

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DO CEARÁ
PRÓ-REITORIA DE PESQUISA, PÓS-GRADUAÇÃO E INOVAÇÃO
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Renato Alves de Oliveira

**Uma proposta de Arquitetura Genérica e Linguagem de Design
de Gêmeos Digitais no contexto de Internet das Coisas**

Fortaleza – CE

13 de maio de 2023

Renato Alves de Oliveira

**Uma proposta de Arquitetura Genérica e Linguagem
de Design de Gêmeos Digitais no contexto de
Internet das Coisas**

Dissertação submetida ao Programa de Pós-Graduação em Ciência da Computação do Instituto Federal de Educação, Ciência e Tecnologia do Ceará, como requisito parcial para obtenção do grau de Mestre em Ciência da Computação.

Linha de Pesquisa: Redes de Computadores e Sistemas Distribuídos (RCSD)

Orientador: Dra. Carina Teixeira de Oliveira

Coorientador: Dra. Atslands Rego da Rocha

Instituto Federal de Educação, Ciência e Tecnologia do Ceará (IFCE)

Pró-Reitoria de Pesquisa, Pós-Graduação e Inovação (PRPI)

Programa de Pós-Graduação em Ciência da Computação (PPGCC)

Fortaleza – CE

13 de maio de 2023

Dados Internacionais de Catalogação na Publicação
Instituto Federal do Ceará - IFCE
Sistema de Bibliotecas - SIBI
Ficha catalográfica elaborada pelo SIBI/IFCE, com os dados fornecidos pelo(a) autor(a)

- O48p Oliveira, Renato Alves de.
 Uma proposta de Arquitetura Genérica e Linguagem de Design de Gêmeos Digitais no contexto de Internet das Coisas / Renato Alves de Oliveira. - 2023.
 194 f. : il. color.
- Dissertação (Mestrado) - Instituto Federal do Ceará, Mestrado em Ciência da Computação, Campus Fortaleza, 2023.
 Orientação: Profa. Dra. Carina Teixeira de Oliveira.
 Coorientação: Profa. Dra. Atslands Rego da Rocha.
1. Internet das Coisas. 2. Gêmeos Digitais . 3. Arquitetura Genérica. 4. Linguagem de Design. 5. Experimentação. I. Título.

CDD 004



INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DO CEARÁ
Av. 13 de maio, 2081 , - Bairro Benfica - CEP 60040-531 - Fortaleza - CE - www.ifce.edu.br

AVALIAÇÃO

Processo: 23256.002052/2023-19

Interessado: Renato Alves de Oliveira

RENATO ALVES DE OLIVEIRA

Uma proposta de Arquitetura Genérica e Linguagem de Design de Gêmeos Digitais

Dissertação submetida à Coordenação do Curso de Pós-graduação em Ciência da Computação do Instituto Federal de Educação, Ciência e Tecnologia do Ceará, como requisito parcial para a obtenção do título de mestre em Ciência da Computação, área de concentração Sistemas de Computação.

Aprovada em **27 / 02 / 2023**.

BANCA EXAMINADORA

Profª. Dra. Carina Teixeira de Oliveira (Orientador)

Instituto Federal de Educação, Ciência e Tecnologia do Ceará - PPGCC/IFCE

Prof. Dr. Lincoln Souza Rocha

Universidade Federal do Ceará - UFC

Prof. Dr. Antônio Wendell de Oliveira Rodrigues

Instituto Federal de Educação, Ciência e Tecnologia do Ceará - PPGCC/IFCE

Profª. Dra. Atslands Rêgo da Rocha

Universidade Federal do Ceará - UFC



Documento assinado eletronicamente por **Atslands Rego da Rocha, Usuário Externo**, em 28/02/2023, às 14:10, com fundamento no art. 6º, § 1º, do [Decreto nº 8.539, de 8 de outubro de 2015](#).



Documento assinado eletronicamente por **Lincoln Souza Rocha, Usuário Externo**, em 28/02/2023, às 15:15, com fundamento no art. 6º, § 1º, do [Decreto nº 8.539, de 8 de outubro de 2015](#).



Documento assinado eletronicamente por **Antonio Wendell de Oliveira Rodrigues, Professor do Ensino Básico, Técnico e Tecnológico**, em 28/02/2023, às 20:17, com fundamento no art. 6º, § 1º, do [Decreto nº 8.539, de 8 de outubro de 2015](#).



Documento assinado eletronicamente por **Carina Teixeira de Oliveira, Professora do Ensino Básico, Técnico e Tecnológico**, em 28/02/2023, às 20:22, com fundamento no art. 6º, § 1º, do [Decreto nº 8.539, de 8 de outubro de 2015](#).



A autenticidade do documento pode ser conferida no site https://sei.ifce.edu.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0 informando o código verificador **4612884** e o código CRC **48455A9D**.

*Este trabalho é dedicado a quem busca tornar o mundo mais agradável, justo e
igualitário através da ciência!*

Agradecimentos

À todas as consciências que, de alguma forma, me incentivaram, apoiaram, orientaram ou torceram por mim. Mãe, noiva, familiares, orientadora, professores e amigos.

Guarda ancora quanta vita c'è

Resumo

A área de Internet das Coisas (*Internet of Things* - IoT) tem crescido substancialmente nos últimos anos. Tanto cresce na quantidade de dispositivos IoT fabricados ao redor do globo, quanto na quantidade de dados gerados a partir da interação entre esses objetos. Além disso, a área vem recebendo investimentos financeiros bilionários, em diversas áreas da sociedade humana, como manufatura, cidades inteligentes, saúde, dentre outras. Dentro do contexto de IoT, uma área que vem recebendo grande atenção na última década é a área de *Digital Twin* (DT). O DT é responsável por criar clones exatos (denominados Entidades Virtuais - EV) de objetos, locais ou pessoas (denominados Entidades Físicas - EF), e tem aplicabilidade em diversas áreas, como manufatura, aeronáutica, saúde, agricultura, dentre outras. Apesar de sua popularidade, na área de DT ainda há vários temas de Pesquisa e Inovação a serem estudados e desenvolvidos. Dentre eles, dois que se destacam são as Arquiteturas e Linguagens de Design. Em se tratando de arquiteturas, uma de suas lacunas mais notáveis diz respeito à falta de padronização arquitetural, não havendo um consenso global em como se construir uma arquitetura de DT. Se houvesse um padrão único de arquitetura de DT, o processo de desenvolvimento/implantação de novas aplicações na área tenderia a ser mais rápido, dispensando a necessidade de projetistas se preocuparem com que tipo de arquitetura usar. Além disso, a adesão de um padrão unificado agilizaria o entendimento do tema por parte de novos pesquisadores. Já em se tratando de linguagens de design (as quais são responsáveis por descrever toda a estrutura da EF sob diferentes aspectos), há duas principais lacunas: a escassez de trabalhos que desenvolvam LDs completas e de código aberto para cenários específicos, como o agrícola; e a escassez de LDs de modo geral, pois algumas grandes plataformas de IoT (como Google Cloud IoT, AWS IoT etc), embora possuam implementações de EVs, as quais podem ser acessadas e manipuladas por seus clientes, não disponibilizam de forma aberta o código de suas LDs. Neste sentido, este trabalho apresenta (i) uma Proposta de Arquitetura Genérica de DT e (ii) uma Proposta de Linguagem de Design de DT para a Área Agrícola, a AgroLD. Além disso, é apresentada uma revisão bibliográfica realizada para compreender como os trabalhos atuais validam arquiteturas de DT. Como forma de validação, baseando-se nos mecanismos encontrados na literatura, a Arquitetura Genérica e a AgroLD foram aplicadas a um estudo de caso na área agrícola, a partir do qual foram realizados experimentos em um *testbed* real. Ao final dos experimentos, pôde-se averiguar que a arquitetura genérica foi submetida a uma maior variedade de experimentos que os trabalhos encontrados na literatura revisada, e que a AgroLD possui uma série de vantagens em relação a outra linguagem de design encontrada na literatura revisada.

Palavras-chave: Internet das Coisas (IoT). Gêmeos Digitais (DT). Arquitetura Genérica. Linguagem de Design (LD). Agricultura. Experimentação.

Abstract

The Internet of Things (IoT) area has grown substantially in recent years. Both the number of IoT devices manufactured around the globe and the amount of data generated from the interaction between these objects grows. In addition, the area has been receiving billionaire financial investments in various areas of human society, such as manufacturing, smart cities, health, among others. Within the context of IoT, an area that has received great attention in the last decade is the Digital Twin (DT) area. DT is responsible for creating exact clones (called Virtual Entities - EV) of objects, places or people (called Physical Entities - EF), and has applicability in several areas, such as manufacturing, aeronautics, health, agriculture, among others. Despite its popularity, in the TD area there are still several Research and Innovation themes to be studied and developed. Among them, two that stand out are Design Architectures and Languages. When it comes to architectures, one of its most notable shortcomings concerns the lack of architectural standardization, with no global consensus on how to build a DT architecture. If there were a single standard for DT architecture, the process of developing/implementing new applications in the area would tend to be faster, eliminating the need for designers to worry about what type of architecture to use. In addition, the adherence to a unified standard would speed up the understanding of the subject by new researchers. When it comes to design languages (which are responsible for describing the entire FE structure under different aspects), there are two main gaps: the scarcity of works that develop complete and open source LDs for specific scenarios, such as agriculture; and the scarcity of LDs in general, as some large IoT platforms (such as Google Cloud IoT, AWS IoT etc), although they have EVs implementations, which can be accessed and manipulated by their customers, do not make the code openly available of your LDs. In this sense, this work presents (i) a DT Generic Architecture Proposal and (ii) a DT Design Language Proposal for the Agricultural Area, AgroLD. In addition, a bibliographic review carried out to understand how current works validate DT architectures is presented. As a form of validation, based on the mechanisms found in the literature, the Generic Architecture and AgroLD were applied to a case study in the agricultural area, from which experiments were carried out in a real testbed. At the end of the experiments, it was possible to verify that the generic architecture was submitted to a greater variety of experiments than the works found in the reviewed literature, and that AgroLD has a series of advantages in relation to another design language found in the reviewed literature.

Keywords: Internet of Things (IoT). Digital Twin (DT). Generic Architecture. Design Language (DL). Agriculture. Experimentation.

Lista de ilustrações

Figura 1 – Exemplo de Arquitetura de Três Camadas da IoT.	32
Figura 2 – Arquitetura Publicação/Assinatura do MQTT.	33
Figura 3 – Arquitetura Requisição/Resposta do CoAP.	34
Figura 4 – Exemplo de <i>Digital Twin</i> de <i>Smartphone</i>	39
Figura 5 – Arquitetura baseada em funcionalidades de <i>middleware</i>	44
Figura 6 – Arquitetura aplicada ao contexto de gestão de automóveis movidos a baterias.	46
Figura 7 – Arquitetura de 3 componentes para DT.	48
Figura 8 – Arquitetura com 8 componentes para DT.	49
Figura 9 – Arquitetura de 5 componentes voltada para o ambiente fabril. . .	50
Figura 10 – Arquiteturas Físico-Virtual e Virtual-Físico.	52
Figura 11 – Arquitetura de três componentes.	53
Figura 12 – Arquitetura de cinco dimensões.	54
Figura 13 – Arquitetura orientada a serviços.	55
Figura 14 – Arquitetura de três componentes para monitoramento de áreas alagadas.	57
Figura 15 – Arquitetura Original do Plante.	62
Figura 16 – Metodologia utilizada para desenvolvimento da dissertação.	65
Figura 17 – Componentes da Arquitetura DT e suas equivalências na literatura. .	69
Figura 18 – Proposta de Arquitetura Genérica de DT.	70
Figura 19 – Arquitetura Genérica aplicada ao estudo de caso do Plante.	76
Figura 20 – Estrutura da AgroLD aplicada ao Plante Box.	94
Figura 21 – Estrutura da DTDL aplicada ao Plante Box.	95
Figura 22 – <i>Testbed</i> utilizado para avaliação da Arquitetura Genérica.	125
Figura 23 – Mensagens Enviadas em função do Tempo de Transmissão.	126
Figura 24 – Mensagens Recebidas em função do Tempo de Transmissão.	127
Figura 25 – Tempo de Consumo de CPU em função do Tempo de Transmissão. .	127
Figura 26 – Tempo Médio de Consumo de CPU (por Mensagem Recebida) em função do Tempo de Transmissão.	128
Figura 27 – Tempo Médio de Entrega (por Mensagem Recebida) em função do Tempo de Transmissão.	129
Figura 28 – Mensagens Enviadas em função da Quantidade de Entidades Físicas. .	130
Figura 29 – Mensagens Recebidas em função da Quantidade de Entidades Físicas.	130
Figura 30 – Taxa de Entrega em função da Quantidade de Entidades Físicas. .	131
Figura 31 – Tempo de Entrega em função da Quantidade de Entidades Físicas. .	132
Figura 32 – Consumo de CPU em função da Quantidade de Entidades Físicas .	132
Figura 33 – Consumo de RAM em função da Quantidade de Entidades Físicas .	133
Figura 34 – Tempo para a Interpretação da AgroLD e DTDL em função da quantidade de Acessos ao Componente Linguagem de Design . . .	134

Figura 35 – Trocas de Contexto na CPU para a Interpretação da AgroLD e DTDL em função da quantidade de Acessos ao Componente Linguagem de Design	135
Figura 36 – Tempo de Consumo de CPU para a Interpretação da AgroLD e DTDL em função da quantidade de Acessos ao Componente Linguagem de Design	135
Figura 37 – Consumo de RAM para a Interpretação da AgroLD e DTDL em função da quantidade de Acessos ao Componente Linguagem de Design	136

Lista de tabelas

Tabela 1 – Número sequencial, ano de publicação, quantidade de citações e título (+ referência) dos 28 trabalhos que compõem o <i>corpus</i> primário.	38
Tabela 2 – Grupos de camadas/componentes arquiteturais de DT.	68
Tabela 3 – Informações técnicas do microcontrolador Node MCU ESP 32S. .	80
Tabela 4 – Detalhes dos sensores e atuador do Plante <i>Box</i>	81
Tabela 5 – Camadas da Arquitetura de (CATARCI et al., 2019).	114
Tabela 6 – Camadas da Arquitetura de (REDELINGHUYS; BASSON; KRUGER, 2020).	114
Tabela 7 – Camadas da Arquitetura de (KYCHKIN; NIKOLAEV, 2020).	115
Tabela 8 – Camadas da Arquitetura de (BOTTANI; VIGNALI; TANCREDI, 2020). .	115
Tabela 9 – Camadas da Arquitetura de (AHELEROFF et al., 2021).	116
Tabela 10 – Camadas da Arquitetura de (ZHENG et al., 2020).	116
Tabela 11 – Camadas da Arquitetura de (DALIBOR et al., 2020).	117
Tabela 12 – Camadas da Arquitetura de (MOURTZIS et al., 2021).	117
Tabela 13 – Camadas da Arquitetura de (PAN et al., 2021).	117
Tabela 14 – Camadas da Arquitetura de (LIM et al., 2021).	118
Tabela 15 – Camadas da Arquitetura de (ZOU et al., 2022).	118
Tabela 16 – Camadas da Arquitetura de (GAO et al., 2022).	118
Tabela 17 – Camadas da Arquitetura de (THÜRER; LI; QU, 2022).	119
Tabela 18 – Camadas da Arquitetura de (NWOGU et al., 2022).	119
Tabela 19 – Componentes da camada <i>Digital Twin</i> da Arquitetura de (NWOGU et al., 2022).	119
Tabela 20 – Comparação das Arquiteturas	120
Tabela 21 – Consumo de dados da LD e da DTDL	122
Tabela 22 – Comparação Resumida entre AgroLD e DTDL	137
Tabela 23 – Comparação da Proposta e Validação da Arquitetura Genérica com Trabalhos Relacionados do Capítulo 6	137

Lista de abreviaturas e siglas

DT	Gêmeo Digital (<i>Digital Twin</i>)
IoT	Internet das Coisas (<i>Internet of Things</i>)
MIT	Instituto de Tecnologia de Massachusetts (<i>Massachusetts Institute of Technology</i>)
M2M	Máquina para Máquina (<i>Machine to Machine</i>)
TI	Tecnologia da Informação
MQTT	Transporte de Telemetria de Enfileiramento de Mensagens (<i>Message Queuing Telemetry Transport</i>)
CoAP	Protocolo de Aplicativo Restrito (<i>Constrained Application Protocol</i>)
TCP	Protocolo de Controle de Transmissão (<i>Transmission Control Protocol</i>)
IETF	Força-Tarefa de Engenharia de Internet (<i>Internet Engineering Task Force</i>)
RFC	Solicitação de Comentários (<i>Request for Comments</i>)
QoS	Qualidade de Serviço (<i>Quality of Service</i>)
EF	Entidade Física
EFF	Entidade Física Fabril
EV	Entidade Virtual
IHM	Interface Homem-Máquina
OL(s)	Objeto(s) Lógico(s)
OF(s)	Objeto(s) Físico(s)
IA	Inteligência Artificial
SOX	Estado de X (<i>State-of-X</i>)
SCF/CPS	Sistemas Ciber-Físicos (<i>Cyber-Physical Systems</i>)
IH	Interação Humana
SoS	Sistema de Sistema (<i>System of System</i>)
GDOS	Gêmeo Digital Orientado a Serviços
SGBD	Sistema Gerenciador de Banco de Dados

CPU	Unidade de Processo Central (<i>Central Process Unit</i>)
RAM	Memória de Acesso Aleatório (<i>Random Access Memory</i>)
JSON	Notação de Objeto JavaScript (<i>JavaScript Object Notation</i>)
JSON-LD	Notação de Objeto JavaScript - Dados de Vinculação (<i>JavaScript Object Notation - Linking Data</i>)
LD	Linguagem de Design
API	Interface de Programação de Aplicativos (<i>Application Programming Interface</i>)
Wi-Fi	Fidelidade Sem Fio (<i>Wireless Fidelity</i>)
MCU	Unidade Microcontroladora (<i>MicroController Unit</i>)
ESP	<i>Espressif</i>
IEEE	Instituto de Engenheiros Eletricistas e Eletrônicos
GPIO	Entrada/Saída de Uso Geral (<i>General Purpose Input/Output</i>)
SRAM	Memória Estática de Acesso Aleatório (<i>Static Random-Access Memory</i>)
DTDL	Linguagem de Definição de Gêmeos Digitais (<i>Digital Twins Definition Language</i>)
API	Interface de Programação de Aplicações <i>Application Programming Interface</i>

Sumário

1	INTRODUÇÃO	25
1.1	Contextualização	25
1.2	Motivação	26
1.3	Problemática	26
1.4	Objetivos	28
1.4.1	Objetivo Geral	28
1.4.2	Objetivos Específicos	28
1.5	Publicações	28
1.6	Estrutura da Dissertação	28
2	FUNDAMENTAÇÃO TEÓRICA	31
2.1	Internet das Coisas	31
2.1.1	Protocolos de Comunicação IoT	32
2.1.1.1	<i>Message Queuing Telemetry Transport</i> - MQTT	33
2.1.1.2	<i>Constrained Application Protocol</i> - CoAP	34
2.2	<i>Digital Twin</i>	35
2.2.1	Metodologia	35
2.2.1.1	Planejamento	35
2.2.1.2	Definição dos Portais e <i>String</i> de Busca	35
2.2.1.3	Critérios de Inclusão/Exclusão	35
2.2.1.4	Condução	36
2.2.2	Resultados das Questões de Pesquisa	37
2.2.2.1	QP1: Quais trabalhos revisam o estado da arte de <i>Digital Twin</i> (DT) no contexto de IoT?	37
2.2.2.2	QP2: Quais as definições de DT e os conceitos similares apresentados nos trabalhos?	37
2.2.2.3	QP3: Quais são as principais aplicações de DT apresentadas nos trabalhos?	40
2.2.2.3.1	Indústria	40
2.2.2.3.2	Cidades Inteligentes	42
2.2.2.3.3	Construção Civil	42
2.2.2.3.4	Saúde	43
2.2.2.3.5	Automobilismo	43
2.2.2.3.6	Aeronáutica e Astronáutica	43
2.2.2.4	QP4: Quais são as arquiteturas de DT apresentadas nos trabalhos?	44
2.2.2.4.1	Arquitetura de (MINERVA; LEE; CRESPI, 2020)	44
2.2.2.4.2	Arquitetura de (WU et al., 2020)	45
2.2.2.4.3	Arquitetura de (TAO et al., 2019)	47
2.2.2.4.4	Arquitetura de (STARK; FRESEMANN; LINDOW, 2019)	47
2.2.2.4.5	Arquitetura de (TAO; ZHANG, 2017)	49
2.2.2.4.6	Arquitetura de (JONES et al., 2020)	50
2.2.2.4.7	Arquitetura de (BOJE et al., 2020)	52
2.2.2.4.8	Arquitetura de (QI et al., 2021)	53

2.2.2.4.9	Arquitetura de (LONGO; NICOLETTI; PADOVANO, 2019)	54
2.2.2.4.10	Arquitetura de (LU et al., 2020)	56
2.2.2.4.11	Arquitetura de (FULLER et al., 2020)	56
2.2.2.4.12	Arquitetura de (GRIEVES, 2015)	57
2.2.2.5	QP5: Quais são as linguagens de design de DT apresentadas nos trabalhos?	58
2.2.2.5.1	Conceitos	58
2.2.2.5.2	Grupos de características	58
2.2.2.5.3	Qualidades para o desenvolvimento de uma LD	60
2.2.2.5.4	Exemplo de Linguagem de Design	61
2.3	Plante - Uma Plataforma Inteligente para o Monitoramento e Irrigação Agrícola	61
3	METODOLOGIA	65
3.0.1	Fase 1 - Revisão da literatura	65
3.0.2	Fase 2 - Definição de uma Arquitetura Genérica de DT	65
3.0.3	Fase 3 - Definição de uma Linguagem de Design	66
3.0.4	Fase 4 - Compreensão das formas de avaliação de arquiteturas de DT	66
3.0.5	Fase 5 - Avaliação experimental	66
4	DEFINIÇÃO DE UMA ARQUITETURA GENÉRICA DE DT	67
4.1	Etapa 1: Identificação de Grupos de Camadas e Componentes	67
4.2	Etapa 2: Nomeação dos Grupos Identificados	69
4.3	Etapa 3: Proposta e Detalhamento da Arquitetura Genérica .	70
4.3.1	Fluxo de Dados	70
4.3.2	Entidade Física	71
4.3.2.1	Espaço de Programação	71
4.3.2.2	Hardware	72
4.3.3	Rede e Conexão	72
4.3.4	Entidade Virtual	73
4.3.4.1	Serviços	73
4.3.4.2	Linguagem de Design	74
4.3.4.3	Dados	75
4.4	Etapa 4: Aplicação da Arquitetura Genérica ao Estudo de Caso do Plante	75
5	DEFINIÇÃO DE UMA LINGUAGEM DE DESIGN	79
5.1	Etapa 1: Entendimento do Domínio de Estudo	80
5.2	Etapa 2: Definição do Cenário do Estudo de Caso	80
5.3	Etapa 3: Definição da Linguagem de Design AgroLD	80
5.4	Etapa 4: Integração das Linguagens de Design com o Plante Box	93
5.4.1	Sensors	94
5.4.1.1	AgroLD	94
5.4.1.2	DTDl	96
5.4.2	Actuators	98
5.4.2.1	AgroLD	98
5.4.2.2	DTDl	98

5.4.3	CPU e Battery	99
5.4.3.1	AgroLD	99
5.4.3.2	DTDl	100
5.4.4	Memory e Communication	104
5.4.4.1	AgroLD	104
5.4.4.2	DTDl	105
5.4.5	Functions	108
5.4.5.1	AgroLD	108
5.4.5.2	DTDl	110
6	COMPREENSÃO DAS FORMAS DE AVALIAÇÃO DE ARQUITETURAS DE DT	113
6.1	Etapla 1: Metodologia	113
6.2	Etapla 2: Detalhamento das Arquiteturas	114
6.2.1	Arquitetura de (CATARCI et al., 2019)	114
6.2.2	Arquitetura de (REDELINGHUYS; BASSON; KRUGER, 2020)	114
6.2.3	Arquitetura de (KYCHKIN; NIKOLAEV, 2020)	114
6.2.4	Arquitetura de (BOTTANI; VIGNALI; TANCREDI, 2020)	115
6.2.5	Arquitetura de (AHELEROFF et al., 2021)	115
6.2.6	Arquitetura de (ZHENG et al., 2020)	116
6.2.7	Arquitetura de (DALIBOR et al., 2020)	116
6.2.8	Arquitetura de (MOURTZIS et al., 2021)	116
6.2.9	Arquitetura de (PAN et al., 2021)	116
6.2.10	Arquitetura de (LIM et al., 2021)	117
6.2.11	Arquitetura de (ZOU et al., 2022)	118
6.2.12	Arquitetura de (GAO et al., 2022)	118
6.2.13	Arquitetura de (THÜRER; LI; QU, 2022)	118
6.2.14	Arquitetura de (NWOGU et al., 2022)	119
6.3	Etapla 3: Comparação das Arquiteturas	120
7	AVALIAÇÃO EXPERIMENTAL	121
7.1	Comparativo entre AgroLD e DTDl	121
7.2	Avaliação da Arquitetura Genérica	123
7.2.1	Detalhamento do <i>testbed</i> experimental	123
7.2.2	Cenário 1 - Integração entre as 3 Camadas da Arquitetura	124
7.2.2.1	Cenário Não Saturado (variando o tempo de envio de mensagens)	124
7.2.2.2	Cenário Saturado (variando a quantidade de Entidades Físicas)	128
7.2.3	Cenário 2 - Camada Entidade Virtual	133
7.2.3.1	Consumo de Recursos Computacionais na Entidade Virtual ao Interpretar as Estruturas das Linguagens de Design	133
7.3	Comparação das Propostas de Linguagem de Design e Arquitetura Genérica com Trabalhos Relacionados	136
7.3.1	Linguagem de Design	136
7.3.2	Arquitetura Genérica	136
7.3.2.1	Característica 1 - Propõe arquitetura a partir do estudo comparativo de múltiplas arquiteturas.	137
7.3.2.2	Característica 2 - Apresenta ou cita estudo de caso.	138

7.3.2.3	Característica 3 - Apresenta experimentos.	138
7.3.2.4	Característica 4 - Implementa a arquitetura para realização dos experimentos.	138
7.3.2.5	Característica 5 - Experimenta diferentes cenários.	138
7.3.2.6	Característica 6 - Experimenta, isoladamente, uma ou mais camadas / componentes da arquitetura.	138
7.3.2.7	Característica 7 - Experimenta, de forma integrada, diferentes camadas / componentes da arquitetura.	139

8	CONCLUSÕES	141
----------	-----------------------------	------------

	REFERÊNCIAS	143
--	------------------------------	------------

	APÊNDICES	153
--	------------------	------------

	APÊNDICE A – AGROLD, A LINGUAGEM DE DESIGN (LD) PRO- POSTA, APLICADA AO PLANTE BOX	155
--	---	------------

	APÊNDICE B – DIGITAL TWINS DEFINITION LANGUAGE (DTD), APLICADA AO PLANTE BOX	161
--	---	------------

	APÊNDICE C – MENSAGENS NECESSÁRIAS PARA A DEFINIÇÃO DOS VALORES PADRÃO DOS COMPONENTES CPU, BATERIA, MEMÓRIA E COMUNICAÇÃO NA DTD	183
--	--	------------

	APÊNDICE D – TELEMETRIA DE REQUISIÇÃO	187
--	--	------------

	APÊNDICE E – TELEMETRIA DE RESPOSTA	189
--	--	------------

1 Introdução

1.1 Contextualização

Na última década, houve um aumento considerável na quantidade de dispositivos conectados à Internet. Um estudo realizado pela *IoT Analytics* - um provedor líder de percepções de mercado para a Internet das Coisas (*Internet of Things* - IoT), *Machine to Machine* (M2M) e Indústria 4.0 - estima que em 2021 haviam 23,9 bilhões de dispositivos conectados, em contraste com 9,7 bilhões em 2011 (IOT-ANALYTICS, 2020). Desse total, 1,1 bilhão (em 2011) e 13,8 bilhões (em 2021) foram categorizados como dispositivos IoT, o que representa um crescimento de 1.254,5% de dispositivos da categoria IoT. Além disso, a *IoT Analytics* espera que em 2025 o número de dispositivos conectados à rede seja de 30,9 bilhões, indicando que a área só tende a crescer.

Outra pesquisa, realizada pela AI Multiple (MULTIPLE, 2022), aponta os investimentos financeiros globais com a área de IoT. A pesquisa destaca que, enquanto em 2018 e 2020 foram gastos, respectivamente, 646 e 749 bilhões de dólares na área, prevê-se que em 2023 e 2027, os investimentos sejam de US\$ 1,1 trilhão e mais de US\$ 2 trilhões, respectivamente. Além disso, o estudo destaca que as áreas de manufatura, transporte/logística e serviços públicos foram as que mais tiveram investimento em IoT no ano de 2020. Cada uma das áreas recebeu US\$ 40 bilhões.

Além disso, Statista (STATISTA, 2020) destaca que a área de IoT foi responsável por gerar, a nível mundial, um volume de dados de 13,6 zettabytes em 2019. A pesquisa prevê 79,4 zettabytes sejam gerados em 2025, um aumento percentual de 583,8% em apenas 6 anos.

Devido ao grande crescimento no número de dispositivos de IoT, aos investimentos volumosos e ascendentes, e ao aumento no volume de dados relativos à Internet das Coisas, é perceptível a relevância dessa temática para o mundo contemporâneo, bem como seu potencial de pesquisa e desenvolvimento por parte da comunidade acadêmica e do mercado.

Em especial, a IoT já vem impactando diretamente a vida cotidiana das pessoas, seja na forma contemporânea de como muitas pessoas passaram a monitorar sua saúde, utilizando dispositivos vestíveis - *smartwatches*, *smartbands* - que monitoram pressão cardíaca, oxigenação no sangue, calorias queimadas por dia, quantidade de passos dados etc (PIZZA et al., 2016); seja através da utilização da IoT para automação de atividades domésticas simples, como regular o ar condicionado de acordo com a temperatura ambiente ou limpar o chão da casa através de dispositivos construídos para essa finalidade (PALACIN et al., 2004). As empresas também têm se beneficiado com a IoT, com elevação do nível de segurança, através da utilização de dispositivos de reconhecimento facial e leitura de dados biométricos (M.S. RANA S.P., 2019) para o acesso de ambientes restritos a funcionários, por exemplo. Além do que hoje já existe, a área de IoT tem potencial para inovações futuras.

1.2 Motivação

Dentro do contexto de IoT, um tema que vem ganhando destaque ultimamente é o *Digital Twin* (DT), ou Gêmeo Digital, em português. A ideia do DT foi originalmente concebida por Michael Grieves e apresentada em 2003 na Universidade de Michigan (GRIEVES, 2005). O termo remete, basicamente, a uma representação ou cópia virtualizada exata, denominada Entidade Virtual (EV), de uma Entidade Física (EF) (WEI et al., 2017). Essa tecnologia é utilizada no ramo de IoT no processo de trocas contínuas de informações entre ambientes virtuais e físicos, no qual se deseja manter atualizado o estado, em tempo real, de uma entidade. Um dos objetivos do DT é a facilitação de atividades de monitoramento ou controle automático de dispositivos IoT (VUKOVIĆ et al., 2021).

Para facilitar o entendimento, pode-se imaginar que uma determinada empresa de aviação deseja que sua torre de controle aéreo monitore em tempo real uma aeronave que está em pleno voo. Para isso, os profissionais de Tecnologia da Informação (TI) da empresa podem construir um clone virtual (um *Digital Twin*) do avião, no qual o seu estado (como uma pane no motor, um nível de gasolina abaixo do esperado, alguma falha no casco externo, bem como as características gerais da aeronave) é sincronizado em tempo real com o DT que está localizado na torre. Esse mecanismo permite que os profissionais estejam a par de tudo o que acontece com a aeronave, podendo tomar medidas corretivas, preventivas, ou mesmo utilizar algum mecanismo de Inteligência Artificial (IA) para automatizar determinados processos.

Vislumbrada a potencialidade de aplicação do DT, não é de se estranhar que este seja um tema que tem recebido grande destaque nos últimos anos, tanto no meio acadêmico quanto mercadológico. O consórcio acadêmico da *China Association for Science and Intelligent Manufacturing* selecionou a montagem de manufatura inteligente de DT como um dos 10 principais avanços científicos e tecnológicos em manufatura inteligente (HE; BAI, 2021). Uma das maiores empresas de consultoria e pesquisa de TI do mundo, a Gartner, escolheu o DT como uma das dez principais tendências de tecnologia estratégica em 2019 (GARTNER, 2019). Além disso, a Cisco, uma gigante multinacional da área de TI, definiu o DT como um dos pilares da Indústria 4.0 (CISCO, 2019).

1.3 Problemática

Apesar de sua popularidade, na área de DT ainda há vários temas de Pesquisa e Inovação a serem estudados e desenvolvidos.

Um dos temas existentes mais notáveis diz respeito à falta de padronização arquitetural, não havendo um consenso global em como se construir uma arquitetura de DT (LU et al., 2020). Caracteriza-se como um grande desafio de pesquisa a criação de uma arquitetura genérica (de referência) relevante capaz de suprir a necessidade de uma integração bidirecional entre EF e EV (KOULAMAS; KALOGERAS, 2018). Se houvesse um padrão único de arquitetura de DT, o processo de desenvolvimento/implantação de novas aplicações na área tenderia a ser mais rápido, dispensando a necessidade de projetistas se preocuparem com que tipo de

arquitetura usar. Em outras palavras, a adesão de um padrão unificado garantiria maior domínio e compreensão do usuário, facilitando os estágios de desenvolvimento e implementação de um projeto de DT (FULLER et al., 2020).

Como fica claramente evidenciado em artigos como (TAO; ZHANG, 2017; TAO et al., 2019; LONGO; NICOLETTI; PADOVANO, 2019; LU et al., 2020; WU et al., 2020; AHELEROFF et al., 2021), o maior interesse de pesquisa na área de arquiteturas de DT é focado no meio industrial, onde cada trabalho propõe diferentes camadas/componentes arquiteturais, diferentes nomes e diferentes características. Nota-se que mesmo em arquiteturas propostas para uma mesma área possuem uma série de diferenças entre si. Além disso, aqueles trabalhos que apresentam arquiteturas com cunho mais aberto/genérico, como (GRIEVES, 2015; STARK; FRESEMAN; LINDOW, 2019; MINERVA; LEE; CRESPI, 2020; JONES et al., 2020; BOJE et al., 2020; FULLER et al., 2020; QI et al., 2021), não possuem uma padronização. Três grandes pontos problemáticos nesses trabalhos são a ausência de (i) um estudo comparativo vasto e aprofundado que compare diversas outras arquiteturas para então propor uma nova, (ii) falta de implantação da arquitetura em um estudo de caso concreto para demonstração de sua utilização e (iii) ausência da realização de experimentos que demonstrem como as diferentes camadas e componentes da arquitetura respondem a diferentes cenários. Esta dissertação trata esses três pontos, apresentando um estudo comparativo, implementação da arquitetura e realização de experimentos.

Outro tema relevante de DT é referente à denominada Linguagem de Design (LD), que, conforme será aprofundado ao longo deste trabalho, é responsável pela definição do modelo e das estruturas da EV. A Linguagem de Design tem vários termos sinônimos na literatura: linguagem de definição (MICROSOFT, 2020a), modelo (HAAG; ANDERL, 2018), design (QI et al., 2021), modelo físico (SCHROEDER et al., 2021), “o virtual” (BOJE et al., 2020), dentre outros. Em essência, todos os termos remetem a mesma coisa: a descrição detalhada de uma EF. Destaca-se que a LD não é uma linguagem de programação, e sim uma linguagem de sintaxe, que pode ser escrita através de uma linguagem de notação, como JSON¹ ou XML², por exemplo (BAO et al., 2019), (MICROSOFT, 2020a). Nesse contexto, duas grandes problemáticas são a escassez de trabalhos com LDs (i) de código aberto e completas e (ii) que possuam foco específico em uma área e/ou em um tipo de dispositivo (conforme detalhadamente discutido na Seção 2.2.2.5). Destaca-se que esta dissertação não apenas propõe uma LD em código aberto, mas também a construiu com base em uma área e tipo de dispositivo específico.

Além disso, dentre todos os trabalhos citados por esta dissertação, inclusive os utilizados para montar a Fundamentação Teórica, nenhum propõe uma Linguagem de Design ou, sequer apresenta um esboço de LD. A única exceção é a *Digital Twins Definition Language* (DTDLD), a qual possui documentação aberta e bem detalhada (MICROSOFT, 2020a).

¹ <https://www.json.org/json-pt.html>

² <https://www.xml.com/>

1.4 Objetivos

1.4.1 Objetivo Geral

O objetivo geral desta dissertação é apresentar uma proposta de Arquitetura Genérica e Linguagem de Design de Gêmeos Digitais no contexto de Internet das Coisas.

1.4.2 Objetivos Específicos

Os objetivos específicos desta dissertação são:

- Apresentar uma pesquisa bibliográfica de cunho geral sobre o DT, compreendendo suas definições, aplicações, arquiteturas e linguagens de design.
- Apresentar um estudo de caso na área agrícola para aplicação e avaliação da arquitetura e linguagem de design propostas.
- Propor uma arquitetura genérica de DT para IoT baseada na pesquisa bibliográfica realizada.
- Propor a AgroLD, uma linguagem de design de DT para a área agrícola.
- Apresentar uma revisão sistemática sobre arquiteturas de DT modernas, visando entender como os trabalhos atuais validam propostas de arquitetura de DT.
- Apresentar uma avaliação para a arquitetura proposta, conforme as metodologias encontradas na revisão sistemática, e para a AgroLD.

1.5 Publicações

Foi publicado um artigo intitulado *Um Panorama Comparativo de Arquiteturas de Digital Twin no Contexto de Internet das Coisas* na X Escola Regional de Computação do Ceará, Maranhão e Piauí (ERCEMAPI 2022) (OLIVEIRA; ROCHA; OLIVEIRA, 2022). Ele corresponde ao Capítulo 4.

Foi submetido um artigo, que está em avaliação, intitulado *Arquiteturas de Digital Twin: Uma Revisão da Literatura e uma Proposta de Arquitetura Genérica* para o XLI Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC 2023). Ele corresponde aos Capítulos 6 e 7.

1.6 Estrutura da Dissertação

O restante do documento está estruturado da seguinte forma: no Capítulo 2 é apresentada uma pesquisa bibliográfica de cunho geral sobre o DT, compreendendo suas definições, aplicações, arquiteturas e linguagens de design. Além disso, é detalhado o estudo de caso na área agrícola para aplicação e avaliação da arquitetura e linguagem de design propostas; no Capítulo 3 é apresentada a metodologia utilizada para o desenvolvimento e validação das contribuições desta dissertação; no Capítulo 4 é apresentada uma proposta de arquitetura genérica de DT para IoT baseada na pesquisa bibliográfica realizada; no Capítulo 5 é proposta a AgroLD, uma linguagem

de design de DT para a área agrícola; no Capítulo 6 é apresentada uma revisão sistemática sobre arquiteturas de DT modernas, visando entender como os trabalhos atuais validam propostas de arquitetura de DT; no Capítulo 7 são apresentadas uma comparação entre AgroLD e DTDL e a avaliação da arquitetura proposta conforme as metodologias encontradas na revisão sistemática; por fim, no Capítulo 8 são apresentadas as conclusões desta dissertação e direcionamentos para trabalhos futuros.

2 Fundamentação Teórica

Este capítulo apresenta um estudo acerca de Internet das Coisas (*Internet of Things* - IoT) e *Digital Twin* (DT). São abordados tanto conceitos básicos de IoT e DT, quanto temas mais específicos da área de DT, como aplicações, arquiteturas e linguagens de design. Também é dedicada uma seção para a apresentação da plataforma Plante. O objetivo de capítulo é fundamentar as temáticas importantes para a compreensão da proposta.

2.1 Internet das Coisas

A ideia de Internet das Coisas surgiu no ano de 1999, quando Kevin Ashton, então diretor executivo da Auto-ID Labs do *Massachusetts Institute of Technology* (MIT), idealizou um futuro no qual os computadores possuiriam autonomia para se comunicarem uns com os outros e coletar os dados de seu interesse, sem necessidade de supervisão humana direta (Haghi Kashani et al., 2021) (MADAKAM S.; TRIPATHI, 2015). De 1999 até o atual momento, o conceito sofreu algumas atualizações e refinamentos, não havendo um conceito único e global aceito por toda a comunidade acadêmica e industrial.

Para (GUBBI et al., 2012), a IoT consiste em uma interconexão de dispositivos de sensoriamento e atuação, que oferece a capacidade de compartilhamento de informações entre diferentes plataformas através de uma estrutura unificada, resultando em uma imagem operacional comum para permitir aplicativos inovadores.

Já (LAGHARI A.A., 2021), também cita que a IoT consiste na interconexão de sistemas, sensores, automóveis e muito mais, mas dá ênfase que o resultado dessa interconexão é a criação de grandes quantidades de dados em tempo real que podem ser usadas posteriormente para análise e geração de serviços.

Para (PATEL et al., 2016), a IoT é uma nova revolução da Internet, na qual os objetos tornam-se reconhecíveis e obtêm inteligência ao tomar ou permitir decisões relacionadas ao contexto, graças ao fato de poderem comunicar informações sobre si mesmos. Eles podem acessar informações que foram agregadas por outras coisas ou podem ser componentes de serviços complexos.

De forma resumida, pode-se ver a IoT como um conceito no qual as coisas (dispositivos inteligentes) passam a ser o sujeito ativo da Internet. Diferente do modelo tradicional de Internet, no qual o ser humano era o principal fornecedor e consumidor, na IoT são as próprias coisas que fornecem e consomem o conteúdo.

Em se tratando de arquiteturas de IoT, atualmente, não há uma padrão, apesar de muitos esforços de padronização por parte da área acadêmica e industrial (ČOLAKOVIĆ; HADŽIALIĆ, 2018). Entretanto, uma das arquiteturas mais comuns é a arquitetura de três camadas. Conforme ilustrado na Figura 1, as três camadas são: Percepção, Rede e Aplicação (LOMBARDI M., 2021). A camada de Percepção representa a camada física dos objetos e agrupa todos os objetos de hardware

inteligentes da IoT, como os sensores e os atuadores. A camada de Rede representa a camada de comunicação responsável pela transmissão de dados para a camada de aplicação por meio de diversas tecnologias e protocolos. Essa camada também pode ser responsável pelo armazenamento de dados. Já a camada de Aplicação possui softwares (instalados em *smartphones*, assistentes de voz inteligentes, aplicativos *web* etc.) que oferecem serviços específicos (AL-FUQAHA et al., 2015).

Figura 1 – Exemplo de Arquitetura de Três Camadas da IoT.



Fonte: Autor.

Dentre os benefícios da IoT, há diversos, para as mais diversas áreas. Dentre os benefícios, estão o compartilhamento em tempo real de informações entre diferentes dispositivos inteligentes (em áreas como a indústria, agricultura, automobilismo etc); controle automatizado de linhas de produção gerando redução de custos e desperdícios operacionais e de linhas produtivas; análise automática de saúde de legumes, vegetais, e alimentos em geral; habilitação da computação ubíqua, através de dispositivos vestíveis, sensores de casas inteligentes; gestão automatizada de ativos, monitoramento em tempo real e diversas aplicações de inteligência (ELIJAH et al., 2018), (PAWAR; TRIVEDI, 2019).

Dentre os desafios abertos na área de IoT, pode-se citar custos associados à implantação da infraestrutura IoT; falta de conhecimento adequado sobre as boas práticas e padrões de se implementar estruturas IoT; interferência *wireless* causada quando há uma grande quantidade de sensores sem fio próximos uns dos outros; gerenciamento conciso e consistente de uma grande quantidade de dispositivos, uma vez que a IoT pode lidar com diversos dispositivos de forma simultânea; dificuldades de roteamento e comunicação quando os dispositivos IoT estão em movimento; além de questões de segurança, confiabilidade e escalabilidade (ELIJAH et al., 2018), (PAWAR; TRIVEDI, 2019).

2.1.1 Protocolos de Comunicação IoT

No contexto da IoT, os dois protocolos de comunicação mais populares são o *Message Queuing Telemetry Transport* (MQTT) e o *Constrained Application Protocol*

(CoAP) (ALMHEIRI; MAAMAR, 2021) (THANGAVEL et al., 2014). A seguir, ambos são detalhados. E mais à frente, no Capítulo 7, ambos protocolos serão implantados, experimentados e comparados em um *testbed* real.

2.1.1.1 Message Queuing Telemetry Transport - MQTT

O MQTT, definido pela OASIS (STANDARD, 2019), é um protocolo de transporte de mensagens baseado no paradigma Publicação/Assinatura (do inglês *Publish/Subscribe*). Atualmente, o MQTT se encontra na versão 5.0. É um protocolo leve, aberto, simples e projetado com o intuito de ser fácil de implementar. Tais características o tornam ideal para uso diversificado e em vários contextos, incluindo ambientes restritos (com requisitos de baixo poder de processamento e rede), como para comunicação *Machine to Machine* (M2M) e IoT (SONI; MAKWANA, 2017). O MQTT pode ser executado sobre o *Transmission Control Protocol* (TCP) ou em outros protocolos de transporte que forneçam conexões bidirecionais ordenadas, sem perdas (STANDARD, 2019).

Conforme ilustra a Figura 2, o MQTT utiliza uma arquitetura que inclui dois tipos de entidades: os Clientes MQTT e os *Brokers* MQTT (OASIS, 2020).

Figura 2 – Arquitetura Publicação/Assinatura do MQTT.



Fonte: Adaptada/Traduzida de (OASIS, 2020).

O cliente é um programa ou dispositivo que usa MQTT. Um cliente tem alguns papéis: 1) abre a conexão de rede com o servidor; 2) publica mensagens que podem interessar a outros clientes; 3) inscreve-se para solicitar mensagens que está interessado em receber; 4) cancela a assinatura para remover uma solicitação de mensagens; e 5) fecha a conexão de rede com o servidor.

A Figura 2 mostra, do lado esquerdo, um sensor publicando no servidor (através do tópico *Temperatura*) a temperatura coletada. Do lado direito, há um Dispositivo Móvel e um Sistema *Backend* que estão assinando o tópico *Temperatura*, e, portanto, recebendo a atualização de temperatura publicada pelo sensor.

O *Broker* é um programa ou dispositivo que atua como intermediário entre clientes que publicam mensagens e clientes que fizeram assinaturas. Um servidor: 1) aceita conexões de rede de clientes; 2) aceita mensagens publicadas por clientes; 3) processa solicitações de assinatura e cancelamento de assinatura de clientes;

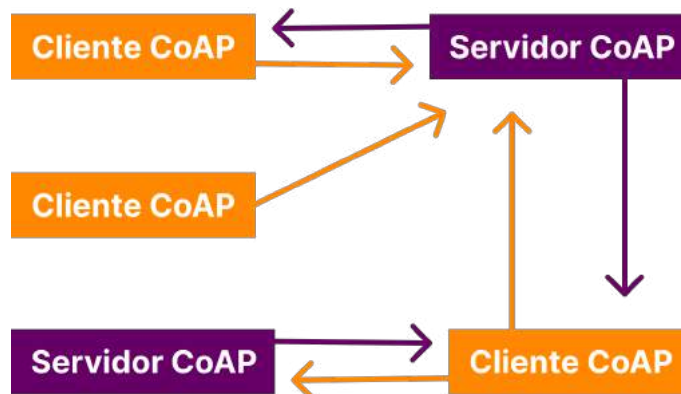
4) encaminha mensagens que correspondem a assinaturas de cliente; e 5) fecha a conexão de rede do cliente.

2.1.1.2 Constrained Application Protocol - CoAP

O *Constrained Application Protocol* (CoAP), definido pela *Request for Comments* (RFC) 7252 (SHELBY KLAUS HARTKE, 2014), do *Internet Engineering Task Force* (IETF), é um protocolo desenvolvido para atuar com o modelo de comunicação M2M, lançado em 2014.

Diferentemente do MQTT, o paradigma de comunicação é o Requisição/Resposta (do inglês *Request/Response*). Nesse modelo, como mostra a Figura 3, há dois agentes, o cliente e o servidor, no qual o primeiro atua como fornecedor de dados e o segundo como consumidor. No CoAP, entretanto, o cliente pode atuar como fornecedor de dados, enviando dados ao servidor para posterior consumo por parte, inclusive, de outros clientes (SHELBY ARM, 2014) (BORMANN, 2016).

Figura 3 – Arquitetura Requisição/Resposta do CoAP.



Fonte: Autor.

Outra diferença do CoAP é que ele usa o *User Datagram Protocol* (UDP) para o transporte, ao invés do TCP comumente usado pelo MQTT. Além disso, o UDP é mais leve e fácil de ser implementado em microcontroladores. Sua comunicação é feita usando *datagramas* sem conexão. O cabeçalho do CoAP é de apenas 4 bytes, e também poder usar características do *broadcast* e *multicast* do UDP (UFRJ, 2016).

O papel dos clientes CoAP (similar ao que ocorre no MQTT) é enviar/ler dados para/do servidor. A principal diferença é que o CoAP utiliza o padrão *Representational State Transfer* (REST) para realizar tal comunicação. Algumas operações definidas pelo padrão REST são: *Get* (através da qual o cliente pode solicitar a leitura de algum dado ao servidor), *Post* (através da qual o cliente pode publicar/criar algum dado no servidor), *Put* (através da qual o cliente pode editar/atualizar um dado existente no servidor) e *Delete* (através da qual o cliente pode solicitar a exclusão de algum dado no servidor) (UFRJ, 2016).

2.2 Digital Twin

A seguir é apresentada a metodologia usada para realização de uma revisão da literatura que tem como objetivo geral realizar um levantamento sobre o estado-da-arte a respeito do tema Gêmeo Digital (*Digital Twin* - DT) no contexto de Internet das Coisas (IoT).

2.2.1 Metodologia

Esta pesquisa de estado-da-arte teve dois principais objetivos: fazer um estudo amplo sobre o tema definido; e detectar oportunidades de pesquisa e desenvolvimento na área em questão.

2.2.1.1 Planejamento

Primeiramente, foram definidas as Questões de Pesquisa (QP) a serem respondidas antes da busca por trabalhos na literatura. As cinco QP que conduziram esse estudo foram:

- **QP1:** *Quais trabalhos revisam o estado da arte de Digital Twin (DT) no contexto de IoT?*
- **QP2:** *Quais as definições de DT e os conceitos similares apresentados nos trabalhos?*
- **QP3:** *Quais são as principais aplicações de DT apresentadas nos trabalhos?*
- **QP4:** *Quais são as arquiteturas de DT apresentadas nos trabalhos?*
- **QP5:** *Quais são as linguagens de design de DT apresentadas nos trabalhos?*

2.2.1.2 Definição dos Portais e *String* de Busca

A busca pelos artigos científicos foi realizada em quatro portais científicos internacionais, listados a seguir:

- ACM Digital Library¹
- IEEE Xplore²
- Science Direct³
- Springer Link⁴

Para obter um *corpus* bibliográfico condizente com o tema especificado foi definida a seguinte string de busca:

("iot" OR "internet of things") AND ("digital twin" OR "digital twins").

2.2.1.3 Critérios de Inclusão/Exclusão

Os critérios de inclusão e exclusão representam uma prática padrão e necessária na elaboração de protocolos de pesquisa de alta qualidade (PATINO; FERREIRA,

¹ <https://dl.acm.org/>

² <https://ieeexplore.ieee.org/Xplore/home.jsp>

³ <https://www.sciencedirect.com/>

⁴ <https://link.springer.com/>

2018). Os parâmetros que determinam se o trabalho deve ou não ser avaliado por esta revisão da literatura são elencados abaixo:

Crítérios de Inclusão

- Trabalhos que utilizem mecanismos de *Digital Twins* no contexto de Internet das Coisas.
- Trabalhos de pesquisa, como surveys, revisões sistemáticas, revisões de literatura, estudos sobre estado-da-arte e afins.

Crítérios de Exclusão

- Trabalhos com menos de 30 citações.
- Trabalhos publicados antes de 2017.

2.2.1.4 Condução

A condução foi efetuada em três etapas descritas a seguir.

Etapas 1 - Aplicação da String de Busca

Nesta etapa foi aplicada a string de busca nos quatro portais, dos quais foram obtidos 1.677 trabalhos científicos para análise primária. No portal ACM *Digital Library* foram obtidos 113 artigos, dos tipos *Proceedings* e *Research Article*; no *IEEE Xplore* foram 255 *papers*, dos tipos *Magazines*, *Conferences*, *Journals* e *Early Access Articles*; no *Science Direct* foram 1380⁵ publicações, dos tipos *Review Articles* e *Research Articles*; e, por fim, no *Springer Link* foram 415 trabalhos, do tipo *Article*.

Etapas 2 - Triagem dos Títulos e Remoção de Duplicatas

Após a separação desse aporte primário de publicações, passou-se para etapa de triagem dos títulos. É válido destacar que os portais de busca aplicam a *string de busca* em todo o corpo do artigo, e não apenas no título e/ou *abstract*, então há artigos que citam o *Digital Twin*, mas que não o têm como tema principal. Diante disso, nesta etapa foi verificado se o título do artigo continha um dos seguintes termos: (“digital twin” OR “digital twins”).

Após a triagem e remoção de duplicatas foram obtidos 418 artigos, aproximadamente 25% dos trabalhos da Etapa 1. Foram 27 do ACM, 120 do *IEEE Xplore*, 215 do *Science Direct* e 58 do *Springer*.

Etapas 3 - Análise das Citações e Tipos de Trabalho

Como dito na Seção 2.2.1.3 só foram levados em consideração trabalhos com 30 ou mais citações, o que resultou em, exatamente, 60 trabalhos. Após a obtenção dos 60 trabalhos que atendiam ao critério do número de citações passou-se para análise dos tipos de trabalhos. Aqui foram excluídos os trabalhos que propunham uma solução aplicada de *Digital Twin* (DT) e foram levados em consideração aqueles

⁵ Só foi possível analisar os 1000 primeiros artigos, pois o site só ofereceu uma paginação com 10 páginas, contendo 100 artigos cada. Logo, os últimos 380 não foram exibidos pelo site, impossibilitando o acesso a eles. Foi enviado um *feedback* para o *Science Direct* reportando o *bug*!

com maior foco em pesquisa, como *surveys*, revisões sistemáticas, revisões de literatura, estudos sobre estado-da-arte e afins. Ao final dessa etapa chegou-se a 28 trabalhos.

Destaca-se que, embora tenham sido selecionados 28 artigos na revisão da literatura, também foram retiradas informações de outros artigos citados por esses 28 trabalhos. Isso deve-se ao fato de que eles eram compostos integralmente por revisões sistemáticas, pesquisas de estado da arte, *surveys* e revisões bibliográficas em geral; logo, a maior parte de seu conteúdo provinha de citações de outros artigos.

2.2.2 Resultados das Questões de Pesquisa

2.2.2.1 QP1: Quais trabalhos revisam o estado da arte de *Digital Twin* (DT) no contexto de IoT?

Respondendo à QP1, a Tabela 1 reflete o número sequencial, o ano de publicação, o número de citações e o título (+ referência) dos 28 trabalhos que compõem o *corpus* primário desta pesquisa. Um detalhe interessante a se destacar é que todos os artigos selecionados dentro dos critérios definidos na Seção 2.2.1.1 foram publicados com título e conteúdo integralmente em inglês.

2.2.2.2 QP2: Quais as definições de DT e os conceitos similares apresentados nos trabalhos?

A ideia do DT foi originalmente concebida por Michael Grieves e apresentada em 2003 na Universidade de Michigan (GRIEVES, 2005). O termo remete, basicamente, a uma representação virtual de uma entidade física (WEI et al., 2017) que permite a troca contínua de informações entre os mundos virtual e físico, facilitando atividades como monitoramento (VUKOVIĆ et al., 2021) e controle automático de dispositivos IoT.

A Figura 4 exemplifica o DT de um *Smartphone*, o qual está fisicamente localizado no mundo físico, mas que possui uma representação virtualizada de suas características - tais como CPU, memória, tela e bateria - no mundo virtual. Destaca-se que ambos os mundos se comunicam de forma bidirecional.

A partir do estudo das definições foram encontrados 6 conceitos-chave comumente utilizados pelos autores para definir o conceito de DT, são eles:

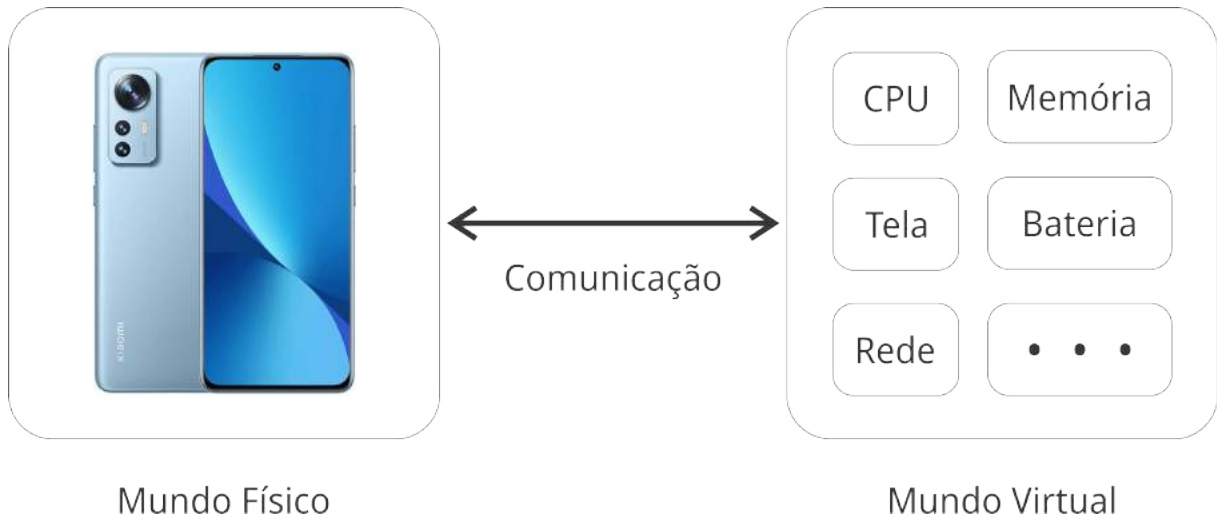
1. **Entidade Física (EF):** um objeto/ambiente físico ou um ser vivo, presente no mundo físico/real.
2. **Entidade Física Fabril (EFF):** funcionários, sistemas, processos, equipamentos ou fenômenos fabris específico do ramo industrial e manufatureiro. A diferença deste conceito para o anterior, é que este é usado especificamente no contexto da indústria, enquanto o anterior é genérico.
3. **Entidade Virtual (EV):** o gêmeo/clone digital (DT) da EF/EFF, presente no mundo virtual, atualizado em tempo real com base em mudanças na EF/EFF.

Tabela 1 – Número sequencial, ano de publicação, quantidade de citações e título (+ referência) dos 28 trabalhos que compõem o *corpus* primário.

Nº	Ano	Citações	Título (+ referência)
1	2021	66	<i>Digital twin-based sustainable intelligent manufacturing: a review</i> , (HE; BAI, 2021)
2	2021	53	Review of <i>Digital Twin</i> about concepts, technologies, and industrial applications, (LIU et al., 2021)
3	2021	47	<i>Disaster City Digital Twin: A vision for integrating artificial and human intelligence for disaster management</i> , (FAN et al., 2021)
4	2020	265	<i>Digital Twin-driven smart manufacturing: Connotation, reference model, applications and research issues</i> , (LU et al., 2020)
5	2020	133	<i>Digital Twin: Enabling Technologies, Challenges and Open Research</i> , (FULLER et al., 2020)
6	2020	124	<i>Characterising the Digital Twin: A systematic literature review</i> , (JONES et al., 2020)
7	2020	94	<i>Towards a semantic Construction Digital Twin: Directions for future research</i> , (BOJE et al., 2020)
8	2020	45	<i>How to tell the difference between a model and a digital twin</i> , (WRIGHT; DAVIDSON, 2020)
9	2020	38	<i>Digital Twin in the IoT Context: A Survey on Technical Features, Scenarios, and Architectural Models</i> , (MINERVA; LEE; CRESPI, 2020)
10	2020	32	<i>The Digital Twin of discrete dynamic systems: Initial approaches and future challenges</i> , (GANGULI; ADHIKARI, 2020)
11	2020	30	Battery digital twins: Perspectives on the fusion of models, data and artificial intelligence for smart battery management systems, (WU et al., 2020)
12	2019	204	Digital Twins and Cyber-Physical Systems toward Smart Manufacturing and Industry 4.0: Correlation and Comparison, (TAO et al., 2019)
13	2019	142	Enabling technologies and tools for digital twin, (QI et al., 2021)
14	2019	107	<i>Review of Digital Twin applications in manufacturing</i> , (CIMINO; NEGRI; FUMAGALLI, 2019)
15	2019	105	<i>A Survey on Digital Twin: Definitions, Characteristics, Applications, and Design Implications</i> , (BARRICELLI; CASIRAGHI; FOGLI, 2019)
16	2019	99	<i>A state-of-the-art survey of Digital Twin: techniques, engineering product lifecycle management and business innovation perspectives</i> , (LIM; ZHENG; CHEN, 2020)
17	2019	74	<i>Digital Twin: Vision, Benefits, Boundaries, and Creation for Buildings</i> , (KHAJAVI et al., 2019)
18	2019	35	<i>Ubiquitous knowledge empowers the Smart Factory: The impacts of a Service-oriented Digital Twin on enterprises performance</i> , (LONGO; NICOLETTI; PADOVANO, 2019)
19	2018	1076	<i>Digital twin-driven product design, manufacturing and service with big data</i> , (TAO et al., 2018)
20	2018	512	<i>Digital Twin in manufacturing: A categorical literature review and classification</i> , (KRITZINGER et al., 2018)
21	2018	168	<i>Digital Twin Service towards Smart Manufacturing</i> , (QI et al., 2018)
22	2018	106	Part data integration in the Shop Floor Digital Twin: Mobile and cloud technologies to enable a manufacturing execution system, (Urbina Coronado et al., 2018)
23	2018	77	<i>Exploring the role of Digital Twin for Asset Lifecycle Management</i> , (MACCHI et al., 2018)
24	2018	64	<i>From Surveillance to Digital Twin: Challenges and Recent Advances of Signal Processing for Industrial Internet of Things</i> , (HE; GUO; ZHENG, 2018)
25	2018	37	<i>Cyber-Physical Systems and Digital Twins in the Industrial Internet of Things [Cyber-Physical Systems]</i> , (KOULAMAS; KALOGERAS, 2018)
26	2017	573	<i>A Review of the Roles of Digital Twin in CPS-based Production Systems</i> , (NEGRI; FUMAGALLI; MACCHI, 2017)
27	2017	432	<i>Digital Twin Shop-Floor: A New Shop-Floor Paradigm Towards Smart Manufacturing</i> , (TAO; ZHANG, 2017)
28	2017	240	<i>The Digital Twin: Demonstrating the Potential of Real Time Data Acquisition in Production Systems</i> , (UHLEMANN et al., 2017)

Fonte: Autor.

- Dados:** *Big Data* composto por dados em tempo real coletados pela EF/EFF e enviados à EV; histórico do ciclo de vida da EF/EFF usado para aprimorá-la; e dados sobre os aspectos físicos, geométricos, comportamentais e contextuais relativas à EF/EFF que resultam na modelagem/design da EV.

Figura 4 – Exemplo de *Digital Twin* de *Smartphone*.

Fonte: Autor.

5. **Comunicação/conexões:** meios e tecnologias de comunicação de IoT através dos quais os dados são transmitidos. A comunicação sempre ocorre de forma bidirecional.
6. **Serviços:** diversos tipos de serviços que podem ser fornecidos com base na interação dos componentes anteriores.

Baseado nesses conceitos-chave, foram extraídos dois conceitos predominantes. Um genérico e outro com foco no ramo industrial.

O primeiro deles é de que o DT é uma EV que representa fielmente e em tempo real uma EF. Ambas entidades possuem comunicação bidirecional uma com a outra visando a sincronização de estado e geração de dados para a criação e prestação de serviços (HE; BAI, 2021), (SADDIK, 2018), (FAN et al., 2021), (MOHAMMADI; TAYLOR, 2017), (ANGRISH et al., 2017), (JONES et al., 2020), (BOJE et al., 2020), (WRIGHT; DAVIDSON, 2020), (MINERVA; LEE; CRESPI, 2020), (TAO et al., 2019), (GRIEVES, 2015), (QI et al., 2021), (TAO et al., 2018), (BARRICELLI; CASIRAGHI; FOGLI, 2019), (KRITZINGER et al., 2018), (QI et al., 2018), (MACCHI et al., 2018), (TAO; ZHANG, 2017). Essa definição possui um amplo escopo, não limitando o DT a uma área específica (saúde, educação etc.) e nem mesmo limitando a EF a objetos específicos, podendo se tratar não apenas de um equipamento eletrônico, mas, inclusive de um ambiente ou ser vivo.

O outro conceito, com foco industrial (maior campo de atuação do DT), é de que o DT é uma EV que mapeia o ciclo de vida de uma EFF. Ambas entidades possuem comunicação bidirecional uma com a outra visando a sincronização entre EV e EFF - acionada a partir de mudanças operacionais - e geração de *Big Data* para aprimoramento da EFF e prestação de Serviços manufatureiros (BOLTON JANET R. MCCOLL-KENNEDY,), (LIU et al., 2021), (LU et al., 2020), (FULLER et al., 2020), (LIU; MRAD, 2018), (VRABIČ et al., 2018), (MADNI; MADNI; LUCERO, 2019), (MINERVA; LEE; CRESPI, 2020), (GANGULI; ADHIKARI, 2020), (CIMINO; NEGRI;

FUMAGALLI, 2019), (LIM; ZHENG; CHEN, 2020), (KANNAN; ARUNACHALAM, 2018), (KHAJAVI et al., 2019), (EASTMAN et al., 2018), (LONGO; NICOLETTI; PADOVANO, 2019), (Urbina Coronado et al., 2018), (HE; GUO; ZHENG, 2018), (KOULAMAS; KALOGERAS, 2018), (NEGRI; FUMAGALLI; MACCHI, 2017), (BAJAJ; COLE; ZWEMER, 2016). Esta segunda definição, presente em diversos artigos, trata de um DT específico à área industrial, na qual o DT tem maior representatividade no atual momento.

Nota-se que a maior diferença em relação à definição anterior é o enfoque no mapeamento do ciclo de vida de uma EEF, limitada ao escopo industrial. É importante frisar que uma EEF também pode representar um ser vivo ou um ambiente, mas ambos dentro do escopo fabril.

2.2.2.3 QP3: Quais são as principais aplicações de DT apresentadas nos trabalhos?

Dentre as áreas de aplicação citadas na revisão de literatura estão: Aeroespacial, Agricultura, Automobilística, Cidades inteligentes, Construção civil, Saúde, Elétrica, Manufatura/indústria, Naval/náutica, Plataforma de perfuração/petrolífera. Entretanto, não foi encontrada na literatura exemplificação de uso para todas. A seguir, estão descritas aquelas que foram melhor abordadas nos trabalhos revisados.

2.2.2.3.1 Indústria

O principal ramo de aplicação do DT, segundo os trabalhos revisados, é o industrial/manufatureiro. Os artigos revisados demonstraram uma ênfase especial nas etapas do ciclo-de-vida do produto dentro dos processos industriais, são elas: Design, Manufatura/Produção e Serviços.

Dentro da etapa de **Design** foram citadas algumas aplicações, como: 1) Modelagem digital, processo no qual se cria uma EV para posterior construção no mundo físico; 2) Otimização iterativa das especificações do produto, que são implementações de melhorias no modelo à medida que detectadas oportunidades para tal; 3) Gestão de integridade de dados entre EV e EEF, que representa uma verificação constante entre o estado físico e virtual, comparando se ambos estão sincronizados; 4) Simulação digital de adaptabilidade e desempenho, para verificar, virtualmente, se o produto se adapta de maneira adequada e com desempenho adequado ao cenário no qual ele será usado; 5) Avaliação e verificação virtual entre o comportamento real e o esperado, para verificar se no mundo físico o comportamento do produto é igual ao planejado no mundo virtual; 6) Validação de modelos digitais, no qual o usuário final (seja pessoa física, jurídica ou até mesmo um sistema) analisa o modelo e informa se ele está de acordo com o esperado (HE; BAI, 2021), (LIU et al., 2021), (LIM; ZHENG; CHEN, 2020), (KRITZINGER et al., 2018), (NEGRI; FUMAGALLI; MACCHI, 2017).

Na etapa de **Manufatura** (também chamada de Produção) foram elencadas as seguintes aplicações: 1) Monitoramento em tempo real, para acompanhar o estado e funcionamento do equipamento através da EV; 2) Previsão de estados futuros, baseado na análise de *big data* através de métodos de inteligência artificial implementados na EV; 3) Gerenciamento, controle e otimização de linhas de produção

inteligentes / digitais / autônomas, também realizado através de mecanismos de inteligência computacional de análise de *big data* via EV; 4) Controle de produção através de operações pré-programadas, no qual se controla EFF através de programação; 5) Reação inteligente a distúrbios e inconsistências, através da análise das reações da EV é possível detectar distúrbios em sua contraparte física (EF/EFF) e corrigi-los.; 6) Previsão de desempenho da peça, baseado em estatísticas testadas na EV; 7) Colaboração e interação humano-robô, permitindo operações manufatureiras manuais e automáticas, permitindo, dentre outras coisas, que o ser humano possa interromper a execução programada do algoritmo e fazer correções ou controles manuais; 8) Análise do comportamento e guia de operação do usuário, utilizado para obter as operações realizadas dos usuários e / ou orientar o usuário para visualizar as atualizações do sistema com uma IHM (Interface Homem-Máquina) de fácil utilização; 9) Avaliação e otimização de processos, quando o gêmeo digital interage com seu ambiente de manufatura por meio de interfaces de interoperabilidade, avaliando e otimizando a combinação de processos e parâmetros; 10) Gerenciamento de ativos, para espelhar de forma dinâmica, realista e precisa o andamento do trabalho e o status dos ativos de manufatura, gerenciar a degradação dos ativos; 11) Planejamento e simulação do processo de produção, possibilitando um plano de produção otimizado globalmente de acordo com a mudança de status em tempo real; 12) Monitoramento do status do equipamento, monitoramento e visualização em tempo real dos processos e equipamentos. Isso permite a detecção de falhas e comportamentos anômalos, além de contribuir para o processo de tomada de decisão e otimização da produção; 13) Análise de consumo de energia, buscando sempre utilizar métricas que mantenham o equilíbrio entre consumo de energia e desempenho (HE; BAI, 2021), (LIU et al., 2021), (LU et al., 2020), (FULLER et al., 2020), (MINERVA; LEE; CRESPI, 2020), (CIMINO; NEGRI; FUMAGALLI, 2019), (BARRICELLI; CASIRAGHI; FOGLI, 2019), (LIM; ZHENG; CHEN, 2020), (KRITZINGER et al., 2018), (MACCHI et al., 2018), (NEGRI; FUMAGALLI; MACCHI, 2017).

Destaca-se que as modificações e ações que ocorrem na EV influenciam em uma reação na EF(F) e vice-versa, por isso o termo “Gêmeo” Digital. O que acontece com um gêmeo gera reações com o outro gêmeo.

Para a etapa de **Serviços** foram elencadas algumas aplicações similares às etapas anteriores, e outras novas. São elas: 1) Manutenção preditiva, é a aplicação de gêmeo digital mais popular, tanto na pesquisa acadêmica quanto na prática da indústria, desde o início do desenvolvimento de gêmeos digitais até o presente; 2) Manutenção virtual e operações virtuais do produto: dado um ambiente ou software 3D, esses serviços elaboram e mostram as operações ou a estratégia de manutenção ao usuário; 3) Detecção, alerta, notificação e diagnóstico de falhas, monitorar os desvios entre os dados coletados e os valores esperados e executar alguma ação; 4) Monitoramento de estado; 5) Previsão de desempenho; 6) Teste virtual de falha da operação física, pode testar virtualmente certas operações sob circunstâncias em que a falha da operação física leva a grandes perdas e danos; 7) Otimização do índice de produção, com a análise dos dados, pode-se modificar os parâmetros de configuração dos produtos, aprimorando o desempenho e os índices de produção; 8) Gerenciamento do ciclo de vida do produto, gerenciando todas as etapas de

vida do produto; 9) Planejamento de processos inteligentes; 10) Recondicionamento e monitoramento de maquinário; 11) Monitoramento abrangente de máquinas e processos de usinagem, oferecendo suporte de tomada de decisão para humanos; 12) Monitoramento da operação da máquina-ferramenta, previsão da rugosidade da superfície; 13) DTs de impressoras 3d para reconstruir a visualização digital de peças e dados; e 14) Ambiente de teste de Realidade Virtual (RV) habilitado para *Digital Twin* (HE; BAI, 2021), (LIU et al., 2021), (LU et al., 2020), (MINERVA; LEE; CRESPI, 2020), (CIMINO; NEGRI; FUMAGALLI, 2019), (BARRICELLI; CASIRAGHI; FOGLI, 2019), (LIM; ZHENG; CHEN, 2020), (LONGO; NICOLETTI; PADOVANO, 2019), (KRITZINGER et al., 2018), (MACCHI et al., 2018), (NEGRI; FUMAGALLI; MACCHI, 2017).

2.2.2.3.2 Cidades Inteligentes

No contexto de cidades inteligentes foram encontrados três exemplos de aplicações de DT.

A primeira aplicação é utilizar o DT para a prevenção e detecção de desastres (naturais e provocados pelo homem) e crises em cidades inteligentes (FAN et al., 2021). Os autores defendem que a necessidade de respostas a desastres e ações de ajuda humanitária está crescendo rapidamente no planeta, e que o DT pode ajudar ativamente nesse contexto. O funcionamento da solução se daria através de, dentre outras coisas, o monitoramento contínuo de mudanças climáticas de uma cidade ou região através de uma Entidade Virtual aliada a algoritmos preditivos de inteligência artificial. A EF nesse contexto seria a própria cidade ou região.

Para (FULLER et al., 2020), o DT pode ser utilizado para facilitar o crescimento das Cidades Inteligentes ao ser capaz de virtualizar um *testbed* “vivo”, permitindo, ao menos, dois serviços: 1) Teste de cenários e, 2) Permitir que os gêmeos digitais aprendam com o ambiente, analisando as mudanças nos dados coletados, resultando em algoritmos de análise e monitoramento. Percebe-se que esta proposta é bastante similar à elencada por (FAN et al., 2021), embora com contexto mais amplo.

Outra aplicação seria um DT de patrimônios culturais (como praças, igrejas, monumentos históricos em geral) para analisar, acompanhar e monitorar a deterioração e estado de conservação e planejar possíveis reformas, sejam de cunho preventivo ou corretivo (MINERVA; LEE; CRESPI, 2020).

2.2.2.3.3 Construção Civil

Os autores de (KHAJAVI et al., 2019) elencaram três exemplos de aplicações na construção civil, são elas: 1) Gestão e melhoria da eficiência energética de edifícios. Nesse caso o DT ajudaria nos processos de aquecimento/resfriamento e distribuição de luz quando e onde necessário. Ou seja, uma distribuição de recursos inteligente e por demanda; 2) Fornecimento de dados sobre as necessidades de manutenção do edifício, alertando a equipe responsável sempre que uma anomalia for detectada; e 3) Uso da EV pelos arquitetos para melhorar o desempenho de

futuras construções, através de análises de desempenho energético, ergonômico, uso de materiais e outras métricas.

Além do exposto, o planejamento de plantas de construção civil através da EV é uma boa ideia de acordo com (UHLEMANN et al., 2017). Para os autores, a aquisição, armazenamento e análise de dados através da DT também contribuem para a criação e desenvolvimento de cenários de otimização no ramo da construção civil.

Em resumo, a DT não atua apenas como tecnologia apoiadora de planejamento, mas também atua como fator evolutivo, permitindo a melhoria constante nos processos de construção civil.

2.2.2.3.4 Saúde

Para (FULLER et al., 2020) e (BARRICELLI; CASIRAGHI; FOGLI, 2019), a criação de DT de seres humanos é uma boa ideia. Isso resultaria em serviços como análise do corpo em tempo real; simulação de efeitos colaterais de drogas; planejamento e execução de procedimentos cirúrgicos; além da simulação de ambientes hospitalares (atuais e futuros) que melhor se adequassem a determinado tipo de paciente, a depender de seu estado de saúde.

A prevenção de ocorrências de doenças ao analisar o histórico pessoal da EF (nesse caso, o ser humano) e o contexto ambiental no qual a EF está inserida, segundo (BARRICELLI; CASIRAGHI; FOGLI, 2019), é uma possibilidade não muito distante graças aos avanços na área de DT.

2.2.2.3.5 Automobilismo

Monitoramento e controle de automóveis movidos a baterias utilizando DT já são uma realidade hoje em dia (WU et al., 2020). Serviços específicos inseridos nesse contexto são histórico de carga e descarga de baterias, predição de hora ideal para a próxima recarga, e otimização de processos mecânicos e de inteligência para redução de consumo de energia. Embora tais serviços tenham sido descritos como aplicados ao ramo automobilístico, eles também podem se aplicar a outros equipamentos alimentados por bateria.

2.2.2.3.6 Aeronáutica e Astronáutica

O *Digital Twin* como habilitador de manutenção, suporte de decisão, otimização e diagnóstico (BARRICELLI; CASIRAGHI; FOGLI, 2019). Um exemplo prático é a detecção de mudanças perigosas na estrutura da aeronave - como uma falha no motor - e, se possível, o acionamento de mecanismos de autocorreção.

Outro exemplo de aplicação é a construção de uma EV de voo de alta fidelidade, com objetivos de permitir (i) a realização de previsão de fadiga, (ii) simulação e avaliação da confiança nas previsões de modelos aerotérmicos, (iii) detecção e monitoramento de danos na estrutura de aeronave, (iv) detecção em tempo real, (v) inspeção, (vi) reparo e (vii) substituição de veículos aeroespaciais (TAO et al., 2018).

2.2.2.4 QP4: Quais são as arquiteturas de DT apresentadas nos trabalhos?

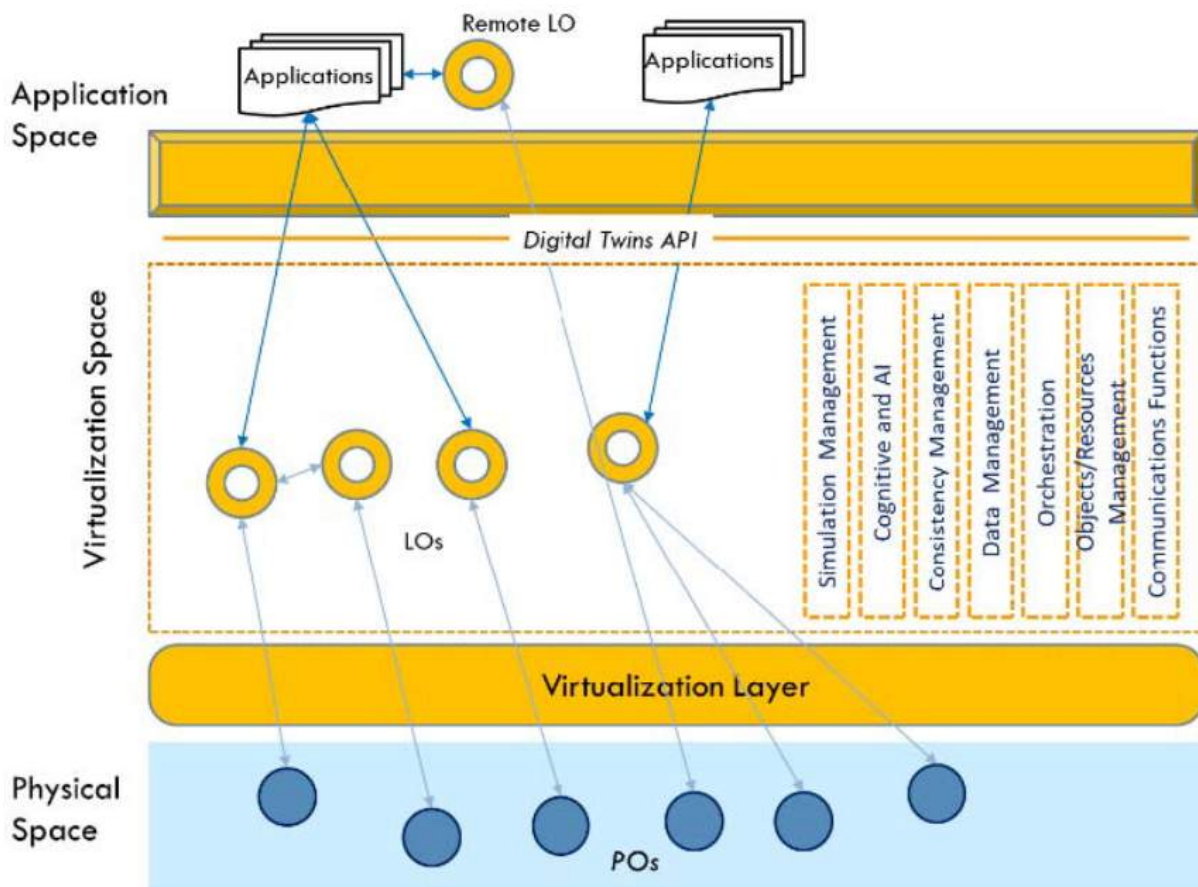
Em meio aos 28 artigos que compõem o *corpus* primário, foram encontradas 13 arquiteturas existentes no contexto de *Digital Twin* para a Internet das Coisas. Destaca-se que, a partir da leitura dos trabalhos, foi detectada como uma problemática da área de DT: a não existência de uma arquitetura genérica, global ou unificada (LU et al., 2020) (FULLER et al., 2020) (KOULAMAS; KALOGERAS, 2018).

2.2.2.4.1 Arquitetura de (MINERVA; LEE; CRESPI, 2020)

Os autores de (MINERVA; LEE; CRESPI, 2020) propõem uma arquitetura baseada em funcionalidades de *middleware* que são implantadas em diferentes sistemas de computação como terminais, *edge* e nuvem. A arquitetura também oferece a possibilidade de programar e estender as funções por meio de *APIs* externas e internas.

Conforme ilustra a Figura 5, a arquitetura possui três componentes: Espaço de Aplicativo (*Application Space*), Espaço de Virtualização (*Virtualization Space*) e Espaço Físico (*Physical Space*).

Figura 5 – Arquitetura baseada em funcionalidades de *middleware*.



Fonte: (MINERVA; LEE; CRESPI, 2020)

- O primeiro componente é o **Espaço de Aplicativo**, no qual estão localizados os Aplicativos (*Applications*) e os Objetos Lógicos Remotos (*Remote*

LO). Os Aplicativos têm comunicação bidirecional com os Objetos Lógicos Remotos presentes neste componente e também os Objetos Lógicos (OLs, ou, em inglês, *LOs*) do componente inferior (o Espaço de Virtualização). Os Objetos Lógicos Remotos têm comunicação bidirecional com os Aplicativos presentes neste componente e também os Objetos Lógicos do componente inferior (Espaço de Virtualização).

O componente 1 é interligado ao componente 2 através de APIs de DT (*Digital Twins API*).

- O segundo componente é o **Espaço de Virtualização**, no qual estão presentes os OLs, ou seja, as versões virtualizadas (os gêmeos digitais) dos objetos do componente inferior (Espaço Físico). Os OLs possuem comunicação bidirecional tanto com os Objetos Físicos (OFs) do componente inferior, quanto com os OLs-Remotos e Aplicações do componente superior. Também estão presentes neste componente alguns serviços, entretanto não há detalhamento sobre seus papéis dentro da arquitetura ou seu funcionamento geral. São o Gerenciamento de Simulação (*Simulation Management*), a Inteligência Artificial (IA) e Cognição (*Cognitive and AI*), o Gerenciamento de Consistência (*Consistency Management*), o Gerenciamento de Dados (*Data Management*), a Orquestração (*Orchestration*), o Gerenciamento de Objetos e Recursos (*Objects/Resources Management*) e as Funções de Comunicação (*Communications Functions*).

Este componente é interligado ao componente inferior através de uma Camada de Virtualização (*Virtualization Layer*).

- O último componente é o **Espaço Físico**, no qual estão os Objetos Físicos, que possuem uma comunicação bidirecional com os Objetos Lógicos do componente superior. Esses objetos podem ser equipamentos eletrônicos inteligentes ou até pessoas e ambientes, a depender da área e tipo de aplicação onde a arquitetura será usada.

2.2.2.4.2 Arquitetura de (WU et al., 2020)

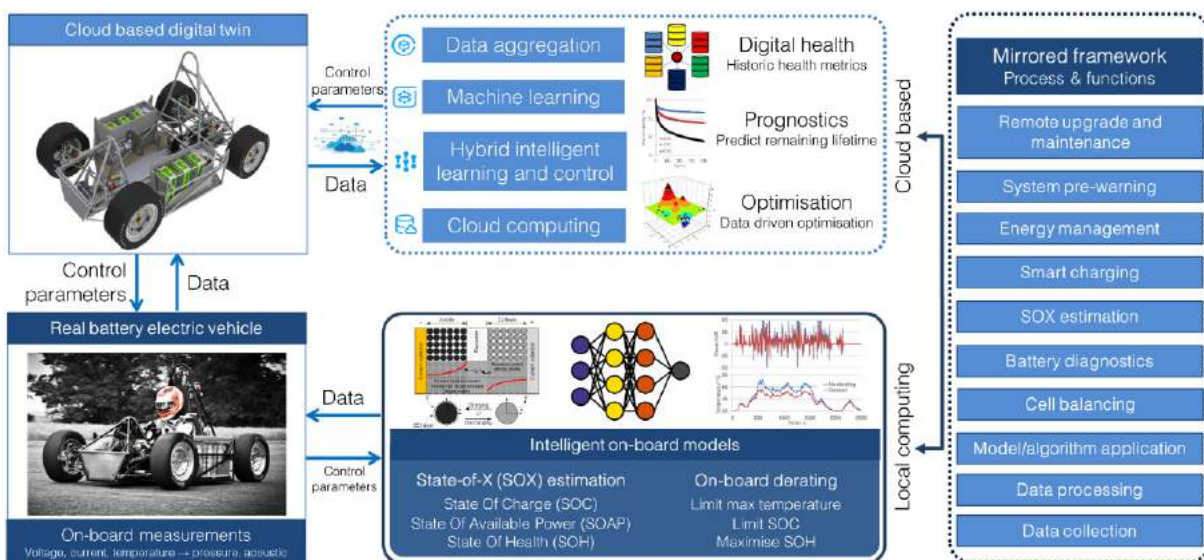
Os autores de (WU et al., 2020) propõem uma arquitetura de DT aplicada ao contexto de gestão de automóveis movidos a Baterias de Lítio.

Conforme ilustra a Figura 6, a arquitetura possui 5 componentes: Computação Local (*Local computing*), Objeto Físico (*Real battery electric vehicle*), Objeto Lógico (*Cloud based digital twin*), Computação em Nuvem (*Cloud based*) e a Estrutura Espelhada (*Mirrored framework*).

- A **Computação Local**, possui “modelos *on-board* inteligentes” que representam as funções de sensoriamento que podem ser executadas localmente, bem como algumas variáveis e funções relativas à bateria. Este componente envia dados ao Objeto Físico e recebe Parâmetros de Controle. A computação local é dividida em dois blocos: o *State-of-X (SOX) Estimation*, responsável por medir o estado de carregamento, estado de bateria disponível e estado de saúde da bateria; e o *On-board derating*, que armazena variáveis como o limite máximo de temperatura, limite de carga e maximização de saúde da bateria.

- O **Objeto Físico** possui “medições *on-board*” que coletam dados contextuais referentes à bateria, como voltagem, corrente e temperatura. Esse componente é o veículo em si. No caso específico desta arquitetura os pontos relevantes definidos pelos autores são todos relativos à bateria, desconsiderando outros diversos componentes que um veículo tem.
- O **Objeto Lógico** corresponde à virtualização do OF. Ou seja, é, de fato, o gêmeo digital do objeto físico.
- A **Computação de Nuvem** é onde está localizada a parte de inteligência e processamento de dados. Possui os seguintes componentes: Agregação de dados (*Data aggregation*), Aprendizado de máquina (*Machine learning*), Aprendizado e controle híbrido inteligente (*Hybrid intelligent learning and control*). Além disso, este componente fornece serviços como histórico de métricas de saúde (*Historic health metrics*), predição de tempo restante de bateria (*Predict remaining lifetime*) e otimização orientada a dados (*data driven optimisation*).
- A **Estrutura Espelhada** representa todas as funções que podem ser extraídas/executadas a partir da interação entre os outros 4 componentes. As funções são atualização e manutenção remota, sistema de pré-aviso, gerenciamento de energia, carregamento inteligente, estimativa de estado, diagnóstico de bateria, balanceamento celular, aplicação de modelo / algoritmo, processamento de dados e coleta de dados.

Figura 6 – Arquitetura aplicada ao contexto de gestão de automóveis movidos a baterias.



Fonte: (WU et al., 2020)

Um detalhe importante desta arquitetura é seu fluxo de troca de informações. Dados de Estado são enviados da Computação Local (passando pelo Objeto Físico e Lógico) até chegar na Computação de Nuvem; enquanto que da Computação de Nuvem são enviados Parâmetros de Controle (passando pelo Objeto Lógico e Físico) até chegar à Computação Local. Isso vai ao encontro de um dos requisitos do DT: a

comunicação bidirecional, na qual a EF e EV mantêm-se atualizadas e sincronizadas e pode-se extrair serviços a partir dessa interação e troca de dados.

2.2.2.4.3 Arquitetura de (TAO et al., 2019)

Os autores de (TAO et al., 2019) propõem uma arquitetura de DT de 3 componentes aplicada ao contexto industrial.

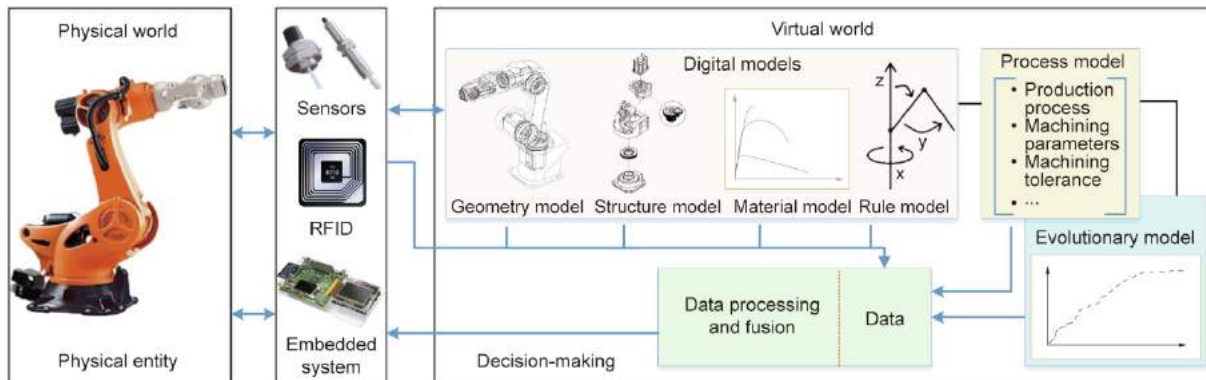
Conforme ilustra a Figura 7, os três componentes são os seguintes: Objeto Físico (*Physical entity*), Sensoriamento e Conexão (que não possui um nome na imagem, mas que contém 3 componentes: *Sensors*, *RFID* e *Embedded system*, os quais são referentes a atividades de sensoriamento e conexão) e Mundo Virtual (*Virtual world*).

- O **Objeto Físico** é o objeto a ser virtualizado, monitorado e controlado. Na arquitetura ele está representado como um braço robótico, demonstrando uma das possíveis representações de EFs.
- O **Sensoriamento e Conexão** possui sensores (responsáveis por coletar dados contextuais) e dispositivos de conexão à rede para se conectar ao Objeto Físico e Mundo Virtual.
- No componente **Mundo Virtual** ocorre todo o processo de análise da dados, inteligência e controle remoto do Objeto Físico. Ele é dividido em 4 componentes: Os Modelos Digitais (*Digital models*) que possuem os modelos geométrico, estrutural, material e de regras. Cada um desses modelos serve para virtualizar/descrever um aspecto do objeto físico. Destaca-se que nem sempre é necessário usar todos esses modelos para implementar uma *Digital Twin*. Qual ou quais modelos devem ser utilizados depende do contexto em que a DT está sendo aplicada. O Modelo de Processo (*Process model*) que descreve, exclusivamente, os processos e detalhes relacionados ao objeto físico, como os processos de produção, os níveis de resistência das peças usadas, dentre outros. O Modelo Evolucionário (*Evolutionary model*) possui o histórico dos dados coletados do Objeto Físico, bem como de alterações em seus modelos. E, por último, a Fusão e Processamento de Dados (*Data processing and fusion*) é responsável por analisar os dados e passar por um processo de tomada de decisão, enviando parâmetros de controle ao componente de Sensoriamento e Conexão que, por sua vez, os repassará ao Objeto Físico.

2.2.2.4.4 Arquitetura de (STARK; FRESEMANN; LINDOW, 2019)

Os autores de (STARK; FRESEMANN; LINDOW, 2019) propõem uma arquitetura de DT com 8 componentes. O objetivo da arquitetura é demonstrar as diferentes escolhas que se pode fazer ao criar uma arquitetura de DT. Por exemplo, os autores descrevem que uma das decisões que se deve tomar é sobre a frequência de atualização da EF e EV, que, de acordo com deles, pode ser imediata (orientada a eventos), a cada hora, a cada dia ou a cada semana. Logo, a proposta dos autores serve para guiar o desenvolvimento de novas arquiteturas a partir da descrição de um conjunto de características pré-definidas.

Figura 7 – Arquitetura de 3 componentes para DT.



Fonte: (TAO et al., 2019)

Conforme ilustra a Figura 8, os 8 componentes são: Amplitude de integração (*Integration breadth*), Modos de conectividade (*Connectivity modes*), Frequência de atualização (*Update frequency*), Inteligência dos Sistemas Ciber-Físicos (CPS) (*CPS Intelligence*), Capacidades de simulação (*Simulation capabilities*), Riqueza do modelo digital (*Digital model richness*), Interação humana (*Human interactions*), Ciclo de vida do produto (*Product life cycle*).

- A **Amplitude de integração** pode ser global, representando a integração completa do objeto; por campo/ambiente de fábrica, representando uma integração apenas a um tipo de ambiente fabril; por campo próximo/sistema de produção, representando uma integração apenas a um sistema específico dentro de um dado ambiente fabril; e por produto/máquina, quando for integrado apenas a um outro determinado objeto.
- Os **Modos de conectividade** podem ser automático, sem necessidade de intervenção humana; direcionada pelo contexto; bidirecional, quando tanto a EF e EV atuam simultaneamente como publicadoras e consumidoras de dados; e uni-direcional.
- A **Frequência de atualização** pode ser imediata/em tempo real, geralmente impulsionada por eventos; por hora; por dia; ou por semana.
- A **Inteligência dos Sistemas Ciber-Físicos (CPS)** pode ser desenvolvida de forma autônoma, com plena atuação cognitiva e independência da atuação humana; parcialmente autônoma, com baixo suporte de Inteligência Artificial (IA) e necessidade, em algumas situações, de intervenção humana; automatizada, independente de controle humano, entretanto, configurado por humanos; e desencadeado/controlado por humanos.
- As **Capacidades de simulação** podem ser do tipo *look-ahead prescriptive*; *ad-hoc*; dinâmica; e estática. Importante destacar que os autores não explicaram ou deram detalhes de como funciona cada um desses tipos de capacidades de simulação.
- A **Riqueza do modelo digital** pode descrever características geométricas, referente ao comprimento, área e volume; cinemáticas, referente à mecânica e aos movimentos da EF; de controle de comportamento, englobando o comportamento cinemático e geométrico; e de comportamento (multi)

físico, englobando o comportamento geométrico, cinemático e lógico.

- O nível de **Interação humana (IH)** pode ser baixo, no caso de dispositivos inteligentes utilizados em *smart houses*, como luzes que ascendem automaticamente ao detectar ausência de iluminação ambiente; pode ser médio, através de realidade virtual/ aumentada, cuja imersão visual é completa; e e alta, através de sistemas híbridos inteligentes que interagem com mais de dois sentidos humanos, como visão, audição e tato.
- O gerenciamento de **Ciclo de vida do produto** pode se limitar ao início; início-meio; ou início-meio-fim/total.

Figura 8 – Arquitetura com 8 componentes para DT.



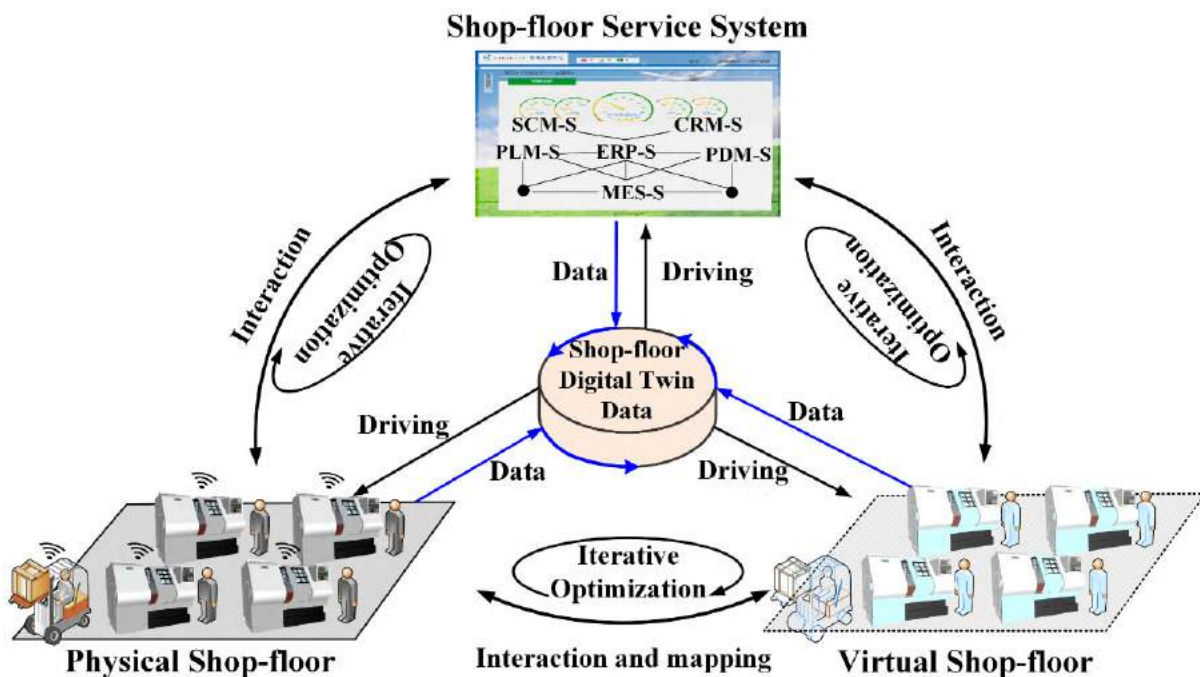
Fonte: (STARK; FRESEMAN; LINDOW, 2019), (citado por (LIU et al., 2021)).

2.2.2.4.5 Arquitetura de (TAO; ZHANG, 2017)

Os autores de (TAO; ZHANG, 2017) propõem uma arquitetura com 5 componentes para o contexto industrial/manufatureiro. A Figura 9 ilustra a arquitetura. Abstraindo os termos específicos de manufatura para termos mais generalistas, tem-se os seguintes componentes: Entidade Física Fabril (*Physical Shop-Floor*), Entidade Virtual (*Virtual Shop-Floor*), Serviços (*Shop-floor Service System*), Base de Dados de DT (*Shop-Floor Digital Twin Data*) e Conexão (*Driving/Data*).

- A **Entidade Física Fabril** engloba várias entidades, como pessoas (funcionários da fábrica), máquinas, veículos, equipamentos e materiais, que existem no espaço físico fabril;
- A **Entidade Virtual** consiste nos modelos virtuais que descrevem em múltiplas dimensões (incluindo geometria, física, comportamento e regra) as EFFs;
- Os **Serviços** são geradas a partir de uma plataforma integrada que disponibiliza ferramentas auxiliadas por computador, modelos e algoritmos em sub-serviços e, em seguida, os combina para formar serviços compostos para demandas específicas das EFFs e EVs;
- A **Base de Dados de DT** inclui dados fundidos dos três componentes anteriores, fornecendo informações abrangentes e consistentes, além de métodos existentes para modelagem, otimização e previsão;
- A **Conexão** descreve as comunicações existentes entre os demais componentes. Os componentes 1, 2 e 3 trocam informações continuamente com o 4. Além disso a EV interage e evolui constantemente com a EFF, fornecendo ordens de controle para a EFF e estratégias de otimização para o componente de serviços. A EFF também interage diretamente com o componente de serviços, mas essa interação não é detalhada.

Figura 9 – Arquitetura de 5 componentes voltada para o ambiente fabril.



Fonte: (TAO; ZHANG, 2017).

2.2.2.4.6 Arquitetura de (JONES et al., 2020)

O trabalho (JONES et al., 2020) apresentou duas arquiteturas (Figura 10) baseadas no fluxo de informações através de quatro componentes padrão. Essas

duas arquiteturas juntas compõem o *Digital Twin*. Uma foi chamada de conexão física para virtual e a outra de conexão virtual para física.

Arquitetura 1 - Físico para Virtual

Conforme o bloco esquerdo da Figura 10, há 4 componentes: Entidade Física (*Physical Entity*), Metrologia (*Metrology*), Realização (*Realisation*) e Entidade Virtual (*Virtual Entity*).

Na arquitetura **Físico para Virtual** as conexões são os meios pelos quais o estado da entidade física é transferido para o ambiente virtual, ou seja, a atualização dos parâmetros virtuais de forma que reflitam os valores dos parâmetros físicos. Os quatro componentes são os seguintes:

- A **Entidade Física** que representa o objeto, ser vivo ou ambiente a ser virtualizado;
- A **Metrologia**, na qual o estado da entidade física é capturado/monitorado/lido. Por exemplo, uma mudança na temperatura de um motor físico é medida usando um termômetro da Internet das Coisas;
- A **Realização**, na qual a diferença entre as entidades físicas e digitais é determinada e a entidade virtual é atualizada em conformidade. Por exemplo: a medição da temperatura é transferida para o ambiente virtual por meio de um serviço da web, um processo virtual determina a diferença de temperaturas entre o objeto físico e o objeto virtual, e então atualiza o objeto virtual de forma que ambas as medidas sejam iguais;
- A **Entidade Virtual** que é o objeto virtual a ser atualizado conforme as mudanças ocorridas na entidade física.

Arquitetura 2 - Virtual para Físico

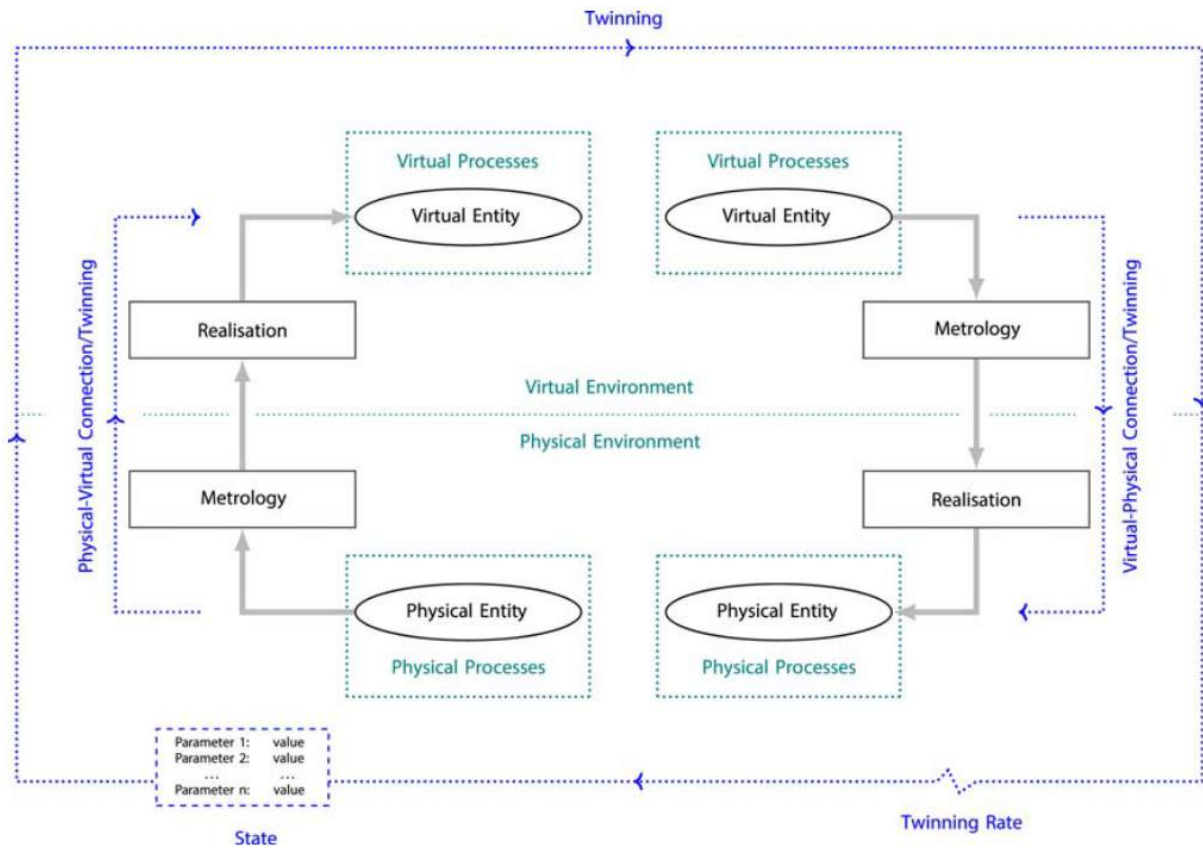
Conforme o bloco direito da Figura 10, há 4 componentes: Entidade Física (*Physical Entity*), Metrologia (*Metrology*), Realização (*Realisation*) e Entidade Virtual (*Virtual Entity*).

Na arquitetura **Virtual para Físico** as conexões são os meios pelos quais se realiza fisicamente uma mudança no estado físico desencadeada por uma mudança no estado virtual. Segundo os autores esse conceito nem sempre aparece nas definições de DT presentes nos trabalhos revisados. Os componentes são descritos a seguir.

- A **Entidade Virtual** é objeto virtual usado para controlar a entidade física.
- A **Metrologia** neste caso, diferentemente da arquitetura anterior, é utilizada para determinar e medir um conjunto ideal de valores de parâmetros dentro de uma entidade física ou ambiente com base em um algoritmo pré-definido. Por exemplo: verifica-se que a temperatura está acima do ideal, e é definido que para regularizar/baixar essa temperatura se faz necessário iniciar um processo de atuação no qual a temperatura será reduzida (ligar o ar-condicionado por exemplo).
- A **Realização** envia uma mensagem de controle ao objeto físico com os parâmetros definidos na metrologia. Exemplo: ligar o ar-condicionado.

- A **Entidade Física** é o objeto a ser controlado e atualizado.

Figura 10 – Arquiteturas Físico-Virtual e Virtual-Físico.



Fonte: (JONES et al., 2020).

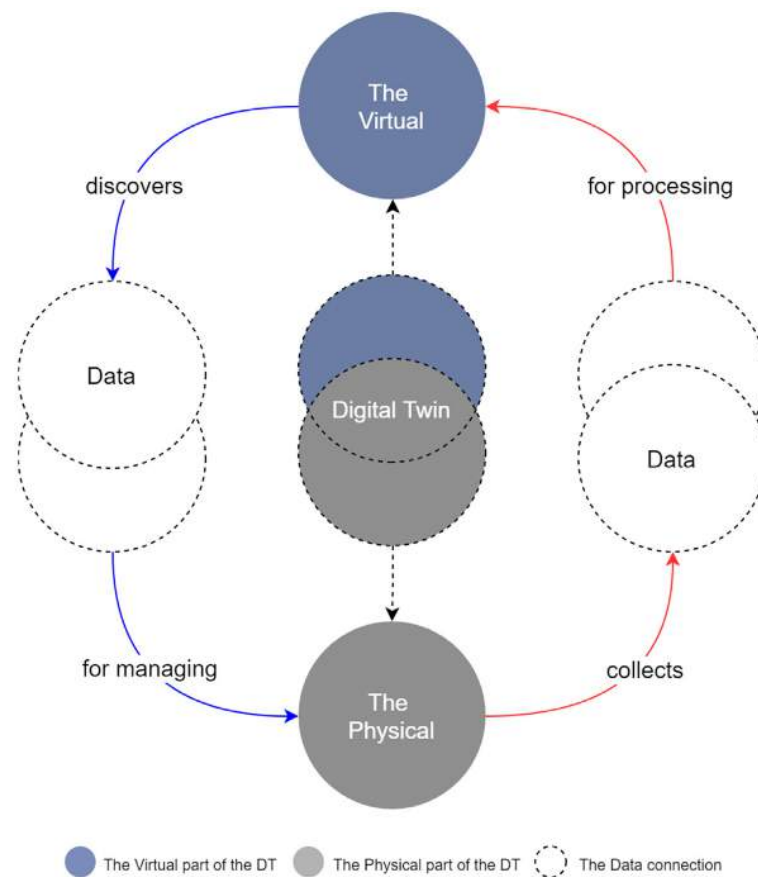
2.2.2.4.7 Arquitetura de (BOJE et al., 2020)

Os autores de (BOJE et al., 2020) propõem uma arquitetura baseada em três componentes. Conforme a Figura 11, os três componentes são chamados de Virtual (*The Virtual*), Dados (*Data*) e Físico (*The Physical*). Essa arquitetura é representada como um “ciclo fechado” entre os três componentes. Na arquitetura proposta o Físico coleta Dados, os envia ao Virtual para serem processados; a Virtual analisa os dados e envia mensagens de controle / gerenciamento para o Físico. O *Digital Twin* é resultado da interação entre esses três componentes. Para cada uma das três partes os autores descreveram as habilidades necessárias.

- O **Físico** possui três habilidades: o sentido, que é a capacidade de observar o mundo físico em tempo real por meio do uso de sensores; o monitoramento, que é a capacidade de rastrear, informar e emitir avisos sobre alterações físicas relevantes; e a atuação, que é a capacidade de alterar / ativar / desativar componentes físicos com base em decisões / estímulos virtuais.
- Os **Dados** possuem três capacidades: a IoT, que é a capacidade de integrar e compartilhar dados comunicados por dispositivos da Internet das Coisas;

- o Link de dados, que é a capacidade de integrar e compartilhar dados por meio de protocolos da Web Semântica; e o armazenamento de conhecimento, que é a capacidade de armazenar fatos sobre o sistema, regras de suporte e capacidades de raciocínio.
- O **Virtual** possui 4 habilidades: simular, que é a capacidade de aplicar modelos de simulação de engenharia de vários domínios de aplicação; prever, a capacidade de prever o comportamento físico com base em simulações e sensores digitais; otimizar, a capacidade de aplicar métodos de otimização e recomendar alocação inteligente de recursos de forma dinâmica; e agência, a capacidade de delegar agentes de IA capazes de gerenciar (e atuar) o físico com base em dados digitais, seguindo comportamentos, protocolos e objetivos bem definidos.

Figura 11 – Arquitetura de três componentes.



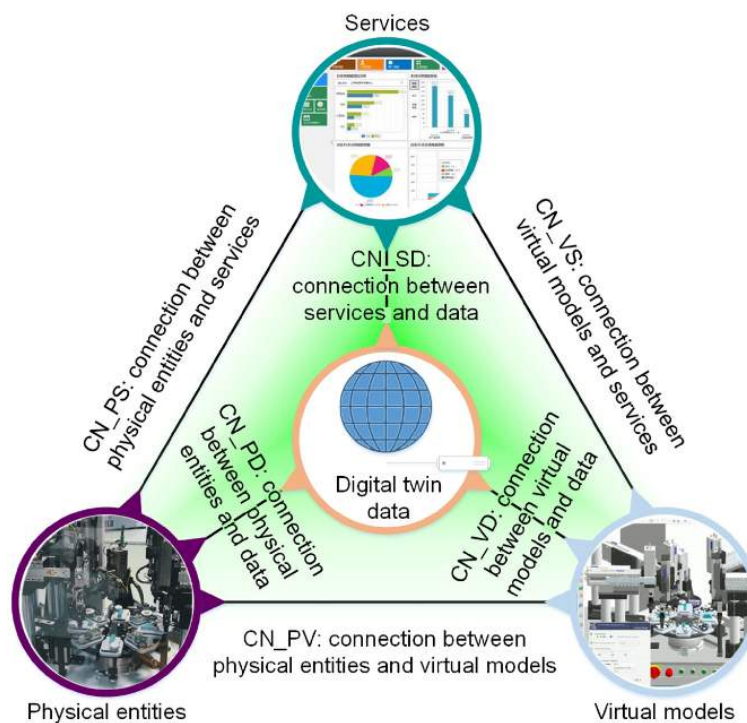
Fonte: (BOJE et al., 2020).

2.2.2.4.8 Arquitetura de (QI et al., 2021)

Os autores de (QI et al., 2021) propõem uma arquitetura com 5 componentes. Conforme a Figura 12, os componentes são Entidades físicas (*Physical entities*), Modelos virtuais (*Virtual models*), Dados (*Digital Twin data*), Serviços (*Services*) e Conexões (as interações entre os componentes anteriores).

- As **Entidades físicas** podem ser divididas em três níveis de acordo com a função e estrutura, que são nível de unidade, nível de sistema e nível de sistema de sistema (SoS).
- Os **Modelos virtuais** são réplicas fiéis de entidades físicas, que reproduzem as geometrias físicas, propriedades, comportamentos e regras.
- Os **Dados** são os principais impulsionadores do DT. O DT lida com dados de escala multi-temporal, multidimensional, multi-fonte e heterogêneo. Alguns dados são obtidos de entidades físicas, incluindo dados de atributos estáticos e dados de condições dinâmicas;
- A DT fornece aos usuários **serviços** de aplicativos relativos a simulação, verificação, monitoramento, otimização, diagnóstico, prognóstico, gerenciamento de saúde etc.
- As **Conexões** são usadas para conectar dinamicamente as representações digitais (modelos virtuais) com suas contrapartes reais para permitir simulação, operação e análise avançadas. As conexões entre entidades físicas, modelos virtuais, serviços e dados permitem a troca de informações e dados. Há 6 tipos de conexões nesta arquitetura: Dados - Entidade Física, Dados - Modelo Virtual, Dados - Serviços, Serviços - Entidade Física, Serviços - Modelo Virtual e Entidade Física - Modelo Virtual.

Figura 12 – Arquitetura de cinco dimensões.



Fonte: (QI et al., 2021).

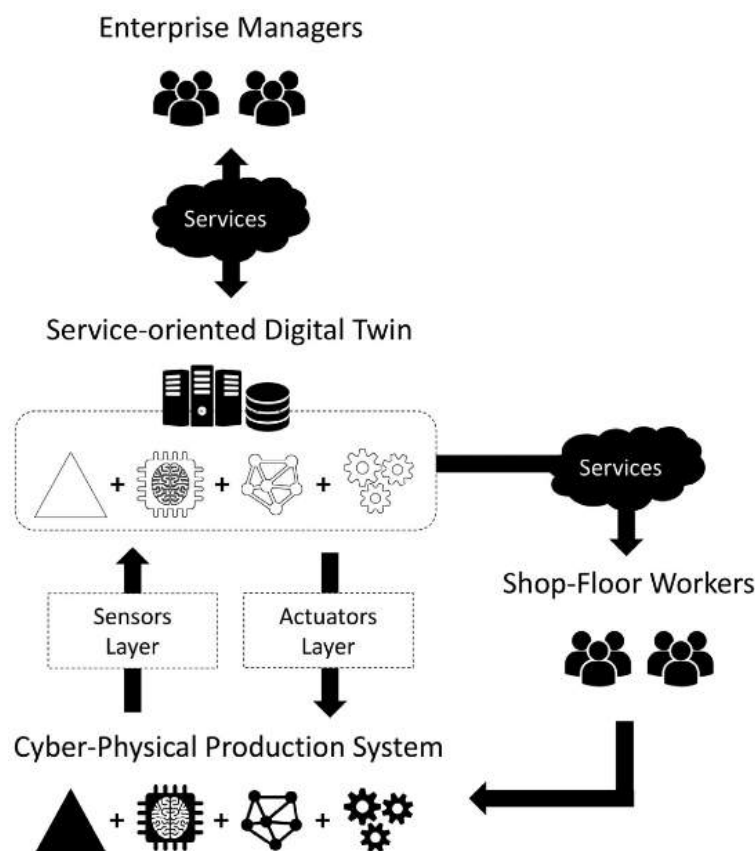
2.2.2.4.9 Arquitetura de (LONGO; NICOLETTI; PADOVANO, 2019)

Os autores de (LONGO; NICOLETTI; PADOVANO, 2019) propõem uma arquitetura de DT com 4 componentes, orientada a serviços. Por meio de aquisição

e análise de dados, os recursos físicos de manufatura podem ser virtualizados e seu *status* e comportamento podem ser rastreados, monitorados e ajustados dinamicamente pelos funcionários de manufatura (gerentes de empresa e operários de fábrica) graças a um conjunto de aplicativos e serviços que podem ser acessados em qualquer lugar e a qualquer hora via internet.

Conforme a Figura 13, os componentes são o Sistema Ciber-Físico (*Cyber-Physical Production System*), Camada de Sensores (*Sensors Layer*), Gêmeo Digital Orientado a Serviços (*Service-oriented Digital Twin*) e Camada de Atuadores (*Actuators Layer*). Além deles, há dois grupos de agentes humanos, que são os Operários (*Shop-Floor Workers*) e Gestores (*Enterprise Managers*).

Figura 13 – Arquitetura orientada a serviços.



Fonte: (LONGO; NICOLETTI; PADOVANO, 2019).

- O **Sistema Ciber-Físico** (SCF) é composto por materiais, ferramentas de máquinas, centros de usinagem, robôs, produtos, recursos de software etc.
- A **Camada de Sensores** é responsável por coletar os recursos do SCF, como atributos básicos, status em tempo real, parâmetros de processo, andamento do processamento, registros de manutenção, dados de operação de produtos, bem como dados de demandas, dados de comportamento e dados de transações de usuários.
- O **Gêmeo Digital Orientado a Serviços** (GDOS) é a representação virtual do SCF, que reside em sistemas de computação em rede (computadores, servidores) e sistemas de armazenamento (discos rígidos). O conhecimento

reside nos sistemas de armazenamento e é encapsulado em diferentes serviços padronizados. Tais serviços representam elementos de informação (ou seja, modelos como serviços, algoritmos como serviços, simulação como serviço, otimização como serviço etc.).

- A **Camada de Atuadores** é acionada/controlada automaticamente ou manualmente via internet/GDOS para realizar algum tipo de atuação no SCF.
- Os **Operários** possuem comunicação bidirecional com o GDOS para conduzir o processo de manufatura e atividades relacionadas (por exemplo, manutenção, definição do plano de trabalho etc).
- Os **Gestores** possuem comunicação bidirecional com o GDOS para desempenhar atividades gerenciais. Alguns serviços fornecidos por esse modelo são: Diagnóstico de Falhas e Monitoramento de Estado, Otimização de prognósticos e cenário, Execução do cenário de fabricação, Notificação de Erros.

2.2.2.4.10 Arquitetura de (LU et al., 2020)

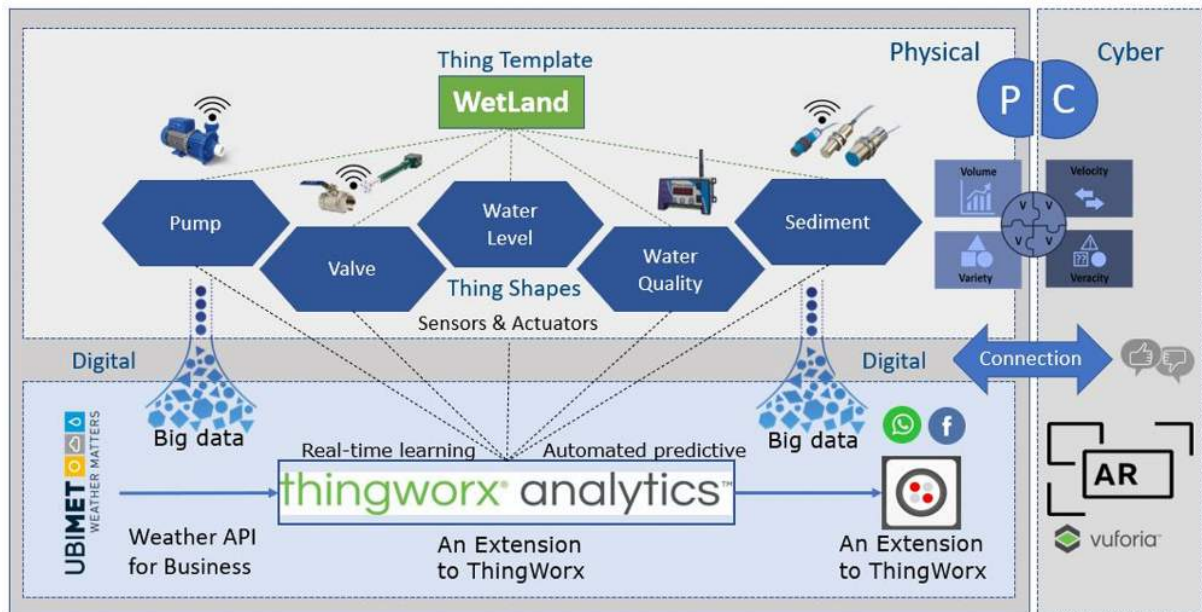
Os autores de (LU et al., 2020) apresentam uma arquitetura de *Digital Twin* com três componentes aplicada ao contexto ambiental no monitoramento de zonas naturais alagadas. Conforme a Figura 14, os componentes são denominados Ciber (*Cyber*), Físico (*Physical*) e Digital.

- O **Físico** é a camada que possui os dispositivos físicos dispostos nas zonas irrigadas. São 5 dispositivos, dentre sensores e atuadores: uma bomba capaz de bombear a água, uma válvula para impedir/permitir a passagem de água, sensores para medir o nível e a qualidade da água, e sensores para analisar os sedimentos. Importante destacar que todos possuem tecnologia de comunicação sem fio para se conectar com a camada Digital.
- A camada **Digital** de modo genérico serve para replicar virtualmente a camada Física. Especificamente ela possui algoritmos de *Big Data* que realizam atividades de aprendizado em tempo real (entendendo os diversos comportamentos do ambiente alagado) e predição automatizada (que visa prever os próximos estados do ambiente em questão). Além disso ela é capaz de controlar os atuadores da camada Física.
- O **Cyber** é uma camada que possui recursos de Realidade Aumentada, possibilitando aos consumidores de tal serviço visualizar em tempo real características do ambiente físico monitorado. para extrair informações de dados heterogêneos de múltiplas fontes visando a construção da representação em tempo real de um objeto físico.

2.2.2.4.11 Arquitetura de (FULLER et al., 2020)

Os autores de (FULLER et al., 2020) apresentam os quatro principais componentes de uma arquitetura de DT.

Figura 14 – Arquitetura de três componentes para monitoramento de áreas alagadas.



Fonte: (LU et al., 2020).

- A **Aplicação** é dividida em Arquitetura e visualização do modelo, utilizada para a visualização e modelagem de arquitetura do objeto virtual, a contraparte do objeto físico; *Software* e API, usados para auxiliar na modelagem dessa arquitetura *Digital Twin* facilitando a terceira camada; e Pré-processamento e coleta, garantindo que os dados sejam coletados corretamente para facilitar o uso de IoT e análise de dados para um *Digital Twin*.
- O **Middleware** é dividido em Tecnologia de armazenamento, que facilita o armazenamento de dados através da utilização de um Sistema Gerenciador de Banco de Dados (SGBD); e Processamento de dados, essencial para a transferência dos dados armazenados entre *Middleware* e Rede.
- A **Rede** é dividido em Camada de tecnologia de comunicação, visando a garantia de que os dados coletados sejam comunicados entre diferentes domínios; e Camada de comunicação sem fio, para garantir que a transmissão de dados sem fio siga o protocolo correto dentro de uma arquitetura *Digital Twin*, além de trazer dados para o próximo domínio, o objeto físico.
- O **Objeto** é composto por uma Plataforma de hardware e Tecnologia de sensores, responsáveis pela coleta de dados em tempo real para, dentre outras coisas, a criação fiel de sua contraparte digital.

2.2.2.4.12 Arquitetura de (GRIEVES, 2015)

O autor de (GRIEVES, 2015) propôs uma arquitetura simples com três componentes: os **Produtos físicos no Espaço Real** equivalentes ao conceito de EF; os **Produtos virtuais no Espaço Virtual**, equivalente ao conceito de EV, através do qual é possível visualizar digitalmente o produto, e testá-lo para avaliar capacidades

de desempenho e as conexões de dados e informações que une os produtos virtuais e reais.

2.2.2.5 QP5: Quais são as linguagens de design de DT apresentadas nos trabalhos?

Visando responder a QP5, a seguir são descritos os conceitos, grupos de características, qualidades para o desenvolvimento de uma LD, bem como citado um exemplo de LD.

2.2.2.5.1 Conceitos

Primeiramente, é importante citar que na literatura podem ser encontrados vários termos sinônimos de Linguagem de Design, tais como: linguagem, modelo, modelagem, design, design de produto, modelo de design, modelo de informações, linguagem de definição, modelo físico, “o virtual” etc. Neste trabalho, foi escolhido o termo Linguagem de Design (LD) como forma de padronização no texto.

A Linguagem de Design é parte fundamental e essencial de um sistema de DT. De forma sucinta, uma LD remete ao conjunto de informações necessárias para criar, descrever, desenhar e modelar uma Entidade Virtual (EV), que atua como gêmeo digital de uma Entidade Física (EF) (QI et al., 2021).

Para entender melhor o conceito de LD, pode-se imaginar um cenário agrícola no qual se objetiva coletar dados climáticos e irrigar plantas através de um dispositivo de IoT (chamado, por exemplo, de *Plante Box*). Dentro desse contexto, o objetivo da LD é montar uma EV do *Plante Box* (que é uma EF) através da descrição detalhada de recursos lógicos (funções que a EF executa), sensores (de temperatura/umidade) e atuadores (irrigador) a ele pertencentes. Além disso, a LD serve para descrever seus objetivos (medir a temperatura/umidade ambiente), suas funções (ligar/desligar o irrigador) e suas regras (se o sensor de umidade indicar valor abaixo de 25% então ligar o irrigador).

Além de definir os recursos, objetivos, funções e regras da EF, a LD é útil para definir quais tipos de dados devem ser sincronizados em tempo real entre EV e EF. Como exemplo, pode-se citar os dados climáticos coletados pelo *Plante Box* (como temperatura, umidade, luminosidade) que necessitam ser enviados à EV para permitir que o agricultor acompanhe em tempo real os dados climáticos de sua plantação.

Pode-se imaginar também uma LD descrevendo recursos de hardware como CPU, memória RAM e secundária, ano de fabricação etc.

Também é possível, através da EV, acessar recursos e controlar remotamente a EF através das informações que a LD fornece à EV.

2.2.2.5.2 Grupos de características

Durante a leitura dos artigos desta revisão bibliográfica, foram encontrados 14 grupos de características que uma LD pode usar para descrever uma EF. São elas:

- Características de **identificação básica**, como nome e identificador, tanto da EF quando da EV (QI et al., 2018).
- Características **físicas ou de forma**, descrevendo detalhes da EF, tais como cor, geometria (dimensões, tamanho e demais coordenadas), desgaste e resistência, tolerância e dureza. Além disso, no contexto industrial, há requisitos de espaço, mecânicos, materiais, hidrodinâmicos e de eventos discretos (LIU et al., 2021), (JONES et al., 2020), (QI et al., 2021), (TAO et al., 2018), (QI et al., 2018).
- Características de **saúde**, que é a diferença entre o estado real e o estado ideal da EF. É possível obter esses dados através de análise e gerenciamento do estado da bateria, CPU, temperatura da EF, dentre outros (JONES et al., 2020).
- Características de **tempo**, como o tempo de ociosidade (tempo em que a EF não faz nada), de processamento e produção (tempo em que a EF está ativa, executando uma atividade), de trabalho (tempo total em que a EF ficou executando alguma tarefa) e exposição (tempo total de vida ociosa e ativa da EF) (JONES et al., 2020), (WRIGHT; DAVIDSON, 2020), (TAO et al., 2018).
- Características de **localização**, referentes à posição geográfica da EF em relação ao ambiente no qual a entidade está (em um campo agrícola), ao *layout* (na quadra A do campo agrícola) e ao local de fabricação (fabricado pela Empresa X) (JONES et al., 2020), (WRIGHT; DAVIDSON, 2020).
- Características de **funcionalidade**, que se refere aos tipos de movimentos que a EF pode executar, ou ao propósito dela. Alguns exemplos são a capacidade funcional (mover um braço mecânico, ativar uma lâmpada), controle (ativar/desativar recursos de hardware, como sensores), parâmetros da máquina (velocidade do fuso, taxa de avanço), modelo de função (diferentes padrões lógicos para se desenvolver funções/rotinas), bioquímico (reações biológicas e/ou químicas causadas pela EF) (JONES et al., 2020) (TAO et al., 2018).
- Características de **processo**, que são as atividades nas quais uma EF está envolvida dentro de, por exemplo, uma linha de produção fabril. Alguns exemplos são parâmetros de programação (sequência, tempo ocioso), modelos de trabalho, detalhes logísticos e informações gerais (JONES et al., 2020).
- Características sobre o **estado** atual medido a partir de todos os parâmetros de entidade e ambiente. O estado pode ser medido através do uso (da CPU, memória e outros recursos de hardware), ambiente (temperatura, umidade), integridade (se a EF sofreu algum dano de hardware ou software), processos (quantos processos estão ativos), estresse humano (no caso de a EF ser um humano) e informações gerais (JONES et al., 2020) (QI et al., 2021) (TAO et al., 2018).
- Características de **desempenho**, que é o nível de operação real em comparação com sua operação ótima/máxima, podendo-se analisar o desempenho parcial (de partes específicas da EF) ou geral (de toda a EF) (JONES et al., 2020), (TAO et al., 2018).

- Características de **ambiente**, tanto o ambiente físico quanto o virtual dentro do qual a EF/EV existe. Em se tratando da EF, pode-se analisar a luminosidade, temperatura, umidade e outras diversas informações (JONES et al., 2020) (KHAJAVI et al., 2019) (TAO et al., 2018).
- Características de **dados qualitativos**, que são informações qualitativas, não mensuráveis por sensores IoT tradicionais, como dados do pedido do produto (tempo de entrega, satisfação do cliente, custos, similaridade entre o produto planejado e o real), alguns requisitos da EF (estabelecidos pelos *stakeholders*), qualificações do funcionário e missão da EF (JONES et al., 2020) (QI et al., 2018).
- Características de **sensores** (através de metadados) como o tipo de sensor e suas capacidades (nível de precisão, incerteza e sensibilidades), data da última calibração do sensor e qualquer outra informação de rastreabilidade, hora da coleta de dados, localização do sensor e nível de bateria (WRIGHT; DAVIDSON, 2020) (KHAJAVI et al., 2019) (TAO et al., 2018).
- Características de **regras** como a decisão de iniciar uma ação com base em uma condição, por exemplo, ligar uma luz (ação) se estiver escuro (condição). É necessário permitir a extração, descrição, associação e evolução dessas regras (QI et al., 2021) (TAO et al., 2018).
- Características dos **proprietários/desenvolvedores** das EFs (através de metadados), sejam os proprietários pessoas físicas, jurídicas, sistemas ou, inclusive, outras EFs. Exemplos de metadados seriam nome, identificação (CPF, CNPJ, ID do sistema), endereço (físico ou endereço IP), dentre outros (QI et al., 2018)

2.2.2.5.3 Qualidades para o desenvolvimento de uma LD

Destaca-se que um único projeto de LD para um contexto específico pode englobar diferentes grupos de características. Além disso, de acordo com os autores de (LU et al., 2020), é comum que um único grupo de características não atenda aos requisitos de um projeto de *Digital Twin*.

Em conformidade com a visão de (LU et al., 2020), os autores de (TAO et al., 2018) e (BARRICELLI; CASIRAGHI; FOGLI, 2019) defendem que as Linguagens de Design devem utilizar uma modalidade de desenvolvimento colaborativo. A ideia é que a LD usada para criar uma EV (que por sua vez é o gêmeo digital de uma EF) deve ser elaborada por uma equipe de desenvolvimento de forma colaborativa por meio da troca e compartilhamento de dados e ideias de design. Os autores acreditam que os diferentes “especialistas em domínio” - usuários finais com diferentes experiências culturais e profissionais - devem atuar em conjunto e colaborar em equipes, aplicando o respectivo conhecimento para ampliar ou modificar um sistema DT. Essa colaboração em grupo e a longo prazo permite satisfazer necessidades e requisitos que não foram identificados no momento do projeto.

Além do desenvolvimento colaborativo, é elencado como um requisito do DT a sincronização em tempo real de informações entre a EV e a EF, havendo a necessidade de as EVs (e, portanto, as LDs) serem leves, enxutas e específicas (LIU

et al., 2021), (KHAJAVI et al., 2019), (HE; BAI, 2021), (LU et al., 2020), (CIMINO; NEGRI; FUMAGALLI, 2019), (BOJE et al., 2020), (MINERVA; LEE; CRESPI, 2020), levando em consideração apenas os dados e modelos relevantes para a finalidade específica a qual se destinam (HAAG; ANDERL, 2018).

Para (GRIEVES, 2015) é necessário criar versões leves da EV, utilizando os grupos de características estritamente necessários sem carregar detalhes dispensáveis, o que reduz drasticamente o tamanho das EVs e permite um processamento mais rápido. Desta forma, permite-se que o processo de germinação (ou seja, a sincronização entre EV e EF) ocorra em tempo real e com custos de computação aceitáveis.

Os autores de (WRIGHT; DAVIDSON, 2020) defendem ainda que a rapidez e a precisão da EV sejam requisitos cruciais, o que não pode ser alcançado com uma EV que contenha detalhes desnecessários, em excesso, genéricos ou pouco específicos. Por exemplo, se detalhes a respeito do fabricante da EF não são utilizados na prática, então eles não devem constar na EV; se, posteriormente, eles passarem a ser necessários, só então deverão ser incluídos.

2.2.2.5.4 Exemplo de Linguagem de Design

Como exemplo de LD é possível citar a *Digital Twins Definition Language* (DTDL), uma LD *open-source* criada pela Microsoft em 2020 (MICROSOFT, 2022a). A DTDL é utilizada para a criação de EVs dentro da plataforma Gêmeos Digitais do Azure, da Microsoft. Atualmente, a DTDL encontra-se na versão 2 e é baseada no padrão *JavaScript Object Notation - Linking Data* (JSON-LD) (MICROSOFT, 2020a).

Pode-se dizer que a DTDL é uma LD genérica, ou seja, não é focada em uma área específica ou em um grupo de características específicas. Em outras palavras, o objetivo da DTDL é permitir a germinação de dispositivos de IoT independente de suas características ou áreas de atuação (MICROSOFT, 2020b). De acordo com a Microsoft é possível utilizar a DTDL para germinação (criação de EVs) de “edifícios, fábricas, fazendas, redes de energia, ferrovias, estádios e muito mais, até mesmo cidades inteiras” (MICROSOFT, 2022b).

A DTDL será utilizada como LD comparativa à Linguagem de Design proposta nesta dissertação, a AgroLD (Capítulo 5).

2.3 Plante - Uma Plataforma Inteligente para o Monitoramento e Irrigação Agrícola

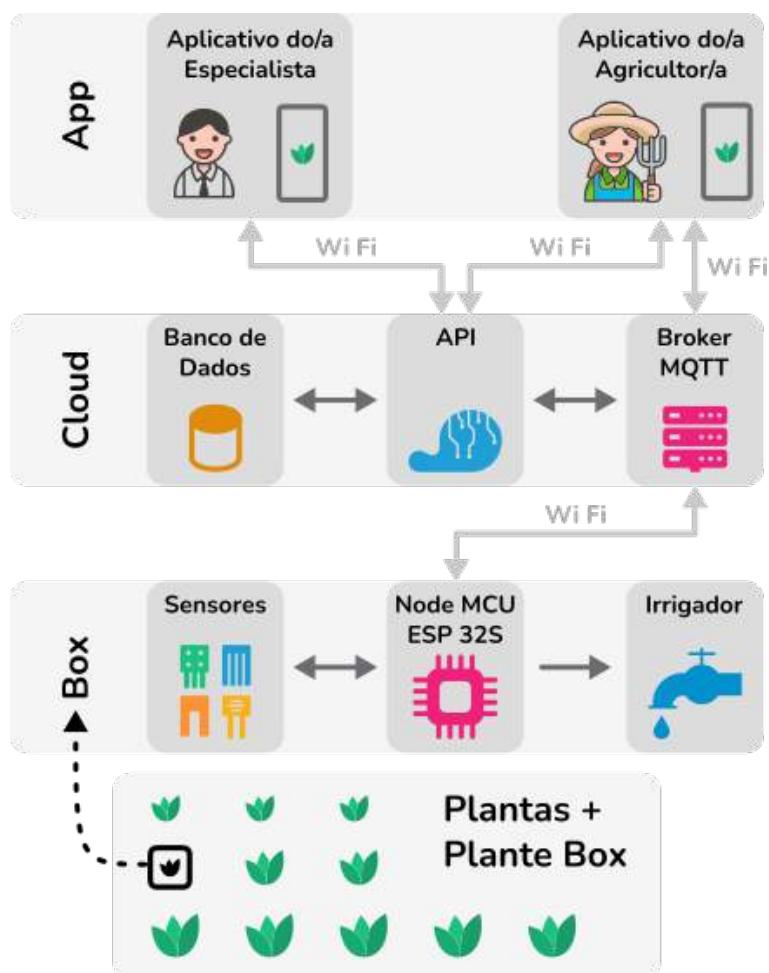
O Plante será utilizado como estudo de caso nas propostas da Arquitetura Genérica de DT, descrita no Capítulo 4 e da Linguagem de Design, a AgroLD, descrita no Capítulo 5. Por isso, é importante apresentar suas principais características e sua arquitetura original.

O Plante foi desenvolvido com intuito de ser uma plataforma inteligente para o monitoramento e irrigação agrícola (OLIVEIRA; BRAGA; OLIVEIRA, 2020; OLIVEIRA; OLIVEIRA; NUNES, 2020). A ideia é que a plataforma viabilize o cuidado

de plantas com um grau reduzido de intervenção humana, seja em grandes áreas (como espaços agricultáveis, parques, praças e zoológicos) ou pequenas áreas (como hortas comunitárias, jardins internos e terrários).

A arquitetura original do Plante (Figura 15) é dividida em três camadas principais: o *Box*, o *Cloud* e o *App*.

Figura 15 – Arquitetura Original do Plante.



Fonte: (OLIVEIRA; OLIVEIRA; NUNES, 2020)

O Plante *Box* consiste nos dispositivos de hardware da plataforma. É composto por três componentes principais: NodeMCU ESP 32s, sensores e irrigador. O NodeMCU controla o irrigador e interage com os sensores (ligando, desligando e lendo os dados coletados). Também tem função de *gateway*, enviando os dados coletados para o Plante *Cloud* e recebendo comandos de controle para manipular o irrigador. A conexão entre o NodeMCU e a camada *Cloud* se dá através do Wi-Fi. Os sensores são quatro e medem, ao todo, cinco tipos de dados: o DHT11 mede a umidade do ar e a temperatura; o YL69 mede o nível de umidade do solo; o LDR mede a intensidade de luz; e o YL83 mensura a intensidade de chuva e, consequentemente, se está ou não chovendo. Já o irrigador é composto por dois componentes: uma válvula solenoide adaptada para encanamentos e um módulo relé (para ligar e desligar a solenoide) (OLIVEIRA; OLIVEIRA; NUNES, 2020).

O Plante *Cloud* é responsável por implementar alguns mecanismos de computação em nuvem e é composto por três partes: o *Broker Message Queuing Telemetry Transport* (MQTT), o Banco de Dados e a *Application Programming Interface* (API). O *Broker* é usado para possibilitar a comunicação entre o Plante *App*, o *Cloud* e o *Box*. O Banco de Dados armazena dados sobre usuários e suas respectivas plantas ou plantações. Além disso, possui um modelo construído para armazenar dados de plantas (espécie, gênero e família botânicas) e suas características ideais de cultivo (umidade, luz, temperatura etc).

A API é composta por três serviços principais:

- Coleta de dados sensoriais: coleta os dados de umidade do ar e solo, temperatura, luz e chuva.
- Automação do irrigador: analisa os fatores ambientais da plantação (coletados pelos sensores) e a espécie de planta cultivada (levando em consideração suas características ideais de cultivo) e decide ligar ou desligar.
- Alertas: a API acessa os dados sensoriais de um ambiente e os compara às características climáticas ideais da planta(ção) monitorada. Se houver diferença entre a faixa ideal e real de características ambientais, então um alerta é gerado.

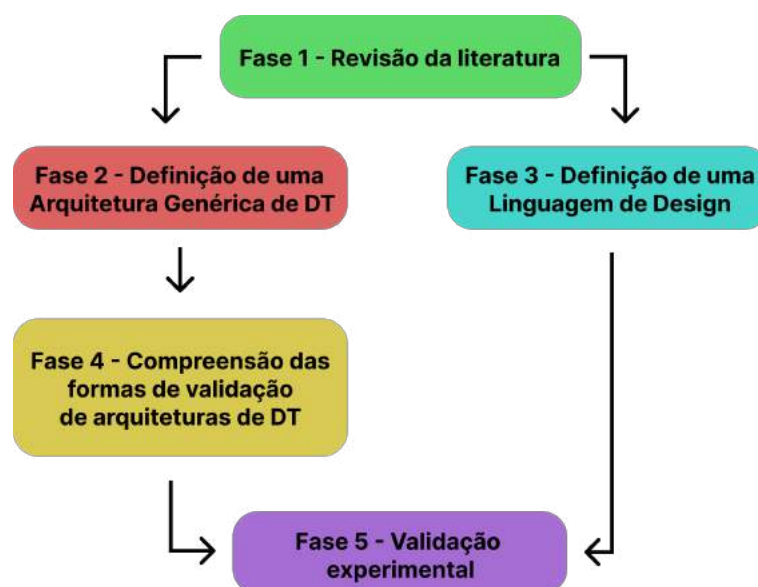
O Aplicativo do/a Especialista é usado por especialistas de domínio agrícola com objetivo de gerar uma base de dados com plantas e seus perfis ideais de cultivo. Enquanto as plantas são representadas por suas famílias, gêneros e espécies, os perfis de cultivo funcionam como um manual que possui as características ideais de clima, solo, luz e nutrientes que cada espécie de planta necessita para obter um bom desenvolvimento. Os/as especialistas gravam as informações no App, que são enviadas à API e então ao Banco de Dados.

Através do App do/a Agricultor/a os usuários podem interagir com os serviços fornecidos pela API. Os gestores tem acesso aos dados de suas plantas ou plantações, aos alertas, aos dados ambientais em tempo real (e também em forma de histórico), e também têm a opção (des)ligar o irrigador via aplicativo ou deixar tudo por conta da irrigação automática. Esse aplicativo é usado principalmente para tarefas de monitoramento e tomada de decisão.

3 Metodologia

Este capítulo apresenta a metodologia utilizada para o desenvolvimento e avaliação das contribuições desta dissertação. A Figura 16 ilustra a metodologia, composta por 5 fases principais. Neste capítulo, cada fase é apresentada de maneira geral para que, nos próximos capítulos, os resultados alcançados em cada fase sejam apresentados em detalhes.

Figura 16 – Metodologia utilizada para desenvolvimento da dissertação.



Fonte: Autor.

3.0.1 Fase 1 - Revisão da literatura

A primeira fase foi a realização de uma revisão da literatura nas temáticas de IoT (abordando os protocolos de comunicação MQTT e CoAP) e DT (abordando suas definições, aplicações, arquiteturas e linguagens de design). Os principais resultados dessa fase foram apresentados na Fundamentação Teórica (Capítulo 2). Esse estudo serviu para extração de um conjunto de desafios e problemas em aberto na temática de DT, incluindo a não existência de uma arquitetura genérica globalmente aceita e linguagem de design específica a uma determinada área ou dispositivo.

3.0.2 Fase 2 - Definição de uma Arquitetura Genérica de DT

Considerando que foi detectado na Fase 1 que não há uma definição de arquitetura genérica, foi proposta uma arquitetura genérica de DT a partir do estudo de 13 arquiteturas de DT descritas na Fase 1. Conforme descrito no Capítulo 4, esse estudo consistiu na análise detalhada das camadas e componentes das 13 arquiteturas, objetivando encontrar similaridades entre elas, para então propor uma

arquitetura que englobasse todas essas similaridades. O objetivo de tal arquitetura é apresentar, em linhas gerais, como uma arquitetura de DT funciona, o que facilitará o entendimento do tema por parte de novos pesquisadores.

3.0.3 Fase 3 - Definição de uma Linguagem de Design

Também levando em consideração as descobertas da Fase 1, em específico a escassez de propostas de linguagem de design com código aberto (e completas) para uma área específica, foi proposta uma linguagem de design para a área agrícola, denominada AgroLD. A metodologia usada, conforme descrito no Capítulo 5, consistiu no entendimento pleno do estudo de caso. Para tal foi realizada uma análise de todos os componentes e características do *Plante Box* (Seção 2.3), para então propor uma linguagem que descreva todas as suas características. O objetivo de tal linguagem é servir de modelo base, para projetistas de DT, de como se construir uma LD leve e ao mesmo tempo completa o suficiente para um determinado contexto.

3.0.4 Fase 4 - Compreensão das formas de avaliação de arquiteturas de DT

Findada a proposta da Fase 2, surgiu uma dúvida de quais são as maneiras corretas e adequadas de se validar uma arquitetura de DT. Para sanar tal dúvida foi realizada uma nova revisão da literatura, conforme será descrito no Capítulo 6, a qual resultou em 14 trabalhos que apresentavam ou propunham arquiteturas de DT. A partir dessas arquiteturas foi realizada uma análise de suas camadas e de como os autores realizavam as avaliações. Foram detectadas duas formas de avaliação: (i) a aplicação a um estudo de caso e (ii) a realização de experimentos (aplicados ao estudo de caso) em uma ou mais camadas da arquitetura.

3.0.5 Fase 5 - Avaliação experimental

Já que esta arquitetura se propõe a servir de base de pesquisa para novos pesquisadores na área, é necessário apresentar uma metodologia de avaliação para a mesma, uma vez que isto é uma boa prática encontrada na literatura. Conforme será detalhado no Capítulo 7, esta fase apresenta uma avaliação da arquitetura proposta da Fase 2 através da comparação entre a AgroLD (definida na Fase 3) e DTDL (Seção 2.2.2.5.4) e de dois protocolos IoT, o MQTT e o CoAP. Destaca-se que para a realização de tal avaliação, a arquitetura proposta é aplicada ao estudo de caso do *Plante*.

4 Definição de uma Arquitetura Genérica de DT

Este capítulo detalha a Fase 2 da metodologia descrita no Capítulo 3.

A metodologia utilizada para elaboração da proposta da arquitetura é apresentada em 4 etapas. A Etapa 1 (Seção 4.1) é a identificação de grupos de camadas e componentes arquiteturais encontrados na literatura revisada. A Etapa 2 (Seção 4.2) consiste na nomeação desses grupos, utilizando uma nomenclatura que resume a carga semântica de cada grupo. A Etapa 3 (Seção 4.3) apresenta a Arquitetura Genérica e uma descrição detalhada e abrangente de suas camadas e componentes, à luz dos trabalhos revisados. A Etapa 4 (Seção 4.4) apresenta a Arquitetura proposta aplicada ao Plante.

4.1 Etapa 1: Identificação de Grupos de Camadas e Componentes

Para propor a arquitetura genérica, foi necessário, antes de mais nada, analisar as arquiteturas revisadas na Seção 2.2.2.4 e identificar os pontos em comum. A partir dessa análise foram identificados 8 grupos de camadas/componentes, os quais possuem características similares entre si. Destaca-se que cada grupo possui camadas/componentes propostos por diferentes autores. A partir dessa análise foi gerada a Tabela 2, que mostra, à esquerda, um nome genérico para identificar o grupo; e à direita, as camadas/componentes pertencentes ao grupo. É importante frisar que havia grupos com características mais genéricas (denominados de Camadas), e grupos com características mais específicas (denominados de Componentes).

O Grupo 1 corresponde a uma camada, que engloba todo o processo de virtualização, servitização e gestão de dados de DT. Dentro dele foram identificados 3 subgrupos. O Subgrupo 1.1 relaciona-se diretamente ao processo de servitização, englobando os mais diversos tipos de serviços que o DT pode oferecer, desde básicos, como monitoramento de sensores até mais complexos, como otimização e atualização inteligente baseada em técnicas de IA (CIMINO; NEGRI; FUMAGALLI, 2019). O Subgrupo 1.2 equivale a etapa de modelagem, design e virtualização de entidades físicas, diretamente relacionada ao conceito de Linguagem de Design. O Subgrupo 1.3 representa o sítio físico e lógico onde ficam armazenados todos os dados relativos ao comportamento da entidade física e ao contexto no qual ela está inserida.

O Grupo 2 corresponde à camada de infraestrutura de comunicação, compreendendo tecnologias de rede, protocolos de troca de mensagens, arquiteturas de comunicação, servidores e afins.

O Grupo 3 é uma camada correlata à entidade física como um todo, ele possui 2 subgrupos. O Subgrupo 3.1 está relacionado ao espaço de programação / parcela de código programável capaz de interagir (controlar, monitorar, trocar dados, se

Tabela 2 – Grupos de camadas/componentes arquiteturais de DT.

Nome do Grupo	Camadas/Componentes Equivalentes (+ referência)
Grupo 1	Mundo Virtual (TAO et al., 2019), Gêmeo Digital Orientado a Serviços
Subgrupo 1.1	Espaço de Aplicativo (MINERVA; LEE; CRESPI, 2020), Computação de Nuvem e Framework Espelhado (WU et al., 2020), Frequência de Atualização, Inteligência dos Sistema Ciber-Físicos e Capacidades de Simulação (LIU et al., 2021), Metrologia (Virtual para Físico) e Realização (JONES et al., 2020), Serviços (QI et al., 2021), Módulo de Processamento de Dados (LU et al., 2020), <i>Middleware</i> (FULLER et al., 2020)
Subgrupo 1.2	Espaço de Virtualização (MINERVA; LEE; CRESPI, 2020), Objeto Lógico (WU et al., 2020), Entidade Virtual (TAO; ZHANG, 2017) e (JONES et al., 2020), Virtual (BOJE et al., 2020), Modelos Virtuais (QI et al., 2021), Modelo de Informação (LU et al., 2020), Aplicação (FULLER et al., 2020)
Subgrupo 1.3	Base de Dados de DT (TAO; ZHANG, 2017), Dados (BOJE et al., 2020), Dados (QI et al., 2021)
Grupo 2	Conexão (TAO et al., 2019) e (TAO; ZHANG, 2017), Modo de Conectividade (LIU et al., 2021), Conexões (QI et al., 2021), Mecanismo de Comunicação (LU et al., 2020), Rede (FULLER et al., 2020)
Grupo 3	Espaço Físico (MINERVA; LEE; CRESPI, 2020), Entidade Física (TAO; ZHANG, 2017) e (JONES et al., 2020), Físico (BOJE et al., 2020), Entidades Físicas (QI et al., 2021), Sistema Ciber-Físico (LONGO; NICOLETTI; PADOVANO, 2019), Objeto (FULLER et al., 2020)
Subgrupo 3.1	Computação Local (WU et al., 2020), Metrologia (Físico para Virtual) (JONES et al., 2020)
Subgrupo 3.2	Objeto Físico (WU et al., 2020), Objeto Físico e Sensoriamento (TAO et al., 2019), Camada de Sensores e Camada de Atuadores (LONGO; NICOLETTI; PADOVANO, 2019)

Fonte: Autor.

comunicar) com os componentes de hardware do objeto físico. Enquanto o Subgrupo 3.2 compreende todos os componentes de hardware atrelados ao objeto, como sensores, atuadores, placa Wi-Fi etc; além de informações pertinentes ao hardware, como número de série, fabricação etc.

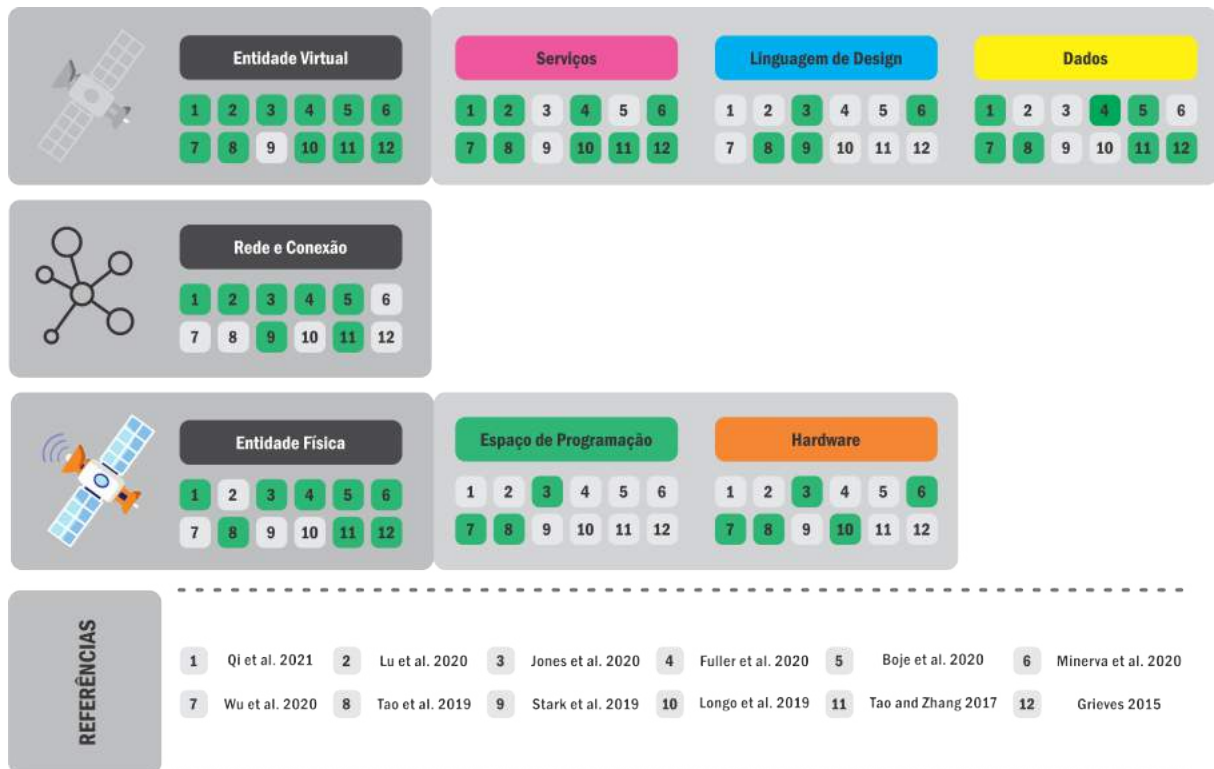
4.2 Etapa 2: Nomeação dos Grupos Identificados

Após a identificação dos grupos e a breve explicação sobre os papéis de cada um, ficou decidido por nomeá-los da seguinte forma:

- Grupo 1 = Entidade Virtual
 - Subgrupo 1.1 = Serviços
 - Subgrupo 1.2 = Linguagem de Design
 - Subgrupo 1.3 = Dados
- Grupo 2 = Rede e Conexão
- Grupo 3 = Entidade Física
 - Subgrupo 3.1 = Espaço de Programação
 - Subgrupo 3.2 = Hardware

O resultado final, com os nomes dos 8 componentes e suas equivalências na literatura revisada na Seção 2.2.2.4, está descrito na Figura 17.

Figura 17 – Componentes da Arquitetura DT e suas equivalências na literatura.



Fonte: Autor.

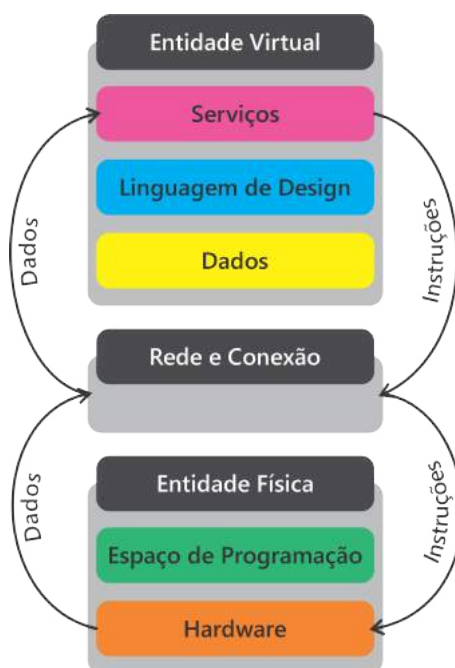
É possível notar números destacados em verde ou branco, que fazem referência aos IDs dos artigos revisados. Na camada Entidade Virtual, por exemplo, o

número 1 está verde, indicando que a arquitetura do trabalho com ID 1 de (QI et al., 2021) possui uma camada equivalente. Já no componente Linguagem de Design o número 1 está branco, indicando que o trabalho com ID 1 não possui um componente equivalente. Tomando como exemplo o trabalho com ID 7, nota-se que ele não possui equivalência com a Entidade Física, já que não apresentou uma camada que represente uma EF de forma completa. Porém, no trabalho foi identificada uma equivalência com seus dois componentes: Espaço de Programação e Hardware.

4.3 Etapa 3: Proposta e Detalhamento da Arquitetura Genérica

Baseado nos dados da Tabela 2 e da Figura 17 foi proposta uma arquitetura genérica de DT, como demonstra a Figura 18. A seguir é dada uma descrição detalhada e abrangente de cada camada e componente da arquitetura proposta. Para tal descrição, leva-se em consideração a visão e as análises dos autores dos trabalhos citados na Figura 17, a qual foi usada compor a proposta de arquitetura.

Figura 18 – Proposta de Arquitetura Genérica de DT.



Fonte: Autor.

4.3.1 Fluxo de Dados

Os dados geralmente têm origem no Hardware da Entidade Física (os quais podem ser coletados do ambiente através de sensores). Após a coleta, eles são enviados (através de uma interface de rede ou antena do Hardware) à Rede e Conexão, a partir da qual são encaminhados ao componente Serviços (da EV). Imagine que tais dados possuam informações a respeito da temperatura atual da

CPU, que está a 120 graus Celsius. Chegando no componente Serviços, esses dados serão analisados, será detectado que a temperatura está muito elevada, e será criada uma instrução (por exemplo: reduzir o número de operações da CPU pela metade, durante 2 minutos), que será encaminhada para a camada Rede e Conexão, e então chegará à antena (ou interface de rede) localizada no Hardware da EF. Após isso, essa instrução será lida e executada a partir do Espaço de Programação.

4.3.2 Entidade Física

A **Entidade Física** foi representada nos artigos de formas diferentes. Alguns autores apresentaram a visão de um espaço físico - como uma sala, galpão ou compartimento de modo geral - onde estariam armazenados objetos, máquinas, veículos, equipamentos, materiais e componentes eletrônicos, especialmente aqueles relacionados a ambientes fabris, manufatureiros e linhas de produção (MINERVA; LEE; CRESPI, 2020) (TAO; ZHANG, 2017).

A **Entidade Física** também pode ser interpretada como um ser humano, seja um paciente hospitalizado, alguém praticando atividades físicas, ou um funcionário trabalhando em uma fábrica (TAO; ZHANG, 2017). Além disso, há a visão de **Entidade Física** como um ser vivo qualquer (humano, animal ou planta) (JONES et al., 2020). O ponto em comum nessas visões é de que o ser vivo em questão necessita, por algum motivo, de monitoramento constante. Pode-se citar o exemplo de um paciente doente que necessita de acompanhamento 24h para averiguar seu estado de saúde, coletando dados como pressão arterial, temperatura corporal, nível de hidratação, dentre outros.

Alguns autores também defendem que a **Entidade Física** pode ser um sistema computacional ou quaisquer recursos de software passíveis de monitoramento (QI et al., 2021) (LONGO; NICOLETTI; PADOVANO, 2019). Para compreender esse conceito, pode-se imaginar o software de um braço robótico utilizado na linha de produção de automóveis. Esse software precisa ser monitorado constantemente para averiguar, por exemplo, o tempo de trabalho do braço, o nível de precisão com que ele está atuando, o desgaste do equipamento, se há necessidade de alguma manutenção etc.

Outra visão de **Entidade Física** diz respeito a uma entidade de hardware atrelada a um software, na qual o hardware teria habilidades como sentido, capaz de observar o mundo físico em tempo real por meio de sensores; e atuação, capaz de alterar/ativar/desativar componentes físicos com base em decisões/estímulos virtuais. O software teria o papel de monitoramento, capaz de rastrear, informar e emitir avisos sobre alterações físicas relevantes (BOJE et al., 2020) (QI et al., 2021) (FULLER et al., 2020). De modo geral, pode-se afirmar que a **Entidade Física** possui uma parte mecânica e uma lógica. À parte mecânica foi dado o nome de **Hardware**, e à lógica, **Espaço de Programação**, ambas são descritas a seguir.

4.3.2.1 Espaço de Programação

O componente **Espaço de Programação** está relacionado à parcela de software programável que interage com o **Hardware** da **Entidade Física** através

de atividades de controle, monitoramento, troca de dados e gestão de informações de modo geral.

Partindo do pressuposto que um dos papéis do **Espaço de Programação** é a gestão do **Hardware**, imagine um cenário com um automóvel inteligente movido à bateria. Nesse panorama, o **Espaço de Programação** é responsável por medir dados históricos da bateria, como o estado de carregamento, tempo para descarga, vida útil restante etc. Além disso, ele pode implementar métricas para definir o limite máximo de temperatura da bateria, visando a maximização da saúde e vida útil (WU et al., 2020).

O **Espaço de Programação** também tem um papel de intermediação da comunicação, uma vez que tem que gerenciar os dados que são enviados do **Hardware** para a camada **Rede e Conexão**, e aqueles que transitam no sentido contrário. Um exemplo desses dados são as informações contextuais, tanto do estado do **Hardware** em si, quanto os dados ambientais coletados através de sensores (JONES et al., 2020).

4.3.2.2 Hardware

Esse componente compreende todos os componentes físicos atrelados à **EF**, e diz respeito, exclusivamente, à parte mecânica/física. Fazendo um paralelo com um *smartphone*, o **Hardware** engloba componentes como memória, tela, placa de Wi-Fi, enquanto o **Espaço de Programação** é composto pelo Sistema Operacional e os Firmwares.

Um exemplo proveniente da literatura revisada é um veículo automotivo movido à bateria (WU et al., 2020), o qual possui componentes, como rodas, capô, engrenagens, direção, freio, tanque de gasolina, motor, bateria etc. Ele também possui sensores capazes de medir voltagem, corrente e temperatura da bateria. Entretanto, cada sensor só pode medir e armazenar esses dados através da intervenção de um software (WU et al., 2020).

Em geral, os sensores que fazem parte do **Hardware** são responsáveis por coletar atributos básicos, estado em tempo real, parâmetros de processo, andamento do processamento, registros de manutenção, dados de operação de produtos, bem como dados de demandas, de comportamento e de transações de usuários. As possibilidades são amplas e mudam de acordo com o contexto. Outro componente importante são os atuadores, acionados/controlados automaticamente ou manualmente via internet/software para realizar algum tipo de ação no mundo físico (LONGO; NICOLETTI; PADOVANO, 2019) (TAO et al., 2019).

4.3.3 Rede e Conexão

Por via de regra, esta camada corresponde à infraestrutura de comunicação como um todo, compreendendo tecnologias de rede, protocolos de troca de mensagens, arquiteturas de comunicação, servidores e afins. O principal objetivo é a interconexão entre **EF** e **EV** (GRIEVES, 2015), conforme ilustrado na Figura 1.

Alguns autores citaram exemplos dos modos de conectividade possíveis. Para (STARK; FRESEMAN; LINDOW, 2019) a conexão pode ser (i) automática, sem necessidade de intervenção humana; (ii) direcionada pelo contexto; (iii) bi-direcional, quando a **Entidade Física** e a **Entidade Virtual** atuam simultaneamente como publicadoras e consumidoras de dados; e (iv) uni-direcional, quando apenas uma das entidades publica dados.

O entendimento geral é de que essa camada é utilizada de modo bi-direcional e em tempo real, e possui tecnologia sem fio, seja Wi-Fi, Bluetooth etc (TAO et al., 2019) (TAO; ZHANG, 2017) (LU et al., 2020). Além disso, a escolha das tecnologias de comunicação utilizadas nessa camada devem variar conforme o contexto. Essa variação busca garantir que os dados coletados sejam comunicados sem erros e em tempo hábil entre as entidades pertencentes à arquitetura, permitindo atividades como simulação, operação e análise avançadas (QI et al., 2021) (FULLER et al., 2020).

4.3.4 Entidade Virtual

A camada **Entidade Virtual** engloba os componentes que descrevem de maneira genérica atividades de geminação (criação de um clone virtual a partir de uma **Entidade Física**) e gestão de dados e serviços extraídos desse processo de geminação.

Pode-se detectar três aspectos principais da **Entidade Virtual**. O primeiro é referente aos **Serviços** que podem ser obtidos através da aplicação de algoritmos (LONGO; NICOLETTI; PADOVANO, 2019) e mecanismos de inteligência nos dados. Os serviços podem ir desde a avaliação do estado de saúde de um equipamento, até projeções à longo prazo do consumo de energia elétrica de uma cidade, por exemplo. O segundo refere-se a uma **Linguagem de Design** (LD), que reside em sistemas de computação em rede (computadores e servidores) e sistemas de armazenamento (discos rígidos) (GRIEVES, 2015). A LD descreve um clone digital perfeito (WU et al., 2020) de uma **Entidade Física** (que pode ser um produto, equipamento, cidade, ou até um ser vivo). O terceiro aspecto engloba os **Dados** extraídos a partir da interação entre LD e **Entidade Física**.

4.3.4.1 Serviços

Esse componente engloba os mais diversos tipos de serviços que o DT pode oferecer, desde básicos, como monitoramento de sensores, até mais complexos, como otimização e atualização inteligente baseada em técnicas de Inteligência Artificial (CIMINO; NEGRI; FUMAGALLI, 2019).

Um exemplo de serviço é utilizar o processamento de dados para analisar informações e aplicar processos de tomada de decisão, enviando mensagens de controle à **Entidade Física** (TAO et al., 2019). Imagine um cenário de uma horta inteligente onde a **Entidade Física** possui sensores de umidade do solo e temperatura. Essa **Entidade Física** monitora diariamente a horta e envia informações contextuais à camada **Entidade Virtual**, a qual decide o momento certo de realizar uma irrigação.

Os serviços geralmente são disponibilizados de duas formas: i) através de aplicativos que interagem com usuários finais, no caso de usuários humanos; e ii) serviços que são consumidos diretamente por APIs, sem interação humana direta. Alguns serviços consumidos por humanos são o gerenciamento de simulação (MINERVA; LEE; CRESPI, 2020) (onde o usuário visualiza em tempo real um clone virtual da **EF**); e a visualização do histórico de métricas de saúde da **Entidade Física** (WU et al., 2020), ou daquilo que a **Entidade Física** está gerenciando (no exemplo da horta, a **Entidade Física** seria os sensores, enquanto aquilo que está sendo gerenciado são as plantas).

Outras aplicações envolvem a utilização de Inteligência Artificial, aprendizado de máquina e controle híbrido inteligente, gestão de consistência da **EV**, agregação e gerenciamento de dados, objetos e recursos (MINERVA; LEE; CRESPI, 2020) (LU et al., 2020). Ainda outras, seriam a predição de tempo restante de bateria e otimização orientada a dados, atualização e manutenção remota, sistema de pré-aviso, gerenciamento de energia, carregamento inteligente, estimativa de estado, diagnóstico de bateria, balanceamento celular, aplicação de modelo/algoritmo, processamento e coleta de dados (WU et al., 2020) (QI et al., 2021).

4.3.4.2 Linguagem de Design

Majoritariamente, esse componente visa descrever todos os aspectos possíveis de uma **EF** através de diferentes modelos. Alguns desses modelos são: i) geométrico, referente ao comprimento, área e volume; ii) cinemático, referente à mecânica e aos movimentos; iii) estrutural, descrevendo os tipos de materiais, a resistência, as bases de sustentação; iv) de regras, descrevendo os comportamentos definidos por regras via software; v) de processo ou comportamental, que descreve os processos de produção; vi) evolucionário, que contém o histórico dos dados coletados da **Entidade Física**, e todas as alterações realizadas em seus modelos (TAO et al., 2019) (STARK; FRESEMAN; LINDOW, 2019) (LU et al., 2020).

A LD também é utilizada para o permitir o gerenciamento da **Entidade Virtual**, isso porque tanto a criação da estrutura da **Entidade Virtual**, quanto a atualização dessa estrutura ao longo do tempo dependem dos modelos aplicados através da **Linguagem de Design** (MINERVA; LEE; CRESPI, 2020) (JONES et al., 2020). Importante destacar que, em regra, a **Linguagem de Design** deve ser atualizada conforme ocorrem alterações de alguma natureza na **Entidade Física**.

A LD também é usada para: i) aplicar modelos de simulação de engenharia de vários domínios de aplicação; ii) prever o comportamento físico com base em simulações e sensores digitais; iii) aplicar métodos de otimização e recomendar alocação inteligente de recursos; iv) delegar agentes de Inteligência Artificial capazes de gerenciar a **Entidade Física** com base em dados digitais, seguindo comportamentos, protocolos e objetivos bem definidos; v) gerar mensagens de controle que são enviadas à **Entidade Física**, vi) bem como permitir a visualização e modelagem de arquitetura da **Entidade Virtual** (JONES et al., 2020) (BOJE et al., 2020) (FULLER et al., 2020).

4.3.4.3 Dados

O componente **Dados** representa o sítio físico e lógico onde ficam armazenadas todas as informações relativas ao comportamento da **Entidade Física** e ao contexto no qual ela está inserida. De modo geral, os **Dados** possuem escala multi-temporal, multi-dimensional, multi-fonte e são heterogêneos, incluindo dados de atributos estáticos (como os modelos presentes da **Linguagem de Design**) e dados de condições dinâmicas (como os dados do estado atual da **Entidade Física**) (QI et al., 2021).

Os **Dados** também podem ser resultado da fusão de informações da **Entidade Física** com estatísticas de rede e os demais componentes da arquitetura, fornecendo informações abrangentes e consistentes, além de alimentar métodos de modelagem, otimização e previsão (TAO; ZHANG, 2017). Também devem possuir capacidades como compartilhamento entre dispositivos de IoT usando uma sintaxe unificada, além de serem estruturados o suficiente a ponto de armazenar conhecimento, como fatos sobre o sistema, regras de suporte e capacidades de raciocínio (BOJE et al., 2020) (FULLER et al., 2020).

4.4 Etapa 4: Aplicação da Arquitetura Genérica ao Estudo de Caso do Plante

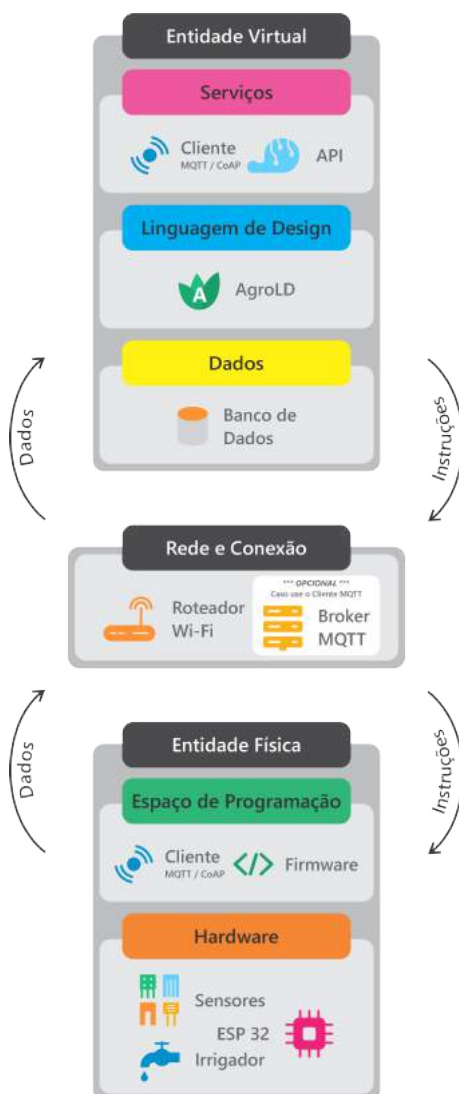
Esta etapa consiste na integração da Arquitetura Genérica proposta com o Plante, no contexto do estudo de caso descrito na Seção 2.3. O objetivo desta integração com o estudo de caso é permitir a criação de um *testbed* para a realização de uma avaliação experimental descrita no Capítulo 7. Dado o exposto, foi elaborada a Figura 19 que contém as especificidades do Plante aplicadas à arquitetura genérica da Figura 18.

Na camada **Entidade Física**, o componente *Hardware* é composto por três módulos. O primeiro é um NodeMCU ESP 32s, uma plataforma IoT *open source* que utiliza como microcontrolador o ESP 32, de baixo custo e baixo consumo de energia, que possui os chips Wi-Fi e Bluetooth incorporados nativamente. O segundo módulo é formado por Sensores que medem cinco tipos de dados: o DHT11, que mede a umidade do ar e a temperatura; o YL69, a umidade do solo; o LDR, a intensidade de luz; e o YL83, a intensidade de chuva. Por fim, para possibilitar a adequação do hardware com a proposta de cultivar plantas com pouca intervenção humana, tem-se um atuador para irrigar a plantação.

Com relação ao *Espaço de Programação*, o módulo *firmware* é gravado no ESP 32, sendo responsável por controlar o irrigador e interagir com os sensores (ligando-os e lendo os dados coletados). Também há um módulo para o Cliente de comunicação (que pode ser utilizado com o MQTT ou CoAP), enviando os dados ambientais coletados para a EV e recebendo comandos de controle para manipular o atuador de irrigação.

A conexão entre a **Entidade Física** e a **Rede e Conexão** se dá através do Wi-Fi. Caso seja usando o Cliente MQTT na EV também deve-se usar o Broker MQTT, que

Figura 19 – Arquitetura Genérica aplicada ao estudo de caso do Plante.



Fonte: Autor.

serve para interconectar os dois Clientes de comunicação, um que está na EF e outro na EV. Destaca-se que a camada de Rede e Conexão é utilizada em duas situações principais: quando novos dados são coletados da plantação (os quais são tratados, formatados e enviados da EF para a EV); e quando a EV, através do componente de *Serviços* automaticamente envia uma requisição de execução/interrupção da irrigação da plantação.

Com relação aos *Serviços* da camada **Entidade Virtual**, a API tem como uma de suas funcionalidades a análise de dados visando a automação do irrigador e a colaboração com a tomada de decisão dos usuários do Plante. O algoritmo que manipula o irrigador é sensível ao contexto, analisando questões como os fatores ambientais da plantação (coletados pelos sensores) e a espécie de planta cultivada (levando em consideração suas características ideais de cultivo). O que o algoritmo faz é decidir a hora certa de ligar e desligar o irrigador e enviar o comando *on/off*: i) em caso do uso do MQTT, ao Broker MQTT, que o repassará ao Cliente MQTT da

EF, ou ii) em caso de uso do CoAP, diretamente ao Cliente CoAP. A partir disso, o comando recebido será executado por meio do *firmware*.

A *API* também conta com um sistema de alertas para ajudar os usuários na tomada de decisões. O fluxo de execução ocorre da seguinte maneira: (i) os dados coletados pela EF são enviados à EV; (ii) a API acessa os dados (temperatura, umidade do ar e solo, luminosidade e chuva) e os compara às características climáticas ideais da plantação monitorada; (iii) as características que estiverem desconformes com o ideal são gravadas no banco de dados (dentro do componente *Dados*); (iv) o Plante App (Aplicativo do Agricultor) captura esses dados e os exibe ao usuário, deixando-o ciente de que, naquele momento, o clima não está ideal para a planta cultivada. No componente *Linguagem de Design* encontra-se a AgroLD (descrita no Capítulo 5).

5 Definição de uma Linguagem de Design

Este capítulo detalha a Fase 3 da metodologia descrita no Capítulo 3.

No contexto de *Digital Twin* (DT), toda Entidade Virtual (EV) é um clone digital exato de uma Entidade Física (EF). Já a Linguagem de Design (LD), como explanado na Seção 2.2.2.5, é o meio através do qual se constrói a EV. Levando-se em conta que a EV possui uma natureza específica - ou seja, ela é um clone de uma EF específica (o *Plante Box*, por exemplo) - pode-se afirmar que a LD (responsável por construir uma EV) também possui uma natureza específica.

Conforme discutido na Seção 2.2.2.5, para propor LD de DT é recomendado definir o contexto no qual essa LD será aplicada, para que o processo de geminação (criação do clone digital (EV) idêntico a uma EF) ocorra com “alta fidelidade” e “alta precisão” (BOSCHERT; ROSEN, 2018). Uma LD deve ser leve, enxuta e específica, o que é difícil de ser alcançado sem a definição de um contexto. Com um contexto definido é possível selecionar apenas as características e atributos necessários sem carregar detalhes supérfluos. Isso reduz drasticamente o tamanho dos modelos e permite um processamento mais rápido e em tempo hábil (GRIEVES, 2015) (WRIGHT; DAVIDSON, 2020).

Baseado na revisão bibliográfica do Capítulo 2, duas áreas apresentaram maior enfoque de estudos sobre o tema *Digital Twin*: indústria e manufatura. Além delas, áreas como cidades inteligentes, automobilística, aeroespacial e cuidados em saúde também são citadas, entretanto, com menor ênfase. Porém, dentre os artigos revisados, não havia citações de aplicações detalhadas para a área agrícola, demonstrando uma carência de trabalhos que pesquisem integração de agricultura e *Digital Twin*.

Dado o exposto, e com intuito de contribuir com a pesquisa de áreas pouco exploradas no cenário de *Digital Twin*, o contexto escolhido para a proposta da Linguagem de Design foi o agrícola.

Assim, a proposta deste trabalho consiste em uma Linguagem de Design para DT para cenários agrícolas. A LD é denominada AgroLD. E como uma das boas práticas para a construção de uma LD é o desenvolvimento colaborativo, o código fonte da AgroLD está disponível em um repositório público no GitHub: <https://github.com/RenatoAlves0/agrold>.

A metodologia utilizada para elaboração da AgroLD é apresentada em 4 etapas. A Etapa 1 (Seção 5.1) é o entendimento do domínio de estudo. A Etapa 2 (Seção 5.2) define o cenário do estudo de caso. A Etapa 3 (Seção 5.3) apresenta a AgroLD, que é a linguagem de design proposta para a área agrícola, com foco em dispositivos de IoT de pequeno porte, como o *Plante Box*. A Etapa 4 (Seção 5.4) apresenta uma comparação entre a AgroLD e a DTDL, ambas integradas ao *Plante Box*.

5.1 Etapa 1: Entendimento do Domínio de Estudo

A primeira etapa da pesquisa consistiu no entendimento do domínio de estudo. Para tal foram analisados os estudos teóricos realizados na Fundamentação Teórica (Capítulo 2), especificamente na Seção 2.2.2.5.

5.2 Etapa 2: Definição do Cenário do Estudo de Caso

Neste trabalho, foi escolhida uma aplicação já existente e sem integração com o DT para a proposta da Linguagem de Design. Trata-se do Plante, uma plataforma inteligente para o monitoramento e irrigação agrícola (OLIVEIRA; OLIVEIRA; NUNES, 2020; OLIVEIRA; BRAGA; OLIVEIRA, 2020), detalhada na Seção 2.3.

Como o objetivo de uma Linguagem de Design é criar um gêmeo digital de uma entidade física, então é necessário compreender mais a fundo os componentes de hardware do Plante Box, a entidade física do Plante neste estudo de caso. Assim, a Tabela 3 apresenta informações técnicas referentes ao microcontrolador Node MCU ESP. A Tabela 4 apresenta detalhes sobre os sensores e o atuador do Box. Em se tratando da segunda tabela, a coluna Tipo de Dado diz respeito aos dados coletados pelos sensores e ao dado gerado pelo irrigador. Já a coluna Funções Relacionadas apresenta as ações que se pode usar para interagir com os sensores e o atuador (ex.: ler os dados do sensor).

Tabela 3 – Informações técnicas do microcontrolador Node MCU ESP 32S.

Nome	Característica(s)
Wi-Fi	IEEE 802.11 b/g/n; 2.4 GHz; HT20/40; até 150 Mbps
Bluetooth	BR/EDR + Bluetooth LE v4.2
Temperatura de Operação	De -40 até 105 °C
GPIO	34 unidades
Memória Flash	4 MB
Memória SRAM	520 KB
Memória ROM	448 KB
Frequência do Microcontrolador	240 MHz
Tamanho do Microcontrolador	5 x 5 mm

Fonte: Autor.

5.3 Etapa 3: Definição da Linguagem de Design AgroLD

Nesta seção, é apresentada a AgroLD, uma linguagem de design com enfoque na área agrícola capaz de descrever os recursos do Plante Box e de dispositivos agrícolas similares a ele. A AgroLD está escrita em inglês e representada pelo padrão *JavaScript Object Notation* (JSON). A decisão de utilizar o JSON como padrão foi

Tabela 4 – Detalhes dos sensores e atuador do Plante Box

Nome	Tipo de Dado	Funções Relacionadas
(Sensor) DHT11	temperatura, umidade ambiente	ligar, desligar, ler
(Sensor) YL69	umidade do solo	ligar, desligar, ler
(Sensor) YL83	chuva	ligar, desligar, ler
(Sensor) LDR	luz	ligar, desligar, ler
(Atuador) Irrigador	irrigação	ligar, desligar

Fonte: Autor.

devido ao fato de ele ser amplamente utilizado no mundo da tecnologia e possuir uma formatação leve de dados (BASSETT, 2015).

Antes de definir a AgroLD, realizou-se uma etapa de identificação de dados úteis para um projeto como o Plante. Para isso, foram levadas em consideração todas as informações apresentadas na Seção 2.2.2.5; os oito componentes de “Linguagem de Design” presentes na Tabela 2, na qual foram analisados os detalhes arquitetônicos elencados pelos autores; e os detalhes de hardware do Plante Box, apresentados na Seção 5.2. A partir dessa análise, definiu-se que a AgroLD descreveria 6 tipos de informação do Plante Box, descritos a seguir:

- **Informações Básicas:** podem detalhar informações técnicas e metadados a respeito da Entidade Física, tais como modelo, nome, número de série, fabricante e data de fabricação, versão do *firmware* padrão, dentre outras. (Descreve características relacionadas à **identificação básica**, detalhada na Seção 2.2.2.5.2).
- **Atuadores:** podem conter dados sobre o nome e tipo do atuador, bem como funções padrões a ele atreladas, como ligar, desligar, girar 90° à direita etc.
- **Sensores:** podem englobar detalhes sobre o nome e tipo do sensor, bem como funções padrões a ele atreladas, como ligar, desligar, ler temperatura ambiente etc. (Descreve características relacionadas à **ambiente e sensores** detalhadas na Seção 2.2.2.5.2).
- **Recursos de Hardware:** compreende a descrição de outros recursos de hardware e suas características pertinentes à aplicação como, por exemplo, bateria (capacidade máxima, valor atual), memória (tipo, capacidade), processador (tipo, *clock*, núcleos) etc.
- **Conectividade:** inclui informações sobre recursos de hardware relativos à conectividade e rede, como Wi-Fi (protocolo, tipo de antena), Bluetooth (versão, alcance), dentre outros.
- **Recursos Lógicos:** responsáveis por descrever as funções e métodos relativos aos componentes anteriores, ou até mesmo relativos à comunicação com componentes externos. (Descreve características relacionadas à **saúde, funcionalidade, estado e ambiente**, detalhadas na Seção 2.2.2.5.2).

Fazendo um paralelo com a Seção 2.2.2.5.2 onde são apresentados os diferentes grupos de características que uma LD pode usar para descrever uma EF, destaca-se que a AgroLD se encaixa em 6, dos 14 grupos de características identificados, são eles: **identificação básica, saúde, funcionalidade, estado, ambiente e sensores**.

Conforme defende (WRIGHT; DAVIDSON, 2020), a linguagem de design tem que ser rápida para que as decisões sobre o aplicativo possam ser feitas dentro do prazo exigido. Para tal, é importante que uma linguagem de design só se preocupe em descrever as características que sejam necessárias ao contexto no qual ela será utilizada, não sendo uma boa prática descrever características que não serão utilizadas na prática.

Após a definição genérica dos 6 tipos de informações que a AgroLD é responsável por gerenciar, foram definidos, de maneira específica, os componentes (e sua estrutura) responsáveis por descrever os 6 tipos de informação.

Conforme apresentado na Listing 5.1, a AgroLD possui um id para identificar a entidade virtualizada e 7 componentes: os sensores (*sensors*), os atuadores (*actuators*), a Unidade Central de Processamento (*Central Process Unit* - CPU), a memória (*memory*), a bateria (*battery*), a comunicação (*communication*) e as funções (*functions*). A seguir, cada componente é explicado em detalhes.

Os *sensors* são uma lista, na qual cada sensor deve conter um id e um data (uma lista representando os tipos de dados coletados, como temperatura e umidade).

Os *actuators* também estão em formato de lista e devem conter id e actions (uma lista representando os tipos de ações/propósitos do atuador, como irrigar).

Há também o componente CPU, com os campos *model*, quantidade de cores, *lithography*, que é o tamanho do processador com *value* (exemplo: mm, cm etc) e *unity* (exemplo: 5, 10), *clock* com *value* (exemplo: 1.000, 700) e *unity* (exemplo: MHz, GHz) e *temperature* de operação com valores *min*, *max* e *unity* (exemplo: graus celsius, fahrenheit).

O componente *memory* é uma lista, pois podem haver diferentes memórias na mesma entidade. Ela possui *id*, *model*, *type* (exemplo: Flash, ROM), *value* e *unity* da *capacity* (exemplo: 1024, MB), *value* e *unity* da *speed* (exemplo: 266, Mbps).

A *battery* contém *model*, *type* (exemplo: lítio), *value* e *unity* da *capacity* (exemplo: 6000, mAh) e a *temperature* de operação (com valores *min*, *max* e *unity* exemplo: graus celsius, fahrenheit).

O componente *communication* também é uma lista, pois pode haver mais de um meio de comunicação, como Wi-Fi e Bluetooth. Ele contém *id*, *protocol* (exemplo: Wi-Fi, Bluetooth), *range* com *value* e *unity* (exemplo: 100, km), *speed* com *value* e *unity* (exemplo: 100, Gbps).

A lista de *functions* serve para representar as formas de controlar ou de se comunicar com os componentes de hardware, como ler, desligar, ligar etc. Cada função contém um *id* (o nome/identificador da função), um *alias* do *id* (um apelido do *id* para ser usado na comunicação entre Entidade Física e o Gêmeo Digital ao

invés do próprio id, tendo como objetivo economizar largura de banda no processo de comunicação) e uma *description* (informando com mais detalhes o propósito da função). No mais, os *parameters* informam se a função requer ou suporta alguma passagem de valor para a função; ele contém *id* (exemplo: altura, cor), *type* (exemplo: *float*, *string*), *description*, *enum* (caso o parâmetro seja limitado a um tipo específico de valores, então deve ser usado o *enum*, informando quais os valores possíveis), *aliasEnum* (um apelido para os valores do *enum*, visando economia de dados no processo de comunicação) e *required* (informando se o parâmetro é obrigatório ou não). Dentro da função existe também a *response*, informando o formato de resposta da função; contém um *id*, um *alias* do *id*, um *type* (exemplo: *string*) e um *unity* (exemplo: graus celsius, percentual, centímetros).

Listing 5.1 – Estrutura da AgroLD

```

1 {
2   "entity": {
3     "id": "",
4     "sensors": [{ "id": "", "data": [] }],
5     "actuators": [{ "id": "", "actions": [] }],
6     "CPU": {
7       "model": "",
8       "cores": "",
9       "lithography": { "value": "", "unity": "" },
10      "clock": { "value": "", "unity": "" },
11      "temperature": { "min": "", "max": "", "unity": "" }
12    },
13    "memory": [
14      {
15        "id": "",
16        "model": "",
17        "type": "",
18        "capacity": { "value": "", "unity": "" },
19        "speed": { "value": "", "unity": "" }
20      }
21    ],
22    "battery": {
23      "model": "",
24      "type": "",
25      "capacity": { "value": "", "unity": "" },
26      "temperature": { "min": "", "max": "", "unity": "" }
27    },
28    "communication": [
29      {
30        "id": "",
31        "protocol": "",
32        "range": { "value": "", "unity": "" },
33        "speed": { "value": "", "unity": "" }
34      }

```

```

35 ],
36 "functions": [
37   {
38     "id": "",
39     "alias": "",
40     "description": "",
41     "parameters": [
42       {
43         "id": "",
44         "type": "",
45         "description": "",
46         "enum": [],
47         "aliasEnum": [],
48         "required": ""
49       }
50     ],
51     "response": [
52       {
53         "id": "",
54         "alias": "",
55         "type": "",
56         "unity": ""
57       }
58     ]
59   }
60 ]
61 }
62 }

```

Após definida a AgroLD, foi definida também uma versão em JSON *Schema* (como demonstrado a seguir), a qual define as regras de sintaxe da linguagem, sendo utilizada como validador das estruturas, auferindo se uma estrutura JSON obedece às regras estabelecidas para esta linguagem.

Listing 5.2 – Estrutura da AgroLD em JSON *Schema*

```

1 {
2   "type": "object",
3   "properties": {
4     "entity": {
5       "type": "object",
6       "properties": {
7         "id": {
8           "type": "string"
9         },
10      "sensors": {
11        "type": "array",
12        "items": [

```

```
13         {
14             "type": "object",
15             "properties": {
16                 "id": {
17                     "type": "string"
18                 },
19                 "data": {
20                     "type": "array",
21                     "items": {}
22                 }
23             },
24             "required": [
25                 "id",
26                 "data"
27             ]
28         }
29     ],
30 },
31 "actuators": {
32     "type": "array",
33     "items": [
34         {
35             "type": "object",
36             "properties": {
37                 "id": {
38                     "type": "string"
39                 },
40                 "actions": {
41                     "type": "array",
42                     "items": {}
43                 }
44             },
45             "required": [
46                 "id",
47                 "actions"
48             ]
49         }
50     ]
51 },
52 "cpu": {
53     "type": "object",
54     "properties": {
55         "model": {
56             "type": "string"
57         },
58         "cores": {
```

```
59         "type": "string"
60     },
61     "lithography": {
62         "type": "object",
63         "properties": {
64             "value": {
65                 "type": "string"
66             },
67             "unity": {
68                 "type": "string"
69             }
70         },
71         "required": [
72             "value",
73             "unity"
74         ]
75     },
76     "clock": {
77         "type": "object",
78         "properties": {
79             "value": {
80                 "type": "string"
81             },
82             "unity": {
83                 "type": "string"
84             }
85         },
86         "required": [
87             "value",
88             "unity"
89         ]
90     },
91     "temperature": {
92         "type": "object",
93         "properties": {
94             "min": {
95                 "type": "string"
96             },
97             "max": {
98                 "type": "string"
99             },
100             "unity": {
101                 "type": "string"
102             }
103         },
104         "required": [
```

```
105         "min",
106         "max",
107         "unity"
108     ]
109 }
110 },
111 "required": [
112     "model",
113     "cores",
114     "lithography",
115     "clock",
116     "temperature"
117 ]
118 },
119 "memory": {
120     "type": "array",
121     "items": [
122         {
123             "type": "object",
124             "properties": {
125                 "id": {
126                     "type": "string"
127                 },
128                 "model": {
129                     "type": "string"
130                 },
131                 "type": {
132                     "type": "string"
133                 },
134                 "capacity": {
135                     "type": "object",
136                     "properties": {
137                         "value": {
138                             "type": "string"
139                         },
140                         "unity": {
141                             "type": "string"
142                         }
143                     },
144                     "required": [
145                         "value",
146                         "unity"
147                     ]
148                 },
149                 "speed": {
150                     "type": "object",
```

```
151         "properties": {
152             "value": {
153                 "type": "string"
154             },
155             "unity": {
156                 "type": "string"
157             }
158         },
159         "required": [
160             "value",
161             "unity"
162         ]
163     },
164 },
165 "required": [
166     "id",
167     "model",
168     "type",
169     "capacity",
170     "speed"
171 ],
172 },
173 ],
174 },
175 "battery": {
176     "type": "object",
177     "properties": {
178         "model": {
179             "type": "string"
180         },
181         "type": {
182             "type": "string"
183         },
184         "capacity": {
185             "type": "object",
186             "properties": {
187                 "value": {
188                     "type": "string"
189                 },
190                 "unity": {
191                     "type": "string"
192                 }
193             },
194             "required": [
195                 "value",
196                 "unity"
```

```
197     ]
198   },
199   "temperature": {
200     "type": "object",
201     "properties": {
202       "min": {
203         "type": "string"
204       },
205       "max": {
206         "type": "string"
207       },
208       "unity": {
209         "type": "string"
210       }
211     },
212     "required": [
213       "min",
214       "max",
215       "unity"
216     ]
217   }
218 },
219 "required": [
220   "model",
221   "type",
222   "capacity",
223   "temperature"
224 ]
225 },
226 "communication": {
227   "type": "array",
228   "items": [
229     {
230       "type": "object",
231       "properties": {
232         "id": {
233           "type": "string"
234         },
235         "protocol": {
236           "type": "string"
237         },
238         "range": {
239           "type": "object",
240           "properties": {
241             "value": {
242               "type": "string"
```

```
243         },
244         "unity": {
245             "type": "string"
246         }
247     },
248     "required": [
249         "value",
250         "unity"
251     ]
252 },
253 "speed": {
254     "type": "object",
255     "properties": {
256         "value": {
257             "type": "string"
258         },
259         "unity": {
260             "type": "string"
261         }
262     },
263     "required": [
264         "value",
265         "unity"
266     ]
267 }
268 },
269 "required": [
270     "id",
271     "protocol",
272     "range",
273     "speed"
274 ]
275 }
276 ]
277 },
278 "functions": {
279     "type": "array",
280     "items": [
281         {
282             "type": "object",
283             "properties": {
284                 "id": {
285                     "type": "string"
286                 },
287                 "alias": {
288                     "type": "string"
```

```
289     },
290     "description": {
291       "type": "string"
292     },
293     "parameters": {
294       "type": "array",
295       "items": [
296         {
297           "type": "object",
298           "properties": {
299             "id": {
300               "type": "string"
301             },
302             "type": {
303               "type": "string"
304             },
305             "description": {
306               "type": "string"
307             },
308             "enum": {
309               "type": "array",
310               "items": {}
311             },
312             "aliasEnum": {
313               "type": "array",
314               "items": {}
315             },
316             "required": {
317               "type": "string"
318             }
319           },
320           "required": [
321             "id",
322             "type",
323             "description",
324             "enum",
325             "aliasEnum",
326             "required"
327           ]
328         }
329       ]
330     },
331     "response": {
332       "type": "array",
333       "items": [
334         {
```

```
335         "type": "object",
336         "properties": {
337             "id": {
338                 "type": "string"
339             },
340             "alias": {
341                 "type": "string"
342             },
343             "type": {
344                 "type": "string"
345             },
346             "unity": {
347                 "type": "string"
348             }
349         },
350         "required": [
351             "id",
352             "alias",
353             "type",
354             "unity"
355         ]
356     },
357 ]
358 }
359 },
360 "required": [
361     "id",
362     "alias",
363     "description",
364     "parameters",
365     "response"
366 ]
367 }
368 ]
369 }
370 },
371 "required": [
372     "id",
373     "sensors",
374     "actuators",
375     "cpu",
376     "memory",
377     "battery",
378     "communication",
379     "functions"
380 ]
```

```
381   }  
382 },  
383 "required": [  
384   "entity"  
385 ]  
386 }
```

5.4 Etapa 4: Integração das Linguagens de Design com o Plante Box

Esta etapa consiste na integração da AgroLD e da *Digital Twins Definition Language* (DTDL) versão 2 (MICROSOFT, 2020a) (apresentada na Seção 2.2.2.5.4) com o Plante Box no contexto do estudo de caso da Etapa 2 deste capítulo. Destaca-se que, ao se considerar o nível arquitetural, a AgroLD e DTDL ficam armazenadas dentro do componente Linguagem de Design ilustrado na arquitetura da Figura 19.

A AgroLD é capaz de representar em um único arquivo o modelo (conjunto de atributos de uma entidade) e os valores (valores dos atributos) de todos os componentes de hardware, bem como as funções associadas ao Plante Box, conforme ilustrado na estrutura da Figura 20.

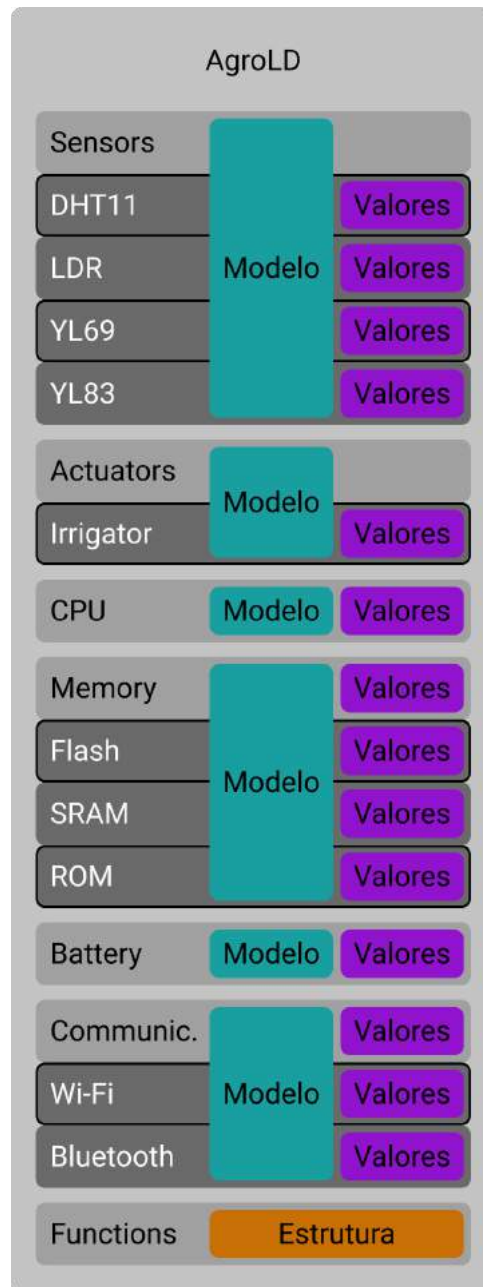
Já a DTDL necessita de arquivos separados. Primeiramente são necessários 9 arquivos para descrever o modelo de cada tipo de componente de hardware, conforme o bloco **DTDL (Interfaces)** na Figura 21. Cada linha representa um arquivo diferente, contendo uma Interface diferente. Além dos 9, é necessário outro arquivo para representar a estrutura do Plante Box como um todo. Nesse arquivo separado são realizadas as ligações com as Interfaces criadas, e são descritas as funções existentes.

As demais diferenças (que possuem um cunho mais específico) entre as linguagens estão elencadas a seguir, através da comparação entre os códigos JSON de alguns dos componentes do Plante Box.

Os códigos JSON apresentados a seguir foram formatados automaticamente pelo Prettier¹, uma ferramenta utilizada para formatar linhas de código.

¹ <https://prettier.io/>

Figura 20 – Estrutura da AgroLD aplicada ao Plante Box.



Fonte: Autor.

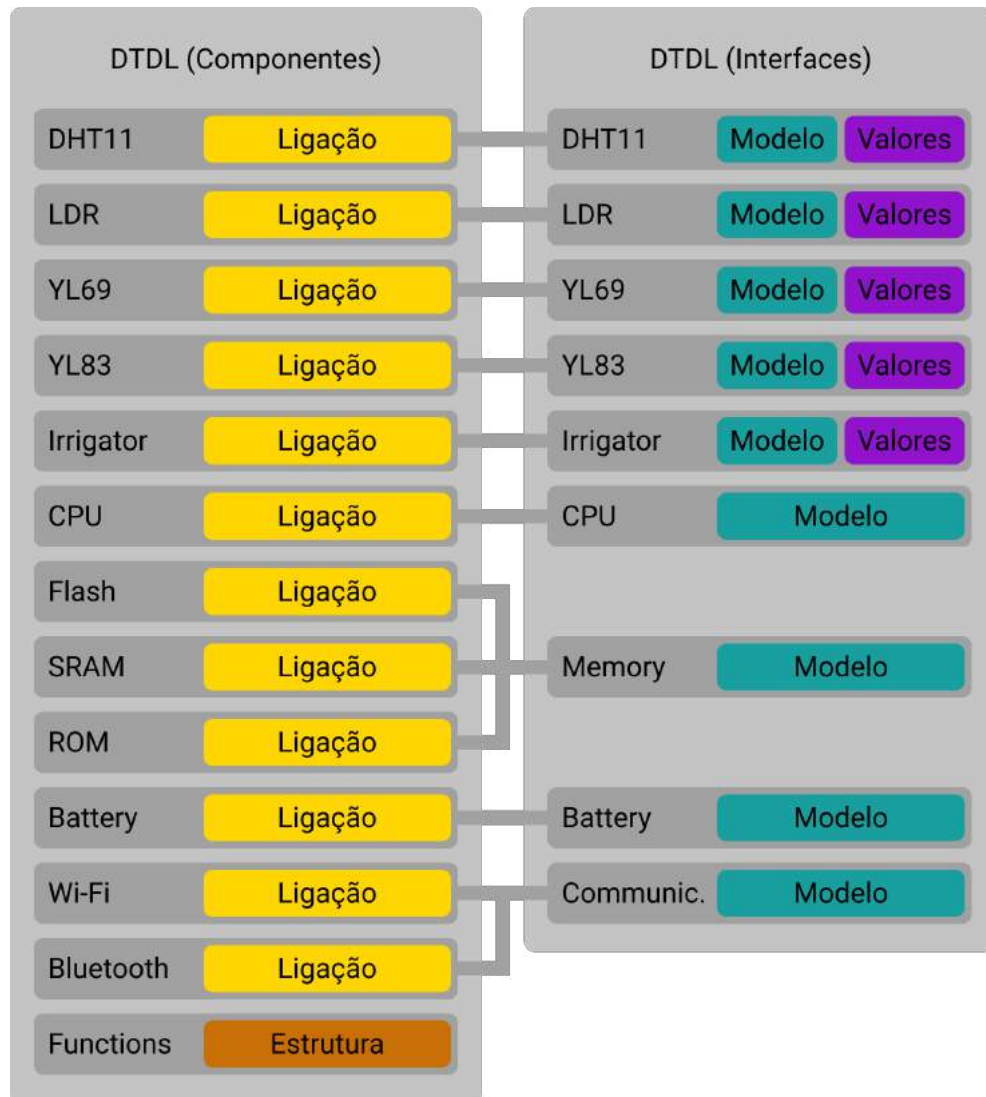
5.4.1 Sensors

Esta seção apresenta os mecanismos utilizados pela AgroLD e DTDL para descrever os sensores do Plante Box.

5.4.1.1 AgroLD

Na Listing 5.3 é mostrado como os sensores têm seu modelo e valores representados na AgroLD. Para cada sensor informa-se o id e os tipos de dados (data) que ele coleta. Por exemplo, o Plante Box possui um sensor chamado “DHT11” (o

Figura 21 – Estrutura da DTDL aplicada ao Plante Box.



Fonte: Autor.

id) responsável por coletar “temperatura” e “umidade” (os data).

Listing 5.3 – Sensores na AgroLD

```

1 "sensors": [
2   {
3     "id": "dht11",
4     "data": ["temperatura", "umidade"]
5   },
6   {
7     "id": "ldr",
8     "data": ["luz"]
9   },
10  {
11    "id": "yl69",
12    "data": ["umidade do solo"]

```

```

13 },
14 {
15     "id": "yl83",
16     "data": ["chuva"]
17 }
18 ],

```

5.4.1.2 DTDL

Em se tratando da DTDL, a Listing 5.4 representa as ligações entre o arquivo principal e as Interfaces criadas para cada sensor. Por exemplo, está sendo criado um componente (pertencente ao *Plante Box*) denominado “dht11” o qual herda as características da Interface “dtmi:com:example:dht11;1”².

Levando em consideração o fato de que cada sensor coleta diferentes tipos de dados (denominados *Telemetry*, de acordo com (MICROSOFT, 2020b)), e que esses tipos de dados são declarados de forma estática dentro das Interfaces, então não foi possível criar um componente de sensor genérico para posteriormente editar seus valores, sendo necessária uma Interface para cada sensor. Esse fator torna a DTDL uma linguagem menos eficiente do ponto de vista do consumo de dados, ocupando mais espaço de memória (se comparada à AgroLD) dentro do componente LD (Figura 19).

Listing 5.4 – Sensores na DTDL

```

1 {
2     "@type": "Component",
3     "name": "dht11",
4     "schema": "dtmi:com:example:dht11;1",
5     "@id": "",
6     "comment": "",
7     "description": "",
8     "displayName": ""
9 },
10 {
11     "@type": "Component",
12     "name": "ldr",
13     "schema": "dtmi:com:example:ldr;1",
14     "@id": "",
15     "comment": "",
16     "description": "",
17     "displayName": ""
18 },
19 {

```

² O prefixo de identificação “dtmi:com:example:” é utilizado como padrão na documentação oficial (MICROSOFT, 2020b), bem como o sufixo “;1”. Sendo assim, eles também são utilizados nos códigos aqui apresentados.

```
20  "@type": "Component",
21  "name": "yl69",
22  "schema": "dtmi:com:example:yl69;1",
23  "@id": "",
24  "comment": "",
25  "description": "",
26  "displayName": ""
27 },
28 {
29   "@type": "Component",
30   "name": "yl83",
31   "schema": "dtmi:com:example:yl83;1",
32   "@id": "",
33   "comment": "",
34   "description": "",
35   "displayName": ""
36 }
```

Na Listing 5.5 está exemplificada a estruturação da Interface do componente DHT11, responsável por coletar dados temperatura e umidade. Como se trata de um sensor é necessário criar um componente *Telemetry* para cada tipo de dado coletado, informando, ao menos, o nome (temperatura, por exemplo), schema (string, por exemplo) e unit (graus celsius, por exemplo).

Listing 5.5 – Interface do DHT11 na DTDL

```
1 {
2   "@context": "dtmi:dtdl:context;2",
3   "@id": "dtmi:com:example:dht11;1",
4   "@type": "Interface",
5   "contents": [
6     {
7       "@type": ["Telemetry"],
8       "name": "temperatura",
9       "schema": "string",
10      "@id": "",
11      "comment": "",
12      "description": "",
13      "displayName": "",
14      "unit": "degreeCelsius"
15    },
16    {
17      "@type": ["Telemetry"],
18      "name": "umidade",
19      "schema": "string",
20      "@id": "",
21      "comment": "",
```

```
22     "description": "",
23     "displayName": "",
24     "unit": "percent"
25   }
26 ]
27 }
```

5.4.2 Actuators

Esta seção apresenta os mecanismos utilizados pela AgroLD e DTDL para descrever o atuador do Plante *Box*.

5.4.2.1 AgroLD

Conforme a Listing 5.6, para cada atuador informa-se o id e os tipos de ações (actions) que ele realiza. No caso específico do Plante *Box* o único atuador é o irrigador, cuja ação é irrigar. Para tal, são utilizadas seis linhas de código.

Listing 5.6 – Atuadores na AgroLD

```
1 "actuators": [
2   {
3     "id": "irrigador",
4     "actions": ["irrigar"]
5   }
6 ]
```

5.4.2.2 DTDL

Baseado no DTDL, é utilizado o código demonstrado na Listing 5.7 para realizar a ligação da Interface do Irrigador com o arquivo principal. Nesse arquivo, deve-se informar, ao menos o @type (que no caso do irrigador, seria um Component), o name (denominado irrigador), e o schema que deve possuir o mesmo valor que o @id da Interface do irrigador, demonstrado na Listing 5.8.

Listing 5.7 – Irrigador na DTDL

```
1 {
2   "@type": "Component",
3   "name": "irrigador",
4   "schema": "dtmi:com:example:irrigador;1",
5   "@id": "",
6   "comment": "",
7   "description": "",
8   "displayName": ""
9 }
```

Como demonstrado na Listing 5.8, na interface, para representar os tipos de ações desempenhadas, utiliza-se o tipo *Property*, informando o nome (irrigar) e o schema (*boolean*, informando que há apenas dois estados: ligado e desligado). Novamente, a DTDL se mostra menos eficiente no quesito eficiência de armazenamento, uma vez que são necessárias apenas 6 linhas para a AgroLD representar o irrigador, contra 27 (9 + 18) da DTDL para fazer o mesmo.

Listing 5.8 – Interface do Irrigador na DTDL

```
1 {
2   "@context": "dtmi:dtdl:context;2",
3   "@id": "dtmi:com:example:irrigador;1",
4   "@type": "Interface",
5   "contents": [
6     {
7       "@type": "Property",
8       "name": "Irigar",
9       "schema": "boolean",
10      "@id": "",
11      "comment": "",
12      "description": "",
13      "displayName": "",
14      "unit": "",
15      "writable": ""
16    }
17  ]
18 }
```

5.4.3 CPU e Battery

Esta seção apresenta os mecanismos utilizados pela AgroLD e DTDL para descrever CPU e Battery do Plante Box. Ambos componentes possuem similaridades estruturais, a nível de código. Logo, será utilizada apenas a CPU para exemplificar o funcionamento de ambos os componentes.

5.4.3.1 AgroLD

A CPU (assim como a Battery) é um tipo de componente que necessita ser descrito apenas uma vez, isto é, uma vez que um dispositivo eletrônico possua uma CPU as características básicas dela (modelo, núcleos, velocidade etc) só mudarão caso ela seja substituída por outra peça. Logo, as características gerais desse tipo de componente tendem a não ser atualizadas com o tempo, com raras exceções. Tomando o Plante como exemplo, componentes como CPU, Memory, chips de Communication são soldados e integrados (*on-board*) à placa-mãe, não sendo algo modular, dificultando a troca de tais peças.

A Listing 5.9 demonstra como a CPU é descrita utilizando a AgroLD. Destaca-se que todos os valores definidos como padrão podem ser atualizados, ao longo do tempo, através de funções destinadas a tal. Tais funções são acionadas a partir da troca de mensagens entre a EV e EF no processo de sincronização, garantindo que o estado representado na EV seja idêntico ao estado da EF, e vice-versa.

Ao todo são utilizadas 11 linhas para descrever a CPU, englobando detalhes como modelo, quantidade de núcleos, litografia, velocidade e temperatura de operação.

Listing 5.9 – CPU na AgroLD

```
1 "CPU": {  
2   "model": "ESP 32S",  
3   "cores": "1",  
4   "lithography": { "value": "5", "unity": "mm" },  
5   "clock": { "value": "240", "unity": "MHz" },  
6   "temperature": {  
7     "min": "-40",  
8     "max": "150",  
9     "unity": "degreeCelsius"  
10  }  
11 }
```

5.4.3.2 DTDL

A Listing 5.10 demonstra a parcela de código responsável por realizar a ligação do componente CPU no arquivo principal com sua interface correspondente. Lembrando que o arquivo principal/raiz seria a parcela à esquerda da Figura 21, enquanto todas as interfaces estão descritas na parcela à direita. O que determina essa ligação é o campo `schema` que aponta para o `@id` da interface correspondente, demonstrado na Listing 5.11.

Listing 5.10 – CPU na DTDL

```
1 {  
2   "@type": "Component",  
3   "name": "CPU",  
4   "schema": "dtmi:com:example:CPU;1",  
5   "@id": "",  
6   "comment": "",  
7   "description": "",  
8   "displayName": ""  
9 }
```

A Listing 5.11 apresenta a implementação da interface CPU. Uma diferença importante entre a implementação desse tipo de componente na DTDL e na AgroLD é

que na DTDL não é possível definir valores padrão para cada característica do modelo da CPU. É possível apenas definir o modelo. Para definir os valores é necessário, após definir o modelo, estabelecer uma comunicação entre a EV e a EF, onde a Entidade Física envia uma mensagem informando os valores. Em outras palavras, os valores só podem ser estabelecidos após toda a infraestrutura de *Digital Twin* estar funcionando. Este fator torna a AgroLD mais vantajosa, podendo-se declarar um valor padrão no próprio código do modelo, sem necessidade de realizar uma etapa adicional.

Destaca-se que esse comportamento ocorre com os componentes CPU, Memory, Battery e Communication, sendo necessária uma etapa adicional para definir os valores padrão, uma vez que não é possível defini-los no próprio modelo (MICROSOFT, 2020b). Esse fator é desvantajoso para a DTDL, pois esses componentes não mudam constantemente com o passar do tempo, a não ser que apresentem algum tipo de defeito ou sejam substituídos, fazendo com que a definição de valores padrão no próprio modelo seja mais coerente.

É importante frisar que, na AgroLD, os valores definidos como padrão podem ser atualizados ao longo do tempo. Eles não são estáticos ou absolutos. Para realizar sua alteração, é necessária apenas a criação de uma função que verifica o estado dos componentes de hardware, averiguando se houve alguma mudança, e atualizando os valores do modelo caso tenha havido.

Outra desvantagem do DTDL é a quantidade de linhas utilizadas para descrever o componente CPU. Foram necessárias 11 linhas para a AgroLD e 151 para a DTDL. Frisa-se, novamente, que estão sendo levados em consideração todos os campos opcionais não preenchidos, os quais ocuparão espaço em memória no componente *Digital Twin* (Figura 19), mesmo que não estejam preenchidos.

Listing 5.11 – Interface da CPU na DTDL

```
1 {
2   "@context": "dtmi:dtdl:context;2",
3   "@id": "dtmi:com:example:CPU;1",
4   "@type": "Interface",
5   "contents": [
6     {
7       "@type": "Property",
8       "name": "model",
9       "schema": "string",
10      "@id": "",
11      "comment": "",
12      "description": "",
13      "displayName": "",
14      "unit": "",
15      "writable": ""
16    },
17    {
18      "@type": "Property",
```

```
19     "name": "cores",
20     "schema": "string",
21     "@id": "",
22     "comment": "",
23     "description": "",
24     "displayName": "",
25     "unit": "",
26     "writable": ""
27 },
28 {
29     "@type": "Property",
30     "name": "lithography",
31     "schema": {
32         "@type": "Object",
33         "fields": [
34             {
35                 "name": "value",
36                 "schema": "string",
37                 "@id": "",
38                 "comment": "",
39                 "description": "",
40                 "displayName": ""
41             },
42             {
43                 "name": "unity",
44                 "schema": "string",
45                 "@id": "",
46                 "comment": "",
47                 "description": "",
48                 "displayName": ""
49             }
50         ],
51         "@id": "",
52         "comment": "",
53         "description": "",
54         "displayName": ""
55     },
56     "@id": "",
57     "comment": "",
58     "description": "",
59     "displayName": "",
60     "unit": "",
61     "writable": ""
62 },
63 {
64     "@type": "Property",
```

```
65     "name": "clock",
66     "schema": {
67         "@type": "Object",
68         "fields": [
69             {
70                 "name": "value",
71                 "schema": "string",
72                 "@id": "",
73                 "comment": "",
74                 "description": "",
75                 "displayName": ""
76             },
77             {
78                 "name": "unity",
79                 "schema": "string",
80                 "@id": "",
81                 "comment": "",
82                 "description": "",
83                 "displayName": ""
84             }
85         ],
86         "@id": "",
87         "comment": "",
88         "description": "",
89         "displayName": ""
90     },
91     "@id": "",
92     "comment": "",
93     "description": "",
94     "displayName": "",
95     "unit": "",
96     "writable": ""
97 },
98 {
99     "@type": "Property",
100     "name": "temperature",
101     "schema": {
102         "@type": "Object",
103         "fields": [
104             {
105                 "name": "min",
106                 "schema": "string",
107                 "@id": "",
108                 "comment": "",
109                 "description": "",
110                 "displayName": ""
```

```

111     },
112     {
113         "name": "max",
114         "schema": "string",
115         "@id": "",
116         "comment": "",
117         "description": "",
118         "displayName": ""
119     },
120     {
121         "name": "unity",
122         "schema": "string",
123         "@id": "",
124         "comment": "",
125         "description": "",
126         "displayName": ""
127     }
128 ],
129 "@id": "",
130 "comment": "",
131 "description": "",
132 "displayName": ""
133 },
134 "@id": "",
135 "comment": "",
136 "description": "",
137 "displayName": "",
138 "unit": "",
139 "writable": ""
140 }
141 ]
142 }

```

5.4.4 Memory e Communication

Esta seção apresenta os mecanismos utilizados pela AgroLD e DTDL para descrever Memory e Communication do Plante Box. Para exemplificar o funcionamento de ambos componentes (que estão definidos como listas) será usado como base o componente Communication.

5.4.4.1 AgroLD

O componente comunicação é uma lista usada para descrever os meios/recursos de comunicação disponíveis ao Plante Box. No caso, trata-se do Wi-Fi e do Bluetooth. A Listing 5.12 demonstra como o componente é descrito. Basicamente se pode ver uma lista, chamada communication, e dentro dessa lista dois objetos, um

representando o wifi e outro o bluetooth. Os valores definidos são correspondentes às características do Plante Box.

Listing 5.12 – Comunicação na AgroLD

```
1 "communication": [  
2   {  
3     "id": "wifi",  
4     "protocol": "IEEE 802.11 b/g/n",  
5     "range": { "value": "100/180", "unity": "m" },  
6     "speed": { "value": "150", "unity": "Mbps" }  
7   },  
8   {  
9     "id": "bluetooth",  
10    "protocol": "BLE v4.2",  
11    "range": { "value": "60", "unity": "m" },  
12    "speed": { "value": "1", "unity": "Mbps" }  
13  }  
14 ]
```

5.4.4.2 DTDL

A forma de declarar esse tipo de lista é diferente entre a AgroLD e a DTDL. A DTDL requer a criação separada de uma interface genérica (DTDL (Interfaces)/Comunicação), e, após isso, a criação de uma instância para cada componente de comunicação (Wi-Fi, Bluetooth) no arquivo principal. Além dessa diferença, há o fato de, na DTDL, ser definido apenas o modelo, sem os valores padrão; sendo necessário, para tal, que a EV se comunique com sua contra-parte física (Entidade Física) e colete os valores. Para exemplificar, a Listing 5.13 demonstra a criação do componente wifi, que herda o modelo da Interface Comunicação.

Listing 5.13 – Wi-Fi na DTDL

```
1 {  
2   "@type": "Component",  
3   "name": "wifi",  
4   "schema": "dtmi:com:example:comunicacao;1",  
5   "@id": "",  
6   "comment": "",  
7   "description": "",  
8   "displayName": ""  
9 }
```

A Listing 5.14 descreve a Interface Comunicação, na qual são definidas as seguintes propriedades: range (representa o alcance que a tecnologia de comunicação sem fio alcança, por exemplo, 100 metros), speed (a velocidade da tecnologia, por exemplo, 100 Mbps).

Listing 5.14 – Interface da Comunicação na DTDL

```
1 {
2   "@context": "dtmi:dtdl:context;2",
3   "@id": "dtmi:com:example:comunicacao;1",
4   "@type": "Interface",
5   "contents": [
6     {
7       "@type": "Property",
8       "name": "protocol",
9       "schema": "string",
10      "@id": "",
11      "comment": "",
12      "description": "",
13      "displayName": "",
14      "unit": "",
15      "writable": ""
16    },
17    {
18      "@type": "Property",
19      "name": "range",
20      "schema": {
21        "@type": "Object",
22        "fields": [
23          {
24            "name": "value",
25            "schema": "string",
26            "@id": "",
27            "comment": "",
28            "description": "",
29            "displayName": ""
30          },
31          {
32            "name": "unity",
33            "schema": "string",
34            "@id": "",
35            "comment": "",
36            "description": "",
37            "displayName": ""
38          }
39        ],
40        "@id": "",
41        "comment": "",
42        "description": "",
43        "displayName": ""
44      },
45      "@id": "",
```

```
46     "comment": "",
47     "description": "",
48     "displayName": "",
49     "unit": "",
50     "writable": ""
51 },
52 {
53     "@type": "Property",
54     "name": "speed",
55     "schema": {
56         "@type": "Object",
57         "fields": [
58             {
59                 "name": "value",
60                 "schema": "string",
61                 "@id": "",
62                 "comment": "",
63                 "description": "",
64                 "displayName": ""
65             },
66             {
67                 "name": "unity",
68                 "schema": "string",
69                 "@id": "",
70                 "comment": "",
71                 "description": "",
72                 "displayName": ""
73             }
74         ],
75         "@id": "",
76         "comment": "",
77         "description": "",
78         "displayName": ""
79     },
80     "@id": "",
81     "comment": "",
82     "description": "",
83     "displayName": "",
84     "unit": "",
85     "writable": ""
86 }
87 ]
88 }
```

5.4.5 Functions

A título de exemplo foram criadas 4 funções: **ler** (para ler dados dos sensors), **ligarDesligar** (para ligar ou desligar o actuador ou os sensors), **temperaturaOperacao** (para ler temperatura de operação da CPU e battery), **medirConsumo** (para ler consumo da CPU, battery e memória flash). Para exemplificar a descrição das funções na AgroLD e DTDL, foi tomada como exemplo a função **ler**. As demais funções podem ser consultadas nos Apêndices A e B, onde estão descritos os códigos completos da AgroLD e DTDL, inclusive as funções.

5.4.5.1 AgroLD

A Listing 5.15 descreve a estrutura da função **ler**, conforme o padrão da AgroLD. Nela são definidos, dentre outras coisas, o **id** (identificar), **alias** (utilizado na troca de mensagens, para identificar a função, ao invés do **id**, com objetivo de economia de dados), os **parameters** (parâmetros) que a função deve receber (que no caso da função ler, não são necessários, estando todos os campos definidos como vazios), e o **formato de response** (resposta), indicando os campos que serão enviados como resposta à chamada da função.

Listing 5.15 – Funções na AgroLD

```
1 "functions": [  
2 {  
3   "id": "ler",  
4   "alias": "l",  
5   "description": "ler dados dos sensores",  
6   "parameters": [  
7     {  
8       "id": "",  
9       "type": "",  
10      "description": "",  
11      "enum": [],  
12      "aliasEnum": [],  
13      "required": ""  
14    }  
15  ],  
16  "response": [  
17    {  
18      "id": "temperatura",  
19      "alias": "t",  
20      "type": "float",  
21      "unity": "degreeCelsius"  
22    },  
23    {  
24      "id": "umidade",  
25      "alias": "u",  
26      "type": "float",
```

```

27     "unity": "percent"
28   },
29   {
30     "id": "umidade do solo",
31     "alias": "us",
32     "type": "float",
33     "unity": "percent"
34   },
35   {
36     "id": "chuva",
37     "alias": "c",
38     "type": "float",
39     "unity": "percent"
40   },
41   {
42     "id": "luz",
43     "alias": "l",
44     "type": "float",
45     "unity": "percent"
46   }
47 ]
48 }
49 ]

```

A função **ler**, para ser acionada, necessita que a EV envie uma mensagem à Entidade Física. Do mesmo modo, quando a Entidade Física receber a requisição ela necessita enviar uma mensagem, para a EV, contendo a resposta.

Conforme a Listing 5.16, o formato de requisição para a função **ler** requer apenas que se envie o *alias* (apelido) da função (que neste caso é a letra “l”) e os parâmetros da função (que neste caso não existem). O mecanismo de *alias* serve para economizar dados no processo de comunicação entre a Entidade Física e a Digital, requerendo uma menor largura de banda de rede. Se, ao invés do *alias*, se utilizasse o *id* (nome completo) da função, isso resultaria em mais dados transferidos via rede.

Listing 5.16 – Função Ler (Requisição) na AgroLD

```

1 { "l": {} }

```

O formato de resposta, exemplificado na Listing 5.17, também utiliza os recursos de *alias* (t é temperatura, u é umidade, us é umidade do solo, c é chuva e l é luminosidade), propiciando uma comunicação mais leve. Destaca-se que os números são meros exemplos de valores.

Listing 5.17 – Função Ler (Resposta) na AgroLD

```

1 { "l": { "t": 25, "u": 70, "us": 60, "c": 30, "l": 55 } }

```

5.4.5.2 DTDL

Na Listing 5.18 está descrita a estrutura função **ler**, conforme o padrão da *Digital Twins Definition Language*. Nesse caso, por se tratar de uma função, sua descrição já fica no arquivo principal, não necessitando de uma interface em um arquivo separado.

Listing 5.18 – Função Ler na DTDL

```
1 {
2   "@type": "Command",
3   "name": "ler",
4   "@id": "",
5   "comment": "",
6   "description": "ler dados dos sensores",
7   "displayName": "",
8   "request": {
9     "name": "",
10    "schema": "",
11    "@id": "",
12    "comment": "",
13    "description": "",
14    "displayName": ""
15  },
16  "response": {
17    "name": "dados dos sensores",
18    "schema": {
19      "@type": "Object",
20      "fields": [
21        {
22          "name": "temperatura",
23          "schema": "float",
24          "@id": "",
25          "comment": "unidade: degreeCelsius",
26          "description": "",
27          "displayName": ""
28        },
29        {
30          "name": "umidade",
31          "schema": "float",
32          "@id": "",
33          "comment": "unidade: percent",
34          "description": "",
35          "displayName": ""
36        },
37        {
38          "name": "umidade do solo",
```

```

39     "schema": "float",
40     "@id": "",
41     "comment": "unidade: percent",
42     "description": "",
43     "displayName": ""
44 },
45 {
46     "name": "chuva",
47     "schema": "float",
48     "@id": "",
49     "comment": "unidade: percent",
50     "description": "",
51     "displayName": ""
52 },
53 {
54     "name": "luz",
55     "schema": "float",
56     "https://www.overleaf.com/project/61154f99b34f18f478f95fbe"
57     "@id": "",
58     "comment": "unidade: percent",
59     "description": "",
60     "displayName": ""
61 },
62 {
63     "@id": "",
64     "comment": "",
65     "description": "",
66     "displayName": ""
67 },
68 {
69     "@id": "",
70     "comment": "",
71     "description": "",
72     "displayName": ""

```

Diferentemente da AgroLD, a DTDL não implementa o recurso de *alias*, sendo necessário utilizar o nome completo da função, bem como os nomes completos dos tipos de dados coletados pelos sensores. As Listing 5.19 e 5.20 descrevem, respectivamente, os formatos de requisição e resposta.

Listing 5.19 – Função Ler (Requisição) na DTDL

```
1 { "ler": {} }
```

Listing 5.20 – Função Ler (Resposta) na DTDL

```
1 {  
2   "ler": {  
3     "temperatura": 25,  
4     "umidade": 70,  
5     "umidade do solo": 60,  
6     "chuva": 30,  
7     "luz": 55  
8   }  
9 }
```

Os modelos completos da AgroLD e da DTDL, ambas descrevendo por completo os recursos do Plante Box, estão descritos nos Apêndices A e B, respectivamente. Destaca-se que na realização da Avaliação Experimental, realizada no Capítulo 7, são utilizados por completo todas as estruturas da LDs descritas nos Apêndices.

6 Compreensão das formas de avaliação de arquiteturas de DT

Este capítulo detalha a Fase 4 da metodologia descrita no Capítulo 3. A metodologia utilizada para elaboração deste capítulo é apresentada em 3 etapas. A Etapa 1 (Seção 6.1) é a apresentação da metodologia usada para a realização de uma revisão sistemática que visa compreender as formas de avaliação de arquiteturas de DT. A Etapa 2 (Seção 6.2) consiste no detalhamento das arquiteturas revisadas respondendo às questões de pesquisa definidas na Etapa 1. E a Etapa 3 (Seção 6.3) apresenta uma breve comparação entre as arquiteturas revisadas e esta dissertação, apontando as metodologias de avaliação.

6.1 Etapa 1: Metodologia

A metodologia utilizada para a revisão da literatura sobre arquitetura de DT e suas formas de avaliação foi dividida em 6 partes.

1. Definição das Questões de Pesquisa (QP).
 - Para qual área/domínio a arquitetura foi proposta?
 - Quais as camadas presentes nas arquiteturas de DT?
 - Quais as características de cada camada?
 - Como o artigo valida a arquitetura proposta?
2. Busca pelo termo “*digital twin*” nos portais *ACM Digital Library*, *IEEE Xplore*, *Science Direct* e *Springer*, aplicando o filtro de data para publicações de 2019 a 2022, resultando em 1.306 artigos.
3. Remoção de artigos duplicados, resultando em 1.192 artigos.
4. Análise dos títulos dos trabalhos, selecionando aqueles que incluíam os termos “*digital twin*” ou “*digital twins*”, resultando em 495 artigos.
5. Análise dos títulos dos trabalhos, selecionando aqueles que incluíam os termos “*architecture*” ou “*arquitetura*”, resultando em 28 artigos.
6. Leitura e seleção dos artigos com proposta de arquitetura de DT estruturada em camadas suficientemente descritas e distinguíveis, resultando em 14 artigos.

A seguir, os 14 artigos são apresentados, respondendo às QP da Etapa 1. Os trabalhos são apresentados em ordem cronológica.

6.2 Etapa 2: Detalhamento das Arquiteturas

6.2.1 Arquitetura de (CATARCI et al., 2019)

Os autores de (CATARCI et al., 2019) apresentaram uma arquitetura de DT no contexto da Indústria, com aplicação específica em Fabricação Inteligente baseado em gêmeos digitais orientados a serviços, com 4 camadas. Não é apresentada avaliação para a arquitetura.

Tabela 5 – Camadas da Arquitetura de (CATARCI et al., 2019).

Nome da Camada	Descrição da Camada
<i>Digital Twins</i>	Envolve entidades físicas incluídas no processo. Essas entidades físicas podem ser máquinas de fabricação ou operadores humanos. Um DT expõe uma API Web composta por 3 partes: síncrona, interface de consulta e assíncrona.
Espaço de dados	Todos os dados disponíveis para o processo. Esses dados são heterogêneos em sua natureza do ponto de vista da tecnologia de acesso, do esquema empregado e do vocabulário empregado.
Mediador	Para atingir o objetivo definido pelo supervisor humano, os gêmeos e os dados disponíveis devem ser integrados. Há duas fases: a fase de síntese (as especificações das APIs expostas pelos gêmeos digitais e os metadados são compostos para construir um processo mediador e a fase de execução (o mediador executa seu programa preparando as mensagens de entrada para os DTs).
Supervisor humano	Quem define os objetivos do processo em termos de resultados finais e indicadores-chave de desempenho a serem obtidos.

6.2.2 Arquitetura de (REDELINGHUYS; BASSON; KRUGER, 2020)

Os autores de (REDELINGHUYS; BASSON; KRUGER, 2020) apresentaram uma arquitetura na área da Indústria, no contexto de Manufatura, que possui 6 camadas. Como forma de avaliação foi apresentado um estudo de caso utilizando uma garra robótica como gêmeo físico e um *dashboard* de DT com leituras dos seguintes dados: medições de fluxo de ar, velocidade do curso, tempo de ciclo, tempo de abertura, tempo de fechamento e tempo de aperto da garra.

Tabela 6 – Camadas da Arquitetura de (REDELINGHUYS; BASSON; KRUGER, 2020).

Nome da Camada	Descrição da Camada
Gêmeo Físico - 1	Inclui vários dispositivos físicos, como atuadores e sensores, que podem fornecer ou consumir sinais trocados com o controlador local ou dispositivo de aquisição de dados, localizado na camada “Gêmeo Físico - 2”.
Gêmeo Físico - 2	A camada 2 representa então a fonte de dados para o gêmeo físico.
Repositório Local de Dados	Interface de comunicação neutra em relação ao fornecedor entre o gêmeo físico e as outras camadas.
Gateway IoT	Adiciona contexto aos dados recebidos do “Repositório Local de Dados” e/ou processa os dados em formulários mais úteis para as camadas superiores.
Repositórios de informações baseados em nuvem	Serviços em nuvem que armazenam informações históricas recebidas do “Gateway IoT”.
Emulação e simulação	Funcionalidade de uma interface de usuário (ou painel) que conecta o usuário a informações suaves em tempo real e históricas sobre o gêmeo físico.

6.2.3 Arquitetura de (KYCHKIN; NIKOLAEV, 2020)

Os autores de (KYCHKIN; NIKOLAEV, 2020) apresentaram uma arquitetura de DT no contexto da Indústria, com aplicação específica em Sistema de Controle de Ventilação de Minas com 4 camadas. Como forma de avaliação da arquitetura

foi realizado um experimento computacional com a tarefa de previsão online do consumo de energia para o sistema de ventilação utilizando dados de simulação. Foram comparados os resultados de 5 algoritmos: *CDL*, *Ndays*, *HW*, *AR&RLS* e *Feature Extraction*. Não é dito qual algoritmo se saiu melhor, apenas apresentados os dados obtidos com cada um.

Tabela 7 – Camadas da Arquitetura de (KYCHKIN; NIKOLAEV, 2020).

Nome da Camada	Descrição da Camada
Sistema de objeto físico	Possui um ventilador, um aquecedor e um medidor de energia para monitorar estes dois dispositivos. Também conta com sensores de pressão, temperatura e fluxo de ar, que enviam dados ao broker MQTT da camada Infraestrutura de rede e computação.
Infraestrutura de rede e computação	Tem-se um broker MQTT que intermedeia a comunicação com o Sistema de objeto físico agregado a um banco de dados externo e um módulo com métricas de coleta e reportagem de dados. Também há um bloco de controle e sincronização de dados que realiza a comunicação com o <i>Digital Twin</i> .
<i>Digital twin</i>	Acessa uma base de dados de séries temporais, e algumas funcionalidades tais como detecção de acidentes e alarmes, modelos de simulação, bloco para executar funções disparadas por usuários e afins.
Interface homem-máquina	Apresenta um dashboard possibilitando a visualização de dados em tempo real, estatísticas, eventos e alarmes, bem como opção de controlar alguns parâmetros das demais camadas

6.2.4 Arquitetura de (BOTTANI; VIGNALI; TANCREDI, 2020)

Os autores de (BOTTANI; VIGNALI; TANCREDI, 2020) apresentaram uma arquitetura de DT no contexto da Indústria, com aplicação específica em Sistemas de Pasteurização para Bebidas Alimentícias com 6 camadas (e 1 agente humano). Como forma de avaliação os autores apresentaram a tendência da pressão de entrada para suco de maçã e damasco em função da variação do fluxo, em um ambiente com temperatura constante de 25°C.

Tabela 8 – Camadas da Arquitetura de (BOTTANI; VIGNALI; TANCREDI, 2020).

Nome da Camada	Descrição da Camada
Planta piloto	Camada física, incluindo os maquinários usados no sistema de pasteurização.
Sensores	Medir a temperatura e pressão de entrada dos fluidos usados para compor a bebida.
Módulo de aquisição de dados para entrada analógica	Obtém os dados dos sensores de temperatura e pressão e os repassa à Ferramenta de detecção de anomalias.
Ferramenta de detecção de anomalias	Detecta se a temperatura e pressão estão acima/abaixo do ideal.
Ferramenta de simulação	Trabalha online em conjunto com a Ferramenta de detecção de anomalias como DT, ou offline atuando como ferramenta de simulação em ambiente virtual.
Terminal de Interação Homem-Máquina	Informa possíveis anomalias de temperatura. A partir disso o agente humano pode executar alguma intervenção manual.

6.2.5 Arquitetura de (AHELEROFF et al., 2021)

Os autores de (AHELEROFF et al., 2021) apresentaram uma arquitetura, no contexto da Indústria 4.0, que possui 5 camadas. É apresentado um estudo de caso no contexto de Recursos Hídricos, que consiste em um sistema de *Digital Twin* para

monitorar e controlar parâmetros climáticos de áreas ambientais alagadas. Não é apresentada avaliação para a arquitetura.

Tabela 9 – Camadas da Arquitetura de (AHELEROFF et al., 2021).

Nome da Camada	Descrição da Camada
Físico	Equipamentos eletrônicos utilizados em processos industriais.
Comunicação	Meio usado para interconectar todas as camadas.
Digital	Uma cópia digital imitada de coisas físicas, incluindo dados geométricos e pode existir em formato digital armazenada em alguma mídia física.
Cibernético	Atua no contexto de Big Data e algoritmos para aplicativos avançados, incluindo gerenciamento de risco, otimização e análise preditiva.
Aplicação	Software/aplicativo resultante dos processos e objetivos presentes nas demais camadas.

6.2.6 Arquitetura de (ZHENG et al., 2020)

Os autores de (ZHENG et al., 2020) apresentaram uma arquitetura de DT no contexto da Indústria, com foco em Processos de Manufatura, com 5 camadas. Não é apresentada avaliação para a arquitetura.

Tabela 10 – Camadas da Arquitetura de (ZHENG et al., 2020).

Nome da Camada	Descrição da Camada
Entidades físicas	Podem consistir em dispositivo ou produto, sistema físico, processo de atividades, até mesmo uma organização.
Entidades virtuais	São réplicas fiéis de entidades físicas, que integram modelos de engenharia digital e múltiplas variáveis, escalas e habilidades das entidades físicas para reproduzir suas geometrias, propriedades físicas, comportamentos e regras no espaço virtual.
Componente Multi-Agent-System	Agentes autônomos e cooperativos representando diferentes componentes de manufatura dispostos ao longo de um sistema de produção.
Modelos semânticos	Suporte aos modelos comportamentais do Componente Multi-Agent-System e das Entidades virtuais.
Serviços	<i>Feedbacks</i> para entidades físicas e virtuais. Permite serviços de aplicação, como monitoramento e otimização de processos de produção, diagnóstico e prognóstico de falhas, manutenção preditiva etc.
Emulação e simulação	Funcionalidade de uma interface de usuário (ou painel) que conecta o usuário a informações suaves em tempo real e históricas sobre o gêmeo físico.

6.2.7 Arquitetura de (DALIBOR et al., 2020)

Os autores de (DALIBOR et al., 2020) apresentaram uma arquitetura de DT na área da Indústria, no contexto específico de Cabines de Comando, que possui 5 camadas. Não é apresentada avaliação para a arquitetura.

6.2.8 Arquitetura de (MOURTZIS et al., 2021)

Os autores de (MOURTZIS et al., 2021) apresentaram uma arquitetura, no contexto de Manufatura, que possui 3 camadas. Como estudo de caso foi proposto um Sistema de Modelagem 3D baseado em gêmeo digital. Não foram apresentados experimentos.

6.2.9 Arquitetura de (PAN et al., 2021)

Os autores de (PAN et al., 2021) apresentaram uma arquitetura na área da Indústria, no contexto de Logística de Produção, que possui 2 camadas. Como forma

Tabela 11 – Camadas da Arquitetura de (DALIBOR et al., 2020).

Nome da Camada	Descrição da Camada
Ciber-Físico	Sistemas de Produção Ciber-Física (SPCF), que é monitorado e controlado pelo DT. Ele fornece uma interface que permite a leitura dos valores de sensores.
Dados	É um extenso armazenamento de dados que pode abranger vários bancos de dados contendo dados do “Ciber-Físico” e seu contexto operacional.
Conexão	Comunica-se com a camada “Ciber-Físico” e fornece dados para a camada de “Aplicação”. Processador de dados: cria e compartilha conhecimento sobre o estado do sistema produzindo sombras digitais com base nos dados contidos na camada “Dados”. Executor: obtém uma solução descrevendo o que o SPCF deve fazer e transformando as descrições em comandos enviados ao SPCF.
Aplicação	Contém a principal funcionalidade do DT, incluindo sua capacidade de detectar comportamentos não intencionais do SPCF e decidir sobre as reações a ele. Avaliador : monitora o comportamento do sistema e detecta possíveis avarias. Ele consulta informações sobre o sistema ou seu contexto do processador de dados e recebe sombras digitais em troca. Raciocinador : se o Avaliador detectar um comportamento não intencional, ele cria uma meta e a envia ao Raciocinador, que por sua vez usa o conhecimento sobre o SPCF, sistemas similares e o contexto operacional do sistema para decidir como atingir esse objetivo. Digital Twin da Cabine de Comando: Agregador de Dados : consulta, processa e envia dados para o Processador Lógico. Processador Lógico : trata e avalia os dados e estados relevantes (recebidos pelo Agregador de Dados). Os dados resultantes são enviados para o frontend para visualizar os dados e estados do sistema. Os comandos são usados para gravar dados de volta no sistema ou para definir metas específicas para o DT.
Visualização	Inclui todos os componentes gráficos do DT da cabine de comando, usados para visualizar as sombras digitais e configurar o DT e o SPCF.

Tabela 12 – Camadas da Arquitetura de (MOURTZIS et al., 2021).

Nome da Camada	Descrição da Camada
Ambiente físico	Execução de código G ¹ , arquivo de código G baixado, interpretação, saída de impressão.
Desenho/Manufatura assistido por computador	Modelo 3D, Manufatura assistida por computador, Processamento de fatias.
Ambiente de simulação	Visualização de realidade aumentada de código G, saída prevista de visualização de realidade aumentada, visualização de simulação, simulação com <i>feed</i> de dados em tempo real.

de avaliação da proposta foram apresentados experimentos comparando o consumo de memória de dois métodos, o PLS (logística de produção sincronizada) e o N-PLS (logística de produção não sincronizada).

Tabela 13 – Camadas da Arquitetura de (PAN et al., 2021).

Nome da Camada	Descrição da Camada
Virtual	Consiste em 4 componentes. Modelos : biblioteca de modelos para construção de DT. Dados : contém dados em tempo real, básicos, derivados e históricos. Espelho : realiza o mapeamento real da camada física através do acoplamento do modelo e da fusão dos dados, garantindo agilidade, precisão e consistência entre a camada física e a camada virtual. Controle de sincronização : é o núcleo/ cérebro do DT, contém módulos necessários para sua operação e é usado para apoiar a decisão e controle de todo o sistema.
Física	Possui recursos de processamento, coleta de informações, cálculo, lógica de controle e otimização de algoritmos.

6.2.10 Arquitetura de (LIM et al., 2021)

Os autores de (LIM et al., 2021) apresentaram uma arquitetura de DT em contexto Genérico, que possui 4 camadas. Não é apresentada avaliação para a arquitetura.

Tabela 14 – Camadas da Arquitetura de (LIM et al., 2021).

Nome da Camada	Descrição da Camada
Interação	Recebe os dados tratados e convertidos em conhecimento da camada “Computação”. Possui uma interface com o usuário para gerenciamento de interrupções e otimização de processos, ao mesmo tempo em que facilita os processos de reconfiguração/reconfiguração de ativos.
Computação	Possui um modelo de ontologia, que inclui requisitos das partes interessadas, regras de segurança e detalhes da operação, é então preenchido com dados em tempo real ou previstos para criar um gráfico de conhecimento. Ferramentas de análise de dados e aprendizado de máquina permitem obter estatísticas importantes, facilitando a tomada de decisões por parte dos operadores.
Consolidação de Dados	Coleta os dados da camada “Ciber-físico” e os armazena em bancos de dados NoSQL adequados para lidar com grandes quantidades de dados de sensores heterogêneos. Realiza a conversão, indexação, filtragem e armazenamento de dados.
Ciber-físico	Possui um conjunto de sensores que coletam dados em tempo real do layout do ambiente ou dos produtos, gerando um clone virtual.

6.2.11 Arquitetura de (ZOU et al., 2022)

O trabalho (ZOU et al., 2022) apresenta uma arquitetura de DT no contexto de Paisagística contemplando 3 camadas. Não é apresentada avaliação para a arquitetura.

Tabela 15 – Camadas da Arquitetura de (ZOU et al., 2022).

Nome da Camada	Descrição da Camada
Físico	Coleta e análise de dados do ambiente físico
Modelo	Simulação e predição.
Aquisição, Pré-processamento, Processamento e Transmissão de Dados	Monitoramento, feedback e <i>self-cure</i> .

6.2.12 Arquitetura de (GAO et al., 2022)

Os autores de (GAO et al., 2022) apresentaram uma arquitetura de DT de forma genérica, no contexto da Indústria, com 5 camadas. Não é apresentada avaliação para a arquitetura.

Tabela 16 – Camadas da Arquitetura de (GAO et al., 2022).

Nome da Camada	Descrição da Camada
Suporte de fundação	Equipamentos físicos específicos, incluindo equipamentos de construção urbana, equipamentos de construção industrial, etc.
Interação de dados	Aquisição de dados, armazenamento de dados, transmissão de dados e processamento de dados.
Construção de modelos e análise de simulação	Modelagem de dados, análise de simulação de dados e processamento inteligente.
Aplicação comum	Consiste em quatro links: descrição, diagnóstico, previsão e tomada de decisão.
Aplicativo da indústria	Status do aplicativo do ecossistema de gêmeos digitais em vários setores, como cidade inteligente, cérebro digital, fabricação inteligente, saúde inteligente etc.

6.2.13 Arquitetura de (THÜRER; LI; QU, 2022)

Os autores de (THÜRER; LI; QU, 2022) apresentaram uma arquitetura de DT no contexto da Indústria, com foco em Logística de Produção, com 3 camadas. Foi desenvolvido um protótipo da Entidade Física para simular a primeira camada

da arquitetura proposta, entretanto não foram apresentados experimento ou testes realizados com tal protótipo.

Tabela 17 – Camadas da Arquitetura de (THÜRER; LI; QU, 2022).

Nome da Camada	Descrição da Camada
Entidade física	Fábrica de modelos da <i>Fischertechnik</i> que já vem com um número considerável de sensores, que podem capturar todas as ações e operações dos recursos da fábrica de modelos em tempo real.
Camada de comunicação	Transmite os dados para o Arduino Mega 2560, um microprocessador que realiza a comunicação em tempo real entre a Entidade física e o Espaço virtual.
Espaço virtual	Suporta simulação, tomada de decisão e controle da entidade física. A sombra digital é criada nesta camada por um banco de dados em tempo real no MySQL. Se o sistema ficar fora de controle, um chamado serviço é realizado e uma ação é tomada.

6.2.14 Arquitetura de (NWOGU et al., 2022)

Os autores de (NWOGU et al., 2022) apresentaram uma arquitetura de DT em contexto Genérico, com 3 camadas. Não é apresentada avaliação para a arquitetura.

Tabela 18 – Camadas da Arquitetura de (NWOGU et al., 2022).

Nome da Camada	Descrição da Camada
Gêmeo Físico	É o sistema físico ou percebido do mundo real, por exemplo, um sistema de fabricação ou um processo aeroportuário, que é conectado dinamicamente ao DT por meio de um meio de comunicação ou integração.
Meio de Comunicação ou Integração	Usando protocolos de comunicação com a capacidade de fornecer uma comunicação bidirecional entre o Gêmeo Físico e o DT. Um fluxo de dados bidirecional distingue um DT de outras formas digitais, como modelo digital, sombra digital, etc.
Digital Twin	O DT consiste em 7 componentes.

Tabela 19 – Componentes da camada *Digital Twin* da Arquitetura de (NWOGU et al., 2022).

Nome do Componente	Descrição
Aquisição de Dados	Para melhorar a entidade do mundo real que está representando ou refletir efetivamente seu comportamento, um DT deve ter a capacidade de automatizar a aquisição de dados.
Análise de Dados	É o motor de um DT, que aplica várias técnicas de análise para fornecer o mecanismo para gerenciar, fundir e processar os dados amplamente heterogêneos adquiridos pelo Sistema de DT.
Banco de Dados	Engloba o repositório interno ou qualquer forma de mecanismo de armazenamento de dados que aloja o modelo de dados de um DT.
Gerenciador de Cenários	Realiza análises hipotéticas com base no resultado da análise de dados e/ou experimentação de simulação.
Simulação	Pode ser implementado usando eventos discretos, baseados em agentes, dinâmica de sistema (ou seja, simulação contínua) ou simulação híbrida.
Otimização	Graças à capacidade de fundir dados díspares e sincronizar com uma entidade do mundo real, um DT é capaz de otimizar um sistema ao longo de todo o seu ciclo de vida, que inclui seus estágios de projeto, desenvolvimento, operação e retirada.
Controle	Concluídos os ciclos de simulação-otimização, o gerenciador de cenários pode coletar os resultados e convertê-los em um conjunto de instruções para o sistema físico.

6.3 Etapa 3: Comparação das Arquiteturas

A Tabela 20 mostra a comparação das arquiteturas revisadas com a arquitetura proposta neste artigo. Das 14 arquiteturas, 8 foram desenvolvidas com foco na área da indústria, e 8 não apresentaram qualquer tipo de avaliação. Destaca-se que, 2020 foi o ano com maior número de publicações sobre a temática, com 5 estudos.

Tabela 20 – Comparação das Arquiteturas

Arquitetura	Domínio	Camadas	Ano	Tem avaliação?
1 - (CATARCI et al., 2019)	Indústria	4	2019	Não
2 - (REDELINGHUYTS; BASSON; KRUGER, 2020)	Indústria	6	2019	Estudo de caso: Utilizando uma garra robótica como gêmeo físico e um <i>dashboard</i> de DT com leituras dos dados.
3 - (KYCHKIN; NIKOLAEV, 2020)	Indústria	4	2020	Experimento: Com a tarefa de previsão online do consumo de energia para o sistema de ventilação.
4 - (BOTTANI; VIGNALI; TANCREDI, 2020)	Indústria	6	2020	Experimento: Tendência da pressão de entrada para suco de maçã e damasco em função da variação do fluxo, em ambiente com temperatura constante de 25°C.
5 - (AHELEROFF et al., 2021)	Indústria	5	2020	Estudo de caso: Em um sistema de <i>Digital Twin</i> para monitorar e controlar parâmetros climáticos de áreas ambientais alagadas.
6 - (ZHENG et al., 2020)	Manufatura	5	2020	Não
7 - (DALIBOR et al., 2020)	Indústria	5	2020	Não
8 - (MOURTZIS et al., 2021)	Manufatura	3	2021	Estudo de caso: Sistema de Modelagem 3D baseado em gêmeo digital.
9 - (PAN et al., 2021)	Indústria	2	2021	Experimento: Comparando o consumo de memória de dois métodos, PLS e N-PLS.
10 - (LIM et al., 2021)	Genérico	4	2021	Não
11 - (ZOU et al., 2022)	Paisagem	3	2022	Não
12 - (GAO et al., 2022)	Genérico	5	2022	Não
13 - (THÜRER; LI; QU, 2022)	Indústria	3	2022	Estudo de caso: protótipo da Entidade Física para simular a primeira camada da arquitetura proposta.
14 - (NWOGU et al., 2022)	Genérico	3	2022	Não
Proposta	Genérico	3	2022	Estudo de caso e experimento de análise de desempenho.

Nota-se que há duas metodologias de avaliação de arquiteturas: (i) aplicação da arquitetura em um estudo de caso, (ii) realização de experimentos, seja comparando o desempenho de diferentes tecnologias aplicadas à arquitetura, seja testando métodos ou funcionalidades relativas ao estudo de caso. Os experimentos podem ser realizados em uma ou mais camadas da arquitetura.

7 Avaliação Experimental

Este capítulo detalha a Fase 5 da metodologia descrita no Capítulo 3, apresentando os resultados dos experimentos feitos com intuito de validar a proposta de Arquitetura Genérica de DT (Fase 2) e a Linguagem de Design AgroLD (Fase 3).

As validações são apresentadas da seguinte forma: a Seção 7.1 apresenta uma comparação entre AgroLD e DTDL analisando quando espaço em memória secundária suas estruturas ocupam; a Seção 7.2 apresenta as validações da arquitetura, que está dividida na subseções 7.2.2 (na qual são realizadas experimentações que testam de forma integrada as três camadas da arquitetura proposta no Capítulo 4) e 7.2.3 (na qual se testa, de forma isolada, a camada Entidade Virtual da arquitetura proposta); e a Seção 7.3 que apresenta um comparativo entre esta Dissertação e os trabalhos do Capítulo 6.

7.1 Comparativo entre AgroLD e DTDL

É importante destacar que a primeira comparação deste capítulo é feita entre as LDs, ao invés das arquiteturas, pelo fato de que ambas LDs serão utilizadas dentro da Arquitetura Genérica nos cenários experimentais. Dado o exposto, é importante entender previamente as diferenças básicas entre ambas as LDs, para depois correlacionar tais diferenças com os resultados obtidos na Seção 7.2, os quais também são influenciados pelas LDs. Além disso, um fator de importante impacto nos experimentos é a diferença de tamanho (em bytes) entre ambas as LDs. Logo, a seguir será apresentada uma comparação entre o tamanho das estruturas da AgroLD e DTDL.

A quantidade de bytes que cada LD ocupa em memória reflete diretamente no tamanho de sua estrutura, e, portanto, no grau de simplicidade e objetividade de cada linguagem. Quanto mais objetiva for a linguagem menor será o tamanho dos modelos, o que permite um processamento mais rápido e em tempo hábil, o que se trata de um importante requisito para as LDs (GRIEVES, 2015) (WRIGHT; DAVIDSON, 2020).

A Tabela 21 apresenta um resumo do consumo de dados das linguagens AgroLD e DTDL. Destaca-se que esse consumo de dados é referente à quantidade de bytes que cada linguagem ocupa dentro do componente Serviços, localizado na camada Entidade Virtual (Figura 19). Esse consumo ocorre na memória secundária, isto é, em memórias não-voláteis, como HD, SSD e afins.

Conforme mostra a tabela, para a definição da cada linguagem foram consumidos 5.414 bytes na AgroLD e 2094 bytes na DTDL. A definição da linguagem compreende todas as estruturas das linguagens aplicadas ao *Plante Box*, descrevendo todos os seus recursos, os quais foram apresentados na Seção 5.2.

A terceira linha da Tabela 21 apresenta o consumo de memória para descrever os valores padrão dos componentes CPU, battery, memory e communication. O

Tabela 21 – Consumo de dados da LD e da DTDL

	LD	DTDL
Espaço em memória ocupado pelas estruturas (Figuras 20 e 21) dentro do componente <i>Digital Twin</i> (Figura 19)	5,414 KB	20,694 KB
Definição dos valores padrão da CPU, bateria, memória e comunicação	0 B	1,241 KB
Telemetria de requisição	105 B	178 B
Telemetria de resposta	230 B	569 B
TOTAL	5.749 KB	2282 KB

Fonte: Autor.

grande detalhe é que a AgroLD descreve esses valores dentro de sua própria estrutura (os quais podem ser alterados posteriormente via troca de mensagens entre EV e EF), entretanto a DTDL determina que esses valores sejam definidos via mensagem, enviada da EF (ou de um componente intermediário) para a EV, para, só então, realizar a atribuição dos valores. Em outras palavras, a AgroLD é capaz de atribuir valores padrão, já a DTDL necessita de uma mensagem inicial para, só então, definir os valores. Destaca-se também que ambas as linguagens podem ter seus valores atualizados ao longo do tempo através da troca de mensagens, as quais são detalhadas no Apêndice C.

Essa necessidade de atualização dos valores ao longo do tempo ocorre para que o estado da EV fique sincronizado com o estado da EF. Ao planejar a implementação da AgroLD foi detectado que os componentes CPU, battery, memory e communication não teriam seus valores atualizados com frequência, durante a vida útil da EF, pois esses são componentes, em geral, soldados no *hardware* da EF. Por exemplo, o *hardware* principal do Plante Box (que é o NodeMCU ESP 32s) não permite que esses componentes sejam mudados. Para mudar tais componentes seria necessário comprar outro NodeMCU. Após tal análise, decidiu-se que CPU, battery, memory e communication teriam seus valores definidos por padrão dentro da própria estrutura da AgroLD, sem a necessidade de uma troca de mensagens inicial entre a EV e EF. Esse planejamento permitiu que a AgroLD consumisse zero bytes para a definição dos valores padrão, enquanto a DTDL consome 1,241 KB.

A quarta linha da Tabela 21 mostra o consumo de dados com a telemetria de requisição (definida no Apêndice D). Essa telemetria corresponde às mensagens que podem ser enviadas da Entidade Virtual à Entidade Física, requisitando a execução de uma função, por exemplo. Para contabilizar o tamanho em bytes são levadas em conta todas as possibilidades de requisições definidas pelas funções. A AgroLD consome 105 bytes, enquanto a DTDL consome 178 bytes. Isso deve-se principalmente ao recurso de *alias*, que a AgroLD possui e a DTDL não. Conforme explicado no

Capítulo 5 o *alias* serve para criar apelidos para nomes de funções ou atributos, reduzindo o tamanho de bytes enviados através da rede.

A última linha da Tabela 21 mostra o consumo com a telemetria de resposta (definida no Apêndice E). Essa telemetria corresponde às mensagens que devem ser enviadas da Entidade Física para a Entidade Virtual, como resposta ou confirmação de execução de uma função. Enquanto a AgroLD consome 230 bytes, a DTDL consome 569 bytes. Essa diferença também deve-se ao fato do *alias* que a AgroLD possui.

7.2 Avaliação da Arquitetura Genérica

Como forma de avaliação da arquitetura proposta no Capítulo 4, foram realizados experimentos que visam testar, comparar e analisar a integração da arquitetura com (i) duas linguagens de design (AgroLD e DTDL) e (ii) dois protocolos da camada de transmissão de IoT, o *Message Queuing Telemetry Transport* (Seção 2.1.1.1) e *Constrained Application Protocol* (Seção 2.1.1.2). Para tanto foi montado um *testbed* real, baseado na arquitetura proposta, descrito a seguir. Além disso, foram testados dois tipos de cenários, um que testa a integração das três camadas da arquitetura, e um que analisa a Entidade Virtual de forma isolada. O objetivo de testar a arquitetura de diferentes formas é apresentar uma avaliação arquitetural mais robusta que os trabalhos apresentados no Capítulo 6, já que eles faziam, em geral, apenas um tipo de testagem.

7.2.1 Detalhamento do *testbed* experimental

Conforme mostra a Figura 22, o *testbed* é baseado na Arquitetura Genérica aplicada ao Plante, e é dividido em algumas etapas: a Etapa 1 consiste no envio dos dados coletados pelos sensores para o ESP 32; na Etapa 2 o *firmware* faz a leitura dos dados analógicos, convertendo-os em digitais; na Etapa 3 o *firmware* envia os dados ao cliente (MQTT ou CoAP), e na Etapa 4 o Cliente os envia para o Roteador Wi-Fi, que está na camada de Rede e Conexão. Para que os dados cheguem ao Cliente MQTT (no componente Serviços da camada Entidade Virtual) são necessárias as Etapas 5 e 6, nas quais esses dados passam, respectivamente, para o Broker MQTT, e depois para o Cliente MQTT. Já para o CoAP é necessária apenas a Etapa 5, isto porque o CoAP é um protocolo um-para-um e não conta com um *Broker* (BORMANN, 2016). Na última etapa, a mensagem é enviada para a Interface de Programação de Aplicações (API), que possui um serviço de geração de alertas, o qual avalia e informa se as condições ambientais da lavoura estão adequadas para o tipo de planta cultivada. Para decodificar os dados a API faz uma consulta à camada Linguagem de Design e detecta o formato (AgroLD ou DTDL) da mensagem, para então fazer a checagem e gerar os alertas.

O *testbed* em questão utiliza dois notebooks e um roteador Wi-Fi. O notebook 1 foi utilizado para implementar a camada Entidade Física, contando com um processador Intel Core i7-5500U, 1TB de HD e 8GB de memória RAM; o notebook 2 foi usado para representar a camada Entidade Virtual, possuindo um processador

Intel Core i5-11400H, 500GB de SSD e 24GB de memória RAM; enquanto Roteador Wi-Fi, que representa a camada Rede e Conexão, possui suporte aos padrões 802.11n, 802.11g e 802.11b, frequência única de 2.4 GHz (até 300 Mbps) e 2 antenas externas de 5dBi. Os notebooks tinham suporte ao padrão 802.11n, logo, foi o padrão usado nos experimentos. Destaca-se também que as camadas Entidade Física e Entidade Virtual, bem como seus componentes, foram implementados em NodeJS (JavaScript), respectivamente, nos notebooks 1 e 2. Ou seja, a arquitetura foi de fato implementada, e não meramente simulada, como demonstrado em <https://github.com/RenatoAlves0/mestrado>.

Destaca-se que o poder de processamento do Intel Core i7-5500U (4 MB de cache, até 2,40 GHz) é inferior ao Intel Core i5-11400H (12 MB de cache, até 4,50 GHz). Foi escolhido o notebook com maior poder de processamento e maior volume de memória RAM para representar a Entidade Virtual porque em cenários reais a EV (que implementa diversos serviços e usa diversas APIs) precisa ser mais potente que a EF (que normalmente se limita a funções de leitura de sensores e manipulação de atuadores). Além disso, em cenários reais, como um campo agrícola automatizado, por exemplo, o que ocorre é a existência de uma EV (servidor, APIs) que gerencia várias EFs (sensores, atuadores).

É importante deixar claro que os experimentos não foram executados através de *software* automatizado ou simulador. Destaca-se que para cada ponto dos gráficos apresentados na Seção a seguir, o experimento foi executado três vezes, objetivando obter o intervalo de confiança. Cada execução de experimento foi inicializada de forma manual, seguindo este passo-a-passo: (i) abrir terminal NodeJs, (ii) executar comando de inicialização do *software* do experimento, (iii) aguardar o *software* finalizar a execução, (iv) copiar os dados resultantes da execução e colar numa planilha, (v) copiar os dados da planilha e plotar nos gráficos.

7.2.2 Cenário 1 - Integração entre as 3 Camadas da Arquitetura

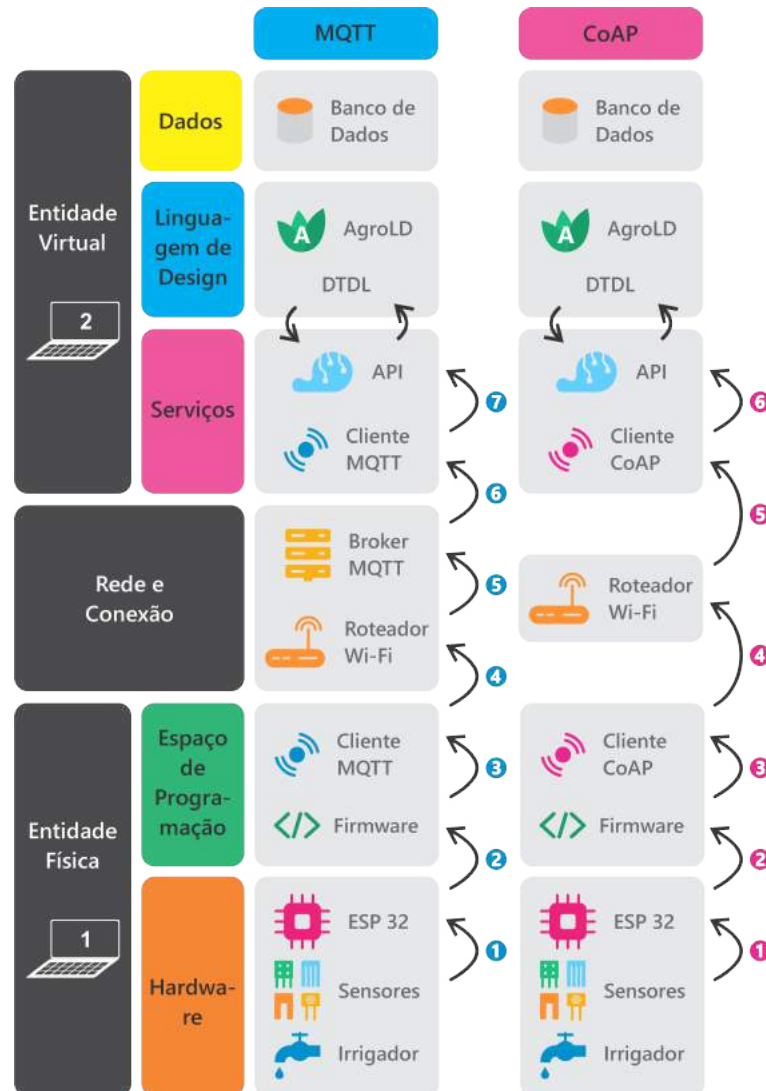
O primeiro grupo de experimentos visa testar a **integração de todas as camadas** da arquitetura, bem como analisar uma das principais funcionalidades do Plante, o monitoramento em tempo real de lavouras, através da coleta e envio de dados ambientais da Entidade Física para a Entidade Virtual. Destaca-se que as mensagens enviadas são formatadas de acordo com o que está definido no Apêndice E, e cada mensagem enviada continha o equivalente a 15 leituras de dados ambientais. A seguir são apresentados dois cenários de experimentação diferentes.

7.2.2.1 Cenário Não Saturado (variando o tempo de envio de mensagens)

Primeiro, objetivou-se analisar a arquitetura integrada a um cenário não saturado, e entender como as linguagens de design e os protocolos de IoT se comportam em cenários pouco congestionados.

Para tal, o cenário conta com uma única Entidade Física enviando dados ambientais para a EV. Nesse cenário a EF tem à sua disposição 0.01, 0.1, 1, 2, 3, 4, 5, 6, 7 e 8 segundos para enviar quantas mensagens conseguir para a EV. Isto é, no

Figura 22 – Testbed utilizado para avaliação da Arquitetura Genérica.



Fonte: Autor.

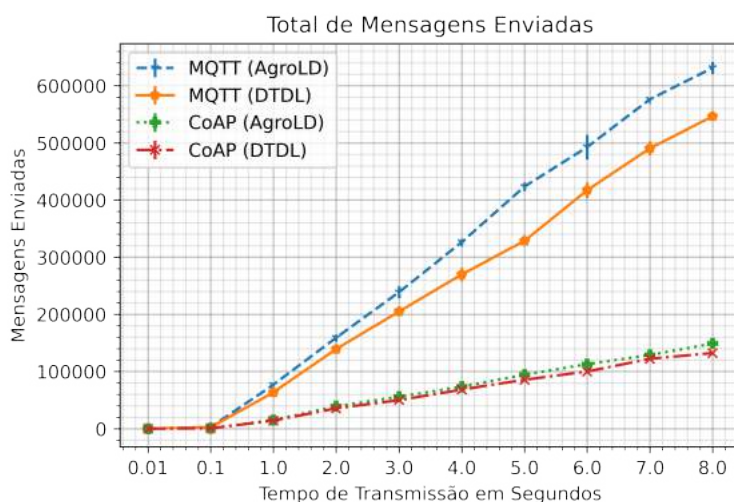
cenário de 1 segundo, a EF fica ativa durante um segundo, enviando mensagens constantemente para a EV, ao final desse tempo, a EF é automaticamente desativada.

A execução desses experimentos resultou nos resultados a seguir.

O primeiro aspecto analisado (Figura 23) foi a quantidade de mensagens que a EF conseguiu enviar. O esperado era que os cenários com uso da AgroLD conseguissem enviar mais mensagens, já que possui uma estrutura mais simplificada que a DTDL. E, de fato, foi isso que ocorreu nos experimentos. Nota-se que quanto mais tempo a EF fica ativa (eixo x), maior é a diferença entre a quantidade de mensagens enviadas com base em AgroLD e DTDL para ambos protocolos de comunicação. Em se tratando dos protocolos IoT, o MQTT conseguiu enviar mais mensagens que o CoAP, demonstrando ser o protocolo mais eficiente nesse cenário. O principal motivo dessa superioridade do MQTT é que ele usa o TCP (um protocolo orientado à conexão) e o CoAP usa o UDP (protocolo sem conexão). Na prática, ao analisar os códigos

implementados em JavaScript¹, o CoAP necessita (i) criar/inicializar um objeto de requisição POST (para envio de mensagens), (ii) enviar a requisição, e (iii) encerrar a requisição e destruir o objeto. Esse processo é feito para cada mensagem enviada. Já no caso do MQTT, o processo é mais simples: (i) abrir/iniciar uma conexão com o Broker, (ii) enviar todas as mensagens que puder, e (iii) encerrar a conexão. Note que o MQTT não precisa realizar o processo de criar/encerrar uma requisição para cada mensagem enviada, um processo similar a isso é feito apenas na primeira e na última mensagem enviadas. Destaca-se também que a pouca diferença entre AgroLD e DTDL nos experimentos com CoAP deve-se a esse processo dispendioso no envio de cada mensagem, o que acaba ofuscando, parcialmente, a diferença em bytes (que não é tão grande²) entre as mensagens formatadas com base na AgroLD e DTDL.

Figura 23 – Mensagens Enviadas em função do Tempo de Transmissão.



Fonte: Autor.

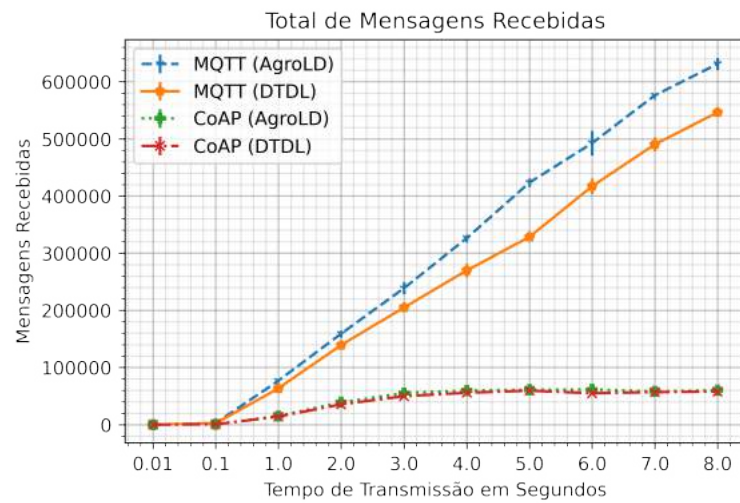
O segundo aspecto analisado (Figura 24) foi a quantidade de mensagens que a EV recebeu com sucesso. No caso do MQTT (com AgroLD e DTDL) foram recebidas com sucesso todas as mensagens enviadas. Já em se tratando do CoAP, a partir dos 4 segundos o protocolo começou a perder pacotes. Nota-se que de 4 a 8 segundos o CoAP recebeu sempre uma quantidade constante de mensagens. Isso se deve ao fato de o cliente CoAP (no componente Serviços) ter saturado, atingindo seu limite máximo de consumo de recursos computacionais nesse cenário. Isso deve-se à presença do Broker nos cenários baseados em MQTT, o qual pré-processa parte dos dados e gerencia o processo de recebimento/distribuição de mensagens e conexão de novos clientes. Isso faz com que o Cliente MQTT fique menos sobrecarregado que o Cliente CoAP, o qual não conta com um Broker ou nó intermediário similar.

No quesito consumo de CPU, conforme mostra a Figura 25, o cenário com MQTT alocou a CPU por mais tempo que o CoAP, uma vez que o MQTT conseguiu entregar mais mensagens, o que requisitou maior tempo de processamento. O mesmo ocorreu com a AgroLD, que consumiu mais CPU que a DTDL, uma vez que

¹ <https://github.com/RenatoAlves0/mestrado/tree/main/physical-entity/programing-space/clients>

² A diferença é de 339 Bytes a menos na AgroLD, na telemetria de resposta

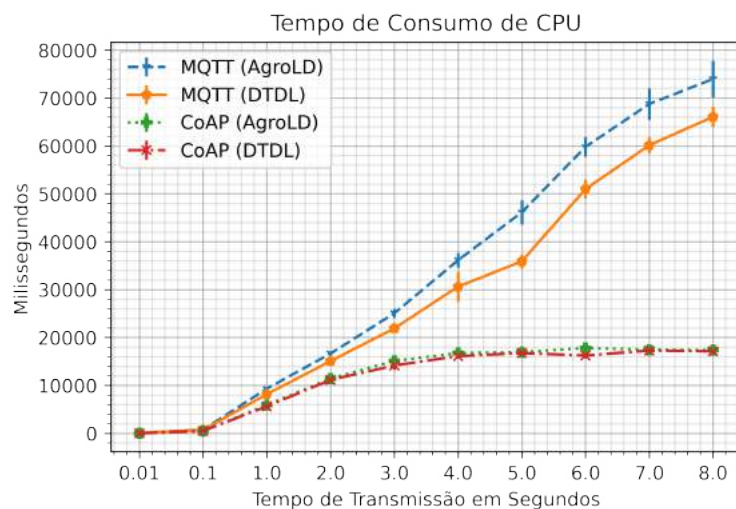
Figura 24 – Mensagens Recebidas em função do Tempo de Transmissão.



Fonte: Autor.

conseguiu enviar mais mensagens. Para obter o tempo de consumo de CPU em milissegundos foi utilizado o método `process.cpuUsage()` do NodeJs.

Figura 25 – Tempo de Consumo de CPU em função do Tempo de Transmissão.

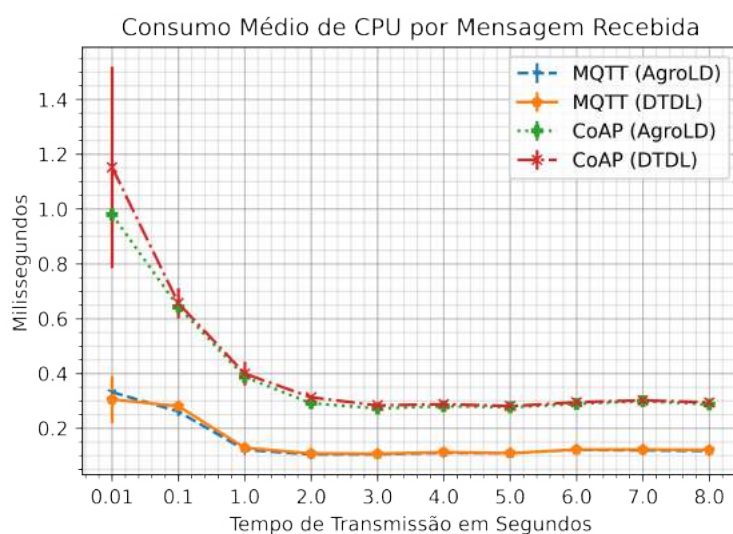


Fonte: Autor.

A Figura 26 mostra o tempo de consumo (em milissegundos) médio por mensagem recebida. O objetivo de mostrar o consumo médio, é entender o consumo por mensagem recebida, ao invés do consumo total. Essa abordagem mostra que cada mensagem recebida, baseada na AgroLD consome menos recursos que cada mensagem recebida baseada na DTDl. Como houve pouca diferença gráfica entre AgroLD e DTDl foi utilizada a seguinte fórmula $((agro/dtdl) * 100) - 100$ para obter a diferença percentual entre os valores. No cenário 0.01 segundo no CoAP, a DTDl foi 8.04% mais econômica, já nos demais cenários a AgroLD foi, respectivamente, 7.59%, 6.02%, 3.34%, 1.81%, 2.17%, 0.37%, 06%, 28% e 3.33% mais econômica.

No caso do MQTT, a AgroLD foi mais econômica em todos os cenários: obtendo 172%, 2.05%, 3.23%, 7.30%, 4.08%, 2.78%, 1.41%, 2.18%, 1.00% e 2.58% a menos de tempo de alocação de CPU. Esses resultados devem-se ao fato de as mensagens baseadas em AgroLD conseguirem ter sido enviadas em maior quantidade para a EV. Em se tratando dos protocolos IoT, o MQTT foi mais econômico que o CoAP, já que conseguiu enviar mais mensagens com sucesso.

Figura 26 – Tempo Médio de Consumo de CPU (por Mensagem Recebida) em função do Tempo de Transmissão.



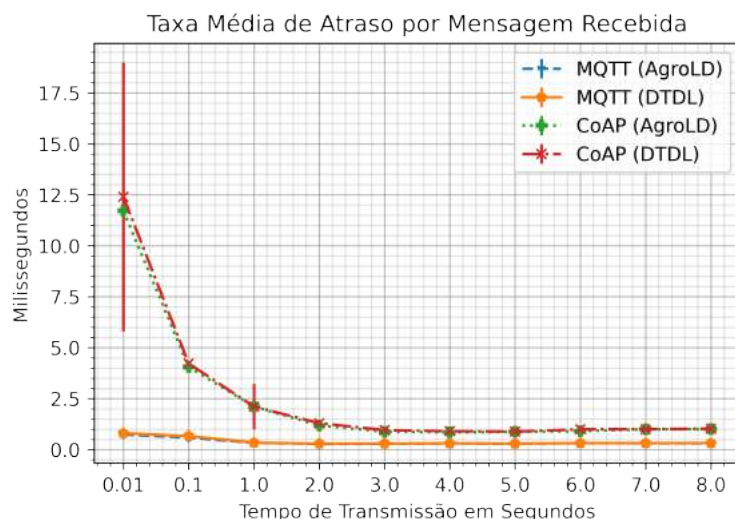
Fonte: Autor.

A Figura 27 mostra quantos milissegundos, em média, cada mensagem demorou para sair da EF e chegar na EV. Assim como na Figura 26, há pouca diferença visual entre as Linguagens de Design, havendo uma diferença mais acentuada nos protocolos IoT, onde o MQTT se sai melhor. Apesar da pouca diferença visual entre AgroLD e DTDL, as mensagens da AgroLD chegaram mais rápido ao destino em todos os cenários. A vantagem percentual da AgroLD sobre a DTDL foi a seguinte: (i) no cenário com MQTT 8.23%, 12.48%, 3.04%, 4.58%, 3.49%, 3.73%, 3.83%, 1.45%, 3.08% e 3.13%, e (ii) no cenário com CoAP 5.83%, 4.20%, 0.07%, 9.54%, 8.80%, 6.41%, 1.44%, 9.95%, 0.41% e 10%.

7.2.2.2 Cenário Saturado (variando a quantidade de Entidades Físicas)

Nesta segunda parte de experimentos, objetivou-se analisar o comportamento da arquitetura quando aplicada a um cenário mais saturado em relação ao cenário anterior. Para tal, foi montado um cenário com 10, 100, 200, 300, 400 e 500 Entidades Físicas (EF), de forma que cada EF envie, simultaneamente, 1000 mensagens para a Entidade Virtual (EV). A escolha de realizar este experimento com várias EFs, ao invés de saturar mais o cenário com uma EF, foi baseada no fato de que: no dia-a-dia de aplicação como Plante (estudo de caso) é natural que existam várias EFs na lavoura, comunicando-se simultaneamente com a EV. Logo, este cenário,

Figura 27 – Tempo Médio de Entrega (por Mensagem Recebida) em função do Tempo de Transmissão.



Fonte: Autor.

aproxima-se mais da realidade que o cenário anterior. Destaca-se que todas as EF foram instanciadas dentro do Notebook 1.

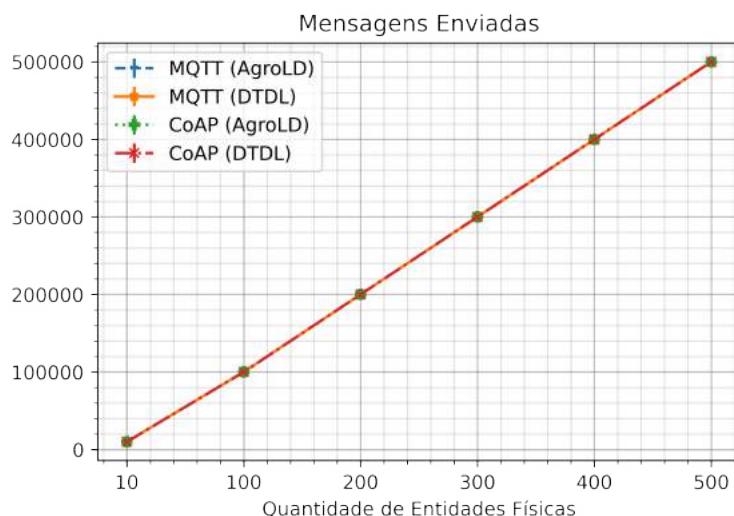
Outro fato importante a se destacar para este cenário saturado é que ao realizar a primeira leva de experimentos não foi possível saturar o MQTT (reduzir a taxa de entrega para próximo de zero) na EV, apenas o CoAP. Em busca de saturar ambos protocolos, e não apenas o CoAP, foi testado um cenário com 1000 Entidades Físicas, o que aumentaria a quantidade de mensagens enviadas e poderia saturar os dois protocolos. Entretanto, neste cenário com 1000 EFs o Notebook 1 travou (por consumo excessivo de CPU e memória), impossibilitando a realização dos experimentos. Para contornar essa situação, e apresentar um cenário experimental no qual fosse possível saturar os dois protocolos, o NodeJs do MQTT e CoAP foram configurados, no Notebook 2, para consumir no máximo 40MBs de memória *Heap*³, o que possibilitou que ambos protocolos fossem saturados sem travar o Notebook 1.

A Figura 28 mostra a quantidade de mensagens enviadas em função da quantidade de EFs. Como é possível notar, todos os protocolos conseguiram enviar todas as mensagens, em igual quantidade. Isso ocorreu, porque, diferente do cenário anterior (não saturado), este cenário não tem limitação de tempo, o *software* apenas informa: "Instancie 100 Entidades Físicas, cada uma enviando 1000 mensagens", e o protocolo leva o tempo que for necessário para enviar todas as mensagens.

A Figura 29 mostra a quantidade de mensagens recebidas na EV em função da quantidade de EFs. Nota-se que o MQTT conseguiu enviar mais mensagens que o CoAP, o motivo é que o MQTT geriu de forma mais eficiente o consumo de memória *Heap*, e conseguiu ficar ativo (recebendo mensagens) por mais tempo. Parte dessa melhor gestão de memória deve-se ao fato de o MQTT possuir um *Broker*, o que faz

³ Memória *heap* é o local onde o NodeJs armazena, de forma não ordenada, informações do código, como variáveis, objetos e dados

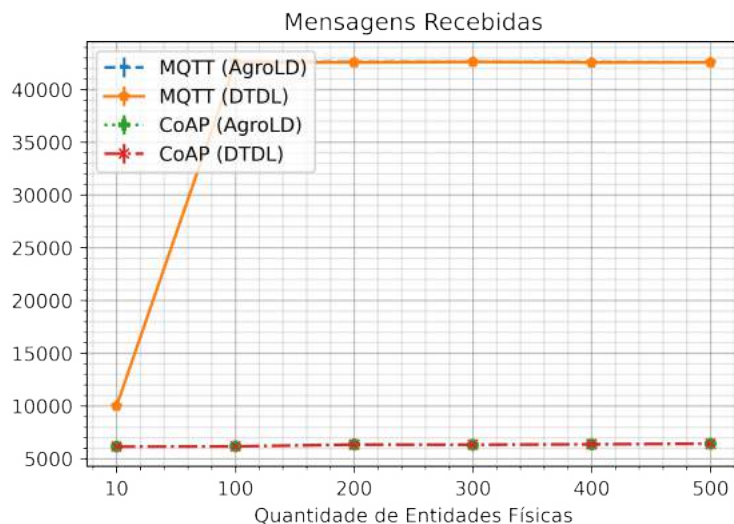
Figura 28 – Mensagens Enviadas em função da Quantidade de Entidades Físicas.



Fonte: Autor.

com que a carga de processamento não fique toda concentrada apenas no Cliente, o mesmo não ocorre com o CoAP, uma vez que ele não possui um *Broker* ou algo similar.

Figura 29 – Mensagens Recebidas em função da Quantidade de Entidades Físicas.

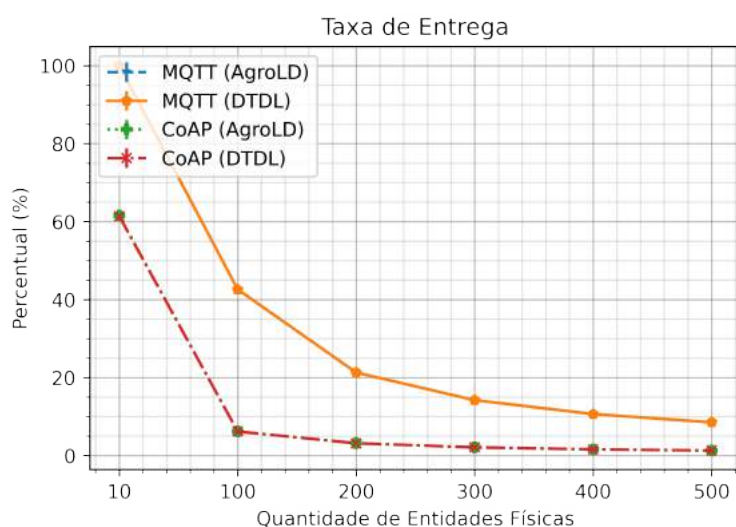


Fonte: Autor.

A Figura 30 mostra a taxa de entrega em função da quantidade de EFs. Destaca-se que, no cenário anterior não foi incluído um gráfico com taxa de entrega, porque o MQTT obteve 100% de entrega em todos os pontos, uma vez que cenário não estava saturado o suficiente. De imediato, pode-se perceber que todas as curvas caem quando o número de EFs aumenta, já que a quantidade de mensagens enviadas aumenta, e a quantidade de mensagens recebidas diminui, devido ao aumento de saturação. Nota-se também que o MQTT conseguiu uma maior taxa de entrega em

todos os cenários, isso demonstra que em cenários mais saturados a presença do Broker ajuda a manter o cliente estável; uma vez que o CoAP não conta com essa tecnologia, seu cliente acaba ficando sobrecarregado. Destaca-se que esse mesmo comportamento também foi visto no cenário anterior. O cliente (CoAP ou MQTT) em questão é o que está na EV. Destaca-se também que a perda de pacotes deste cenário ocorreu nos clientes CoAP/MQTT da camada Entidade Virtual. Quanto às LDs, no cenário com MQTT, a AgroLD obteve uma pequena vantagem percentual, sobre a DTDL, a partir do cenário com 100 EF: entregando 0.08%, 0.11%, 0.03%, 0.09% e 0.01% a mais de mensagens. Já no cenário com CoAP, a vantagem da AgroLD foi de 0.24%, 0.05%, 0.32%, 0.10%, 0.13% e 0.47%.

Figura 30 – Taxa de Entrega em função da Quantidade de Entidades Físicas.



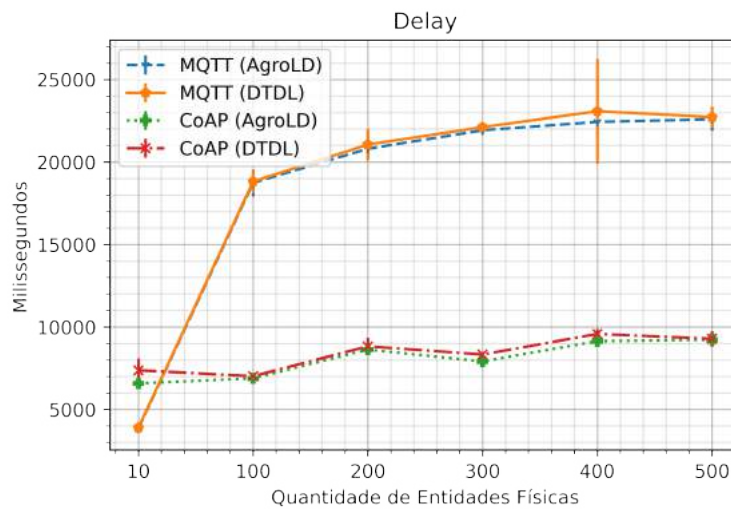
Fonte: Autor.

A Figura 31 mostra o tempo decorrido desde a saída das mensagens da EF até sua chegada à EV. Nota-se em todos os cenários que a AgroLD demorou menos tempo para chegar ao destino, isso deve-se ao fato de o tamanho das mensagens baseadas em AgroLD serem menores, uma vez que sua estrutura consome menos bytes devido ao recurso de *alias*. Nota-se também que na maior parte dos cenários o CoAP obteve um menor tempo, isso deve-se ao fato de que ele conseguiu enviar menos mensagens que o MQTT em tais cenários.

No cenário com MQTT, DTDL e 400 EFs nota-se que o intervalo de confiança ficou maior que os demais, Uma hipótese para isso é que um dos notebooks estava acessando a rede de forma mais intensa que o normal, possivelmente baixando alguma atualização de *software* ou atualizando dados de clima, por exemplo.

Analisando o consumo de CPU, na Figura 32, nota-se que em todos os cenários baseados em AgroLD houve uma leve vantagem no consumo em comparação à DTDL. Isso deve-se ao fato da estrutura com menor tamanho em bytes da AgroLD. Já em se tratando dos protocolos IoT, no cenário menos saturado (10 EFs) o MQTT foi mais econômico que o CoAP (é importante lembrar que nesse cenário houve uma diferença de aproximadamente 5000 mensagens a mais recebidas pelo MQTT, e mesmo assim

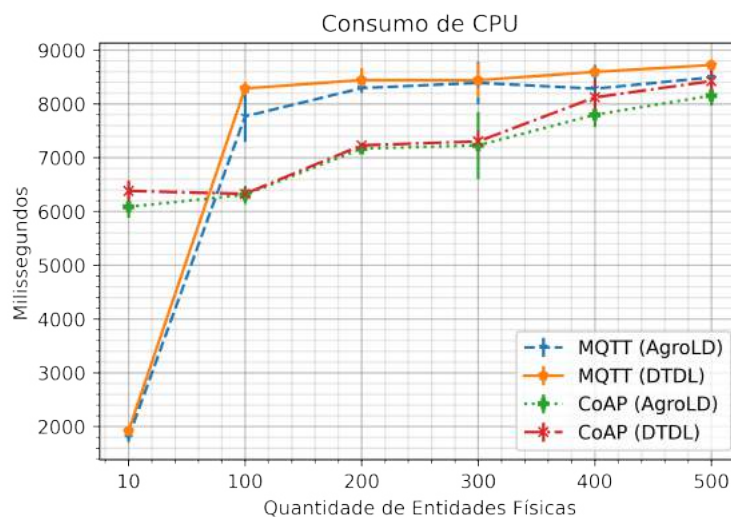
Figura 31 – Tempo de Entrega em função da Quantidade de Entidades Físicas.



Fonte: Autor.

ele conseguiu ser mais econômico); já nos demais cenários o CoAP apresentou um menor consumo (isso porque ele recebeu em torno de 37000 mensagens a menos que o MQTT, logo, já era esperado que ele fosse mais econômico, uma vez que tinha menos mensagens para processar).

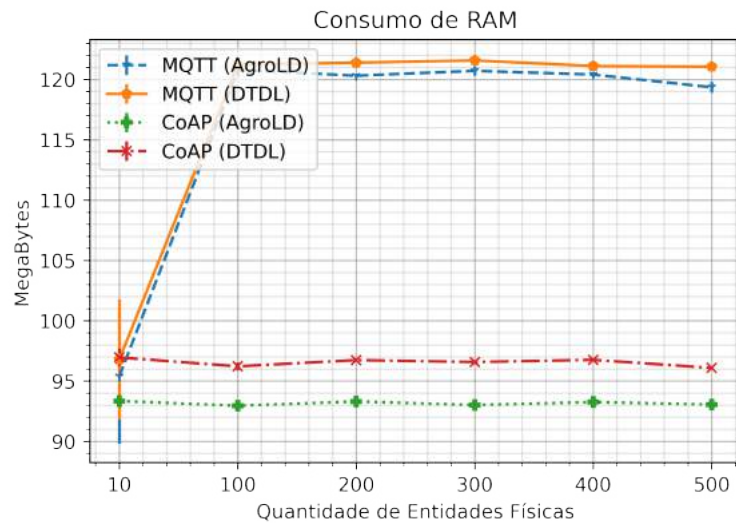
Figura 32 – Consumo de CPU em função da Quantidade de Entidades Físicas



Fonte: Autor.

Por último, a respeito do consumo de RAM (Figura 33) percebe-se que o CoAP consumiu menos memória que o MQTT, uma vez que também recebeu menos mensagens. Além disso, a AgroLD se mostrou mais econômica que a DTDL, uma vez que possui uma estrutura mais simplificada, objetiva e eficiente.

Figura 33 – Consumo de RAM em função da Quantidade de Entidades Físicas



Fonte: Autor.

7.2.3 Cenário 2 - Camada Entidade Virtual

Diferentemente dos demais experimentos até então apresentados, este cenário visa analisar/experimentar **isoladamente a camada Entidade Virtual** da arquitetura, analisando como esta camada se comporta ao *interpretar*⁴ todas as estruturas das linguagens de design. Em um ambiente real, essa interpretação (ou leitura, processamento, inicialização, carregamento em memória, que são termos similares) é feita sempre que o componente Serviços precisa acessar o componente Linguagem de Design.

7.2.3.1 Consumo de Recursos Computacionais na Entidade Virtual ao Interpretar as Estruturas das Linguagens de Design

Como exemplo é possível citar o seguinte cenário: imagine que há dois dispositivos de IoT (Entidades Físicas) em uma lavoura monitorando seus recursos ambientais. Às 10:20 um desses dispositivos envia os dados coletados à EV, e às 10:30 o outro dispositivo também envia os dados à EV. Isso significa que o componente Serviços precisou fazer duas consultas (às 10:20 e às 10:30) ao componente Linguagem de Design, e assim, fazer a interpretação de suas estruturas.

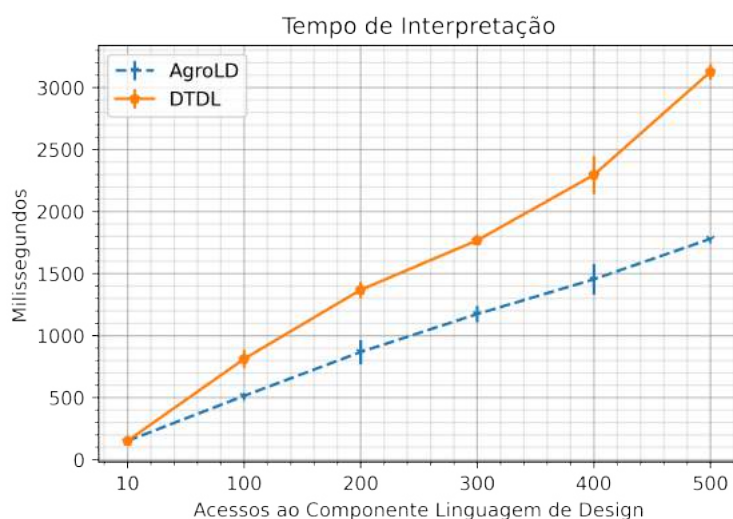
Para tal, foi montado um cenário no qual sejam necessários 10, 100, 200, 300, 400 e 500 acessos ao componente Linguagem de Design para interpretação de suas estruturas. Destaca-se que, já que o objetivo é analisar de forma isolada a EV, então foi criado um *script* (<https://github.com/RenatoAlves0/mestrado/tree/main/virtual-entity/services/api/initialization>) dentro da própria EV, que solicita/requisita esses acessos à LD, caso contrário, não seria possível testar isoladamente esta camada.

⁴ Como a arquitetura foi implementada em NodeJS (JavaScript) utiliza-se o termo "interpretar" ao invés de "compilar", que é o utilizado no contexto de Java, C# e algumas outras linguagens de programação. Diferente dessas linguagens, o JavaScript não necessita transformar o código em um *Machine Code*.

Os resultados a seguir mostram o consumo de recursos computacionais por parte da Entidade Virtual, componente Serviços, para fazer essas interpretações da AgroLD e da DTDL.

A Figura 34 mostra o tempo total que o algoritmo de interpretação leva para iniciar e finalizar a execução. O algoritmo em questão consulta as estruturas do componente LD e as carrega em memória dentro do componente Serviços. Notadamente, nos cenários com AgroLD o algoritmo necessitou de menos tempo para finalizar com sucesso sua execução. No cenário mais saturado (500 acessos) a AgroLD demorou 1800ms contra 3200ms da DTDL. Isso é resultado da diferença de tamanho entre as estruturas, como mostra a Tabela 22 a DTDL possui 16,933 KBs a mais que a AgroLD.

Figura 34 – Tempo para a Interpretação da AgroLD e DTDL em função da quantidade de Acessos ao Componente Linguagem de Design



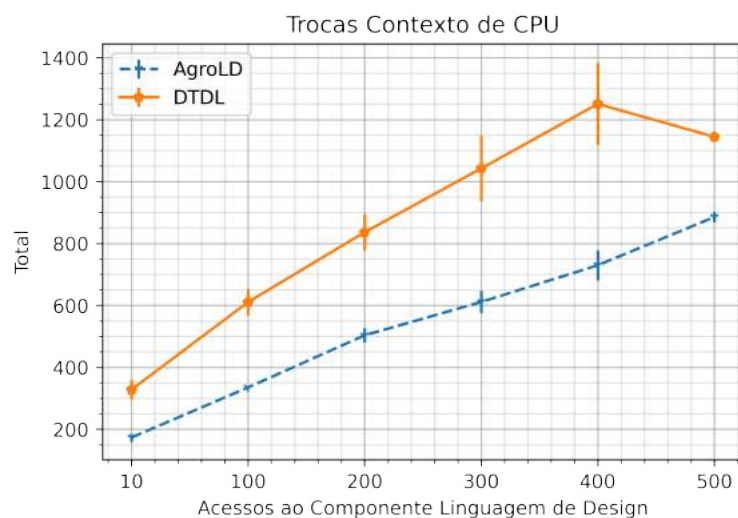
Fonte: Autor.

A troca de contexto é o processo de armazenar e restaurar o estado (contexto) de uma CPU de forma que múltiplos processos possam compartilhar uma única instância de CPU. A Figura 35 exibe a quantidade de trocas de contexto da AgroLD e DTDL. Como nos cenários com AgroLD o algoritmo levou menos tempo de execução, é natural que ele troque de contexto menos vezes. Enquanto o cenário com AgroLD variou entre 180 e 900 trocas, o cenário com DTDL variou entre 350 e 1250 trocas. Já era esperado que uma estrutura com 5.749 KBs seja inicializada em menos tempo que a estrutura da DTDL, com 2282 KBs.

A Figura 36 mostra o tempo que a interpretação das estruturas alocou a CPU, desconsiderando o tempo das trocas de contexto (ou seja, o momento no qual ficou em espera), e considerando apenas o tempo real de uso da CPU. No cenário mais saturado a AgroLD alocou a CPU por 1600ms, enquanto a DTDL a utilizou por 2500ms.

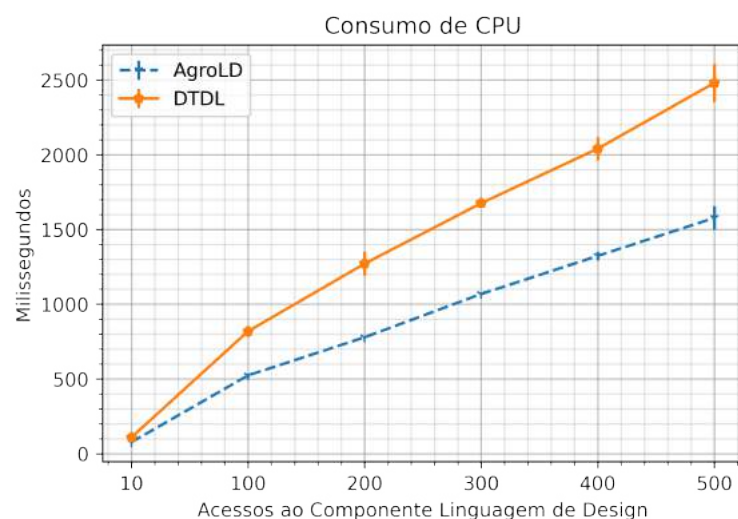
No quesito consumo de memória RAM, a Figura 37 mostra quantos MBs o algoritmo consumiu. Diferentemente dos parâmetros anteriores, o consumo de

Figura 35 – Trocas de Contexto na CPU para a Interpretação da AgroLD e DTDL em função da quantidade de Acessos ao Componente Linguagem de Design



Fonte: Autor.

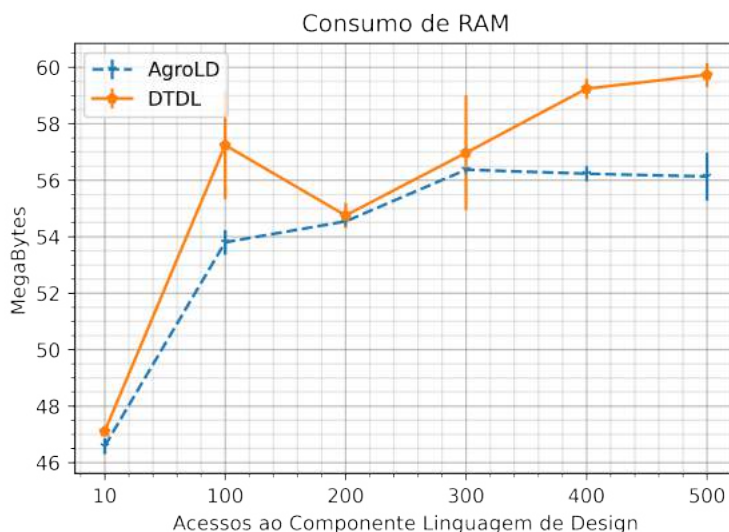
Figura 36 – Tempo de Consumo de CPU para a Interpretação da AgroLD e DTDL em função da quantidade de Acessos ao Componente Linguagem de Design



Fonte: Autor.

RAM não apresentou uma diferença acentuada (embora todos os cenários tenham mostrado um consumo de memória inferior na AgroLD). Isso demonstra que, para este cenário, o tamanho das estruturas das linguagens de design não influencia tanto no consumo de memória RAM total do algoritmo. O consumo da AgroLD variou de 46.5 a 56.5 MB, enquanto a DTDL variou de 47.1 a 59.5 MB.

Figura 37 – Consumo de RAM para a Interpretação da AgroLD e DTDL em função da quantidade de Acessos ao Componente Linguagem de Design



Fonte: Autor.

7.3 Comparação das Propostas de Linguagem de Design e Arquitetura Genérica com Trabalhos Relacionados

Esta seção visa comparar, resumidamente, (i) as vantagens da AgroLD sobre a DTDL, e (ii) as vantagens deste trabalho (no que diz respeito à arquitetura genérica) em relação aos trabalhos apresentados no Capítulo 6.

7.3.1 Linguagem de Design

Como demonstrado na Tabela 22 e ao longo deste trabalho, a AgroLD tem algumas vantagens em relação à DTDL, quando aplicada ao estudo de caso do Plante. Algumas delas são: (i) menor tamanho das estruturas, uma vez que a AgroLD não necessita de interfaces, (ii) o fato de os valores padrão já serem definidas dentro da própria estrutura, não sendo necessário inicializar tais valores posteriormente, (iii) tempo de interpretação (inicialização, leitura, carregamento em memória) menor, uma vez que seu tamanho em bytes é menor, (iv) menor consumo de RAM e CPU no momento da interpretação, (v) *payload* menor, devido ao fato de a AgroLD possuir o recurso de *alias*, sendo possível criar apelidos (pequenos) para nomes de funções e atributos.

7.3.2 Arquitetura Genérica

A Tabela 23 compara 7 características (apresentadas na tabela como C1, C2 etc) utilizadas no desenvolvimento da proposta e avaliação da arquitetura de *Digital Twin*, dos 14 trabalhos (apresentados no Capítulo 6) e desta Dissertação. Logo abaixo da tabela essas características são detalhadas e discutidas.

Tabela 22 – Comparação Resumida entre AgroLD e DTDL

	DTD	AgroLD
Menor tamanho (em bytes) das estruturas		X
Não necessita de interfaces		X
Já define os valores padrão na própria estrutura		X
Tempo de interpretação menor		X
Consome menos RAM e CPU na interpretação		X
<i>Payload</i> menor		X
<i>Alias</i> de funções		X

Fonte: Autor.

Tabela 23 – Comparação da Proposta e Validação da Arquitetura Genérica com Trabalhos Relacionados do Capítulo 6

Trabalho	C1	C2	c3	C4	C5	C6	C7
1 - (CATARCI et al., 2019)		X					
2 - (REDELINGHUY; BASSON; KRUGER, 2020)		X	X	X			X
3 - (KYCHKIN; NIKOLAEV, 2020)		X	X				
4 - (BOTTANI; VIGNALI; TANCREDI, 2020)		X	X				
5 - (AHELEROFF et al., 2021)		X					
6 - (ZHENG et al., 2020)		X					
7 - (DALIBOR et al., 2020)		X					
8 - (MOURTZIS et al., 2021)		X	X	X			X
9 - (PAN et al., 2021)		X	X		X		
10 - (LIM et al., 2021)							
11 - (ZOU et al., 2022)		X					
12 - (GAO et al., 2022)		X					
13 - (THÜRER; LI; QU, 2022)		X					
14 - (NWOGU et al., 2022)							
DISSERTAÇÃO	X	X	X	X	X	X	X

Fonte: Autor.

7.3.2.1 Característica 1 - Propõe arquitetura a partir do estudo comparativo de múltiplas arquiteturas.

Considera-se que possuem esta característica aqueles trabalhos que propuseram um arquitetura a partir do exercício de identificação de similaridades entre

camadas/componentes de diferentes arquiteturas propostas por outros autores. O único, dos 15 trabalhos, que possui esta característica é esta dissertação.

7.3.2.2 Característica 2 - Apresenta ou cita estudo de caso.

Considera-se que possuem esta característica aqueles trabalhos que (i) citaram um estudo de caso hipotético para posterior/possível aplicação à arquitetura apresentada no trabalho, (ii) apresentaram um estudo de caso aplicado à arquitetura apresentada, sem detalhar como se deu essa aplicação ou (iii) apresentam estudo de caso aplicado/integrado à arquitetura e detalham como se deu tal integração. Com exceção dos trabalhos (LIM et al., 2021) e (NWOGU et al., 2022), todos os demais apresentaram estudos de caso.

7.3.2.3 Característica 3 - Apresenta experimentos.

Considera-se que possuem esta característica aqueles trabalhos que apresentaram quaisquer tipo de experimentos, sejam eles (i) integrados à arquitetura, (ii) realizados através de simulador sem integração com a arquitetura ou (iii) que sejam resultado da execução de um algoritmo que tenha alguma relação com a arquitetura, mesmo que ele não tenha sido integrado a ela. Além desta dissertação, os trabalhos (REDELINGHUY; BASSON; KRUGER, 2020), (KYCHKIN; NIKOLAEV, 2020), (BOTTANI; VIGNALI; TANCREDI, 2020), (MOURTZIS et al., 2021) e (PAN et al., 2021) também apresentam experimentos.

7.3.2.4 Característica 4 - Implementa a arquitetura para realização dos experimentos.

Considera-se que possuem esta característica aqueles trabalhos que deixam claro em seu texto a existência de um *testbed* no qual a arquitetura foi de fato implantada / desenvolvida / implementada, e que os experimentos foram extraídos a partir deste *testbed*. Além da presente dissertação, apenas os artigos (REDELINGHUY; BASSON; KRUGER, 2020) e (MOURTZIS et al., 2021) deixam claro que implementaram a arquitetura para a realização de experimentos.

7.3.2.5 Característica 5 - Experimenta diferentes cenários.

Considera-se que possuem esta característica aqueles trabalhos que apresentaram mais de um cenário de experimentação, resultando em mais de um tipo de gráfico com os cenários experimentados. Apenas esta dissertação e (PAN et al., 2021) experimentaram diferentes / múltiplos cenários.

7.3.2.6 Característica 6 - Experimenta, isoladamente, uma ou mais camadas / componentes da arquitetura.

Considera-se que possuem esta característica aqueles trabalhos que deixam claro em seu texto que os resultados de um ou mais experimentos foram obtidos através da testagem de apenas uma camada ou componente da arquitetura citada

no trabalho. Apenas esta dissertação apresentou testes isolados de um componente da arquitetura.

7.3.2.7 Característica 7 - Experimental, de forma integrada, diferentes camadas / componentes da arquitetura.

Considera-se que possuem esta característica aqueles trabalhos que apresentam um resultado de experimento que seja fruto da comunicação entre duas ou mais camadas/componentes de sua arquitetura. Apenas esta dissertação e os trabalhos (REDELINGHUYS; BASSON; KRUGER, 2020) e (MOURTZIS et al., 2021) possuem esta característica.

8 Conclusões

Os objetivos desta dissertação foram: 1. Apresentar uma revisão da literatura no contexto de *Digital Twin*, 2. Propor uma arquitetura genérica de DT, 3. Propor uma linguagem de design de DT, 4. Compreender as formas de avaliação de arquiteturas de DT e 5. Apresentar uma avaliação experimental para a arquitetura proposta, baseando-se nas formas de avaliação encontradas na literatura.

Atendendo ao objetivo 1 foi apresentada no Capítulo 2 uma revisão da literatura com ênfase na compreensão do DT e suas definições, aplicações, arquiteturas e linguagens de design.

Atendendo ao objetivo 2 foi apresentada no Capítulo 4 a proposta de uma arquitetura genérica, a qual contribui para a redução da lacuna bibliográfica que diz respeito à falta de padronização arquitetural na área de Digital Twin. Além disso, os resultados deste trabalho demonstram que é possível a utilização de uma arquitetura genérica em casos concretos aplicados a contextos específicos (não genéricos). Isso foi demonstrado pelo estudo de caso do Plante, que, mesmo possuindo originalmente outra arquitetura e sendo aplicado ao contexto agrícola, foi adaptado à arquitetura genérica proposta, sem perder suas características e funcionalidades originais.

Atendendo ao objetivo 3 foi apresentada no Capítulo 5 a proposta de uma linguagem de design, denominada AgroLD. A AgroLD contribui para a redução de duas lacunas bibliográficas: a escassez de LDs (i) de código aberto e completas e (ii) que possuam foco específico em uma área e/ou em um tipo de dispositivo. O código JSON da AgroLD foi publicado no GitHub¹, e foi aplicado ao contexto agrícola, através do estudo de caso do Plante.

Atendendo ao objetivo 4 foi apresentada no Capítulo 6 uma revisão da literatura para compreender as formas de avaliação de arquiteturas de DT. O resultado do estudo foi a detecção de duas metodologias de avaliação de arquiteturas: (i) aplicação da arquitetura em um estudo de caso, (ii) realização de experimentos, seja comparando o desempenho de diferentes tecnologias aplicadas à arquitetura, seja testando métodos ou funcionalidades relativas ao estudo de caso. Os experimentos podem ser realizados em uma ou mais camadas da arquitetura.

Atendendo ao objetivo 5 foram apresentados no Capítulo 7 experimentos feitos com intuito de validar a proposta de arquitetura genérica. Ficou claro neste capítulo que esta dissertação apresenta uma avaliação da arquitetura com mais etapas, e mais bem detalhada que os outros trabalhos revisados: (i) propondo a arquitetura a partir de um estudo comparativo entre diversas outras encontradas na literatura, (ii) aplicando a arquitetura a um estudo de caso, (iii) apresentando experimentos, (iv) implantando, de fato, a arquitetura para realizar os experimentos, (v) experimentando diferentes cenários, (vi) experimentando de forma integrada todas as camadas propostas e (vii) experimentando de forma isolada uma camada arquitetural. Além disso, é notável que a AgroLD possui algumas características

¹ https://github.com/RenatoAlves0/agrold/tree/main/plante_box_study_case/agrold

superiores à DTDL, como (i) menor tamanho das estruturas, (ii) não necessitando de interfaces, (iii) definição dos valores padrão na própria estrutura, (iv) menor tempo de inicialização, (v) menor consumo de RAM e CPU ao inicializar/ler as estruturas, (vi) *payload* menor, (vii) *alias*/apelidos de funções.

Como trabalhos futuros pretende-se aplicar a Arquitetura Genérica proposta a outros estudos de caso e realizar mais formas de experimentação. Em relação à AgroLD almeja-se compará-la a outras linguagens de design de DT que surjam no futuro.

Referências

- AHELEROFF, S. et al. Digital twin as a service (dtaas) in industry 4.0: An architecture reference model. *Advanced Engineering Informatics*, v. 47, p. 101225, 2021. ISSN 1474-0346. Citado 7 vezes nas páginas 17, 23, 27, 115, 116, 120 e 137.
- AL-FUQAHA, A. et al. Internet of things: A survey on enabling technologies, protocols, and applications. *IEEE Communications Surveys Tutorials*, v. 17, n. 4, p. 2347–2376, 2015. Citado na página 32.
- ALMHEIRI, A.; MAAMAR, Z. Iot protocols – mqtt versus coap. In: . New York, NY, USA: Association for Computing Machinery, 2021. (NISS2021). ISBN 9781450388719. Disponível em: <<https://doi.org/10.1145/3454127.3456594>>. Citado na página 33.
- ANGRISH, A. et al. A flexible data schema and system architecture for the virtualization of manufacturing machines (vmm). *Journal of Manufacturing Systems*, v. 45, p. 236–247, 2017. ISSN 0278-6125. Disponível em: <<https://www.sciencedirect.com/science/article/pii/S0278612517301371>>. Citado na página 39.
- BAJAJ, M.; COLE, B.; ZWEMER, D. Architecture to geometry - integrating system models with mechanical design. In: . [S.l.: s.n.], 2016. Citado na página 40.
- BAO, J. et al. The modelling and operations for the digital twin in the context of manufacturing. *Enterprise Information Systems*, Taylor Francis, v. 13, n. 4, p. 534–556, 2019. Disponível em: <<https://doi.org/10.1080/17517575.2018.1526324>>. Citado na página 27.
- BARRICELLI, B. R.; CASIRAGHI, E.; FOGLI, D. A survey on digital twin: Definitions, characteristics, applications, and design implications. *IEEE Access*, v. 7, p. 167653–167671, 2019. Citado 6 vezes nas páginas 38, 39, 41, 42, 43 e 60.
- BASSETT, L. *Introduction to JavaScript object notation: a to-the-point guide to JSON*. [S.l.]: "O'Reilly Media, Inc.", 2015. Citado na página 81.
- BOJE, C. et al. Towards a semantic construction digital twin: Directions for future research. *Automation in Construction*, v. 114, p. 103179, 2020. ISSN 0926-5805. Disponível em: <<https://www.sciencedirect.com/science/article/pii/S0926580519314785>>. Citado 11 vezes nas páginas 21, 27, 38, 39, 52, 53, 61, 68, 71, 74 e 75.
- BOLTON JANET R. MCCOLL-KENNEDY, L. C. A. G. C. O. L. M. Z. R. N. Customer experience challenges: bringing together digital, physical and social realms. *Journal of Service Management*. Citado na página 39.
- BORMANN, C. *CoAP - RFC 7252 Constrained Application Protocol*. 2016. Disponível em: <<https://coap.technology/>>, Último acesso em: 19/01/2022. Citado 2 vezes nas páginas 34 e 123.

- BOSCHERT, C. H. S.; ROSEN, R. Next generation digital twin. *Organizing Committee of TMCE*, 2018. Disponível em: <https://www.researchgate.net/profile/Stefan-Boschert/publication/325119950_Next_Generation_Digital_Twin/links/5af952ca0f7e9b026bf6e553/Next-Generation-Digital-Twin.pdf>. Citado na página 79.
- BOTTANI, E.; VIGNALI, G.; TANCREDI, G. P. C. A digital twin model of a pasteurization system for food beverages: tools and architecture. In: *IEEE International Conference on Engineering, Technology and Innovation (ICE/ITMC)*. [S.l.: s.n.], 2020. p. 1–8. Citado 6 vezes nas páginas 17, 23, 115, 120, 137 e 138.
- CATARCI, T. et al. A conceptual architecture and model for smart manufacturing relying on service-based digital twins. In: *IEEE International Conference on Web Services (ICWS)*. [S.l.: s.n.], 2019. p. 229–236. Citado 5 vezes nas páginas 17, 23, 114, 120 e 137.
- CIMINO, C.; NEGRI, E.; FUMAGALLI, L. Review of digital twin applications in manufacturing. *Computers in Industry*, v. 113, p. 103130, 2019. ISSN 0166-3615. Disponível em: <<https://www.sciencedirect.com/science/article/pii/S0166361519304385>>. Citado 7 vezes nas páginas 38, 40, 41, 42, 61, 67 e 73.
- CISCO. *Industry 4.0: How Cisco is Enabling the Future of Manufacturing*. 2019. Disponível em: <https://www.cisco.com/c/dam/en_us/solutions/industries/manufacturing/white-paper-c11-742529.pdf>, Último acesso em: 19/08/2021. Citado na página 26.
- DALIBOR, M. et al. Towards a model-driven architecture for interactive digital twin cockpits. In: DOBBIE, G. et al. (Ed.). *Conceptual Modeling*. Cham: Springer International Publishing, 2020. p. 377–387. ISBN 978-3-030-62522-1. Citado 6 vezes nas páginas 17, 23, 116, 117, 120 e 137.
- EASTMAN, C. et al. *BIM Handbook: A Guide to Building Information Modeling for Owners, Managers, Designers, Engineers and Contractors*. [S.l.]: Wiley Publishing, 2018. ISBN 0470185287. Citado na página 40.
- ELIJAH, O. et al. An overview of internet of things (iot) and data analytics in agriculture: Benefits and challenges. *IEEE Internet of Things Journal*, v. 5, n. 5, p. 3758–3773, 2018. Citado na página 32.
- FAN, C. et al. Disaster city digital twin: A vision for integrating artificial and human intelligence for disaster management. *International Journal of Information Management*, v. 56, p. 102049, 2021. ISSN 0268-4012. Disponível em: <<https://www.sciencedirect.com/science/article/pii/S0268401219302956>>. Citado 3 vezes nas páginas 38, 39 e 42.
- FULLER, A. et al. Digital twin: Enabling technologies, challenges and open research. *IEEE Access*, v. 8, p. 108952–108971, 2020. Citado 14 vezes nas páginas 22, 27, 38, 39, 41, 42, 43, 44, 56, 68, 71, 73, 74 e 75.
- GANGULI, R.; ADHIKARI, S. The digital twin of discrete dynamic systems: Initial approaches and future challenges. *Applied Mathematical Modelling*, v. 77, p. 1110–1128, 2020. ISSN 0307-904X. Disponível em: <<https://www.sciencedirect.com/science/article/pii/S0307904X19305670>>. Citado 2 vezes nas páginas 38 e 39.

GAO, B. et al. Design and typical application of digital twin architecture in smart grid. In: *7th Asia Conference on Power and Electrical Engineering (ACPEE)*. [S.l.: s.n.], 2022. p. 2140–2144. Citado 5 vezes nas páginas 17, 23, 118, 120 e 137.

GARTNER. *Gartner Top 10 Strategic Technology Trends for 2019*. 2019. Disponível em: <<https://www.gartner.com/smarterwithgartner/gartner-top-10-strategic-technology-trends-for-2019/>>, Último acesso em: 19/08/2021. Citado na página 26.

GRIEVES, M. Product lifecycle management: Driving the next generation of lean thinking. 01 2005. Citado 2 vezes nas páginas 26 e 37.

GRIEVES, M. Digital twin: Manufacturing excellence through virtual factory replication. 03 2015. Citado 9 vezes nas páginas 22, 27, 39, 57, 61, 72, 73, 79 e 121.

GUBBI, J. et al. Internet of things (iot): A vision, architectural elements, and future directions. *Future Generation Computer Systems*, v. 29, 07 2012. Citado na página 31.

HAAG, S.; ANDERL, R. Digital twin – proof of concept. *Manufacturing Letters*, v. 15, p. 64–66, 2018. ISSN 2213-8463. Industry 4.0 and Smart Manufacturing. Disponível em: <<https://www.sciencedirect.com/science/article/pii/S2213846318300208>>. Citado 2 vezes nas páginas 27 e 61.

Haghi Kashani, M. et al. A systematic review of iot in healthcare: Applications, techniques, and trends. *Journal of Network and Computer Applications*, v. 192, p. 103164, 2021. ISSN 1084-8045. Disponível em: <<https://www.sciencedirect.com/science/article/pii/S1084804521001764>>. Citado na página 31.

HE, B.; BAI, K. Digital twin-based sustainable intelligent manufacturing: a review. *Advances in Manufacturing*, p. 1–21, 2021. Citado 7 vezes nas páginas 26, 38, 39, 40, 41, 42 e 61.

HE, Y.; GUO, J.; ZHENG, X. From surveillance to digital twin: Challenges and recent advances of signal processing for industrial internet of things. *IEEE Signal Processing Magazine*, v. 35, n. 5, p. 120–129, 2018. Citado 2 vezes nas páginas 38 e 40.

IOT-ANALYTICS. *Cellular IoT LPWA Connectivity Market Tracker 2010-2025*. 2020. Disponível em: <<https://iot-analytics.com/state-of-the-iot-2020-12-billion-iot-connections-surpassing-non-iot-for-the-first-time/>>, Último acesso em: 19/08/2023. Citado na página 25.

JONES, D. et al. Characterising the digital twin: A systematic literature review. *CIRP Journal of Manufacturing Science and Technology*, v. 29, p. 36–52, 2020. ISSN 1755-5817. Disponível em: <<https://www.sciencedirect.com/science/article/pii/S1755581720300110>>. Citado 12 vezes nas páginas 21, 27, 38, 39, 50, 52, 59, 60, 68, 71, 72 e 74.

KANNAN, K.; ARUNACHALAM, N. A Digital Twin for Grinding Wheel: An Information Sharing Platform for Sustainable Grinding Process. *Journal of Manufacturing*

Science and Engineering, v. 141, n. 2, 12 2018. ISSN 1087-1357. 021015. Disponível em: <<https://doi.org/10.1115/1.4042076>>. Citado na página 40.

KHAJAVI, S. H. et al. Digital twin: Vision, benefits, boundaries, and creation for buildings. *IEEE Access*, v. 7, p. 147406–147419, 2019. Citado 5 vezes nas páginas 38, 40, 42, 60 e 61.

KOULAMAS, C.; KALOGERAS, A. Cyber-physical systems and digital twins in the industrial internet of things [cyber-physical systems]. *Computer*, v. 51, n. 11, p. 95–98, 2018. Citado 4 vezes nas páginas 26, 38, 40 e 44.

KRITZINGER, W. et al. Digital twin in manufacturing: A categorical literature review and classification. *IFAC-PapersOnLine*, v. 51, n. 11, p. 1016–1022, 2018. ISSN 2405-8963. 16th IFAC Symposium on Information Control Problems in Manufacturing INCOM 2018. Disponível em: <<https://www.sciencedirect.com/science/article/pii/S2405896318316021>>. Citado 5 vezes nas páginas 38, 39, 40, 41 e 42.

KYCHKIN, A.; NIKOLAEV, A. Iot-based mine ventilation control system architecture with digital twin. In: *2020 International Conference on Industrial Engineering, Applications and Manufacturing (ICIEAM)*. [S.l.: s.n.], 2020. p. 1–5. Citado 7 vezes nas páginas 17, 23, 114, 115, 120, 137 e 138.

LAGHARI A.A., W. K. L. R. A review and state of art of internet of things (iot). *Arch Computat Methods Eng*, 2021. Citado na página 31.

LIM, K. Y. H. et al. Digital twin architecture and development trends on manufacturing topologies. In: _____. *Implementing Industry 4.0: The Model Factory as the Key Enabler for the Future of Manufacturing*. Cham: Springer International Publishing, 2021. p. 259–286. ISBN 978-3-030-67270-6. Citado 7 vezes nas páginas 17, 23, 117, 118, 120, 137 e 138.

LIM, K. Y. H.; ZHENG, P.; CHEN, C.-H. A state-of-the-art survey of digital twin: techniques, engineering product lifecycle management and business innovation perspectives. *Journal of Intelligent Manufacturing*, 08 2020. Citado 4 vezes nas páginas 38, 40, 41 e 42.

LIU, M. et al. Review of digital twin about concepts, technologies, and industrial applications. *Journal of Manufacturing Systems*, v. 58, p. 346–361, 2021. ISSN 0278-6125. Digital Twin towards Smart Manufacturing and Industry 4.0. Disponível em: <<https://www.sciencedirect.com/science/article/pii/S0278612520301072>>. Citado 9 vezes nas páginas 38, 39, 40, 41, 42, 49, 59, 61 e 68.

LIU, N. M. Z.; MRAD, N. The role of data fusion in predictive maintenance using digital twin. *AIP Conference Proceedings 1949, 020023*, 2018. Citado na página 39.

LOMBARDI M., P. F. . S.-D. Internet of things: A general overview between architectures, protocols and applications. *Information*, 2021. Citado na página 31.

LONGO, F.; NICOLETTI, L.; PADOVANO, A. Ubiquitous knowledge empowers the smart factory: The impacts of a service-oriented digital twin on enterprises performance. *Annual Reviews in Control*, v. 47, p. 221–236, 2019. ISSN 1367-5788. Disponível em: <<https://www.sciencedirect.com/science/article/pii/>

S1367578818301299>. Citado 11 vezes nas páginas 22, 27, 38, 40, 42, 54, 55, 68, 71, 72 e 73.

LU, Y. et al. Digital twin-driven smart manufacturing: Connotation, reference model, applications and research issues. *Robotics and Computer-Integrated Manufacturing*, v. 61, p. 101837, 2020. ISSN 0736-5845. Disponível em: <<https://www.sciencedirect.com/science/article/pii/S0736584519302480>>. Citado 15 vezes nas páginas 22, 26, 27, 38, 39, 41, 42, 44, 56, 57, 60, 61, 68, 73 e 74.

MACCHI, M. et al. Exploring the role of digital twin for asset lifecycle management. *IFAC-PapersOnLine*, v. 51, n. 11, p. 790–795, 2018. ISSN 2405-8963. 16th IFAC Symposium on Information Control Problems in Manufacturing INCOM 2018. Disponível em: <<https://www.sciencedirect.com/science/article/pii/S2405896318315416>>. Citado 4 vezes nas páginas 38, 39, 41 e 42.

MADAKAM S., R. R.; TRIPATHI, S. Internet of things (iot): A literature review. *Journal of Computer and Communication*, 2015. Citado na página 31.

MADNI, A. M.; MADNI, C. C.; LUCERO, S. D. Leveraging digital twin technology in model-based systems engineering. *Systems*, v. 7, n. 1, 2019. ISSN 2079-8954. Disponível em: <<https://www.mdpi.com/2079-8954/7/1/7>>. Citado na página 39.

MICROSOFT. *Digital Twins Definition Language (DTDL)*. 2020. Disponível em: <<https://docs.microsoft.com/pt-br/azure/digital-twins/concepts-models>>, Último acesso em: 04/11/2021. Citado 3 vezes nas páginas 27, 61 e 93.

MICROSOFT. *Digital Twins Definition Language (DTDL)*. 2020. Disponível em: <<https://github.com/Azure/opensdtw-dtdl/blob/master/DTDL/v2/dtdlv2.md>>, Último acesso em: 04/11/2021. Citado 3 vezes nas páginas 61, 96 e 101.

MICROSOFT. *Documentação dos Gêmeos Digitais do Azure*. 2022. Disponível em: <<https://docs.microsoft.com/pt-br/azure/digital-twins/>>, Último acesso em: 27/01/2022. Citado na página 61.

MICROSOFT. *O que são Gêmeos Digitais do Azure?* 2022. Disponível em: <<https://docs.microsoft.com/pt-br/azure/digital-twins/overview>>, Último acesso em: 27/01/2022. Citado na página 61.

MINERVA, R.; LEE, G. M.; CRESPI, N. Digital twin in the iot context: A survey on technical features, scenarios, and architectural models. *Proceedings of the IEEE*, v. 108, n. 10, p. 1785–1824, 2020. Citado 11 vezes nas páginas 21, 27, 38, 39, 41, 42, 44, 61, 68, 71 e 74.

MOHAMMADI, N.; TAYLOR, J. E. Smart city digital twins. In: *2017 IEEE Symposium Series on Computational Intelligence (SSCI)*. [S.l.: s.n.], 2017. p. 1–5. Citado na página 39.

MOURTZIS, D. et al. A digital twin architecture for monitoring and optimization of fused deposition modeling processes. *Procedia CIRP*, v. 103, p. 97–102, 2021. ISSN 2212-8271. 9th CIRP Global Web Conference – Sustainable, resilient, and agile manufacturing and service operations : Lessons from COVID-19. Disponível em: <<https://www.sciencedirect.com/science/article/pii/S2212827121008568>>. Citado 8 vezes nas páginas 17, 23, 116, 117, 120, 137, 138 e 139.

M.S. RANA S.P., M. T.-G. D. D. S. O. Biometric security and internet of things (iot). In: . [S.l.: s.n.], 2019. Citado na página 25.

MULTIPLE, A. *Capture More IoT Market Share in 2022*. 2022. Disponível em: <<https://research.aimultiple.com/iot-future/>>, Último acesso em: 08/02/2022. Citado na página 25.

NEGRI, E.; FUMAGALLI, L.; MACCHI, M. A review of the roles of digital twin in cps-based production systems. *Procedia Manufacturing*, v. 11, p. 939–948, 2017. ISSN 2351-9789. 27th International Conference on Flexible Automation and Intelligent Manufacturing, FAIM2017, 27-30 June 2017, Modena, Italy. Disponível em: <<https://www.sciencedirect.com/science/article/pii/S2351978917304067>>. Citado 4 vezes nas páginas 38, 40, 41 e 42.

NWOGU, C. et al. Towards a requirement-driven digital twin architecture. *Procedia CIRP*, v. 107, p. 758–763, 2022. ISSN 2212-8271. Leading manufacturing systems transformation – Proceedings of the 55th CIRP Conference on Manufacturing Systems 2022. Disponível em: <<https://www.sciencedirect.com/science/article/pii/S2212827122003420>>. Citado 6 vezes nas páginas 17, 23, 119, 120, 137 e 138.

OASIS. *MQTT: The Standard for IoT Messaging*. 2020. Disponível em: <<https://mqtt.org/>>, Último acesso em: 20.09.2021. Citado na página 33.

OLIVEIRA, R.; BRAGA, R.; OLIVEIRA, C. Plante: An Intelligent Platform for Monitoring and Controlling of Agricultural Environments. In: MENA, F. M. et al. (Ed.). *8th International Workshop on ADVANCEs in ICT Infrastructures and Services (ADVANCE 2020)*. Cancún, Mexico, 2020. (Proc. of the 8th International Workshop on ADVANCEs in ICT Infrastructures and Services (ADVANCE 2020)), p. 1–8. Disponível em: <<https://hal.archives-ouvertes.fr/hal-02495231>>. Citado 2 vezes nas páginas 61 e 80.

OLIVEIRA, R.; OLIVEIRA, C.; NUNES, F. *Plante: uma plataforma inteligente para o monitoramento e irrigação agrícola*. 2020. Disponível em: <biblioteca.ifce.edu.br/index.asp?codigo_sophia=98979>, Último acesso em: 04/10/2021. Citado 3 vezes nas páginas 61, 62 e 80.

OLIVEIRA, R.; ROCHA, A.; OLIVEIRA, C. Um panorama comparativo de arquiteturas de digital twin no contexto de internet das coisas. In: *Anais da X Escola Regional de Computação do Ceará, Maranhão e Piauí*. Porto Alegre, RS, Brasil: SBC, 2022. p. 41–50. ISSN 0000-0000. Disponível em: <<https://sol.sbc.org.br/index.php/ercemapi/article/view/21958>>. Citado na página 28.

PALACIN, J. et al. Building a mobile robot for a floor-cleaning operation in domestic environments. *IEEE Transactions on Instrumentation and Measurement*, v. 53, n. 5, p. 1418–1424, 2004. Citado na página 25.

PAN, Y. et al. Digital twin based real-time production logistics synchronization system in a multi-level computing architecture. *Journal of Manufacturing Systems*, v. 58, p. 246–260, 2021. ISSN 0278-6125. Digital Twin towards Smart Manufacturing and Industry 4.0. Citado 7 vezes nas páginas 17, 23, 116, 117, 120, 137 e 138.

PATEL, K. et al. *Internet of Things-IOT: Definition, Characteristics, Architecture, Enabling Technologies, Application Future Challenges*. 2016. Citado na página 31.

PATINO, C.; FERREIRA, J. Inclusion and exclusion criteria in research studies: definitions and why they matter. *Jornal Brasileiro de Pneumologia*, v. 44, p. 84–84, 04 2018. Citado na página 36.

PAWAR, P.; TRIVEDI, A. Device-to-device communication based iot system: Benefits and challenges. *IETE Technical Review*, Taylor Francis, v. 36, n. 4, p. 362–374, 2019. Disponível em: <<https://doi.org/10.1080/02564602.2018.1476191>>. Citado na página 32.

PIZZA, S. et al. Smartwatch in <i>vivo</i>. In: *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*. New York, NY, USA: Association for Computing Machinery, 2016. (CHI '16), p. 5456–5469. ISBN 9781450333627. Disponível em: <<https://doi.org/10.1145/2858036.2858522>>. Citado na página 25.

QI, Q. et al. Enabling technologies and tools for digital twin. *Journal of Manufacturing Systems*, v. 58, p. 3–21, 2021. ISSN 0278-6125. Digital Twin towards Smart Manufacturing and Industry 4.0. Disponível em: <<https://www.sciencedirect.com/science/article/pii/S027861251930086X>>. Citado 15 vezes nas páginas 21, 27, 38, 39, 53, 54, 58, 59, 60, 68, 70, 71, 73, 74 e 75.

QI, Q. et al. Digital twin service towards smart manufacturing. *Procedia CIRP*, v. 72, p. 237–242, 2018. ISSN 2212-8271. 51st CIRP Conference on Manufacturing Systems. Disponível em: <<https://www.sciencedirect.com/science/article/pii/S2212827118302580>>. Citado 4 vezes nas páginas 38, 39, 59 e 60.

REDELINGHUYS, A. J. H.; BASSON, A. H.; KRUGER, K. A six-layer architecture for the digital twin: A manufacturing case study implementation. *J. Intell. Manuf.*, Springer-Verlag, Berlin, Heidelberg, v. 31, n. 6, p. 1383–1402, aug 2020. ISSN 0956-5515. Disponível em: <<https://doi.org/10.1007/s10845-019-01516-6>>. Citado 7 vezes nas páginas 17, 23, 114, 120, 137, 138 e 139.

SADDIK, A. E. Digital twins: The convergence of multimedia technologies. *IEEE MultiMedia*, v. 25, n. 2, p. 87–92, 2018. Citado na página 39.

SCHROEDER, G. N. et al. A methodology for digital twin modeling and deployment for industry 4.0. *Proceedings of the IEEE*, v. 109, n. 4, p. 556–567, 2021. Citado na página 27.

SHELBY ARM, K. H. C. B. U. B. T. Z. The constrained application protocol (coap). In: *Internet Engineering Task Force (IETF)*. [S.l.: s.n.], 2014. Citado na página 34.

SHELBY KLAUS HARTKE, C. B. Z. *The Constrained Application Protocol (CoAP)*. 2014. Disponível em: <<https://datatracker.ietf.org/doc/html/rfc7252>>, Último acesso em: 20/01/2021. Citado na página 34.

SONI, D.; MAKWANA, A. A survey on mqtt: a protocol of internet of things (iot). In: *International Conference On Telecommunication, Power Analysis And Computing Techniques (ICTPACT-2017)*. [S.l.: s.n.], 2017. v. 20. Citado na página 33.

STANDARD, O. *MQTT Version 5.0*. 2019. Disponível em: <<https://docs.oasis-open.org/mqtt/mqtt/v5.0/mqtt-v5.0.html>>, Último acesso em: 19.08.2021. Citado na página 33.

STARK, R.; FRESEMANN, C.; LINDOW, K. Development and operation of digital twins for technical systems and services. *CIRP Annals*, v. 68, n. 1, p. 129–132, 2019. ISSN 0007-8506. Disponível em: <<https://www.sciencedirect.com/science/article/pii/S0007850619300502>>. Citado 6 vezes nas páginas 21, 27, 47, 49, 73 e 74.

STATISTA. *Data volume of IoT connected devices worldwide 2019 and 2025*. 2020. Disponível em: <<https://www.statista.com/statistics/1017863/worldwide-iot-connected-devices-data-size/>>, Último acesso em: 08/02/2022. Citado na página 25.

TAO, F. et al. Digital twin-driven product design, manufacturing and service with big data. *The International Journal of Advanced Manufacturing Technology*, v. 94, 02 2018. Citado 5 vezes nas páginas 38, 39, 43, 59 e 60.

TAO, F. et al. Digital twins and cyber-physical systems toward smart manufacturing and industry 4.0: Correlation and comparison. *Engineering*, v. 5, p. 653–661, 08 2019. Citado 10 vezes nas páginas 21, 27, 38, 39, 47, 48, 68, 72, 73 e 74.

TAO, F.; ZHANG, M. Digital twin shop-floor: A new shop-floor paradigm towards smart manufacturing. *IEEE Access*, v. 5, p. 20418–20427, 2017. Citado 10 vezes nas páginas 21, 27, 38, 39, 49, 50, 68, 71, 73 e 75.

THANGAVEL, D. et al. Performance evaluation of mqtt and coap via a common middleware. In: *2014 IEEE Ninth International Conference on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP)*. [S.l.: s.n.], 2014. p. 1–6. Citado na página 33.

THÜRER, M.; LI, S. S.; QU, T. Digital twin architecture for production logistics: The critical role of programmable logic controllers (plcs). *Procedia Computer Science*, v. 200, p. 710–717, 2022. ISSN 1877-0509. 3rd International Conference on Industry 4.0 and Smart Manufacturing. Disponível em: <<https://www.sciencedirect.com/science/article/pii/S1877050922002782>>. Citado 6 vezes nas páginas 17, 23, 118, 119, 120 e 137.

UFRJ. *Constrained Application Protocol*. 2016. Disponível em: <<https://www.gta.ufrj.br/ensino/eel878/redes1-2019-1/vf/coap/>>, Último acesso em: 12/02/2023. Citado na página 34.

UHLEMANN, T. et al. The digital twin: Demonstrating the potential of real time data acquisition in production systems. *Procedia Manufacturing*, v. 9, p. 113–120, 12 2017. Citado 2 vezes nas páginas 38 e 43.

Urbina Coronado, P. D. et al. Part data integration in the shop floor digital twin: Mobile and cloud technologies to enable a manufacturing execution system. *Journal of Manufacturing Systems*, v. 48, p. 25–33, 2018. ISSN 0278-6125. Special Issue on Smart Manufacturing. Disponível em: <<https://www.sciencedirect.com/science/article/pii/S027861251830013X>>. Citado 2 vezes nas páginas 38 e 40.

- VRABIČ, R. et al. Digital twins: Understanding the added value of integrated models for through-life engineering services. *Procedia Manufacturing*, v. 16, p. 139–146, 2018. ISSN 2351-9789. Proceedings of the 7th International Conference on Through-life Engineering Services. Disponível em: <<https://www.sciencedirect.com/science/article/pii/S2351978918312897>>. Citado na página 39.
- VUKOVIĆ, M. et al. Digital twins in industrial iot: a survey of the state of the art and of relevant standards. In: *2021 IEEE International Conference on Communications Workshops (ICC Workshops)*. [S.l.: s.n.], 2021. p. 1–6. Citado 2 vezes nas páginas 26 e 37.
- WEI, C.-c. et al. Implementation of a data acquisition system for heterogeneous machines. In: *2017 IEEE/SICE International Symposium on System Integration (SII)*. [S.l.: s.n.], 2017. p. 232–235. Citado 2 vezes nas páginas 26 e 37.
- WRIGHT, L.; DAVIDSON, S. How to tell the difference between a model and a digital twin. *Advanced Modeling and Simulation in Engineering Sciences*, v. 7, 12 2020. Citado 8 vezes nas páginas 38, 39, 59, 60, 61, 79, 82 e 121.
- WU, B. et al. Battery digital twins: Perspectives on the fusion of models, data and artificial intelligence for smart battery management systems. *Energy and AI*, v. 1, p. 100016, 2020. ISSN 2666-5468. Disponível em: <<https://www.sciencedirect.com/science/article/pii/S2666546820300161>>. Citado 10 vezes nas páginas 21, 27, 38, 43, 45, 46, 68, 72, 73 e 74.
- ZHENG, X. et al. A quality-oriented digital twin modelling method for manufacturing processes based on a multi-agent architecture. *Procedia Manufacturing*, v. 51, p. 309–315, 2020. ISSN 2351-9789. Disponível em: <<https://www.sciencedirect.com/science/article/pii/S2351978920318990>>. Citado 5 vezes nas páginas 17, 23, 116, 120 e 137.
- ZOU, S. et al. A preliminary study on the development and application of digital twin landscape architectures in the context of smart city. In: *Global Conference on Robotics, Artificial Intelligence and Information Technology (GCRAIT)*. [S.l.: s.n.], 2022. p. 322–327. Citado 5 vezes nas páginas 17, 23, 118, 120 e 137.
- ČOLAKOVIĆ, A.; HADŽIALIĆ, M. Internet of things (iot): A review of enabling technologies, challenges, and open research issues. *Computer Networks*, v. 144, p. 17–39, 2018. ISSN 1389-1286. Disponível em: <<https://www.sciencedirect.com/science/article/pii/S1389128618305243>>. Citado na página 31.

Apêndices

APÊNDICE A – AgroLD, a Linguagem de Design (LD) proposta, aplicada ao Plante Box

Todos os códigos JSON apresentados a seguir estão escritos exatamente como ficariam dispostos dentro do componente Linguagem de Design da arquitetura de Digital Twin proposta (Figura 19). Isto é, levando em consideração o modelo completo dos componentes, mesmo os campos que não estão preenchidos, ou seja, que estão com valor nulo.

O código abaixo é equivalente à estrutura da Figura 20.

```

1 {
2   "entity": {
3     "id": "planteBox",
4     "sensors": [
5       {
6         "id": "dht11",
7         "data": ["temperatura", "umidade"]
8       },
9       {
10        "id": "yl69",
11        "data": ["umidade do solo"]
12      },
13      {
14        "id": "yl83",
15        "data": ["chuva"]
16      },
17      { "id": "ldr", "data": ["luz"] }
18    ],
19    "actuators": [
20      {
21        "id": "irrigador",
22        "actions": ["irrigar"]
23      }
24    ],
25    "cpu": {
26      "model": "ESP 32S",
27      "cores": "1",
28      "lithography": { "value": "5", "unity": "mm" },
29      "clock": { "value": "240", "unity": "MHz" },
30      "temperature": {

```

```

31     "min": "-40",
32     "max": "150",
33     "unity": "degreeCelsius"
34 }
35 },
36 "memory": [
37     {
38         "id": "Flash",
39         "model": "",
40         "type": "SSD",
41         "capacity": { "value": "4", "unity": "MB" },
42         "speed": { "value": "", "unity": "" }
43     },
44     {
45         "id": "SRAM",
46         "model": "",
47         "type": "SSD",
48         "capacity": { "value": "520", "unity": "KB" },
49         "speed": { "value": "", "unity": "" }
50     },
51     {
52         "id": "ROM",
53         "model": "",
54         "type": "SSD",
55         "capacity": { "value": "448", "unity": "KB" },
56         "speed": { "value": "", "unity": "" }
57     }
58 ],
59 "battery": {
60     "model": "",
61     "type": "",
62     "capacity": { "value": "", "unity": "" },
63     "temperature": { "min": "", "max": "", "unity": "" }
64 },
65 "communication": [
66     {
67         "id": "wifi",
68         "protocol": "IEEE 802.11 b/g/n",
69         "range": { "value": "100/180", "unity": "m" },
70         "speed": { "value": "150", "unity": "Mbps" }
71     },
72     {
73         "id": "bluetooth",
74         "protocol": "BLE v4.2",
75         "range": { "value": "60", "unity": "m" },
76         "speed": { "value": "1", "unity": "Mbps" }

```

```
77     }
78 ],
79 "functions": [
80     {
81         "id": "ler",
82         "alias": "l",
83         "description": "ler dados dos sensores",
84         "parameters": [
85             {
86                 "id": "",
87                 "type": "",
88                 "description": "",
89                 "enum": [],
90                 "aliasEnum": [],
91                 "required": ""
92             }
93         ],
94         "response": [
95             {
96                 "id": "temperatura",
97                 "alias": "t",
98                 "type": "float",
99                 "unity": "degreeCelsius"
100             },
101             {
102                 "id": "umidade",
103                 "alias": "u",
104                 "type": "float",
105                 "unity": "percent"
106             },
107             {
108                 "id": "umidade do solo",
109                 "alias": "us",
110                 "type": "float",
111                 "unity": "percent"
112             },
113             {
114                 "id": "chuva",
115                 "alias": "c",
116                 "type": "float",
117                 "unity": "percent"
118             },
119             {
120                 "id": "luz",
121                 "alias": "l",
122                 "type": "float",
```

```

123         "unity": "percent"
124     }
125 ]
126 },
127 {
128     "id": "ligarDesligar",
129     "alias": "ld",
130     "description": "ligar ou desligar atuador ou sensores",
131     "parameters": [
132         {
133             "id": "",
134             "type": "",
135             "description": "",
136             "enum": [
137                 "ligarAtuador",
138                 "ligarSensores",
139                 "desligarAtuador",
140                 "desligarSensores"
141             ],
142             "aliasEnum": ["la", "ls", "da", "ds"],
143             "required": "1"
144         }
145     ],
146     "response": [
147         {
148             "id": "executado com sucesso",
149             "alias": "es",
150             "type": "boolean",
151             "unity": ""
152         }
153     ]
154 },
155 {
156     "id": "temperaturaOperacao",
157     "alias": "to",
158     "description": "ler temperatura de operacao da cpu e bateria",
159     "parameters": [
160         {
161             "id": "",
162             "type": "",
163             "description": "",
164             "enum": [],
165             "aliasEnum": [],
166             "required": ""
167         }
168     ],

```

```
169     "response": [  
170         {  
171             "id": "temperatura da cpu",  
172             "alias": "tc",  
173             "type": "float",  
174             "unity": "degreeCelsius"  
175         },  
176         {  
177             "id": "temperatura da bateria",  
178             "alias": "tb",  
179             "type": "float",  
180             "unity": "degreeCelsius"  
181         }  
182     ]  
183 },  
184 {  
185     "id": "medirConsumo",  
186     "alias": "mc",  
187     "description": "ler consumo da cpu, bateria e memoria flash",  
188     "parameters": [  
189         {  
190             "id": "",  
191             "type": "",  
192             "description": "",  
193             "enum": [],  
194             "aliasEnum": [],  
195             "required": ""  
196         }  
197     ],  
198     "response": [  
199         {  
200             "id": "consumo da cpu",  
201             "alias": "cc",  
202             "type": "float",  
203             "unity": "percent"  
204         },  
205         {  
206             "id": "consumo da bateria",  
207             "alias": "cb",  
208             "type": "float",  
209             "unity": "percent"  
210         },  
211         {  
212             "id": "consumo da memoria flash",  
213             "alias": "cmf",  
214             "type": "float",
```

```
215     "unity": "percent"  
216   }  
217 ]  
218 }  
219 ]  
220 }  
221 }
```

APÊNDICE B – Digital Twins Definition Language (DTDL), aplicada ao Plante Box

Todos os códigos JSON apresentados a seguir estão escritos exatamente como ficariam dispostos dentro do componente Linguagem de Design da arquitetura de Digital Twin proposta (Figura 19). Isto é, levando em consideração o modelo completo dos componentes, mesmo os campos que não estão preenchidos, ou seja, que estão com valor nulo.

O código abaixo é equivalente à estrutura da Figura 21.

```

1 {
2   "@context": "dtmi:dtdl:context;2",
3   "@id": "dtmi:com:example:planteBox;1",
4   "@type": "Interface",
5   "contents": [
6     {
7       "@type": "Component",
8       "name": "dht11",
9       "schema": "dtmi:com:example:dht11;1",
10      "@id": "",
11      "comment": "",
12      "description": "",
13      "displayName": ""
14    },
15    {
16      "@type": "Component",
17      "name": "yl69",
18      "schema": "dtmi:com:example:yl69;1",
19      "@id": "",
20      "comment": "",
21      "description": "",
22      "displayName": ""
23    },
24    {
25      "@type": "Component",
26      "name": "yl83",
27      "schema": "dtmi:com:example:yl83;1",
28      "@id": "",
29      "comment": "",
30      "description": "",

```

```
31     "displayName": ""
32   },
33   {
34     "@type": "Component",
35     "name": "ldr",
36     "schema": "dtmi:com:example:ldr;1",
37     "@id": "",
38     "comment": "",
39     "description": "",
40     "displayName": ""
41   },
42   {
43     "@type": "Component",
44     "name": "irrigador",
45     "schema": "dtmi:com:example:irrigador;1",
46     "@id": "",
47     "comment": "",
48     "description": "",
49     "displayName": ""
50   },
51   {
52     "@type": "Component",
53     "name": "cpu",
54     "schema": "dtmi:com:example:cpu;1",
55     "@id": "",
56     "comment": "",
57     "description": "",
58     "displayName": ""
59   },
60   {
61     "@type": "Component",
62     "name": "Flash",
63     "schema": "dtmi:com:example:memoria;1",
64     "@id": "",
65     "comment": "",
66     "description": "",
67     "displayName": ""
68   },
69   {
70     "@type": "Component",
71     "name": "SRAM",
72     "schema": "dtmi:com:example:memoria;1",
73     "@id": "",
74     "comment": "",
75     "description": "",
76     "displayName": ""
```

```
77 },
78 {
79     "@type": "Component",
80     "name": "ROM",
81     "schema": "dtmi:com:example:memoria;1",
82     "@id": "",
83     "comment": "",
84     "description": "",
85     "displayName": ""
86 },
87 {
88     "@type": "Component",
89     "name": "bateria",
90     "schema": "dtmi:com:example:bateria;1",
91     "@id": "",
92     "comment": "",
93     "description": "",
94     "displayName": ""
95 },
96 {
97     "@type": "Component",
98     "name": "wifi",
99     "schema": "dtmi:com:example:comunicacao;1",
100     "@id": "",
101     "comment": "",
102     "description": "",
103     "displayName": ""
104 },
105 {
106     "@type": "Component",
107     "name": "bluetooth",
108     "schema": "dtmi:com:example:comunicacao;1",
109     "@id": "",
110     "comment": "",
111     "description": "",
112     "displayName": ""
113 },
114 {
115     "@type": "Command",
116     "name": "ler",
117     "@id": "",
118     "comment": "",
119     "description": "ler dados dos sensores",
120     "displayName": "",
121     "request": {
122         "name": "",
```

```
123     "schema": "",
124     "@id": "",
125     "comment": "",
126     "description": "",
127     "displayName": ""
128 },
129 "response": {
130     "name": "dados dos sensores",
131     "schema": {
132         "@type": "Object",
133         "fields": [
134             {
135                 "name": "temperatura",
136                 "schema": "float",
137                 "@id": "",
138                 "comment": "unidade: degreeCelsius",
139                 "description": "",
140                 "displayName": ""
141             },
142             {
143                 "name": "umidade",
144                 "schema": "float",
145                 "@id": "",
146                 "comment": "unidade: percent",
147                 "description": "",
148                 "displayName": ""
149             },
150             {
151                 "name": "umidade do solo",
152                 "schema": "float",
153                 "@id": "",
154                 "comment": "unidade: percent",
155                 "description": "",
156                 "displayName": ""
157             },
158             {
159                 "name": "chuva",
160                 "schema": "float",
161                 "@id": "",
162                 "comment": "unidade: percent",
163                 "description": "",
164                 "displayName": ""
165             },
166             {
167                 "name": "luz",
168                 "schema": "float",
```

```

169         "@id": "",
170         "comment": "unidade: percent",
171         "description": "",
172         "displayName": ""
173     }
174 ],
175     "@id": "",
176     "comment": "",
177     "description": "",
178     "displayName": ""
179 },
180     "@id": "",
181     "comment": "",
182     "description": "",
183     "displayName": ""
184 }
185 },
186 {
187     "@type": "Command",
188     "name": "ligarDesligar",
189     "@id": "",
190     "comment": "",
191     "description": "liga ou desliga atuador ou sensores",
192     "displayName": "",
193     "request": {
194         "name": "",
195         "schema": {
196             "@type": "Enum",
197             "valueSchema": "string",
198             "enumValues": [
199                 {
200                     "name": "ligarAtuador",
201                     "enumValue": "la",
202                     "@id": "",
203                     "comment": "",
204                     "description": "",
205                     "displayName": ""
206                 },
207                 {
208                     "name": "ligarSensores",
209                     "enumValue": "ls",
210                     "@id": "",
211                     "comment": "",
212                     "description": "",
213                     "displayName": ""
214                 }

```

```

215         {
216             "name": "desligarAtuador",
217             "enumValue": "da",
218             "@id": "",
219             "comment": "",
220             "description": "",
221             "displayName": ""
222         },
223         {
224             "name": "desligarSensores",
225             "enumValue": "ds",
226             "@id": "",
227             "comment": "",
228             "description": "",
229             "displayName": ""
230         }
231     ],
232     "@id": "",
233     "comment": "",
234     "description": "",
235     "displayName": ""
236 },
237 "@id": "",
238 "comment": "",
239 "description": "",
240 "displayName": ""
241 },
242 "response": {
243     "name": "executado com sucesso",
244     "schema": "boolean",
245     "@id": "",
246     "comment": "",
247     "description": "",
248     "displayName": ""
249 }
250 },
251 {
252     "@type": "Command",
253     "name": "temperaturaOperacao",
254     "@id": "",
255     "comment": "",
256     "description": "ler temperatura de operacao da CPU e bateria",
257     "displayName": "",
258     "request": {
259         "name": "",
260         "schema": "",

```

```

261     "@id": "",
262     "comment": "",
263     "description": "",
264     "displayName": ""
265 },
266 "response": {
267     "name": "dados dos sensores",
268     "schema": {
269         "@type": "Object",
270         "fields": [
271             {
272                 "name": "temperatura da cpu",
273                 "schema": "float",
274                 "@id": "",
275                 "comment": "unidade: degreeCelsius",
276                 "description": "",
277                 "displayName": ""
278             },
279             {
280                 "name": "temperatura da bateria",
281                 "schema": "float",
282                 "@id": "",
283                 "comment": "unidade: degreeCelsius",
284                 "description": "",
285                 "displayName": ""
286             }
287         ],
288         "@id": "",
289         "comment": "",
290         "description": "",
291         "displayName": ""
292     },
293     "@id": "",
294     "comment": "",
295     "description": "",
296     "displayName": ""
297 }
298 },
299 {
300     "@type": "Command",
301     "name": "medirConsumo",
302     "@id": "",
303     "comment": "",
304     "description": "ler consumo da CPU, bateria e memoria flash",
305     "displayName": "",
306     "request": {

```

```
307     "name": "",
308     "schema": "",
309     "@id": "",
310     "comment": "",
311     "description": "",
312     "displayName": ""
313 },
314 "response": {
315     "name": "dados dos sensores",
316     "schema": {
317         "@type": "Object",
318         "fields": [
319             {
320                 "name": "consumo da cpu",
321                 "schema": "float",
322                 "@id": "",
323                 "comment": "unidade: percent",
324                 "description": "",
325                 "displayName": ""
326             },
327             {
328                 "name": "consumo da bateria",
329                 "schema": "float",
330                 "@id": "",
331                 "comment": "unidade: percent",
332                 "description": "",
333                 "displayName": ""
334             },
335             {
336                 "name": "consumo da memoria flash",
337                 "schema": "float",
338                 "@id": "",
339                 "comment": "unidade: percent",
340                 "description": "",
341                 "displayName": ""
342             }
343         ],
344         "@id": "",
345         "comment": "",
346         "description": "",
347         "displayName": ""
348     },
349     "@id": "",
350     "comment": "",
351     "description": "",
352     "displayName": ""
```

```

353     }
354   }
355 ]
356 }

```

Os códigos abaixo são equivalentes aos arquivos do **DTD** (**Coleção de Interfaces**) da Figura ??.

Interface do DHT11:

```

1 {
2   "@context": "dtmi:dtdl:context;2",
3   "@id": "dtmi:com:example:dht11;1",
4   "@type": "Interface",
5   "contents": [
6     {
7       "@type": ["Telemetry"],
8       "name": "temperatura",
9       "schema": "string",
10      "@id": "",
11      "comment": "",
12      "description": "",
13      "displayName": "",
14      "unit": "degreeCelsius"
15    },
16    {
17      "@type": ["Telemetry"],
18      "name": "umidade",
19      "schema": "string",
20      "@id": "",
21      "comment": "",
22      "description": "",
23      "displayName": "",
24      "unit": "percent"
25    }
26  ]
27 }

```

Interface do LDR:

```

1 {
2   "@context": "dtmi:dtdl:context;2",
3   "@id": "dtmi:com:example:ldr;1",
4   "@type": "Interface",
5   "contents": [
6     {
7       "@type": ["Telemetry"],

```

```
8     "name": "luz",
9     "schema": "string",
10    "@id": "",
11    "comment": "",
12    "description": "",
13    "displayName": "",
14    "unit": "percent"
15  }
16 ]
17 }
```

Interface do YL69:

```
1 {
2   "@context": "dtmi:dtdl:context;2",
3   "@id": "dtmi:com:example:yl69;1",
4   "@type": "Interface",
5   "contents": [
6     {
7       "@type": ["Telemetry"],
8       "name": "umidade do solo",
9       "schema": "string",
10      "@id": "",
11      "comment": "",
12      "description": "",
13      "displayName": "",
14      "unit": "percent"
15    }
16  ]
17 }
```

Interface do YL83:

```
1 {
2   "@context": "dtmi:dtdl:context;2",
3   "@id": "dtmi:com:example:yl83;1",
4   "@type": "Interface",
5   "contents": [
6     {
7       "@type": ["Telemetry"],
8       "name": "chuva",
9       "schema": "string",
10      "@id": "",
11      "comment": "",
12      "description": "",
13      "displayName": "",

```

```

14     "unit": "percent"
15   }
16 ]
17 }

```

Interface do Irrigador:

```

1 {
2   "@context": "dtmi:dtdl:context;2",
3   "@id": "dtmi:com:example:irrigador;1",
4   "@type": "Interface",
5   "contents": [
6     {
7       "@type": "Property",
8       "name": "Irigar",
9       "schema": "boolean",
10      "@id": "",
11      "comment": "",
12      "description": "",
13      "displayName": "",
14      "unit": "",
15      "writable": ""
16    }
17  ]
18 }

```

Interface da CPU:

```

1 {
2   "@context": "dtmi:dtdl:context;2",
3   "@id": "dtmi:com:example:cpu;1",
4   "@type": "Interface",
5   "contents": [
6     {
7       "@type": "Property",
8       "name": "model",
9       "schema": "string",
10      "@id": "",
11      "comment": "",
12      "description": "",
13      "displayName": "",
14      "unit": "",
15      "writable": ""
16    },
17    {
18      "@type": "Property",

```

```
19     "name": "cores",
20     "schema": "string",
21     "@id": "",
22     "comment": "",
23     "description": "",
24     "displayName": "",
25     "unit": "",
26     "writable": ""
27 },
28 {
29     "@type": "Property",
30     "name": "lithography",
31     "schema": {
32         "@type": "Object",
33         "fields": [
34             {
35                 "name": "value",
36                 "schema": "string",
37                 "@id": "",
38                 "comment": "",
39                 "description": "",
40                 "displayName": ""
41             },
42             {
43                 "name": "unity",
44                 "schema": "string",
45                 "@id": "",
46                 "comment": "",
47                 "description": "",
48                 "displayName": ""
49             }
50         ],
51         "@id": "",
52         "comment": "",
53         "description": "",
54         "displayName": ""
55     },
56     "@id": "",
57     "comment": "",
58     "description": "",
59     "displayName": "",
60     "unit": "",
61     "writable": ""
62 },
63 {
64     "@type": "Property",
```

```

65     "name": "clock",
66     "schema": {
67         "@type": "Object",
68         "fields": [
69             {
70                 "name": "value",
71                 "schema": "string",
72                 "@id": "",
73                 "comment": "",
74                 "description": "",
75                 "displayName": ""
76             },
77             {
78                 "name": "unity",
79                 "schema": "string",
80                 "@id": "",
81                 "comment": "",
82                 "description": "",
83                 "displayName": ""
84             }
85         ],
86         "@id": "",
87         "comment": "",
88         "description": "",
89         "displayName": ""
90     },
91     "@id": "",
92     "comment": "",
93     "description": "",
94     "displayName": "",
95     "unit": "",
96     "writable": ""
97 },
98 {
99     "@type": "Property",
100    "name": "temperature",
101    "schema": {
102        "@type": "Object",
103        "fields": [
104            {
105                "name": "min",
106                "schema": "string",
107                "@id": "",
108                "comment": "",
109                "description": "",
110                "displayName": ""

```

```

111     },
112     {
113         "name": "max",
114         "schema": "string",
115         "@id": "",
116         "comment": "",
117         "description": "",
118         "displayName": ""
119     },
120     {
121         "name": "unity",
122         "schema": "string",
123         "@id": "",
124         "comment": "",
125         "description": "",
126         "displayName": ""
127     }
128 ],
129 "@id": "",
130 "comment": "",
131 "description": "",
132 "displayName": ""
133 },
134 "@id": "",
135 "comment": "",
136 "description": "",
137 "displayName": "",
138 "unit": "",
139 "writable": ""
140 }
141 ]
142 }

```

Interface da Memória:

```

1 {
2   "@context": "dtmi:dtdl:context;2",
3   "@id": "dtmi:com:example:memoria;1",
4   "@type": "Interface",
5   "contents": [
6     {
7       "@type": "Property",
8       "name": "model",
9       "schema": "string",
10      "@id": "",
11      "comment": "",

```

```

12     "description": "",
13     "displayName": "",
14     "unit": "",
15     "writable": ""
16 },
17 {
18     "@type": "Property",
19     "name": "type",
20     "schema": "string",
21     "@id": "",
22     "comment": "",
23     "description": "",
24     "displayName": "",
25     "unit": "",
26     "writable": ""
27 },
28 {
29     "@type": "Property",
30     "name": "capacity",
31     "schema": {
32         "@type": "Object",
33         "fields": [
34             {
35                 "name": "value",
36                 "schema": "string",
37                 "@id": "",
38                 "comment": "",
39                 "description": "",
40                 "displayName": ""
41             },
42             {
43                 "name": "unity",
44                 "schema": "string",
45                 "@id": "",
46                 "comment": "",
47                 "description": "",
48                 "displayName": ""
49             }
50         ],
51         "@id": "",
52         "comment": "",
53         "description": "",
54         "displayName": ""
55     },
56     "@id": "",
57     "comment": "",

```

```

58     "description": "",
59     "displayName": "",
60     "unit": "",
61     "writable": ""
62 },
63 {
64     "@type": "Property",
65     "name": "speed",
66     "schema": {
67         "@type": "Object",
68         "fields": [
69             {
70                 "name": "value",
71                 "schema": "string",
72                 "@id": "",
73                 "comment": "",
74                 "description": "",
75                 "displayName": ""
76             },
77             {
78                 "name": "unity",
79                 "schema": "string",
80                 "@id": "",
81                 "comment": "",
82                 "description": "",
83                 "displayName": ""
84             }
85         ],
86         "@id": "",
87         "comment": "",
88         "description": "",
89         "displayName": ""
90     },
91     "@id": "",
92     "comment": "",
93     "description": "",
94     "displayName": "",
95     "unit": "",
96     "writable": ""
97 }
98 ]
99 }

```

Interface da Bateria:

```
1 {
```

```

2  "@context": "dtmi:dtdl:context;2",
3  "@id": "dtmi:com:example:bateria;1",
4  "@type": "Interface",
5  "contents": [
6    {
7      "@type": "Property",
8      "name": "model",
9      "schema": "string",
10     "@id": "",
11     "comment": "",
12     "description": "",
13     "displayName": "",
14     "unit": "",
15     "writable": ""
16   },
17   {
18     "@type": "Property",
19     "name": "type",
20     "schema": "string",
21     "@id": "",
22     "comment": "",
23     "description": "",
24     "displayName": "",
25     "unit": "",
26     "writable": ""
27   },
28   {
29     "@type": "Property",
30     "name": "capacity",
31     "schema": {
32       "@type": "Object",
33       "fields": [
34         {
35           "name": "value",
36           "schema": "string",
37           "@id": "",
38           "comment": "",
39           "description": "",
40           "displayName": ""
41         },
42         {
43           "name": "unity",
44           "schema": "string",
45           "@id": "",
46           "comment": "",
47           "description": "",

```

```

48         "displayName": ""
49     }
50 ],
51     "@id": "",
52     "comment": "",
53     "description": "",
54     "displayName": ""
55 },
56     "@id": "",
57     "comment": "",
58     "description": "",
59     "displayName": "",
60     "unit": "",
61     "writable": ""
62 },
63 {
64     "@type": "Property",
65     "name": "temperature",
66     "schema": {
67         "@type": "Object",
68         "fields": [
69             {
70                 "name": "min",
71                 "schema": "string",
72                 "@id": "",
73                 "comment": "",
74                 "description": "",
75                 "displayName": ""
76             },
77             {
78                 "name": "max",
79                 "schema": "string",
80                 "@id": "",
81                 "comment": "",
82                 "description": "",
83                 "displayName": ""
84             },
85             {
86                 "name": "unity",
87                 "schema": "string",
88                 "@id": "",
89                 "comment": "",
90                 "description": "",
91                 "displayName": ""
92             }
93         ],

```

```

94     "@id": "",
95     "comment": "",
96     "description": "",
97     "displayName": ""
98 },
99     "@id": "",
100    "comment": "",
101    "description": "",
102    "displayName": "",
103    "unit": "",
104    "writable": ""
105 }
106 ]
107 }

```

Interface da Comunicação:

```

1 {
2   "@context": "dtmi:dtdl:context;2",
3   "@id": "dtmi:com:example:comunicacao;1",
4   "@type": "Interface",
5   "contents": [
6     {
7       "@type": "Property",
8       "name": "protocol",
9       "schema": "string",
10      "@id": "",
11      "comment": "",
12      "description": "",
13      "displayName": "",
14      "unit": "",
15      "writable": ""
16    },
17    {
18      "@type": "Property",
19      "name": "range",
20      "schema": {
21        "@type": "Object",
22        "fields": [
23          {
24            "name": "value",
25            "schema": "string",
26            "@id": "",
27            "comment": "",
28            "description": "",
29            "displayName": ""

```

```

30     },
31     {
32         "name": "unity",
33         "schema": "string",
34         "@id": "",
35         "comment": "",
36         "description": "",
37         "displayName": ""
38     }
39 ],
40 "@id": "",
41 "comment": "",
42 "description": "",
43 "displayName": ""
44 },
45 "@id": "",
46 "comment": "",
47 "description": "",
48 "displayName": "",
49 "unit": "",
50 "writable": ""
51 },
52 {
53     "@type": "Property",
54     "name": "speed",
55     "schema": {
56         "@type": "Object",
57         "fields": [
58             {
59                 "name": "value",
60                 "schema": "string",
61                 "@id": "",
62                 "comment": "",
63                 "description": "",
64                 "displayName": ""
65             },
66             {
67                 "name": "unity",
68                 "schema": "string",
69                 "@id": "",
70                 "comment": "",
71                 "description": "",
72                 "displayName": ""
73             }
74         ],
75         "@id": "",

```

```
76     "comment": "",
77     "description": "",
78     "displayName": ""
79 },
80     "@id": "",
81     "comment": "",
82     "description": "",
83     "displayName": "",
84     "unit": "",
85     "writable": ""
86 }
87 ]
88 }
```


APÊNDICE C – Mensagens necessárias para a definição dos valores padrão dos componentes CPU, bateria, memória e comunicação na DTDL

Destaca-se que essa etapa adicional é necessária na DTDL devido ao fato de não ser possível declarar um valor padrão dentro do modelo da linguagem. Então, após a construção do modelo se faz necessário requisitar os valores padrão de alguns componentes.

Para o componente Comunicação, é necessária a declaração do Bluetooth e do Wi-Fi.

```
1 {  
2   "bluetooth": {  
3     "protocol": "BLE v4.2",  
4     "range": { "value": "60", "unity": "m" },  
5     "velocity": { "value": "1", "unity": "Mbps" }  
6   }  
7 }
```

```
1 {  
2   "wifi": {  
3     "protocol": "IEEE 802.11 b/g/n",  
4     "range": { "value": "100/180", "unity": "m" },  
5     "velocity": { "value": "150", "unity": "Mbps" }  
6   }  
7 }
```

Para o componente Memória, é necessária a declaração das memórias Flash, SRAM e ROM.

```
1 {  
2   "Flash": {  
3     "model": "",  
4     "type": "SSD",  
5     "capacity": { "value": "4", "unity": "MB" },  
6     "velocity": { "value": "", "unity": "" }  
7   }  
8 }
```

```
1 {
2   "sram": {
3     "model": "",
4     "type": "SSD",
5     "capacity": { "value": "520", "unity": "KB" },
6     "velocity": { "value": "", "unity": "" }
7   }
8 }
```

```
1 {
2   "rom": {
3     "model": "",
4     "type": "SSD",
5     "capacity": { "value": "448", "unity": "KB" },
6     "velocity": { "value": "", "unity": "" }
7   }
8 }
```

Para o componente CPU.

```
1 {
2   "cpu": {
3     "model": "ESP 32S",
4     "cores": "1",
5     "lithography": { "value": "5", "unity": "mm" },
6     "clock": { "value": "240", "unity": "MHz" },
7     "temperature": {
8       "min": "-40",
9       "max": "150",
10      "unity": "degreeCelsius"
11    }
12   }
13 }
```

Para o componente Bateria não são declarados valores, pois o Plante Box não possui bateria, sendo ligado diretamente a uma fonte da alimentação de corrente contínua. Entretanto, outros componentes similares a ele podem utilizar baterias, e, por isso, foi adicionado um componente de bateria para exemplificar tal situação.

```
1 {
2   "battery": {
3     "model": "",
4     "type": "",
5     "capacity": { "value": "", "unity": "" },
```

```
6     "temperature": { "min": "", "max": "", "unity": "" }
7   }
8 }
```


APÊNDICE D – Telemetria de Requisição

Corresponde às mensagens que devem ser enviadas do Digital Twin à Entidade Física, requisitando a execução de uma função.

Para a LD proposta existem as seguintes formas de requisição.

Desligando o atuador.

```
1 { "ld": "da" }
```

Desligando os sensores.

```
1 { "ld": "ds" }
```

Ligando o atuador.

```
1 { "ld": "la" }
```

Ligando os sensores.

```
1 { "ld": "ls" }
```

Lendo os dados dos sensores.

```
1 { "l": {} }
```

Medindo o consumo de CPU, memória e bateria.

```
1 { "mc": {} }
```

Medindo a temperatura de operação da CPU e bateria.

```
1 { "to": {} }
```

Para a DTDL existem as seguintes formas de requisição.

Desligando o atuador.

```
1 { "ligarDesligar": "da" }
```

Desligando os sensores.

```
1 { "ligarDesligar": "ds" }
```

Ligando o atuador.

```
1 { "ligarDesligar": "la" }
```

Ligando os sensores.

```
1 { "ligarDesligar": "ls" }
```

Lendo os dados dos sensores.

```
1 { "ler": {} }
```

Medindo o consumo de CPU, memória e bateria.

```
1 { "medirConsumo": {} }
```

Medindo a temperatura de operação da CPU e bateria.

```
1 { "temperaturaOperacao": {} }
```

APÊNDICE E – Telemetria de Resposta

Corresponde às mensagens que devem ser enviadas da Entidade Física ao Digital Twin, como resposta / confirmação de execução de uma função.

Para a LD proposta existem as seguintes formas de requisição.

Resposta para: ligar/desligar atuador/sensores.

```
1 { "ld": { "es": 1 } }
```

Resposta para: ler os dados dos sensores.

```
1 { "l": { "t": 25, "u": 70, "us": 60, "c": 30, "l": 55 } }
```

Resposta para: medir o consumo de CPU, memória e bateria.

```
1 { "mc": { "cc": 12, "cb": 30, "cmf": 70 } }
```

Resposta para: medindo a temperatura de operação da CPU e bateria.

```
1 { "to": { "tc": 40, "tb": 35 } }
```

Para a DTDL existem as seguintes formas de requisição.

Resposta para: ligar/desligar atuador/sensores.

```
1 { "ligarDesligar": { "executado com sucesso": 1 } }
```

Resposta para: ler os dados dos sensores.

```
1 {
2   "ler": {
3     "temperatura": 25,
4     "umidade": 70,
5     "umidade do solo": 60,
6     "chuva": 30,
7     "luz": 55
8   }
9 }
```

Resposta para: medir o consumo de CPU, memória e bateria.

```
1 {  
2   "medirConsumo": {  
3     "consumo da cpu": 12,  
4     "consumo da bateria": 30,  
5     "consumo da mem ria flash": 70  
6   }  
7 }
```

Resposta para: medindo a temperatura de operação da CPU e bateria.

```
1 {  
2   "temperaturaOperacao": {  
3     "temperatura da cpu": 40,  
4     "temperatura da bater": 35  
5   }  
6 }
```