# Redundância de Controladores de Processo Industriais como um Serviço

**Lucas B. Rodrigues de Sá** [*] **Ricardo P. Pontarolli** [**]
**Eduardo P. Godoy** [***]

[*] *Centro Industrial Nuclear de Aramar (CINA), Marinha do Brasil, Iperó-SP, (lucas.sa@marinha.mil.br).*
[**] *Instituto Federal de Educação, Ciência e Tecnologia de São Paulo (IFSP) , Boituva-SP, (pasquati@ifsp.edu.br)*
[***] *Universidade Estadual Paulista (Unesp), Sorocaba-SP, (eduardo.godoy@unesp.br)*

**Abstract:** The Industry 4.0 starts the digitalization process and for this have been using new technologies as Internet of Things, Cloud Computing, Service Oriented Architecture and Control Systems in network. The integrated use of these technologies in automation and process control applications provide advantages, such as vertical interoperability and standardization of communication through networked service sharing. In this article, the redundancy of industrial process controllers is studied and the development of a redundant controller is investigated using an automation architecture oriented to microservices. The redundant controller is based on a modification of the *PIDPlus* algorithm to network control for operation as a microservice. Additionally, a redundancy application (orchestrator) is responsible for orchestrating microservices and recording control variables, allowing the sharing of data from process variables among redundant controllers (replica). The experimental results obtained in a pilot plant in different scenarios demonstrated feasibility and efficacy in the redundancy implementation of process controllers as a service, as well as the advantages of this type of implementation.

**Resumo**: A Indústria 4.0 iniciou um processo de digitalização e para isso vêm utilizando novas tecnologias como a Internet das Coisas, Computação em nuvem, Arquitetura Orientada a Serviços e Sistemas de Controle Via Rede. O uso integrado dessas tecnologias em aplicações de automação e controle de processos proporcionam vantagens, como a interoperabilidade vertical e a padronização da comunicação, através do compartilhamento de serviços em rede. Nesse artigo é estudado a redundância de controladores de processos industriais e investigado o desenvolvimento de um controlador redundante usando uma arquitetura de automação orientada a microsserviços. O controlador redundante é baseado numa modificação do algoritmo *PIDPlus* para controle em rede para operação como um microsserviço. Adicionalmente, uma aplicação de redundância (orquestrador) é responsável pela orquestração dos microsserviços e o registro das variáveis de controle, permitindo o compartilhamento de dados das variáveis de processo entre os controladores redundantes (em réplica). Os resultados experimentais obtidos numa planta piloto em diferentes cenários demonstraram viabilidade e eficácia na implementação de redundância de controladores de processo como um serviço, bem como as vantagens deste tipo de implementação.

*Keywords:* Industry 4.0; Control Systems; Service Oriented Architectur; Controler Redundancy; Hardware.

*Palavras-chaves:* Indústria 4.0; Sistemas de Controle; Arquitetura Orientada a Serviços; Redundância de Controlador; Hardware.

## 1. INTRODUÇÃO

A automação industrial está evoluindo rapidamente chegando em uma nova revolução chamada Indústria 4.0, através da integração de novas tecnologias com o intuito de aumentar a eficiência e produtividade. Essa revolução é baseada na convergência entre tecnologias da automação (TA) e tecnologias da informação (TI). Para melhorar os processos industrias é fundamental utilizar tecnologias de automação, que com sua alta capacidade em acelerar processos de manufatura e produção, obtém maior eficiência e qualidade com menores custos e tempos. Dessa forma, a indústria tem evoluído de forma a incorporar novas soluções de TI e consequentemente obter maior produtividade (Sauter et al., 2011; Bangemann et al., 2014).

O desenvolvimento e evolução das tecnologias digitais contribuíram para a criação de novos métodos de produção industrial baseados na automação, robótica, inteligência

artificial, Internet das Coisas e inteligência de dados, dentre outras inovações. Dentro da indústria a utilização dessas tecnologias de forma coordenada a fim de aumentar a competitividade dos negócios, otimizar a eficiência da cadeia produtiva, agregar valor ao produto, utilizar os recursos de forma mais sustentável e customizar as soluções tecnológicas é chamada de Indústria 4.0 (I4.0).

Delsing (2017) apresenta uma Arquitetura Orientada a Serviços (SOA) como infraestrutura de integração e comunicação, utilizando o conceito de nuvem em uma rede local. Nessa arquitetura, os componentes e dispositivos de automação são disponibilizados como serviços, provendo interoperabilidade total entre as aplicações industriais e desta forma propõe um modelo de arquitetura em nuvem para substituir o antigo ISA-95, trazendo maior flexibilidade entre os dispositivos IoT.

SOA representa uma arquitetura de TI onde se busca fragmentar e modularizar aplicações convencionais em serviços (Xiao et al., 2016). Cada serviço é independente de plataforma ou linguagem de desenvolvimento, fornecendo uma arquitetura escalável e em rede (Theorin et al., 2014). Microsserviços representam uma evolução do SOA, onde temos serviços com poucas responsabilidades, pequenos e autônomos, que pode trabalhar em conjunto ou de forma independente com outros serviços.

A adoção das arquiteturas orientadas a serviços em aplicações industriais também forneceu vantagens significativas, como a interoperabilidade vertical, a padronização da comunicação e a virtualização de soluções (Fernandes et al., 2021). Por exemplo, tornou-se possível virtualizar equipamentos e dispositivos industriais, disponibilizando seus recursos e funcionalidades como um serviço (Givehchi et al., 2014).

Redundância, na automação industrial, significa manter sistemas redundantes (ex: duplicados) para garantir a disponibilidade de processos e dispositivos críticos, sendo utilizada em uma variedade de segmentos da indústria, principalmente de processos. No entanto, aplicações com redundância são mais complexas de implantar que as tradicionais, além de impactar significativamente em maiores custos.

Com base nos avanços proporcionados pela I4.0, este artigo apresenta uma proposta de desenvolvimento de aplicações de automação industrial como um serviço, dando continuidade ao trabalho de Pontarolli et al. (2020). Dessa forma, apresenta-se o desenvolvimento de controlador redundante de processos como um microsserviço, como uma alternativa à implantação tradicional de redundância de controladores industriais.

Este artigo será apresentado da seguinte forma, a seção 2 apresenta uma revisão de literatura sobre aplicações de virtualização de controle e redundância. Na seção 3 será apresentado a arquitetura orientada a microsserviços utilizada, na seção 4 o desenvolvimento dos serviços e aplicações. A seção 5 discute e compara os resultados dos serviços e aplicações de controle de processos com o controlador redundante desenvolvido. Por fim na seção 6 as conclusões, contribuições e futuras aplicações.

## 2. VIRTUALIZAÇÃO DE CONTROLE E REDUNDÂNCIA

A virtualização pode fornecer uma camada de abstração para o *software* aumentando a agilidade do sistema, além de fornecer um controle melhor dos recursos distribuídos em diferentes aplicações/sistemas operacionais. Em aplicações de automação e controle, os equipamentos industriais representam *hardware* e *software* específico, dedicado e de custo elevado. Dessa forma, a virtualização desses equipamentos pode originar vantagens relevantes a nível de comissionamento, flexibilidade e custo de implantação.

Givehchi et al. (2014) apresentam uma virtualização de um Controlador Lógico Programável (CLP). O artigo avalia e demonstra que é possível virtualizar a funcionalidade de um CLP em um software e este ter um desempenho compatível a um CLP com hardware dedicado. Azarmipour et al. (2019) também apresentam uma virtualização de um CLP, chamado de PLC 4.0, porém desenvolvendo uma nova arquitetura para este tipo de sistema, fazendo uma ponte entre aplicações de automação industrial e TI.

Fernandes et al. (2020) evoluíram as propostas desses trabalhos citados anteriormente através do desenvolvimento de um CLP como um serviço. Esta forma de implementação incorporou vantagens das arquiteturas orientadas a serviço para aplicações de automação e controle, como a criação de aplicações industriais descentralizadas, com grande modularidade, escalabilidade e independência entre sistemas. A facilidade de replicação e reuso do serviço também foi destacada, fornecendo um potencial interessante para o desenvolvimento de aplicações de automação e controle com redundância.

Redundância em processos industriais objetiva garantir a operação contínua desse sistema. A forma mais comum de redundância consiste em manter uma "réplica"do sistema funcionando em paralelo. Em caso de falha do sistema original, a réplica entra em operação sendo capaz de executar as mesmas funções do original com o mesmo nível de confiabilidade. Nas aplicações de automação e controle, existem vários tipos de redundância, como redundância de CPU, de fonte de alimentação, de rede de comunicação e de equipamentos como controladores, sensores e atuadores.

A redundância de controladores usando tecnologias como a comunicação em rede e a computação em nuvem tem sido investigada recententemente. Hegazy and Heffeda (2015), apresentam uma proposta de redundância em nuvem usando controladores redundantes alocados em diferentes localidades, com um compensador de possíveis atrasos inerentes à rede. Para ter confiabilidade no sistema de controle com redundância em nuvem, foi proposto um algoritmo de tolerância a falha que roda de forma assíncrona nos controladores redundantes, da seguinte forma:

- Se o primeiro controlador falhar, o controlador secundário estará automaticamente pronto para assumir. Essa é uma importante garantia para controlador redundante. Automaticamente se temos uma redundância tripla e o primeiro e segundo falharem, automaticamente o terceiro irá assumir o controle.

- Se o primeiro controlador estiver disponível novamente este irá voltar a assumir o controle, forçando o controlador secundário a sair do controle.

## 3. ARQUITETURA ORIENTADA A MICROSSERVIÇOS (MOA)

Para o desenvolvimento dessa pesquisa foi usado o *framework Moleculer*, que contém ferramentas, bibliotecas, sistemas e componentes que simplificam o processo de desenvolvimento de uma solução baseada em microsserviços. O Moleculer é um *framework* de código aberto utilizado na criação de microsserviços em linguagem JavaScript para a plataforma Node.js.

Na Arquitetura Orientada a Microsserviço (MOA) do *framework Moleculer*, os serviços são executados em nós de forma individual, se comunicando via protocolo de comunicação (Transporter), ou seja, um intermediário de mensagem. Nesta arquitetura, a latência da comunicação pode ser um ponto de preocupação pois não é insignificante, mas os serviços podem ser replicados para proporcionar resiliência e tolerância a falhas.

Nessa arquitetura MOA do *Moleculer*, não existe nenhum tipo de hierarquia ou prioridade, fazendo com que todos os microsserviços tenham a mesma importância. Uma boa vantagem na utilização deste *framework* é que os microsserviços desenvolvidos possuem disponível um recurso automático de registro e descoberta. Assim é informado a todos os serviços existentes quando houver um novo serviço ou disponibilização de uma nova funcionalidade em um serviço. Outra funcionalidade importante é o balanceamento de carga automático, que distribui dinamicamente e de forma uniforme a comunicação entre os serviços.

## 4. DESENVOLVIMENTO DOS SERVIÇOS E APLICAÇÕES

Em uma arquitetura a microsserviços, a aplicação é desenvolvida através da composição dos microsserviços desenvolvidos. Com essa composição é definido quais microsserviços serão utilizados e sua sequência de execução para obtermos a funcionalidade desejada. Essa composição pode ser feita via Orquestração (onde uma aplicação maestro comanda a execução dos serviços) ou via Coreografia (onde os serviços conhecem previamente a sequência de execução). Neste trabalho foi utilizado o formato de Orquestração.

Para este artigo utilizaremos os microsserviços de controle e de aquisição de dados (DAQ) e o experimento será realizado em uma planta piloto descrita em Pontarolli et al. (2020). Os serviços foram embarcados em duas Raspberry Pi 3 e em um computador. Foi desenvolvido um microsserviço de controle com redundância (Controle redundante) utilizando uma versão modificada do algoritmo *PIDPlus* (Song et al., 2006), sendo responsável pelo controle de todas as malhas de controle. Para este serviço foi utilizado as seguintes equações no algoritmo para calcular o termo integrativo e derivativo, que são fundamentais para implementarmos a redundância:

$$I = I_{n-1} + (U_{n-1} - I_{n-1})(1 - \frac{-\Delta T}{e^T reset}) \qquad (1)$$

Onde: I = termo integrativo, $I_{n-1}$ = termo integrativo anterior, $U_{n-1}$ = saída do controlador anterior, $\Delta T$ = intervalo de tempo desde que o último valor medido foi recebido e $T_{reset}$ = constante de tempo da planta somado ao tempo morto.

$$D = K_D(\frac{e_n - e_{n-1}}{\Delta T}) \qquad (2)$$

Onde: D = termo derivativo, $K_D$ = ganho derivativo, $e_n$ = erro atual, $e_{n-1}$ = erro anterior.

Os termos das equações (1) e (2) são fundamentais para o desenvolvimento do serviço de controle com redundância. Para habilitar a operação do microsserviço, alguns desses itens das equações (1) e (2) devem ser armazenados em um registrador. Toda vez que o microsserviço for atuar é necessário que busque os dados no registrador e ao final do serviço que este seja atualizado. Os itens que devem ser armazenados são, erro = e, termo integrativo = I, saída do controlador = U e o tempo atual que será utilizado para calcular o $\Delta T$ do próximo serviço. O diferencial é que ao ser disponibilizado como um serviço, o microsserviço de controle redundante pode ser replicado em múltiplas plataformas como computador, nuvem ou sistema embarcado, gerando uma grande economia de *hardware*.

O microsserviço DAQ é responsável pela aquisição de dados de variáveis utilizando módulos de *hardware* colocados no processo. Este serviço tem a função de fazer a leitura de entradas (sensores) e outro para atualizar as saídas (atuadores).

A figura 1 faz uma ilustração de como os serviços estão alocados e distribuídos na planta, desta forma conseguiremos demonstrar a função de cada serviço e como ele atua em todo o processo.

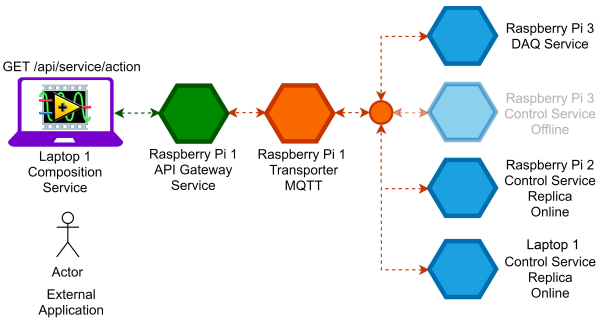


Figura 1. Estrutura dos Serviços.

Na figura 1 a parte representada pela aplicação externa é responsável por fazer a orquestração dos serviços, bem como o registro das variáveis utilizadas no controle e realizar a monitoração do processo. Essa aplicação foi desenvolvida utilizando o *software* Labview. Como essa aplicação é externa sua comunicação com os demais microsserviços é feita utilizando o padrão REST via o microsserviço Gateway representado pelo hexágono verde e embarcado na Raspberry Pi 1. A aplicação externa pode ser feita e desenvolvida em plataformas de *software* já consolidadas e tradicionais na industria.

O hexágono laranja representa o serviço de transporter e este faz a comunicação entre os microsserviços e entre estes e as possíveis aplicações externas. Está embarcado na Raspberry Pi 1 e faz a comunicação entre os serviços utilizando o MQTT. O microsserviço DAQ representado por um hexágono azul está embarcado na Rasberry Pi 3 e é responsável por fazer a aquisição de dados do processo conforme já mencionado.

O serviço de controle redundante, foco de estudo deste artigo, está representado pelo hexágono azul e tem um serviço alocado na Rasberry Pi 3, uma replica no computador e outra na Raspberry Pi 2. Este serviço tem a função de fazer o calculo de controle e atualizar as variáveis de controle no registrador, desta forma permitindo a redundância dos controladores embarcados em *hardwares* diferentes.

Neste artigo contamos com duas aplicações, a de monitoramento e a de controle com redundância. A aplicação de monitoramento faz a aquisição e monitoramento de dados das malhas do processo com o MOA. Os dados de processos são adquiridos através do serviço DAQ. Essa aplicação tem o objetivo de fazer uma interface para o monitoramento das variáveis de processo, contando com uma análise de comportamento e desempenho ao longo do tempo da ferramenta de controle. A aplicação de controle tem como responsabilidade a efetivação do controle em malha fechada das variáveis de interesse, através da composição dos serviços DAQ e controle redundante, além de sua função fundamental de redundância.

Na figura 2 apresentamos as etapas dos microsserviços, feita por uma aplicação externa, no nosso caso feita através do *software* Labview. Neste diagrama é possível verificar a sequência de execução dos microsserviços definida em cada aplicação. Neste passo a passo é possível observar todo o caminho realizado pelos pacotes de comunicação, desde a aplicação externa até o controle e sua atuação no atuador. Ao termos uma solicitação externa, a mesma será enviada para o API *Gateway* que encaminhará ao transporter e depois para o serviço DAQ que fará a leitura dos dados do processo, retornando a aplicação externa. Depois a aplicação externa enviará esses dados para o microsserviço de controle redundante, via API Gateway, que encaminhá ao transporter e este encaminhará para o serviço de controle retornando a aplicação externa. Por fim a aplicação externa novamente irá passar os dados via API Gateway, que transmitirá para o transporter, que enviará para o serviço DAQ, que atualizará os atuadores do processo e depois retornará a aplicação externa.

## 5. RESULTADOS E DISCUSSÕES

Para a análise dos resultados do serviço de controle redundante com redundância fizemos um experimento de controle de pressão de linha na planta piloto com os serviços 1, 2 e 3. Neste experimento contamos com três serviços que são capazes de realizar o controle do processo individualmente ou com qualquer configuração, todos os serviços on-line, dois ou apenas 1. Para demonstrar a operação da redundância de controle da malha de processo, o experimento começou com todos os serviços on-line e a cada 30 segundos geramos uma mudança na quantidade de serviços on-line de forma aleatória e essas etapas do

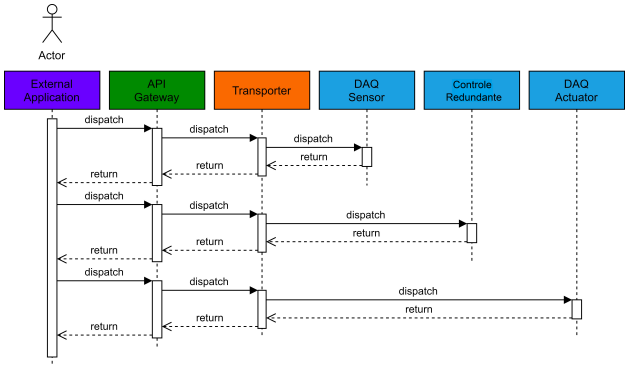


Figura 2. Sequência de comunicação dos Microsserviços.

experimento podem ser vistas no gráfico da figura 3, onde a linha em azul representa em cada intervalo de tempo quantos serviços estão on-line em cada instante.

O gráfico da figura 3, traça o perfil de *Setpoint* em azul e o da Variável de Processo em vermelho referente a malha de pressão da planta piloto. Com isso verificamos que o processo foi controlado conforme esperado e demandado pelo perfil de *Setpoint*, consolidando a eficácia do controle redundante realizado pelo controlador redundante. Este controle foi realizado no processo através do algorítimo desenvolvido em *JavaScript*.

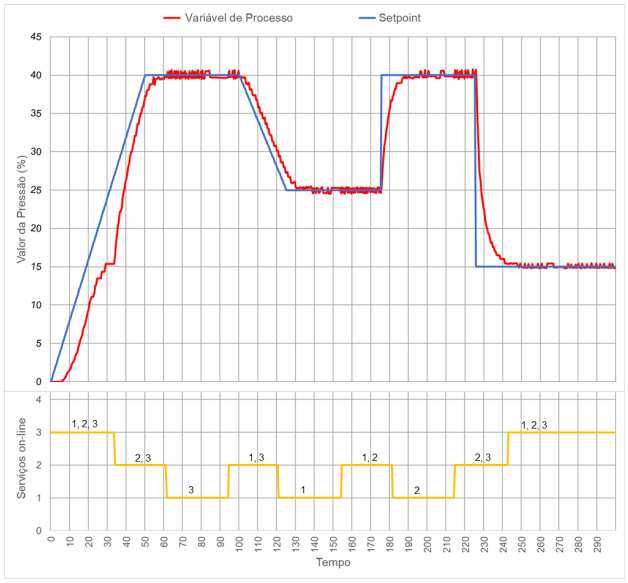


Figura 3. Controle malha de pressão.

Na figura 3 , através da linha em amarelo que mostra a quantidade de serviços on-line, temos uma analise que demonstra melhor como foi feito o experimento para consolidar a redundância do controlador redundante. Neste gráfico podemos observar por exemplo que nos primeiros 30 segundos temos os três serviços online representado pela linha amarela e acima desta linha quais serviços especificamente estão on-line naquele momento. Após os 30 segundos colocamos o serviço 1 em off-line, representado pela linha amarela que naquele momento de 30 a 60 segundos tínhamos 2 serviços on-line o 2 e 3. Passado mais 30 segundos colocamos o serviço 2 em off-line, restando somente o serviço 3 para realizar o controle e desta forma

fomos alterando a quantidade de serviços ao longo do experimento conforme pode ser visto na figura 3.

Com este experimento conseguimos simular uma possível falha em qualquer um dos serviços de controle, forçando que o mesmo fique off-line e é possível observar que a função de controle continua funcionando perfeitamente durante todo o processo, não tendo nenhuma influência a quantidade de serviços on-line naquele momento, ou seja, podemos ter diversos serviços redundantes on-line ao mesmo tempo.

Uma importante configuração fornecida pelo *Moleculer* a ser considerado é o balanceamento da rede, que define a política de requisição dos serviços existentes pelo Transporter. No caso dos controladores redundantes desse artigo, essa política define qual dos serviços (origianl ou réplicas) serão requisitados para operação em cada iteração da malha de controle. Os tipos possíveis de configuração são Round-Robin strategy, Random strategy, CPU usage-based strategy e Latency-based strategy:

- Round-Robin strategy - Com essa estrategia o nó é selecionado através de um algoritmo round-robin que vai chamando os serviços de forma sequencial, conforme sequência que foram colocados on-line por exemplo ($1 \rightarrow 2 \rightarrow 3 \rightarrow 1 \rightarrow 2 \rightarrow 3$).
- Random strategy - Utilizando essa estrategia o nó é selecionado de forma aleatória.
- CPU usage-based strategy - Utilizando essa estrategia o nó selecionado será o que tiver a menor utilização da CPU. Se tivermos uma grande quantidade de nós, essa estrategia irá fazer uma amostra e direcionar para o nó com menor uso de CPU, ao invés de fazer uma analise na CPU de todos os nós.
- Latency-based strategy - Com essa estrategia o nó que tiver a menor latência será o selecionado, que será medido através de comandos de ping feito periodicamente. Também se tivermos muitos nós será feito uma amostra e será selecionado o de menor latência.

Importante salientar que para o experimento realizado na figura 3 foi utilizado a estrategia Round-Robin o que significa que durante cada intervalo de tempo os serviços de controle foram requisitados de forma sequencial conforme os disponíveis naquele momento, por exemplo nos primeiros 30 segundos foi feito da seguinte forma $3 \rightarrow 2 \rightarrow 1 \rightarrow 3 \rightarrow 2 \rightarrow 1$.

Para verificarmos todas as estrategias de balanceamento da rede fizemos experimentos para cada uma das estrategias, o que será demonstrado através de gráficos. Para as estrategias Round-Robin e Random o experimento foi realizado com os três serviços on-line e conforme podemos observar na figura 4 para os primeiros 60 segundos, o primeiro gráfico da figura representa o Round-Robin, pois como podemos verificar no gráfico é seguido uma sequência ($3 \rightarrow 2 \rightarrow 1 \rightarrow 3 \rightarrow 2 \rightarrow 1$) ao longo de todo o período confirmando que funciona corretamente quando configurado. O segundo gráfico da figura 4 representa o Random, que tem como estrategia escolher de forma aleatória entre todos os serviços disponíveis naquele período o que podemos confirmar através do gráfico seu formato aleatório.

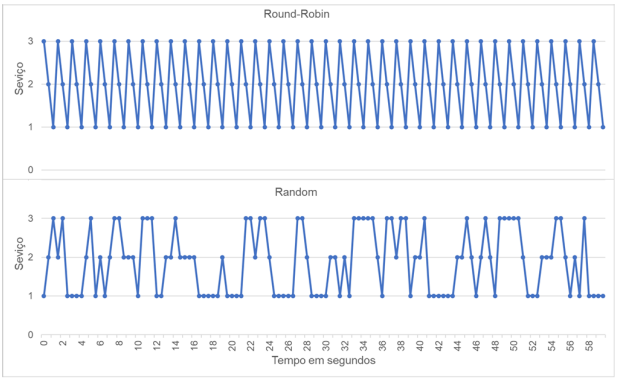


Figura 4. Estrategia Round-Robin e Random.

Para o experimento da estrategia CPU Usage, que verifica a porcentagem de uso de cada CPU dos serviços disponíveis e escolhe o de menor uso, foi instalado um programa que gera um stress na CPU de cada um dos *hardwares*. Começamos o experimento com os três serviços on-line e durante este período foi selecionado o serviço 3 por um período e o dois, porém após 40 segundos onde o programa que gera o stress na CPU do serviço 2 e 3 foi efetivado conforme figura 5, é possível observar que por todo o período em que o programa estava em funcionamento o serviço escolhido foi o 1 que estava com menor uso da CPU. Em 100 segundos foi retirado o stress da CPU dos serviços 2 e 3 e em aproximadamente 115 segundos o stress foi gerado na CPU dos serviços 1 e 3 conforme gráfico 5, fazendo com que o serviço escolhido durante este período fosse o 2 que tinha o menor uso da CPU. Por fim em 170 segundos foi retirado o stress da CPU dos serviços 1 e 3 e colocado nos serviços 1 e 2 fazendo com que o serviço escolhido fosse o 3 até o final do experimento pois era o que tinha o menor uso da CPU conforme demonstrado no gráfico 5.

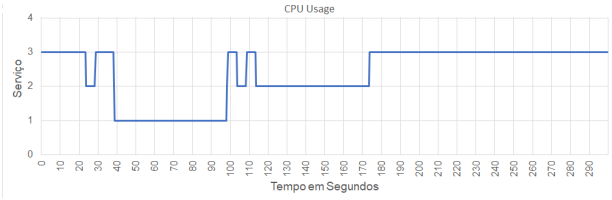


Figura 5. Estrategia CPU Usage

Por fim para a estrategia de latência que tem grande importância para este tipo de sistema baseado em serviços foi observado que com os três serviços on-line a tendência era escolher o serviço 1 que estava embarcado no computador e ligado a rede via cabo que tem menor latência do que os serviços 2 e 3 que estão embarcados nas *Raspbarry PI* e estão ligados à rede via wi-fi. Para esta estrategia não foi possível verificar com certeza que o serviço escolhido era o de menor latência, porque não foi possível segregar a rede e sobrecarregar somente o ponto onde queríamos realizar o teste, pois ao sobrecarregar em qualquer ponto afetava a rede como um todo e não trouxe o resultado esperado. Importante ressaltar que para todas as estrategias de balanceamento de rede testadas a curva da variável de processo foi semelhante a apresentada na figura 3.

Em aplicações de automação e controle, a comunicação em rede entre os serviços realizada pelo Transporter pode impactar no desempenho das malhas de controle. O Moleculer possui alguns tipos de transporters disponíveis a serem utilizados como TCP, NATs, Redis, MQTT, AMQP 1.0 e Kafka Transporter. Para este artigo utilizamos o MQTT transporter, mas para trabalhos futuros uma analise detalhada da influência de cada um desses tipos será importante.

## 6. CONCLUSÃO

A motivação deste artigo foi o desenvolvimento do microsserviço de controle redundante com o intuito de permitir a redundância deste tipo de controle, gerando uma grande economia de controladores físicos na planta e outros dispositivos, além de diminuir a complexidade do sistema de redundância. O microsserviço de controle flexibiliza e otimiza o desenvolvimento de algoritmo de controle conforme a necessidade do projeto, podendo ser adaptado com pequenas alterações. A malha de processo de pressão se mostrou adequada para o controle redundante via rede.

Com os experimentos realizados na planta piloto, conseguimos comprovar a efetividade da aplicação e validação do controlador redundante como um microsserviço. Os resultados apresentados demonstram que o conceito de controlador redundante desenvolvido também pode ser usado para implementação de outros tipos de controladores e processos. Para este trabalho, utilizamos até 3 serviços de controle redundantes de até 3 serviços de controle funcionando ao mesmo tempo em *hardwares* diferentes, podendo esse ser de maior número ou menor dependendo da aplicação.

Outra contribuição importante foi a utilização das ferramentas de balanceamento de rede disponíveis no *framework Moleculer*, o que permite uma personalização do projeto de automação, permitindo uma configuração conforme a necessidade do projeto.

## AGRADECIMENTOS

Os autores agradecem à Fundação de Amparo à Pesquisa do Estado de São Paulo (FAPESP), processo nº 2018/19984-4.

## REFERÊNCIAS

Azarmipour, M., Elfaham, H., Gries, C., and Epple, U. (2019). PLC 4.0: A Control System for Industry 4.0. *IEEE*. doi:978-1-7281-4878-6/19.

Bangemann, T., Karnouskos, S., Camp, R., Carlsson, O., Riedl, M., McLeod, S., Harrison, R., W., A., and Stluka, C. (2014). State of the art in industrial automation. *Industrial cloud-based cyber-physical systems*, 23–47. doi: 10.1007/978-3-319-05624-1_2.

Delsing, J. (2017). Local Cloud Internet of Things Automation. *IEEE INDUSTRIAL ELECTRONICS MAGAZINE,*. doi:10.1109/MIE.2017.2759342.

Fernandes, M.M., Bigheti, J.A., Pontarolli, R.P., and Godoy, E.P. (2020). Controlador Lógico Programável como um Serviço: Uma Proposta para a Indústria 4.0. *Congresso Brasileiro de Automática-CBA*, 2(1). doi:https://doi.org/10.48011/asba.v2i1.1740.

Fernandes, M.M., Bigheti, J.A., Pontarolli, R.P., and Godoy, E.P. (2021). Industrial Automation as a Service: A New Application to Industry 4.0. *IEEE Latin America Transactions*, 19(12), 2046–2053. doi:10.1109/TLA.2021.9480146.

Givehchi, O., Trsek, H., Jasperneite, J., and Imtiaz, J. (2014). Control-as-a-Service from the Cloud: A Case Study for using Virtualized PLCs. *IEEE*.

Hegazy, T. and Heffeda, M. (2015). Industrial Automation as a Cloud Service. *IEEE Transactions on Parallel and Distributed Systems*, 26(10). doi:pp.2750-2763.

Pontarolli, R.P., Bigheti, J.A., Risso, S.L., Domingues, F.O., Fernandes, M.M., and Godoy, E.P. (2020). Planta Piloto 4.0: Uma Abordagem para a Automação e Controle de Processos Orientada a Serviços. *Congresso Brasileiro de Automática - CBA*. doi:10.48011/asba.v2i1.1741.

Sauter, T., Stefan, S., Wolfgang, K., and Wolfgang, K. (2011). The Evolution of Factory and Building Automation. *IEEE Industrial Electronics Magazine*, 5(3), 35–48. doi:10.1109/mie.2011.942175.

Song, J., Mok, A.K., Chen, D., Nixon, M., Blevins, T., and Wojsznis, W. (2006). Improving PID Control with Unreliable Communications. *ISA EXPO 2006*.

Theorin, A., Hagsund, J., and Johnsson, C. (2014). Service orchestration with OPC UA in a graphical control language. *19th IEEE Int. Conf. Emerg. Technol. Fact. Autom.*

Xiao, Z., Wijegunaratne, I., and Qiang, X. (2016). Reflections on SOA and Microservices. *4th International Conference on Enterprise Systems*.