UNIVERSIDADE DO VALE DO RIO DOS SINOS - UNISINOS
UNIDADE ACADÊMICA DE GRADUAÇÃO
CURSO DE CIÊNCIA DA COMPUTAÇÃO

SÉRGIO LUIZ BONEMBERGER JUNIOR

Desenvolvimento de sensores inteligentes aplicados para a manutenção preditiva na Industria 4.0

SÃO LEOPOLDO
2019

Sérgio Luiz Bonemberger Junior

DESENVOLVIMENTO DE SENSORES INTELIGENTES APLICADOS PARA A MANUTENÇÃO PREDITIVA NA INDÚSTRIA 4.0

Artigo apresentado como requisito parcial para obtenção do título de Bacharel em Ciência da Computação, pelo Curso de Bacharelado em Ciência da Computação da Universidade do Vale do Rio dos Sinos - UNISINOS

Orientador: Prof. Dr. Rafael Kunst

São Leopoldo

2019

DESENVOLVIMENTO DE SENSORES INTELIGENTES APLICADOS PARA A MANUTENÇÃO PREDITIVA NA INDÚSTRIA 4.0

Sérgio Luiz Bonemberger Junior *

Rafael Kunst **

Resumo: A quarta revolução industrial também chamada de Indústria 4.0, está, cada vez mais, integrando o mundo físico ao mundo digital, através da implementação de sistemas ciberfísicos em plantas de produção industrial. O desenvolvimento deste tipo de sistema utiliza a Internet das Coisas para permitir que equipamentos troquem informações entre si e com os seus arredores, constituindo assim, um ambiente industrial inteligente. Entre os vários conceitos que estão surgindo ou se modernizando devido a quarta revolução industrial, um que pode ser destacado é o de Manutenção Preditiva, que busca prever quando uma máquina irá apresentar defeito com base em sinais previamente emitidos pela mesma. Considerando a relevância da Manutenção Preditiva utilizando sensores de Internet das Coisas no contexto da Indústria 4.0, neste trabalho foi realizada uma pesquisa nas áreas de Internet das Coisas e Computação em Neblina, que resultou na implementação de uma arquitetura de baixo custo para coletar dados relativos ao funcionamento de equipamentos em indústrias. Foram desenvolvidos sensores para coletar os dados de vibração de máquinas, com a periodicidade de medições configurável, permitindo maximizar a vida útil dos mesmos mantendo a qualidade dos dados coletados. Para validar o funcionamento do modelo proposto foram realizados testes com e sem a utilização da Camada de Neblina, com uso de diferentes frequências na coleta de dados e com ou sem aplicação de um algoritmo de compressão de dados. Os resultados não comprovaram que a utilização de uma Camada de Neblina teve impacto significativo na redução do consumo de energia. Nos testes com diferentes frequências na coleta de dados, no entanto, pode ser verificado uma relação entre o volume de dados transmitidos com a energia dispendia nas operações.

Palavras-chave: Internet das Coisas. Industria 4.0. Sensores. Manutenção Preditiva Computação em Neblina.

Abstract: The fourth industrial revolution, also called Industry 4.0 is increasingly integrating the physical world with the digital world through the use of cyberphysical systems in industrial production plants. The development of this type of system uses the Internet of Things to enable equipment to exchange information with each other and their surroundings, thus constituting an intelligent industrial environment. Among many concepts that are emerging due to the fourth industrial revolution, one that can be highlighted is Predictive Maintenance, which seeks to predict when a machine will fail based on signals previously emitted by it. Considering the relevance of Predictive Maintenance using IoT sensors in the context of Industry 4.0, this work carried out a

* Email: sergio.bonemberger@gmail.com
** Email: rafaelkunst@unisinos.br

research in the areas of IoT and Fog Computing, which resulted in the development of a low cost architecture to collect data related to equipment operation in industries. Sensors were developed to collect machine vibration data, with configurable measurement periodicity, allowing them to maximize machine lifetime while maintaining the behavior of the collected data. To validate the proposed model, tests were performed with and without the use of the Fog Layer, with the use of different frequencies in the data collection and with or without the application of a data compression algorithm. The results did not prove that the use of a Fog Layer had a significant impact on reducing energy consumption. However, in tests with different frequencies in data collection, a relationship could be verified between the amount of data transmitted with the energy spent on operations.

Keywords: Internet of Things. Industry 4.0. Sensors. Predictive Maintenance. Fog Computing.

1 INTRODUÇÃO

A Internet das Coisas (*Internet of Things* – IoT) é um conceito utilizado para descrever ambientes onde objetos se comunicam uns com os outros através de uma estrutura de rede, de forma a trocarem informações sobre si próprios, sobre demais objetos ao seu redor e sobre o ambiente em que estão inseridos (FISCHER, 2019). A aplicabilidade desse conceito tem crescido com velocidade, sendo estimado que até 2025 todos os objetos do nosso dia-a-dia estarão conectados com o mundo digital através de IoT (SARHAN, 2018).

Uma das áreas onde a aplicação de IoT tem se destacado é a área de automação industrial. Nesse campo, o uso de IoT vem sendo denominado IIoT (*Industrial Internet of Things* ou Internet das Coisas Industriais). A IIoT representa um ambiente industrial interconectado, onde máquinas trocam informações em tempo real entre si e com demais sistemas e serviços, através da utilização de componentes inteligentes (CIVERCHIA 2017). O desenvolvimento de ambientes IIoT é entendido como um dos pilares de uma nova revolução industrial, a qual está sendo referenciada como Indústria 4.0. Esse termo é frequentemente entendido como a aplicação do conceito de sistemas ciberfísicos (*Cyber Physical Systems* – CPS) a sistemas de produção industrial (DRATH; HORTH, 2014).

Dentro da Indústria 4.0, a Manutenção Preditiva (*Predictive Maintenance* – PdM) é abordada como um tema central, uma vez que a aplicabilidade desse conceito cria um ambiente para o desenvolvimento de ferramentas de PdM. Nessas ferramentas, a utilização de IoT atua como uma plataforma de comunicação de

dados que permite a implementação de CPS's dotados de algoritmos de predição, os quais têm por objetivo indicar quando a manutenção do maquinário deverá ser feita, tornando o processo de produção mais eficiente e por consequência reduzindo os custos (YU-CHEN, 2017).

Neste trabalho é proposta uma arquitetura de software de baixo custo, que permite a coleta de dados de vibração de máquinas, através da implementação de sensores baseados em IoT em conjunto com a utilização do paradigma de Computação em Neblina (conhecida como *Fog Computing*) para redução do tempo gasto na comunicação dos dados. Também foi aplicado um algoritmo de compressão de dados para reduzir o volume dos dados enviados para a Camada de *Fog*. A implementação dessa arquitetura visa prover dados para trabalhos futuros, que se proponham a desenvolver algoritmos baseados em aprendizagem de máquina para identificar a necessidade de manutenção preditiva em equipamentos.

A arquitetura implementada foi validada através da verificação de diferentes cenários, com e sem a utilização da Camada de *Fog,* com diferentes frequências de coleta de dados e com ou sem utilização de um algoritmo de compressão de dados. Os testes não demonstraram uma redução significativa no consumo de energia ao ser utilizada a Camada de *Fog*. No entanto foi possível verificar que o volume de dados transmitidos possui sim uma relação com consumo de energia.

O presente trabalho está organizado em 6 capítulos. Após esta introdução o Capítulo 2 irá tratar do Referencial Teórico que foi utilizado para o desenvolvimento do trabalho, apresentando os principais conceitos necessários para o entendimento da solução proposta nos capítulos posteriores. O Capítulo 3 é dedicado a apresentação e análise das características de trabalhos que possuem semelhanças com o projeto que foi desenvolvido nesse trabalho. No Capítulo 4, é apresentada a solução proposta. Este capítulo explana sobre os dispositivos que foram desenvolvidos e os detalhes da arquitetura. No Capítulo 5 são analisados os resultados observados com os experimentos. Por fim o Capítulo 6 traz as conclusões e contribuições obtidas com desenvolvimento desse trabalho e proposições de possíveis trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capitulo serão introduzidos os conceitos que são pilares para o desenvolvimento do presente trabalho. O mesmo está organizado em 5 seções principais, conforme segue. Na seção 2.1, são apresentados os fundamentos referentes à Indústria 4.0. Em seguida, na seção 2.2, serão introduzidos os conceitos relacionados a IoT. A seção 2.3, aborda microcontroladores utilizados para o desenvolvimento de soluções de IoT. A seção 2.4, apresenta os principais tópicos relacionados à *Cloud Computing*. Finalmente, a seção 2.5, discorre sobre o paradigma de *Fog Computing* e suas aplicabilidades.

2.1 Indústria 4.0

O termo Indústria 4.0 está sendo utilizado para representar a quarta revolução industrial que está ocorrendo nos últimos anos. O termo inicialmente foi proposto em 2011 na feira de Hannover na Alemanha, sendo posteriormente anunciado em 2013 como uma estratégia do governo alemão para desempenhar um papel pioneiro no setor industrial, estando em constante expansão desde então (XU; XU; LI, 2018).

De acordo com Vogel-Heuser e Hess (2016), existe mais de uma definição para a Indústria 4.0, mas a maioria delas concorda com os seguintes princípios:

- **Orientação a Serviços**: CPPS (*Cyber Physical Production Systems* ou Sistema de Produção Ciberfísicos) devem oferecer serviços através da internet e de uma arquitetura orientada a serviços;
- **Sistemas inteligentes:** desenvolvimento de sistemas capazes de tomar decisões sozinhos;
- **Acompanhamento em tempo real:** capacidade de sistemas se comunicarem com usuários e entre si, provendo aos usuários a possibilidade de acompanhar/analisar tudo que está ocorrendo na planta de produção de forma virtual em tempo real;
- **Modularização:** capacidade de poder adaptar ou alterar requisitos através da expansão ou troca de módulos individuais;
- ***Big Data***: uso de algoritmos e tecnologias de *Big Data* disponibilizadas em tempo real.

- **Eficiência**: aumento da eficiência dos equipamentos com base em dados fornecidos pelos sistemas;
- **Padronização:** integração de dados durante todo o ciclo produtivo baseada na implementação de modelos de dados padronizados;
- **Segurança:** acesso seguro a dados armazenados em *cloud* ou intranet.

Para Lu (2017), a Industria 4.0 é caracterizada pelo desenvolvimento de processos altamente automatizados e digitalizados através do uso de componentes eletrônicos combinados com projetos na área de tecnologia da informação. Dentre as principais tecnologias se destacam Computação Móvel, *Cloud Computing*, *Big Data* e IoT.

2.1.1 Manutenção Preditiva (*Predictive Maintenance* – PdM)

PdM não é um conceito novo na indústria, que consiste em detectar sinais que um equipamento começa a apresentar antes de realmente ocorrer uma falha significativa com o mesmo. Inicialmente, esses sinais eram percebidos através da observação de especialistas, sendo a inspeção visual a forma mais comum de PdM. Apesar da inspeção visual permanecer como um método amplamente utilizado, esse processo de detecção de sintomas evoluiu para métodos automatizados, baseados na utilização de sensores para coleta de dados, que posteriormente são analisados através do uso de técnicas de reconhecimento de padrões, permitindo assim evitar trocas de peças desnecessárias e aumentando a segurança e eficiência do processo (HASHEMIAN, 2010).

Para Selcuk (2017), a PdM é empregada para atingir dois objetivos principais: prevenção de falhas e melhora na eficiência dos processos. A prevenção de falhas é o principal objetivo para o uso de PdM, uma vez que 99% das falhas em máquinas são precedidas por algum sinal de discrepância na operação da mesma. Do ponto de vista de melhoria na eficiência do processo PdM ajuda a aumentar segurança do processo, aumentar a qualidade dos produtos, melhorar a confiabilidade do processo, aumentar a disponibilidade dos recursos, reduzir custos com trocas de partes e trabalho manual, reduzir o desperdício de matéria prima e de consumíveis e reduzir o consumo de energia das máquinas.

2.2 Internet das Coisas (*Internet of Things* – IOT)

IoT é um paradigma que está crescendo rapidamente no cenário da comunicação entre dispositivos. A ideia central por trás desse conceito é a presença de objetos inteligentes inseridos no contexto das atividades dos usuários, que sejam capazes interagir e cooperar entre si e com seus vizinhos, com o intuito de atingir algum objetivo comum (ATZORI; IERA; MORABITO, 2010). O termo foi inicialmente proposto para se referir à utilização da tecnologia RFID (*Radio Frequency Identification* ou Identificação por Rádio Frequência) para produzir objetos interconectados, com funcionalidades interoperáveis e providos de um identificador único. Posteriormente, pesquisadores começaram a utilizar o termo de forma mais ampla para se referir também a outras tecnologias (DA XU; HE; LI, 2014)

A aplicação do paradigma de IoT é principalmente sustentada no alto impacto que o mesmo pode ter nos aspectos do dia-a-dia dos seus usuários. Na perspectiva de usuários privados, esse impacto deverá ser claramente percebido, tanto em atividades domésticas, quanto em ambientes de trabalho. (ATZORI; IERA; MORABITO, 2010).

No contexto da Indústria 4.0, a IoT é uma tecnologia estruturante, sendo uma base sólida para o desenvolvimento e evolução de CPS's. Desta forma, permitindo o desenvolvimento de uma nova geração de sistemas, que integram informações do mundo físico com o cibernético. Possibilitando que empresas inseridas no setor produtivo executem a sua visão de modernização de seus processos, com o objetivo de torna-los mais inteligentes e eficientes. Neste sentido IoT pode ser aplicado em diversos domínios da indústria, como por exemplo a manutenção de equipamentos, onde a evolução de sensores aliada a técnicas de IA (Inteligência Artificial), permitem inferir com maior precisão o estado do maquinário, ou a rastreabilidade da cadeia produtiva de produtos, permitindo identificar todo o ciclo de vida de cada produto individualmente (XU; XU; LI, 2018).

2.3 Microcontroladores

Microcontroladores são módulos que provêm para o usuário os principais elementos necessários para execução de um programa em um único chip, o que permite reduzir o custo para execução de aplicações com menor complexidade.

Dessa forma, pode-se ter um sistema funcionando, sendo composto apenas por um microcontrolador em conjunto com poucos componentes externos para permitir que o mesmo se comunique com o seu ambiente e demais sistemas (BATES, 2011).

Com a evolução da tecnologia e surgimento da IoT, a aplicação de microcontroladores para o desenvolvimento de objetos inteligentes através de sistemas embarcados aumentou, uma vez que as possibilidades de aplicações que podem ser desenvolvidas com os mesmos são ilimitadas, e já existindo diversas opções de hardware e software que podem ser utilizados nesse processo, como por exemplo o Arduino, o Raspberry PI e o ESP8266 (SINGH; KAPOOR, 2017).

2.3.1 Raspberry Pi

O Raspberry Pi é um computador em única placa. O mesmo possui por padrão processador ARM, uma central para processamento de gráficos, portas USB e Ethernet, conector HDMI e um *slot* para cartão micro-SD. Também é importante ressaltar que o Raspberry Pi suporta diversos sistemas operacionais, como por exemplo o Raspian Linux, o Ubunto Matae e o Windows 10 IoT Core, assim como várias linguagens de programação (SINGH; KAPOOR, 2017).

Conforme descrito por Vujovic e Maksimovic (2015), o Raspberry Pi pode funcionar como um computar pessoal, recebendo comandos de periféricos, como teclado e mouse e exibindo os resultados em um monitor, mas também pode ser utilizado como um servidor *web*, respondendo a chamadas feitas por outros dispositivos.

2.3.2 ESP8266

O ESP8266 é um microcontrolador de baixo custo, produzido pela empresa Espressif System’s como parte da sua plataforma de conectividade. O módulo possui um processador de 32 bits, 64 KB de memória SRAM, 96 KB de memória DRAM, 16 pinos GPIO, 2,4 GHz Wi-Fi e de acordo com a fabricante, pode operar em temperaturas entre -40 ºC e 125 ºC (RODRIGUES; CASTRO, 2018).

De acordo com Polianytsia e Herasymenko (2016), o ESP8266 é o microcontrolador adequado para cenários onde os objetivos da aplicação não irão fazer uso de todas as funcionalidades providas por plataformas mais robustas como

o Raspberry Pi ou o Arduino, visto que para essas demandas o ESP8266 pode entregar o mesmo resultado a custos mais baixos.

2.4 Computação em Nuvem (*Cloud Computing)*

O sucesso da Internet somado ao rápido desenvolvimento de tecnologias de processamento e armazenamento de dados permitiu o surgimento do modelo de computação baseado na nuvem. O conceito é que usuários podem usar e liberar recursos sob demanda sem a necessidade de eles mesmos realizarem a aquisição de *hardwares* para atender as suas necessidades. O modelo se popularizou com gigantes como Google, Amazon e Microsoft desenvolvendo plataforma robustas para *Cloud Computing* enquanto empresas buscam adequar sua estrutura a esse novo modelo (ZHANG; CHENG; BOUTABA, 2010).

De acordo com Zhang, Cheng e Boutaba (2010), *Cloud Computing* possui uma série de características que a tornam atrativa para empresas: I Não há necessidade investimento inicial; II Redução dos custos operacionais; III Alta escalabilidade; IV Fácil acesso e V Redução de riscos para o negócio.

Conforme Pflanzner e Kertesz (2016), provedores de *Cloud Computing* podem ser categorizados em 3 grupos: PaaS (*Platform as a Service*), IaaS (*Infrastructure as a Service*) e SaaS (*Software as a Service*).

Serviços de IaaS se referem a disponibilização de recursos de infraestrutura sob demanda (ZHANG; CHENG; BOUTABA, 2010). Provedores desse tipo de serviço geralmente realizam o gerenciamento sobre o hardware, garantindo que o datacenter tenha um alto índice de disponibilidade e buscando preços competitivos para a alocação de recursos. Fica a cargo de quem contrata o serviço a responsabilidade de instalar e manter os sistemas que estiverem fazendo uso da infraestrutura da *cloud*.

Serviços de PaaS se propõem a disponibilizar um ambiente para que os consumidores possam desenvolver suas aplicações. Não se trata apenas da disponibilização de máquina como em IaaS, mas também de componentes que são necessários para o desenvolvimento de sistemas, como sistemas operacionais e banco de dados servidor *web* (PFLANZNER; KERTEZ, 2016).

Serviços de SaaS são propriamente sistemas providos pela Internet. Nesse modelo, os usuários não estão interessados em recursos físicos da *cloud*, mas nas funcionalidades dos sistemas em si (PFLANZNER; KERTEZ, 2016).

2.5 Computação em Neblina (*Fog Computing)*

Fog Computing é uma plataforma que busca prover serviços de computação, armazenamento e rede entre dispositivos e a *cloud*. Em geral, ela fica localizada nos limites da rede, tendo surgido com o objetivo inicial de atender aplicações que produzem um alto volume de dados, como por exemplo aplicações baseadas em IoT ou que precisam de baixos tempos de latência como por exemplo *streaming* de vídeos (YI; LI; LI, 2015). A *Fog Computing* não surgiu para substituir a *Cloud Computing*, mas sim para permitir o desenvolvimento de novos tipos de aplicação que não são possíveis ou viáveis apenas com o emprego da *Cloud Computing* (BONOMI et al., 2012).

O termo *Fog Computing* é uma analogia ao termo *Cloud Computing*, pois essa plataforma busca aproximar da rede local algumas características da *Cloud Computing*, assim como a neblina é uma nuvem próxima do chão (BONOMI et al., 2012). De acordo com Puliafito et al. (2019), *Fog Computing* ganhou força devido a necessidade da utilização de uma nova estratégia de comunicação para atender dispositivos IoT, uma vez que a comunicação dos mesmos diretamente com *clouds* pode significar distâncias geográficas consideráveis.

Um exemplo de aplicação da Camada de *Fog* é apresentado por Krishnan, Bhagwat e Uptat (2015), no qual é utilizado um Raspberry Pi para atuar como um dispositivo da Camada de *Fog,* sendo responsável, não apenas, por receber os dados de um sensor de temperatura, mas também por realizar um processamento dos dados antes de enviá-los para a *cloud*. De acordo com os autores, a arquitetura contemplando a Camada de *Fog* se mostrou mais eficiente. Os resultados do experimento são apresentados na Tabela 1 a seguir:

Tabela 1 – Resultados dos experimentos com e sem *Fog Computing*

Atributo	Arquitetura de Cloud Tradicional	Fog Computing
Latência da Predição	5 segundos	1,5 segundos
Latência da Exibição da Página Web	8 segundos	3 segundos
Trafego de Internet	75 KB/s	10 KB/s

Fonte – Adaptado de Krishnan, Bhagwat e Uptat (2015).

3 TRABALHOS RELACIONADOS

Este capitulo irá apresentar trabalhos que possuem relação com a solução proposta. Os mesmos foram selecionados por proporem soluções a problemas similares ao do presente trabalhou ou por contribuírem diretamente com a solução proposta de alguma forma. As próximas seções apresentam os trabalhos selecionados ordenados por ano de publicação de forma decrescente e depois ordem alfabética.

3.1 An Industry 4.0-Enabled Low Cost Predictive Maintenance Approach for SMEs (SEZER et al., 2018)

Sezer et al. (2018) realizaram um estudo de caso buscando identificar todos os componentes que seriam necessários para desenvolver um processo de PdM de baixo custo orientado aos conceitos da Indústria 4.0. Com base na sua pesquisa foram identificados os seguintes conceitos como sendo chave para um processo alinhado com as necessidades da Indústria 4.0: I Internet das Coisas Industriais (*Industrial Internet of* Things – IIoT); II Sistemas Ciberfísicos (*Cyber-Physical Systems* – CPS); III Computação em Nuvem (*Cloud Computing*); IV Aprendizado de Máquina (*Machine Learning*) e V Manutenção Baseada em Condição (*Condition-Based Maintenance* – CBM).

Para atender a esses princípios é proposto um modelo de PdM de baixo custo, voltado para industrias de pequeno porte. Para realizar coleta de dados foi utilizado um Raspberry Pi 3 Modelo B em conjunto com a placa Sense-HAT, que possui diversos sensores. As técnicas utilizadas para realizar a predição do

problema foram Particionamento Recursivo (*Recursive Partitioning*) e Árvores de Regressão (*Regression Trees*).

3.2 Vibration Analysis for IoT Enabled Predictive Maintenance (JUNG; ZHANG; WINSLETT, 2017)

Jung et al. (2017) propõe um modelo para realização de análise de dados de vibração coletados através de sensores de IoT acoplados a bombas de vácuo. O trabalho não detalha qual exatamente é o tipo de sensor utilizado, deixando claro apenas se trata de um sensor do tipo MEMS. A arquitetura proposta busca aumentar o tempo de vida dos sensores através do uso de um escalonador controlado por um servidor, de forma que os sensores possam atuar em dois modos. No modo ativo o sensor realiza as medições e após realizar o envio dos dados ele entra no modo inativo, reduzindo o seu consumo de energia. Para implementar esse acionamento remoto dos sensores é proposto dois ciclos de comunicação entre o servidor e o sensor. No ciclo de operação o sensor de fato realiza comunicação dos dados coletados enquanto o ciclo de prova de vida apensa serve para que o sensor comunique para o servidor que ainda está operacional.

O algoritmo proposto no modelo primeiramente busca classificar cada máquina dentro de categorias previamente definidas, baseado na saúde das mesmas e posteriormente projetar quando uma máquina irá entrar em uma zona de risco. Essa classificação é realizada através da técnica de Distância Entre Picos Harmônicos (*Peak Harmonic Distance*).

O modelo foi testado durante alguns meses utilizando 12 bombas idênticas cuja coleta de dados ocorreu a cada 10 minutos. Segundo o autor, o modelo permitiu um melhor aproveitamento do maquinário, sendo possível verificar uma redução de 20% no custo de troca de peças.

3.3 Fog Computing Based Efficient IoT Scheme for the Industry 4.0 (PERALTA et al., 2017)

Peralta et al. (2017) apresenta as vantagens da utilização de *Fog Computing* e propõe uma arquitetura considerando esse conceito para utilização de sensores IoT na indústria. A arquitetura proposta é constituída por 3 camadas: a primeira é a

Camada de Dispositivos IoT, a segunda é a Camada de *Fog* e a terceira Camada a de *Cloud*. A Camada de Dispositivos IoT representa os clientes MQTT (*Message Queueing Telemetry Transport*), que são os responsáveis pela coleta de dados e envio dos mesmos para as camadas superiores. A Camada de *Fog* é responsável por gerenciar as mensagens recebidas da camada anterior, representando o MQTT *broker.* A segunda camada é constituída por dispositivos simples, mas com a capacidade de realizar algumas operações básicas, transferindo para a *cloud* a execução e operações mais complexas. No modelo proposto a segunda camada também deve ser capaz de executar algoritmos de predição que são utilizados para reduzir o consumo de energia dos sensores da camada anterior. A Camada de *Cloud* representa os servidores e é responsável por executar as tarefas que são computacionalmente muito pesadas para serem executadas na Camada de *Fog*.

Segundo o autor, a introdução da Camada de *Fog* permitiu baixar consideravelmente a latência da comunicação. No trabalho é apresentado que comunicação direta entre os sensores com serviços AWS ocorria com latência variando entre 800ms e 2,5s, enquanto a latência utilizando um Mosquitto *broker* na *fog* estava em torno 200ms.

3.4 An Intelligent Door System using Raspberry Pi and Amazon Web Services IoT (BASHA; JILANI; ARUN, 2016)

Basha, Jilani e Arun (2016) desenvolveram um sistema para identificar e notificar quando algum intruso abrir uma porta que deveria estar fechada. Para desenvolver o sensor os autores utilizaram um Raspberry Pi modelo B+ combinado com um acelerômetro ADXL 345. A programação foi feita usando a linguagem Phyton.

Uma vez que o sensor identifica que a porta foi aberta, o mesmo realiza a comunicação com o serviço da Amazon AWS IoT, utilizando o protocolo MQTT para se autenticar e trocar mensagens. O AWS IoT por sua vez avisa o proprietário através do seu modulo de notificação SNS (*Simple Notification Service*).

Juntamente com a notificação o modelo construído também utiliza OAuth2 para realizar a autenticação no Google *Spreadsheet* e registrar os dados referentes a intrusão em uma planilha online.

3.5 Análise

Durante a realização da pesquisa diversos trabalhos nas áreas de IoT, *Cloud Computing*, *Fog Computing* e Indústria 4.0 foram encontrados, sendo selecionados trabalhos que possuem características similares ao presente trabalho ou que abordem com maior profundidade ao menos um tema que integrou a arquitetura desenvolvida. A Tabela 2 apresenta uma comparação entre características dos artigos que foram selecionados:

- Indústria 4.0: identifica se o trabalho em questão está relacionado ao contexto da Indústria 4.0;
- Analise de Consumo de Energia: identifica se no trabalho em questão foi realizado alguma análise do consumo de energia de sensores;
- *Fog Computing*: identifica se o trabalho em questão implemento uma Camada de *Fog*;
- *Cloud Computing*: identifica se o trabalho em questão implementou o uso de uma Camada de *Cloud.*

Tabela 2 – Trabalhos relacionados

Trabalho	**Indústria 4.0**	**Análise do Consumo de Energia**	***Fog Computing***	***Cloud Computing***
Sezer et al. (2018)	X			X
Jung, Zhang e Winslett (2017)	X	X		X
Peralta et al. (2017)	X	X	X	X
Basha. Jilani e Arun (2016)				X
Solução proposta	X	X	X	X

Fonte – Elaborado pelo autor.

Existem muitas pesquisas na área da Indústria 4.0, muitas delas utilizando IoT ou IA para melhorar processos de produção. No entanto, não foram identificados trabalhos que propusessem o desenvolvimento de sensores capazes de serem reconfigurados em tempo de produção, permitindo um melhor aproveitamento dos mesmos, o que foi visto como uma oportunidade para ser explorada no presente trabalho.

4 METODOLOGIA

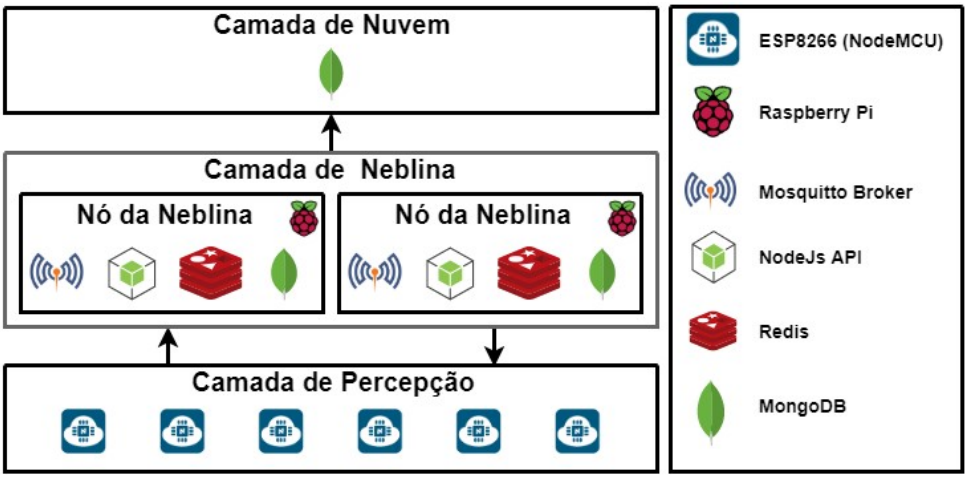
Este capitulo irá detalhar a solução proposta. Será abordado o planejamento de etapas em que o trabalho foi divido, detalhando o que foi feito em cada uma.

O desenvolvimento da solução foi dividido em 4 etapas. Na primeira etapa foi realizado o desenvolvimento dos dispositivos para medição de vibração em si, utilizando o microcontrolador ESP8266, juntamente com um módulo de acelerômetro MPU6050. Na sequência, foi realizado o desenvolvimento das camadas de *fog* e *cloud* responsáveis por armazenar os dados coletados pelos dispositivos e hospedar os serviços para configuração do escalonamento dos mesmos. Posteriormente, foi implementado o algoritmo de compressão de dados *Delta Encoding*, com o objetivo de diminuir o consumo de energia. Após ter finalizado o desenvolvimento da arquitetura proposta, foram analisados os resultados obtidos, referente ao consumo de energia e pontos coletados pelos sensores, sendo feita uma comparação entre os dados obtidos usando e não usando o algoritmo de compressão de dados e com diferentes variações nos parâmetros de configuração dos dispositivos.

4.1 Arquitetura do modelo proposto

A arquitetura do modelo proposto neste trabalho é apresentada na Figura 1, sendo composta por 3 camadas: I Camada de Percepção, II Camada de *Fog* e III Camada de *Cloud*. A Camada de Percepção é constituída pela rede de dispositivos IoT e é responsável pela coleta dos dados dos equipamentos monitorados. A Camada de *Fog* é situada entre a Camada de Percepção e a Camada de *Cloud,* sendo responsável pela gerência do processo de medição, assim como reduzir o tráfego de dados para a *cloud*. Por fim, a Camada de *Cloud* recebe os dados de medição enviados pela Camada de *Fog,* e armazena os mesmos, para que em trabalhos futuros estes dados possam ser utilizados para o desenvolvimento de algoritmos baseados em aprendizagem de máquina, para serem aplicados à PdM.

Figura 1 – Arquitetura da solução proposta



Fonte – Elaborado pelo autor.

4.2 Implementação dos sensores de vibração de IoT

O desenvolvimento dos dispositivos IoT foi feito utilizando o microcontrolador ESP8266, pois o mesmo possui um custo baixo e um módulo Wi-Fi para comunicação sem fio (RODRIGUES; CASTRO, 2018). Além dessas características, conforme Polianytsia e Herasymenko (2016), o ESP8266 é o microcontrolador adequado para aplicações simples, que utilizam poucos recursos, que é o caso dos dispositivos que foram construídos. Para realizar a medição do espectro de vibração da máquina foi utilizado o módulo de acelerômetro MPU6050.

Visando a economia de energia, os dispositivos são programados para possuírem dois modos de operação: ativo e passivo. No modo passivo, o microcontrolador ESP8266 é colocado em modo *Deep-sleep*, no qual consome o mínimo de energia possível. Da mesma forma, o MPU6050 é colocado em modo de baixo consumo, através da escrita do bit SLEEP no seu registrador PWR_MGMT_1. A Tabela 3 ilustra a comparação entre os modos de operação de baixo consumo de energia do ESP8266, enquanto a Tabela 4 demonstra o consumo do MPU6050 quando colocado no seu modo de baixo consumo.

Tabela 3 – Modos de operação do ESP8266

Item	Modem-sleep	Light-sleep	Deep-sleep
Wi-Fi	Desligado	Desligado	Desligado
Clock do sistema	Ligado	Desligado	Desligado
RTC	Ligado	Ligado	Ligado
CPU	Ligado	Pendente	Desligado
Consumo de Energia	15 mA	0.4 mA	~20 µA

Fonte – Adaptado de Expressif (2015).

Tabela 4 – Consumo de energia do MPU6050

Modo de operação	Consumo de energia
Normal	500 µA
Baixo consumo – frequência em 1.25Hz	10 µA
Baixo consumo – frequência em 5Hz	20 µA
Baixo consumo – frequência em 20Hz	60 µA
Baixo consumo – frequência em 40Hz	110 µA

Fonte – Adaptado de Ivensense (2013).

No modo ativo, os dispositivos de fato realizam as medições de vibração por um período de tempo, e enviam os dados para a Camada de *Fog*. A comunicação com a Camada de *Fog* é realizada através do protocolo MQTT, pois conforme Peralta et al. (2017), esse protocolo oferece uma alta escalabilidade com um baixo consumo de banda, sendo bastante adequado para implementação do uso de sensores inteligentes na Industria 4.0. Dessa forma os dispositivos serão MQTT *clients* e os nodos da Camada de *Fog* MQTT *brokers.*

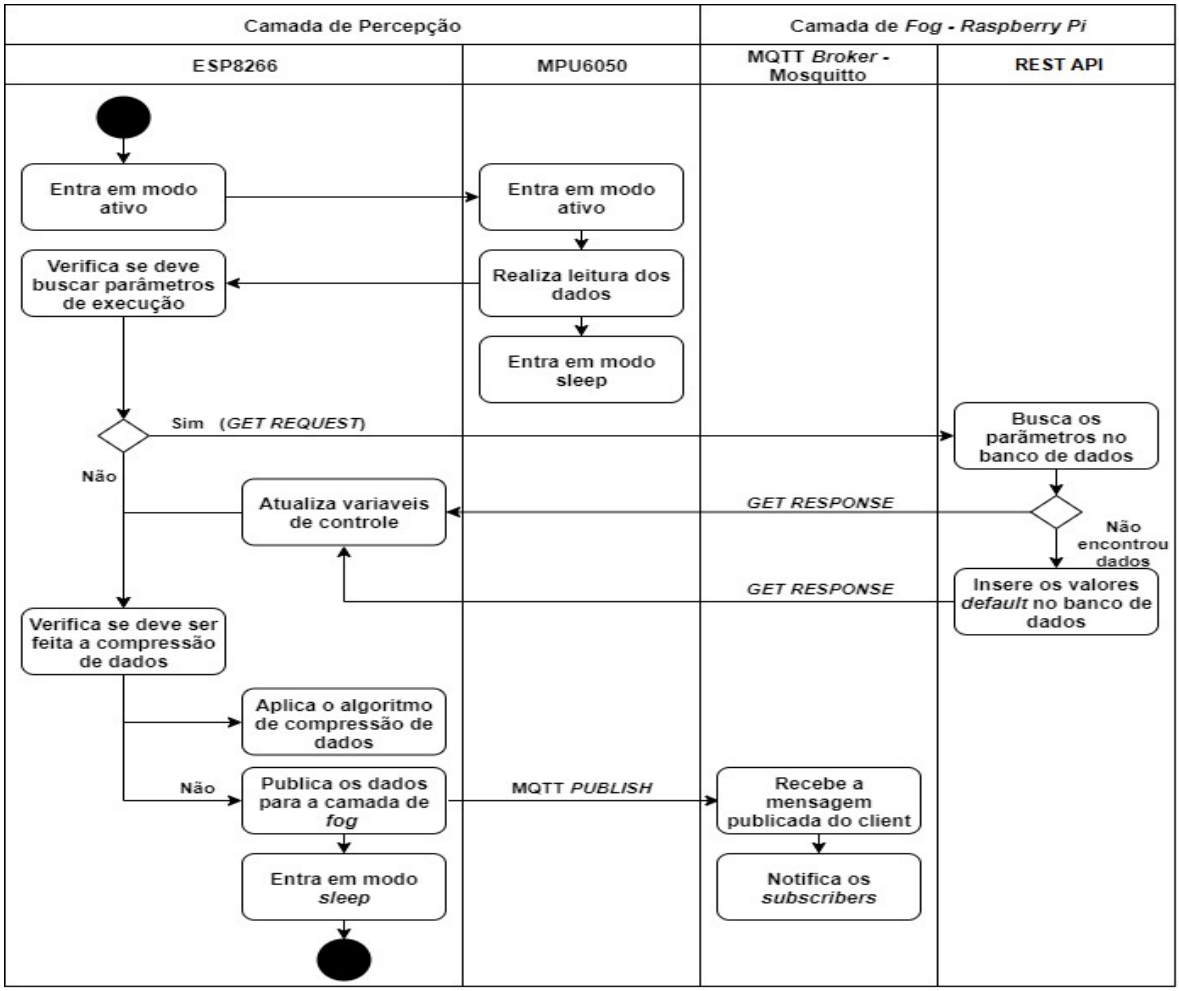
Os dispositivos são programados de forma que o seu funcionamento seja configurável, permitindo que, em trabalhos futuros algoritmos de IA possam inferir na operação de cada um, permitindo que a arquitetura como um todo se torne mais eficiente. Para que possam ser configuráveis, em determinados ciclos, os dispositivos solicitam à Camada de *Fog* os seus parâmetros de operação. Essa requisição é feita através de uma chamada REST (*Representational State Transfer*) para um API (*Application Programming Interface)* hospedada no nodo de *fog* que atende o mesmo. Se a Camada de *Fog* não encontrar parâmetros para o dispositivo em questão, então o mesmo será registrado com uma parametrização padrão. Os parâmetros que podem ser configuráveis para cada dispositivo são:

- *sensorDataSleepTime*: define o tempo em que o dispositivo irá ficar em modo *Deep-sleep*;
- *maxReadsBeforeReeschedule*: define a quantidade de vezes em que o código do dispositivo irá executar antes de tentar buscar novamente a sua parametrização;
- *sensorReadDuration*: define por quanto tempo a leitura do acelerômetro irá ser realizada;
- *compressionAlgorithm*: define se será utilizado algum algoritmo de compressão de dados antes do envio dos dados para *fog.* No presente trabalho foi implementado o algoritmo *Delta Encoding*;
- *discardedPoints*: define o percentual de leituras que serão descartadas antes da realização do envio dos dados;

A escolha pelo algoritmo *Delta Encoding* foi realizada, pois conforme o trabalho de Arrabi e Lach (2009), esse algoritmo, assim como o *Huffman Encoding,* pode ser executado em sensores, por necessitarem de um baixo poder computacional. Durante o desenvolvimento do trabalho, também foi realizado o desenvolvimento do algoritmo *Huffman Encoding*, visando aplicar uma combinação de ambos algoritmos, conforme proposto por Marcelloni e Vecchio (2008). No entanto, a execução do algoritmo *Huffman Encoding* apresentou problemas referentes ao consumo de memória do ESP8266, que inviabilizaram a utilização do mesmo para a coleta dos resultados.

O acelerômetro MPU6050 permite a medição de temperatura e de giroscópio. No entanto, neste trabalho apenas foram utilizados os dados dos seus acelerômetros (eixos X, Y e Z). A mensagem MQTT enviada pelo dispositivo é composta pelos dados referentes aos 3 eixos, juntamente com um identificador único do sensor (*macAddress* do chip) e com um identificador do algoritmo de compressão aplicado ainda no microcontrolador. A Figura 2 apresenta a lógica do algoritmo que é executado nos dispositivos. Como tamanho máximo de uma mensagem MQTT suportada pela biblioteca utilizada é de 128 bytes, para mensagens maiores do que o tamanho máximo foi necessário realizar a divisão das mesmas e mais de uma mensagem.

Figura 2 – Execução de um ciclo do ESP8266



Fonte – Elaborado pelo autor.

De forma geral, o funcionamento dos dispositivos da Camada de Percepção pode ser descrito conforme os seguintes passos: I Realizar a leitura dos dados do acelerômetro; II Verificar se deve buscar seus parâmetros de configuração. Se deve buscar, então realizar a requisição dos mesmos para a API da Camada de *Fog*; III Atualizar as variáveis de controle com a resposta que recebeu; IV Verificar se deve aplicar algum algoritmo de compressão de dados. Se deve realizar a compressão dos dados, então aplica o algoritmo Delta Encoding; V Publicar os dados para o MQTT *broker* hospedado na Camada de *Fog* e VI Entrar modo passivo após o envio dos dados.

4.3 Implementação da Camada de *Fog*

A Camada de *Fog* implementada atua entre a Camada de Percepção e Camada de *Cloud,* e serve para diminuir tanto o volume de dados trafegados para a *cloud*, quanto a latência na comunicação com os sensores. Para a criação da Camada de *Fog* foi utilizado um Raspberry Pi, pois em comparação com o ESP8266 o Raspberry Pi possui mais recursos operacionais. A opção por um hardware mais potente, visa atender o objetivo da arquitetura de manter o custo baixo. Para isso, um nodo da Camada de *Fog* precisa atender a vários dispositivos da Camada de Percepção.

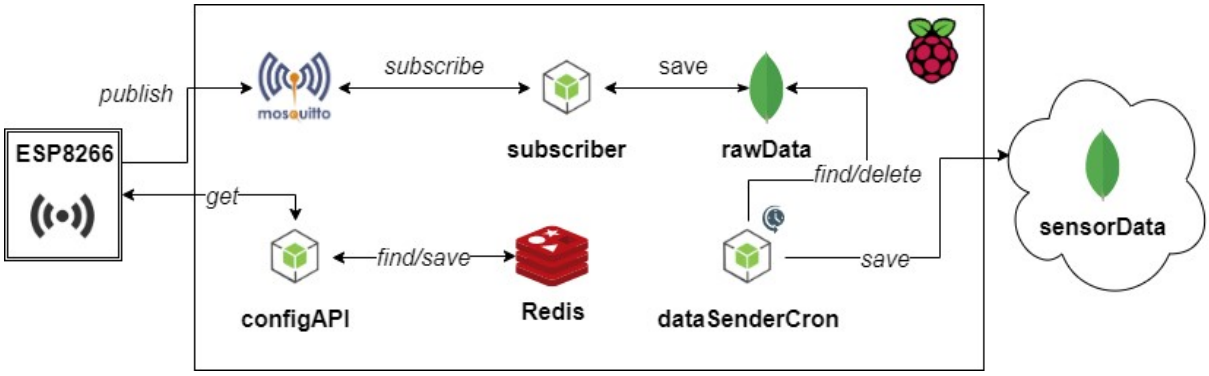
A escolha do Raspberry Pi foi feita, porque durante a pesquisa foram identificados trabalhos relacionados como o realizado por Bellavista e Zanin (2017) e o de Constant et al. (2017), que utilizaram o Raspberry Pi como uma solução para a implementação da Camada de *Fog*.

Esta camada é responsável por executar as seguintes ações:

1. Funcionar como um MQTT *broker,* sendo responsável por processar as mensagens enviadas pelos sensores da Camada de Percepção. Para isso, cada nodo da Camada de *Fog* hospeda uma instância do *broker* Mosquitto;
2. Hospedar um *subscriber* para capturar as mensagens publicadas pelos sensores. Sempre que um sensor publica uma mensagem no *broker* o *subscriber* recebe a mesma, realiza o *parse* da mesma e armazena os dados de vibração dos sensores em uma instância de banco de dados MongoDB, a qual também é executada localmente no nodo da *fog*;
3. Hospedar a API REST de onde os sensores buscam os seus parâmetros de operação. O armazenamento dos parâmetros de operação está sendo feito através do banco de dados em memória Redis;
4. Realizar o envio dos dados de vibração de sensores já processados para a Camada de *Cloud*. Para isso cada nodo possui uma *cron,* que busca os dados da coleção do MongoDB que armazena os dados já processados. Após o envio dos dados para a *cloud* os dados são eliminados de ambas coleções no MongoDB.

   A Figura 3 apresenta a arquitetura de um nodo da Camada de *Fog.*

Figura 3 – Arquitetura de um nodo da Camada de *Fog*



Fonte – Elaborado pelo autor.

4.4 Implementação da Camada de *Cloud*

A Camada de *Cloud* foi prevista na arquitetura para prover o armazenamento e disponibilização dos dados coletados pelos sensores. Neste trabalho apenas foi realizado a implementação do armazenamento dos dados na *cloud*. Para isso foi utilizado um MongoDB Atlas, o qual é alimentado pela Camada de *Fog*. O restante da Camada de *Cloud* não foi implementado, pois não era necessário para a análise dos resultados, ficando com um ponto em aberto para trabalhos futuros.

5 RESULTADOS

Após a implementação da arquitetura, foram realizados testes com cenários diferentes para verificar a comparação entre os dados coletados com a energia consumida para realização da coleta. A análise da energia consumida foi feita através da medição da tensão emitida por uma bateria LiPo (*Lithium Polymer*) de 3.7V, que foi utilizada para energizar o sensor. Cada cenário foi executado por 30 minutos e após a execução de cada um foi medido a tensão da bateria com o uso de um multímetro. Os cenários testados foram os seguintes: I Envio dos dados do acelerômetro sem utilizar compressão de dados e realizando a transmissão diretamente para a *cloud*; II Envio dos dados do acelerômetro utilizando o algoritmo *Delta Encoding* e realizando a transmissão diretamente para a *cloud*; III Envio dos dados do acelerômetro sem utilizar compressão de dados e realizando a transmissão para a Camada de *Fog* e IV Envio dos dados do acelerômetro utilizando o algoritmo *Delta Encoding* e realizando a transmissão para a Camada de *Fog*;

Cada um dos cenários foi testado com 4 variações dos parâmetros *sensorDataSleepTime, sensorReadDuration* e *discardedPoints.* Também foi feita uma análise dos pontos coletados para verificar se o espectro de vibração se manteve, mesmo alterando os parâmetros de operação do dispositivo. Essa análise foi feita com os dados eixo X do Cenário I para cada uma das variações de parâmetro.

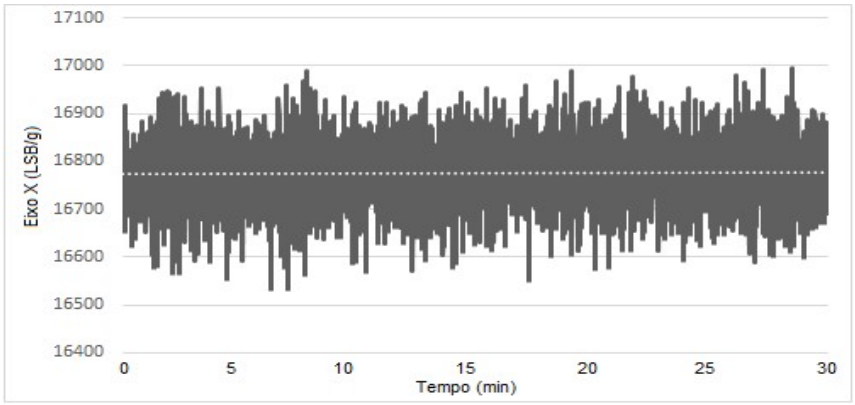
A primeira variação de parâmetros testada foi a seguinte: *sensorDataSleepTime* = 30 segundos, *sensorReadDuration* = 2 segundos; *discardedPoints* = 0%. Desta forma, o sensor estava lendo os dados do acelerômetro durante 2 segundos e transmitindo os mesmos sem descartar nenhuma informação. Após o término de cada operação, o sensor entrava no modo *Deep-sleep* por 30 segundos antes de começar um novo ciclo de medição novamente. A Tabela 5 demonstra a os resultados obtidos do ponto de vista de consumo de energia para cada um dos cenários testados e o Gráfico 1 apresenta os pontos do eixo X que foram coletados durante todo os 30 minutos em que o Cenário I foi executado.

Tabela 5 – Consumo de energia para a combinação de parâmetros I

Cenário	**Pontos coletados**	**Compressão**	***Fog Computing***	**Consumo total**	**Consumo por ciclo**
I	3956	Não	Não	≅ 300 mA	≅ 13,04 mA
II	3956	*Delta Encoding*	Não	≅ 200 mA	≅ 8,70 mA
III	4300	Não	Sim	≅ 300 mA	≅ 12 mA
IV	4300	*Delta Encoding*	Sim	≅ 200 mA	≅ 8 mA

Fonte – Elaborado pelo autor.

Gráfico 1 – Pontos do eixo X coletados para a combinação de parâmetros I



Fonte – Elaborado pelo autor.

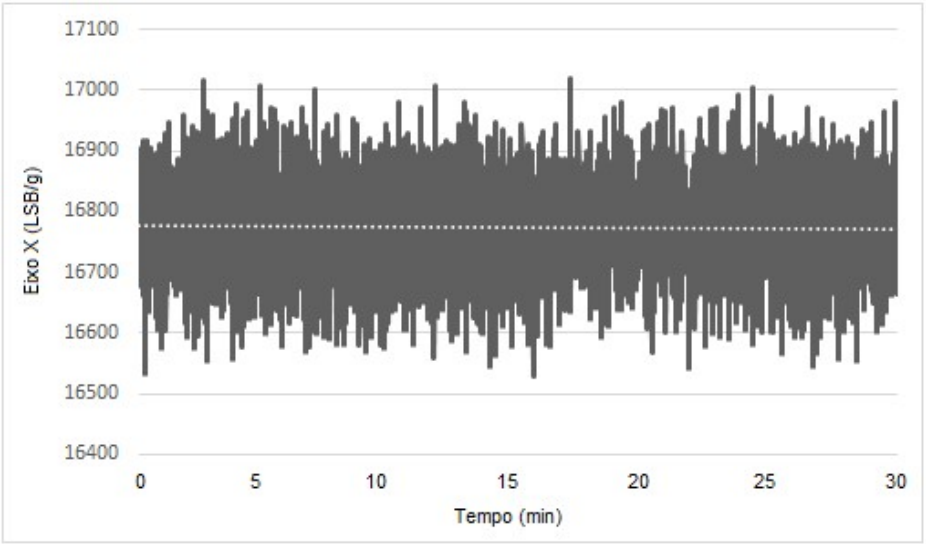
A segunda variação de parâmetros testada foi a seguinte: *sensorDataSleepTime* = 60 segundos, *sensorReadDuration* = 4 segundos; *discardedPoints* = 33%. Desta forma, o sensor estava lendo os dados do acelerômetro durante 4 segundos, descartando 1 ponto a cada 3 pontos coletados. Após o término de cada operação o sensor entrava no modo *Deep-sleep* por 60 segundos antes de começar um novo ciclo de medição. A Tabela 6 demonstra a os resultados obtidos do ponto de vista de consumo de energia para cada um dos cenários testados e o Gráfico 2 apresenta os pontos do eixo X que foram coletados durante todo os 30 minutos em que o Cenário I foi executado.

Tabela 6 – Consumo de energia para a combinação de parâmetros II

Cenário	**Pontos coletados**	**Compressão**	***Fog Computing***	**Consumo total**	**Consumo por ciclo**
I	3447	Não	Não	≅ 200 mA	≅ 13,33 mA
II	3272	*Delta Encoding*	Não	≅ 100 mA	≅ 7,69 mA
III	3018	Não	Sim	≅ 100 mA	≅ 8,33 mA
IV	3209	*Delta Encoding*	Sim	≅ 100 mA	≅ 7,69 mA

Fonte – Elaborado pelo autor.

Gráfico 2 – Pontos do eixo X coletados para a combinação de parâmetros II



Fonte – Elaborado pelo autor.

A terceira variação de parâmetros testada foi: *sensorDataSleepTime* = 90 segundos, *sensorReadDuration* = 6 segundos; *discardedPoints* = 50%. Desta forma o sensor estava lendo os dados do acelerômetro durante 6 segundos, descartando 1

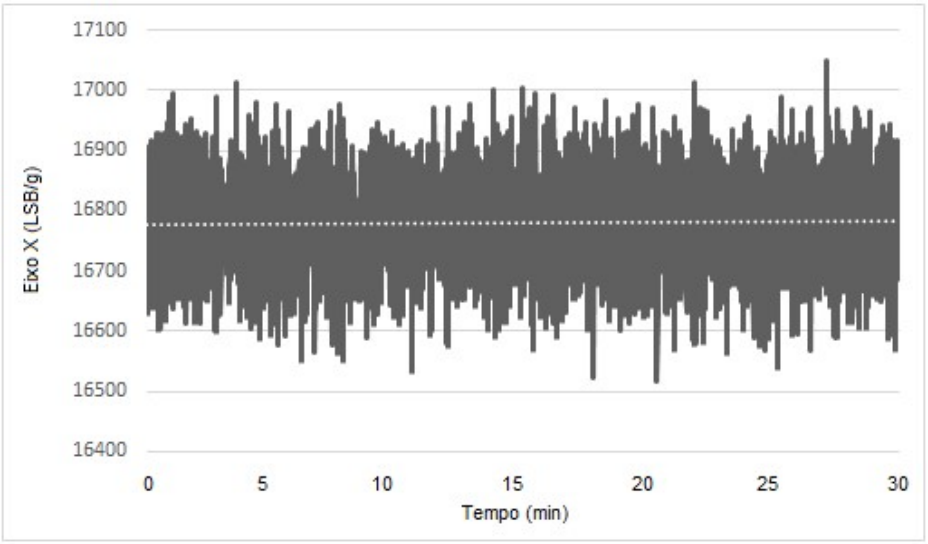
ponto para cada 2 pontos coletados. Após o término de cada operação o sensor entrava no modo *Deep-sleep* por 90 segundos antes de começar um novo ciclo de medição. A Tabela 7 demonstra a os resultados obtidos do ponto de vista de consumo de energia para cada um dos cenários testados e o Gráfico 3 apresenta os pontos do eixo X que foram coletados durante todo os 30 minutos em que o Cenário I foi executado.

Tabela 7 – Consumo de energia para a combinação de parâmetros III

Cenário	**Pontos coletados**	**Compressão**	***Fog Computing***	**Consumo total**	**Consumo por ciclo**
I	3144	Não	Não	≅ 100 mA	≅ 10 mA
II	2886	*Delta Encoding*	Não	≅ 100 mA	≅ 11,11 mA
III	3486	Não	Sim	≅ 150 mA	≅ 15 mA
IV	3062	*Delta Encoding*	Sim	≅ 150 mA	≅ 16,67 mA

Fonte – Elaborado pelo autor.

Gráfico 3 – Pontos do eixo X coletados para a combinação de parâmetros III



Fonte – Elaborado pelo autor.

A última variação de parâmetros testada foi: *sensorDataSleepTime* = 10 segundos, *sensorReadDuration* = 1 segundo; *discardedPoints* = 50%. Desta forma o sensor estava lendo os dados do acelerômetro durante 1 segundo, descartando 1 ponto para cada 2 pontos coletados. Após o término de cada operação o sensor entrava no modo *Deep-sleep* por 10 segundos antes de começar um novo ciclo de medição. A Tabela 8 demonstra a os resultados obtidos do ponto de vista de

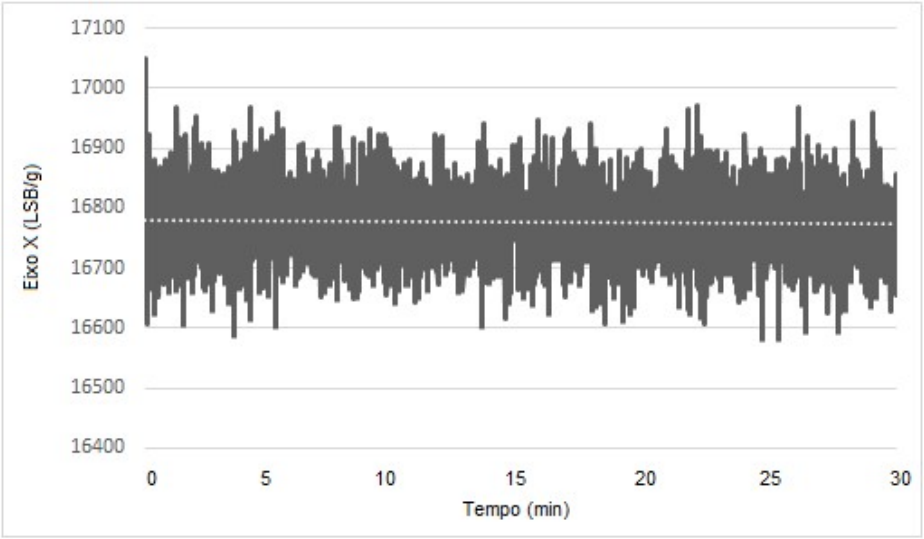
consumo de energia, para cada um dos cenários testados e o Gráfico 4 apresenta os pontos do eixo X que foram coletados durante todo os 30 minutos em que o Cenário I foi executado.

Tabela 8 – Consumo de energia para a combinação de parâmetros IV

Cenário	**Pontos coletados**	**Compressão**	***Fog Computing***	**Consumo total**	**Consumo por ciclo**
I	3420	Não	Não	≅ 150 mA	≅ 2,63 mA
II	2520	*Delta Encoding*	Não	≅ 150 mA	≅ 3,57 mA
III	3538	Não	Sim	≅ 200 mA	≅ 3,45 mA
IV	1980	*Delta Encoding*	Sim	≅ 200 mA	≅ 6,06 mA

Fonte – Elaborado pelo autor.

Gráfico 4 – Pontos do eixo X coletados para a combinação de parâmetros IV

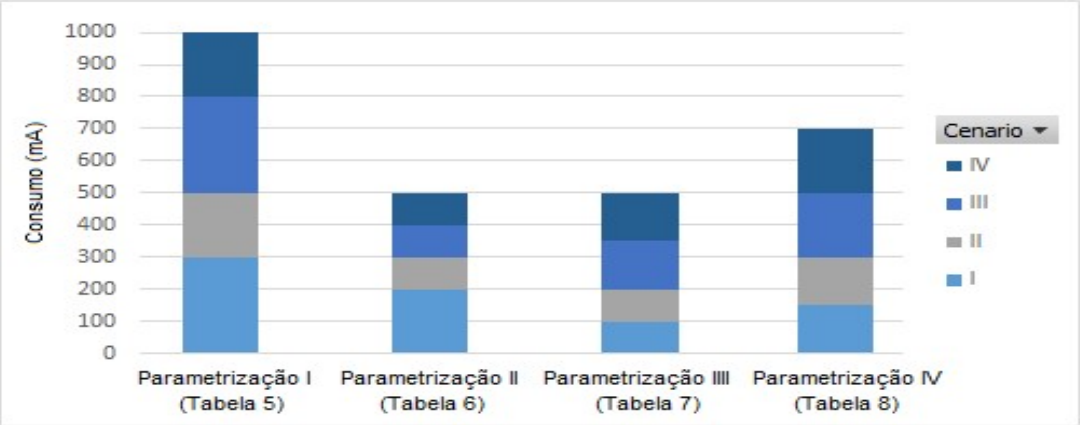


Fonte – Elaborado pelo autor.

Os testes realizados não demonstraram que a utilização da Camada de *Fog* teve um impacto significativo na redução do consumo de energia do sensor. Também não foi possível verificar que a utilização do algoritmo *Delta Encoding* resultou em um menor consumo de dados. Os testes, no entanto, demonstraram que o volume de dados a ser transmitido a cada medição aumentou sim o consumo de energia, o que pode ser verificado ao comparar os valores obtidos na execução com a primeira configuração de parâmetros, exibidos na Tabela 5, onde não estava sendo descartado nenhuma informação com as execuções realizadas em cenários

onde estavam sendo descartados dados antes do envio, tabelas 6, 7 e 8, como pode ser verificado no Gráfico 5.

Gráfico 5 – Comparação do consumo de energia entre as combinações de parâmetros e cenários testados



Fonte – Elaborado pelo autor.

Essa constatação, indica, que ter sensores com parâmetros de execução configuráveis pode levar ao aumento do tempo de vida dos mesmos, pois permite que um algoritmo de IA possa inferir na programação desses conforme identificar a necessidade de coletar mais ou menos dados para determinar o estado de uma máquina.

Por fim, a Tabela 9 demonstra a comparação entre os gráficos que foram construídos com base nos resultados do eixo X do Cenário I de cada uma das 4 diferentes configurações de parâmetros que foram testados.

Tabela 9 – Comparação entre os gráficos apresentados

Cenário	**Maior valor (LSB/g)**	**Menor valor (LSB/g)**	**Média (LSB/g)**	**Desvio Padrão**
Gráfico 1	16996	16536	16676,16	66,67
Gráfico 2	17020	16528	16674,39	82,80
Gráfico 3	17048	16516	16780,06	84,56
Gráfico 4	17048	16584	16777,72	58,47

Fonte – Elaborado pelo autor.

6 CONCLUSÃO

O presente trabalho propôs uma arquitetura de baixo custo para a coleta de dados de vibração de equipamentos. A arquitetura proposta contempla a construção de sensores não invasivos e a implementação de uma Camada de *Fog*, para buscar maximizar a vida útil dos sensores.

Através dos testes que foram realizados não foi possível verificar que a implementação de uma Camada de *Fog* teve um impacto significativo no consumo de energia da bateria do sensor. Resultado que era esperado conforme os trabalhos encontrados no levantamento do referencial teórico. O fato de não ter sido constatado essa melhora que era esperada nos resultados, não descarta o uso de uma Camada de *Fog* em uma arquitetura para medição de dados, visto que a mesma pode trazer benefícios para outros fins, como redução do tráfego de dados para a *cloud*. Também não foi constatado que a utilização do algoritmo de compressão de dados escolhido para ser aplicado no processamento feito no sensor, antes da transmissão dos dados para *fog*, tenha resultados significativos no consumo da bateria. Entretanto pode ser constatado que o volume de dados enviados pelo sensor possui uma relação com a sua vida útil. Desta forma, ter a possibilidade de poder determinar quando uma medição com maior precisão é necessária para identificação de problemas é uma característica desejável em um sistema de PdM baseado em dados coletados com sensores.

O trabalho realizado possui diversas oportunidades para evolução em trabalhos futuros, podendo ser destacado a implementação de algoritmos de compressão de dados também na Camada de *Fog* para reduzir ainda mais os dados que serão enviados para a Camada de C*loud* e a implementação de um algoritmo de IA, que consuma os dados coletados por esta arquitetura, sendo capaz de contribuir no processo de PdM e retroalimentar os parâmetros de operação dos sensores, tornando assim a arquitetura mais eficiente.

REFERÊNCIAS

ARRABI, Saad; LACH, John. Adaptive lossless compression in wireless body sensor networks. In: **Proceedings of the Fourth International Conference on Body Area Networks**. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2009. p. 19.

ATZORI, Luigi; IERA, Antonio; MORABITO, Giacomo. The internet of things: A survey. **Computer networks**, v. 54, n. 15, p. 2787-2805, 2010.

BASHA, S. Nazeem; JILANI, Dr SAK; ARUN, Mr S. An Intelligent Door System using Raspberry Pi and Amazon Web Services IoT. **International Journal of Engineering Trends and Technology (IJETT)**, v. 33, n. 2, 2016.

BATES, Martin. **PIC microcontrollers: an introduction to microelectronics**. Elsevier, 2011.

BELLAVISTA, Paolo; ZANNI, Alessandro. Feasibility of fog computing deployment based on docker containerization over raspberrypi. In: **Proceedings of the 18th international conference on distributed computing and networking**. ACM, 2017. p. 16.

BONOMI, Flavio et al. Fog computing and its role in the internet of things. In: **Proceedings of the first edition of the MCC workshop on Mobile cloud computing**. ACM, 2012. p. 13-16.

CIVERCHIA, Federico et al. Industrial Internet of Things monitoring solution for advanced predictive maintenance applications. **Journal of Industrial Information Integration**, v. 7, p. 4-12, 2017.

CONSTANT, Nicholas et al. Fog-assisted wiot: A smart fog gateway for end-to-end analytics in wearable internet of things. **arXiv preprint arXiv:1701.08680**, 2017.

DA XU, Li; HE, Wu; LI, Shancang. Internet of things in industries: A survey. **IEEE Transactions on industrial informatics**, v. 10, n. 4, p. 2233-2243, 2014.

DRATH, Rainer; HORCH, Alexander. Industrie 4.0: Hit or hype?[industry forum]. **IEEE industrial electronics magazine**, v. 8, n. 2, p. 56-58, 2014.

"ESP8266EX Datasheet", *Espr. Syst. Datasheet*, pp. 1-31, 2015.

FISCHER, Gabriel Souto. Elhealth: utilizando internet das coisas e predição computacional para gerenciamento elástico de recursos humanos em hospitais inteligentes. 2019.

HASHEMIAN, Hashem M. State-of-the-art predictive maintenance techniques. **IEEE Transactions on Instrumentation and measurement**, v. 60, n. 1, p. 226-236, 2010.

INVENSENSE, INC. DATASHEET MPU6050, Revision 3.4, 2013.

JUNG, Deokwoo; ZHANG, Zhenjie; WINSLETT, Marianne. Vibration analysis for iot enabled predictive maintenance. In: **2017 IEEE 33rd International Conference on Data Engineering (ICDE)**. IEEE, 2017. p. 1271-1282.

KRISHNAN, Y. Navaneeth; BHAGWAT, Chandan N.; UTPAT, Aparajit P. Fog computing—Network based cloud computing. In: **2015 2nd International Conference on Electronics and Communication Systems (ICECS)**. IEEE, 2015. p. 250-251.

MARCELLONI, Francesco; VECCHIO, Massimo. A simple algorithm for data compression in wireless sensor networks. **IEEE communications letters**, v. 12, n. 6, p. 411-413, 2008.

MUNIR, Arslan; KANSAKAR, Prasanna; KHAN, Samee U. IFCIoT: Integrated Fog Cloud IoT: A novel architectural paradigm for the future Internet of Things. **IEEE Consumer Electronics Magazine**, v. 6, n. 3, p. 74-82, 2017.

PERALTA, Goiuri et al. Fog computing based efficient IoT scheme for the Industry 4.0. In: **2017 IEEE International Workshop of Electronics, Control, Measurement, Signals and their Application to Mechatronics (ECMSM)**. IEEE, 2017. p. 1-6.

PFLANZNER, Tamás; KERTÉSZ, Attila. A survey of IoT cloud providers. In: **2016 39th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)**. IEEE, 2016. p. 730-735.

POLIANYTSIA, Andrii; STARKOVA, Olena; HERASYMENKO, Kostiantyn. Survey of hardware IoT platforms. In: **2016 Third International Scientific-Practical Conference Problems of Infocommunications Science and Technology (PIC S&T)**. IEEE, 2016. p. 152-153.

PULIAFITO, Carlo et al. Fog computing for the internet of things: A Survey. **ACM Transactions on Internet Technology (TOIT)**, v. 19, n. 2, p. 18, 2019.

RODRIGUES, Carlos; CASTRO, Bruno. A vision of Internet of Things in Industry 4.0 with ESP8266. **International Journal of Eletronics and Communication Engineering and Technology**, v. 9, p.1-12, 2018.

SARHAN, Qusay Idrees. Internet of things: a survey of challenges and issues. **International Journal of Internet of Things and Cyber-Assurance**, v. 1, n. 1, p. 40-75, 2018.

SELCUK, Sule. Predictive maintenance, its implementation and latest trends. **Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture**, v. 231, n. 9, p. 1670-1679, 2017.

SEZER, Erim et al. An industry 4.0-enabled low cost predictive maintenance approach for smes. In: **2018 IEEE International Conference on Engineering, Technology and Innovation (ICE/ITMC)**. IEEE, 2018. p. 1-8.

SINGH, Kiran Jot; KAPOOR, Divneet Singh. Create Your Own Internet of Things: A survey of IoT platforms. **IEEE Consumer Electronics Magazine**, v. 6, n. 2, p. 57-68, 2017.

VOGEL-HEUSER, Birgit; HESS, Dieter. Guest editorial industry 4.0–prerequisites and visions. **IEEE Transactions on Automation Science and Engineering**, v. 13, n. 2, p. 411-413, 2016.

VUJOVIĆ, Vladimir; MAKSIMOVIĆ, Mirjana. Raspberry Pi as a Sensor Web node for home automation. **Computers & Electrical Engineering**, v. 44, p. 153-171, 2015.

XU, Li Da; XU, Eric L.; LI, Ling. Industry 4.0: state of the art and future trends. **International Journal of Production Research**, v. 56, n. 8, p. 2941-2962, 2018.

ZHANG, Qi; CHENG, Lu; BOUTABA, Raouf. Cloud computing: state-of-the-art and research challenges. **Journal of internet services and applications**, v. 1, n. 1, p. 7-18, 2010.