ANAIS DOS TRABALHOS DE DIPLOMA – JULHO/2022
UNIVERSIDADE FEDERAL DE ITAJUBÁ
ENGENHARIA DE CONTROLE E AUTOMAÇÃO

DESENVOLVIMENTO DE UM GÊMEO DIGITAL "*DIGITAL TWIN*" DE UM PROCESSO INDUSTRIAL USANDO O SOFTWARE ABB ROBOTSTUDIO

Gustavo Amaral Leal de Souza **Rebeca Ribeiro Pinto**

Orientador: Prof. Kleber Roberto da Silva Santos
Instituto de Engenharia de Sistemas e Tecnologia da Informação (IESTI)

***Resumo* - Este artigo apresenta o desenvolvimento de uma representação virtual de um processo industrial robotizado (*Digital Twin*) utilizando o software ABB RobotStudio. Dessa forma, foi construído um modelo CAD que representa o processo utilizado. Esse arquivo foi implementado e configurado juntamente a um robô manipulador no ambiente do RobotStudio. Sendo assim, essa maquete comunica-se com um CLP simulado no software B&R Automation Studio, contendo uma lógica de automação que controla a execução completa do processo industrial. Ademais, utilizou-se o protocolo OPC-UA para realizar a comunicação entre ambos os softwares utilizados.**

Palavras-Chave: *Digital Twin*, OPC-UA, Automação, CLP, RobotStudio, Automation Studio, Indústria 4.0.

I. INTRODUÇÃO

Ao longo dos anos, a Indústria 4.0 tem ganhado um amplo espaço nos processos de manufatura do Brasil, com destaque para o setor automotivo [1]. Essa importância se deve aos pilares da 4ª Revolução Industrial. Um desses pilares é a Internet das Coisas (*IoT*), que de acordo com [2] "é a rede global de dispositivos físicos inter-relacionados, tais como, sensores, atuadores, aplicativos inteligentes, objetos, dispositivos computacionais, máquinas mecânicas, e pessoas que estão se tornando uma parte essencial da internet."

Ademais, a Indústria 4.0 está relacionada com Sistemas Cyber-Físicos e, devido ao *IoT*, é possível elaborar uma representação virtual de equipamentos que permite o tráfego de dados em tempo real, e consequentemente possibilitando atividades autônomas [3]. Essa representação virtual, denominada Gêmeo Digital, proporciona a realização de análises, seja de equipamentos isolados ou plantas industriais completas [4].

Dessa maneira, tem-se que o *Digital Twin* trata-se de uma ferramenta importante para o comissionamento virtual. Segundo [5] comissionar é criar e realizar testes em um programa que controla um processo virtual e é associado com a aplicação real. Por conseguinte, executar testes dessa forma garante diversos benefícios. Dentre eles, pode-se citar a previsão de cenários, tendo uma visão mais robusta das condições e eventos. Têm-se também, uma economia de recursos pois é possível analisar previamente a viabilidade do processo. Além disso, agrega uma maior qualidade e eficiência para o projeto através da otimização do mesmo [6].

Portanto, a partir dessa motivação, o principal foco deste projeto é validar se o software RobotStudio é uma ferramenta eficaz para a criação de um *Digital Twin*. Sendo assim, o presente artigo apresenta o desenvolvimento de um Gêmeo Digital (*Digital Twin*) de um processo industrial robotizado utilizando o ambiente do software ABB RobotStudio. A lógica do processo industrial utilizada foi a de uma envasadora de doces descrita e contida em [7].

Para tal, este documento começa explicando os conceitos e softwares utilizados no projeto na seção II. Na seção seguinte, Processo e Criação da Maquete Virtual, é apresentado o funcionamento da envasadora em questão, além do modo de criação do ambiente visual de simulação. Ademais, na seção IV, Elementos do Processo e do Software RobotStudio, tem-se a definição de recursos próprios do RobotStudio que foram essenciais para a elaboração do *Digital Twin*. Já no capítulo subsequente, Configuração do Robô, é mostrado a criação da garra, além dos mecanismos utilizados para a movimentação desta ferramenta, bem como do robô. Na seção VI é mostrado como a movimentação das peças da maquete foi feita, além da lógica responsável por esse processo. A próxima seção, Integração e Comunicação, apresenta como os dois softwares utilizados se comunicam e as devidas configurações necessárias. A seção VIII é responsável pelos resultados do projeto, ou seja, se é possível realizar a criação do *Digital Twin* a partir da ferramenta RobotStudio. Na Conclusão, tem-se algumas considerações que foram observadas durante a elaboração

do presente trabalho. Finalmente, na última seção, apresenta possíveis alterações e implementações que podem ser realizadas neste projeto.

II. Fundamentação Teórica

Nessa secção será abordado os programas e as definições das tecnologias utilizadas para o desenvolvimento da aplicação em questão.

II.1 RobotStudio

De acordo com o manual do usuário [8], o RobotStudio se trata de um software de modelagem, programação off-line, e simulação de células robóticas. Segundo o site do fabricante [9], a ferramenta possibilita que a programação do robô seja feita sem interferir no processo em tempo real. Dessa forma, é possível realização de treinamentos, testes e otimização do sistema de forma off-line.
Além disso, o software possui diversas características, tais como, comissionamento virtual, tecnologia de realidade aumentada e simulação da posição de parada. Aliado a esses atributos, a possibilidade de criar um ambiente virtual que seja a cópia da planta industrial (*Digital Twin*) e a facilidade de integração do robô com as demais partes da célula de manufatura, foram os principais motivos da escolha do programa RobotStudio.

II.2 B&R Automation Studio

Assim como o RobotStudio, o B&R Automation Studio é um membro do Grupo ABB. Ademais, esse software permite uma facilidade para comunicação entre máquinas, além de possibilitar a programação de CLP's (Controladores Lógicos Programáveis) em diversas linguagens [10].
Dentre as principais características desse programa, as que contribuíram para a utilização deste foram a programação versátil em linguagens IEC 61131-3, atualização dinâmica dos componentes da planta e o acesso simples e controlado aos dados da máquina com OPC (*Open Plataform Comunication*) [10].

II.3 Digital Twin

O conceito de Digital Twin foi definido, em 2002, por Dr. Michael Grieves e John Vickers [11], que consiste na ideia de uma informação digital construída a partir de um sistema físico.
Além disso, segundo [12], o gêmeo digital trata-se de um perfil virtual, que contém o comportamento dos objetos ou processos. Dessa forma, é possível prever e detectar tendência de defeitos no sistema, otimizar a performance dos equipamentos de manufatura, diminuição do custo para o desenvolvimento de um novo produto.
Portanto, essas qualidades e importâncias do *Digital Twin* para a automação serviram de motivação para o desenvolvimento do presente artigo. Sendo assim, foi elaborado uma célula de manufatura no RobotStudio com um robô integrado, que utilizando o protocolo OPC-UA (*Open Plataform Comunication – Unified Architecture*), comunica-se com a lógica CLP feita em Texto Estruturado no B&R Automation Studio.

III. Processo e Criação da Maquete Digital

III.1 Apresentação do processo:

A fim de elaborar um *Digital Twin* para a validação do RobotStudio, foi escolhido um processo robotizado de uma envasadora de doces.
Esse processo se inicia com o posicionamento de uma embalagem, que chamaremos de balde na frente do atuador pneumático duplex geminado, isto é feito pelo robô IRB_120_3_58_1 fabricado pela ABB. Em seguida, ocorre a detecção do objeto pelo sensor de presença permitindo a atuação do cilindro pneumático, que movimenta o balde para o local em que o silo se encontra. Após o término do despejo dos doces, o mesmo atuador geminado posiciona o balde no início da esteira.
Ao chegar na esteira, o balde se desloca para o ponto em que acontece o processo de colocação da tampa. Primeiramente, ocorre o bloqueio do movimento do balde, realizado pela atuação de um cilindro pneumático localizado na esteira. Em seguida, outro pistão é responsável por posicionar a tampa em cima do balde. Com o término dessa ação, e o recuo dos atuadores, o recipiente é liberado para continuar o processo.
Na etapa posterior, para a selagem da tampa, é necessário que o balde novamente seja travado na esteira. Após esse bloqueio, um atuador pneumático vertical executa a fixação da tampa no recipiente. Por último, o objeto é liberado e se desloca para o final da esteira, em que ocorre o armazenamento do produto.

III.2 Criação da maquete:

Para que seja possível o desenvolvimento de uma aplicação *Digital Twin*, é necessário a criação de uma maquete virtual que represente o processo. Para tal, primeiramente foram importados, de bibliotecas *Open Source* [13], diversos modelos CAD's dos equipamentos necessários, dentre eles: atuadores pneumáticos, mesa, esteira, canaleta e silo.
Além disso, é importante destacar que para gerar movimento, os objetos devem ter cada uma de suas peças separadas dentro dos respectivos arquivos CAD's. Por exemplo, para movimentar um atuador pneumático é necessário que a haste deste seja passível de ser manipulada isoladamente.
Esses objetos foram organizados e dispostos no software Autodesk Inventor, de maneira a criar uma linha de produção de uma envasadora de doces, seguindo a ideia apresentada na seção *III.1*. A Figura 1 abaixo mostra a solução CAD criada no programa citado.

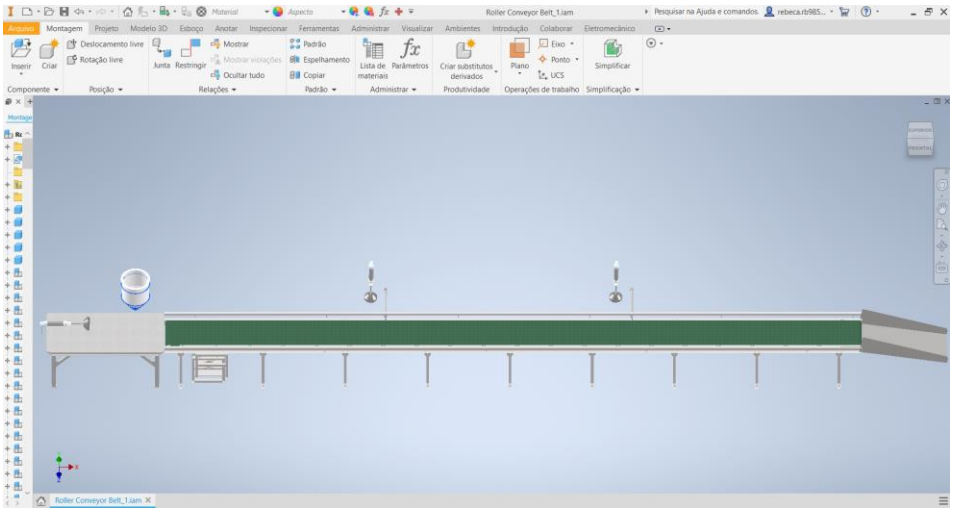


Figura 1: Modelo CAD do processo no Inventor
Fonte: Autoral

Em seguida, o arquivo ".STEP/.STP" da montagem do CAD foi importado na *Station* do RobotStudio. Após inserido, fez-se necessário o ajuste de posição e rotação do modelo a fim de que esteja na forma correta de acordo com plano cartesiano.

IV. Elementos do Processo e do Software RobotStudio

Este item apresenta a definição das ferramentas utilizadas para a criação do *Digital Twin* no RobotStudio, incluindo conceitos nativos do programa.

IV.1 Atuadores:

Nesta seção será apresentado os atuadores pneumáticos usados na aplicação. Esses cilindros são apresentados a seguir:

- Cilindro Duplex Geminado: Esse atuador é composto por dois cilindros unidos, mas que podem operar independentemente. Dessa forma, obtém-se, 3 ou 4 posições de trabalho dependendo dos cursos dos cilindros. Nesse projeto, esse atuador pneumático tem a função de levar o balde tanto para a posição de envase dos doces quanto para a esteira. Para tal, como descrito na seção III (Processo e Criação da Maquete Digital), é possível manipular a haste desse cilindro separadamente, podendo assim criar movimentos diferentes para esta, e obter as 3 posições de trabalho necessárias para a aplicação.
- Cilindro de Dupla Ação: Esse cilindro é composto de um pistão que é acionado pelo ar comprimido em ambos os cursos. Neste projeto, são usados 4 atuadores pneumáticos de dupla ação, posicionados tanto na ação de tampar e selar o balde, quanto como trava, para que essas operações sejam possíveis. A movimentação desses cilindros é feita da mesma forma que no duplex geminado utilizado na maquete, através da criação de movimentos de avanço e recuo da haste.

IV.2 Sensores:

Este item apresenta os sensores utilizados na maquete virtual permitindo a detecção das ações dos objetos. Tais sinais podem ser externados para o controlador do processo, que executará a lógica de automação, ou utilizados no acionamento de eventos internos do RobotStudio, permitindo a interação entre objetos da maquete. Esses elementos são definidos a seguir:

- Sensores Lineares (*LinearSensor*): Este sensor detecta o contato entre um objeto da maquete e uma linha sensora. Isto dificulta que outros objetos ativem o sensor de maneira indesejada, pois, ao colocá-lo na superfície da mesa, por exemplo, para ativá-lo, é necessário que o objeto esteja em contato com a mesa e toque a linha do sensor. Ou seja, os cilindros pneumáticos e o braço robótico nunca irão ativá-lo, uma vez que eles não tocam o plano da mesa.
- Sensores Planos (*PlaneSensor*): Esses sensores têm uma superfície de detecção que pode ser regulada tanto em tamanho quanto em orientação. Deve-se atentar que qualquer objeto que entrar em contato com o plano, ativará o sensor, portanto este deve ser posicionado corretamente, para evitar ativações indesejadas.
- Sensores de Colisão (*CollisionSensor*): Para que o cilindro pneumático fosse capaz de empurrar o balde, foram utilizados os chamados sensores de colisão. Para tal, deve-se configurar quais objetos devem se colidir e a distância da proximidade entre os objetos para que o sensor se ative. Ademais, alerta-se que, sempre que ocorrer a

colisão entre esses dois objetos configurados, o sensor de colisão se ativará. Portanto, caso a ativação seja necessária somente em algum momento em específico, como é o caso desse projeto, é preciso fazer uma alteração da lógica, de forma a ativar e desativar a entrada *Active* do sensor, como mostrado na Figura 2.

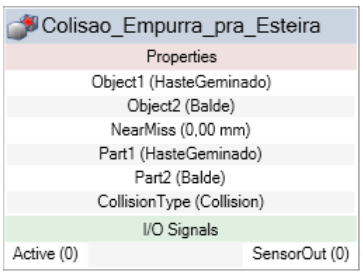


Figura 2: Sensor de Colisão
Fonte: Autoral

IV.3 Formas Geométricas e Equipamentos:

Nesta seção será mostrado as formas geométricas e equipamentos que foram utilizados para a criação de determinadas ferramentas do processo. Esses objetos são citados a seguir:

- Cilindro *(Solid Cylinder)*: Esse cilindro se trata de um item pertencente a uma biblioteca nativa do RobotStudio, chamada *Solid.* Essa forma geométrica foi utilizada para a representação do balde, com altura de 180 *mm* e um raio de 80 *mm.*
- Paralelepípedo *(Solid Box)*: Esse objeto se trata também de um item pertencente a biblioteca *Solid.* Essa forma geométrica foi utilizada para a criação da garra, sendo o braço com altura *150mm*, comprimento 20 *mm* e largura 20 *mm.* A base da garra possui 220 *mm* de largura, 20 *mm* de comprimento e 10 *mm* de altura.
- *IsoPlate Small Bizel:* Esse item pertence a biblioteca de equipamentos nativa do RobotStudio. Trata-se de um disco de cor metálica que é utilizado no projeto como conector entre o robô e o efetuador.

IV.4 Blocos Lógicos:

Nesta seção será apresentado as ferramentas que constituem a parte lógica no RobotStudio. Esses blocos lógicos são apresentados a seguir:

- Movimento Linear (*LinearMover*): Esse bloco é responsável por realizar os movimentos lineares de todos os objetos de um ponto a outro, de acordo com a direção, sentido, velocidade e objeto configurado. Essa ação foi utilizada no projeto para criar a movimentação dos cilindros e do balde, através do acionamento e paralização do sinal de ativação do bloco na entrada *Execute*.
- *LogicSRLatch*: A partir de um comando *Set*, a saída desse bloco é ativad*a*, e continua dessa forma mesmo quando essa instrução é cessada. Esse sinal somente é desativado após o bloco receber o comando *Reset*. No presente projeto foi utilizado o bloco *LogicSRLatch* para garantir que o sinal vindo dos temporizadores permanecesse ativo pelo tempo desejado. Na Figura 3, é apresentado este bloco, com as entradas *Set* e *Reset*, e as respectivas saídas. Acrescenta-se que a saída *Output* é referente ao *Set*, enquanto *InvOutput* refere-se ao *Reset*.

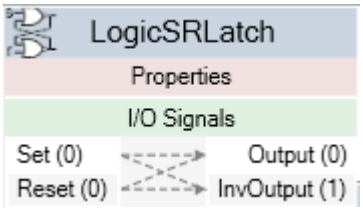


Figura 3: Bloco *Set/Reset*
Fonte: Autoral

- Blocos de Visibilidade (*Hide/Show*): O bloco *Hide* e o bloco *Show* são utilizados para alterar a visibilidade de algum objeto. Dessa forma, no projeto da envasadora, na falta de um CAD apropriado para o cilindro duplex geminado, foi necessário a utilização de hastes auxiliares. Sendo assim, a visibilidade era alterada, a partir desses blocos, sempre que necessário, para garantir uma movimentação fluida e contínua do atuador pneumático.
- Blocos de Comunicação (*OpcUa Client)*: Este bloco é responsável por realizar a comunicação entre o RobotStudio e outros softwares através do protocolo OPC-UA. Ao ser configurado corretamente, esse bloco atua como o cliente recebendo as informações coletadas pelo servidor (*OPC-UA Server*). Nesta ferramenta, pode-se configurar as características da comunicação, como a porta e protocolo de transporte, sendo que neste caso foi utilizado o OPC TCP.

IV.5 Configuração e Programação:

Nesta seção serão apresentados os seguintes itens:

- *RobotWare*: Trata-se de uma das famílias de controladores da ABB, que garante uma otimização do projeto. Na aplicação foi utilizada a versão 6.12.04.00 disponível para o robô escolhido e selecionada na criação do projeto.
- *RAPID*: É a linguagem de programação utilizada para controlar os robôs da ABB. Além disso, trata-se de um dos métodos existentes para a programação do robô, sendo que os outros métodos são traduzidos automaticamente para essa linguagem. Neste projeto, criou-se pontos

ensinados para o robô, que foram organizados no caminho a ser percorrido. Esse caminho posteriormente foi convertido para a linguagem *RAPID*.

V. CONFIGURAÇÃO DO ROBÔ

Inicialmente, foi adicionado uma nova mesa, paralela com a já existente na maquete, a fim de posicionar o robô em cima desta. Dessa forma, o IRB_120_3_58_1 foi acrescentado na mesa, e o controlador deste foi escolhido, respeitando o modelo do robô e a versão do *RobotWare* (6.12.04.00).
Este capítulo apresenta as ações realizadas para a configuração do robô, desde a criação da garra, até realização do movimento do autômato.

IV.1 Criação da garra:

Com a falta de uma opção de ferramenta para manipular o balde de forma adequada, e devido ao RobotStudio ter ferramentas necessárias para criação e adição de mecanismos, foi criada uma nova garra. Para isso, como base foi utilizado um *IsoPlate Small Bizel* e uma forma geométrica do tipo *Solid Box*. Essas peças foram unidas em um grupo chamado *GripperBase*. Para representar os dedos da garra foi utilizado dois novos paralelepípedos renomeados para *Left Finger* e *Right Finger*. A imagem da Figura 4 apresenta a garra desenvolvida.

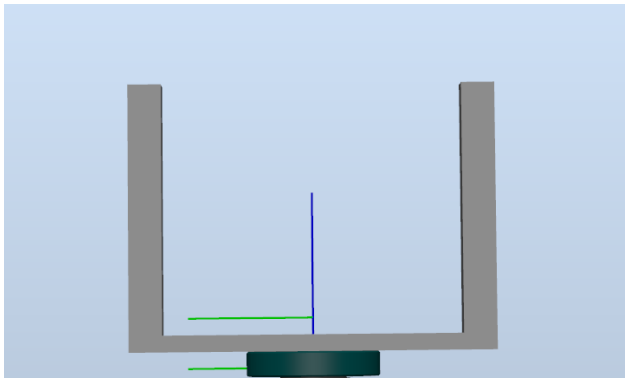


Figura 4: Construção da garra
Fonte: Autoral

No RobotStudio, existe opção de criação de mecanismos associando movimento às peças. Para isso, é necessário fazer a configuração dos *links*, *joints*, *tooldata* e *poses* de acordo com a ferramenta desejada para a aplicação.
Dessa forma, os *links* são criados de acordo com as peças da ferramenta que está sendo desenvolvida. Além disso, a base do equipamento é configurada como *BaseLink*.
No caso das juntas, é necessário definir se o movimento realizado será prismático ou rotacional de acordo com a ação de cada junta. Ademais, deve ser estabelecida a relação de hierarquia de cada elo que compõe a junta. Para o *tooldata* é preciso definir especificações da ferramenta, como massa, centro de gravidade e momento de inércia. Finalmente deve-se definir as *poses* que são necessárias para o movimento. Após essas configurações pode se criar sinais de modo a associar eventos a este, e posteriormente, adicionar ações a partir desses eventos.
Portanto, após a construção da garra, fez-se necessária a criação de um mecanismo para a realização da captura do balde. Sendo assim, foram criados três *links* a partir dos *Solid Box* usados para representação dos componentes da garra. Em relação as juntas, as formas geométricas referentes aos dedos da garra foram configuradas como "filhas" do componente *GripperBase*. Além disso, como o movimento para realizar a captura do balde é linear, foi utilizado duas juntas do tipo prismáticas. No caso do *tooldata* configurou-se a massa de 2.5 *kg* e 20 *mm* no eixo Z. Por fim, foram definidas as posições necessárias para o movimento da garra. Dessa maneira criou-se a *HomePose* que representa a posição da garra totalmente aberta, definida em 90 *mm* para ambos os dedos. Foi criada também, a *SyncPose* que corresponde a posição de captura do balde definida em 80 *mm* para os dois dedos.
Após a criação do mecanismo, foi adicionado um sinal relacionado a garra, e criado dois eventos atrelados a esse sinal. Esses eventos são responsáveis pela movimentação da garra para atingir as posições *HomePose* e *SyncPose*. Em seguida, foram associadas ações a estes eventos a fim de realizar a dinâmica de captura e a liberação do balde.
Ademais, após a configuração do manipulador, fez-se o acoplamento da garra ao robô, através do reposicionamento do equipamento. Para um maior detalhamento de como criar e realizar a configuração do mecanismo da garra, ver Apêndice A.

IV.2 Movimentação do robô:

Para que o robô faça os movimentos desejados, é necessário a programação deste no RobotStudio, através da criação de elementos do tipo *targets e paths*. Primeiramente, deve-se criar uma posição inicial do robô (*Home_Position*), entretanto precisa-se atentar para que não ocorra erro de singularidade. Um caso de singularidade identificado durante o desenvolvimento do projeto, foi quando duas juntas estão com seus eixos de movimentação alinhados, ocasionando assim a perda de um grau de liberdade o que limita as possibilidades de movimentação do robô.
Acrescenta-se também, que no RobotStudio é possível realizar o ensino de posições do robô através do mecanismo *Teach Target*. Dessa forma, utilizando as ferramentas de movimento do robô, define-se um ponto que o manipulador deve percorrer. A partir de um ponto já criado, é possível também criar outros pontos pela função *Offset*, em que se define um deslocamento fixo em relação ao original. Além disso, pode-se realizar essa criação de pontos através do mecanismo *SetPosition*, no qual define-se uma posição clicando em algum local da maquete virtual. Após essa definição, pode-se ainda realizar rotações e translações como ajustes para atingir a posição desejada. Caso o robô necessite realizar um caminho muito extenso, é recomendado a criação de mais pontos intermediários a fim de controlar as *poses* do manipulador

para que ele faça os movimentos necessários para percorrer o caminho total.
Esse caminho pode ser definido através do conjunto de pontos previamente criados em que o robô deve percorrer de acordo com o papel na aplicação. Essa trajetória pode ser percorrida com movimentos lineares ou de junta, além de se definir a velocidade utilizada, bem como o parâmetro *zone*, que representa o quão preciso o robô vai atuar para atingir os pontos configurados. Ademais, é possível a criação de instruções de ações, adicionadas ao caminho para que o robô realize todo o processo desejado.
Para o presente projeto da criação de um *Digital Twin* de uma envasadora de doces, foram criados, além do *Home_Position*, a posição de captura do balde (*PickPoint*) e a posição de liberação do balde na mesa (*PlacePoint*). Além disso, criou-se pontos intermediários devido ao percurso que o robô deveria realizar, evitando assim que ocorresse movimentos abruptos ou até erros em razão de pontos fora do alcance. Dessa forma, foi totalizado oito pontos, apresentados na Figura 5.
A partir desses pontos, criou-se o caminho responsável pela ação de movimentação robô, como mostrado na Figura 5, pelas linhas em amarelo que levam a garra do robô ao balde. Dessa forma, inseriu movimentos junta (*MoveJ*) para os pontos respeitando a ordem do processo. Esses movimentos têm velocidade de 200 mm/s e o parâmetro *zone* foi definido como *fine*, garantindo a maior precisão para a realização da trajetória.
Para concluir o caminho foi necessário criar os eventos de captura e liberação do balde a partir do sinal previamente criado, ou seja, os eventos do movimento da garra até as posições, *HomePose* e *SyncPose*. Em seguida, adicionou ações *Attach* e *Dettach* para balde. Posteriormente, inseriu-se duas instruções de ação do tipo *Reset*, a partir do sinal criado. Uma instrução foi adicionada no início para garantir que a garra comece na posição inicial, ou seja, totalmente aberta, e a outra foi inserida no momento da liberação do balde, para que a garra se abra e o balde seja solto.
Além disso, foi criado outra instrução do tipo *Set* no momento da captura do balde. Adicionou-se também duas instruções de ações do tipo *WaitTime*, a fim criar um atraso de tempo
A primeira é inserida antes de capturar o balde para segurança da aplicação, e a segunda, no final, para configurar o tempo de simulação do restante do processo.
Após a realização de todo esse desenvolvimento, é necessário fazer a sincronização dos pontos e caminhos criados para o código RAPID, responsável pela atuação do robô.
Finalmente, para ser possível o retorno da maquete para o estado inicial do processo, essa posição foi salva através do recurso *Reset* da simulação do RobotStudio. Para um maior detalhamento de como realizar a configuração do robô, ver o Apêndice B.

VI. Movimentação e Elaboração da Lógica CLP

V.1 Movimentação das peças:

Para que a peça faça os movimentos desejados, é necessário o posicionamento de sensores e a criação de ações de movimentos. Esses componentes podem ser inseridos a partir do item *Station Logic*.
No caso do *Digital Twin* da envasadaora de doces, em questão, a dinâmica necessária para realizar a movimentação foi dividida nas seguintes etapas:

V.1.1 Pré-esteira:

Esta etapa começa após o posicionamento do balde pelo robô no local determinado, sendo a presença de um balde detectada por um sensor de presença. O sensor escolhido foi do tipo linear (*LinearSensor*), para evitar que ele fosse ativado pelo pistão em movimento, como explicado na seção IV.

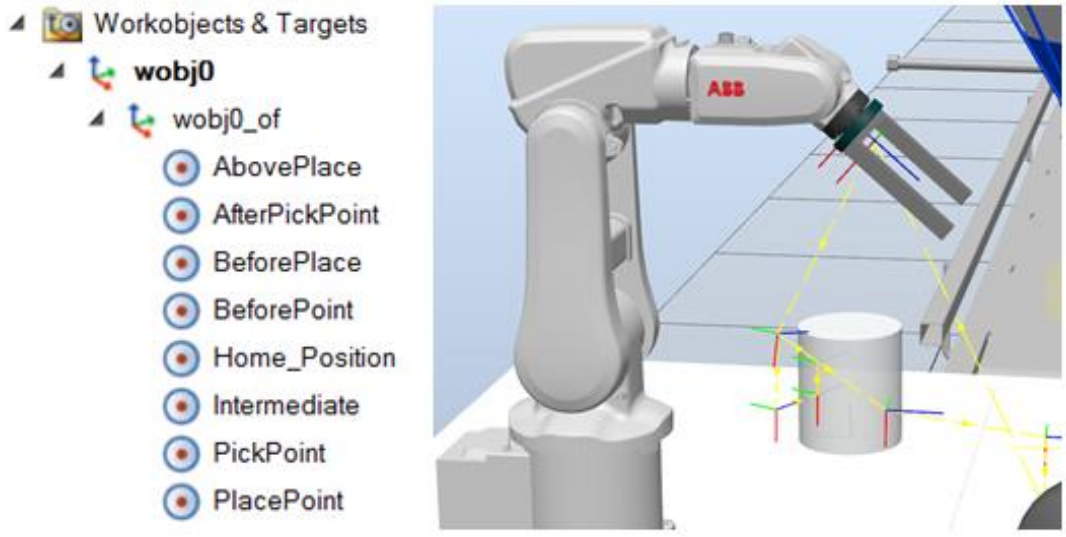


Figura 5: Criação dos Targets e Path
Fonte: Autoral

Em seguida, após um intervalo de tempo criado por um temporizador, a fim de esperar o robô retornar para a posição inicial, ocorre a primeira movimentação do cilindro geminado, ou seja, o balde se move até o silo. Essa movimentação se trata apenas do deslocamento da haste do modelo CAD inserido, sendo que, essa translação da peça de forma isolada, só é possível devido à importação dos arquivos com componentes passíveis de edição. Dessa forma, primeiramente, foi criado blocos manipuladores *LinearMover* para o balde e para o pistão, além de um sensor de fim de curso, do tipo *PlaneSensor*, localizado dentro do cilindro e um sensor de colisão entre o atuador e o balde.

Para que o pistão atinja a posição mais distante, ou seja, para que alcance a esteira, é necessário que exista novamente blocos do tipo *LinearMover*, um sensor linear de presença embaixo do silo, um novo sensor de colisão, além de um sensor plano no início da esteira. Ressalta-se que antes do movimento acontecer, existe um segundo temporizador representando o envasamento dos doces no balde. Para esses atrasos de tempo, como comentado na seção IV, foi necessário o uso de blocos do tipo *LogicsSRLatch*, a fim de que a saída do temporizador tivesse duração suficiente para o acionamento da lógica seguinte. Acrescenta-se que devido ao tamanho da mesa, por questão de simulação, foi utilizado ações do tipo *Hide* e *Show* para criar hastes intermediárias durante esse processo.

V.1.2 Processo de colocação e selagem da tampa:

Ambas as etapas ocorrem de maneira semelhante, entretanto, primeiro acontece o processo de colocação da tampa, e após a conclusão desta, e a movimentação do balde pela esteira até a estação de selamento, começa o processo de selagem. Essas etapas se iniciam quando o balde atinge o respectivo *PlaneSensor* que auxilia a parada do balde em conjunto com a trava. Para isso, existem manipuladores *LinearMover*, um sensor plano no final do recuo do pistão referente a trava, bem como um de avanço. No processo de colocar ou selar a tampa utiliza-se um sensor de colisão, além do bloco *LinearMover* responsável pelo movimento. Após o término do último processo, o recipiente se move até atingir o *PlaneSensor* localizado no final da esteira.

V.2 Lógica do processo:

O código que controla o processo foi elaborado na linguagem texto estruturado (ST), no programa B&R Automation Studio, sendo considerado o hardware PC para efeito de simulação. Dessa forma, desenvolveu-se uma máquina de estado contendo todas as ações que devem ser realizadas durante a simulação do processo da envasadora de doces.

Como complemento a lógica do B&R, no RobotStudio, foi criado um sinal "Liga" no controlador, responsável pelo início do processo. Para que a simulação aguarde o sinal "Liga" a ser ativado, foi adicionado no código RAPID uma ação do tipo "*WaitDI*".

V.2.1 Lógica da máquina de estado:

Para que o programa se inicie, é preciso ativar o sinal "Liga", e aguardar a ação do robô. Com o balde posicionado no sensor de presença na frente do cilindro geminado, ativa-se um temporizador para esperar o robô voltar para a posição inicial. Quando o robô completar essa ação, o cilindro geminado inicia o primeiro movimento da haste, e ao entrar em contato com balde, pelo sensor de colisão, o objeto também começa o deslocamento em direção ao silo. Ao atingir o sensor de presença de baixo do silo, um temporizador representando o envase de doces se inicializa. Simultaneamente, inicia-se o movimento de recuo do cilindro geminado que se encerra quando esse atinge o sensor no interior do atuador pneumático. Após a finalização do envase dos doces, o cilindro geminado avança novamente, e ao colidir com o balde, o recipiente move em direção a esteira. Quando o balde atinge o sensor do início da esteira, a haste do geminado recua para a posição inicial.

Todo esse movimento da haste que o cilindro geminado realiza é feito através de uma lógica utilizando a visibilidade do objeto, por meio de sinais que ativam os blocos *Hide* e Show do RobotStudio. Para isso, na aplicação existem sensores planos que identificam a posição da haste original, alternando a visibilidade das auxiliares a fim de criar um pistão contínuo.

O balde se desloca até atingir o sensor de posição referente à área em que ocorre a ação de tampar o balde, onde é travado pelo acionamento do pistão de trava, que avança até alcançar o sensor fim de curso. Com esse sensor ativo, inicia-se o deslocamento vertical do pistão até que haja uma colisão com o balde, ou seja, que a tampa foi posicionada. Logo em seguida, ocorre o retorno do pistão vertical, e ao atingir a posição inicial acontece a liberação do balde, pelo recuo da trava que é cessado quando atinge o sensor no interior do cilindro.

O recipiente então se movimenta até a etapa de selagem, em que a lógica é similar ao processo de tampar o balde. Dessa forma, aciona-se a trava e então um pistão vertical se desloca até encontrar o balde, sendo em seguida recuado, e a trava finalmente liberada. Por fim, o balde se desloca até o sensor de posição localizado no final da esteira, onde existe uma canaleta para a retirada da peça.

VII. INTEGRAÇÃO E COMUNICAÇÃO

De modo a integrar a lógica CLP feita no B&R com a lógica do RobotStudio, criou-se uma comunicação entre os dois programas através do protocolo OPC-UA.

VI.1 B&R Automation Studio:

A fim de permitir a comunicação com o RobotStudio, foi necessário, primeiramente, habilitar o item *Activate OPC-UA System* na seção *OPC-UA System*, como mostrado na Figura 6.

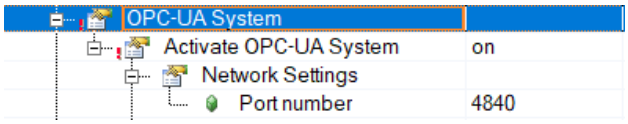


Figura 6: Seção OPC-UA System no B&R
Fonte: Autoral

Em seguida, é preciso ativar as variáveis utilizada na lógica a fim de torná-las visíveis para o protocolo, através do arquivo *OPC UA Default View*, responsável pelo mapeamento das variáveis utilizadas.
Além disso, deve-se configurar em qual dos ciclos de execução o programa criado tem que rodar. Dessa forma, no presente projeto, o ciclo utilizado foi de 100ms para que a simulação ocorresse de maneira correta. Finalmente deve-se fazer a transferência de toda configuração no B&R Automation, através da opção *Trasnfe*r.

VI.1 RobotStudio:

Após a preparação do B&R, deve-se realizar os devidos ajustes no RobotStudio. Inicialmente, insere-se na *Station Logic* um bloco *OpcUa Client*, em que é possível fazer a configuração da comunicação. Na janela de propriedades é possível verificar se a porta corresponde com a utilizada no B&R, sendo que na simulação em questão a porta utilizada foi a padrão 4840, como foi mostrado na Figura 6. Em seguida deve-se realizar a conexão com o B&R, no menu de contexto do bloco.
Ademais, é necessário incluir as entradas e saídas do processo no bloco adicionado. Tal ação é realizada a partir do menu contextual na opção *Configure*, em que as variáveis se encontram na pasta *Program*, que devem ser deslocadas para as seções *Input Signal* e *Output Signal*, como mostrado na Figura 7.

Após a inserção das variáveis, deve-se conectar os demais blocos de acordo com a lógica do programa elaborado, sendo que para o *Digital Twin* criado as ligações ficaram da seguinte forma apresentada a seguir na Figura 8.
Para um maior detalhamento de como configurar a comunicação entre os softwares, ver Apêndice C.

III. RESULTADOS

Após a conclusão do projeto e a simulação do mesmo, nota-se que é possível a elaboração de um *Digital Twin* utilizando o software da ABB RobotStudio, como mostrado na Figura 9. Este desenvolvimento é viável uma vez que, como apresentado, esse programa tem os devidos recursos de criação de uma maquete digital. Dentre esses

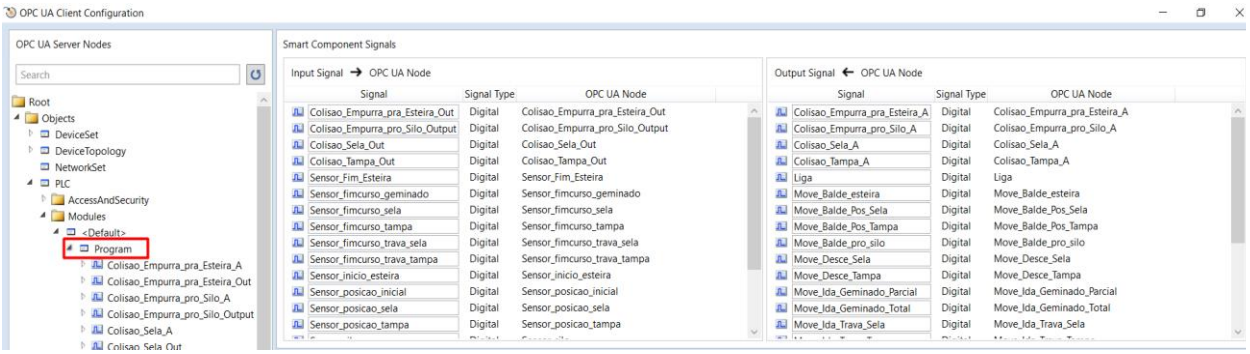


Figura 7: Configuração *OPC-UA Client*
Fonte: Autoral

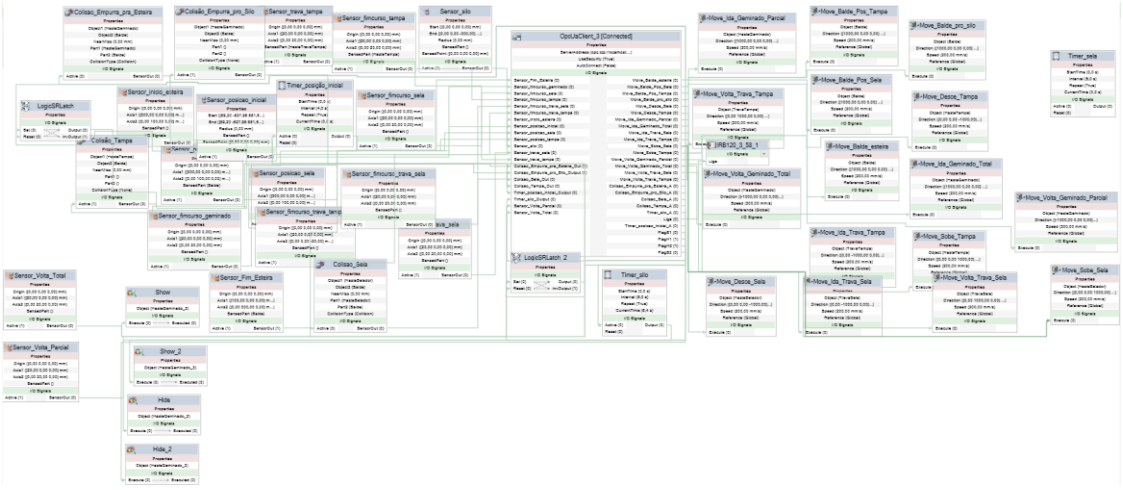


Figura 8: *Station Logic* do RobotStudio
Fonte: Autoral

se pode citar a capacidade de realizar movimento em arquivos do tipo CAD, criar uma lógica através de sensores e os demais blocos citados, além de permitir a comunicação desta lógica com um CLP físico ou virtual.
Como o RobotStudio se trata de um programador off-line, é possível realizar toda a elaboração da lógica e posteriormente a transmitir para o sistema real. Dessa maneira, o *Digital Twin* criado acompanha o processo físico real garantindo os benefícios propostos por esta tecnologia.
Ademais, acrescenta-se que para o presente trabalho, utilizou-se o software B&R Automation Studio, para criação da lógica. Entretanto, outros softwares capazes de programar CLP podem ser usados, com a condição de serem usados como *OPC-UA Server*. Um dos motivos de utilização do software B&R Automation Studio foi a facilidade de encontrar tutoriais de comunicação com o RobotStudio, já que ambos os programas são do fabricante ABB.
Além disso, foi observado empiricamente que quanto menor o ciclo de execução da lógica utilizado no processo, mais preciso a simulação se tornava, evitando problemas de atraso nas ativações dos sinais. Considerando que no presente projeto o CLP foi simulado, tem-se que com o uso de CLP's reais, em que o tempo do ciclo são menores que os de simulações, a aplicação se tornaria mais eficaz.
Finalmente, tem-se que o projeto criado representa um *Digital Twin* e que a ferramenta escolhida, RobotStudio, é eficaz para esse desenvolvimento, atendendo as características de um Gêmeo Digital, como por exemplo a validação de uma lógica CLP. A Figura 10 mostra um esquemático da interligação entre os diversos softwares e hardwares que permite que essa validação seja possível.

IX. Conclusão

Portanto, este projeto se propôs a realização de um *Digital Twin*, criando uma representação de um processo real. Sendo assim, com base nas instruções presentes neste artigo torna-se possível a criação do gêmeo digital em questão. Além disso, a partir da simulação nota-se que a lógica do CLP elaborada no B&R é capaz de controlar a maquete desenvolvida no RobotStudio utilizando o protocolo OPC-UA. Ademais, pode-se inferir que o RobotStudio permite a construção de lógicas para controle de um robô, além de criação de movimentos e ações dentro de uma simulação sem a necessidade de um ambiente físico.

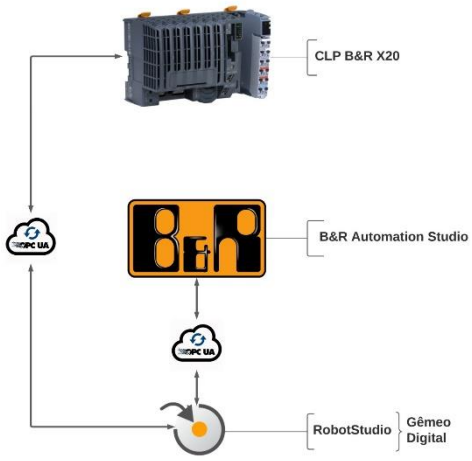


Figura 10: Interligação entre softwares e hardwares
Fonte: Autoral

Dessa forma, em casos de lógicas menos complexas seria possível o desenvolvimento de uma aplicação utilizando apenas este software.
Enfatize-se também, a facilidade em programar o robô, mesmo sem conhecimento da linguagem RAPID, uma vez que é possível fazer essa configuração de outras maneiras, e posteriormente realizar a transferência para o código em RAPID de forma automática.
Além disso, destaca-se a capacidade do RobotStudio de criar novas ferramentas com a possibilidade de adição de mecanismos a estas, como foi feito com a garra no presente trabalho. Dessa maneira, garante-se uma maior flexibilidade de criação das maquetes digitais, o que torna o RobotStudio um software muito poderoso, sendo possível até mesmo criação de novos robôs customizados.

VIII. Trabalhos Futuros

De forma a complementar o projeto desenvolvido sugere-se algumas otimizações na aplicação. Dessa maneira, a partir do *Digital Twin* criado no RobotStudio, outras lógicas CLP em outros programas que utilizam protocolo OPC-UA poderiam ser elaboradas, por exemplo como atividades de laboratório.
Paralelamente, tem-se que o posicionamento do balde no silo e no início da esteira também poderia ser feito pelo robô. Além disso, um novo robô seria responsável pela retirada do recipiente da esteira no final do processo.

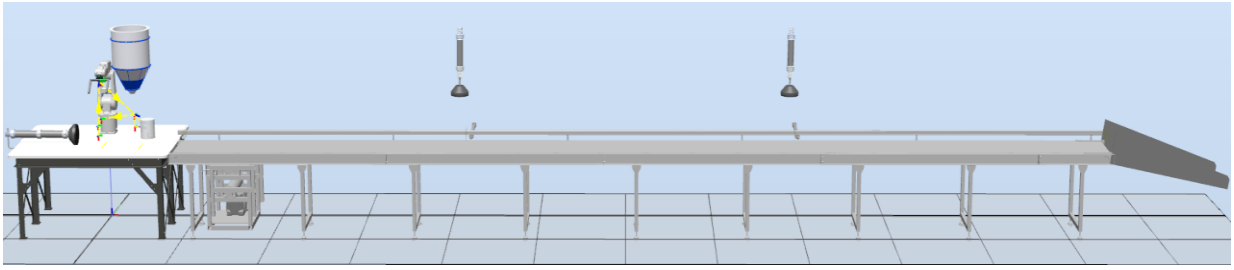


Figura 9: Gêmeo Digital
Fonte: Autoral

Ainda em relação ao robô para que a operação seja mais segura em relação aos operadores, pode-se usar um robô colaborativo da ABB. Dessa maneira, evita-se colisões entre pessoas e novos objetos, sendo possível simular essa funcionalidade utilizando manequins presentes nas bibliotecas do RobotStudio.

IX. AGRADECIMENTOS

Gostaríamos de agradecer este projeto aos nossos familiares que nos apoiaram durante a jornada da graduação. Agradecemos também à Deus e aos nossos velhos e novos amigos por nos acompanharem ao longo desse processo. Por fim, agradecemos ao nosso orientador, Kleber Santos, que nos auxiliou durante todo esse trajeto e sempre estava disposta a ajudar.

X. REFERÊNCIAS

[1] FIRJAN SENAI. **Indústria 4.0**. Disponível em: https://firjansenai.com.br/cursorio/a-firjan-senai/industria-40. Acesso em: 14 jun. 2022.

[2] Saqlain, M., Piao, M., Shim, Y., & Lee, J. Y. Framework of an IoT-based Industrial Data Management for Smart Manufacturing.

[3] PEREIRA, Adriano; DE OLIVEIRA SIMONETTO, Eugênio. Indústria 4.0: conceitos e perspectivas para o Brasil. **Revista da Universidade Vale do Rio Verde**, v. 16, n. 1, 2018.

[4] MUSSOMELI, Adam. **Expecting digital twins**: Adoption of these versatile avatars is spreading across industries. Deloitte Insigths. 8 p. Disponível em: https://www2.deloitte.com/content/dam/insights/us/articles/3773_Expecting-digital-twins/DI_Expecting-digital-twins.pdf. Acesso em: 14 jun. 2022.

[5] HOFFMANN, P. et al. **Virtual Commissioning Of Manufacturing Systems** - A Review And New Approaches For Simplification. In: 24th European Conference on Modelling and Simulation. [S.l.: s.n.], 2010. p. 175 – 181.

[6] TOTVS, Equipe. Digital Twin: saiba como aplicar e principais benefícios. ln: TOTVS. TOTVS. [S.l.]. 30 ago. 2021. Disponível em: https://www.totvs.com/blog/inovacoes/digital-twin/#:~:text=O%20digital%20twin%20permite%2C%20inclusive,a%20demanda%20bem%20mais%20precisa.. Acesso em: 1 jul. 2022.

[7] PROJETO Final do Curso Técnico em Automação Industrial: Apresentação do Processo Rodando (4/4). Gian Fachini. Disponível em: https://www.youtube.com/watch?v=NKhOzAqWjzo. Acesso em: 6 out. 2021.

[8] ABB ROBOTICS. **Operation Manual**: RobotStudio. 4.0.378 ed. Sweden, 2013. 548 p. Disponível em: https://www.inf.uszeged.hu/~groszt/teach/Robotika/Segedanyag/3HAC032104-en.pdf. Acesso em: 14 jun. 2022.

[9] ABB ROBOTICS. **RobotStudio®**: The world's most used offline programming tool for robotics. ABB. Disponível em: https://new.abb.com/products/robotics/robotstudio Acesso em: 14 jun. 2022.

[10] B&R. **Automation Studio 4.** Disponível em: https://www.br-automation.com/ptbr/produtos/software/. Acesso em: 14 jun. 2022.

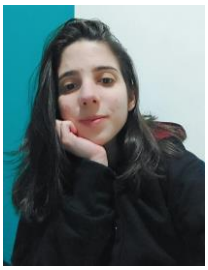
[11] GRIEVES, Michael. **Digital Twin**: Mitigating Unpredictable, Undesirable Emergent Behavior in Complex Systems (Excerpt). Disponível em: https://www.researchgate.net/profile/Michael-Grieves/publication/307509727_Origins_of_the_Digital_Twin_Concept/links/57c6f44008ae9d64047e92b4/Origins-of-the-Digital-Twin-Concept.pdf. Acesso em: 14 jun. 2022.

[12] PARROTT, Aaron; WARSHAW, Lane. **Industry 4.0 and the digital twin:** Manufacturing meets its match. Deloitte Insights. Disponível em: https://www2.deloitte.com/us/en/insights/focus/industry-4-0/digital-twin-technology-smart-factory.html. Acesso em: 14 jun. 2022.

[13] GRABCAD Community. Disponível em: https://grabcad.com/library. Acesso em: 10 mar. 2022.

BIOGRAFIA:

Gustavo Amaral Leal de Souza
Nasceu em Sorocaba (SP), em 1999. Concluiu o ensino médio em Brazópolis (MG) no ano de 2016. Ingressou na UNIFEI (Campus Itajubá) em 2017, no curso de Engenharia Hídrica, sendo que em 2018 transferiu para Engenharia de Controle e Automação, na mesma universidade. No primeiro semestre de 2022, iniciou o estágio na empresa Braskem, no núcleo de pesquisa de químicos renováveis, auxiliando a equipe do laboratório, em tempo e recursos, através de soluções computacionais, majoritariamente feitas em Python.



Rebeca Ribeiro Pinto
Nasceu em Volta Redonda (RJ), em 1998. Concluiu o ensino médio em Itajubá (MG) no ano de 2015. Ingressou na UNIFEI (Campus Itajubá) em 2017, no curso de Engenharia Hídrica, sendo que em 2018 transferiu para Engenharia de Controle e Automação. No ano de 2019, participou do projeto social fundação ASIMO na área de desenvolvimento de projetos, além da coleta e pesquisa sobre novos destinos para o lixo eletrônico. No primeiro semestre de 2021, atuou no Projeto Semestral UNIFEI em parceria com a ABB, e no semestre seguinte iniciou o estágio na empresa Automalógica na área de Engenharia/SCADA.

APÊNDICE A – TUTORIAL DA CRIAÇÃO DA GARRA

Inicialmente, foi utilizado um componente chamado *IsoPlate Small Bizel*, presente na biblioteca de equipamentos RobotStudio, além de três formas geométricas de modelagem do tipo *Solid Box* nativo do software. Sendo assim, um dos sólidos foi posicionado no *IsoPlate* representando a base da garra. Esses dois componentes foram inseridos em um novo *Component Group*, criado a partir da aba *Modelling* e renomeado para *GripperBase*. Os demais sólidos foram renomeados para *Left Finger* e *Right Finger*, e posicionados paralelamente entre si, a fim de fazer o movimento mecânico do efetuador. Essas formas geométricas foram dimensionadas de acordo com o tamanho do balde a ser criado. Ademais, após a configuração do manipulador, fez-se o acoplamento da garra ao robô, através do reposicionamento do equipamento.

Após a criação física da garra, fez-se necessária a configuração do mecanismo para a realização da captura do balde. Para tal, deve-se acessar a aba *Modelling* e na seção *Mechanism*, fazer a criação de um mecanismo (em, *Create Mechanism*). Na janela exibida, foi alterado o *Mechanism Model Name* para *CustomGripper* e o *Mechanism Type* foi definido como *Tool*.

Em seguida, a partir do modelo de mecanismo criado, *CustomGripper*, deve-se configurar os *links*, acessando-os através de um duplo clique, como mostrado a seguir na Figura 11:

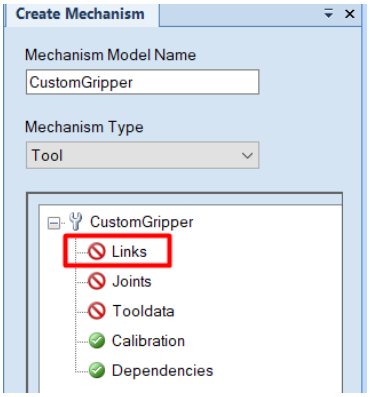


Figura 11: Configuração do mecanismo *CustomGripper*
Fonte: Autoral

O primeiro *link*, L1, representa a base do efetuador, portanto, deve-se defini-lo como *BaseLink*, marcando a opção *Set as BaseLink*. Deve-se selecionar o *GripperBase* na seção *Selected Component*, e em seguida, transferir para a coluna *Added Components*, de acordo com a Figura 12.

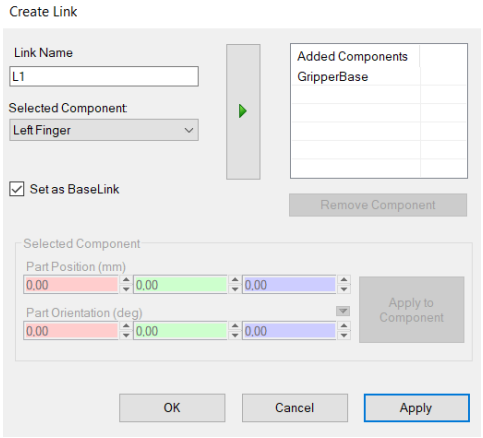


Figura 12: Configuração do *link* L1
Fonte: Autoral

Da mesma forma, para os próximos *links*, L2 e L3, deve-se selecionar os componentes restantes da garra (*Left Finger* e *Rigth Finger*) na opção *Selected Components*, porém diferente do L1, não são marcados como *BaseLink*.

Com os três *links* configurados, deve-se estabelecer as juntas da garra, acessando agora a opção *Joint* com um duplo clique. Dessa forma, como o movimento de ambas juntas do manipulador é linear, deve-se definir a propriedade *Joint Type* para prismática na janela exibida. Ademais, para a primeira junta criada (J1), a propriedade *Child Link* deve ser o L2, e o parâmetro *Second Position*, no eixo Y, precisa ser alterado para -30. Além disso, os limites máximo e mínimo necessitam ser alterados de acordo com o tamanho da base, de maneira que o valor máximo, represente o extremo lateral do *GripperBase*, sendo possível fazer o teste pelo comando *Jog Axis*. Essa configuração é apresentada na Figura 13.

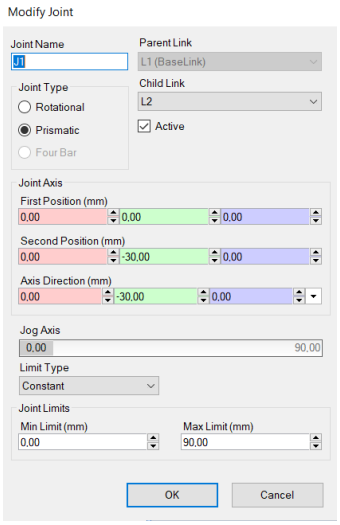


Figura 13: Configuração da junta J1
Fonte: Autoral

Em relação à segunda junta (J2), como se trata de uma ação espelhada, o valor do *Second Position* no eixo Y, deve ser 30, mantendo os mesmos limites máximo e mínimos. Enfatiza-se que o *Parent Link* de ambos os dedos têm que ser definido como o L1 (*BaseLink*), e no caso de J2, o *Child Link* é o L3.

Após a criação das juntas, acessa-se a opção *Tooldata*, em que se configura os parâmetros de criação dessa ferramenta, como a propriedade *Belongs to Link* que deve ser alterada para o *BaseLink*. Além disso, em relação as características referentes a massa e centro de gravidade, é necessário definir os valores de acordo com o processo em questão, como mostrado na Figura 14.

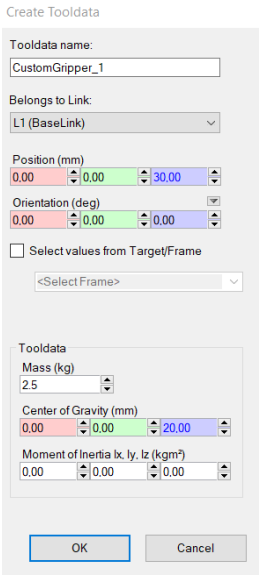


Figura 14: Configuração do *CustomGripper_1*
Fonte: Autoral

Ao finalizar essas configurações, é necessário compilar o mecanismo criado, como mostrado na Figura 15, abaixo.

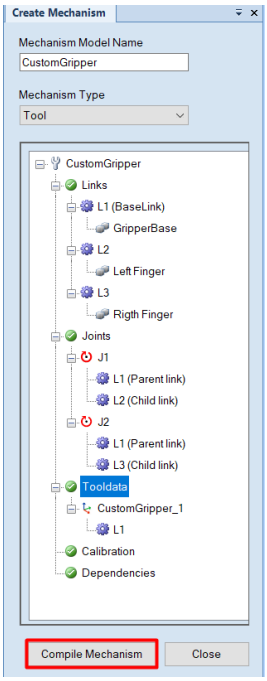


Figura 15: Configuração do CustomGripper_1
Fonte: Autoral

Em seguida, no campo *Poses* na parte inferior, deve-se acessar as características do *SyncPose*. Esse parâmetro representa a posição da garra fechada, ou seja, enquanto segura o objeto. Dessa forma, como ambas as garras têm limite máximo de 90, definiu-se os *Joint Value* como 80 a fim de se ajustar ao tamanho do balde, como mostrado na Figura 16.

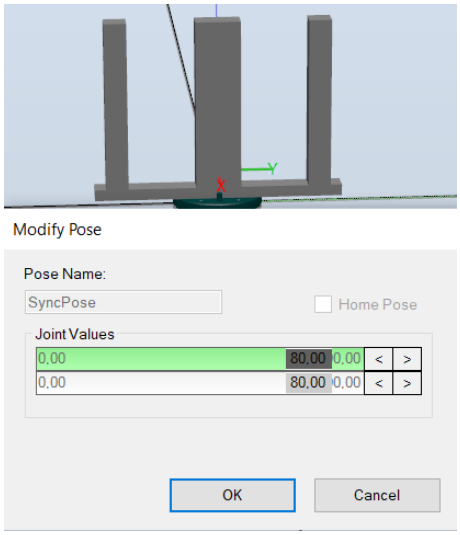


Figura 16: Configuração do *SyncPose*
Fonte: Autoral

Para representar a posição da garra totalmente aberta, é necessário criar uma *pose*, e defini-la como *HomePose*, estabelecendo os *Joint Values* para os valores máximos pré-definidos anteriormente. Dessa forma, o mecanismo da garra foi finalizado, e ela pode ser acoplada ao robô.

Com a intenção de criar os sinais para comandar os movimentos do mecanismo criado, é necessário acessar a aba *Controller* e em sequência, na opção *Configuration* selecionar *IO System*. Na tela aberta, cria-se um novo sinal, a

partir do botão direito, no item *Signal* na coluna *Type*. Como o sinal se trata de uma saída no controlador em questão, deve-se configurar da seguinte forma, mostrada na Figura 17.

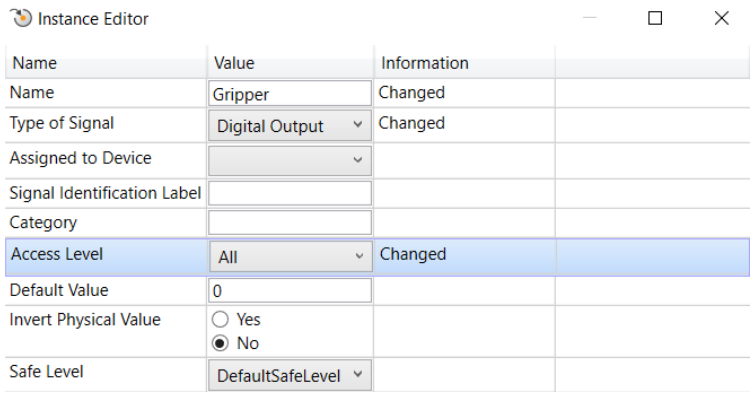


Figura 17: Criação do sinal *Gripper*
Fonte: Autoral

Esse sinal é responsável por controlar os eventos de abertura e fechamento da garra. Dessa forma, deve-se criar os eventos em *Event Manager*, na seção *Configure* da aba *Simulation*. Um evento é adicionado clicando no botão *Add*. Na primeira tela, para ambos os eventos, o *Activation* e *Event Trig Type* devem permanecer no modo padrão. Na janela seguinte, deve-se selecionar o sinal criado, *Gripper*, e para o evento de fechamento da garra, a opção *Trigger Condition* precisa estar em *Signal is true (1)*. Na próxima etapa, deve-se escolher *Move Mechanism to Pose* como *Set Action Type*. Na última janela, define-se como mecanismo o *CustomGripper* e como *Pose* o *SyncPose*. Para finalizar o fechamento da garra, é preciso atribuir uma ação para o evento criado. Dessa forma, acessa-se a seção *Action* e faz a adição de uma ação *Attach Object* através do botão *Add Action*. Em seguida, seleciona-se a opção *Keep P*osition, altera a propriedade *Attach Objec*t para <*Find Closest Object*> e finalmente define *CustomGripper* para *Attach t*o.

De forma análoga, cria-se o evento de abertura da garra a partir do sinal *Gripper*, entretanto o *Trigger Condition* deve ser *Signal is false (0)*. Novamente, tem-se que definir *Move Mechanism to Pose* como *Set Action Type*, o mecanismo como *CustomGripper*, porém a *Pose* deve ser *HomePose*. Selecionando o último evento criado, adiciona-se uma ação *Dettach Object*. Na tela aberta, a propriedade *Dettach Object* deve ser alterada para <*Any Object*>, assim como a *Dettach From* para *CustomGripper*. Na Figura 18 pode-se ver os eventos criados.

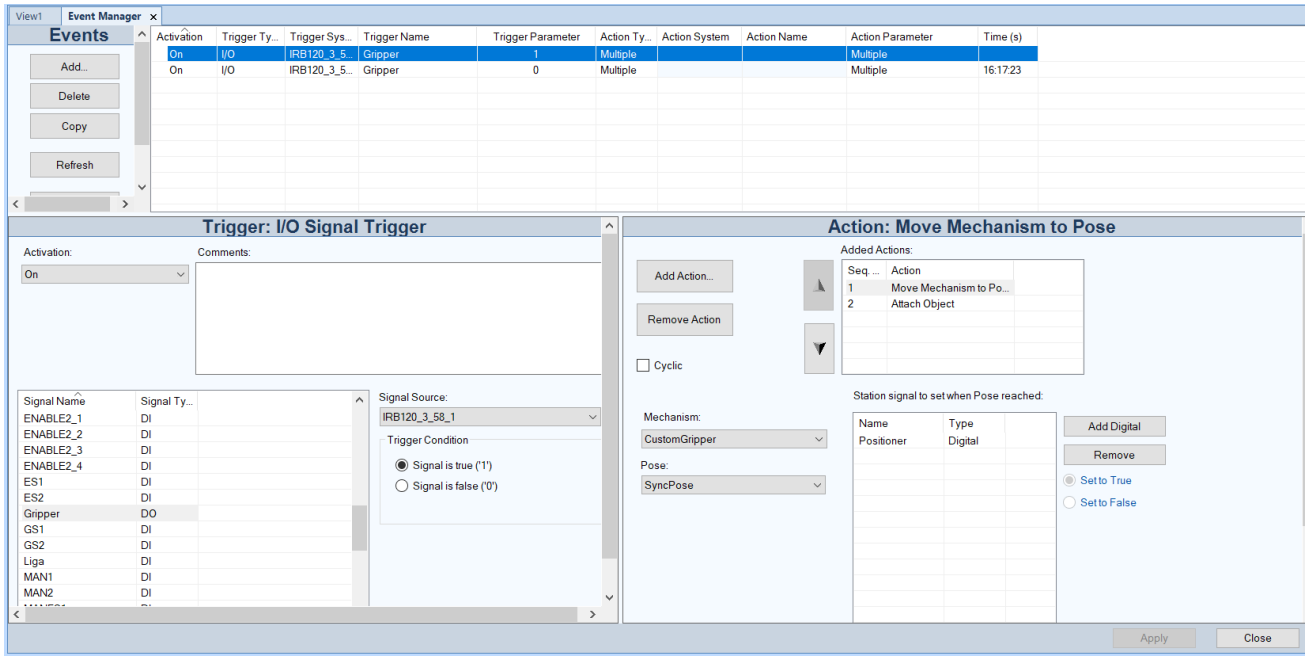


Figura 18: Criação do evento de abertura e fechamento da garra
Fonte: Autoral

APÊNDICE B – TUTORIAL DA CRIAÇÃO DA TRAJETÓRIA DO ROBÔ

Primeiramente, deve-se criar uma posição inicial do robô (*Home*) a partir da aba Home no ícone *Teach Target*. Dessa forma, um *target* é criado na aba *Paths and Targets* e foi renomeado para *HomePosition*, como mostrado na Figura 19 a seguir:

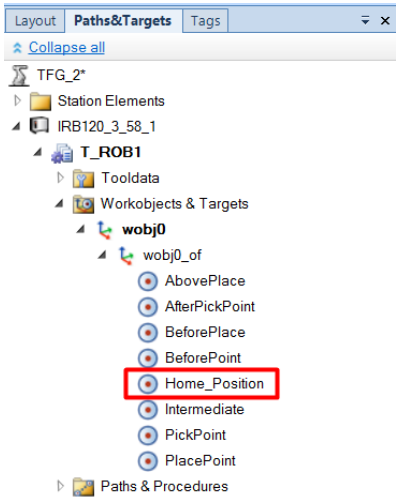


Figura 19: Criação do *Target Home_Position*
Fonte: Autoral

Em seguida, para que o robô encontre a localização do balde, deve-se criar um *target*, chamado *PickPoint*, a partir da aba *Home*, no botão *Target* opção *Create Target*. Posteriormente, precisa-se definir a posição deste ponto criado, ou seja, a posição em que o robô pega o balde. Para isso, aciona-se o recurso *Snap Position* na parte superior da tela *View*, depois, na janela de posição do *target*, clica-se em um dos eixos e em seguida na parte superior do balde. Após a criação, a partir do menu contextual do *target*, seleciona-se a opção *View tool at target* e escolha o *CustomGripper*, de modo a visualizar como seria a realizado a ação de pegar o balde. A partir da função *Set Position* no menu contextual é possível ajustar através de offset e rotações a orientação e posição da garra, a fim de capturar o balde da maneira mais adequada para o processo.

Definido os pontos que o robô precisa passar, foi criado o caminho da movimentação do robô. Dessa forma, através do menu *Home*, na opção *Path*, um *Empty Path* deve ser criado. Esse caminho pode ser encontrado na pasta *Path and Procedures* também mostrado na Figura 17.

Para a segurança da aplicação, é necessário adicionar uma instrução de ação do tipo *Reset* no início do processo, a partir da opção *Insert Action Instruction* no botão direito do caminho criado. Na janela aberta, altera-se a propriedade *Instruction Templates* para *Reset*. Posteriormente, copia-se os *targets* para o caminho, arrastando-os para dentro do *path* criado, atentando-se para a ordem da movimentação. Ademais, alguns parâmetros do robô devem ser configurados, a partir da opção *Modify Instruction* do menu contextual. Entre esses, pode-se citar a velocidade do robô e o parâmetro *Zone*. No processo da envasadora em questão, foi escolhido uma velocidade de 200 mm/s, e uma zona *fine* para evitar a formação de cantos e garantir que o robô alcance o *target* criado com a maior precisão possível. É importante ressaltar também que, caso movimento seja complexo ou não linear deve-se alterar o *Motion Type* na opção *Edit Instruction* para movimento de junta (*MoveJ*).

Após a instrução de movimentação do *PickPoint*, ou seja, posição em que o robô pega o objeto, é necessário adicionar mais duas instruções de ação. A primeira deve ser um pequeno intervalo de tempo antes do fechamento da garra, sendo assim, o *Instruction Templates* é configurado como *WaitTime*. Na opção *Instruction Arguments* o valor *\InPos* tem que ser *Enabled* e no *Time* inserido o valor do intervalo de tempo, nesse caso 1 (um segundo).

Em seguida, a segunda ação, que se trata do acionamento da garra, deve ser adicionada, e configurando como *Set* a propriedade *Instruction Templates*. É necessário enfatizar que depois do robô ter chegado na posição de soltar o objeto, uma nova ação *Reset* deve ser inserida.

Finalmente, depois dos caminhos e pontos criados, a fim de executar a simulação do robô é preciso, na aba *RAPID*, realizar através da sincronização, a transferências dos caminhos e *targets* da estação para o código *RAPID*.

Adverte-se que, antes de realizar a simulação, como o balde não tem um mecanismo de retorno para a posição inicial, aconselha-se salvar a posição inicial como um estado passível de reset, através da aba *Simulation*, no botão *Reset* e na opção *Save Current State*, incluindo todo o projeto. Acrescenta-se também, a necessidade da adição de um *WaitTime* através do *Path and Procedures*, com o valor do tempo necessário para que a simulação termine.

APÊNDICE C – TUTORIAL DA INTEGRAÇÃO ENTRE OS SOFTWARES

Inicialmente, para permitir a comunicação com o RobotStudio, deve-se, no B&R Automation Studio, habilitar o item *Activate OPC-UA System* na seção *OPC-UA System*, como mostrado na Figura 6. Essa opção pode ser encontrada na configuração do hardware utilizado, a partir da *Physical View*.

Posteriormente, no mesmo software, para ativar as variáveis utilizada na lógica, é necessário acessar a aba *Configuration View*, na pasta *Connectivity*, e na subpasta OpcUA, e então inserir através do toolbox, o arquivo "*OPC UA Default View*". Editando esse arquivo, ao selecionar a variável é possível habilitá-la clicando no ícone verde mostrado na Figura 20.

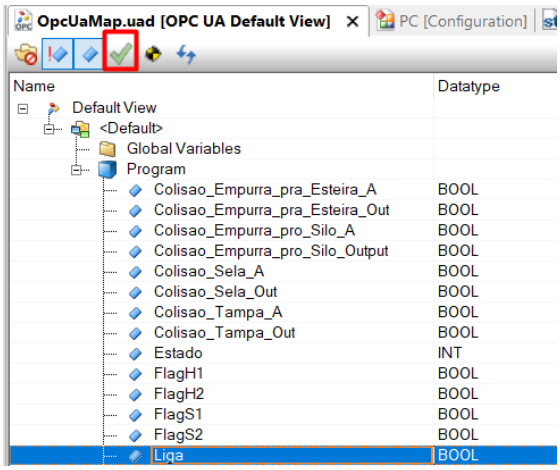


Figura 20: Mapeamento das variáveis
Fonte: Autoral

Além disso, na *Physical View* deste programa, com botão direito no hardware adicionado, acessa-se a opção Software. Nessa janela, deve-se configurar em qual dos ciclos de execução o programa criado tem que rodar. Para fazer a transferência de toda configuração do B&R, deve-se clicar no ícone *Trasnfe*r (Ctrl+F5).

Em seguida, no RobotStudio, deve-se inserir um bloco *OpcUa Client* na janela *Station Logic*, e a partir do menu contextual é possível fazer a configuração da comunicação. Na janela de propriedades é possível verificar se a porta corresponde com a utilizada no B&R, como mostrado na Figura 6. Posteriormente, deve-se realizar a conexão com o B&R, no menu de contexto do bloco.

Ademais, para incluir as entradas e saídas do processo no bloco adicionado deve-se acessar o menu contextual do bloco na opção *Configure*, em que as variáveis se encontram na pasta *Program*, que devem ser deslocadas para as seções *Input Signal* e *Output Signal*, como mostrado na Figura 7.

Como mencionado, após a inserção das variáveis, deve-se conectar os demais blocos de acordo com a lógica do programa elaborado, sendo que para o *Digital Twin* criado as ligações ficaram da forma apresentada na Figura 8.