Sistema baseado em nuvem para o monitoramento de uma planta industrial

Matheus Rangel Pimentel * **Celso Jose Munaro** *

* *Departamento de Engenharia Elétrica, Centro Tecnológico, Universidade Federal do Espírito Santo, ES, (e-mails: matheusrpimentel@gmail.com, celso.munaro@ufes.br).*

Abstract: Programmable Logic Controllers (PLCs) are central elements in industrial automation systems, with specialized control functions. Algorithms that involve complex mathematical operations are difficult to implement. However, its communication capabilities make it possible to make data available to cloud systems with a huge variety of tools for the development of monitoring systems based on machine learning. In this work, native PLC features were configured to send data to the cloud and receive the desired diagnostics. The cloud environment was developed for training models and monitoring an industrial plant, taking advantage of existing libraries. The result is a cloud environment with high computational power that provides important diagnostics to a SCADA (Supervisory Control and Data Acquisition) system in an industrial plant. Any browser screen on the local operating station allows parameterization of algorithms programmed in the cloud. In the developed application, algorithms for fault detection in the controlled plant were implemented in the cloud, using operating data for model training and monitoring. The results can be viewed both on the SCADA system graphical interface and on the cloud system dashboard. The proposed environment can be easily extended to industrial processes in general.

Resumo: Os Controladores Lógicos Programáveis (CLPs) são os elementos centrais em sistemas de automação industrial, com funções especializadas de controle. Algoritmos que envolvam operações matemáticas mais complexas são de difícil implementação. Entretanto, seus recursos de comunicação permitem disponibilizar dados a sistemas em nuvem com uma enorme variedade de ferramentas para o desenvolvimento de sistemas de monitoramento baseados em aprendizado de máquina. Neste trabalho, recursos nativos do CLP foram configurados para enviar dados para a nuvem e receber os diagnósticos desejados. O ambiente da nuvem foi desenvolvido para o treinamento de modelos e o monitoramento de uma planta industrial, aproveitando as bibliotecas existentes. Resultou disso um ambiente em nuvem com elevado poder computacional que fornece importantes diagnósticos a um sistema SCADA (Sistemas de Supervisão e Aquisição de Dados, do inglês *Supervisory Control and Data Acquisition*) de uma planta industrial. Uma tela de um navegador qualquer na estação de operação local permite a parametrização dos algoritmos programados na nuvem. Na aplicação desenvolvida, algoritmos para detecção de falhas na planta controlada foram implementados na nuvem, utilizando dados de operação para treinamento de modelos e o monitoramento. Os resultados podem ser visualizados tanto na interface gráfica do sistema SCADA quanto em um navegador acessando o sistema em nuvem. O ambiente proposto pode ser facilmente estendido a processos industriais em geral.

Keywords: Process monitoring, cloud computing, MQTT protocol, programmable logical controller, fault detection, 4.0 industry

Palavras-chaves: Monitoramento de processos, computação em nuvem, protocolo MQTT, controlador lógico programável, detecção de falhas, indústria 4.0

1. INTRODUÇÃO

Aplicações baseadas no paradigma de computação em nuvem têm crescido continuamente devido a seu baixo custo de implantação e gerenciamento. Os serviços online e aplicações incluem computação remota, software como serviço, armazenamento, entretenimento. Uma extensa revisão revelou o impacto nas áreas de tecnologia de informação (TI) industriais antes e depois da computação em nuvem, mostrando o avanço de tais serviços desde 1970 (Srivastava e Khan, 2018). Muitos trabalhos têm sido publicados apresentando novos paradigmas e aplicações. A importância de casar as contribuições da literatura com as reais necessidades da indústria foi avaliada entrevistando fornecedores e usuários de serviços de computação em nuvem (Venters e Whitley, 2012). Concluiu-se que tais serviços são difíceis de explorar para muitas empresas, embora elas anseiem por seus benefícios.

⋆ Os autores agradecem a bolsa de iniciação científica concedida pela UFES para a realização desse projeto.

Diversas soluções integrando à nuvem aos dispositivos de campo têm sido propostas. O trabalho de Bacci di Capaci e Scali (2020) é um exemplo de resultados concretos na utilização de algoritmos implantados e mantidos em nuvem para o monitoramento do desempenho de malhas de controle em sistemas SCADA. Os dados são recebidos e tratados na nuvem, e as malhas de controle são avaliadas, disponibilizando um relatório de seus desempenhos.

Em Arévalo et al. (2018) algoritmos para o monitoramento da condição de um sistema de armazenamento de produtos a granel foram implementados em uma arquitetura baseada em nuvem, discutindo alternativas para esta arquitetura. A entrega de controle como serviço via nuvem foi proposta por Givehchi et al. (2014), utilizando CLPs virtuais para realizar o controle de plantas industriais.

Tem-se observado um uso crescente de algoritmos usando inteligência artificial para monitoramento de desempenho e detecção de falhas executados na nuvem com dados fornecidos continuamente por CLPs (Hasır et al., 2021). Os controladores lógicos programáveis (CLP) estão presentes na grande maioria de sistemas de automação industrial. Os dados de sensores e atuadores utilizados para controle e monitoramento são usualmente disponibilizados a sistemas de armazenamento ou otimização via servidores de dados padrão OPC. Este mesmo padrão tem sido utilizado para a comunicação de sistemas de computação em nuvem (Beño et al., 2019). Outra alternativa que tem sido bastante utilizada para a comunicação do CLP com a nuvem é o protocolo MQTT (*Message Queeing Telemetry Transport*). Em Amoretti et al. (2020) esta alternativa somada a proposta de autenticação das mensagens foi discutida e avaliada. Em Gavlas et al. (2018) uma planta conectada a um CLP foi monitorada e controlada remotamente na nuvem, usando para isso o ambiente `Node-RED` para a visualização e interface com o usuário.

Nesse artigo, um ambiente em nuvem foi desenvolvido para fornecer diagnósticos de operação para uma planta industrial conectada a um sistema SCADA. O CLP utilizado e seu ambiente de programação permitem a utilização dos protocolos MQTT e OPC para a transmissão de dados. A escolha pelo MQTT foi baseada na comparação entre esses dois padrões de comunicação realizada em Rocha et al. (2019). O ambiente em nuvem foi utilizado para obter modelos baseados em dados para monitorar a planta. Os diagnósticos são retornados ao CLP e podem ser visualizados nas telas de operação. O ambiente em nuvem é acessado via navegador permitindo a parametrizacão e acesso aos relatórios gerados. Várias decisões relacionadas a provedor, banco de dados, visualização e interconexão, aliadas ao uso de softwares livres e uma grande variedade de bibliotecas disponíveis foram consideradas para o desenvolvimento. Uma aplicação de detecção de falhas para um sistema de controle de nível de um reservatório mostra a forma transparante como os recursos computacionais em nuvem podem enriquecer a operação um sistema de operação SCADA tradicional.

2. PROTOCOLOS PARA COMUNICAÇÃO ENTRE CLP E NUVEM

No contexto da indústria 4.0 e da IoT, diversos são os protocolos utilizados para transporte de informação. Exemplos desses são os protocolos: Transporte Enfileirado de Mensagens por Telemetria (MQTT, do inglês *Message Queeing Telemetry Transport*), Plataforma de Comunicação Aberta de Arquitetura Unificada (OPC UA, do inglês *Open Platform Communications Unified Architecture*), Sistema de Operação de Robôs (ROS, do inglês *Robot Operating System*) e sistemas de distribuição de dados (DDS, em inglês Data Distribution System) (Profanter et al., 2019).

Rocha et al. (2019) realizou um estudo comparativo entre dois desses protocolos: OPC-UA e MQTT. Testes sobre a carga útil das mensagens, tempo de entrega e número de clientes foram realizados. O MQTT para todas as qualidades de serviço (QoS, do inglês *Qualities of Service*) apresentou resultados melhores ou semelhantes ao OPC UA. Isso ocorre porque a pilha de protocolos deste é complexa, e ele apresenta diversos serviços na troca de dados, ao contrário do MQTT, que não é estruturado.

O MQTT é um protocolo aberto de mensagens criado pela IBM no ano de 1999. Ele é muito atraente aos desenvolvedores de serviços de IoT, por ser leve e flexível, e permitir a comunicação assíncrona entre os componentes do sistema, o que possibilita leves interrupções na conexão entre os dispositivos sem perdas significativas. Além disso, o protocolo exige muito pouco da rede para comunicação, o que reforça o seu uso em aplicações onde a largura de banda pode ser limitada e a latência pode ser alta (IBM e Eurotech, s.d.).

Toda conexão via MQTT dispõe de dois tipos de agentes: os clientes e o intermediador (*broker*). Os clientes são os hardwares conectados à internet que trocam informações (mensagens) via protocolo. Eles podem ser publicadores (*publishers*) e/ou assinantes (*subscribers*) (Kashyap et al., 2018). Os publicadores enviam e os assinantes recebem as mensagens. Os clientes podem ser controladores, softwares de programação, dispositivos supervisionados ou até mesmo sensores.

O intermediador é um servidor, geralmente executado em um serviço de nuvem, que recebe as mensagens dos publicadores juntamente com o tópico de publicação e envia para os assinantes que solicitaram as mensagens daquele tópico. Essa estrutura permite desacoplar os clientes entre si, tornando necessário apenas o endereço do intermediador e o tópico de publicação para troca de mensagens, o que possibilita que a comunicação possa ser facilmente diversificada quanto ao número de assinantes e quanto ao número de publicadores.

3. SERVIÇOS DE NUVEM

A computação em nuvem pode ser considerada um modo de computação terceirizado onde os recursos computacionais são agrupados em grandes centros de dados externos e acessados pelos clientes através da Internet, geralmente apresentando facilidade em lidar com grandes quantidades de dados. Além da computação em nuvem, há também a computação em borda, onde a fonte dos dados e a máquina responsável pelo processamento são posicionadas próximas, o que permite menores latências, se comparada à computação em nuvem, além de maior cibersegurança, já que os dados não saem da empresa. Por fim, a computação

em névoa é também um modo de computação terceirizado muito utilizado. Nela a distância do servidor fica entre as distâncias da computação em nuvem e em borda, sendo uma possível solução para processamentos de conjuntos de dados de tamanho intermediário, além de ser uma proposta para soluções que permitam de latência média (Zhang et al., 2021).

Tratando-se da computação em nuvem, existem diversos serviços que podem ser oferecidos por esta. Quando se carrega uma aplicação via navegador na internet ao invés de baixar o software em uma CPU física, tem-se o que se chama de programa como serviço (*Software as a Service - SaaS*). Famosos exemplos de SaaS são Google, Facebook, Overleaf. Já a disponibilidade de um ambiente de desenvolvimento onde o usuário tem a liberdade de utilização e produção de seus próprios códigos, trata-se da plataforma como serviço (*Platform as a Service - PaaS*). Exemplos deste tipo de serviço são Linux, Apache, MySQL, Ruby. Por fim, tem-se o que a literatura chama de infraestrutura como serviço (*Infrastructure as a Service - IaaS*), onde os provedores disponibilizam armazenamento, rede, sistema operacional, hardware, tudo sob demanda, o que permite uma escalabilidade do serviço, pagando de acordo com a quantidade utilizada. Grandes plataformas fornecedoras deste tipo de serviço são a Amazon Web Services (AWS), a International Business Machine (IBM) Watson, a Microsoft Azure e a DigitalOcean.

No âmbito das máquinas virtuais em nuvem enquadradas na categoria IaaS, métodos para acesso remoto ao terminal ou para a emulação da área de trabalho da CPU são necessários. O SSH (*Secure Shell*) é o protocolo para acesso a terminais de máquinas mais utilizado. Ele é um protocolo de rede criptográfico fornecedor de um canal seguro para comunicação em uma rede insegura.

Quanto aos programas como serviço (SaaS), o intermediador deve ser instalado em nuvem para a utilização do protocolo MQTT. Exemplos de intermediadores disponíveis são: `mosquitto`, `HiveMQ`, `ActiveMQ` e `MQTTRoute`. Em um teste comparativo feito por Mishra e Mishra (2020) com os quatro citados, o `mosquitto` é o que apresenta a maior vantagem quanto a latência média e utilização da CPU em todas as QoS do MQTT, além de ser um software grátis e de código aberto(Eclipse Foundation, 2009).

No trabalho presente, foram utilizadas nuvens do tipo SaaS, os programas utilizados para implementação da operação remota; e IaaS, a máquina virtual em nuvem onde os softwares serão instalados e executados. O sistema operacional `Ubuntu` foi selecionado. A provedora selecionada para o serviço foi a `DigitalOcean`, por ter apresentado a melhor relação custo-benefício para esta aplicação. Este provedor também foi escolhido em Arévalo et al. (2018).

O acesso ao terminal da máquina foi realizado pelo SSH nativo do Windows. A partir do acesso ao terminal foram instalados o intermediador `mosquitto`, o banco de dados `InfluxDB` , o serviço de visualização de dados temporais `Grafana` e a ferramenta de programação visual `Node-RED` (InfluxData, 2013; Grafana Labs, 2014). Para que os softwares não se encerrassem ao fechar o terminal, o comando `setsid` foi utilizado para executá-los inicialmente. É importante ressaltar que o acesso às interfaces gráficas do `Node-RED` e do `Grafana` é feito via URL que contém o IP do servidor e a porta utilizada pelo serviço.

Após devidamente instalados e inicializados, algumas configurações foram feitas feitas para efetivar a conexão. Dentro da interface do `Node-RED` existem blocos de funções nativas - `mqtt in` e `mqtt out` - que permitem o recebimento e o envio de dados para os tópicos desejados. Como o `Node-RED` é uma ferramenta de programação visual, existem algumas limitações quanto a criação de algoritmos de tratamento de dados e de detecção de falhas. Portanto, foi utilizado um componente auxiliar do `Node-RED`, o `pynodered`, que permite a utilização de funções criadas em `Python` na interface de programação visual.

Uma das grandes vantagens de utilizar esse tipo de ferramenta de programação visual é a possibilidade de criar interfaces de interação com usuário apenas utilizando blocos. Para isso, dentro do `Node-RED` a biblioteca adicional `node-red-dashboard` foi instalada. Ela permite a utilização de diversos recursos visuais facilmente implementáveis, os quais foram utilizados para criar a interface.

Quanto ao manuseio de dados disponíveis no `InfluxDB`, a biblioteca `influxdb` do `Python` foi utilizada, permitindo a escolha e captura de uma série temporal pelo usuário. O `Grafana` foi utilizado para visualização e análise gráfica dos dados no banco de dados, tendo em vista a riqueza de recursos que dispõe e a qualidade da apresentação visual. Nesse trabalho, ele foi utilizado como uma ferramenta auxiliar para análise dos dados armazenados e seleção de trechos de dados para treinamento dos algoritmos de detecção de falhas.

4. COMUNICAÇÃO DO CLP COM A NUVEM

O OPC UA é o padrão de conectividade para a indústria 4.0 e pode ser usado em praticamente qualquer equipamento industrial, seja por disponibilidade nativa no dispositivo ou através de IIoT Edge Gateways. Muitos fabricantes de CLP têm introduzido no *firmware* desses equipamentos bibliotecas que implementam o protocolo MQTT, um protocolo mais leve mas com funcionalidades muito atraentes para aplicações em nuvem. Nesse trabalho, este foi o protocolo utilizado, nativo no CLP `Altus Nexto NX3005`. O processo de configuração é simples e tem início ativando a biblioteca nativa do CLP `LibMQTT`. As variáveis já em uso no código implementado no CLP que serão publicadas e assinadas via MQTT devem ser alocadas em tópicos de recepção (*subscribe*) ou envio (*publish*), de modo que possam ser acessadas em todo programa, sendo para isso declaradas como variáveis globais. Quanto aos tópicos, o usuário que envia os dados pode escolhê-los, e quem recebe tais dados apenas deve se conectar a estes de forma correta. Por fim, o número IP da máquina onde o intermediador está localizado deve ser informado.

Com essa configuração, o CLP estará pronto para enviar os dados para a aplicação de monitoramento desenvolvida na nuvem, e receber os diagnósticos, que podem ser visualizados nas telas de operação.

5. APLICAÇÃO E DISCUSSÃO

Na Figura 1 é mostrado o ambiente desenvolvido para a aplicação, que permite a operação e o monitoramento de uma planta industrial. Os recursos disponibilizados e sua implementação são agora apresentados. Considera-se nesta estrutura a operação local, da forma usual, e a operação em nuvem, com os acessos que serão descritos. A operação em nuvem visa coletar os dados de treinamento, avaliar os modelos, parametrizar os algoritmos, e iniciar o monitoramento na nuvem. Importante destacar que todas tarefas de operação em nuvem podem ser realizadas na mesma estação de trabalho do sistema SCADA.

5.1 Planta industrial

Foi utilizado um CLP da série `Nexto NX3005`. A solução apresentada pode ser aplicada a qualquer planta industrial compatível com os recursos deste CLP. Dadas as restrições de acesso aos laboratórios de pesquisa durante a pandemia, optou-se por simular um sistema de controle de nível de reservatório similar ao existente no laboratório, de modo que a solução proposta pudesse ser imediatamente utilizada na planta real substituindo as variáveis de simulação pelas tags dos instrumentos da planta real. A simulação é feita usando as equações e parâmetros que descrevem esse sistema. A planta (ver Figura 1) dispõe de um reservatório cilíndrico de $10cm$ de raio, $1m$ de altura e aproximadamente $31.4l$ de volume. Ele recebe água bombeada com vazão variável, para simular perturbações. O nível é medido por um transmissor e seu controle é feito por um controlador PI, que atua sobre uma válvula pneumática de controle, que disponibiliza a informação de posição (via comunicação Hart). O nível, a abertura da válvula e o modo de operação (*flag*) são as variáveis a serem publicadas. A referência de nível, o alarme de falha e o sinal liga/desliga da planta são as variáveis assinadas, sendo usadas quando a planta está sendo operada remotamente.

Para simular uma falha, foi introduzida uma válvula manual no reservatório que emula um vazamento. Esta variável será também assinada pelo CLP, permitindo a inserção das falhas com o objetivo de gerar dados para treinamento e teste de algoritmos de detecção de falhas.

5.2 Operação local e em nuvem

A simulação e controle de nível do reservatório são feitos pelo CLP. Em operação local, o operador escolhe a referência de nível e os parâmetros do controlador PI que controla a abertura da válvula de forma a alcançar o nível especificado. A vazão de entrada é uma perturbação que o operador pode gerar de forma a avaliar o desempenho do controle de nível. A tela de operação local (ver Figura 2) contém todos esses elementos usados para a operação. Ela foi desenvolvida no ambiente de programação do CLP (`Mastertool`).

Para a operação em nuvem, o operador local pressiona o botão e uma *flag* é ativada. A partir disso, a escolha da referência de nível, vazão de entrada e vazão de falha (todas variáveis assinadas) passam a ser realizadas na interface gráfica em execução na nuvem (ver canto superior esquerdo na Figura 3). O nível e a abertura da válvula são

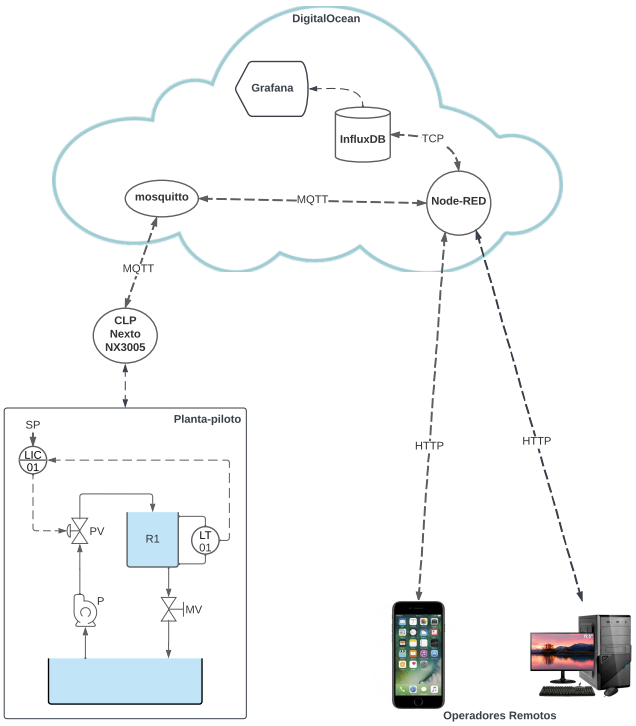

Figura 1. Elementos que compõem a aplicação.

Figura 2. Tela de operação local.

informadas nas telas de operação local e remota (Figura 1). A qualquer momento, o operador local pode retornar a operação para o modo local. A operação em nuvem visa principalmente o treinamento e uso dos algoritmos de detecção de falhas. Para a execução dessas atividades, não é necessário que a operação esteja no modo remoto, pois todos os algoritmos e dados estão disponíveis na nuvem.

A tela de acesso ao ambiente de nuvem, via navegador, é mostrada na Figura 3. O operador deve ter o link bem como o login e a senha, o que garante a segurança de acesso.

5.3 Armazenamento de dados

Os dados que são armazenados continuamente via `Influx-DB` são: nível, abertura da válvula, vazão de entrada e os resultados dos algoritmos de detecção de falhas. Tem-se

assim o histórico dos dados que pode ser visualizado a qualquer momento via `grafana`. Este ambiente de visualização de dados é facilmente configurado. O `InfluxDB` foi escolhido como o banco de dados a ser utilizado e o seu endereço deve ser informado. A partir disso as variáveis presentes no banco de dados estão disponíveis para serem manuseadas e apresentadas.

5.4 Treinamento para detecção de falha

Uma característica de grande valor desse ambiente é a possibilidade de execução de algoritmos escritos em linguagem Python para processar e analisar os dados armazenados no banco de dados ou que chegam do CLP. Nessa aplicação, foram implementados dois algoritmos para detecção da falha. Um modelo baseado em regressão linear foi utilizado para estimar a abertura da válvula a partir das medidas de nível e da vazão de entrada. Uma carta de controle é utilizada para monitorar a média do erro entre a abertura da válvula e sua estimativa (Kruger e Xie, 2012). O segundo algoritmo de detecção de falha foi uma árvore de decisão, treinada com a abertura da válvula, nível, vazão de entrada (Aldrich e Auret, 2013). Nesse caso, foram introduzidas falhas e rotulados os dados de forma a viabilizar o treinamento.

Ambos os algoritmos foram implementados a partir da biblioteca `scikit-learn` do `Python` e geram os modelos (regressão linear e árvore de decisão) usados para detectar a falha. O alarme de falha é gerado quando o limiar da carta de controle é ultrapassado três vezes consecutivas. O mesmo procedimento é utilizado para a árvore de decisão, que gera o alarme após a terceira indicação seguida da falha. Posteriormente um sistema de votação pode ser utilizado para selecionar ou ponderar a saída desses algoritmos (não implementado nesse artigo).

O treinamento da regressão linear é feito com dados de operação normal. Os dados utilizados são armazenados no `InfluxDB`. Um modelo de regressão é treinado, o resíduo (erro de predição) é calculado, e a partir dele calcula-se os limites superior e inferior da carta de controle. A qualquer momento um novo treinamento pode ser realizado, selecionando um novo conjunto de dados de operação normal. Isso é aconselhável quando o número de falsos positivos ultrapassa o nível de confiança (5%), devido, por exemplo, à operação da planta em uma região de operação onde não foi treinada. Para isso, basta selecionar o período no qual estão os dados que se quer incluir no treinamento. Eles serão concatenados aos dados existentes e um novo modelo de regressão é gerado. Usa-se validação cruzada, método *holdout*, para verificar a capacidade de generalização do modelo e gerar estatísticas para avaliar seu desempenho. Os dados são divididos aleatoriamente em dois grupos mutuamente exclusivos, com 2/3 das amostras para treino e 1/3 para teste. Este procedimento é repetido 20 vezes e a média e o desvio padrão da raiz quadrada média normalizada do erro é calculada para cada modelo, gerando 20 valores da qualidade do modelo (fit) dado pela (equação 1), que são representados por sua média e desvio padrão. Na equação 1, y é a abertura da válvula medida e $\bar{y}$ é seu valor médio. $r = 100\%$ indica um ajuste perfeito do modelo entre y e $\hat{y}$, o valor estimado pelo modelo da regressão linear.

$$r = 100\left(1 - \frac{|| y - \hat{y} ||}{|| y - \bar{y} ||}\right) \quad (1)$$

O fit r obtido é comparado ao obtido pelo modelo anterior. Caso o novo modelo tenha desempenho semelhante, ele passa a ser utilizado para detecção de falhas.

Para o treinamento da árvore de decisão são necessários dados rotulados, gerados no início da operação tanto em normalidade quanto em falha. A validação cruzada é utilizada aqui da mesma forma que na regressão linear, porém calculando a média e o desvio padrão da acurácia e da precisão do classificador. Havendo a necessidade de um novo treinamento, devido ao aumento dos falsos positivos, os dados de operação normal são assim rotulados e concatenados aos dados existentes na base de dados. Um novo treinamento é feito, comparando a qualidade dos modelos, e caso o usuário aceite, o novo modelo passa a ser utilizado para detecção de falhas.

Na figura 3 são mostrados os resultados dos algoritmos de detecção de falhas. Na carta de controle é mostrada a média do resíduo e os limiares superior e inferior, calculados de acordo com (Kruger e Xie, 2012, p. 8). Os resultados da carta de controle e da árvore de decisão são convertidos em alarmes no gráfico correspondente. Eles assumem valor 1 quando três ultrapassagens de limiar consecutivas ocorrerem para a carta de controle, ou três classificações de falha ocorrerem no caso da árvore de decisão. Estes alarmes são enviados para o CLP e são mostrados na tela de operação (ver Figura 2).

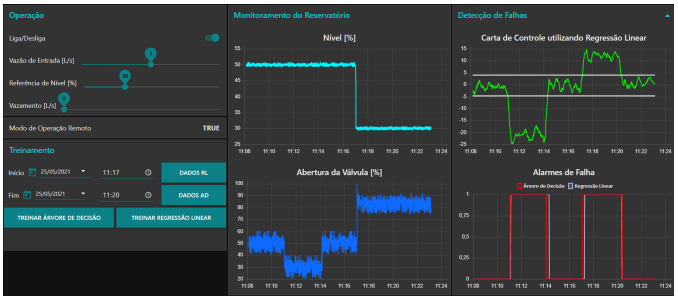

Figura 3. Exemplo de operação.

5.5 Discussões

As figuras 2 e 3 resumem a proposta de monitoramento de processos industriais em plataforma de nuvem. A planta pode ser operada localmente usando a tela mostrada na Figura 2. Caso o operador pressione o botão de operação remota, a planta passa a ser comandada via interface de nuvem, que usa para isso a tela mostrada na Figura 3. Ele pode alterar a referência de nível e a vazão de entrada, para mudar o ponto de operação da planta, avaliando seu efeito na tela mostrada na Figura 3. Pode também introduzir falhas, gerando assim dados para treinamento dos algoritmos de detecção.

Para o monitoramento de falhas e treinamento dos algoritmos a planta pode estar operando tanto no modo local quanto no modo remoto, pois os dados e algoritmos utilizados estão na nuvem. Havendo falhas, elas são indicadas tanto na interface de nuvem (Figura 3) quanto na tela local de operação (Figura 2).

A grande vantagem do ambiente na nuvem é a possibilidade de treinar os algoritmos de detecção de falhas, uma vez que tem mais recursos e é mais adequado para fazê-lo. Havendo falhas indicadas erroneamente na tela de operação (Figura 2), basta acessar a tela remota e refazer o treino dos modelos de detecção de falhas, ou então, utilizar diferentes algoritmos já disponíveis na biblioteca Python. Na Figura 3 tais procedimentos são ilustrados. Os dados estão sendo armazenados a cada dois segundos no banco de dados. Até o instante 11:20, os algoritmos de detecção treinados inicialmente estão em execução. No instante 11:11 uma falha é introduzida, sendo prontamente detectada pelos dois algoritmos. Ela é removida (a falha é sanada) no instante 11:14, quando cessam os alarmes. Às 11:17 a referência de nível é reduzida de 50% para 30%. A carta de controle e a árvore de decisão indicam falha, pois não foram treinados para essa região de operação. Como o operador sabe que não se trata de falha, seleciona as 90 amostras dos últimos três minutos (11:17 à 11:20), que são concatenados aos dados armazenados para o treinamento anterior, e os dois algoritmos de detecção são novamente treinados e após 11:20 não indicam mais falhas. O desempenho dos modelos antes e depois do novo treinamento são avaliados e mostrados na Tabela 1. O *fit* foi utilizado como métrica de avaliação para regressão linear, e a acurácia e a precisão foram utilizadas para a árvore de decisão. Conclui-se então que o desempenho de ambos algoritmos para os dois pontos de operação se manteve próximo.

Tabela 1. Avaliação de desempenho de modelos obtidos.

Referência de nível	**Regressão Linear**	**Árvore de Decisão**	
	fit	Acurácia	Precisão
50%	77.6%(±3.5)	95.9%(±3.1)	96.1%(±2.9)
30%	66.7%(±5.06)	91.7%(±5.9)	92.5%(±4.8)

Novas variáveis da planta podem ser incluídas na base de dados e novos algoritmos de monitoramento podem ser implementados utilizando esses dados, de forma similar.

6. CONCLUSÕES

Uma proposta para o monitoramento de processos industriais em plataforma de nuvem usando protocolo MQTT foi apresentada. Um CLP no qual as bibliotecas com esse protocolo estavam implementadas foi configurado para publicar e assinar variáveis em um intermediador na nuvem. Na nuvem, foram selecionados e configurados o intermediador `mosquitto`, o banco de dados para séries temporais `InfluxDB`, os softwares `Grafana` e `Node-RED`. As variáveis usadas para monitoramento foram persistidas no banco de dados e dois algoritmos de detecção de falhas foram programados em linguagem Python. Os diagnósticos são comunicados diretamente ao CLP que pode mostrá-los na tela de operação. Uma interface gráfica desenvolvida no `Node-RED` permite parametrizar e treinar os modelos usados para diagnóstico. Uma aplicação para o monitoramento de uma planta de controle de nível foi mostrada, onde se observa que os serviços de nuvem são executados de forma transparente para o sistema SCADA. O estudo de caso tratado nesse trabalho pode ser facilmente estendido a qualquer planta industrial. Tanto a comunicação via MQTT quanto via OPC podem ser utilizadas. Os recursos computacionais usados no ambiente de nuvem estão disponíveis na maioria dos provedores deste tipo de serviço. As escolhas feitas neste trabalho levaram em consideração o CLP de pequeno porte utilizado e o baixo esforço computacional requerido para a aplicação.

7. AGRADECIMENTOS

Os autores agradecem a bolsa de iniciação científica concedida pela Ufes para a realização desse projeto, e ao suporte dado pela Altus para uso da biblioteca LibMQTT.

REFERÊNCIAS

Aldrich, C. e Auret, L. (2013). *Unsupervised process monitoring and fault diagnosis with machine learning methods*. Springer.

Amoretti, M., Pecori, R., Protskaya, Y., Veltri, L., e Zanichelli, F. (2020). A scalable and secure publish/subscribe-based framework for industrial iot. *IEEE Transactions on Industrial Informatics*, 17(6), 3815–3825.

Arévalo, F., Diprasetya, M.R., e Schwung, A. (2018). A cloud-based architecture for condition monitoring based on machine learning. In *2018 IEEE 16th International Conference on Industrial Informatics (INDIN)*, 163–168. IEEE.

Bacci di Capaci, R. e Scali, C. (2020). A cloud-based monitoring system for performance assessment of industrial plants. *Industrial & Engineering Chemistry Research*, 59(6), 2341–2352.

Beňo, L., Pribiš, R., e Leskovskỳ, R. (2019). Processing data from opc ua server by using edge and cloud computing. *IFAC-PapersOnLine*, 52(27), 240–245.

Eclipse Foundation (2009). mosquitto. Acessado em: 14 de maio de 2021. Disponível em: <https://mosquitto.org/>.

Gavlas, A., Zwierzyna, J., e Koziorek, J. (2018). Possibilities of transfer process data from plc to cloud platforms based on iot. *IFAC-PapersOnLine*, 51(6), 156–161.

Givehchi, O., Imtiaz, J., Trsek, H., e Jasperneite, J. (2014). Control-as-a-service from the cloud: A case study for using virtualized plcs. In *2014 10th IEEE Workshop on Factory Communication Systems (WFCS 2014)*, 1–4. IEEE.

Grafana Labs (2014). Grafana. Acessado em: 14 de maio de 2021. Disponível em: <https://grafana.com/>.

Hasır, M., Cekli, S., e Uzunoğlu, C. (2021). Simultaneous remote monitoring of transformers' ambient parameters by using iot. *Internet of Things*, 14, 100390.

IBM e Eurotech (s.d.). Mqtt v3.1 protocol specification. Acessado em: 14 de maio de 2021. Disponível em: <https://public.dhe.ibm.com/software/dw/webservices/ws-mqtt/mqtt-v3r1.html>.

InfluxData (2013). Influxdb. Acessado em: 14 de maio de 2021. Disponível em: <https://www.influxdata.com/>.

Kashyap, M., Sharma, V., e Gupta, N. (2018). Taking mqtt and nodemcu to iot: communication in internet of things. *Procedia computer science*, 132, 1611–1618.

Kruger, U. e Xie, L. (2012). *Statistical monitoring of complex multivatiate processes: with applications in industrial process control*. John Wiley Sons.

Mishra, B. e Mishra, B. (2020). Evaluating and analyzing mqtt brokers with stress testing. In *Proceedings of the*

12th Conference of PHD Students in Computer Science, CSCS, 32–35.

Profanter, S., Tekat, A., Dorofeev, K., Rickert, M., e Knoll, A. (2019). Opc ua versus ros, dds, and mqtt: performance evaluation of industry 4.0 protocols. In *2019 IEEE International Conference on Industrial Technology (ICIT)*, 955–962. IEEE.

Rocha, M.S., Sestito, G.S., Dias, A.L., Turcato, A.C., Brandão, D., e Ferrari, P. (2019). On the performance of opc ua and mqtt for data exchange between industrial plants and cloud servers. *ACTA IMEKO*, 8(2), 80–87.

Srivastava, P. e Khan, R. (2018). A review paper on cloud computing. *International Journal of Advanced Research in Computer Science and Software Engineering*, 8(6), 17–20.

Venters, W. e Whitley, E.A. (2012). A critical review of cloud computing: researching desires and realities. *Journal of Information Technology*, 27(3), 179–197.

Zhang, T., Li, Y., e Chen, C.P. (2021). Edge computing and its role in industrial internet: Methodologies, applications, and future directions. *Information Sciences*, 557, 34–65.