Fatec Cruzeiro Prof. Waldomiro May
Revista Científica H-TEC Humanidades & Tecnologia

INTEGRAÇÃO DE SENSORES DE CARGA E TECNOLOGIA IOT EM ARMAZENS INDUSTRIAIS

Autores
Leonardo Henrique da Silva Paixão [1]
Rafael Ferreira dos Santos[2]
Eugênio Sper de Almeida[3]

Resumo
Com o avanço tecnológico em máquinas e equipamentos industriais, o mercado de trabalho vem mudando cada vez mais para aperfeiçoar, reduzir custo e tempo de manufatura. Para colaborar com essa evolução, utilizando a Tecnologia IoT e sensores de carga, esse presente trabalho visa ajudar na melhoria do processo de logística, mais especificamente na reposição de materiais nas linhas de montagem, apresentando uma alternativa de automação nos armazéns, utilizando sistema de baixo custo, com comunicação realizada via Wi-Fi, com o auxílio do Protocolo MQTT, utilizando o ESP8266, que realiza comunicação wireless e células de carga Half Point para a captura dos dados. Como microcontrolador foi utilizado o NodeMCU ESP8622 12E e para a leitura e captura dos dados é utilizado Células de Carga Meia Ponte, foi necessário também o Módulo HX711 para a integração entre o NodeMCU e as Células de Carga. Obtivemos resultados satisfatórios e o software de automação tem como uma grande vantagem a sua mobilidade e praticidade, podendo ser utilizado como um equipamento doméstico.
Palavras-Chave: Célula de carga. Iot. Armazém. Protocolo MQTT.

Abstract
With technological advances in industrial machinery and equipment, the labor market has been changing more and more to improve, reduce cost and manufacturing time. To collaborate with this evolution, using IoT Technology and load sensors, this present work aims to help improve the logistics process, more specifically in the replacement of materials in assembly lines, presenting an alternative for automation in warehouses, using a low system cost, with communication carried out via Wi-Fi, with the aid of the MQTT Protocol, using the ESP8266, which performs wireless communication and Half Point load cells for data capture. The NodeMCU ESP8622 12E was used as a microcontroller and Half Bridge Load Cells were used to read and capture the data. The HX711 Module was also needed for the integration between the NodeMCU and the Load Cells. We got satisfactory results and the automation software has as a great advantage its mobility and practicality, being able to be used as a domestic equipment.
Keywords*: Loading cell. Iot. Storage. MQTT protocol.*

INTRODUÇÃO

A quarta revolução industrial, também denominada de Indústria 4.0, tem promovido a integração de sistemas ciber-físicos, Inteligência Artificial, IoT, entre outros. Observa-se uma

[1] Graduação em Análise e Desenvolvimento de Sistemas pela Faculdade de Tecnologia do Estado de São Paulo – Fatec. E-mail: contato@fateccruzeiro.edu.br
[2] Graduação em Análise e Desenvolvimento de Sistemas pela Faculdade de Tecnologia do Estado de São Paulo – Fatec. E-mail: contato@fateccruzeiro.edu.br
[3] Doutorado em Computação Aplicada pelo Instituto Nacional de Pesquisas Espaciais – INPE e docente na Faculdade de Tecnologia do Estado de São Paulo – Fatec. E-mail: eugenio.almeida@fatec.sp.gov.br

fusão do real com o virtual e conectando sistemas digitais, físicos e biológicos, possibilitando a produção personalizada em massa (SCHWAB, 2016).

A Automação avançou com a Indústria 4.0, tornando-se objeto de estudo e desenvolvimento em várias áreas. As empresas tomam conta deste tema, pois com ela percebem a possibilidade de diminuir suas despesas, seja com mão de obra humana que tem um custo mais alto, seja com a economia esperada de energia ou o melhor aproveitamento do tempo na produção (WENDLING, 2013).

Na automação de sistemas industriais, comerciais, automobilísticos, domésticos, entre outros, é preciso determinar as (condições ou variáveis) do sistema. É necessário obter os valores das (condições ou variáveis) do ambiente a ser monitorado, e esta é o função de sensores (WENDLING, 2010).

Para Arozo (2003), os sistemas de gerenciamento de depósitos e armazéns (*Warhouses management System* - WMS) são os sistemas que tem a responsabilidade de gerenciar as operações do dia a dia dos armazéns e depósitos. Sua utilização está restrita a decisões totalmente operacionais, tais como: definição de rotas de coleta, definição de endereçamento dos produtos, entre outras.

Este trabalho tem como objetivo apresentar um solução de gerenciamento de depósitos e armazéns, utilizando as tecnologias e recursos da Indústria 4.0 e sensores de carga.

Simularemos um armazém de linha de produção, onde serão implementados sensores de carga que irão receber os dados de quantidade e peso dos materiais disponíveis, avisando ao repositor, se houver, a necessidade de reposição, diminuindo a possibilidades de procrastinação. Esta soluç˜são permitirá também a análise dos dados gerados e a criação de gráficos que auxilie ainda os gestores nas tomadas de decisões.

1. FUNDAMENTAÇÃO TEÓRICA

Neste capítulo será abordada a fundamentação teórica das tecnologias utilizadas para o desenvolvimento desta solução. Serão abordados temas como IoT, Protocolo MQTT, Banco de Dados NoSQL.

1.1. Internet das Coisas (IoT)

Na chegada da Indústria 4.0, trazendo consigo tecnologias como Big Data, Inteligência Artificial, entre outros, surge também o conceito de Internet das Coisas, oriunda do inglês *Internet of Things* (IoT). Termo primeiramente utilizado por Ashton (2009) ao trabalhar com

conceito de RFID (*Radio Frequency Identification*). Hoje em dia a definição é mais abrangente, engloba em várias aplicações, como na saúde, serviços públicos, transporte etc.(Gubbi et al, 2013). A IoT trouxe inteligência aos objetos, ou como afirmam Zhu, Leung, Shu, & Ngai (2015), é um facilitador inteligente do mundo, que tem na sua estrutura, itens como a internet e objetos inteligentes.

O principal objetivo do IoT é a conexão de usuários com objetos e dispositivos diversos, capaz de transmitir e receber dados, permitindo a visualização do funcionamento dos mesmos (PONTE, 2018). Um dos usos com maior potencial da Internet das Coisas é para o monitoramento de processos de diversos setores, desde a indústria até residências. Seu uso propicia uma melhora de qualidade e produtividade, porém ainda não é muito aplicado devido a objetos com inteligência primitiva, ou seja, objetos com tecnologias antigas ou desatualizados (ATZORI et al., 2010).

Segundo ATZORI et al. (2010), a *IoT* necessita de algumas tecnologias para seu funcionamento completo. Dentre elas estão protocolos de comunicação da Internet e dispositivos.

Como a IoT não possui uma arquitetura definida, vários especialistas opinam, ou seja, alguns defendem que a IoT possui uma arquitetura de 3 camadas e outros que possui 5 camadas.

1.1.1. Modelos de Arquitetura IoT

Algumas pessoas acreditam que a *IoT* possui uma arquitetura de 3 camadas, e outros, com o passar dos anos, essa arquitetura não representava mais a *IoT* por completo, sendo assim, apenas uma parte dela, surgindo uma Arquitetura de 5 camadas.

1.1.1.1. Arquitetura de 3 Camadas

Aproveitando as entradas de vários projetos destes eventos, um modelo de arquitetura em três camadas para IoT foi descrito por Wu *et al.* (2010). Segundo Wu *et al.* (2010) o modelo da IoT consiste em três camadas, na quais elas são: Camada de Percepção (*Perception Layer),* a Camada de Rede (*Network Layer)* e Camada de Aplicação (*Application Layer),* conforme mostra a Figura 1.

Figura 1: Arquiteturas de 3 Camadas para IoT

Fonte: Elaborado pelo Autor(2020)

Cada camada tem um propósito no funcionamento da *IoT*, e suas funções são:

- **Camada de percepção:** Esta camada é responsável pela parte de sensores da *IoT*, ou seja, tem a responsabilidade de identificar objetos e coletar informações deles. Ela se refere as "coisas" em *IoT*, como sensores, atuadores, câmeras, GPS, entre outros;
- **Camada de rede:** Esta camada tem a responsabilidade de enviar as informações já processados para a camada de aplicação;
- **Camada de aplicação:** Esta camada tem a responsabilidade de utilizar as informações coletadas pela camada de percepção e enviada pela camada de rede.

1.1.1.2. Arquitetura de 5 Camadas

Devido ao avanço da IoT, o modelo de arquitetura de três camadas não tem se mostrado mais suficiente para representar a complexidade dos sistemas IoT atuais (MASHAL et al., 2015). Segundo MASHAL *et al.* (2015) o novo modelo de cinco camadas tem sido sugerido com a adição de duas novas camadas: Camada de Processamento(*Processing Layer*) e Camada de Negócios(*Business Layer*). Esse modelo tem sido amplamente utilizado em diversos projetos de sistemas IoT e é o mais utilizado.

Figura 2: Arquitetura de 5 Camadas

Fonte: Elaborado pelo Autor (2020)

Cada camada tem um propósito no funcionamento da *IoT*, e suas funções são:

- **Camada de percepção:** Esta camada é responsável pela parte de sensores da *IoT*, ou seja, tem a responsabilidade de identificar objetos e coletar informações deles. Ela se refere as "coisas" em *IoT*, como sensores, atuadores, câmeras, GPS, entre outros;
- **Camada de rede:** Esta camada tem a responsabilidade de enviar as informações já processados para a camada de aplicação;
- **Camada de aplicação:** Esta camada tem a responsabilidade de utilizar as informações coletadas pela camada de percepção e enviada pela camada de rede.
- **Camada de processamento:** Esta camada tem como função armazenar, analisar e processar as informações transmitidas pela camada de rede;
- **Camada de negócios:** Esta camada é o "administrador" da *IoT*.

1.2. Protocolo de Comunicação

A Internet utiliza uma variedade protocolos de comunicação, responsáveis por enviar e/ou receber informações de um ou mais dispositivos para outros. Quando se fala em transmissão de dados entre dispositivos conectados em uma rede, surge a necessidade de pensar em protocolos que gerencie essas transmissões, ou seja, a troca de mensagens e/ou dados entre os dispositivos.

Neste trabalho surgem dois protocolos de comunicação que podem ser utilizados: *Messaging Queue Telemetry Transport* (MQTT) e o *Constrained Application Protocol* (CoAP).

1.2.1. Protocolo CoAP

O *Constrained Application Protocol* (CoAP) é um protocolo de comunicação da web especializado para uso com nós restritos e redes restritas (por exemplo, baixa potência, com perdas). Os nós geralmente têm microcontroladores de 8 bits com pequenas quantidades de ROM e RAM. As redes restritas, como IPv6 em Redes de Área Pessoal Sem Fio de Baixa Potência (6LoWPANs), muitas vezes têm altas taxas de erro de pacote e uma taxa de transferência típica de 10s de kbit / s. O protocolo é projetado para aplicações máquina a máquina (M2M), como energia inteligente e automação predial.(SHELBY, 2014)

O CoAP fornece um modelo de interação de solicitação/resposta entre terminais de aplicativos, oferece suporte à descoberta integrada de serviços e recursos e inclui conceitos-chave da Web, como URIs e tipos de mídia da Internet. O CoAP foi projetado para interagir facilmente com HTTP para integração com a Web enquanto atende a requisitos especializados, como suporte broadcast multiplexado, sobrecarga muito baixa e simplicidade para ambientes restritos. Na Figura 3, mostra o modelo de interação do CoAP.(SHELBY,2014)

Figura 3: Modelo de interação do CoAP.

Fonte: SHELBY (2014)

O modelo de interação do CoAP é semelhante ao modelo cliente/servidor do HTTP. No entanto, o CoAP usa de uma abordagem de duas camadas, uma camada de mensagens CoAP usada para lidar com o UDP e a natureza assíncrona das interações, e as interações de requisição/respostas usando códigos de método e resposta (SHELBY,2014).

1.2.2. Protocolo MQTT

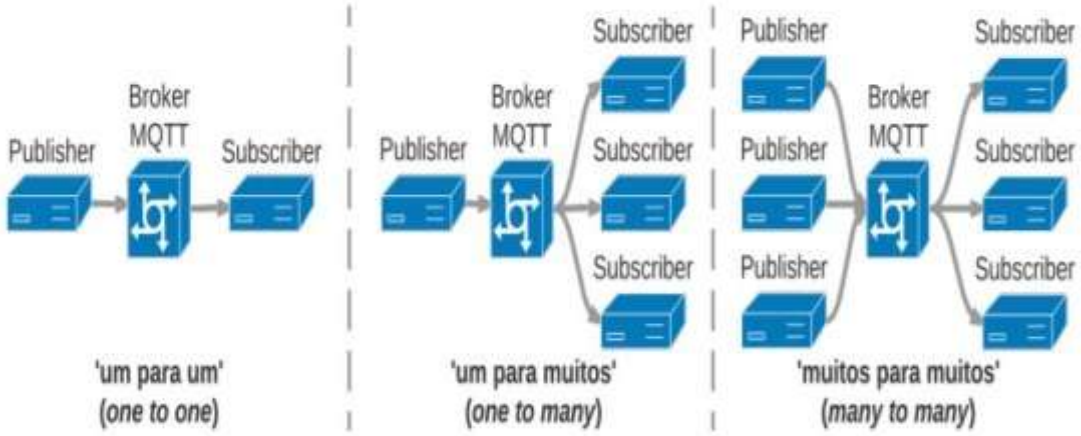
O *Message Queue Telemetry Transport* (MQTT) é um protocolo de comunicação entre máquinas através da Internet. Ele foi desenvolvido baseado no conjunto de protocolos TCP/IP e permite conexão com a Internet e consequentemente com outros dispositivos (YUAN,2017). O protocolo MQTT foi inventado pela empresa IBM, que tinha como intenção a comunicação entre satélites e pipelines de petróleo. Contudo, com a evolução, chegou a Quarta Revolução Industrial trazendo a Tecnologia *IoT* e o protocolo de comunicação MQTT passoua ser amplamente utilizados para projetos *IoT*.

A utilização do protocolo de comunicação MQTT em projetos IoT se torna ideal por alguns fatores, um deles é a característica de ser uma rede menos exigente, ou seja, permite a sua implementação em dispositivos com hardware muito restrito quando se trata de banda limitada e alta latência.(YUAN,2017).

Segundo Jaffey (2014), o protocolo baseia-se no modelo cliente/servidor. Os dispositivos sensores e/ou aplicações são clientes que se conectam a um servidor (*Broker*) usando TCP ou TCP/IP. As mensagens a serem enviadas são publicadas para um endereço chamado tópico, é bem semelhante a uma estrutura de diretórios em um sistema de arquivos, por exemplo, setor1/armazém/peso. Os clientes têm duas funções, serem os publicadores destas informações, ou então, serem aqueles que recebe esta informação, denominando aqueles clientes que subscrevem.

Dentre os diversos protocolos existentes para IoT, tais como MQTT, CoAP, este trabalho adotou o MQTT. Uma vantagem de utilização do MQTT é que somente a origem (publicador) precisa saber o endereço do *broker* (Negociador) permitindo três métodos de envio como mostra a Figura 4 (TORRES, 2016).

Figura 4: Métodos de Envio de dados suportados pelo MQTT



Fonte: Torres(2016)

A Figura 4 apresenta 3 métodos que o protocolo MQTT utiliza, primeiro “um para um”, onde há apenas 2 clientes, um que envia e outro que recebe. “Um para muitos”, o mesmo conceito do anterior, porém possui mais clientes para receber a informação do cliente que publica. E o terceiro, “muitos para muitos”, onde pode possuir N clientes que publica e N clientes que recebe a informação.

1.2.2.1. Comparação entre os Protocolos

Todo protocolo possui suas diferenças, mesmo que tenham o mesmo fim. Neste caso, comparamos dois protocolos muitos utilizados, como MQTT e CoAP, a fim de utilizar aquele mais adequado para o Trabalho.

Tabela 1: Comparação entre os Protocolos

	MQTT	**COAP**
CONEXÃO	Conexão de 1 a N Clientes.	Conexão de apenas um Cliente por vez.
PADRÃO	Padrão Publish-Subscribe	Padrão Request-Response
ARQUITETURA DE MENSAGENS	O Protocolo utiliza a infraestrutura e a integração dos Protocolos TCP e IP	O Protocolo transita suas mensagens pelo Protocolo UDP
PADRÃO DE ACESSO	Os Clientes têm uma relação mais próxima a Redes Internas.	Os Clientes precisam estar conectado ao um Servidor Local e somente o Servidor tem acesso a Redes Internas.

Fonte: Elaborado pelo Autor (2020)

Na tabela 1 mostra dois protocolos que são usados na *IoT*, ambos os protocolos possuem a mesma função, só que cada um tem à sua maneira de processar essa função, como por exemplo o padrão de ambos, onde o protocolo MQTT, pode ter mais de 1 cliente, tem o padrão de um cliente publicar uma informação e o outro receber a mesma, já o CoAP, trabalha com apenas um cliente conectado por vez, sendo assim seu padrão sendo diferente. Neste trabalho foi adotado o protocolo MQTT, porque seu padrão é ideal, e como este trabalho visa trabalhar com 1 ou mais clientes conectados ao negociador utilizando o protocolo MQTT.

1.3. Banco de Dados NoSQL

Inicialmente, as propostas de bancos de dados não relacionais foram desenvolvidas por pequenas empresas e por comunidades de software livre. Tais soluções foram então agrupadas em um termo, NoSQL (*Not Only SQL*), que significa "não apenas SQL". Este termo faz referência a SGBDs que não adotam o modelo relacional e são mais flexíveis quanto às propriedades ACID. (LÓSCIO; De OLIVEIRA; PONTES, 2011).

Eles têm como objetivo de atender aos requisitos de gerenciamento de grandes volumes de dados, semiestruturados ou não estruturados, que necessitam de alta disponibilidade e escalabilidade. (LÓSCIO; De OLIVEIRA; PONTES,2011).

Dentre as principais características dos BDs do tipo NoSQL estão a normalização das informações e o alto desempenho no armazenamento paralelo de dados em grande escala. Eles apresentam a capacidade de ser escalados horizontalmente, instalando servidores com menor capacidade de processamento um ao lado do outro. Desta forma conseguem tratar da leitura e escrita de uma enorme quantidade de dados. (PEREIRA; BORGES; RUBENS; SANTANA, 2013).

O processamento de dados distribuídos é possível através da utilização do conceito de MapReduce (Mapa Reduzido), que realiza o mapeamento dos dados para os servidores que irão processá-los e gerar um resultado (VIEIRA *et al*., 2012).

2. Matérias e Métodos

Neste capítulo são abordados os materiais, suas funcionalidades e os métodos de utilização.

2.1. Materiais

Os materiais utilizados no trabalho são de baixo custo e ferramentas *Open-Source.*

2.1.1. Célula de Carga

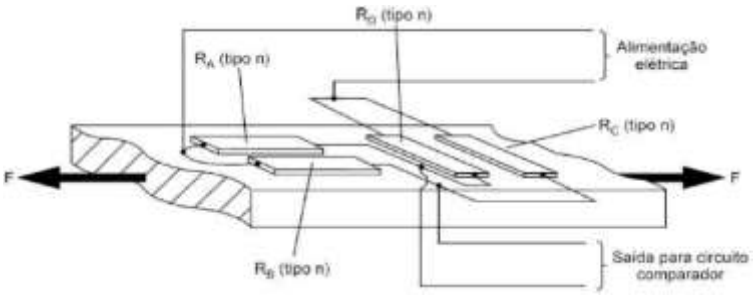
Célula é frequentemente utilizada para descrever um transdutor compacto (ex. célula de pressão, célula de carga, célula de torque). Alguns transdutores possuem seu nome familiar de acordo com o fenômeno físico que descrevem (BECK,1983).

Transdutores pode ser fabricado com base em diferentes princípios de operação (resistivo, indutivo, piezoelétrico, capacitivo etc.), a depender da natureza física da transformação de energia (BECK,1983).

As células (sensores) de carga (*Load Cell*), tem como princípio de funcionamento a variação de resistência ôhmica de um extensômetro (strain-gauge), quando submetido a uma deformação. A célula de carga mede a deformação da peça a ser medida pela sua própria deformação e traduz em seção transversal do extensômetro (THOMAZINI; ALBUQUERQUE, 2011). Vale ressaltar que algumas células já possuem um extensômetro, porém para células de carga antigas, se faz necessário adquirir um extensômetro.

Em geral, são utilizadas pelo menos duas células de carga. Uma para medir compressão e outra para medir tensão (forças aplicadas em direções diferentes). Com duas células de carga, tem-se uma ponte de Wheatstone completa. Para usar apenas uma célula de carga, é necessário completar a ponte com outros dois resistores (VIDAL, 2017). Neste trabalho optou em usar duas células de carga, duplicando o peso suportado de 50Kg para 100Kg.

Figura 5: Detalhes de Construção



Fonte: BECK (2013)

A célula de carga que utilizamos neste trabalho possui as seguintes especificações:

- Tempo de Resposta: 0,2 segundos
- Dimensões(mm): 28 x 28 x 8 mm
- Capacidade(kg): 50
- Margem de erro: 0,2 %
- Temperatura de operação: -20 a 65 °C
- Tensão de excitação(V): 5 a 10

- Preço atual: 15,80R$

2.1.2 Módulo HX711

As células de carga oferecem dados com grandezas elétricas muito baixa e já que a faixa de leitura da grande maioria dos equipamentos de controle requer níveis elevados de grandeza elétrica, com isso faz-se necessário usar amplificador de sinal e um conversor análogo/digital.(SEMICONDUCTOR,2014)

Segundo Araújo, Ferrando e Kakuno (2017), o Arduino possui um Conversor Analógico Digital (ADC) de 10 bits de resolução, que atende diversas demandas. No entanto, existem algumas aplicações em que o sinal a ser mensurado é menor que a sensibilidade do ADC do Arduino, como, por exemplo, em aplicações com células de carga.

Nesses casos, o amplificador/conversor HX711 é utilizado para converter o sinal analógico das células de carga e para digital de 24bits e amplificá-lo. Um chip amplificador SOP16L também faz parte do módulo. O sinal convertido e amplificado é enviado para as saídas DT (*Double tagged*) e SCK (*Serial Clock)* do conversor. (SILVA *et al*., 2017). Vale ressaltar que estas saídas são as mesmas entradas D1 e D2 de um ESP8622 NodeMCU, ou seja, o Módulo é a conexão entre as células de carga e um Microcontrolador, para que seja possível o envio dos dados. O Conversor/Amplificador Hx711 possui estas especificações:

- Tensão de operação: 4,8 a 5,5 VDC
- Corrente de operação: 1,6 mA
- Temperatura de operação: -20 a 85 °C
- Interface SPI
- Dimensões: 29 x 17 x 4 mm
- Preço Atual: 12,90R$

Segundo SEMICONDUCTOR(2014), o Módulo HX711 tem sua aparência como na Figura 6.

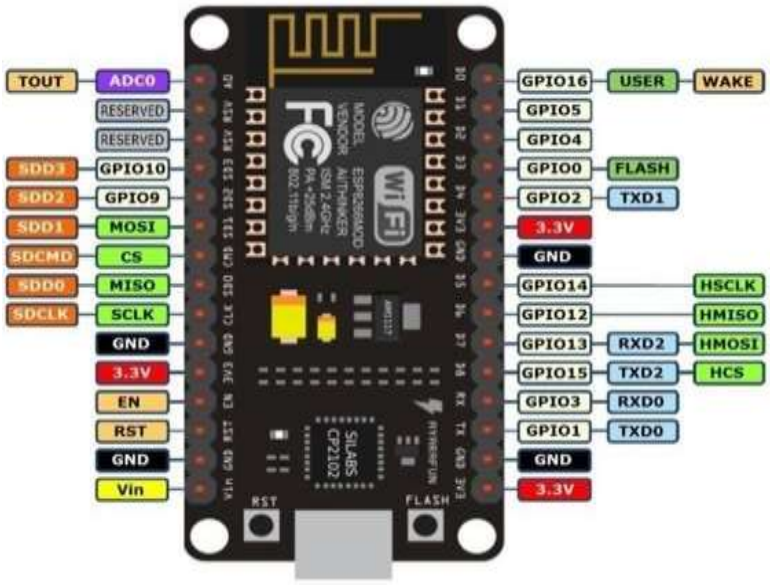
Figura 6: Imagem de um Módulo HX711

Fonte: SEMICONDUCTOR(2014)

2.1.3 Microcontrolador ESP8266 NodeMCU v3 (12E)

O NodeMCU é um *firmware* baseado no Microcontrolador ESP8266, surgiu para facilitar construção de projetos que utilizam o mesmo, pois apresenta um regulador de tensão de 3,3 V (tensão de operação do Microcontrolador ESP8266), e com ele não é mais necessário a utilização de outros componentes como adaptadores SPI (*Serial Peripheral Interface*) e UART (*Universal Asychronous Receiver Transmitter*) para realizar a conexão (OLIVEIRA,2017). O NodeMCU surgiu logo após o lançamento do Microcontrolador ESP8266, sendo lançado com o intuito de ser uma placa para desenvolvimento de projetos IoT. Outra grande vantagem deve- se ao fato de apresentar uma interface Micro USB-*serial* acoplada, o que facilita para tanto a parte de alimentação do microcontrolador, que pode ser realizada através de um carregador celular, como a parte de transmissão do programa escrito do computador para a placa (OLIVEIRA, 2017). A Placa NodeMCU pode ser programada em linguagem LUA ou C++, pela plataforma do IDE Arduino, como foi realizado neste trabalho.

Figura 7: Microcontrolador e suas respectivos entradas.



Fonte: FALLOWS(2016)

O Microcontrolador NodeMCU V3 (12E) possui estas especificações:

- Tensão de operação: 4,5 a 9 V
- Corrente de operação: 1,6 mA
- Temperatura de operação: -20 a 100 °C
- Interface SI (*Serial Interface)* / SPI
- Dimensões: 49 x 25,5 x 7 mm
- Suporta 5 conexões TCP/IP

- Preço Atual: R$. 50,90

2.1.4 InfluxBD

O Influx é um banco de dados de séries temporais de código aberto que possui alta disponibilidade para armazenar dados de séries temporais, especialmente os dados de sensores e de IoT para ser analisado em tempo real, é desenvolvido pela InfluxData (NAQVI,2017).

Desenvolvido na linguagem de Programação Go, ele é otimizado para lidar com dados de series temporais. Fornece uma linguagem de consulta semelhante ao SQL denominada InfluxQL, com a principal diferença na incapacidade de fazer junções, ou seja, o InfluxQL não tem a capacidade de fazer um agrupamento de tabelas, tendo que consultar dados de uma tabela por vez, a menos que seja feita por meio de consultas contínuas(NAQVI,2017).

O Influx usa na sua estrutura de dados interna a TSM(*Time Structured Merge Tree*). Semelhante a uma arvore LSM(*Log Structured Merge Tree*) em certo sentido, usa *log write-ahead*, indexa arquivos que são somente leitura e ocasionalmente realiza compactação para combinar arquivo de índice (INFLUXDATA,2019).

Consultas contínuas são consultas que fazem *loop* em intervalos específicos e reduzem a amostra de dados para pontos de dados únicos. Isso reduz a carga no banco de dados (SALENIUS,2017). Outras maneiras de reduzir a carga com o InfluxDB são feitas por meio de Fragmentos (*sharding*), que divide grandes bancos de dados em partes menores, gerenciadas com mais facilidade em diferentes instâncias do servidor de banco de dados (SALENIUS,2017). Segundo a DB-Engines (2020), a tendência de popularidade do InfluxDB, vem aumentando exponencialmente cada ano.

Gráfico 1: Tendência da Popularidade do Influx

Fonte: DB-ENGINES (2020)

O gráfico mostra a Popularidade do Banco de Dados InfluxDB desde 2015 até o mês de janeiro deste ano, aumentando de 200 a 21 mil.

2.1.5 Grafana

O Grafana é um software gratuito de código aberto, rico em recursos, analítico e de monitoramento que é executado como um aplicativo da Web para compor o painel e o gráfico. Grafana é o único das ferramentas populares de visualização e monitoramento que suporta mais de 30 gráficos abertos e vários tipos de fontes de dados que armazenam dados de séries temporais (KILI, 2018). As suas principais características consistem em uma interface de usuário fácil de usar por criar vários tipos de gráficos e flexibilidade devido ao seu código aberto e foco em sendo a melhor ferramenta de visualização para dados de séries temporais (SALENIUS,2017).

O Grafana mantém uma relação estreita com bancos de dados para realizar algumas operações de análise, visualização e monitoramento. No entanto, todos os gráficos visualizados Grafana realizam consultas no banco de dados, todas as funções são executadas através do banco de dados, ou seja, é como o a ferramentas fosse uma extensão dos bancos de dados (YIGAL, 2018).

2.1.5.1 Comparação entre Ferramentas de Monitoramento

Para maior entendimento, a Tabela 2 mostram as Ferramentas de Monitoramento e suas diferenças, como Limitações, Preços, Banco de Dados compatíveis e Gráficos.

Tabela 2: Comparativo entre Ferramentas de Monitoramento

Ferramentas	**Grafana**	**Kibana**	**Splunk**
Limitação	Não Possui	Não Possui	1 usuário tem 500Mb/dia.
Preço	Gratuito com opção comercial com preço de US$49/mês.	Gratuito com opção comercial em nuvem com preço entre US$16 a US$80/mês.	Gratuito com opção comercial com preço de US$150/mês.
Banco de Dados	InfluxDB, Elasticsearch, Graphite, MySQL, OpenTSDB, PostgreSQL, Prometheus, MongoDB, Redis, entre outros.	Utiliza Elasticsearch para pesquisa e Banco de Dados.	Splunk DB Connect(Banco de Dados Baseado em Arquivos)
Gráficos	Gráfico de série temporal com linhas, barras e pontos, status simples, tabela de valores, mapas de calor temporais, lista de alertas e stat.	Gráfico de série temporal, área, mapa de calor, barras horizontais, gráfico de linha, gráfico de torta, barras verticais, entre outros.	Gráfico de linhas, gráfico de área, gráfico de colunas, gráfico de barras, gráfico de pizza, gráfico de dispersão, gráfico de série temporal com linhas, barras, entre outros.

Fonte: Elaborado pelo Autor(2020)

A Tabela 2 apresenta as ferramentas de monitoramento de dados e suas especificações, mostrando se há limites, como um tempo determinado até a versão gratuita finalizar ou de armazenamento, preços, se o caso for necessário a utilização da versão paga, compatibilidades com banco de dados e seus respectivos gráficos. Neste trabalho adotou a Ferramenta Grafana, tem uma capacidade gigantesca de conexão com banco de dados e pelo seu recurso Alerta, que foi utilizado.

2.1.6 Telegraf

O *Telegraf* é uma das ferramentas desenvolvidas pela InfluxData, empresa especializada em plataformas de monitoramento de dados em séries temporais. Faz parte de um conjunto de ferramentas, mas pode ser utilizado separadamente e de forma gratuita (INFLUXDATA,2019).

Desenvolvido em Go e sob a licença de *open-source* MIT, consiste em um software agente que coleta, processa, agrega e escreve métricas. A arquitetura baseia-se em *plugins* de fácil instalação e configuração, o que proporciona uma fácil customização (GITHUB,2019).

No início só havia instalações Linux. Após a versão 1.20 do Banco de Dados InfluxDB, foi possível instalar em Sistemas Operacionais Windows, trazendo o primeiro agente coletor de métricas para o Windows em 2019 (INFLUXDATA,2019).

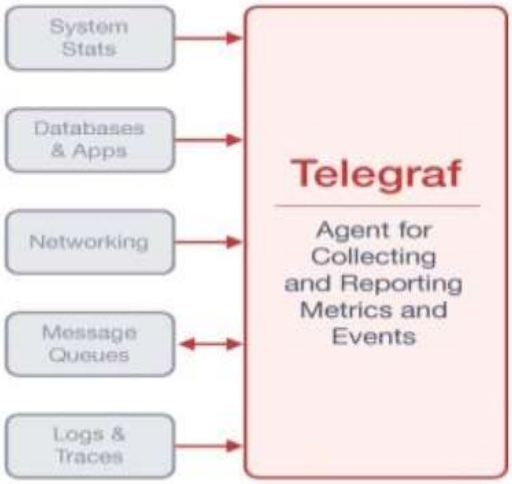
Os *plugins* acompanham o Telegraf em segundo plano onde se classificam em quatro categorias (GITHUB, 2019). São elas:

- Input: Plugins que fazem a coleta de métricas do sistema, serviços ou API de terceiros;
- Processor: Plugins que convertem e filtram métricas;
- Aggregator: Plugins que agregam métricas(calculam médias, valores, mínimos, máximos...);
- Output: Plugins que armazenam ou enviam métricas para ambientes externos.

Todas as métricas selecionadas possuem *plugins* nativos para a coleta de dados, e é possível enviar tais dados para ambientes externos(como Graphite, InfluxDB, entre outros) por meio de *plugins* específicos. O intervalo de coleta também é configurável através do arquivo de configuração principal do software.

A Figura 8 exibe um esquema simplificado das possibilidades de coleta de métricas de sistema.

Figura 8: Esquema simplificado de entrada de dados no *Telegraf*

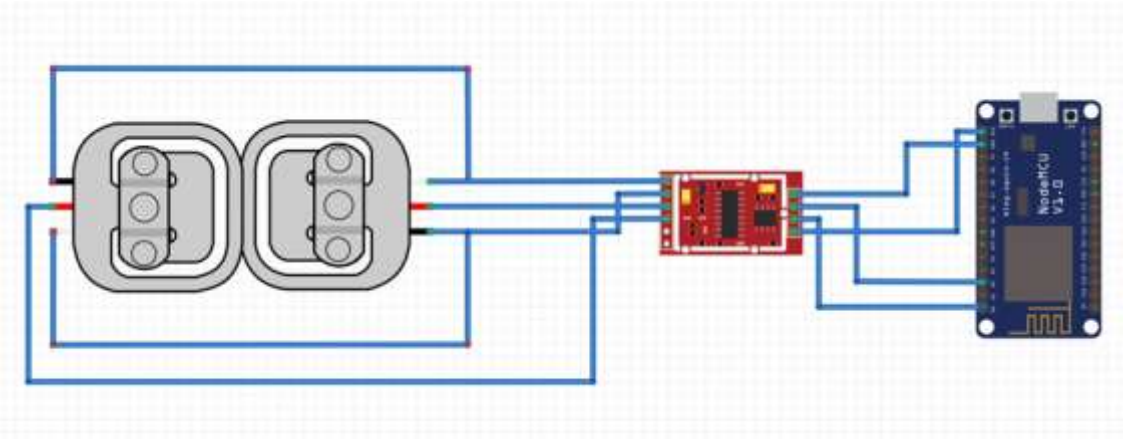


Fonte: (INFLUXDATA,2019)

2.2 Métodos

O protótipo observado na Figura 9, foi montado e projetado para ser capaz de medir grandezas como peso e guardar esses dados para que possam ser analisados posteriormente.

Figura 9: Esquema do circuito montado no Fritzing



Fonte: Elaborado pelo Autor(2020)

As células de carga são responsáveis por realizar a medida da sua grandeza. O valor da sua resistência é convertido e amplificado pelo HX711 conversor A/D, e o sinal digital resultante deste conversor é enviado para o NodeMCU ESP8266 V3 (12E). Vale ressaltar que, para que a célula seja capaz de medir sua grandeza correspondente, é necessário que a parte inferior das células fique suspensa, pois senão os pinos que prendem o transdutor impedirão a flexão do sensor. Nesse caso, utilizando como base madeira nas laterais das células, como é mostrado na Figura 10.

Figura 10: Células de Cargas com Base de Madeira

Fonte: Elaborado pelo Autor(2020)

O microcontrolador NodeMCU ESP8266 V3 (12E) recebe toda a programação necessária para realizar a interface entre os dados coletados e o computador/notebook. Nos equipamentos, se encontra instalado o Banco de Dados InfluxDB e Ferramenta de Monitoramento Grafana. No InfluxDB, foi feito um banco de dados chamado **medição**, onde os dados são armazenados, é enviado o dado de Peso ao InfluxDB e com a linguagem InfluxQL transforma os dados de Peso em Quantidade. Na ferramenta Grafana os dados coletados têm um intervalo de 5 segundos, ou seja, cada dado é exibido após o outro entre 5 segundos e são exibidos de duas formas diferentes:

O primeiro modo de exibição é com um gráfico comum, figura 11, mostrando a zona de risco (linha vermelha), zona de cuidado (linha amarela) e a zona normal(linha verde).

Figura 11: Gráfico Comum

Fonte: Elaborado pelo Autor(2020)

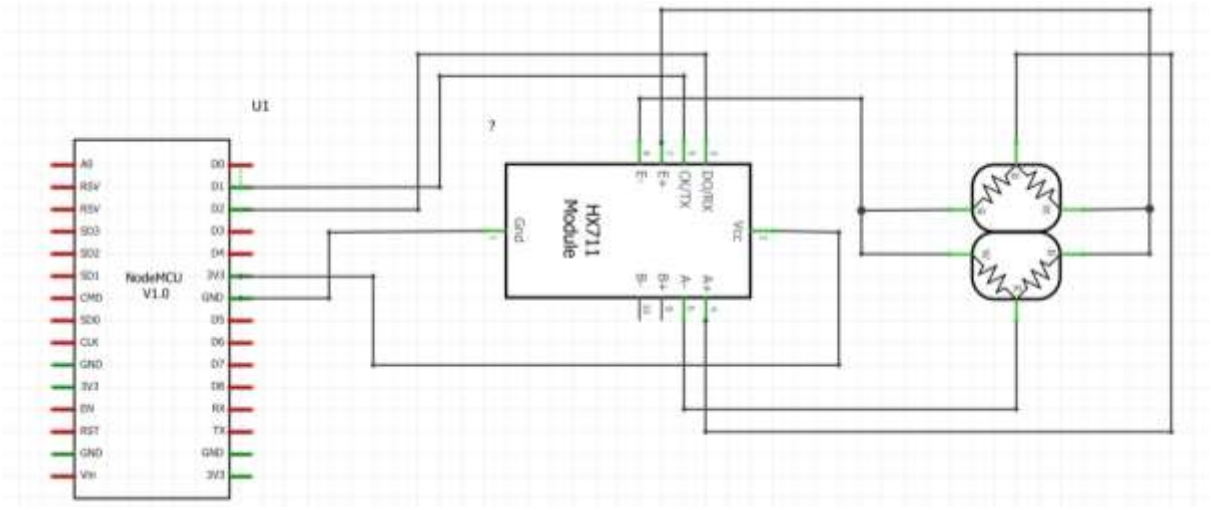
O segundo modo de exibição é por gráfico Gauge, mostrando também as limitações de cada medida coletadas das células de carga. Sua diferença é bem visível do outro, onde mostra apenas o valor e suas limitações. Na vertical, eixo Y, mostra o limite em kg, de 0kg a 100kg.

Figura 12: Gráfico Gauge

Fonte: Elaborado pelo Autor (2020)

Este gráfico mostra em unidades melhor a informação coletada pelos sensores, mostrando também, em cima, ao chegar em certos valores, os limites. A ferramenta Grafana possui diversos tipos de gráficos, porém, para este trabalho foi escolhido apenas esses dois tipos de gráficos pela sua simplicidade e praticidade.

Figura 13: Circuito montado em modo esquemático



Fonte: Elaborado pelo Autor (2020)

A Figura 13mostra a ligações dos componentes em suas respectivas portas. Para o funcionamento correto de uma Ponte de Wheatstone, foram necessárias duas Células de Carga, alterando assim, a capacidade do trabalho.

O Módulo HX711, possui apenas 1 entrada positiva e negativa para a alimentação, com isso foi necessário interligar ambas as células, Fio Branco de uma com o Preto da outra, e o Fio Preto de uma com o Fio Branco da outra, assim ambas as Células de carga ficam ligadas nas portas E+,E-, A+ e A- do Módulo HX711, onde correspondem: E+ e E-, Energia Positivo e Negativo, A+, Entrada Analógica Positivo e A-, Entrada Analógica Negativa.

O microcontrolador NodeMCU é ligado nas seguintes portas: VCC (tensão de

alimentação positiva), GNU (tensão de alimentação negativa), DK (saída digital) e SCK (entrada digital).

3 RESULTADOS

Os resultados esperados foram alcançados com êxito. Seguindo a metodologia do trabalho foi possível compreender o funcionamento da solução de automação de forma simples. Foram utilizados os materiais anteriores como testes: Pregos de 22x48 mm pesando 1kg a cada pacote de 50 unidades, com duas Células de carga, a capacidade é aumentada, 100kg, ou seja, 5000 unidades de pregos 22x48 mm no total.

A solução de automação utilizou duas Células de Carga e apenas um Módulo HX711. Deste modo, foi possível gravar os dados pelo programa de automação no Microcontrolador NodeMCU v3 para serem consultados quando necessário, para consultar estes dados, foi utilizado o aplicativo Mosquitto, que trabalha com Protocolo MQTT, ou seja, os dados são enviados do NodeMCU v3 para o Equipamento via *WiFi* (*Wireless Fidelity*) e organizados no Intermediário, *Mosquitto Broker*, onde o *Broker* funciona como um servidor entre aqueles que recebem um dado e aqueles que publicam o mesmo, neste caso, nosso cliente publicador foi o microcontrolador NodeMCU e a Ferramenta *Telegraf* foi um cliente recebedor. No *Telegraf*, é configurado para se tornar um cliente recebedor do nosso intermediário, figura 14.

Figura 14: Configuração do *Telegraf*.

```
[[inputs.mqtt_consumer]]

# Servidor Local da Maquina que esta rodando o Broker MQTT e a Porta
servers = ["localhost:1883"]

# Os Tópicos onde estão sendo publicados os dados
topics = ["medicao/peso","medicao/quantidade"]

# É o valor retirado do Broker Mosquitto
data_format = "value"

# Tipo de dado que esta sendo retirado do Broker e enviado para o Influx
data_type = "float"

#Configurações do InfluxDB
[[outputs.influxdb]]

# Servidor onde o Influx funciona e a porta
urls = ["http://localhost:8086"]

# Banco de Dados utilizado para a operação
database = "medicao"

# skip_database_creation = true

timeout = "5s"

#É o nome dado aos Measurements do InfluxDB
name_override = "dados"
```

Fonte: Elaborado pelo Autor(2020)

A Figura 14 mostra a configuração do *Telegraf*, onde o *inputs.mqtt_consumer* é responsável de tornar o *Telegraf* um cliente recebedor do *broker* e o *outputs.influxdb* é responsável fazer a conexão no InfluxDB, para armazenar os dados. Um recurso interessante do *Telegraf* é a criação de um banco de dados, se não houver, onde o *Telegraf* tem a capacidade criar um banco de dados no InfluxDB ou em outros SGDB. Após a configuração, é feito duas consultas para receber os dados de peso e quantidade no Grafana. A configuração das consultas então na figura 15**.**

Figura 15: Configuração de Consulta

Fonte: Elaborado pelo Autor(2020)

Após a realização da configuração das consultas, os gráficos são exibidos (Figura 16)**.**

Figura 16: Gráfico exibindo dados de Peso.

Fonte: Elaborado pelo Autor(2020)

O gráfico da Figura 16 mostra os dados de Peso (Weight), no eixo Y do gráfico mostra o mínimo de 0kg até o máximo suportado que no caso é 100kg. Os Dados do Grafana levam 5 segundos para alterar, ou seja, mesmo que os dados cheguem no InfluxDB e o Grafana consulte-os, ainda demora 5 segundos para, de fato, mostrar nos gráficos.

Anteriormente utilizamos o *Thresholds* do Grafana para mostrar as limitações, porém ao utilizar o recurso Alerta, ocorreu a desativação dos *Thresholds*, ou seja, nesta versão atual, 7.3.2, que foi utilizado neste trabalho, não foi possivel implementar ambos os recursos em um gráfico.

Ao remover materiais, os gráficos vão alterando conforme as remoções dos materiais, até chegar na faixa avermelhada a Ferramenta Grafana começa a utilizar o recurso Alerta de modo excessivo, pois como a reposição é aleatória, se não houver uma reposição até o momento, o Grafana irá retornar a mandar uma mensagem com a situação do gráfico para o usuário, e isto acontece até que os materiais sejam devidamente colocados.

Vale ressaltar que, se não houver nenhuma reposição, os dados continuam a descer, chegando ao zero, como se trata de uma simulação, apenas fica no aguardo da reposição ser feita, em casos reais, pode resultar em paradas na produção. Quando chega a esta situação, tanto a quantidade quanto o peso devem chegar em valores menor do que 5kg e/ou 250 unidades. A reposição é aleatória, ou seja, nem sempre, ao repor, vai voltar em 100kg e/ou 5000 Unidades, tem vezes que mesmo repondo, parou ainda na faixa de perigo.

4. CONCLUSÃO

O objetivo deste trabalho foi desenvolver um sistema para monitorar os materiais de uma linha de produção usando tecnologia IoT. Atualmente o processo de monitoramento e reposição de peças depende de serviços manuais.

Para isso foram utilizadas células de carga, microcontrolador, Broker MQTT, *Telegraf*, InfluxDB e Grafana. Foi implementado em um workflow que tinha a função de coletar os dados do sensor, armazenar em um banco de dados e visualizar na forma de peso e quantidade. Com essas soluções desenvolvidas espera contribuir na área de logística, na reposição e controle dos materiais.

Utilizando como protocolo o MQTT, é possivel aumentar a distância da conexão utilizando *Acess Point* para que os dados cheguem ainda mais longe. O recurso Alerta do Grafana, mesmo ainda sendo Beta, ou seja, uma versão ainda no estágio de desenvolvimento, funcionou devidamente, além de possuir uma grande variedade de lugares onde possa mandar as mensagens de Alerta.

O *Telegraf* foi utilizado como ferramenta de coleta de métricas e é possivel configurá-lo facilmente. InfluxDB, o Banco de Dados que foi essencial para este trabalho por trabalhar em Serie Temporal.

Referências

AROZO, R *et al*. Softwares de *supply chain management*: Definições, principais funcionalidades e implantação por empresas brasileiras. Logística e gerenciamento da cadeia de suprimentos*:* Planejamento do fluxo de produtos e dos recursos. São Paulo: Atlas, 2003.

BECK, J.C.P. (1983). Projeto, construção e análise de células de carga de placa e de anel. Escola de Engenharia de UFRGS. Rio Grande do Sul.

CASSINELLI, F.; MIRANDA, L.; VALE, L. (2018). Constrained Application Protocol.
DALLY, J. W.; RILEY, W. F.; MCCONNELL, K. G. (1993) Instrumentation for Engineering Measurements. University of Maryland, Iowa State University.

DAL PONTE, G. B. (2019). Desenvolvimento de sistema de monitoramento remoto de consumo de gás envasado utilizando tecnologia IOT. Instituto Federal de Santa Catarina, Santa Catarina.

DEVOLVIAM, Ali & Chen, Liu & Liu, MENGCHI. (2018). A survey on NoSQL stores. ACM Computing Sorves. 51. 1-43. 10.1145/3158661.

DOCS, GitHub (2019). The plugin-driven server agent for collecting & reporting metrics. [*S. l.*].

GUBBI, J *et al*. (2013). Internet of Things (IoT): A vision, architectural elements, and future directions. Future Generation Computer Systems, 29(7), 1645–1660. https://doi.org/10.1016/j.future.2013.01.010

L. ATZORI et al, LERA A., MORABITO G. (2010) The Internet of Things: A Survey Computer Networks. DIEE, University of Cagliari, Italy.

LAVARDA, M.D.; ZADUSKI, S.; GOMES L. G.; GAMBA, H. R.; BORBA G.B. (2014) PLATAFORMAS DE FORÇA PARA A REABILITAÇÃO POR BIOFEEDBACK DO CONTROLE MUSCULAR DOS MEMBROS INFERIORES EM HEMIPARÉTICOS.

Congresso Brasileiro de Engenharia Biomédica.

LIMA, B. L. S. de; SILVA, A. N. R. da. (2006). Extensômetros de resistência elétrica fabricados por serigrafia para aplicação como sensores de pressão e célula de carga. Universidade de São Paulo, São Paulo.

MIAO, Wu. *et. al* (2010). Research on the architecture of Internet of things. In.: 3rd IEEE International Conference on Advanced Computer Theory and Engineering (ICACTE). Sichuan, China, 21 ago. 2010. p. 484-487.

NAQVI, et al; YFANTIDOU, S.; ZIMÁNYI, E. (2017) Time series databases and influxdb. Studienarbeit, Université Libre de Bruxelles. Bruxelas.

OSHIMA, C. K. (2019). Desenvolvimento e validação de sistema de aferição de esforço físico em equipamento de treinamento de surfe – um estudo piloto. Instituto de Educação, Ciência e tecnologia de Santa Catarina. Santa Catarina.

PEREIRA, F. S. *et al*; BORGES, H. P.; RUBENS, H.; SANTANA S. A. (2013) Utilização de

Banco de Dados NoSQL em Ambientes Corporativos. Unitri – Centro Universitário do Triângulo, v. 3, n. 1.

SANTOS, F.C.F.(2011). Projeto e Construção de Sensores de Carga e Deslocamento.
Faculdade Ciências e Tecnologia, Universidade Nova de Lisboa.

SALENIUS, M. (2017) Designing A Data Solution for A Game Company: Case InfluxDB and

Grafana. Bachelor of Business Administration.

SEMICONDUCTOR. (2014) Converter and Amplifier HX711. [S. I.].

SHELBY, Z. *et al (*2014). The Constrained Application Protocol (CoAP). Transfer Protocol, Universitaet Bremen TZI, p. 1-112, 5 jun. 2014.

SCHWAB, K. *et al (*2016). The Fourth Industrial Revolution, Alemanha, v. 1, n. 1, 11 Jan. Indústria, p. 1-192.

THOMAZINI, D.; ALBUQUERQUE, P.U.B. de. (2011) Sensores Industriais: Fundamentos e Aplicações.8.Ed. p.113. Brasil. Editora Érica.

TORRES, B. A. ROCHA, R. A. SOUZA, N. J (2016). Análise de Desempenho de Brokers MQTT em Sistema de Baixo Custo, Fortaleza. Grupo de Redes de Computadores, Engenharia de Software e Sistemas (GREat) - Universidade Federal do Ceará.

VIEIRA, M. R., *et al.* (2012) "Bancos de Dados NoSQL: Conceitos, Ferramentas, Linguagens e Estudos de Casos no Contexto de Big Data". In: Simpósio Brasileiro de Bancos de Dados, São Paulo.

VIDAL, V. *et al (*2017*)*. Célula de Carga(Strain Gauge). Balança, CEFET - Mina Gerais, p. 1-78, 11 nov. 2017.

YUAN, M. et al. (2017). Conhecendo o MQTT. Transfer Protocol, IBM [S. l.].

WENDLING, M. (2010) Sensores. Universidade Estadual Paulista, Guaratinguetá.