UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE
UNIDADE ACADÊMICA ESPECIALIZADA EM CIÊNCIAS AGRÁRIAS
CURSO SUPERIOR DE TECNOLOGIA EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

ASSIS LUCAS ROMÃO BERNARDO

MEPIM: *Middleware* para Economia de Energia em Redes de Sensores sem Fios.

Macaíba
Junho, 2018

Assis Lucas Romão Bernardo

MEPIM: *Middleware* para Economia de Energia em Redes de Sensores sem Fios

Trabalho de conclusão de curso de graduação apresentado à Unidade Acadêmica Especializada em Ciências Agrárias – Escola Agrícola de Jundiaí da Universidade Federal do Rio Grande do Norte como requisito parcial para a obtenção do grau de Tecnólogo em Análise e Desenvolvimento de Sistemas.

Orientador: Prof. Dr. Taniro Chacon Rodrigues
Coorientadora: Profª. Drª. Fabiana Rodrigues de Arruda Câmara

Macaíba

Junho, 2018

Universidade Federal do Rio Grande do Norte - UFRN
Sistema de Bibliotecas - SISBI
Catalogação de Publicação na Fonte. UFRN - Biblioteca Central Zila Mamede

Bernardo, Assis Lucas Romao.
MEPIM: Middleware para economia de energia em redes de sensores sem fios / Assis Lucas Romao Bernardo. - 2018.
59 f.: il.

Monografia - (graduação) - Universidade Federal do Rio Grande do Norte, Unidade Acadêmica Especializada em Ciências Agrárias, Tecnologia em Análise e Desenvolvimento de Sistemas. Macaíba, RN, 2018.
Orientador: Prof. Dr. Taniro Chacon Rodrigues.
Coorientadora: Profª. Drª. Fabiana Rodrigues de Arruda Câmara.

1. Redes de sensores sem fios - Monografia. 2. Middleware - Monografia. 3. RSSFs - Monografia. 4. Microalgas - Monografia. 5. Economia de energia - Monografia. I. Rodrigues, Taniro Chacon. II. Câmara, Fabiana Rodrigues de Arruda. III. Título.

RN/UF/BCZM CDU 004.73

Elaborado por Ana Cristina Cavalcanti Tinôco - CRB-15/262

Assis Lucas Romão Bernardo

MEPIM: *Middleware* para Economia de Energia em Redes de Sensores sem Fios.

Trabalho de conclusão de curso de graduação apresentado à Unidade Especializada em Ciências Agrárias – Escola Agrícola de Jundiaí da Universidade Federal do Rio Grande do Norte como requisito parcial para a obtenção do título de Tecnólogo em Análise e Desenvolvimento de Sistemas.

Aprovado em: ____ de _____ de _____.

BANCA EXAMINADORA

_____

Prof. Dr. Taniro Chacon Rodrigues – EAJ/UFRN - Orientador

_____

Profª. Dra. Fabiana Rodrigues de Arruda Câmara – EAJ/UFRN - Coorientadora

_____

Profª. Dra. Laura Emmanuella Alves dos Santos Santana de Oliveira – EAJ/UFRN- Avaliador

_____

Prof. Dr. Josenalde Barbosa de Oliveira – EAJ/UFRN- Avaliador

RESUMO

As tecnologias são utilizadas como aliados na construção de soluções que poderão permitir a maximização da produtividade. A utilização de Redes de Sensores sem Fios (ou RSSFs) vêm ganhando espaço no desenvolvimento de aplicações devido à sua capacidade de construção de redes ad hoc para monitoramento e controle de ambientes. Dispositivos que compõem uma RSSF possuem restrições de energia e necessitam do uso de mecanismos para maximizarem a disponibilidade dos dispositivos que compõe a rede. O objetivo desse trabalho foi desenvolver um middleware para RSSFs capaz de aplicar políticas de economia de energia aos nós, que pertencem a rede, e controlar seus respectivos ciclos de trabalho, otimizando o consumo de energia e evitando desperdícios, assim, sendo capaz de viabilizar sua aplicação em domínios como a aquicultura. A avaliação foi realizada através do estabelecimento de métricas para quantificação da eficiência em termos de economia de energia e escalabilidade. Os resultados obtidos para a diferença de consumo entre as três políticas de uso de energia implementados foram de 74.8mA, 74.5mA e 1.78mA. Para a avaliação da escalabilidade em relação a perda de pacotes, os resultados foram de 100% de pacotes enviados sem perdas. Para a avaliação de perda de pacotes, considerando a inserção de novos nós na rede, os resultados foram de 100% dos pacotes enviados e para a avaliação de escalabilidade em relação à perda de pacotes com remoção de nós na rede durante seu funcionamento 100% dos pacotes foram enviados corretamente. Realizou-se ainda, um estudo de caso que analisou o comportamento da aplicação em um sistema para monitoramento num ambiente de cultivo de microalgas, sendo o mesmo capaz de realizar a coleta de dados físico-químicos de um ambiente.

Palavras-chave: *middleware*, RSSFs, economia, microalgas.

ABSTRACT

The technology is used as an ally in the construction of solutions that could allow the maximization of productivity. The use of Wireless Sensor Networks (or WSNs) has been gaining space in application development due to its ability to build ad hoc networks for monitoring and control of environments. Devices that make up an WSNs have energy restrictions and need to use mechanisms to maximize the availability of the devices that make up the network. The objective of this work was to develop a WSNs middleware capable of applying energy saving policies to the nodes belonging to the network and controlling their respective work cycles, optimizing energy consumption and avoiding waste, thus being able to make feasible areas such as aquaculture. The evaluation was performed through the establishment of metrics to quantify efficiency in terms of energy savings and scalability. The results obtained for the consumption difference between the three energy use policies implemented were 74.8mA, 74.5mA and 1.78mA. For the evaluation of scalability in relation to package loss, the results were 100% of packages sent without loss. For the evaluation of package loss, considering the insertion of new nodes in the network, the results were 100% of the packages sent and for the evaluation of scalability in relation to the loss of packages with removal of nodes in the network during its operation 100% of the successfully sent. A case study was carried out that analyzed the behavior of the application in a system for monitoring in a microalgae culture environment, being able to perform the collection of physicochemical data of an environment.

Keywords: middleware, WSNs, economy, microalgae.

AGRADECIMENTOS

Gostaria de agradecer primeiramente a Deus, por permitir a construção de novos laços durante o processo de formação, por ser meu maior motivador. Sou grato também por me mostrar que qualquer passo à frente, por menor que seja, é um grande avanço.

Agradeço a minha família, meu pai Francisco de Assis, minha mãe Isaura Lucia e minha vó Maria Félix a quem dedico esse trabalho. Muito obrigado por todo suporte, ensinamentos, carinho, conselhos e principalmente pelo amor compartilhado. Aos meus irmãos por todo apoio, motivação e por todas as palavras que me fazem seguir e diante.

Agradeço ao meu orientador Taniro Chacon, pelos conhecimentos transmitidos, pela dedicação, influência, atenção prestada durante o curso e no desenvolvimento deste trabalho e pela amizade.

Agradeço a todos os meus professores pelos ensinamentos, empenho e por contribuírem para meu crescimento na busca do saber.

Agradeço aos meus amigos, especialmente a: Jonatas Melo por ser meu irmão de outra mãe, por sempre estar presente e pelo companheirismo; João Paulo Araújo pelas aventuras e pelos bons momentos da infância à fase adulta; Juvanderson Nicacio por sempre estar disponível para ajudar; Washington da Silva (compadre) por ser meu freguês nos mais diversos jogos, pela boa amizade e pela ajuda que me deu no primeiro semestre do curso; Suéliton Santos (Duda) pelas dicas e conselhos para a vida e Laércio Medeiros por compartilhar sua experiência de vida, pelos conselhos e principalmente pela amizade.

Aos amigos que tive o prazer de conhecer ao longo do curso: Jonas Florêncio, parceiro e professor no mundo da eletrônica; Elidiel Dantas professor de matemática e amigo para qualquer momento; Larysse Savanna professora de inglês, amiga e companheira dos momentos difíceis do curso. Agradeço por todos os bons momentos proporcionados, por toda experiência que pude adquirir, pelos ensinamentos.

Por fim, agradeço a todos que puderam participar de alguma forma no desenvolvimento desse trabalho.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS

API	Application Programming Interface
CPU	Unidade de Processamento Central
DAO	Data Access Object
GQM	Goal Question Metric
HTTP	HyperText Transfer Protocol
IDE	Integrated Development Environment
MEPIM	Middleware for Energy Policy Management
MQTT	Message Queuing Telemetry Transport
MOM	Middleware Orientado a Mensagens
NoSQL	Not Only SQL
ORB	Object Request Broker
RPC	Remote Procedure Call
REST	Representational State Transfer
RSSFs	Redes de Sensores Sem Fios
SOA	Service Oriented Architecture
URIs	Uniform Resource Locator
UML	Unified Modeling Language
WSNs	Wireless Sensor Networks

SUMÁRIO

1. INTRODUÇÃO

Os recentes avanços tecnológicos estimulam o uso de recursos computacionais como solução para automatizar processos e consequentemente agilizar os sistemas de produção, de modo a melhorar a qualidade produtiva. No segmento agrícola, o atual uso de novas tecnologias pode ser utilizado para construir sistemas de apoio capazes de reduzir desperdícios e maximizar a produtividade (LAMAS, 2017). Nesse contexto, o uso da tecnologia como forma de auxílio, por exemplo, no monitoramento e controle remoto de equipamentos em ambiente de produção pode apoiar produtores na tomada de decisões e melhorar a produtividade na cultura a qual se aplica.

Uma Rede de Sensores Sem Fios (RSSFs) é formada por diversos dispositivos, também chamados de nós, com capacidade de comunicação através do meio sem fio com o objetivo de monitorar algum fenômeno físico, como temperatura, ou químico, como concentração de amônia, dentre ouros. As RSSFs diferem de uma tradicional rede de computadores por serem redes *ad hoc* compostas de dezenas a milhares de nós capazes de trocar informações diretamente entre si (RODRIGUES, 2015). Uma rede *ad hoc* é um tipo de rede em que seus dispositivos funcionam como roteadores, encaminhando informações vindas dos dispositivos vizinhos de forma comunitária, descartando o uso de um ponto de acesso.

Os nós de uma RSSF são compostos por um microprocessador, uma antena de rádio e, normalmente, usam uma bateria como fonte de alimentação. Tais nós, quando equipados com sensores, possuem a capacidade de capturar informações do ambiente como temperatura, umidade, luminosidade, movimento, pressão ou qualquer outra grandeza físico-química detectável. Quando equipados com atuadores, esses dispositivos podem executar tarefas como modificar parâmetros ou ativar ações a serem realizadas no ambiente monitorado (RODRIGUES, 2011).

Apesar da aplicabilidade em uma vasta gama de domínios, os nós de uma RSSF, em geral, possuem limitações severas de memória, capacidade de processamento e energia (CARVALHO, 2012; PEREIRA 2014; OLIVEIRA, 2015). Segundo (PANTAZIS, et al., 2013), é de grande importância que os nós que compõem uma RSSF tenham a vida útil de suas baterias prolongadas o máximo possível independentemente da aplicação, mesmo diante da inexistência de limitações de energia. Além disso, os nós são frequentemente dispersos em áreas de difícil acesso, o que impede que seja feita qualquer tipo de manutenção. Dessa forma, são necessários

mecanismos que possibilitem a auto-organização, adaptação e reconfiguração diante de problemas de comunicação ou perda de nós da rede e, principalmente, pelas restrições de energia (LOUREIRO et al., 2003).

Dentro do conjunto de aplicações para RSSF é possível destacar o seu uso no domínio rural. As RSSFs aliadas ao ramo agrícola e aquícola podem proporcionar ganhos de produtividade e evitar desperdícios como mostra o trabalho de Brasil (2017), onde é proposto um sistema capaz de receber dados providos por sensores, independentemente do hardware que é utilizado, e disponibilizar as informações obtidas ao produtor sobre os parâmetros de produção em uma interface web.

No contexto da aquicultura é possível destacar as aplicações para o cultivo de microalgas. As microalgas são microrganismos aquáticos que podem ter seu cultivo destinado a uma miríade de usos, como: (i) alimentação de animais e seres humanos, na forma de suplemento alimentar; e (ii) em indústrias, na forma de cosméticos ou fonte de biocombustíveis (KIRROLIA et al. 2013). Geralmente, o cultivo de microalgas se inicia em uma sala com condição de temperatura e luminosidade controladas até atingir concentração celular adequada em volume de aproximadamente 20 litros. Em seguida, a biomassa é transformada para volumes maiores em tanques alojados ou fotobiorreatores caracterizando o cultivo massivo que ocorre em áreas existentes.

O cultivo de microalgas é um processo delicado que necessita de higienização tanto das pessoas envolvidas quanto do ambiente e instrumentos a fim de evitar a proliferação de microrganismos que possam contaminar o cultivo. Além disso, como as microalgas são organismos extremamente frágeis é necessário controlar com precisão todas as características físico-químicas do ambiente de produção que afetam seu desenvolvimento como temperatura da água e ambiente, intensidade da luz, nutrientes, concentração de pH da água e salinidade (HAKALIN, 2014). Atualmente, cultivos feitos com fotobiorreatores tubulares são construídos com materiais resistentes e transparentes permitindo exposição das microalgas à luz e possuindo seções que possibilitam a injeção de CO_2.

Dessa forma, um sistema computacional para controle e monitoramento de ambientes de produção de microalgas tem grande potencial para melhorar a produção através da coleta de dados físico-químicos do ambiente, envio de tais dados ao gerente da produção, uso de atuadores para modificar o ambiente remotamente e redução do contato do ambiente de

produção com fatores externos. No entanto, tal sistema deve considerar as restrições energéticas de uma RSSF e implementar mecanismos para economia de energia para que o funcionamento dos dispositivos acompanhe todo o ciclo de produção de microalgas que dura entre 10 a 15 dias em média, dentro de laboratório.

1.1 JUSTIFICATIVA

Existem na literatura trabalhos recentes desenvolvidos como propostas de sistemas para o cultivo de microalgas como em PALOMINO (2017) e MARIANO et al. (2017).

Em PALOMINO (2017) é proposto um sistema eletrônico com o objetivo de avaliar o impacto de diferentes condições na produtividade de biomassa seca a partir do cultivo de microalgas em tanque fotobiorreator com um controle de parâmetros físicos como luminosidade, temperatura e CO_2 utilizando o microcontrolador Arduino (“Arduino”, 2017). No trabalho de (PALOMINO 2017), a fim de quantificar a produtividade de biomassa, os seguintes testes foram realizados utilizando sistemas desenvolvidos no mesmo: avaliação do cultivo sem controle de luminosidade durante o período de cultivo; cultivo com controle de luminosidade de doze horas de luz e doze horas de escuro; cultivo com presença de monitoramento de temperatura ambiental a 20Cº e cultivo com injeção de CO_2. Apesar da eficiência do sistema, são apresentadas limitações referentes a capacidade do banco de dados para armazenamento de informações e a falta de uma API (do inglês *Application Programming Interface*) para requisição em tempo real dos dados coletados pelos sensores conectados ao Arduino, além de não haver um tratamento do consumo energético gerado pelos dispositivos.

No trabalho apresentado em (MARIANO et al., 2017), os autores desenvolveram uma aplicação para o monitoramento automatizado de parâmetros no cultivo de microalgas. Nessa aplicação o sistema é capaz de realizar o monitoramento de forma automatizada, onde as análises são feitas sem supervisão após serem programadas. Para isso, esse sistema conta com dois equipamentos para controle de microalgas, para amostra individual e outro para um conjunto de amostras. Ambos equipamentos possuem, objetivos em comum, que são o controle em tempo real de pH e concentração de substâncias através da internet, mas cada um realiza uma tarefa específica. O primeiro, controlador de microalgas de uma amostra, tem a função de controlar a vazão de entrada e saída em um recipiente e o segundo, pode realizar medições de temperatura da amostra e fornecer iluminação local. Apesar dos resultados favoráveis, o trabalho de (MARIANO et al., 2017), não apresenta uma preocupação com o consumo de

energia dos microcontroladores, dessa forma a falha de um dos nós da rede poderia comprometer todo um cultivo de microalgas.

Portanto, uma solução para o problema do consumo de energia é o uso de um *middleware*. O *middleware* trata-se de uma camada de software que abstrai dispositivos e aplicações, introduzindo níveis de transparência e interoperabilidade, além de oferecer serviços para usuários finais e aplicações. Um *middleware* oculta as complexidades como heterogeneidade de *hardware*, camadas de protocolo de rede, plataformas e dependências de sistemas operacionais dos usuários, e facilita o gerenciamento de recursos do sistema. Além disso, um *middleware* fornece acesso padronizado aos dados e serviços providos pelos dispositivos inteligentes de uma rede por meio de uma interface de alto nível e ainda possibilita a reutilização de serviços genéricos (PIRES, 2014).

Diante dos problemas comumente enfrentados por desenvolvedores e pesquisadores em relação às restrições de *hardware* e em especial às limitações de energia dos nós que compõem uma RSSFs, as soluções emergentes precisam considerar a escassez dos recursos dos nós pertencentes a RSSFs para satisfazer tais demandas. Assim, o uso de plataformas de *middleware* pode contribuir para a atender a uma série de requisitos como gerencia de dispositivos, dados e de recursos.

Com o uso de um *middleware* é possível adotar algumas estratégias que possam prolongar a vida útil das baterias dos objetos que compõe uma rede. Uma possível estratégia é a aplicação de políticas de consumo de energia nos nós da rede para modificar seus ciclos de trabalho (em inglês, *duty cycle*). O ciclo de trabalho trata-se do tempo inativo pelo qual uma aplicação espera para a emissão ou recepção de dados (RODRIGUES, 2015). Em outras palavras, é a fração de tempo em que um dispositivo está em pleno funcionamento. Com a aplicação de políticas de consumo de energia, o ciclo de trabalho dos nós são modificados, desabilitando alguns componentes de *hardware* como a placa de comunicação sem fio e a unidade de processamento central (CPU) por um intervalo de tempo definido nessas políticas, ou diminuindo seu ritmo de trabalho para gerar um menor consumo de energia.

1.2 OBJETIVOS

Esta seção tem a finalidade de apresentar os objetivos deste trabalho. A seção 1.2.1 expõe o objetivo geral do trabalho que se refere a ideia central. Na seção 1.2.2 são apresentados os objetivos específicos, ilustrando o que se almeja alcançar com o desenvolvimento e pesquisa.

1.2.1 Objetivo geral

O objetivo geral do presente trabalho é desenvolver o MEPIM (do inglês *Middleware for Energy Policy Management,* ou *middleware* para gerenciamento de políticas de energia), um *middleware* para RSSF capaz de aplicar políticas de consumo de energia aos nós pertencentes a rede de forma a controlar seus respectivos ciclos de trabalho, para otimizar o consumo de energia e evitar desperdícios, viabilizando a aplicação de RSSF em domínios como a aquicultura.

1.2.2 Objetivos específicos

Os objetivos específicos desse trabalho serão listados a seguir, onde cada um representa um módulo do MEPIM:

1) Desenvolvimento do módulo para publicação de políticas de consumo de energia e implantação das políticas;
2) Desenvolvimento do módulo responsável pelo envio de dados coletados por sensores (pH, luminosidade, temperatura do ambiente, nível de bateria, etc.) para o *Middleware* para posterior armazenamento dos danos em uma base de dados;
3) Implementação do módulo para comunicação com os nós da rede e banco de dados;
4) Implementação da comunicação com um banco de dados de acesso em tempo real para armazenamento de dados dos dispositivos da rede e políticas de consumo de energia.

2. CONCEITOS BÁSICOS

Neste Capítulo são abordados os conceitos básicos para compreensão do presente trabalho. Nas seções a seguir serão abordados assuntos como arquitetura orientada a serviços (Seção 2.1), padrão arquitetural REST (Seção 2.3), paradigma de comunicação Publish/Subscribe (Seção 2.4), definição de plataformas de *middleware* (Seção 2.5), explicação de algumas funcionalidades do servidor CloudMQTT (Seção 2.6), conceitos de *middleware* (Seção 2.6), conceitos sobre bancos de dados de tempo real (Seção 2.7), componentes do módulo Wi-Fi ESP8266 NodeMCU juntamente com definições dos modos de suspenção e conceitos sobre microalgas (Seção 2.9).

2.1 ARQUITETURA ORIENTADA A SERVIÇOS - SOA

A orientação a serviços permite o uso da Arquitetura Orientada Serviços ou (SOA, do inglês *Service Oriented Architecture*) de forma que seja possível a comunicação entre aplicações independentemente da plataforma e linguagem de programação utilizada (Cunha, 2006). Seu objetivo é desenvolver aplicações distribuídas interoperáveis e com capacidade de evolução.

No contexto de orientação a serviços, o paradigma SOA define que a construção de sistemas deve ser organizada em um conjunto de módulos, ou aplicações pequenas e simples. Esses módulos ou partes de software são conhecidos como serviços. Os serviços podem ser desenvolvidos independente de linguagem de programação e possibilita ser usado como cliente e servidor por meio de sua interface padrão, ou seja, sua interface permite que requisite um serviço e ainda possa ser solicitado por outro serviço (CUNHA, 2006).

Essa abordagem permite lidar com a heterogeneidade entre sistemas. Em outras palavras, fornece uma estratégia que possibilita a comunicação entre aplicações executadas em diferentes plataformas e em diferentes tecnologias. O SOA promove a interoperabilidade entre serviços por meio da utilização de tecnologias e protocolos de comunicação padronizados como o REST (FIELDING, 2000) e SOAP (CHAPPELL e JEWELL, 2002).

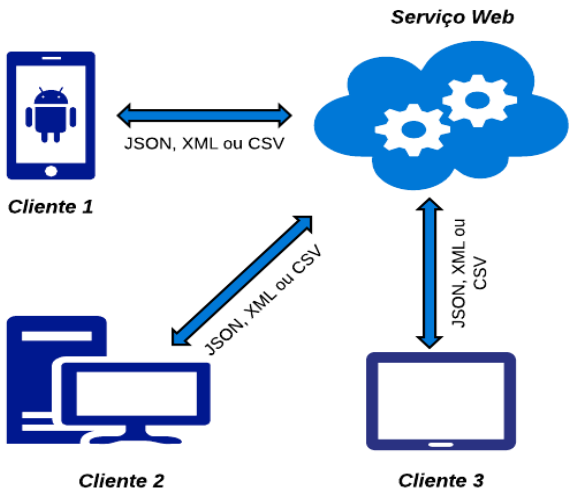
Assim, sistemas baseados em SOA não dependem de linguagens de programação, plataformas, nem do contexto ao qual serão usados, tornando os provedores de serviços independente dos seus consumidores.

2.2 SERVIÇO WEB

Um serviço web (do inglês *Web Service*) trata-se de um padrão de comunicação, que atende as necessidades de uma SOA, cujo principal objetivo é possibilitar a interoperabilidade entre diferentes *softwares* e aplicações, executando em uma vasta gama de plataformas. As mensagens compartilhadas entre serviços web são escritas utilizando tecnologias padronizadas como JSON, XML ou outros padrões relacionados a *Web* (BOOTH et al, 2004). A identificação de um serviço web se dá por meio do uso de URIs (*Uniform Resource Locator*), permitindo a interação entre diferentes máquinas em uma rede.

O objetivo de um serviço web é prover funcionalidades e recursos que possam ser acessadas através de rede, por um cliente solicitante que deseja utilizar essas funcionalidades. A comunicação entre um cliente e o serviço é baseada em requisição e resposta, onde o cliente realiza uma requisição a um provedor de serviço e o servidor retorna uma resposta ao cliente em um formato como JSON ou XML. O diagrama abaixo (Figura 1) ilustra como ocorre a iteração entre o serviço web e as aplicações clientes.

Figura 1- Comunicação Entre o Serviço Web e Solicitantes.



FONTE: Elaborada pelo autor.

2.3 REST

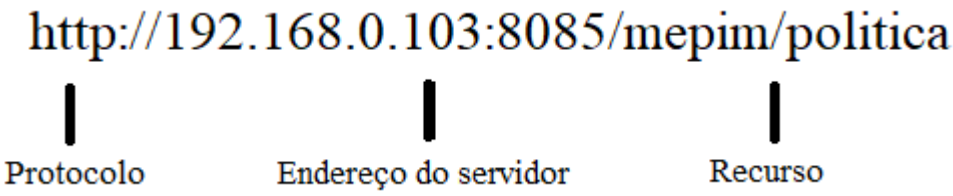
O REST (abreviação do inglês *Representational State Transfer*) é um estilo arquitetural de software para implementação de serviços web proposto por Fielding em (FIELDING, 2000),

um dos criadores do protocolo HTTP. A abordagem do REST é baseada na utilização de requisições HTTP (*HyperText Transfer Protocol*) e no conceito de recursos. Com isso, alguns princípios são considerados para viabilização de construções de serviços em conformidade com o REST, como:

- Utilização de métodos HTTP de forma explícita;
- A identificação de recursos através de URIs;
- A transferência padronizada de respostas em formatos como JSON ou XML;
- A interligações entre diferentes recursos.

Em REST, os recursos são funcionalidades e dados que são trafegados pelo protocolo. Estes recursos são identificados através de URIs, normalmente acessados por meio de um endereço (link). Uma URI é um dos padrões utilizados por aplicações web para identificação de um recurso do servidor.

Figura 2 - Exemplo de URL.



FONTE: Elaborada pelo autor.

Uma URL trata-se de um formado utilizado para localização de recursos ou serviços disponibilizados na *internet*. Na Figura 2, é possível observar que uma URL é composta por várias partes, onde cada uma possui um significado particular. A primeira parte “http://”, especifica o protocolo de comunicação utilizado pelo cliente e o provedor para acessar um recurso ou serviço. A segunda parte diz respeito ao nome ou endereço do provedor de recursos, seguida de uma barra “/”, indicando o nome do recurso a ser acessado.

Os serviços web RESTful são serviços que aplicam os princípios de REST, os recursos são expostos por meio de URIs e acessados através do uso de métodos do protocolo HTTP. Os principais métodos HTTP são GET, PUT, POST e DELETE, onde cada método é um verbo que

pode ser usado para realizar uma alteração no recurso requisitado. A Tabela 1 mostra os principais verbos do protocolo HTTP seguida de suas respectivas descrições.

Tabela 1 - Métodos do protocolo HTTP.

Verbo	Descrição
GET	Realiza uma requisição de dados do provedor.
POST	Realiza a inserção de dados no provedor.
PUT	Usado para atualizar algum dado no provedor.
DELETE	Remove um determinado dado do provedor.

FONTE: Elaborada pelo autor.

2.4 PUBLISH/SUBSCRIBE

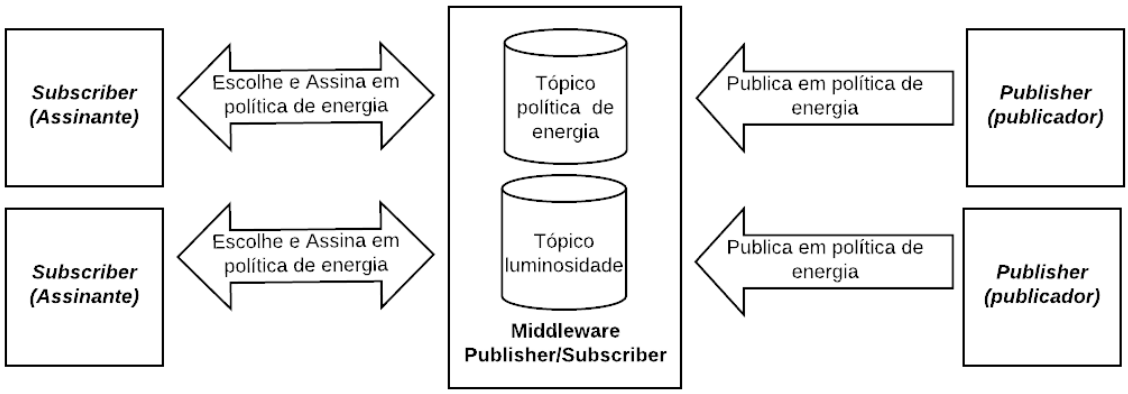
"O *Publish/Subscribe* é um paradigma de comunicação com desacoplamento entre produtores e assinantes em termos de tempo, espaço e sincronização", (BRASIL, 2017, p. 17). Nesse paradigma, os produtores de conteúdo, ou *Publishers*, produzem mensagens em forma de eventos para que os assinantes, ou *Subscribers*, possam obtê-los. Os assinantes podem declarar interesse para receber informações (através de tópicos) de outros publicadores. As inscrições são uma forma de filtrar os eventos produzidos pelos produtores de conteúdo, ou seja, garante que os assinantes receberam apenas os dados de interesse (CORSARO et al, 2006). Uma requisição é feita a um outro elemento da rede sempre que um *Subscriber* se inscreve para obter um dado, o mesmo ocorre quando um Publisher publica um dado. As requisições são enviadas a um elemento chamado *broker*, responsável por intermediar a comunicação entre os produtores de conteúdo e os assinantes.

As publicações realizadas pelos *Publishers* são disponibilizadas de maneira assíncrona, ou seja, emissores e os receptores não precisam estar disponíveis no mesmo momento para que haja comunicação. Já nas mensagens síncronas a comunicação ocorre de maneira direta, ou seja, o emissor e o receptor devem estar disponíveis no mesmo momento. O broker possui um serviço de notificações que tem a funcionalidade de desacoplar os *Subscribers* dos *Publishers*. Para que isso ocorra, as notificações recebidas do produtor são encaminhadas, pelo serviço de notificações, para todos os *Subscribers* interessados em receber essas informações (ZIMMERMANN, 2006).

Outro ponto importante a respeito do *Publish/Subscribe* é o fato de permitir o desacoplamento temporal e espacial. No desacoplamento temporal o *Publisher* e o *Subscriber* têm tempos de vida independentes, não havendo necessidade estarem conectados ao mesmo tempo para que ocorra o encaminhamento de dados. O desacoplamento espacial sugere que os *Publishers* não necessitam conhecer os *Subscribers*, possibilitando que os *Subscribers* possam ser distribuídos, duplicados e modificados.

Para melhor entendimento do funcionamento do paradigma *Publish/Subscribe*, a Figura 3 ilustra seu funcionamento básico, onde os publicadores divulgam uma informação em um tópico e os assinantes se inscrevem em um tópico de interesse para obtenção dessa informação e o broker intermediando a comunicação entre esses componentes.

Figura 3 - Comunicação Publish/Subscribe.



FONTE: Elaborada pelo autor.

Um exemplo de implementação do Publish/Subscribe é o CloudMQTT (“CloudMQTT”, 2018). No CloudMQTT estão implementados todos protocolos de comunicação necessários para que a comunicação entre o publicador e o assinante ocorra de maneira segura e com garantia de que os dados serão encaminhados corretamente aos seus destinos.

2.5 *MIDDLEWARE*

O *middleware* (em português, mediador), é um software que atua como mediador entre outros sistemas e aplicações. Esse software possui a capacidade prover abstrações como invocação remota de métodos, notificação de eventos, transmissão e replicação de dados compartilhados, fazendo com que o nível da comunicação entre outros sistemas seja acrescido.

Em outras palavras, um *middleware* se refere uma camada de software que fornece abstração de programação, assim como o mascaramento de tipos de redes, linguagens de programação e sistemas operacionais (COULOURIS, 2013). Além disso, no contexto da comunicação cliente-servidor, onde a escalabilidade é limitada, ou seja, o crescimento de requisições feitas pode comprometer a performance por limitações de *hardware*, o uso de *middleware* pode adicionar atributos de qualidade para que seja possível escalar a quantidade de dispositivos na rede sem que ocorra perda de pacotes.

Um *middleware* pode abranger várias categorias diferentes, onde cada um é baseado em um paradigma de comunicação. Os principais modelos de arquitetura são serviço web, sistemas *publish/subcribe,* filas de mensagens, objetos distribuídos e componentes distribuídos (COULOURIS, 2013).

De acordo com a RedHat (“RedHat”, 2018), alguns exemplos dos vários domínios de *middleware* existentes que abrangem uma gama de software são:

- Chamada de procedimento remoto (do inglês *Remote procedure call* - RPC): trata-se de uma comunicação cliente-servidor, permitindo que aplicações e serviços sejam distribuídos entre outras plataformas.

- Interfaces de programação e aplicações (do inglês *Application Programming Interface - API*): trata-se de um conjunto de rotinas e padrões de programação para a criação de softwares. Permitindo que serviços e aplicações interajam entre si sem a necessidade de saber detalhes de suas implementações;

- *Middleware* orientado a mensagens (MOM): baseado em RPC, mas com a adição de mecanismos de filas, possibilitando que a interação cliente-servidor ocorra de forma assíncrona quando um dispositivo destinatário não estiver disponível.

- Broker de solicitação de objeto (do inglês *Object Request Broker* - ORB): utiliza comunicação cliente-servidor para que serviços remotos possam ser acessados como serviços locais. Os processos do servidor são registrados com o ORB, o qual os clientes fazem contato para acessar esses serviços.

2.5.1 CloudMQTT

O MQTT (da sigla em inglês *Message Queuing Telemetry Transport*) é um protocolo de mensagens projetado para comunicação *Machine-to-Machine* (M2M) baseado em *Publish/Subscribe*, onde a comunicação entre os clientes e servidor (*broker*) é realizada via protocolo TCP (*Transmission Control Protocol*).

O CloudMQTT é uma implementação de um *broker* disponibilizado na web, que utiliza o protocolo MQTT para fornecer métodos de troca de mensagens para dispositivos de baixo poder computacional. Este software possibilita interoperabilidade entre dispositivos e aplicações independentemente da plataforma utilizada, além de suportar todos os níveis de QoS (Quality of Service) e fornece métodos de autenticações para os dispositivos da rede (“CloudMQTT”, 2018).

Em uma rede MQTT o broker desempenha o papel de intermediador na comunicação entre os elementos. Todos as publicações de dados dos elementos da rede são feitas por meio do broker que fará o encaminhamento de seus dados.

Outro ponto importante do CloudMQTT é o fato de implementar filas para garantir que as mensagens publicadas possam ser encaminhadas posteriormente para os nós inscritos em um tópico sem a necessidade de estarem conectados no momento da publicação do dado.

2.6 FIREBASE

Uma das tendências tecnológicas usadas como propostas em problemas com grandes quantidades de dados é o NoSQL (do inglês *Not Only SQL*). O NoSQL são bancos de dados não relacionais comumente usados para o armazenamento e processamento de grandes quantidades de dados por serem otimizados para uma performance escalável.

Um exemplo de ferramenta que implementa o NOSQL é o Firebase. O Firebase é uma plataforma da empresa Google para armazenamento de dados de aplicações. Esta ferramenta fornece uma gama de recursos que agilizam o desenvolvimento de aplicações sem a necessidade do gerenciamento de infraestrutura. Provê funcionalidades como análises, bancos de dados, mensagens e relatórios de erro e ainda possui escalonamento automático (“Firebase”, 2018).

O Firebase possui uma ferramenta chamada *Firebase RealTime Database* que, trata-se de um banco de dados para armazenamento e sincronização de dados na nuvem NoSQL em

formato JSON, que é um formato leve para troca de dados computacionais, e sincronizados em tempo real, o que favorece sua utilização em aplicações que exigem respostas rápidas.

2.7 MÓDULO WIFI ESP8266 NODEMCU

O NodeMCU, módulo Wi-Fi ESP8266 (Figura 3), é uma placa para desenvolvimento projetada para aplicações móveis com restrições de espaço e potência (“Espressif”, 2018). O NodeMCU ESP8266 possui um processador ESP8266-12E com arquitetura de 32 bits, podendo operar na faixa de 80 a 160 MHz, com memória FLASH de 4MB, 64 Kb para instruções e 69Kb para dados. Possui ainda módulo para comunicação com rede Wi-Fi. Suporta redes (802.11 b/g/n) que são padrões de conectividade sem fio mais utilizados (MAGNO, 2013). Possui um conector micro USB que pode ser alimentado a 5VDC, com tensão de operação de 4.5 à 9V, possui 11 pinos digitais, sendo que 1 pino é analógico com resolução de 10 bits. O ESP8266 pode ser programado usando a linguagem de programação LUA (“LUA”, 2018) e é compatível com a IDE Arduino (“Arduino”, 2018), baseada em C++.

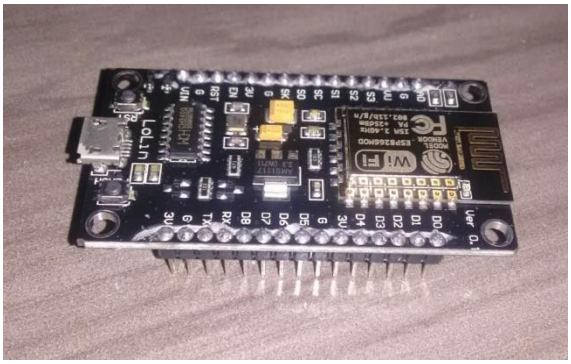
O ESP8266 possui três modos configuráveis de suspensão que modificam o *duty cycle*, são eles o *modem-sleep, light-sleep e deep-sleep*. Todos têm funções diferentes que podem ser modificadas conforme necessário:

- **Modem-sleep:** É o modo padrão para o ESP8266. Porém, só é ativado quando o mesmo estiver conectado a um ponto de acesso. Nesse modo, o modem wifi é desativado sempre que possível entre intervalos de mapa de identificação de tráfego (ou *Delivery Traffic Indication Map* - DTIM) definidos pelo roteador.

- **Light-sleep:** Esse modo é semelhante ao Modem-sleep, no entanto desativa também o clock do sistema e suspende a unidade de processamento (CPU). A CPU trabalha em um ritmo mais econômico, ou marcha lenta, sem ser desligada.

- **Deep-sleep:** Tudo é desligado nesse modo, com exceção do relógio de tempo real (RTC), responsável por manter o tempo do dispositivo permitindo que o mesmo acorde após um período.

Figura 4 - Módulo NodeMCU ESP8266.



FONTE: Própria (2018).

3. ARQUITETURA DO SISTEMA

De acordo com o que foi apresentado no Capítulo 1, o presente trabalho tem como objetivo o desenvolvimento de um *middleware* capaz de aplicar políticas de consumo energético aos nós da rede, alterando seus modos de suspensão, de forma que reduzam a frequência de trabalho através da desativação de componentes físicos como a CPU e o módulo Wi-Fi para reduzir o consumo de energia dos nós.

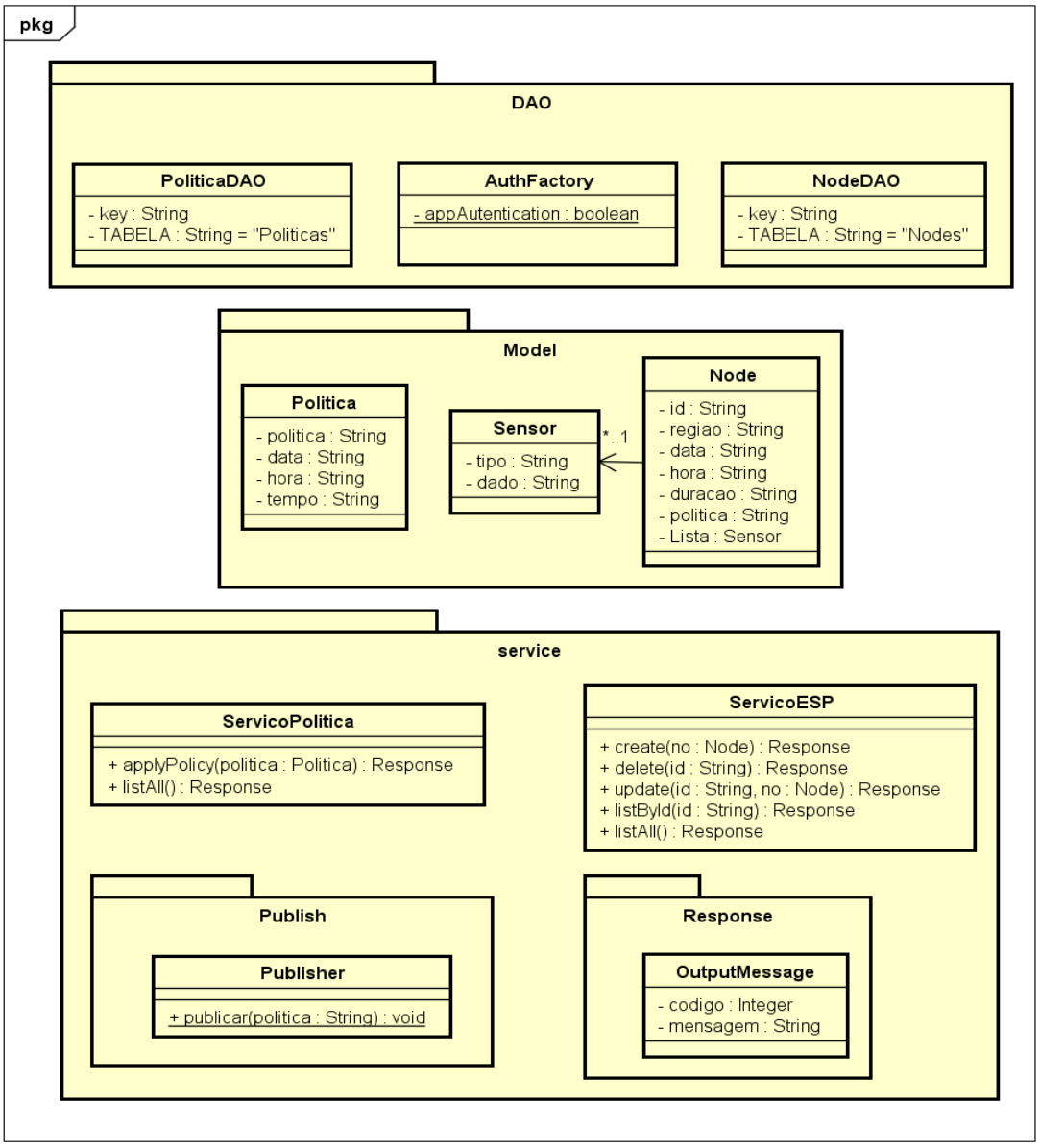
Esse capítulo está organizado da seguinte forma: na seção 3.1 é apresentada a organização do MEPIM em termos de classes do sistema e na seção 3.2 é apresenta a arquitetura da aplicação desenvolvida para alcançar o objetivo citado.

3.1 ORGANIZAÇÃO DO MEPIM

Essa seção apresenta um diagrama de classes UML (Linguagem de Modelagem Unificada, do inglês *Unified Modeling Language*) onde cada elemento nesse diagrama representa uma entidade do sistema, cada qual com sua finalidade específica na arquitetura como mensagens, serviços, e suas relações. O uso de um diagrama de classes é fundamental para auxiliar o desenvolvimento de aplicações orientadas a objetos (LOPES, 2011). A Figura 6 a seguir mostra o diagrama de classes do sistema MEPIM onde é possível notar o mapeamento de toda a estrutura das classes e seus relacionamentos.

A classes “PoliticaDAO” e “NodeDAO” implementam o padrão de projeto *Data Access Object* (DAO) (GAMMA, 2000) para prover métodos para a comunicação com a base de dados do *Firebase.* A classe “AuthFactory” fornece os métodos de autenticação necessários para que haja a comunicação com o *Firebase*. A classe “Politica” representa uma política de energia que pode ser implantada nos nós da rede. As classes “Node” e “Sensor” representam os nós da rede juntamente com os dados de seus sensores, assim, um Node pode possuir N sensores conectados a ele. Enquanto “ServicoPolitica” e ServicoESP representam serviços onde cada método corresponde a uma ação baseada nos verbos HTTP. “Publish” é a classe responsável por fornecer os métodos necessários para a publicação de um dado em um tópico e “OutputMessage” tem o objetivo de retornar mensagens às aplicações clientes.

Figura 5 - Diagrama de Classes do MEPIM.



FONTE: Elaborada pelo autor.

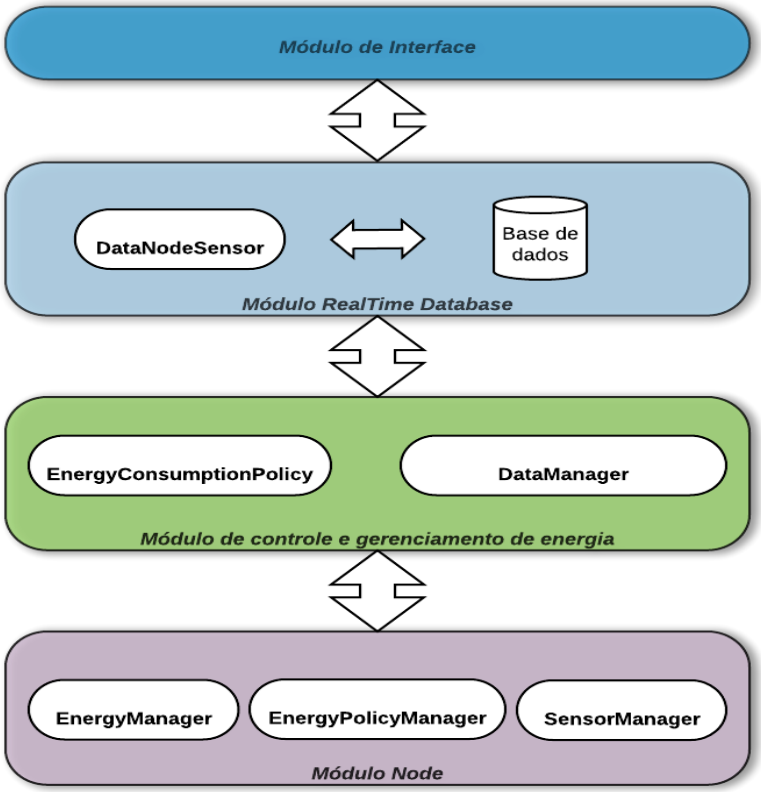
3.2 ARQUITETURA EM CAMADAS

A arquitetura proposta é composta por três camadas, sendo elas: Real Time Database, Controle e Gerenciamento de Energia e Node como ilustrado na Figura 4.

Para armazenar os dados enviados pelos nós da rede de objetos, neste trabalho foi utilizado o banco de dados de acesso em tempo real da plataforma Firebase (“Firebase”, 2018). Os dados armazenados são: (i) dados de energia residual das baterias dos nós; (ii) dados dos sensores conectados aos nós e (iii) dados das políticas de consumo de energia aplicadas aos nós da rede.

O módulo de Controle e Gerenciamento de Energia é responsável por intermediar a comunicação entre o banco de dados e os nós da rede. Essa comunicação possibilita a aplicação de novas políticas de consumo aos nós e o armazenamento dos dados coletados pelos sensores. O módulo de interface permite a disponibilização de uma API para consulta aos dados dos sensores e implantação de políticas de energia. A imagem abaixo ilustra as camadas do MEPIM.

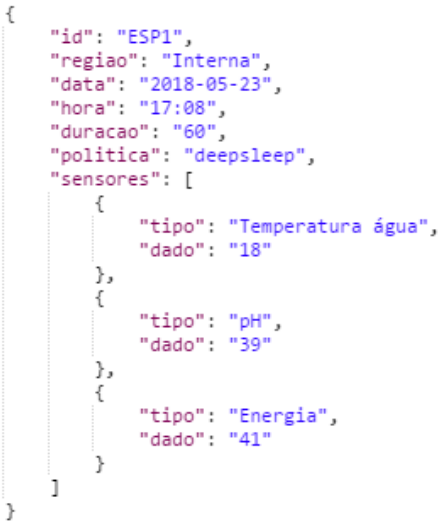
Figura 6 - Arquitetura do MEPIM.



FONTE: Elaborada pelo autor.

O ***Módulo de Interface*** fornece uma API REST que permite aos demais módulos a interação entre si, podendo acessar serviços e funcionalidades do MEPIM. Em outras palavras, esse módulo tem a principal finalidade de prover interoperabilidade, funcionando como uma ponte para comunicação entre os demais módulos que compõem o sistema, tornando o processo de comunicação abstrato e fornecendo meios de acesso padronizados aos recursos através do protocolo HTTP. O ***Módulo de Interface*** utiliza o formato JSON como resposta às requisições feitas aos serviços como mostra a Figura 8. Esses serviços são providos pelo ***Módulo de controle e gerenciamento de energia,*** que possibilita as requisições aos dados dos sensores e envio de tais dados para o ***Módulo RealTime Database***.

Figura 7 - Estrutura dos Dados de Um Nó.



```
{
    "id": "ESP1",
    "regiao": "Interna",
    "data": "2018-05-23",
    "hora": "17:08",
    "duracao": "60",
    "politica": "deepsleep",
    "sensores": [
        {
            "tipo": "Temperatura água",
            "dado": "18"
        },
        {
            "tipo": "pH",
            "dado": "39"
        },
        {
            "tipo": "Energia",
            "dado": "41"
        }
    ]
}
```


FONTE: Elaborada pelo autor.

O ***Módulo RealTime Database*** é responsável por armazenar os dados de sensores e políticas de energia aplicadas providos pelos nós da rede. Esses dados são recebidos pela ***Modulo de controle e gerenciamento de energia*** e encaminhados para armazenamento pela ***Módulo RealTime Database*** em uma base de dados de tempo real, utiliza o formado JSON para persistir e suportar o armazenamento de grandes quantidades de informações sem que isso degrade seu desempenho.

A função do ***Módulo de controle e gerenciamento de energia*** é permitir a aplicação de políticas de energia e mediar a comunicação entre a ***Módulo RealTime Database*** e o ***Módulo Node***. O *middleware* publica (Publish) as políticas de energia em um tópico para que os nós assinantes os obtenham. Os dados providos pelos nós da rede são recebidos e posteriormente enviados para a ***Módulo RealTime Database*** para que sejam armazenados no banco de dados.

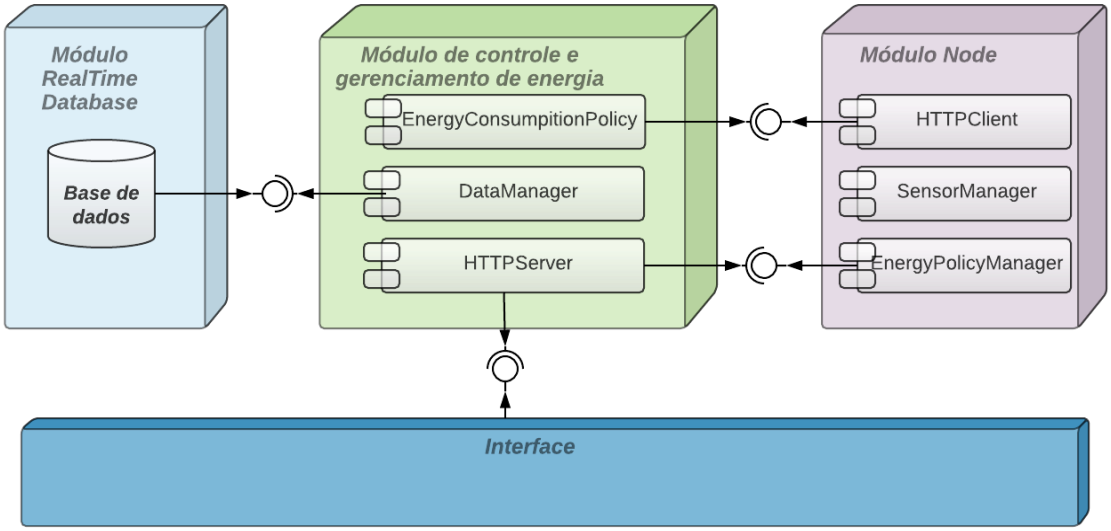
O ***Módulo Node*** tem como objetivo controlar os estados dos nós, alterando seus modos de suspensão, e executar ações por meio de seus atuadores. Além disso, esse módulo é responsável por captar e enviar os dados dos sensores e dado de sua bateria para a camada ***Módulo de controle e gerenciamento de energia*** e acordo com a política aplicada e ainda assinar (Subscribe) o tópico para obtenção das políticas de energia publicadas também pelo ***Módulo de controle e gerenciamento de energia***. Os nós da rede assinam o tópico onde são publicadas as políticas de energia e passam a ouvir a rede. Quando uma política é publicada, o MQTTRoute encaminha o dado até os nós. Ao detectar a publicação de algum dado, os nós recebem a política de energia e, conforme especificado na política recebida (modem-sleep,

light-sleep e deepsleep), altera o modo de suspensão do hardware fazendo com que os nós diminuam o ritmo de trabalho ou desabilitem alguns componentes físicos.

3.3 IMPLANTAÇÃO

Esta seção aborda o processo de implantação de cada camada do MEPIM apresentada na seção anterior.

Figura 8 - Diagrama de Componentes do MEPIM.

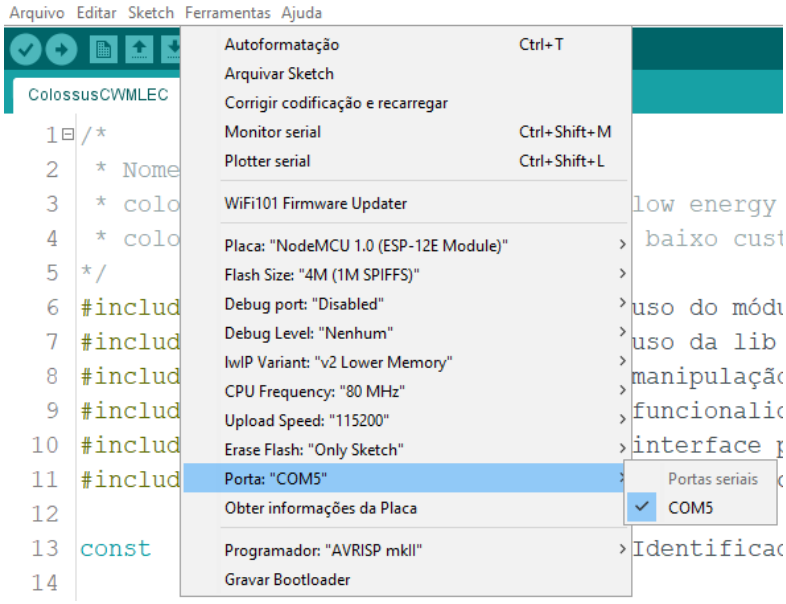


FONTE: Elaborada pelo autor.

A implantação do ***Módulo de aquisição e controle*** dá-se através da criação de uma aplicação web utilizando a linguagem de programação Java (“JAVA”, 2018) onde são definidos os recursos e serviços a serem disponibilizados. O acesso aos recursos através da Internet ou rede local é feito através da implantação de tal aplicação web em um servidor Tomcat (“Tomcat”, 2018), com isso é possível atender as requisições de outras aplicações através de uma rede.

O processo de implantação dos nós da rede foi realizado através da IDE Arduino onde é realizada a definição da placa de desenvolvimento *“NodeMCU 1.0 ESP-12E Module”* como mostra a Figura 10. Com isso, os nós podem ser configurados e implantados na rede para interação com os demais módulos do MEPIM.

Figura 9 - Definição da Placa de Desenvolvimento.



FONTE: Elaborada pelo autor.

A implantação da base de dados é feita através da plataforma do *Firebase*, sendo necessário apenas a definição do nome do projeto (nome do banco de dados) e a definição das regras de acesso aos dados. Essas regras restringem os dados salvos na base de dados, fazendo com que não possam ser acessados por qualquer pessoa ou aplicação. Seu objetivo é proteger os dados do banco contra uso indevido. A Figura 11 mostra um exemplo de regra de autenticação que concede acesso de gravação e leitura de dados à usuários autenticados, onde o uid (linhas 19 e 20) é o identificador do usuário do Firebase e o campo “Nodes” é o local onde os dados estão armazenados.

Figura 10 - Exemplo de Regras de Acesso.

```
18    }, "Nodes" : {
19      ".read": $uid === auth.uid",
20      ".write": "$uid === auth.uid",
```

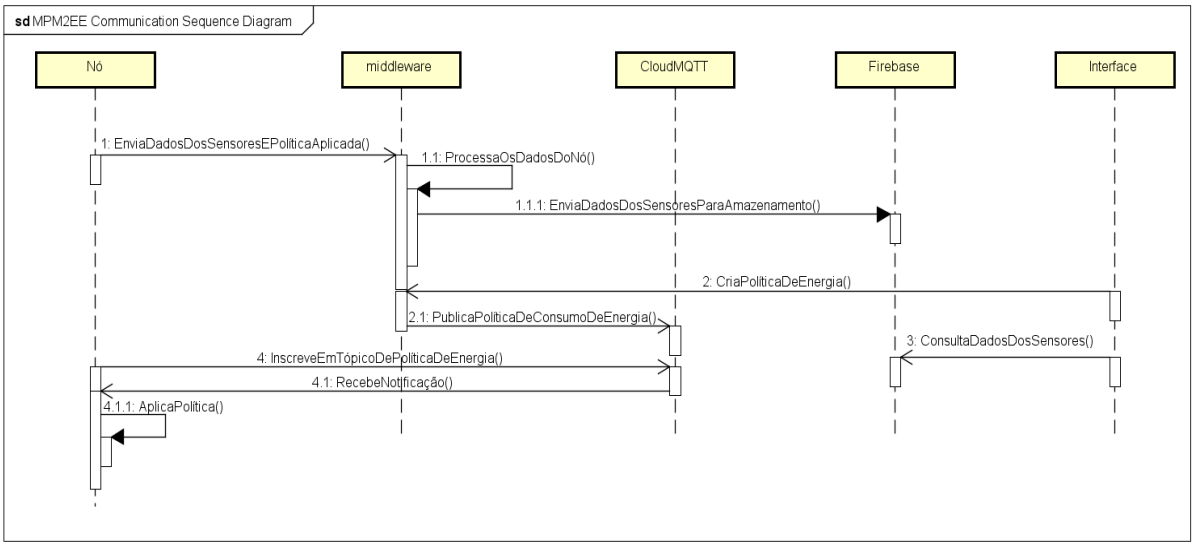
FONTE: Elaborada pelo autor.

3.4 IMPLEMENTAÇÃO

Esta seção aborda os principais aspectos da implementação de cada camada do MEPIM apresentadas na seção 3.3. Os processos estão divididos em: (i) coleta e envio de dados dos nós; (ii) armazenamento de dados dos nós no Firebase; (iii) implementação da criação, publicação e aplicação das políticas de consumo de energia e (iv) assinatura em tópicos da rede.

O diagrama de sequência ilustrado na Figura 12, mostra de forma simplificada como ocorre a comunicação entre as camadas do MEPIM. Os nós enviam os dados dos sensores através do protocolo HTTP para o MEPIM que se encarregue de encaminhar os dados providos pelos nós para armazenamento no Firebase. Os nós assinam o tópico de publicação das políticas de consumo de energia. Ao ser notificado pelo CloudMQTT sobre a publicação de uma política, o nó recebe esse dado e altera o *duty cycle* de acordo com a política recebida.

Figura 11 - Sequência da Comunicação Entre as Camadas.



FONTE: Elaborada pelo autor.

3.4.1 Envio dos dados dos sensores e política aplicada

Para envio dos dados dos sensores ao *middleware*, é necessário o uso da biblioteca HTTPClient IDE Arduino. Essa biblioteca fornece uma API, que abstrai detalhes complexos de sua implementação, simplificando o processo de comunicação do ESP8266 via protocolo HTTP com um provedor de serviços (*middleware*). Os métodos providos pela biblioteca são o conjunto de verbos HTTP, a saber: GET, POST, PUT e DELETE.

Figura 12 - Coleta e Envio dos Dados dos Nós.

```
130     HTTPClient http;
131     http.begin("http://"+IPSERVER+":8085/mepim/esp");
132     http.addHeader("Content-Type", "application/json");
133     http.addHeader("Accept", "application/json");
134     http.POST(getSensores ());
135     http.end();
```

FONTE: Elaborada pelo autor.

A Figura 13 demostra a implementação, na IDE Arduino, da função responsável por enviar os dados dos sensores para o *middleware*. Nas linhas 130 e 131 é criado um objeto http e em seguida é especificado a URL para acesso ao recurso *create* do *middleware*. Em 132 e 133 é definido o tipo de conteúdo que será enviado e tipo de dado que será aceito no retorno da URI, sendo esse tipo um JSON para ambos. Em seguida, o método POST é invocado para enviar os dados dos sensores ao *middleware* na linha 134 e por fim, na linha 135 a comunicação HTTP é encerrada.

Os dados enviados pelos nós contém campos que vão desde a identificação do nó que envia o dado aos tipos de sensores conectados, a saber: id, regiao, data, hora, duraçao, politica, além de uma lista contendo os tipos de sensores e dados coletados como ilustra a imagem abaixo Figura 14.

Figura 13 - Detalhes dos Dados Enviados pelos Nós.

```
{
    "id":"ESP1",
    "regiao":"Interna",
    "data":"-",
    "hora":"-",
    "duracao":3,
    "politica":"deepsleep",
    "sensores":[
        {
            "tipo":"PH",
            "dado":"98"
        },
        {
            "tipo":"temp_agua",
            "dado":"48"
        },
        {
            "tipo":"energia",
            "dado":"48"
        }
    ]
}
```

FONTE: Elaborada pelo autor.

O campo *id* na linha 1, é o identificador do nó, onde cada nó da rede possui um *id* único. O campo *região* linha 2, indica a região a qual os sensores estão coletando informações, ou

seja, se é interna ao tanque ou externa. Cada região contem tipos de sensores diferentes, a externa pode conter sensores por exemplo de temperatura do ambiente, luminosidade do ambiente e a externa pode conter sensores como temperatura da água e Ph, porém independente da região, cada nó possui um sensor em comum que é o de nível de energia (linha 18 e 19), usado para medir a energia restante da bateria o alimenta. Os demais campos são a *data* (linha 4), que representa a data da coleta dos dados, o campo *hora* (linha 5), que é o momento em que a coleta foi feita, e o campo *politica* (linha 7), que indica qual política de energia está instalada e o campo *duração* (linha 3) é o tempo em que um nó permanece em estado de sono profundo (*deepsleep*).

3.4.2 Envio dos dados para armazenamento

Para que o *middleware* possa atender às requisições dos nós da rede, obtendo os dados dos sensores para posteriormente enviá-los ao Firebase, foi implementado um método POST do protocolo HTTP que recebe um arquivo JSON contendo os dados enviados pelos nós. Nessa seção, são abordados detalhes do processo de obtenção e armazenamento dos dados dos sensores no Firebase.

A partir do envio dos dados dos nós é gerada uma requisição ao método POST do *middleware*. Essa requisição contém o arquivo JSON com os dados dos nós que são e enviados pelo *middleware* via HTTP para que sejam armazenados na plataforma *Firebase*.

Figura 14 - Envio dos Dados para a Plataforma Firebase.

```
48      @POST
49      @Consumes(MediaType.APPLICATION_JSON)
50      @Produces(MediaType.APPLICATION_JSON)
51      public Response create(Node no) {
52          try {
53              NodeDAO dao = new NodeDAO();
54              dao.salvar(no);
55              dao.gerarToken();
56          } catch (Exception e) {
57              return Response.status(Response.Status.INTERNAL_SERVER_ERROR)
58                      .entity(new OutputMessage(500, e.getMessage()))
59                      .build();
60
61          }
62          return Response.status(Response.Status.CREATED)
63                  .entity(new OutputMessage(200, "Processo realizado com sucesso!"))
64                  .build();
65      }
```

FONTE: Elaborada pelo autor.

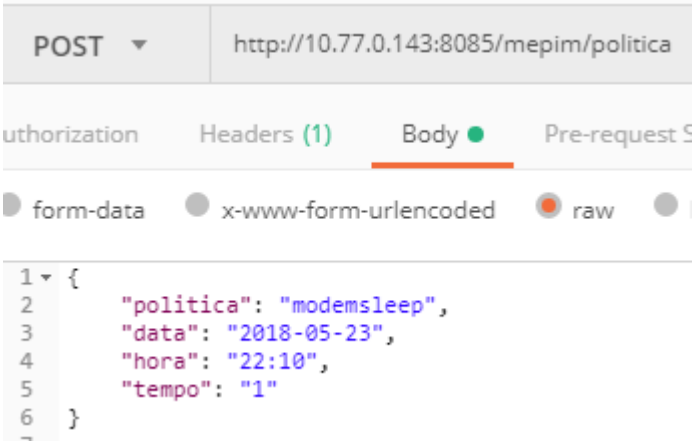
A Figura 15 demonstra a implementação do método POST do MEPIM, responsável por atender às requisições dos nós da rede e invocar o método para envio dos dados recebidos para o *Firebase.* Como exibido nos códigos da imagem acima, a linha de 48 é definido, para acesso ao serviço *create*(), a notação que corresponde ao método HTTP POST. Em seguida é especificado o tipo de consumo de mídia, ou seja, o formato de dado que o recurso irá consumir que é um JSON. Na linha 50, é usada a notação @Produces especificando o formato de retorno da URI solicitada definida como JSON. Na linha 51 é definido que o serviço *create*() receberá, em uma requisição POST, um dado "no" que conterá os dados dos sensores. Das linhas 53 à 55 é realizado a tentativa de encaminhamento dos dados recebidos para o *Firebase.* Caso ocorra algum erro que impeça o envio dos dados, o fluxo de código parte para a linha 57, onde é criada uma mensagem de retorno para o cliente que fez a requisição, contendo detalhes da falha. Caso contrário, esse fluxo parte para a linha 62, onde retornada uma resposta informando que o processo foi concluído com êxito.

3.4.3 Implantação das políticas de consumo de energia

A criação de uma política de consumo de energia é possível através de uma requisição, que pode ser feita por qualquer aplicação que seja capaz de usar o verbo POST do protocolo HTTP. A criação de uma política de consumo é feita por meio de uma URL, que é o endereço do serviço fornecido pelo MEPIM. Sendo necessário que os dados da política sejam escritos em formato JSON.

As políticas de consumo de energia aplicáveis são *ModemSleep*, *LightSleep* e *Deepsleep*, onde os detalhes dos efeitos de cada uma é mostrado na seção 2.8, baseadas nos modos de suspensão do *NodeMCU*. O processo de aplicação de uma política de consumo de energia modifica o *duty cycle* de todos os nós da RSSFs e os componentes físicos da placa podem ser desabilitados, de acordo com especificações da política implantado a fim de se prolongar a vida útil da rede. A Figura 16 demonstra a criação de uma política de consumo de energia.

Figura 15 - Exemplo de Política de Consumo de Energia.



FONTE: Elaborada pelo autor.

Para a criação de uma política é necessário informar os campos *politica* linha 2, que é o tipo da política de consumo, os campos *data* e *hora* linhas 3 e 4 respectivamente (preenchidos automaticamente pelo *middleware*), que representam o momento da sua criação e o campo *tempo* na linha 5 que indica a duração em minutos em que um nó permanecerá no estado de suspensão *DeepSleep* quando esse estado for aplicado.

A partir da criação de uma política, uma requisição é feita ao *middleware* via verbo POST, onde o *middleware* irá receber seus dados no formato JSON e publicá-los em um tópico. A publicação das políticas de consumo é realizada por meio do uso da API Eclipse Paho. O Paho é um projeto de código aberto desenvolvido na linguagem JAVA, que utiliza protocolos de troca de mensagens MQTT e MQTT-SN, baseados em *Publishe/Subscribe*, destinado para aplicações M2M (*Machine-to-Machine*) e IoT (*Internet Of Things*) (“Eclipse”, 2018).

Figura 16 - - Implementação de Publicação de Política.

```
39          MqttClient cliente = new MqttClient(broker, clientId, persistence);
40          MqttConnectOptions connOpts = new MqttConnectOptions();
41          connOpts.setCleanSession(true);
42          connOpts.setUserName(usuario);
43          connOpts.setPassword(senha.toCharArray());
44
45          cliente.connect(connOpts);
46
47          MqttMessage message = new MqttMessage(politica.getBytes());
48          message.setQos(qos);
49          cliente.publish(topic, message);
50          cliente.disconnect();
```

FONTE: Elaborada pelo autor.

O trecho de código da Figura 17, mostra a implementação da publicação de uma política de consumo de energia via Paho, onde são necessários inserir os dados como o endereço do broker e identificação da aplicação Publisher (clientId) (linha 39) e dados de autenticação como usuário e senha (linhas 42 e 43), para que haja conexão com o broker CloudMQTT. Além disso, é necessário a definição do tópico e a mensagem (política de consumo) a ser encaminhada (linha 49) e o QoS (linha 48). O QoS (*Quality of Service*) indica como ocorrerá a relação entre o publicador e os assinantes podendo ser em três níveis diferentes. O QoS 0, indica que não há confirmação da entrega dos dados. No QoS 1 há confirmação de entrega de dados e o QoS 2, nível utilizado nessa implementação, ocorre confirmação de entrega e recebimento dos dados.

Após a publicação de uma política de consumo, o *broker* notifica os nós assinantes do tópico onde foram publicadas as políticas e as encaminha. Dessa forma, os nós assinantes do tópico "*mepim/Policy*", onde as políticas são publicadas, ao serem notificados sobre o surgimento de um novo dado, recebem (linha 208 e 209) a política e as implementa (linha 212) Figura 18.

Figura 17 - Exemplo de Aplicação de Política de Consumo.

```
206 void callBackMqtt(char* topic, byte* payload, unsigned int length) {
207
208     for (int i = 0; i < length; i++) {
209         char c = (char)payload[i];
210         policy += c;
211     }
212     applyPolicy(policy);
213 }
```

FONTE: Elaborada pelo autor.

3.4.4 Assinatura em tópicos de política de energia

A assinatura dos nós é feita utilizando o protocolo Publish/Subscribe para que seja possível obter as notificações de quando uma nova política de consumo de energia é publicada pelo *middleware*. A imagem abaixo mostra detalhes da implementação de um assinante.

Figura 18 - Implementação do Assinante.

```
66      client.setServer(IPBROKER, 1883);
67      client.setCallback(callBackMqtt);
68
69      int timeOut = 0;
70      while (!client.connected()) {
71            if (client.connect(IDNODE, USERMQTT, PASSWDMQTT))
72                client.subscribe("mepim/Policy");
73            else {
74                delay(1000);
75                timeOut++;
76                if (cont == 7)
77                    break;
78            }
79       }
```

FONTE: Elaborada pelo autor.

De acordo com a figura acima Figura 19, são especificados o endereço e a porta necessários para comunicação com o *broker* (linha 66) e a definição da função responsável por receber os dados no momento de sua detecção (linha 67). Em seguida, são informados os dados de autenticação do *broker* que são o identificador do assinante, usuário e senha (IDNODE, USERMQTT, PASSWDMQTT), necessários para que seja possível assinar um tópico (linha 71). Caso o dispositivo assinante consiga se conectar ao servidor, ele define o tópico de interesse "*mepim/Policy*" para receber os dados pulicados. Caso contrário, os códigos das linhas 71 à 78 serão repetidos até obter sucesso na conexão e assinatura ou o *timeOut* de 7 segundos se esgotar (linhas 74 à 77). Com isso, os nós serão capazes de assinar um tópico e receber uma notificação sempre que houver a publicação de uma nova política de energia a ser implantada.

4. AVALIAÇÃO

Esse Capítulo apresenta a avaliação do MEPIM realizada através da execução de um método baseada em objetivos, questões e métricas, com a finalidade de avaliar se os objetivos definidos na seção 1.2 do Capítulo 1 foram atingidos.

4.1 OBJETIVOS DA AVALIAÇÃO

Para definição dos objetivos desta avaliação foi utilizado o método GQM, ou *Goal/Question/Metric* proposto em (BASILI, 1992). O GQM trata-se de uma abordagem para medição de produtos e processos de software. Sua metodologia parte da condição de que as medições devem ser orientadas a metas, onde toda coleta de dados deve ser baseada em um fundamento lógico, que é documentado de maneira explícita.

No GQM, o primeiro passo para atingir os objetivos da avaliação é a definição das metas. Considerando os objetivos apresentados no Capítulo 1: *(i) desenvolvimento do módulo para publicação de políticas de consumo de energia e implantação das políticas; (ii) desenvolvimento do módulo responsável pelo envio de dados coletados por sensores (pH, luminosidade, temperatura do ambiente, nível de bateria, etc.) para o Middleware para posterior armazenamento dos danos em uma base de dados; (iii) implementação do módulo para comunicação com os nós da rede e banco de dados; e (iv) Implementação da comunicação com um banco de dados de acesso em tempo real para armazenamento de dados dos dispositivos da rede e políticas de consumo de energia*. Os objetivos do trabalho foram aprimorados para permitir o surgimento das metas (*Goals*) da avaliação, a saber:

Primeira Meta (G1): Avaliar a eficiência do MEPIM em relação a redução de consumo de energia dos nós da RSSF através do processo de aplicação de políticas de consumo de energia.

Segunda Meta (G2): Avaliar a eficiência do MEPIM em termos de escalabilidade.

Uma vez definidas as metas da avaliação, foram derivadas as questões (*Questions*) para especificar se a meta foi alcançada. A seção abaixo (4.2) mostradas as questões elaboradas.

4.2 QUESTÕES

As questões caracterizam uma definição operacional das metas da avaliação, identificando informações fundamentais para alcança-las. Seguindo a abordagem GQM, foram levantadas uma série de questões a partir de cada meta com a finalidade de determinar medições adequadas para a avaliação. Assim, as questões Q1 e Q2 refinam a meta G1, tendo como foco a redução do consumo de energia. As questões Q3, Q4 e Q5 estão relacionadas a meta G2,

tendo como foco avaliar a escalabilidade do MEPIM. A seguir são listadas as questões levantadas:

Q1: Qual a diferença de consumo entre as três políticas de consumo de energia do MEPIM?

Q2: O processo de aplicação das políticas de consumo de energia, por meio do MEPIM, é eficiente em termos de consumo de energia se comparado com o NodeMCU sem essa aplicação?

Q3: O uso do MEPIM causa perda de pacotes?

Q4: A inserção de novos nós na rede causa perda de pacotes?

Q5: A retirada de nós na rede causa perda de pacotes?

4.3 MÉTRICAS

A partir do refinamento das questões, foram elaboradas as métricas (*Metrics*) para definir os dados que necessitam ser coletados e responder as questões mencionadas na seção 4.2. As métricas definidas são classificadas por M_i, onde $_i$ corresponde ao índice da questão. A finalidade das métricas é quantificar a eficiência do MEPIM em termos de: (i) diferença de consumo entre as políticas de consumo de energia, (ii) eficiência elétrica com a aplicação das políticas de consumo de energia, (iii) eficiência em termos de entrega de pacotes, (iv) eficiência em termos de entrega de pacotes com a inserção de novos nós na rede e (v) a eficiência em relação a entrega de pacotes com a remoção de nós da rede.

A métrica **M_1** foi criada para medir o consumo entre as políticas de consumo de energia. Seu objetivo foi calcular a diferença de consumo de energia entre cada uma das políticas do MEPIM. Esta métrica coleta dados em *miliamperes* que fazem referência ao consumo das políticas de consumo de energia do MEPIM. A definição dessa métrica é dada a seguir:

$$|D| = P_i - P_n$$

Onde:

|D| Número da diferença de consumo;

P_i Número do consumo de uma política i;

P_n Número do consumo de uma política n.

A métrica **M_2** foi especificada para avaliar o termo eficiência elétrica. Seu objetivo é calcular quantas horas uma determinada bateria pode alimentar um nó com a aplicação de cada uma das políticas de consumo de energia implementadas pelo MEPIM. Esta métrica captura

dados que representam o valor em horas que uma bateria pode alimentar um nó até o esgotamento total de sua carga. Essa métrica é definida da seguinte forma:

$$H = \frac{CB}{C}$$

Onde:

H Número de horas que uma determinada bateria pode alimentar o nó;

CB Capacidade da bateria em *miliamperes/h*;

C Consumo de energia do nó em *miliamperes*.

A métrica **M3** foi elaborada para aferir a eficiência da entrega de dados do MEPIM. Seu objetivo é calcular possíveis perdas de pacotes na rede durante seu funcionamento. Seu objetivo é a captura dados referentes a quantidade de envios de dados pelos nós. Sua definição é dada da seguinte forma:

$$P = \left(\sum QPN\right) - \left(\left(\frac{D}{I}\right) \times QN\right)$$

Onde:

P Número da perda de pacotes;

QPN Somatório da quantidade de pacotes enviados pelos nós;

D Valor da duração em minutos do teste realizado;

I Intervalo de tempo para envio de dados;

QN Valor da quantidade de nós utilizados.

A métrica **M4** foi especificada para avaliar o impacto da inserção de nós no MEPIM. Seu objetivo é calcular o impacto da adição de novos nós na rede através perda de pacotes. Esta métrica, assim como a métrica **M3** é definida da seguinte forma:

$$P = \left(\sum QPN\right) - \left(\left(\frac{D}{I}\right) \times QN\right)$$

Onde:

P Número da perda de pacotes

QPN Somatório da quantidade de pacotes enviados pelos nós;

D Valor da duração em minutos do teste;

I Intervalo de tempo de envio de dados;

QN Valor da quantidade de nós utilizados.

A métrica **M5** foi criada com o objetivo de avaliar o impacto da remoção de nós da rede. Esta métrica, assim como as métricas **M3** e **M4** captura dados referentes a quantidade de capotes enviados pelos nós da rede a fim de verificar quantos pacotes foram perdidos, caso haja alguma perda. Assim, sua definição é dada da seguinte forma:

$$P = \left(\sum QPN\right) - \left(\left(\frac{D}{I}\right) \times QN\right)$$

Onde:

P Número da perda de pacotes

QPN Somatório da quantidade de pacotes enviados pelos nós

D Valor da duração em minutos do teste

I Intervalo de tempo de envio de dados

QN Valor da quantidade de nós utilizados

4.4 ANÁLISE DOS RESULTADOS

Esta seção aborda uma análise dos dados e resultados obtidos a partir das avaliações feitas com base nos objetivos, questões e métricas definidos nas seções anteriores. As métricas relacionadas aos objetivos G1 e G2 foram obtidas através da execução do MEPIM. Os dados de todas as métricas foram obtidos através de medições durante o funcionamento do MEPIM. Os resultados estão organizados de acordo com os índices das métricas.

Os dados de consumo dos nós foram obtidos através de medições realizadas com um amperímetro para cada uma das políticas de consumo aplicadas em um nó. O processo de medições foi repetido por 3 (três) vezes utilizando 3 (três) nós, com resistores de 10 quiloohm (kΩ) e sem a utilização de sensores conectados ao nó. A Tabela 2 mostra o resultado do consumo em miliamperes (mA) de um nó com a aplicação das políticas ModemSleep, LightSleep e DeepSleep.

Tabela 2 - Dados da Métrica M1 Com a Diferença de Consumo de energia entre Políticas.

Política	Consumo

ModemSleep	~ 74.8 mA
LightSleep	~ 74.5 mA
DeepSleep	~ 1.78 mA

A imagem abaixo (Figura 20) mostra o momento do processo de realização de uma das medições feitas para obtenção dos dados de consumo dos nós.

Figura 19 - Momento da Medição de Consumo de Um Nó.

FONTE: Próprio autor.

A métrica **M_1** foi calculada baseada no consumo de energia para cada uma das políticas de consumo implementadas pelo MEPIM. Como apresentado na Tabela 2, ao analisar os resultados obtidos com a aplicação das políticas ModemSleep e Lightsleep, em relação ao consumo, não possuem uma diferença significativa, porém o resultado obtido com a política DeepSleep mostra a diferença de consumo de aproximadamente 97,7% em comparação com as políticas ModemSleep e LightSleep. Tal resultado indica que, dependendo do cenário de uso e da necessidade da aplicação do MEPIM, a política DeepSleep possui melhor eficiência de consumo de energia.

A métrica **M2** assim como a **M1**, foi calculada com base no consumo de energia de um nó com a aplicação das políticas de consumo e sem essa aplicação a fim de, a partir dos dados obtidos, calcular quantas horas uma determinada bateria pode alimentar esse nó até o esgotamento total de sua carga. Para esse teste, considerando que a bateria utilizada possui 9V (*volts*) de 250 mAh (*miliamperes* por hora), os valores obtidos paras as políticas ModemSleep, LightSleep, DeepSleep e para o consumo do nó sem uma política aplicada são ilustrados na tabela abaixo (Tabela 3).

Tabela 3 - Dados da Métrica M2 Com Exemplos de Duração Para Cada Duty Cycle Configurado.

Política	**Horas**
ModemSleep	3:34h
LightSleep	3:35h
DeepSleep	140:44h
Sem política	3:34h

Os resultados obtidos apresentados na Tabela 3, mostram que a duração da bateria entre as políticas ModemSleep, LightSleep e a duração sem a aplicação de uma política não houveram alterações significativas, enquanto que com a aplicação da política DeepSleep há um grande aumento dessa duração em comparação com os outros resultados. Dessa forma, a política DeepSleep mostra-se mais eficiente em comparação com os outros resultados, podendo manter um nó alimentado com uma bateria de 250mAh por cerca de 140 horas e 44 minutos, enquanto que um nó com a mesma bateria usando as demais políticas ou sem o uso delas mostram durações de 3 horas e 34 minutos aproximadamente.

A métrica **M3** foi calculada com base na perda de pacotes ocorrida durante o funcionamento do MEPIM. Os dados obtidos representam a quantidade de pacotes que os nós devem entregar em um determinado período de tempo. A Tabela 4 mostra os resultados obtidos durante o funcionamento do MEPIM com duração de aproximadamente 6 horas e intervalo para envio de dados dos nós de 5 minutos utilizando 3, 4 e 6 nós.

Tabela 4 - Dados da Métrica M3 Com Resultados da Avaliação de Enviados de Pacotes.

Total de nós	Quantidade de pacotes enviados	Quantidade de pacotes recebidos
3	216	216
4	288	288
6	432	432

Para a contagem dos pacotes enviados pelos nós, foi utilizado um contador em cada nó, onde seu valor é incrementado cada vez que o nó tenta enviar um pacote. Já a contagem dos pacotes recebidos, foi realizada a partir de uma busca pelos dados salvos no período da realização do teste, considerando que cada dado salvo contém os campos data e hora do envio.

Os resultados apresentados na Tabela 4, mostram a eficiência do MEPIM para os testes realizados em relação a perda de pacotes, considerando que a quantidade de pacotes entregues pelos nós é igual a quantidade de pacotes esperados.

A métrica **M_4**, que se refere ao impacto da inserção de novos nós na rede, foi calculada com base nos dados de perda de pacotes. Os testes feitos foram semelhantes aos realizados para a métrica **M_3**, o que difere é a inserção de novos nós na rede durante seu funcionamento, a fim de analisar o comportamento do sistema e verificar se algum dado deixou de ser entregue.

Tabela 5 - Dados Obtidos Através da Métrica M3 Para a Inserção de Novos Nós.

Total de nós	Quantidade de pacotes enviados	Quantidade de pacotes recebidos
3	180	180
4	246	246
6	372	372

Os resultados apresentados na Tabela 5 mostram que, para os testes realizados com 3, 4 e 6 nós, 100% dos pacotes enviados foram recebidos. Com base na análise dos dados obtidos, pode-se concluir que o MEPIM obteve bons resultados para a inserção de novos nós na rede durante seu funcionamento, considerando que para os testes feitos, nenhum pacote foi perdido.

A métrica **M5**, que se refere ao impacto da retirada de nós da rede durante seu funcionamento. Os testes realizados foram baseados nos testes da métrica **M4**, porém com a realização da remoção dos nós da rede durante seu funcionamento para verificar o impacto de tal ação em relação a perda de pacotes.

Tabela 6 - Dados Obtidos Através da Métrica M5 Resultado da Remoção de Nós.

Total de nós	Quantidade de pacotes enviados	Quantidade de pacotes recebidos
3	186	186
4	246	246
6	360	360

Os resultados apresentados na Tabela 6 mostram que para ambos os testes feitos, 100% dos pacotes enviados foram recebidos. Baseado nos dados apresentados, o MEPIM mostrou um bom desempenho para os testes realizados com a remoção dos nós, sendo assim, pode-se concluir que, dentro dos testes realizados, a remoção de nós não causa perda de pacotes para os demais nós que permaneceram em funcionamento na rede.

Na seção a seguir é apresentada um estudo de caso a fim de mostrar a estrutura do MEPIM em possíveis diferentes cenários de cultivo de microalgas.

4.5 ESTUDO DE CASO

Esta seção ilustra o uso do MEPIM em cenários hipotéticos da aquicultura, mais especificamente em um cultivo de microalgas, explicando a organização do *middleware*, nós da rede e sensores necessários para o monitoramento em cultivo com a finalidade de avaliar seu comportamento em diferentes cenários. Para isso, o objetivo específico enunciado na Introdução (seção 1.3) de *"(ii) Desenvolvimento do módulo responsável pelo envio de dados coletados por sensores (pH, luminosidade, temperatura do ambiente, nível de bateria, etc.) para o Middleware para posterior armazenamento dos dados em uma base de dados*", deve ser avaliado através de diferentes cenários com foco na análise do comportamento do MEPIM. Os diferentes cenários da prova de conceito objetivam demonstrar a estrutura do trabalho aqui proposto e analisar seu funcionamento. A seguir, são descritos os cenários da prova de conceito:

i. Monitoramento de parâmetros externo aos tanques de cultivo de microalgas (Subseção 4.4.1);

ii. Monitoramento de parâmetros interno aos tanques de cultivos de microalgas (Subseção 4.4.2);

iii. Monitoramento dos parâmetros interno e externo aos tanques de cultivo de microalgas (Subseção 4.4.3);

4.4.1 Cenário 1: Monitoramento de parâmetros externos aos tanques de cultivo de microalgas

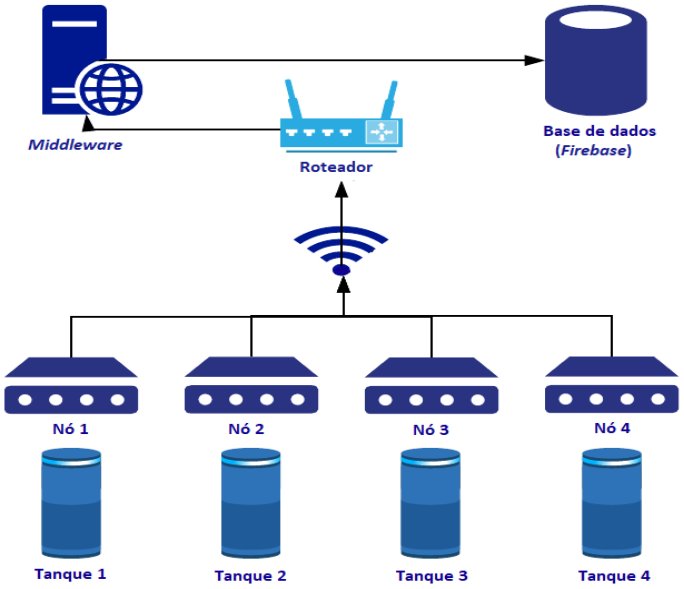
Este cenário considera o uso do MEPIM no monitoramento de parâmetros de algumas variáveis externas aos tanques de cultivos de microalgas, sendo elas temperatura do ambiente e luminosidade do ambiente.

Por não possuir os sensores necessários para o monitoramento das variáveis especificadas, seus dados foram gerados de forma randômica e variando de maneira análoga a do ambiente.

A estrutura dos nós é organizada de forma que cada nó tenha a responsabilidade de monitorar a temperatura e luminosidade de ambiente em apenas um tanque, com a política de consumo de energia DeepSleep com intervalo de sono de 1 hora. Assim, em caso de possíveis falhas ou perda de algum nó, o cultivo não seria comprometido completamente, ou seja, apenas o tanque ao qual o nó responsável parou de funcionar teria seu monitoramento encerrado.

Organizada essa estrutura, para que haja comunicação entre a rede de nós, o *middleware* e o *Firebase*, é necessário o uso de um roteador no ambiente de cultivo, uma vez que a comunicação entre esses módulos é realizada exclusivamente por uma rede Wi-Fi via *Internet*. Na Figura 21 é ilustrada a organização da estrutura do MEPIM, mostrando como foi estruturado cada um dos módulos.

Figura 20 - Estrutura do MEPIM.



FONTE: Elaborada pelo autor.

4.4.2 Cenário 2: Monitoramento de parâmetros internos aos tanques de cultivos de microalgas

Neste cenário é considerado o uso do MEPIM no monitoramento de parâmetros de algumas variáveis internas aos tanques de cultivo de microalgas. As variáveis monitoradas são níveis de Ph e temperatura da água.

Este cenário considera toda a estrutura mencionada no Cenário 1 especificado na seção 4.4.1, considerando que apenas os sensores foram trocados para o monitoramento da região interna dos tanques de cultivos.

Dito isso, a estratégia para geração dos dados dos sensores de forma aleatória também foi mantida para analisar o comportamento do MEPIM em tal cenário.

4.4.3 Cenário 3: Monitoramento dos parâmetros internos e externos aos tanques de cultivo de microalgas

Este cenário o uso do MEPIM é considerado em monitoramento de parâmetros de ambas as regiões (interna e externa) dos tanques de cultivo de microalgas. Sendo as variáveis monitoradas: temperatura e luminosidade do ambiente, temperatura da água e níveis de pH.

Considerando que a única modificação necessária neste cenário são os sensores inseridos para monitoramento das variáveis mencionadas, toda a estrutura dos módulos citada na subseção anterior (4.4.1) foi mantida.

4.4.4 microalgas

Microalgas são seres que pertencem a um grupo abundantemente heterogêneo de organismos aquáticos em sua maioria e, geralmente, microscópicos unicelulares. As microalgas possuem grande importância biológica, ecológica e econômica. O valor biológico desse grupo de organismos está na estrutura da atmosfera da terra, possibilitando a vida sobre a superfície da terra para todos os seres vivos aeróbicos. Seu valor ecológico se dá pelo fato de que as microalgas são produtores primários que sustentam a vida marina desempenhando, assim, um papel ecológico fundamental na manutenção destes ecossistemas. Já sua importância econômica é definida pela sua diversidade de uso em vários países no mundo que vão desde a indústria alimentícia à de medicamentos, da cosmética à agricultura (VIDOTTI, 2004).

O cultivo de microalgas é feito, geralmente, em tanques abertos pouco profundos com inclusão de nutrientes e com luz solar ou artificial, ou em tanques alongados para garantir qualidade da luz solar, podendo ainda serem cultivadas em tanques fotobiorreatores (BERTOLDI et al. 2008). A imagem abaixo mostra um exemplo de tanque fotobiorreator Figura 5.

Figura 21 - Exemplo de tanques fotobiorreatores.

FONTE: (Maniezo e Saleh, 2014).

Seu cultivo um processo delicado que necessita de higienização do ambiente, instrumentos e pessoas envolvidas nesse processo a fim de evitar a proliferação de microrganismos que venham a competir com as microalgas. Devido ao fato de serem organismos extremamente frágeis, é importante controlar com precisão todas as características físico-químicas do ambiente de produção que afetam seu desenvolvimento como temperatura

da água e ambiente, luminosidade, nutrientes, pH da água e salinidade (HAKALIN, 2014). Atualmente, cultivos feitos com fotobiorreatores tubulares, são construídos com materiais resistentes e transparentes permitindo exposição das microalgas à luz e possuindo seções que possibilitam a injeção de ar e troca de gases, entrada de CO_2 e saída de O_2.

Na literatura existem trabalhos propostos para automação no cultivo de microalgas como em PALOMINO (2017), é apresentado um sistema eletrônico com o objetivo de avaliar o impacto de diferentes condições na produtividade de biomassa seca a partir do cultivo de microalgas em tanque fotobiorreator com um controle de parâmetros físicos como luminosidade, temperatura e CO_2 utilizando o microcontrolador Arduino ("Arduino", 2017). Outra proposta é apresentada no trabalho de Mariano (MARIANO et al., 2017), onde os autores desenvolveram uma aplicação para o monitoramento automatizado de parâmetros no cultivo de microalgas. Nessa aplicação o sistema é capaz de realizar o monitoramento de forma automatizada, onde as análises são feitas sem supervisão após serem programadas.

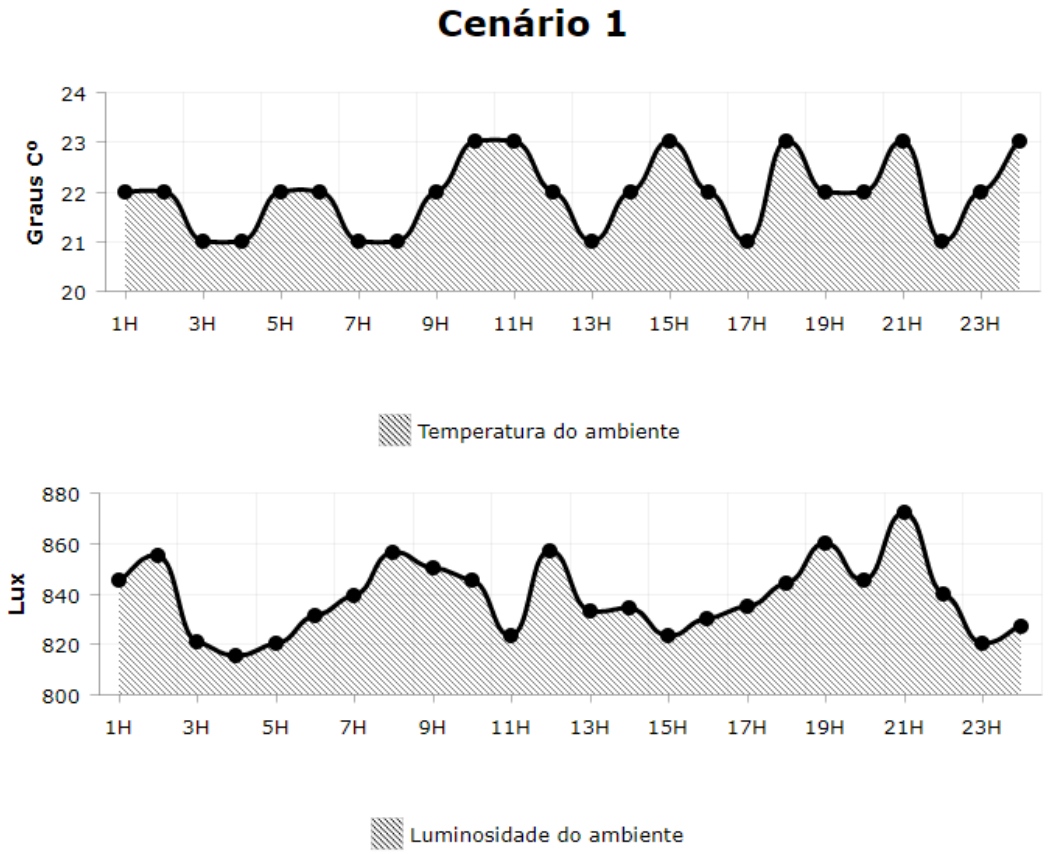
Dessa forma, diante do seu alto valor econômico, a produção de microalgas vem sendo feita com foco em produzir biomassa para fabricação de produtos alimentícios e para aquisição de compostos naturais de elevado valor no mercado mundial (DENER et al., 2006). Nesse contexto, o uso da tecnologia aliada a atividades como o cultivo de microalgas é uma forma de potencializar a produtividade.

4.6 ANÁLISE DO ESTUDO DE CASO

Nesta seção são apresentados os resultados obtidos em cada cenário da prova de conceitos e uma análise desses resultados. A partir dos dados coletados pelos nós da rede, foram construídos gráficos que permitem a um usuário do MEPIM analisar os dados de temperatura do ambiente, temperatura da água, nível de pH e luminosidade do ambiente de acordo com os sensores usados em cada cenário.

No cenário 1, o MEPIM funcionou como esperado, sem apresentar falhas na comunicação ou coleta de dados, possibilitando a coleta dos dados de temperatura do ambiente e luminosidade do ambiente. Ao fim do teste, foi gerado o gráfico a partir dos dados coletados como mostra a Figura 22 abaixo.

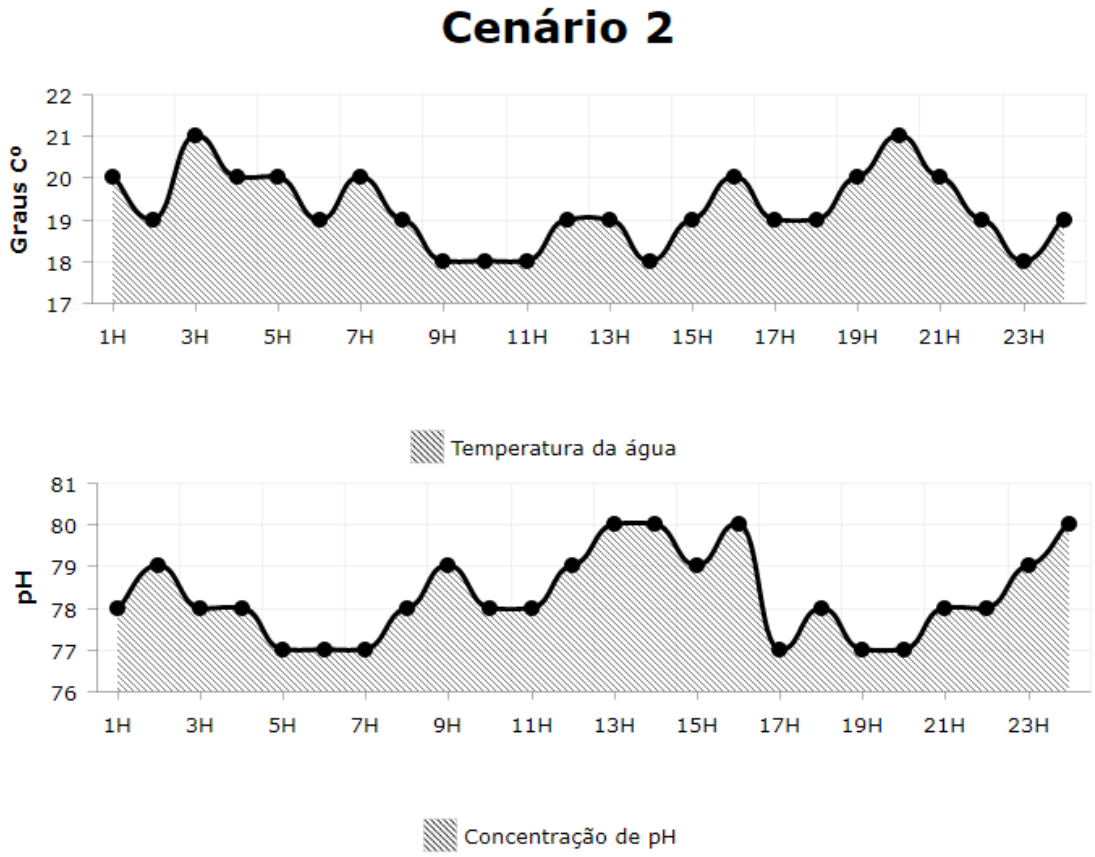
Figura 22 - Dados dos Sensores do Cenário 1.



FONTE: Elaborada pelo autor.

No cenário 2, o MEPIM também obteve o comportamento esperado, coletando os dados dos sensores no intervalo de 1 hora para cada nova coleta. A Figura 23 mostra o gráfico dos dados dos sensores de temperatura da água e pH, obtidos a partir da coleta feita nos testes do cenário 2.

Figura 23 - Dados dos Sensores do Cenário 2.

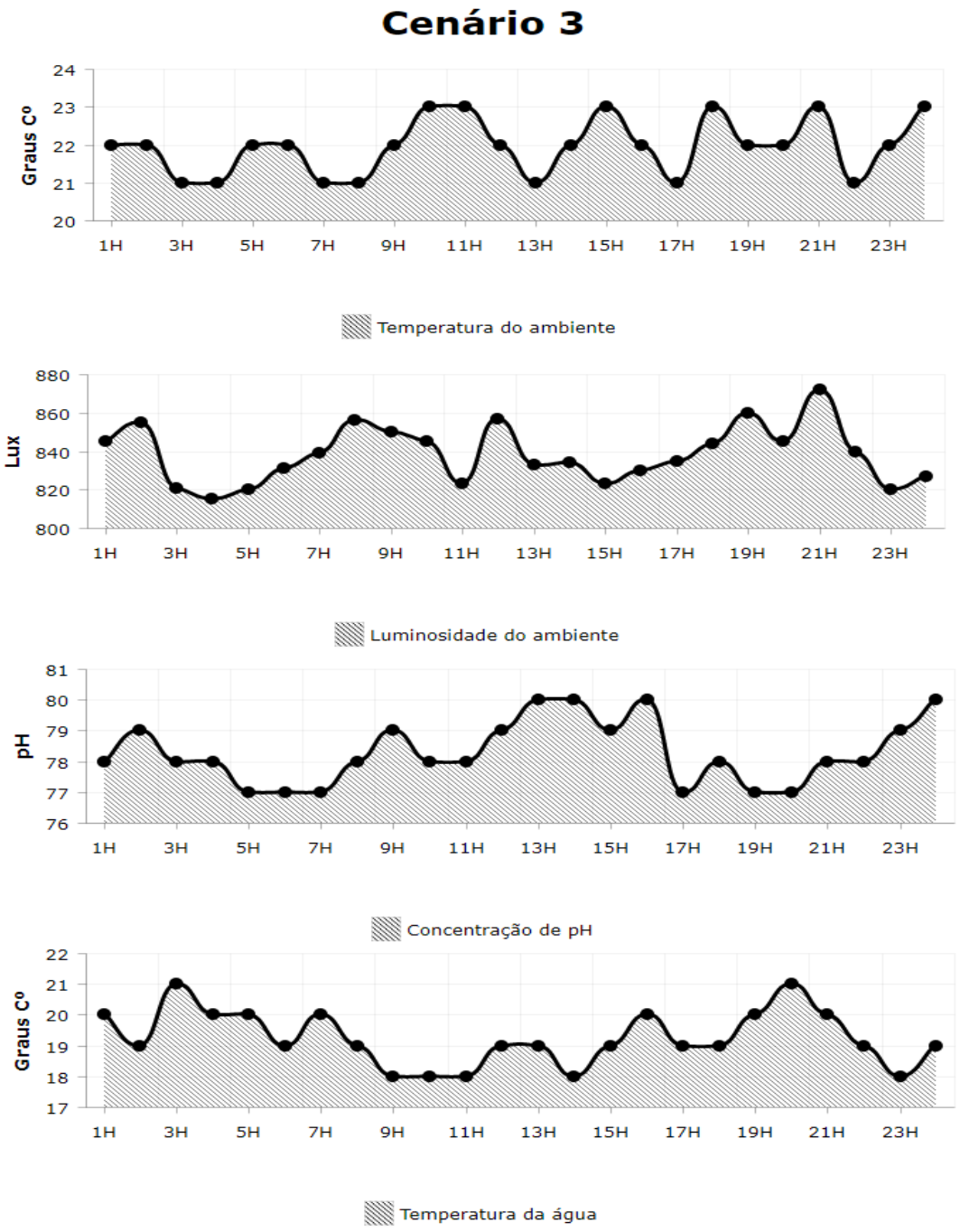


FONTE: Elaborada pelo autor.

No cenário 3, uso do MEPIM obteve êxito nos testes e funcionou como desejado. A coleta dos dados dos sensores de temperatura da água, pH, temperatura do ambiente e luminosidade ocorreu normalmente dentro do intervalo definido, possibilitando a geração do gráfico como ilustrado na Figura 24.

Diante dos resultados obtidos nos 3 cenários, o MEPIM se mostrou eficiente para os testes realizados, obtendo o comportamento esperado em todos eles e sem apresentar falhas ou erros que impedissem seu funcionamento.

Figura 24 - Dados dos Sensores do Cenário 3.



FONTE: Elaborada pelo autor.

5. CONSIDERAÇÕES FINAIS

O uso de um *middleware* como proposta para a redução de consumo de energia dos nós de uma RSSFs aplicado as ciências agrárias, funcionando como intermediador entre a RSSFs e uma base de dados, utilizando as tecnologias apropriadas, demonstra ser uma solução viável, uma vez que seu uso possibilita uma redução considerável no consumo de energia desses nós. As características e arquitetura MEPIM, permitem a sua aplicação não somente ao cultivo de microalgas, mas também para qualquer outro objetivo que necessite de monitoramento do ambiente. O presente trabalho foi motivado pelos conceitos de RSSFs, onde o uso do MEPIM é proposto como ferramenta para redução de consumo energético dos nós de uma rede aplicado ao monitoramento de um cultivo de microalgas, coletando dados referentes aos parâmetros considerados essenciais para o desenvolvimento deste cultivo, com dados gerados randomicamente. Com isso, espera-se que a utilização deste sistema, juntamente com os avanços da área de RSSFs, possa contribuir de forma significativa considerando sua usabilidade e benefícios.

5.1 TRABALHOS FUTUROS

Diante do que foi abordado neste trabalho, as aplicações de *middleware* para RSSFs possui uma vasta gama de aplicabilidade, o que possibilita o desenvolvimento de diversos trabalhos futuros. Como por exemplo, a geração de gráficos a partir dos dados coletados pelo módulo node e notificações de quando um determinado sensor obtem um valor de uma variável considerada crítica para o ambiente de monitoramento.

O MEPIM não conta com uma interface de usuário que possibilite a interação dos usuários com o sistema de forma agradável para a realização de consulta dos dados coletados pela rede e a gerencia das políticas de consumo de energia. Dessa forma, o desenvolvimento de uma interface de usuário que possibilite tal interação é um trabalho futuro atraente.

A arquitetura do MEPIM foi baseada na arquitetura orientada a serviços, possibilitando uma futura expansão para incluir novos módulos como, por exemplo um módulo para realizar a predição de dados coletados pelos nós de uma RSSFs para economia de energia. Onde esse módulo permitiria que, diante da obtenção de dados de sensores com pouca variação, os nós entrem em estado de hibernação para poupar energia de suas baterias.

Outro trabalho futuro interessante é a implementação da autodetecção dos sensores conectados aos nós da rede. Na versão atual do MEPIM, cada sensor a ser utilizado precisa ser especificado via linha de código, no nó ao qual ele será conectado, para que esse sensor funcione corretamente.

REFERÊNCIAS

ARDUINO. Disponível em: <https://www.arduino.cc/>. Acesso em: 10 abril. 2018.

BRASIL, T, V, L. Aplicação de *Middleware* para o Monitoramento de Unidade Produtiva na Aquicultura. 2017. 57 f. TCC (Graduação) - Curso de Análise e Desenvolvimento de Sistemas, Unidade Especializada em Ciências Agrárias, Universidade Federal do Rio Grande do Norte, Macaíba, 2017, p-17.

BASILI, V. 1992. "Software Modeling and Measurement: The Goal/Question/Metric Paradigm". College Park, MD, USA: University of Maryland. Technical Report CS-TR-2956. Dsponível em: <http://www.cs.umd.edu/~basili/publications/technical/T78.pdf>. Acessado em 30 maio 2018.

BERTOLDI, F. C.; SANT'ANNA, E.; OLIVEIRA, J. L.B. Revisão: Biotecnologia de Microalgas. Boletim CEPPA, v.26, n. 1, p. 9-20, 2008.

BOOTH, D. et al. 2004. Web Services Architecture. Disponível em <http://www.w3.org/TR/2004/NOTE-ws-arch-20040211/>. Acessado em 18 de Abril de 2018.

CLOUDMQTT. Disponível em: < https://www.cloudmqtt.com/>. Acessado em: 01 abril 2018.

COULOURIS, G. et al. Sistemas Distribuídos: Conceitos e Projetos. 4º Edição. Bookman. 2007.

CORSARO, A. et al. (2006). Quality of Service in Publish/Subscribe *Middleware*.

CUNHA, M. X. C.; MARCILIO, F. S. J.; e Dornelas, J. S. 2006. "O uso da arquitetura SOA como estratégia de integração de sistemas de informação em uma instituição pública de ensino". 1–13. Disponível em: < https://www.aedb.br/seget/arquivos/artigos08/498_integracao_seget_final.pdf >. Acessado em: 07 jul. 2018.

CHAPPELL, D. A. and JEWELL, T. (2002), Java Web Services: Up and Running – O'Reilly Media; 1 edition.

CARVALHO, F. B. S. et al. 2012. "Aplicações Ambientais de Redes de Sensores Sem Fio". Bibliotekevirtual.Org 14–19. Disponível em: <http://www.bibliotekevirtual.org/revistas/RTIC/v02n01/v02n01a03.pdf>. Acessado em: 21 maio 2018.

DERNER, R. B.; OHSE, S.; VILLELA, M.; CARVALHO, S. M. de; FETT, R. Microalgas, produtos e aplicações. Ciência Rural, v.36, n. 6, p. 1959-1967, 2006.

ESPRESSIF. Disponível em: <https://www.espressif.com/>. Acessado em 18 abril 2018.

FIREBASE. Disponível em: <https://firebase.google.com/>. Acessado em 01 abril 2018.

FIELDING, R. T. Architectural Styles and the Design of Network-based Software Architectures. Disponível em: <https://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm>. Acessado em 28 maio 2018.

GILL, C. D. e SMART, W.D. (2002). *Middleware* for Robots?, Proceedings of AAAI Spring Symposium on Intelligent Distributed and Embedded Systems.

GAMMA, E. et al. Padrões de Projetos. 1. Ed. Bookman, 2000.

HAKALIN, N. L. (2014). OTIMIZAÇÃO DAS CONDIÇÕES DE CULTIVO DA MICROALGA Scenedesmus sp. PARA A PRODUÇÃO DE BIODIESEL. Tese de Doutorado (Biologia Molecurar), Instituto de Ciências Biológicas, Departamento de Biologia Celular, Brasília, p. 38. 2014.

JAVA. Disponível em <https://java.com/>. Acessado em 5 abril 2018.

KIRROLIA, A.; BISHNOI, N. R.; & SINGH, R. 2013. Microalgae as a boon for sustainable energy production and its future research & development aspects. Renewable Sustainable Energy Rev., 20: 642-656.

LAMAS, F. M. A Tecnologia na Agricultura. Embrapa, 2017. Disponível em: < https://www.embrapa.br/busca-de-noticias/-/noticia/30015917/artigo-a-tecnologia-na-agricultura>. Acessado em 5 jun. 2018.

LOPES, B. S. e DIAS, J. W. 2011. "Utilizando os Diagramas da UML (Linguagem Unificada de Modelagem) para desenvolver aplicação em JSF".

LOUREIRO, A. a. F.; NOGUEIRA, J. M. S.; RUIZ, L. B.; MINI, R. A. D. F.; NAKAMURA, E. F.; & FIGUEIREDO, C. M. S. (2003). Redes de Sensores Sem Fio. XXI Simpósio Brasileiro de Redes de Computadores, 179–226. Disponível em: <http://homepages.dcc.ufmg.br/~loureiro/cm/docs/sbrc03.pdf>. Acessado em: 12 maio 2018.

LUA. Disponível em: <https://www.lua.org/portugues.html>. Acessado em 18 abril 2018.

MANIEZO, G. SALEH, D. jornalcomunicacao.ufpr.br. Energia produzida com o uso de microalgas é pioneira na UFPR, 2014. Disponível em: <https://guiadamonografia.com.br/como-citar-imagens-e-graficos-no-tcc/>. Acessado em: 11 jun. 2018. Il color.

MARIANO, A. B.; ALMEIDA, A. R.; MOREIRA, G. F. N.; BOEHME, J. H.; LIMA, J. M; MONTEIRO, L. V.; LOPES, P. H. P. M.; BELTRÃO, R. K, V; GANDIN, Y. E. M. (2017). Monitoramento Automatizado De Parâmetros No Cultivo De Microalgas Em Sistemas Fotoautotrófico Com Foco Na Produção De Biocombustíveis. In: CRICTE XXVIII? Congresso Regional de Iniciação Científica e Tecnológica em Engenharia, Ijuí - RS. CRICTE XXVIII.

MAGNO, R.; RECHARTE, D.; GONÇALVES, D.; e FERRÃO, D. 2013. "Como evoluiram as normas Wi-Fi IEEE 802.11?" 1–24. Disponível em: <https://paginas.fe.up.pt/~projfeup/submit_13_14/uploads/relat_1MIEEC01_3.pdf>. Acessado em: 07 jul. 2018.

OLIVEIRA, Marcelo Eduardo de. Desenvolvimento de sistema automatizado de monitoramento de ambientes de produção animal, utilizando uma rede de sensores sem fio. 2016. Dissertação (Mestrado em Gestão e Inovação na Indústria Animal) - Faculdade de Zootecnia e Engenharia de Alimentos, Universidade de São Paulo, Pirassununga, 2016. doi:10.11606/D.74.2016.tde-01062016-102252. Acesso em: 07 jul. 2018.

OASIS. Disponível em: <http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=soa-rm>. Acessado em 23 abril de 2018.

PANTAZIS, N. A.; NIKOLIDAKIS, S. A.; & VERGADOS, D. D. 2013. Energy-e-cient routing protocols in wireless sensor networks: A survey. Communications Surveys Tutorials, IEEE, 15(2):551591.

PEREIRA, P. H. C. et al. Agricultura de precisão com rede de sensores sem fio. Revista de Tecnologia da Informação e Comunicação, Campina Grande, v. 4, n. 2, p. 19-27, out. 2014.

PIRES, P. F. et al. 2015. “Plataformas para a Internet das Coisas”. Anais do Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos 110–69.

PALOMINO, R. L. Q. (2017). Desenvolvimento de sistemas eletrônicos para operação e monitoramento de parâmetros envolvidos na produção de biomassa a partir de microalgas em fotobiorreatores. Universidade Estadual Paulista (UNESP).

RODRIGUES, T. C. 2015. “ArchWiSeN: Uma Estratégia Baseada em Modelos para Desenvolvimento de Aplicações para Redes de Sensores e Atuadores Sem Fio ArchWiSeN : Uma Estratégia Baseada em Modelos Sensores e Atuadores Sem Fio”. Disponível em: < https://repositorio.ufrn.br/jspui/bitstream/123456789/19895/1/TaniroChaconRodrigues_TESE.pdf >. Acessado em: 20 maio 2018.

RODRIGUES, T. C. 2011. “Abordagem dirigida a modelos para redes de sensores sem fio”. Disponível em: <https://repositorio.ufrn.br/jspui/bitstream/123456789/18017/1/TaniroCR_DISSERT.pdf >. Acessado em: 20 maio 2018.

REDHAT. Disponível em: < https://canaltech.com.br/software/o-que-e-api/ >. Acessado em 7 de jun. 2018.

TOMCAT. Disponível em <http://tomcat.apache.org/>. Acessado em 18 abril 2018.

VIDOTTI, E. C.; ROLLEMBERG, M. do C. E. Algas: da economia nos ambientes aquáticos à bioremediação e à química analítica. Quím. Nova, v.27, n. 1, 2004.

W3C, Web Services Architecture. Disponível em: <https://www.w3.org/>. Acessado em 24 abril de 2018.

ZIMMERMANN, M. M. D. O. (2006). LUAPS - LUA PUBLISH-SUBSCRIBE. PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO DE JANEIRO.