

# Sistema de Sensoriamento de Baixo Custo Baseado em IoT

Daiane de Cássia Ribeiro & Suelem Regina de Freitas

**Abstract**— This article describes the implementation of a sensing system based on IoT and Industry 4.0 concepts where process variables are monitored by a remote device. The proposed solution consists in an OPC-UA client embedded in a *Raspberry Pi* development board connected to field sensors. The sensors data are sent to an OPC-UA server. This data can be viewed by a monitoring screen via an HTTP server.

**Index Terms**—Sensing, IoT, *Raspberry Pi*, HTTP Protocol

**Resumo**— Esse artigo descreve a implementação de um sistema de sensoriamento baseado em IoT e conceitos da indústria 4.0, onde variáveis de processo são monitoradas por meio de um dispositivo remoto. A solução proposta consiste em um cliente OPC-UA embarcado em uma placa de desenvolvimento *raspberry pi* conectada à sensores de campo. Esta placa recebe dados dos sensores e os envia para um servidor OPC-UA. Esses dados podem ser visualizados em uma tela de monitoramento por meio de um servidor HTTP (*HyperText Transfer Protocol*).

**Palavras chave**—Indústria 4.0, IoT, Protocolo HTTP, *Raspberry Pi*, Sensoriamento.

## I. INTRODUÇÃO

O avanço do setor industrial aponta a necessidade de monitoramento de dados e criação de processos mais inteligentes, o que demanda das indústrias a busca de alternativas para melhorá-los e otimizá-los. [1]

O primeiro marco para esse avanço foram as Revoluções Industriais. Desde a primeira revolução que transformou a mão de obra manual em mecânica, seguida da segunda revolução que se caracterizou pela manufatura em massa. Na terceira a eletrônica, a tecnologia da informação e as telecomunicações foram destaque. Por fim tem-se a presente quarta revolução industrial também chamada de Indústria 4.0. [1]

Conforme relatório apresentado pela BCG (*Boston Consulting Group*) são dez as tecnologias aplicadas a Indústria 4.0: Robôs Inteligentes, manufatura aditiva e híbrida, simulação virtual, integração horizontal e vertical dos sistemas, IoT (*Internet of Things*), *Big Data & Analytics*, computação em nuvem, segurança cibernética, realidade aumentada e ética. [2]

Trabalho de Conclusão de Curso apresentado ao Instituto Nacional de Telecomunicações (INATEL) como parte dos requisitos para obtenção do Título de Bacharel em Engenharia de Controle e Automação. Aprovado em 03/12/2021 pela comissão julgadora: Prof. MSc. Egídio Raimundo Neto / INATEL – Orientador e Presidente da Comissão Julgadora, Prof. MSc. João Paulo Carvalho Henriques / INATEL – Membro da Comissão Julgadora, Prof. João Pedro Magalhães de Paula Paiva / INATEL – Membro da Comissão Julgadora. Coordenador do Curso de Engenharia de Controle e Automação: Prof. Dr. Alexandre Baratella Lugli.

No setor industrial, os dispositivos se tornam cada vez mais conectados e geradores de dados que são utilizadas no *Big Data*, juntamente ao IoT, para gerenciamento e análise dos dados gerados. Visto isso, o uso de recursos em nuvem é necessário e o protocolo OPC-UA (*Open Platform Communication – Unified Architecture*) é capaz de realizar a comunicação entre os dispositivos do chão de fábrica com sistemas de computação em nuvem. [2]

O trabalho desenvolvido tem como propósito apresentar um sistema de sensoriamento de baixo custo para a aquisição de dados utilizando conceitos da indústria 4.0 e demonstrar o OPC-UA sendo embarcado em outras plataformas de desenvolvimento. Nesta aplicação o cliente OPC-UA está embarcado em uma *raspberry-pi*, onde estão conectados todos os sensores, e os dados coletados são enviados para o servidor OPC-UA localizado na nuvem. Esta solução está integrada a um servidor HTTP onde os dados podem ser visualizados e monitorados. A aplicação proposta pode ser utilizada no setor industrial, para desenvolvimento de equipamentos e processos mais inteligentes e mais conectados.

O trabalho é organizado em capítulos, apresentados como descrito a seguir. No Capítulo II apresentam-se os fundamentos teóricos sobre os temas os temas abordados. Em seguida apresentam-se os materiais e métodos utilizados no Capítulo III. Já no Capítulo IV o desenvolvimento da aplicação prática é reportado. O Capítulo V apresenta os testes e os resultados obtidos e, por fim, no Capítulo VI apresenta-se a conclusão.

## II. FUNDAMENTOS TEÓRICOS

Neste capítulo descrevem-se os conceitos teóricos necessários à compreensão da aplicação prática desenvolvida neste trabalho.

### A. IoT – Internet of Things

O termo IoT foi criado em 1999 por Kevin Ashton, um visionário e precursor da tecnologia que imaginou um futuro onde todas as “coisas” fossem conectadas à internet. O IoT é uma solução simples e eficiente, onde seu principal objetivo é conectar dispositivos à rede e possibilitar a troca de dados entre si. [3]

Alguns avanços recentes fizeram o IoT ser possível, dentre eles: o acesso à tecnologia de sensores de baixo custo e baixa potência, a conectividade, plataformas de computação em nuvem e análise avançada. [3]

As indústrias utilizam este conceito para criar processos mais conectados, principalmente na área de instrumentação e

sensoriamento remoto, surgindo assim um novo conceito de internet das coisas, o IIoT (*Industrial Internet of Things*). [3]

Dentre as vantagens de se utilizar o IoT existem os chamados três "Ces" do IoT: Comunicação, Controle e Automação e Custos Reduzidos. [4]

Na Comunicação, informações são expostas às pessoas e aos sistemas, desde o estado e integridade do equipamento (ligado, desligado, alto, baixo), bem como sensores que detectam sinais vitais de uma pessoa, antes disso essas informações eram coletadas manualmente e com pouca frequência. Em Controle e Automação, quando conectada, uma empresa conseguirá de forma remota acessar, controlar dispositivos, visualizar possíveis alarmes e até mesmo reconhecê-los. A adoção do IoT gera economia, pois as medições realizadas fornecem dados em tempo real, assim facilitando a detecção de possíveis falhas de equipamentos e dispositivos. [4]

### B. Computação em Nuvem

O serviço em nuvem parte de dois princípios de funcionamento: o receptor e o provedor. O provedor armazena grande quantidade de informações, analisa e as distribui, para serem utilizadas em seguida pelo receptor, assim deixando o usuário com acesso à documentos e informações às quais não necessitam mais ser armazenadas em sua máquina. [5]

Desde seu surgimento em 1997, em uma palestra ministrada por Ramnath Chellappa, o termo computação em nuvem vem ganhando espaço nas indústrias, pois não há necessidade de ter e manter servidores físicos e *datacenters*, os serviços podem ser acessados e instalados em uma plataforma em nuvem, além de fornecer mais agilidade de comunicação entre setores. [5]

Os serviços em nuvem podem facilitar muitos processos de fabricação e se tornou uma parte fundamental da indústria 4.0 integrando toda a cadeia produtiva e permitindo acesso de dados de qualquer lugar e por qualquer dispositivo. [5]

Dentre os serviços oferecidos pela nuvem, destaca-se três, são eles: SaaS (*Software as a Service*), Paas (*Platform as a Service*) e IaaS (*Infrastructure as a Service*). [6]

No modelo de nuvem SaaS o usuário pode acessar um *software* sem a necessidade do usuário manter um recurso físico para armazenamento desses *softwares*. Neste tipo de serviço o fornecedor do *software* é o responsável por toda estrutura oferecida ao usuário final. Muitos sistemas CRMs (*Customer Relationship Management*) e ERPs (*Enterprise Resource Planning*) utilizados na indústria trabalham no SaaS. [6]

No serviço PaaS é oferecido uma plataforma, onde são desenvolvidas e implementadas as soluções de tecnologia para nuvem. É neste modelo que o usuário pode desenvolver, compilar e testar suas aplicações sem se preocupar com a arquitetura, pois o ambiente de execução já é preparado pelo provedor do serviço em nuvem. [6]

No modelo IaaS a empresa contrata uma infraestrutura computacional, geralmente são ambientes virtualizados que podem ser acessados remotamente. Os pacotes de serviços oferecidos podem conter desde servidores até roteadores. As principais vantagens do IaaS são: redução de custo, risco de operação e escalabilidade, pois os serviços são cobrados de acordo com o uso. [6]

A Figura 1 apresenta os modelos de computação em nuvem.

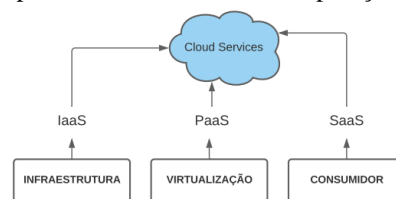


Figura 1 – Modelos de Serviços na Nuvem. [6]

Na aplicação descrita neste artigo serão utilizados os modelos IaaS e SaaS. O IaaS será utilizado devido ao uso de um computador virtual para executar os servidores. A tela de monitoramento será um modelo SaaS, pois o usuário pode acessá-la sem a necessidade de nenhuma instalação em seu ambiente local. [6]

### C. Sistemas de Supervisão

Os sistemas de supervisão têm a função de monitorar dados importantes em um processo e disponibilizá-los ao operador ou gerente e tudo isso de forma autônoma. [8]

Os primeiros sistemas de monitoramento utilizavam painéis com lâmpadas e *buzzers* para sinalizar ao operador o estado do seu processo. Com a evolução da automação, os sistemas de supervisão se tornaram mais inteligentes e capazes de monitorar um grande número de dados em tempo real. [8]

Os sistemas supervisórios também são conhecidos como sistema SCADA (*Supervisory Control and Data Acquisition*) e dentro do processo industrial são utilizados para monitorar dados das variáveis de processo. As principais vantagens de se utilizar esse tipo de sistema para supervisão são: sinalização de alarmes em tempo real, operação remota no processo industrial, geração de relatórios e gráficos. [9]

### D. OPC (Open Platform Communications)

O OPC é um protocolo de comunicação para troca de dados de forma segura e confiável no ambiente industrial. Ele surgiu da necessidade de criar um protocolo padronizado que pudesse ser utilizado nas indústrias sem a necessidade de criar *drivers* específicos para cada fabricante. Essa solução veio para extinguir o problema de interconexão entre os diversos dispositivos e facilitar e comunicação entre eles dentro da planta industrial. [10]

O primeiro padrão foi denominado OPC Classic e seu uso era limitado apenas ao sistema operacional Windows. Ele possuía três especificações:

- OPC DA (*Data Access*) - Define a leitura e escrita dos dados das variáveis de processo. Seu principal objetivo é transmitir dados dos CLPs (Controlador Lógico Programável) ou outros dispositivos de aquisição de dados. [10]
- OPC AE (*Alarms & Events*) - Define a troca de dados de alarmes e eventos do sistema. [10]
- OPC HDA (*Historical Data Access*) – Define métodos de consulta e análises que são aplicados a dados históricos. [10]

Devido às limitações de sistema operacional do OPC *Classic*, houve a necessidade de criar um OPC compatível com outras plataformas, como Linux e MAC, então em 2008 surgiu o protocolo OPC UA, esta versão possui todas as funcionalidades do OPC *Classic*, porém supera as limitações e adiciona novas funcionalidades. [10]

O funcionamento do OPC é baseado na tecnologia OLE (*Object Linking and Embedding*), que foi originado dos padrões COM (*Component Object Model*) e DCOM (*Distributed Component Object Model*), ambos da Microsoft. Ele utiliza a arquitetura cliente/servidor, onde um cliente OPC pode interagir com um servidor OPC, realizando assim a troca de dados. [10]

A Figura 2 demonstra a arquitetura de comunicação do protocolo OPC.

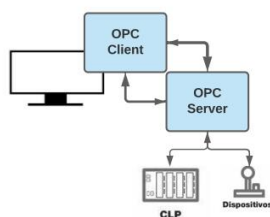


Figura 2 – Comunicação OPC. [11]

No protocolo OPC UA, a troca de informações com a *web* é muito importante para monitoramento de dados. As informações obtidas são enviadas para *web* através de protocolos padrões, como SOAP (*Simple Object Access Protocol*), HTTP (*Hypertext Transfer Protocol*) ou OPC TCP (*Transmission Control Protocol*). [11]

O OPC-UA traz consigo conceitos da Indústria 4.0, tais como: Comunicação independente de plataforma e fornecedor, segurança de dados, padronização e inteligência descentralizada. O OPC-UA foi projetado para sistemas de escalabilidade e oferece suporte para uma ampla gama de domínios, desde de dispositivos de campo ao gerenciamento empresarial. [11]

### III. MATERIAS E MÉTODOS

Neste capítulo são descritos os materiais e os métodos utilizados para o desenvolvimento deste projeto.

#### A. Raspberry Pi 4

O *raspberry Pi* é um microcomputador embarcado criado pela *raspberry Pi Foundation*, que tinha como objetivo disponibilizar minicomputadores de baixo custo para democratizar o estudo da computação em diversas partes do mundo. A primeira versão desse dispositivo foi lançada em 2012 e hoje já conta com mais de 15 versões diferentes. [12]

Além de oferecer funcionalidades de um computador, ele pode ser usado para outras aplicações, geralmente embarcadas, como: central de mídia, sistema de monitoramento, entre outros. [12]

O modelo utilizado neste projeto possui uma arquitetura baseada em ARM (*Advanced RISC Machine*) e conta com um

processador de 64bits. Suas principais características são:

- Processador Broadcom BCM2711, quad-core Cortex-A72 (ARM v8) 64-bit SoC;
- Clock 1.5 GHz;
- Memória RAM: 4GB DDR4;
- Adaptador Wifi 2.4GHz e 5.0GHz IEEE 802.11b/g/a/ac *wireless* LAN integrado;
- Adaptador Wifi 2.4GHz e 5.0GHz IEEE 802.11b/g/a/ac *wireless* LAN integrado;
- Bluetooth 5.0 integrado;
- Gigabit Ethernet (PoE);
- 2 portas micro HDMI com suporte a vídeos 4K e 60fps;
- 2 portas USB 3.0 e 2 portas USB 2.0;
- Alimentação: 5.1V com conector USB-C;
- Interface para câmera (CSI);
- Interface para *display* (DSI);
- Slot para cartão microSD;
- Conector de áudio e vídeo;
- GPIO de 40 pinos. [12]

A Figura 3 mostra o dispositivo e suas entradas e saídas.



Figura 3 – Entradas e saídas Raspberry Pi 4. [12]

#### B. Sensores e Periféricos do Sistema

Os sensores são dispositivos sensíveis a parâmetros físicos, como temperatura, pressão, luminosidade, vazão, entre outros. Os sensores podem ser divididos em analógicos e digitais, sendo que a principal diferença entre eles é o tipo de sinal entregue. Os sensores analógicos entregam um sinal contínuo e sensores digitais fornecem um sinal discreto (0 ou 1). [13]

Neste projeto utiliza-se o sensor BMP280, o sensor DHT11, o módulo de entrada e saída PCF8591 e um módulo relé de 2 canais. O BMP280 é um sensor de temperatura e pressão fabricado pela Bosch. O DHT11 é um sensor de temperatura e umidade fabricado pela empresa chinesa Aosong *Electronics* Co. Ltd. O módulo PCF8591 possui entradas analógicas e um sensor de luminosidade e é fabricado pela NXP *Semiconductors*. O módulo relé possui 2 canais e sua alimentação é de 5V. [14][15][16].

As Figuras 4, 5, 6 e 7 mostram os sensores BMP280, DHT11, o módulo PCF8591 e o módulo relé, respectivamente.

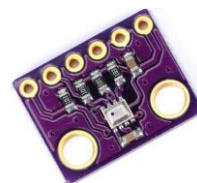


Figura 4 – Sensor BMP280. [14]



Figura 5 – Sensor DHT11. [15]



Figura 6 – Sensor PCF8591. [16]



Figura 7 – Módulo Relé 2 canais.

### C. Linguagem de programação Python e bibliotecas utilizadas

Python é uma linguagem de programação fácil de ser entendida e que permite trabalhar mais rapidamente e integrar sistemas de forma eficaz. Essa linguagem pode ser usada em diversas aplicações, como: *data Science*, *machine learning*, desenvolvimento *web*. [17]

Integrado ao Python existem diversas bibliotecas que facilitam o desenvolvimento e auxiliam na criação dos sistemas. Para execução deste projeto foi necessário utilizar as bibliotecas *flask*, *smbus*, *dht11* e *FreeOPC-UA*, todas as bibliotecas podem ser encontradas na documentação do Python. [17]

- Biblioteca *Smbus* – é utilizada para ler e tratar informações vindas do barramento *I2C* (*Inter-Integrated Circuit*). Esta biblioteca facilita a comunicação com este barramento, devido a sua facilidade de implantação. [18]
- Biblioteca *DHT11* – é utilizada para leitura de dados do sensor *DHT11*. Essa biblioteca é de fácil utilização e auxilia na leitura de temperatura e umidade do sensor. [19]
- Biblioteca *FreeOPC-UA* – é baseada em programação orientada a objetos e é feita para enviar e receber dados e estruturas do protocolo *OPC-UA*, permitindo criar um cliente e um servidor de forma simples. [20]
- Biblioteca *Flask* – é um *framework* para criar aplicativos *web* através de servidores *HTTP* e é ideal para quem busca aplicações simples e rápidas. Com essa biblioteca é possível criar servidores *HTTP* e utilizar requisições do tipo *GET* e *POST*. [21]

O método *GET* é utilizado quando precisa enviar dados do servidor. No caso da aplicação proposta o *GET* é usado para enviar os dados do servidor *HTTP* para a *web*. Já o método *POST* é utilizado quando é necessário enviar dados para o servidor. Neste projeto utiliza-se a requisição *POST* para acionamento de carga remotamente. [22]

A Figura 8 é exemplifica o funcionamento desses métodos.



Figura 8 – Requisições *HTTP GET* e *POST*. [22]

### D. Javascript

JavaScript é uma linguagem de programação baseada em objetos, criada principalmente para desenvolvimento de *frontend web*, neste caso é usado juntamente com *HTML* e *CSS* (*Cascading Style Sheets*). Com Javascript é possível criar páginas *web* mais dinâmicas, pois existem diversos *frameworks* nessa linguagem que facilitam a criação de elementos dinâmicos na página, como: gráficos, tabelas, formulários. [23]

### E. Bootstrap

Bootstrap é um *framework web* gratuito e *open source* que oferece padrões para desenvolvimento *HTML*, Javascript e *CSS*, permite criação de projetos para dispositivos móveis na *web* com biblioteca de componentes *frontend*. Se encontra na versão 4, contando com ampla gama de *plugins* em JavaScript/JQuery que facilita a implementação de plataformas *web*. [24]

Suas principais vantagens são a facilidade de utilização, pois não requer muito conhecimento em computação; permite utilização de padrões prontos já pré-codificados e apresenta resposta rápida às requisições; é personalizável, pois pode ser selecionado apenas o que for necessário ao projeto e é *open source*, já que possui código 100% aberto. [24]

### F. Google Charts

O Google *Charts* é um interativo serviço *web* que permite criação de gráficos a partir de informações fornecidas pelo usuário, esses dados são tratados em JavaScript e incorporados numa página *web*, e o serviço envia a resposta em forma gráfica. [25]

Essa biblioteca pode ser considerada como uma das melhores por sua facilidade de utilização e excelente documentação. Com ela é possível desenhar gráficos de linha, pizza, barras, tabelas, entre outros. Neste projeto, foi utilizado o gráfico de linha para visualização dos dados dos sensores em tempo real. [25]

### G. Amazon Web Service (AWS) e Amazon Elastic Compute Cloud (Amazon EC2)

A AWS é um provedor de serviços *online* em *websites* ou aplicações cliente/servidor baseado em plataforma em nuvem. [26]

O Amazon *EC2* é uma instância da AWS, que oferece uma capacidade de computação escalável na nuvem. É uma solução que facilita a aquisição de servidores virtuais. [27]

Dentre as principais vantagens de se utilizar a Amazon *EC2*, tem-se: agilidade de implantação e as aplicações não



saem do ar enquanto a implantação é feita, simplicidade na utilização, disponibilidade dos serviços já que oferece capacidade de escalar sob demanda do usuário. Com o uso da Amazon EC2 elimina-se a necessidade de investir em hardware e é possível desenvolver aplicativos com mais agilidade. [27]

#### IV. APLICAÇÃO PRÁTICA

Neste capítulo são descritos os processos utilizados para o desenvolvimento da aplicação prática, que consiste em um cliente e um servidor OPC-UA, desenvolvidos utilizando a biblioteca FreeOPC-UA comunicando entre os mesmos.

##### A. Arquitetura do sistema

A arquitetura da aplicação é cliente/servidor. O cliente é embarcado em um *raspberry-pi*, onde há cargas e sensores de campo conectados. O cliente coleta os dados e os envia através do protocolo OPC-UA para uma plataforma na nuvem, nesta aplicação foi utilizada a instância EC2 da AWS, onde está embarcado o servidor OPC-UA. Este servidor recebe os dados e os expõe para um servidor HTTP desenvolvido em Python, utilizando o *framework* Flask.

O servidor HTTP envia os dados para *web* utilizando a requisição GET HTTP e o formato JSON (*JavaScript Object Notation*). Na *web* esses dados são tratados e exibidos na tela, este tratamento é feito utilizando o Google Charts. Na aplicação também será possível acionar cargas conectadas no *raspberry*, através da requisição POST HTTP. A página *web* foi criada a partir do *framework* Bootstrap.

Na Figura 9 é possível visualizar a arquitetura da aplicação.

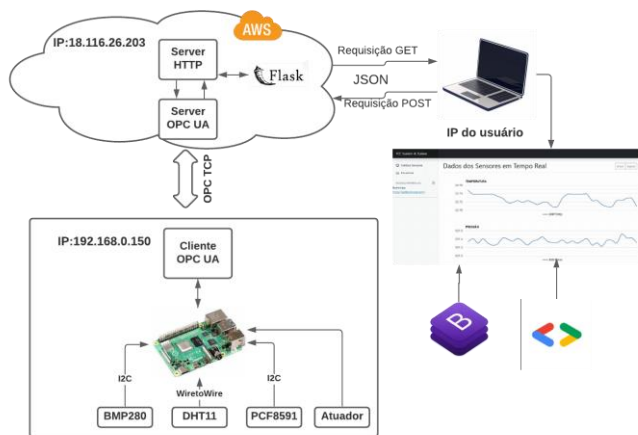


Figura 9 – Arquitetura do sistema.

##### B. Desenvolvimento feito no Raspberry-Pi 4

O *Raspberry-pi* 4 é uma placa utilizada para desenvolvimento. Seu sistema operacional é o Raspbian, baseado em Linux e linha de comando. No *raspberry* está embarcado o Cliente OPC-UA, com os sensores e módulos conectados.

O Cliente OPC-UA foi desenvolvido a partir da versão 3.9 da linguagem Python, assim se fez necessário configurar o *raspberry* com a mesma versão. Utilizou-se a biblioteca

FreeOPC-UA, baseada em programação orientada a objetos, com documentação de fácil entendimento. Para utilizá-la é necessário fazer o *download* utilizando o comando: `pip install opcua`. O cliente precisa se comunicar com o servidor para criar os objetos e as variáveis do OPC-UA no servidor, seguido da configuração do cliente de modo a realizar essa comunicação.

A Figura 10 mostra como foi feito a configuração do cliente e variáveis no projeto.

```
#Informações vindas do servidor
datasensor = OPCClient.get_root_node()
date = datasensor.get_child(["0:Objects", "2:MyObject", "2:MyDataDatetime"])
sensor_Temp1 = datasensor.get_child(["0:Objects", "2:MyObject", "2:Temp1"])
sensor_Temp2 = datasensor.get_child(["0:Objects", "2:MyObject", "2:Temp2"])
sensor_Press = datasensor.get_child(["0:Objects", "2:MyObject", "2:Press1"])
Poti = datasensor.get_child(["0:Objects", "2:MyObject", "2:Poti1"])
sensor_Umidade = datasensor.get_child(["0:Objects", "2:MyObject", "2:Umidade1"])
atuador = datasensor.get_child(["0:Objects", "2:MyObject", "2:Atuador1"])
```

Figura 10 – Configuração das variáveis dos sensores no Cliente OPC-UA.

No código também é necessário apontar o endereço IP (*Internet Protocol*) do servidor na nuvem e a porta TCP (*Transmission Control Protocol*) do protocolo OPC-UA, para que o cliente consiga comunicar com o mesmo.

A Figura 11 mostra como é feita essa configuração.

```
OPCClient = Client("opc.tcp://18.116.26.203:4840")
```

Figura 11 – Configuração de rede no cliente OPC-UA.

Na placa também estão conectados a carga e os sensores do sistema. O sensor BMP280 e o módulo PCF8591 são conectados no *raspberry* via protocolo I2C e o tratamento dos dados é feito utilizando a biblioteca *smbus* do python.

O sensor DHT11 é ligado ao pino 24 do *raspberry*, onde é configurado como entrada, para tratamento dos dados de temperatura e umidade foi utilizado a biblioteca *dht11*, disponível na documentação do python.

No sistema, também é conectado um módulo relé para acionamento de cargas, e este é conectado ao pino 26, configurado como saída.

A Figura 12 mostra a distribuição dos sensores no *raspberry*.

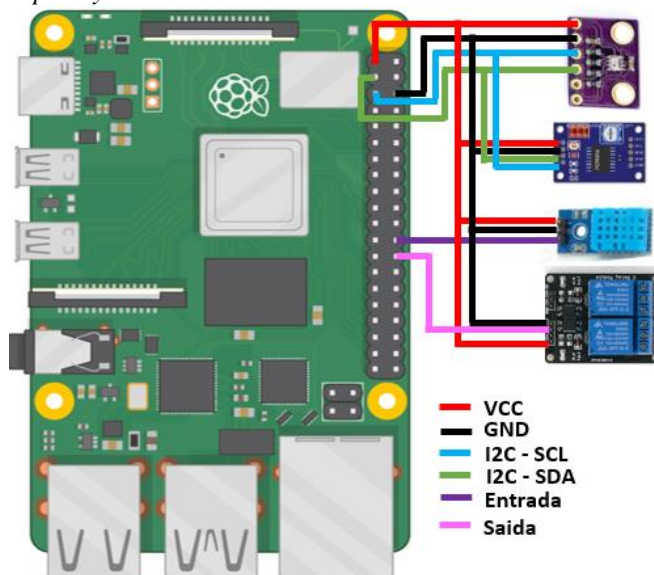


Figura 12 – Ligação dos sensores e módulos conectados no raspberry.

### C. Desenvolvimento na AWS

A AWS é uma plataforma para desenvolvimento em nuvem e oferece diversas ferramentas que podem ser utilizadas. Neste projeto utilizou-se a instância EC2, onde foram embarcados o servidor OPC-UA e o servidor HTTP. O EC2 permite que usuários aluguem um computador virtual para rodar suas próprias aplicações.

Nesse projeto foi alugado uma instância de 24GiB de HD e 4GiB de RAM, com IOPS (*Input/Output Operations Per Second*) de 100 dados de entrada e saída por segundo. [28]

A Figura 13 mostra a instância EC2 configurada.




Detalhes	
ID de volume	Tamanho
 vol-0b2bff38aa2d3f840	 24 GiB
Estado do volume	IOPS
 Em uso	100

Figura 13 – Detalhes da instância utilizada.

Na instância alugada também foi necessário configurar a rede para que houvesse acesso à internet e o cliente e servidor pudessem comunicar entre si. [28]

Na Figura 14 é mostrado a configuração de rede e os IP's disponíveis.

Instância: i-0533183dbc7e81471 (Instância TCC SuelemDaiane)		
ID de instância i-0533183dbc7e81471 (Instância TCC SuelemDaiane)	Endereço IPv4 público 18.116.26.203   <a href="#">endereço aberto</a>	Endereços IPv4 privados 172.31.20.236
Endereço IPv6 -	Estado da instância  Executando	DNS IPv4 público ec2-18-116-26-203.us-east-2.compute.amazonaws.com   <a href="#">endereço aberto</a>

Figura 14 – Detalhes da configuração de rede da instância.

O servidor OPC-UA e servidor HTTP foram desenvolvidos utilizando a versão 3.9 da linguagem Python, portanto, se fez necessário a configuração da AWS na mesma versão. O servidor OPC-UA foi desenvolvido utilizando a biblioteca FreeOPC-UA. Neste servidor são criadas as variáveis referentes aos sensores e ao atuador do sistema. A Figura 15 apresenta a criação dessas variáveis.

```
# Variáveis criadas - Referentes aos sensores e atuadores
sensores = node.add_object(OPC_server["index"], "MyObject")
sensor_Temp1 = sensores.add_variable(OPC_server["index"], "Temp1", 0)
sensor_Temp2 = sensores.add_variable(OPC_server["index"], "Temp2", 0)
sensor_Press = sensores.add_variable(OPC_server["index"], "Press1", 0)
Pot1 = sensores.add_variable(OPC_server["index"], "Pot1", 0)
sensor_Umidade = sensores.add_variable(OPC_server["index"], "Umidade1", 0)
atuador = sensores.add_variable(OPC_server["index"], "Atuador1", 0)
myDataDatetime = sensores.add_variable(OPC_server["index"], "MyDataDatetime", None)
```

Figura 15 – Variáveis criadas no servidor OPC-UA

Com cliente e o servidor OPC-UA configurados pode-se estabelecer comunicação entre eles e o servidor já consegue ler os dados vindos dos sensores no *raspberrypi*.

Após a coleta dos dados dos sensores, o servidor OPC-UA expõe esses dados para o servidor HTTP e então, é executada uma API (*Application Programming Interface*) simples, onde

existem duas rotas HTTP, *GET* e *POST*. Na rota *GET* os dados são formatados e enviados no formato JSON para o cliente, que neste caso é o navegador *web* do dispositivo que está acessando a página *web*. Os dados são coletados e os dados são atualizados na página *web* a cada 2 segundos.

A Figura 16 apresenta a rota que faz o envio do JSON para *web*.

```
@app.route('/sensor_value')
def getWebSensorData():
    global OPC_server
    BMPTemp1 = OPC_server["server"].get_node("ns=2;i=2").get_value()
    BMPPress = OPC_server["server"].get_node("ns=2;i=3").get_value()
    DHT11Temp1 = OPC_server["server"].get_node("ns=2;i=4").get_value()
    Potenciometro = OPC_server["server"].get_node("ns=2;i=5").get_value()
    DHT11_Umi = OPC_server["server"].get_node("ns=2;i=6").get_value()
    atuador = OPC_server["server"].get_node("ns=2;i=7")
    Data = OPC_server["server"].get_node("ns=2;i=8").get_value().timestamp()
    if(Data == None):
        print('Aguardando Conexão Client')
    if DHT11_Umi == 0:
        DHT11_Umi = OPC_server["server"].get_node("ns=2;i=6").get_value()

    sensor_data = {
        'dadosgraficoT': dadosArrayT, #Coloca os dados do vetor no JSON
        'dadosgraficoU': dadosArrayU,
        'dadosgraficoP': dadosArrayP,
        'dadosgraficoL': dadosArrayL,
        'myDate': Data #variavel que será encapsulada no json
    }

    return jsonify(sensor_data)
```

Figura 16 – Rota *GET* HTTP para envio dos dados para *web*.

Este JSON é interpretado no *frontend web*, onde esses dados são processados pelo javascript e posteriormente, com auxílio do *Google Charts* os gráficos são gerados e atualizados em tempo real, assim permitindo a visualização destes dados ao longo do tempo.

A página *web* foi desenvolvida utilizando HTML e a biblioteca chamada *bootstrap* para montagem do *layout* e *design* da página dos dados. Para acessar a página *web* deve-se utilizar o seguinte endereço: <http://18.116.26.203:5000/index>.

A rota *POST* é responsável pelo acionamento remoto das cargas conectadas ao *raspberrypi*. Nesta rota o corpo da requisição em formato JSON é interpretado para determinar quais acionamentos devem ser realizados através do protocolo OPC-UA, além de fornecer botões para o controle da carga útil no *raspberrypi*.

A Figura 17 apresenta a rota do *POST* e as configurações para acionamento da carga.

```
@app.route('/atuador', methods=['POST'])
def setActuator():
    global atuador

    if request.is_json:
        payload = request.get_json()

        if payload["shouldEnable"] is not None:
            atuador.set_value(1 if payload["shouldEnable"] else 0)
            return jsonify({'msg': "Success", 'status': payload["shouldEnable"]})
        else:
            return "Invalid value", 400
    else:
        return "No JSON found", 400
```

Figura 17 – Rota *POST* HTTP para acionamento da carga.

### V. TESTES E RESULTADOS

Neste capítulo é demonstrado as etapas de teste do projeto, a fim de comprovar o funcionamento dos sensores e módulos

instalados no sistema. Foram definidos dois métodos para testar a aplicação, são eles: teste de funcionalidade dos sensores e teste fim-a-fim, para verificar a comunicação entre o servidor HTTP e a API em execução.

O primeiro método de teste realizado tem a função de verificar o funcionamento dos sensores e módulos. Os testes foram realizados alterando as condições dos sensores e módulos, de modo a visualizar a alteração dos valores de temperatura, umidade, pressão e verificar o acionamento da carga instalada.

A primeira variável de processo a ser testada é a temperatura, e para isso foi utilizado os sensores BMP280 e DHT11, ambos de temperatura. A Figura 18 apresenta a medição de temperatura desses sensores coletada no *raspberry* em um determinado horário.

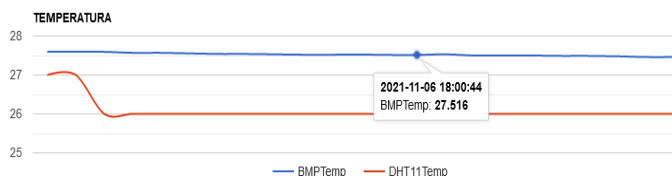


Figura 18 – Gráfico com dados de temperatura em tempo real.

Para validar o funcionamento, os sensores foram aquecidos a fim de visualizar a alteração da temperatura no gráfico. Nas Figuras 19 e 20 é possível visualizar a variação de temperatura dos sensores.

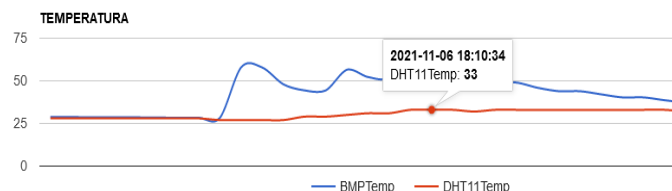


Figura 19 – Variação de temperatura no sensor DHT11.

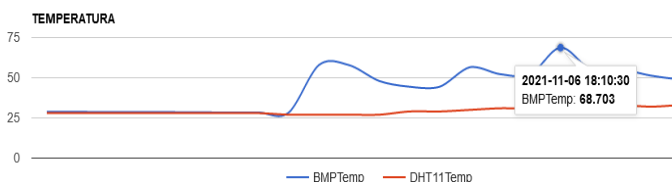


Figura 20 – Variação de temperatura no sensor BMP280.

Nos gráficos é possível observar que o BMP280 tem uma resposta mais rápida e é mais preciso comparado ao DHT11, por estar conectado ao barramento I2C.

O próximo teste foi feito com a variável umidade e utilizando o sensor DHT11 para verificar o funcionamento. Na Figura 21 é possível visualizar o sensor de umidade antes de ocorrer a variação.

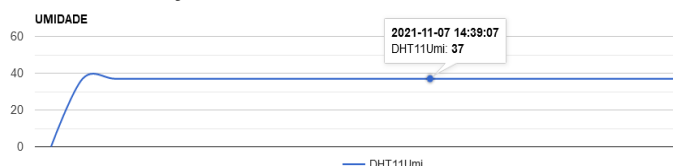


Figura 21 – Dados de umidade em tempo real.

Para validação do funcionamento, alterou-se a umidade próximo ao sensor, aquecendo-o. A Figura 22 apresenta a variação de umidade.

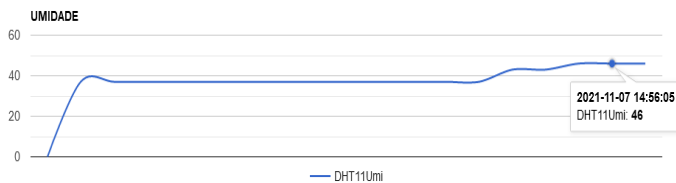


Figura 22 – Variação de umidade no sensor DHT11.

O próximo teste é referente a pressão atmosférica e é realizado com o BMP280 medindo a pressão. Como a leitura é feita pelo barramento I2C é possível visualizar uma pequena variação no gráfico. Na Figura 23 é possível visualizar as medidas realizadas por esse sensor.



Figura 23 – Variação de pressão no sensor BMP280.

Utilizando o módulo PCF8591 e um potenciômetro ligado a entrada analógica deste módulo variou-se a resistência e houve uma variação no gráfico. Na Figura 24 é possível visualizar a variação deste potenciômetro.

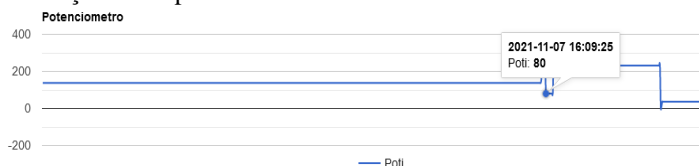


Figura 24 – Variação de resistência no potenciômetro do PCF8591.

O último teste de funcionamento, é realizado com o módulo relé para verificar o acionamento da carga conectada ao mesmo. Este acionamento é realizado através da *dashboard* na *web* e utiliza o método de requisição *POST*. A Figura 25 mostra o módulo relé antes de ativar a carga IN1.



Figura 25 – Módulo relé antes da ativação da carga.

A Figura 26 mostra o modulo relé após o acionamento da carga IN1.

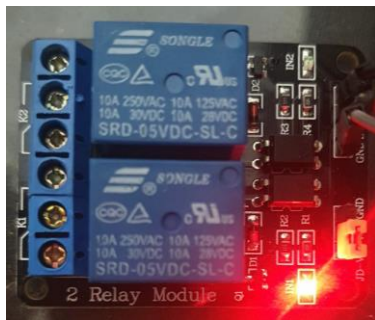


Figura 26 – Carga IN1 ativada.

O teste fim-a-fim foi realizado utilizando o *Postman*, um *software* para teste de API. Primeiramente testou-se a rota que retorna o JSON com os dados dos sensores, para isso foi necessário enviar uma requisição *GET* no endereço do servidor com o *endpoint* *sensor\_value*, referente a rota do envio dos dados. Na Figura 27 é possível visualizar a configuração feita no *software* para enviar essa requisição.

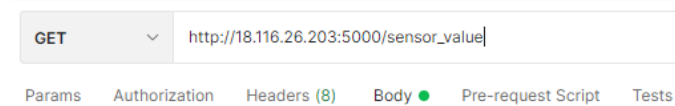


Figura 27 – Configuração do *Postman* para enviar requisição *GET*.

Para validar o funcionamento da comunicação o *Postman* retorna o JSON com as informações do sensor e o status 200 OK, indicando que a requisição foi bem-sucedida. A Figura 28 apresenta a resposta dessa requisição, retirada do *Postman*.

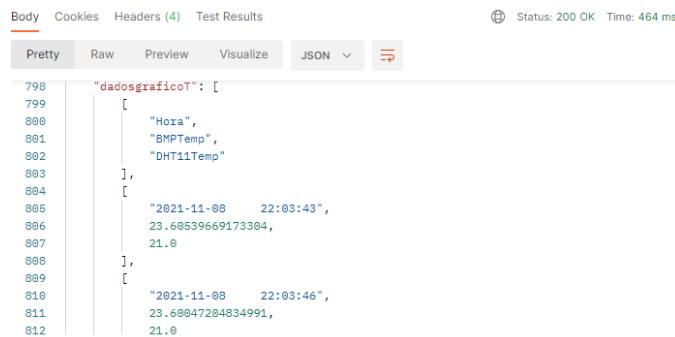


Figura 28 – Resposta da requisição *GET* na rota *sensor\_value*.

O próximo teste fim-a-fim foi feito com a rota que retorna a página *web*, para a realização do teste foi necessário configurar o *endpoint* para *index*. A Figura 29 mostra a configuração feita no *software* para esse teste.

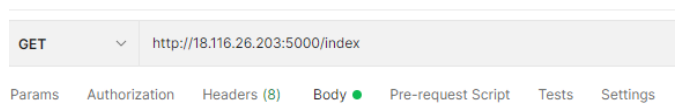


Figura 29 – Configuração do *Postman* para enviar essa requisição.

O *Postman* retorna o código fonte da página *web* e o status 200 OK, indicando que está funcionando corretamente. A Figura 30 mostra a resposta a essa requisição.

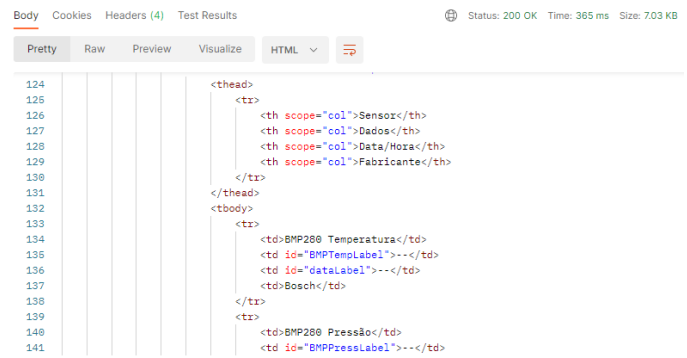


Figura 30 – Resposta da requisição *GET* na página *web*.

Para validar a requisição *POST* que é responsável pelo acionamento da carga, também foi utilizado o *Postman*, agora configurado para enviar um *POST* ao *endpoint* *actuator*. Para envio dessa requisição é necessário preencher o *body* com um JSON indicando a ação que deve ser aplicada no atuador, neste caso a carga foi acionada e a ação foi *true* na variável *shouldEnable*. A Figura 31 mostra a configuração realizada.

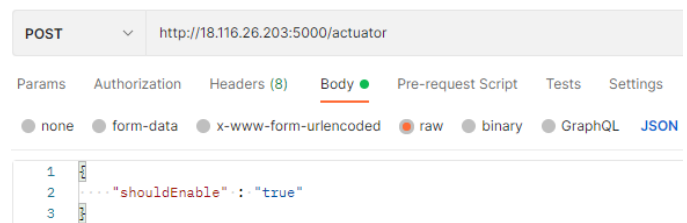


Figura 31 – Configuração do *Postman* para envio da requisição *POST*.

A aplicação foi programada de modo que ao receber uma requisição *POST* retorne um JSON com uma mensagem de sucesso e o status. A Figura 32 mostra a resposta contendo o JSON.

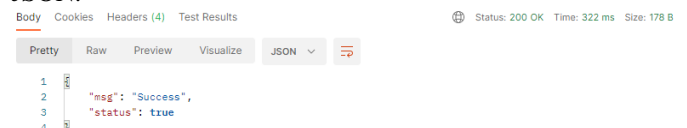


Figura 32 – Resposta da requisição *POST* na rota *actuator*.

Os testes realizados tiveram um resultado satisfatório e pôde-se testar o funcionamento de todos os sensores e validar a comunicação fim-a-fim, mostrando que o servidor HTTP e a aplicação estão funcionando corretamente.

## VI. CONCLUSÃO

Com o avanço das indústrias e a chegada de diversas tecnologias é cada vez mais necessário o desenvolvimento de aplicações inteligentes para utilização em ambientes industriais, seja na área produtiva, na área de testes ou na manutenção industrial. Neste projeto, buscou-se desenvolver uma aplicação de baixo custo utilizando conceitos, como: IoT, computação em nuvem e o protocolo OPC-UA que facilitam o desenvolvimento de aplicações mais inteligentes e conectadas para determinadas áreas da indústria.

A integração permitiu que os dados de campo fossem disponibilizados em uma aplicação *web*. O uso de *frameworks*,



como: Bootstrap e *Flask* e as bibliotecas do Python, facilitaram o desenvolvimento da aplicação, pela simplicidade de implementação e integração.

Diante dos testes realizados e resultados obtidos, foi observado que a aplicação é satisfatória diante do quadro ao qual foi proposto visto que atingiu o objetivo de facilitar o acesso e monitoramento de variáveis reais e em tempo real, utilizando uma aplicação IoT.

#### REFERÊNCIAS

- [1] REVOLUÇÃO INDUSTRIAL. O que é a 4ª revolução industrial. Disponível em: < <https://www.bbc.com/portuguese/geral-37658309>>. Acesso em: 15 ago. 2021.
- [2] INDÚSTRIA 4.0. Industry 4.0: The Future of Productivity and Growth in Manufacturing Industries. Disponível em: < [https://www.bcg.com/pt-br/publications/2015/engineered\\_products\\_project\\_business\\_industry\\_4\\_future\\_productivity\\_growth\\_manufacturing\\_industries](https://www.bcg.com/pt-br/publications/2015/engineered_products_project_business_industry_4_future_productivity_growth_manufacturing_industries)>. Acesso em: 17 ago. 2021.
- [3] IoT - O que é IoT. Disponível em: <<https://www.oracle.com/br/internet-of-things/what-is-iot/>>. Acesso em: 27 ago. 2021.
- [4] IoT. Uma introdução à Internet da Coisas. Disponível em: <[https://www.cisco.com/c/dam/global/pt\\_br/assets/brand/iot/iot/pdfs/lopez\\_research\\_an\\_introduction\\_to\\_iiot\\_102413\\_final\\_portuguese.pdf](https://www.cisco.com/c/dam/global/pt_br/assets/brand/iot/iot/pdfs/lopez_research_an_introduction_to_iiot_102413_final_portuguese.pdf)>. Acesso em: 27 ago. 2021.
- [5] COMPUTAÇÃO EM NUVEM. História e o Futuro da Computação em Nuvem: Como Surgiu Cloud Computing. Disponível em: <[https://www.dell.com/learn/br/pt/brsdt1/sb360/social\\_cloud](https://www.dell.com/learn/br/pt/brsdt1/sb360/social_cloud)>. Acesso em 10 set. 2021.
- [6] PINTO, G. S.; IRION, C. Cloud Computing e Datacenter. 2013. Trabalho de conclusão de curso de pós-graduação. 2013. Trabalho de conclusão de Engenharia de Redes e Tecnologia da Informação – Instituto Nacional de Telecomunicações, Santa Rita do Sapucaí, 2013, 7p.
- [7] PEREIRA JUNIOR, T. J. Internet das Coisas e sua evolução tecnológica para as cidades inteligentes. 2016. Trabalho de conclusão de curso de Engenharia de Redes e Tecnologia da Informação – Instituto Nacional de Telecomunicações, Santa Rita do Sapucaí, 2016, 9p.
- [8] COUTO, E. A.; COELHO J. R. U. Supervisão de sistema de controle difuso embarcado via Node-RED. 2021. Trabalho de conclusão de curso de Engenharia de Controle e Automação – Instituto Nacional de Telecomunicações, Santa Rita do Sapucaí, 2021, 9p.
- [9] SUPERVISÓRIO. O que são sistemas supervisórios. Disponível em: < <https://kb.elipse.com.br/o-que-sao-sistemas-supervisórios/>>. Acesso em 29 set. 2021.
- [10] OPC UA. UA Companion Specifications. Disponível em: <<https://opcfoundation.org/about/opc-technologies/opc-ua/ua-companion-specifications/>>. Acesso em: 05 ago. 2021.
- [11] MARQUES NETO, Nery Silva. Desenvolvimento e teste de uma plataforma de comunicação de baixo custo para planta laboratorial. 2020. Trabalho de conclusão de curso de Engenharia de Controle e Automação – Universidade Federal do Rio Grande do Sul, Porto Alegre, 2020, 45p.
- [12] RASPBERRY PI 4. Raspberry Pi 4. Disponível em: < <https://www.raspberrypi.org/products/raspberry-pi-4-model-b/>>. Acesso em: 20 set. 2021.
- [13] ANTONIOLLI, Alessandro. Sistema de monitoramento automatizado para controle de qualidade de água em sistema aquapônico. 2019. Trabalho de conclusão de curso de Engenharia da Computação – Universidade do Vale do Taquari, Lajeado, 2019, 95p.
- [14] SENSOR BMP280. BMP280. Disponível em: < <https://www.bosch-sensortec.com/products/environmental-sensors/pressure-sensors/bmp280/>>. Acesso em: 20 set. 2021.
- [15] SENSOR DHT11. Temperature and humidity module. Disponível em: < [https://s3-sa-east-1.amazonaws.com/robocore-lojavirtual/791/Datasheet\\_DHT11.pdf](https://s3-sa-east-1.amazonaws.com/robocore-lojavirtual/791/Datasheet_DHT11.pdf)>. Acesso em: 20 set. 2021.
- [16] MÓDULO I2C PARA RASPBERRY.PCF8591. Disponível em: < <https://www.nxp.com/docs/en/data-sheet/PCF8591.pdf> >. Acesso em: 20 set. 2021.
- [17] LINGUAGEM DE PROGRAMAÇÃO PYTHON. Python is powerful and fast; plays well with others; runs everywhere; is friendly & easy to learn; is Open. Disponível em: < <https://www.python.org/about/>>. Acesso em: 22 set. 2021.
- [18] I2C PYTHON. Smbus2 0.4.1. Disponível em: < <https://pypi.org/project/smbus2/>> Acesso em: 22 set. 2021
- [19] DHT11 PYTHON3. DHT11 0.1.0. Disponível em: < <https://pypi.org/project/dht11/>> Acesso em: 22 set. 2021
- [20] OPC-UA PYTHON. Python OPC-UA Documentation. Disponível em: < <https://python-opcua.readthedocs.io/en/latest/>> Acesso em: 22 set. 2021
- [21] FLASK. Flask 2.0.2 Disponível em: < <https://pypi.org/project/Flask/>>. Acesso em: 22 set. 2021
- [22] REQUISITOS HTTP FLASK. HTTP Methods Disponível em: < <https://flask.palletsprojects.com/en/2.0.x/quickstart/#routing>>. Acesso em: 22 set. 2021
- [23] JAVASCRIPT. JavaScript. Disponível em: < <https://developer.mozilla.org/pt-BR/docs/Web/JavaScript> >. Acesso em: 22 set. 2021
- [24] BOOSTRAP. Bootstrap. Disponível em: < <https://getbootstrap.com.br/> > Acesso em: 25 set. 2021
- [25] GOOGLE CHARTS. Interactive charts for browsers and mobile devices. Disponível em: < <https://developers.google.com/chart> >. Acesso em: 25 set. 2021.
- [26] AWS. Computação em nuvem com a AWS. Disponível em: < [https://aws.amazon.com/pt/what-is-aws/?nc1=f\\_cc](https://aws.amazon.com/pt/what-is-aws/?nc1=f_cc) >. Acesso em: 25 set. 2021.
- [27] INSTÂNCIAS AWS. O que é o Amazon EC2? Disponível em: < [https://docs.aws.amazon.com/pt\\_br/AWSEC2/latest/UserGuide/concepts.html](https://docs.aws.amazon.com/pt_br/AWSEC2/latest/UserGuide/concepts.html) >. Acesso em: 25 set. 2021.
- [28] EC2 AWS. Configuração para usar o Amazon EC2. Disponível em: <[https://docs.aws.amazon.com/pt\\_br/AWSEC2/latest/UserGuide/get-set-up-for-amazon-ec2.html](https://docs.aws.amazon.com/pt_br/AWSEC2/latest/UserGuide/get-set-up-for-amazon-ec2.html) > Acesso em: 15 out. 2021.