas modernas de IoT, e julgar a possibilidade de serem utilizadas para substituir a plataforma existente da empresa.

Aqui é apresentada uma visão geral do funcionamento e funcionalidades das plataformas estudadas, enquanto no capítulo 3 esta análise é aprofundada por um teste prático de utilização da plataforma.

2.2.1 Amazon

A plataforma Amazon dispõe de diversos serviços que podem ser utilizados para soluções de IoT. Dentre estes, os de maior interesse estão listados a seguir:

2.2.1.1 Amazon FreeRTOS

Recentemente patronado pela Amazon, o já estabelecido e conhecido FreeRTOS tem ganhado muito interesse e suporte devido a sua leveza e portabilidade, adequados a ser instalado em edge-devices e nós IoT de baixa energia.

2.2.1.2 AWS GreenGrass

O GreenGrass é um framework de software com bibliotecas que permitem a integração de serviços da AWS em dispositivos de edge-processing, garantindo funcionalidades como o AWS Lambda em baixa latência e com tráfego de dados reduzido.

2.2.1.3 Amazon IoT Core

O AWS IoT Core é uma plataforma gerenciada na qual podem ser conectados dispositivos e aplicações para uma integração fácil e interação com os demais serviços da Amazon. Os serviços fornecidos englobam a recepção e o redirecionamento dos dados a outros serviços, não havendo uma armazenagem ou portal/interface para acesso aos dados. Uma visão geral do conjunto de ferramentas é oferecida em Figura 4

- **Device Gateway** O Device Gateway é o principal ponto de entrada para conexões ao IoT Core, suportando os protocolos MQTT, WebSockets e HTTP. Ele funciona

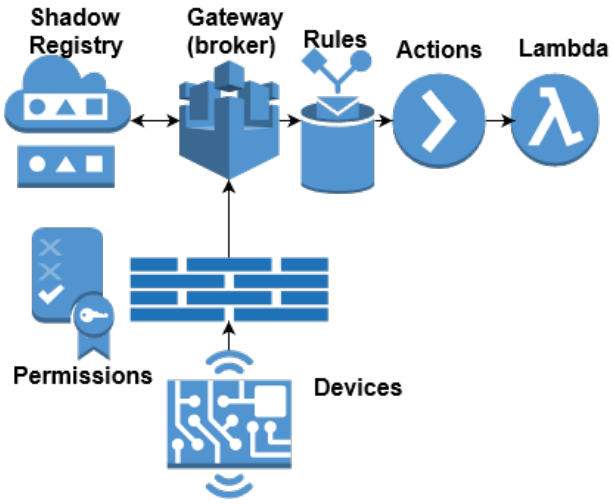

Figura 4 – Visão geral do conjunto IoT Core.

como um Broker MQTT de alto desempenho, mas com limitações no que diz respeito ao QoS das mensagens, sessões persistentes e retenção de mensagens. As conexões são aceitas somente em canal seguro (TLS) e perante autenticação do dispositivo, segundo o protocolo utilizado, sendo para o MQTT necessária uma autenticação mútua por certificados tipo x.509.

- **Registry** O Registro é um gerenciador de dispositivo que armazena metadados dos dispositivos registrados no sistema, os certificados de autenticação associados a estes dispositivos e os conjuntos de permissões associadas a estes certificados.

- **Device Shadow** Dada a falta de persistência de mensagem e sessão no broker utilizado pelo Device Gateway, foi implementada uma funcionalidade de armazenamento de mensagens através de uma API denominada Device Shadow, que armazena informações de estado enviadas do dispositivo ou para o dispositivo.

- **Rules Engine** A Rules Engine permite que ações sejam efetuadas segundo o tópico ou conteúdo de mensagens que chegam no sistema, permitindo a utilização de diversos serviços da AWS como armazenamento em base de dados, disparo de funções Lambda, ou até mesmo o envio de mensagens MQTT de resposta para os dispositivos pelo Device Gateway.

O AWS IoT Core conta com diversas SDKs para permitir uma integração rápida com dispositivos embarcados em várias linguagens de programação.

Enquanto o broker e os protocolos utilizados são agnósticos aos formato de payload enviado nas mensagens, para uma correta integração com a *engine* de regras e os demais serviços da AWS, é necessário utilizar *payloads* em formato JSON, porém com uma estrutura interna que permanece livre.

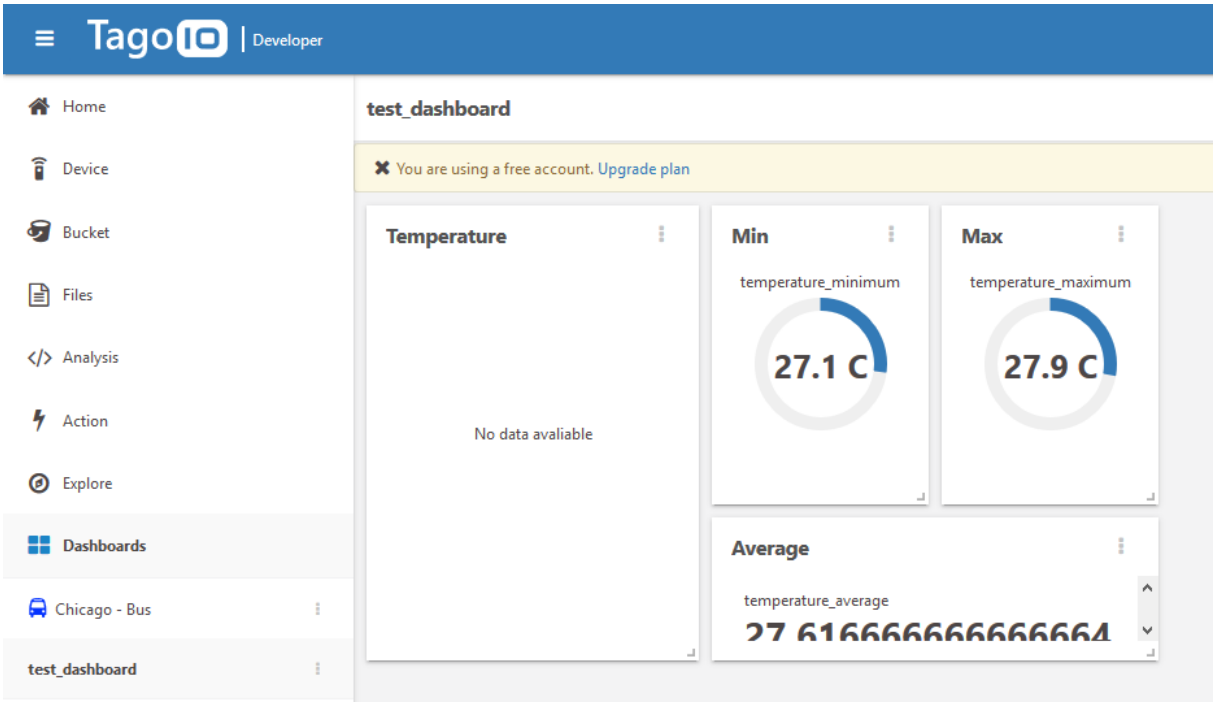

Figura 5 – Dashboard Tago.

A configuração de novos dispositivos, credenciais e permissões pode ser feita manualmente pela interface web, ou em batelada pela interface de linha de comando disponibilizada para a configuração dos serviços AWS (AWS CLI).

2.2.2 Tago.io

A Tago disponibiliza um serviço de recepção de dados por MQTT e HTTP, armazenamento e portal com várias opções para a exibição gráfica dos dados coletados, como exemplificado em Figura 5.

As ferramentas de gestão de dispositivo se limitam à geração de tokens de autenticação e gestão da destinação dos dados de cada dispositivo, sendo os aspectos de controle de acesso aos tópicos/urls geridos automaticamente.

As mensagens recebidas, que devem ter payload tipo JSON em um formato predefinido, tem seus campos extraídos e armazenados em Buckets, e podem ser recuperadas através de ferramentas de visualização ou API para processamento externo.

Por fim, é possível a programação de scripts em Python ou JavaScript que são disparados em um cronograma ou na recepção de mensagens, permitindo o disparo de novas mensagens, criação ou modificação de dados, ou a execução de funções de analítica sobre os novos dados e os dados já armazenados.

Agumas SDKs estão disponibilizadas por conveniência, com bibliotecas para realizar

a conexão e o envio dos dados nas linguagens Java, JavaScript e Python.

2.2.3 Konker(lab)

De forma muito semelhante a Tago (e a muitas outras plataformas), a Konker disponibiliza uma plataforma para a recepção, armazenamento, visualização, transformação e roteamento de dados de sensores. A plataforma e as informações dos dispositivos podem ser configuradas pela web manualmente ou através de uma API REST.

As mensagens são recebidas por HTTP e MQTT, em uma estrutura de tópicos/urls pré-definida devido ao controle de acesso, e com uma payload de tipo JSON, sem restrições quanto ao seu formato interno.

Os dados são recuperáveis através de inscrições MQTT e requisições HTTP, bem como pela interface web do sistema. Os dados também podem ser redirecionados de um dispositivo para outro.

2.3 Estudo de arquiteturas para a interoperabilidade

Dentro do escopo da telemetria, a interoperabilidade entre sistemas significa a habilidade de se obter e utilizar os dados disponibilizados pelos dispositivos em plataformas desenvolvidas independentemente destes.

Nota-se que este é um escopo mais restrito do que a típica aplicação de IoT, pois na Telemetria ha pouca ou nenhuma comunicação entre dispositivos, e até mesmo a comunicação da plataforma para o dispositivo muitas vezes não é necessária.

Entende-se então por interoperabilidade a facilidade de integração de novos dispositivos dada a similaridade de funcionamento com dispositivos previamente integrados.

2.3.1 Protocolos de aplicação abertos

Como mencionado em 1.3.3, a necessidade por protocolos de comunicação e aplicação no universo IoT não foi respondida de forma unânime. Diversas soluções tanto proprietárias como abertas foram adotadas, sendo estas últimas de grande interesse para a interoperabilidade entre sistemas. Vemos a seguir algumas das escolhas mais populares e mais mencionadas em *surveys* no que diz respeito a protocolos de aplicação para a IoT.

2.3.1.1 HTTP REST

REST, ou REpresentional State Transfer, é um modelo de aplicação que utiliza o protocolo HTTP para realizar transações simplificadas orientadas a recursos, tipicamente

reduzidas a uma única operação, de forma a reduzir a complexidade das interações entre dispositivo e servidor, e aumentar as chances de uma correta integração e interoperabilidade.

Seu conceito se traduz por efetuar transações que não são afetadas por nenhum possível estado definido fora da mesma, excluindo assim o uso de sequências complexas de transações e a necessidade de armazenamento, gestão e sincronização do estado de clientes e servidores. Informações de autenticação ou sessão como login, senha ou tokens são incluídos na própria transação e re-enviados a cada nova transação.

Embora o REST não constitua em si um protocolo bem especificado, é um conjunto de boas-práticas que permite o uso do protocolo HTTP para comunicações M2M orientada a recursos de forma simples, permitindo o acesso a recursos em formato JSON, XML e outros de forma padronizada [8].

2.3.1.2 WebSockets

O Websockets é um protocolo complementar ao protocolo HTTP para permitir uma comunicação assíncrona e bidirecional baseada em mensagens entre cliente e servidor, enquanto utilizando a mesma porta e regras de firewall presentes para conexões HTTP [8].

Embora o uso de WebSockets reduza consideravelmente o overhead em transações complexas ou no envio de múltiplas requisições, a abertura de um WebSocket ainda envolve uma requisição ao servidor com formato parecido ao HTTP.

2.3.1.3 CoAP

Desenvolvido entre 2010 e 2013 por um sub-grupo da *IETF*, o CoAP [9] é um protocolo versátil e de alto desempenho, inspirado no modelo de aplicação orientado a recursos usado em HTTP-REST, porém projetado para o uso em dispositivos com recursos computacionais limitados, redes de baixa energia e conexões intermitentes [7].

Este protocolo binário conciso apresenta serviços de *request-response*, descoberta de recursos, e inscrição para o recebimento de informações atualizadas. Sua estrutura é extensível e versátil, e prevê o uso de interfaces que fazem a tradução entre *requests* HTTP e CoAP para acesso aos recursos de dispositivos através da web.

2.3.1.4 MQTT

O MQTT, ou Message Queue Telemetry Transport, é um protocolo desenvolvido pela IBM em 1999, que foi posteriormente aberto e estandardizado pela OASIS entre 2013 e 2014. Originalmente parte da série MQ (Websphere) da IBM de serviços de comunicação, este protocolo foi concebido para a integração de sistemas de telemetria em aplicações de supervisão, tendo em vista minimizar os requisitos de hardware e o tráfego de rede [10].

Este protocolo consiste em um mecanismo de inscrição e publicação para a troca de mensagens assíncronas de clientes associados a um servidor de aplicação denominado *broker*. Os clientes se conectam ao broker para enviar mensagens ou para se inscrever em tópicos de recebimento. Enquanto permanecerem conectados, os clientes recebem de forma assíncrona as mensagens publicadas nos tópicos aos quais se inscreveram, e da mesma forma podem enviar mensagens destinadas a tópicos no broker para que sejam distribuídas.

O MQTT abrange diversos casos de uso, como o da comunicação de dispositivo para dispositivo, ou a comunicação de um (publicador) para muitos (inscritos) [11], e tem a flexibilidade para que novas rotas de encaminhamento de mensagem sejam facilmente construídas.

Diversas configurações como a denominada *qualidade de serviço* e *persistência de sessão* permitem se obter a robustez e confiabilidade necessárias nas mais diversas aplicações conforme necessário, sem sobrecarregar o seu funcionamento nos casos de uso menos exigentes.

2.3.1.5 XMPP

O XMPP, ou *Extensible Messaging and Presence Protocol*, é um protocolo aberto para troca de mensagens instantâneas, presença e gestão de contatos baseado em XML. Este protocolo foi criado em 1999 e foi revisado pela IETF nos anos seguintes, tornando-se um padrão para aplicações de *instant-messaging*.

Clientes se conectam a um *broker* usando um identificador, e múltiplos servidores podem se comunicar entre si. O serviço conta com várias extensões que podem ser utilizadas de acordo com a aplicação, ganhando capabilidades como descoberta de serviços, transferência de arquivos, *publish-subscribe* e outros.

O XMPP é um protocolo baseado em XML, e sendo um protocolo em modo texto, tem um considerável *overhead* em suas transações. O XMPP também não conta com mecanismos de garantia de entrega (*Quality of Service*) [11].

2.3.1.6 AMQP

O AMQP, ou *Advanced Message Queuing Protocol*, é um protocolo de troca de mensagens M2M que tem foco em uma arquitetura de produtores e consumidores. O protocolo permite criar filas de entrada (denominadas *exchanges*), de saída (denominadas *queues*), e rotas (denominados *bindings*) para distribuir e repassar mensagens na aplicação. Mecanismos de garantia de entrega são bem definidos e robustos, tornando essa solução popular em software corporativo e financeiro.

2.3.2 Middleware

Para os casos onde uma comunicação direta entre dispositivo e os servidores de aplicação não é possível, faz se uso dos denominados *middlewares*, ou intermediários. Estes intermediários podem ter a forma de plataformas, ou serviços de tradução e tunelamento de protocolos. O uso de middlewares é o que tem permitido uma integração ágil de novas aplicações no universo da IoT [12].

2.3.2.1 Plataformas de IoT

Diversas "plataformas de IoT" estão disponíveis para uso em um modelo PaaS (*Platform as a Service*), algumas das quais foram analisadas no início deste capítulo. Estas plataformas lidam com a recepção de dados dos dispositivos, sua armazenagem, e em boa parte dos casos, a sua exibição em *Dashboards* configuráveis com alarmes e outras funcionalidades semelhantes a softwares SCADA.

Como é possível observar das plataformas analisadas em 2.2 e em [12], a tendência observada é que a esmagadora maioria destas plataformas tem suporte à comunicação HTTP-REST, e, em segundo lugar, MQTT. Elas contam, na maior parte dos casos, com SDKs e bibliotecas para a programação dos dispositivos, de forma a torná-los compatíveis com a plataforma, dado que a API a ser utilizada deve ser aquela da plataforma, e que tipicamente varia bastante de uma plataforma para a outras.

Os dados armazenados nestas plataformas podem então ser recuperados para uso em outros sistemas e aplicações, fazendo delas um efetivo middleware que não constitui a aplicação em si, mas um mecanismo de integração entre dispositivos e *back-end*.

Esta arquitetura de integração baseada em plataformas é suportada por [13], onde é defendida uma padronização dos mecanismos para a descoberta e recuperação dos dados destas plataformas, de forma a minimizar o esforço de integração entre plataformas e serviços.

2.3.2.2 Message Oriented Middleware

Outras soluções de middleware fazem parte do conjunto mais genérico denominado *Message Oriented Middleware*, ou *MOM*. Estas plataformas permitem a integração e o roteamento de mensagens para um grande número de dispositivos, segundo regras que podem ser definidas e alteradas conforme a aplicação evolui, com grande flexibilidade. Também são colocados nesta categoria por alguns autores os *brokers* de protocolos orientados a mensagem que possuem arquitetura e funcionalidade semelhante.

Um dos exemplos mais conhecidos de MOM é o projeto Apache Camel, um projeto open-source baseado em Java que permite a troca de mensagens em diferentes protocolos, e a criação de regras e rotas para tais mensagens. Baseado em *EIPs* (*Enterprise Integration*

Patterns), o Camel suporta uma grande variedade de protocolos de transporte, mas no que diz respeito ao conteúdo das mensagens, uma completa integração só é possível com o uso de alguns formatos padrão envolvendo objetos Java.

2.3.2.3 Middlewares específicos

Com suficiente conhecimento dos protocolos envolvidos, é possível desenvolver software que realize a tradução entre um protocolo não suportado por determinada plataforma e um protocolo suportado pela mesma, permitindo a integração de dispositivos que não podem ser reconfigurados e estão presos a um protocolo fixo.

Esta integração no entanto não pode ser vista como uma solução de interoperabilidade, a menos que o esforço de se repetir integrações semelhantes para uma ampla gama de dispositivos seja bastante reduzido.

2.4 Avaliação de empresas parceiras e integrações desejáveis

A Sensorweb tem feito projetos colaborativos com diversas empresas de hardware em Florianópolis e no Brasil para a fabricação de dispositivos de Telemetria para adicionar ao seu catálogo.

Em outra frente com propósito semelhante, a Sensorweb procura possibilidades de negócios com fornecedores já estabelecidos no ramo de dispositivos de telemetria e data-loggers.

2.4.1 Smetro

SMETRO é uma startup dedicada ao mundo hardware e dos sensores IoT. A SMETRO posiciona-se como parceira industrial de empresas que buscam sensores conectados de alta qualidade, robustez e fácil instalação. A SMETRO vem desenvolvendo dispositivos de monitoramento com diversos protocolos de comunicação, como LoRa, BLE, WiFi e 3G, para atender a diversos casos de uso, em um formato compacto como o apresentado em Figura 6.

Em parceria com a Sensorweb, foi realizada a integração de uma linha de seus dispositivos 3G com a plataforma Endrixx utilizando o protocolo descrito em 2.1.3 programado diretamente nos dispositivos, porém frente às dificuldades de projeto e manutenção de uma tal bifurcação, é de interesse da Sensorweb que se passe a um modelo de integração por protocolos abertos.

Os dispositivos SMETRO WiFi tem suporte, entre outros, ao protocolo MQTT, com payload em formato JSON. A conexão com o broker MQTT é possível por *tcp* somente, não havendo suporte a *tls* no momento, e o *QoS* utilizado está fixo em 0.

Figura 6 – Dispositivo Smetro WiFi.

O formato de payload utilizado pelos dispositivos SMETRO WiFi para as mensagens MQTT é conforme o ilustrado em Bloco de Código 2.1.

2.4.2 Khomp

A Khomp está atualmente desenvolvendo uma linha de sensores e gateways para uso em sistemas de monitoramento. Alguns destes modelos foram projetados para serem integrados à solução da Sensorweb, enquanto outros seguem padrões internos para a comunicação.

Em uma parceria recente com a Sensorweb, a Khomp desenvolveu um modelo de gateway que pode vir a substituir o modelo atualmente utilizado na empresa, atendendo às especificações e aos protocolos determinados pela Sensorweb. Com o conhecimento adquirido neste projeto, eles elaboraram também a sua própria solução genérica de gateways e sensores de temperatura de umidade, atualmente em etapa de testes.

As linhas genéricas desenvolvidas foram denominadas ITG e ITS, sendo a primeira baseada em transmissores de baixa energia e gateways, e a segunda fazendo uso de

Bloco de Código 2.1 – Formato de payload Smetro

```
{
    "status":{
        "value":1.00, // Versão do firmware
        "timestamp":1234567891234, // Unix Timestamp em milissegundos
        "context":{
            "power":"connected",
            "battery":"disconnected"
        }
    },
    "i0":{
        "value":1, // Leitura digital
        "timestamp":1234567891234
    },
    "intr0":{
        "value":0, // Contagem de interrupções
        "timestamp":1234567891234
    },
    "s0":{
        "value":25.93, // Leitura da sonda externa
        "timestamp":1234567891234,
        "context":{
            "id":123456789123456789 // Número de série da sonda
        }
    }
}
```

Bloco de Código 2.2 – Formato de payload Khomp

```
{
    "seq":1234, // Número de sequência para controle de fluxo
    "data":[
        {
            "time":1234567891234, // Unix Timestamp em millisegundos
            "unit":2224179556, // Identificador de unidade no padrão IEEE 1451
            "value":123456798, // Valor da leitura
            "dev_id":123465, // Código do dispositivo que efetuou a leitura
            "ref":"", // Código do sensor lido, se existir
        },
        ...
    ]
}
```

comunicação 3G. Ambas utilizam um protocolo bastante semelhante baseado em uma API HTTP complexa à qual são enviadas mensagens de status e de dados das leituras de sensores. O conteúdo das mensagens contendo os dados de sensores é um documento JSON na forma apresentada em Bloco de Código 2.2.

2.4.3 Novus

A Novus é uma empresa fundada no Rio Grande do Sul que se tornou referência nacional em instrumentos de medição e controle, e que lançou recentemente uma linha de data-loggers para telemetria chamada LogBox, apresentada em Figura 7. Com diversos protocolos de comunicação disponíveis, os LogBox se encaixam em diversos casos de uso. Eles possuem sensor interno de temperatura, e podem utilizar uma grande gama de

Figura 7 – Dispositivo Novus Logbox.

Bloco de Código 2.3 – Formato de payload Novus

```
{
    "n_channels":4, // Canais de leitura
    "timestamp":1531945548, // Unix Timestamp em segundos
    "battery":5.69, // Tensão da bateria
    "value_channels":[0.000,24.200,0.000,24.200], // Valores lidos
    "alarm_low":[0,1,0,0], // Disparos de alarmes
    "alarm_high":[0,0,0,1],
    "buzzer_state":0 // Estado do alarme sonoro
}
```

sensores externos.

O dispositivo LogBox WiFi funciona como um data-logger com conectividade oportunista, sem interromper o seu funcionamento no caso de falta de energia ou internet. Ele tem suporte (dentre outros) ao protocolo MQTT, enviando mensagens com a suas configurações e com os valores de suas leituras em formato JSON. O formato para os documentos JSON enviados pelo LogBox WiFi é apresentado em Bloco de Código 2.3.

2.4.4 Fanem

A Fanem é uma fabricante de material médico-hospitalar com uma longa história de parceria com a Sensorweb. Em 2015, a Fanem contactou a equipe de desenvolvimento da Sensorweb, que trabalhava sob o nome de MCA Sistemas, para o desenvolvimento de um sistema de monitoramento local para câmaras frias e geladeiras. Após a finalização deste projeto, denominado SoftChamber, a Fanem demonstrou interesse pela solução de monitoramento remoto proposta pela Sensorweb.

Tomando participação na empresa, a Fanem tem interesse em desenvolver projetos conjuntos para a integração de sistemas de telemetria em diversos de seus produtos nas linhas de câmaras frias e equipamentos do setor neo-natal.

Um dos projetos em curso atualmente é um módulo de conectividade denominado Multiconnect, a ser compatibilizado com diversos equipamentos da FANEM, tais como câmaras de conservação (refrigeradores), estufas e incubadoras, e trazendo conectividade multi-protocolo e funcionalidades de telemetria a estes equipamentos

A placa Multiconnect suporta os protocolos Bluetooth, WiFi e Ethernet, e prevê conectividade com os serviços da Sensorweb, com um aplicativo de celular via Bluetooth para monitoramento local, e com serviços MQTT.

2.4.5 Monnit

A Monnit é um fornecedor norte-americano de sensores e gateways para o monitoramento de umidade, temperatura e outras grandezas. Sendo um grande fabricante, a Monnit disponibiliza uma vasta gama de sensores com um projeto robusto, longo alcance e alta duração de bateria.

A Sensorweb tem interesse em potenciais fornecedores de hardware como a Monnit. No momento, os dispositivos disponibilizados por ela não atendem a todas as especificações impostas pelo sistema atual da Sensorweb, o que freou os esforços de integração destes dispositivos, mas ha um interesse futuro em se fazer a integração dos dispositivos deste fornecedor.

Um um pequeno kit com um sensor e um gateway foi adquirido para testes em Novembro (Figura 8), porém já se observa que os dispositivos da Monnit se comunicam exclusivamente com a plataforma própria da empresa, *iMonnit*, sendo necessária uma integração através da API desta plataforma.

Figura 8 – Dispositivo Monnit e Gateway.

2.5 Protocolos de aplicação para telemetria e payloads

Como apresentado em 1.3.3, ha uma grande fragmentação no que diz respeito aos protocolos utilizados em aplicações de IoT. Esta fragmentação apareceu primeiro com novas soluções de redes de baixa energia não baseadas em IP, e se propagou na forma de stacks de protocolos desenvolvidos independentemente para tais tecnologias, como é o caso do ZigBee, LoRaWAN, Sigfox, e diversos outros.

O que se observa em soluções de telemetria no momento presente, como se pode ver dos produtos listados em 2.4, é que muitas aplicações de telemetria encontraram uma solução "boa o suficiente" no uso do protocolo HTTP REST, devido à assimetria da comunicação e baixa complexidade das operações realizadas.

2.5.1 A camada de aplicação da Telemetria

O funcionamento de sistemas de telemetria pura como os da Sensorweb consiste (como descrito em mais detalhes em 2.1) em dispositivos em campo que fazem medições de temperatura, umidade ou outras grandezas, possivelmente em aglomeradores que fazem

interface entre redes não-ip e a internet, e por fim em plataformas que fazem a recepção de dados, tipicamente na nuvem.

A informação que trafega dentro deste esquema consiste em:

1. Identificação do dispositivo
2. Identificador da variável lida
3. Valor da leitura
4. Instante da leitura
5. Informações adicionais sobre o dispositivo

Os itens 2, 3 e 4 podem se repetir no caso de várias variáveis lidas. Vários destes itens podem ser omitidos ou inferidos de acordo com a forma de operação. Por exemplo, um dispositivo conectado com um IP fixo, e que transmite uma única variável, não tem necessidade de transmitir informações de identificação, e um dispositivo que transmite o valor lido em tempo real (*soft*) não tem necessidade de fornecer uma timestamp com o instante da leitura.

2.5.2 Payloads em formato de documento JSON

Observa-se das análises de plataforma e dispositivo feitas em 2.2 e 2.4 que o formato de documento JSON foi adotado como um padrão *de-facto* para mensagens de telemetria.

O formato de documento JSON consiste em uma notação de representação de objetos, tendo uma estrutura flexível e adaptável a muitas aplicações. No caso da telemetria, dentro deste documento JSON se encontram tipicamente os valores lidos pelos dispositivos, como pode ser visto nos diferentes exemplos de payloads JSON foram apresentadas em 2.4.

Nota-se destes exemplos que a estrutura interna dos documentos JSON apresentados difere bastante, não havendo uma notação consistente ou caminho universal para se obter as leituras de variáveis e outras informações úteis dentro dos documentos. Em outras palavras, não é possível escrever código que acesse dados na forma `Objeto.valor` e `Objeto.timestamp` pois a estrutura do documento não segue um padrão.

Ha um esforço de estandardização do formato dos documentos JSON enviados em aplicações de telemetria denominado SenML [14], que permite a transmissão de mensagens de telemetria de forma concisa e universal. Este protocolo no entanto é recente e ainda não muito adotado.

2.6 Avaliação do protocolo MQTT

Devido à escolha realizada pela implementação de um cliente utilizando o protocolo MQTT, foi feito um estudo aprofundado sobre o funcionamento deste protocolo. Os detalhes mais técnicos são aqui omitidos, sendo esta seção somente para familiarizar o leitor com o funcionamento do protocolo, suas características, e por fim justificar algumas das escolhas de projeto realizadas.

2.6.1 Origens e características

Como mencionado em 2.3.1, o MQTT é um protocolo de encaminhamento de mensagens desenvolvido pela IBM para situações de baixo poder de processamento de dispositivos e baixo consumo de rede.

Este protocolo permite que clientes enviem mensagens destinadas a determinados tópicos, e que clientes se inscrevam a tópicos para a recepção das mesmas, permitindo que a arquitetura da aplicação seja alterada ou ampliada sem a necessidade de se reconfigurar os dispositivos já conectados.

2.6.2 Funcionamento

O funcionamento deste protocolo consiste em uma conexão TCP entre o cliente e um servidor com o serviço denominado *broker*. O estabelecimento da conexão envolve uma mensagem de conexão e autenticação, que é respondida com uma confirmação pelo broker.

Uma vez estabelecida a conexão, o envio de mensagens é feito com uma mensagem contendo um cabeçalho de 2 *bytes*, um tópico, e a payload ou corpo da mensagem. O mecanismo de envio de mensagem pode aguardar uma confirmação ou não, conforme o *QoS* solicitado.

A inscrição em tópicos para a recepção de mensagens se dá por uma mensagem e uma confirmação. Por fim, a entrega de mensagens é assíncrona e parte do broker, devendo ou não ser confirmada pelo cliente de acordo com o QoS solicitado durante a inscrição tópico. As mensagens são enviadas a seus destinatários finais de acordo com o tópico ao qual ela está associada, como ilustrado em Figura 9.

Uma variante do protocolo MQTT denominada MQTT-SN foi desenvolvida especificamente para redes de sensores e dispositivos embarcados com limitações de conectividade. Não sendo mais baseado em conexões TCP, este protocolo permite que os dispositivos hibernem entre envios de mensagens, além de prever outras funcionalidades específicas para conexões extremamente limitadas e intermitentes. Este protocolo no entanto é relativamente recente e não é comumente utilizado ainda no momento.

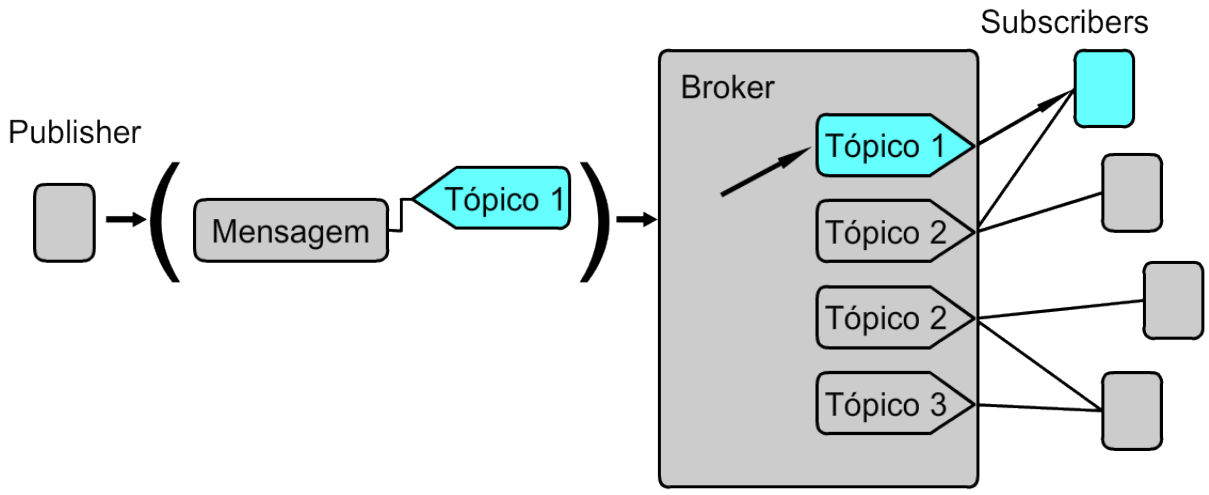

Figura 9 – Diagrama de envio de mensagem via MQTT.

2.6.3 Abrangência e versatilidade

Tratando-se de um protocolo de transporte de mensagens, o MQTT não define diretamente uma aplicação, permanecendo bastante flexível para que se defina a aplicação em alguma camada acima dele. O MQTT pode transportar mensagens em qualquer formato, podendo suportar tanto protocolos baseados em texto quanto protocolos em binário. A única particularidade definida neste protocolo é a utilização de tópicos, que funcionam como endereços ou caixas postais para o redirecionamento das mensagens.

Este protocolo tem suporte a funcionalidades de persistência e controle de entrega de mensagens, porém a não-mandatoriedade de tais funcionalidades permite manter um funcionamento leve em dispositivos com pouco poder de processamento ou conectividade limitada.

Este protocolo tem a vantagem de ter suas informações de controle codificadas em binário, sendo bastante conciso e com baixo overhead, além da possibilidade do uso de mensagens não-respondidas como é o caso no uso de *QoS 0*, reduzindo assim o tráfego de rede, e no caso de dispositivos a bateria, o tempo que o dispositivo permanece ativo.

2.6.4 Comparação com o HTTP

O protocolo MQTT tem ganhado espaço no universo IoT e em diversas outras aplicações devido não somente ao overhead reduzido em suas mensagens, mas sobretudo devido à capacidade de receber mensagens do servidor de forma assíncrona e eficiente.

Um dos aspectos mais interessantes do MQTT é a capacidade de se realizar o envio de mensagens em transações de complexidade variável de forma transparente, ou seja, definindo-se a “qualidade de serviço” ou *QoS* para cada mensagem de acordo com as

necessidades da aplicação sem a necessidade de modificar a mesma. Este mecanismo torna o MQTT adequado a diversos casos de uso, desde os de comunicação extremamente restrita até sistemas robustos e altamente confiáveis. O protocolo HTTP REST em contrapartida opera com um único modelo de transação de tipo pedido-resposta.

Por fim, diversas aplicações recentes tem utilizado sistemas de notificação no qual cabe ao servidor notificar aos dispositivos sobre novos eventos e mensagens. Para que tais eventos sejam recebidos pelo cliente em tempo suficientemente breve usando o protocolo HTTP, é necessário realizar uma abordagem por *polling*, na qual o cliente periodicamente solicita atualizações do servidor. Uma estratégia alternativa é a que faz uso de mecanismos de *long-poll*, no qual o dispositivo faz uma solicitação ao servidor, que não responde de imediato, mas sim posterga a resposta por um longo período, no aguardo de que ocorra algum evento a ser notificado ao dispositivo, ou que a conexão se encerre. O protocolo MQTT em contrapartida prevê o envio de mensagens assíncronas do broker ao cliente de forma simples e mais eficiente.

2.6.5 Conclusões sobre o uso do protocolo

O protocolo MQTT vem sendo utilizado cada vez mais para soluções de IoT por ser versátil o bastante para atender a diversas arquiteturas de aplicação e leve o bastante para atender às restrições de dispositivos embarcados. É um dos protocolos mais mencionados na literatura recente para aplicações de IoT ([5, 7, 8, 11, 15])

Embora o protocolo de mensagens em si não defina um protocolo de aplicação único a ser utilizado, o uso de um protocolo comum para a troca de mensagens é um passo na direção da integração de sistemas de IoT. Enquanto isso, as tendências convergentes na utilização de JSON e, futuramente, SenML para aplicações de telemetria prometem uma melhor integração de sistemas no futuro próximo.

2.7 Considerações

Os desenvolvimentos recentes da Internet das Coisas observados nos últimos anos sugerem que esta nova tecnologia está atingindo um certo grau de maturidade, no qual uma gama de protocolos mais comumente adotados foram refinados para atender às aplicações mais comuns de forma satisfatória, freando a necessidade por novos protocolos.

Dentre aplicações de telemetria, observa-se a preferência por protocolos leves e já de certa maturidade em aplicações de IoT, permitindo a utilização de hardware mais simples. Esta convergência é observável nas semelhanças entre plataformas e entre dispositivos.

Dentre estes protocolos, destacam-se o protocolo de troca de mensagens MQTT, o protocolo de transações HTTP e o formato de documento JSON, que apareceram repetidas

vezes nos dispositivos e plataformas estudados. A integração destes protocolos em um sistema de monitoramento deve prover interoperabilidade e facilitar a integração com uma vasta gama de dispositivos e plataformas.

3 Implementação de Soluções de Middleware

O estudo das plataformas apresentado em 2.2 foi acompanhado de várias implementações com o intuito de se explorar mais a fundo cada solução e ganhar conhecimento prático.

O estudo das arquiteturas de soluções para a interoperabilidade apresentado em 2.3 foi uma tarefa constante ao longo de todo o trabalho, sendo reforçado pelas atividades práticas realizadas, onde tais arquiteturas foram implementadas.

No início das atividades, muitos dos dispositivos a serem integrados não estavam disponíveis para testes, e fez-se uso de microcontroladores programáveis para a prototipagem de dispositivos comunicantes em diversos protocolos com a finalidade de realizar os testes com diversas plataformas.

3.1 Soluções baseadas em plataformas

Aqui será abordado com um ponto de vista mais técnico o uso de plataformas de terceiros para a recepção de dados de dispositivos em campo conforme proposto no capítulo anterior. A escolha destas plataformas foi anterior à fundamentação teórica, tendo sido definida já na proposta do projeto, e o conhecimento prático adquirido com estes experimentos reforçou o estudo bibliográfico realizado em paralelo.

3.1.1 Objetivo e Escopo

Tendo em vista o objetivo de integração de dispositivos pertencentes a outras soluções verticais, optou-se inicialmente, pelo que pareceu ser uma das abordagens mais rápidas: Diversas plataformas de IoT estão disponíveis para a recepção de dados de telemetria de dispositivos, capazes de comunicar em diversos protocolos.

O uso de plataformas de terceiros para a recepção de dados de telemetria é uma solução corriqueira na integração de soluções, pois em geral a parte de gestão e comunicação com o dispositivo é abstraída, e os dados obtidos podem ser diretamente acessados e imediatamente utilizados.

A praticidade de uma integração através de plataforma ou middleware se torna visível para dispositivos com protocolos de comunicação complexos, onde o esforço para a obtenção dos dados diretamente no dispositivo é bastante amplificado, e assim como para a gestão de grandes frotas de dispositivos, onde uma ferramenta de consulta centralizada pode acelerar o processo da análise de dados.

Na maioria dos casos, o fabricante de determinado dispositivo possui sua própria plataforma, à qual o seu produto se conecta por padrão para a o envio de dados a menos que reconfigurado. Estes costumam ser os casos de integração mais triviais, onde a recuperação dos dados obtidos se faz por requisições HTTP à plataforma, permitindo obter séries temporais de leituras sem grande esforço. Esta forma de integração por middleware, como descrita em 2.3.2, é provavelmente uma das soluções com menor *time to market*.

Estabeleceu-se como objetivo intermediário para o estudo de integração por plataforma uma prova de conceito rápida para a utilização das principais plataformas estudadas.

3.1.2 Metodologia

Para testar as funcionalidades de cada plataforma, foi realizada a integração de um dispositivo pelo protocolo determinado como "preferencial" para aquela plataforma. Este experimento teve como objetivo também verificar a viabilidade e a simplicidade de uso de cada plataforma escolhida.

Cada experimento consistiu em se programar um dispositivo para a conexão à plataforma, em se realizar o envio de mensagens, e por fim em se testar as funcionalidades de maior interesse em cada plataforma.

Cada plataforma por fim foi julgada em aspectos relacionados à praticidade da solução proposta, à facilidade de instalação e migração de sensores, e a completeza das funções disponibilizadas.

3.1.3 Dispositivo utilizado

As atividades listadas no restante desta sessão foram realizadas ainda nas primeiras semanas dentro da empresa, e não se dispunha ainda dos dispositivos listados em 2.4.

Para permitir testes rápidos de plataforma, foi elaborado um dispositivo simples, baseado em um módulo *ESP8266*, conectado a uma sonda Dallas DS18b20 para a leitura de temperatura. O dispositivo conta com uma fonte externa, o módulo ESP01 com conectividade wifi, e a sonda de temperatura. Apelidado "ESPeixe", o dispositivo é apresentado em Figura 10.

Este módulo conta com um processador RISC da *Tensilica* a 80 MHz, 80 KB de RAM interna, 1 MB de ROM externa, somente 4 pinos utilizáveis para GPIO (com determinadas restrições) e conectividade WiFi.

Através da IDE do Arduino, e de bibliotecas disponíveis online, foi possível realizar implementações que se comunicavam nos protocolos HTTP e MQTT, necessários para testar as plataformas escolhidas.

Uma alternativa proposta inicialmente teria sido o uso da placa de prototipagem

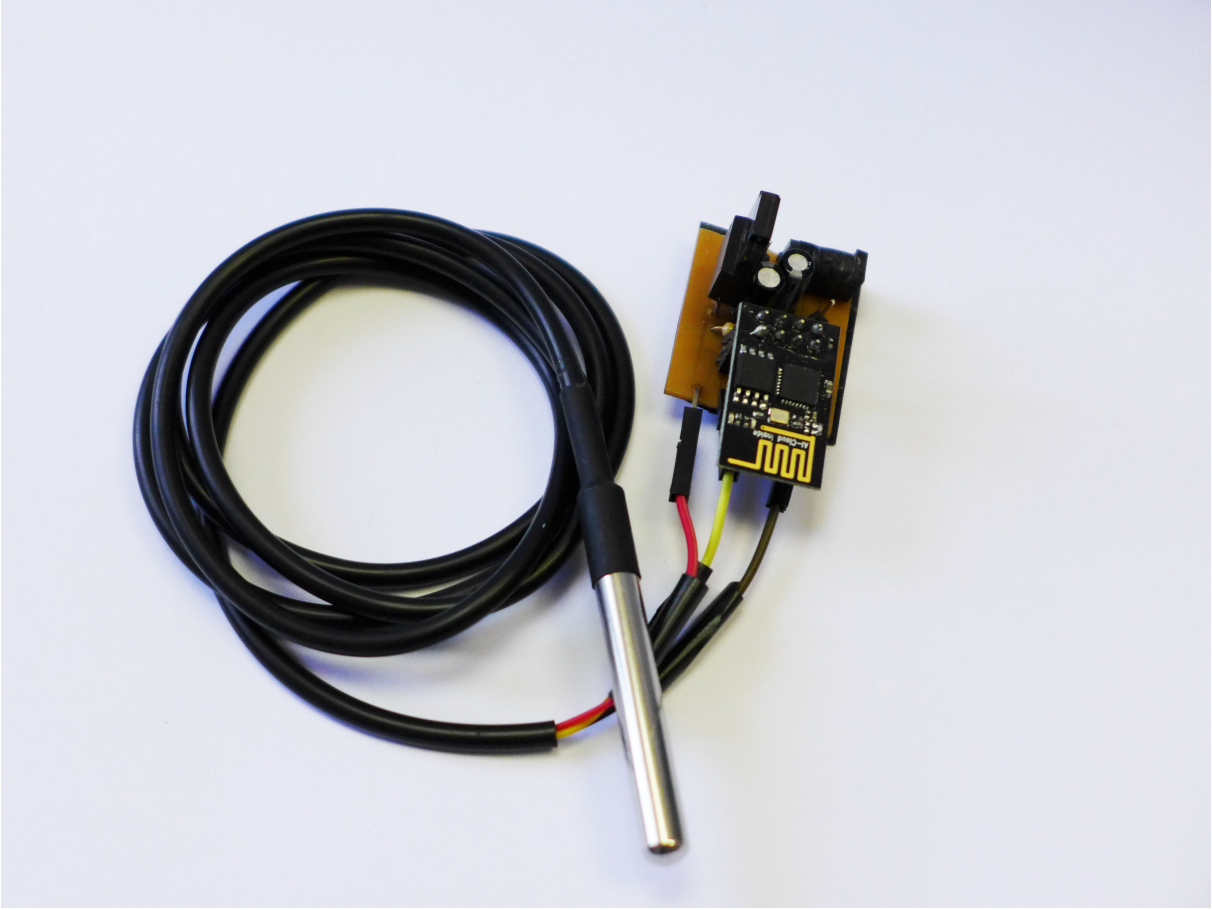

Figura 10 – Dispositivo ESPeixe.

rápida *Linkit One*, de mais fácil utilização, porém a falta de bibliotecas de TLS atualizadas para este processador foi um fator imediatamente limitante e impediu o seu uso nos trabalhos realizados.

3.1.4 Integração com o Amazon IoT Core

Como uma das primeiras atividades a serem realizadas, foi proposto o teste da plataforma IoT Core da Amazon para a conexão de dispositivos se comunicando por MQTT.

3.1.4.1 Programação do dispositivo

Para a integração com a plataforma IoT Core da Amazon, ha diversas SDKs disponíveis para linguagens como Java e Python, bem como soluções em C para dispositivos embarcados com sistema operacional. O uso do sistema operacional sugerido no entanto (FreeRTOS) estava fora de questão, devido às limitações de memória do módulo ESP8266.

Uma alternativa encontrada foi o projeto Mongoose OS, que pode ser instalado

no ESP, e possui bibliotecas para comunicação com alguns dos serviços AWS, incluindo o IoT Core. No entanto, sua instalação pareceu não-trivial, e optou-se por utilizar as bibliotecas nativas do projeto Arduino para o ESP8266. As bibliotecas utilizadas foram *clientWifiSecure* para a conexão SSL (TLS), parte do SDK distribuído pela *Espressif*, fabricante do ESP8266, e a biblioteca de código aberto *pubsubclient* [16] para a conexão com o broker pelo protocolo MQTT.

3.1.4.2 Conexão

A conexão ao broker do IoT Core somente pode ser feita por um canal TLS com autenticação mútua através de certificados X.509. Para isto, foi necessário gerar certificados para o dispositivo, utilizando ferramentas da Amazon, e incluí-los em forma binária no programa embarcado. Para a conversão entre formatos de certificado em modo texto e modo binário, foi utilizada a ferramenta *openssl*, e para a sua inclusão no código fonte em forma codificada hexadecimal, foi utilizada a ferramenta Unix *xxd*.

O processamento das funções de criptografia necessário para o estabelecimento da conexão leva vários segundos no microcontrolador, e pode limitar os casos de uso em dispositivos com baixo poder de processamento e alimentados por bateria. Também é inviável armazenar e processar as longas cadeias de certificados do servidor como é previsto pelo protocolo na validação de certificados X.509, e portanto a identidade do servidor não é verificada nesta implementação.

Uma vez estabelecido o canal TLS, o dispositivo se conecta efetivamente ao broker MQTT, usando um identificador de cliente único que deve ser registrado na plataforma. O certificado utilizado em combinação com o identificado servem como mecanismo de autenticação, e definem as permissões atribuídas ao dispositivo.

3.1.4.3 Envio de mensagens

Uma vez estabelecida a conexão, o dispositivo entra em um loop, fazendo a leitura do termômetro, formulando uma mensagem JSON, e a enviando para o broker em um tópico determinado. O formato das mensagens enviadas é o apresentado a seguir:

[float com a mensagem JSON MQTT]

Este formato simples não corresponde a nenhuma aplicação em particular, e foi utilizado somente como exemplo de uma implementação mínima.

3.1.4.4 Aplicação

O broker presente no serviço IoT Core da Amazon permite que aplicações sejam conectadas aos dispositivos pelo mecanismo de *subscribe* do MQTT. Além disso, também é possível definir regras que podem disparar diversas ações baseado no conteúdo e no

tópico das mensagens sendo processadas. A integração para armazenar valores em bancos de dados, ou disparar funções de processamento de dados através do serviço de Lambda são bastante triviais, porém tipicamente requerem que o conteúdo das mensagens seja um documento JSON válido.

O IoT core conta com uma ferramenta de depuração (debug) no navegador que consiste basicamente em um cliente MQTT com as devidas permissões de acesso para dar *publish* e *subscribe* em quaisquer tópicos e verificar o funcionamento do sistema. Não ha, contudo, qualquer ferramenta simples para a visualização de gráficos ou estatísticas dos valores recebidos, sendo necessário o desenvolvimento de um front-end através de outras ferramentas.

3.1.4.5 Conclusão

A plataforma IoT Core da Amazon constitui essencialmente um broker MQTT pré-configurado e com integrações inteligentes aos demais serviços AWS. Este serviço não conta com qualquer interface gráfica dedicada para a visualização do histórico dos dados recebidos, ou outras funcionalidades de sistema supervisório, sendo necessário o desenvolvimento da mesma através de outros serviços.

As configurações necessárias no dispositivo, como o uso de criptografia e certificados de cliente, são um fator limitante, pois nem todos os dispositivos embarcados são capazes de estabelecer um canal de comunicação TLS com autenticação mútua devido aos requisitos computacionais das operações de criptografia envolvidas no processo.

3.1.5 Integração com a plataforma Tago

Dando continuidade ao estudo das plataformas elencadas, estabeleceu-se como objetivo conectar um objeto (thing) também à plataforma Tago.

3.1.5.1 Programação do dispositivo

A programação do microcontrolador seguiu um caminho semelhante ao encontrado para a implementação pela plataforma da Amazon. Dado que as SDKs disponibilizadas davam suporte somente a Java e JavaScript, optou-se por descarta-las completamente e utilizar as bibliotecas padrão do projeto Arduino para o ESP8266.

Para a comunicação, usou-se apenas a biblioteca *clientWifiSecure* para o estabelecimento de uma conexão segura, enquanto a requisição HTTP foi feita manualmente devido a sua simplicidade.

Bloco de Código 3.1 – Formato de mensagem Tago

```
{
    "variable":"temperature", // Nome da variável
    "unit":"F", // Unidade da variável
    "value":55, // Valor enviado
    "time":"2015-11-03 13:44:33", // Hora em formato
         // ISO 8601 com uso de espaço, opcional
    "location":{ // Coordenadas geográficas, opcional
       "lat":42.2974279,
       "lng":-85.628292
    }
}
```

3.1.5.2 Conexão

A plataforma Tago recebe dados de dispositivos pelo protocolo HTTPS, ou seja, HTTP por um canal seguro TLS. O uso de criptografia é extremamente recomendado, porém também é possível realizar conexões por HTTP simples. Embora não seja feita a menção para tal na página principal da sua documentação (no momento em que foi consultada para este projeto), consta em outros documentos de ajuda que é possível fazer o envio de dados para a plataforma via MQTT.

Para o sucesso da conexão, os únicos requisitos são um token de dispositivo válido e uma mensagem válida.

3.1.5.3 Envio de mensagens

O envio de mensagens a esta plataforma é bastante trivial. É feita uma requisição de tipo POST a um endereço web, com o corpo de mensagem contendo um documento JSON com os dados de leitura no formato predeterminado ilustrado em Bloco de Código 3.1.

O dispositivo que faz o envio é identificado por um *token* no cabeçalho (*header*) da requisição HTTP.

Para a implementação prática, o dispositivo faz o envio de temperatura lida do termômetro a cada 5 segundos, sem informações de horário ou localização.

3.1.5.4 Aplicação

A aplicação *TagoIO* conta com uma interface para a gestão de dispositivos, configuração de ações e transformações e visualização das informações nas dashboards.

É possível criar análises sobre os dados armazenados, para se obter dados como mínimo e máximo de valores, média e outras grandezas estatísticas. As análises são programadas em JavaScript (Node.js), e os valores resultados são armazenados de forma semelhante às variáveis enviadas por dispositivos.

A configuração de *Actions* permite efetuar o disparo das funções de análise com

base na chegada de mensagens. As configurações de disparo de Actions permitem se obter funcionalidade semelhante à de Alarmes em um sistema SCADA. É possível efetuar o disparo de Emails, SMSs e outros baseado em valores de referência para máximo e mínimo de variáveis, por exemplo.

Por fim, a aplicação permite a criação de Dashboards aos quais é possível adicionar *widgets* para a exibição das variáveis armazenadas.

Com exceção da configuração de dispositivos, a plataforma não conta com ferramentas para a automatização da configuração, sendo necessário configurar cada action, dashboard e widget manualmente a partir da interface gráfica.

3.1.5.5 Conclusão

A plataforma Tago faz o recebimento de dados via HTTPS em um protocolo simples e de fácil implementação. Trata-se no entanto de um formato predefinido e sem a flexibilidade necessária para a integração de dispositivos arbitrários, sendo necessária a programação do mesmo para o uso do protocolo estabelecido.

Aspectos de armazenagem, transformação dos dados e análise são todos bem integrados e automatizáveis. A plataforma conta com uma interface gráfica elegante para a apresentação dos dados monitorados.

A falta de uma API para a configuração automática de *Actions* e *Dashboards* é um fator limitante para a utilização dessa plataforma pois impossibilita uma migração rápida de uma grande quantidade de sensores.

3.1.6 Integração com a plataforma Konker

A Konker é uma empresa brasileira que está criando uma plataforma para soluções de IoT.

3.1.6.1 Programação do dispositivo

A plataforma Konker não conta com qualquer SDK, sendo necessário a programação de dispositivos através de outras bibliotecas. No entanto, devido ao suporte ao MQTT, a integração com esta plataforma foi bastante semelhante à integração com a Amazon, sendo utilizadas as mesmas bibliotecas para o ESP8266.

3.1.6.2 Conexão

A plataforma Konker tem suporte a HTTP e MQTT ambos com ou tem o uso de TLS. Ambas as opções tem funcionalidade muito parecida, havendo uma analogia entre URL no HTTP e tópico no MQTT.

O uso de criptografia neste projeto foi facilitado por não haver necessidade de se utilizar certificados para o dispositivo e servidor. A autenticação do dispositivo é feita por *tipo Http-Basic* para o HTTP e fornecendo usuário e senha na conexão com o broker para o MQTT.

3.1.6.3 Envio de mensagens

O envio de mensagens está restrito a uma arquitetura de "canais" que define a quais endereços ou tópicos cada dispositivo tem acesso. Qualquer comunicação entre dispositivos é mediada por regras ("rotas") definidas na plataforma.

O envio de mensagens por MQTT é feito através de um tópico na forma "pub/<device_username>/<canal>" e deve conter um documento JSON válido.

O recebimento de mensagens recebidas da plataforma através de rotas se dá pelo tópico equivalente "sub/<device_username>/<canal>".

3.1.6.4 Aplicação

A plataforma Konker permite a visualização de mensagens recebidas para cada dispositivo individualmente, conta com retenção de dados por um período delimitado, e permite a criação de rotas de mensagens entre dispositivos e a chamada a APIs externas. O conjunto de funcionalidades oferecido é bastante básico, porém flexível.

3.1.6.5 Conclusão

O suporte a múltiplos protocolos, mensagens JSON genéricas, combinado com a capacidade de processar e redirecionar mensagens, faz da Konker uma escolha interessante para uma plataforma de *middleware* para uma integração de IoT.

A estrutura fixa de tópicos e URLs no entanto é um inconveniente, por ser justamente o aspecto que não pode ser configurado em vários dos dispositivos estudados para a integração.

3.1.7 Soluções Khomp e Smetro

Tanto a empresa Khomp quanto a Smetro (2.4) contam com suas próprias plataformas para a recepção dos dados. Ambas contam com armazenagem e API de acesso aos dados.

A integração com estas plataformas foi julgada como sendo não-prioritária, devido ao suporte ao protocolo interno da Sensorweb já instalado nos dispositivos, e a especificidade das mesmas, e portanto ficou fora do escopo deste trabalho.

3.2 Soluções de tradução entre protocolos

Visto que muitas vezes dispositivos estão atrelados a um único protocolo, ou impõem limitações sobre a forma de utilização do protocolo, ou utilizam protocolos de aplicação semelhantes porém sobre uma camada de comunicação distinta, se faz necessária a criação de ferramentas especializadas para a integração dos mesmos com uma plataforma. Tais ferramentas podem ser vistas como uma solução de Middleware, e podem constituir uma ferramenta de integração relativamente versátil.

A integração de dispositivos de terceiros com a plataforma Endrixx utilizada pela Sensorweb está limitada pelos protocolos suportados na plataforma, que no caso são HTTP-POST com payloads em formato *form-data*, e HTTP-GET com dados de retorno que devem ser interpretados com o uso de *expressões regulares*.

3.2.1 Objetivo e Escopo

Teve-se como objetivo para esta etapa do trabalho elaborar ferramentas constituindo um middleware que permitisse a integração de dispositivos à plataforma Endrixx através da tradução entre protocolos.

Como escopo, limitou-se a atividade ao o uso de ferramentas prototipagem rápida como scripts e serviços prontamente disponíveis.

Identificou-se grande potencial no uso de ferramentas da Amazon AWS, em parte também devido à familiaridade de boa parte da equipe da empresa com as mesmas. Atualmente, os serviços de máquinas virtuais da Amazon são utilizados para hospedar os serviços da Sensorweb.

3.2.2 Revisão dos serviços da Amazon AWS

3.2.2.1 Instâncias EC2

Um dos principais serviços da Amazon consiste no uso de maquinas virtuais em ambiente de nuvem, sem custos de aquisição de hardware. Estas instâncias podem ser acessadas remotamente, configuradas para fornecer serviços web, e executar praticamente qualquer tipo de aplicação.

Conjecturou-se o uso deste serviço para hospedar uma aplicação responsável pela interface entre diferentes protocolos, respondendo aos dispositivos em seu protocolo e em seguida fazendo o re-envio dos dados de telemetria ao Endrixx no protocolo correto.

3.2.2.2 IoT Core

Como visto em 2.2.1, a Amazon conta com um serviço de broker MQTT com a capacidade de integração a outros serviços.

3.2.2.3 Lambdas

O serviço de Lambdas da Amazon constitui um ambiente de execução de tarefas *serverless*, ou seja, sem uma alocação permanente de recursos, permitindo uma tarifação pelo uso efetivo de recursos. Lambdas são essencialmente scripts em Python ou em Node.JS que são executados em resposta a requisições de outros serviços da Amazon, tendo acesso a web e integração com os demais recursos da AWS.

A interface para a criação de funções Lambda da Amazon permite rapidamente criar e configurar novas funções, programar a função diretamente no navegador, assim como definir cenários e teste com dados fictícios. Por fim, é possível medir o uso de recursos para se estimar o custo da utilização do serviço.

3.2.2.4 API Gateway

O API Gateway é um serviço da Amazon que permite definir uma API HTTP em determinado endereço e fazer o redirecionamento de páginas ou o disparo de outros serviços, como Lambdas, em resposta às requisições realizadas aos endereços definidos.

Quando utilizado em conjunto com Lambdas (e possivelmente algum serviço de base de dados), é possível obter o comportamento semelhante ao de um servidor em uma aplicação *serverless*.

3.2.3 Solução MQTT - HTTP

Para uma integração do Endrixx com o protocolo MQTT, foi elaborada uma solução a partir do protótipo de integração realizado em 3.1.4, fazendo ainda uso do dispositivo prototipado em 3.1.3.

A integração entre os dois protocolos (MQTT e HTTP) foi feita através do uso do broker MQTT do serviço IoT Core da Amazon, e funções Lambda disparadas pelo recebimento de mensagens.

A integração se dá da seguinte forma:

1. O dispositivo se conecta ao broker do IoT Core.

2. As mensagens enviadas ao broker são processadas por regras e disparam a execução de uma função Lambda.

3. A função Lambda recebe o conteúdo da mensagem e, após as modificações necessárias, faz o envio dos dados por HTTP-POST ao Endrixx.

Para esta integração, o dispositivo foi programado para se conectar ao broker do serviço IoT Core, e realizar o envio de mensagens em determinado tópico via MQTT, contendo a leitura da sonda de temperatura já no formato esperado pela aplicação Endrixx. Foi criada uma regra no IoT Core que é executada a cada mensagem que chega no tópico especificado e causa a execução de uma função Lambda com a mensagem inteira como parâmetro. Por fim, a função executada faz uma requisição HTTP-POST ao Endrixx, fazendo assim o envio dos dados do corpo da mensagem.

Para limitar o escopo da atividade realizada, o corpo da mensagem enviada pelo dispositivo já se encontra no formato esperado pela aplicação, ou seja, *x-www-form-urlencoded*, sendo efetuado somente uma redireção de mensagens entre um protocolo e outro. Nota-se que para a recepção de uma mensagem não-JSON como argumento em uma aplicação Lambda, a mesma é codificada em *base64* pela regra que fez o disparo da função no IoT Core.

3.2.4 Solução HTTP - HTTP

Para a integração entre o dispositivo ITG da Khomp (2.4.2) e a plataforma Endrixx, foi implementada uma solução que faz o recebimento de dados do dispositivo conforme a sua API específica em HTTP, para em seguida fazer o re-envio destes dados para a plataforma no formato esperado pela mesma (sua própria API).

Esta solução foi realizada com o uso do serviço de API Gateway da Amazon e funções Lambda disparadas pelos acessos realizados pelo dispositivo.

O dispositivo ITG da Khomp faz requisições aos seguintes endereços do servidor ao qual está configurado para a transmissão de dados:

- <server_address>/api/v1_1/json/gateways/connection_status
  (status da conexão)
- <server_address>/api/v1_1/json/gateways/signal_status
  (status da rede móvel)
- <server_address>/api/v1_1/json/gateways/data
  (dados dos sensores)

A integração se dá da seguinte forma:

1. O dispositivo faz requisições HTTP a diferentes endereços no API Gateway.

2. Os acessos ao API Gateway provocam a execução de funções Lambda, passando parâmetros importantes como o endereço acessado e o corpo da mensagem.

3. Conforme o tipo de mensagem recebido, a função Lambda provê a resposta adequada a ser retornada pelo API Gateway.

4. Para o caso de uma requisição de envio de dados, a função Lambda interpreta as informações recebidas e constrói a requisição no formato adequado a ser enviada à plataforma Endrixx.

5. A função Lambda faz qualquer envio necessário à plataforma Endrixx.

Para esta integração, foi criada uma API de tipo *proxy* no API Gateway, permitindo que todos os acessos a sub-endereços em um caminho definido sejam processados pela mesma função Lambda, abrangendo assim todos os endereços utilizados com uma única configuração. Esta API foi então configurada para realizar uma chamada de função Lambda, passando o conteúdo da mensagem e o endereço acessado. Por fim, a função Lambda criada foi programada para identificar o tipo de operação realizado com base no endereço acessado, e tomar as ações necessárias em cada caso.

O dispositivo ITG da Khomp faz o envio de dados de sensores em formado JSON, com os identificadores, valores e timestamps das leituras de cada sensor. O processamento necessário para converter estas informações em um formato válido para a plataforma Endrixx consiste em construir o nome para a variável lida com base no identificador do dispositivo e do sensor, em re-formular a timestamp presente no formato Unix para o formato aceito pela plataforma, e por fim aglomerar os dados no formato de pares chave-valor a ser enviado.

3.2.5 Avaliação do funcionamento de middleware com funções Lambda

Ambas as implementações realizadas obtiveram sucesso no seu funcionamento com os dispositivos testados. Os dados dos dispositivos foram recuperados com sucesso na plataforma Endrixx através do protocolo HTTP padrão para entrada de dados da plataforma sem a necessidade de modificações no sistema.

3.2.5.1 Agilidade de desenvolvimento

As atividades realizadas aconteceram de forma bastante rápida e ágil, tomando cerca de um dia para a arquitetura da solução e um dia para a sua implantação e teste, e não causaram qualquer parada em outros serviços prestados pela plataforma Endrixx.

A interface utilizada para a criação e configuração de funções Lambda permite rapidamente criar funções a partir de funções existentes, permitindo a reutilização de

código. De fato, a implementação da função Lambda da solução HTTP-HTTP foi feita a partir da solução anterior, somente sendo acrescentadas as modificações necessárias.

Esta agilidade de desenvolvimento permite que uma solução com a mesma arquitetura seja rapidamente repetida com baixo esforço para novos modelos de dispositivos em uma abordagem específica para cada caso.

3.2.5.2 Escalabilidade e custo

O serviço de Lambda da Amazon permite a execução de um número virtualmente ilimitado de funções em paralelo, permitindo que seja aumentado o número de dispositivos conectados sem que haja colisões no acesso à plataforma e consequente degradação do serviço.

A tarifação destes serviços é feita pelo uso de recursos durante a execução da função, sendo este calculado como o produto entre a memória alocada e o tempo de execução da função. Por fim, outros serviços como mensagens MQTT e o disparo de ações no IoT Core, bem como o trafego de dados de requisições HTTP são tarifados.

Uma estimativa de custo para a utilização de uma solução MQTT - HTTP como a implementada é apresentada a seguir.

Cenário:

- Um dispositivo MQTT permanece conectado 24h/7 e faz o envio de uma mensagem a cada 5 minutos (105120 mensagens por ano).
- O broker MQTT executa uma regra em cada mensagem recebida, e esta regra dispara uma função Lambda
- Uma função Lambda é executada por mensagem recebida, utilizando 128 MB de memória por 500 milissegundos (valores observados durante os testes com margem).
- A função Lambda utiliza 5 kB de dados para realizar o envio da mensagem ao Endrixx.

Serviço IoT Core - conectividade, mensagens e regras	$0.178656
Serviço Lambda - execução e recursos	$0.133174426
Outros - tráfegos de dados	$0.047304
Total por dispositivo-ano	**$0.423980851**

Tabela 1 – Custo em USD para a solução MQTT - HTTP.

Neste cenário, o custo de operação desta solução, por dispositivo, por ano, é apresentado na Tabela 1.

O cálculo de custo para a solução HTTP - HTTP se dá de forma semelhante:

Cenário:

- Um dispositivo se conecta ao API Gateway e faz o envio de uma mensagem a cada 5 minutos (105120 mensagens por ano).
- Uma função Lambda é executada por mensagem recebida, utilizando 128 MB de memória por 500 milissegundos (valores observados durante os testes com margem).
- A função Lambda utiliza 5 kB de dados para realizar o envio da mensagem ao Endrixx.

Neste cenário, o custo de operação desta solução, por dispositivo, por ano, é apresentado na Tabela 2.

Serviço API Gateway - requests HTTP	$0.36792
Serviço Lambda - execução e recursos	$0.133174426
Outros - tráfegos de dados	$0.047304
Total por dispositivo-ano	$0.548398426

Tabela 2 – Custo em USD para a solução HTTP - HTTP.

3.3 Considerações

Observa-se das especificações das plataformas analisadas que são impostas restrições sobre o funcionamento da aplicação, como a definição de APIs organizadas em endereços ou tópicos específicos, e formatos de mensagens predefinidos. Estas restrições presumivelmente partem da suposição de que o desenvolvedor da aplicação tem controle sobre o hardware e firmware do dispositivo a ser conectado, o que pode ser falso em um cenário onde se busca a interoperabilidade entre dispositivos e plataformas de terceiros em um ambiente onde ainda não existe uma padronização de mensagens.

O uso de ferramentas de middleware para esta integração permite contornar alguns destes problemas, ao custo de tempo de desenvolvimento, treinamento, e supostamente custo de manutenção posterior. Este tipo de solução no entanto é suficientemente repetível e escalável para justificar o seu uso em integrações de sistemas.

4 Implementação de módulo MQTT

Um dos objetivos principais do trabalho realizado na empresa Sensorweb foi a implementação de um módulo para a recepção de dados via MQTT no software SCADA utilizado na empresa, o Endrixx.

Baseado em uma análise prévia a este trabalho pela equipe de desenvolvimento da empresa, justificado pelas tendências observadas em 2.4 e pelas análises feitas em 2.6, o protocolo MQTT é uma escolha interessante para a integração de soluções de telemetria na presente situação do universo IoT.

Neste capítulo é primeiramente definido em mais detalhe o escopo do trabalho planejado. São estabelecidas as especificações do que deve ser implementado na plataforma. Em seguida, aprofunda-se sobre a estrutura interna do Endrixx, para favorecer o entendimento das atividades realizadas neste trabalho. Por fim, as atividades de projeto e programação são explicadas.

4.1 Objetivo e escopo

Foi definido para trabalho o objetivo de implementar no software Endrixx um módulo para a recepção de dados de sensores através do protocolo MQTT.

Este módulo deve idealmente ser compatível com a maneira como a plataforma Endrixx é atualmente utilizada, sem maiores modificações na arquitetura da solução atual.

Prevê-se que este módulo deve ser implementado de forma semelhante aos demais módulos para a recepção de dados já presentes no software Endrixx, mantendo uma estrutura organizada da solução e facilitando o treinamento dos usuários.

Este módulo deve ser projetado para abranger o máximo de dispositivos e plataformas que suportam o protocolo MQTT dentre aqueles listados em 2.

4.2 Estrutura interna do Endrixx

Para familiarizar o leitor com software Endrixx como um todo e permitir um melhor entendimento do projeto proposto, aqui é apresentada de forma simplificada a estrutura global do software Endrixx.

O software Endrixx utilizado na empresa é um software SCADA. Ele é uma solução integrada programada majoritariamente em Java e executado como um servlet Java em um servidor. Isto permite que várias instâncias sejam executadas em paralelo na mesma

máquina em endereços diferentes sem que haja conflitos de porta.

O software Endrixx, de forma semelhante a muitos softwares SCADA, pode ser dividido em diversos módulos com funcionamento quase independente. De um ponto de vista simplificado, a divisão se dá como:

- Interface Web
- Gestão de usuários e permissões
- Módulos de entrada de dados
- Gestão de alarmes e eventos
- Registro de dados históricos
- Relatórios e Auditorias
- API e Batch Scripting

Dos módulos listados acima, muitos ainda podem ser divididos em *runtime* e *persistência*, separando o funcionamento de execução e de configuração.

No restante desta seção serão apresentados superficialmente os módulos de maior importância para este trabalho.

4.2.1 Interface Web

O Endrixx conta com uma interface web que pode ser acessada remotamente, permitindo não somente o acompanhamento e monitoramento de sistemas, bem como a configuração dos demais módulos do sistema, que normalmente opera em máquinas virtuais "headless" na nuvem.

A interface é implementada com uso das ferramentas *Spring MVC* com páginas em *JSP* e utilizando *DWR*. Isto permite a exibição de páginas dinâmicas com o uso de Java e a execução remota de métodos no serviço a partir de interações no navegador.

4.2.2 Módulos de entrada de dados

O Endrixx possui um sistema de aquisição de dados bastante versátil. O sistema conta com uma vasta biblioteca de módulos de diferentes protocolos, conhecidos como *Data Sources*, os quais podem ser instanciados e configurados pelo usuário a partir da interface web para a recepção de dados de dispositivos.

Internamente, cada tipo de módulo de dados é implementado como um conjunto de classes Java:

- **DataSourceRT** – Inicialização, execução e terminação das rotinas de aquisição de dados.
- **DataSourceVO** – Leitura, validação e escrita das configurações necessárias para o funcionamento das rotinas de aquisição de dados.
- **PointLocatorRT** – Uma instância de um valor monitorado, permite acesso a métodos para o registro dos valores recebidos.
- **PointLocatorVO** – Leitura, validação e escrita das configurações necessárias para a interpretação e armazenamento dos dados recebidos em relação a um valor monitorado.

Estas classes são instanciadas por um *Runtime Manager* que durante a inicialização do sistema fica responsável pela inicialização dos módulos de aquisição de dados configurados.

4.2.3 Interface de Configuração de Data Sources

A criação e configuração de novos *datasources* é feita a partir da página de *Data Sources* do Endrixx, onde é possível visualizar os datasources existentes, modificá-los, excluí-los, e criar novos datasources selecionando um dos protocolos disponíveis.

A criação de um novo datasource abre a janela de edição de datasources, que é a mesma para a modificação de datasources já existentes. Nesta página, algumas configurações universais a todos os tipos de datasources, bem como configurações específicas àquele protocolo. Nesta página também são criados e configurados os *datapoints* que identificam as variáveis recebidas do dispositivo, bem como quaisquer parâmetros necessários para a extração destas variáveis dos dados enviados pelo dispositivo.

Ao clicar no botão de *salvar*, os valores inseridos na página são recuperados via javascript, e através de uma chamada DWR, um objeto *DataSourceVO* do tipo selecionado é criado e os valores inseridos são configurados e por fim validados. No caso de sucesso na validação, o sistema instancia um objeto *DataSourceRT* correspondente. É então possível iniciar o Data Source, e começar o recebimento de dados.

4.3 Especificações

Das análises de dispositivos e plataformas efetuadas em 2.2 e 2.4, foram levantadas diversas especificações. Estas buscam garantir a interoperabilidade do módulo criado com o maior número de plataformas e dispositivos possível. Estas especificações também definem quais devem ser os parâmetros configuráveis pelo usuário para permitir o suporte a estas plataformas.

4.3.1 Funcionamento e Conexão

Em primeiro lugar, foi estabelecido que, para que o Endrixx possa continuar a ser utilizado como *servlet* e instanciado diversas vezes na mesma máquina, não é possível a utilização de um *broker* MQTT devido ao compartilhamento das portas TCP do sistema.

Portanto, especificou-se que a solução seria implementada como um *cliente* MQTT a se conectar a algum broker externo. O endereço e porta do broker ao qual se conectar foram estabelecidos como parâmetros configuráveis.

4.3.2 Autenticação e Criptografia

Optou-se por incluir o a opção para o uso de criptografia, de maneira a permitir o uso de todas as plataformas estudadas. O uso de certificados especiais para o cliente e o servidor se faz necessário para estas conexões, e portanto também é previsto. Parâmetros como identificador, usuário e senha para a conexão com o broker também devem ser parâmetros configuráveis.

Durante testes iniciais, foram utilizados servidores MQTT públicos que possuíam certificados expirados. Optou-se então por incluir a opção de se ignorar o certificado apresentado pelo servidor e estabelecer a conexão mesmo assim. Quaisquer consequências em termos de segurança ficam como responsabilidade do usuário.

4.3.3 Outras opções de conexão

Observou-se que alguns serviços de Broker MQTT impões restrições a parâmetros como o intervalo entre mensagens do tipo *keep-alive* do protocolo MQTT. Assim, foram expostos como parâmetros o intervalo entre mensagens *keep-alive* e também o *timeout* para a conexão com o servidor.

4.3.4 Recebimento de mensagens

O recebimento de mensagens MQTT envolve a inscrição em tópicos (2.6) para a separação de mensagens de diversos dispositivos e aplicações. O uso de *wildcards* permite a inscrição a múltiplos tópicos simultaneamente, podendo assim receber dados de múltiplos dispositivos ao mesmo tempo. Desta forma, o tópico, com suporte a *wildcards*, é um parâmetro importante para a utilização do protocolo.

Vários provedores de MQTT impões restrições sobre o *QoS* utilizado, portanto a configuração de QoS para a inscrição realizada também está exposta como parâmetro ao usuário.

Por fim, faz sentido permitir o agrupamento de diversos dispositivos similares em um único *Data Source*, sendo os dispositivos representados por *Data Points*. Desta forma,

prevê-se o uso de tópicos mais genéricos (com *wildcards*) para o *Data Source*, e um segundo mecanismo de separação para cada *Data Point*.

4.3.5 Recuperação dos dados

Todas as plataformas listadas em 2.2, bem como a maior parte dos dispositivos listados em 2.4, utilizam documentos JSON para o envio de dados. Este será portanto o formado suportado para as mensagens recebidas. Estes documentos tem diferentes formatos e não permitem a recuperação dos dados de sensores de uma maneira unificada (como mencionado em 2.5.2). É necessário permitir portanto que seja configurado o endereço dentro do documento JSON onde o valor de cada ponto deve ser recuperado. O mesmo se aplica para a recuperação da *timestamp* do valor lido.

Esta configuração se refere já aos pontos de dado lidos em cada mensagem, sendo portanto uma configuração associada ao *Data Point* e não mais ao *Data Source*.

4.3.6 Interpretação dos dados

Os valores de um sensor de telemetria podem assumir diversas formas, como valores binários *on-off*, números inteiros, números com vírgula, e mensagens de texto, entre outros. A seleção do tipo de valor para uma correta interpretação dos dados é portanto uma opção do *Data Point*.

Da mesma forma, o instante de leitura de um sensor pode ter diferentes representações. Dentre os sensores analisados, foram encontradas implementações de timestamps baseadas no tempo desde o *unix epoch* medido em segundos, milissegundo e microssegundos. Outras formas de representação possíveis utilizam datas em formatos baseados no *ISO 8601*. Estabelece-se que todos estes formatos devem ser suportados. Imaginando que para dado conjunto de dispositivos, o formato adotado seja o mesmo, esta será uma configuração do *Data Source*.

Por fim, para o caso de dispositivos que usam o formato *ISO 8601* sem informações de fuso horário, é necessário que esta informação seja incluída nas configurações para permitir que o Endrixx realize a conversão necessária.

4.4 Detalhes da implementação

4.4.1 Escolha de bibliotecas

Para a implementação deste módulo de comunicação MQTT em Java, foi escolhido uma biblioteca de cliente MQTT *open-source* bem reputada, o *Eclipse Paho*. Esta biblioteca

lida com todos os aspectos da conexão com o broker MQTT, o envio e recebimento de mensagens.

Para a criação de *sockets TCP* com *TLS* e o uso de certificados foram utilizadas as funções da suite *JSSE* (`javax.net.ssl`) com o auxílio de um utilitário disponibilizado pela Amazon (`com.amazonaws.auth.PEM`) para a leitura de certificados de forma mais versátil.

Para a interpretação das mensagens, leitura dos documentos JSON e a recuperação dos dados dentro do documento JSON recebido, foi utilizada a biblioteca *JsonPath* (`com.jayway.jsonpath`). Esta biblioteca permite realizar *queries* em documentos JSON e recuperar dados encontrados no caminho informado. Esta biblioteca utiliza uma linguagem de *queries* que permite endereçar campos em documentos JSON complexos de forma bastante versátil.

4.4.2 Data Source e Data Points

Aqui é explicado o conteúdo dos principais arquivos de código do módulo implementado. O prefixo *JSONMQTT* foi adotado para refletir o tipo específico de comunicação que o módulo suporta, isto é, documentos JSON pelo protocolo MQTT. As classes aqui descritas derivam de suas respectivas primitivas dentro do modelo de objeto utilizado no sistema conforme apresentado em 4.2.

4.4.2.1 JSONMQTTPointLocatorVO

Esta classe guarda as configurações que são específicas a cada ponto, listadas a seguir:

- Tipo de dado – Um número identificando o tipo de valor lido como binário, inteiro, numérico ou texto.
- Filtro de tópico – Um tópico MQTT para a filtragem de mensagens.
- Path do valor – Uma *query string* na sintaxe prevista pela biblioteca JsonPath para encontrar a informação do valor.
- Path da timestamp – Idem acima, para encontrar a informação da timestamp.

Os métodos desta classe estão ligados aos *gets* e *sets* para cada parâmetro, bem como a serialização e de-serialização destas informações para a sua persistência em disco. Ha também um método de validação que é invocado após a criação e edição do ponto para validar os parâmetros inseridos.

Outros campos herdados nesta classe são compartilhados com os pontos de demais módulos para as funcionalidades comuns aos pontos de dados, como a sua identificação global no sistema.

4.4.2.2 JSONMQTTPointLocatorRT

Esta classe representa um ponto monitorado. No caso específico deste módulo, a sua implementação é mínima, pois não se especificou qualquer método para realizar a *escrita* dos valores do ponto, somente o seu monitoramento.

4.4.2.3 JSONMQTTDataSourceVO

Esta classe guarda as configurações necessárias para o funcionamento do cliente MQTT e aspectos gerais da aquisição de dados, como é apresentado a seguir:

- URL e porta do Broker MQTT – O endereço do serviço ao qual se conectar.
- Identificador, usuário e senha – Os parâmetros para a conexão com o Broker MQTT.
- Opção do uso de SSL, certificado e senha de cliente, certificado de servidor – Necessários para o estabelecimento de uma conexão segura.
- Intervalo de Keep-Alive e Timeout – Parâmetros de funcionamento do protocolo MQTT.
- Filtro de tópico e QoS – Parâmetros para a recepção de mensagens MQTT.
- Formato de Data e Fuso – Parâmetros para a interpretação dos Timestamps dos valores recebidos.

Os métodos desta classe estão ligados aos *gets* e *sets* para cada parâmetro, bem como a serialização e de-serialização destas informações para a sua persistência em disco. Ha também um método de validação que é invocado após a criação e edição do data source para validar os parâmetros inseridos.

4.4.2.4 JSONMQTTDataSourceRT

Esta é a classe principal do módulo implementado. Ela contém os métodos de inicialização, operação e finalização do protocolo. A operação desta classe é orquestrada pelo *Runtime Manager*, responsável por inicializar e encerrar o sistema de aquisição de dados, entre outros.

A rotina de inicialização (`JSONMQTTDataSourceRT.initialize`) consiste em preparar os objetos que serão utilizados na operação do protocolo, como:

- O cliente MQTT com os parâmetros para conexão.
- Os objetos ligados ao uso de SSL e certificados.
- As configurações para o *parser* JsonPath.
- Os objetos para a leitura de timestamps e timezones.

A ativação do módulo se dá pelo método `JSONMQTTDataSourceRT.beginPolling`, que estabelece a conexão do cliente MQTT com o broker. Este método resolve imediatamente, sendo o estabelecimento da conexão concluído de forma assíncrona. Após estabelecida a conexão, o cliente se inscreve no tópico configurado e começa a receber mensagens MQTT dos sensores.

O recebimento de mensagens acontece de forma assíncrona, disparando uma chamada ao método `JSONMQTTDataSourceRT.messageArrived`. Após o *parsing* do documento JSON é percorrida a lista dos *Data Points* configurados, e é aplicado o filtro de tópico para cada um. No caso de um resultado positivo deste filtro, é efetuada a *query* no documento para recuperar o valor lido do ponto, e de forma semelhante a sua timestamp.

Os valores lidos do documento JSON são interpretados conforme o formato definido na configuração do tipo de dado daquele ponto. A timestamp lida é interpretada segundo o formato customizado fornecido, sendo feito o uso da classe Java `java.text.SimpleDateFormat` para a interpretação do formato e da data lida. Caso o formato não seja informado se assume uma timestamp Unix em segundos. Caso a data resultante por este último método seja posterior ao ano 2970, assume-se uma timestamp Unix em milissegundos ao invés de segundos. Novamente, o valor é re-interpretado como microssegundos caso a data resultante ainda seja posterior ao ano 2970.

O fim da operação deste módulo se dá em duas etapas, uma assíncrona e uma bloqueante (*join*), que cancelam a inscrição para a recepção de mensagens e desconectam o cliente do servidor para a finalização do sistema.

4.4.3 Interface de configuração

O módulo implementado é configurado através da interface web do Endrixx, sendo a configuração realizada em uma página contendo o formulário apresentado em Figura 11. Os campos para a configuração são escritos em HTML com uso de Java (JSP) para nomes de atributos e o carregamento de valores já salvos nas configurações.

O arquivo em questão também contem o código JavaScript que executa no navegador para realizar as chamadas remotas via DWR que fazem a configuração e validação dos objetos *DataSourceVO* e *PointLocatorVO* com os dados inseridos. As funções equivalentes

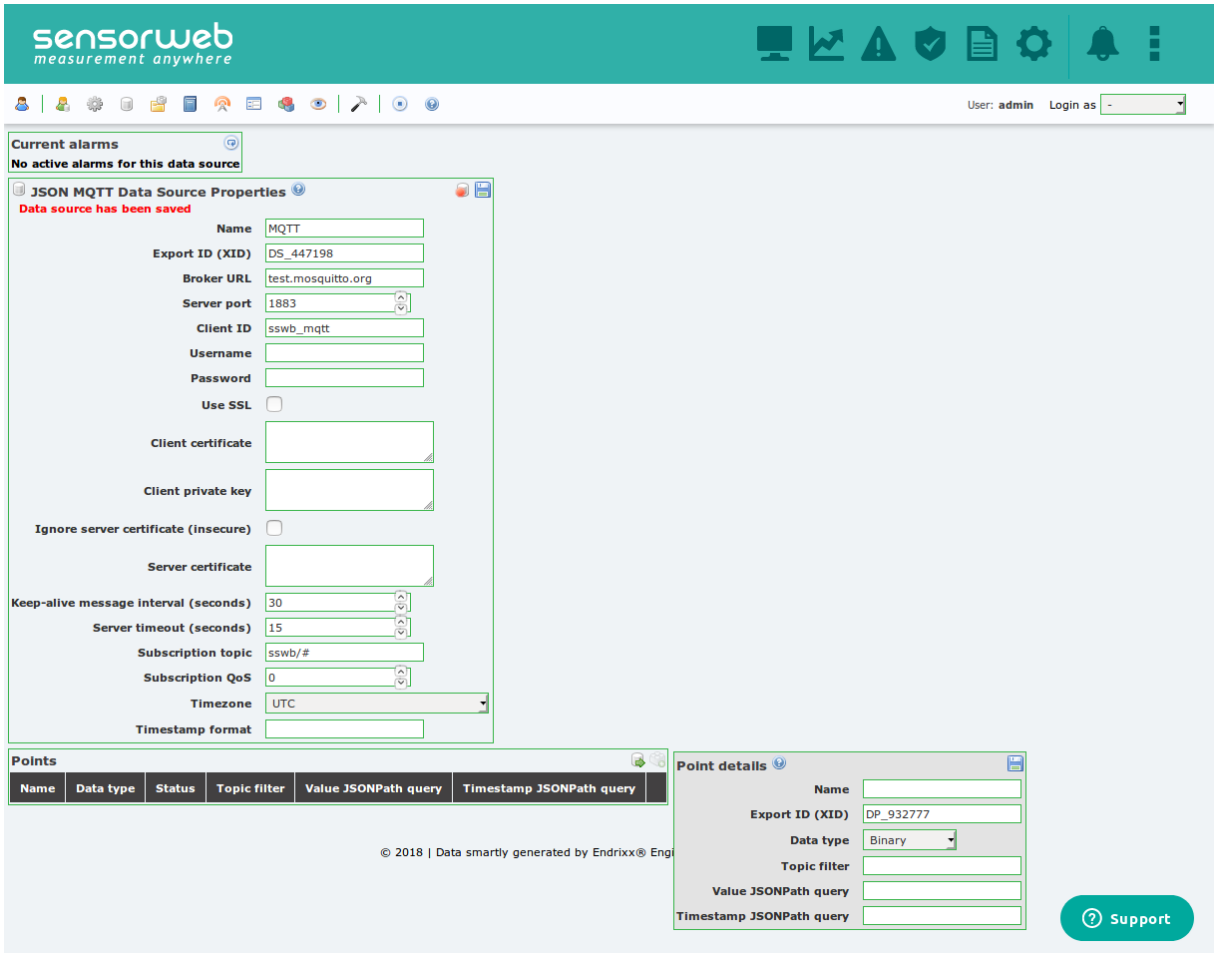

Figura 11 – Página para a configuração do Data Source MQTT.

em Java para as chamadas DWR foram implementadas em um arquivo compartilhado com os demais Data Sources para tal fim.

Todos os textos da interface são definidos por identificadores para que o texto adequado seja carregado dos arquivos de localização com a língua utilizada durante a execução.

4.5 Avaliação do funcionamento

Para a avaliação do funcionamento e da interoperabilidade do módulo criado, foram testados diversos dispositivos conectados via MQTT, assim como diversos Brokers MQTT.

Os testes foram realizados em um computador com Ubuntu, hosteando uma instância do software Endrixx na própria rede local da Sensorweb. Devido ao funcionamento de *cliente* do módulo implementado, não foi necessária qualquer intervenção no firewall ou regras de portas nos roteadores da empresa.

4.5.1 ESPeixe

Em uma implementação muito semelhante a 3.1.4, o dispositivo *ESPeixe* foi conectado ao Broker da Amazon IoT Core, fazendo o envio periódico de uma mensagem com o valor da temperatura lida.

Após a geração de permissões e certificados no serviço IoT Core, foi criado um Data Source do tipo *JsonMQTT*, e este foi configurado para se conectar ao mesmo broker. Criou-se um único ponto de dado, correspondente ao valor lido pela sonda de temperatura. Tópico e caminho para o valor foram configurados conforme o implementado no dispositivo. O caminho para a timestamp foi deixado em branco, sendo este caso tratado em software como um caso excepcional, e sendo usado o instante de recebimento da mensagem para o registro do valor.

O resultado obtido foi conforme o esperado, os valores enviados pelo dispositivo sendo recebidos no software Endrixx e sendo armazenadas com o instante do recebimento da mensagem.

4.5.2 Smetro WiFi

O dispositivo Smetro WiFi permite a configuração de endereço e porta do broker MQTT ao qual ele se conecta, bem como os campos de usuário e senha. Ele no entanto não tem suporte a criptografia. Para este teste, um dos dispositivos disponibilizados pela Smetro foi configurado para se conectar a um broker público mantido pelo projeto open-source Mosquitto, ligado ao projeto Eclipse.

Foi criado no Endrixx um Data Source, configurado para se conectar ao broker `test.mosquitto.org` na porta padrão 1883, sem o uso de criptografia ou autenticação. O tópico de inscrição foi definido de forma genérica para atender a todos os dispositivos Smetro.

Um ponto de dado foi criado, com o filtro de tópico configurado com o número de série do dispositivo utilizado. O caminho para o valor dentro do documento JSON enviado foi definido como é ilustrado em Bloco de Código 4.1. Novamente, os resultados foram exatamente como o esperado, com a recepção correta dos dados.

4.5.3 Novus WiFi

O dispositivo Novus Wifi possui um software de configuração que permite definir o endereço do Broker MQTT ao qual ele se conecta, assim como porta, usuário e senha. Para este teste, o dispositivo disponibilizado pela Novus foi configurado para se conectar a um broker público mantido pela HiveMQ, uma versão empresarial de broker MQTT.

De forma semelhante ao teste anterior, foi criado um Data Source configurado para

Bloco de Código 4.1 – Payload Smetro e respectivo JSONPath

```
Payload:
{
    ...
    "s0":{
        "value":27.0,
        "timestamp":1543047700000,
        "context":{
            "id":11539632678245893
        }
    }
    ...
}

JsonPath:
    $[?(@.context.id == 11539632678245893)].value
(elemento da raiz cujo context.id seja o procurado,
    então elemento value)
```

Bloco de Código 4.2 – JSONPath para payload Novus

```
JsonPath:
    $.value_channels[1]
(elemento da raiz value_channels, índice 1 no vetor)
```

se conectar ao broker `broker.hivemq.com`, sem o uso de criptografia ou autenticação. O tópico de inscrição foi definido de forma genérica para atender a todos os dispositivos Novus.

Um ponto de dado foi criado, com o filtro de tópico configurado para receber somente mensagens de log do dispositivo utilizado. O caminho para o valor dentro do documento JSON enviado foi definido a partir da estrutura da mensagem enviada pelo dispositivo, disponível em 2.4.3 e é apresentado em Bloco de Código 4.2. Novamente, os resultados foram exatamente como o esperado, com a recepção correta dos dados.

4.6 Considerações

Sendo o uso de software SCADA essencial para a solução oferecida pela empresa Sensorweb, a implementação de soluções de interoperabilidade dentro do próprio software utilizado é naturalmente preferível frente a outras soluções envolvendo ferramentas externas.

A complexidade envolvida na extensão de uma ferramenta tão grande e integrada como um software SCADA torna este caminho mais lento, e requer escolhas bem justificadas para que o resultado seja suficientemente versátil e abrangente para a integração do maior número possível de dispositivos comunicantes.

Os resultados obtidos, no entanto, sugerem que o trabalho proposto foi realizado corretamente e com a versatilidade desejada, sendo possível conectar três dispositivos completamente não-relacionados com a plataforma, fazendo uso somente da parametrização

do módulo programado, sem alterações de código.

5 Conclusões

5.1 Uso de plataformas IoT

Diversas plataformas se propõem como ferramentas de integração para soluções de IoT. Enquanto estas plataformas de fato podem funcionar como agregadoras, coletando informações de diversos dispositivos, as restrições comumente impostas sobre os protocolos de comunicação e mensagens suportados restringem significativamente o seu caso de uso. Várias destas restrições somente podem ser contornadas com grande controle sobre o firmware dos dispositivos comunicantes em questão, o que raramente é o caso quando se busca a integração de dispositivos manufaturados por terceiros.

O uso destas plataformas traz a vantagem da terceirização de muitas das preocupações com escalabilidade e manutenção. Enquanto os custos de operação de algumas delas são bastante significativos, o serviço contratado se propõe a suportar grandes quantidades de dispositivos em funcionamento ininterrupto.

Nota-se no entanto que para as implementações realizadas, integrações com estas plataformas somente foram possíveis com hardware e firmware próprios, customizados para cada plataforma, o que ilustra as dificuldades em se obter interoperabilidade em soluções de IoT ainda no momento.

5.2 Uso de tradutores de protocolo

O uso de ferramentas de middleware para a integração entre sistemas é bastante útil em cenários onde faz-se uso de mecanismos e mensagens padronizados, como é o caso de sistemas que seguem EIPs (*Enterprise Integration Patterns*). No entanto, para cenários heterogêneos no que diz respeito ao formato de mensagens, não existem ferramentas prontas de integração, e se faz necessário o uso de mecanismos mais versáteis, porém com maior custo de desenvolvimento.

A solução escolhida para a implementação deste tipo de middleware, o serviço de Lambda da Amazon, provê os mecanismos para que a solução realizada seja facilmente modificável e replicável. O serviço oferecido é também consideravelmente escalável.

Para as duas soluções realizadas, observou-se a necessidade de serviços adicionais para que se realize a completa interface com o dispositivo, nominalmente o API Gateway para o protocolo HTTP, e o IoT Core para o protocolo MQTT. O uso de tais conjuntos de serviços pode atrelar uma solução a determinada ferramenta que possui as funcionalidades desejadas e restringir sua portabilidade, ou ocasionar um ponto único de falha onde as

alternativas são limitadas ou inexistentes.

5.3 Módulo MQTT para o Endrixx

Dado a complexidade de um software SCADA, a implementação de módulos adicionais envolve um significativo esforço de desenvolvimento. A necessidade de compreender o funcionamento de tal software e os ciclos de vida de seus componentes impõe um barreira inicial considerável ao desenvolvimento. No entanto, o nível de integração obtido por tal método é inigualável, e com uma ferramenta obtida suficientemente útil e versátil, justifica tais custos.

Em termos de escalabilidade, a modularidade do software Endrixx permite que sejam instanciadas quantos clientes o necessário para a operação do sistema, embora o funcionamento do protocolo MQTT seja tal que um único cliente pode ser utilizado para receber a comunicação de um sem-número de dispositivos. No que diz respeito à manutenção, o mecanismo implementado demonstrou ser suficientemente parametrizável e não necessita customizações em software para interoperar com diferentes dispositivos utilizando mensagens JSON/MQTT.

A pesar do sucesso obtido na implementação de um protocolo moderno e bastante adotado, a implementação realizada conta apenas com um cliente MQTT, e portanto depende de um broker externo. A grande disponibilidade de brokers MQTT open-source e gratuitos assim como pagos não torna isso um problema, mas a necessidade de componentes externos para o funcionamento da solução inerentemente aumenta a complexidade do sistema e a quantidade de possíveis pontos de falha.

5.4 Soluções propostas porém não implementadas

5.4.1 Reprogramação da placa Multiconnect

Dentre as propostas de atividades levantadas no início deste PFC, constava a proposta de incorporação do protocolo MQTT ao firmware do módulo *Multiconnect* desenvolvido pela empresa Fanem. Esta atividade permitiria a posterior integração dos dispositivos da Fanem que farão uso deste módulo com a plataforma Endrixx.

5.4.2 Integração com a pilha HL7

O protocolo HL7 é bastante utilizado por dispositivos hospitalares para o monitoramento local, porém com o uso de protocolos internet, é possível utilizá-lo para telemetria. A implementação de um módulo para a recepção de dados via este protocolo permitiria a integração de dispositivos não somente da cadeia do frio, mas sim de todo o ambiente

de um hospital conectado. A prioridade deste projeto no entanto foi considerada baixa, tratando-se de uma ação exploratória para a qual não foram investigados os ganhos efetivos que este tipo de solução traria.

5.4.3 Programação de cliente MQTT nas centrais

O uso de um protocolo mais padronizado nas centrais utilizadas na empresa seria um passo tanto na direção de uma futura troca pelo hardware de terceiros, como na direção de facilitar a integração de sistemas de monitoramento de terceiros ao sistema da Sensorweb. Este é às vezes o caso para certos clientes que desejam integrar os dados obtidos pelos sensores instalados em suas próprias soluções de monitoramento.

Referências

1 LEE, I.; LEE, K. The internet of things (iot): Applications, investments, and challenges for enterprises. *Business Horizons*, v. 58, n. 4, p. 431 – 440, 2015. ISSN 0007-6813. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0007681315000373>. Citado na página 26.

2 AL-FUQAHA, A. et al. Toward better horizontal integration among iot services. *IEEE Communications Magazine*, v. 53, n. 9, p. 72–79, September 2015. ISSN 0163-6804. Citado 2 vezes nas páginas 26 e 28.

3 SHENG, Z. et al. Recent advances in industrial wireless sensor networks toward efficient management in iot. *IEEE Access*, v. 3, p. 622–637, 2015. ISSN 2169-3536. Citado na página 27.

4 SANITáRIA, A. N. de V. *RESOLUÇÃO RDC Nº 17, DE 16 DE ABRIL DE 2010. Dispõe sobre as Boas Práticas de Fabricação deMedicamentos.* 2010. Citado na página 27.

5 VERMA, P. K. et al. Machine-to-machine (m2m) communications. *J. Netw. Comput. Appl.*, Academic Press Ltd., London, UK, UK, v. 66, n. C, p. 83–105, maio 2016. ISSN 1084-8045. Disponível em: <http://dx.doi.org/10.1016/j.jnca.2016.02.016>. Citado 2 vezes nas páginas 28 e 50.

6 ARTIGO: "Why The Internet Of Things Hasn't Gone Cellular Yet". <https://web.archive.org/web/20171022194622/https://www.fastcompany.com/3056442/why-the-internet-of-things-hasnt-gone-cellular-yet>. Acessado em 20/11/2018. Citado na página 28.

7 ISHAQ, I. et al. Ietf standardization in the field of the internet of things (iot): a survey. *JOURNAL OF SENSOR AND ACTUATOR NETWORKS*, n. 2, p. 235–287, 2013. ISSN 2224-2708. Disponível em: <http://dx.doi.org/10.3390/jsan2020235>. Citado 3 vezes nas páginas 28, 38 e 50.

8 ASIM, M. A survey on application layer protocols for internet of things (iot). *International Journal of Advanced Research in Computer Science*, v. 8, n. 3, p. 996–1000, 2017. ISSN 0976-5697. Disponível em: <http://www.ijarcs.info/index.php/Ijarcs/article/view/3143>. Citado 2 vezes nas páginas 38 e 50.

9 SHELBY, Z.; HARTKE, K.; BORMANN, C. *The Constrained Application Protocol (CoAP).* [S.l.], 2014. <http://www.rfc-editor.org/rfc/rfc7252.txt>. Disponível em: <http://www.rfc-editor.org/rfc/rfc7252.txt>. Citado na página 38.

10 ARTIGO: "Conhecendo o MQTT". <https://web.archive.org/web/20181127132333/https://www.ibm.com/developerworks/br/library/iot-mqtt-why-good-for-iot/index.html>. Acessado em 20/11/2018. Citado na página 38.

11 YASSEIN, M. B.; SHATNAWI, M. Q.; AL-ZOUBI, D. Application layer protocols for the internet of things: A survey. In: *2016 International Conference on Engineering MIS (ICEMIS).* [S.l.: s.n.], 2016. p. 1–4. Citado 2 vezes nas páginas 39 e 50.

12 CRUZ, M. A. da et al. Performance evaluation of iot middleware. *Journal of Network and Computer Applications*, v. 109, p. 53 – 65, 2018. ISSN 1084-8045. Disponível em: <http://www.sciencedirect.com/science/article/pii/S108480451830064X>. Citado na página 40.

13 BLACKSTOCK, M.; LEA, R. Iot interoperability: A hub-based approach. In: *2014 International Conference on the Internet of Things (IOT)*. [S.l.: s.n.], 2014. p. 79–84. Citado na página 40.

14 JENNINGS, C. et al. *Sensor Measurement Lists (SenML)*. [S.l.], 2018. Citado na página 47.

15 KARAGIANNIS, V. et al. A survey on application layer protocols for the internet of things. *Trans. IoT Cloud Comput.*, v. 3, p. 11–17, 01 2015. Citado na página 50.

16 KNOLLEARY. *knolleary/pubsubclient*. 2018. Disponível em: <https://github.com/knolleary/pubsubclient>. Citado na página 56.