REVISTA BRASILEIRA DE MECATRÔNICA
FACULDADE SENAI DE TECNOLOGIA MECATRÔNICA

DESENVOLVIMENTO DE UM *GATEWAY* PARA OS PROTOCOLOS OPC-UA E MQTT

DEVELOPMENT OF A GATEWAY FOR THE OPC-UA AND MQTT PROTOCOLS

Robson Vieira Soares [1,i]
Douglas da Serra Ogata *(in memoriam)* [2,ii]
André Luis dos Santos [3,iii]
Paulo André dos Santos [4,iv]
Pedro André Braga de Oliveira [5,v]

Data de submissão: (05/02/2024) Data de aprovação: (14/12/2024)

RESUMO
O presente trabalho baseia-se na Indústria 4.0 e mostra a interação entre os ambientes de automação industrial e Internet das Coisas através de um *gateway* que atua como intermediário entre dois protocolos, utilizando OPC-UA para coleta e MQTT para transporte de dados. Essa integração é realizada através de uma aplicação desenvolvida em linguagem de programação Python conectada a um *broker* em nuvem, permitindo que as informações do ambiente industrial podem ser monitoradas através de aplicativos para celular, como o MQTTDash. Apesar de não configurados na aplicação apresentadas, os protocolos disponibilizam mecanismos de autenticação e criptografia, assegurando a integridade e confidencialidade dos dados. Espera-se que este trabalho possa auxiliar no desenvolvimento de soluções voltadas à Indústria 4.0 e à Internet das Coisas que necessitem de acesso remoto aos dados de plantas industriais.

Palavras-chave: Indústria 4.0; Internet das Coisas; OPC-UA; MQTT; g*ateway*.

ABSTRACT
This work is based on Industry 4.0 and demonstrates the interaction between industrial automation environments and the Internet of Things through a gateway that acts as an intermediary between two protocols. It utilizes OPC-UA for data collection and MQTT for data transport. This integration is achieved through an application developed in the Python programming language, connected to a cloud-based broker, allowing information from the

[1] Especialista em Automação e Controle pelo Centro Universitário SENAI São Paulo – Campus “Mariano Ferraz”, e-mail: robson.vieira@senaisp.edu.br
[2] Docente no Centro Universitário SENAI São Paulo – Campus “Mariano Ferraz”, e-mail: douglas.ogata@sp.senai.br
[3] Docente no Centro Universitário SENAI São Paulo – Campus “Mariano Ferraz”, e-mail: andre.lsantos@sp.senai.br
[4] Docente no Centro Universitário SENAI São Paulo – Campus “Mariano Ferraz”, e-mail: paulo.andre@sp.senai.br
[5] Docente no Centro Universitário SENAI São Paulo – Campus “Mariano Ferraz”, e-mail: pedro.braga@sp.senai.br

industrial environment to be monitored via mobile applications, such as MQTTDash. Although not configured in the presented application, the protocols provide authentication and encryption mechanisms, ensuring the integrity and confidentiality of the data. It is hoped that this work can assist in the development of solutions focused on Industry 4.0 and the Internet of Things that require remote access to data from industrial plants.

Keywords: Industry 4.0; Internet of Things; OPC-UA; MQTT; gateway.

1. INTRODUÇÃO

Para se manterem competitivas em um mercado globalizado, as empresas são submetidas a uma pressão cada vez maior sobre preço, qualidade e funcionalidade de seus produtos. Para mitigar essas demandas e atender as necessidades cada vez mais diversificadas dos consumidores, elas têm buscado adotar tecnologias mais avançadas que permitam melhorar seus processos produtivos (Yang et. al., 2018). Os avanços recentes nas tecnologias de automação e de informação têm permitido a criação de sistemas de manufatura inteligentes, reconfiguráveis e flexíveis, que possibilitam um aumento da competitividade das empresas e o atendimento às necessidades dos consumidores.

A adoção de novas tecnologias nos processos produtivos levou ao surgimento de uma quarta revolução industrial, também chamada de Indústria 4.0 (I4.0). O *Big Data*, sistemas ciber-físicos (CPS – *Cyber-physical systems*), Internet das Coisas (IoT), tecnologias de informação e comunicação (TIC) e computação em nuvem são tecnologias chave para a criação de sistemas de manufatura inteligentes (Kumar *et. al*, 2019, p. 1, 2).

O termo Indústria 4.0 surgiu pela primeira vez na feira de Hannover de 2011. Teve apoio do governo alemão com intuito de aumentar a competitividade da indústria no mercado globalizado. Seu objetivo foi promover a integração dos sistemas ciber-físicos nos processos industriais e permitindo a criação de sistemas de manufatura flexíveis que possam atender às necessidades dos consumidores (Popkova; Ragulina; Bogoviz, 2019).

A IoT possibilita a coleta e o compartilhamento de dados em tempo real entre os diversos elementos envolvidos nos processos industriais (Kumar *et. al*., 2019). Os dados desempenham um papel importante, pois eles podem ser transformados em informações e, depois, transformados em conhecimento. Estes conhecimentos aumentam a capacidade de adaptação, de tomada de decisão, e de otimização dos processos, resultando em aumento de produtividade, lucro e eficiência operacional (Gilchrist, 2016).

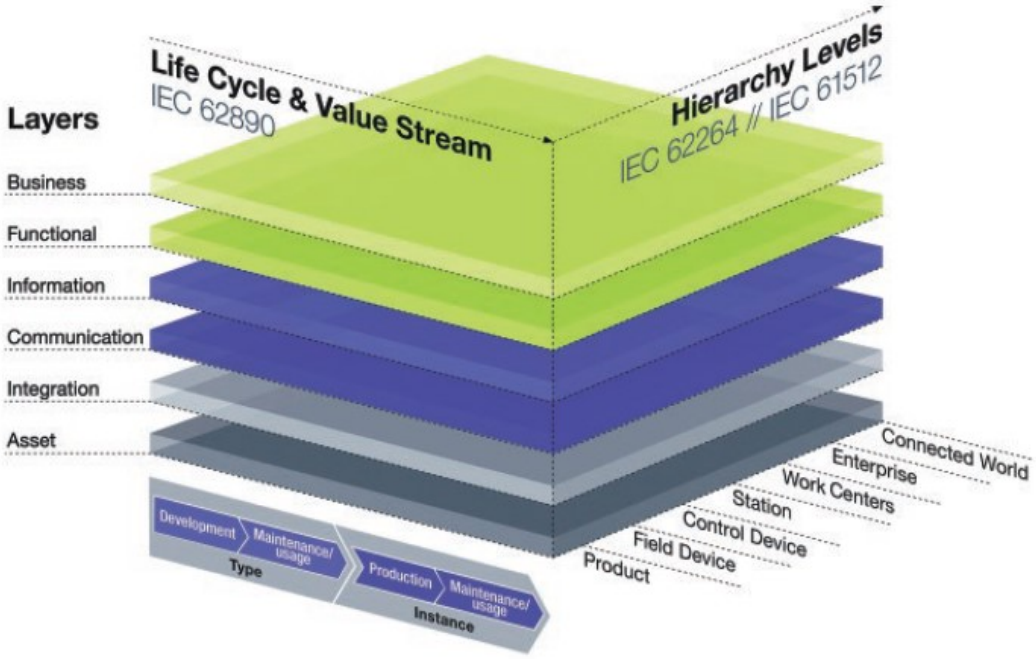
O desenvolvimento de um sistema de manufatura inteligente depende do emprego de diversas tecnologias. A integração destas diferentes tecnologias apresenta diversos desafios à sua implementação. Portanto, para identificar as tecnologias a serem aplicadas e direcionar as decisões tomadas para a criação de um sistema de manufatura inteligente, deve ser adotada uma abordagem sistematizada. Para atingir este objetivo, algumas arquiteturas de referência foram criadas para o desenvolvimento de soluções voltadas à I4.0 e, também, para soluções de IoT.

Para a implementação da I4.0, uma das arquiteturas de referência que se destaca é o modelo de arquitetura de referência para a Indústria 4.0 (RAMI 4.0 – *Reference Architectural Model for Industrie* 4.0), criado na Alemanha.

O RAMI 4.0 é um modelo tridimensional que contempla os principais aspectos envolvidos no desenvolvimento de uma solução para I4.0. O eixo horizontal esquerdo trata dos aspectos relativos ao ciclo de vida do produto. O eixo horizontal direito relaciona os locais

onde os elementos se encontram dentro da estrutura das fábricas. E o eixo vertical é dividido em seis camadas que descrevem os vários aspectos que compõem as entidades físicas e virtuais do sistema (Adolphs *et. al*., 2015). O RAMI 4.0 é apresentado na figura 1.

Figura 1 - Modelo de Referência de Arquitetura para a Indústria 4.0 (RAMI 4.0)



Fonte: Adolphs *et al*. (2015)

Dentre as várias arquiteturas propostas para o desenvolvimento de uma solução para IoT, uma que se destaca é composta de 5 camadas: Percepção, Transporte, Processamento, Aplicação e Negócio (Sethi; Sarangi, 2017).

Nesta arquitetura, as funções executadas pelas camadas são as seguintes:

- **Percepção** – Nesta camada estão os sensores que coletam informações sobre o ambiente.
- **Transporte** – Realiza a troca de dados entre a camada de percepção e a camada de processamento por meio de redes wireless, 3G, LAN, Bluetooth, RFID ou NFC.
- **Processamento** – Responsável pelo armazenamento, análise e processamento dos dados vindos da camada de transporte. Esta camada gerencia e fornece serviços para as camadas inferiores. Esta camada emprega tecnologias como bancos de dados, computação em nuvem e processamento de *Big Data*.
- **Aplicação** – Responsável por fornecer serviços de aplicação específicos aos usuários. É nesta camada que são desenvolvidas as aplicações específicas onde a solução de IoT será empregada como, por exemplo, casas inteligentes, cidades inteligentes e fábricas inteligentes.
- **Negócio** – Responsável pelo gerenciamento do sistema de IoT como um todo, incluindo as aplicações, os modelos de negócio e a privacidade dos usuários (Sethi; Sarangi, 2017).

Apesar de serem desenvolvidas para atender a soluções diferentes, I4.0 e IoT, as duas arquiteturas apresentadas apresentam algumas similaridades. Os sensores estão localizados

nas camadas de ativo do RAMI 4.0 e na camada de percepção da arquitetura de IoT. As tecnologias e protocolos de comunicação são tratados pelas camadas de integração e comunicação do RAMI 4.0 e pela camada de transporte da arquitetura de IoT. O tratamento dos dados é realizado pela camada de informação do RAMI 4.0 e pela camada de processamento da arquitetura de IoT. Os serviços são implementados na camada funcional do RAMI 4.0 e na camada de aplicação da arquitetura de IoT. E os aspectos relacionados ao negócio são abordados pelas camadas de negócio presentes nas duas arquiteturas. Estas similaridades evidenciam a semelhança entre as soluções apresentadas para o desenvolvimento dos dois tipos de solução.

Dentro deste contexto, o presente trabalho apresenta uma solução para envio de dados coletados de um equipamento de automação via protocolo OPC-UA para um servidor na internet (*broker*) usando o protocolo MQTT (*Message Queue Telemetry Transport*), desempenhando o papel de um *gateway*, convertendo dados vindos do protocolo OPC-UA para o protocolo MQTT. Desta forma, estes dados poderão ser acessados por qualquer aplicação que possa se comunicar com o *broker* por meio do protocolo MQTT. O acesso a estes dados facilitará o desenvolvimento de sistemas de manufatura inteligentes e flexíveis e, consequentemente, o desenvolvimento de uma solução para a implementação da I4.0.

A grande vantagem do uso de acesso remoto via internet em aplicações industriais é: a possibilidade de as empresas com unidades distribuídas conseguirem acessar, compartilhar, analisar, e processar informações de chão de fábrica em tempo real e com maior agilidade, e a possibilidade de terceirização de serviços técnicos ou administrativos especializados com maior grau de interação entre os parceiros, evitando assim a necessidade de especialistas em seu quadro de funcionários. (Fernandes Júnior, 2009).

Com relação às arquiteturas, o escopo deste trabalho está restrito às camadas de integração e de comunicação do RAMI 4.0 e à camada de transporte da arquitetura para IoT.

O restante deste trabalho está organizado da seguinte maneira. Os objetivos e justificativas estão no restante desta seção 1. A seção 2 apresenta uma revisão bibliográfica dos principais temas relacionados a este trabalho. A seção 3 descreve o desenvolvimento do aplicativo que realizará a função de *gateway* convertendo os dados entre os protocolos OPC-UA e MQTT, permitindo o acesso destes dados de forma remota pela internet. E por fim, a seção 4 apresenta as considerações finais sobre este trabalho.

1.1 Objetivo

O objetivo deste trabalho é desenvolver uma aplicação que realize a função de um *gateway*, convertendo os dados e realizando as ações necessárias para o correto funcionamento da comunicação entre os protocolos OPC-UA e MQTT, permitindo que o usuário possa acessar esses dados remotamente.

Os objetivos específicos deste trabalho são:

- Realizar a leitura e a gravação de dados em um servidor OPC-UA;
- Realizar a leitura e o envio de dados para um servidor MQTT localizado na internet;
- Realizar a troca de dados entre os protocolos OPC-UA e MQTT; e,
- Acessar os dados coletados de forma remota por meio de um aplicativo de celular.

1.2 Justificativa

Para melhorar seus processos, as empresas têm buscado soluções que possam facilitar o acesso aos dados e, desta forma, melhorar o controle e o monitoramento dos equipamentos industriais. A criação de um aplicativo *gateway* usando a linguagem Python e bibliotecas gratuitas para trabalhar com os protocolos OPC-UA e MQTT possibilita uma redução no custo de desenvolvimento de uma aplicação de controle e monitoramento de processos, sendo uma opção atrativa para pequenas empresas. Este trabalho se propõe a apresentar uma solução para que pequenas empresas possam iniciar a adoção das tecnologias de suporte à IoT e à I4.0 e, assim, obter os benefícios que estas tecnologias oferecem. O principal benefício que este projeto pretende demonstrar é a possibilidade de acesso remoto aos dados de um processo industrial por meio da internet.

1.3 Recursos necessários

Para o desenvolvimento deste trabalho, foi necessária a utilização dos seguintes recursos:

- **Python** – Linguagem de programação de alto nível e tipagem forte usada na construção do aplicativo *gateway*;
- **KepserverEX** – Servidor OPC-UA que, neste projeto, simula os dados lidos pelo *gateway*; e
- **MQTTDash** – Aplicativo para celular com sistema operacional Android que realiza a leitura dos dados publicados no *broker*.

2. REVISÃO DE LITERATURA

Este capítulo apresenta uma revisão bibliográfica dos principais temas relacionados a este trabalho. Inicialmente, na seção 2.1, é apresentada, resumidamente, uma definição de Indústria 4.0 com destaque para sistemas ciber-físicos e IoT, que requerem integrações como a abordada neste estudo para serem implementadas. Os protocolos integrados neste trabalho, OPC-UA e MQTT, são abordados nas seções 2.2 e 2.3. Ao final, na seção 2.4, o termo *gateway* utilizado para definir a implementação realizada é explanado.

2.1 Indústria 4.0

A quarta revolução industrial, ou Indústria 4.0 (I4.0), se trata realmente de uma nova revolução industrial, pois tem provocado mudanças profundas nos sistemas econômicos e nas estruturas sociais por meio da adoção de novas tecnologias nos processos de manufatura (Schwab, 2017).

Tecnologias como computação em nuvem, IoT e CPS têm permitido a criação de fábricas inteligentes com processos de manufatura reconfiguráveis, inteligentes e flexíveis que atendem de forma mais eficiente às demandas do mercado global. A IoT desempenha um papel fundamental na criação de sistemas de manufatura inteligentes pois possibilita a comunicação em tempo real entre os diferentes elementos que os compõem (Kumar; Zindani; Davim, 2019).

2.1.1 Sistemas ciber-físicos

Os recentes avanços obtidos nos sistemas de computação e nas comunicações digitais possibilitaram o surgimento de uma internet industrial. As diferenças existentes entre os sistemas de computação e os sistemas de comunicação exigem um conhecimento interdisciplinar para o desenvolvimento de produtos que contemplem, ao mesmo tempo, poder de processamento e capacidade de comunicação (Gilchrist, 2016).

Sistemas embarcados são sistemas de informação incorporados em dispositivos físicos. Os sistemas ciber-físicos (CPS) vão um passo além, pois possuem um grau maior de integração com o ambiente em que operam. Os CPS integram os processos de computação, de comunicação e os físicos. Nos CPS, ocorre uma integração maior entre os processos físicos e os processos computacionais, onde os processos físicos afetam os processos computacionais e vice-versa. Sendo assim, os CPS promovem uma integração entre os mundos físico e virtual (Gilchrist, 2016).

Se, por um lado, os sistemas embarcados enfatizam o componente computacional, os CPS também enfatizam os domínios físico e de comunicação além do domínio computacional. Ao contrário dos sistemas embarcados onde, eventualmente, pode existir capacidade de comunicação, os CPS dependem da capacidade de comunicação para trocar dados com outros dispositivos (Gilchrist, 2016).

2.1.2 IoT

A Internet das Coisas (IoT) representa uma visão em que cada objeto no mundo tem o potencial de se conectar à Internet e fornecer seus dados para gerar *insights* por conta própria ou por meio de outros objetos conectados (Balani; Hathi, 2016).

Qualquer coisa pode ser um objeto da IoT. De um veículo conectado, pode-se entender o comportamento do motorista e os padrões de uso do veículo; de máquinas conectadas, pode-se determinar o momento em que é necessário executar alguma manutenção; em um aeroporto conectado, pode-se descobrir o tempo gasto pelos passageiros para executar os procedimentos operacionais como *check-in* e verificações de segurança e assim melhorar a movimentação de passageiros (Balani; Hathi, 2016).

A IoT não se trata apenas de conectividade, mas de como utilizar os dados dentro do contexto da aplicação para obter *insights* que não poderiam ser obtidos anteriormente. A IoT possibilitou a aquisição de uma grande quantidade de dados brutos que, se não tratados corretamente, são desperdiçados, representando uma oportunidade perdida para a construção de um ambiente inteligente ao nosso redor. Portanto, cada vez mais é necessário processar estes dados, aplicando correlações, de modo a se obter insights e prever os resultados por meio de análises apropriadas (Balani; Hathi, 2016).

2.2 OPC-UA

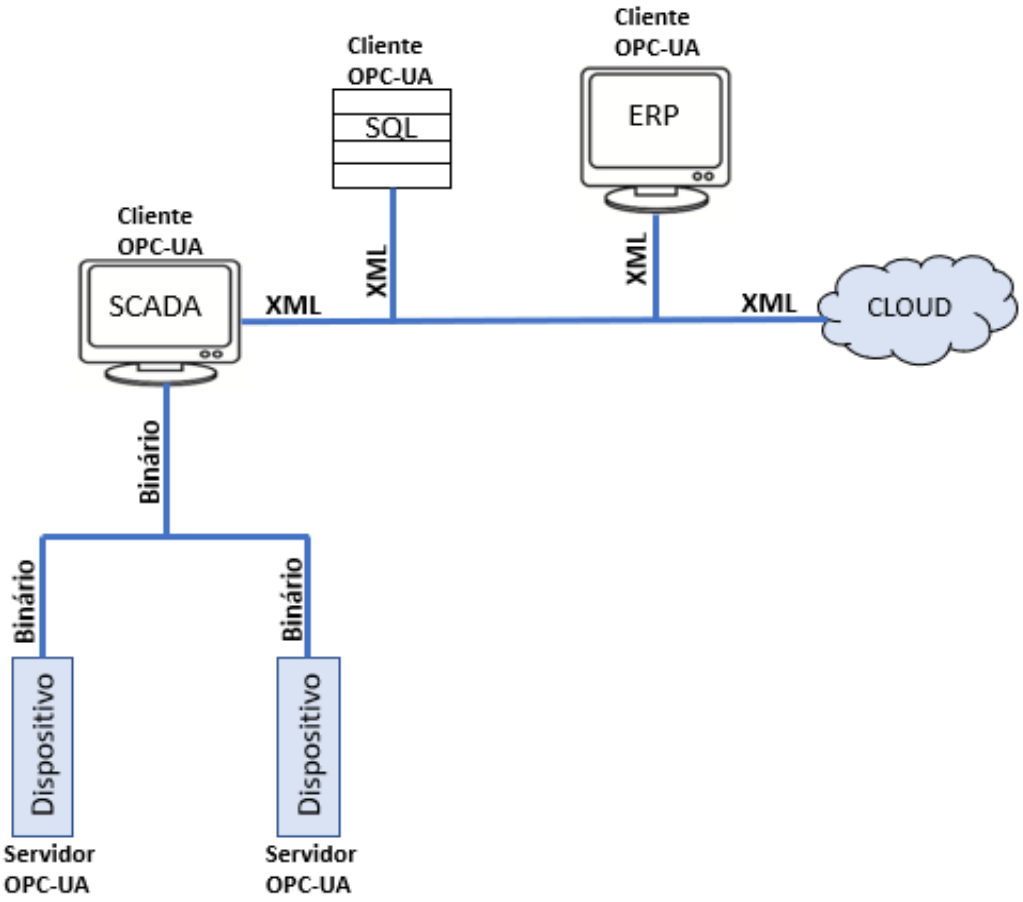
OPC-UA é um protocolo de comunicação industrial e sua sigla significa *Open Platform Communications – Unified Architecture* ou, em português, Plataforma de Comunicação Aberta – Arquitetura Unificada. Ele utiliza um modelo de informação orientado a objetos, que suporta estruturas, objetos e máquinas de estado. Sendo uma plataforma que garante o fluxo de informação contínuo entre os dispositivos de diversos fornecedores (Venturelli, 2021). Este protocolo vem buscando se tornar padrão para a comunicação entre equipamentos industriais

ao contrário de protocolos proprietários que ainda são bastante empregados em sistemas de automação industrial.

São suportados formatos de comunicação binário e XML (*Extensible Markup Language*). O formato binário é simples e de baixo custo, geralmente empregado no nível de dispositivo, onde o poder de processamento é limitado e o desempenho é prioridade. O formato XML é uma linguagem de marcação de dados. Isso facilita a interpretação por parte de diversos dispositivos, independentemente da plataforma e pode utilizar-se de esquemas de SOA (Biondani *et al*., 2017).

A figura 2 mostra um exemplo de comunicação entre dispositivos e software supervisório usando o formato binário e a comunicação entre o software supervisório e um banco de dados, um software ERP e a nuvem (*cloud*) usando a linguagem XML.

Figura 2 - Rede industrial usando protocolo OPC-UA



Fonte: Elaborado pelo autor

Os dados são extremamente protegidos, pois o sistema usa um padrão industrial de certificados digitais X.509 e autenticação de endereços, autorização, criptografia e integridade de dados (Biondani *et al*., 2017).

Com o uso do protocolo OPC-UA consegue-se atender aos requisitos de segurança de dados da I4.0, promovendo o envio de informações para outras camadas da automação e a interação com qualquer dispositivo de chão de fábrica, funcionando em qualquer sistema operacional e plataformas de hardware como PCs e servidores em nuvem. Em resumo, o OPC-UA permite que o usuário aproveite toda a tecnologia moderna que contribui para a criação

de uma fábrica inteligente com visão industrial, inteligência artificial, *machine learning*, manutenção preditiva e mais (Cognex, 2019).

A arquitetura cliente/servidor não é a ideal para soluções de IoT. Para este tipo de aplicação, a arquitetura ideal é a publicador/assinante (*publisher*/*subscriber*), onde o publicador envia dados para seus assinantes e os assinantes também podem enviar dados para o publicador (Venturelli, 2021).

2.3 MQTT

O protocolo MQTT (*Message Queue Telemetry Transport*) foi desenvolvido pela IBM no final dos anos 1990, por Andy Stanford-Clark (IBM) e Arlen Nipper (Cirrus Link, Eurotech). O objetivo da dupla era ter um protocolo leve de dados, com economia de banda e baixo consumo de energia e de hardware para conectar sistemas de telemetria de oleodutos via satélite que, na época, era um recurso extremamente caro (Silveira *et al*., 2021).

Em 2010, a IBM publicou a versão 3.1 do protocolo, a primeira livre de *royalties* que foi, em 2014, padronizada pela *Organization for the Advancement of Structured Information Standards* (OASIS) com o lançamento da versão 3.1.1 e passou por atualizações até chegar a versão atual (OASIS Standard, 2019), lançada em 2019. Com isso, transformou-se oficialmente um padrão aberto com suporte nas linguagens de programação populares e sendo usado em diversas implementações de software livre.

O protocolo MQTT é leve, aberto e foi projetado para ser de fácil implantação. Sendo assim, ideal para comunicação M2M (*Machine to Machine*) e IoT, onde as redes geralmente possuem uma alta latência, pequenas mensagens para serem entregues e onde a largura de banda disponível geralmente é mínima. Esse protocolo é baseado em TCP/IP (*Transmission Control Protocol/Internet Protocol*), possui 3 níveis de QoS (*Quality of Service*) e um sistema de alerta sobre desconexões anormais (OASIS Standard, 2019).

A arquitetura do protocolo MQTT é baseada em tópicos, onde um dispositivo publica uma mensagem e o outro a recebe. O sistema é baseado em duas instâncias, cliente e *broker*. O *broker* é um servidor virtual que armazena todos os dados publicados pelos clientes, separa as informações em tópicos e disponibiliza para quem queira assinar. Não existe identificação do cliente perante o *broker*, seja para publicar ou assinar. Os clientes são aplicações embarcadas em dispositivos ou em sistemas operacionais que utilizam o *broker* como intermediário para enviar dados uns aos outros. As informações podem ser usadas para tomada de decisão humana ou programada (Tietz, 2017).

2.4 Gateway

O termo *gateway* vem da língua inglesa que, traduzido para a língua portuguesa, significa porta de entrada, portão ou portal. Com base nisso, pode-se dizer que um *gateway* é um intermediário ou uma porta de passagem entre dois ambientes distintos (Frenet, 2022). Dentro do contexto de rede, ele atua como uma ponte entre redes distintas, convertendo dados entre dois diferentes sistemas para que possam se entender (Tanenbaum; Wetherall, 2018).

Neste projeto, o *gateway* representa a porta intermediária entre os ambientes industrial e de IoT. Na prática, os dados saem do ambiente industrial e são transportados pelo protocolo OPC-UA até o *gateway*, que os recebe e converte para o protocolo MQTT de modo que possam ser enviados ao *broker* que está na nuvem, como mostra a figura 3.

Figura 3 – Comunicação do *Gateway*

Fonte: Elaborado pelo autor

3. METODOLOGIA

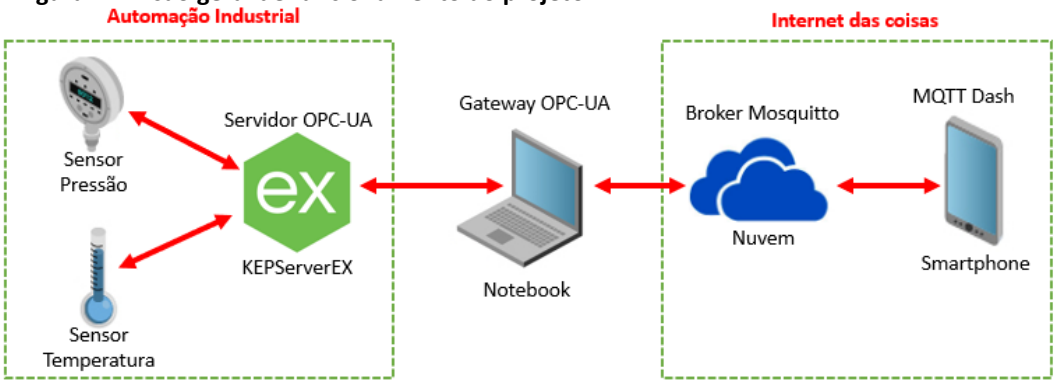
Este projeto tem o objetivo de integrar um ambiente industrial com um ambiente de IoT dentro de um dos pilares da I4.0 que é a computação em nuvem, possibilitando o acesso e monitoramento remoto de dados de um sistema de manufatura industrial. Esse tipo de integração é possível através do uso de um protocolo de automação industrial, o OPC-UA, e de um protocolo de IoT, o MQTT, que interagem através de um *gateway*.

O *gateway* foi construído com uso de uma linguagem de programação de alto nível, muito popular no cenário de desenvolvimento de aplicações, o Python. O uso desta linguagem neste trabalho permitiu a criação de um aplicativo capaz de buscar dados em um servidor OPC-UA e encaminhá-los para um sistema de nuvem formado por um *broker* através do protocolo MQTT.

As informações podem ser acessadas à longa distância por assinantes do tópico no *broker*, desde que o usuário tenha um aplicativo com funcionalidades de visualização de dados em um *broker* MQTT, como o MQTTDash, por exemplo, instalado em seu celular ou em seu computador.

A figura 4 apresenta uma visão geral do funcionamento do projeto. O lado esquerdo representa um ambiente industrial formado por sensores mandando informações ao servidor OPC-UA e o lado direito representa um ambiente de IoT onde um celular acessa remotamente os dados enviados pelos sensores localizados no ambiente industrial. Os dois ambientes interagem entre si através de um *gateway* que faz a ponte de transição de dados. O *gateway* é formado por um aplicativo que extrai os dados do servidor, converte e os transporta, através do protocolo MQTT, para um *broker* que está na nuvem; estes dados podem ser acessados pelos assinantes dos tópicos no *broker*.

Figura 4 – Visão geral de funcionamento do projeto



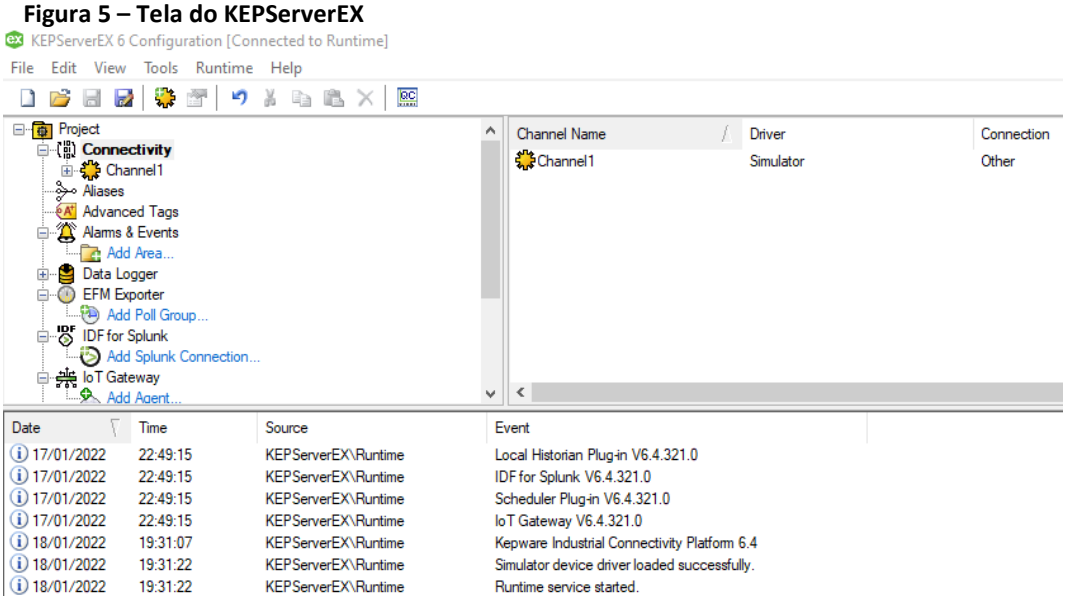
Fonte: Elaborado pelo autor

Os próprios servidores de conectividade industrial possuem simuladores de sinais analógicos para serem usados na realização de testes de uma aplicação. Nesse caso está sendo utilizado um servidor OPC-UA da empresa PTC, chamado KEPServerEX. Este software é privado e a aquisição da licença envolve custo financeiro, porém, o fabricante possui uma versão de demonstração que pode ser adquirida através de um cadastro no site da empresa.

Neste projeto, foi usada uma versão de demonstração deste software adquirida mediante cadastro. O KEPServerEX é uma plataforma de conectividade industrial que permite ao usuário o gerenciamento, monitoramento e controle de diversos dispositivos de automação e aplicativos de software por meio de uma interface intuitiva (Kepware, 2021).

Ao executar o aplicativo, é necessário criar as *tags* que serão usadas para a simulação e configurar as propriedades do servidor OPC-UA. A tela inicial do KEPServerEX possui três janelas de interação, sendo que a do lado esquerdo mostra a estrutura do projeto como alarmes, *tags*, dados, conectividade e outros. Clicando em um desses tópicos, abre-se, na janela do lado direito, uma espécie de tabela formada por linhas e colunas que expõe com mais detalhes as informações do item acessado. Na parte inferior, a terceira janela, mostra os eventos do software, conforme mostra a figura 5.

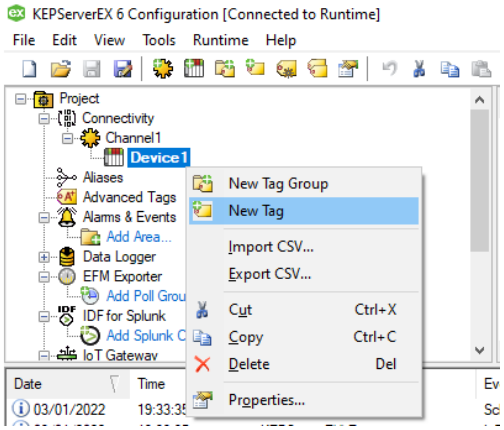
Figura 5 – Tela do KEPServerEX



Fonte: Elaborado pelo autor

Para criação das duas *tags* que serão usadas na simulação de sinais no *gateway*, foi necessário clicar com o botão direito no item *Connectivity* → *Device* e selecionar a opção *New Tag,* conforme mostra a figura 6.

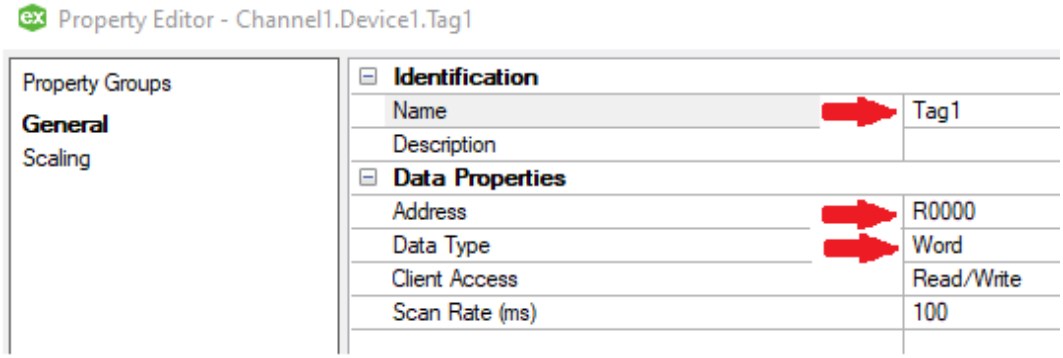
Figura 6 – Acessando *New Tag* do KEPServerEX



Fonte: Elaborado pelo autor

A janela *Property Editor* é, então, aberta, para criação da *tag*. O item *General* precisa estar selecionado para que as informações da *tag* sejam apresentadas na janela da direita. As informações *Name*, *Address* e *Data type* devem ser ajustadas, respectivamente, com os valores *Tag1*, *R0000* e *Word*, enquanto que *Client Access* (valor *Read/Write*) e *Scan Rate* (valor 100ms) podem manter seus valores padrão. A figura 7 mostra os dados ajustados.

Figura 7 – Tela *Property* Editor KEPServerEX

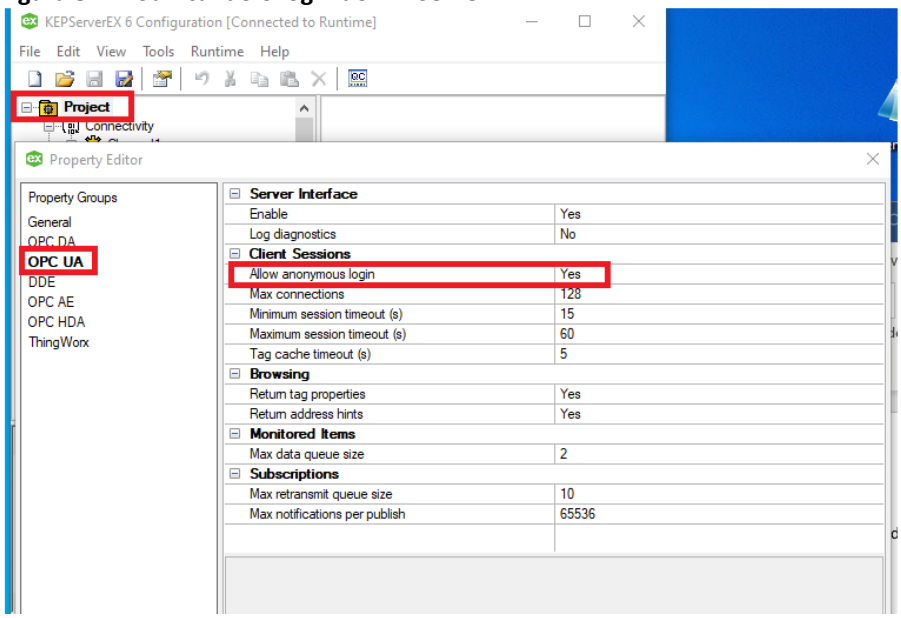


Fonte: Elaborado pelo autor.

Para nome (*Name*) poderia ser utilizado outro texto e outros tipos de dados (*Data Type*) poderiam ser selecionados (dentre as opções disponibilizadas). O item *Address* foi preenchido com R0000, onde a letra representa o tipo da variável que pode ser registro (R), constante (K) e *Strings* (S). O valor numérico define um endereçamento da variável entre 0000 e 9999 para registros e constantes ou entre 000 e 999 para *strings*.

Após a criação das duas variáveis é necessário configurar as propriedades do OPC-UA. Para isso é necessário acessar a janela *Property Editor*, conforme mostra a figura 8, clicando com o botão direito no item *Project,* selecionando, em seguida, a opção *Properties*. Nesta janela, deve ser selecionado o item OPC-UA e, no tópico *Client Sessions*, o parâmetro *Allow anonymous login* deve ser alterado para *Yes* de modo que o acesso às variáveis não necessite da identificação do usuário.

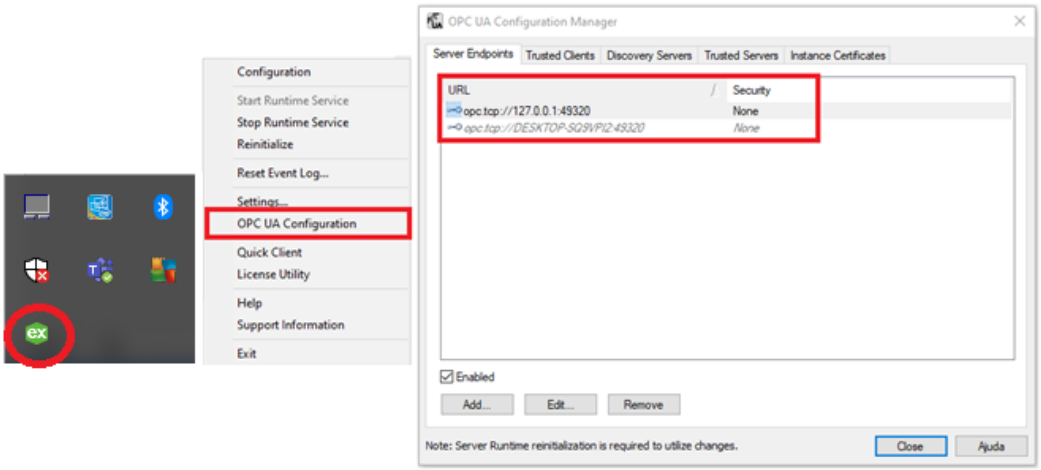
Figura 8 – Modificando o login do KEPServerEX



Fonte: Elaborado pelo autor

A figura 9 mostra como acessar os detalhes de conexão. Clicando com o botão direito no ícone do KEPServerEX, que está localizado na barra de tarefas, e selecionando a opção *OPC UA Configuration* é apresentada uma janela de configurações. Na aba *Server Endpoints* podem ser visualizados/editados os endereços de rede e portas de acesso das variáveis. O número da porta e o tipo de segurança podem ser alterados através de um duplo clique no item.

Figura 9 – Acessando a segurança do KEPServerEX



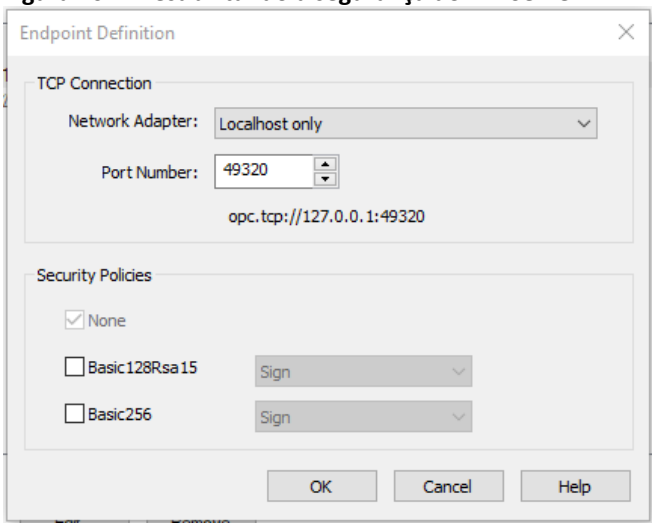
Fonte: Elaborado pelo autor

Os endereços de rede vêm configurados com duas camadas de seguranças ativas que são: Basic128Rsa15 e Basic256. Esse tipo de proteção permite uma comunicação segura, sendo responsável por assinar, codificar e decodificar mensagens trocadas entre clientes e servidores OPC-UA. Para realização de um teste de comunicação envolvendo o *gateway* com o KEPServerEX não há necessidade de que esse tipo de segurança esteja ativado, sendo assim,

o nível de segurança pode ser modificado para *None* o que significa um sistema sem criptografia.

A modificação é feita através de um duplo clique no endereço de rede para acessar a janela *Endpoint Definition* e fazer a desativação, conforme apontado figura 10. No tópico de *Security Policies* é preciso desmarcar os itens Basic128Rsa15 e Basic256. Feitas essas modificações, o sistema de segurança ficará com a mensagem exposta de *None*.

Figura 10 – Desabilitando a segurança do KEPServerEX



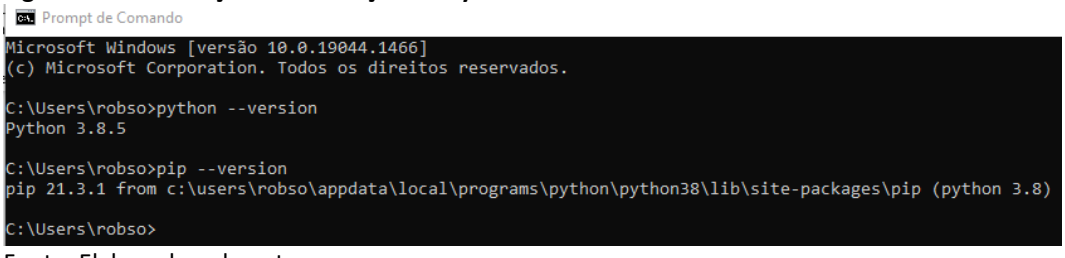
Fonte: Elaborado pelo autor

Feitas as modificações no KEPServerEX, é necessária a construção do algoritmo que irá atuar como *gateway*. Para isso, é preciso fazer a instalação da linguagem de programação Python no notebook que irá operar como *gateway*.

A instalação é simples: basta acessar o site <https://www.python.org/downloads/> e baixar o aplicativo conforme o sistema operacional da máquina. No caso em questão, foi feito o *download* para o Windows 10. Durante o processo de instalação é essencial que a opção *Add Python 3.8 to PATH* esteja selecionado para que o *Prompt Command* o reconheça e funcione normalmente para criação da aplicação. Após a instalação do Python, pode-se acessar o *Prompt Command* para verificar se a instalação foi feita corretamente e do sistema de gerenciamento de pacotes *pip*. A vantagem do *pip* é que possui uma interface de linha de comando simples, o que facilita a adição de bibliotecas no Python.

A verificação das instalações é feita através dos comandos *python –version* e *pip –version,* como mostra a figura 11.

Figura 11 – Verificação da instalação do Python



Fonte: Elaborado pelo autor

A linguagem Python possui diversas bibliotecas prontas que ajudam a reduzir o tempo no desenvolvimento de uma aplicação. Biblioteca é uma coleção de subprogramas ou

módulos desenvolvidos por programadores que os disponibilizam no próprio repositório da linguagem e podem ser importadas sem ônus financeiros. Para o desenvolvimento do *gateway* foram usadas duas bibliotecas especificas: uma para atender o protocolo MQTT e a outra para OPC UA.

A biblioteca usada na comunicação com protocolo MQTT foi a *paho-mqtt* (https://projects.eclipse.org/projects/iot.paho) desenvolvida pela Fundação Eclipse. Essa biblioteca permite a conexão com um broker MQTT para publicar mensagens, assinar tópicos e receber mensagens publicadas. O protocolo OPC UA necessitou do uso da biblioteca *asyncua* (https://github.com/FreeOpcUa). No algoritmo esse código fonte está ajudando o *gateway* na comunicação com o servidor KEPservEX para fazer a leitura dos dados.

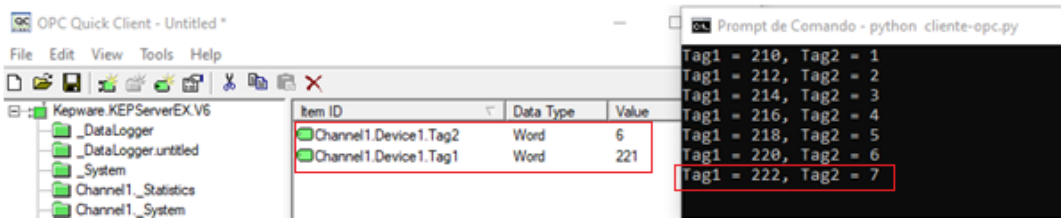
A instalação de bibliotecas é feita através do comando <*pip install* (nome da biblioteca)> no *prompt de comandos*. Para construir um algoritmo que faz a leitura dos dados do KEPservEX foram necessárias algumas bibliotecas adicionais: *sys, asyncio, logging e time*.

A biblioteca *sys* é responsável pela interação com o interpretador Python e fornece uma série de funções e variáveis para controlar o ambiente de execução do aplicativo. A biblioteca *asyncio* é usada como uma base para várias estruturas assíncronas do Python e para escrever os códigos da sintaxe *async/await*. A biblioteca *logging* é usada para rastrear os eventos que acontecem durante a execução do algoritmo e, por fim, a biblioteca *time* é usada para fazer a medição do tempo de execução dos programas ou trechos de código algoritmo, pois o módulo ajuda a informar a hora em segundos.

Na sintaxe, *async def main()* é o módulo principal do programa, onde fica localizado o endereço do URL e o corpo do algoritmo descrito por linhas de comandos. O endereço do URL deve ser o mesmo que está no KEPServerEX como mostrado na figura 10.

A figura 12 mostra, no lado esquerdo, os valores das *tags* do servidor KEPServerEX e, no lado direito, os valores lidos pelo algoritmo em testes realizados. A execução do algoritmo foi feita no *Prompt Comando*s.

Figura 12 – Leitura das variáveis feita pelo algoritmo



Fonte: Elaborado pelo autor

O próximo passo é fazer com que os valores das *tags* possam ser vistos remotamente. Dessa forma, é preciso utilizar o protocolo MQTT para que os dados sejam enviados para um *broker* via nuvem de modo que possam ser visualizados por assinantes dos tópicos.

A interação irá contar com uso do *broker* gratuito da Eclipse Foundation para realização dos testes; seu endereço é mqtt.eclipseprojects.io. O uso desse *broker* não necessita de cadastro. Basta o usuário configurar os tópicos e publicar as informações no endereço citado anteriormente para que o assinante do tópico possa visualizar. Para o setor industrial é recomendado o uso de *broker* com restrições de segurança devido a privacidade dos dados e alta disponibilidade.

Antes da função *main*, foram definidas as três principais configurações do *broker* que são:

- Endereço do *broker* (mqtt.eclipseprojects.io);
- Nome do projeto (SENAI-MQTT), utilizado como nome do cliente na conexão;
- Nome do tópico (foram criados dois tópicos: *Tag*1 e *Tag*2).

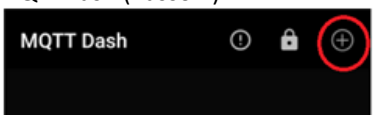
Apesar de terem sido utilizados um tópico para cada *tag* poderia ter sido definida uma estrutura json ou xml para se utilizar um único tópico para compartilhamento dos valores de múltiplas *tags*.

No algoritmo, o comando *publish.single* realiza a publicação de cada tópico (*tag*) recebendo, entre parênteses, as quatro informações necessárias: nome do tópico, valor do dado, endereço do *broker* e nome do projeto. No apêndice está contido o algoritmo atualizado que será usado para fazer a publicação dos tópicos no *broker* em nuvem.

As informações que foram enviadas ao *broker* podem ser vistas remotamente, basta baixar no *smartphone* o aplicativo MQTTDash, ou outro cliente MQTT, e assinar os tópicos para poder visualizar os dados.

Considerando o uso do MQTTDash, deve-se, após a instalação do aplicativo, realizar a sua configuração. Primeiramente, clicando no sinal de + que está no canto superior direito da tela, como mostra a figura 13, para adicionar um *broker*.

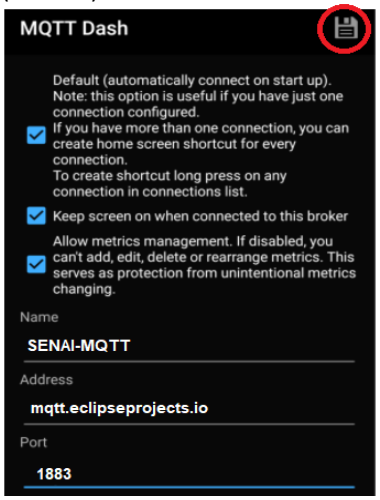
Figura 13 – Configuração
MQTTDash (Passo 1)



Fonte: Elaborado pelo autor.

O próximo passo é fazer a pré-configuração do sistema *mobile*, sendo necessário preencher os campos *Name*, *Adress* e *Port*. Os campos *Name* e *Adress* devem ser o mesmo que foi descrito na linha de comando do algoritmo, no caso, *Name* foi SENAI-MQTT e *Adress* preenchido com ***mqtt.eclipseprojects.io***. A porta de comunicação do MQTT é sempre 1883. Feito o preenchimento das linhas, é necessário salvar as informações através do clique no disquete que está no canto superior direito da tela, como mostra a figura 14.

Figura 14 – Configuração MQTTDash
(Passo 2)

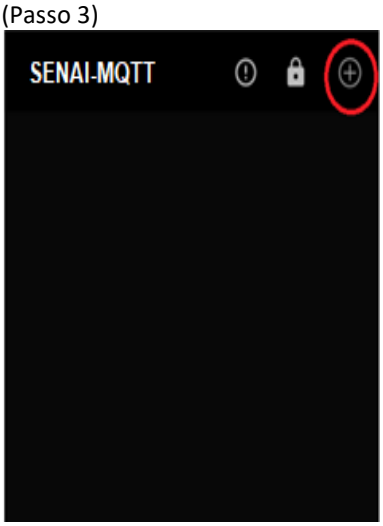


Fonte: Elaborado pelo autor

.

A tela ficará igual a figura 15 e, para adicionar os tópicos no *dashboard,* foi necessário clicar no símbolo “+” no canto superior direito da tela.

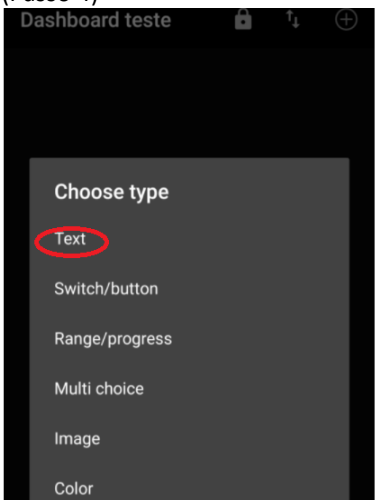
Figura 15 – Configuração MQTTDash
(Passo 3)



Fonte: Elaborado pelo autor.

Na próxima tela aparecerão algumas opções de itens a serem adicionados para criação da caixa de texto, botão e outros. Para visualização do valor da *tag* foi adotada a opção *text,* como mostra a figura 16.

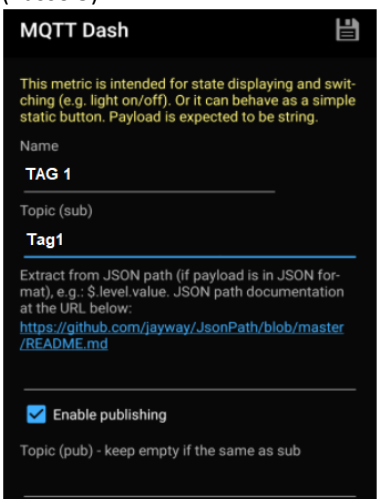
Figura 16 – Configuração MQTTDash
(Passo 4)



Fonte: Elaborado pelo autor

Após clicar em *Text*, aparecerá uma tela na qual é necessário configurar a caixa de texto que irá aparecer na tela do *smartphone*. O item *Name* receberá o nome da variável de TAG 1, no item *Topic* tem que ter o mesmo nome a que foi dado no algoritmo para variável, no caso, tag1. A figura 17 mostra os campos que foram preenchidos.

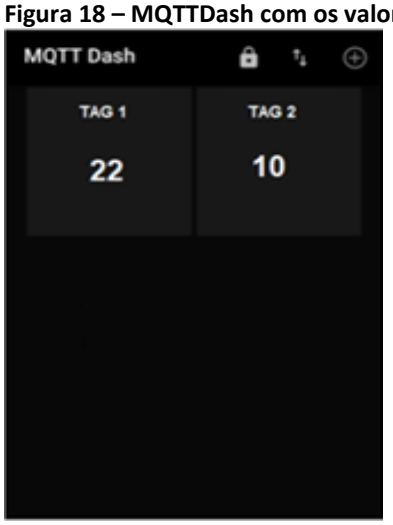
Figura 17 – Configuração MQTTDash
(Passo 5)



Fonte: Elaborado pelo autor

Criadas as duas *tags*, a tela do dashboard terá duas caixas de texto com os nomes de TAG1 e TAG2. Voltando para tela inicial e clicando no tópico SENAI-MQTT, o sistema irá acessar os dados que estão na nuvem e serão transmitidos para as caixas de textos que são assinantes dos tópicos, como mostra a figura 18. Os valores das *tags* foram obtidos através da assinatura dos tópicos que estão em nuvem.

Figura 18 – MQTTDash com os valores do *Broker* em nuvem



Fonte: Elaborado pelo autor

Assim, se confirma o sucesso no estabelecimento da conexão entre os ambientes de automação industrial e IoT através do *gateway* que fez a ponte de transição de dados para a leitura no *smartphone*.

4. RESULTADOS E DISCUSSÕES

O *gateway* foi desenvolvido com uso da linguagem de programação Python que é uma tecnologia de código aberto e possui diversas bibliotecas livres para utilização. Para comunicar o ambiente industrial foi necessário o uso da biblioteca *asyncua* que interage com o protocolo OPC-UA. Para a IoT foi usada o código fonte *paho-mqtt* que se comunica com o protocolo MQTT e possibilita a publicação de dados via nuvem para o *broker*.

O algoritmo foi desenvolvido obteve resultado satisfatório, uma vez que o objetivo era a leitura de dados via OPC-UA, o envio desses dados pelo protocolo MQTT e informações podendo ser vistas remotamente pelo assinante do tópico no *broker* em nuvem. Alterações no código podem implementar validação de certificados e autenticação para implementar a segurança necessária a uma aplicação em nuvem.

5. CONCLUSÃO

Este trabalho teve como foco um dos pilares da indústria 4.0 que é a computação em nuvem, onde é possível visualizar e monitorar dados de um ambiente industrial remotamente. Isso é possível devido a construção do *gateway* que faz a ponte de transição de dados entre os ambientes industrial e IoT, integrando os protocolos OPC-UA e MQTT.

Espera-se que esse trabalho possa auxiliar em futuros projetos da indústria 4.0 e IoT que necessitam que os dados do ambiente industrial possam ser visualizados remotamente, como também, a comunicação entre máquinas que precisam de informações para ocorrer uma tomada de decisão programada. O uso de tecnologias de apresentação e servidores de nuvem podem ser adicionados visando monitoramento de equipamento industriais (Botelho *et al*., 2023) e/ou o armazenamento em banco de dados em nuvem permite análise posteriores como, por exemplo, decisões relacionadas a manutenção preventiva (Araujo; Rossato, 2022; Pestana *et al*., 2022).

REFERÊNCIAS

ADOLPHS, P. et al. **GMA status report:** reference architecture model industrie 4.0 (RAMI4.0). ZVEI, 2015. Disponível em: https://www.zvei.org/en/press-media/publications/gma-status-report-reference-archtitecture-model-industrie-40-rami-40/ . Acesso em: 12 dez. 2024.

ARAUJO, Thiago Sales; ROSSATO, Daniel Barbuto. Utilizando rapidminer para manutenção preditiva. **Revista Científica SENAI-SP-Educação, Tecnologia e Inovação**, v. 1, n. 1, p. 37-53, 2022. Disponível em: https://periodicos.sp.senai.br/index.php/rcsenaisp/article/view/4 . Acesso em: 12 dez. 2024.

BALANI, Naveen; HATHI, Rajeev. **Enterprise IoT:** a definitive handbook. [s. l.]: CreateSpace Independent Publishing Platform, 2016.

BIONDANI, Francesco; CHENG, Dong Seon; FUMMI, Franco. Adopting OPC UA for efficient and secure firmware transmission in Industry 4.0 scenarios. *In*: INTERNATIONAL SYMPOSIUM ON INDUSTRIAL ELECTRONICS (ISIE), 33., University of Verona, Verona: IEEE, 2024. DOI:10.1109/ISIE54533.2024.10595820

BOTELHO, Rodrigo Sousa et al. Sistema IoT para monitoramento de bombeamento de efluentes. **Revista Brasileira de Mecatrônica**, v. 5, n. 3, p. 62-79, 2023. Disponível em: https://revistabrmecatronica.sp.senai.br/ojs/index.php/revistabrmecatronica/article/view/149 . Acesso em: 10 dez. 2024.

COGNEX. **O que é o OPC UA e porque é essencial para a automação industrial.** 2019. Disponível em: https://www.cognex.com/pt-br/blogs/machine-vision/why-opc-ua-is-essential-for-factory-automation. Acesso em: 6 fev. 2022.

FERNANDES JÚNIOR, R. F. **Identificação remota de plantas industriais utilizando tecnologias OPC e Cyberopc.** 2009. Dissertação (Mestrado em Engenharia Elétrica) - Faculdade de Engenharia Elétrica e de Computação, Universidade de São Paulo, São Carlos, 2009. Disponível em: https://repositorio.usp.br/item/001741666. Acesso em: 6 fev. 2022.

FRENET. **Gateway:** o que é e como funciona? 2022. Disponível em: https://www.frenet.com.br/blog/gateway-o-que-e-como-funciona/. Acesso em: 9 fev. 2022.

GILCHRIST, A. **Industry 4.0:** the industrial internet of things. California: Apress Media, 2016.

KEPWARE. **About kepware.** 2021. Disponível em: https://www.kepware.com/en-us/about/overview/. Acesso em: 22 mar. 2021.

KUMAR, K.; ZINDANI, D.; DAVIM, J. P. **Industry 4.0:** developments towards the fourth industrial revolution. Singapore: Springer, 2019.

MELO, Pablo Felipe Soares de. Dispositivo de controle para a Indústria 4.0 baseado no RAMI 4.0 e OPC UA. 2020.

PESTANA, Allan dos Anjos *et al*. Automação em estação de tratamento de esgoto: big data e machine learning no saneamento. **Revista Brasileira de Mecatrônica**, v. 4, n. 4, p. 82-111, 2022. Disponível em: https://revistabrmecatronica.sp.senai.br/ojs/index.php/revistabrmecatronica/article/view/170 . Acesso em: 10 nov. 2022.

POPKOVA, E. G.; RAGULINA, J. V.; BOGOVIZ, A. V. **Industry 4.0:** industrial revolution of the 21st century. Cham: Springer, 2019.

SETHI, P.; SARANGI, S. R. Internet of things: architectures, protocols, and applications. *In:* **Journal of Electrical and Computer Engineering.** 2017. Disponível em: https://www.hindawi.com/journals/jece/2017/9324035/. Acesso em: 9 fev. 2022.

SCHWAB, K. **The fourth industrial revolution.** Nova York: Crown Business, 2017.

SILVEIRA, Rodrigo Silvério da *et al*. Aplicação do protocolo MQTT para integração de dispositivos IIOT a sistemas de automação industrial em um ambiente de testes. **Revista Brasileira de Mecatrônica**, v. 4, n. 2, p. 01-15, 2021. Disponível em: https://revistabrmecatronica.sp.senai.br/ojs/index.php/revistabrmecatronica/article/view/141 . Acesso em: 10 dez. 2024.

OASIS STANDARD. **MQTT Version 5.0**. 2019. Disponível em: https://docs.oasis-open.org/mqtt/mqtt/v5.0/mqtt-v5.0.pdf. Acesso em: 12 dez. 2024.

TANENBAUM, A. S.; WETHERALL, D. J. **Redes de computadores**. 5. ed. São Paulo, SP: Pearson, 2011. *E-book*.

TIETZ, Fabricio. **Desenvolvimento de um gateway de internet das coisas para automação industrial**. 2017. Trabalho de Conclusão de Curso (Graduação) – Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos, 2017. Disponível em: https://bdta.abcd.usp.br/directbitstream/ff31faba-15c5-475d-9142-955edf890162/Tietz_Fabricio_tcc.pdf . Acesso em: 12 dez. 2024.

VENTURELLI, Márcio. OPC UA na automação industrial. **Automação Industrial Info**. 12 novembro 2018. Disponível em: https://www.automacaoindustrial.info/opc-ua-na-automacao-industrial/ . Acesso em: 20 mar. 2021.

YANG, C.; SHEN, W.; WANG, X. The internet of things in manufacturing: Key issues and potential applications. **IEEE Systems, Man, & Cybernetics Magazine**, v. 4, n. 1, p. 6-15 jan. 2018.

APÊNDICE A - ALGORITMO DO GATEWAY

```
import paho.mqtt.client as mqtt
import asyncio
import time
import sys
sys.path.insert(0, "..")
import logging

from asyncua import Client, ua

# URL do broker MQTT
broker_url = "test.mosquitto.org"
# Nome da conta usada no servidor MQTT
client_name = "SENAI-MQTT-2"
# Tópico referente ao valor enviado ao broker MQTT
topico1 = "Tag1"
# Tópico referente ao valor lido do broker MQTT
topico2 = "Tag2"

tag1value = 0
tag2value = 0

# Callback responável por receber uma mensagem publicada no tópico
def on_message(client_mqtt, userdata, msg):
    global tag2value
    try:
        # O valor publicado no servidor MQTT é convertido int
        # e salvo na variável tag2value
        s = str(msg.payload.decode("utf-8"))
        tag2value = int(s)
    except:
        return

client_mqtt = mqtt.Client(client_name)
client_mqtt.on_message = on_message #Especifica função de callback
print("Estabelecendo conexao com o broker: ", broker_url)
client_mqtt.connect(broker_url)
client_mqtt.loop_start() #Inicia o loop do cliente MQTT
# Tópico referente ao valor que vai ser lido do broker MQTT
print("Assinando o tópico: ", topico2)
client_mqtt.subscribe(topico2)

async def main():
    # URL para conexão com o servidor OPC-UA
    url = "opc.tcp://127.0.0.1:49320"
    async with Client(url=url) as client_opcua:

        # Especificação das tags que manipuladas no servidor OPC-UA
        tag1 = client_opcua.get_node("ns=2;s=Channel1.Device1.Tag1")
        tag2 = client_opcua.get_node("ns=2;s=Channel1.Device1.Tag2")

        while 1:
            global tag1value, tag2value

            # Leitura do valor da Tag1 a partir do servidor OPC-UA
            tag1value = await tag1.read_value()

            # Preparação do valor a ser gravado na Tag2
            dv = ua.DataValue(ua.Variant(tag2value, ua.VariantType.UInt16))
```

```
            # Gravação do valor na Tag2 no servidor OPC-UA
            await tag2.set_value(dv)

            # Publicação do valor da Tag1 no broker MQTT
            client_mqtt.publish(topico1, str(tag1value))

            # Impressão dos valores da Tag1 e da Tag2 na tela
            print(f"Tag1 = {tag1value}, Tag2 = {tag2value}")

            time.sleep(1)

if __name__ == "__main__":
    logging.basicConfig(level=logging.WARN)
    asyncio.run(main())
```

Sobre os Autores

---

[i]Robson Vieira Soares

Formado em Engenharia de Automação e Controle pela Universidade Paulista - UNIP (2016), possui 3 formações técnicas pelo SENAI-SP, sendo duas concluídas no SENAI “Suíço-Brasileira Paulo Ernesto Tolle” em mecânica de precisão (2007) e Desenhos de projetos (2011). Uma no SENAI “Ary Torres” de Eletroeletrônica (2018). Aluno do curso de pós-graduação em automação e controle da Faculdade de Tecnologia SENAI “Mariano Ferraz”. Tem experiência na área de Automação Industrial e Mecânica.

[ii]Douglas da Serra Ogata (*In memoriam*)

Possui graduação em Engenharia de Automação e Controle pela Universidade Paulista - UNIP (2002), e Especialização em Automação e Controle pela Faculdade de Tecnologia SENAI “Mariano Ferraz” (2015). Aluno de mestrado do Programa de Pós-graduação em Engenharia Mecânica da Escola Politécnica da USP com tema voltado à Indústria 4.0. Tem experiência na área de Automação Industrial e Mecatrônica.

[iii]André Luis dos Santos

Possui graduação em Engenharia Mecatrônica pela Universidade Paulista, mestrado em Engenharia pela Universidade de São Paulo e é docente na Faculdade de Tecnologia SENAI Mariano Ferraz. Tem experiência na área de automação industrial, desenvolvimento de software, processamento de sinais e redes de comunicação. https://orcid.org/0000-0001-6627-3886

[iv]Paulo André dos Santos

Possui graduação em Mecatrônica Industrial - SENAI - Departamento Regional de São Paulo (2005), mestrado em Engenharia Elétrica pela Universidade Federal do ABC (2014) e doutorado pela USP (2024). Atualmente é professor no Centro Universitário SENAI-SP. Tem experiência na área de Engenharia Elétrica, com ênfase em Energia, Automação e Eletrônica embarcada, atuando principalmente nos seguintes temas: eficiência energética, sistemas de visão embarcadas, instrumentação industrial e FPGA . https://orcid.org/0000-0002-1177-1949

[v]Pedro André Braga de Oliveira

Possui graduação em Matemática Aplicada e Computacional com habilitação em sistemas e controle pela Universidade de São Paulo - USP (2009), especialização em Ciência de Dados (Big Data Analytics) pela Universidade Presbiteriana Mackenzie (2019). Atualmente é professor da Faculdade SENAI São Paulo “Mariano Ferraz”. https://orcid.org/0009-0006-3195-1759