UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

GEOVANY APARECIDO DUARTE CÂNCIO

ARQUITETURA PARA A IMPLEMENTAÇÃO DE CONTROLE SUPERVISÓRIO COMUNICANTE

PATO BRANCO

2022

GEOVANY APARECIDO DUARTE CÂNCIO

ARQUITETURA PARA A IMPLEMENTAÇÃO DE CONTROLE SUPERVISÓRIO COMUNICANTE

Architecture for the implementation of communicating supervisory control

Trabalho de Conclusão de Curso apresentado como requisito para obtenção do título de Bacharel em Engenharia de Computação da Universidade Tecnológica Federal do Paraná (UTFPR).

Orientador: Prof. Dr. Marcelo Teixeira
Coorientador: Prof. Dr. Gustavo Weber Denardin

PATO BRANCO

2022

GEOVANY APARECIDO DUARTE CÂNCIO

ARQUITETURA PARA A IMPLEMENTAÇÃO DE CONTROLE SUPERVISÓRIO COMUNICANTE

Trabalho de Conclusão de Curso apresentado como requisito para obtenção do título de Bacharel em Engenharia de Computação da Universidade Tecnológica Federal do Paraná (UTFPR).

Data de Aprovação: 13 de setembro de 2022.

Prof. Dr. Marcelo Teixeira
Universidade Tecnológica Federal do Paraná

Prof. Dr. Cesar Rafael Claure Torrico
Universidade Tecnológica Federal do Paraná

Prof. Msc. Luis Cassiano Goularte Rista
Universidade Tecnológica Federal do Paraná

PATO BRANCO

2022

AGRADECIMENTOS

Agradeço primeiramente a toda minha família, pelo apoio e incentivo na busca pelos meus sonhos, que em todos os momentos estiveram ao meu lado e me deram forças nesta jornada. Meus pais, irmãos, primos, tios e minhas avós, em especial a memória de Eulina Duarte Correa, que partiu, mas merece meus mais sinceros agradecimentos por toda a cumplicidade dedicada a mim enquanto ainda em vida, nesses anos de graduação.

Agradeço ao meu orientador, professor Marcelo Teixeira, por ter aceitado me guiar nesta jornada, me auxiliando de todas as formas possíveis desde o início até a conclusão deste trabalho, e que ainda teremos uma longa e produtiva jornada com novos objetivos para serem alcançados, agora no mestrado. E ao coorientador, professor Gustavo Weber Denardin, pelos direcionamentos técnicos deste trabalho.

Agradeço a todos os professores que me instruíram com ótimas aulas e toda a dedicação para repassar seus conhecimentos. Em especial a professora Kathya Silvia Collazos Linares, que me acompanhou desde o início do curso e teve atitudes de uma grande e pura pessoa, principalmente quando me visitou no hospital após uma delicada cirurgia.

Agradeço também as várias outras pessoas, que de uma forma ou de outra, contribuíram na minha jornada acadêmica até aqui. Não mencionarei nomes, para não esquecer de ninguém, mas todos sabem a contribuição que tiveram.

A todos, meus mais sinceros agradecimentos.

RESUMO

A indústria moderna contempla um perfil de sistemas que detectam e reagem, em tempo de execução, às mudanças no contexto físico, passando a operar de diferentes modos. Ainda que essa característica exponha potencial para agregar melhorias aos processos produtivos, ela também impõe desafios à comunidade de Engenharia de Automação e Controle, como, por exemplo, na capacidade limitada para processar, em paralelo, múltiplas e complexas malhas de controle. Uma alternativa nasce da combinação de métodos formais de síntese, como a Teoria de Controle Supervisório (TCS), com arquiteturas que descentralizam a implementação do sistema de controle. Isso beneficia tanto a programação do controlador quanto a sua implementação. Entretanto, modelos atuais de implementação descentralizada não consideram aspectos emergentes da indústria, como a comunicação via internet de módulos descentralizados de controle, algo que apresenta potencial para a personalização da produção e produção cooperativa. Este trabalho propõe uma arquitetura que descentraliza o sistema de controle resultante da TCS e implementa a comunicação entre diferentes agentes de controle. Uma peça chave nesta proposta vem do conceito de um agente central e agente local de controle. O agente central executa uma abstração do comportamento dos agentes locais sendo responsável pela coordenação, em alto nível, da lógica do sistema de controle que resulta nas ações físicas locais. Nessa abstração, o controle global comanda ações aos agentes locais após o processamento de certos eventos reportados pelos agentes locais. Esses eventos, em geral, representam contextos de operação que abstraem cadeias de eventos observáveis localmente, mas não observáveis globalmente, deixando assim o controle global mais simples e concentrado em ações chaves do sistema, operacionalizadas apenas localmente. Em termos de resultado, foi observada uma redução significativa da máquina de estados implementada no agente central (com abstração) comparada com uma máquina de estados equivalente (sem abstração), ambas utilizando uma abordagem monolítica para a síntese do supervisor, que resultou em 207 estados e 464 transições de uma máquina com 251 estados e 526 transições, respectivamente. Além disso, demais aspectos de implementação indicam alto potencial de escalabilidade, que implica em possíveis ganhos a TCS, com possibilidade de expansão para diversas pesquisas e uso em ambientes produtivos.

Palavras-chave: Indústria 4.0; Teoria de controle supervisório; API REST; Cloud; Microsserviços.

ABSTRACT

Modern industry contemplates a profile of systems that detect and react, at runtime, to changes in the physical context, starting to operate in different ways. Although this characteristic exposes the potential to add improvements to production processes, it also poses challenges to the Automation and Control Engineering community, such as, for example, the limited capacity to process, in parallel, multiple and complex control loops. An alternative arises from the combination of formal synthesis methods, such as the Supervisory Control Theory (TCS), with architectures that decentralize the implementation of the control system. This benefits both controller programming and implementation. However, current models of decentralized implementation do not consider emerging aspects of the industry, such as communication via the Internet of decentralized control modules, something that has potential for customization of production and cooperative production. This work proposes an architecture that decentralizes the control system resulting from TCS and implements the communication between different control agents. A key piece in this proposal comes from the concept of a central agent and a local control agent. The central agent performs an abstraction of the behavior of local agents and is responsible for coordinating, at a high level, the logic of the control system that results in local physical actions. In this abstraction, global control commands actions to local agents after processing certain events reported by local agents. These events, in general, represent operational contexts that abstract chains of events that are locally observable, but not globally observable, thus making global control simpler and more focused on key system actions, operationalized only locally. In terms of results, a significant reduction of the state machine implemented in the central agent (with abstraction) compared to an equivalent state machine (without abstraction) was observed, both using a monolithic approach to the supervisor synthesis, which resulted in 207 states and 464 transitions of a machine with 251 states and 526 transitions, respectively. In addition, other implementation aspects indicate a high potential for scalability, which implies possible gains to TCS, with the possibility of expansion for several researches and use in productive environments.

Keywords: Industry 4.0; Supervisory control theory; API REST; Cloud; Microservices.

LISTA DE ILUSTRAÇÕES

SUMÁRIO

1 INTRODUÇÃO

A indústria moderna vem migrando para arquiteturas de produção sensíveis ao contexto, ou seja, sistemas personalizáveis de produção que detectam e reagem, em tempo de execução, às mudanças no contexto do sistema físico, passando assim a operar em múltiplos modos. Esse perfil de produção se alinha aos *Sistemas Flexíveis de Manufatura* (SFM) e quando associados a tecnologias computacionais modernas, resultam em métodos avançados para a indústria do futuro, denominada *Industria 4.0* (BANGEMANN *et al.*, 2016; LIU *et al.*, 2017; WEN; GUO, 2016).

Ainda que a indústria do futuro exponha potencial para agregar melhorias aos processos produtivos, ela também impõe desafios à comunidade de Engenharia de Automação e Controle, que pode ser observada na dificuldade de programação lógica de controladores flexíveis (SILVA; RIBEIRO; TEIXEIRA, 2017), aumento da capacidade de memória necessária para embarcá-lo, maior número de canais de comunicação, entre outros.

Esses sistemas de manufatura industrial são geralmente vistos como sistemas que evoluem a eventos, denominados como *Sistemas a Eventos Discretos* (SEDs) (CASSANDRAS; LAFORTUNE, 2009) e um dos métodos que torna possível a obtenção do controlador lógico para esses sistemas é a Teoria do Controle Supervisório (TCS), proposta por (RAMADGE; WONHAM, W. M., 1989). A TCS facilita algumas etapas na obtenção do controlador, como a programação, pois parte de um modelo e sintetiza o controlador automaticamente, além de possuir propriedades que garantem uma produção segura.

Recentemente, a literatura tem explorado aspectos relacionados à implementação eficiente de controladores (SCHOUTEN *et al.*, s.d.; ROSA *et al.*, 2017) por meio de arquiteturas descentralizadas e distribuídas, que se comunicam umas com as outras. Porém, estas iniciativas focam essencialmente na economia de memória e nas limitações do sistema distribuído, como eventuais *delays* de comunicação que interferem na estrutura de atuação do SED.

Outra abordagem que complementa decisivamente a TCS é a modularização da síntese de supervisores (PRENZEL; PROVOST, 2018; QUEIROZ; CURY, 2000), cujo foco central é o problema de memória e quantidade de portas lógicas necessárias à implementação. Nesse caso, a preocupação está em descentralizar um mesmo problema de controle em várias partes, que não necessariamente são implementadas via sistemas distribuídos. Isso difere deste trabalho, cujo foco é justamente descentralizar e distribuir um ou vários sistemas de controle particulares, bem definidos localmente (via modularização ou não), mas que em conjunto cumprem tarefas de

comum acordo como, por exemplo, em prol de objetivos comuns.

Diferentemente das abordagens da literatura (SCHOUTEN *et al.*, s.d.; ROSA *et al.*, 2017; PRENZEL; PROVOST, 2018), o principal desafio neste trabalho é explorar os módulos locais a partir de uma estrutura de comunicação que os permita requisitar e reportar quais comportamentos devem ter ou possuem em determinados instantes, que poderiam então ser coordenados mais eficientemente, ou até mesmo colaborativamente, por um agente centralizado com visão parcial sobre os módulos.

A construção desta infraestrutura descentralizada e comunicante via Internet é, então, a premissa principal deste trabalho. O desafio é explorar a descentralização dos controladores e manter as propriedades de segurança impostas por meio da TCS, integrados a um agente centralizador que tem a função de coordenar esses agentes remotos. Estima-se, assim, poder prover uma infraestrutura com suporte a aspectos voltados à indústria do futuro, envolvendo principalmente personalização de produção. A personalização, em si, não é objetivo deste trabalho, mas está sendo explorada em paralelo, em trabalhos complementares, que serão oportunamente integrados aos resultados aqui propostos.

1.1 OBJETIVO GERAL

Este Trabalho de Conclusão de Curso (TCC) tem como objetivo propor uma arquitetura para implementação de controle supervisório descentralizado, de tal modo que diferentes módulos do sistema de controle sejam implementados localmente e coordenados por um agente central que observa partes chaves de seus comportamentos e toma decisões baseadas no contexto reportado por cada um deles. Uma arquitetura de comunicação é proposta para viabilizar a ação sistemática do controlador central sobre os locais, por meio da troca de mensagens.

1.2 OBJETIVOS ESPECÍFICOS

- Construir um caso prático correspondente a um processo flexível de manufatura, modelá-lo e efetuar a síntese do controlador conforme a arquitetura proposta;
- Implementação de um sistema de comunicação eficiente para que viabilize a arquitetura proposta;
- Implementação de um servidor que contemple o controlador central para a exposição de interfaces que recebem requisições de controle.

- Implementação de clientes que possam executar comandos para diferentes tipos de máquinas industriais, e em paralelo, que possa se comunicar com o servidor por meio do protocolo de comunicação estabelecido.
- Com a arquitetura implementada, realizar testes para a comprovação de sua eficiência, como a comparação entre o tamanho de duas máquinas de estados, obtidas com e sem a abstração de eventos, provenientes do mesmo método para o cálculo do supervisor via TCS.
- Efetuar a implantação do agente central desta arquitetura em um ambiente remoto de produção.

2 REFERENCIAL TEÓRICO

Atualmente, avanços tecnológicos têm tomado dimensões tais que dificultam a concepção de sistemas informatizados, especialmente aqueles voltados à indústria. O principal empecilho é a capacidade humana limitada para processar, em paralelo, múltiplas e complexas instâncias de um determinado problema. Essa característica abre, assim, espaço para o desenvolvimento de abordagens formais para a síntese automática de sistemas.

Este capítulo apresenta a fundamentação teórica de uma classe de sistemas que é capaz de tirar proveito de mecanismos formais de modelagem para o processamento e a obtenção automática da estrutura lógica de um sistema de controle, que possui a capacidade se ser modularizada para o uso sobre uma abordagem moderna de desenvolvimento. Tal fundamentação fornece um aparato robusto para o tratamento de sistemas industriais complexos, sua representação, processamento e eventual otimização. Para explicações mais detalhadas, o leitor deve se dirigir ao (CASSANDRAS; LAFORTUNE, 2009).

2.1 SISTEMAS A EVENTOS DISCRETOS

Os SEDs possuem dois fundamentos básicos, os estados, que é um conjunto discreto identificando o status do sistema, e os procedimentos de transição de estados, ditando a forma como seu comportamento atual evolui. Esses sistemas se diferenciam daqueles dirigidos pelo tempo, pois compartilham de uma dinâmica em que os eventos acontecem de forma abrupta, imprevisível e, em pontos discretos, síncronos ou não no tempo.

2.2 TEORIA DOS AUTÔMATOS E LINGUAGENS

No modelo de um SED, um *alfabeto* é definido como um conjunto composto por todos os eventos possíveis, no qual o conjunto é finito, não-vazio e denotado por Σ. O símbolo Σ^* denota todas as possíveis combinações finitas de cadeias dos eventos de Σ, incluindo-se a cadeia vazia denotada por ϵ (CASSANDRAS; LAFORTUNE, 2009).

Uma linguagem L pode ser definida formalmente como um subconjunto de cadeias em Σ^*, ou seja, $L \subseteq \Sigma^*$, que determina quais possíveis combinações de eventos do Σ são sequências reconhecidas. Dado uma linguagem $L \subseteq \Sigma^*$, seu prefixo-fechamento é denotado por $\bar{L}$, define-se como: $\bar{L} = \{s \in \Sigma^* : (\exists t \in \Sigma^*), st \in L\}$. A linguagem pode ser expressa por um *Autômato*

Finito (AF) (SIPSER, 2012), também conhecido como máquina de estados finitos, que é definido como uma 5-tupla $\langle Q,\Sigma,\rightarrow,q_0,F\rangle$, onde Q é o conjunto finito de estados, Σ é o alfabeto finito, $\rightarrow\subseteq Q\times\Sigma\times Q$ é a relação de transição de estados, $q_0\in Q$ é o estado inicial, $F\subseteq Q$ é o subconjunto de estados marcados.

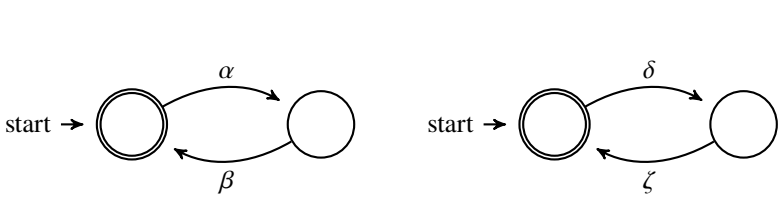
Dois tipos de linguagens podem ser definidas a partir de um autômato qualquer G, a linguagem gerada e a linguagem marcada, definidas respectivamente como: $L(G)=\{s\in\Sigma^*: q_0\xrightarrow{s}q\in Q\}$; $L^w(G)=\{s\in\Sigma^*: q_0\xrightarrow{s}q\in F\}$. Assim a linguagem gerada $L(G)$ abrange todas as possíveis cadeias do autômato G, ao passo que a linguagem marcada $L^w(G)$ representa todo o conjunto de cadeias que atingem os estados marcados, tal que $L^w(G)\subseteq L(G)$ (CASSANDRAS; LAFORTUNE, 2009).

Uma operação útil que pode ser aplicada em linguagens e autômatos é denotado por $||$, e dá-se o nome de composição síncrona (WONHAM, W., 2002). Neste trabalho, por questões de praticidade, define-se somente a operação de sincronização para autômatos, pois este é adotado como formalismo para simplificar a modelagem de SEDs.

Portanto, sejam dois autômatos, $G_1=\langle Q_1,\Sigma_1,\rightarrow_1,q_1^0,F_1\rangle$ e $G_2=\langle Q_2,\Sigma_2,\rightarrow_2,q_2^0,F_2\rangle$, representado na figura 1. O resultado da composição pode ser observado na figura 2. A composição síncrona de G_1 e G_2, denotada por $G_1||G_2$, é definida pelo seguinte autômato $G=G_1||G_2=\langle Q_1\times Q_2,\Sigma_1\cup\Sigma_2,\rightarrow_1\,||\,\rightarrow_2,(q_1^0,q_2^0),F_1\times F_2\rangle$, tal que suas transições são definidas pelas regras da equação 1.

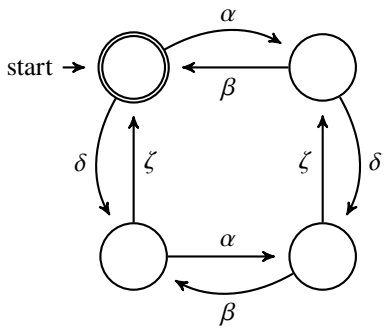
$$\rightarrow_1\,||\,\rightarrow_2=\begin{cases}(q_1,q_2)\xrightarrow{\sigma}(q_1',q_2')) & \text{se}\quad \sigma\in\Sigma_1\cap\Sigma_2;\\ (q_1,q_2)\xrightarrow{\sigma}(q_1',q_2)) & \text{se}\quad \sigma\in\Sigma_1\backslash\Sigma_2;\\ (q_1,q_2)\xrightarrow{\sigma}(q_1,q_2')) & \text{se}\quad \sigma\in\Sigma_1\backslash\Sigma_2;\\ \text{Indefinido caso contrário}\end{cases} \tag{1}$$

Figura 1 – Modelagem dos autômatos



Fonte: Autoria própria.

Figura 2 – Composição síncrona de autômatos
$G = G1 \,||\, G2$:



Fonte: Autoria própria.

2.3 TEORIA DE CONTROLE SUPERVISÓRIO

A modelagem de SEDs por meio de autômatos e linguagens habilita o processamento matemático-computacional da TCS. A partir das plantas de um sistema e suas especificações, a TCS calcula um autômato supervisor com características de controlabilidade, não bloqueio e de máxima permissividade. Isso garante que o sistema tenha um comportamento em que possa executar todas as tarefas possíveis sem comprometer aspectos de segurança com más combinações de eventos.

Na abordagem proposta pela TCS, o alfabeto é dividido em dois subconjuntos disjuntos $\Sigma = \Sigma_c \cup \Sigma_u$. O simbolo Σ_c representando os eventos controláveis e Σ_u os eventos não controláveis, esse segundo subconjunto traz os eventos que não podem ser diretamente impedidos de ocorrer. O supervisor tem a responsabilidade de coordenar as ações da planta permitindo alcançar um comportamento $L(K)$, habilitando ou desabilitando apenas eventos do subconjunto Σ_c.

Formalmente, seja E um autômato modelando as restrições impostas a uma planta G, Dizemos que $L(K) = L^w(G||E) \subseteq L^w(G)$ é controlável em relação a G se $\overline{L(K)}\Sigma_u \cap L(G) \subseteq \overline{L(K)}$. Ou seja, se após qualquer evolução da planta, um evento não controlável é elegível, então ele não é desabilitado.

A operação que calcula o supervisor controlável S é denominada de síntese e o supervisor representa o máximo comportamento controlável, ou seja, o que mais se aproxima de $L(K)$. Deste modo, S pode ser associado ao comportamento menos restritivo possível de ser implementado por um supervisor não-bloqueante sobre G, de maneira a respeitar o conjunto de especificações.

2.4 CONTROLE DESCENTRALIZADO

Uma vez que o supervisor foi sintetizado, é possível implementá-lo por meio de uma conversão da sua máquina de estados em código sobre alguma linguagem de programação. Sua implementação pode ser efetuada de vários modos, uma delas em um único *Hardware* de maneira centralizada (e.g., Controladores Lógicos Programáveis (CLPs) (LEAL; DA CRUZ; HOUNSELL, 2009). Essa com a desvantagem de ter uma lógica fixa e pouca flexibilidade. Outra forma, de modo descentralizado, em que o controlador pode ser implementado em vários dispositivos (AFZALIAN; NOORBAKHSH; NABAVI, 2008).

A ideia de implementação descentralizada está muito alinhada a cooperação de sistemas e aos moldes da indústria moderna (*indústria 4.0*), logo quando se opta por descentralizar uma arquitetura de controle, os dispositivos tornam-se mais efetivos, configurados localmente, e possibilitando maior flexibilidade, o que resulta em um sistema distribuído (HARRISON; VERA; AHMAD, 2016).

Aqui, assume-se que partes do sistema de controle sejam implementados localmente, de tal maneira que esses módulos locais reportam informações a uma estrutura global, que é responsável pela coordenação das entidades locais. Portanto, ao se optar por não implementar a máquina de estados em um mesmo *Hardware*, surge a necessidade de que esses subsistemas estejam sobre uma estrutura de rede para que possam se comunicar.

2.4.1 COMUNICAÇÃO CLIENTE-SERVIDOR

Em uma arquitetura cliente-servidor, a comunicação ocorre entre um programa hospedeiro, denominado servidor, com vários outros hospedeiros, denominados clientes. A troca de mensagens se dá em um esquema de pergunta e resposta, nos quais os clientes enviam as perguntas, ou requisições, para que o servidor possa receber, processá-las e devolver as respostas (KUROSE; ROSS, 2013). Esta arquitetura de comunicação domina o cenário moderno da *web*, com servidores fornecendo diversos conteúdos aos navegadores utilizados por bilhões de usuários.

O primeiro passo para entender uma aplicação distribuída na Internet, é saber que, quando ocorre uma comunicação entre diferentes máquinas, toda uma estrutura de rede deve existir, além de suas regras, ou protocolos, que ditam a forma de como cada pacote vai se comportar, desde o processo de origem até o destino.

O conjunto de protocolos que determinam a transmissão de pacotes por meio de uma

rede, é conhecido como a pilha TCP/IP (*Transmission Control Protocol/Internet Protocol*). Essa pilha é encarregada de realizar diversas funções, com protocolos separados em camadas que definem serviços estruturados para a comunicação entre aplicações.

A partir disso, cada hospedeiro deve possuir um conjunto de bits que o identifica de maneira exclusiva, denominado endereço IP, e a porta a qual a aplicação atua, pois a mesma máquina pode executar diferentes aplicações na rede. Adicionalmente, a comunicação é possível por meio de uma interface de software denominada *socket*, que é a interface entre a camada de aplicação e a de transporte dentro de um sistema final. A aplicação pode optar por qual protocolo de transporte o *socket* irá atuar, neste projeto é utilizado o TCP (*Transfer Control Protocol*), pois garante que o servidor e o cliente troquem informações de controle antes que os dados sejam transmitidos, bem como a confiabilidade de sua entrega (KUROSE; ROSS, 2013).

De acordo com o contexto desse trabalho, o protocolo da camada de aplicação que faz parte da pilha TCP/IP e melhor se encaixa aos requisitos necessários para uma aplicação de controle descentralizada, e com um agente centralizador, é o protocolo HTTP.

2.4.2 PROTOCOLO HTTP

O protocolo HTTP pertence a camada de aplicação e é a base de qualquer troca de dados na Web. Ele possui a arquitetura de um cliente-servidor e define a estrutura e o modo de como essas entidades trocam mensagens. O início da comunicação ocorre pelo lado do cliente, em que solicita ao servidor um documento, por exemplo, uma página HTML (*Linguagem de Marcação de HiperTexto*), uma imagem JPEG (*Joint Photographics Experts Group*), entre outros. Esses objetos são formatos que representam a estrutura de dados que é transmitida pelos programas (KUROSE; ROSS, 2013).

Um formato que tem sido utilizado em larga escala por inúmeras aplicações é o JSON (*JavaScript Object Notation*). Este padrão é facilmente entendido por humanos, leve para as máquinas e independente de qualquer linguagem de programação, isto é, pode ser escrito e reconhecido por qualquer linguagem. Sua sintaxe estrutura os dados em pares propriedade/valor, como o exemplo abaixo:

$$\{\text{“}Nome\text{”} : \text{“}Geovany\text{”}\}$$

O HTTP utiliza o TCP como protocolo da camada de transporte, de modo que o cliente HTTP inicia uma conexão TCP com o servidor, e após estabelecida, os processos acessam ao

TCP fazendo uso da interface de *socket*. Dessa maneira, tanto o cliente quanto o servidor enviam e recebem, respectivamente, as requisições por meio desta interface. Uma vantagem da utilização deste protocolo é que o TCP garante a chegada dos dados a ambos os lados, o que implica que toda requisição e resposta chegará intacta (KUROSE; ROSS, 2013).

Para que uma requisição seja efetuada pelo cliente, é necessário saber a URL (*Uniform Resource Locator*) do servidor, ou o endereço IP ao qual se encontra. Além dos métodos que a requisição poderá efetuar, que segundo (TANENBAUM, 2003), podem ser, principalmente, do tipo GET, POST, PUT ou DELETE. Assim, é possível a implementação de uma API (*Application Programming Interface*) REST (*Representational State Transfer*), que define uma estrutura para a criação de APIs (REST. . . , 2020).

Uma vez que o sistema de controle se torna modular, uma forma de implementar várias partes deste sistema de maneira eficiente, rápida e altamente escalável é fazendo uso de uma arquitetura orientada a microsserviços, que é explorada na sequência.

2.5 MICROSSERVIÇOS

Uma arquitetura orientada a microsserviços implementa cada componente de maneira independente, mas trocam informações para alcançar o objetivo geral do sistema. Tais componentes são, preferencialmente, implementados em máquinas diferentes, e um dos métodos que permitem uma rápida implementação é o *Docker* e *Docker Compose*, que realizam a conteinerização de aplicações.

2.5.1 DOCKER E DOCKER COMPOSE

O *docker* é uma plataforma para a virtualização de uma aplicação, que ocorre a nível de software e permite o isolamento entre aplicações. Isto ocorre fazendo uso de uma imagem que contempla a base para a construção do projeto, como, por exemplo, uma imagem do *Python* para a execução de uma API utilizando o *framework FastAPI*. Esta imagem será utilizada pelos mecanismos do *docker* para a execução de um contêiner na máquina com apenas um único comando (DOCKER. . . , 2022).

Para a configuração e execução de vários serviços de maneira centralizada, o *docker* disponibiliza um pacote adicional denominado *docker compose*. Este componente permite a execução de vários contêineres com a utilização de apenas um único arquivo de configuração

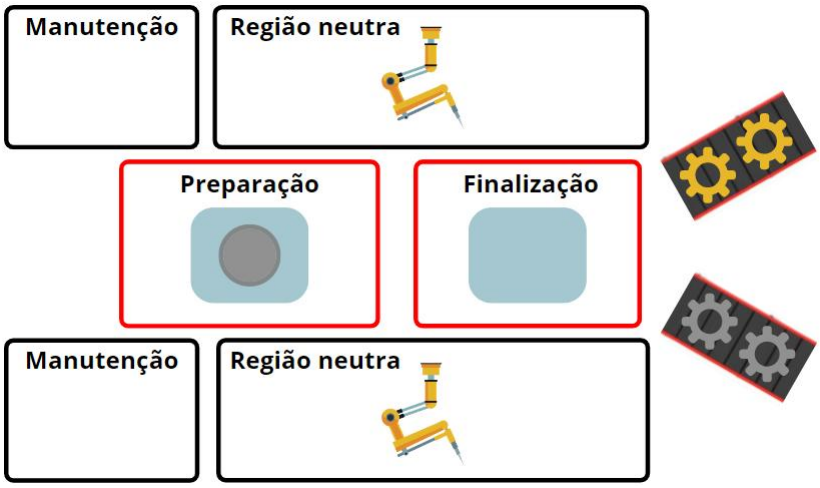
e um comando, que poderiam ser, por exemplo, um serviço de banco de dados, uma API em *Python*, entre outros (OVERVIEW. . . , 2022).

Na seção 3 é apresentada todas as etapas relacionadas a construção desta arquitetura de controle descentralizado, que possui um agente global coordenando vários dispositivos locais, isto é, uma descentralização baseada em um agente centralizador, que pode ser utilizado de maneira interna a uma fábrica (e.g., linha de produção X), nela como um todo (e.g., vários setores ou processos), ou além, para várias fábricas. Isso só é possível devido a estrutura moderna e descentralizada desta arquitetura, que se comunica via Internet e se posiciona além do que foi explorado atualmente na literatura.

3 ARQUITETURA PARA CONTROLE DESCENTRALIZADO

Esta seção apresenta a arquitetura comunicante proposta para o controle supervisório de SEDs. Para a melhor ilustração desta arquitetura, parte-se de um pequeno exemplo prático de dois braços robóticos que disputam por recursos para a produção de diferentes tipos de peças, podendo haver cooperação entre os dois componentes, a depender de decisões tomadas por um agente centralizado de controle, capturando aspectos de produção cooperativa. A arquitetura do exemplo pode ser vista na figura 3.

Figura 3 – Cenário para a implementação da arquitetura proposta



Fonte: Autoria própria.

A área de preparação é uma espécie de *buffer* que contém um único insumo, de modo que ambos os braços robóticos irão se direcionar até esta região e efetuar uma série de ações, como um furo ou corte na peça, que definem se a peça será de um modelo A ou B, respectivamente. Já a região de finalização é o local onde a peça será colocada pelos braços robóticos, derivados da preparação, para novas ações que preparam esta peça para ser enviada a duas distintas esteiras.

As regiões demarcadas em vermelho são compartilhadas entre ambos os braços robóticos, isto quer dizer que a robustez do sistema de controle deve atuar para evitar possíveis colisões. Já as regiões demarcadas em preto são áreas individuais a cada braço, não havendo problemas de colisão. Finalmente as duas esteiras são apenas ilustrativas, pois quando a peça é finalizada, os mecanismos da própria região de finalização são responsáveis por direcionar a peça tratada até uma das esteiras, cujo controle não faz parte do escopo deste trabalho.

A partir desse cenário, define-se a modelagem que abstrai as principais sequências de cada braço robótico para a formação de uma planta global do sistema, que junto de suas especificações, é calculado um supervisor compacto que será implementado em um agente

centralizador. Então, os módulos locais irão implementar um autômato diferente dos quais foram utilizados para suas representações, mas ainda assim será similar, pois adiciona os eventos que não foram modelados. No total, três módulos de controle serão implementados sobre à arquitetura de comunicação proposta, o agente central e dois módulos locais, que definem a modularização do sistema de controle advindo da TCS. Por fim, mostram-se indícios de possíveis ganhos com a abordagem, por meio do mesmo exemplo de controle.

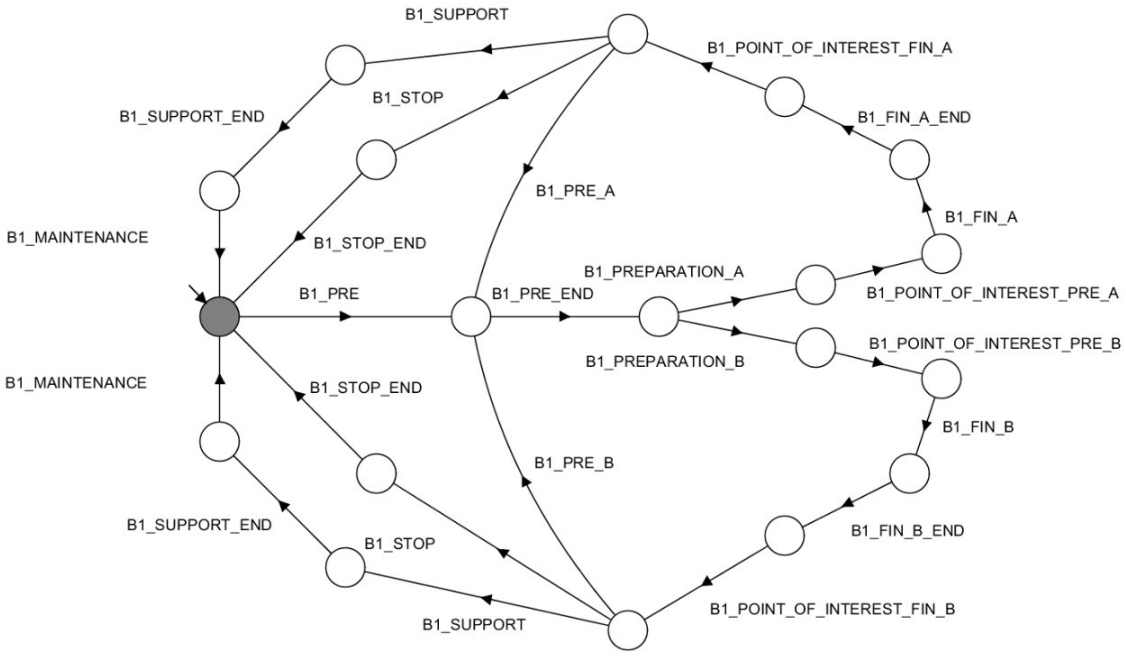
3.1 SISTEMA DE CONTROLE DO AGENTE CENTRAL

Esta seção expõe as etapas referentes a obtenção do sistema de controle que será implementado pelo agente central, partindo da modelagem até o cálculo do supervisor.

A modelagem foi realizada utilizando a ferramenta de software supremica (AKESSON *et al.*, 2019), que disponibiliza um ambiente gráfico de modelagem, simulação, síntese de um supervisor via TCS e verificação de SEDs. Assim é assegurado a obtenção de um supervisor com as propriedades advindas da TCS e que pode ser representado por uma máquina de estados.

A máquina de estados que representa uma abstração das ações disponíveis do braço robótico 1 para suas atividades de manufatura denomina-se G_{abs1} e é apresentada na figura 4, ao passo que o autômato G_{abs2} representa as respectivas ações do braço robótico 2 e é equivalente ao primeiro, apenas com alterações nos rótulos dos eventos.

Figura 4 – Autômato do braço robótico 1



Fonte: Autoria própria.

A tabela 1 mapeia os eventos do autômato ilustrado na figura 4, bem como suas respectivas descrições semânticas. Para a coluna de controlabilidade, a letra *C* representa que o evento é controlável, ao passo que *NC* condiz com um evento não controlável, e se a coluna 'abstração' possuir um *Sim*, isso significa que o evento em questão possui um perfil de abstrair outros eventos. Isto representa que no contexto de controle lógico, tais eventos não são necessários para o cálculo do agente central, como o abrir ou fechar a garra do braço robótico, por exemplo, que deve ser implementado localmente mas no contexto geral do sistema não causa impacto.

Vale ressaltar que o braço robótico 2 executa eventos equivalentes ao braço robótico 1, ou seja, o evento B1_PRE existirá no autômato G_{abs2} como o evento B2_PRE, por exemplo. Tal evento representa a saída do braço robótico 2 da posição inicial em direção a área de preparação, os demais eventos serão definidos igualmente.

Tabela 1 – Eventos do autômato G_{abs1}

Braço robótico 1 (G_{abs1})			
Sinal	Descrição	Controlabilidade	Abstração
B1_PRE	Sai da posição inicial em direção a área de preparação	C	-
B1_PRE_END	Chega na área de preparação	NC	-
B1_PREPARATION_A	Prepara a região para produzir a peça A	C	-
B1_PREPARATION_B	Prepara a região para produzir a peça B	C	-
B1_POINT_OF_INTEREST_PRE_A	Abstrai eventos que preparam, de fato, a peça do tipo A	NC	Sim
B1_POINT_OF_INTEREST_PRE_B	Abstrai eventos que preparam, de fato, a peça do tipo B	NC	Sim
B1_FIN_A	Sai da preparação em direção a área de finalização com a peça A	C	-
B1_FIN_B	Sai da preparação em direção a área de finalização com a peça B	C	-
B1_FIN_A_END	Chegou na área de finalização com a peça A	NC	-
B1_FIN_B_END	Chegou na área de finalização com a peça B	NC	-
B1_POINT_OF_INTEREST_FIN_A	Abstrai eventos que finalizam, de fato, a peça do tipo A	NC	Sim
B1_POINT_OF_INTEREST_FIN_B	Abstrai eventos que finalizam, de fato, a peça do tipo B	NC	Sim
B1_PRE_A	Se movimenta até a área de preparação, advindo da região de finalização de outra peça do tipo A	C	-
B1_PRE_B	Se movimenta até a área de preparação, advindo da região de finalização de outra peça do tipo B	C	-
B1_STOP	Se movimenta em direção a posição inicial do braço robótico	C	-
B1_STOP_END	Chegou na posição inicial do braço robótico	NC	-
B1_SUPPORT	Se movimenta em direção a região de manutenção	C	-
B1_SUPPORT_END	Chegou na região de manutenção	NC	-
B1_MAINTENANCE	Abstrai um conjunto de eventos realizados na manutenção do equipamento	NC	Sim

Fonte: Autoria própria.

Para explicitar o funcionamento do autômato da figura 4 e seus eventos, utiliza-se o contexto dos braços robóticos dispostos fisicamente conforme o layout da figura 3. Portanto para manufatura de um produto, um braço se direciona para uma área inicial demarcada como *Preparação* no layout, esses movimentos são traduzidos pelos eventos B1_PRE que representa o sinal de início do movimento e B1_PRE_END de término, na figura 3, que determina quando o dispositivo já se encontra sobre a área de preparo.

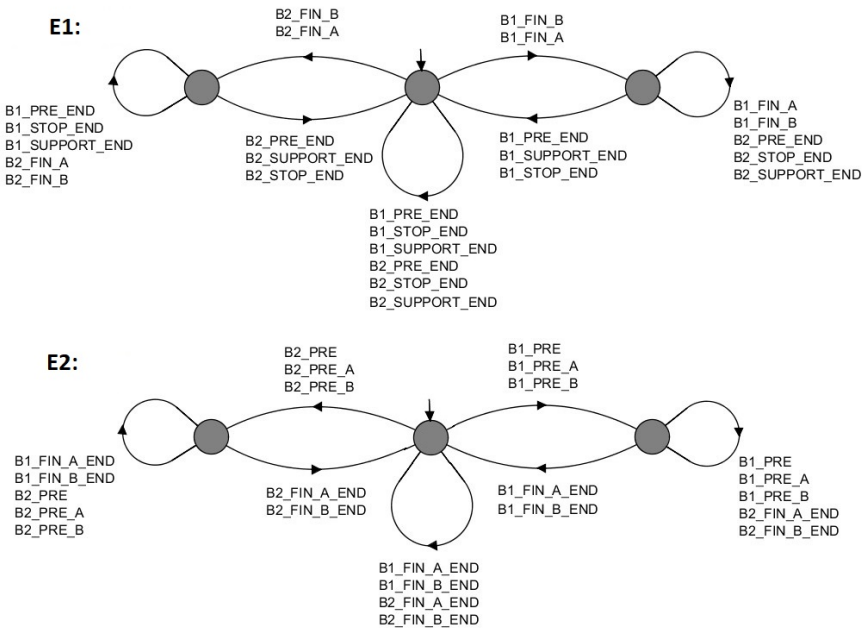
Os próximos eventos disponíveis no autômato G_{abs1} habilitam a opção de escolher o início de qual tipo de produto será preparado neste local, B1_PREPARATION_A ou B1_PREPARATION_B, já o evento subsequente B1_POINT_OF_INTEREST_PRE_A, por exemplo, é o evento que representa uma abstração de pormenores eventos de preparação do produto. Este evento representa fisicamente etapas em que o braço robótico atua na área de preparação e não requer a modelagem para o cálculo do supervisor.

Após o término da etapa de preparo o próximo evento dará sequência na manufatura do produto, e para isso, deve ser transportado para outra área. Isso quer dizer que fisicamente o evento de início B1_FIN_A, faz com que o dispositivo se mova com o produto semi-preparado para outra área (região de finalização), deste modo, o evento de término B1_FIN_A_END indica que o dispositivo chegou com sucesso na região de finalização do produto, consequentemente o evento B1_POINT_OF_INTEREST_FIN_A também representa uma versão abstraída dos eventos realizados para o acabamento e descarte final do produto.

A partir deste momento, o dispositivo tem as opções de iniciar o evento de transição para a área de produção inicial com o evento B1_PRE_A, ir para uma região neutra desocupando a área de acabamento por meio de B1_STOP ou por fim ir para a área de realização de manutenção escolhendo B1_SUPPORT. Assim o ciclo se repete, tanto para a produção de uma peça do tipo A ou B.

A composição dos autômatos G_{abs1} e G_{abs2} resulta na planta geral do sistema G_{abs}, que por si só não possui nenhuma ação de controle. Para tal, são modeladas especificações que impõem restrições de eventos exclusivamente sobre a planta G_{abs}, expostas na figura 5, que serão utilizadas para o cálculo e obtenção da máquina de estados do supervisor.

Figura 5 – Restrições impostas a planta global G_{abs}



Fonte: Autoria própria.

As especificações e suas respectivas finalidades sobre a planta são assim descritas:

- E1: esta especificação visa implementar um *mutex* na região de finalização exposta na figura 3. Isto significa que se um dos dois braços robóticos adentrar nesta região, o outro ficará proibido de entrar até que o primeiro saia desta região.
- E2: já esta especificação, possui a responsabilidade de efetuar um *mutex* na região de preparação exposta na figura 3. Isto representa que se um dos dois braços robóticos adentrar nesta região, o outro ficará proibido de entrar até a finalização dos trabalhos do braço nesta área.

Em posse da planta global do sistema G_{abs} e suas respectivas especificações (E1 e E2), o supervisor é sintetizado por meio do *software* supremica e resulta numa máquina de estados contendo 207 estados, com 434 eventos de transição, que representa o sistema de controle obtido via TCS ao qual faz uso do conceito de abstração de eventos permitido por esta arquitetura. O supervisor denomina-se S_{abs} e é implementado no agente central.

A abordagem para a síntese do supervisor utilizada via TCS foi a monolítica, no entanto mesmo que seja a maneira mais básica de se obter um sistema de controle via TCS, gerou-se ganhos com um supervisor mais compacto. Uma comparação com um supervisor sem abstração será efetuado na sequência, a fim de validar este argumento. Outra característica a ser destacada é que os autômatos até aqui utilizados foram construídos apenas para o cálculo de S_{abs}, diferentemente dos quais serão implementados nos módulos locais.

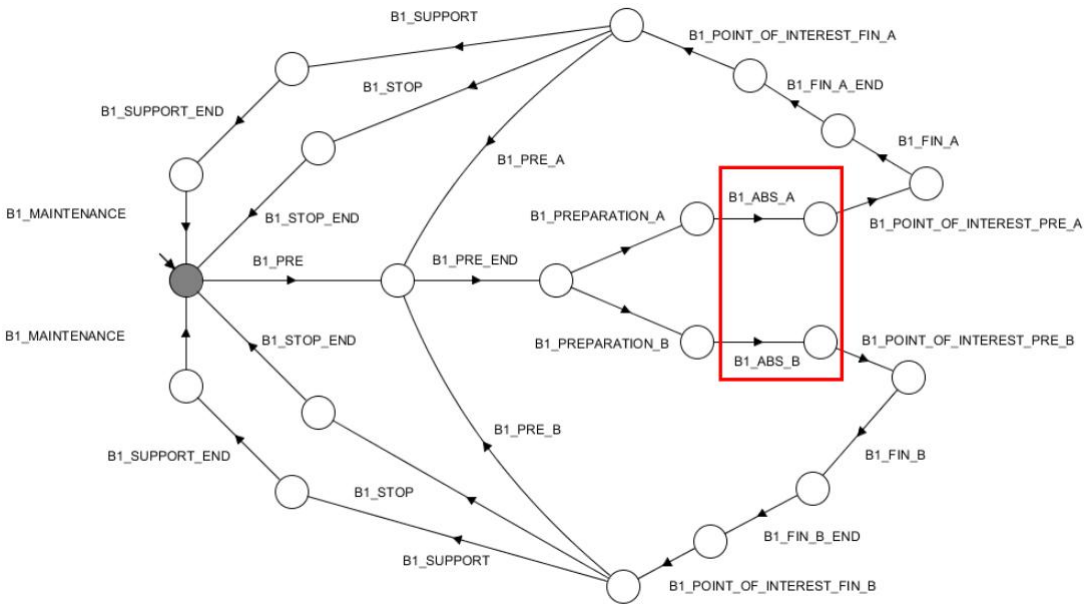
A seguir é apresentado quais as máquinas de estados serão implementadas para o controle de cada módulo local, a forma com que os eventos abstraídos favorecem no cálculo de S_{abs} e, finalmente, como é a relação entre os sistemas de controle central e local no que diz respeito aos eventos mapeados apenas localmente.

3.2 SISTEMA DE CONTROLE DOS MÓDULOS LOCAIS

Cada módulo local irá implementar uma máquina de estados que representa todas as suas possíveis ações físicas, muito parecida com o autômato apresentado na figura 4, no entanto, apenas inclui os eventos abstraídos, pois devem existir em algum contexto de todo o sistema. Para expor a diferença entre a representação do braço robótico 1, que houve por meio do autômato G_{abs1} e o sistema de controle que será implementado localmente, a figura 6 expõe o autômato $G_{B1NotAbs}$, que não possui abstração e demonstra dois eventos adicionais utilizados para representar o comportamento local do braço robótico 1.

Este autômato será implementado no módulo de controle do braço robótico 1, e o mesmo ocorre para o braço robótico 2, que também insere dois eventos para representar seu total funcionamento por meio do autômato $G_{B2NotAbs}$. O evento B1_ABS_A representa o furo do insumo para a produção da peça do tipo A, e o B1_ABS_B representa um corte no insumo para se produzir uma peça do tipo B. No segundo braço robótico o autômato é similar, apenas o nome dos eventos que seriam diferentes (B2_ABS_A e B2_ABS_B).

Figura 6 – Autômato com todas as possíveis ações do braço robótico 1



Fonte: Autoria própria.

Tabela 2 – Comparação entre supervisores que fazem ou não o uso da abstração de eventos

Supervisor	Estados	Transições
S_{NotAbs}	251	526
S_{abs}	207	434

Fonte: Autoria própria.

Os autômatos $G_{B1NotAbs}$ e $G_{B2NotAbs}$ em conjunto com o supervisor S_{abs} formam um sistema de controle modular proposto por esta arquitetura. A partir deste ponto, o autômato $G_{B1NotAbs}$ será denominado como L_1 e o $G_{B2NotAbs}$ com L_2, que remetem aos módulos locais para o controle das ações físicas do braço robótico 1 e 2, respectivamente.

Se os autômatos L_1 e L_2 fossem compostos para formar uma planta global do sistema sem abstração e sintetizados com as especificações E1 e E2, obtém-se um sistema de controle também monolítico S_{NotAbs}, equivalente para o controle do sistema apresentado na figura 3, entretanto não faz uso do conceito de abstração, cujo tamanho e comparação com o sistema de controle do agente central proposto por esta arquitetura é apresentado na tabela 2.

A tabela 2 expõe os ganhos em termos de tamanho do supervisor abstraído S_{abs} frente ao que não utiliza abstração S_{NotAbs}. Estes ganhos são consideráveis, visto que apenas 4 eventos foram abstraídos em todo o sistema.

3.3 AGENTE CENTRAL DE CONTROLE BASEADO NA ABSTRAÇÃO DE EVENTOS

O agente central nada mais é do que uma abstração das plantas locais, que observa apenas partes dos controladores locais L_1 e L_2, partes estas que competem ao controle central de ações S_{abs}. Essa abstração é conveniente principalmente por simplificar a modelagem do controlador central e, ao mesmo tempo, permitir a autonomia dos controladores locais.

Na arquitetura proposta, a planta global é vista como 2 plantas, G_{abs1} que abstrai L_1 e G_{abs2} que abstrai L_2. Elas são construídas conforma a figura 4 que representa modelos que abstraem os eventos que representam as ações dos dispositivos, de modo que um único evento possa representar um conjunto de outros eventos. Isso é possível graças a descentralização do sistema de controle em módulos locais e global, em que apenas eventos que iniciam e completam sequências locais de produção precisam ser observados pelo agente central, ao passo que no contexto local de cada equipamento, todos os eventos serão implementados, que se inclui os que foram abstraídos. Uma metodologia para ditar quais eventos devem ser abstraídos ou mantidos na modelagem dos autômatos não é explorada por esta pesquisa, mas é citada como mais uma

possibilidade para trabalhos futuros complementares a este.

A abstração da modelagem poderia ocorrer por meio da utilização dos seguintes eventos, mas para exemplificar a utilidade desta arquitetura, a figura 6 não abstraiu apenas B1_POINT_OF_INTEREST_PRE_A e B1_POINT_OF_INTEREST_PRE_B, que são eventos para representar um conjunto de outros eventos que poderiam existir apenas localmente, sendo eles:

- B1_POINT_OF_INTEREST_PRE_A.
- B1_POINT_OF_INTEREST_PRE_B.
- B1_POINT_OF_INTEREST_FIN_A.
- B1_POINT_OF_INTEREST_FIN_B.
- B1_MAINTENANCE.

Ao fim de alguma sequência de eventos que existe apenas no contexto local, seja em L_1 ou L_2, um dos eventos da lista acima será mapeado e enviado ao agente central S_{abs}, que vai reconhecer o fim de uma série de eventos abstraídos e atualizará seus status.

A seguir, os conceitos referentes à implementação são explorados com mais detalhes, a fim de facilitar o entendimento de como esses componentes remotos trabalham em conjunto.

3.4 IMPLEMENTAÇÃO DISTRIBUÍDA DO SISTEMA DE CONTROLE

A implementação foi distribuída de modo que um servidor implementa a máquina de estados S_{abs} calculada anteriormente, mas como alguns eventos foram abstraídos, devem existir em algum contexto, que serão nos módulos locais. Cada módulo local terá um sistema de controle apresentado a seguir, que são: L_1 e L_2. Esta característica é o principal ponto desta arquitetura, pois calcula um supervisor por meio da mesma abordagem monolítica da TCS, mas a modelagem para o cálculo é efetuada sobre modelos com menos estados e transições, resultando em um servidor mais compacto, que em conjunto com os módulos locais, definem a modularização de todo o sistema de controle.

3.4.1 AGENTE GLOBAL (SERVIDOR)

O agente centralizador ou global é implementado em uma máquina que executa o servidor HTTP, que é responsável pela coordenação lógica de todo o sistema de controle, isto é, está no servidor a decisão de liberar ou não eventos para cada módulo do cliente, ou seja, os

módulos locais.

Os eventos estão dispostos em uma lista, que foi retirada como uma sequência válida de eventos por meio da máquina de estados S_{abs} calculada via TCS, que provê a produção de uma ou mais peças para o cenário da figura 3. Esta mesma máquina de estados poderia ser percorrida por uma metaheurística para coordenar a cooperação entre agentes remotos, pois é o sistema de controle responsável por ditar a forma com que os módulos locais devem atuar.

3.4.2 AGENTES LOCAIS (CLIENTES)

Os módulos locais implementam o conjunto de todas as ações físicas de cada componente industrial, no caso do braço robótico 1, implementa o autômato L_1, que nada mais é que uma ampliação do autômato G_{abs1}. Este comportamento ocorre de maneira similar ao braço robótico 2, cujo sistema de controle local é o L_2, que é a expansão do autômato G_{abs2}.

Cada módulo de controle local é responsável por requisitar ao servidor alguma ação para ser efetuada, bem como reportar ações finalizadas. Isto significa que cada agente local possui um comportamento passivo de atuação, pois só devem dar início a eventos mapeados no servidor a partir do momento que o agente global permitir.

Em resumo, as características deste agente são positivas, porque removem a sobrecarga do global (que atua executando uma abstração) e, ao mesmo tempo, remove sobrecarga também dos locais, que não precisam enxergar comportamentos que não lhes compete localmente.

3.4.3 FLUXO DE COMUNICAÇÃO ENTRE CLIENTE E SERVIDOR

A comunicação fornece um caminho para que os módulos locais possam ser controlados pelo agente global. A figura 7 ilustra um fluxograma que expõe em etapas a forma com que as requisições foram implementadas.

Cada módulo local requisita um único evento do servidor (Etapa 1), que disponibiliza o primeiro da lista de eventos. Se o evento recebido pertence a máquina de estados do agente local e condiz com o qual este agente aguarda, o módulo local enviará uma nova requisição informando ao servidor que o evento foi recebido pelo módulo correto (Etapa 2), para então iniciar o processamento. Por fim, quando o módulo local finaliza as etapas referente ao evento, uma nova confirmação será enviada ao servidor para informá-lo sobre algum evento de término (Etapa 3).

Figura 7 – Fluxo de comunicação

Fonte: Autoria própria.

Deste modo, as requisições de confirmação e término de eventos atualizam a lista do servidor, que os exclui da lista. Os módulos locais ficam em processo de *loop* até que os eventos enviados sejam iguais aos que se espera receber. Isso garante que se um evento diferente for lido, apenas será ignorado.

Até este ponto a discussão foi pautada, principalmente, em questões conceituais relacionadas à implementação distribuída do sistema de controle. A seguir, será apresentado as ferramentas e tecnologias utilizadas para ilustrar os aspectos técnicos deste trabalho.

3.5 FERRAMENTAS E TECNOLOGIAS DA ARQUITETURA PROPOSTA

Para que a arquitetura proposta seja construída de maneira robusta e eficiente, o conjunto correto de práticas, ferramentas e tecnologias devem ser as mais adequadas ao que se propõe. Assim, esses aspectos foram levados em consideração desde a escolha das linguagens de programação, até o preparo para a execução dessa aplicação em diferentes ambientes, dentre eles, de desenvolvimento ou produtivo.

Cada módulo local foi implementado fazendo uso dos microcontroladores ESP32 para a efetivação do controle das ações físicas de cada equipamento, no caso deste trabalho, da simulação de dois braços robóticos. Este micro foi utilizado devido a capacidade de comunicação *wi-fi*, bem como a programação por meio do *framework* oficial de desenvolvimento, chamado ESP-IDF, que disponibiliza uma série de recursos para a compilação, gravação e monitoramento de logs em tempo de execução, além de ser *Open Sorce* (OFFICIAL. . . , 2022).

O agente central é o componente da arquitetura que irá prover toda a lógica do sistema de controle, e toda a sua estruturação deve ser leve e escalável. Para tal, o servidor foi construído

fazendo uso da linguagem de programação *Python*, que é uma linguagem interpretada, orientada a objetos, *Open Source* e possui uma imensa quantidade de usuários, que fornecem amplas soluções para diversos tipos de aplicações. Dentre elas, está o *framework FastApi*, que foi utilizado para a construção da API implementada pelo agente global, pois é rápido, também *Open Source* e de fácil desenvolvimento (FASTAPI. . . , 2022).

Outra característica de grande importância que foi adicionada na construção do servidor é a utilização de um ambiente virtual do próprio *Python* (VENV. . . , 2022). Este ambiente provê o isolamento de todos os recursos utilizados pela aplicação, que em conjunto com as ferramentas de conteinerização como o *Docker* e *Docker Compose*, encapsularam a aplicação para ser inicializada com um único comando, seja num ambiente de implantação de *software* local, de homologação ou produtivo.

Isto ocorre devido ao fato deste trabalho ter o intuito de coordenar agentes remotos, que concretiza uma arquitetura para ser expandida até a coordenação de várias fábricas. Assim, a implantação do controle global levou em consideração uma perspectiva moderna de desenvolvimento, que se situa em uma abordagem orientada a microsserviços para a implantação em ambientes de alta disponibilidade, como em *Cloud*.

3.5.1 SERVIDOR IMPLANTADO EM AMBIENTE DE CLOUD

O desenvolvimento de uma aplicação ocorre em etapas e fazendo uso de diversos ambientes para a efetivação de testes. De início, geralmente o desenvolvimento é efetuado em ambiente local, isso quer dizer que a própria máquina do desenvolvedor é responsável por executar todos os processos referentes à aplicação. No entanto, quando se opta por disponibilizar os recursos do projeto a um grande número de usuários a implantação pode ocorrer a nível remoto, como por exemplo, em ambiente de *Cloud*.

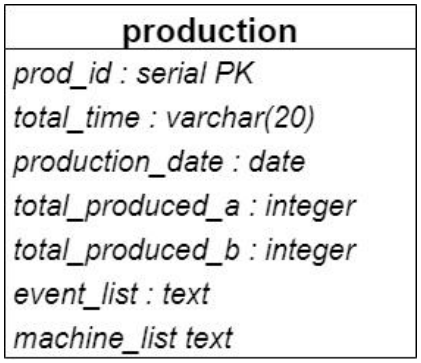
Para testar e efetivar o conceito apresentado por esta arquitetura sobre o controle de agentes remotos, este trabalho implantou o agente global em uma máquina na *Amazon Web Services* (AWS), pois é um serviço de computação em *Cloud* seguro, escalável, confiável e que garante alta disponibilidade as aplicações (AWS. . . , 2022). De modo que dois contêineres do *Docker* foram implantados por meio do *Docker compose*, um deles sendo o próprio agente global para o caso apresentado na figura 3, e o outro, o banco de dados postgresql para o armazenamento das principais informações referentes a cada produção.

3.5.2 ARMAZENAMENTO DE INFORMAÇÕES

O armazenamento dos dados é um fator que obrigatoriamente deve ser suportado pelas modernas arquiteturas de controle, porque fornece todas as informações referente a cada produção. Tais análises podem ser efetuadas para futuras tomadas de decisão e, até mesmo, para a avaliação da arquitetura proposta.

O banco de dados utilizado foi o *postgresql*, pois é um banco de dados relacional, *Open Source*, contempla um grande número de usuários e funcionalidades, e se conecta facilmente com uma aplicação *Python*. A figura 8 ilustra a tabela criada pelo servidor para armazenar as principais informações de cada produção iniciada.

Figura 8 – Tabela para armazenamento de informações



Fonte: Autoria própria.

Cada item da tabela é listado e conceituado a seguir:

- user_id: identificador de cada produção inicializada.
- total_time: tempo total, em segundos, para se concluir uma determinada produção.
- production_date: data em que a produção foi iniciada.
- total_produced_a: total de peças do tipo A que foram produzidas.
- total_produced_b: total de peças do tipo B que foram produzidas.
- event_list: lista de todos os eventos utilizados na produção.
- machine_list: lista de todos os equipamentos utilizados na produção.

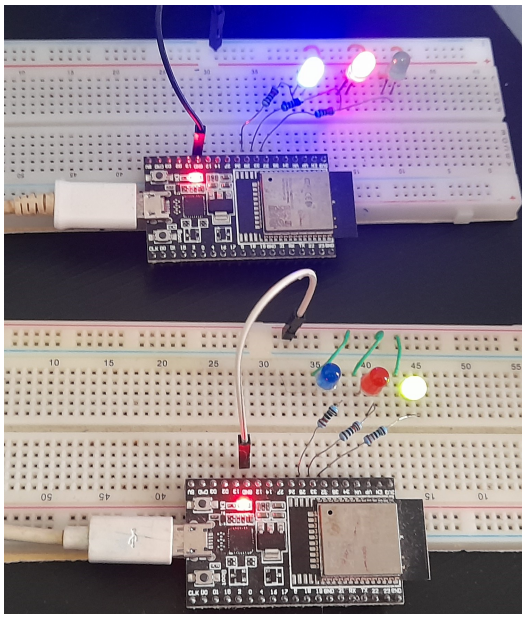
Na sequência, será ilustrado o funcionamento da arquitetura proposta com o servidor implantado localmente e na *Cloud*, fornecendo alta disponibilidade.

3.6 ILUSTRAÇÃO DA ARQUITETURA PROPOSTA

Nos testes de bancada, LEDs ilustram ações tomadas pelos dispositivos físicos. Isto ocorre a partir da implementação das máquinas de estados completa de cada braço robótico, L_1 e L_2, que estão implementadas em seus respectivos módulos locais, bem como o servidor, que possui a lista retirada do supervisor S_{abs} calculado por meio do caso prático apresentado. Embora este seja um teste de bancada, a inserção e integração física dos braços robóticos reais iria requerer apenas configurar os movimentos representados por cada evento. Para verificação do código, se dirigir ao *github* (ARQUITETURA. . . , 2022).

O resultado da simulação distribuída para os módulos locais, que possui o mesmo comportamento para a simulação local e na *Cloud*, é exposto na figura 9, que representa três possíveis estados de atuação para os dois módulos implementados.

Figura 9 – Atuação específica de cada módulo local



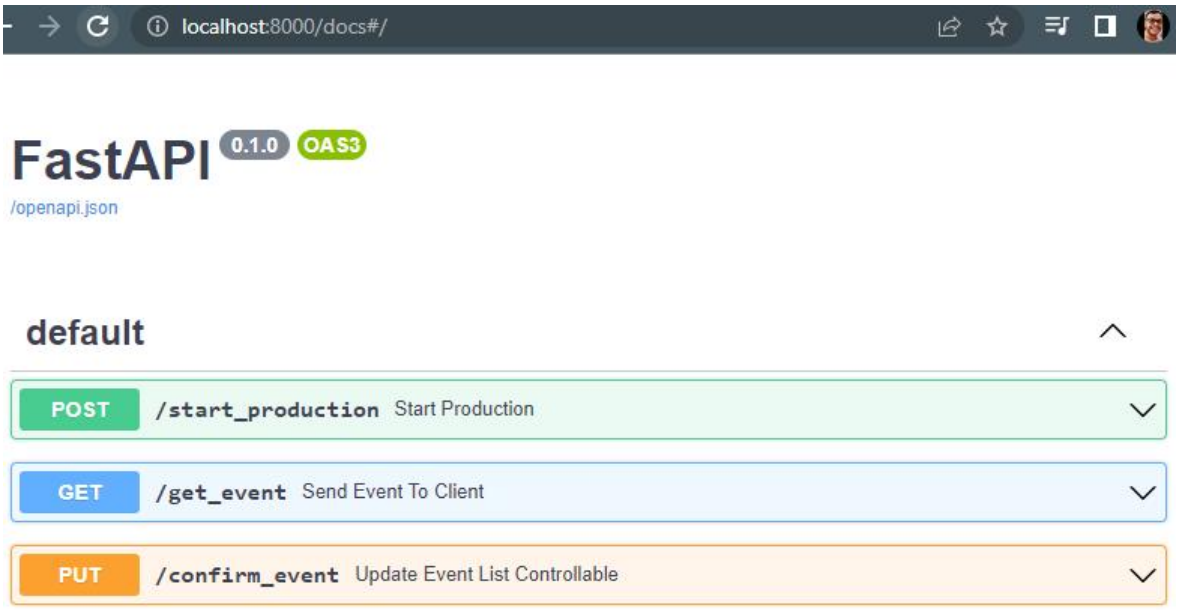
Fonte: Autoria própria.

O LED verde representa a configuração de quando o módulo local está requisitando um evento para dar início às suas ações. Quando recebe um evento esperado e se encontra numa região determinada como crítica, pois é uma área em que mais de um componente local pode atuar, o LED vermelho será ativado. Por fim, o LED azul indica que o agente local já está em fase de processamento, isto significa que o evento requisitado é o mesmo que o evento esperado. Assim, a lista de eventos pertencentes ao servidor vai sendo consumida até que uma nova produção seja iniciada, e com ela, uma nova lista é gerada. Essa geração pode ser obtida empiricamente, ou por meio do uso de métodos inteligentes. Neste trabalho a lista foi obtida por

meio de um algoritmo bio-inspirado em colônia de formigas, resultado de pesquisa complementar à aqui apresentada.

A atuação do servidor pode ser observado na figura 10, por meio de uma API interativa para a efetivação de diversos testes manuais, ou até mesmo automático, que validam seu funcionamento.

Figura 10 – API interativa do servidor na rede local



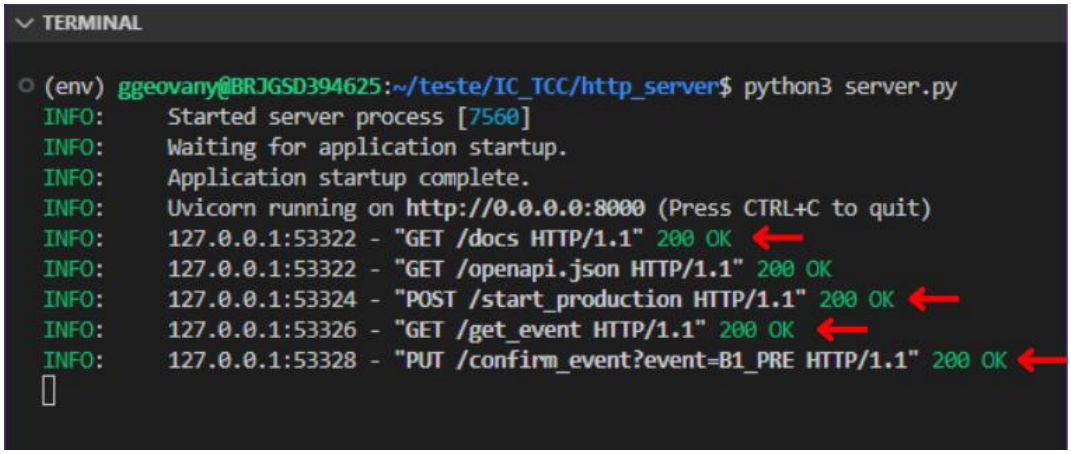
Fonte: Autoria própria.

Este teste ocorreu em ambiente de rede local, que pode ser observado na *URL* inserida no navegador. As rotas e seus respectivos significados são disponibilizadas na listagem a seguir:

- POST /start_production: inicialização de produção, podendo adicionar quantas peças de cada tipo deve ser produzida.
- GET /get_event: recuperação do primeiro evento da lista.
- PUT /confirm_event: confirmação de um evento recebido ou finalizado.

A primeira rota será utilizada para dar início a produção de um número finito de peças do tipo A e B, ao passo que as demais serão acessadas por cada módulo local, para recebimento de qual evento deve ser processado, envio de uma confirmação de que o evento recebido é igual ao esperado ou confirmação de um evento de finalização e, por fim, a rota da própria documentação (GET /docs). Para testar e validar o funcionamento de cada rota, os *logs* da figura 11 ilustram o servidor atuando no recebimento de cada requisição.

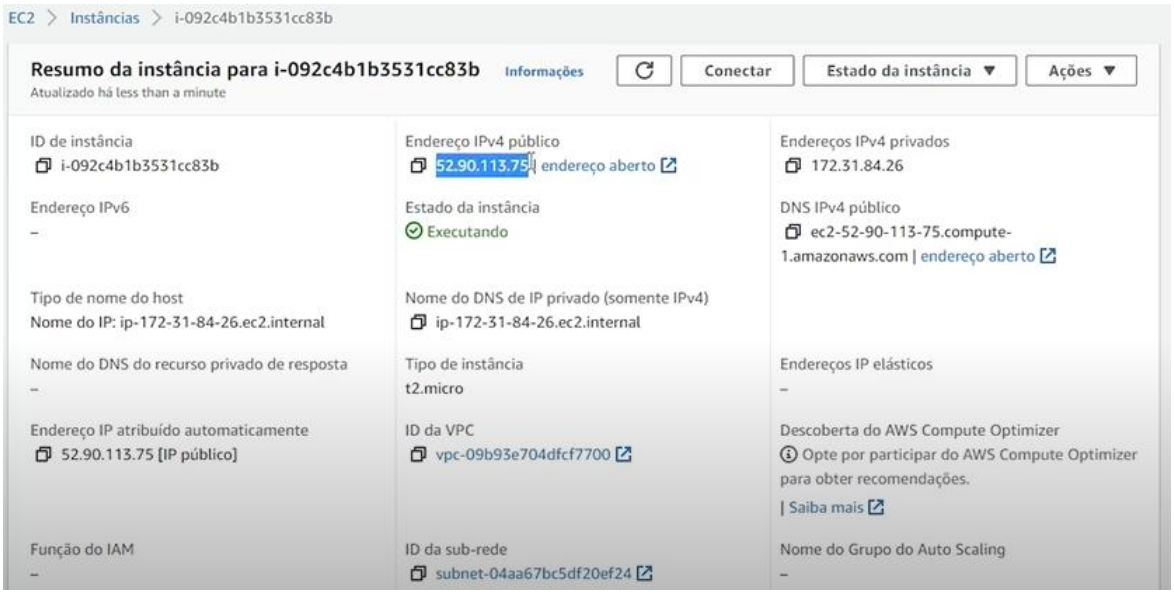
Figura 11 – *Logs* de funcionamento do servidor



Fonte: Autoria própria.

Para a efetivação dos testes com o componente lógico (agente central) implantado na AWS, uma máquina foi criada nesta plataforma, cujo endereço de IP foi atribuído automaticamente para o devido acesso, e pode ser observado na figura 12.

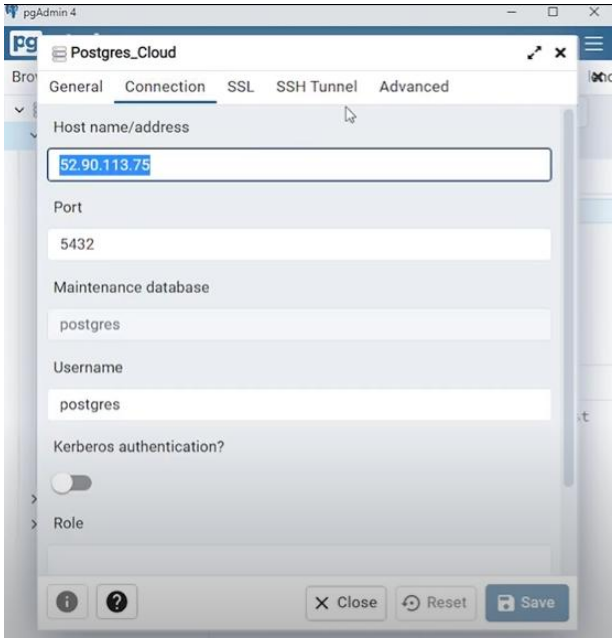
Figura 12 – Instância na *Cloud* para implantação do agente global e banco de dados



Fonte: Autoria própria.

A interface interativa do agente global é a mesma apresentada na figura 10, no entanto, a *URL* do navegador foi *52.90.113.75:80/docs*, que era o endereço público da instância. Enquanto que o banco de dados pôde ser acessado pela ferramenta *PgAdmin4*, que é *Open Source* e utilizada para gerenciar o *Postgresql*, indicando apenas o endereço da máquina e as credenciais de acesso, cujo apresentação é feita na figura 13.

Figura 13 – Acesso ao banco de dados na *Cloud*



Fonte: Autoria própria.

3.7 DISCUSSÃO DOS RESULTADOS

A arquitetura proposta nesta pesquisa possibilita uma série de benefícios, que vão desde a teoria que disponibiliza os sistemas de controle, até a possibilidade da construção de interfaces para configuração e acompanhamento de uma produção. Isso garante segurança, robustez e aspectos modernos para aplicações de cunho industrial.

A abstração na modelagem de cada equipamento industrial fornece a vantagem de um supervisor mais compacto, em contrapartida, aspectos como abstração, controle local e observabilidade parcial causam maior esforço para a construção do sistema, podendo se tornar um inconveniente.

Essa arquitetura pode ser implementada utilizando a abordagem monolítica para o cálculo do supervisor implementado no agente global, bem como fazendo uso de outras abordagem modulares advindas da TCS. Neste trabalho foi utilizado para o cálculo do sistema de controle S_{abs} a mesma abordagem para a obtenção do supervisor mais básico, que é o monolítico. No entanto, o sistema de controle como um todo é modular, pois o subconjunto de ações físicas que foram abstraídas estão implementadas nos sistemas de controle locais (L_1 e L_2).

A flexibilização de produção também pôde ser alcançada, porém fica a cargo da pesquisa complementar a disponibilização da melhor sequência de eventos para a produção de peças com características distintas. Isto quer dizer que os resultados dessa pesquisa apenas ilustram sua

viabilidade de uso em um sistema flexível de manufatura, sem, no entanto, explorar os aspectos de política de cooperação.

O servidor que implementa uma abstração da lógica de controle será acessado via *Internet*. Isto significa que esta estrutura pode controlar uma fábrica sem estar implementada necessariamente no mesmo local dos módulos supervisores locais. Além disso, como é uma máquina que pode implementar diversas aplicações na rede, poderia implementar o servidor e paralelamente um cliente (novo módulo local), para então ser controlado por outro agente global, ou seja, múltiplas fábricas ou módulos fabris poderiam ser controlados nessa arquitetura.

O *framework* utilizado para a construção da API em Python implementa, de forma nativa, uma documentação interativa para a execução de testes. Cada rota é disponibilizada com todas as informações referentes ao envio ou atualização como, por exemplo, de parâmetros que devem ser enviados ao executar um método GET ou um campo para a inserção do conteúdo da requisição.

A API que implementa o acesso ao servidor desta aplicação disponibiliza uma rota para a inicialização de produção, que provê a inserção de todas as configurações iniciais para se produzir um número 'x' de peças. Ou seja, representa o local que pode ser construído uma interface gráfica para o gerenciamento da fábrica.

Por fim, quando uma produção de um número de peças chega ao fim, seus dados são armazenados no banco de dados com as mais importantes características relacionadas a cada produção, como a quantidade de peças produzidas, tempo de produção, equipamentos relacionados, entre outros. Isso permite a retirada de diversas estatísticas quanto a tudo que foi produzido e a possibilidade de se monitorar com mais detalhes a aplicação.

3.8 TRABALHOS FUTUROS

A arquitetura base, aqui proposta, pode ser usada para a estruturação de outras aplicações industriais. Dentre as pesquisas conjecturadas para a sequência, inclui-se:

- Estruturação de aspectos de segurança de um sistema distribuído na *Internet*, que fornece criptografia, autenticidade, controle de acesso, etc.
- Construção de uma interface gráfica para o gerenciamento de produção e monitoramento da aplicação.
- Implementação dos movimentos físicos de dois braços robóticos em um ambiente controlado, para visualização física da arquitetura em funcionamento com um equipa-

mento de fábrica.

- Ampliação dessa estrutura para o controle de mais de uma fábrica.
- Criação de uma metodologia para ditar quais eventos desta arquitetura podem ser abstraídos.
- Por fim, utilização de outras abordagens mais eficientes para a síntese do supervisor via TCS para a implementação do agente central, como a modularização, por exemplo.

3.9 EXPOSIÇÃO PÚBLICA DA ARQUITETURA

Esta seção apresenta meios para visualização de todos os recursos do projeto, para eventuais contribuições ou melhor entendimento técnico das práticas e ferramentas adotadas na construção desta arquitetura, que inclui o repositório e um vídeo apresentação da aplicação em funcionamento.

3.9.1 DISPONIBILIDADE DO CÓDIGO

Para a verificação do código que implementa toda a arquitetura deste trabalho, tanto do servidor quanto dos dois módulos locais, o leitor deve se direcionar ao repositório do *github*, em (ARQUITETURA. . . , 2022).

Além do código que pode ser clonado para estudo ou até mesmo adição de novas funcionalidades, este repositório possui uma ampla documentação com todas as etapas para inicializar o agente global (servidor) e os dois módulos locais (clientes), respectivos aos braços robóticos 1 e 2.

Para o servidor, a documentação provê a inicialização fazendo uso de quatro abordagens, que são: instalando as dependências diretamente na máquina, por meio do ambiente virtual do *python*, por um único contêiner do *docker* e, finalmente, por meio do *docker compose*, que é o único meio que inicia adicionalmente o serviço do *postgresql*, não requisitando outras instalações deste banco de dados para o uso completo da aplicação.

Além disso, descrições detalhadas foram feitas para o funcionamento correto de toda arquitetura, como o encaminhamento de porta no sistema operacional (*Windows*), por exemplo, bem como a inserção de *links* que realizam o redirecionamento para páginas de conceitos ou instalações.

3.9.2 VÍDEO DE APRESENTAÇÃO DO FUNCIONAMENTO

O vídeo encontra-se em (TCC. . . , 2022), e fornece uma sequência de dois testes, o primeiro teste é executado no ambiente local e o outro com o servidor implementado numa instância da *AWS*. Todas as etapas foram incluídas na apresentação, desde a requisição para a produção de peças até o armazenamento das informações no banco de dados.

4 CONCLUSÕES FINAIS

A maturidade atingida por esta arquitetura no que diz respeito aos conceitos e tecnologias utilizadas, demonstra indícios de inúmeros ganhos para a TCS. O primeiro resultado obtido foi uma redução significativa entre as máquinas de estados, ambas foram obtidas de maneira monolítica, porém a primeira é sem abstração e a segunda, devido a arquitetura proposta, possui abstração de eventos, que foi de 251 estados e 526 transições para 207 estados e 434 transições, respectivamente.

Vale ressaltar que apenas quatro eventos foram abstraídos do sistema como um todo, mas em um caso real, um número muito maior de eventos seriam abstraídos. Ainda assim, a máquina de estados utilizada por esta arquitetura na implementação do agente central se manteria com um tamanho constante, ao passo que uma equivalente, obtido sem abstração, cresceria exponencialmente.

O que permite um ganho significativo na redução do tamanho do supervisor implementado pelo agente central comparado com um sistema de controle obtido de maneira tradicional, é a utilização do conceito da abstração de eventos permitido por esta arquitetura. Apesar de ambas as maneiras fazerem uso da abordagem monolítica para a síntese do supervisor, essa comparação não se limita apenas a esta forma de síntese. Pois mesmo que outras abordagens mais eficientes de síntese fossem utilizadas, devido a abstração, o agente central desta arquitetura sempre será mais compacto comparado a um sistema de controle obtido de maneira equivalente e sem abstração, como na modularização, por exemplo.

O agente central foi construído de modo que fosse escalável e leve, e este requisito foi atingido com esta arquitetura, pois foi implementado com técnicas e ferramentas modernas de desenvolvimento, como o *Docker* e *docker compose*. Podendo ser implantado em um ambiente que pode ser configurado para prover alta disponibilidade (*Cloud*) com um único comando. Isto é positivo, porque uma máquina com poucos recursos tem a capacidade de executar múltiplos contêineres, ou seja, múltiplas linhas de produção podem ser controladas por um agente central, e um novo agente central poderia controlar até mesmo outras fábricas.

Os módulos locais fazem uso de um microcontrolador muito utilizado para conexões com a internet, além de possuir um *framework* completo e nativo de desenvolvimento (ESP-IDF). No entanto, este módulo não foi explorado em maiores detalhes, no que diz respeito às suas limitações em um ambiente industrial e em quais equipamentos pode ser utilizado. Assim, recomenda-se uma maior concentração na implementação deste agente em trabalhos futuros.

Por fim, conclui-se que este trabalho provê uma base sólida para a construção de inúmeras aplicações de cunho industrial, pois possui características que garantem segurança à produção, via TCS, seja a nível lógico ou de infraestrutura de software e hardware e, por fim, que adentra aos aspectos modernos de produção, denominada indústria 4.0.

REFERÊNCIAS

AFZALIAN, A.; NOORBAKHSH, M.; NABAVI, S. PLC implementation of decentralized supervisory control for dynamic flow controller. In: IEEE. 2008 IEEE International Conference on Control Applications. [*S. l.: s. n.*], 2008. P. 522–527.

AKESSON, K. *et al.* **Supremica**. [*S. l.*], 2019. Disponível em: http://www.supremica.org/.

ARQUITETURA para a implementação de controle supervisório descentralizado. [*S. l.*], 2022. Disponível em: https://github.com/GeovanyADC/IC_TCC.

AWS. [*S. l.*], 2022. Disponível em: https://aws.amazon.com/pt/.

BANGEMANN, T. *et al.* Integration of Classical Components Into Industrial Cyber-Physical Systems. **Proceedings of the IEEE**, v. 104, n. 5, p. 947–959, mai. 2016.

CASSANDRAS, C. G.; LAFORTUNE, S. **Introduction to discrete event systems**. [*S. l.*]: Springer Science & Business Media, 2009.

DOCKER overview. [*S. l.*], 2022. Disponível em: https://docs.docker.com/get-started/overview/.

FASTAPI. [*S. l.*], 2022. Disponível em: https://fastapi.tiangolo.com/.

HARRISON, R.; VERA, D.; AHMAD, B. Engineering Methods and Tools for Cyber-Physical Automation Systems. **Proceedings of the IEEE**, v. 104, n. 5, p. 973–985, mai. 2016.

KUROSE, J. F.; ROSS, K. W. **Redes de Computadores e a Internet - Uma Abordagem Top-Down**. 6. ed. [*S. l.*]: Pearson Education do Brasil Ltda, 2013.

LEAL, A. B.; DA CRUZ, D. L.; HOUNSELL, M. d. S. Supervisory control implementation into programmable logic controllers. In: IEEE. 2009 IEEE Conference on Emerging Technologies & Factory Automation. [*S. l.: s. n.*], 2009. P. 1–7.

LIU, Y. *et al.* Review on cyber-physical systems. **IEEE/CAA Journal of Automatica Sinica**, v. 4, n. 1, p. 27–40, jan. 2017.

OFFICIAL IoT Development Framework. [*S. l.*], 2022. Disponível em: https://www.espressif.com/en/products/sdks/esp-idf#:~:text=Open%2DSource,under%20a%20compatible%20permissive%20license..

OVERVIEW. [*S. l.*], 2022. Disponível em: https://docs.docker.com/compose/.

PRENZEL, L.; PROVOST, J. PLC implementation of symbolic, modular supervisory controllers. **IFAC-PapersOnLine**, Elsevier, v. 51, n. 7, p. 304–309, 2018.

QUEIROZ, M. H. D.; CURY, J. E. R. Modular Supervisory Control of Large Scale Discrete Event Systems. In: DISCRETE Event Systems: Analysis and Control. Proc. WODES'00. [*S. l.*]: Kluwer Academic, 2000. P. 103–110.

RAMADGE, P. J.; WONHAM, W. M. The control of discrete event systems. **Proceedings of the IEEE**, IEEE, v. 77, n. 1, p. 81–98, 1989.

REST API (Introduction). [*S. l.*], 2020. Disponível em: https://www.redhat.com/pt-br/topics/api/what-is-a-rest-api.

ROSA, M. *et al.* Efficient Implementation of Distinguished Controllers for Discrete-Event Systems. **IFAC-PapersOnLine**, v. 50, p. 1187–1192, 2017.

SCHOUTEN, R. *et al.* Synthesis and Implementation of Distributed Supervisory Controllers with Communication Delays. IEEE.

SILVA, A. L.; RIBEIRO, R.; TEIXEIRA, M. Modeling and control of flexible context-dependent manufacturing systems. **Information Sciences**, v. 421, p. 1–14, 2017.

SIPSER, M. **Introdução à Teoria da Computação**. 2. ed. [*S. l.*]: Cengage Learning, 2012.

TANENBAUM, A. S. **Computer Networks**. [*S. l.*]: Pearson Education, inc, 2003.

TCC - Geovany Aparecido Duarte Câncio. [*S. l.*], 2022. Disponível em: https://youtu.be/z4tXGZeBhdk.

VENV - Criação de ambientes virtuais. [*S. l.*], 2022. Disponível em: https://docs.python.org/pt-br/3/library/venv.html.

WEN, S.; GUO, G. Control and resource allocation of cyber-physical systems. **IET Control Theory Applications**, v. 10, n. 16, p. 2038–2048, 2016.

WONHAM, W. **Notes on Discrete Event Systems**. [*S. l.: s. n.*], 2002. University of Toronto.