

UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
ESCOLA DE ENGENHARIA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

PEDRO HENRIQUE MORGAN PEREIRA

**INTEROPERABILITY MIDDLEWARE
FOR IIOT GATEWAYS BASED ON
INTERNATIONAL STANDARD
ONTOLOGIES AND STANDARDIZED
DIGITAL REPRESENTATION**

Porto Alegre
2022

PEDRO HENRIQUE MORGAN PEREIRA

**INTEROPERABILITY MIDDLEWARE
FOR IIOT GATEWAYS BASED ON
INTERNATIONAL STANDARD
ONTOLOGIES AND STANDARDIZED
DIGITAL REPRESENTATION**

Thesis presented to Programa de Pós-Graduação em Engenharia Elétrica of Universidade Federal do Rio Grande do Sul in partial fulfillment of the requirements for the degree of Master in Electrical Engineering.

Minor: Control and Automation

ADVISOR: Prof. Dr. Edison Pignaton de Freitas

Porto Alegre
2022

PEDRO HENRIQUE MORGAN PEREIRA

**INTEROPERABILITY MIDDLEWARE
FOR IIOT GATEWAYS BASED ON
INTERNATIONAL STANDARD
ONTOLOGIES AND STANDARDIZED
DIGITAL REPRESENTATION**

This thesis was considered adequate for obtaining the degree of Master in Electrical Engineering and approved in its final form by the Advisor and the Examination Committee.

Advisor: _____

Prof. Dr. Edison Pignaton de Freitas, UFRGS
PhD from Halmstad University, Sweden, and Federal
University of Rio Grande do Sul, Brazil

Examination Committee:

Prof. Dr. Ivan Müller, UFRGS
PhD from Federal University of Rio Grande do Sul, Brazil

Prof. Dr. João Cesar Netto, UFRGS
PhD from Université Catholique de Louvain, Belgium

Prof. Dr. Maria Claudia Reis Cavalcanti, IME
PhD from Federal University of Rio de Janeiro, Brazil

Coordinator of PPGEE: _____

Prof. Dr. Sérgio Luís Haffner

Porto Alegre, Maio 2022.

***"Shoot for the moon. Even if you miss,
you'll land among the stars."***

Norman Vincent Peale

ABSTRACT

Recent advances in the areas of microelectronics, information technology, and communication protocols have made the development of smaller devices with greater processing capacity and lower energy consumption. This context contributed to the growing number of physical devices in industrial environments which are interconnected and communicate via the internet, enabling concepts such as Industry 4.0 and the Industrial Internet of Things (IIoT). These nodes have different sensors and actuators that monitor and control environment data. Several companies develop these devices, including diverse communication protocols, data structures, and IoT platforms, which leads to interoperability issues. In IoT scenarios, interoperability is the ability of two systems to communicate and share services. Therefore, communication problems can make it unfeasible to use heterogeneous devices, increasing the project's financial cost and development time. In an industry, interoperability is related to different aspects, such as physical communication, divergent device communication protocols, and syntactical problems, referring to the distinct data structure. Developing a new standard for solving these matters may bring interoperability-related drawbacks rather than effectively solving these issues. Therefore, to mitigate interoperability problems in industrial applications, this work proposes the development of an interoperability middleware for Edge-enabled IIoT gateways based on international standards. The middleware is responsible for translating communication protocols, updating data from simulations or physical nodes to the assets' digital representations, and storing data locally or remotely. The middleware adopts the IEEE industrial standard ontologies combined with assets' standardized digital models. As a case study, a simulation replicates the production of a nutrient solution for agriculture, controlled by IIoT nodes. The use case consists of three devices, each equipped with at least five sensors or actuators, communicating in different communication protocols and exchanging data using diverse structures. The performance of the proposed middleware and its proposed translations algorithms were evaluated, obtaining satisfactory results for mitigating interoperable in industrial applications.

Keywords: Interoperability, IoT, IIoT, Industry 4.0, Communication Protocols, Ontology, AAS.

RESUMO

Devido a recentes avanços nas áreas de microeletrônica, tecnologia da informação, e protocolos de comunicação tornaram possível o desenvolvimento de dispositivos cada vez menores com maior capacidade de processamento e menor consumo energético. Esse contexto contribuiu para o crescente número desses dispositivos na indústria que estão interligados via internet, viabilizando conceitos como Indústria 4.0 e Internet das Coisas Industrial (IIoT). Esses nós possuem diferentes sensores e atuadores que monitoram e controlam os dados do ambiente. Esses equipamentos são desenvolvidos por diferentes empresas, incluindo protocolos de comunicação, estruturas de dados e plataformas de IoT distintos, acarretando em problemas de interoperabilidade. Em cenários de IoT, interoperabilidade, é a capacidade de sistemas se comunicarem e compartilharem serviços. Portanto, esses problemas podem inviabilizar o uso de dispositivos heterogêneos, aumentando o custo financeiro do projeto e seu tempo de desenvolvimento. Na indústria, interoperabilidade se divide em diferentes aspectos, como comunicação e problemas sintáticos, referentes à estrutura de dados distinta. O desenvolvimento de um padrão industrial pode trazer mais desvantagens relacionadas à interoperabilidade, em vez de resolver esses problemas. Portanto, para mitigar problemas relacionados a intoperabilidade industrial, este trabalho propõe o desenvolvimento de um middleware de interoperável para gateways IIoT baseado em padrões internacionais e ontologias. O middleware é responsável por traduzir diferentes protocolos de comunicação, atualizar os dados dos ativos industriais por meio de suas representações digitais, esses armazenados localmente ou remotamente. O middleware adota os padrões ontológicos industriais da IEEE combinadas com modelos digitais padronizados de ativos industriais. Como estudo de caso, são realizadas simulações para a produção de uma solução nutritiva para agricultura, controlada por nós IIoT. O processo utiliza três dispositivos, cada um equipado com pelo menos cinco sensores ou atuadores, por meio de diferentes protocolos de comunicação e estruturas de dados. O desempenho do middleware proposto e seus algoritmos de tradução foram avaliados e apresentados no final do trabalho, os quais resultados foram satisfatórios para mitigar a interoperabilidade em aplicações industriais.

Palavras-chave: Interoperabilidade, IoT, Indústria 4.0, Protocolos de comunicação, Ontologia, AAS.

LIST OF FIGURES

Figure 1: IoT Solution	15
Figure 2: IoT generations concept.	20
Figure 3: Industrial revolution.	23
Figure 4: IoT interoperability taxonomy.	30
Figure 5: Using AAS to transform a I4.0 asset to an I4.0 component. . .	32
Figure 6: General structure of an AAS.	33
Figure 7: A system communication network via I4.0-compliant commu- nication.	40
Figure 8: IoT-Lite Ontology.	43
Figure 9: Semantic gateway as service architecture.	45
Figure 10: System Overview.	49
Figure 11: Middleware’s architecture overview.	50
Figure 12: Proposed communication block.	51
Figure 13: Proposed data storage block.	52
Figure 14: Proposed user interface block.	52
Figure 15: Part of the proposed IIoT ontology.	53
Figure 16: AAS sensors submodules.	56
Figure 17: Sequemce UML diagram for message exchange.	57
Figure 18: Proposed JSON message structure.	58
Figure 19: Proposed IDL data type.	58
Figure 20: Piping and instrumentation diagram for nutrient solution mod- ule	60
Figure 21: State machine diagram of the NSM	61
Figure 22: Use Case Assets Grouping	62
Figure 23: Description part of the project’s approach.	63
Figure 24: Execution part of the project’s approach.	64
Figure 25: Actual System Overview.	66
Figure 26: IIoT Ontology Classes in Protégé software.	68
Figure 27: IIoT Ontology Abstract Subclasses in Protégé software.	69
Figure 28: IIoT Ontology Physical Subclasses in Protégé software.	69
Figure 29: Screenshot of AAS_A240 in SiOME.	70
Figure 30: Screenshot of AAS_P230 in SiOME.	70
Figure 31: Screenshot of OPC UA server in UaExpert Software.	71
Figure 32: Ontology JSON config file.	72
Figure 33: Ontology YAML config file.	73

Figure 34: Screenshot of the developed SCADA like software.	74
Figure 35: NodeRED block flow for creating and updating SVG data. . . .	75
Figure 36: NodeRED block flow for executing scripts in terminal.	76
Figure 37: Screenshot of the IIoT ontology OOPS! results.	78
Figure 38: Screenshot of the Case study Simulation executing in the de- veloped SCADA like.	79
Figure 39: Simulation data from 12th experiment execution in influxDB. .	80
Figure 40: Simulation data from 12th experiment in influxDB using data filtering.	83
Figure 41: Average gateway's CPU usage during each experiment.	84
Figure 42: Average gateway's RAM usage during each experiment.	84

LIST OF TABLES

Table 1:	Comparison of the main related works.	48
Table 2:	Ontologies used for the development of the IIoT ontology . . .	55

LIST OF ABBREVIATIONS

AAS	Asset Administrator Shell
AI	Artificial Intelligence
AMQP	Advanced Message Queuing Protocol
API	Application Programming Interface
COAP	Constrained Application Protocol
CORA	Core Ontology for Robotics and Automation
CPS	Cyber-Physical Systems
CPU	Central Processing Unit
CSV	Comma-Separated Values
DDS	Data Distribution Service
DOB	DDS-OPC UA Bridge
DT	Digital Twin
EC	Electrical Conductivity
HMI	Human-Machine Interface
HTTP	HyperText Transfer Protocol
IDL	Interface Definition Language
IDTA	Industrial Digital Twin Association
IEEE	Institute of Electrical and Electronics Engineers
IIoT	Industrial Internet of Things
IoS	Internet of Services
IoT	Internet of Things
IRI	Internationalized Resource Identifier
I4.0	Industry 4.0
JSON	JavaScript Object Notation
LoRa	Long Range

MOB MQTT-OPC UA Bridge

MoM Message-Oriented Middlewares

MQTT Message Queuing Telemetry Transport

M2M Machine-to-machine communication

NOSQL Not only SQL

NSM Nutritional solution modules

OOPS! Ontology Pitfall Scanner!

OPC UA Open Platform Communications Unified Architecture

ORArch Ontology for Robotic Architectural

OS Operating System

OSI Open System Interconnect

OWL Web Ontology Language

PLC Programmable logic controller

QOS Quality of service

RAM Random access memory

RFID Radio-Frequency Identification

ROA Ontology for Autonomous Robotics

ROS2 Robot Operating System 2

SCADA Supervisory control and data acquisition

SGS Semantic Gateway as Service

SiOME Siemens OPC UA Modeling Editor

SNS Social Networks Services

SQL Structured query language

SSN Semantic Sensor Network

SUMO Suggested Upper Merged Ontology

SVG Scalable Vector Graphics

TCP Transmission Control Protocol

UFRGS Federal University of Rio Grande do Sul

UML Unified Modeling Language

WSN Wireless Sensor Network

WWW World Wide Web

W3C World Wide Web Consortium

YAML YAML Ain't Markup Language

XML Extensible markup language

XMPP Extensible Messaging and Presence Protocol

LIST OF ALGORITHMS

1	Translator algorithms flowchart.	73
---	--	----

CONTENTS

1	INTRODUCTION	15
1.1	Motivation	17
1.2	Objectives and Contribution	18
1.3	Dissertation Organization	19
2	BACKGROUND CONCEPTS REVIEW	20
2.1	Internet of Things	20
2.1.1	Middleware	21
2.1.2	Gateway	22
2.2	Industry 4.0	23
2.2.1	Cyber Physical Systems	24
2.3	Industrial Internet of Things	25
2.4	Digital Twins	27
2.5	Communication Protocols	27
2.5.1	MQTT	28
2.5.2	DDS	29
2.5.3	OPC-UA	29
2.6	Interoperability	30
2.7	Asset Administrator Shell	32
2.8	Ontologies	33
2.8.1	OWL - Web Ontology Language	35
2.8.2	Protégé	35
3	ANALYSIS OF THE STATE OF THE ART	37
3.1	Communication Protocol Translator	37
3.2	Digital Representation of Industrial Assets	39
3.3	Semantic Technologies in Automation	42
3.4	Discussion	46
4	PROPOSED IIOT INTEROPERABILITY MIDDLEWARE	49
4.1	Proposal Overview	49
4.2	Proposed Middleware Architecture	50
4.2.1	Communication Block	50
4.2.2	Data Storage Block	51
4.2.3	User Interface Block	51
4.3	Developed Ontology	52

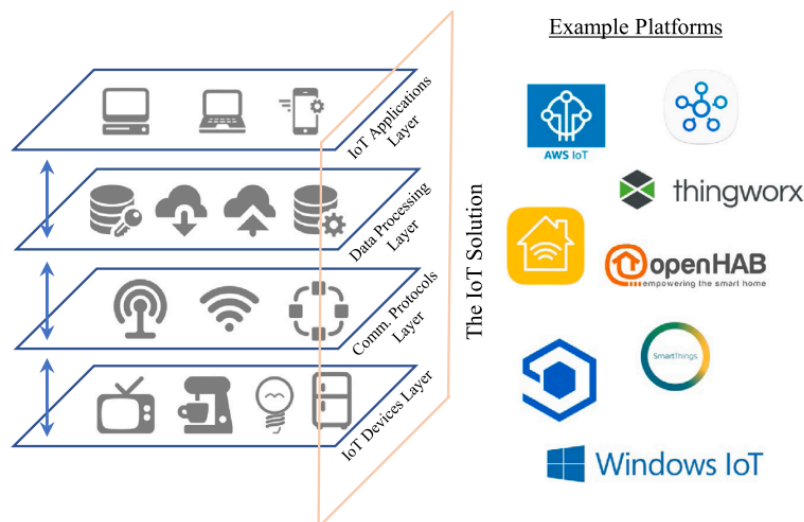
4.4 AAS Submodels	55
4.5 Communication Translators	56
4.6 Proposed standardization	57
4.6.1 Proposed topic terminology	57
4.6.2 Proposed data structure	57
5 CASE STUDY	59
6 IMPLEMENTATION DETAILS	63
6.1 Selected technologies	63
6.1.1 Description Part	63
6.1.2 Execution Part	64
6.2 System Architecture Overview	66
6.3 IIoT Ontology	67
6.4 AAS	68
6.5 Communication Translators	71
6.6 SCADA	74
7 EXPERIMENTS AND RESULTS	77
7.1 Ontology Evaluation	77
7.2 Use case simulation	78
7.3 Simulation's conformance to the use case	79
7.4 Gateway Performance	81
7.5 Execution time for protocol translations	81
8 CONCLUSIONS AND FUTURE WORKS	85
REFERENCES	87

1 INTRODUCTION

Recent advances in the areas of microelectronics, information technology, and communication protocols have made the development of smaller devices with greater processing capacity and lower energy consumption. This context contributed to the growing number of physical devices interconnected and communicated via the Internet in our quotidian life and industrial environments, enabling concepts such as Internet of Things (IoT), Industrial Internet of Things (IIoT), and Industry 4.0.

The IoT concept was proposed in 1999 by Kevin Ashton to describe a system composed of physical devices connected to the Internet using sensors for monitoring real-life application data (GUBBI et al., 2013). The first IoT application used Radio-Frequency Identification (RFID) tags attached to the Internet for identifying, counting, and tracing specific objects in corporate supply chains (SHENG et al., 2013). However, IoT is more than RFID tags; it is an extended and expanded system network based on the Internet. These systems can accomplish interaction between different things, machines, and humans to solve industrial or even everyday needs within time critical applications (WANG et al., 2021).

Figure 1: IoT Solution



Source: Adopted from (BABUN et al., 2021).

Fig.1 depicts an overview architecture of an IoT system and its components/layers. This solution demands synchronization among many IoT devices that communicate using diverse communication protocols and data processing capabilities (BABUN et al., 2021). The solution has four interdependent layers: (1) IoT Devices, (2) Communication Protocols, (3) Data Processing, and (4) IoT Applications (Fig.1 (BABUN et al., 2021)). The lowest layer comprehends the IoT devices that monitor and control the environmental data using specific sensors and actuators. The upper layer is the communication protocol that enables communication between devices and gateways, cloud servers, or other devices while reducing power consumption and increased reliability. These protocols differ in their interaction models, i.e., request-reply (REST HTTP and CoAP) and publish-subscribe (Message Queuing Telemetry Transport [MQTT] and Data Distribution Service [DDS]) (DIZDAREVIĆ et al., 2019). At the system's top, the application layer permits the IoT device's physical data analysis and provides a user interface for managing the devices' data.

IoT has been an essential topic in the technology industry, smart cities, home automation, and healthcare services in the last few years. Using advancements in computing power, electronics miniaturization, and network interconnections, IoT, offers new capabilities for reducing cost and time in essential applications (ROSE; ELDRIDGE; CHAPIN, 2015). This technology increases the availability of environment information along the production value chain using networked nodes equipped with heterogeneous sensors and actuators.

With each new industrial age, technological advances have had a fundamental impact on increasing the productivity of enterprises. The German industrial sector first proposed the so-called fourth industrial revolution, Industry 4.0, in 2011, encompassing automation and data exchange technologies (LIAO et al., 2017). The digital transformation process is one of the characteristics of this revolution, which takes place by incorporating four base technologies: the IoT, cloud computing, big data, and artificial intelligence (AI) (FRANK; DALENOGARE; AYALA, 2019). Today, Industry 4.0 is the new stage for manufacturing companies that combine these base technologies to create cyber-physical systems (CPS), providing the integration of the company's processes (BENITEZ; AYALA; FRANK, 2020). The use of CPS enables the development of autonomous systems, monitor physical processes by creating a virtual copy of the physical world that makes decentralized decisions (HERMANN; PENTEK; OTTO, 2016), and allows communication with one another, increasing industrial efficiency, productivity, safety, and transparency.

Another perspective in the industrial sphere relies on using not only CPS but also embedded systems, cloud computing, edge computing, and the generic technologies associated with the smart factory (EMMRICH et al., 2015). Combining multiple connected assets in manufacturing environments that operate as part of a more extensive system or system of systems empowers the definition of a smart manufacturing enterprise (CONWAY, 2016). The smart manufacturing enterprise is essentially an industrial environment monitored and controlled using the IIoT concept, which aims to connect industrial assets like

engines, power grids, and sensors to the cloud over a network (HELMIO et al., 2017). Therefore, the IIoT notion is a system comprising networked smart objects, including cloud or edge computing platforms, CPS, that "enables real-time, intelligent, and autonomous access, collection, analysis, communications, and exchange of process, product and/or service information, within the industrial environment, to optimize overall production value." (BOYES et al., 2018).

There are several potential benefits of using these technologies in the industry: decreasing production cost, development time, energy consumption and improving productivity, storing and tracking goods, and service delivery. Nevertheless, along with the advantages came uncertainties and challenges, such as infrastructure, communications, interfaces, protocols, interoperability, and standards (LI; XU; ZHAO, 2015).

Interoperability definition by IEEE is the ability of a system or component to exchange information, among others, and use the exchanged information for accomplishing its goal (RADATZ; GERACI; KATKI, 1990). The interoperability issues in IoT have different perspectives due to its high heterogeneity. The diverse elements of IoT, Industry 4.0, and IIoT, such as its devices, communication protocols, services, and applications, must cooperate to accomplish shared objectives. The leading IoT interoperability perspectives are device, networking, syntactic, semantic, and platform interoperability (NOURA; ATIQUZZAMAN; GAEDKE, 2019). Each deals with interoperability issues at a different level, but all are important to ensure an interoperable system.

1.1 Motivation

The interoperability concept is increasingly present in automation systems due to the growing number of physical devices used in industrial environments interconnected and communicating via the Internet. These industrial assets are developed by several companies and are heterogeneously modeled, such as based on different communication protocols, data structures, and IoT platforms, leading to interoperability issues. According to the European project Unify IoT (AAZAM; ZEADALLY; HARRAS, 2018), more than 300 IoT platforms are developed annually in the current market.

Several concepts are being studied and proposed to reduce the interoperability problems in an industrial manner, such as semantic-related technologies. These semantic models make it possible to create a common language, using the aspects of a system and their relationships for the various components, and ensure that the parties involved can understand the information exchanged. Enabling machine-to-machine communication (M2M) and allowing a more significant contribution between humans and machines (MAYER et al., 2017). Some semantic models developed for this domain in the literature, but there is a need to evolve these models, adding new concepts as the system and technologies grow. An essential item to ensure complete integration is maintaining a reliable representation of the system, including its sensors and actuators.

Also, due to the use in the last decades of CPS in industrial environments, digital representation of assets is increasingly present in the industry, benefiting and allowing an understanding of information between different systems and components. These representations provide semantical meta-data of the physical devices, including their main features and system characteristics (NAGORNY et al., 2018), i.e., communication protocols and data structures. The digital models have a pivotal role in establishing communication among I4.0 Components and managing interoperability between the applications and the manufacturing systems (YONG; LEE; LAZARUS, 2021). These and new contributions can leverage the adoption of emerging and innovative concepts, leading industries with traditional methods to use current technologies and make them even more competitive in the market.

1.2 Objectives and Contribution

Using a middleware based on semantic models can contribute as a solution to ensure system interoperability from the lowest IoT to its highest layer (Fig.1). Enabling systems communication and, above all, that they can have the same understanding of certain information. Therefore, this work proposes a middleware for Edge-enabled IIoT gateways (PAPCUN et al., 2020) to mitigate interoperability problems in the context of Industry 4.0 and IIoT. The approach adopts international standard ontologies to represent system elements combined with standardized digital representation using the Asset Administration Shell (AAS) to integrate multiple devices' data. Three of the interoperability perspectives presented in (NOURA; ATIQUZZAMAN; GAEDKE, 2019) are treated more prominently in this work: device, syntactical and semantic interoperability.

Nevertheless, for the developed work to be an adopted approach for an organization is crucial that it is viable and brings tangible benefits to the business. The main contributions of this dissertation are:

- A general Industry 4.0/IIoT oriented ontology based on international Standards (IEEE 1872-2015) and worldwide utilized ontologies. The developed ontology allows the description of different industrial systems to follow the exact specifications, reducing possible human errors;
- Representation of the assets' most relevant information in the digital world using AAS. In this way, an asset's digital version is created (digital twin);
- A middleware oriented to reducing interoperability problems in industrial environments by combining the developed IIoT ontology and AAS use. The middleware enables the communication between industrial assets with different communication protocols and permits heterogeneous information understanding from divergent sensors and actuators using specific data structures.
- A Supervisory Control and Data Acquisition (SCADA) like system for

users/engineers to control the simulation and monitor devices' data using the Node-RED development tool.

1.3 Dissertation Organization

This work has 8 chapters. The first is the Introduction. In this chapter, there is a contextualization of the theme and the objectives of the work.

Chapter 2 presents essential concepts for the development and understanding of research, such as the concept of IoT, Industry 4.0, and IIoT.

Chapter 3 presents the works in the literature related to the research topic. The end of the chapter presents a discussion of related results and gaps that this work will address.

Chapter 4 presents the proposed middleware details for the software architecture and for the IIoT ontology developed in this project.

Chapter 5 presents the a case study in agriculture selected for evaluating the developed middleware an ontology.

Chapter 6 presents the implementation of the interoperability middleware, describing the specific softwares and tools in the context of IIoT and Industry 4.0 and how they were used for its development.

Chapter 7 presents the simulation experiments and the analysis of the obtained results.

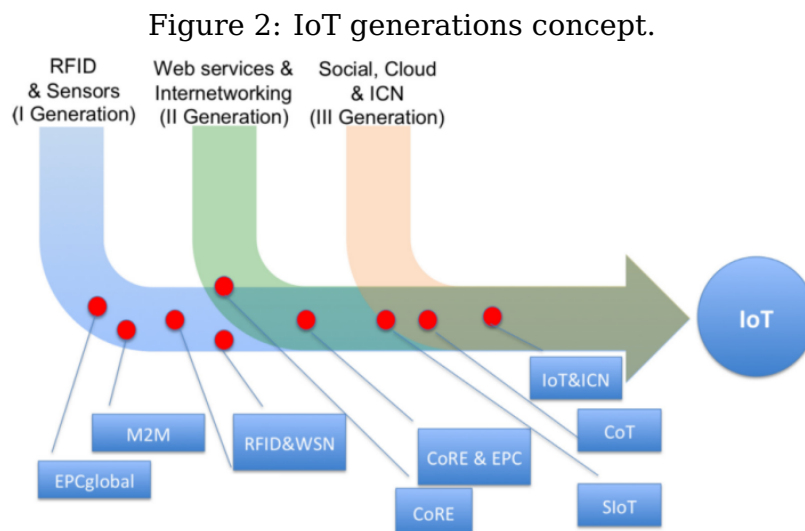
Chapter 8 concludes the work by highlighting contributions and future work.

2 BACKGROUND CONCEPTS REVIEW

This chapter reviews the main concepts of the Internet of Things, Industry 4.0, Ontologies, Asset Administrator Shell, and insight into the tools used for this work.

2.1 Internet of Things

The Internet of Things was first used in 1999 in a presentation on Supply Chain (ASHTON, 2009). This term became famous for its success with RFID tags attached to the Internet for identifying, counting, and tracing specific objects in corporate supply chains (KORTUEM et al., 2010). Throughout technological development, the concept of IoT has evolved; its evolution has three generations (Fig2 (BABUN et al., 2021)) (ATZORI; IERA; MORABITO, 2017).



Source: Adopted from (BABUN et al., 2021).

The first generation is related to RFID tags to identify assets and the use of different sensors to perceive the environment (ATZORI; IERA; MORABITO, 2017). In addition to setting a unique identification for each device, the device can share its data through the Internet Protocol (IP) (DEERING; HINDEN et al., 1998), allowing communication with other nodes. The Internet-enabled communication between devices, machine to machine (m2m) (HOLLER et al.,

2014), and permitted people to communicate with one another or even with machines.

M2M refers to solutions that allow communication between devices in a specific application via wired or wireless communication. It is a technology that enables machines to communicate without human intervention, automating data transfer between intelligent elements, producers, and consumers of data (HOLLER et al., 2014). The central concept of IoT is connecting several nodes(things) through the Internet, which aligns with the idea of M2M. Furthermore, seeking to combine real objects so they can connect, communicate and interact with a person using the web.

The second generation provides ordinary objects with the ability to connect to the Internet like any other host (ATZORI; IERA; MORABITO, 2017). The concept of the Web of Things emerged (GUINARD; TRIFA, 2009), where devices can connect to the World Wide Web (WWW) (BERNERS-LEE, 1999) as resources. In addition, arises the concept of Social Networks Services (SNS) (RICHTER; KOCH, 2008), in which devices can interact with social media getting even closer to the user (ATZORI; IERA; MORABITO, 2017).

At last, the third generation encompasses more modern concepts and brings people closer to the objects involved in an IoT system. This generation is usually called the Future Internet, which will explore cloud computing technology and focus on people, content, and services (ATZORI; IERA; MORABITO, 2017). Additionally, things can participate in communities, with similar interest groups collaborating in their actions.

To make these concepts viable, it is of paramount importance that there is a communication standard, communication protocol, and a way of representing information between systems, machines, and people to reduce ambiguous interpretations of the exchanged data.

Finally, According to Kagermann (KAGERMANN et al., 2013), the integration of the IoT and the Internet of Services (IoS) in the manufacturing process initiated the fourth industrial revolution. Due to concepts such as M2M, autonomous machines allowed an industrial environment to achieve more outstanding production, cost reduction, and increased safety (HOLLER et al., 2014).

2.1.1 Middleware

Middleware is a set of software and technologies that can assist hide the complexity and heterogeneity of underlying hardware and network platforms, simplify system resource management, and improve application execution predictability (WANG et al., 2008). They are used for different areas, such as distributed systems (BAKKEN, 2001), WSN (WANG et al., 2008) and IoT (RAZ-ZAQUE et al., 2015).

In the IoT context, a middleware combines heterogeneous computing and communications devices supporting interoperability within the varied applications and services operating on these devices; a middleware can provide standard applications and make application development easier (BANDYOPADHYAY et al., 2011). A fully functional IoT middleware needs to integrate

essential components such as Wireless Sensor Network (WSN), RFID, M2M communications, and Supervisory control and data acquisition (SCADA) to support the envisioned diverse application domains (ZHOU, 2012) (PERERA et al., 2013).

Middlewares serve as an abstraction of the system or hardware, allowing users to concentrate on the application without being distracted by orthogonal problems at the system or hardware level (NEELY; DOBSON; NIXON, 2006). It bridges applications, the operating system, and the network communications layers, facilitating and coordinating cooperative processing. Since many IoT applications are data-centric, data management middleware provides applications' data gathering, processing, and storage capabilities (WANG et al., 2008). Some of the middlewares' main characteristics are:

- Supply appropriate system abstractions so that the application programmer can concentrate on the application logic rather than the implementation details at a lower level.
- Offers reusable code services, such as code update, and data services, such as data filtering, so that the application programmer can deploy and run the app without worrying about sophisticated and time-consuming methods.
- Provides efficient resource services, such as power management, to assist the programmer in network infrastructure management and adaptation. It also facilitates system integration, monitoring, and security.

2.1.2 Gateway

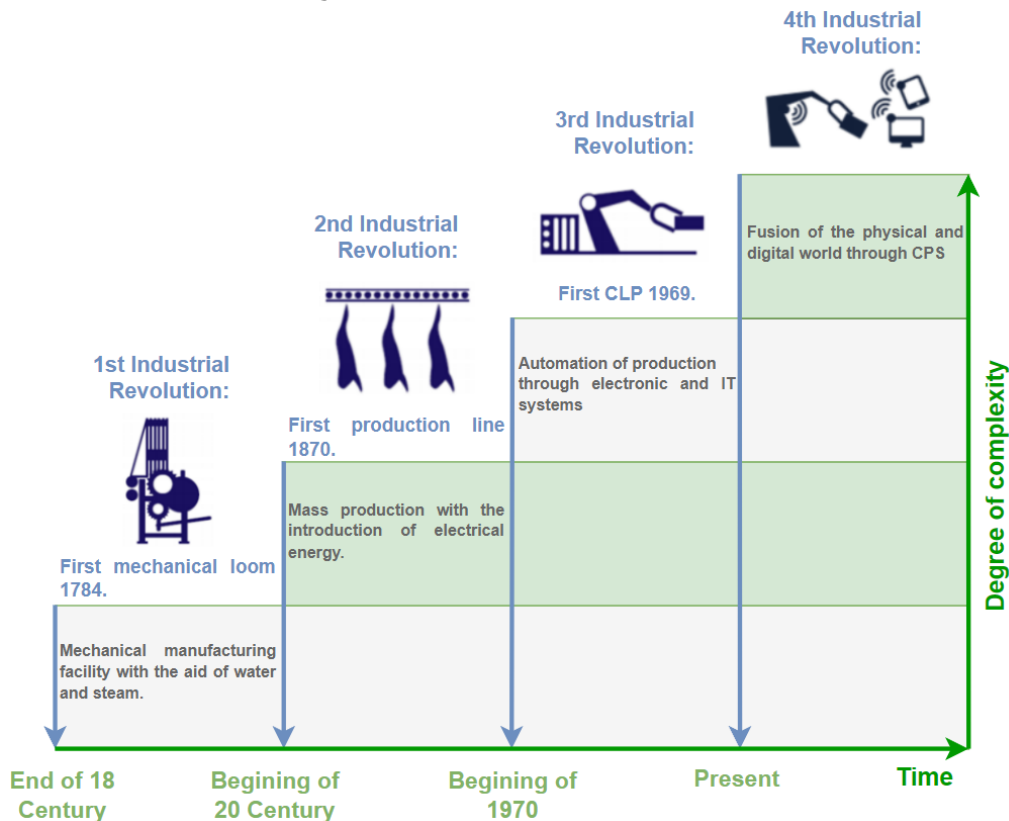
An IoT gateway is an intelligent component that works with an IoT platform (LEA; BLACKSTOCK, 2014). It usually communicates via the Internet between a network of M2M devices and remote peers (e.g., clients). The gateway's primary purpose is to address the heterogeneity of multiple sensor networks to their communication to mobile communication networks or the Internet (KANG; CHOO, 2018). As gateways link the Internet and IoT devices, it provides connections among these devices and cloud serves, enabling intelligent big data analysis and data-driven decision-making (KANG; CHOO, 2018).

Gateways typically have more processing power, memory, and capacity than IoT devices (BUTUN et al., 2020). Gateways come in various shapes and sizes, ranging from small embedded Linux systems with radio modules and some data preprocessing to traditional wireless routers used in the consumer market to small battery-operated radio transceivers with an uplink to communication satellites (TRAN; MISRA, 2019). This equipment supports various communication protocols and data types and can realize the conversion of data formats communicated between an array of nodes and translate communication protocols to permit interoperability (ZHONG; ZHU; HUANG, 2015). It has, also the function of safety protection and preventing outside intrusion using authentication methods. (ZHONG; ZHU; HUANG, 2015)

2.2 Industry 4.0

As the technology evolved, different creations marked the progress of industrial dynamics. They led to new revolutions: the steam engine, electricity, digital revolution, and cyber-physical systems, as illustrated in Figure 3.

Figure 3: Industrial revolution.



Source: Adapted from (KAGERMANN et al., 2013).

In 2011, the fourth industrial revolution took place, proposed by the German industrial sector. This revolution uses automation and data exchange technologies using concepts of CPS, IoT, and Cloud computing (LIAO et al., 2017). The German Federal Government supported the initiative and announced the first recommendations for implementing Industry 4.0 in 2013 (KAGERMANN et al., 2013). Consequently, several countries used the German advance as an initiative to develop new technological proposals. In the USA, the Industry started the process of developing smart factories (WANG et al., 2016), in China, in 2015, the "Made in China 2025" (WÜBBEKE et al., 2016) project, and the European Union created several other initiatives.

I4.0 introduces a series of paradigm shifts that change how the Industry works and how the product reaches the consumer. Using CPS and M2M techniques to create virtual copies of the physical devices and enabling communication between machines allows an entire system to make decentralized decisions autonomously (HERMANN; PENTEK; OTTO, 2016). Allowing manufacturing to become more flexible by having concomitant decentralized processes with resource efficiency and using individualized products with short product

development periods (LASI et al., 2014). With the exponential advancement of computer capabilities, an immense amount of digitized information, and innovation strategies, several industries can put into practice the concept of industry 4.0. These industries save resources, increase profitability, reduce waste, predict errors and delays, speed up production, intervene quickly in case of production problems, etc. (BRETTEL et al., 2014).

Finally, Hermann (HERMANN; PENTEK; OTTO, 2016) also states that I4.0 has six design principles. These principles guide companies to identify and implement the scenarios foreseen in Industry 4.0: interoperability, virtualization, decentralization, service orientation, and modularity.

2.2.1 Cyber Physical Systems

A CPS combines and coordinates computational and physical elements in an identical system. CPS integrates the performance capability of the physical world and the intelligence of the cyber world to add new capabilities to real-world physical scenarios (LEE; BAGHERI; KAO, 2015). Operations are monitored, coordinated, controlled, and integrated into these systems by a computing and communication core. These systems intend to incorporate objects from the physical world and information systems, performing interconnections and information sharing.

A CPS aggregates computing, communication, and storage resources to monitor and control physical world entities reliably, securely, efficiently, and in real-time (SHA et al., 2008). CPS is an emerging and enabling technology to face future challenges regarding industrial automation. In this domain, CPS comprises intelligent machines, storage systems, and production capacity to perform information exchange autonomously, trigger actions, and control each other independently (JAZDI, 2014).

Gorecky (GORECKY et al., 2014) addresses the relationship between humans and CPS and says that this relationship happens with the support of interfaces. The man will have the role key to determining the production strategy. Questions may then arise about the autonomy and decision-making power on the part of man. Furthermore, workers do not will need to have a fixed place of work, as they will have access to information via mobile networks.

According to Hermann (HERMANN; PENTEK; OTTO, 2016), integrating processes computational and physical computers and networks for monitoring and controlling processes are essential features of cyber-physical systems. And the integration of Embedded systems and the Internet of Things serve as the foundation for cyber-physical systems. Embedded systems can perform operations with other systems on networks closed. The Internet of Things is the interconnection of physical objects through global or local data networks. With the Internet of Things, objects can interact with each other.

With their modular structures, CPS and IoT facilitate the vision and execution of “Smart Factories” (ZUEHLKE, 2010). They allow physical processes to be monitored, virtual copies of the physical world, and they make decentralized decisions. Combining both concepts makes it possible for systems to

intercommunicate and cooperate with humans, and through cloud computing, services are offered and used by participants across the factory.

2.3 Industrial Internet of Things

The Industrial Internet of Things (IIoT) definition by (BOYES et al., 2018) is: "A system comprising networked smart objects, cyber-physical assets, associated generic information technologies and optional cloud or edge computing platforms, which enable intelligent, and autonomous access, collection, analysis, communications, and exchange of process, product and/or service information, within the industrial environment, to optimize overall production value. This value may include; improving product or service delivery, boosting productivity, reducing labor costs, reducing energy consumption, and reducing the build-to-order cycle."

In a simplified way, the IIoT is a network of physical objects, systems platforms, and applications containing embedded technology to communicate and share intelligence, the external environment, and people. IIoT is concerned with heavy-duty processes such as manufacturing, monitoring, and so on, instead of IoT, which focuses on everyday items with Internet connectivity. As a result, assets used in IIoT applications are more precise and long-lasting (BUTUN et al., 2020).

Unlike conventional applications, industrial environments create more precise, continuous, and sensitive data. Also, it requires high security and reliability for data exchange, and most of these applications are time-critical. Unlike conventional applications, industrial environments create more precise, continuous, and sensitive data. Also, it requires high security and reliability for data exchange, and most of these applications are time-critical. Even though IoT and IIoT rely on similar concepts, such as data management, network, security, and cloud, the first is not recommended for high security and time-critical applications (BOYES et al., 2018).

The vast volume of data generated necessitates data streaming, big data, machine learning, or artificial intelligence technologies. Besides, the volume of generated data, scalability, and specialized data management techniques are the primary differences between technologies (BUTUN et al., 2020).

Companies hope to boost productivity, competitiveness, efficiency, stability, and safety by implementing IIoT in production lines or other industrial projects. Application areas for IIoT range from smart cities to precision agriculture, smart traffic to smart grid (the future of the electric grid), and so on (BOYES et al., 2018).

The IIoT system can monitor, collect, analyze, and intelligently change the behavior or environment without human intervention (BOYES et al., 2018). Nonetheless, this system not only emphasizes the nonexistence of human intervention (MUMTAZ et al., 2017) but also focuses on interoperability between manufacturing systems to trigger automation and synchronization for closed ecosystems (YOUNAN et al., 2020).

IIoT takes benefits of IoT communications in business applications focusing

on interoperability between machines (YOUNAN et al., 2020). The IIoT gateways need to support a variety of protocols. As a result, a distinguishing aspect of IIoT environments is the integration of protocols, standards, and buses from many technologies (FIGUEROA-LORENZO; AÑORGA; ARRIZABALAGA, 2020).

According to a keynote speech delivered by Tom Bradicich, the seven principles of the IIoT are (BUTUN et al., 2020):

- Big amount of analog data: Many sensors generate analog data, which is digitalized before being processed, analyzed, or stored.
- The IIoT's devices are always connected. There are three significant advantages to this: (1) It is feasible to monitor real-time. (2) Constant monitoring can aid in the distribution of software and firmware upgrades and fixes. (3) The value of linked devices encourages individuals and businesses to buy products.
- Real-time data streaming: Many safety systems are used in the industry, continually producing data. In the case of a nuclear power plant, safety is crucial to operations. Monitoring necessitates real-time data streaming because a data delay could result in catastrophic occurrences. As a result, real-time data aggregation and streaming are critical.
- Data insights: "What are you trying to achieve?" is the question that data insights (Spectrum of Value) in IoT aims to answer.
- Time-to vs. depth-of-insight trade-off: It is equivalent to the immediacy-of knowledge compared to the depth of knowledge. For instance, while monitoring or analyzing nuclear power plant data, immediate attention is required, whereas analyzing scientific experiment data (data by CERN or NASA) can take years to uncover scientific problems.
- Visibility from Big Data: Once data has been collected and stored in a big data environment, it should be accessible anytime analysis or other tasks are required.
- Edge computing: Data center-class computing and analytics carried out to the edge (latency, bandwidth, cost, security, duplication, reliability, corruption, compliance, and data sovereignty).

The IIoT aspires to create intelligent manufacturing goods and smart factories with close customer and business partner connectivity. With the rise of IIoT, Industry 4.0 has emerged as a subset that focuses on digitizing and integrating all physical operations throughout the entire business (SENGUPTA; RUJ; BIT, 2020).

The digital connectivity (promoted by integrating protocols, standards, and buses) of industrial machinery and industrial equipment with any physical asset with an IT platform is a unique advance that sets a social precedent for business opportunities. This convergence of the physical world and cyber on an industrial scale allows operations for several applications (FIGUEROA-LORENZO; AÑORGA; ARRIZABALAGA, 2020).

IIoT is an enabling technology for Digital Twin (DT) by providing the functionalities of remote sensing of the sensors and remote controlling of the actuators. IIoT provides the means of communication to the link between the actual factory machinery and the digital twin equivalent at the command center of the factory (BUTUN et al., 2020).

2.4 Digital Twins

Artificial intelligence (AI) and big data processing are two of the emerging technologies that are combined to create the Digital Twin (DT) technology. Due to the lack of a defined definition for DT as of yet, it can be described in a variety of ways depending on its purpose and scope.

Michael Grieves (GRIEVES, 2014) defines that Digital Twin concept model is divided into three parts: a) a real space containing physical things, b) a virtual space containing virtual objects, and c) the data and information flow connections or links that connect the virtual and real space objects. Element "c" facilitates data exchange, allowing the convergence and synchronization of data flow from virtual to real space.

For Barricelli (BARRICELLI; CASIRAGHI; FOGLI, 2019) DTs can be defined as (physical and/or virtual) machines or computer-based models that simulate, emulate, mirror, or "twin" the existence of a physical thing. These things can be an ordinary object, a process, a human, or a human-related trait. Each DT has a link to its physical twin by a defined unique key that identifies and differentiates each other allowing the creation of a bijective relationship. A DT monitors, controls, and optimizes its operations and functions by following the lifecycle of its physical twin.

2.5 Communication Protocols

The need to exchange data between different devices or systems made it necessary to formulate standard descriptions, formats, and rules for expressing these data in a common language. These rules created consistency and universality for sending and receiving messages understandable by two or more communication system entities.

There are several communication protocols, each specialized for an application, such as facilitating everyday activities using smart devices, IoT, to robust industrial environments. These protocols define the syntax, semantics, and synchronization of communication. Each protocol can even check possible errors and recover data using specific methods. Following standard systems with the same protocol can transmit and process data from files to analog and digital signals.

The Ethernet has been the most commonly used communication technology within the office domain for several decades, which entails decreasing components prices through mass production. The lower prices lead to an adaptation of this technology in industrial applications (INDUSTRIAL COMMUNICATION PROTOCOLS, 2009). Ethernet has become ubiquitous and cost-effective with

its evolution, with standard physical links and increased speed. Industrial Ethernet protocols use a modified Media Access Control (MAC) layer to achieve low latency, and deterministic responses (LIN; PEARSON et al., 2013).

However, the development of IoT and I4.0 technologies enables multiple devices equipped with sensors and actuators to work together, exchanging essential data using wireless communication. Therefore, wireless protocols in the industry have increased, allowing data monitoring and control using WSN (BRETTEL et al., 2014). These protocols differ in their interaction models, i.e., request-reply and publish-subscribe (DIZDAREVIĆ et al., 2019).

The request-reply communication model is one of the most basic communication paradigms, based on widespread client/server architectures. The client must request information from the server, which receives the request message, processes it, and answer the client with the corresponding data. Therefore, centralizing the system data in a server that answers requests from multiple clients. The two most known protocols based on the request/reply model are REST HTTP and CoAP (DIZDAREVIĆ et al., 2019).

The publish-subscribe model is a distributed, asynchronous, loosely coupled communication between data generators and destinations. Widely used in current industrial applications in the form of numerous publish-subscribe Message-Oriented Middlewares (MoM) (JIA et al., 2014). This model has several subscribers different from clients and does not have to request information from a server. Instead of asking for information, the subscriber must subscribe to the broker, the central entity in this architecture. The broker is responsible for filtering and forwarding incoming messages between its publishers and subscribers. Finally, the publishers provide information whenever specific events occur to a topic by sending this data to the broker. All subscribers will receive data sent to the broker on the subscribed topics. For these reasons, a publish-subscribe interaction model is an event-based architecture (HINZE; SACHS; BUCHMANN, 2009).

2.5.1 MQTT

The Message Queue Telemetry Transport (MQTT) is a lightweight wireless communication protocol that follows the publish-subscribe paradigm (PROFANTER et al., 2019). It was developed by IBM (LOCKE, 2010), with its latest version, MQTT v3.1, adopted for IoT by the Organization for the Advancement of Structured Information Standards (OASIS) (BANKS; GUPTA, 2014). MQTT is an open message protocol that mainly focuses on a small code footprint and low bandwidth usage while handling high latency or lousy network connections (PROFANTER et al., 2019).

This protocol is one of the most prominent protocol solutions in constrained environments for its minimal message header and low power requirements compared to other messaging protocols. Besides, it runs on top of the TCP transport protocol, ensuring its reliability and reliability (DIZDAREVIĆ et al., 2019). With the publish-subscribe, the exchanged data is stored by the broker, meaning that small devices report data to the broker and do not need to keep it themselves (PROFANTER et al., 2019). Also, the broker can send commands

for controlling these devices, even by grouping data hierarchically.

2.5.2 DDS

The Data Distribution Service (DDS) is a real-time data-centric interoperability standard based on the publish-subscribe interaction model (DIZDAREVIĆ et al., 2019). The Object Management Group standardizes (OMG) this protocol, which specifies it as one of many communication protocols used in industry sectors such as railway networks, air traffic control, medical services, military, and industrial automation (HARISH, 2015).

Unlike other publish-subscribe protocols, DDS is decentralized since it does not depend on the broker component. The publishers and subscribers can communicate asynchronously as peers through the data bus. Without a broker, the probability of system failures is lower since it doesn't rely on a central/single point that concentrates all data, creating a more reliable system (DIZDAREVIĆ et al., 2019).

The DDS protocol runs on top of the UDP transport protocol as default but also supports TCP. The protocol also offers a broad set of QoS policies (over 20 QoS as defined by the standard). Whenever a node exchanges data, a QoS policy is set for its topic, which the publishers set. However, the subscribers control the behavior when receiving data. It is essential to point out that the communication only occurs when both publishers and subscribers match through topics (same name, type, and a compatible QoS) (DIZDAREVIĆ et al., 2019).

The extensive set of QoS parameters, e.g., durability, lifespan, presentation, reliability, and deadlines (PROFANTER et al., 2019), is a prominent feature of DDS, allowing the discovery of distributed remote entities, data delivery, data availability, time, and resource utilization (INGLÉS-ROMERO et al., 2017). Therefore, DDS is an essential solution for IoT-based environments for its decentralized publish/subscribe architecture and its support for implementation in both powerful and constrained devices (AL-FUQAHA et al., 2015).

2.5.3 OPC-UA

The Open Platform Communications Unified Architecture (OPC UA) is a service-oriented machine-to-machine communication protocol that allows for standardized read and writes access to current data in automation devices via communication drivers (MAHNKE; LEITNER; DAMM, 2009). The OPC UA is an IEC standard known as IEC 62541.

The primary use cases are interfaces for industrial automation applications like Human-Machine Interfaces (HMI) and SCADA systems providing a cross-platform communication protocol while using an information model to describe the transferred data. Its primary strength is the semantic description of the address space model with various companion specifications, which extend the basic semantic descriptions for multiple domains like Programmable logic controller (PLC), robotics, or computer vision (PROFANTER et al., 2019).

OPC uses the client-server approach for information exchange. OPC UA Client can read and write one or more attributes of Nodes maintained into the

OPC UA server's address space. The clients can access data without understanding the whole model exposed by a standard specific protocol allowing the addition of new protocols in the future (MAHNKE; LEITNER; DAMM, 2009). Applications consuming and providing data can be both client and server.

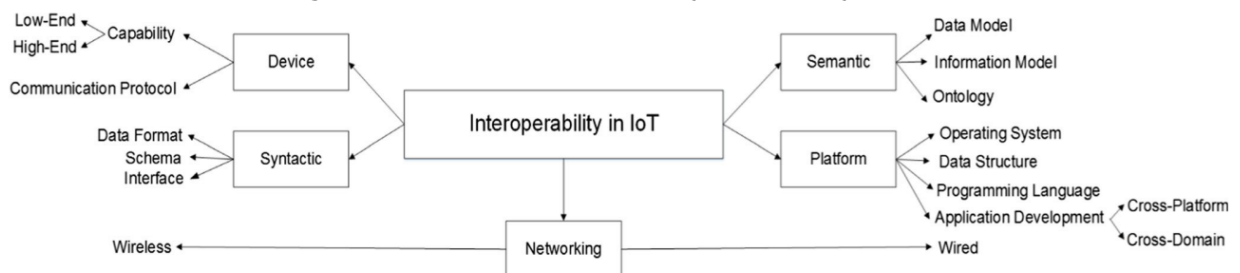
OPC UA has two main components transport mechanisms and data modeling. For the transport mechanisms, OPC UA uses the Transmission Control Protocol (TCP) as its transport protocol for high-performance intranet communication and accepts internet standards like Web Services, Extensible markup language (XML), and HyperText Transfer Protocol (HTTP) for firewall-friendly internet communication. The data modeling defines the rules and base building blocks necessary to expose an information model with OPC UA. That represents enhanced concepts like describing state machines used in different information models (MAHNKE; LEITNER; DAMM, 2009).

2.6 Interoperability

Interoperability is the capacity of various computerized products or systems to connect promptly and share data without encountering any restrictions (BUTUN et al., 2020). According to a McKinsey (MANYIKA et al., 2015) analysis, 40 percent of the Internet of Things' potential is unavailable due to a lack of interoperability.

Variations in IoT applications bring unique challenges to integration for its device architecture, infrastructure, connectivity protocols, platforms, and data models (NEGASH; WESTERLUND; TENHUNEN, 2019). Additionally, the resource constraints in most IoT devices make the challenge more restrictive. IIoT standard initiatives aim at facilitating interoperability, simplifying development, easing implementation, and identifying possible threats and vulnerability issues.

Figure 4: IoT interoperability taxonomy.



Source: Adopted from (NOURA; ATIQUZZAMAN; GAEDKE, 2019).

The IoT interoperability taxonomy comprises five different perspectives (Figure 4 (NOURA; ATIQUZZAMAN; GAEDKE, 2019)): device interoperability, networking interoperability, syntactic interoperability, semantic interoperability, and platform interoperability (NOURA; ATIQUZZAMAN; GAEDKE, 2019).

- Device interoperability refers to enabling the integration and interoperability of heterogeneous devices with various communication protocols

and standards. It concerns (i) the exchange of information between heterogeneous devices and heterogeneous communication protocols and (ii) the ability to integrate new devices into any IoT platform.

- Network interoperability deals with mechanisms that allow end-to-end communication between systems via multiple networks (networks of networks). The network interoperability level manages challenges such as addressing, routing, resource optimization, security, Quality of service (QoS), and mobility support due to the IoT's dynamic and heterogeneous network environment.
- Syntactical interoperability, also known as data exchange interoperability (TOLK, 2004), refers to the divergence of the format and the data structure used in any exchanged information or service between heterogeneous IoT system entities. Syntactic interoperability problems arise when the sender's encoding rules are incompatible with the receiver's decoding rules or vice-versa, which leads to mismatching message parse trees.
- Semantic interoperability refers to the exchange of information, data, and knowledge in a meaningful way for both entities (TOLK, 2004). Data can be measured in various units and can reflect different information meanings. Devices have diverse rules for understanding the meaning of information content and create a domain-specific information model, known as the semantic model. As a result of the semantic incompatibility between data and information models, systems cannot interact dynamically and autonomously.
- Platform interoperability issues in IoT arise due to the availability of diverse operating systems (OSs), programming languages, data structures, architectures, and access mechanisms for things and data. A cross-platform IoT application can access different IoT platforms and integrate data from various platforms. After cross-platform interoperability is enabled, cross-domain interoperability (e.g., health, home, transport, etc.) can be achieved in which divergent platforms within heterogeneous domains are federated to build horizontal IoT applications.

With a greater focus on industrial applications, Butun (BUTUN et al., 2020) presents two sorts of IIoT interoperability:

- The cross-layer interoperability, also known as heterogeneity, orchestrates the Open System Interconnect (OSI) layers in a seamless and trouble-free manner. Heterogeneity in the IoT means implementing different communication protocols, data formats, and technologies (YOUNAN et al., 2020). The variety of IoT devices and hardware-based, inflexible cellular infrastructures make efficient connectivity even more difficult. It is causing cross-layer communication functionalities between heterogeneous IoT devices and cellular systems.

- The cross-system interoperability refers to the capacity to maintain networks and systems running despite distinct system architectures. An application with several devices communicating based on different technologies can be interoperable using semantic system models to comprehend raw sensor data better, enabling AI-based machines to make autonomous decisions based on simple rules. By specifying complete centralized metadata and translating disparate communication protocols, gateways can assist in mitigating interoperability issues.

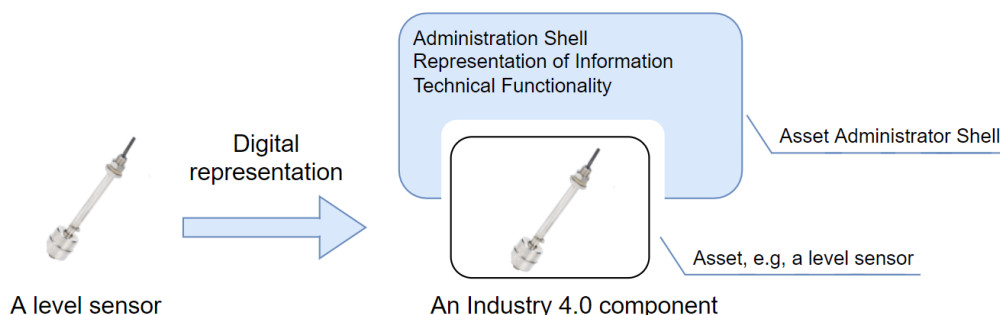
Furthermore, due to the wide range of technological solutions offered by different suppliers, interoperability challenges are bound to develop, mainly if no standard Application Programming Interface (API), or communication protocol, has been officially adopted (DI MARTINO et al., 2018). From the perspective of IoT providers, lack of interoperability means that service providers are bound to a single IoT device or software offered by a single provider, potentially resulting in increased operational costs, product functionality, and stability difficulties (NOURA; ATIQUZZAMAN; GAEDKE, 2019).

2.7 Asset Administrator Shell

With the widespread implementation of Industry 4.0, digital twins (DTs) have become increasingly viable to model on-site manufacturing resources and have improved data interoperability (MINERVA; LEE; CRESPI, 2020).

The Asset Administration Shell (AAS) is a standardized electronic representation of industrial assets (BARNSTEDT et al., 2018) based on the Architecture Model for I4.0 (RAMI 4.0) (SCHWEICHHART, 2016), enabling the implementation of DTs. The use of the AAS facilitates information processing from different machines, making these data discoverable, identifiable, and accessible, thus easing interoperability among the applications of a manufacturing company (LÜDER et al., 2020; YE et al., 2020). AAS supports data communication via OPC UA, an Ethernet-based protocol frequently used in industrial control systems (YE et al., 2022).

Figure 5: Using AAS to transform a I4.0 asset to an I4.0 component.

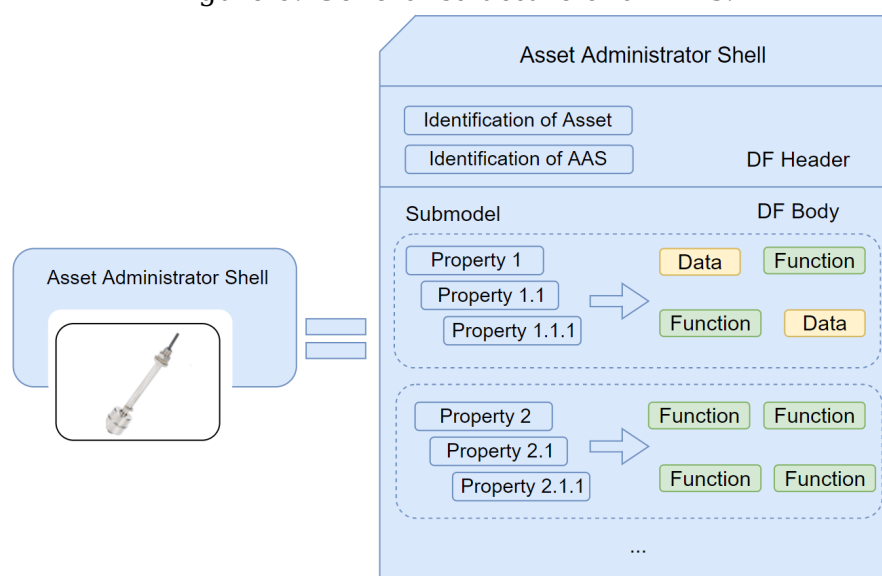


Source: Adapted from (YE et al., 2022).

An AAS transforms a manufacturing asset, an I4.0 asset, into an Industry 4.0 component that can be a module, a device, or a system (GRANGEL-

GONZÁLEZ et al., 2016). The essential elements of an I4.0 Component structurally are the asset and the administration shell (Figure 5) (63088., 2017). Assets are valuable elements of an organization, either physical or nonphysical objects, such as materials and products, devices, machines, software, and digital services have a respective digital version (BADER; MALESHKOVA, 2019). The AAS provides semantical meta-data of a CPS, including functional and non-functional properties: intrinsic information, operational parameters, and technical functionalities (NAGORNY et al., 2018). Using digital models of various aspects enables direct interactions over standardized and secure communication links with other Industry 4.0 Components and managing interoperability between the applications and the manufacturing systems (SAKURADA; LEITÃO, 2020).

Figure 6: General structure of an AAS.



Source: Adapted from (YE; HONG, 2019).

Figure 6 shows the general structure of an AAS. It has a header and a body. The header contains a list of parameters identifying the AAS and the physical asset. The body stores data related to the capabilities of an asset and its operational data (YE; HONG, 2019). The asset's information describes as submodels in the AAS. The AAS may incorporate general submodels (e.g. identification) and also specific submodels (e.g. communication) (YE; HONG, 2019). The submodels contain submodel elements like SubmodelCollection and Properties. A property, a submodel element type, can contain a value representing a physical variable of the asset. It can be of several types as INT, BOOL, or STRING. An example is the current status of a pump (ON / OFF)

2.8 Ontologies

The ontology concept is still somewhat overloaded with several different meanings. The philosophical notion of an 'ontology' has been addressed for

over two thousand years by the expression "what exists?", a systematic explanation of existence, becoming a relevant notion in the 1990s (GÓMEZ-PÉREZ; BENJAMINS, 1999). However, this work approaches the idea of ontologies from a Computer Science point of view.

Guarino (GUARINO; OBERLE; STAAB, 2009) explicit that nowadays, the most relevant ontology definition is from 1998, (STUDER; BENJAMINS; FENSEL, 1998): "An ontology is a formal, explicit specification of a shared conceptualization.". Guarino defines the 'Explicit' term as "refers to the to the fact that all elements of an ontology are explicitly defined, whereas 'formal' means that the ontology specification is given in a language that comes with a formal syntax and semantics, thus resulting in machine executable and machine interpretable ontology descriptions."

Ontologies provide a generic way of representing domain knowledge for a common understanding between applications (CHANDRASEKARAN; JOSEPHSON; BENJAMINS, 1999). By using these models is possible to create a shared vocabulary of an area, with the meanings of each term and the relationships between them (GÓMEZ-PÉREZ; BENJAMINS, 1999). Furthermore, an ontology provides the same understanding for people who need to share information in a specific domain, further enabling the basic concepts of the field and the relationships to be machine-interpretable (WANG et al., 2016). Using a familiar vocabulary is critical for communicating between machines or humans (FIORINI et al., 2017). For applications nowadays, using ontologies enables detailed structured data sharing among different devices and allows them to perform reasoning over the shared data.

There are different types of ontologies, such as upper level and domain ontologies. A domain ontology (or domain-specific ontology) represents concepts to a specific domain, such as engineering or biology. These ontologies provide a shared, reusable definitions of domain-wide knowledge, offering an unambiguous formalized representation (EL-GOHARY; EL-DIRABY, 2010). With the aid of a domain ontology, it is possible to recognize attributes in requirements descriptions and provide a semantic foundation for requirement descriptions (KAIYA; SAEKI, 2006).

An upper or top level ontologies enable the semantic integration of domain ontologies and direct the creation of new ontologies. They include broad categories consisting of general terms that support broad semantic interoperability among different domains. Rich definitions and axioms are typically provided for each category in upper level ontologies. Based on the types of entities they include, their theories of space and time, as well as how people relate to space and time, various upper level ontologies offer various distinctions (HOEHNDORF, 2010) (SOWA, 1995).

Some of the benefits of using ontology-based models are as follows (WANG et al., 2016):

- Knowledge sharing: ontologies enable computational entities, such as machines in the industry, to have a standard set of concepts related to this domain.

- Knowledge reuse: it is possible to unite ontologies from different domains to have a larger one involving all concepts. An example is using an ontology with concepts related to units of measure, which is suitable for several application domains.
- Logical inference: it is possible to make logical inferences based on the data obtained from a low-level context, that is, data provided by the sensors. This data goes through a verifying process, which can correct inconsistencies in the context due to failures in acquiring these data.

2.8.1 OWL - Web Ontology Language

The use of ontologies in the Semantic Web context requires a language to describe them, meeting some basic requirements: well-defined syntax and semantics, reasoning support, and expression convenience (ANTONIOU; HARMELEN, 2004). One of the possible languages to be used is the Web Ontology Language (OWL) (ANTONIOU; HARMELEN, 2004) recommended by the World Wide Web Consortium (W3C) (JACOBS, 2001), based on existing languages and standards.

OWL allows the user, in addition to presenting ontologies' information, to process the content of information. This language provides additional vocabulary among machines using formal semantics, facilitating greater machine interoperability of Web content. Representing the meaning of terms in vocabularies and the relationships between those terms (MARTIN et al., 2004). Class, individuals, and properties are fundamental elements for building ontologies with OWL.

Classes in ontologies can be related to the concept of class in object-oriented programming languages. Objects in the real world can be grouped or set with similar characteristics or functionalities (parameters)(LACY, 2005).

A property is a relationship of individuals, OWL distinguished properties into two categories that an ontology builder may want to define. Object properties link individuals to individuals; Data type properties link individuals to data values (W3C, 2009). Properties provide attributes to individuals that use methods to access and reuse object data by being able to relate a property to multiple classes (LACY, 2005).

Individuals represent instances of the classes described in the ontology. These instances are similar to objects in object-oriented programming, differing in that they do not have associated methods. Individuals can represent both virtual concepts, and physical objects (LACY, 2005).

2.8.2 Protégé

Protégé (UNIVERSITY, 2001) is a consolidated, open-source tool that assists users in constructing sizeable electronic knowledge bases (NOY et al., 2003) developed at Stanford University. The tool enables developers to create, edit and visualize domain ontologies.

This tool can help users create other applications to acquire knowledge from specialists in particular areas that define essential concepts and relationships. Protégé automatically constructs a visual knowledge-acquisition system

helping develop and facilitate new users' understanding. Nevertheless, it is possible to create an extension, or a new tool for a specific use, based on Protégé (MUSEN, 1989).

The Protégé's user community is populated. In 2003 (NOY et al., 2003) there were more than 7000 registered users and an active discussion list with more than 1200 subscribers. Even today, most ontologies developed for different areas, such as medicine, engineering, and history, use this tool.

3 ANALYSIS OF THE STATE OF THE ART

This chapter presents the analysis of the state of the art that meets the purpose of this work in the areas of IoT, semantic models, and works on the fourth industrial revolution.

3.1 Communication Protocol Translator

Middleware solutions are deployed on the Cloud (remote data centers) or the Edge Network (nearby IoT Gateways). Cloud-related middlewares are typically based on deep analytics; otherwise, Edge related can support near-real-time applications. Most middlewares are hosted on the Cloud, which may introduce communication latency, preventing time-critical applications in industrial environments.

Budakoti (BUDAKOTI; GAUR; LUNG, 2018) proposes a lightweight Middleware at the Edge network. It is an event-based Middleware, publish and subscribe mechanism that supports multiple protocol bindings. It must be able to do multiprotocol translations to provide a horizontal unified data integration scheme for interoperability for applications on the Cloud and IoT devices on the Edge. The lightweight Middleware is deployed on an IoT gateway based on Raspberry Pi 3, implementing interoperability between devices transmitting data over WiFi, Bluetooth, and Serial by IoT applications built over protocols like HTTP, Constrained Application Protocol (COAP), MQTT, and Advanced Message Queuing Protocol (AMQP). Supporting local data analysis and can also be deployed on Cloud for deep analytics. The developed Middleware uses a single database for storing sensors' data in JavaScript Object Notation (JSON) from different protocols, based on structured query language (SQL) or not only SQL (NoSQL). As a result, clients can access the same resources regardless of which protocol they choose for communication.

An IoT gateway, the Samsung Artik 1020 development kit, was presented by Castellanos (CASTELLANOS et al., 2021) for reducing interoperability problems. The gateway connects wireless nodes for simultaneous data transmission using ZigBee, WiFi, and Bluetooth protocols. It also implements a flexible algorithm to translate sensors' data into a uniform format, JSON, for information to a cloud server over the MQTT protocol. However, the proposal is restricted to IoT platforms that use only the MQTT protocol and follow the data formatting presented by the authors, which is not standardized by any

already consolidated service.

Palavras (PALAVRAS et al., 2018) presents another IoT gateway based on a secure multiprotocol integration bridge for the IoT (SeMIBIoT). The gateway can provide hop-by-hop or end-to-end secure communications between an array of heterogeneous nodes and standardized IoT protocols. Its main task is to translate messages between protocols and relay them to the corresponding devices covering the HTTP, Websockets, CoAP, MQTT, and Extensible Messaging and Presence Protocol (XMPP) protocols. The author rely on real-time translation, not using persistent storage. SeMIBIoT defines IDs for each protocol to verify the corresponding translation method for each device. The bridging mechanism acts whenever a device sends a request and, consequently, in the response.

Experiments using physical IoT devices were carried out in Budakoti (BUDAKOTI; GAUR; LUNG, 2018), and Palavras (PALAVRAS et al., 2018) works. Both showed promising results using their bridging mechanism between IoT protocols, the first for MQTT and AMQP and the last for XMPP, MQTT, HTTP, and CoAP. However, since the two works are specific to IoT applications, none of the developed middlewares support industry-specific protocols.

Another IoT protocol interoperability solution is presented by Derhamy (DERHAMY; ELIASSON; DELSING, 2017), a protocol translator for SOA-based systems. It combines two translation techniques, the direct protocol-to-protocol and the establishment of an intermediate protocol. The former has the lowest packet loss per conversion. Still, new translators must be developed with the increasing number of protocols, resulting in a longer development time and greater need for processing power. On the other hand, in the second format, the user must define a protocol as an intermediary, so fewer translators will be needed (number of used protocols - 1). The translator system is autonomous and not coupled to the orchestration system. The experiment results were not compared with translators using different techniques, so it cannot be proved that the proposed one is the most adequate. Furthermore, only HTTP and CoAP protocols were used, neither of which are industry standards.

Ferreira (FERREIRA et al., 2019) describes a 4.0 multiprotocol use case, focusing specifically on LoRaWAN and WiFi connectivity. The case study comprehends two multiprotocol devices, a gateway and a sensor node. The node has an energy metering system based on a commercially available ATM90E26 single-phase meter. Different tests are performed on an industrial ground floor, composed of open spaces and electrical/electronic machinery using the same devices. In one of these, the sensor node sends energy data using WiFi through the MQTT protocol and another through the Long Range technology (LoRa) interface using the LoRaWAN protocol. The functioning of the developed devices has been partially proven since the experiments did not concern multiple nodes communicating with different protocols. Nevertheless, the gateway enables the communication throw devices that use LoRa and WiFi technologies; still, it doesn't allow the translation of application protocols.

Wang (WANG et al., 2022) proposes an integration and intercommunication technology based on the convergence of MQTT and OPC UA. To establish

the communication, the authors use a unified name ID to create mappings from one protocol to another, which is stored on an XML configuration file. In addition, a shared database is used to realize integration between both protocols, allowing reading and writing data, likewise a data acquisition drive that triggers whenever a database data is updated. However, no physical experiments or simulations were carried out to verify the functionality of the proposed translator.

3.2 Digital Representation of Industrial Assets

The core feature of an I4.0 Component is the combination of objects from both the physical world and the information world, offering dedicated functionalities and flexible services inside and outside a network of I4.0 Components. Integrating advanced information and communication technologies (ICTs) and OTs is a popular method for the digitalization process, as it promotes research with CPS technologies.

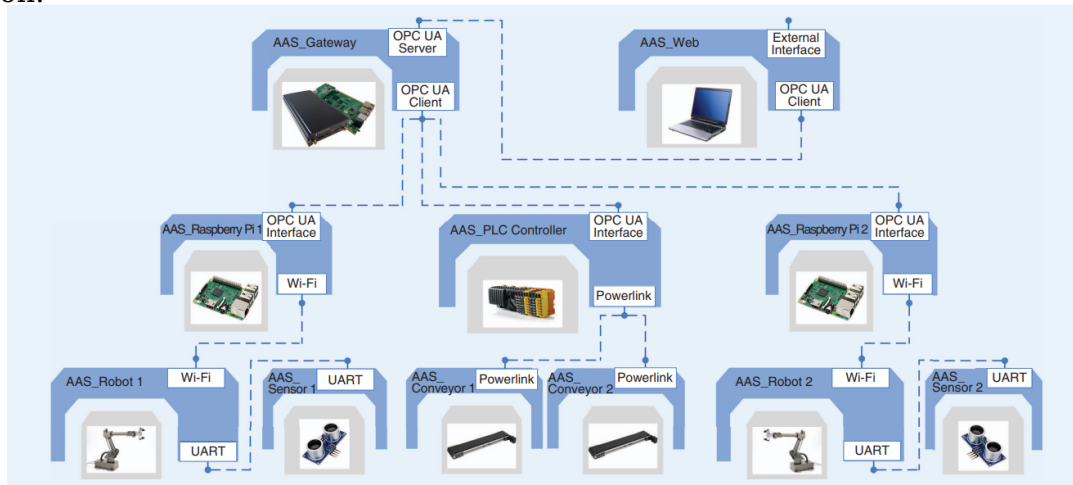
Schroeder (SCHROEDER et al., 2016) presents a methodology to create a high-level model of DT and use its information to enable data exchange between divergent systems. The proposed method comprises three phases: creating a physical device's (any device that provides data exchange) high-level model, extracting the model's data by middleware, and information system development.

A case study was conducted using a modeled and simulated industrial valve to validate the method. For the modeling phase, the authors used AutomationML; the FIWARE middleware was used to collect the models' data. At last, another system requests data using REST API, which is returned as a JSON format. Models allow regular users, without programming knowledge, to model heterogeneous devices' DT, enabling data exchange among systems. It's worth noting that the project's goal (SCHROEDER et al., 2016) was to allow data transmission between different systems using DT models. The tests comprised only one device, and its data was sent only to one system. Since DTs are widely used in industrial applications with numerous devices, it would be essential to validate different models exchanging information with more than one system.

Ye (YE; HONG, 2019) describes the details of the AAS and presents a proof of concept (POC) in a manufacturing demonstration. The demonstration (7 (YE; HONG, 2019)) comprises several assets, such as a web application, a gateway, Raspberry Pi single-board computers, a PLC controller, robots, ultrasonic sensors, conveyors, and their respective AAS.

The boundary of the AAS features a communication interface that provides access to any assets' data and functions within the application using its AAS's submodels. This communication is established by using a submethod for identifying the AAS communication method. The AAS must guarantee the continuous acquisition of runtime data generated by physical assets, such as a linear axis's actual position and rotation speed. Thus, the assets' data will be available in their respective AAS. A control flow generated by the gateway forwards

Figure 7: A system communication network via I4.0-compliant communication.



Source: Adopted from (YE; HONG, 2019).

production tasks to AASs of the field devices. Feedback parameters are eventually loaded back into the database of the AAS web. Although (YE; HONG, 2019) presented a case study with different types of sensors, actuators, and gateway, this was not implemented, evidencing a lack in the literature of an application similar to the one used in IIoT applications.

An adaptable factory contains modular components that can be added, removed, and adjusted according to production needs, thus plug and produce (PnP) of industrial field devices. New field devices or production modules may be integrable without (or with minimal) human intervention at each runtime to perform different tasks (e.g., manufacturing a new product), significantly increasing production flexibility.

Authors of (YE et al., 2020) present a use case for digitizing following PnP methodology via AAS. Three robots from different vendors structure the AAS-based system, each equipped with an I4.0 adaptor. The adaptor adopts an OPC UA server, open62541 toolkit, for each robot [embedded in the Linux operating system (OS) evaluation boards]. The I4.0 adaptors first manage asset runtime data standardized and then send it to the AAS. The AAS of the three robots interacts so that their assets can accomplish a specific system function or task, communicating through the OPC UA protocol. The author uses an I.4 adapter for each asset to establish the OPC UA server and client for each AAS. So that as the number of devices in a network increases, we will also have an increase in the cost necessary to develop it.

Bouter (BOUTER et al., 2021) presents a methodology to identify and develop AAS's submodules for specific application scenarios, which other researchers should apply to further standardization. Each system comprehends different assets that have divergent functionalities and properties. The physical asset's properties are stored in its AAS using different Submodels linked with its data or functions. These characteristics are thought to express themselves in the AAS DF body, which should adhere to a standardized data for-

mat. New models should be created whenever a standard is unavailable for an application intended to be reused in future works. The use of OWL ontological models is based on Internationalized Resource Identifier (IRI) as globally unique identifiers in common with the AAS for new submodules. Allowing to transparently develop new submodules from ontologies, many of which are taken as standard by different researchers.

The authors rely on a small industry use case to evaluate the proposed model. Developing several AAS submodules for the specific application was necessary, even for a simple application. Emphasizing that a multi-agent is demanded when using the developed method in more robust industrial applications. Bouter's methodology (BOUTER et al., 2021) was developed to establish a new pattern for developing submodules, reusing the models of other authors. However, the developed submodules did not agglutinate already developed in the literature, such as (YE; HONG, 2019) (YE et al., 2020) (IÑIGO et al., 2020).

A case study demonstrates the use of ASSs for individual components in an I4.0 virtual assembly line designed to produce plastic models of cars (ARM et al., 2021). All virtual asset has their AASs, which communicate with each other and negotiate the production priorities and requirements according to a pre-specified set of rules. Each AAS must have the negotiation submodel fully implemented. The AAS communication driver integrates a TCP/IP connection and sends a TCP stream; thus, MQTT-based communication was also employed, exhibiting communication latencies lower than those achieved by the OPC UA. The researchers discarded using the MQTT protocol since when performing tests with a more significant number of AAS, it obtained results much lower than those achieved with OPC UA.

The limitations of this study reside in two aspects. The first refers to the number of AAS used in the case study that was fewer than expected for industrial applications. Finally, as in (YE et al., 2020) each AAS must have both a server and an OPCUA client that can cause several problems with network scaling and when using devices that lack processing capabilities.

Iñigo (IÑIGO et al., 2020) presents a case study on applying the AAS in an industrial context. The use case considers a plant composed of a robotic arm, a grinding machine, and a semantic harmonization layer. Two AASs (Robotic Arm AAS and Grinding Machine AAS) were developed and tested using a semantic integrator for experimentally validated interoperability. The demonstrator has been transformed to consider the RoboticArm as an Asset. A Raspberry Pi was used as the OPCUA server that published the Administration Shell of the RoboticArm asset instance. The developed architecture provided a protocol translator that exchanges information using UMATI (Universal Machine Tool Interface), standardizing how machine tools share information over OPCUA. In conclusion, the AAS has been validated to represent heterogeneous industrial assets and their digital twins and enable interoperability between them in a manufacturing plant.

Although the authors of (IÑIGO et al., 2020) have presented experimental tests using industrial devices, the experiments were performed with only two

machines, a small number compared to the number of devices in an actual industrial application. In addition, both machines use the same communication protocol, OPC-UA, so it was impossible to assess whether the developed integrator allows multi-protocol communication.

3.3 Semantic Technologies in Automation

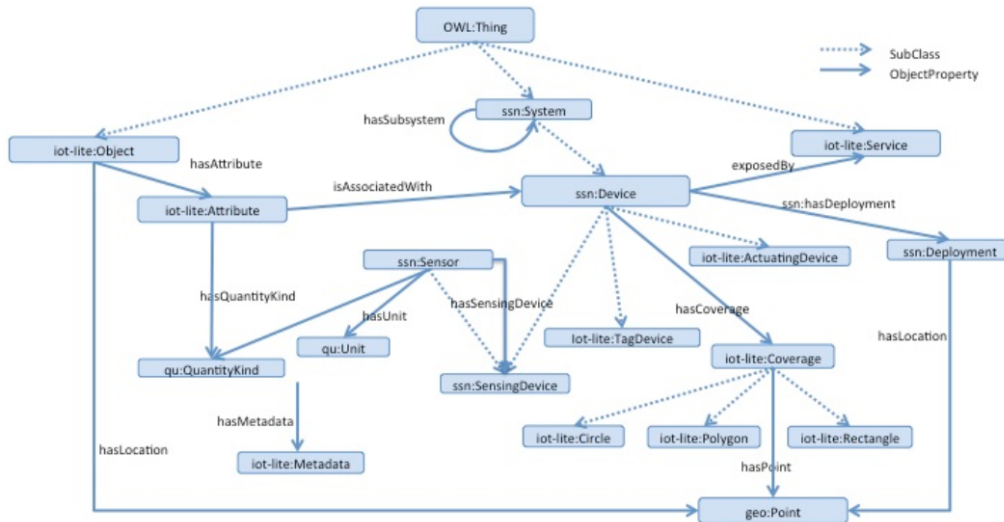
In industrial informatics, various technical systems exchange information, including engineering, manufacturer, process control, production planning, and maintenance, to optimize the industrial process (INTEROPERABILITY, 2017). Each system has its tasks, functionality, models, and semantics. In an industrial environment, the research and deployment of Semantic Technologies, semantic links, and ontologies have increased considerably in recent years. These are the potential technological solutions to address interoperability problems for IoT environments.

The Semantic Sensor Network Ontology (COMPTON et al., 2012), also known as SSN, was first created by the W3C Semantic Sensor Network Incubator Group. The SSN ontology has been employed in a web of things architecture, sensing for manufacturing, and representing persons and personal gadgets as sensors. It can define physical sensors and their characteristics: accuracy, measurement capabilities, outputs, observation value, and feature of interest. Knowledge Engineers can utilize the SSN ontology as a foundation for future projects by establishing equivalence relations to link significant aspects of sensors specified in this ontology. It is worth mentioning that this ontology is the basis of several others developed for research and projects until today, used to create IoT and IIoT-specific ontologies.

Bermudez-Edo (BERMUDEZ-EDO et al., 2016) used the SSN ontology as a basis to describe a lightweight semantic IoT model, IoT-Lite (Figure 8). The IoT-Lite ontology can be used as the core part of a semantic model; it is designed for large-scale and provides means to update and change the semantic annotations. Depending on the applications, semantic modules can be updated to provide additional domain-specific concepts and relationships. This ontology describes IoT-related ideas for developing IoT and smart city applications and services. IoT-Lite does not intend to be a complete ontology for the IoT; it explains critical IoT concepts, such as tag and actuating devices, sensory data, location, and type. These descriptions allow interoperability and discovery of sensory data in heterogeneous IoT platforms.

Agarwal (AGARWAL et al., 2016) discusses FIESTA-IoT, a unified ontology to resolve interoperability issues. This ontology is developed for the EU H2020 FIESTA-IoT project, which seeks to use semantic-based methodologies to allow interoperability among multiple orthogonal testbeds. To avoid overloading the ontology domain, the FIESTA-IoT links existing IoT solution concepts from SSN, IoT-lite, M3-lite taxonomy, Time, DUL, and WGS84 ontologies. It ensures better interoperability with existing semantic-based IoT platforms, projects, and standardizations. The ontology is a solution to achieve semantic interoperability among heterogeneous testbeds, focusing on describing its resources

Figure 8: IoT-Lite Ontology.



Source: Adopted from (BERMUDEZ-EDO et al., 2016).

(i.e., sensors, tags, etc.) and their observations.

The ontologies developed by Compton (COMPTON et al., 2012), Bermudez (BERMUDEZ-EDO et al., 2016), and Agarwal (AGARWAL et al., 2016) can be used as a starting point for developing new ontologies for IoT technologies, as they share multiple classes and attributes. They can be adapted for Industry 4.0 and IIoT applications to address typical interoperability concerns. None of the three suggested ontologies address the type and structure of shared data and communication devices or specify the communication protocols used by nodes and devices.

Kumar (KUMAR et al., 2019) discusses the current state of ontologies for I4.0, including existing ontological frameworks and efforts to standardize ontologies in the field. Developing an industry-related ontology requires interoperable M2M communication and describes autonomous robotics requirements. This way, it is possible to create concepts semantically standardized by different equipment in the manufacturing domain.

Several industry-related ontologies have the definitions used by the IEEE 1872-2015 (IEEE, 2015) standard as their fundamental principles. Nevertheless, the newly developed ontologies include I4.0-specific ontological notions to create an industry's ontological standard. The Core Ontology for Robotics and Automation (CORA) (PRESTES; FIORINI; CARBONERA, 2014) was developed within the IEEE 1872-2015. As the name suggests, CORA is a core ontology for robotics applications that outlines essential commitments for creating well-defined models of robots. The Ontology for Autonomous Robotics (ROA) (OLSZEWSKA et al., 2017) defines robotic notions as fundamental for Autonomous Robotics, including behavior, function, goal, and task definitions. It reuses ontologies such as the Suggested Upper Merged Ontology (SUMO) upper ontology, the CORA core, and specialized ontologies. The Ontology for Robotic Architectural (ORArch) delves into hardware and software concepts and how they might be expressed in mixed architecture descriptions. Its goal

is to make it possible to explain numerous architectural viewpoints of a single robot that mixes hardware and software.

The ontologies presented by Kumar (KUMAR et al., 2019) are all tailored to industrial applications involving robotics. A robot can be a programmable machine containing actuators and sensors in ontologies, as specified in CORA ontology, although specific attributes of these devices are omitted. Furthermore, robot communication standards, such as its communication protocols, are not addressed, which may make it impossible to establish communication.

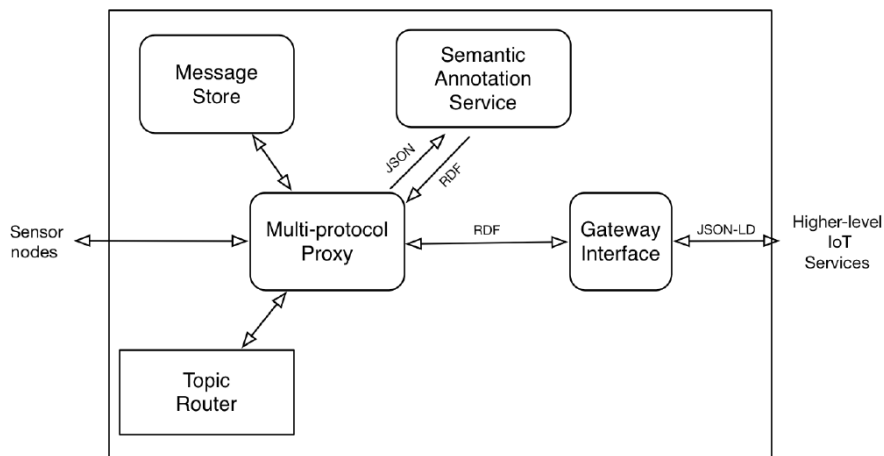
Steinmetz (STEINMETZ et al., 2018) presents how to map industrial elements into a semantic model, integrating Industry 4.0 objects easily and understandably for the users. These semantic models can support services and allow data exchange between virtual and physical assets through an IoT middleware. The semantic model used was an I4.0 ontology composed of other ontologies found in the literature that bring complementary concepts for IoT application. Such as the IoT Lite ontology (BERMUDEZ-EDO et al., 2016) which is an extension of Semantic Sensor Network (SSN) ontology (COMPTON et al., 2012), that describes crucial concepts of IoT, such as devices, sensors, attributes, and so on. Nevertheless, the author added new classes related to the type of information visualization to make it possible to model how data should be presented to different types of users. The model was used to generate interfaces to the FIWARE IoT middleware and validated through an industrial valve actuator use case. The valve circuit was instrumented with different sensors to monitor its data, like battery level, temperature, pressure, and torque. Although the author has experimented with various sensors, they were not treated as different devices. Therefore, the developed ontology does not handle problems related to communication between other devices with heterogeneous communication protocols, such as the one presented in this work.

Besides, some authors combine semantic techniques and protocol translators in IoT and industrial environments. A novel middleware solution that enables effective communication between OPC UA and DDS communication protocols is proposed by Endeley (ENDELEY et al., 2019), allowing transparent communication between devices that utilize these protocols without changing each framework's native rules. Both protocols are leading industry standards in the industry4.0 environment, OPC UA for the request/reply and DDS for the publish/subscribe pattern. The proposed translator requires a method of mapping the OPC UA service sets that allow DDS actors to access OPC UA server functions such as Read, Write, and Browse. A simplified view of the OPC UA AddressSpace is also provided by mapping the DDS subscription model using specific DDS topics. Although the translation of the OPC UA protocol to DDS was proven, the opposite has not been evaluated. In addition, the authors performed experiments with only one device. In IoT and IIoT applications, several devices are connected in the same network, constantly sending data, evidencing the need to perform new tests with a more significant number of devices.

Kannoth (KANNOTH; SCHNICKE; ANTONINO, 2021), similarly to Endeley

(ENDELEY et al., 2019), uses a mapping technique to integrate different protocols with OPC UA for Industry 4.0 applications. The solution uses the Industry 4.0 Virtual Automation Bus architecture (VAB), which provides a standard technology-independent type system to map technology-related functions and types. The authors mapped the VAB architecture of HTTP and OPC UA protocols with their classes and services, such as HTTP GET and OPC UA Invoke primitives. The model was evaluated using a gateway that forwarded requests via HTTP from the industry 4.0 application to the OPC-UA server. The gateway runs an HTTP server and an OPC UA client simultaneously, collecting data from the OPCUA server and forwarding it to HTTP requests. Although the authors have indicated using an industry 4.0 application, it only employed the HTTP protocol.

Figure 9: Semantic gateway as service architecture.



Source: Adopted from (DESAI; SHETH; ANANTHARAM, 2015).

The current IoT interoperability provides end-to-end message delivery and lacks accessibility to semantic data. Desai (DESAI; SHETH; ANANTHARAM, 2015) proposes the concept of Semantic Gateway as Service (SGS) that offers an intelligent solution by integrating Semantic Web technologies with existing sensor and services standards. The SGS also provides a mechanism to incorporate communication protocols into a single system. Its architecture has three core components (Figure 9 (DESAI; SHETH; ANANTHARAM, 2015)): (1) multiprotocol proxy, (2) semantic annotation service, and (3) gateway service interface; the gateway can also have components for required capabilities such as message store and topics router, which assists multiprotocol proxy and gateway service interface in translation between messaging protocols.

The SGS combines the use of ontologies, such as SSN ontology (COMP-TON et al., 2012), and a communication translator providing communication between widely used CoAP, MQTT and XMPP protocols, making their semantic integration possible and seamless. Although the framework allows the translation of several communication protocols, the developed ontology does not describe them. Furthermore, no specific protocols for the industrial context

are suggested, limiting the scope of IoT systems to the protocols specified by the author.

3.4 Discussion

In this chapter, the most relevant works found in the literature related to the purpose of this dissertation were presented. These works' results were analyzed to verify technologies that could be reused in this dissertation. In addition, their gaps were identified, including possible contributions to state of the art. Table 1 compares the main related works from the perspective of four main aspects: communication protocols, semantic technologies, digital representation, and industry domain.

- **Communication protocols:** Works that use communication protocols that are well disseminated, both industrially and not, such as MQTT, OPC-UA, and HTTP, are classified as "totally relevant"; works that do not deal directly with communication protocols but somehow carry out data exchange between systems are classified as "partially relevant"; those that do not fit the other classifications are classified as "Not applicable" definition.
- **Semantic technologies:** Works that deal with semantics for divergent systems using ontologies, Unified Modeling Language (UML) among others, are classified as "totally relevant"; works that create or use some custom data structure as a standard for communication are classified as "partially relevant"; those that do not fit the other classifications are classified as "Not applicable" definition.
- **Digital representation:** Works that use the virtualization of physical devices, digital twins, to develop simulations or research in a specific area are classified as "totally relevant"; works that only discuss CPS techniques but do not make any implementation or in-depth study are classified as "partially relevant"; those that do not fit the other classifications are classified as "Not applicable" definition.
- **Industry domain:** Works specified as "totally relevant" are those that somehow refer to the industrial sphere (Industry 4.0, IIoT, industrial communication protocols); works that are not used in industry, but can be adapted, as in the case of specific ontologies for IoT, are classified as "partially relevant"; those that do not fit the other classifications are classified as "Not applicable" definition.

In brief, many of the researched works dealing with physical interoperability problems use middlewares and multiprotocol gateways to translate different communication protocols. However, many of these end up not dealing with issues related to the data structure, exposing the system to semantic problems. Semantic interoperability is commonly addressed in the literature using semantic technologies, such as ontologies and UML, some of which use international standards as their base (IEEE). At last, in the previous few years,

some authors have worked with industrial assets digitization standards to develop studies concerning the use of the AAS.

Yet, none of the related works combines the three technologies in a single system. In this sense, this dissertation seeks to solve this gap by developing an interoperability middleware for IIoT gateways based on international standard ontologies and standardized digital representation. This work combines multi-communication protocols translator algorithms, specific IIoT ontology based on international standards, and the so-to-be digital asset standard AAS.

Table 1: Comparison of the main related works.

Work	Communication Protocols	Semantic Technologies	Digital Representation	Industry Domain
Budakoti, Gaur, and Lung (2018)				
Castellanos et al. (2021)				
Palavras et al. (2018)				
Derhamy, Eliasson, and Delsing (2017)				
Ferreira et al. (2019)				
Wang et al. (2022)				
Schroeder et al. (2016)				
Ye and Hong (2019)				
Ye et al. (2020)				
Bouter et al. (2021)				
Arm et al. (2021)				
Iñigo et al. (2020)				
Bermudez-edo et al. (2016)				
Agarwal et al. (2016)				
Kumar et al. (2019)				
Steinmetz et al. (2018)				
Endeley et al. (2019)				
Kannoht, Schnicke and Antonino (2021)				
Desai, Sheth, and Anantharam (2015)				
Related Works				

Source: The author

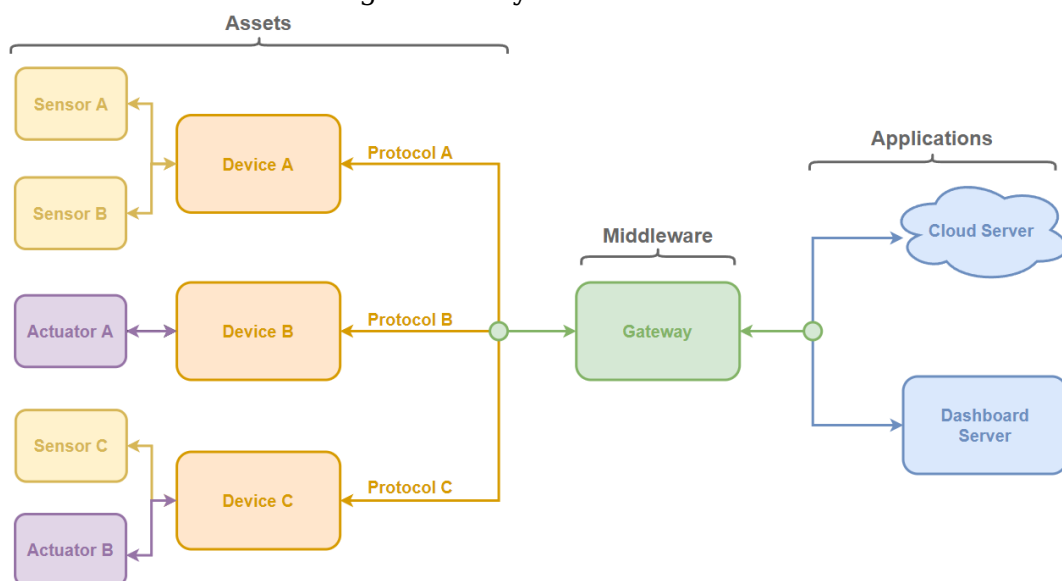
4 PROPOSED IIOT INTEROPERABILITY MIDDLEWARE

This chapter describes the research approach for mitigating interoperability problems between industrial assets using multiprotocol translators, semantic models, and digital representations. An overview of the project and its main characteristics are presented, enabling a broad understanding of all the developed subsystems used to solve the gap found in the literature presented in Section 3.4.

4.1 Proposal Overview

The proposed middleware is a fundamental tool for mitigating interoperability problems in the industrial environment. It allows communication between industrial assets using protocol translators and techniques for semantic compatibility of the exchanged data: increasing flexibility and the possibilities for accomplishing various applications using heterogeneous devices.

Figure 10: System Overview.



Source: The author.

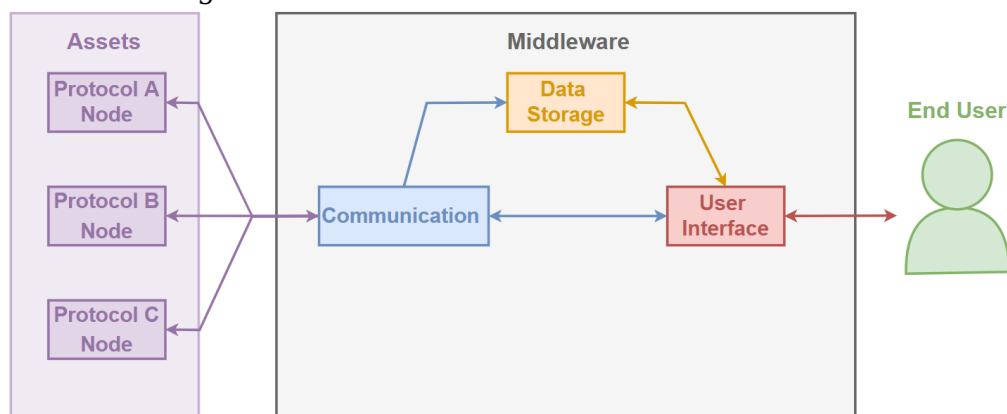
The system comprises three essential components: the assets, the middleware, and applications (Figure 10). Assets range from sensors, actuators,

robots, and PLCs to large industrial machines, whose primary function is monitoring and controlling environment data. The middleware can be a gateway, as already mentioned in Section 2.1.2, which is responsible for different tasks, such as forwarding the assets' data to the applications, acting as a protocol translator, storing data, etc. The applications, in turn, comprise different possibilities, such as cloud servers, dashboard servers, and specific platforms.

4.2 Proposed Middleware Architecture

The architecture for the developed gateways' interoperable middleware is explored in this section. The architecture comprises three main blocks (Figure 11): Communication, Data Storage, and User Interface. These blocks will be presented in the following sections.

Figure 11: Middleware's architecture overview.



Source: The author.

4.2.1 Communication Block

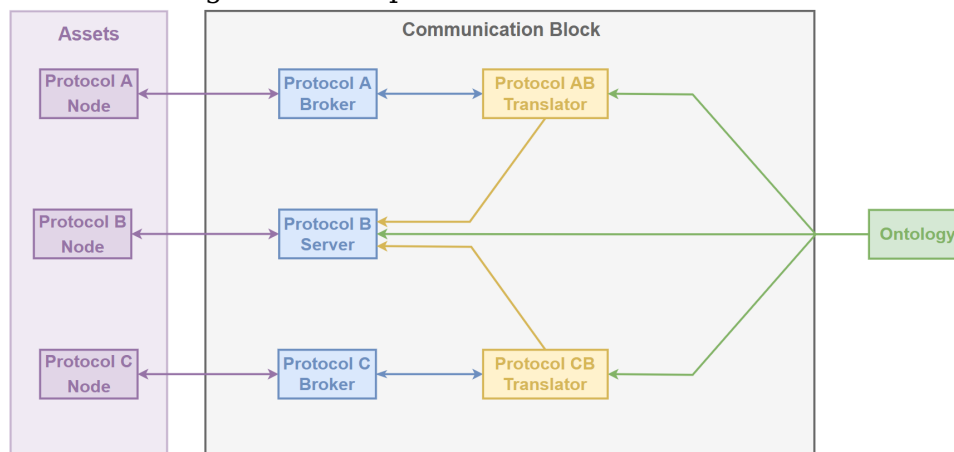
The first block, communication (Figure 12), allows data transmission to/from different industrial assets by incorporating different communication protocol servers and brokers. The approach selected for this dissertation uses three industrial communication protocols widely used in real applications, but the middleware could comprise "n" translators. A shared server is used to store all messages exchanged in the system by defining an intermediate protocol, reducing the necessary number of translators.

The communication protocols translators are responsible for converting data from one protocol to another, allowing interoperability between devices that use distinct and usually incompatible protocols. The translators use data from the ontology developed for the specific system to carry out this conversion, mapping several essential characteristics, such as functions and topics. The complexity of this block is directly related to the number of protocols used since a more significant amount of servers, brokers, and translators are needed.

As an example, in Figure 12 is depicted an application using three communication protocols, being the Protocol B the intermediary. Therefore, re-

maintaining protocols must be all translated to the Protocol B . So, to translate information from protocols A to C, it is mandatory to perform the translation from A to B and then from B to C.

Figure 12: Proposed communication block.



Source: The author.

4.2.2 Data Storage Block

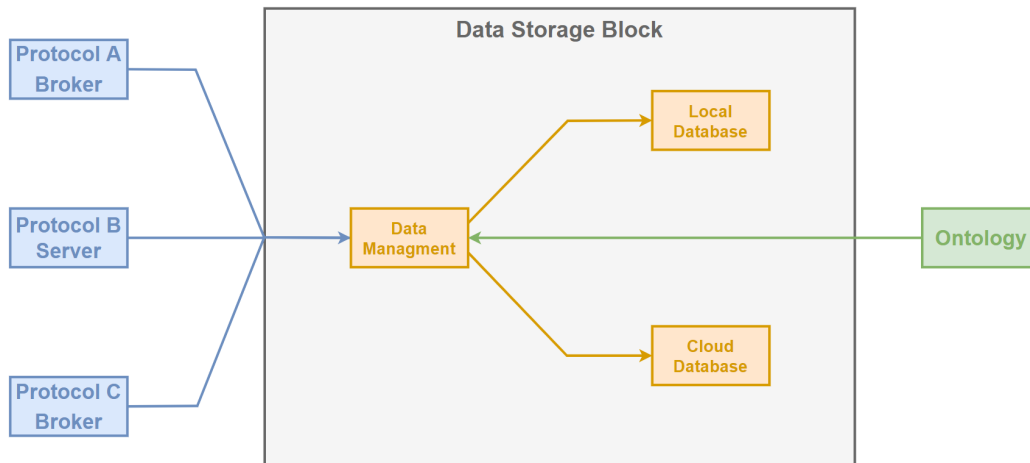
As the name suggests, the "Data Storage" block (Figure 13) is responsible for storing industrial assets' physical and digital data (digital twins) in a local or cloud database. Nevertheless, all incoming messages are processed before being saved. The ontology description makes it possible to identify various information from this message, such as the id of the device that sent the message and its communication protocol, determining the type of asset (a sensor or an actuator), etc.

The database used is based on structure data, storing information regard each sensor and actuator individually. Each message has a time stamp and includes different labels, defining the senders communication protocol, assets' type and so on, facilitating the reading and writing of processed and incoming data. It is worth mentioning that the saved data refer to industrial processes, such as production lines, maintenance systems, etc. Therefore, the stored data can be used as a dataset for machine learning algorithms, predicting machine failure and improving manufacturing time.

4.2.3 User Interface Block

The user interface block (Figure 14) aims to make the data available in a simplified way so that the end-user can monitor the system's primary information, such as momentary sensors and actuators data. Also, allow the user to control the system, such as controlling the status of specific actuators, start, stop and resume an industrial process. It is necessary to select a well-distributed framework and adapt it to the specified use case. For this reason, the creation of SCADA like software, as SCADA is already widely used for industrial applications mainly involving PLCs, is proposed. SCADA allows

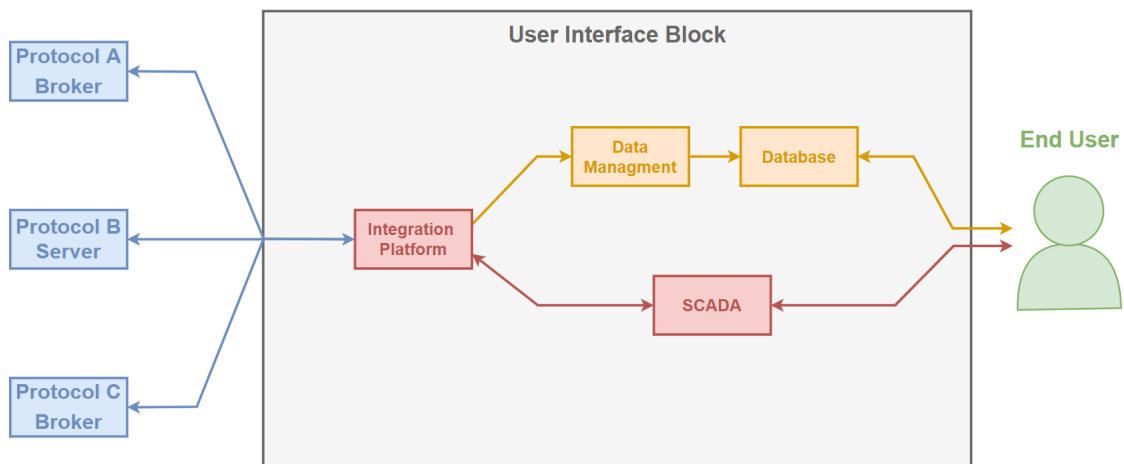
Figure 13: Proposed data storage block.



Source: The author.

the user, locally or remotely, to monitor process data from an industrial production plant and act manually or automatically in the process. In addition to data visualization and control, it is necessary to allow the user to interact with the database: verify data from performed simulations and change relevant information such as its bucket name.

Figure 14: Proposed user interface block.



Source: The author.

4.3 Developed Ontology

This work combines several ontologies well disseminated in the literature, such as IoT-Lite (BERMUDEZ-EDO et al., 2016) and QUDT (RIJGERSBERG; VAN ASSEM; TOP, 2013), with ontologies standardized by the IEEE (IEEE, 2015) to develop one specialized for IIoT applications. In addition, the proposed ontology includes essential concepts related to industrial applications yet not presented in the base ontologies. Figure 15 represents the combination of classes from different base ontologies, identified by different colors,

location in space-time. From Physical, the Object notion is presented. Object constitute everyday objects that exist in space with spatial components that are parallel to time. Last but not least, an Artifact is a manufactured object.

From FIESTA-IoT (AGARWAL et al., 2016) the DOI concept is reused, which comprises the domain of interest which is correlated to assets by the property hasDOI (not shown in the image so as not to impair the reader's understanding). Some of the examples of DOI are Agriculture, Energy, Health Care, and Smart cities.

More than just a collection of vocabularies for different quantity and unit standards, the QUDT (RIJGERSBERG; VAN ASSEM; TOP, 2013) ontology provides a conceptual framework. This work's proposed ontology reuses the definition of Quantity Kind and Unit. For QUDT, Quantity Kind refers to any observable property that can be measured and quantitatively quantified. Physical characteristics include length, mass, time, force, energy, power, electric charge, etc. The definition of a Unit is a specific quantity of a particular sort that is used as a scale to measure other quantities of the same kind. For instance, the BIPM has empirically defined the Meter as a unit of length. Any length can be written as a number that has the unit meter multiplied by it.

From the IoT lite ontology (BERMUDEZ-EDO et al., 2016), at least four classes (represented in Figure 15) are merged into the IIoT ontology. The Coverage concept corresponds to a communication device, i.e. an IoT device equipped with a temperature sensor inside a room has coverage of that room. An Actuating Device is defined as a device that can actuate over an object or QuantityKind. A Tag Device is used to set an identification number, to different devices that act in the same application, even if these are similar equipment, such as QR code, RFID, or bar code.

Classes for describing the assets' connectivity were added, such as Communication Protocol and Communication Type, related to the "Communication Device" class defined by the SSN ontology (COMPTON et al., 2012). As the names suggest, Communication Protocol represents the protocol used by the IIoT node, such as LoRa, WiFi, NB-IoT, and so on. As for Communication Type, sets if the communication used by the devices is wired or wireless, which can impact the positioning of a node within an environment.

The IIoT ontology also introduces two new classes for industrial assets IElement and IThing. IElement includes elements monitored and controlled throughout processes, such as industrial tanks, ovens, etc. The IThing class represents the gateways and equipment with sensors, actuators, and radios to monitor, control, and exchange interest data. An IThing can have a sensor attached to it, but an IElement is not necessarily connected to the sensor, even though both a Device and an IThing are set as assets.

Moreover, classes and properties with relevant data for creating assets' individual AAS, facilitating their development, and creating module standards based on the proposed ontology, were included. The based ontologies are described in Table 2.

Table 2: Ontologies used for the development of the IIoT ontology

Name	Description	Reference
SUMO	It is a top-level ontology that defines basic ontological categories across all domains. Is used as the basis for an interchange language for morphosyntactic to resolve the meaning of terms in web search.	IEEE, 2015
ROCO	Defines the main concepts and relations regarding robot architecture for autonomous systems and inherits from SUMO/CORA ontologies (Prestes et al., 2013). Its primary goal is to enhance information sharing in cloud robotics (CR).	FREITAS et al., 2020
SSN	It is an ontology for defining sensors and their observations, the techniques used to make them, the studied features of interest, the samples used, the observed properties, and actuators.	COMPTON et al., 2012
IoT Lite	It is a lightweight ontology to represent Internet of Things (IoT) resources, entities, and services. IoT-Lite is an instantiation of the SSN ontology (COMPTON et al., 2012).	BERMUDEZ-EDO et al., 2016
FIESTA IoT	It aims to achieve semantic interoperability among heterogeneous testbeds. It takes inspiration from the well-known methodology for reusing and interconnecting existing ontologies, such as SSN (COMPTON et al., 2012) and IoT Lite (BERMUDEZ-EDO et al., 2016).	AGARWAL et al., 2016
QUDT	It provides semantic specifications for units of measure, quantity kind, dimensions, and data types. QUDT is an advocate for the development and implementation of standards to quantify data expressed in RDF and JSON.	RIJGERSBERG; VAN ASSEM; TOP, 2013

Source: The author.

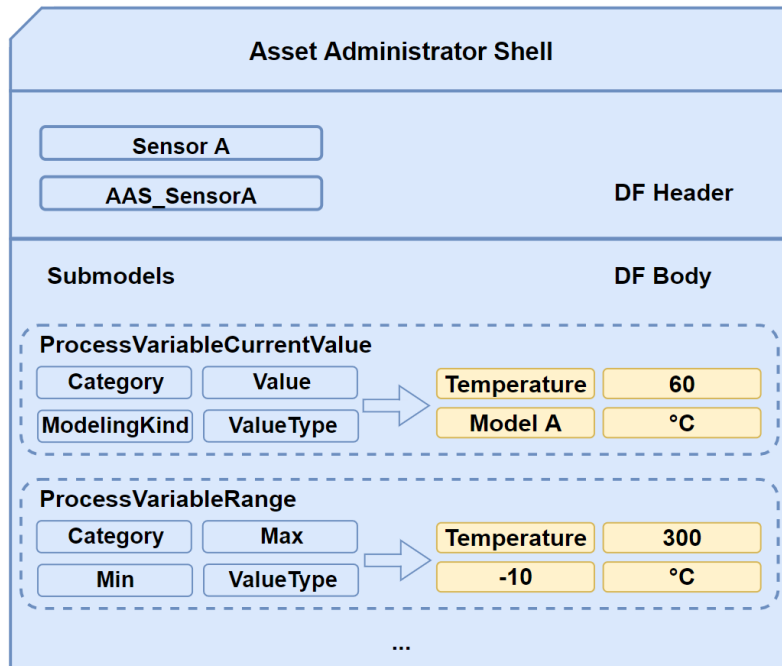
4.4 AAS Submodels

As already indicated in the section 2.7, AAS includes submodules that describe the functional and non-functional properties of the assets. Industrial Digital Twin Association (IDTA) is working on standardizing the AAS submodules. Once the submodel template is proposed, a group of experts in the field develops the template. After elaboration, the template is available for use. There are two standardized and published models; the rest is under development or revision. The default templates are Generic technical data (Plattform Industrie 4.0, 2020a) and Nameplate (Plattform Industrie 4.0, 2020b). The Nameplate template has information regarding the manufacturer and serial number used in case of maintenance or replacement of the Asset.

Due to the lack of AAS submodules standards, three unique ones are recommended to manage sensitive data in typical IIoT applications. Two of these are specific to sensors: ProcessVariableCurrentValue and ProcessVariableRange Figure 16. Each represents, respectively, information regarding the sensor's current measurement and its measurement range. The Status submodule was created for actuators, containing four properties, one of which is Value, representing the actuator's current Boolean data. The information used to describe the modules is based on the information described in the ontology.

The assets are described straightforwardly in this project, with only a few submodules. On the other hand, this information is sufficient to validate the proposed methodology. Based on the ontology information, other submodules could be developed to enrich the digital description of the assets, but this was not the focus of the research.

Figure 16: AAS sensors submodules.



Source: The author.

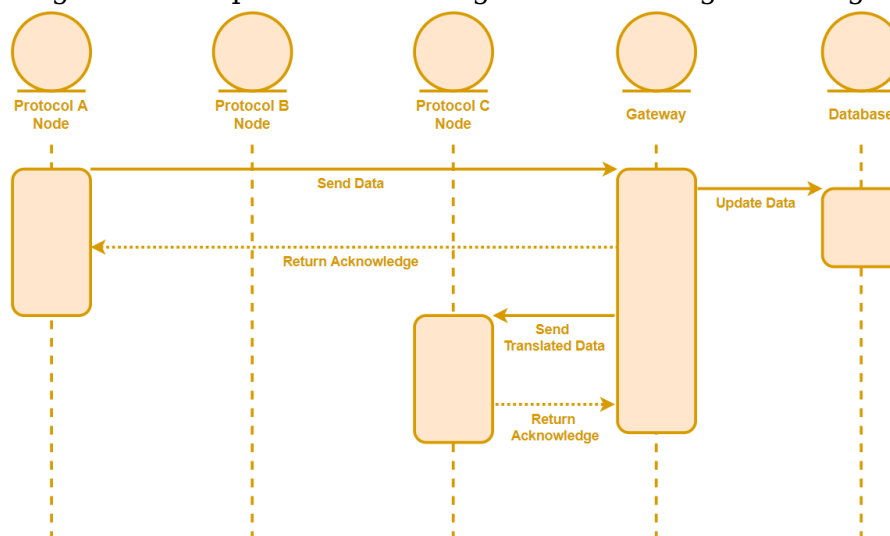
4.5 Communication Translators

This dissertation suggests defining an intermediate protocol to minimize the complexity of the system by using a smaller number of communication protocol translators, as stated in section 4.2.1. As a result, any new protocol introduced to the application requires only one new translator. All developed translators must be bidirectional, allowing for sending and receiving data from any protocol to the intermediary one.

Additionally, a shared database is used, preferably the server/broker from the protocol defined as an intermediary, for storing assets' data. Consequently, it is possible to create an event-based system, sending updated data to specific devices whenever an event happens. For example, in the ontology description, "DeviceA" relies on "SensorD" data; an event is triggered anytime this data is updated. As a result, the gateway's middleware selects the appropriate translator that correctly encapsulates and sends the data.

Figure 17 represents the UML sequence diagram for exchanging messages between three devices based on three different communication protocols, each named by its used protocol (A, B, and C). Also, Node C requires data transmitted by Node A. In the case presented, Node A sends a message to the gateway, which will save the received data to the database and return an acknowledgment message. After that, the gateway's middleware performs the translation from protocol A to protocol B, the intermediate protocol, and finally translates the data from protocol B to C. The translated data will be forwarded to Node C, which will send an acknowledgment message to the gateway.

Figure 17: Sequence UML diagram for message exchange.



Source: The author.

4.6 Proposed standardization

One of the focuses of this work is to allow communication between heterogeneous devices that communicate with different types of communication protocols. Most protocols follow publish-subscribe or client-server patterns, so specific characteristics, such as standard nomenclature and data structure.

4.6.1 Proposed topic terminology

Thus, it was stated that all topics used by publish-subscribe protocols, such as MQTT, must use the device id as their topic, for example, "/DeviceA". However, when an event is triggered, the middleware must send updated information using the name of the sensor/actuator as the topic, for example, "/SensorD".

4.6.2 Proposed data structure

Another definition was the standardization of the data structure used by the devices. It is worth mentioning that different types of formats are used, therefore two were selected: JSON (BRAY, 2014) and the OMG Interface Definition Language (IDL) (SIEGEL, 1998).

JSON is a lightweight open standard for data format, text-based and language-independent. It is human-readable and straightforward for computers to parse and utilize. JSON is often used in various systems and protocols because it uses objects represented by an array. Figure 18 illustrates the structure of a JSON message following the patterns used in this project.

The developed pattern uses three elements: "*timestamp*" (Fig. 18, line 3), "*sensors*" (Fig. 18, line 4), and "*actuators*" (Fig. 18, line 7). The "*timestamp*" element is a `time_t` data type that indicates the timestamp in Unix time added by the device before sending the message. The "*sensors*" and "*actuators*" elements are sub-arrays with varied sizes, which contain the values of the sensors and

Figure 18: Proposed JSON message structure.

```

1 {
2   "timestamp": 1644692844,
3   "sensors": {
4     "SensorA": 0.465,
5     "SensorB": 12,
6   },
7   "actuators": {
8     "ActuatorA": true,
9   },
10 }

```

Source: The author.

actuators of the devices, respectively. In this example, the device that sent the message has two sensors and an actuator.

Figure 19: Proposed IDL data type.

```

1   Header header
2   float32 SensorA
3   int32 SensorB
4   bool ActuatorA

```

Source: The author.

IDL is a descriptive language for defining data types and interfaces in a language-independent manner regardless of the programming language or operating system/processor platform. The IDL defines the syntax for defining data types and interfaces. Data types are required to establish parameters and return the value of interfaces' operations. Figure 19 illustrates an IDL data type developed for a specific device.

The data type created was based on the same fictitious device used to present the message structure using JSON format 18. It is composed of a Header structure and specific variables for the device's sensors and actuators data. The Header (Fig.19, line 1) is composed of the timestamp in Unix time and a message counter. There is a variable for each sensor, SensorA (Fig.19, line 2), defined as a float32, and SensorB (Fig.19, line 3), as an int32. Furthermore, the device includes an actuator, ActuatorA (Fig.19, line 4), whose data is expressed by a boolean variable. Therefore, all equipment that uses this type of structure must have its data type created.

5 CASE STUDY

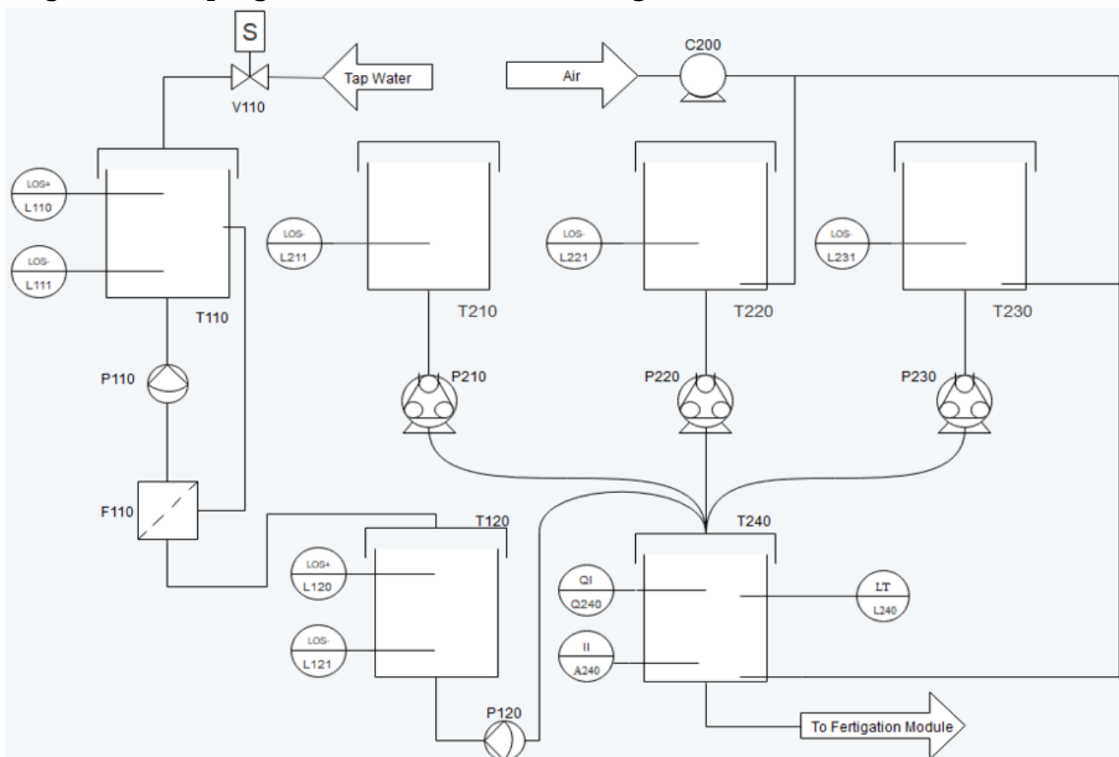
This chapter describes the case study selected to conduct the experiments to validate the functionality and obtained results using the middleware developed in this work. The chosen use case is from the agriculture and chemical domain, available at Los Andes University in Colombia, a research partner from GCAR/UFRGS.

The case study refers to the production process of a nutrient solution for soilless culture agriculture, e.g., hydroponics agriculture techniques. This cultivation technique is known for growing plants in an inert medium utilizing a solution of mineral fertilizers dissolved in water. It relies on various agronomic and environmental variables that must be managed to maximize input returns while conserving resources. It can deliver intensive food production in limited areas with more efficient resource management, including non-arable places such as deserts and cities. Soilless culture is seen as a viable instrument for food security as new agricultural systems are needed to meet food demands while reducing the environmental impacts of production.

The testbench used for the nutritional solution modules (NSM) (BARBIERI et al., 2021), depicted in Figure 20 (BARBIERI et al., 2021), is a double stock open system, which means that only pH and EC are controlled throughout each cycle. Therefore, for each irrigation, a new solution is produced. Several sensors and actuators are used in the testbench: eight level sensors, a pH sensor, an electrical conductivity (EC) sensor, a valve, five pumps, and an air compressor. So that these assets have standardized names that comprise one character and three numeric digits, the character indicates the asset's type, such as L for level sensors and P for pumps. The first and second numeric digits represent the assembly unit and the tank in which it is placed. Finally, the third digit is reserved for digital level sensors; it specifies whether the sensor is high-level (equals zero) or low-level (defined as one).

The NSM consists of two subsystems: filtration unit and recipe preparation unit. The filtration unit is initialized when the V110 solenoid valve is activated, filling the T110 tank with tap water. When both level sensors of the first tank are triggered, the P110 pump supplies the water into the F110 inverse osmosis filter to reduce its chloride and TSS content. The subsystem is finished when the T120 tank is filled with freshly filtered water. In the recipe preparation, the C200 air compressor agitates the fertilizer tanks T220 and T230 to prevent the concentrated nutrients from settling down. Air is also delivered to the

Figure 20: Piping and instrumentation diagram for nutrient solution module

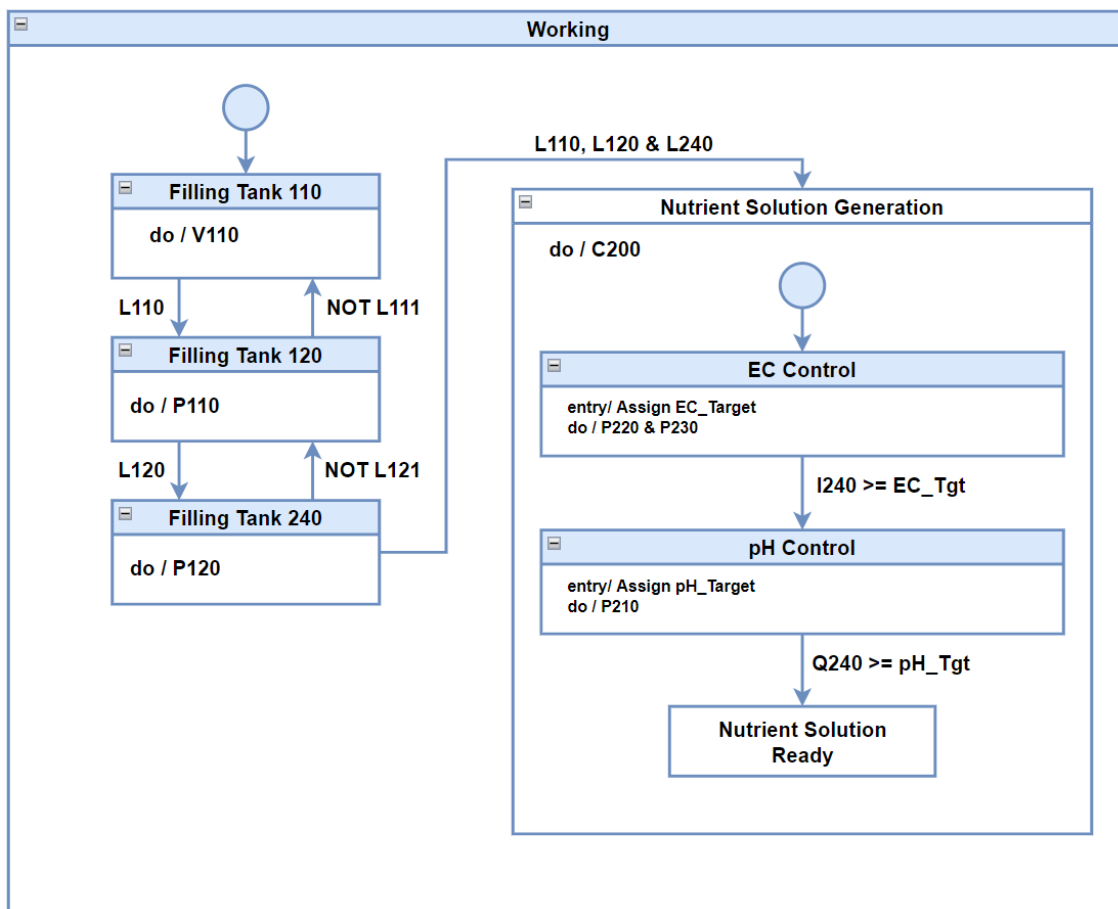


Source: Adopted from (BARBIERI et al., 2021).

T240 mixing tank for mixing the nutrients during the solution preparation. On tanks T210 (acid stock), T220, and T230 (A and B stocks), a low-level sensor is installed to alert the operator when the acid and concentrated nutrients need to be refilled. Acid and nutrients are delivered via peristaltic pumps P210, P220, and P230 to control the pH and EC of the fluid. The solution's data in the T240 tank is monitored by three sensors, Q240, A240, and L240, which are used to measure the pH, EC, and level, respectively.

The state machine diagram of the NSM is shown in Figure 21. The first two states were set to fill the first two tanks, T110 and T120. Then, filtered water is sent to the T240 mixing tank to start the preparation of the solution. However, the three tanks, T110, T120, and T240, must be filled to speed up the ascending solution production before continuing the process. Due to the module's responsibility for managing two distinct nutrient solutions, a particular EC and pH target value is given depending on the considered sample of plants. Throughout the subsequent phases, the C200 air compressor will be active. Firstly, the Ec value will be controlled; for that, the pumps P220 and P230 will be activated until the value measured by the sensor is the one set by the operator. The pH control will be carried out in the next state, starting the P210 pump and pumping the acid stock into the solution. After stabilizing the Ec and pH in the range indicated by the user, C200 is deactivated. This process concludes one of the cycles so that the nutrient solution is ready and can be stored for future use. The system can start a new cycle, but now from

Figure 21: State machine diagram of the NSM



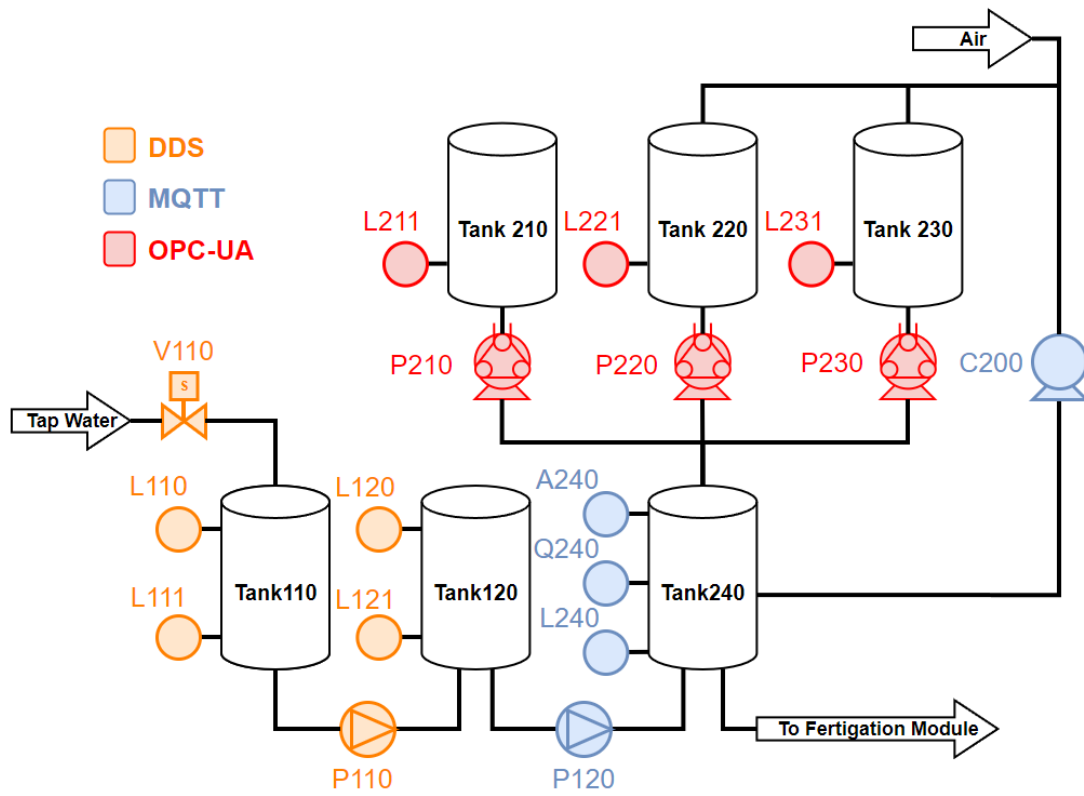
Source: Adapted from (BARBIERI et al., 2021).

the "Filling Tank 240" state.

The automation of the production of the nutrient solution is of paramount importance to achieve the system goals. It demands continuous monitoring of several process variables using different sensors and remote control to ensure the proper sequence of production steps described in the state machine. Using heterogeneous devices with specific sensors and actuators should allow the process to occur autonomously and safely. Therefore, three IIoT devices are identified in the case study presented above, each communicating using a different communication protocol. The sensors and actuators are connected to these devices. As depicted in Figure 22, three groups corresponding to three different communication protocols (DDS, MQTT, and OPC UA) can be identified by the colors orange, blue, and red.

The next chapter will describe the implementation of the developed middleware using ontologies, communication translators, and digital representations. The use case described above will be adopted to illustrate how sensors and actuators connected via distinct communication can interoperate using the concepts proposed in this work.

Figure 22: Use Case Assets Grouping



Source: The author.

6 IMPLEMENTATION DETAILS

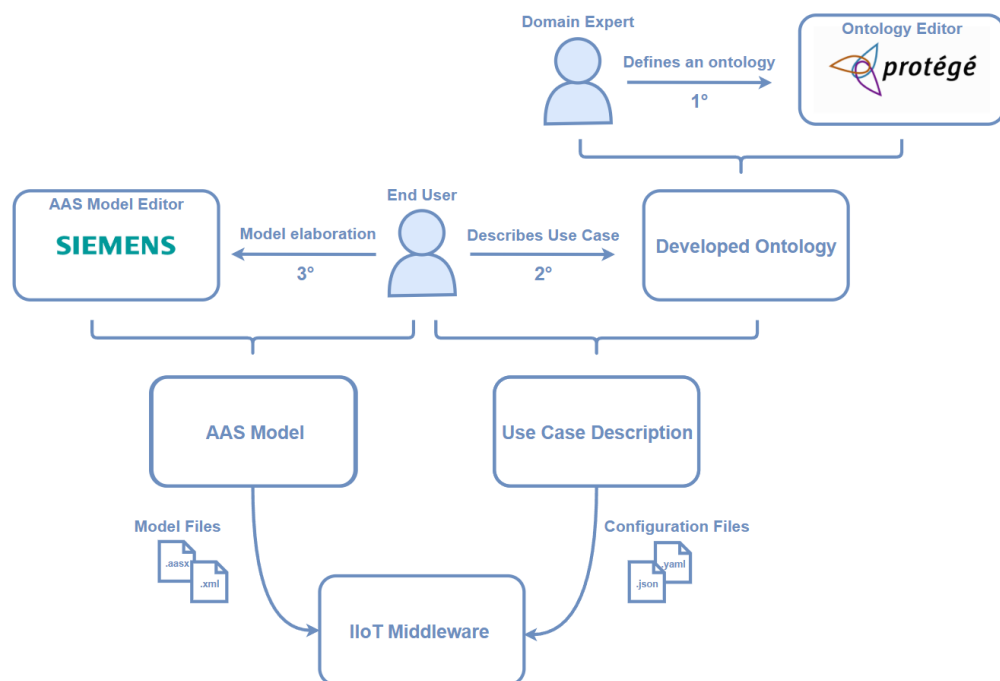
This chapter presents the implementation that was developed in order to validate the proposal presented in the previous chapters. The selected technologies and implementation details are discussed, showing how the proposed ideas can be deployed using state-of-the-art technologies.

6.1 Selected technologies

The approach used to implement the IIoT middleware discussed in this work is composed of two parts: system description and system execution. Each part is treated individually throughout this section. In addition, selected software and hardware components are presented.

6.1.1 Description Part

Figure 23: Description part of the project's approach.



Source: The author.

The first part comprises the description of the system. The IIoT ontology

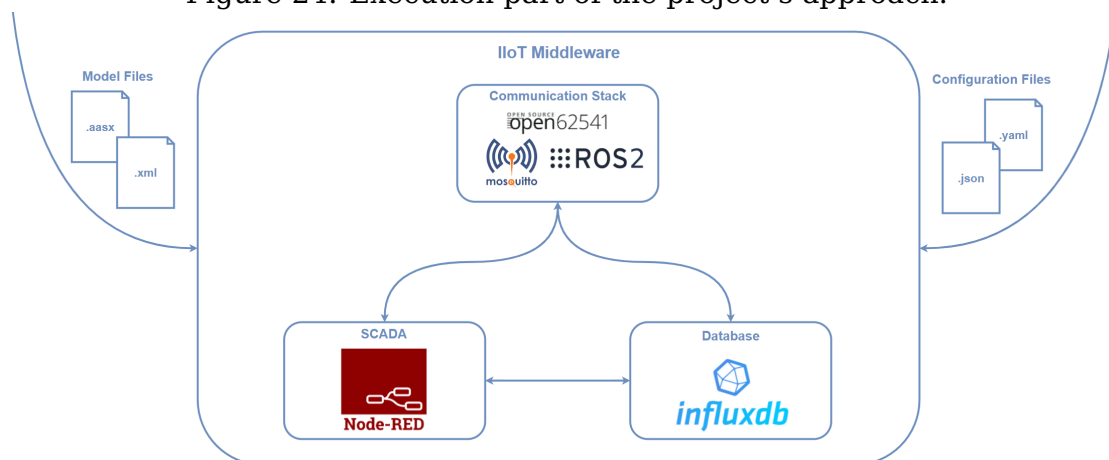
presented in section 4.3 is first described using Protégé software (Stanford, 2022). Protégé was selected since it is one of the most adopted tools for this purpose. Based on this generated ontology, an end-user, typically an automation engineer or similar, will then be able to instantiate the components of a particular industrial plant, such as sensors, actuators, PLCs, etc. The description should also include the IIoT device's data structure and communication characteristics. After modeling the ontology, it generates two configuration files, JSON and YAML Ain't Markup Language (YAML) formats. One of these files describes the communication characteristics of all IIoT nodes in the system, comprising MQTT URL, port, and topics, for example. The other will identify each device's communication protocol, sensors, and actuators with their respective data dependencies. These files are vital for configuring the gateway's communication protocol translator scripts, which will be described later in this chapter.

Additionally, based on the use case description using the IIoT ontology, the user creates the asset's digital representation using the Siemens OPC UA Modeling Editor (SiOME) (SIEMENS, 2022). The description follows the OPC UA information model and is then used by SiOME to generate an AAS model compatible with the OPC UA protocol. After elaborating the model in the editor, the file is converted to the "OPC UA Nodeset XML schema" (Nodeset2.xml). Then, the open62541 library (OPEN62541, 2021) transforms the model (file extracted from SiOME) into a .c file that will be the basis of the OPC UA server.

It is important to note that the use case description must be adjusted in case the user adds a new sensor/actuator to a node or a new device to the system. If not, the gateway's configuration files will lead to communication and syntactical errors.

6.1.2 Execution Part

Figure 24: Execution part of the project's approach.



Source: The author.

The execution part (Figure 24) can start right after the modeling part is done. Configuring the servers and brokers for the communication protocols

is the first step in the execution. Given that the middleware uses AAS and consequently OPC UA for its operation, an OPC UA server must be configured and initialized. The OPC UA server is created using the open62541, an open-source C implementation of OPC UA that is initialized using the parameters from the SIOME software file. Two additional communication protocols were also established to be compatible with the gateway: Eclipse Mosquitto and Robot Operation System 2 (ROS2) (ROS2, 2021).

Eclipse Mosquitto, an open-source server implementation for MQTT versions 5.0, 3.1.1, and 3.1, was installed and configured using the ontology-based configuration files. It was chosen as it is widely used for embedded computers or microcontrollers, such as raspberry pi. This broker also provides SSL support for encrypted network connections and authentication, essential for industrial applications.

Furthermore, the Robot Operating System 2 (ROS2) Galactic version was installed in the gateway, a set of software libraries and tools for building robot applications. The ROS2 framework was chosen since it uses the DDS protocol for serialization and transport messages and device "discovery" and has become a widely utilized platform across many robotic applications, allowing manufacturers to eliminate outmoded equipment and tailor a robot's software to their preferences. ROS2 enables DDS communication protocol publishers and subscribers to communicate through peer-to-peer communication, creating a mesh network without nodes relying on a master.

With the servers and brokers configured and initialized, the protocol translator scripts are started using the two ontology-based configuration files. The translators were developed within a ROS 2 workspace explicitly set to store and run these converters and dedicated scripts for device simulation. Each of the translators is executed as a ROS2 node so that several translators may execute simultaneously or individually as a single process.

As discussed in the previous chapter, SCADA like software was adopted to implement a friendly user interface. The NodeRED (O'LEARY; CONWAY-JONES, 2016) tool was selected for this purpose. With all the communication dependencies set, the user opens a NodeRED dashboard hosted by the same gateway. The developed SCADA like allows easy data monitoring and provides simple user interfaces for executing commands from the terminal, such as device simulation scripts, or for starting/closing communication protocols brokers/servers.

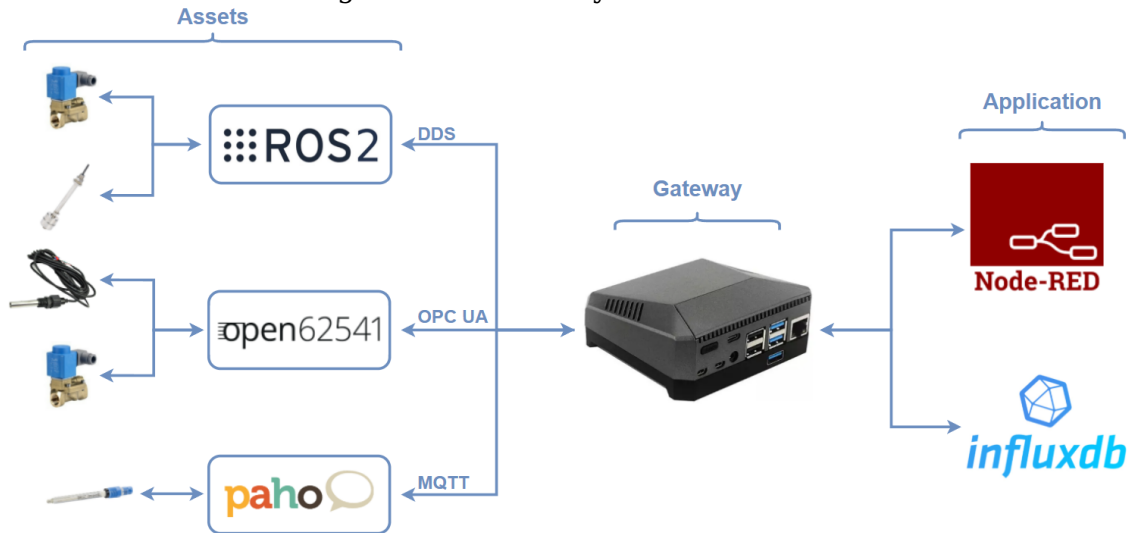
Additionally, a local database, using influxDB, is used to store system data for future analysis. This database was chosen since its primary use is for monitoring applications' data, with a high volume of queries and writes per second without causing much impact on the operating system. It also allows for adding labels to the data before saving it, such as for specific communication protocols or device types (sensor or actuator), enabling easy data filtering.

Finally, the system can receive and forward data from/to the heterogeneous devices described in the ontology. The user can connect to the nodes or use the SCADA like system to execute digital twin scripts to start the experiments. The middleware enables the system to function autonomously until prompted

to stop by the user.

6.2 System Architecture Overview

Figure 25: Actual System Overview.



Source: The author.

Three communication protocols were selected for testing the developed middleware: DDS, OPC UA, and MQTT. The OPC UA and DDS protocols are widely adopted in industrial applications as they are Industry4.0 standards. Each communication protocol follows a different communication pattern; OPC UA is based on the client-server communication paradigm, while DDS uses the publish-subscribe. Furthermore, the MQTT protocol is chosen since it is broadly used in IoT applications and has been widely deployed in industrial settings due to its ease of implementation and wide range of compatible devices. By selecting industrial communication protocols that follow different communication schemas, it was possible to demonstrate the flexible and interoperable way the middleware can be configured.

Three simulated devices (Figure 22) were defined to validate the interoperable middleware functionalities. Each node has at least five sensors and actuators, which communicate by one of the communication protocols. The simulation scripts use the ontology's equipment characteristics to simulate its behaviors. The three devices constantly monitor sensitive data for their operation and transmit relevant data to their respective broker or server whenever an update occurs. Acting accordingly to the pattern of the case study (Figure 21) selected for this work and presented in Chapter 5.

The OPC UA device script is implemented using the open62541 library and connects an OPC UA client to its corresponding gateway's server. The client sends updated data from its sensors and actuators using write functions to specific server positions, storing all current data on the server and available to other devices when required. Moreover, it constantly queries for dependency

data updates, as this information is directly related to the correct functioning of the equipment.

The DDS and MQTT device's scripts follow a similar pattern as both are based on the publish-subscribe model. Initially, the broker's parameters are configured and as soon as the communication is established, the client subscribes to the topics related to sensors and actuators whose information is essential for the device's operation, following the standard from section 4.6.1. The MQTT client is based on the paho python MQTT library (Eclipse Foundation, 2021), while the DDS was implemented using a ROS2 publisher and subscriber node. Both clients publish their data whenever modified, using the topic with their simulated device id. MQTT uses the JSON format (Figure 18), while the DDS uses the IDL structure (Figure 19) for its messages.

The interoperability middleware was deployed on a gateway based on a raspberry pi 4 with 8GB of RAM and 32 GB SD memory card, running Ubuntu Server 20.04 LTS (Canonical Ltd, 2021). The gateway includes the communication protocols brokers (DDS and MQTT) and server (OPC-UA), two bridging mechanisms for industrial communication protocols, a local database for storing the system data (InfluxDB), and a SCADA like (NodeRed dashboard) system to configure, monitor, and act during the simulation. Due to these characteristics, this gateway can be classified as an edge-enabled IIoT gateway (PAPCUN et al., 2020).

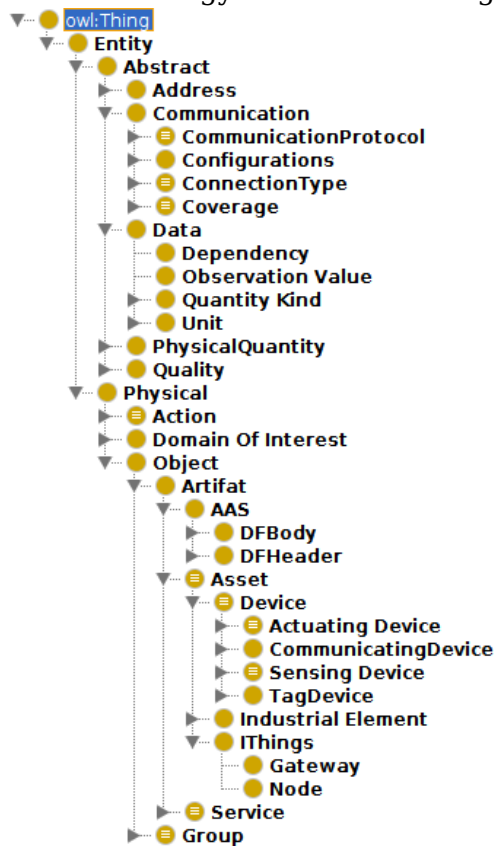
6.3 IIoT Ontology

As described in section 4.3 the IIoT ontology proposed in this work combines different existing ontologies. The implementation of the IIoT ontology started with the study of international standards, including the IEEE, through technical documents. Besides the standards, literary research on widespread ontologies for IoT applications, sensors, and relevant aspects to our approach. After a well-structured database, the ontology was firstly defined using the Protégé software. Protégé made it possible to develop classes and their properties to create a common language for different devices.

Throughout this work, the ontology was adapted according to the project's needs adding industry-specific data so that it now has 482 classes and 47 object property values. The IIoT ontology has similar numbers to other ontologies described in the literature, such as m3-lite and fiesta IoT, which have 451 and 484 classes and 5 and 30 object counts, respectively.

Figure 26 presents some of the classes developed for the ontology. Some of these classes are specific to adding AAS information, such as DFBody and DF-Header. Several classes specify the equipment's communication parameters since these are essential data for IIoT applications. It is worth mentioning the CommunicationProtocol class indicates the adopted communication protocol, and the configurations class defines the communication's server and broker information, such as URL, port, username, and password. In addition to these, the Data class and its subclasses are of paramount importance. They describe important information related to sensors and actuators measured data, such

Figure 26: IIoT Ontology Classes in Protégé software.



Source: The author.

as range, units, etc. This information is used to assemble the messages that will be transferred between system nodes.

The IIoT ontology was also based on IEEE international standards, such as the SUMO top-level ontology. Some of the classes inherited from the standard were: Physical and Abstract. These classes are super-classes to all others defined in the ontology, dividing the model into these two categories. The Physical subclasses are shown in Figure 28, whereas the Abstract subclasses are shown in Figure 27. Both images were generated in the Protégé software using OWLViz, which it is possible to observe the ontology graphically for a better understanding.

6.4 AAS

After describing the case study with the developed ontology, the SiOME software was used to produce the AAS models for each asset.

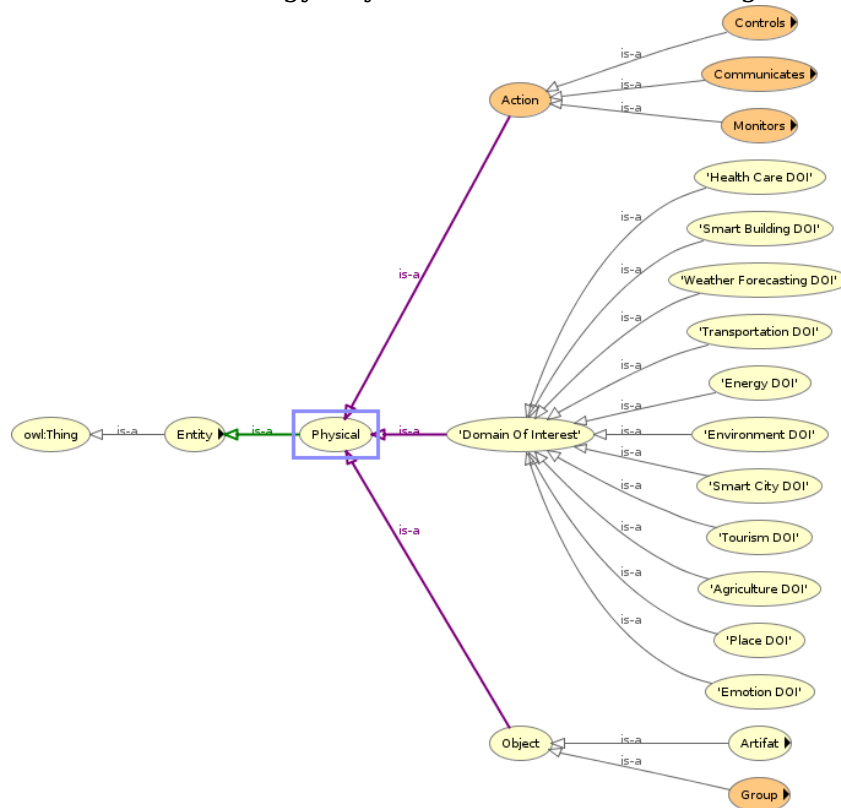
An example of an AAS is the AAS_A240 which is an Ec sensor as defined in chapter 5 that was modeled in SiOME, as depicted in Figure 29. This sensor has the ProcessVariableCurrentValue, and ProcessVariableRange submodules explicitly developed for this project, which was presented in the 4.4 section. Each submodule has several static properties based on ontology data and a variable one, the current sensor's measurement value (Value property of the

Figure 27: IIoT Ontology Abstract Subclasses in Protégé software.



Source: The author.

Figure 28: IIoT Ontology Physical Subclasses in Protégé software.



Source: The author.

ProcessVariableCurrentValue submodule).

As an example for actuators, it can be considered the AAS_P230 shown in

Figure 29: Screenshot of AAS_A240 in SiOME.

The screenshot shows the SiOME interface for AAS_A240. The left pane displays a tree view of the AAS structure, with 'ProcessVariableRange' selected. The right pane shows the 'OPC UA Attributes' and 'References' for this node.

OPC UA Attributes

NodeId	ns=1;i=5180
NodeClass	Object
BrowseName	1:ProcessVariableRange
DisplayName	ProcessVariableRange
Description	null
WriteMask	0

References

ReferenceType	NodeClass	Name	T
<input checked="" type="checkbox"/> HasProperty	Variable	Category	F
<input checked="" type="checkbox"/> HasProperty	Variable	Max	F
<input checked="" type="checkbox"/> HasProperty	Variable	Min	F
<input checked="" type="checkbox"/> HasProperty	Variable	ModelingKind	F
<input checked="" type="checkbox"/> HasProperty	Variable	ValueType	F

Source: The author.

Figure 30: Screenshot of AAS_P230 in SiOME.

The screenshot shows the SiOME interface for AAS_P230. The left pane displays a tree view of the AAS structure, with 'Status' selected. The right pane shows the 'OPC UA Attributes' and 'References' for this node.

OPC UA Attributes

NodeId	ns=1;i=5153
NodeClass	Object
BrowseName	1:AAS_V110
DisplayName	AAS_V110
Description	null
WriteMask	0

References

ReferenceType	NodeClass	Name	TypeDefinition
<input checked="" type="checkbox"/> HasComponent	Object	Asset	1:AASAssetType
<input type="checkbox"/> HasComponent	Object	DerivedFrom	1:AASReferenceType

Type Defined Placeholder

ReferenceType	NodeClass	Name	TypeDefi
---------------	-----------	------	----------

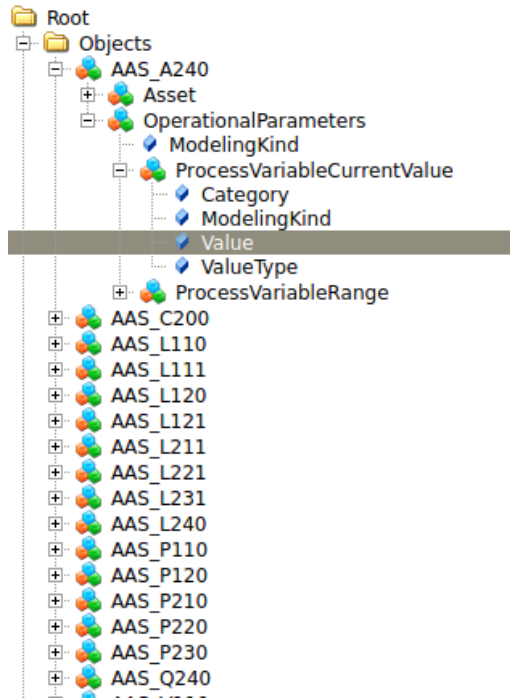
Source: The author.

Figure 30. It is noticeable that this AAS is more straightforward compared to the AAS_A240 sensor (Figure 29) since the first one has only one submodule, which corresponds to the Status of the actuator. The AAS_P230 represents the asset P230, an on/off pump whose current value is easily represented by a Boolean variable. This value is presented by the Value property of the Status submodule.

After the modeling phase, the description is converted to a .c file using the open62541 library. The converted file is used in the OPC UA Aggregation Server since its information model is based on the AAS model. Therefore, all assets' data are stored on the server hosted by the gateway.

Finally, an OPC UA server was built with the open62541 library and initialized based on the model described in the .c file. The UaExpert software, a full-

Figure 31: Screenshot of OPC UA server in UaExpert Software.



Source: The author.

featured OPC UA Client, is used to check whether the assets and submodules were appropriately created. Figure 31 presents the OPC UA server information in the UaExpert software for the AAS_A240 based on the model defined in SiOME (Figure 29). All submodels and properties proposed in SiOME are included on the server; therefore, both are compatible and appropriately created. After this, it is possible to read and write data utilizing scripts with OPC UA clients connected to this server.

6.5 Communication Translators

Essential files and scripts for the functioning of the proposed middleware were developed and stored in a ROS2 workspace, named `iiot_ws`, that was dedicated to this project. Two packages were created within the workspace: `iiot_interfaces` and `opcua_bridges`. The `iiot_interfaces` package includes the proposed IDL data types files (Figure 19) that allow data exchange via the DDS protocol, which is the basis for communication between ROS2 nodes. The `opcua_bridges` package comprises the protocol translators, the configuration files, communication certificates, and the simulated device scripts.

As stated in section 4.2.1, the proposed middleware was validated using three communication protocols. Furthermore, one of these protocols serves as an intermediary protocol, reducing the number of translators required. The OPC UA was selected as its server is already used to store asset data through its AAS. As a result, two translators are required, one for DDS and the other for MQTT, both of which must be translated to OPC UA and are bidirectional.

The two developed communication protocol translators are the MQTT-OPC

UA Bridge (MOB), using the paho python MQTT library, and the DDS-OPC UA Bridge (DOB), based on the opcua-asyncio library. The bridging mechanisms are written in Python and run as independent ROS2 nodes. The two converters use the files developed by the use case description (Chapter 5) using the IIoT ontology for its configuration.

Figure 32: Ontology JSON config file.

```

1 {
2   "device": [
3     {
4       "name": "Device01",
5       "commProtocol": "DDS",
6       "sensors": ["L110", "L111", "L120", "L121"],
7       "actuators": ["V110", "P110"],
8       "dependency": ["P120", "L240"]
9     },
10    {
11      "name": "Device02",
12      "commProtocol": "MQTT",
13      "sensors": ["A240", "Q240", "L240"],
14      "actuators": ["P120", "C200"],
15      "dependency": ["L110", "L120", "L121", "P210", "P220", "P230",
16        ""]
17    },
18    {
19      "name": "Device03",
20      "commProtocol": "OPCUA",
21      "sensors": ["L211", "L221", "L231"],
22      "actuators": ["P210", "P220", "P230"],
23      "dependency": ["A240", "Q240", "L240", "C200"]
24    }
25  ]

```

Source: The author.

The ontology.json is depicted in Figure 32; it has a primary element *"device"* (Fig. 32, line 2) that contains relevant information about the devices used in the proposed system. This device's information include the device's name (Fig. 32, lines 4/11/18), the adopted communication protocol (Fig. 32, lines 5/12/189), and an identification number for its sensors (Fig. 32, lines 6/13/20), actuators (Fig. 32, lines 7/14/21), and dependencies (Fig. 32, lines 8/15/22). It is worth mentioning that the dependencies are essential data for the device's functioning. So that the data referring to specific dependencies sensors and actuators, when updated, will be forwarded to the corresponding equipment.

The other configuration file, config.yaml, is presented in Figure 33. This file contains data regarding the necessary configurations to establish communication with the InfluxDB server, the MQTT broker, and the OPC UA server. For example, the broker's URL, port, user, password, and security certificates' location are described for the MQTT protocol (Figure 33, lines 2-8).

In addition to the configuration files already presented (Figure 32 and 33) the protocol translators use a dictionary file for their operation. The dictio-

Figure 33: Ontology YAML config file.

```

1 mqttConfig:
2   broker: localhost
3   port: 8883
4   user: iiot
5   passwd: interoperability
6   ca_cert_path: ../certificates/ca/ca.crt
7   cli_cert_path: ../certificates/client/iiotGateway.crt
8   cli_key_path: ../certificates/client/iiotGateway.key
9 opcuaConfig:
10  server: "opc.tcp://localhost:4840"
11 influxdbConfig:
12  url: "http://localhost:8086"
13  token: "#####"
14  org: "IIoIT"
15  bucket: "dissertationDB"

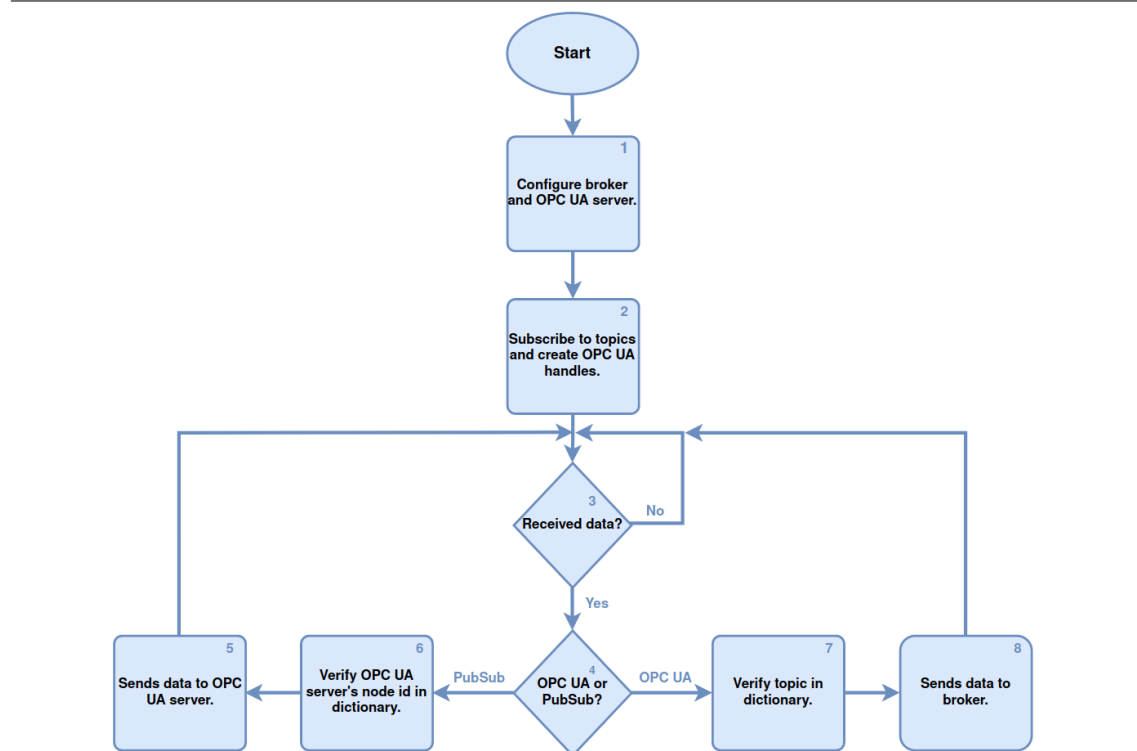
```

Source: The author.

nary indicates the OPC UA server's node id and data type for each sensor and actuator. Such node id and data type are required to use the write or read functions from the OPC UA protocol.

Both developed translators, MOB and DOB, are based on publisher and subscriber standard protocols. As a result, the structure of the converters follows a similar concept. Figure 1 presents the flowchart used for the translators.

Algorithm 1 Translator algorithms flowchart.



The first step comprises the configuration of the DDS or MQTT broker and the OPC UA server. Then, in step 2, the client subscribes to the topics of its dependencies, and a handle is created to monitor the update of its dependency

data directly from the OPC UA server. In the following step, the system waits until a new message arrives. When a message is received, the sender is verified to check if it comes from an MQTT/DDS node (identified as PubSub in the flowchart) or an OPC UA handler.

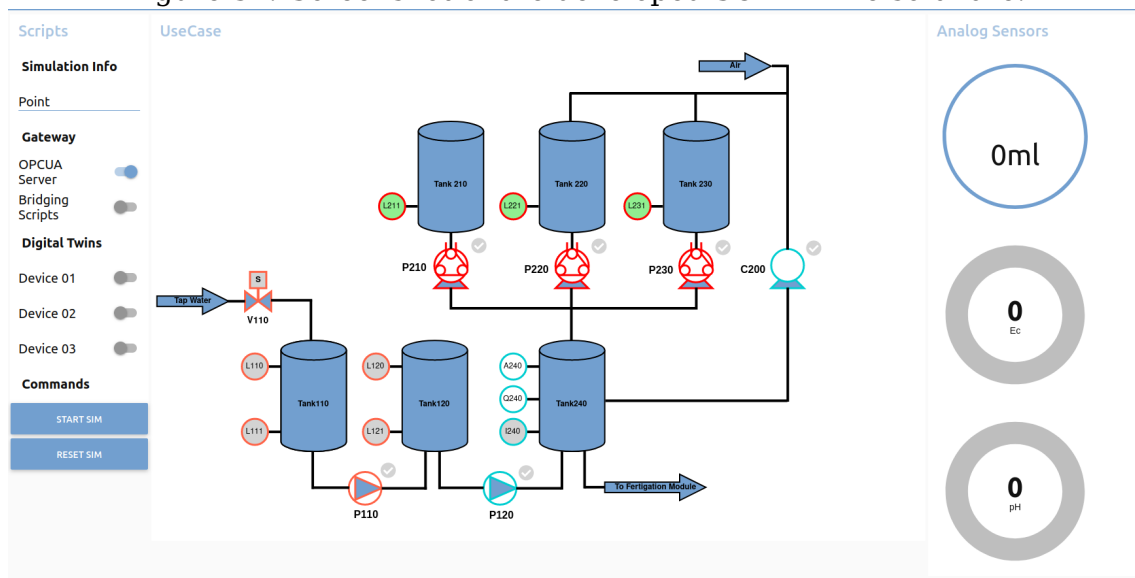
In case the data comes from a PubSub node, it is parsed, and the dictionary is used to check the node id for each sensor and actuator (step 6). The write function then sends the device data to the appropriate OPC UA server in the next step. The system returns to step 3 once the sending is completed.

However, in case the data received in step 3 refers to an OPC UA handle, the system will jump to step 7, where the topic referring to the sensor or actuator of the received data will be verified using the dictionary. The data will then be sent to a DDS or MQTT broker using the topic. After sending the message, the system returns to step 3.

6.6 SCADA

As mentioned in earlier chapters, to implement a friendly user interface, SCADA like software, a system widely used in the industry for its ability to control different functions and monitor various parameters, was developed. This software allows for monitoring and controlling an autonomous system in the context of agriculture – the Use Case described in Chapter 5.

Figure 34: Screenshot of the developed SCADA like software.



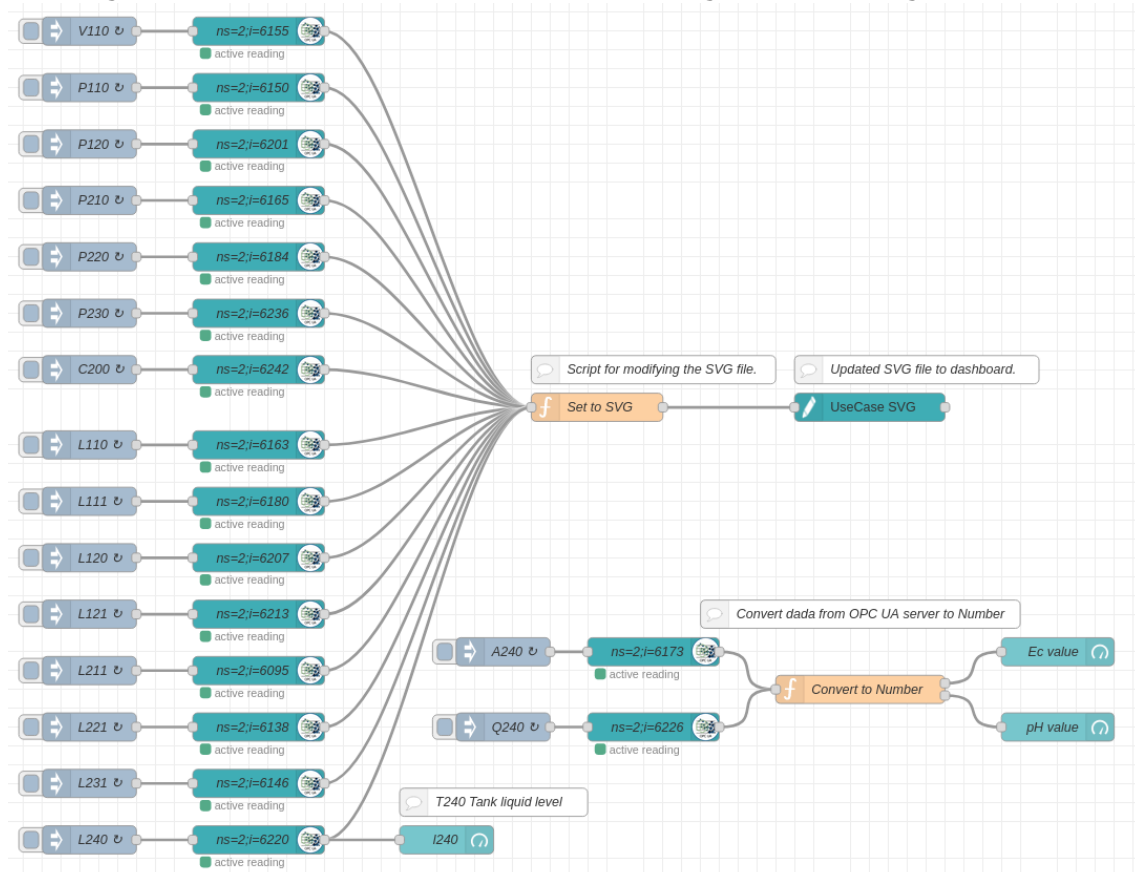
Source: The author.

The system is presented in Figure 34; it comprises an image of the nutrient solution module and UI gadgets, such as buttons and switches (node-red-dashboard), that allow the user to control the application remotely. The operator can monitor the status of valves and pumps, as well as the values measured by digital and analog sensors. The control functionalities are allowed by executing multiple scripts, such as starting the OPC UA server or

simulating the devices proposed in subsection 6.2. Furthermore, the user can configure the InfluxDB point's name where the experiment data is stored.

As previously mentioned, the SCADA like software was deployed using the Node-Red dashboard (Node-RED Org, 2016), hosted in the gateway. The system has two parts: the use case visualization and the execution of specific commands. The first (Figure 35) is responsible for displaying an image of the nutrient solution module, in SVG format, on the dashboard using the node-red-contrib-ui-svg library (BARTBUTENAERS, 2019), which allows modifying different aspects of the image using input data. In addition, several blocks are used, named with the OPC UA node id, whose function is to read data from the assets stored in the OPC UA server, node-red-contrib-opcua (KARAILA; LANDSDORF, 2019). This information is used to input a Javascript (JS) script that updates the Scalable Vector Graphics (SVG) image with sensor and actuators data. In addition, data from the A240, Q40, and L240 sensors are graphically presented.

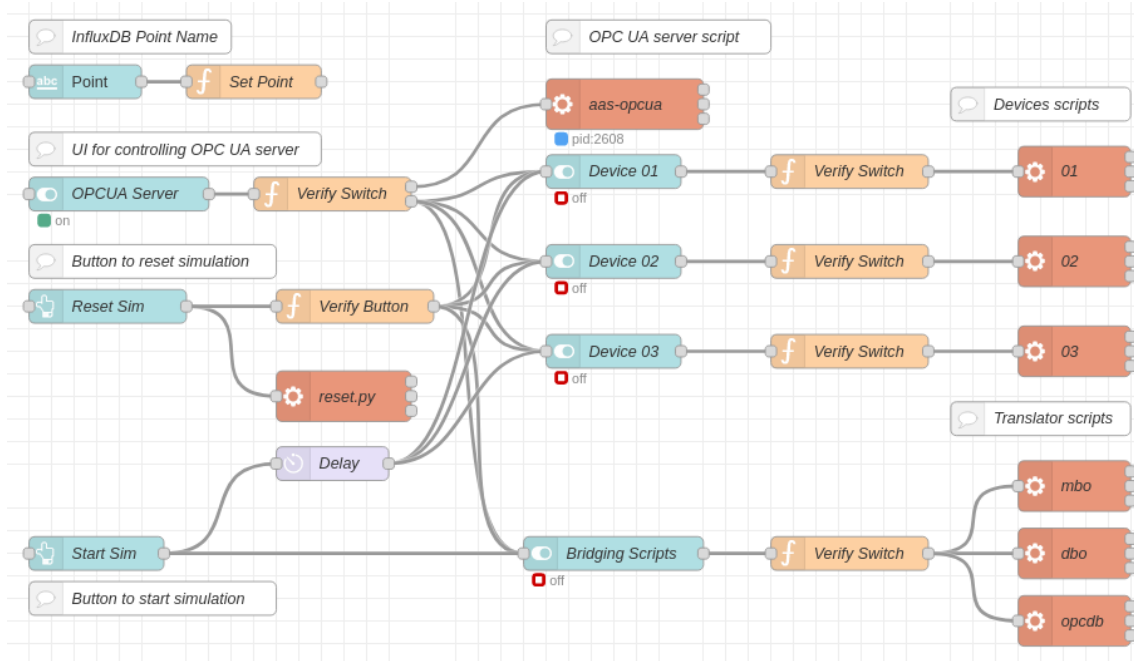
Figure 35: NodeRED block flow for creating and updating SVG data.



Source: The author.

The second part, the execution of specific commands (Figure 37), allows the operator to execute several scripts directly in the operating system through the exec nodes, represented by the color red. JS function blocks, blocks in orange, were developed to verify if the buttons were pressed and if the switches were activated. If the checks are valid, the scripts are executed. Two but-

Figure 36: NodeRED block flow for executing scripts in terminal.



Source: The author.

tons were created to facilitate the execution of the experiments. The START SIM button, when activated, starts the protocol translators' scripts and, after a delay of 2.5 seconds, starts the scripts for the three devices, starting the simulation. On the other hand, the RESET SIM button disables the protocol converters and scripts from the devices and returns the OPC UA server to its initial values.

7 EXPERIMENTS AND RESULTS

In this chapter the experiments conducted to validate the proposed middleware are described. Before executing the experiments, a tool was used to evaluate if the ontology presented in this work is correctly described. Using the system implementation described in section 6.2, a simple automation system to a simulated version of the use case described in chapter 5 was developed and some footprint and runtime characteristics such as the gateway's performance, the time required to complete protocol translations, and others were evaluated.

7.1 Ontology Evaluation

As a first step, it was evaluated whether the IIoT ontology was appropriately constructed and complained to OWL standards. For this purpose, a public domain tool that scans ontologies and checks for potential problems was used, OOPS! (OntOlogy Pitfall Scanner!) (POVEDA-VILLALÓN; GÓMEZ-PÉREZ; SUÁREZ-FIGUEROA, 2014). This tool detects the most common pitfalls in OWL documents, using the ontology URI or its RDF code.







Several problems might occur when modeling an ontology, as the developer has to create different classes and properties. Therefore, it is important to check the consistency of a designed ontology, such as the one developed in this work, before sharing it with implementation flaws. Some of the pitfalls verified by OPPS! are the following:

- The intersection of two or more classes is defined as the domain or range of a relationship. If those classes could not exchange instances, this warning could prevent difficulties with reasoning.
- The identifiers of the ontology elements do not follow any naming convention. The ontology's maintainability, accessibility, and clarity could all be improved in this scenario.

In addition, errors are classified according to their impact on the ontology: critical, important, and minor. Ontologies that have recognized critical pitfalls should be fixed before being shared with other users, as these issues might impair the consistency, reasoning, application, and other aspects of the ontology. On the other hand, important and minor errors do not significantly impact

the ontology. However, it is interesting that errors classified as important are also fixed.

Figure 37: Screenshot of the IIoT ontology OOPS! results.

Results for P04: Creating unconnected ontology elements.	1 case Minor 
Results for P08: Missing annotations.	183 cases Minor 
Results for P13: Inverse relationships not explicitly declared.	46 cases Minor 
Results for P20: Misusing ontology annotations.	5 cases Minor 
Results for P22: Using different naming conventions in the ontology.	ontology* Minor 
Results for P32: Several classes with the same label.	11 cases Minor 

Source: The author.

Only minor pitfalls were identified when applying the OOPS! tool to the IIoT ontology, such as missing annotations and employing alternative naming conventions. It is noteworthy that some of these pitfalls came from ontologies that were taken from the literature. Therefore, they were not corrected since it would not bring essential benefits to the project and would change ontologies developed by other authors and already used in other works.

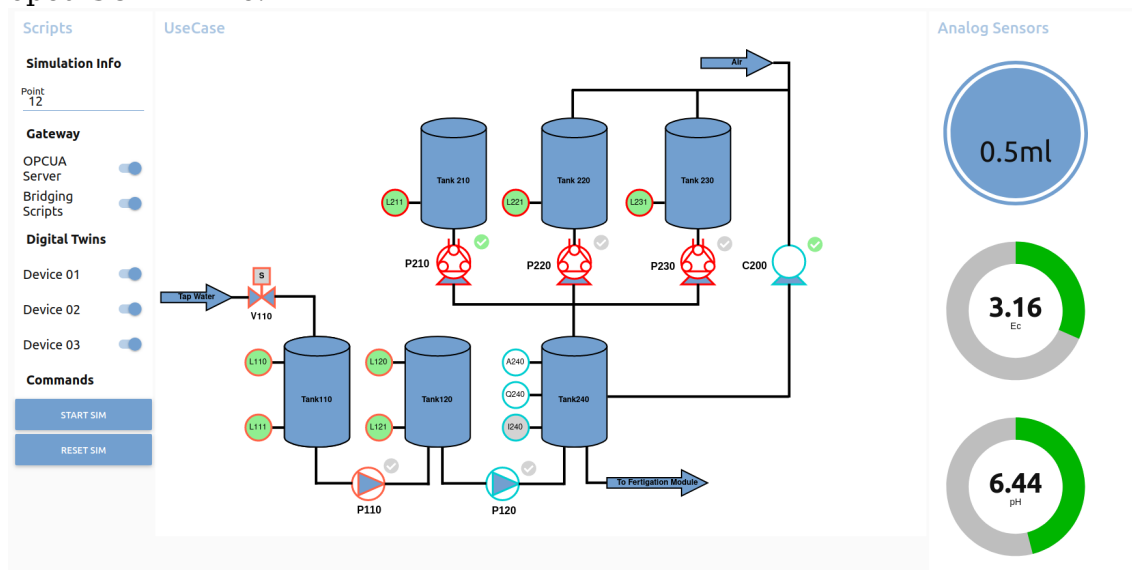
7.2 Use case simulation

The findings of the case study experiment are described in this section. The objective of these simulations is not for performance evaluation compared to related works, but to analyze the feasibility and viability of this dissertation proposed methodology on off-the-shelf hardware. To reproduce the use case simulations, the implemented SCADA like software (Figure 38) plays a vital role, as already mentioned in the previous chapters, since it is through it that the user can start the OPC UA server, protocol translators, and device scripts. Moreover, the software also allows the operator to monitor the exchanged data from sensors and actuators through the end devices. These experiments enabled the analysis of several aspects of the system, three of which will be evaluated:

- The simulation's conformance to the use case's predicted behavior pattern.
- The gateway's performance in terms of Central Processing Unit (CPU) and memory usage.
- The time required to complete the communication protocol translations.

A statistical analysis was performed to calculate the number of repetitions (samples) necessary for the experiment. The analysis was based on a significance level $\alpha = 0.05$, representing a confidence interval equal to $1 - \alpha = 0.95 = 95\%$. Data regarding the gateway's performance experiments was used for calculating the minimum number of repetitions for the experiments (n). The standard deviation for CPU and RAM memory was $\sigma \approx 0.32$ and $\sigma \approx 0.39$, that equals to $n = 19$. Then, 20 repetitions of the experiment were performed, each one lasting twenty minutes.

Figure 38: Screenshot of the Case study Simulation executing in the developed SCADA like.



Source: The author.

7.3 Simulation's conformance to the use case

The three scripts that emulate the tangible assets are employed in the experiments. These scripts represent the active component of a digital twin's virtual representation of an asset. The communication bridges translate the process data from the three devices and store it in the OPC UA server. The SCADA like system collects data from devices via the OPC UA server and allows users to keep track of their current state. All experiment data is saved locally at the specified InfluxDB point. Figure 39 depicts the information from all sensors and actuators from the three devices during the twelfth experiment.

Evaluating the data within the sample period makes it possible to identify seven plateaus: each indicates the $\text{pH} = 7$ (water pH value) of the solution in "Tank 240." This process occurs at the start of the nutrient solution manufacturing cycle. As a result, it is possible to state that seven solutions were created throughout the experiment. Subsequently, at least one production cycle could be completed, ensuring that the devices communicate with one another through different protocols. This demonstrated that the simulation process was executed as expected by the case study.

Furthermore, since InfluxDB adds tagging techniques, the user may also filter data according to the device's connection protocol, asset ID, and type (sensor or actuator). It is possible, for example, to filter data from a particular set of actuators communicating by MQTT protocol such as described in (Figure 40). Therefore, this communication protocol filter becomes a powerful tool to support developers in analyzing the correctness of the implemented system.

7.4 Gateway Performance

As already indicated, one of the aspects to be evaluated is the performance of the gateway that was implemented on a Raspberry Pi 4. During the experiment, the gateway includes all necessary software infrastructure for executing the three communication protocols mentioned previously in this work (DDS, MQTT, and OPC UA). It also included the scripts for the devices and protocol translators.

The Glances tool was utilized to assess CPU and memory performance information. Glances is a Python-based cross-platform monitoring application that uses the psutil package. Its interface provides information about system features such as processor usage (per Core, per processor), Random access memory (RAM) and swap memory usage, processor load, etc. This platform includes an XML-RPC server and a RESTful JSON API that can be used by other client software. It also allows the user to export system statistics to Comma-Separated Values (CSV), InfluxDB, Cassandra, and OpenTSDB, among other formats and servers.

The system performance results obtained in the twenty experiments were saved as a CSV file for further analysis. As already mentioned, for each experiment the CPU and RAM usage was averaged for the whole execution time. Data samples are depicted in Figures 41 and 42, respectively, for CPU and RAM performance.

As it can be observed, the average CPU usage was below 10% (more precisely between 8,81 to 9,97%) and the average RAM usage was below 17%, indicating that the overhead caused by the proposed gateway is low. That suggests that the middleware may be used in a more extensive system than the one described, with a more significant number of assets. Such scalability assessment of the middleware is beyond the scope of this dissertation, whose goal was to propose the interoperability middleware architecture and validate it using a case study.

7.5 Execution time for protocol translations

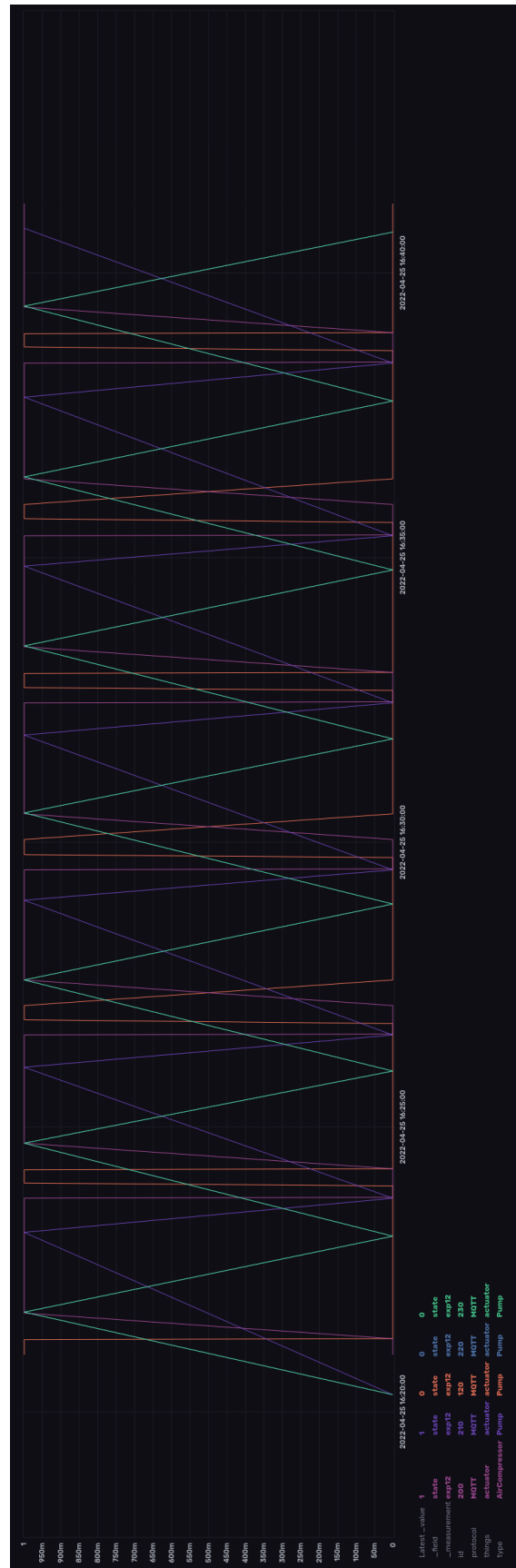
Several I4.0 applications are time critical, therefore an investigation on the required time for converting the proposed industrial communication protocols using the developed translators has been conducted. Initially, the time for exchanging messages between devices using the same protocol was evaluated. In the sequence, a similar analysis with devices communicating via distinct communication protocols using the MOB or DOB translators was conducted.

To standardize the tests, all messages exchanged during the simulation contained data from five sensors/actuators that remained unchanged during execution. It's worth noting that nodes using the DDS and MQTT protocols can convey all five pieces of data as a single message. However, OPC UA clients must write or read one variable at a time. The study was conducted by timestamping the database's recorded data. The average communication time and the jitter was calculated for fifty interactions for each protocol.

Testing devices using identical communication protocols determined that the delay between a publisher submitting a message and a subscriber receiving it was 2 ms for the DDS and MQTT protocols with a jitter not detectable with the timestamp resolution. In the case of the OPC UA protocol, the time required for writing data to the server and then reading was 3ms. Therefore, writing and reading the five pieces of information was concluded in fifteen milliseconds. According to the DOB translation testing findings, it took 17ms from publishing a message by the DDS publisher until this data arrives on the OPC UA server. And for the MOB translator is required 32 ms to read data from an MQTT publisher into the OPC UA server.

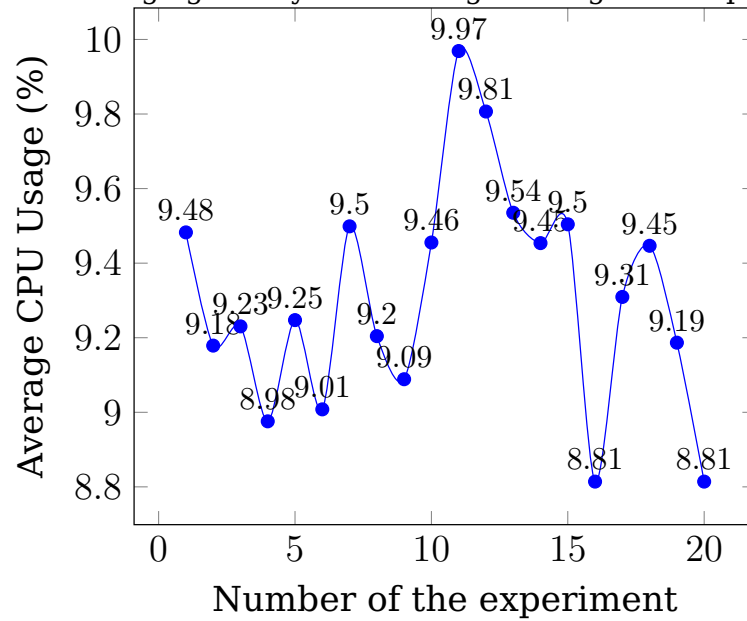
After verifying the values obtained during the experiments, it is possible to identify that the times required to perform the protocol conversions are, as expected, higher when compared to the communication time with devices with the same protocol. However, the time required is still in the sub-second range; they are compatible with the requirements of a variety of industrial applications, including the suggested use case.

Figure 40: Simulation data from 12th experiment in influxDB using data filtering.



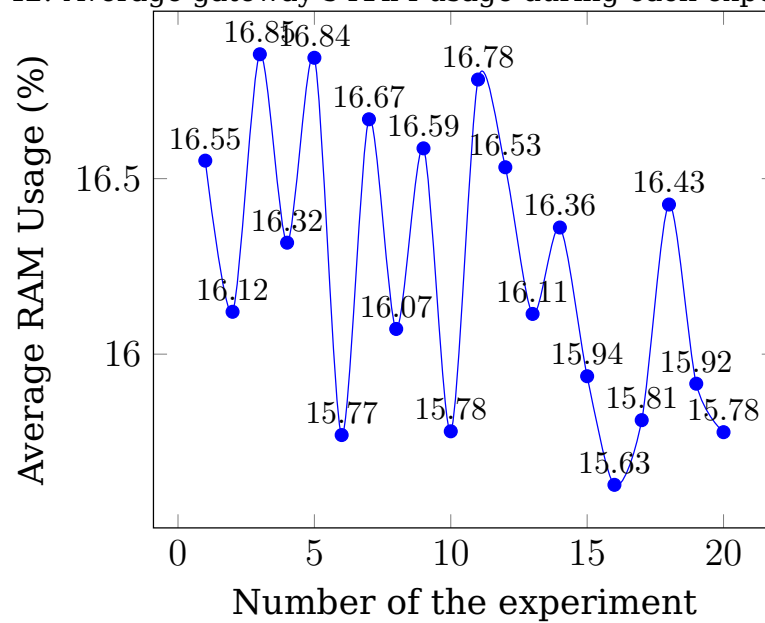
Source: The author.

Figure 41: Average gateway's CPU usage during each experiment.



Source: The author.

Figure 42: Average gateway's RAM usage during each experiment.



Source: The author.

8 CONCLUSIONS AND FUTURE WORKS

This work presents a middleware development for IIoT gateways whose primary goal is to solve interoperability problems in Industry 4.0 domain applications. Its approach adopts ontologies based on international standards (IEEE) and ontologies widely used in the research area, combined with standardized digital representation (AAS) and communication protocols translators to mitigate device, syntactical and semantic interoperability issues.

Using a semantic model, the so-called IIoT ontology, the user can define a system at a high level of abstraction, with the depth required for each project, promoting a system-wide vocabulary for addressing semantic issues. Combining the system description with protocol translators allows for an industrial design with multiple assets enabling an interoperable operation using heterogeneous communication protocols and data structures. Besides, it also provides a digital representation of the industrial assets' most relevant information through the asset administration shell (AAS) concept. In this way, an asset's digital version is created (digital twin). Hence, the proposed middleware improves the development of industrial automation applications by reducing the time required to build new applications and facilitating the development of maintenance applications.

The middleware was deployed on low-cost embedded hardware architecture and validated using a simulated use case. The simulations were conducted using SCADA like software developed specifically for this work, which allows the end-user to perform online monitoring of the system. During the experiments, several metrics were assessed, including the gateway's CPU and memory usage and relevant timing information on the messages exchanged between system devices. Furthermore, a software was applied to the IIoT ontology to detect the most typical mistakes in ontology development.

The simulation results indicate that combining the IIoT ontology with AAS is beneficial for dealing with interoperability concerns in industrial applications. The obtained gateway's performance was adequate for the application and that the proposed concepts for protocols translation allowed a smooth communication. The ontology verification stated that the IIoT ontology was developed according to standards and just required minor tweaks, such as adding comments to classes. The functionality of the developed protocol translators was also proven, which allowed the conversion of Industry 4.0 communication protocols standards: DDS, OPC UA, and MQTT.

The two main contributions of this work are a general Industry 4.0/IIoT oriented ontology based on international standards and a middleware implementation that combines the developed ontology with AAS to mitigate interoperability problems in an industrial environment. The middleware was evaluated in a specific use case. Still, its structure was designed for general industrial applications, allowing the system's diversity, such as the number of devices, the communication protocols, and hardware architecture.

Some possible future work activities that can be developed based on the results obtained in this work are:

- Deploy the gateway used in the simulated use case in a physical plant. The results from both experiments should be compared to increase the confidence in the simulated use case directly in the middleware.
- Extend the IIoT ontology's implementation to other areas of application, having expert domains adding new classes and properties.
- Perform a more detailed description of the assets to create a digital twin similar to its physical form, as possible, by adding new submodules in its AAS.
- Automatic generation of the AAS model files autonomously using a parser based on the ontology configuration files.
- Perform intervention time-series analyses of the industrial assets' data stored in the database through its timestamp.

REFERENCES

- 63088.(2017)., I. P. **Smart Manufacturing-Reference Architecture Model Industry 4.0 (RAMI4. 0)**. 2017.
- AAZAM, M.; ZEADALLY, S.; HARRAS, K. A. Deploying fog computing in industrial internet of things and industry 4.0. **IEEE Transactions on Industrial Informatics**, [S.l.], v.14, n.10, p.4674–4682, 2018.
- AGARWAL, R. et al. Unified IoT ontology to enable interoperability and federation of testbeds. In: IEEE 3RD WORLD FORUM ON INTERNET OF THINGS (WF-IOT), 2016., 2016. **Anais. . .** [S.l.: s.n.], 2016. p.70–75.
- AL-FUQAHA, A. et al. Toward better horizontal integration among IoT services. **IEEE Communications Magazine**, [S.l.], v.53, n.9, p.72–79, 2015.
- ANTONIOU, G.; HARMELEN, F. v. Web ontology language: owl. In: **Handbook on ontologies**. [S.l.]: Springer, 2004. p.67–92.
- ARM, J. et al. Automated design and integration of asset administration shells in components of industry 4.0. **Sensors**, [S.l.], v.21, n.6, p.2004, 2021.
- ATZORI, L.; IERA, A.; MORABITO, G. Understanding the Internet of Things: definition, potentials, and societal role of a fast evolving paradigm. **Ad Hoc Networks**, [S.l.], v.56, p.122–140, 2017.
- BABUN, L. et al. A survey on IoT platforms: communication, security, and privacy perspectives. **Computer Networks**, [S.l.], v.192, p.108040, 2021.
- BADER, S. R.; MALESHKOVA, M. The semantic asset administration shell. In: INTERNATIONAL CONFERENCE ON SEMANTIC SYSTEMS, 2019. **Anais. . .** [S.l.: s.n.], 2019. p.159–174.
- BAKKEN, D. E. Encyclopedia of distributed computing. **Kluwer Academic Press, 2001, ch. Middleware**, [S.l.], 2001.
- BANDYOPADHYAY, S. et al. Role of middleware for internet of things: a study. **International Journal of Computer Science and Engineering Survey**, [S.l.], v.2, n.3, p.94–105, 2011.
- BANKS, A.; GUPTA, R. MQTT Version 3.1. 1. **OASIS standard**, [S.l.], v.29, p.89, 2014.

BARBIERI, G. et al. A mathematical model to enable the virtual commissioning simulation of wick soilless cultivations. **J. Eng. Sci. Technol**, [S.l.], v.16, p.3325–3342, 2021.

BARNSTEDT, E. et al. Details of the asset administration shell. **Tech. Rep.**, [S.l.], 2018.

BARRICELLI, B. R.; CASIRAGHI, E.; FOGLI, D. A survey on digital twin: definitions, characteristics, applications, and design implications. **IEEE access**, [S.l.], v.7, p.167653–167671, 2019.

BARTBUTENAERS. **node-red-contrib-ui-svg**. Available at: <<https://github.com/bartbutenaers/node-red-contrib-ui-svg>>. Accessed on: 16 march 2022.

BENITEZ, G. B.; AYALA, N. F.; FRANK, A. G. Industry 4.0 innovation ecosystems: an evolutionary perspective on value cocreation. **International Journal of Production Economics**, [S.l.], v.228, p.107735, 2020.

BERMUDEZ-EDO, M. et al. IoT-Lite: a lightweight semantic model for the internet of things. In: INTL IEEE CONFERENCES ON UBIQUITOUS INTELLIGENCE & COMPUTING, ADVANCED AND TRUSTED COMPUTING, SCALABLE COMPUTING AND COMMUNICATIONS, CLOUD AND BIG DATA COMPUTING, INTERNET OF PEOPLE, AND SMART WORLD CONGRESS (UIC/ATC/SCALCOM/CBDCOM/IOP/SMARTWORLD), 2016., 2016. **Anais. . .** [S.l.: s.n.], 2016. p.90–97.

BERNERS-LEE, T. **Weaving the Web**: the original design and ultimate destiny of the world wide web by its inventor. [S.l.]: Harper San Francisco, 1999.

BOUTER, C. et al. Towards a Comprehensive Methodology for Modelling Submodels in the Industry 4.0 Asset Administration Shell. In: IEEE 23RD CONFERENCE ON BUSINESS INFORMATICS (CBI), 2021., 2021. **Anais. . .** [S.l.: s.n.], 2021. v.2, p.10–19.

BOYES, H. et al. The industrial internet of things (IIoT): an analysis framework. **Computers in industry**, [S.l.], v.101, p.1–12, 2018.

BRAY, T. **The javascript object notation (json) data interchange format**. [S.l.: s.n.], 2014.

BRETTEL, M. et al. How virtualization, decentralization and network building change the manufacturing landscape: an industry 4.0 perspective. **International Journal of Information and Communication Engineering**, [S.l.], v.8, n.1, p.37–44, 2014.

BUDAKOTI, J.; GAUR, A. S.; LUNG, C.-H. IoT gateway middleware for SDN managed IoT. In: IEEE INTERNATIONAL CONFERENCE ON INTERNET OF THINGS (ITHINGS) AND IEEE GREEN COMPUTING AND

COMMUNICATIONS (GREENCOM) AND IEEE CYBER, PHYSICAL AND SOCIAL COMPUTING (CPSCOM) AND IEEE SMART DATA (SMARTDATA), 2018., 2018. **Anais...** [S.l.: s.n.], 2018. p.154–161.

BUTUN, I. et al. **Industrial IoT**. [S.l.]: Springer, 2020.

Canonical Ltd. **Ubuntu Server 20.04 LTS**. Available at: <<http://ftp.jaist.ac.jp/pub/Linux/ubuntu-cdimage/ubuntu-legacy-server/releases/20.04/release/>>. Accessed on: 16 march 2022.

CASTELLANOS, W. et al. Internet of things: a multiprotocol gateway as solution of the interoperability problem. **arXiv preprint arXiv:2108.00098**, [S.l.], 2021.

CHANDRASEKARAN, B.; JOSEPHSON, J. R.; BENJAMINS, V. R. What are ontologies, and why do we need them? **IEEE Intelligent Systems and their applications**, [S.l.], v.14, n.1, p.20–26, 1999.

COMPTON, M. et al. The SSN ontology of the W3C semantic sensor network incubator group. **Journal of Web Semantics**, [S.l.], v.17, p.25–32, 2012.

CONWAY, J. The Industrial Internet of Things: an evolution to a smart manufacturing enterprise. **Schneider Electric**, [S.l.], 2016.

DEERING, S.; HINDEN, R. et al. **Internet protocol, version 6 (IPv6) specification**. [S.l.]: RFC 2460, december, 1998.

DERHAMY, H.; ELIASSON, J.; DELSING, J. IoT interoperability—on-demand and low latency transparent multiprotocol translator. **IEEE Internet of Things Journal**, [S.l.], v.4, n.5, p.1754–1763, 2017.

DESAI, P.; SHETH, A.; ANANTHARAM, P. Semantic gateway as a service architecture for iot interoperability. In: IEEE INTERNATIONAL CONFERENCE ON MOBILE SERVICES, 2015., 2015. **Anais...** [S.l.: s.n.], 2015. p.313–319.

DI MARTINO, B. et al. Internet of things reference architectures, security and interoperability: a survey. **Internet of Things**, [S.l.], v.1, p.99–112, 2018.

DIZDAREVIĆ, J. et al. A survey of communication protocols for internet of things and related challenges of fog and cloud computing integration. **ACM Computing Surveys (CSUR)**, [S.l.], v.51, n.6, p.1–29, 2019.

Eclipse Foundation. **Eclipse Paho™ MQTT Python Client**. Available at: <<https://github.com/eclipse/paho.mqtt.python>>. Accessed on: 16 march 2022.

EL-GOHARY, N. M.; EL-DIRABY, T. E. Domain ontology for processes in infrastructure and construction. **Journal of Construction Engineering and Management**, [S.l.], v.136, n.7, p.730–744, 2010.

EMMRICH, V. et al. Geschäftsmodell-Innovation durch Industrie 4.0: chancen und risiken für den maschinen-und anlagenbau. **München, Stuttgart: Dr. Wieselhuber & Partner, Fraunhofer IPA**, [S.l.], 2015.

ENDELEY, R. et al. A smart gateway enabling opc ua and dds interoperability. In: IEEE SMARTWORLD, UBIQUITOUS INTELLIGENCE & COMPUTING, ADVANCED & TRUSTED COMPUTING, SCALABLE COMPUTING & COMMUNICATIONS, CLOUD & BIG DATA COMPUTING, INTERNET OF PEOPLE AND SMART CITY INNOVATION (SMARTWORLD/SCALCOM/UIC/ATC/CBDCOM/IOP/SCI), 2019., 2019. **Anais. . .** [S.l.: s.n.], 2019. p.88–93.

FERREIRA, P. et al. Multi-Protocol LoRaWAN/Wi-Fi Sensor Node Performance Assessment for Industry 4.0 Energy Monitoring. In: IEEE-APS TOPICAL CONFERENCE ON ANTENNAS AND PROPAGATION IN WIRELESS COMMUNICATIONS (APWC), 2019., 2019. **Anais. . .** [S.l.: s.n.], 2019. p.403–407.

FIGUEROA-LORENZO, S.; AÑORGA, J.; ARRIZABALAGA, S. A survey of IIoT protocols: a measure of vulnerability risk analysis based on cvss. **ACM Computing Surveys (CSUR)**, [S.l.], v.53, n.2, p.1–53, 2020.

FIORINI, S. R. et al. A suite of ontologies for robotics and automation [industrial activities]. **IEEE Robotics & Automation Magazine**, [S.l.], v.24, n.1, p.8–11, 2017.

FRANK, A. G.; DALENOGARE, L. S.; AYALA, N. F. Industry 4.0 technologies: implementation patterns in manufacturing companies. **International Journal of Production Economics**, [S.l.], v.210, p.15–26, 2019.

GÓMEZ-PÉREZ, A.; BENJAMINS, R. Overview of Knowledge Sharing and Reuse Components: ontologies and problem-solving methods. In: INTERNATIONAL JOINT CONFERENCE ON ARTIFICIAL INTELLIGENCE (IJCAI'99) WORKSHOP KRR5: ONTOLOGIES AND PROBLEM-SOLVING METHODS: LESSON LEARNED AND FUTURE TRENDS, 16., 1999. **Proceedings. . .** IJCAI and the Scandinavian AI Societies. CEUR Workshop Proceedings, 1999. v.18. Ontology Engineering Group ? OEG.

GORECKY, D. et al. Human-machine-interaction in the industry 4.0 era. In: IEEE INTERNATIONAL CONFERENCE ON INDUSTRIAL INFORMATICS (INDIN), 2014., 2014. **Anais. . .** [S.l.: s.n.], 2014. p.289–294.

GRANGEL-GONZÁLEZ, I. et al. An RDF-based approach for implementing industry 4.0 components with Administration Shells. In: IEEE 21ST INTERNATIONAL CONFERENCE ON EMERGING TECHNOLOGIES AND FACTORY AUTOMATION (ETF A), 2016., 2016. **Anais. . .** [S.l.: s.n.], 2016. p.1–8.

GRIEVES, M. Digital twin: manufacturing excellence through virtual factory replication. **White paper**, [S.l.], v.1, n.2014, p.1–7, 2014.

GUARINO, N.; OBERLE, D.; STAAB, S. What is an ontology? In: **Handbook on ontologies**. [S.l.]: Springer, 2009. p.1–17.

GUBBI, J. et al. Internet of Things (IoT): a vision, architectural elements, and future directions. **Future generation computer systems**, [S.l.], v.29, n.7, p.1645–1660, 2013.

GUINARD, D.; TRIFA, V. Towards the web of things: web mashups for embedded devices. In: WORKSHOP ON MASHUPS, ENTERPRISE MASHUPS AND LIGHTWEIGHT COMPOSITION ON THE WEB (MEM 2009), IN PROCEEDINGS OF WWW (INTERNATIONAL WORLD WIDE WEB CONFERENCES), MADRID, SPAIN, 2009. **Anais...** [S.l.: s.n.], 2009. v.15, p.8.

HARISH, K. **Data Distribution Service Based Communication Middleware for Network of Systems**. 2015. Tese (Doutorado em Engenharia Elétrica) — , 2015.

HELMIÖ, P. et al. **Open source in industrial internet of things: a systematic literature review**. 2017. Dissertação (Mestrado em Engenharia Elétrica) — LUT University, Lappeenranta, 2017.

HERMANN, M.; PENTEK, T.; OTTO, B. Design principles for industrie 4.0 scenarios. In: HAWAII INTERNATIONAL CONFERENCE ON SYSTEM SCIENCES (HICSS), 2016., 2016. **Anais...** [S.l.: s.n.], 2016. p.3928–3937.

HINZE, A.; SACHS, K.; BUCHMANN, A. Event-based applications and enabling technologies. In: THIRD ACM INTERNATIONAL CONFERENCE ON DISTRIBUTED EVENT-BASED SYSTEMS, 2009. **Proceedings...** [S.l.: s.n.], 2009. p.1–15.

HOEHNDORF, R. What is an upper level ontology? **Ontogenesis**, [S.l.], 2010.

HOLLER, J. et al. **Internet of things**. [S.l.]: Academic Press, 2014.

IEEE. IEEE Standard Ontologies for Robotics and Automation. **IEEE Std 1872-2015**, [S.l.], p.1–60, 2015.

NOF, S. Y. (Ed.). **Industrial Communication Protocols**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009. p.981–999.

INGLÉS-ROMERO, J. F. et al. A model-driven approach to enable adaptive qos in dds-based middleware. **IEEE Transactions on Emerging Topics in Computational Intelligence**, [S.l.], v.1, n.3, p.176–187, 2017.

IÑIGO, M. A. et al. Towards an Asset Administration Shell scenario: a use case for interoperability and standardization in industry 4.0. In: NOMS 2020-2020 IEEE/IFIP NETWORK OPERATIONS AND MANAGEMENT SYMPOSIUM, 2020. **Anais...** [S.l.: s.n.], 2020. p.1–6.

INTEROPERABILITY, I. Guest editorial semantic technologies in automation systems. **IEEE Transactions on Industrial Informatics**, [S.l.], v.13, n.6, p.3335, 2017.

JACOBS, I. About the world wide web consortium (W3C). **The World Wide Web Consortium**, [S.l.], 2001.

JAZDI, N. Cyber physical systems in the context of Industry 4.0. In: IEEE INTERNATIONAL CONFERENCE ON AUTOMATION, QUALITY AND TESTING, ROBOTICS, 2014., 2014. **Anais. . .** [S.l.: s.n.], 2014. p.1–4.

JIA, Y. et al. Improved reliability of large scale publish/subscribe based moms using model checking. In: IEEE NETWORK OPERATIONS AND MANAGEMENT SYMPOSIUM (NOMS), 2014., 2014. **Anais. . .** [S.l.: s.n.], 2014. p.1–8.

KAGERMANN, H. et al. **Recommendations for implementing the strategic initiative INDUSTRIE 4.0**: securing the future of german manufacturing industry; final report of the industrie 4.0 working group. [S.l.]: Forschungsunion, 2013.

KAIYA, H.; SAEKI, M. Using domain ontology as domain knowledge for requirements elicitation. In: IEEE INTERNATIONAL REQUIREMENTS ENGINEERING CONFERENCE (RE'06), 14., 2006. **Anais. . .** [S.l.: s.n.], 2006. p.189–198.

KANG, B.; CHOO, H. An experimental study of a reliable IoT gateway. **ICT Express**, [S.l.], v.4, n.3, p.130–133, 2018.

KANNOTH, S.; SCHNICKE, F.; ANTONINO, P. O. Enabling industry 4.0 communication protocol interoperability: an opc ua case study. In: CONFERENCE ON THE ENGINEERING OF COMPUTER BASED SYSTEMS, 7., 2021. **Anais. . .** [S.l.: s.n.], 2021. p.1–9.

KARAILA, M.; LANDSDORF, K. **node-red-contrib-opcua**. Available at: <<https://github.com/mikakaraila/node-red-contrib-opcua>>. Accessed on: 16 march 2022.

KUMAR, V. R. S. et al. Ontologies for industry 4.0. **The Knowledge Engineering Review**, [S.l.], v.34, 2019.

LACY, L. W. **OWL**: representing information using the web ontology language. [S.l.]: Trafford Publishing, 2005.

LASI, H. et al. Industry 4.0. **Business & information systems engineering**, [S.l.], v.6, n.4, p.239–242, 2014.

LEA, R.; BLACKSTOCK, M. City hub: a cloud-based iot platform for smart cities. In: IEEE 6TH INTERNATIONAL CONFERENCE ON CLOUD COMPUTING TECHNOLOGY AND SCIENCE, 2014., 2014. **Anais. . .** [S.l.: s.n.], 2014. p.799–804.

- LEE, J.; BAGHERI, B.; KAO, H.-A. A cyber-physical systems architecture for industry 4.0-based manufacturing systems. **Manufacturing letters**, [S.l.], v.3, p.18–23, 2015.
- LI, S.; XU, L. D.; ZHAO, S. The internet of things: a survey. **Information systems frontiers**, [S.l.], v.17, n.2, p.243–259, 2015.
- LIAO, Y. et al. Past, present and future of Industry 4.0-a systematic literature review and research agenda proposal. **International journal of production research**, [S.l.], v.55, n.12, p.3609–3629, 2017.
- LIN, Z.; PEARSON, S. et al. An inside look at industrial Ethernet communication protocols. **Texas Instruments, White Paper**, [S.l.], 2013.
- LOCKE, D. Mq telemetry transport (mqtt) v3. 1 protocol specification. **IBM developerWorks Technical Library**, [S.l.], v.15, 2010.
- LÜDER, A. et al. Generating industry 4.0 asset administration shells with data from engineering data logistics. In: IEEE INTERNATIONAL CONFERENCE ON EMERGING TECHNOLOGIES AND FACTORY AUTOMATION (ETFA), 2020., 2020. **Anais. . .** [S.l.: s.n.], 2020. v.1, p.867–874.
- MAHNKE, W.; LEITNER, S.-H.; DAMM, M. **OPC unified architecture**. [S.l.]: Springer Science & Business Media, 2009.
- MANYIKA, J. et al. **The Internet of Things**: mapping the value beyond the hype. [S.l.]: McKinsey Global Institute New York, NY, USA, 2015. v.24.
- MARTIN, D. et al. Bringing semantics to web services: the owl-s approach. In: INTERNATIONAL WORKSHOP ON SEMANTIC WEB SERVICES AND WEB PROCESS COMPOSITION, 2004. **Anais. . .** [S.l.: s.n.], 2004. p.26–42.
- MAYER, S. et al. An open semantic framework for the industrial Internet of Things. **IEEE Intelligent Systems**, [S.l.], v.32, n.1, p.96–101, 2017.
- 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**, [S.l.], v.108, n.10, p.1785–1824, 2020.
- MUMTAZ, S. et al. Massive Internet of Things for industrial applications: addressing wireless iiot connectivity challenges and ecosystem fragmentation. **IEEE Industrial Electronics Magazine**, [S.l.], v.11, n.1, p.28–33, 2017.
- MUSEN, M. A. Automated support for building and extending expert models. In: **Knowledge Acquisition**: selected research and commentary. [S.l.]: Springer, 1989. p.101–129.
- NAGORNY, K. et al. Semantical support for a CPS data marketplace to prepare Big Data analytics in smart manufacturing environments. In: IEEE INDUSTRIAL CYBER-PHYSICAL SYSTEMS (ICPS), 2018., 2018. **Anais. . .** [S.l.: s.n.], 2018. p.206–211.

NEELY, S.; DOBSON, S.; NIXON, P. Adaptive middleware for autonomic systems. In: ANNALES DES TÉLÉCOMMUNICATIONS, 2006. **Anais. . .** [S.l.: s.n.], 2006. v.61, n.9, p.1099–1118.

NEGASH, B.; WESTERLUND, T.; TENHUNEN, H. Towards an interoperable Internet of Things through a web of virtual things at the Fog layer. **Future Generation Computer Systems**, [S.l.], v.91, p.96–107, 2019.

Node-RED Org. **node-red-dashboard**. Available at: <<https://flows.nodered.org/node/node-red-dashboard>>. Accessed on: 16 march 2022.

NOURA, M.; ATIQUZZAMAN, M.; GAEDKE, M. Interoperability in internet of things: taxonomies and open challenges. **Mobile networks and applications**, [S.l.], v.24, n.3, p.796–809, 2019.

NOY, N. F. et al. Protégé-2000: an open-source ontology-development and knowledge-acquisition environment. In: AMIA... ANNUAL SYMPOSIUM PROCEEDINGS. AMIA SYMPOSIUM, 2003. **Anais. . .** [S.l.: s.n.], 2003. p.953–953.

O'LEARY, N.; CONWAY-JONES, D. **Node-RED programming tool**. Available at: <<https://github.com/node-red/node-red>>. Accessed on: 16 march 2022.

OLSZEWSKA, J. I. et al. Ontology for autonomous robotics. In: IEEE INTERNATIONAL SYMPOSIUM ON ROBOT AND HUMAN INTERACTIVE COMMUNICATION (RO-MAN), 2017., 2017. **Anais. . .** [S.l.: s.n.], 2017. p.189–194.

OPEN62541. **The Open source implementation of OPC UA, open62541**. Available at: <<http://www.open62541.org/>>. Accessed on: 16 march 2022.

PALAVRAS, E. et al. Semibiot: secure multi-protocol integration bridge for the iot. In: IEEE INTERNATIONAL CONFERENCE ON COMMUNICATIONS (ICC), 2018., 2018. **Anais. . .** [S.l.: s.n.], 2018. p.1–7.

PAPCUN, P. et al. Edge-enabled IoT gateway criteria selection and evaluation. **Concurrency and Computation: Practice and Experience**, [S.l.], v.32, n.13, p.e5219, 2020.

PEREIRA, P. M. **IMIIoTG**. Available at: <<https://github.com/morganpereira/IMIIoTG>>. Accessed on: 07 july 2022.

PERERA, C. et al. Context aware computing for the internet of things: a survey. **IEEE communications surveys & tutorials**, [S.l.], v.16, n.1, p.414–454, 2013.

Plattform Industrie 4.0. **Generic Frame for Technical Data for Industrial Equipment in Manufacturing (Version 1.1)**. Available at:

<https://github.com/admin-shell-io/submodel-templates/blob/main/published/Technical_Data/1/1/SMT_Technical_Data_V11.pdf >
.Accessed on : 14march2022.

Plattform Industrie 4.0. **ZVEI Digital Nameplate for industrial equipment (Version 1.0)**. Available at:

<https://github.com/admin-shell-io/submodel-templates/blob/main/published/ZVEI_Digital_Nameplate/1/0/SMT_ZVEI_Digital_Nameplate_V10.pdf >
.Accessed on : 14march2022.

POVEDA-VILLALÓN, M.; GÓMEZ-PÉREZ, A.; SUÁREZ-FIGUEROA, M. C. OOPS! (Ontology Pitfall Scanner!): an on-line tool for ontology evaluation. **International Journal on Semantic Web and Information Systems (IJSWIS)**, [S.l.], v.10, n.2, p.7–34, 2014.

PRESTES, E.; FIORINI, S. R.; CARBONERA, J. Core ontology for robotics and automation. In: WORKSHOP ON KNOWLEDGE REPRESENTATION AND ONTOLOGIES FOR ROBOTICS AND AUTOMATION, 18., 2014. **Proceedings...** [S.l.: s.n.], 2014. p.7.

PROFANTER, S. et al. OPC UA versus ROS, DDS, and MQTT: performance evaluation of industry 4.0 protocols. In: IEEE INTERNATIONAL CONFERENCE ON INDUSTRIAL TECHNOLOGY (ICIT), 2019., 2019. **Anais...** [S.l.: s.n.], 2019. p.955–962.

RADATZ, J.; GERACI, A.; KATKI, F. IEEE standard glossary of software engineering terminology. IEEE Std. 610.12-1990. **Computer Society of the IEEE**, [S.l.], 1990.

RAZZAQUE, M. A. et al. Middleware for internet of things: a survey. **IEEE Internet of things journal**, [S.l.], v.3, n.1, p.70–95, 2015.

RICHTER, A.; KOCH, M. Functions of social networking services. **From CSCW to Web 2.0: European Developments in Collaborative Design Selected Papers from COOP08**, [S.l.], 2008.

RIJGERSBERG, H.; VAN ASSEM, M.; TOP, J. Ontology of units of measure and related concepts. **Semantic Web**, [S.l.], v.4, n.1, p.3–13, 2013.

ROS2, R. O. S. . **The Robot Operating System Galactic Geochelone**. Available at: <<https://docs.ros.org/en/galactic/index.html>>. Accessed on: 16 march 2022.

ROSE, K.; ELDRIDGE, S.; CHAPIN, L. The internet of things: an overview. **The internet society (ISOC)**, [S.l.], v.80, p.1–50, 2015.

SAKURADA, L.; LEITÃO, P. Multi-agent systems to implement industry 4.0 components. In: IEEE CONFERENCE ON INDUSTRIAL CYBERPHYSICAL SYSTEMS (ICPS), 2020., 2020. **Anais...** [S.l.: s.n.], 2020. v.1, p.21–26.

SCHROEDER, G. N. et al. Digital twin data modeling with automationml and a communication methodology for data exchange. **IFAC-PapersOnLine**, [S.l.], v.49, n.30, p.12–17, 2016.

SCHWEICHHART, K. Reference architectural model industrie 4.0 (rami 4.0). **An Introduction. Available online: <https://www.plattform-i40.de>**, [S.l.], v.40, 2016.

SENGUPTA, J.; RUJ, S.; BIT, S. D. A comprehensive survey on attacks, security issues and blockchain solutions for IoT and IIoT. **Journal of Network and Computer Applications**, [S.l.], v.149, p.102481, 2020.

SHA, L. et al. Cyber-physical systems: a new frontier. In: IEEE INTERNATIONAL CONFERENCE ON SENSOR NETWORKS, UBIQUITOUS, AND TRUSTWORTHY COMPUTING (SUTC 2008), 2008., 2008. **Anais...** [S.l.: s.n.], 2008. p.1–9.

SHENG, Z. et al. A survey on the ietf protocol suite for the internet of things: standards, challenges, and opportunities. **IEEE wireless communications**, [S.l.], v.20, n.6, p.91–98, 2013.

SIEGEL, J. OMG Overview: corba and the oma in enterprise computing. **Communications of the ACM**, [S.l.], v.41, n.10, p.37–43, 1998.

SIEMENS. **Siemens OPC UA Modeling Editor V2.5**. Available at: <[https://support.industry.siemens.com/cs/document/109755133/siemens-opc-ua-modeling-editor-\(siome\)-for-implementing-opc-ua-companion-specifications?dti=0lc=en-WW](https://support.industry.siemens.com/cs/document/109755133/siemens-opc-ua-modeling-editor-(siome)-for-implementing-opc-ua-companion-specifications?dti=0lc=en-WW)>. Accessed on: 16 march 2022.

SOWA, J. F. Top-level ontological categories. **International journal of human-computer studies**, [S.l.], v.43, n.5-6, p.669–685, 1995.

Stanford. **site-protege**. Available at: <<https://protege.stanford.edu/>>. Accessed on: 10 june 2022.

STEINMETZ, C. et al. Using Ontology and Standard Middleware for integrating IoT based in the Industry 4.0. **IFAC-PapersOnLine**, [S.l.], v.51, n.10, p.169–174, 2018.

STUDER, R.; BENJAMINS, V. R.; FENSEL, D. Knowledge engineering: principles and methods. **Data & knowledge engineering**, [S.l.], v.25, n.1-2, p.161–197, 1998.

TOLK, A. Composable mission spaces and M&S repositories–applicability of open standards. In: SPRING SIMULATION INTEROPERABILITY WORKSHOP, ARLINGTON (VA), 2004. **Anais...** [S.l.: s.n.], 2004.

TRAN, C.; MISRA, S. The technical foundations of IoT. **IEEE Wireless Communications**, [S.l.], v.26, n.3, p.8–8, 2019.

UNIVERSITY, S. Protege: a free, open source ontology editor and knowledge-base framework. **Stanford Center for Biomedical Informatics Research, Stanford, CA, USA**, [S.l.], 2001.

W3C. **w3-property**. Available at: <<https://www.w3.org/TR/owl-ref/Property>>. Accessed on: 10 june 2022.

WANG, J. et al. The evolution of the Internet of Things (IoT) over the past 20 years. **Computers & Industrial Engineering**, [S.l.], v.155, p.107174, 2021.

WANG, M.-M. et al. Middleware for wireless sensor networks: a survey. **Journal of computer science and technology**, [S.l.], v.23, n.3, p.305–326, 2008.

WANG, S. et al. Implementing smart factory of industrie 4.0: an outlook. **International journal of distributed sensor networks**, [S.l.], v.12, n.1, p.3159805, 2016.

WANG, Z. et al. Multi-protocol Integration and Intercommunication Technology Based on OPC UA and MQTT. In: JOURNAL OF PHYSICS: CONFERENCE SERIES, 2022. **Anais...** [S.l.: s.n.], 2022. v.2173, n.1, p.012070.

WÜBBEKE, J. et al. Made in China 2025. **Mercator Institute for China Studies. Papers on China**, [S.l.], v.2, p.74, 2016.

YE, X. et al. Toward the plug-and-produce capability for industry 4.0: an asset administration shell approach. **IEEE Industrial Electronics Magazine**, [S.l.], v.14, n.4, p.146–157, 2020.

YE, X. et al. Toward Data Interoperability of Enterprise and Control Applications via The Industry 4.0 Asset Administration Shell. **IEEE Access**, [S.l.], 2022.

YE, X.; HONG, S. H. Toward industry 4.0 components: insights into and implementation of asset administration shells. **IEEE Industrial Electronics Magazine**, [S.l.], v.13, n.1, p.13–25, 2019.

YONG, C.; LEE, W.; LAZARUS, C. Industry 4.0 Reference Architectural Models: critical review and opportunities. **International Journal of Advanced Research in Engineering Innovation**, [S.l.], v.3, n.4, p.40–49, 2021.

YOUNAN, M. et al. Challenges and recommended technologies for the industrial internet of things: a comprehensive review. **Measurement**, [S.l.], v.151, p.107198, 2020.

ZHONG, C.-L.; ZHU, Z.; HUANG, R.-G. Study on the IOT architecture and gateway technology. In: INTERNATIONAL SYMPOSIUM ON DISTRIBUTED COMPUTING AND APPLICATIONS FOR BUSINESS ENGINEERING AND SCIENCE (DCABES), 2015., 2015. **Anais...** [S.l.: s.n.], 2015. p.196–199.

ZHOU, H. **The internet of things in the cloud**. [S.l.]: CRC press Boca Raton, FL, 2012.

ZUEHLKE, D. SmartFactory—Towards a factory-of-things. **Annual reviews in control**, [S.l.], v.34, n.1, p.129–138, 2010.