Development of a Wireless Gateway for Industrial Internet of Things Applications

I. Ferreira, J. Bigheti, and E. Godoy

***Abstract*—Industry 4.0 is a new concept representing the evolution of the current production systems through the convergence of new technologies of industrial automation and information systems. The Industrial Internet of Things (IIoT) stands out for the communication among equipment and systems, offering to users a variety of useful information for the management of improvement of production systems. This paper presents the development of a wireless gateway for IIoT applications. This IIoT gateway is based on open source solutions and enables the programming, network communication and supervision of the industrial equipment. The communication protocols supported by the gateway are: Modbus TCP/IP, MQTT and CoAP for connection between the remote modules and the gateway and HTTP using the REST standard with JSON format for external communication with the gateway. An innovation and differential of the gateway was adding the OpenPLC project in order to enable the programming and automation of industrial processes according to the IEC 61131-3. In addition, the gateway stores process information and makes it available online and accessible in real-time to any point connected to the network and/or the Internet. Several operation tests performed validated the development and demonstrated the efficacy of the gateway.**

***Index Terms*—Industry 4.0, Internet of Things, Embedded Devices, OpenPLC.**

I. Introdução

A necessidade de se aumentar a produtividade tem levado o mundo industrial a passar por várias revoluções. A primeira revolução aconteceu no século 18 com o advento da máquina a vapor, revolucionando a produção têxtil. O desenvolvimento da energia elétrica e da produção em massa levaram à segunda revolução industrial, no final do século 19. Em 1969 a crescente utilização da tecnologia da informação e eletrônicos na indústria gerou a terceira revolução industrial. Dessa forma, a indústria tem evoluído de forma a incorporar novas tecnologias e obter maior produtividade [1].

A Indústria 4.0 (I4.0) é um novo conceito que representa uma evolução dos sistemas produtivos a partir da convergência entre novas tecnologias de automação industrial e tecnologia da informação (TI) [2]. A I4.0 se destaca pela evolução na comunicação entre sistemas e equipamentos, disponibilizando informações úteis para o gerenciamento e aperfeiçoamento dos sistemas de produção.

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001.

I.V. Ferreira, J.A. Bigheti, E. P. Godoy are with São Paulo State University (Unesp), Sorocaba, SP 18087-180, Brazil (eduardo.godoy@unesp.br).

É possível monitorar e controlar a produção, melhorando a produtividade e qualidade, através de um melhor ajuste das máquinas [3].

A maioria das tecnologias necessárias à implementação da I4.0 já existem. Entre elas podem-se citar, entre outras, o protocolo IPV6 para ampliação dos pontos de conexão IP a todos os equipamentos, a ampliação do uso de redes sem fio para flexibilização da comunicação e aquisição de dados, o uso de virtualização através da disponibilização de sistemas e serviços a partir de softwares; a Internet das Coisas (IoT – *Internet of Things*), Big Data e Computação em Nuvem visando a conexão de todos os equipamentos em redes e o processamento e compartilhamento das informações na nuvem [4-8]. O grande desafio, portanto, é promover a integração entre essas tecnologias, visando a obtenção de uma nova realidade produtiva, onde tudo estará conectado para que as melhores decisões de produção, custo e segurança sejam tomadas, tudo sob demanda e em tempo real.

Uma das tecnologias base para este conceito é a IoT e o Máquina para Máquina (M2M – *Machine to Machine*) [9]. A IoT contempla a conexão lógica dos dispositivos e meios relacionados ao ambiente produtivo como os sensores, controladores, computadores, células de produção, sistema de planejamento produtivo, sendo as informações compartilhadas através de bancos de dados. A M2M representa a interconexão entre células de produção, onde tais sistemas podem trocar informações entre si, de forma autônoma, e tomar decisões de produção e segurança relacionadas ao processo através de um modelo de inteligência gerenciado pela IoT.

As pesquisas em IoT podem ser encontradas em diferentes áreas como saúde, transporte, comunicação, energia e industrial [9-10]. Cada foco de aplicação possui características e requisitos específicos. Quando aplicada à área industrial, a IoT industrial (IIoT) busca simplificar e criar arquiteturas de sistemas que são mais acessíveis, responsivas e efetivas, com comunicações eficientes e interação entre a produção e sensores, atuadores, entre outros, para aprimorar o desempenho e flexibilidade da indústria [11].

O panorama atual de aplicações de IIoT e I4.0 é bastante heterogêneo, sendo caracterizado por diferentes dispositivos e protocolos de comunicação que necessitam ser integrados aos sistemas corporativos industriais [21]. Neste cenário, protocolos de comunicação tradicionais usados para monitoramento e controle de processos como WirelessHART, ISA100.11a, Profibus e Modbus começam a coexistir com protocolos de IoT como MQTT, CoAP e AMQP, utilizados em aplicações, geralmente hospedadas em nuvem, como

monitoramento do estado e da eficiência (OEE) de máquinas e equipamentos [22]. A integração desses equipamentos e protocolos com sistemas de controle e supervisão industriais (SDCD e SCADA) e sistemas corporativos (ERP) não é transparente. Geralmente, os sistemas industriais ainda não suportam a comunicação com protocolos de IoT e muitas soluções de software de TI não suportam a comunicação com protocolos industriais tradicionais, de forma que muitas vezes as aplicações de automação e IIoT acabam tendo infraestruturas complementares. Uma solução para superar esse desafio de interação é o uso de uma arquitetura com *gateways* [23].

Seja qual for a aplicação de IIoT, temos de um lado o ambiente industrial com as variáveis de interesse disponibilizadas nas máquinas, processos e redes industriais padronizadas. E do outro lado temos o ambiente da Internet e TI com os protocolos da Web (HTTP, nuvem). Portanto, para integrar os dois lados, é necessário um *gateway*, senão a informação de interesse não se torna compatível entre os lados e disponível para uso de forma padronizada e interoperável conforme requerido pelas novas aplicações de IIoT.

O desenvolvimento desses *gateways* não é simples, pois há a necessidade de integração de diferentes soluções de hardware e software, além da necessidade de suportar os diferentes protocolos de comunicação requeridos por cada aplicação. Diversos trabalhos sobre *gateways* podem ser encontrados em diferentes aplicações de IIoT e com diferentes objetivos. Alguns trabalhos focam no desenvolvimento de *gateways* para aplicações definidas, como na integração de remotas de IO e redes industriais tradicionais com sistemas corporativos [24,25] e serviços em nuvem [26] para aplicações de monitoramento de ativos e processos. Outros trabalhos focam na proposição de *gateways* genéricos (sem um caso de aplicação definido), buscando adaptar a comunicação entre protocolos de comunicação diferentes [27-29].

Uma solução para integração de dispositivos de redes sem fio industriais WirelessHART e ISA100.10a com um sistema de supervisão industrial é apresentado em [25]. Como o sistema de supervisão não comunicava diretamente com os protocolos de redes industriais, um *gateway* foi desenvolvido para adaptar a comunicação entre os dispositivos sem fio e um protocolo suportado pelo software de supervisão (Modbus). Nessa mesma linha, referência [26] descreve um *gateway* para integração de equipamentos HART e WirelessHART (via protocolo HART-IP) com serviços em nuvem usando protocolo OPC UA. Uma proposta de *gateway* para IIoT modular e orientada a serviços é apresentada em [27], com suporte a comunicação Modbus e interface via padrão REST. Este trabalho descreve o desafio relacionado à definição e integração de soluções de hardware e software para construção do *gateway,* bem como a validação operacional do mesmo em hardware dedicado. Uma solução genérica de *gateway* para adaptar a comunicação de dispositivos de IoT usando os protocolos MQTT, CoAP, WebSocket e DDS para o padrão HTTP é apresentado em [29].

Diante desse contexto, este trabalho descreve o desenvolvimento de um *gateway* de comunicação sem fio baseado no padrão Wi-Fi, através da integração de *hardware* e *software* de código aberto (*open source*), para aplicações IIoT. Este *gateway* fornece as funcionalidades de comunicação em rede e supervisão dos equipamentos industriais. Uma inovação incorporada ao *gateway* é a possibilidade de programação e automação de processos industriais de acordo com a IEC 61131-3 Esta característica representa um diferencial do *gateway* deste trabalho em relação às soluções encontradas na literatura e discutidas anteriormente. Adicionalmente, este *gateway* é capaz de armazenar e disponibilizar online de forma padronizada e interoperável, as informações do processo ou sistema em questão, para qualquer tipo de plataforma, como dispositivos móveis ou computadores, tornando os dados de monitoramento acessíveis em tempo real, para qualquer ponto conectado à rede e/ou Internet.

Este artigo está apresentado da seguinte forma. A seção II apresenta uma arquitetura para aplicações de IIoT com foco para a proposta do dispositivo *gateway*, suas funcionalidades básicas e tecnologias. A descrição do desenvolvimento do *gateway* é mostrada na seção III, apresentando o sistema embarcado e as APIs de comunicação implementadas. A seção IV discute e compara os resultados de operação do *gateway* em diferentes condições. A seção VII elenca as conclusões e contribuições deste artigo.

II. Arquitetura para IIoT

A área de IoT apresenta uma grande diversidade de abordagens, conceitos e estruturas, bem como diferentes inciativas que propuseram modelos, arquiteturas e ferramentas para aplicações distintas [10]. No entanto, é necessária uma convergência de abordagens e padrões industriais para simplificar a aplicação da IoT na área industrial. Este artigo propõe uma arquitetura de IIoT baseada em um dispositivo funcionando como um portal entre diferentes meios ou protocolos de comunicação (*gateway*) e em uma estrutura de camadas e protocolos apresentada em [12].

A arquitetura para aplicação da IIoT, mostrada na Fig. 1, é composta por um dispositivo *gateway* capaz de gerenciar a comunicação com outros dispositivos responsáveis por coletar dados e executar ações em uma aplicação industrial, possibilitando o monitoramento e controle uma planta industrial. O *gateway* focou em conectividade sem fio no padrão Wi-Fi (pilha TCP/IP), de forma a também permitir sua conexão com dispositivos/equipamentos industriais legados e com plataformas de computação em nuvem.

Os protocolos de comunicação definidos para o *gateway* são o Modbus TCP/IP relacionado às conexões entre equipamentos usando Ethernet Industrial (suporte a outros protocolos como Profinet e EtherNet/IP pode ser desenvolvido), o MQTT e o CoAP provendo as conexões entre equipamentos da IoT, plataformas em nuvem e os protocolos da Internet, como HTTP, para acesso e envio de dados para o *gateway*, permitindo a supervisão e o controle da planta industrial.

O módulo *gateway* é representado por um computador de placa única (Raspberry PI 3) rodando uma distribuição de Linux embarcado. O módulo roda aplicações em ambiente

Node.js, como o *broker* MQTT, clientes CoAP e Modbus TCP/IP, além de um banco de dados MySQL [13] para armazenamento das informações coletadas.

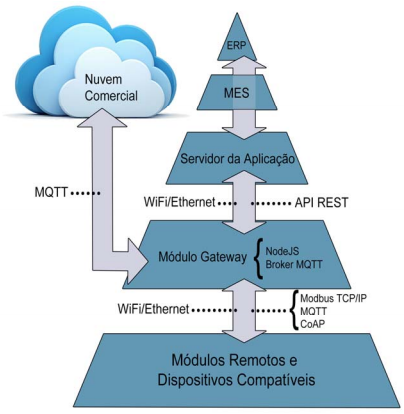

Fig. 1. Arquitetura baseada em gateway para aplicações de IIoT.

O módulo *gateway* é responsável por gerenciar a comunicação com os módulos remotos e interface com o mundo externo. Ele também hospeda o servidor do OpenPLC [14], que é uma aplicação que possibilita a programação dos módulos remotos nas cinco linguagens definidas pela norma IEC-61131-3 (Ladder - LD, Diagrama de Blocos Funcionais - FBD, Sequenciamento Gráfico de Funções – SFC, Lista de Instruções – IL e Texto Estruturado – STL). A programação é realizada através de um aplicativo chamado PLCOpen Editor. A programação é enviada ao sistema através de um *upload* feito em uma página Web, hospedada e provida pelo próprio sistema embarcado do módulo *gateway*.

Os módulos remotos funcionam como uma interface de I/O com o mundo externo, através dos sinais de entrada e saída do sistema. Podem ser adicionados quantos módulos forem necessários e como a comunicação é sem fio, eles podem estar distribuídos pela linha de produção ou malha de processo. Estes módulos são representados por *hardware* livre de baixo custo (ESP8266 ou ESP32) ou equipamentos industriais (CLP, PAC, SCADA, SDCD ou sistemas de supervisão e gerenciamento) que se comunicam com o módulo *gateway* usando um dos protocolos de comunicação suportados (Modbus TCP/IP, MQTT e CoAP).

III. Desenvolvimento do Gateway

As principais funcionalidades do módulo *gateway* são a aquisição de dados de módulos remotos, automação de processos, armazenamento desses dados em um banco de dados e a disponibilização desses dados de uma forma padronizada para serem consumidos por outras aplicações ou dispositivos. Foi utilizada uma Raspberry PI 3 como *hardware do gateway*, por já possuir Wi-Fi de forma nativa e possibilitar a criação de um ponto de acesso para conexão dos módulos remotos através de funcionalidade nativa do Linux, não sendo necessário nenhum *hardware* adicional para prover essa conectividade. Além disso, o uso do *gateway* embarcado e distribuído fisicamente pela área industrial habilita o desenvolvimento de técnicas de computação de borda (*Edge Computing*). Apesar de não ter sido o foco desse trabalho, essa possibilidade está em consonância com trabalhos recentes sobre desenvolvimento de *gateway* para IIoT [30].

O sistema operacional usado foi o *Raspbian Stretch*, uma distribuição Linux baseada no Debian 9. O programa do *gateway* cria uma interface entre módulos remotos (sensores/atuadores) e consumidores de dados, através de uma API REST, criando uma forma padronizada de acesso aos dados e uma abstração dos protocolos utilizados. O módulo Express [15] do Node.js foi usado para criar um servidor Web contendo a API REST. Foram utilizados os módulos jsModbus [16] para comunicação com módulos remotos através do protocolo Modbus TCP/IP, node-coap [17] para comunicação utilizando o protocolo CoAP e o Mosca [18], funcionando como broker MQTT e possibilitando o acesso aos dados. Os dados dos módulos remotos são coletados e armazenados em um banco de dados MySQL.

A. API REST

A API REST foi implementada de modo a distinguir os protocolos utilizados para se comunicar com os módulos remotos diretamente na URL da requisição. Para o Modbus TCP/IP, por exemplo, a URL se inicia com o IP ou endereço do servidor (*gateway*) seguido de "*/api/modbus/*". Para acesso de dados provenientes da comunicação por MQTT, a URL que deve ser acessada é composta pelo endereço do servidor seguido de "*/api/mqtt/*". Essa regra se repete para a comunicação utilizando o protocolo CoAP.

Para protocolos que seguem o modelo requisição-resposta, o dispositivo remoto deve ser cadastrado através de uma requisição com método POST e contendo os dados do dispositivo no formato JSON no corpo da mensagem. Após o cadastro, o gateway se comunica diretamente com o dispositivo remoto de forma cíclica, com período informado no momento do cadastro. Esses dados obtidos são armazenados no banco de dados para futuras consultas, através da API REST. Esse procedimento é mostrado na Figura 2.

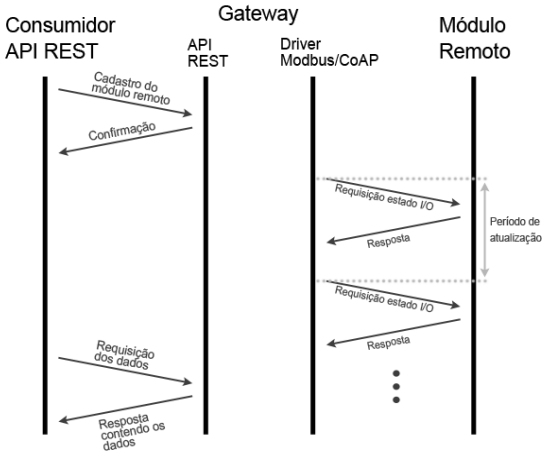

Fig. 2. Comunicação API REST/ gateway e gateway/dispositivo remoto.

O procedimento de comunicação MQTT passa por um intermediário (*broker*), que neste caso é o próprio *gateway*.

Dispositivos remotos podem enviar e receber dados a todo momento para o gateway através do protocolo MQTT, sendo que esses dados só serão armazenados no banco de dados se houver previamente o cadastro do tópico em que eles estão sendo publicados, conforme Figura 3.

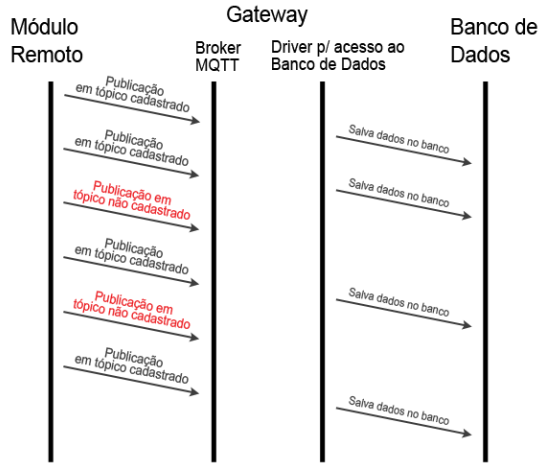

Fig. 3. Operação do gateway para comunicação MQTT.

O cadastro do tópico é feito através de uma requisição com método POST e contendo o tópico que deve ser cadastrado no corpo da mensagem no formato JSON. Os dados armazenados são obtidos através de requisições à API REST (Figura 4).

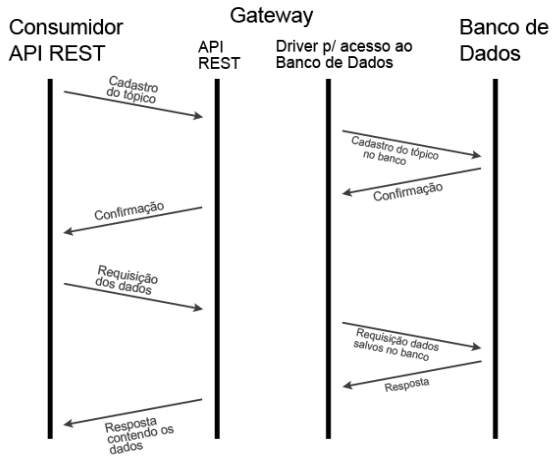

Fig. 4. Obtenção dos dados armazenados no banco de dados.

Uma rota especial foi criada para obtenção de dados sobre a utilização de recursos do *gateway*. Uma requisição HTTP com método GET deve ser enviada para a URL "*http://ip-do-gateway:porta/api/system/*". A resposta do *gateway* contém um JSON com a porcentagem de uso de CPU e memória RAM, além da temperatura do processador.

IV. Resultados e Discussões

Para a validação do desenvolvimento do módulo *gateway*, configurou-se a Raspberry Pi 3 para funcionar como ponto de acesso WiFi. Deste modo, os módulos remotos (ESP8266) podem se conectar a ela diretamente via WiFi (sem a necessidade de um hardware adicional como um roteador sem fio) para troca de informações e conectividade com a rede.

O programa desenvolvido, juntamente com o servidor do OpenPLC e os demais recursos necessários para o funcionamento do gateway foram testados na Raspberry Pi 3. Para a obtenção dos dados sobre o uso de recursos do sistema, como uso de memória, CPU e temperatura do processador, a rota especial da API REST foi utilizada. Uma interface no software LabVIEW envia sucessivas requisições HTTP para essa rota do *gateway* e armazena esses dados, juntamente com o tempo entre cada requisição e sua respectiva resposta, em uma planilha do Microsoft Excel.

A comunicação entre os módulos remotos e o módulo *gateway* foi testada. Os estados das portas de I/O foram requisitados pela aplicação desenvolvida em Node.js e armazenados no banco de dados MySQL. Para a realização de diferentes experimentos e verificação da operação do módulo *gateway*, comparou-se o uso de recursos do sistema em diversas situações, conforme descrito na Tabela I. Buscou-se analisar as capacidades e limitações operacionais do gateway em relação à quantidade de dispositivos e conexões simultâneas, número de I/Os conectados e impacto da comunicação de cada protocolo usado.

O software *Postman* [19] de testes de APIs REST foi utilizado para enviar a requisição responsável por cadastrar um novo dispositivo no *gateway*. Os experimentos realizados estão descritos na Tabela I, enquanto os resultados obtidos para estes testes podem ser vistos na Tabela II.

TABELA I
DESCRIÇÃO DOS TESTES DE OPERAÇÃO COM O GATEWAY

Teste	Descrição	Parâmetros
1	Módulo gateway em espera	Apenas o servidor web em execução
2	OpenPLC em execução	Servidor web e OpenPLC em execução. Nenhum módulo remoto conectado
3	Módulo remoto se comunicando com servidor do OpenPLC	Um módulo remoto conectado com o servidor do OpenPLC, com programa desenvolvido no PLCOpen Editor sendo executado pelo OpenPLC.
4 a)	Diversas conexões com o servidor do OpenPLC adicionadas sucessivamente	Período de atualização de 10s. 8 I/Os digitais requisitados no total.
4 b)	Diversas conexões com o servidor do OpenPLC adicionadas sucessivamente	Período de atualização de 10s. 32 I/Os requisitados de cada tipo.
5	Diversas conexões com o servidor do OpenPLC adicionadas sucessivamente	Período de atualização de 1s
6	Conexões com módulos remotos Modbus TCP/IP adicionadas sucessivamente	Período de atualização de 1s
7	Conexões com módulos remotos CoAP adicionadas sucessivamente	Período de atualização de 1s
8	Diversos módulos remotos MQTT se conectando ao broker sucessivamente	Nenhum tópico sendo monitorado
9	Adição sucessiva de tópicos MQTT monitorados	Adição sucessiva de tópicos a serem monitorados

O teste 1 foi realizado com a Raspberry PI 3 em repouso (o servidor do OpenPLC não havia sido inicializado e não havia nenhum módulo remoto conectado com o *gateway*). A

utilização média de CPU foi de 0,41%, enquanto a de memória RAM foi de 47,55% e a temperatura média do processador foi de 49,38°C, conforme Fig. 5.

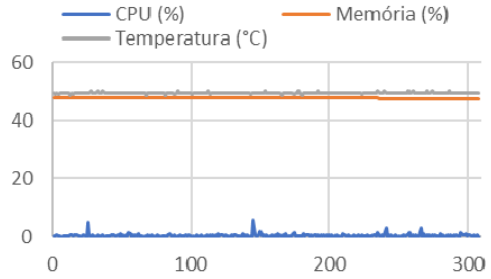

Fig. 5. Utilização de recursos do gateway no teste 1.

Para teste 2 iniciou-se o servidor do OpenPLC, ainda sem a conexão de nenhum módulo remoto com o *gateway*. Não houve grande impacto na utilização da CPU, sendo a utilização média nesse teste igual a 0,41%. A média de uso da memória RAM ficou em 52,03% e a média da temperatura foi de 49,41 °C, conforme Fig. 6.

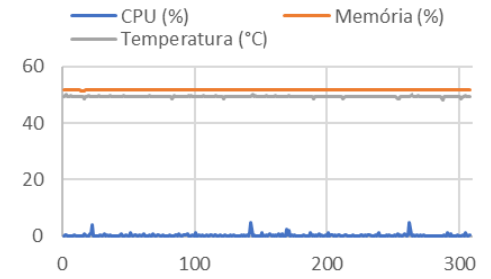

Fig. 6. Utilização de recursos do gateway no teste 2.

Um módulo remoto foi conectado ao servidor do OpenPLC e executou-se um programa em Ladder para a realização do teste 3 (Fig. 7). A utilização média dos recursos do *gateway* foi de 0,55% de CPU e 55,17% de memória RAM, enquanto a temperatura média da CPU foi de 49,49 °C.

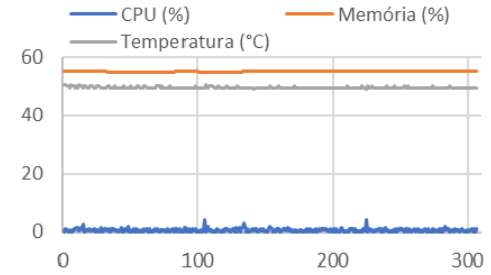

Fig. 7. Utilização de recursos do gateway no teste 3.

Para o teste 4 (Fig. 8), adotou-se a seguinte metodologia: a cada 100 requisições à rota que retorna o status da utilização dos recursos do *gateway*, uma nova conexão com o servidor do OpenPLC foi criada. Isso foi feito cadastrando um novo dispositivo no *gateway* e cada um desses dispositivos requisitando dados de 8 saídas digitais (*coils*) com período de atualização de 10 segundos. Apesar de alguns problemas de comunicação com o *gateway* terem ocorrido durante o experimento (o *gateway* não foi capaz de responder a uma das requisições HTTP, resultando em uma leitura errônea de 0% de utilização dos recursos do sistema, ocasionando uma queda súbita no gráfico da Fig. 8), não houve nenhuma falha severa e foi possível adicionar 50 conexões simultâneas.

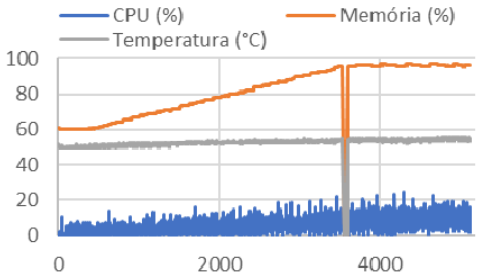

Fig. 8. Utilização de recursos do gateway no teste 4a).

Visando encontrar os limites de operação do módulo *gateway,* o teste 4 foi repetido requisitando 32 endereços de I/O de cada tipo. Nesse cenário (Fig. 9) o número máximo de conexões simultâneas caiu para 14, sendo que a partir da 15ª conexão o módulo *gateway* encontrou problemas para acessar o banco de dados.

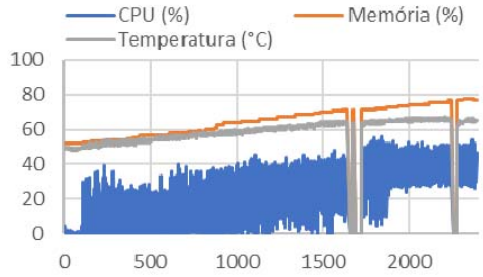

Fig. 9. Utilização de recursos do gateway no experimento 4b).

O teste 5 (Fig.) seguiu a mesma metodologia do teste 4, porém com período de atualização de 1 segundo e requisitando 8 endereços de I/O de cada tipo. O módulo *gateway* apresentou problemas de acesso ao banco de dados a partir da adição do 7º dispositivo.

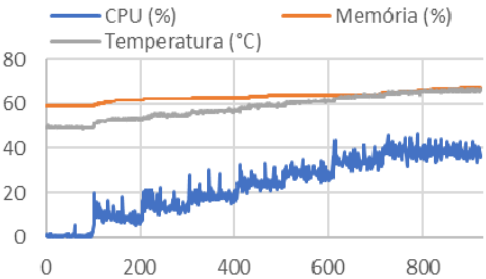

Fig. 10. Utilização de recursos do gateway no teste 5.

O teste 6 (Fig. 10) foi realizado seguindo a mesma metodologia do experimento 5 (requisição 8 endereços de I/O de cada tipo dos dispositivos a cada 1 segundo), com a diferença de cadastro de módulos remotos distintos ao invés de várias conexões com o mesmo módulo. Foram utilizadas ESP8266 como módulos remotos com o *firmware* Modbus.

Novamente os problemas de acesso ao banco surgiram a partir do 7º dispositivo cadastrado.

Seguindo a metodologia do teste 6, para o teste 7 foram adicionadas conexões sucessivas com módulos remotos distintos, comunicando-se com o *gateway* pelo protocolo CoAP com período de atualização de 1 segundo. Diferentemente do teste 6, o ponto de falha no teste 7 foi o *firmware* dos módulos remotos, que não foram capazes de responder a requisições sucessivas e reiniciaram depois de algum tempo. Com esse erro, os módulos remotos não responderam às requisições realizadas pelo *gateway* e consequentemente a utilização de recursos registrada não foi adequada.

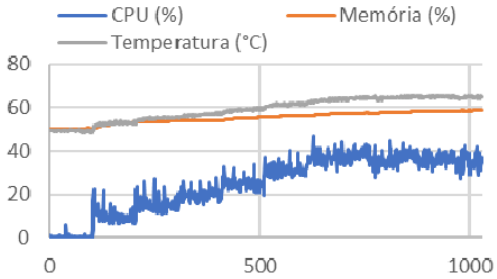

Fig. 10. Utilização de recursos do gateway no teste 6.

No teste 8 módulos remotos com o *firmware* de comunicação MQTT se conectaram ao *gateway* de forma sucessiva. A cada 100 requisições, um novo módulo era conectado. Cada módulo foi configurado com todos os seus 8 pinos de I/O digital como entrada, de modo que esses dados fossem enviados constantemente ao *broker* MQTT. Inicialmente não haviam tópicos cadastrados, ocorrendo somente a comunicação entre os módulos remotos e o *broker* MQTT. Ao final desse teste, 5 módulos remotos estavam conectados ao *gateway*, cada um publicando dados de 8 entradas digitais a cada segundo. Com essa taxa média de 40 publicações por segundo trafegando pelo *broker* MQTT, o uso médio foi de 8,4% de CPU, 60,34% de memória RAM e uma temperatura de 52,82 °C, conforme Fig. 11.

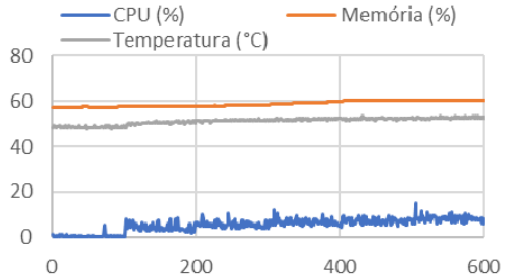

Fig. 11. Utilização de recursos do gateway no teste 8.

O teste 9 é complementar ao teste 8, sendo realizado juntamente com ele. Após os 5 módulos remotos se conectarem ao módulo *gateway* e enviarem os dados das entradas digitais, foram cadastrados os tópicos relativos a cada dispositivo para monitoramento pelo módulo *gateway*. A cada 100 requisições, todos os tópicos relacionados com um dos módulos remotos eram adicionados através da utilização do *wildcard* "#" do MQTT. Isso foi feito através do cadastro no tópico "*endereço-mac-do-dispositivo/#*", possibilitando que todos os dados publicados nos tópicos relativos às entradas digitais do dispositivo identificado pelo endereço MAC cadastrado fossem armazenados no banco de dados. Após todos os tópicos serem cadastrados, foi realizado o cadastro de tópicos individuais de cada entrada digital de um determinado módulo remoto, para verificação do impacto causado na utilização dos recursos do sistema do módulo *gateway*. Os resultados do teste 9 são mostrados na Fig. 12.

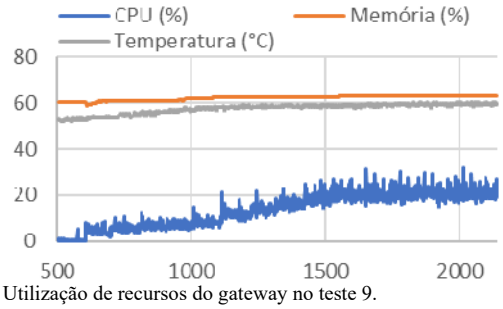

Fig. 12. Utilização de recursos do gateway no teste 9.

Um comparativo dos valores médios observados do uso de CPU (%), memória RAM (%) e temperatura do processador (°C) relativos às últimas 100 requisições executadas nos testes de operação realizados com o gateway são apresentados na Tabela II.

TABELA II
USO MÉDIO DE RECURSO DO GATEWAY PARA CADA TESTE

Teste	Parâmetro analisado			
	Utilização de CPU	Utilização de Memória	Temperatura (°C)	Falha no gateway?
1	0,44%	47,46%	49,42	Não
2	0,39%	52,07%	49,38	Não
3	0,52%	55,21%	49,43	Não
4 a)	9,65%	96,25%	54,62	Não
4 b)	37,11%	76,88%	65,09	Sim
5	38,32%	66,71%	65,84	Sim
6	35,73%	58,44%	65,00	Sim
7	1,98%	54,50%	49,17	Não
8	8,40%	60,34%	52,82	Não
9	22,28%	62,92%	59,43	Não

Analisando os resultados da Tabela II é possível observar um padrão nos casos de falha do módulo *gateway* relacionado à temperatura do processador mais elevada, possivelmente ocasionada por uma quantidade maior de requisições por segundo ao banco de dados nesses experimentos. Mesmo com 50 conexões simultâneas, como visto no experimento 4.a), o módulo *gateway* não apresentou falhas e executou suas funções com normalidade, apesar da maior utilização de memória RAM. Com exceção do teste 7, onde a falha ocorreu no *firmware* dos módulos remotos e não no *gateway*, nos experimentos onde ocorreram falhas estas foram relacionadas ao banco de dados e à taxa de armazenamento de dados.

V. Conclusão

Este trabalho descreveu o desenvolvimento de um *gateway*

sem fio baseado em comunicação Wi-Fi para aplicação da Internet das Coisas Industrial. O diferencial do *gateway* em relação à outras soluções da literatura, é permitir a programação e supervisão dos processos de acordo com padrões industriais (IEC 61131-3), além das tradicionais funcionalidades de aquisição de diferentes fontes de dados e compatibilização da comunicação em aplicações industriais.

Um ponto forte do *gateway* é a flexibilidade e versatilidade de comunicação proporcionada pelos protocolos suportados. O Modbus TCP/IP compatibiliza a comunicação com a Ethernet Industrial em aplicações de automação. Ainda que atualmente não seja suportado, outras soluções de Ethernet Industrial como Profinet e EtherNet/IP poderiam ser futuramente adicionadas ao gateway. O suporte ao MQTT e ao CoAP, que são protocolos otimizados para comunicação em baixa largura de banda, fornecem maior eficiência na distribuição de informações, reduzindo o consumo de banda na rede. Além disso, outro ponto forte é que o modelo é totalmente baseado em soluções de *hardware* e *software* de código aberto, possibilitando a personalização e expansão do sistema para qualquer aplicação.

Os resultados apresentados validaram o desenvolvimento e demonstraram o potencial de uso do *gateway*. A análise experimental avaliou a utilização de recursos (uso de CPU e memória) e limitações operacionais do gateway (quantidade de dispositivos e conexões simultâneas, número de IOs conectados, impacto da comunicação de cada protocolo).

O *hardware* embarcado foi capaz de executar o *software* desenvolvido, juntamente com o OpenPLC e gerenciar a comunicação entre os módulos remotos e o *gateway* de forma eficiente e confiável na maior parte dos cenários analisados. Nos casos em que houve falha, ela estava relacionada com o armazenamento de dados. Trabalhos futuros investigarão essa questão para solução deste problema (uso de outro banco de dados ou de mais de um banco de dados para execução de funções diferentes [20]) e a possibilidade do desenvolvimento de técnicas de computação de borda no *gateway* [30].

REFERÊNCIAS

[1] Sauter, T.; Soucek, S.; Kastner, W.; Dietrich, D. The Evolution of Factory and Building Automation, IEEE Industrial Electronics Magazine, vol.5, no.3, pp.35-48, Sept 2011.

[2] Wollschlaeger, M.; Sauter, T.; Jasperneite, J. The Future of Industrial Communication: Automation Networks in the Era of the Internet of Things and Industry 4.0, IEEE Industrial Electronics Magazine, vol.11, pp.1727, 2017.

[3] Mckinsey Global Institute, The Internet of Things: Mapping the Value Beyond the Hype, 2015.

[4] Zuehlke, D. SmartFactory—Towards a factory-of-things. Annual Reviews in Control, ed. 34, pg 129–138, 2010.

[5] Hegazy, T., Hefeeda, M. Industrial Automation as a Cloud Service, IEEE Transactions on Parallel and Distributed Systems, vol. 26, no.10, pp. 2750-2763, Oct. 1 2015.

[6] Pradilla, J.; Esteve, M.; Palau, C. SOSFul: Sensor Observation Service (SOS) for Internet of Things (IoT), IEEE Latin America Transactions, vol 16, No. 4, pp. 1276-1283, April 2018.

[7] Cravero, A. Big Data Architectures and the Internet of Things: A Systematic Mapping Study, IEEE Latin America Transactions, vol 16, No. 4, pp. 1219-1226, April 2018.

[8] Rocha Filho, G.P.; Mano, L.Y.; Valejo, A.D.B.; Villas, L.A.; Ueyama, J. A Low-Cost Smart Home Automation to Enhance Decision-Making based on Fog Computing and Computational Intelligence), IEEE Latin America Transactions, vol 16, No. 1, pp. 186-191, Jannuary 2018.

[9] Stankovic, J.A., Research Directions for the Internet of Things, IEEE Internet of Things Journal, vol.1, no.1, pp.39, Feb. 2014.

[10] Sundmaeker, H.; Guillemin, P.; Friess, P.; Woelfflé, S. Vision and challenges for realising the Internet of Things, Cluster of European Research Projects on the Internet of Things. 2010.

[11] Lydon, B. Internet of Things Industrial automation industry exploring and implementing IoT, Cover story, ISA. 2014. Disponível em: <https://www.isa.org/standards-and-publications/isa-publications/intech-magazine/2014/mar-apr/cover-story-internet-of-things/>. Acesso em: Agosto, 2016.

[12] Weyrich, M.; Ebert, C. Reference Architectures for the Internet of Things, IEEE Software, vol. 33, no. 1, pp. 112-116, Jan.-Feb. 2016.

[13] MySQL. Sistema de gerenciamento de banco de dados relacional. Disponível em: <http://www.mysql.com/>. Acesso em: Janeiro, 2018.

[14] Alves, T. R., Buratto, M., De Souza, F. M., Rodrigues, T. V. OpenPLC: An Open Source Alternative to Automation, IEEE 2014 Global Humanitarian Technology Conference, 2014.

[15] Express. Framework web rápido, flexível e minimalista para Node.js. Disponível em: <http://expressjs.com/>. Acesso em: Janeiro, 2018.

[16] JSModbus. Simple Modbus TCP Master/Slave implementation for Node.js. Disponível em: <http://github.com/dcvice1967/jsModbus>. Acesso em: Janeiro, 2018.

[17] Node-Coap. A client and server library for CoAP modelled after the http module. Disponível em: <http://github.com/mcollina/node-coap>. Acesso em: Janeiro, 2018.

[18] Mosca. MQTT broker as a module. Disponível em: <https://github.com/mcollina/mosca>. Acesso em: Novembro, 2017.

[19] Postman. Disponível em: <https://www.getpostman.com/>. Acesso em: Novembro, 2017.

[20] Araújo, D.; Costa, J.; Silva, D.R.; Nogueira, M.B.; Rodrigues, M.C. Arquitetura geral de um sistema para automação residencial utilizando plataformas abertas, XIII Simpósio Brasileiro de Automação Inteligente, Porto Alegre, 2017.

[21] Vitturi, S.; Zunino, C.; Sauter, T. Industrial Communication Systems and Their Future Challenges: Next-Generation Ethernet, IIoT, and 5G, Proceedings of the IEEE, vol. 107, no. 6, pp. 944-961, June 2019

[22] Felser, M.; Rentschler, M.; Kleineberg, O. Coexistence Standardization of Operation Technology and Information Technology, Proceedings of the IEEE, vol. 107, no. 6, pp. 962-976, June 2019.

[23] Bloom, G.; Alsulami, b.; Nwafor, E.; Bertolotti, I. C. Design patterns for the industrial Internet of Things, 14th IEEE International Workshop on Factory Communication Systems (WFCS), Imperia, pp. 1-10, 2018.

[24] Nuratch, S. The IIoT devices to cloud gateway design and implementation based on microcontroller for real-time monitoring and control in automation systems, 12th IEEE Conference on Industrial Electronics and Applications (ICIEA), Siem Reap, pp. 919-923, 2017.

[25] Tanyakom, A.; Pongswatd, S.; Julsereewong, A.; Rerkratn, A. Integration of WirelessHART and ISA100.11a field devices into condition monitoring system for starting IIoT implementation, 56th Annual Conference of the Society of Instrument and Control Engineers of Japan (SICE), Kanazawa, pp. 1395-1400, 2017.

[26] Gong, T.; Zheng, S.; Nixon, M.; Rotvold, E.; Han, S. Industrial IoT Field Gateway Design for Heterogeneous Process Monitoring and Control, IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS), Porto, pp. 99-100, 2018.

[27] Resende, C.; Monteiro, M.; Oliveira, J.; Moreira, W.; Cavaleiro, A.; Silva, R.; Carvalho, R.; WGW4IIoT: Wireless Gateway for Industrial IoT, IEEE Symposium on Computers and Communications (ISCC), Natal, pp. 01108-01113, 2018.

[28] De Araújo, P.; Filho, R.; Rodrigues, J.; Oliveira, J.; Braga, S. Infrastructure for Integration of Legacy Electrical Equipment into a Smart-Grid Using Wireless Sensor Networks, Sensors, vol. 18, no. 5, p. 1312, 2018.

[29] Da Cruz, M.A.A.; Rodrigues, J.J.P.C.; Lorenz, P.; Solic, P.; Al-Muhtadi, J.; Albuquerque, V.H.C. A proposal for bridging application layer protocols to HTTP on IoT solutions, Future Generation Computer Systems, vol 97, pp. 145-152, 2019.

[30] Morabito, R.; Petrolo, R.; Loscrì, V.; Mitton, n. LEGIoT: A Lightweight Edge Gateway for the Internet of Things, Future Generation Computer Systems, vol. 81, pp. 1-15, 2018.

Israel Vieira Ferreira received the B.Eng. degree in Control and Automation Engineering and the M.Sc degree in Electrical Engineering at São Paulo State University (Unesp) (SP-Brazil) in 2016 and 2018, respectively. Currently he is a Software Developer of FIT - Instituto de Tecnologia at Sorocaba (SP-Brazil). His research interests include industrial networks, automation and control, networked control systems (NCS), wireless networks, embedded electronics, robotics and expert systems (Fuzzy and Neural Networks). http://lattes.cnpq.br/3958148633705318.

Jeferson André Bigheti received the B.Tech. degree in Electrical Systems in 1990, the Academic specialization in Mechatronics Engineering in 2007 and the M.Sc. in Electrical Engineering in 2011, all from the São Paulo State University (Unesp) at Bauru (SP-Brazil). Currently he is a Ph.D student in Electrical Engineering and a professor of the National Service for Industrial Learning (SENAI) at Lençóis Paulista (SP-Brazil). His research interests include automation, supervision of industrial systems, Internet of Things and Industry 4.0. http://lattes.cnpq.br/6243932250481498.

Eduardo Paciencia Godoy received the B.Eng. degree in Control and Automation Engineering at Itajubá Federal University (MG-Brazil) in 2003 and the M.Sc. and Ph.D. degrees in Mechanical Engineering at University of São Paulo at São Carlos (SP-Brazil) in 2007 and 2011, respectively. Currently he is an Associate Professor of the São Paulo State University (Unesp) at Sorocaba (SP-Brazil). His research interests include industrial networks and automation, networked control systems (NCS), wireless networks and telemetry, embedded electronics, Internet of Things and Industry 4.0. http://lattes.cnpq.br/0072632067545698.