Universidade de Brasília
Faculdade de Tecnologia
Departamento de Engenharia Mecânica

Desenvolvimento de um gêmeo digital para simulação, monitoramento e armazenamento de dados de processo em uma célula robótica de manufatura aditiva por deposição de metal a arco e arame

João Vítor Arantes Cabral

PROJETO FINAL DE CURSO
ENGENHARIA DE CONTROLE E AUTOMAÇÃO

Brasília
2023

Universidade de Brasília
Faculdade de Tecnologia
Departamento de Engenharia Mecânica

Desenvolvimento de um gêmeo digital para simulação, monitoramento e armazenamento de dados de processo em uma célula robótica de manufatura aditiva por deposição de metal a arco e arame

João Vítor Arantes Cabral

Projeto Final de Curso submetido como requisito parcial para obtenção do grau de Engenheiro de Controle e Automação

Orientador: Prof. Dr. Guilherme Caribé de Carvalho
Coorientador: Prof. Dr. Alberto José Álvares

Brasília
2023

Cabral, João Vítor Arantes.
C117d Desenvolvimento de um gêmeo digital para simulação, monitoramento e armazenamento de dados de processo em uma célula robótica de manufatura aditiva por deposição de metal a arco e arame / João Vítor Arantes Cabral; orientador Guilherme Caribé de Carvalho; coorientador Alberto José Álvares. -- Brasília, 2023.
87 p.

Projeto Final de Curso (Engenharia de Controle e Automação) -- Universidade de Brasília, 2023.

1. Manufatura aditiva. 2. Gêmeos Digitais. 3. Internet das Coisas. 4. Células de Manufatura Robotizadas. I. Cabral, João Vítor Arantes II. de Carvalho, Guilherme Caribé, orient. III. Álvares, Alberto José, coorient. IV. Título

Universidade de Brasília
Faculdade de Tecnologia
Departamento de Engenharia Mecânica

Desenvolvimento de um gêmeo digital para simulação, monitoramento e armazenamento de dados de processo em uma célula robótica de manufatura aditiva por deposição de metal a arco e arame

João Vítor Arantes Cabral

Projeto Final de Curso submetido como requisito parcial para obtenção do grau de Engenheiro de Controle e Automação

Trabalho aprovado. Brasília, 21 de dezembro de 2023:

Prof. Dr. Guilherme Caribé de Carvalho, UnB/FT/ENM
Orientador

Prof. Dr. José Maurício Santos Torres da Motta, UnB/FT/ENM
Examinador interno

Prof. Dr. Carlos Humberto Llanos Quintero, UnB/FT/ENM
Examinador interno

Brasília
2023

Agradecimentos

Primeiramente agradeço a Deus por ter me dado o dom da vida e da inteligência, agradeço também aos meus pais por sempre terem me apoiado a vida inteira me dando a base para me tornar o homem que sou hoje e por terem me possibilitado a realização do curso de graduação em Engenharia Mecatrônica.

Sou grato também a todos os meus colegas, amigos e família que me apoiaram ao longo da jornada do curso de graduação; em especial à minha namorada por sempre ter me dado suporte e me encorajar a ser cada dia melhor e à minha cunhada por me ajudar a realizar meu sonho de fazer intercâmbio.

Agradeço aos meus orientadores, prof. Guilherme Caribé de Carvalho e prof. Alberto José Álvares por terem me guiado ao longo deste trabalho, me dando a oportunidade de aplicar os conhecimentos adquiridos ao longo do curso em um projeto tão inovador. Agradeço por fim à Universidade de Brasília, por todas as experiências, conhecimento e crescimento acadêmico e profissional que venho adquirindo nos últimos anos e pelas portas que se abriram para mim graças à universidade.

Resumo

A manufatura aditiva vem em constante crescimento e este tema está recebendo cada vez mais atenção de indústrias e do mundo acadêmico. Diferentemente da manufatura tradicional, a subtrativa, busca-se através da construção por camadas a redução do tempo de produção de peças complexas, o baixo custo e otimização do consumo de material, evitando desperdícios. Este trabalho apresenta um tipo de manufatura aditiva, a deposição de material metálico a arco, que será implementada por meio de uma fonte de soldagem e um robô ABB IBR 2600. A fim de criar um Gêmeo Digital *Digital Twin* de uma célula de manufatura aditiva de metal, utilizaram-se em um primeiro momento os programas RoboDK e ABB RobotStudio. Tal aplicação coloca em prática conceitos relacionados à Internet das Coisas (IoT) e Internet das Coisas Industriais (IIoT), sendo neste trabalho aplicados para o monitoramento em tempo real de um ativo em laboratório. A normatização base para o desenvolvimento do Digital Twin em questão é a ISO 23247, que estabelece diretivas para guiar a criação de sistemas de monitoramento e controle relacionados a serviços inerentes a aplicações recentes da Indústria 4.0. Através dessa norma, é possível determinar o interfaceamento entre domínios que fornecem serviços e aplicações ao usuário e também a outras camadas/domínios subsequentes. Na troca de dados entre cada domínio há a utilização de protocolos de comunicação que são comuns em aplicações IoT como MQTT, protocolos web como HTTP e WebGL para simulação de movimentação 3D no navegador além de protocolos de transmissão de dados entre máquinas como o Ethernet, que neste projeto será utilizado para obter dados da célula de manufatura aditiva robotizada. O objetivo deste trabalho, então, é desenvolver aplicações e serviços modulares que conjuntamente descrevem um Gêmeo Digital para uma célula de manufatura aditiva de material metálico composta pelo robô ABB IRB 2600 com fonte de soldagem Fronius MW5000 GTAW e mesa posicionadora ABB IRBP A250. Nesse sentido, foram utilizados os protocolos de comunicação citados (Ethernet, MQTT e HTTP) para desenvolver um Gêmeo Digital (*Digital Twin*) modular para a célula de manufatura, cujos dados obtidos dos ativos físicos serão publicados em nuvem através de um fluxo Node-RED em servidor local com dashboard de visualização de variáveis e simulação 3D de movimentação espelhada do robô ABB IRB 2600 via RoboDK.

Palavras-chave: Manufatura aditiva. Gêmeos Digitais. Internet das Coisas. Células de Manufatura Robotizadas.

Abstract

The additive manufacturing is constantly growing and this theme is receiving increasingly more attention of industries and the academic world. Differently from the traditional manufacturing, the subtractive one, a reduction on the complex parts production time, a reduction on the cost, an optimization of material usage and reduction of waste are sought after through the construction of those parts by layers. This work presents an additive manufacturing type, the arc deposition of metallic material which will be implemented using a welding power supply and an ABB IRB 2600 robot. In order to create a metal additive manufaturing Digital Twin, it was decided to utilize the software RoboDK and ABB RobotStudio. That application puts into practice concepts related to Internet of Things (IoT) and Industrial Internet of things (IIoT), which will be applied in this work for real-time monitoring of an asset in a laboratory. The standardization basis for the development of the Digital Twin in question is the ISO 23247, which establishes directives for guiding the creation of control and monitoring systems related to services inherent to recent applications of the Industry 4.0. Through this standard, it is possible to determine the interface between domains that provide services and applications to the user and also to other subsequent layers/domains. In the exchange of data between each domain there is the use of communication protocols that are common in IoT applications such as MQTT, web protocols such as HTTP and WebGL for simulation of 3D movement in the browser and data transmission protocols between machines such as Ethernet, which in this project will be used to obtain data from the robotic additive manufacturing cell. The objective of this work, then, is to develop modular applications and services that jointly describe a Digital Twin for an additive manufaturing cell for metallic material composed by the ABB IRB 2600 robot with a Fronius MW5000 GTAW welding source and an ABB IRBP A250 positioning table. In this sense, the reported communication protocols (Ethernet, MQTT and HTTP) were used to develop a modular Digital Twin (*Digital Twin*) for the production cell, whose data obtained from the physical assets will be published in the cloud through a Node-RED flow on local server with variable visualization dashboard and 3D simulation of mirrored movement of the ABB IRB 2600 robot via RoboDK.

Keywords: Additive Manufacturing. Digital Twins. Internet of Things. Robotic Manufacturing Cells.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

ABB	ASEA Brown Boveri 17
AM	Additive Manufacturing 17
DCDCE	Entidade de Coleta de Dados e Controle de Dispositivos 19
DF	Distrito Federal 17
DT	Digital Twin(s) 18
FDM	Fused Deposition Modeling 21
GMAW	Gas Metal Arc Welding 23
GRACO	Grupo de Automação e Controle 17
GTAW	Gas Tungsten Arc Welding 23
I/O	Input/Output 41
IHM	Interface homem-máquina 41
IIoT	Industrial Internet of Things 6
IoT	Internet of Things 18
MA	Manufatura Aditiva 17
OME	Elemento de Manufatura Observável 18
RWS	Robot Web Services 38
TCP	Tool Center Point 37
TCP/IP	Transmission Control Protocol - Protocolo de Controle de Transmissão e Internet Protocol - Protocolo de Internet 38
UnB	Universidade de Brasília 17
WAAM	Wire Arc Additive Manufacturing 23
wobjdata	Work Object Data 26

Sumário

1 Introdução

A manufatura aditiva consiste em um processo de fabricação de geometria a partir de um modelo 3D. Há diversos tipos desta tecnologia, entretanto sua ideia principal se mantém a mesma, a qual é a deposição de material em camadas, construindo a geometria do modelo. O processo inicia na construção de um modelo via Software de CAD (Computer-Aided Design), em sequência é feita a etapa de fatiamento do modelo 3D, com auxílio de um software de fatiamento, ao finalizar essa etapa é gerado o arquivo de descrição dos pontos de trajetória a ser percorrida durante a deposição de material. Essa trajetória pode ser programada em diferentes linguagens, como, por exemplo, o código G, que contém todos os comandos mediante coordenadas cartesianas para o entendimento da máquina e construção de cada camada. A manufatura aditiva vem se tornando solução para conquistar uma maneira mais rápida e barata quando comparada com a maneira convencional de produção de peças.

Esta tecnologia se tornou um dos pilares da Indústria 4.0, pois se torna uma opção alternativa para as Indústrias de confecção de peças, impactando no custo com compra de peças para manutenção das máquinas, além de conseguir reduzir o tempo de chegada dessas peças, pois elas serão fabricadas "in house", isso pode gerar grandes retornos como pode ser visto em (WISHBOX, 2022). Com o avanço dessa tecnologia, é possível obter peças em inox, aço carbono, polímeros de alta resistência, aumentando a performance de linhas produtivas e fazendo as indústrias criarem certa independência de seus fornecedores, gerando um sistema de conhecimento interno de projeto e desenvolvimento de peças e máquinas que é autônomo.

Como pode ser visto na figura 1.1, a Indústria 4.0 possui diversos pilares sobre os quais a tecnologia pode ser implementada.

Figura 1.1 – Pilares da Indústria 4.0

Fonte: Adaptado de (PIJUSH KANTI DUTTA PRAMANIK, 2019)

Todos esses pilares são de extrema importância para a implementação de um conceito tecnológico, avançado, dinâmico, digital no chão de fábrica. O uso de internet das coisas vem sendo cada vez mais expandido por ser a tecnologia capaz de reunir e transmitir dados por meio de uma rede que faz a conexão dos objetos físicos. Nesse contexto entra o processamento em "nuvem"que inclui o uso da computação da "nuvem" e, com isso, tem-se o uso de machine learning e inteligência artificial para definição de modelos de inspeção, buscando uma manutenção preditiva no processo produtivo, além do uso dos conceitos de *big data* e suas análises para trazer confiabilidade aos dados coletados e uma tomada de decisão nível gerencial mais assertiva.

No contexto da Indústria 4.0, este trabalho representa uma aplicação prática dos conceitos de *Digital Twin*, Internet das Coisas (IoT) e de células de manufatura aditiva (AM)

robotizadas. No âmbito da criação de *Digital Twins* (DT), diversas tecnologias relacionadas a protocolos de comunicação são utilizadas; além disso, geralmente a implementação de um DT se baseia em normatizações de fornecimento de serviços e aplicações relacionadas ao Gêmeo Digital. Outro aspecto a ser considerado é a movimentação do ativo físico a ser monitorado no Gêmeo Digital (neste caso o robô IRB ABB 2600), a qual deve ser levada em consideração ao ser utilizado para deposição de liga metálica, é possível que não seja obtida a precisão necessária para a produção de peças de qualidade e que possuam boa resistência mecânica e durabilidade. Ademais, nesse sentido, há várias técnicas de deposição de material metálico que podem ser utilizadas a fim de imprimir as peças tridimensionais.

Ao longo do presente trabalho, um projeto de concepção de uma arquitetura *Digital Twin* para uma célula de manufatura aditiva robotizada com base no braço robótico ABB IRB 2600 é apresentado. Tal arquitetura baseia-se na normatização proposta pela norma ISO 23247 (ISO, 2022a,b,c,d), que será retratada mais adiante na fundamentação teórica deste trabalho. Os protocolos de comunicação utilizados para implementar o *Digital Twin* são os protocolos Ethernet TCP/IP, MQTT e HTTP para publicar dados em *cloud*. A estruturação e conceituação do fluxo de dados composto por esses protocolos será explicada nos capítulos a seguir.

1.1 Objetivos

Os objetivos deste trabalho consistem no desenvolvimento de um Gêmeo Digital para a célula de manufatura aditiva de metal composta pelo robô ABB IRB 2600 e mesa posicionadora ABB IRBP A250, localizados no laboratório GRACO da Universidade de Brasília.

O Gêmeo Digital é composto por diversos algoritmos organizados em módulos que devem ser executados concomitantemente para o pleno funcionamento de todas as aplicações de monitoramento baseadas no Gêmeo Digital implementado. Esses módulos são:

- Adaptador MQTT com *broker* e *publisher* MQTT com servidor Ethernet TCP/IP local para coleta de dados do controlador ICR5 do robô ABB IRB 2600;
- Módulo cliente Ethernet TCP/IP embutido no código RAPID interpretado pelo controlador ICR5;
- Fluxo de estruturação de dados coletados executado em servidor local Node-RED (subscrito nos tópicos do *publisher* MQTT) para publicação em "nuvem";
- Algoritmo em Python com *subscribers* MQTT para simulação de movimentação 3D no software RoboDK.

Todos os módulos citados serão explicados no capítulo 4 deste trabalho. Além disso, como algoritmo à parte do fluxo implementado, desenvolveu-se um algoritmo em Python capaz de inserir o módulo cliente de comunicação com o servidor Ethernet ativo no adaptador MQTT em códigos RAPID, possibilitando habilitar a comunicação *socket* em algoritmos gerados para a célula de manufatura aditiva em questão.

1.2 Estruturação do trabalho

O presente trabalho está dividido em 6 capítulos, descritos a seguir.

- Introdução (1): apresentação inicial do trabalho situando-o no contexto atual da manufatura aditiva. Apresentação de objetivos e estruturação geral do trabalho;
- Fundamentação teórica (2): capítulo para conceituação teórica acerca de protocolos de comunicação e manufatura aditiva por deposição de metal;
- Trabalhos correlatos (3): capítulo onde trabalhos relevantes com temática similar ao do presente trabalho serão comentados e eventualmente comparados com os resultados esperados neste;
- Metodologia (4): neste capítulo a arquitetura baseada na ISO 23247 projetada para o Gêmeo Digital da célula de manufatura aditiva para deposição de metal será apresentada. A metodologia para implementação de cada um dos módulos necessários para o Gêmeo Digital também será discutida;
- Resultados (5): capítulo onde os resultados alcançados na implementação do fluxo de dados para o Gêmeo Digital são apresentados destacando cada módulo implementado;
- Conclusão (6): no último capítulo comenta-se por fim os resultados alcançados, além de propor possíveis aprimoramentos passíveis de serem implementados futuramente no projeto.

2 Fundamentação teórica

A fim de se conceituar a arquitetura de um Gêmeo Digital, faz-se necessário primeiro explicitar conceitos fundamentais referentes às tecnologias necessárias para o desenvolvimento dos serviços e aplicações que o compõem. Dessa forma, baseando-se na ISO 23247 para criação de aplicações DT e situando-a no contexto da manufatura aditiva de metal; apresenta-se nas seções a seguir a definição dos principais protocolos de comunicação utilizados nesse projeto, bem como os principais processos de deposição de material metálico encontrados na indústria, além de um detalhamento dos domínios DT abordados pela norma ISO 23247, explicada na seção a seguir.

2.1 ISO 23247 e *Digital Twins*

A ISO 23247 (ISO, 2022a,b,c,d) é uma norma recente concebida para a padronização de arquiteturas *Digital Twin* em aplicações de indústria e desenvolvimento de pesquisas. Tal norma estabelece um framework de apoio à criação de DT's de Elementos de Manufatura Observáveis (OMEs). A abrangência desses elementos inclui processos, serviços, equipamentos, entre outros aspectos de produção em um chão de fábrica e de integração de dispositivos em laboratórios. A norma em questão se demonstra importante para a difusão de aplicações da Indústria 4.0 por facilitar sua implementação e reprodução em variados tipos de instalações, determinando modelos de análise de dados e manutenção preditiva, design de produtos e controle em tempo real de ativos fabris.

Em tal normatização, os modelos *Digital Twin* são categorizados e há a divisão destes em domínios. Cada um dos domínios possui a responsabilidade de fornecer serviços para domínios adjacentes além de implementar aplicações. Tais aplicações podem ser de monitoramento e/ou de controle como por exemplo interfaces homem-máquina que transmitem informações ao supervisor de produção ou usuário final. Nas bordas de cada um desses domínios habilita-se a troca de dados com os domínios em sua borda por meio de protocolos de comunicação.

O primeiro domínio demonstrado na norma engloba ativos de chão de fábrica e dispositivos a serem monitorados em geral no DT, este domínio é denominado Elemento de Manufatura Observável (OME). Neste domínio, há a transmissão de variáveis dos ativos monitorados por dispositivos de campo ou pelo próprio ativo de manufatura, correspondendo à base de todo o DT; tais informações percorrerão posteriormente todos os domínios consecutivos.

O segundo domínio padronizado pela norma corresponde à Entidade de Coleta de

Dados e Controle de Dispositivos (DCDCE). Neste domínio há o componente responsável pela coleta de dados do maquinário monitorado a fim de estruturar informações e organizá-las de uma forma a serem facilmente compreendidas e transmitidas através do DT. Este é o domínio que irá se comunicar com a Entidade *Digital Twin* alimentando-a com dados coletados do OME.

O terceiro domínio denomina-se Entidade *Digital Twin*, neste há a implementação do controle de acesso a recursos e informações, além de aplicações e serviços que compõem a base da interface DT e da subentidade de operação e administração de dados. Ademais, é neste domínio que há a modelagem para a simulação do DT e a base de serviços para o modelo de interface de exibição de dados que serão utilizados posteriormente pela Interface de Usuário *Digital Twin*.

Por último há o domínio chamado Interface de Usuário *Digital Twin*, o qual possibilita o monitoramento do ativo pelo usuário supervisor mediante *dashboards* supervisórios e/ou simulação 3D de movimentação conforme o ativo real. Além disso, há a possibilidade do monitoramento do histórico de dados coletados e controle em quase tempo real.

2.2 Conceitos relacionados a protocolos de comunicação

Em uma implementação de uma arquitetura *Digital Twin*, faz-se necessário criar várias camadas e/ou domínios de serviços e aplicações que se comunicam umas com as outras. Isso se dá pelo fato de que desde a coleta de dados no nível físico com o ativo a ser monitorado até as Interfaces-Homem-Máquina (IHM) há vários interfaceamentos, transformações e estruturações de dados que devem acontecer a fim de implementar as aplicações relacionadas a um DT.

Tal interfaceamento ocorre por meio dos protocolos de comunicação, sejam eles relacionados à obtenção e transmissão de dados para o adaptador na camada física, estruturação e distribuição de mensagens ou até troca de informações entre aplicações web ou similares.

Nas próximas subseções, serão abordados protocolos utilizados na implementação do DT aqui presente, tais como HTTP, MQTT, MTConnect, entre outros.

2.2.1 Ethernet

O protocolo da camada física que será utilizado para obter dados da célula de manufatura robotizada baseada no robô ABB IRB 2600 será o protocolo Ethernet.

Como descrito por (BERNAL, 2022), o protocolo Ethernet é responsável pela transmissão de dados entre máquinas de uma mesma rede local. Tal transmissão se dá por meio

de pacotes que são transmitidos a receptores identificados por meio de um sistema de endereçamento. Cada pacote possui tamanho variável e possui um cabeçalho contendo o tipo de dados sendo transmitidos além dos endereçamentos de destino e origem do pacote.

Por se tratar de um protocolo de comunicação bem veloz e versátil com uma banda larga bem expressiva, o protocolo Ethernet é comumente utilizado para transmissão de dados entre dispositivos de campo e dispositivos de controle/monitoramento de aplicações IoT. Não obstante, uma aplicação *Digital Twin* é passível de ser implementada com sua base de monitoramento de ativos fundamentada no protocolo Ethernet.

2.2.2 MTConnect

Como descrito pelos documentos disponíveis no site do MTConnect Institute em (MTCONNECT, 2022), o protocolo MTConnect é um padrão de troca de informações entre dispositivos que define uma série de modelos semânticos de dados para uma representação de como a informação se relaciona com uma operação de manufatura.

De uma maneira geral, o protocolo MTConnect define um fluxo de dados que se baseia em um dispositivo adaptador e um agente para estruturação desses dados. Um cliente para visualização de dados em forma de *dashboard*, simulação 3D ou outra forma de supervisório também é passível de ser implementada no final do fluxo citado.

Em resumo, os dados coletados de dispositivos de campo ou ativos de manufatura são estruturados no formato de string SHDR (*Simple Hierarchical Data Representation*) com uma *timestamp* (marcação temporal) correspondente ao momento de coleta dessas variáveis. O agente então disponibilizado em um servidor local de um dispositivo, seja ele o próprio dispositivo adaptador ou outro computador conectado na mesma rede do adaptador com acesso aos dados coletados, organiza o histórico de variáveis coletadas e transmitidas por pacotes HTTP em um formulário XML que separa as informações conforme os dispositivos monitorados configurados. O cliente, por fim, pode se utilizar desse formulário XML para organizar a visualização de interface de usuário de uma maneira mais adequada conforme a aplicação a ser implementada.

2.2.3 MQTT

Da mesma forma, ao se observar a descrição do protocolo MQTT no site de sua organização em (MQTT, 2022) tem-se que aquele é um protocolo de mensagens padrão para aplicações de Internet das Coisas. Uma das premissas desse protocolo é permitir transmitir mensagens através de dispositivos utilizando pouca banda de rede e poucos recursos em geral.

Este protocolo se baseia numa arquitetura *Publisher-Broker-Subscriber* na qual, após a coleta de dados por dispositivos de campo pelos *publishers*, tais dados são estruturados

em mensagens que são publicadas ao *broker*. O *broker*, então, implementado em um servidor local, realiza a distribuição dessas mensagens em tópicos assinados pelos *subscribers*, cada assinante recebe as mensagens relativas aos tópicos que assina. Tal fluxo de dados é exemplificado pela figura 2.2.

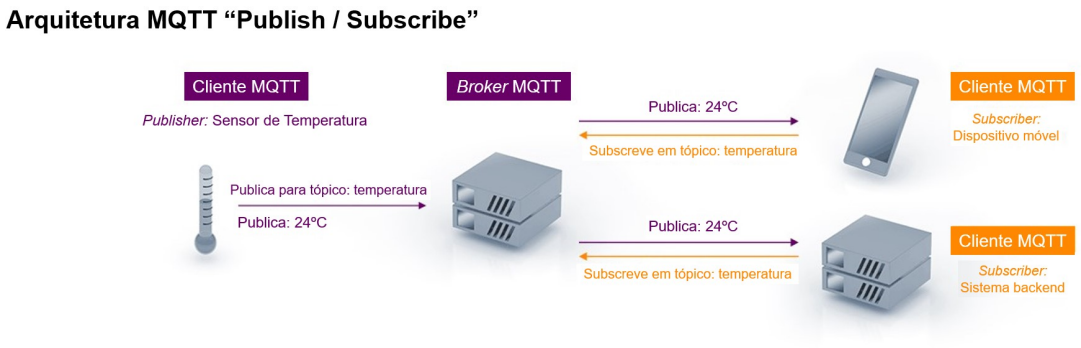

Figura 2.2 – Fluxo de dados e mensagens padrão do protocolo MQTT.

Fonte: (MQTT, 2022)

Por meio da criação de *subscribers* locais, é possível passar os dados coletados para servidores em "nuvem", estruturando as informações em um histórico de variáveis. Desse modo, aplicações como *dashboards* e páginas de simulação 3D de ativos monitorados também podem ser criadas na "nuvem"para visualização remota.

2.3 Deposição de material

Outro aspecto importante para o desenvolvimento de uma célula de manufatura aditiva é a forma pela qual o material é depositado para criar as peças 3D. Tendo em vista a deposição de material metálico, algumas técnicas serão descritas as quais podem ser utilizadas no projeto para a efetivação da manufatura pela célula robotizada com o robô ABB IRB 2600.

2.3.1 FDM - Fused Deposition Modeling

Esse é o processo de impressão mais conhecido atualmente, pois é de fácil utilização e baixo custo, o que o torna com grande acessibilidade. Consiste em uma máquina com três eixos principais e um eixo para o tracionamento do filamento onde o bico de impressão é aquecido a uma determinada temperatura e o filamento é fundido e tem-se a deposição do filamento na plataforma de impressão e com os movimentos do eixo é formada a geometria da peça. O processo pode ser visualizado na figura 2.3:

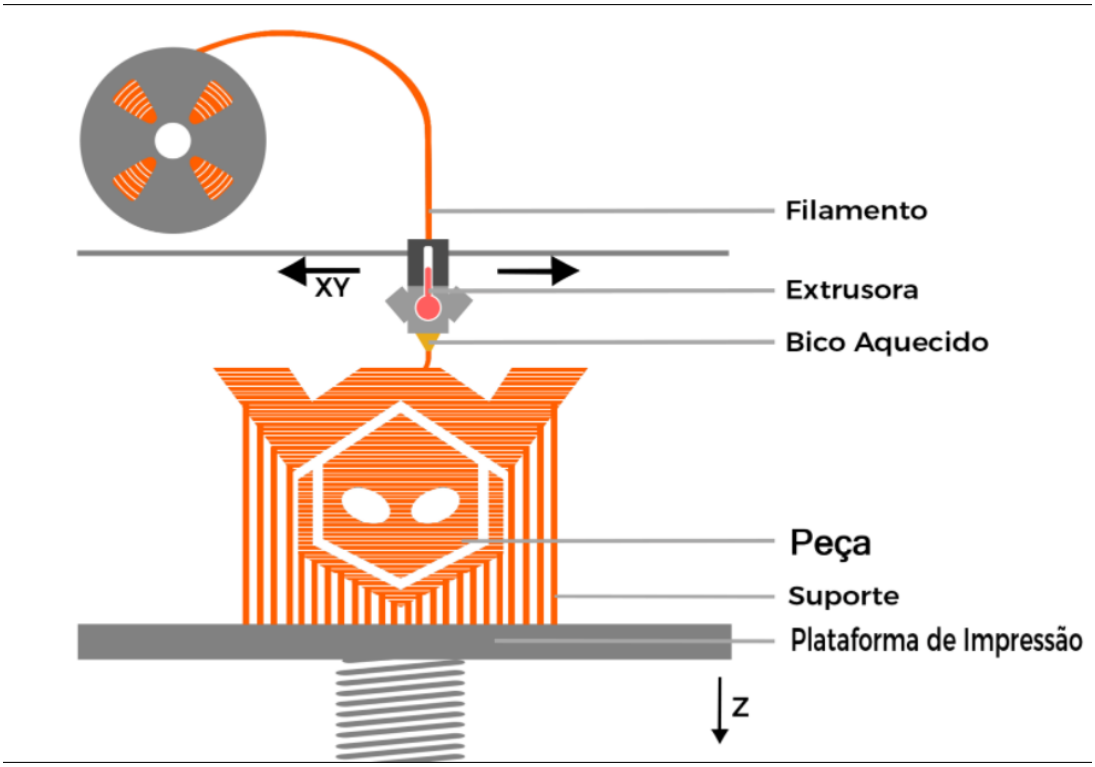

Figura 2.3 – Modelo de Impressão pelo método FDM.

Fonte: (WISHBOX, 2022)

2.3.2 Sinterização seletiva a laser e fusão seletiva a laser

O método de Selective Laser Sintering (SLS) ou Seletive Laser Melting (SLM) para metais é uma forma de impressão muito versátil, entretanto um processo que possui complexidade maior, além disso, impressoras que utilização essa tecnologia são mais caras. A sinterização a laser é um processo onde o material bruto da impressão é um pó podendo ser plástico (náilon) ou metal, onde um feixe de laser seletivo difunde o pó (em estado sólido) camada por camada até a construção da geometria completa da peça. No caso de metais (SLM), o material precisa ser completamente fundido a cada camada de pó aplicada. Após a peça pronta é necessário utilizar alguns processos para retirar o pó que não foi sinterizado ou fundido, como o uso de jato de ar para limpeza da peça. No caso metais é necessário os processos de lavagem e tratamento térmico. O processo pode ser observado na figura 2.4:

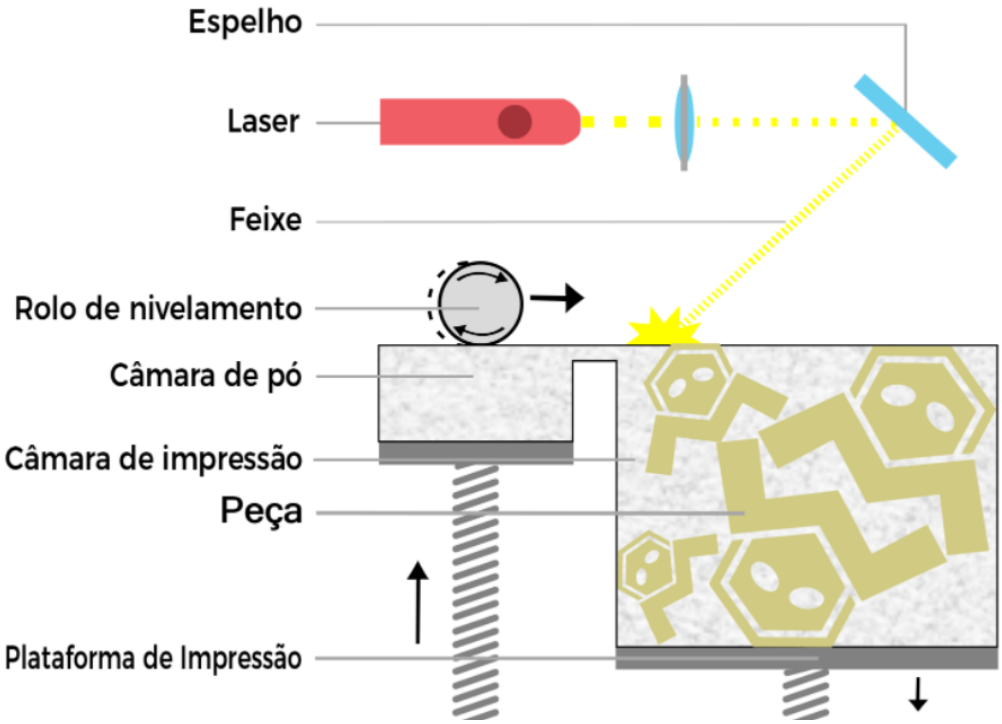

Figura 2.4 – Modelo de Impressão pelo método SLS.

Fonte: (WISHBOX, 2022)

2.3.3 Soldagem a arco gás-metal (GMAW)

Outro método de deposição de material é a soldagem a arco de arame de metal e gás (GMAW), tal metodologia de manufatura aditiva por arco de arame (WAAM) descrita em (HENCKELL et al., 2020) consiste na aplicação constante de um arco aberto entre o arame, continuamente alimentado, e o material previamente depositado sobre o substrato metálico, configurando uma deposição em camadas. Um dos desafios deste tipo de processo é a determinação de geometria, tamanho de grãos e dureza da peça metálica fabricada através do controle de velocidade e de aporte de calor no sistema.

Neste tipo de processo de deposição de material, as taxas de deposição podem chegar até 8 kg/h. Entretanto, existem limitações quanto a exatidão dimensional e qualidade da superfície atreladas ao tamanho de poças de fusão quando se compara com a precisão necessária para tal tipo de deposição. Portanto, muitas vezes se faz necessário aplicar um processo de pós-manufatura como o fresamento.

2.3.4 Soldagem por Gás Inerte de Tungstênio (GTAW)

Método análogo ao GMAW, na soldagem GTAW ou TIG um eletrodo de tungstênio não consumível é utilizado, sendo o eletrodo e a poça de fusão protegidos por um gás inerte que sai da tocha (SINGH, 2020). Os gases utilizados nesse processo são o argônio e gás hélio.

Geralmente esse processo utiliza corrente contínua de polaridade negativa no eletrodo de tungstênio. Além disso, há a possibilidade de usar corrente alternada para produzir diferentes efeitos na soldagem.

Outra distinção entre os métodos GMAW e GTAW é o fato que no último a alimentação do material de adição na forma de arame é realizada em direção perpendicular à orientação do eletrodo, o que produz limitações quanto à movimentação do manipulador robótico na manufatura (rotação e velocidade) mas que também gera um melhor acabamento na peça fabricada. Tal particularidade é evidenciada na figura 2.5

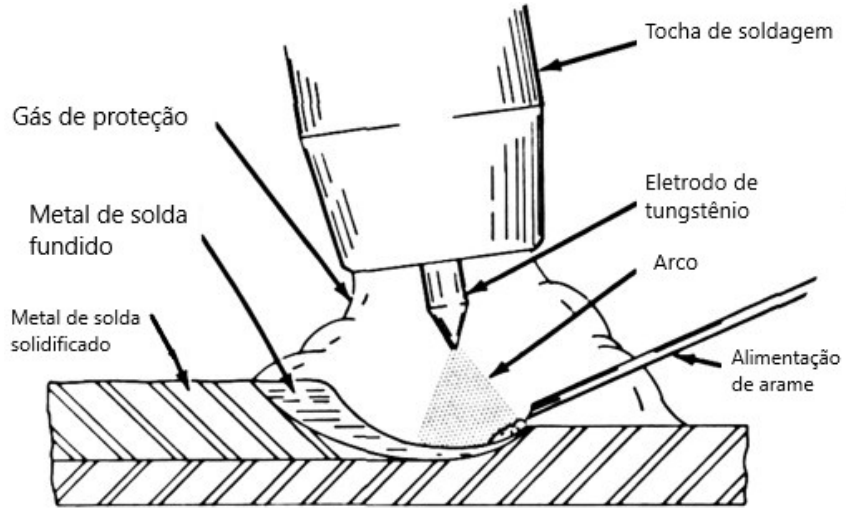

Figura 2.5 – Diagrama de soldagem GTAW.

Fonte: adaptado de (ANTONINI, 2014)

A soldagem GTAW é aplicada no laboratório GRACO através da fonte de soldagem GTAW Fronius MW5000, que está conectada à tocha GTAW instalada no órgão terminal do robô para permitir a fabricação de peças metálicas 3D na célula robótica de manufatura aditiva.

2.3.5 LMD – Laser Metal Deposition

Como descrito em (SELCUK, 2013), esse processo de deposição se utiliza de um laser de alta potência para criar uma poça de fusão de metal na qual se funde metal em pó ou arame para criar uma trilha de metal. Tal metodologia se destaca pelo baixo impacto na resistência do material, além de gerar menores distorções por possuir uma menor zona afetada pelo calor quando comparada com outros métodos, como, por exemplo, a soldagem a arco convencional. Ao se colocar o bocal com a alimentação de material no suporte da ponta do robô juntamente com o laser de fusão, é possível realizar a impressão de peças metálicas através do controle da alimentação de material e ativação do laser, juntamente com a movimentação do robô.

2.3.6 CMT – Cold Metal Transfer

Outro método de deposição de material é a transferência de metal frio, que como descrita por (DUTRA; SILVA; MARQUES, 2013) deriva-se do processo de soldagem a arco GMAW caracterizando-se pela transferência metálica por curto-circuito controlado. No CMT, além da forma da onda de corrente controlada, há o controle da velocidade e do sentido de avanço do arame. A constante potência resultante do rápido chaveamento entre tensões altas e baixas permite a rápida fusão do metal a temperaturas mais baixas que os processos de soldagem GMAW convencionais. Tais características permitem, no contexto da manufatura aditiva, a economia de tempo, materiais e consequentemente custo geral. Um exemplo da observação dessa redução nos custos de manufatura pela utilização do método CMT se verifica em (ZHANG et al., 2022), no qual há a redução de erros na deposição de material e o aumento da precisão desta. Entretanto, por se fazer necessária uma fonte de soldagem no processo de manufatura, a complexidade geral da instalação da célula robótica e sua consequente representação na arquitetura *Digital Twin* é acentuada.

2.4 Robô ABB 2600, RAPID e cinemática

Para o melhor entendimento dos programas desenvolvidos ao longo deste trabalho, é preciso esclarecer alguns conceitos relacionados ao robô ABB IRB 2600, bem como aspectos de sua cinemática e controle de movimentação e I/O do controlador do robô via programação RAPID.

A movimentação em pontos do robô nos códigos em RAPID é descrita por um tipo de estrutura de dados chamada de "robtarget". Tal estrutura é dividida em quatro partes:

- **trans**: translação, diz respeito às variáveis de posição cartesiana x, y e z do TCP. Tal posição é utilizada para calcular os ângulos de rotação das juntas do robô na cinemática inversa de movimentação;
- **rot**: Rotação (ou orientação), descreve a orientação da ferramenta em forma de quaternion, $[q1, q2, q3, q4]$;
- **robconf**: Configuração dos eixos do robô, relaciona-se à configuração das juntas na posição zero;
- **extax**: eixos externos, descreve a movimentação de unidades mecânicas externas ao robô na forma $[eax\_a, eax\_b, eax\_c, eax\_d, eax\_e, eax\_f]$. No caso da célula robótica contemplada, os eixos externos configurados referem-se à movimentação da mesa posicionadora. O eixo *eax_b* rotaciona a mesa onde o prato está fixado enquanto o eixo *eax_c* descreve a rotação do prato em si.

Há mais um aspecto a ser considerado quanto aos programas RAPID utilizados como base para testar o fluxo de dados a ser implementado. As posições do robô nos programas RAPID estão descritas conforme um **wobjdata**, (Work Object Data), tal variável descreve a posição do objeto de trabalho que serve de base de referência para a movimentação do robô em relação à posição global do TCP. Neste caso a referência está no centro do prato da mesa posicionadora, sendo assim é possível calcular os pontos de deposição de material em relação a essa referência, sem precisar calcular as posições conforme a posição global do robô. Isso facilita na programação das rotinas em RAPID, além de padronizar o desenvolvimento dos algoritmos e facilitar o desenvolvimento do fluxo de dados, pois quaisquer alterações na célula robótica podem ser facilmente aplicadas ao programa RAPID modificando somente o **wobjdata**.

3 Trabalhos correlatos

A fim de se nortear todo o desse projeto, se fez necessário realizar uma revisão de literatura acerca de implementações semelhantes de *Digital Twins* na manufatura aditiva, em especial, trabalhos que envolvem células de manufatura aditiva robotizadas. O objetivo aqui é demonstrar aplicações semelhantes para a impressão 3D utilizando deposição de material metálico, principalmente aquelas que demonstram soluções mais factíveis tendo em vista os recursos disponibilizados no laboratório GRACO da Universidade de Brasília.

3.1 Gerenciamento de dados colaborativo para sistemas de manufatura aditiva habilitados por *Digital Twin*

No trabalho desenvolvido por (LIU et al., 2022), uma arquitetura DT baseada em "nuvem" é implementada para sistemas de manufatura aditiva baseados na deposição de material metálico. Neste trabalho são citadas diversas formas de deposição de material metálico encontradas no mercado, tais como: fusão seletiva a laser, extrusão de material e aspersão térmica de material.

A arquitetura do trabalho citado é evidenciada na figura 3.6, na qual se encontram demonstrações de módulos para planejamento de processos de produção, mecanismos de controle de qualidade, pós-processamento, manufatura e design de produto. Além das funcionalidades isoladas implementadas por esses módulos, há a inovação na forma de gerenciamento coletivo de dados por meio do uso da "nuvem".

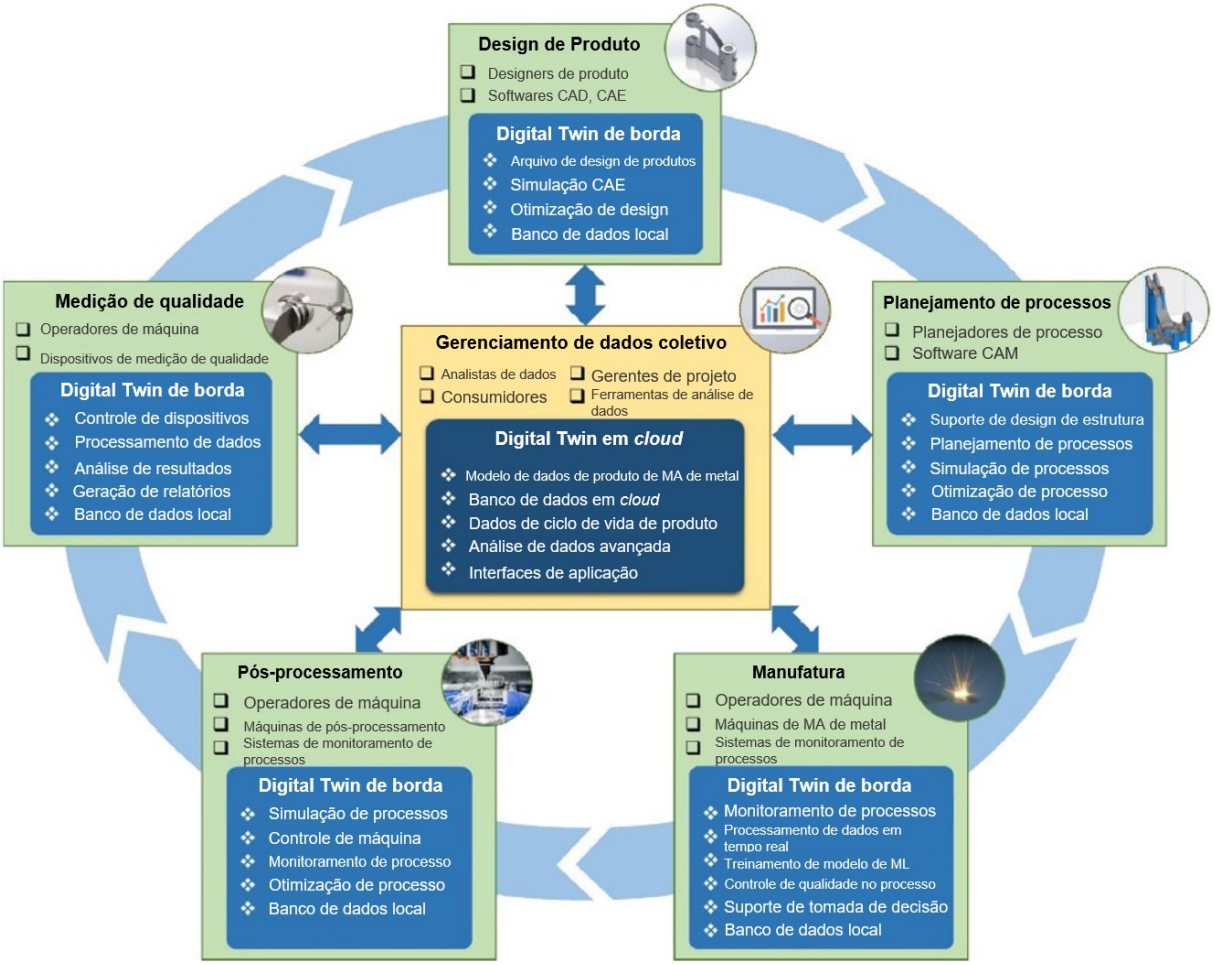

Figura 3.6 – Framework DT conceitual para gerenciamento colaborativo de dados para MA de metal.

Fonte: adaptado de (LIU et al., 2022, p. 5)

As aplicações descritas no trabalho mencionado fornecem visualização em tempo real de variáveis de manufatura, além do processo CAD/CAM de design de produto e simulação 3D da impressão por células de manufatura robotizadas. Ademais, os dados coletados são utilizados para gerar relatórios de pós-manufatura de níveis de qualidade de peças geradas.

A estrutura de visualização dos dados coletados e relatórios de análise por parte do usuário pode ser vista na figura 3.7. As funções descritas passam desde o planejamento de produto, gerenciamento de projetos, planejamento de produção até análise de dados de ciclo de vida do produto juntamente com feedback dos consumidores finais.

Em resumo, tal projeto citado propõe e executa uma arquitetura *Digital Twin* complexa com vários serviços e aplicações sendo disponíveis tanto localmente quanto em "nuvem"para supervisão e controle de linhas de produção por manufatura aditiva.

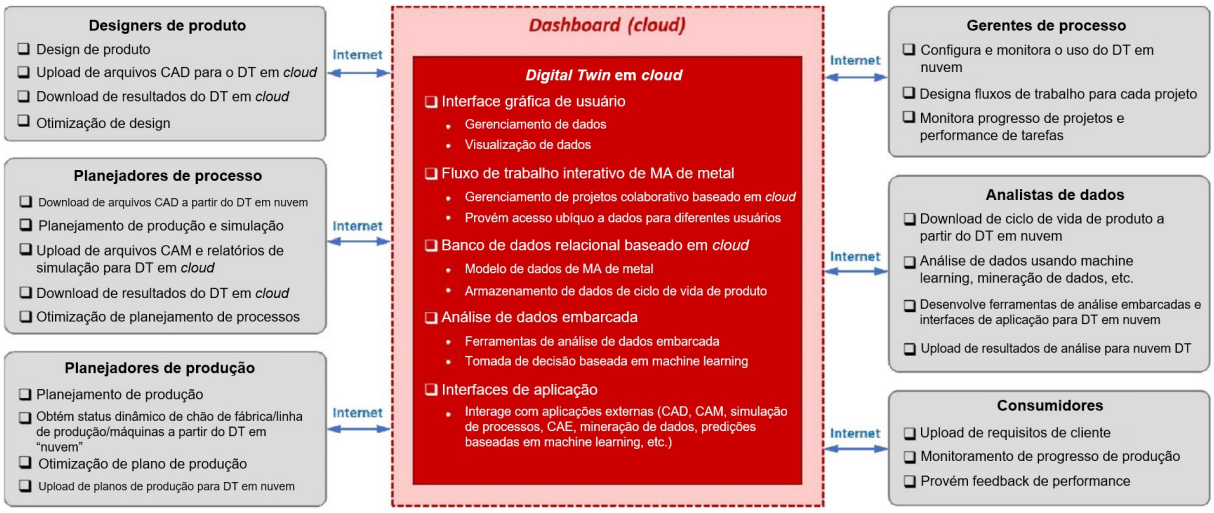

Figura 3.7 – Funções do *dashboard* de usuário *Digital Twin* em "nuvem".

Fonte: adaptado de (LIU et al., 2022, p. 10)

3.2 Interface para células de manufatura aditiva industriais multimarcas

O segundo trabalho utilizado como base para este projeto é o desenvolvido em (ZHU; PIRES; AZAR, 2020), que demonstra a concepção de ambientes de simulação para a manufatura aditiva. Neste estudo, ferramentas CAD/CAM foram utilizadas para melhorar o ciclo de desenvolvimento de peças e produtos por meio de ambientes de simulação de fabricação e de linhas produtivas. Essa melhora pode ser feita através da análise prévia da impressão 3D de peças através de simulações.

Outro foco deste trabalho é a conversão de código G gerado por softwares de fatiamento em um código que pode ser interpretado pelos robôs das células de manufatura. O objetivo disso seria criar um sistema universal de CAD/CAM que funcionasse independentemente da marca do robô, cuja saída seriam instruções de caminho interpretáveis pelo robô que realiza a impressão. No projeto citado foram utilizados robôs Kuka e ABB para simulação e desenvolvimento do *Digital Twin*.

Além da conversão de código realizada, o trabalho também estabelece um modelo de *Digital Twin* para simulação da manufatura aditiva realizada nas células de trabalho. Isso se dá por meio de uma GUI de visualização de variáveis e caminho da ferramenta, além da simulação 3D das células de manufatura robotizada.

3.3 Desenvolvimento *Digital Twin* para manufatura subtrativa baseada na ISO 23247

Por último, um trabalho correlato que se baseia no mesmo padrão internacional que será utilizado no desenvolvimento do *Digital Twin* deste projeto é o trabalho desenvolvido em (CABRAL; GASCA; ALVARES, 2023).

Diferentemente do presente trabalho, no artigo citado o foco é na manufatura subtrativa. Outro aspecto a ser considerado são as diferenças nos protocolos e linguagem utilizados no *Digital Twin*. Como se trata de uma fresadora, o código G é utilizado no ativo físico para a usinagem, ao contrário do código RAPID usado na movimentação do robô no presente trabalho. Em relação aos protocolos de comunicação utilizados, entre os domínios OME e DCDCE, o protocolo utilizado é o RS-232 (serial), enquanto no robô ABB será utilizado o protocolo Ethernet para coleta de dados. Apesar de a ABB disponibilizar o módulo de comunicação serial para o controlador como opcional, no ativo do laboratório GRACO esta opção não está inclusa.

Tais diferenças são melhor evidenciadas na imagem que resume a arquitetura do *Digital Twin* do artigo, que está representada na figura 3.8.

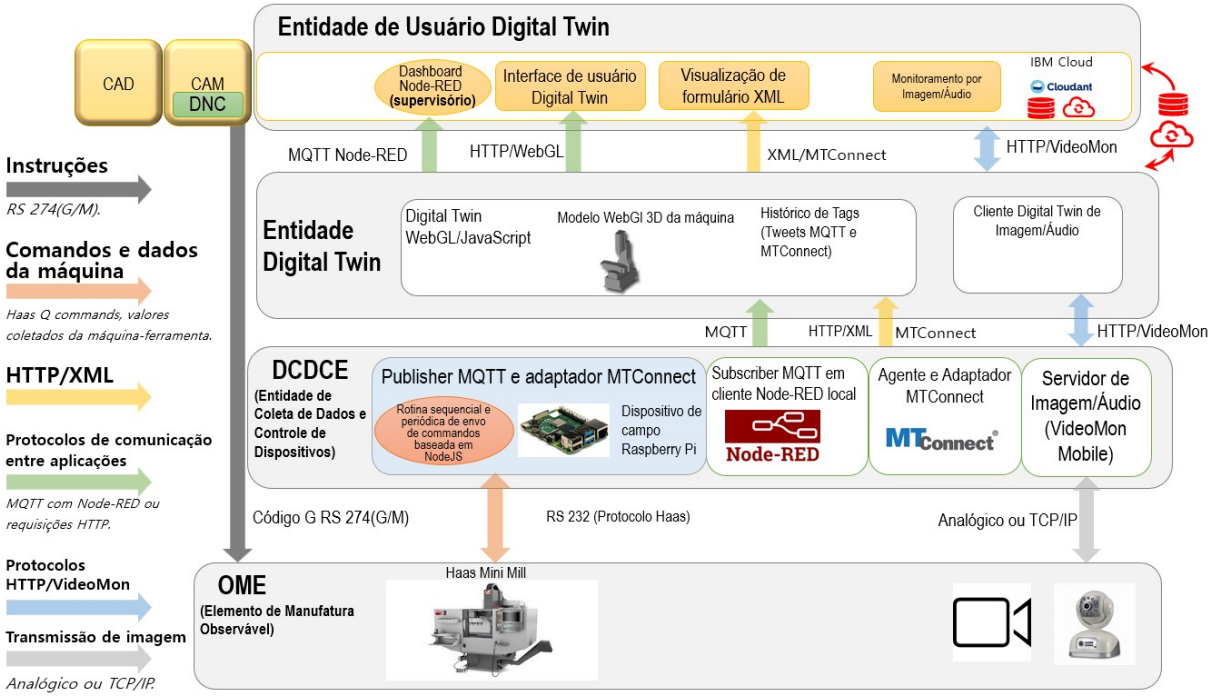

Figura 3.8 – Arquitetura Digital Twin para manufatura subtrativa baseada na ISO 23247.

Fonte: (CABRAL; GASCA; ALVARES, 2023, p. 4)

Em relação aos outros domínios do *Digital Twin*, as aplicações desenvolvidas, protocolos utilizados e domínios DT são bastante similares aos projetados para o presente trabalho. Outra diferença notável é a presença de um fluxo menor MTConnect no trabalho citado, enquanto no presente projeto tal fluxo não será desenvolvido.

4 Metodologia

Como base da metodologia do projeto desenvolvido neste trabalho, optou-se por seguir a norma ISO 23247 descrita na seção 2.1 a fim de planejar domínios de fornecimento de serviços e aplicações-base para a implementação de um *Digital Twin* para a manufatura aditiva.

Nas seções a seguir, a arquitetura *Digital Twin* proposta será destrinchada em módulos a fim de explicar como se estruturou a coleta e transformação de dados ao longo dos domínios DT previstos na norma ISO 23247.

4.1 Arquitetura *Digital Twin* baseada na norma ISO 23247

Fundamentalmente, o protocolo MQTT é utilizado para estabelecer o fluxo de dados entre os principais domínios DT. Tal protocolo funciona juntamente com o protocolo Ethernet TCP/IP para a transmissão de dados dos ativos monitorados (OMEs) para o domínio DCDCE descrito na ISO 23247. As variáveis coletadas são estruturadas por um dispositivo adaptador e passadas por meio de mensagens seguindo a estruturação padrão do protocolo MQTT (*Publisher-Broker-Subscriber*).

Assim, se faz possível implementar um cliente para interfaceamento homem-máquina em um *dashboard* online ou local através do framework Node-RED. Outra visualização a ser implementada é a simulação 3D de movimentação do robô de acordo com variáveis coletadas no software RoboDK.

Tal arquitetura descrita pode ser observada na forma de domínios em conformidade com a norma ISO 23247 na figura 4.9.

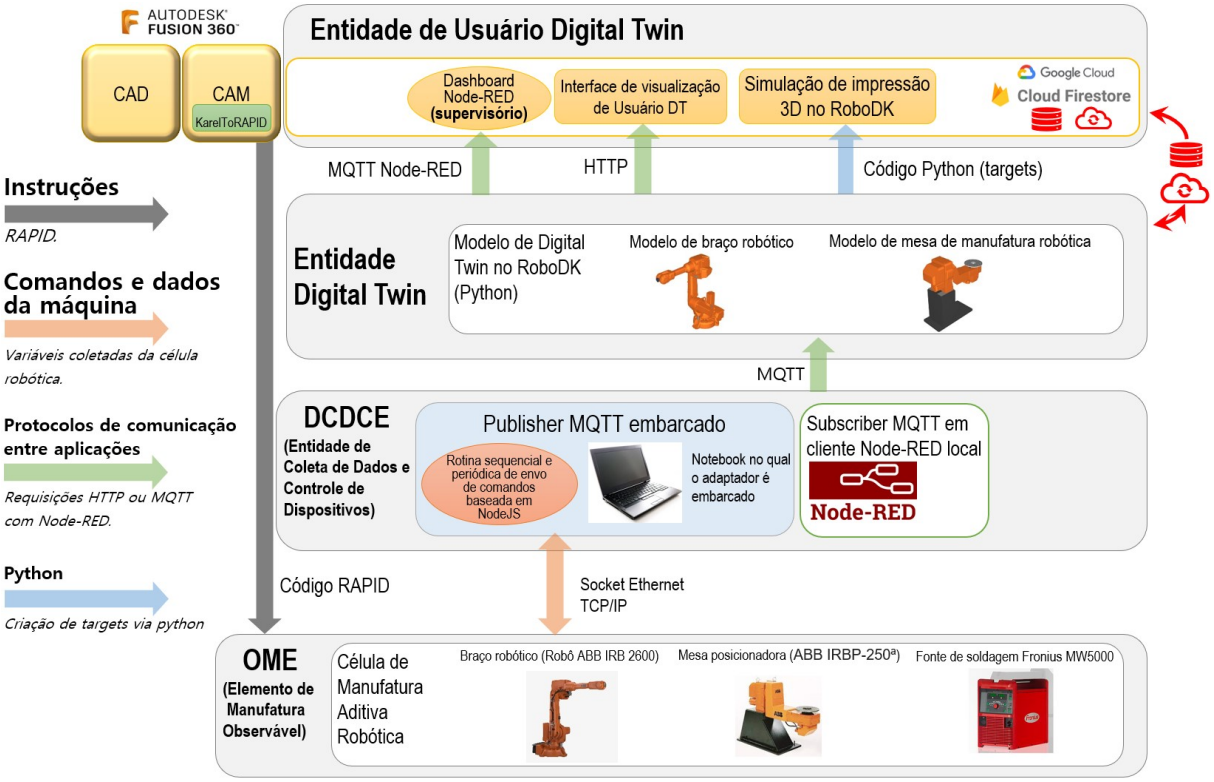

Figura 4.9 – Estruturação do *Digital Twin* planejado com base na norma ISO 23247.

Por fim, o robô IRB 2600 a ser monitorado pelas aplicações presentes na arquitetura *Digital Twin* pode ser observado na figura 4.10. Esse ativo se encontra instalado no laboratório GRACO da Universidade de Brasília e toda a arquitetura *Digital Twin* proposta é baseada nas configurações espaciais e limitações da instalação física do robô ABB IRB 2600 e mesa posicionadora ABB IRBP A250.

Figura 4.10 – Célula de manufatura aditiva instalada no laboratório GRACO da UnB, composta pelo robô ABB IRB 2600 e mesa posicionadora robótica ABB IRBP A250.

4.2 Fatiamento e geração código RAPID

A fim de agilizar o processo de desenvolvimento e de testes do *Digital Twin*, soluções já disponíveis de código aberto serão utilizadas para o fatiamento das peças a serem fabricadas e geração do código RAPID correspondente a esse fatiamento.

Nesse sentido, serão utilizadas as soluções desenvolvidas nos Trabalhos de Graduação descritos em (ANDRADE, 2013) e (SALES DE MATOS, 2022), cujos softwares resultantes serão detalhados nas subseções a seguir.

4.2.1 Software de Fatiamento

No Trabalho de Graduação (ANDRADE, 2013), é descrito o desenvolvimento de um software capaz de realizar o fatiamento de peças 3D pré-projetadas a fim de utilizar esses pontos como ponto de partida para a definição de estratégias de deposição de material na manufatura aditiva.

A estratégia adotada no fatiamento realizado por esse software se baseia na interpolação contínua entre pontos em uma trajetória helicoidal que conecta dois planos. Planos esses que são empilhados na direção de crescimento da peça a ser fabricada até que o plano mais alto seja alcançado. A trajetória helicoidal na direção do eixo de crescimento determinada pelos pontos de deposição de material entre planos é exemplificada (com a distância entre planos propositalmente exagerada) na figura 4.11.

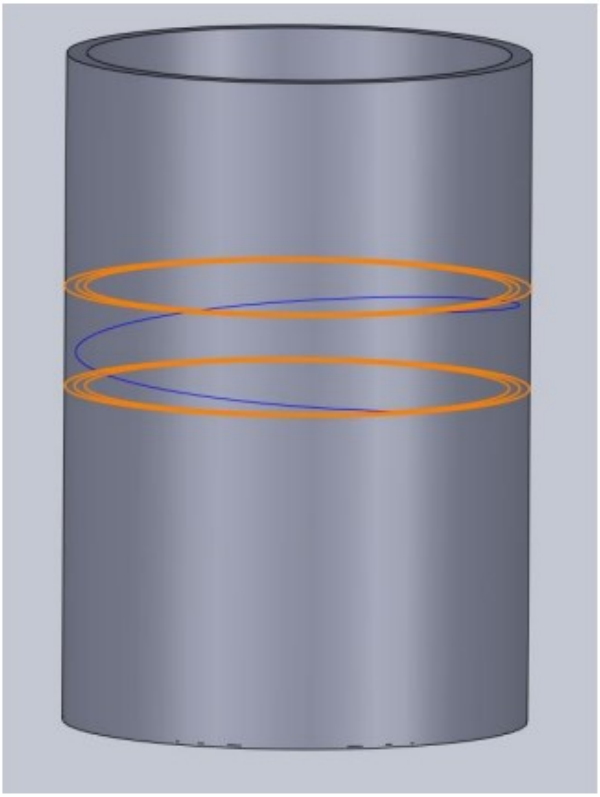

Figura 4.11 – Exemplo de ciclo de fatiamento entre dois planos conforme trajetória helicoidal.

Fonte: (ANDRADE, 2013)

Tal programa pode ser utilizado para realizar a conversão de peças 3D em arquivos ".sat"para código Karel (WorkspaceLT). A saída desse programa será posteriormente utilizada como entrada do software de conversão para código RAPID. No programa de fatiamento, é possível configurar a distância (em milímetros) entre as camadas de material depositado além do número de casas decimais de precisão dos pontos gerados. A representação 3D do fatiamento gerado pelo programa pode ser observada na figura 4.12.

A principal limitação desse programa é que não há variação de espessura na estratégia de pontos do fatiamento, sendo assim não é possível fatiar peças maciças com este software, somente sólidos tipo casca são contemplados.

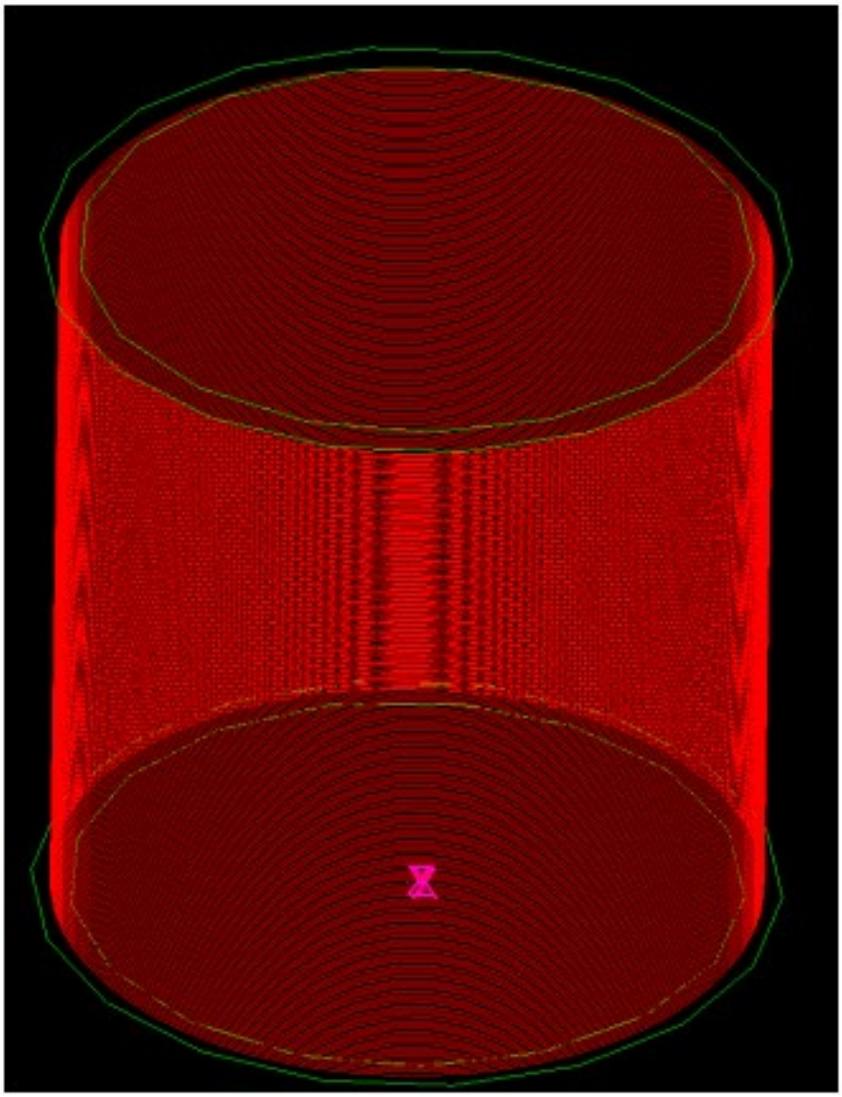

Figura 4.12 – Exemplo de modelo casca cilíndrica fatiada na saída do software descrito (distância entre planos de 1 mm).

Fonte: (ANDRADE, 2013)

4.2.2 Conversão para RAPID software KarelToRAPID

A partir da saída do software de fatiamento executado de maneira prévia, utiliza-se o código de saída Karel como entrada para outro programa que irá realizar a conversão para código RAPID, que pode ser interpretado pelo controlador do robô IRB 2600.

Através do software desenvolvido no Trabalho de Graduação (SALES DE MATOS, 2022), é possível realizar a conversão de código Karel para RAPID do fatiamento de casca. Para simplicidade, a estratégia sem orientação da mesa descrita na subseção **4.2.1.1** do trabalho citado será utilizada para os testes do *Digital Twin*. Nessa estratégia, apenas o prato da mesa giratória é rotacionado, com o braço da mesa permanecendo imóvel, sendo assim, só um grau de liberdade da mesa é aproveitado, porém a estratégia de manufatura e consequente execução em código RAPID e simulação são simplificados.

Tal estratégia é exemplificada na figura **4.39** do trabalho citado, reproduzida a seguir:

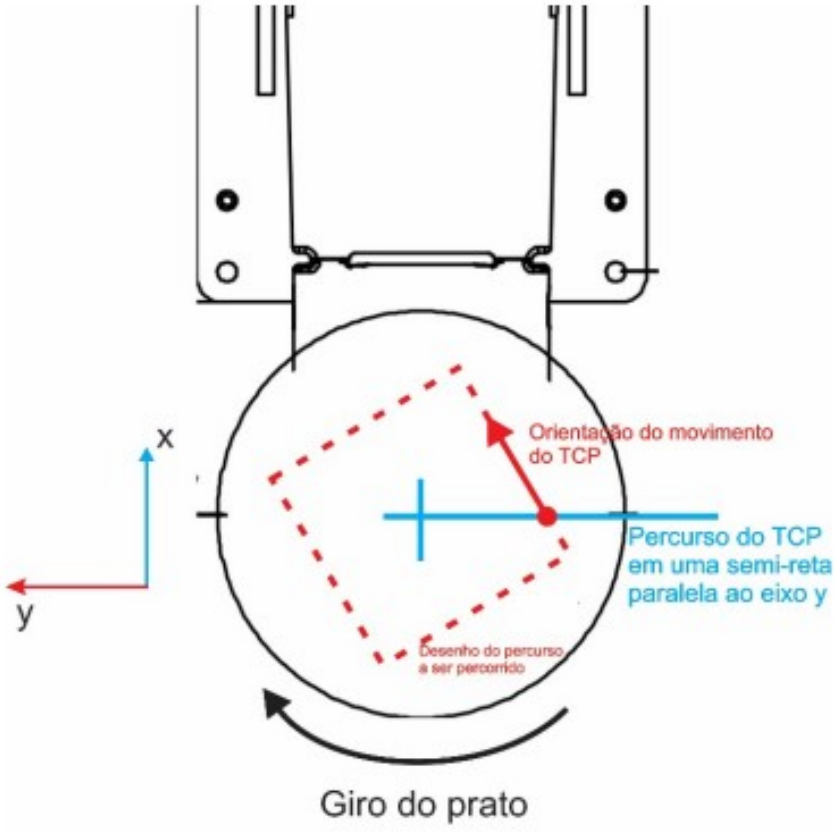

Figura 4.13 – Descritivo do movimento do TCP (Tool Center Point) e rotação da mesa posicionadora IRBP-250A na estratégia sem orientação.

Fonte: (SALES DE MATOS, 2022)

Ao limitar a movimentação da mesa posicionadora, é possível descrever a movimentação do robô por meio de coordenadas cilíndricas (r, θ, z)no código RAPID. A rotação do ângulo θ é realizada exclusivamente pela rotação do prato da mesa giratória, enquanto as coordenadas r e z descrevem a movimentação do robô no eixo y conforme a figura 4.13 e eixo vertical de deposição do material respectivamente. Nessa estratégia também não há rotação do punho do manipulador da tocha GTAW.

Uma limitação dessa estratégia descrita nos resultados de (SALES DE MATOS, 2022) são as distorções nos cantos das arestas geradas pela rotação da mesa posicionadora. Velocidades mais altas de movimentação podem ser utilizadas para tentar corrigir esse problema, mas outras distorções podem ser geradas nesse caso por conta do espirramento de metal líquido, uma vez que leva um tempo até que este se solidifique.

4.3 Coleta de dados

Em relação à coleta de dados do controlador ICR5 do robô ABB IRB 2600, duas abordagens de implementação podem ser utilizadas.

A primeira e mais simples seria criar uma rotina de coleta de dados através da adaptação de um programa em Python ou JavaScript compatível com o Robot Web Services disponibilizado a partir do RobotWare 6 (versão do sistema operacional do controlador ICR5, que compreende todos os programas e módulos necessários para a execução de tarefas pelo robô ABB IRB 2600).

A segunda, que requer alterações diretamente no algoritmo RAPID para execução de comandos pelo controlador, é a descrita no manual de comunicações do robô compatível com RobotWare 5 (ABB, 2004). Tal solução se baseia numa comunicação via *socket* TCP/IP dentro da execução do programa RAPID interpretado pelo controlador do robô.

As duas abordagens serão detalhadas nas subseções a seguir. Por fim, uma metodologia de cálculo de energia de soldagem instantânea ao longo do processo de fabricação também é apresentada, utilizando-se dos dados coletados via I/O do controlador ICR5 provenientes da fonte de soldagem Fronius por meio de sua integração ao controlador do Robô, via conversão da rede Fronius LocalNet para a rede DeviceNet utilizada no IRC5

4.3.1 Rotina de coleta de dados via Robot Web Services

De acordo com a documentação da ABB para o Robot Web Services (RWS) (ABB, 2021), é possível criar uma aplicação REST compatível com as comunicações disponíveis no controlador ICR5 via TCP/IP. O RWS se trata de um framework baseado em REST que habilita o uso do protocolo de aplicação HTTP para o envio de requisições ao controlador.

Segundo o manual, o framework é compatível com formulários html e XML, bem como formatos JSON e comunicação via WebSocket. Todos esses são recursos valiosos para o estabelecimento de conexão e troca de mensagens entre o adaptador do DT e o controlador do robô para coleta de dados.

Por se demonstrar uma forma de acesso mais fácil às informações do robô através dos métodos HTTP (GET principalmente), implementar uma solução via RWS se mostra a forma ideal de habilitar a coleta de dados pelo *Digital Twin*, dispensando alterações no código RAPID interpretado pelo controlador do robô.

4.3.2 Modificação de código RAPID para comunicação via *socket* TCP/IP

Segundo a seção **8** do manual (ABB, 2004), é possível realizar a comunicação entre o controlador IRC5 e qualquer dispositivo que aceita comunicação via TCP/IP (um computador, por exemplo). Para tal, um *socket* TCP/IP é criado como um servidor para a comunicação

em um dispositivo conectado ao controlador via Ethernet. A partir disso, um *socket* cliente é criado dentro do código em RAPID interpretado pelo controlador a fim de estabelecer a comunicação com o servidor, habilitando a troca de mensagens entre os dois pontos.

Um exemplo de implementação de um *socket* TCP/IP cliente em um código RAPID pode ser visto na figura 4.14, extraída do próprio manual da ABB.

Code example for client, contacting server with IP address 192.168.0.2:

```
! WaitTime to delay start of client.
! Server application should start first.
WaitTime 5;
VAR socketdev socket1;
VAR string received_string;
PROC main()
  SocketCreate socket1;
  SocketConnect socket1, "192.168.0.2", 1025;
  ! Communication
  SocketSend socket1 \Str:="Hello server";
  SocketReceive socket1 \Str:=received_string;
  TPWrite "Server wrote - " + received_string;
  received_string := "";
  ! Continue sending and receiving
  ...
  ! Shutdown the connection
  SocketSend socket1 \Str:="Shutdown connection";
  SocketReceive socket1 \Str:=received_string;
  TPWrite "Server wrote - " + received_string;
  SocketClose socket1;
ENDPROC
```

Figura 4.14 – Exemplo de cliente TCP/IP embutido em código RAPID interpretado pelo controlador ICR5.

Fonte: (ABB, 2004)

Para a coleta de dados do *Digital Twin* através dessa abordagem, uma solução em RAPID similar à apresentada no manual seria desenvolvida, com as devidas alterações para o envio de dados dos sinais de entrada e saída do controlador durante a execução do programa de manufatura.

Entretanto, uma evidente desvantagem desse método é a necessidade de adaptação de cada algoritmo de manufatura RAPID para realizar a comunicação via *socket* TCP/IP. Sendo assim, tal solução não se demonstra a ideal por inserir mais um passo entre a geração de código RAPID do fatiamento e sua interpretação pelo controlador do robô; portanto será somente utilizada se nenhuma outra abordagem se mostrar eficaz.

4.3.3 Cálculo de energia de soldagem

Utilizando-se do módulo DeviceNet, que se comunica com a fonte de soldagem Fronius, é possível coletar diversas variáveis de soldagem que podem ser usadas para calcular

a energia de soldagem instantânea durante a execução de um programa RAPID.

Tal energia pode ser calculada através da seguinte equação (EAGAR, T. W., 1993):

$$E = \eta * \frac{V * I}{\upsilon} \tag{4.1}$$

onde a energia de soldagem E (J/mm) é obtida pela tensão de soldagem V (em volts), corrente de soldagem I (em amperes), multiplicados por um fator de eficiência η e divididos pela velocidade de soldagem. O valor do coeficiente de eficiência η admitido para o cálculo é de 0,7, similar a valores encontrados na literatura para o método de soldagem GTAW.

Tal cálculo é realizado a cada ciclo de soldagem, e o valor calculado estará disponível para visualização no *dashboard* Node-RED com os valores também sendo armazenados na "nuvem"a cada ciclo.

4.4 Adaptador MQTT

Se tratando do domínio mais importante de um *Digital Twin*, o software adaptador realiza a ponte entre dados coletados do ativo físico "crus"e o início da definição de um fluxo de dados estruturados de acordo com o protocolo de comunicação a ser utilizado, no caso desse projeto o protocolo MQTT.

No adaptador serão inclusos o servidor TCP/IP para coleta de variáveis enviadas periodicamente pelo controlador do robô e um *broker*/*publisher* MQTT para inicializar o fluxo MQTT de mensagens.

4.5 *Dashboard* de monitoramento

Um dos objetivos de se implementar o *Digital Twin* é habilitar o monitoramento de variáveis do sistema físico em tempo real (ou quase real). A fim de se atingir esse objetivo, o fluxo de dados diagramado e explicado em 4.1 será utilizado.

Após a coleta de dados pelo adaptador, tais variáveis coletadas serão passíveis de visualização através da implementação de um *dashboard* utilizando o framework Node-RED, framework comum no desenvolvimento de aplicações IoT. Os dados serão passados ao *dashboard* através do protocolo de comunicação MQTT, o fluxo Node-RED também poderá ser utilizado para publicação dos dados em *cloud*, o que possibilitará a criação de um *dashboard* com host na "nuvem"se necessário.

4.6 Simulação de impressão 3D

Uma das principais diferenças entre um supervisório comum e um *Digital Twin* é a possibilidade de espelhamento da movimentação realizada pelo ativo real. Dessa forma, é possível observar no computador mediante uma IHM (interface homem-máquina) o exato estado atual do ativo monitorado. No caso da célula de manufatura robótica o modelo 3D espelhará a movimentação do robô, mesa posicionadora, posição do TCP e deposição de material.

Sendo assim, neste trabalho, é simulada a impressão 3D de peças metálicas no momento em que elas estão sendo fabricadas pela célula de manufatura aditiva robótica no laboratório. Isso é feito através do monitoramento de variáveis de posicionamento e de sinais I/O obtidos da fonte de soldagem Fronius MW5000 GTAW através do controlador ICR5.

Tal simulação é feita utilizando-se do software RoboDK, no qual é possível implementar um *socket* para receber as informações obtidas pelo adaptador, dessa forma para cada novo ponto de movimentação do robô um "target"é gerado para movimentação do modelo 3D na simulação no RoboDK. Utilizando-se de macros dentro do RoboDK também é possível simular a deposição de material.

4.7 Publicação de dados em *cloud*

No final do fluxo MQTT, entre os domínios entidade *Digital Twin* e interface de usuário DT em 4.9 há a integração dos dados do fluxo com a "nuvem". Pensou-se inicialmente em utilizar os serviços IBM Cloud Cloudant, baseados no Cloud Foundry da IBM, porém infelizmente esse serviço foi descontinuado.

A outra alternativa pensada então foi utilizar o Firebase Firestore da Google Cloud. As vantagens de se utilizar esse serviço é a fácil integração com o Node-RED e Node.js, além disso por se tratar de um servidor NoSQL, a postagem de dados e também consulta de documentos pode ser feita de maneira mais rápida que muitos bancos de dados SQL tradicionais.

5 Resultados

Como resultado da implementação da metodologia descrita no capítulo anterior, diversos módulos foram desenvolvidos para funcionarem de maneira conjunta, implementando assim o fluxo de dados descrito na arquitetura *Digital Twin* em 4.1.

Nas sessões a seguir, cada um dos módulos será detalhado com suas aplicações descritas conforme os domínios da norma ISO 23247 e arquitetura planejada 4.9.

5.1 Algoritmo RAPID para obtenção de variáveis

A primeira solução pensada seria utilizar a API Robot Web Services descrita em 4.3.1, entretanto tal API foi disponibilizada para os controladores da ABB com RobotWare a partir da versão 6.0, sendo que a versão do RobotWare do controlador ICR5 do robô presente no laboratório é a RobotWare 5.13.0. Como para atualizar a versão do software uma atualização de hardware também era necessária, a solução mais direta foi descartada. Dessa forma, a segunda solução descrita em 4.3.2 foi implementada.

Utilizando como base o módulo de *socket* TCP/IP descrito no manual (ABB, 2004), criou-se um socket cliente dentro de uma sub-rotina para módulos RAPID a fim de trocar mensagens com um servidor TCP/IP rodando em outra máquina (PC ou notebook). Essa subrotina deve ser incluída nos módulos dos programas RAPID a serem executados a fim de habilitar o monitoramento via TCP/IP.

Ao se aplicar o fluxo de dados durante a execução de programas RAPID no robô, atribuiu-se um período de 500 ms (meio segundo no código RAPID) para as interrupções referentes à comunicação. A instrução "ITimer"estabelece uma interrupção cíclica com um intervalo de tempo pré-estabelecido, enquanto uma outra variável serve de "link"entre a interrupção e a medição de tempo, no caso do bloco de código desenvolvido, apresentado no apêndice A (no arquivo "main_start.txt"), essa variável é a "timeint".

Na subseção a seguir, uma estimativa da resolução das medidas de posição para o *Digital Twin* será realizada. As variáveis coletadas durante a execução de programas RAPID e os tópicos MQTT correspondentes serão detalhados na subseção 5.2.3.

5.1.1 Cálculo da resolução da medida de posição

Baseando-se no tempo de 500 ms base entre a coleta de dados para a medida de posição do robô, ao se executar a função RAPID "MaxRobSpeed", que retorna a velocidade máxima de acordo com a ferramenta atual, configuração de juntas do robô e **wobjdata**, é

possível determinar a resolução máxima do *Digital Twin* no pior dos casos.

Como a velocidade máxima do robô obtida ao se executar a função mencionada é de 5000 mm/s, calcula-se que a resolução máxima (em máxima velocidade linear do TCP) para a movimentação do robô com uma amostragem de 2 atualizações de dados por segundo é de 2500 mm, ou seja, o robô poderia se mover até 2,5 metros na velocidade linear máxima entre uma coleta de variáveis e outra. No entanto, tal velocidade é impraticável para movimentos com ajuste fino na manufatura aditiva, sendo assim a resolução dos movimentos média observada nos testes está na ordem de milímetros entre um ciclo de atualização e outro. De fato, trabalhos anteriores que abordam o processo GTAW na literatura (SCHWEDERSKY et al., 2011; MENDEZ; NIECE; EAGAR, T., 2000) apontam que, para intensidades de corrente comumente utilizadas no processo de deposição abordado (até 250 A), há a ocorrência de defeitos na solidificação da poça de fusão - como o efeito *humping*, por exemplo - para velocidades de movimento acima de 1 m/min (ou 16,67 mm/s). Sendo assim, para tal valor de velocidade máxima prevista na deposição GTAW, a resolução do Gêmeo Digital desenvolvido para a movimentação linear do robô está na ordem de 8,33 mm durante a fabricação de uma peça.

É importante ressaltar que a resolução da medida está diretamente atrelada à velocidade de deslocamento durante a deposição. Dessa forma, para velocidades mais baixas tipicamente utilizadas no processo, como por exemplo 8 mm/s, a menor medida (resolução) entre ciclos também será menor, no exemplo citado sendo 4 mm a cada ciclo.

5.2 Adaptador com servidor TCP/IP e *broker*/*publisher* MQTT

A partir da metodologia descrita em 4.4, um software modular foi desenvolvido para estabelecer a comunicação via *socket* TCP/IP com o controlador ICR5 do robô ABB IRB 2600 e habilita o fluxo de dados atrabés do *broker*/*publisher* MQTT.

Nas subseções a seguir, cada um dos módulos presentes no adaptador será detalhado. Todos os algoritmos referentes a esta seção estão presentes no apêndice B. O fluxograma de execução detalhado do adaptador MQTT como um todo encontra-se no apêndice D.

5.2.1 *Broker* MQTT

Correspondendo ao módulo mais simples do adaptador, o *broker* MQTT estabelece a ponte de comunicação entre o *publisher* embutido no adaptador e os potenciais *subscribers*, que no caso deste projeto estão no fluxo Node-RED descrito na seção 5.3.

O *broker* faz uso do handler MQTT *Aedes*, cuja documentação está disponível em (MOSCAJS, 2023), o *broker* é estabelecido através do handler em um servidor local na porta

1883, porta padrão para aplicações MQTT. A partir desse *broker* então, tópicos podem ser criados para conectar mensagens de *publishers* a *subscribers*.

5.2.2 *Socket* servidor TCP/IP e *publisher* MQTT

Um servidor foi criado em um *socket* TCP/IP habilitando a troca de mensagens. A fim de evitar dessincronização entre o controlador e o adaptador MQTT, além de agilizar a obtenção de dados (por evitar que informação seja perdida em cada ciclo), foi estabelecida uma estrutura de "handshake"na troca de mensagens. Dessa forma, o servidor TCP/IP fica na espera de mensagens enviadas pelo cliente no código RAPID executado pelo controlador, e, após cada mensagem enviada pelo último, este aguarda a confirmação do servidor TCP/IP através de uma mensagem "PONG".

O *socket* em questão está disponibilizado em um servidor local na porta 1884 por padrão. Após a obtenção das variáveis de execução do controlador, as mensagens são processadas e estruturadas em tópicos, enviadas pelo mesmo módulo por meio de um *publisher* conectado ao *broker* na porta 1883. Os tópicos das mensagens publicadas assim como as variáveis contidas nelas são detalhados na subseção 5.2.3.

5.2.3 Tópicos *publisher* MQTT e variáveis

A fim de relacionar as variáveis coletadas nos módulos *socket* dos programas RAPID executados com seus tópicos MQTT correspondentes, observa-se a tabela 5.1.

Variável	Método de coleta	Descrição da variável	Tópicos correspondentes
Status	Ao se iniciar o módulo *socket* em um programa RAPID, ele envia para o servidor do adaptador via *socket* TCP/IP a mensagem de "Status;ONLINE"; ao se encerrar o módulo *socket*, a mensagem "Status;OFFLINE"é enviada.	Descreve se o adaptador está online e conectado ao controlador ICR5.	"abb/irb2600/status"
Mode	O controlador sempre envia uma mensagem via *socket* TCP/IP com o conteúdo "auto", uma vez que o módulo *socket* só funciona no modo de execução automática do controlador.	Modo de execução atual do controlador ICR5 (manual ou automática).	"abb/irb2600/mode"
Execution	Nome do programa definido no código RAPID.	Nome do programa sendo executado.	"abb/irb2600/execution"
Tool	Função RAPID CTool() que retorna as variáveis da ferramenta atual.	Descreve a ferramenta de trabalho atual: deslocamento entre o acoplador e o TCP, orientação e inércia se necessário.	"abb/irb2600/tool"
Cwobj	Função RAPID CWobj() que retorna as variáveis do objeto de trabalho atual.	"Current work object", diz respeito às coordenadas atuais de trabalho sobre as quais o robô calcula as coordenadas de movimentação global	"abb/irb2600/cwobj"
Speed	É definido um sinal de saída de sistema no controlador ICR5 no qual a variável de sistema "Tcp Speed"pode ser obtida.	Velocidade atual do TCP.	"abb/irb2600/speed"
Mspeed	Função RAPID MaxRobSpeed().	Velocidade máxima linear do robô em mm/s.	"abb/irb2600/mspeed"
RSpeed	Obtida através da função RAPID CSpeedOverride().	Velocidade de rotação da mesa posicionadora em porcentagem.	"abb/irb2600/rspeed"

Welding	Obtido através do sinal I/O de monitoramento da fonte Fronius pelo controlador ICR5 "doFr1ArcOn".	Define os valores "TRUE"ou "FALSE"para o estado atual do processo de deposição, em termos de arco presente ou não.	"abb/irb2600/welding"
WVoltage	Obtido através do sinal I/O de monitoramento da fonte Fronius pelo controlador ICR5 "aiFr1WeldingVoltage".	Tensão de soldagem atual em volts.	"abb/irb2600/wvoltage"
WCurrent	Obtido através do sinal I/O de monitoramento da fonte Fronius pelo controlador ICR5 "aoFr1Power".	Corrente atual de soldagem em amperes.	"abb/irb2600/wcurrent"
WSpeed	Obtido através do sinal I/O de monitoramento da fonte Fronius pelo controlador ICR5 "aoFr1WireSpeedWfi".	Velocidade atual de alimentação do arame em mm/s.	"abb/irb2600/wspeed"
PosRotation	Rotação dos eixos externos do robô obtidos do atributo "extax"da saída da função RAPID CRobT() "current robot target".	Rotação dos eixos externos do robô (rotação da mesa posicionadora em graus).	"abb/irb2600/posrotation"
CRobT	Atributos translacionais ("trans") e rotacionais ("rot") do "target"atual do robô obtidos através da função RAPID CRobT().	Variáveis de posição ("target") atual do robô, coordenadas cartesianas e orientação em forma de quaternion.	"abb/irb2600/crobt" "abb/irb2600/x_pos" "abb/irb2600/y_pos" "abb/irb2600/z_pos" "abb/irb2600/rorient"
Timestamp	Obtidos através das funções RAPID CDate() e CTime()	Data e hora atual para referência do fim do ciclo de coleta de variáveis pelo controlador.	"abb/irb2600/timestamp"

Tabela 5.1 – Variáveis coletadas no código RAPID e tópicos MQTT correspondentes

É notável que não há uma correspondência exata entra variáveis e tópicos tanto em nomenclatura quanto em número, isso ocorre pois há uma organização diferente entre a coleta de variáveis e sua posterior passagem pelo fluxo de dados MQTT no Node-RED. Essa decisão de projeto se deu a fim de melhor separar e estruturar as informações coletadas no módulo adaptador com o objetivo de serem obtidas mais facilmente nas aplicações de visualização no final do fluxo (*dashboard*, simulação 3D, geração de gráficos).

5.3 Fluxo e dashboard de monitoramento Node-RED

Conforme a metodologia do projeto descrita, criou-se um fluxo Node-RED como intermediário entre o software adaptador e o *dashboard* no próprio Node-RED e banco de dados na "nuvem". A seguir, serão mostrados o fluxo e o *dashboard* elaborado para o robô ABB IRB 2600 no Node-RED, bem como um destaque para o cálculo de latência entre mensagens recebidas pelos *subscribers* nos tópicos MQTT.

5.3.1 Fluxo e *dashboard* no Node-RED

A principal forma de visualização de dados coletados no fluxo de dados MQTT é o *dashboard* estabelecido no framework Node-RED. Tal *dashboard* se apoia em um fluxo de blocos de código que processam as mensagens obtidas nos tópicos subscritos via MQTT para organizar as informações de forma a estabelecer uma visualização concisa e organização para publicação de dados na "nuvem". Tal fluxo pode ser visto na figura 5.15.

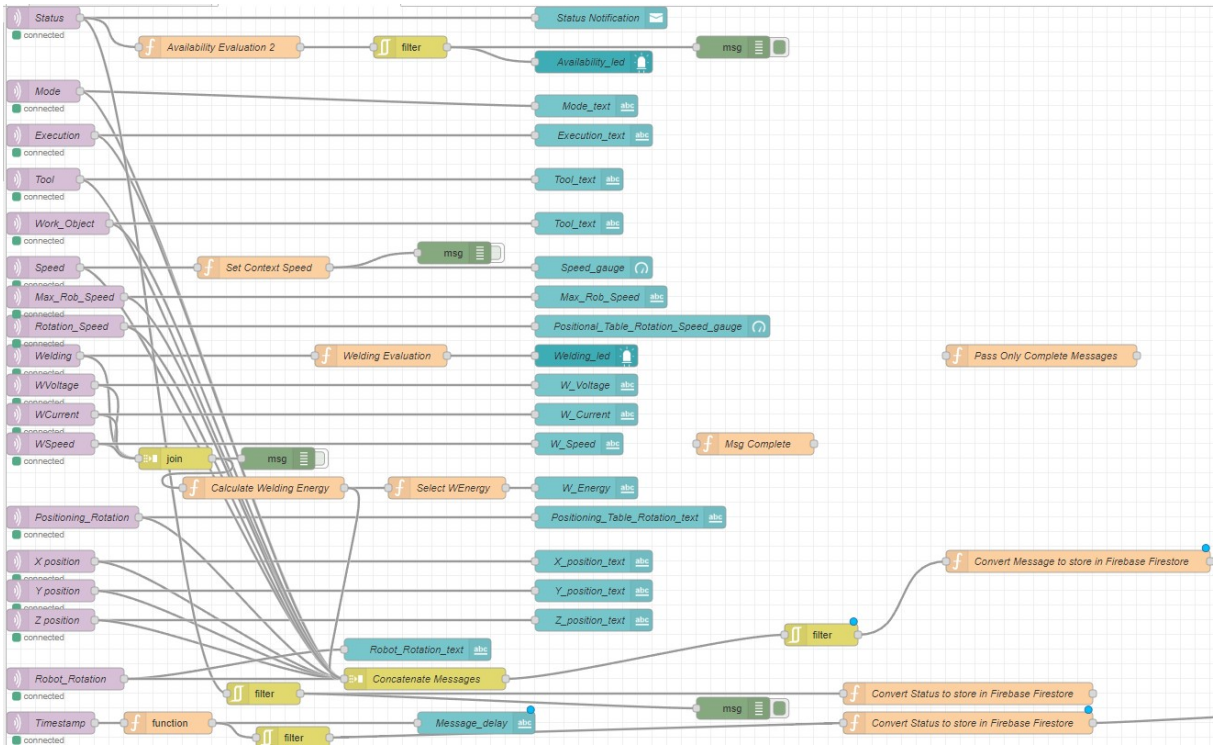

Figura 5.15 – Fluxo Node-RED para processamento e estruturação de mensagens MQTT para *dashboard* e publicação na "nuvem"Firebase Firestore.

Como observado, o fluxo de processamento de mensagens MQTT no Node-RED é composto por diversos blocos de código que são executados a cada mensagem recebida pelos tópicos subscritos MQTT. No final do fluxo há um bloco de função responsável por publicar os documentos na "nuvem"Firebase Firestore, detalhados na seção 5.5.

Além disso, no framework Node-RED também implementou-se um *dashboard* de visualização de variáveis dos tópicos subscritos que permitem a visualização dos dados instantâneos coletados. Destaca-se o *dashboard* na figura 5.16.

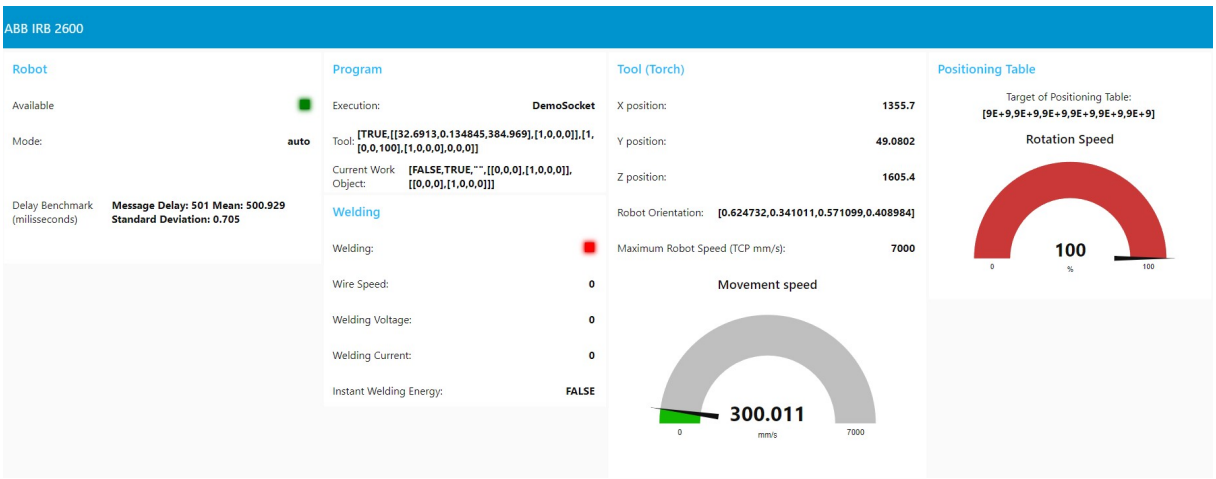

Figura 5.16 – *Dashboard* de monitoramento de informações coletadas via fluxo MQTT no Node-RED.

Como destacado, é possível observar diversas informações, dentre elas "status"do adaptador, variáveis de soldagem, posição e velocidade do TCP e mesa posicionadora, pro-

grama sendo executado, etc. Tais informações permitem uma visão geral do estado atual da célula de manufatura aditiva robótica, que juntamente com a simulação de espelhamento de movimentação detalhada em 5.4 compõem o *Digital Twin* projetado.

5.3.2 Benchmark de latência entre mensagens

Como descrito na seção 5.1, estabeleceram-se interrupções cíclicas de 500 milissegundos para coleta de dados no código RAPID para a comunicação *Socket*.

Sendo assim, entre duas mensagens de um mesmo tópico há 500 ms de tempo de latência base somados com o tempo de processamento do programa RAPID e tempo de envio de mensagens de outros tópicos em uma mesma iteração do módulo do *Socket*.

A fim de realizar um *benchmark* da comunicação entre o software adaptador e o robô, criou-se uma seção no *dashboard* para monitorar o tempo médio de atraso (*delay*) entre duas mensagens de um mesmo tópico, além do desvio padrão para monitorar se há flutuações consideráveis na eficiência da comunicação. Tal indicador pode ser observado enquanto o programa é executado juntamente com as outras variáveis disponibilizadas no *dashboard* Node-RED. Além disso, há a possibilidade da medição do atraso entre a publicação dos documentos correspondentes às mensagens na "nuvem", habilitando uma posterior análise em forma de gráfico das latências envolvidas no fluxo de dados.

Destaca-se o indicador do *delay* entre mensagens na figura 5.17.

Figura 5.17 – Painel de monitoramento do robô com destaque para o benchmark do delay entre mensagens recebidas pelos *subscribers* Node-RED MQTT.

5.4 Simulação 3D de espelhamento da movimentação do robô

Como resultado da simulação do modelo 3D do ativo físico, tem-se a simulação da célula de manufatura robotizada formada pelo braço robótico ABB IRB 2600 e pela mesa posicionadora para soldagem ABB IRBP A250. Tal célula é representada na figura 5.18.

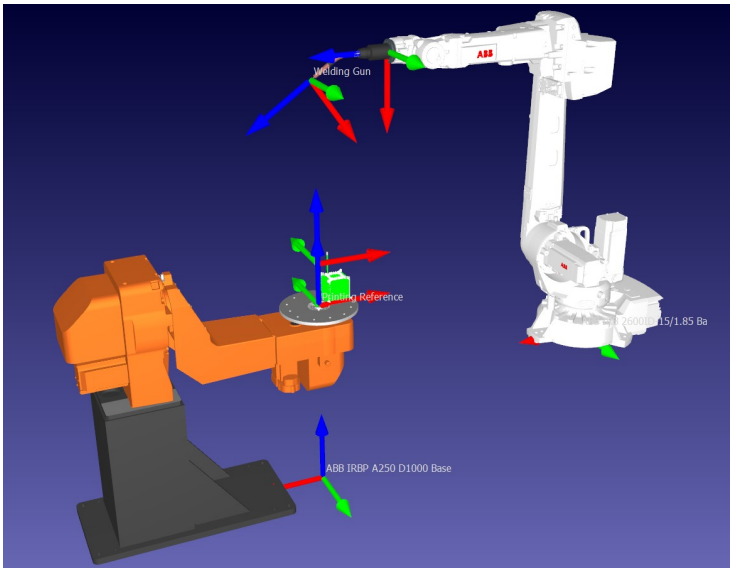

Figura 5.18 – Modelo 3D da célula de manufatura aditiva a ser simulada.

Nas subseções a seguir, serão explicadas as possibilidades de simulação com o modelo 3D no RoboDK. Primeiramente uma simulação prévia a uma manufatura será demonstrada, na segunda subseção a simulação se dará através dos dados coletados por um programa executado no RoboDK inscrito nos tópicos MQTT do fluxo de dados explicado anteriormente.

5.4.1 Simulação impressão peça fatiada via Slic3r

Além de compor o modelo base da movimentação do *Digital Twin* da célula de manufatura em questão, é possível simular o feitio de uma peça de maneira prévia à sua execução pelo robô. Dessa forma, tem-se uma ideia de como a peça ficará após a fabricação. Na figura 5.18 há uma demonstração dessa simulação.

Tal simulação é possível após instalar o módulo do programa de fatiamento de modelos 3D Slic3r, o qual realiza a conversão da peça 3D para um caminho a ser percorrido pela ferramenta, neste caso uma tocha para soldagem controlada pelo braço robótico em questão. Ao carregar a peça para o software RoboDK, então é possível configurar parâmetros para a simulação da impressão que será realizada.

Por fim, tem-se um pequeno vídeo demonstrativo da simulação e os passos necessários para tal no link https://youtu.be/E-wuVhRHnC8 (A. CABRAL J. V. E. F. D. O., L., 2022).

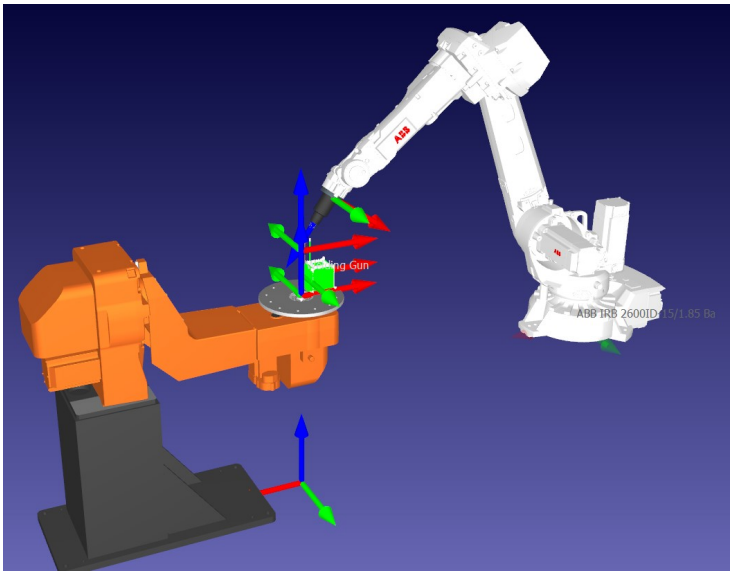

Figura 5.19 – Simulação da impressão 3D de um cubo.

5.4.2 Simulação de movimentação espelhada do *Digital Twin*

Aplicando o planejamento descrito em 4.6, conectou-se o modelo de simulação 3D no RoboDK com o fluxo de dados MQTT através de um algoritmo em Python.

Por estar subscrito nos tópicos de mensagens correspondentes à movimentação e deposição de material no fluxo MQTT, a cada atualização do ciclo de mensagens publicadas no *broker* MQTT o algoritmo no RoboDK interpreta essas variáveis e converte as informações obtidas em "targets" de movimentação. Desses targets, o RoboDK então faz a movimentação de juntas para alcançar o ponto de localização atual do TCP do robô.

Devido à movimentação do robô real sendo monitorada uma vez a cada 500 milissegundos, quando o robô realiza movimentos rápidos a simulação no RoboDK dá a impressão de sofrer travamentos, a simulação só fica suave quando o robô realiza movimentos em baixa velocidade. Isso é claramente devido ao período de atualização de dados, entretanto, para o *Digital Twin* deste projeto, não foi possível estabelecer uma periodicidade menor sem atrapalhar o pleno funcionamento das tarefas executadas pelo controlador do robô e sincronização com o fluxo de dados através do adaptador MQTT.

Outra possibilidade da simulação no RoboDK é a deposição de material via macros disponibilizadas nos módulos Python do RoboDK "arcOn" e "sprayOn". Quando a variável de soldagem "drFr1ArcOn" do monitoramento de sinais do controlador ICR5 está com o valor "1", o tópico "abb/irb2600/welding" publica uma mensagem "TRUE" que então é interpretada pelo *subscriber* no algoritmo Python no RoboDK ativando a macro "arcOn". Dessa forma, o RoboDK simula a deposição de material em pequenas esferas que formam um rastro do caminho percorrido pelo TCP, simulando assim uma peça sendo fabricada.

Uma demonstração da simulação de movimentação do robô espelhando o ativo físico

pode ser vista no vídeo em https://youtu.be/ydOauQ8YwFM (A. CABRAL, J. V., 2023). No vídeo percebe-se o modelo 3D realizando os mesmos movimentos do robô real, caracterizando uma aplicação satisfatória do *Digital Twin* para monitoramento do ativo. Na figura 5.20 evidencia-se um print de tal simulação.

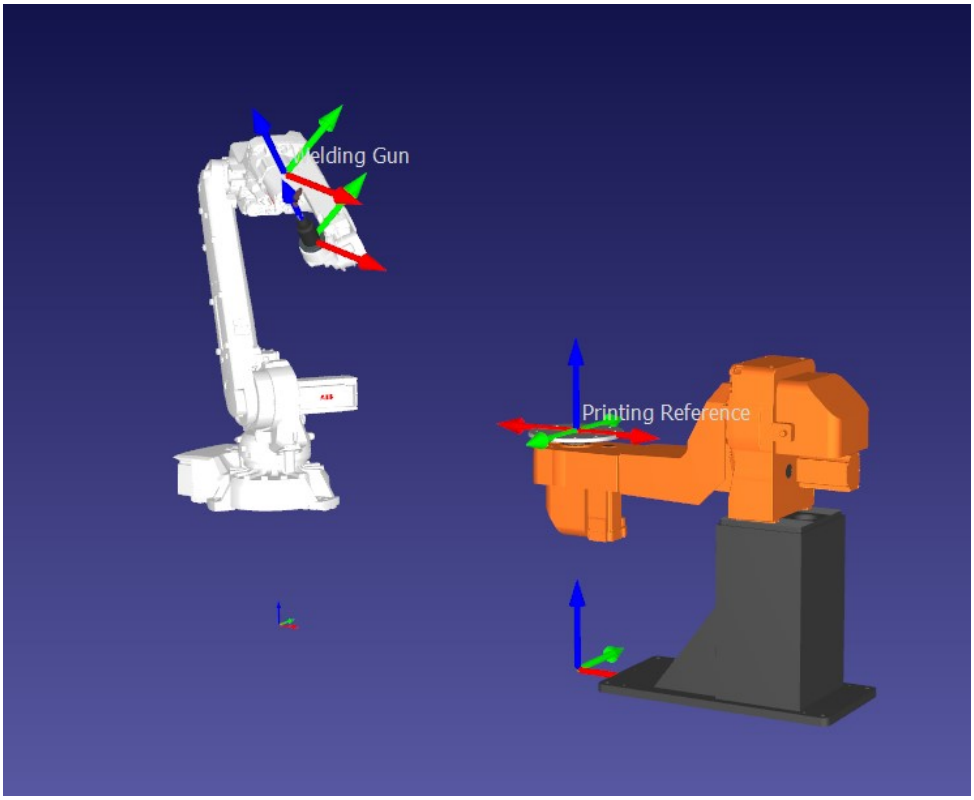

Figura 5.20 – Demonstração de movimentação espelhada do *Digital Twin* 3D da célula de manufatura.

Outro problema evidenciado na movimentação do robô é a possível dessincronização entre a movimentação do modelo 3D e o ativo real, isso ocorre por causa das diferenças entre os modelos de cinemática do RoboDK e o modelo do controlador ICR5 do robô. Em um trabalho futuro isso poderia ser corrigido criando um modelo de cinemática equivalente ao do ABB IRB 2600 através de uma modelagem via React Three.js, no entanto neste projeto tal abordagem seria muito trabalhosa para o pouco tempo de desenvolvimento, por isso a abordagem mais simples utilizando o RoboDK foi escolhida. Entretanto, para movimentações espelhando movimentos precisos em "work objects" em cima da mesa posicionadora, essa dessincronização é mínima, por exemplo quando uma peça é fabricada.

Na simulação 3D, a deposição de material foi habilitada somente para dados simulados (por meio de um controlador virtual no software RobotStudio), pois ao longo do desenvolvimento desse projeto a fonte Fronius estava passando por manutenções e seus parâmetros de soldagem ainda precisavam ser determinados (o que foge do escopo deste projeto). Alguns exemplos de simulação de deposição de material com dados coletados do controlador virtual podem ser vistos nas figuras 5.21 e 5.22.

O código em Python referente à simulação detalhada nesta seção está presente no apêndice C.

5.5 Dados publicados no Firebase Firestore

No fluxo Node-RED, estabeleceu-se uma divisão de tópicos que serão atrelados aos documentos publicados no Firebase Firestore. Foi criada uma coleção para cada estrutura de documento com tópicos a serem publicados. Essas coleções com os tópicos relacionados a cada documento são enumerados a seguir (cada item corresponde a uma coleção no Firebase Firestore enquanto cada subitem corresponde a um tópico dentro de cada documento da coleção).

- ***"status_irb_2600"***: coleção cujos documentos armazenam o status do adaptador MQTT, ONLINE ou OFFLINE;
  - ***"abb/irb2600/status"***: *string*. Valor "ONLINE"ou "OFFLINE".
- ***abb_irb_2600***: coleção cujos documentos armazenam dados gerais coletados do controlador ICR5 através do adaptador MQTT. Esses dados são utilizados para gerar gráficos e poderiam ser utilizados também para a criação de *dashboards* online;
  - ***"abb/irb2600/cwobj"***: *string*. Tópico (atributo) do documento de dados coletados referente às coordenadas do "current work object", referência das coordenadas x, y, z em relação às coordenadas globais do robô (coordenadas do objeto de trabalho na mesa posicionadora);
  - ***"abb/irb2600/execution"***: *string*. Tópico do documento referente ao nome do programa sendo executado;
  - ***"abb/irb2600/mode"***: *string*. Tópico do documento de dados coletados referente ao modo de execução atual do controlador do robô. Como o programa com o *socket* que comunica com o *Digital Twin* é executado somente no modo automático do controlador, este tópico sempre estará com o valor "auto"(automático);
  - ***"abb/irb2600/mspeed"***: *string*. Este tópico faz referência à velocidade linear máxima que o robô pode atingir (em mm/s);
  - ***"abb/irb2600/posrotation"***: *string*. Se refere às posições dos eixos externos do robô, no caso da célula ropótica monitorada há apenas dois eixos referente aos ângulos de rotação da mesa posicionadora;
  - ***"abb/irb2600/rorient"***: *string*. Se refere à orientação do TCP em formato de quaternion, faz parte da estrutura que forma a posição atual do robô;

  - ***"abb/irb2600/rspeed"***: *string*. Se refere à porcentagem atual da velocidade máxima do eixo externo (os valores variam de 0 a 100, o programa RAPID comanda a porcentagem de velocidade a ser interpretada pelo eixo externo);
  - ***"abb/irb2600/speed"***: *string*. Velocidade linear do TCP em mm/s;
  - ***"abb/irb2600/tool"***: *string*. Dados da ferramenta carregada pelo eixo acoplador do robô, se refere ao deslocamento do vetor que parte do eixo acoplador até a ponta da ferramenta (onde fica o TCP), assim como a rotação em formato de quaternion e informações de inércia/cinemática da ferramenta acoplada;
  - ***"abb/irb2600/wvariables"***: coleção de atributos referentes às variáveis de soldagem coletadas da fonte Fronius;
    * ***"abb/irb2600/wcurrent"***: *string*. Corrente de soldagem atual;
    * ***"abb/irb2600/welding"***: *string*. Variável que indica se a fonte de soldagem está ativa ou não;
    * ***"abb/irb2600/wenergy"***: *string*. Energia de soldagem instantânea calculada segundo a fórmula apresentada na subseção 4.3.3;
    * ***"abb/irb2600/wspeed"***: *string*. Variável referente à velocidade de alimentação do arame de solda;
    * ***"abb/irb2600/wvoltage"***: *string*. Tensão atual no arco elétrico de solda;
  - ***"abb/irb2600/x_pos"***: *string*. Valor referente à posição atual do TCP no eixo *x*;
  - ***"abb/irb2600/y_pos"***: *string*. Valor referente à posição atual do TCP no eixo *y*;
  - ***"abb/irb2600/z_pos"***: *string*. Valor referente à posição atual do TCP no eixo *z*;
  - ***"abb/irb2600/timestamp"***: *string*. Valor de referência do *timestamp* no momento de publicação das mensagens aos tópicos MQTT.

- ***"delay_abb_irb_2600"***: coleção cujos documentos armazenam dados referentes ao cálculo do tempo gasto entre o envio de mensagens para o *subscribers* MQTT.
  - ***"abb/irb2600/delay"***: *float*. Valor do tempo gasto (em ms) entre a última mensagem e a mensagem atual;
  - ***"abb/irb2600/delay_mean"***: *float*. Média do valor do tempo gasto entre mensagens da sessão atual do *subscriber* MQTT executado no Node-RED;
  - ***"abb/irb2600/delay_std_deviation"***: *float*. Desvio-padrão do valor do tempo gasto entre mensagens da sessão atual do *subscriber* MQTT executado no Node-RED.

Muitas variáveis numéricas são armazenadas no Firebase Firestore como *string*, pois uma limitação do protocolo de comunicação MQTT é a possibilidade de publicar somente dados embutidos em *strings*, entretanto para manipulações numéricas a conversão é facilmente realizada utilizando-se qualquer linguagem de programação.

É importante frisar que todos os documentos contêm a "timestamp"do momento em que foram criados, sendo possível relacionar o momento da execução do programa RAPID com os estados do *Digital Twin* monitorados e publicados em *cloud*. Dessa forma, é possível realizar uma análise temporal em cima dos dados e geração de gráficos.

5.5.1 Gráficos gerados com dados em *cloud*

Para exemplificar a possibilidade da aplicação de dados coletados na análise e visualização de informações acerca da manufatura aditiva metálica espelhada pelo *Digital Twin*, gráficos foram gerados com base nos dados publicados no banco de dados na "nuvem"Firebase Firestore.

O primeiro gráfico mostrado na figura 5.21 relaciona a altura Z do TCP na execução de um programa RAPID em um controlador virtual simulado com a ativação do arco de soldagem através do sinal de saída "doFr1ArcOn"da fonte de soldagem Fronius MW5000 (sinais também simulados).

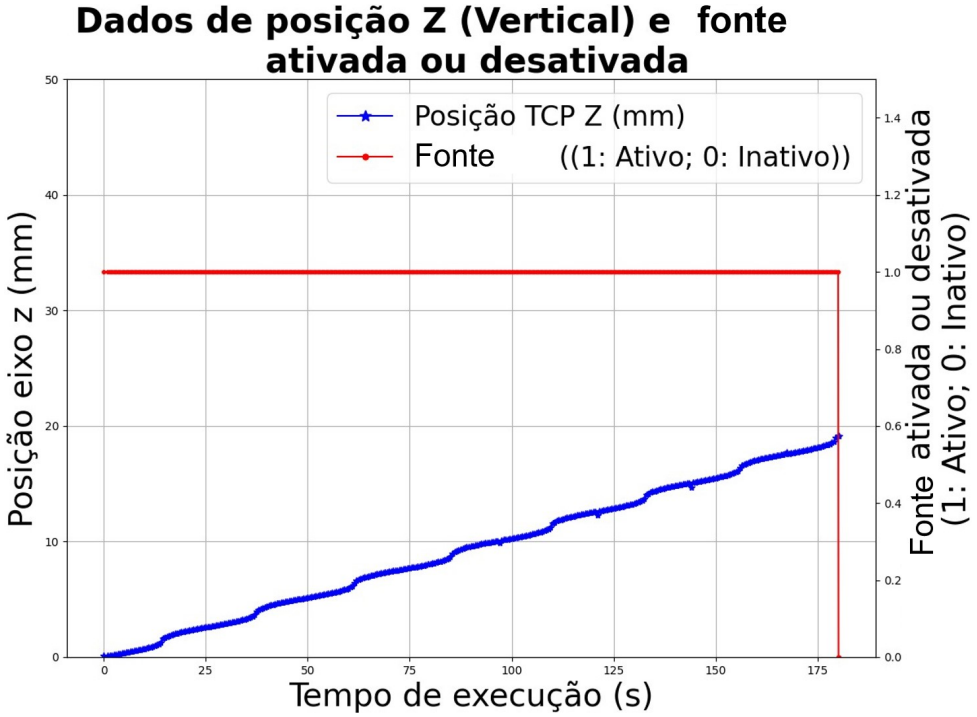

Figura 5.21 – Simulação de ativação da fonte de soldagem de acordo com a altura Z do TCP.

Uma análise que poderia partir desse gráfico seria identificar se o distanciamento entre camadas programado nas estratégias de deposição está sendo respeitado. Também é possível estimar o tempo gasto na deposição de cada camada de material. Outra análise seria a possibilidade de correlacionar defeitos na peça final fabricada com o programa RAPID analisando o gráfico de deposição de material gerado.

Como a estratégia de fatiamento utilizada nos programas RAPID executados nos testes do *Digital Twin* se baseia em uma deposição contínua helicoidal entre dois planos, outro gráfico interessante é o de posição X e Y ao longo do tempo *t*. Tal gráfico evidenciado na figura 5.22 permite ter uma ideia da seção transversal de cada camada de material depositado (pois cada camada começa e termina no mesmo ponto no início da deposição de material).

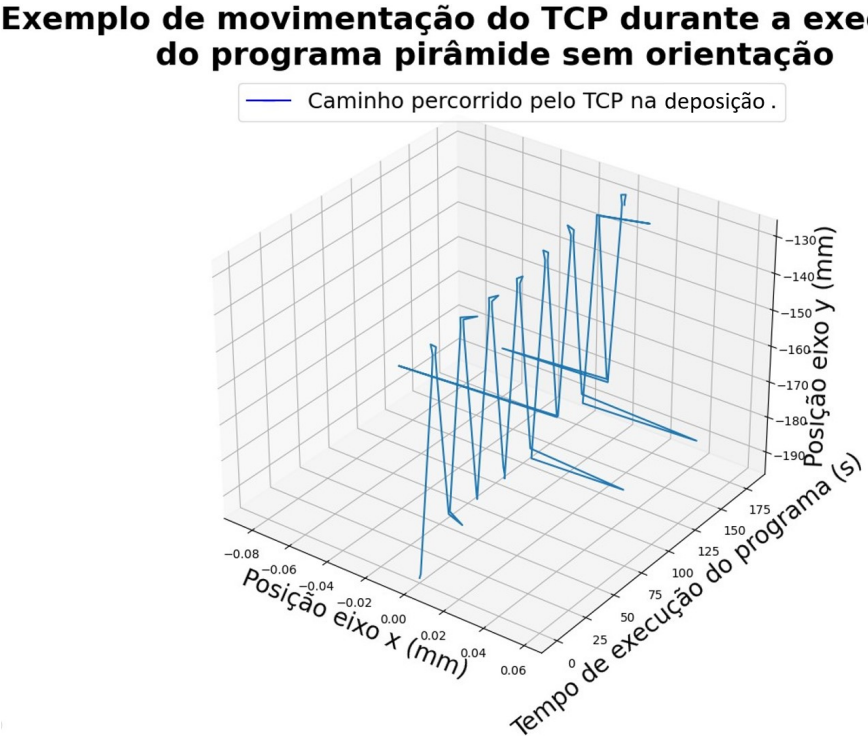

Figura 5.22 – Simulação de movimentação X e Y do TCP ao longo do tempo.

Todos os gráficos mostrados nessa seção foram gerados com dados coletados de um controlador ICR5 virtual utilizando-se do software RobotStudio 2022 da ABB. Tais simulações foram realizadas, pois a fonte Fronius juntamente com a ponta de solda no acoplador do robô estavam passando por manutenções durante o desenvolvimento deste trabalho.

5.6 Algoritmo de inserção de módulo *socket* em programas RAPID

A fim de se facilitar a adaptação de programas RAPID para habilitar a comunicação *socket* necessária para o *Digital Twin*, desenvolveu-se um programa em Python ("insert_socket_to_rapid.py") que tem como entrada um programa RAPID para controle do robô e na saída o algoritmo cria um novo arquivo copiando o programa original e adicionando as rotinas e variáveis necessárias para a comunicação com o adaptador Ethernet. O algoritmo funciona realizando os seguintes passos:

1. Realiza a leitura do arquivo de código RAPID original;

2. Lê os arquivos .txt com os módulos e arquivos;
3. Cria uma lista de linhas para o arquivo de saída;
4. Identifica o início do programa RAPID e adiciona as linhas do arquivo "variables.txt"à lista;
5. Identifica o início da função "main", adiciona as linhas dos arquivos "socket_management.txt", "send_data.txt", "end_socket.txt"antes da função "main". Adiciona as linhas do arquivo "main_start.txt"no início da "main";
6. Identifica o final da rotina "main"e adiciona as linhas do arquivo "main_end.txt"antes do final da "main";
7. Gera um arquivo de saída com o módulo *socket* habilitado.

Tais arquivos comentados encontram-se nas diversas seções do apêndice A. Além disso o fluxograma detalhado de execução do programa se encontra no apêndice E.

6 Conclusão

Neste trabalho apresentou-se uma aplicação de Gêmeo Digital baseado na ISO 23247 que cumpriu com os requisitos de projeto planejados em 1.1. O fluxo de dados implementado, utilizando-se dos protocolos Ethernet TCP/IP e MQTT, se mostrou eficiente na coleta de dados através do controlador ICR5 da célula de manufatura robótica composta pelo braço robótico ABB IRB 2600 e pela mesa posicionadora IRBP A250. Além disso, a simulação 3D espelhando a movimentação do ativo real demonstrou um bom sincronismo e precisão nos movimentos, sendo fiel ao propósito de um Gêmeo Digital.

Em relação ao *dashboard* Node-RED, tal aplicação também demonstrou com sucesso uma forma de visualização das informações coletadas das variáveis de movimentação e de soldagem da célula de manufatura aditiva robótica. Por fim, o programa para adaptar algoritmos RAPID para a inclusão do módulo *socket* para comunicação com o adaptador MQTT também se mostrou eficiente, facilitando a aplicação do Gêmeo Digital implementado e o desenvolvimento de aplicações futuras com códigos RAPID escritos em projetos futuros.

Por fim, os objetivos do presente trabalho foram alcançados através da implementação de um *Digital Twin* da célula de manufatura aditiva composta pelo robô ABB IRB 2600 e pela mesa posicionadora IRBP A250; a eficácia da implementação é evidenciada nos resultados apresentados no vídeo em (A. CABRAL, J. V., 2023), o que ressalta o sucesso no desenvolvimento do projeto. Há melhorias que podem ser feitas em relação ao *Digital Twin* e a coleta de dados que são comentadas na seção 6.1. Os códigos desenvolvidos ao longo deste trabalho também estão disponíveis no GitHub (os arquivos podem sofrer atualizações futuras) no link https://github.com/MASCAM/ABB_IRB_2600_Digital_Twin (CABRAL, 2023).

6.1 Trabalhos futuros

Futuramente pretende-se aprimorar a simulação 3D de espelhamento de movimentação através da construção de um modelo de cinemática via WebGL/React Three.js, dessa forma a movimentação do robô poderá ser mais precisa e mais sincronizada com a realidade.

Outro aspecto que pode ser melhorado é o monitoramento via sensores das camadas de material depositado na manufatura aditiva metálica, dessa forma ocorreria uma validação das camadas depositadas da peça a ser fabricada no monitoramento.

Além disso, é possível melhorar e testar taxas de amostragem maiores para aumentar a resolução das variáveis coletadas. Caso o software RobotWare do controlador for atualizado para a versão 6 ou superior, adotar o Robot Web Services (RWS) no *Digital Twin* seria possível,

o que disponibilizaria mais variáveis para serem monitoradas, além de melhorar o fluxo de coleta de dados possibilitando inclusive taxas maiores de atualização desejadas.

Outra melhoria pensada é a disponibilização de um *dashboard* com *host* na web, dessa forma qualquer pessoa com acesso ao *host* de qualquer lugar do mundo poderia visualizar o *Digital Twin*. Dessa forma, aplicações de laboratório/ensino remoto utilizando o DT do ABB IRB 2600 poderiam ser habilitadas.

Uma proposta para ampliar o escopo do *Digital Twin* seria transformar todo o fluxo em um sistema ciber-físico, aprimorando o módulo adaptador para enviar comandos de volta ao robô de forma a controlá-lo conforme as informações coletadas. Idealmente o RWS facilitaria essa tarefa, porém seria possível implementar a volta de controle também adaptando a solução *socket* TCP/IP apresentada neste trabalho.

Por fim, idealmente seria interessante embarcar algum computador ou dispositivo (Raspberry Pi, por exemplo) junto ao controlador do robô, de forma que quando ligado as variáveis sejam continuamente monitoradas sem a necessidade de conectar um computador a todo momento ao controlador. Entretanto, essa solução necessitaria da disponibilidade do *framework* Robot Web Services mencionado anteriormente.

Referências

ABB, R. P. **Application manual. Robot communication and I/O control. RobotWare 5.15.01**. SE-721, 68, Västerås, Sweden, 2004. Disponível em: <https://library.e.abb.com/public/c1e123190684963fc1257b4b00523470/3HAC024534-001_rev1_en_library.pdf>. Acesso em: 28 ago. 2023. Citado nas pp. 38, 39, 42.

ABB, R. P. **Application Manual - Robot Web Services. RobotWare 7**. SE-721, 68, Västerås, Sweden, 2021. Disponível em: <https://developercenter.robotstudio.com/api/rwsApi/>. Acesso em: 29 ago. 2023. Citado na p. 38.

A. CABRAL, J. V. **Simulação de Espelhamento de Movimentação 3D do Robô ABB IRB 2600 via RoboDK**. https://youtu.be/ydOauQ8YwFM. Acesso em: 7 nov. 2023. Citado nas pp. 52, 58.

A. CABRAL JOÃO VÍTOR E F. DE OLIVEIRA, L. **Demonstração Simulação Deposição Material 3D ABB IRB 2600 RoboDK**. https://www.youtube.com/watch?v=E-wuVhRHnC8. Acesso em: 12 mai. 2022. Citado na p. 50.

ANDRADE, R. C. **Desenvolvimento de software de fatiamento de sólidos tipo casca e geração de trajetórias para fabricação de peças por deposição de metal em camadas sucessivas utilizando o processo GMAW**. Brasília, DF, 2013. P. 82. Trabalho de Graduação em Engenharia de Controle e Automação, Publicação FT. TG-nº 20/2012, Faculdade de Tecnologia, Universidade de Brasília. Citado nas pp. 34–36.

ANTONINI, J. 8.04 - Health Effects Associated with Welding. In: HASHMI, S.; BATALHA, G. F.; VAN TYNE, C. J.; YILBAS, B. (Ed.). **Comprehensive Materials Processing**. Oxford: Elsevier, 2014. P. 49–70. ISBN 978-0-08-096533-8. DOI: https://doi.org/10.1016/B978-0-08-096532-1.00807-4. Disponível em: <https://www.sciencedirect.com/science/article/pii/B9780080965321008074>. Citado na p. 24.

BERNAL, V. B. **Tecnologia de Redes - Protocolo Ethernet**. https://www.lsi.usp.br/~volnys/courses/tecredes/pdf/05ETH-col.pdf. Acesso em: 11 mai. 2022. Citado na p. 19.

CABRAL, J. V. A. **ABB_IRB_2600_Digital_Twin**. https://github.com/MASCAM/ABB_IRB_2600_Digital_Twin. Acesso em: 15 dez. 2023. Citado na p. 58.

CABRAL, J. V. A.; GASCA, E. A. R.; ALVARES, A. J. Digital Twin Implementation for Machining Center Based on ISO 23247 Standard. **IEEE Latin America Transactions**, v. 21, n. 5, p. 628–635, abr. 2023. Disponível em: <https://latamt.ieeer9.org/index.php/transactions/article/view/7651>. Citado nas pp. 30, 31.

DUTRA, J.; SILVA, R.; MARQUES, C. Melting and Welding Power Characteristics of MIG CMT versus Conventional MIG for Aluminum 5183. **Soldagem Inspeção**, v. 18, p. 12–18, mar. 2013. DOI: 10.1590/S0104-92242013000100003. Citado na p. 25.

EAGAR, T. W. Energy Sources Used for Fusion Welding[1]. In: WELDING, BRAZING, AND SOLDERING. ASM International, jan. 1993. ISBN 978-1-62708-173-3. DOI: 10.31399/asm.hb.v06.a0001332. eprint: https://dl.asminternational.org/book/chapter-pdf/475430/a0001332.pdf. Disponível em: <https://doi.org/10.31399/asm.hb.v06.a0001332>. Citado na p. 40.

HENCKELL, P.; GIERTH, M.; ALI, Y.; REIMANN, J.; BERGMANN, J. P. Reduction of Energy Input in Wire Arc Additive Manufacturing (WAAM) with Gas Metal Arc Welding (GMAW). **Materials**, v. 13, n. 11, 2020. ISSN 1996-1944. DOI: 10.3390/ma13112491. Disponível em: <https://www.mdpi.com/1996-1944/13/11/2491>. Citado na p. 23.

ISO, I. 1. 4. **ISO 23247-1:2021 Automation systems and integration — Digital twin framework for manufacturing — Part 1: Overview and general principles, stage 60.60**. https://www.iso.org/standard/75066.html. Acesso em: 8 mai. 2022. Citado nas pp. 16, 18.

ISO, I. 1. 4. **ISO 23247-2:2021 Automation systems and integration — Digital twin framework for manufacturing — Part 2: Reference architecture, stage 60.60**. https://www.iso.org/standard/78743.html. Acesso em: 8 mai. 2022. Citado nas pp. 16, 18.

ISO, I. 1. 4. **ISO 23247-3:2021 Automation systems and integration — Digital twin framework for manufacturing — Part 3: Digital representation of manufacturing elements, stage 60.60**. https://www.iso.org/standard/78744.html. Acesso em: 8 mai. 2022. Citado nas pp. 16, 18.

ISO, I. 1. 4. **ISO 23247-4:2021 Automation systems and integration — Digital twin framework for manufacturing — Part 4: Information exchange, stage 60.60**. https://www.iso.org/standard/78745.html. Acesso em: 8 mai. 2022. Citado nas pp. 16, 18.

LIU, C.; LE ROUX, L.; KÖRNER, C.; TABASTE, O.; LACAN, F.; BIGOT, S. Digital Twin-enabled Collaborative Data Management for Metal Additive Manufacturing Systems. **Journal of Manufacturing Systems**, v. 62, p. 857–874, 2022. ISSN 0278-6125. DOI: https://doi.org/10.1016/j.jmsy.2020.05.010. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0278612520300741>. Citado nas pp. 27–29.

MENDEZ, P.; NIECE, K.; EAGAR, T. Humping Formation in High Current GTA Welding. **Proceedings of the International Conference on Joining of Advanced and Specialty Materials II**, jan. 2000. Citado na p. 43.

MOSCAJS. **Aedes**. https://github.com/moscajs/aedes. Acesso em: 20 set. 2023. Citado na p. 43.

MQTT, O. **MQTT: The Standard for IoT Messaging**. https://mqtt.org/. Acesso em: 11 mai. 2022. Citado nas pp. 20, 21.

MTCONNECT, I. **MTConnect Standard**. https://www.mtconnect.org/. Acesso em: 11 mai. 2022. Citado na p. 20.

PIJUSH KANTI DUTTA PRAMANIK, e. a. **A novel multi-brand robotic software interface for industrial additive manufacturing cells**. Nov. 2019. DOI: 10.1108/IR-11-2019-0237. Citado na p. 15.

SALES DE MATOS, B. **Desenvolvimento de software de geração de trajetórias para construção de modelos metálicos tipo casca por meio do processo GTAW com adição de arame frio**. 2022. F. 102. Projeto Final de Curso (Engenharia de Controle e Automação) – Universidade de Brasília, Brasília. Citado nas pp. 34, 36, 37.

SCHWEDERSKY, M. B.; DUTRA, J. C.; OKUYAMA, M. P.; SILVA, R. H. G. e. Soldagem TIG de elevada produtividade: influência dos gases de proteção na velocidade limite para formação de defeitos. **Soldagem Inspeção**, Associação Brasileira de Soldagem, v. 16, n. 4, p. 333–340, out. 2011. ISSN 0104-9224. DOI: 10.1590/S0104-92242011000400004. Disponível em: <https://doi.org/10.1590/S0104-92242011000400004>. Citado na p. 43.

SELCUK, C. 13 - Joining processes for powder metallurgy parts. In: CHANG, I.; ZHAO, Y. (Ed.). **Advances in Powder Metallurgy**. Woodhead Publishing, 2013. (Woodhead Publishing Series in Metals and Surface Engineering). P. 380–398. ISBN 978-0-85709-420-9. DOI: https://doi.org/10.1533/9780857098900.3.380. Disponível em: <https://www.sciencedirect.com/science/article/pii/B9780857094209500131>. Citado na p. 24.

SINGH, R. 3 - Welding and joining processes. In: SINGH, R. (Ed.). **Applied Welding Engineering (Third Edition)**. Third Edition: Butterworth-Heinemann, 2020. P. 157–186. ISBN 978-0-12-821348-3. DOI: https://doi.org/10.1016/B978-0-12-821348-3.00015-X. Disponível em: <https://www.sciencedirect.com/science/article/pii/B978012821348300015X>. Citado na p. 23.

WISHBOX. **DTUDO SOBRE IMPRESSÃO 3D: O QUE É, COMO FUNCIONA E TIPOS**. https://www.wishbox.net.br/blog/impressao-3d/. Acesso em: 11 mai. 2022. Citado nas pp. 22, 23.

WISHBOX. **AMBEV: REDUÇÃO NOS CUSTOS DE PRODUÇÃO COM IMPRESSORA 3D**. https://www.wishbox.net.br/blog/ambev-producao-com-impressora-3d/. Acesso em: 11 mai. 2022. Citado na p. 14.

ZHANG, Y.; GAO, M.; LU, Y.; DU, W. Deposition geometrical characteristics of wire arc additive-manufactured AA2219 aluminium alloy with cold metal transfer pulse advance arc mode. **The International Journal of Advanced Manufacturing Technology**, v. 123, n. 11, p. 3807–3818, dez. 2022. ISSN 1433-3015. DOI: 10.1007/s00170-022-10460-4. Disponível em: <https://doi.org/10.1007/s00170-022-10460-4>. Citado na p. 25.

ZHU, C. Y.; PIRES, J. N.; AZAR, A. A novel multi-brand robotic software interface for industrial additive manufacturing cells. **Industrial Robot: the international journal of robotics research and application**, Emerald Publishing Limited, v. 47, n. 4, p. 581–592, jan. 2020. ISSN 0143-991X. DOI: 10.1108/IR-11-2019-0237. Disponível em: <https://doi.org/10.1108/IR-11-2019-0237>. Citado na p. 29.

Apêndices

Apêndice A – Blocos de código incluídos nos programas RAPID

A.1 append_socket_to_rapid.py

Código A.1 – Código para inserir *socket* aos algoritmos RAPID

```
#João Vítor Arantes Cabral 17/0126048
#Trabalho de graduacao
#Módulo de inserção do módulo socket aos programas RAPID

import os
import json

#print(lines_to_add_dict)

FILEPATH = "./RAPID_PROGRAMS/"
FILENAME = "MainModule.mod"
OUTPUT_FILE = FILENAME.split(".mod")[0] + "_socket.mod"

#get_variables
def get_variables():

    #L   o arquivo de variáveis e adiciona-as a uma lista
    variables_to_add = open('./lines_to_add/variables.txt', "r",
       encoding='ISO-8859-1')
    variables = list(variables_to_add.readlines())
    variables_to_add.close()
    return variables

#get_send_data
def get_send_data():

    #L   o arquivo de envio de mensagens via socket e adiciona as
       linhas a uma lista
    send_data_to_add = open('./lines_to_add/send_data.txt', "r",
       encoding='ISO-8859-1')
    send_data = list(send_data_to_add.readlines())
    send_data_to_add.close()
    return send_data

#get_end_socket
def get_end_socket():

    #L   o arquivo de envio de encerramento do socket e adiciona
       as linhas a uma lista
```

```
36      end_socket_to_add = open('./lines_to_add/end_socket.txt', "r",
            encoding='ISO-8859-1')
37      end_socket = list(end_socket_to_add.readlines())
38      end_socket_to_add.close()
39      return end_socket
40
41  #get_main_start
42  def get_main_start():
43
44      #L   o arquivo de cabeçalho da função main e adiciona as
            linhas a uma lista
45      main_start_to_add = open('./lines_to_add/main_start.txt', "r",
            encoding='ISO-8859-1')
46      main_start = list(main_start_to_add.readlines())
47      main_start_to_add.close()
48      return main_start
49
50  #get_main_end
51  def get_main_end():
52
53      #L   o arquivo de final da função main e adiciona as linhas a
            uma lista
54      main_end_to_add = open('./lines_to_add/main_end.txt', "r",
            encoding='ISO-8859-1')
55      main_end = list(main_end_to_add.readlines())
56      main_end_to_add.close()
57      return main_end
58
59  #get_socket_management
60  def get_socket_management():
61
62      #L   o arquivo de inicialização do socket e adiciona as linhas
            a uma lista
63      socket_management_to_add =
            open('./lines_to_add/socket_management.txt', "r",
            encoding='ISO-8859-1')
64      socket_management = list(socket_management_to_add.readlines())
65      socket_management_to_add.close()
66      return socket_management
67
68  def add_lines(filepath, filename):
69
70      #Pega todas as linhas a serem adicionadas ao arquivo
71      variables = get_variables()
72      send_data = get_send_data()
73      end_socket = get_end_socket()
74      main_start = get_main_start()
75      main_end = get_main_end()
76      socket_management = get_socket_management()
77
78      #abre o arquivo RAPID e armazena as linhas em uma lista
79      file = open(FILEPATH + FILENAME, "r", encoding='ISO-8859-1')
```

```
80      file_lines = list(file.readlines())
81      file.close()
82
83      output_lines = []
84      main = False
85      append_line = True
86
87      #para cada linha do arquivo
88      for i in range(len(file_lines)):
89
90          append_line = True
91          #verifica o início do módulo Main e adiciona as variáveis
               ao arquivo
92          if "MODULEMainModule" in file_lines[i].replace(" ",
               "").replace("\t", ""):
93
94              output_lines.append(file_lines[i])
95              output_lines.extend(variables)
96              append_line = False
97
98          #verifica se o módulo main iniciou e adiciona as linhas de
               cabeçalho e procedures
99          elif "PROCmain()" in file_lines[i].replace(" ",
               "").replace("\t", ""):
100
101             output_lines.extend(send_data)
102             output_lines.extend(socket_management)
103             output_lines.extend(end_socket)
104             output_lines.append(file_lines[i])
105             output_lines.extend(main_start)
106             append_line = False
107             main = True
108
109         #verifica o fim da função main e adiciona as linhas de
               finalização da main
110         elif "ENDPROC" in file_lines[i].replace(" ",
               "").replace("\t", "") and main:
111
112             output_lines.extend(main_end)
113             main = False
114
115         if append_line:
116
117             output_lines.append(file_lines[i])
118
119     #cria um novo arquivo com os módulos do socket adicionados
120     print(output_lines)
121     output_file = open(FILEPATH + OUTPUT_FILE, "w",
           encoding='ISO-8859-1')
122     output_file.writelines(output_lines)
123     output_file.close()
124
```

```
125     #lines = variables.split("\n")
126     #print(lines)
127     return
128
129
130 def main():
131
132     add_lines(FILEPATH, FILENAME)
133     return 0
134
135 if __name__ == "__main__":
136
137     main()
```

A.2 variables.txt

Código A.2 – Variáveis a serem inseridas nos algoritmos RAPID

```
1     VAR string program_name:= "DemoSocket"; !MODIFIQUE COM O NOME
          DO PROGRAMA ATUAL!!!!!
2
3     VAR string received_string;
4     VAR string message;
5     VAR intnum timeint:=0;
6     VAR intnum i:=0;
7     VAR socketdev socket1;
8     VAR string strTemp;
9     VAR robtarget actual_pos;
10    VAR intnum state;
```

A.3 socket_management.txt

Código A.3 – TRAP de inicialização da comunicação *socket* e chamada para envio de arquivos

```
1     TRAP socket_management
2
3         ISleep timeint;
4         IF i = 1 THEN
5             !127.0.0.1
6             SocketCreate socket1;
7             SocketConnect socket1, "164.41.91.117", 1884;
8             SocketSend socket1 \Str:="Status;ONLINE";
9             i:=2;
10        ENDIF
11        send_data;
12        IF received_string = "Shutdown" THEN
13
14            end_socket;
```

```
15
16          ENDIF
17
18          IWatch timeint;
19
20      ENDTRAP
```

A.4 send_data.txt

Código A.4 – Procedure para envio de mensagens via *socket*

```
1       PROC send_data()
2
3           message:= "Mode;" + "auto";!NumToStr(area, 3);
4           SocketSend socket1 \Str:=message;
5           !WaitTime 0.01;
6           SocketReceive socket1 \Str:= message;
7
8           message:= "Execution;" + program_name;!NumToStr(area, 3);
9           SocketSend socket1 \Str:=message;
10          !WaitTime 0.01;
11          SocketReceive socket1 \Str:= message;
12
13          message:= "T;" + ValToStr(CTool());!NumToStr(area, 3);
14          SocketSend socket1 \Str:=message;
15          !WaitTime 0.01;
16          SocketReceive socket1 \Str:= message;
17
18          message:= "CW;" + ValToStr(CWobj());!NumToStr(area, 3);
19          SocketSend socket1 \Str:=message;
20          !WaitTime 0.01;
21          SocketReceive socket1 \Str:= message;
22
23          !Escolha o work object e ferramenta de acordo com o
               programa
24          !actual_pos:= CRobT(\Tool:=TTW5500);
25          actual_pos:= CRobT(\Tool:=TTW5500_1 \WObj:=wobj1);
26          !converted_boolean:=StrToVal(ValToStr(soTCPSpeed),
               converted_float);
27          message:= "Speed;" +
               ValToStr(1000*AOutput(soTCPSpeed));!NumToStr(area, 3);
28          SocketSend socket1 \Str:=message;
29          !WaitTime 0.01;
30          SocketReceive socket1 \Str:= message;
31
32          message:= "MSpeed;" +
               ValToStr(MaxRobSpeed());!NumToStr(area, 3);
33          SocketSend socket1 \Str:=message;
34          !WaitTime 0.01;
35          SocketReceive socket1 \Str:= message;
```

```
36
37          message:= "RSpeed;" +
               ValToStr(CSpeedOverride());!NumToStr(area, 3);
38          SocketSend socket1 \Str:=message;
39          !WaitTime 0.01;
40          SocketReceive socket1 \Str:= message;
41
42          message:= "Welding;" +
               ValToStr(DOutput(doFr1ArcOn));!NumToStr(area, 3);
43          SocketSend socket1 \Str:=message;
44          !WaitTime 0.01;
45          SocketReceive socket1 \Str:= message;
46
47          message:= "WVoltage;" +
               ValToStr(AInput(aiFr1WeldingVoltage));!NumToStr(area,
               3);
48          SocketSend socket1 \Str:=message;
49          !WaitTime 0.01;
50          SocketReceive socket1 \Str:= message;
51
52          message:= "WCurrent;" +
               ValToStr(AOutput(aoFr1Power));!NumToStr(area, 3);
53          SocketSend socket1 \Str:=message;
54          !WaitTime 0.01;
55          SocketReceive socket1 \Str:= message;
56
57          message:= "WSpeed;" +
               ValToStr(AOutput(aoFr1WireSpeedWfi));!NumToStr(area, 3);
58          SocketSend socket1 \Str:=message;
59          !WaitTime 0.01;
60          SocketReceive socket1 \Str:= message;
61
62          strTemp:=ValToStr(actual_pos.extax);
63          message:= "PR;" + strTemp;!NumToStr(area, 3);
64          SocketSend socket1 \Str:=message;
65          !WaitTime 0.01;
66          SocketReceive socket1 \Str:= message;
67
68          strTemp:=ValToStr(actual_pos.trans) + "," +
               ValToStr(actual_pos.rot);
69          message:= "CR;" + strTemp;!NumToStr(area, 3);
70          SocketSend socket1 \Str:=message;
71          !WaitTime 0.01;
72          SocketReceive socket1 \Str:= message;
73
74          message:= "Timestamp;" + CDate() + ";" +
               CTime();!NumToStr(area, 3);
75          SocketSend socket1 \Str:=message;
76          SocketReceive socket1 \Str:= message;
77
78          IF message = "Shutdown" THEN
79
```

```
80              end_socket;
81
82          ENDIF
83
84      ENDPROC
```

A.5 end_socket

Código A.5 – Procedure que encerra o *socket* no código RAPID

```
1       PROC end_socket()
2
3           IDelete timeint;
4           send_data;
5           SocketSend socket1 \Str:="Status;OFFLINE";
6           SocketReceive socket1 \Str:= message;
7           SocketClose socket1;
8
9       ENDPROC
```

A.6 main_start.txt

Código A.6 – Cabeçalho da "main"pro código RAPID

```
1           IEnable;
2           i:=0;
3           IF i = 0 THEN
4
5               CONNECT timeint WITH socket_management;
6               ITimer 0.5, timeint;
7               i:=1;
8
9           ENDIF
```

A.7 main_end.txt

Código A.7 – Finalização da "main"pro código RAPID

```
1           IDisable;
2           send_data;
3           end_socket;
```

Apêndice B – Adaptador MQTT

B.1 config.json

Código B.1 – Arquivo *json* de configuração do adaptador

```
{

    "ethernet": {

        "host": "164.41.91.117",
        "port": 1884,
        "topic": "abb/irb2600/"

    },

    "publisher": {

        "host": "localhost",
        "port": 1883,
        "topic": "abb/irb2600/",
        "heartbit": 500,
        "simulation": false

    }

}
```

B.2 abb-irb-ethernet.js

Código B.2 – Código referente à comunicação entre *sockets* do adaptador MQTT e código RAPID no controlador ICR5

```
//João Vítor Arantes Cabral 17/0126048
//Trabalho de graduacao
//Módulo de comunicação Socket TCP/IP com controlador ICR5

const net = require('net');
//var util=require('node:util');
//import util from 'util';
const {promisify} = require('util')


const sleep = promisify(setTimeout)

module.exports = function (options, pub) {
```

```
14
15      //Inicialização do servidor
16      let initial_time = Date.now();
17      let number_of_messages = 0;
18      let _port = options.port || 1884;
19      let _host = options.host || "127.0.0.1"
20      this.clients = {};
21      this.pub = pub;
22      pub.publish(options.topic + "test", "test");
23      this.server=net.createServer(function(conn){
24
25          //Grava o endereço do socket do controlador para envio de
                mensagens
26          conn.on('connect',function(){
27
28              console.log('connect');
29              this.clients[socket.id] = socket;
30
31          });
32          //Método para monitorar mensagens enviadas pelo socket no
                controlador
33          conn.on('data',function(data){
34
35              var message = data.toString()
36
37              number_of_messages++;
38              console.log("Delay of message: ", Date.now() -
                    initial_time), "";
39              //console.log('CLIENT:', message, "\n");
40              console.log("Number of messages: ",
                    number_of_messages);
41              extract_data(message, pub, options);
42              initial_time = Date.now();
43              try {
44
45                  conn.write("PONG");
46
47              }
48              catch (e){
49
50                  console.log(e)
51
52              }
53
54          });
55
56      });
57      this.server.listen(_port, _host, function () {
58
59          console.log('Ethernet TCP/IP Socket server started and
                listening on port ', _port);
60
```

```
61      });
62      /*
63      this.server.on('connection', (stream) => {
64
65          stream.connect
66
67      })
68      */
69     //Função para extrair dados das mensagens enviadas pelo
          controlador
70      function extract_data(message, pub, options) {
71
72          var topic = message.split(";")[0]
73          var payload = message.split(";")[1]
74          if (payload.includes("Timestamp")) {
75
76              payload.replace("Timestamp", "")
77
78          } else if (payload.includes("Execution")) {
79
80              payload.replace("Execution", "")
81
82          }
83          console.log(topic)
84          if (topic == "PR"){
85
86              topic = "PosRotation"
87
88          } else if (topic == "CR") {
89
90              topic = "CRobT"
91
92          } else if (topic == "W") {
93
94              topic = "Welding"
95
96          } else if (topic == "T") {
97
98              topic = "Tool"
99
100         } else if (topic == "CW") {
101
102             topic = "CWobj"
103
104         }
105
106         if (topic == "Status") {
107
108             if (payload.includes("ONLINE")) {
109
110                 payload = "ONLINE"
111
```

```
112             } else {
113 
114                 payload = "OFFLINE"
115 
116             }
117 
118             console.log("Status: ", payload, "\n")
119             pub.publish(options.topic + "status", payload)
120 
121         } else if (topic == "Mode") {
122 
123             //console.log("mode: ", payload, "\n")
124             pub.publish(options.topic + "mode", payload)
125 
126         } else if (topic == "Execution") {
127 
128             //console.log("Speed: ", payload, "\n")
129             pub.publish(options.topic + "execution", payload)
130 
131         } else if (topic == "Tool") {
132 
133             console.log("Tool: ", payload, "\n")
134             pub.publish(options.topic + "tool", payload)
135 
136         } else if (topic == "CWobj") {
137 
138             console.log("Work object: ", payload, "\n")
139             pub.publish(options.topic + "cwobj", payload)
140 
141         } else if (topic == "Speed") {
142 
143             //console.log("Speed: ", payload, "\n")
144             payload = payload.replace(/[^\d.-]/g, '')
145             pub.publish(options.topic + "speed", payload)
146 
147         } else if (topic == "MSpeed") {
148 
149             //console.log("Speed: ", payload, "\n")
150             payload = payload.replace(/[^\d.-]/g, '')
151             pub.publish(options.topic + "mspeed", payload)
152 
153         } else if (topic == "RSpeed") {
154 
155             //console.log("Speed: ", payload, "\n")
156             payload = payload.replace(/[^\d.-]/g, '')
157             pub.publish(options.topic + "rspeed", payload)
158 
159         } else if (topic == "Welding") {
160 
161             //console.log("Speed: ", payload, "\n")
162             if (payload == "1") {
163 
```

```
164                 payload = "TRUE"
165
166             } else {
167
168                 payload = "FALSE"
169
170             }
171             pub.publish(options.topic + "welding", payload)
172
173         } else if (topic == "WVoltage") {
174
175             pub.publish(options.topic + "wvoltage", payload)
176
177         } else if (topic == "WCurrent") {
178
179             pub.publish(options.topic + "wcurrent", payload)
180
181         } else if (topic == "WSpeed") {
182
183             pub.publish(options.topic + "wspeed", payload)
184
185         } else if (topic == "PosRotation") {
186
187             //console.log("Speed: ", payload, "\n")
188             //payload = payload.replace(/[a-z]/gi, '');
189             payload = payload.replace("CR", "")
190             let aux_arr = JSON.parse("[" + payload + "]")
191             pub.publish(options.topic + "posrotation", payload)
192
193         }  else if (topic == "CRobT"){
194
195
196             console.log(payload)
197             //payload = payload.replace(/[a-z]/gi, '');
198             console.log(payload)
199             if (payload.includes("Timestamp")) {
200
201                 payload.replace("Timestamp", "")
202
203             }
204             let aux_arr = JSON.parse("[" + payload + "]");
205             console.log(aux_arr)
206             //console.log("CRobT: ", payload, "\n")
207             //console.log(options.topic + "crobt", payload)
208             pub.publish(options.topic + "crobt", "[" + payload +
                    "]")
209             pub.publish(options.topic + "x_pos",
                    String(aux_arr[0][0]))
210             pub.publish(options.topic + "y_pos",
                    String(aux_arr[0][1]))
211             pub.publish(options.topic + "z_pos",
                    String(aux_arr[0][2]))
```

```
212             pub.publish(options.topic + "rorient",
213 `[${String( aux_arr[1][0])},${String(
    aux_arr[1][1])},${String(aux_arr[1][2])},${String(
    aux_arr[1][3])}]`)
214             //pub.publish(options.topic + "crobt", "[" + payload +
                    "]")
215
216         } else if (topic == "Timestamp") {
217
218             console.log("Timestamp: ", payload, "\n")
219             var date = new Date()
220             pub.publish(options.topic + "timestamp",
                    String(Date.now()))
221
222         } else {
223
224             pub.publish(options.topic + "status", "ONLINE")
225
226         }
227
228     }
229
230 }
```

B.3 abb-mqtt-publisher.js

Código B.3 – Código de inicialização do *publisher* MQTT

```
1  //João Vítor Arantes Cabral 17/0126048
2  //Trabalho de graduacao
3  //Módulo publisher do adaptador para publicação de mensagens via
       MQTT
4
5  const Ethernet = require('./abb-irb-ethernet')
6  const Simulator = require('./data_simulator')
7  const config = require('./config.json')
8
9  const mqtt = require('mqtt')
10 const Broker = require('./mqtt-broker')
11 var first = true
12
13 //Inicia o broker e o publisher, e inicializa o módulo do socket
       Ethernet
14 const broker = new Broker(config.publisher)
15 const pub = mqtt.connect(
       `mqtt://${config.publisher.host}:${config.publisher.port}` )
16 const ethernet = new Ethernet(config.ethernet, pub)
17
18 //Encerramento do processo quando usuário aperta ctrl+c
19 process.on('SIGINT', function() {
```

```
20
21      ethernet.server.close()
22      console.log('Finishing process and closing publisher.')
23      pub.publish(config.publisher.topic + 'status', 'OFFLINE')
24      pub.end()
25      broker.server.close()
26      throw new Error("Process ended");
27
28      /*
29      for (key in ethernet.clients) {
30
31          console.log(`${key}: ${ethernet.clients[key]}`);
32
33      }*/
34      //
35
36
37  })
```

B.4 mqtt-broker.js

Código B.4 – Código referente ao *broker* MQTT

```
1   const aedes = require('aedes')();
2   const net = require('net');
3   const httpServer = require("http").createServer();
4   const ws = require("websocket-stream");
5   ws.createServer({ server: httpServer }, aedes.handle);
6   var util=require('util');
7
8
9   module.exports = function (options) {
10
11      let _port = options.port || 1883;
12      this.connected = 0;
13      this.server = net.createServer(aedes.handle);
14
15      //Inicializa o servidor broker na porta determinada pelo
            arquivo config.json
16      this.server.listen(_port, function () {
17
18          console.log('MQTT server started and listening on port ',
                _port);
19
20      });
21      this.server.on('connection', (socket) => {
22
23          const clientAddress = socket.remoteAddress.split(':')[3] +
                ' ' + socket.remotePort;
```

```
24          console.log("New socket client connected: " +
               clientAddress);
25          this.connected += 1;
26
27      });
28
29      //Monitora as mensagens passadas ao servidor
30      this.server.on('data',function(data){
31
32          util.puts(data);
33
34      });
35
36      //Cria um websocket
37      httpServer.listen(7777, () => {
38
39          console.log("MQTT server started and listening on port",
               7777);
40
41      });
42
43
44      httpServer.on("connection", (socket) => {
45
46          const clientAddress = socket.remoteAddress.split(":")[3] +
               " " + socket.remotePort;
47          console.log("New socket client connected: " +
               clientAddress);
48
49      });
50      /*server.on('timeout', (socket) => {
51
52          const clientAddress = socket.remoteAddress.split(':')[3] +
               ' ' + socket.remotePort
53          console.log("Socket client disconnected: " + clientAddress)
54          this.connected -= 1
55
56      })*/
57
58
59  }
```

Apêndice C – Código de simulação de espelhamento de movimentação 3D do robô

Código C.1 – Simulação 3D de movimentação através de subscrição em tópicos MQTT

```
#João Vítor Arantes Cabral 17/0126048
#Trabalho de graduacao
#Módulo de simulação de movimentação 3D do DT com RoboDK

from robodk import robolink    # RoboDK API
from robodk import robomath    # Robot toolbox
from ast import literal_eval

# The following 2 lines keep your code compatible with older versions or RoboDK:
from robodk import *      # RoboDK API
from robolink import *    # Robot toolbox

import paho.mqtt.subscribe as subscribe

import os
import re
import sys
sys.path.insert(0, 'C:/RoboDK/Library/Macros')
import ArcStart as ArcStart
import ArcEnd as ArcEnd
import numpy as np
from scipy.spatial.transform import Rotation

RDK = Robolink()

#Inicializa variáveis e constantes importantes para a movimentação
target = [[876.695,168.553,912.675],[0.556886,-0.0786023,0.825158,0.0530474]]
APPROACH = 300  # Approach distance
SPEED_WELD = 50  # Speed in mn/s of the welding path
SPEED_MOVE = 200  # Speed in mm/s of the approach/retract movements
#[[1400.0000,11.3000,510.0000],[1,0,0,0]], [[0,0,0],[1,0,0,0] ]
wobj = False#[ False,True,"",[[1400.0000,11.3000,510.0000],[1,0,0,0]],[[0,0,0],[1,0,0,0] ] ]
WELDING = False
ROTATION_SPEED = 0

#Identifica o rob  e a mesa no ambiente de simulação RoboDK
robot = RDK.Item('ABB IRB 2600ID-15/1.85')
```

```
38 table = RDK.Item('ABB IRBP A250 D1000')
39 print_reference = RDK.Item('Printing Reference')
40
41 def ABB_2_Pose(CPosT):
42
43     #converte o target ABB para pose RoboDK
44     xyz = CPosT[0]
45     q1234 = CPosT[1]
46     pose = quaternion_2_pose(q1234)
47     pose.setPos(xyz)
48     return pose
49
50 # Radius of the sphere
51 SPHERE_RADIUS = 5
52
53 # Set the particle color as a named color or as AARRGGBB (alpha
       channel goes first!)
54 COLOR = "white"
55
56 def on_CRobT(client, userdata, message):
57
58     #função de monitoramento dos tópicos subscritos
59     global wobj
60     global print_reference
61     if  message.topic == "abb/irb2600/rspeed":
62
63
64         rotation_speed_str = message.payload.decode("utf-8")
65         #rotation_speed_str = re.findall("\d+",
              rotation_speed_str)[0]
66
67         try:
68
69             ROTATION_SPEED = float(rotation_speed_str)
70
71         except Exception as e:
72
73             print(e)
74             pass
75
76     elif message.topic == "abb/irb2600/speed":
77
78         #ajusta a velocidade de movimentação da simulação de
              acordo com a velocidade linear do TCP
79         speed_str = message.payload.decode("utf-8")
80         try:
81
82             SPEED_MOVE = float(speed_str)
83
84         except Exception as e:
85
86             print(e)
```

```
87              pass
88
89      elif message.topic == "abb/irb2600/welding":
90
91          #monitora se a solda está ativa ou desativada
92          welding_str = message.payload.decode("utf-8")
93          WELDING = True if welding_str == "TRUE" else False
94          if WELDING == True:
95
96              # Create a new spray gun
97              tool = 0  # auto detect active tool
98              object = 0 # auto detect object to attach particles
99
100
101             #Ativa o spray para simular soldagem
102             RDK.Spray_Add(tool, object, "PARTICLE=SPHERE(%.2f)
                    NO_PROJECT COLOR=%s" % (SPHERE_RADIUS, COLOR))
103             RDK.Spray_SetState(SPRAY_ON)
104
105         else:
106
107             #Desativa o spray
108             RDK.Spray_SetState(SPRAY_OFF)
109
110     elif message.topic == "abb/irb2600/posrotation":
111
112         #movimenta a mesa posicionadora de acordo com os ângulos
                obtidos dos eixos externos
113         posrotation = message.payload.decode("utf-8")
114         posrotation = literal_eval(posrotation)
115         print("PosRotation: " + str(posrotation))
116
117         #print(posrotation)
118         # Create a rotation object from Euler angles specifying
                axes of rotation
119         rot = Rotation.from_euler('xyz', [posrotation[2], 0,
                posrotation[1]], degrees=True)
120
121         # Convert to quaternions and print
122         rot_quat = rot.as_quat()
123         #[posrotation[2], 0.000, posrotation[1], 1.000]
124         #posrotation = [[0.000, 0.000, 0.000], [posrotation[2] +
                1.3, 0.000, posrotation[1] - 1.3, 1.000]]
125         posrotation = [[0.000, 0.000, 0.000], rot_quat]
126         pose_table = ABB_2_Pose(posrotation)
127         table.MoveJ(pose_table)
128         #print(pose)
129         #robot.MoveJ(transl(0, 0, 0) * pose)
130
131     elif message.topic == "abb/irb2600/cwobj":
132
133         #ajusta o objeto de trabalho atual
```

```
134         wobj = message.payload.decode("utf-8")
135         wobj = literal_eval(wobj.replace("FALSE",
               "False").replace("TRUE", "True"))
136         print(wobj)
137 
138     elif message.topic == "abb/irb2600/crobt":
139 
140         #movimenta o rob
141         position = message.payload.decode("utf-8")
142         position = literal_eval(position)
143         position = list(position)
144         if wobj[3][0] == [0.000, 0.000, 0.000]:
145 
146             position[0] = list(np.add(position[0], [-1400.000,
                   0.000, -400.000]))
147             #position[1] = list(np.add(position[1], [-1.000,
                   0.000, 0.000, 0.000]))
148 
149         #position[0] = list(np.add(position[0], wobj[3][0]))
150         #position[1] = list(np.add(position[1], wobj[3][1]))
151         print("CRobT: " + str(position))
152         #position[0][1] = position[0][1] - 1400.0
153         #position[0][1] = position[0][1] - 11.3
154         #position[0][2] = position[0][2] - 5.0
155         pose = ABB_2_Pose(position)
156         print(pose)
157 
158         try:
159 
160             robot.MoveJ(transl(0, 0, 0) * pose)
161 
162         except Exception as e:
163 
164             print(e)
165             pass
166 
167 #subscreve nos tópicos necessários para simulação de movimentação
168 MY_TOPICS = [("abb/irb2600/rspeed", 0), ("abb/irb2600/speed", 0),
       ("abb/irb2600/cwobj", 0), ("abb/irb2600/welding", 0),
       ("abb/irb2600/posrotation", 0), ("abb/irb2600/crobt", 0)]
169 subscribe.callback(on_CRobT, MY_TOPICS)
170 # Link to RoboDK
171 RDK = robolink.Robolink()
```

Apêndice D – Fluxograma de execução do adaptador MQTT

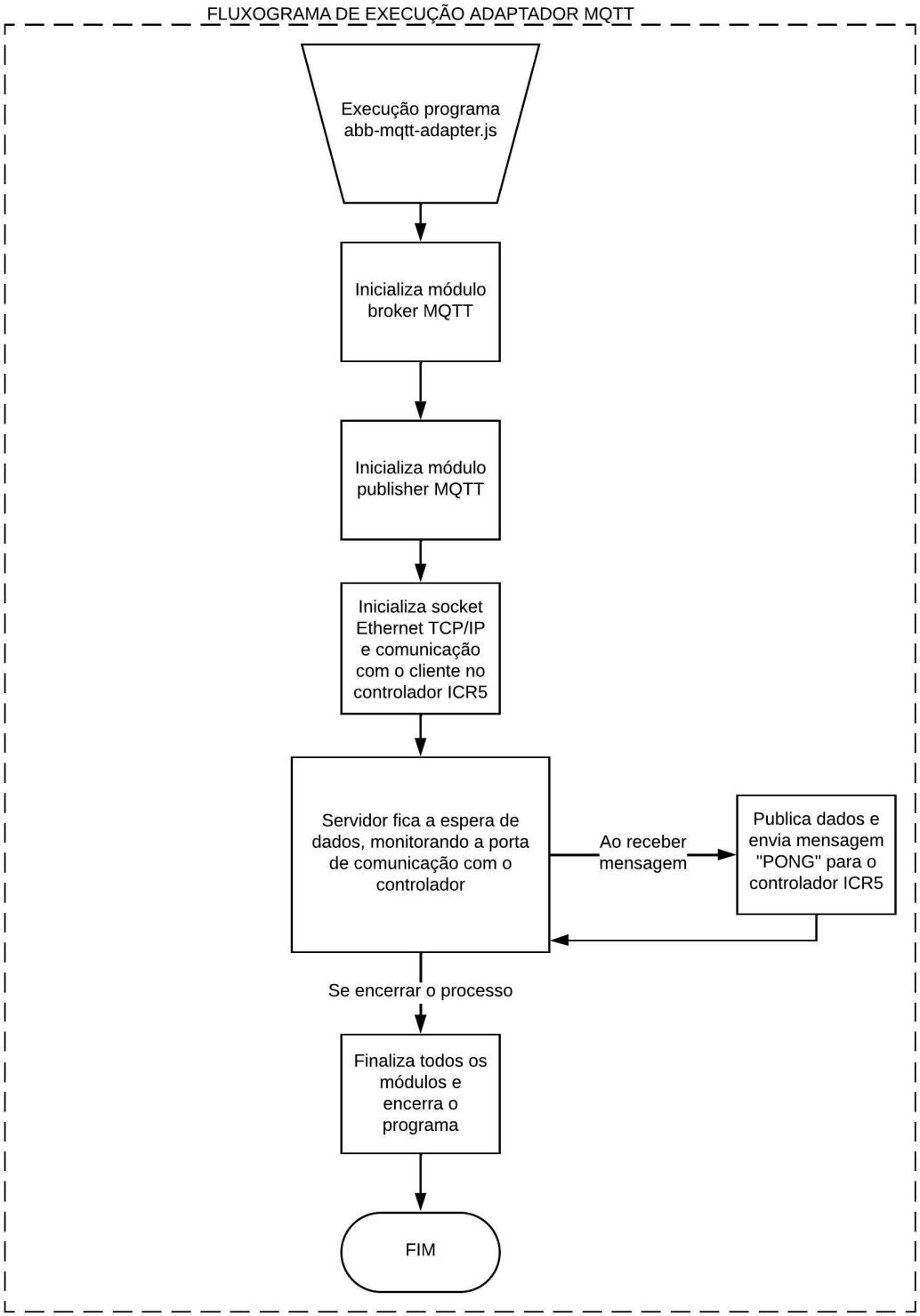

Figura D.23 – Fluxo de execução do programa "abb-mqtt-adapter.js", que funciona juntamente com os módulos do *broker* e *publisher* MQTT.

Apêndice E – Fluxograma do programa que insere módulo *socket* aos programas RAPID

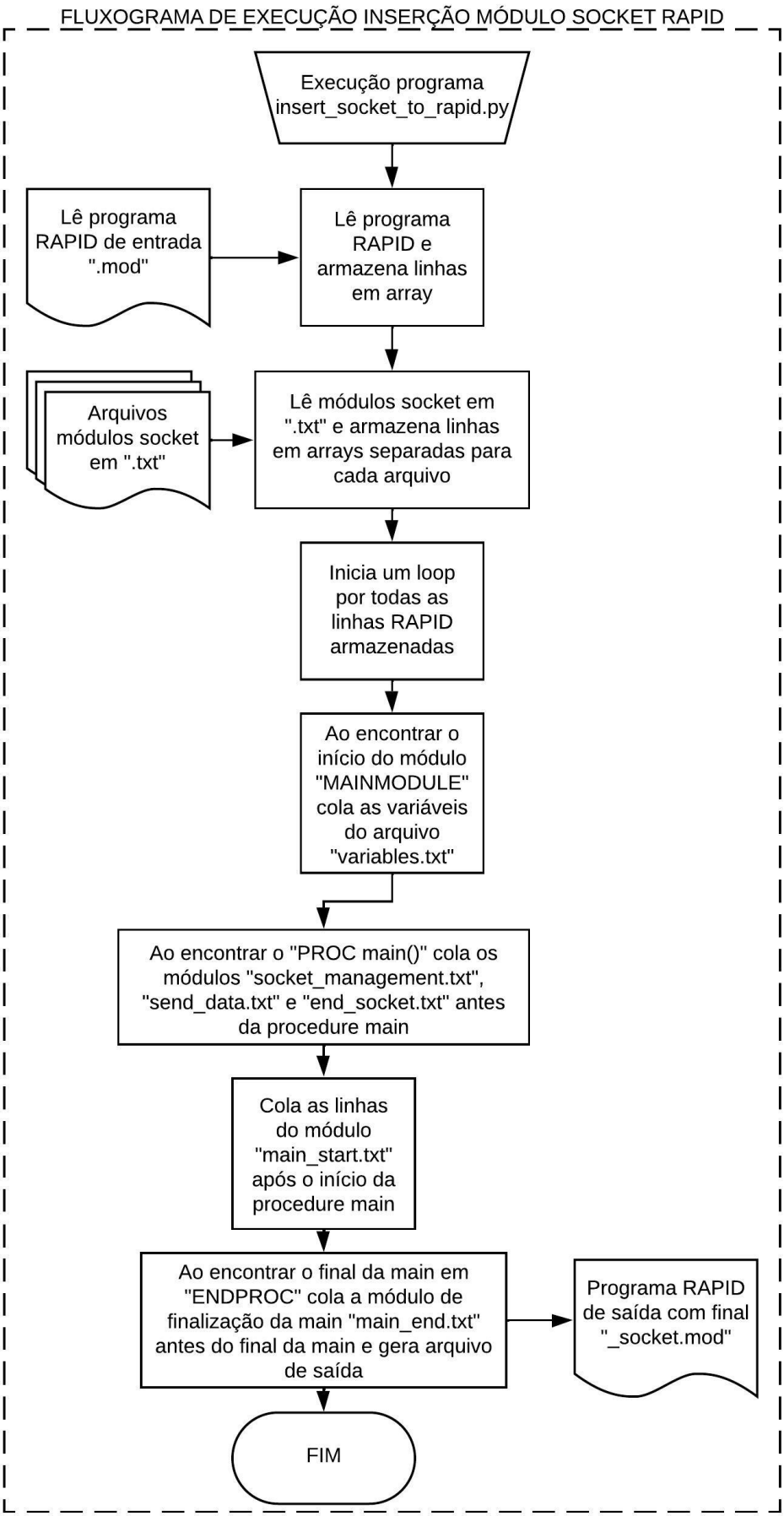

Figura E.24 – Fluxo de execução do programa "insert_socket_to_rapid.py"onde os módulos *socket* são inseridos no arquivo RAPID ".mod"de entrada.