Integrating Industrial Systems with Cloud-Based Platforms for Enhanced Connectivity and Efficiency

Master's degree in Electrical and Electronic Engineering

Judá Imbó

Leiria, Portugal, April 2026

Integrating Industrial Systems with Cloud-Based Platforms for Enhanced Connectivity and Efficiency

Master's degree in Electrical and Electronic Engineering

Judá Imbó

(2232626)

Dissertation Report under the supervision of Professor Paulo Coelho (paulo.coelho@ipleiria.pt) and Professor Eliseu Ribeiro (eliseu.ribeiro@ipleiria.pt).

Leiria, Portugal, April 2026

Originality and Copyright

This dissertation report is original, made only for this purpose, and all authors whose studies and publications were used to complete it are duly acknowledged.

Acknowledgments

To reach this point, I have not made this journey alone. Many people supported me in completing this milestone. Above all, I thank God for helping me to face and overcome the challenges of this chapter of my life, which is now coming to an end. I am grateful to my mother and siblings for always being there for me and encouraging me to get to this point. Without you, none of this would be possible.

I am also grateful for my supervisors for their guidance throughout the development of this dissertation. Especially, I am deeply thankful to Professor Paulo Coelho for his patience, availability, and continuous support.

Finally, I want to thank everyone who, in different ways, helped me make this achievement possible.

May God bless you all.

Abstract

This dissertation investigates the integration of industrial automation systems with cloud-based platforms by designing, implementing, and evaluating a practical Industrial Internet of Things (IIoT) architecture. The work addresses the limited availability of implementation-oriented comparative studies of PLC-Edge-Cloud solutions by combining a PRISMA-based systematic review with a practical case study using Siemens S7-1500 PLCs, a Siemens SIMATIC IOT2050 gateway, Node-RED, and three cloud platforms: Siemens Insights Hub, AWS IoT SiteWise, and Microsoft Azure Digital Twins.

The proposed architecture acquires industrial data via OPC UA, processes it at the edge, and transmits telemetry to cloud services via MQTT. The implementation demonstrates a reproducible gateway-centred approach for connecting industrial assets to cloud environments for monitoring, data modelling, and telemetry processing. The systematic review identifies the most common cloud platforms, protocols, architectures, benefits, and limitations reported in recent IIoT literature. The practical evaluation compares the selected platforms based on ease of use and implementation complexity, using observable criteria such as configuration steps, number of services, integration type, coding effort, dependencies, and conceptual difficulty. End-to-end latency was also analysed for the cloud-based MQTT communication paths implemented with AWS IoT Core and Azure IoT Hub.

The results show that Siemens Insights Hub offers the lowest implementation effort and the best ease of use, mainly due to its native integration mechanisms and reduced architectural complexity. AWS IoT SiteWise provides a balanced solution with moderate implementation effort. At the same time, Azure Digital Twins offers advanced modelling capabilities at the cost of higher configuration complexity, service dependency, and coding effort. In the latency comparison, AWS IoT Core achieved lower and more stable end-to-end latency than Azure IoT Hub under the tested conditions. Overall, this dissertation provides a replicable IIoT integration architecture and practical guidance for selecting cloud-based platforms in Industry 4.0 scenarios.

Keywords: IIoT, Cloud Computing, Industry 4.0, OPC UA, MQTT, Digital Twins

Resumo

Esta dissertação pretende explorar a integração de sistemas de automação industrial com plataformas baseadas na *cloud*, através da conceção, implementação e avaliação de uma arquitetura de *Industrial Internet of Things* (IIoT). O trabalho aborda a limitada disponibilidade de estudos comparativos orientados à implementação de soluções PLC-Edge-Cloud, ao combinar uma revisão sistemática baseada na metodologia PRISMA com um caso de estudo prático ao utilizar PLCs Siemens S7-1500, uma IIoT gateway Siemens SIMATIC IOT2050, Node-RED e três plataformas *cloud*: Siemens Insights Hub, AWS IoT SiteWise e Microsoft Azure Digital Twins.

A arquitetura proposta obtém dados industriais através de OPC UA, processa-os no dispositivo *edge* e envia para os serviços *cloud* por MQTT. A implementação demonstra uma abordagem reprodutível, centrada na *gateway*, para a conecxão de ativos industriais em ambientes *cloud* para monitorização, modelação de dados e processamento de telemetria. A revisão sistemática identifica as plataformas *cloud*, protocolos, arquiteturas, benefícios e limitações mais comuns reportados na literatura recente de IIoT. A avaliação prática compara as plataformas selecionadas com base na facilidade de utilização e na complexidade de implementação, recorrendo a critérios observáveis como o número de etapas de configuração, número de serviços, tipo de integração, esforço de programação, dependências e dificuldade conceptual. A latência ponta-a-ponta foi também analisada para os fluxos de comunicação MQTT na *cloud* implementados com AWS IoT Core e Azure IoT Hub.

Os resultados mostram que o Siemens Insights Hub apresenta o menor esforço de implementação e a melhor facilidade de utilização, principalmente devido aos seus mecanismos de integração nativos e à menor complexidade arquitetónica. O AWS IoT SiteWise constitui uma solução equilibrada, com um esforço de implementação moderado. Por sua vez, o Azure Digital Twins oferece capacidades avançadas de modelação, à custa de maior complexidade de configuração, dependência de serviços e esforço de programação. Na comparação de latência, o AWS IoT Core apresentou valores de latência ponta-a-ponta inferiores e mais estáveis do que o Azure IoT Hub nas condições testadas. De forma global, esta dissertação disponibiliza uma arquitetura de integração IIoT reprodutível e orientações práticas para a seleção de plataformas *cloud* em cenários de Indústria 4.0.

Palavras-Chave: IIoT, Computação em *cloud*, Indústria 4.0, OPC UA, MQTT, Gémeos digitais.

Contents

List of Figures

List of Tables

List of Abbreviations and Acronyms

ADT	Azure Digital Twins
AI	Artificial Intelligence
API	Application Programming Interface
AWS	Amazon Web Services
CLI	Command Line Interface
CPU	Central Processing Unit
DB	Data Blocks
DOI	Digital Object Identifier
DT	Digital Twin
DTDL	Digital Twin Definition Language
FB	Function Block
IAM	Identity and Access Management
IDE	Integrated Development Environment
IEEE	Institute of Electrical and Electronics Engineers
IIoT	Industrial Internet of Things
IoT	Internet of Things
IT	Information Technology
JSON	JavaScript Object Notation
ML	Machine Learning
MQTT	Message Queuing Telemetry Transport
OPC UA	Open Platform Communications Unified Architecture
PC	Personal Computer
PLC	Programmable Logic Controller
PN	PROFINET
PRISMA	Preferred Reporting Items for Systematic Reviews and Meta-Analyses
SAS	Shared Access Signature
SCADA	Supervisory Control and Data Acquisition
SCL	Structured Control Language
SDK	Software Development Kit

TCP	Transmission Control Protocol
TIA	Totally Integrated Automation

1. Introduction

Industry 4.0 relies on the vertical and horizontal integration of industrial assets, cyber-physical systems, and digital services to enable real-time monitoring, data-driven decision-making, and more flexible production systems [1], [2]. In this context, the Industrial Internet of Things (IIoT) provides the connectivity layer through which sensors, machines, controllers, and information systems exchange operational data across industrial environments [3]. However, the practical value of IIoT depends not only on the availability of cloud services, but also on the ability to connect shopfloor automation systems to cloud platforms in a secure, reproducible, and engineering-oriented manner.

Programmable Logic Controllers (PLCs) remain one of the main sources of operational data in industrial automation. At the same time, cloud platforms such as Siemens Insights Hub, Amazon Web Services (AWS), and Microsoft Azure provide services for device connectivity, data ingestion, storage, monitoring, analytics, and digital representation of industrial assets. Technologies such as Open Platform Communications Unified Architecture (OPC UA), Message Queuing Telemetry Transport (MQTT), edge gateways, and digital twins are commonly used to support this integration because they address interoperability, lightweight communication, edge processing, and asset modelling requirements [4], [5], [6], [7]. Nevertheless, the literature still provides limited implementation-oriented comparisons of complete PLC-Edge-Cloud architectures across multiple industrial cloud platforms. This makes it difficult for engineers, students, and researchers to evaluate the practical effort, configuration complexity, and usability differences between platforms in realistic industrial scenarios.

This dissertation addresses this gap by designing, implementing, and evaluating an IIoT architecture based on Siemens S7-1500 PLCs, a Siemens SIMATIC IOT2050 edge gateway, Node-RED, and three cloud platforms: Siemens Insights Hub, AWS IoT SiteWise, and Microsoft Azure Digital Twins. Industrial data are acquired from the PLC via OPC UA, processed at the edge, and transmitted to the cloud via MQTT and platform-specific ingestion mechanisms.

The work makes two main contributions. First, it provides a reproducible end-to-end PLC-Edge-Cloud integration architecture that can be replicated in academic and engineering contexts. Second, it presents a comparative evaluation of the selected cloud platforms using ease of use, implementation complexity, and end-to-end latency as evaluation criteria. Methodologically, the dissertation combines a systematic review with a practical case study. The systematic review identifies existing applications, architectures, protocols, and cloud services reported in recent IIoT literature, while the practical case study implements and compares the selected platforms through device onboarding, secure data transmission, telemetry ingestion, data modelling, and latency testing.

The manuscript is organised as follows: Section 2 gives background on Industry 4.0 concepts, IIoT fundamentals, and relevant cloud computing models. Section 3 presents the state-of-the-art and its findings. Section 4 describes the proposed architecture and the implementation steps for the communication between PLCs, the IIoT gateway, and the explored cloud platforms. Section 5 provides the experimental results and comparative analysis using the defined metrics. Section 6 concludes the dissertation, summarising the main contributions and addressing the objectives that were fulfilled.

2. Background

Discussing the integration of industrial systems with cloud-based platforms, which encompasses concepts such as Industry 4.0, cloud computing, and the Industrial Internet of Things (IIoT), can be quite challenging to understand. This section aims to assist readers with little or no prior knowledge, enabling them to understand the material presented in subsequent sections. Those who are already familiar with these topics may wish to proceed directly to the next section.

2.1. Industry 4.0

Humanity has experienced several moments in which technological advances have led to a change of paradigm in industrial production and organisation, which can be called the 'industrial revolution', as shown in Figure 2.1. In the first industrial revolution, with the scientific development realised during the 17th and early 18th centuries, the abundance of energy (coal) and iron ore, with the end of feudal organisation and the change in the land ownership regime, allowed the production of goods, especially in agriculture and textiles, to be mechanised from water or steam power, leading to a change in the way society was organised and the concept of the factory was developed [8].

From then on, other revolutions emerged. It moved from steam and water mechanisation to the internal combustion engine and the widespread use of electricity, which enabled mass production through the creation of the assembly line. Then, with the introduction of computers, electronics, and automation, the industry has achieved new levels of efficiency through electromechanical systems. All this has led to what we now call the fourth industrial revolution, or Industry 4.0, characterised by cyber-physical systems and digital interconnection, in which factories, machines and products communicate in real time, creating a new paradigm of intelligent production [8]. Despite being in this period, there are already conversations about Industry 5.0, which, according to Mulongo [9], aims to improve the consumer experience by integrating highly accurate machines with the creative capabilities of humans, enabling cost reduction, improved sustainability, easier customisation, and greater operational efficiency.

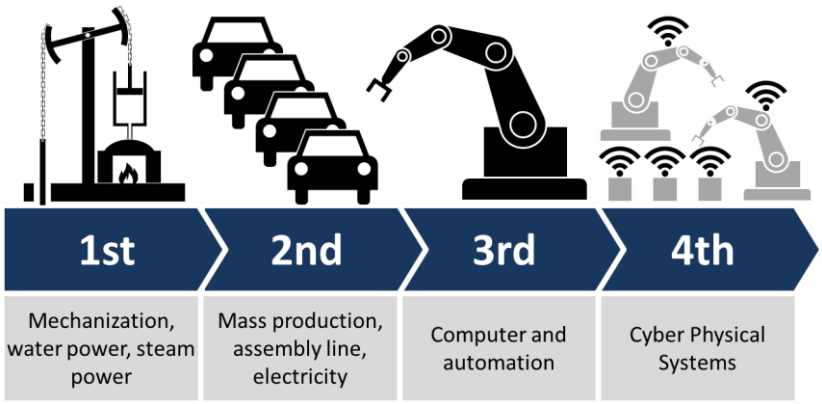

Figure 2.1 – Industrial revolutions

Industry 4.0 can be organised in four main technological areas according to [11], as illustrated in Figure 2.2:

- Connectivity & Communication

This area integrates IIoT and cloud computing, technologies that enable the creation of a smart and secure network with devices, machines, and management, processing, and storage of data services, which communicate with each other and share data in real time (explained in detail in Sections 2.2 and 2.3);

- Data, Intelligence, Analytics

The technologies involved in this area have an important role in the analysis and interpretation of large volumes of data coming from different parts of the shopfloor, helping not only to predict failures but also to optimise processes and facilitate decision-making;

- Advanced Production

This area includes technologies that use methods (e.g., 3D printing) capable of producing customised parts and prototypes only when needed. It also includes the use of collaborative robots capable of managing production phases autonomously, using information from sensors;

- Human-Machine Interaction

This area covers real-time simulation of products, processes or scenarios through the creation of virtual models and the use of augmented reality, which displays relevant information about industrial processes on top of the real environment using digital glasses or interactive tactile surfaces, for example.

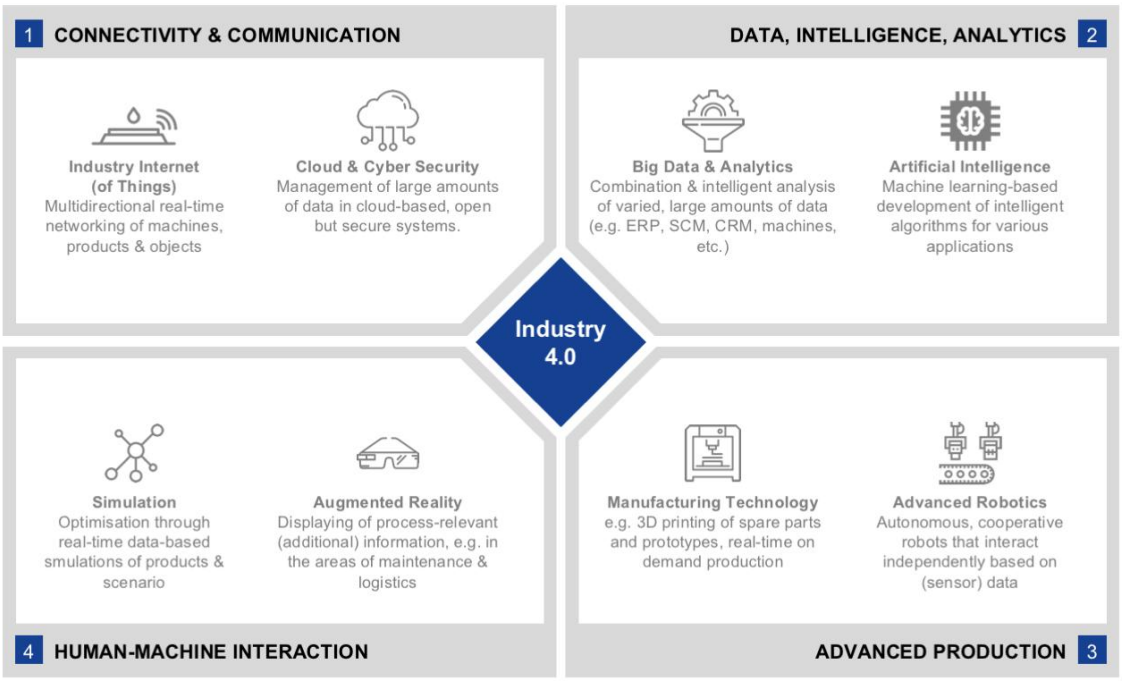

Figure 2.2 – Key areas of technology for Industry 4.0

This dissertation focuses mainly on Connectivity & Communication, although it also addresses Data, Intelligence, Analytics, and Human-Machine Interaction. With these basic notions of Industry 4.0, it is now possible to take a closer look at the concepts and definitions related to IIoT, cloud and metrics used to evaluate cloud architectures, which will be explored in the following subsections.

2.2. Industrial Internet of Things (IIoT) and its relationship with the Cloud

The concept of the Industrial Internet of Things involves integrating industrial components (sensors, machines, and control systems) that can communicate with each other to share and access data in real time, thereby establishing a smart network. This integration enables continuous monitoring of industrial systems, supports technicians and engineers in making quick decisions, and optimises resources to increase production efficiency at low operating costs [12].

According to Khan et al. [12], IIoT differs from the Internet of Things (IoT) in several respects, as summarised in Table 2.1. While IIoT is applied to industrial systems/environments (such as manufacturing, energy, and logistics), IoT focuses on more general applications for personal use (such as wearables and smart homes). In terms of security, the IoT has measures focused on utility and consumer protection. In contrast, the IIoT requires more advanced, robust mechanisms due to its integration into large-scale infrastructure and networks. When it comes to interoperability, IoT focuses on device autonomy, while IIoT emphasises architectures based on cyber-physical systems to enable compatibility across different industrial equipment. Concerning scalability, IIoT operates on large-scale industrial networks capable of supporting large amounts of data and equipment,

while IoT mainly operates on small networks. In terms of precision and accuracy, IoT is designed for critical real-time monitoring, but with a higher tolerance. In contrast, IIoT requires the opposite, needing rigorous synchronisation with millisecond-precision measurements to ensure reliability in industrial processes.

Table 2.1 – Comparison between IIoT and IoT

Concentration	IIoT	IoT
Application	Industrial systems and environments	General and personal-use applications
Security	Advanced and robust security mechanisms	Focused on usability and consumer protection
Interoperability	Focused on cyber-physical system–based architectures to ensure compatibility	Focused on device autonomy and basic interoperability between devices
Scalability	Operates in large-scale industrial networks	Operates in small-scale networks
Precision and Accuracy	Requires synchronization with high precision (in milliseconds)	Designed for real-time monitoring with higher tolerance

IIoT depends on a range of enabling technologies that can be integrated to improve its efficiency and functionality, depending on the adopted architecture. Among the most relevant technologies are Artificial Intelligence (AI), particularly Machine Learning (ML), cybersecurity, cloud computing, edge computing, and data analytics. AI and ML techniques can be employed to analyse the large volumes of data generated by industrial assets and shopfloor equipment, identify patterns and anomalies, and support intelligent decision-making and process automation. Cloud computing provides IT services for processing and storing large amounts of data, while edge computing performs these tasks closer to devices, enabling faster responses. Cybersecurity is the technology that protects devices and data against cyberattacks, enabling them to communicate securely. Data mining is used as a tool to extract useful information from device data to improve industrial processes [13]. In further relation to the technologies discussed, IIoT uses communication protocols that link industrial equipment to one another and to the cloud, as described later in Section 2.

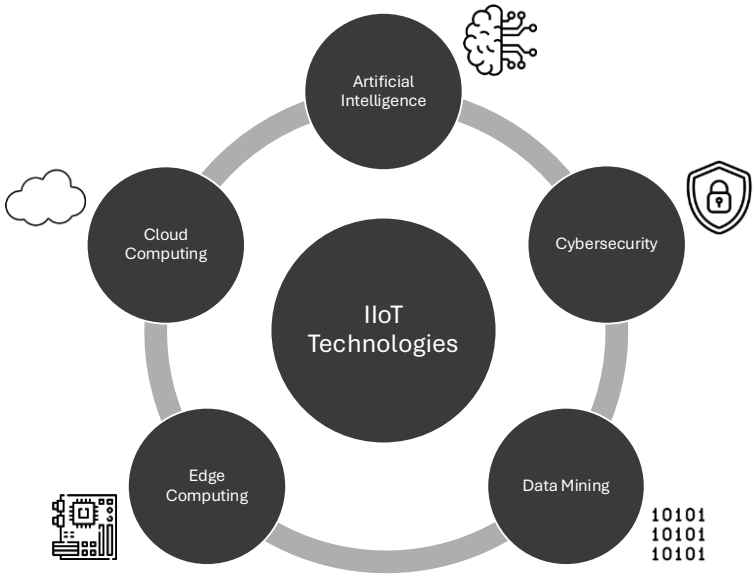

Figure 2.3 – Main IIoT technologies

2.3.Cloud definition, types of cloud and cloud services

To understand the arrival of the cloud, it is necessary to understand what existed before. In the past, each company and factory needed to have its own servers, networks, and databases to process and store information. To do so, they had to purchase, install, maintain and replace them, in addition to needing an IT team, which was extremely expensive. There were also cases where companies purchased more servers than they needed to handle peaks in demand, leaving them with underutilised resources and wasted money. With the arrival of the cloud, everything that companies were responsible for owning and operating is now managed by another entity, which may have several data centres full of servers, networking, storage, and other services, and is responsible for all the installation and maintenance required for these data centres to be operational. These entities are referred to as cloud providers [14].

Cloud providers usually offer many services related to computing (e.g. virtual machines), networking (virtual networks) and databases. But they also offer additional services, such as AI, IoT, and more. If a company needs a cloud service, it can create an account in minutes, use the service as it wishes, paying only for what it uses, and deactivate the service when it no longer needs it.

The definition of Cloud can be described through the five characteristics of cloud computing [14]:

- On-demand self service

Means that there is no need for human intervention to obtain computing, network or storage resources. To use a service, companies do not need to inform the cloud provider that they are interested in obtaining the service;

- Broad network access

This characteristic allows resources or services created by the company in the cloud to be accessed anywhere via the internet. In other words, it is possible to use/create, for example, a service in the United States and access it in Portugal, only needing a good network;

- Resource pooling

This means that cloud providers' physical resources are shared among companies. In other words, if company A creates a virtual machine on a server, there is definitely a possibility that it will share the same server with another company, B. However, it is not and never will be possible to know which company this is or to access its information. It is up to the cloud provider to decide which physical resource to allocate to the company's virtual services;

- Rapid elasticity

In other words, resources can be expanded or reduced automatically when necessary, avoiding the purchase of resources only for a peak demand scenario;

- Measured service

This means that payment is made only for the resources used, either for the time a server was used (per second/minute/hour) or for the amount of data stored in the database per day.

Combining all these characteristics, the cloud can be defined as a set of servers, networks, storage and interfaces that provide computing as a service [15], allowing it to be acquired or released quickly, accessed anywhere, configured when necessary, and only paying for what is used, in a secure way.

The diverse services offered by the cloud can also be divided into three main types, as shown in Figure 2.4: Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS). IaaS provides IT infrastructure such as computing, networking, and storage, leaving the customer or company to handle everything else, including software installation and virtual machine maintenance [16]. PaaS provides a platform for developing and running applications. In other words, the cloud provides computing, networking, storage, security, updates, maintenance, etc., leaving the customer only to develop the software and upload it to the cloud [17]. SaaS offers software that runs entirely in the cloud, with no need to install anything previously. The cloud provider is responsible for all software maintenance and updates [18]. It should be noted that the customer only uses the software and does not have access to the infrastructure that supports it.

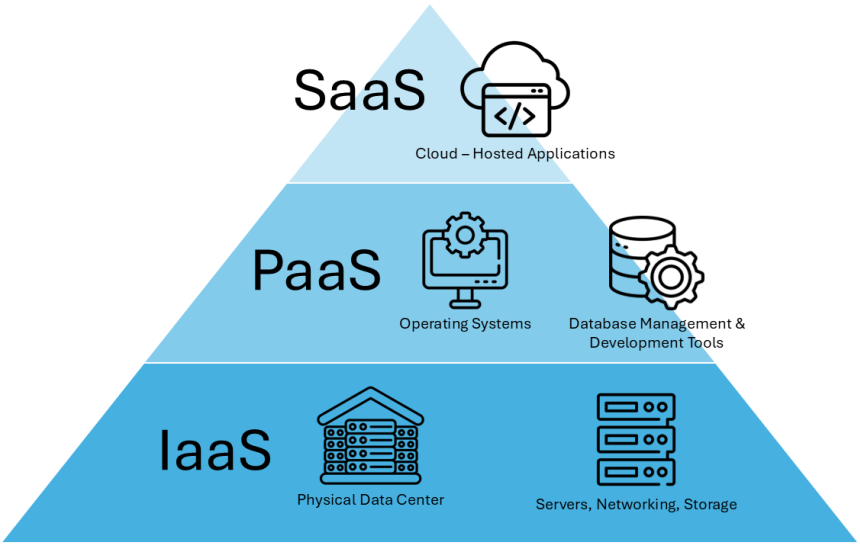

Figure 2.4 – Types of cloud services

There are also three types of cloud: private (which the servers, data centre or network are entirely dedicated to a single company); public (where the infrastructure is managed by a cloud provider with one or more data centres, with the infrastructure being shared between several companies) and hybrid (which is a mixture of private and public, where there is the possibility of having servers installed locally, with the company being able to use the private cloud for some services and the public cloud for others) [19].

With the evolution of the industry and the rising demand for production quality and on-time delivery, the shopfloor must operate at high performance and efficiency. With the use of IIoT (which enables real-time interconnection and communication between different types of machines, sensors and controllers to digitalise industrial processes) and the cloud (which offers a scalable infrastructure for data processing, storage and analysis with easy integration through management software), it will be possible to achieve enhanced connectivity and efficiency on the shopfloor through remote monitoring and intervention, predictive maintenance and optimisation of industrial processes. With clear, processed information, it will be possible to notify those who manage these industrial systems of current or future anomalies anywhere in the world, leading to better performance, productivity, and reduced waste.

2.4.Evaluation Metrics

To analyse the performance of IIoT-to-Cloud architectures and compare with other clouds, it is necessary to use metrics.

Aslanpour et al. [20] classify performance metrics according to the MAPE-K model (Monitoring, Analysis, Planning, Execution), with monitoring-related measures being the most common in practice. Resource utilisation is a metric that measures the consumption of CPU, memory, storage, and network resources. It is essential to evaluate task distribution efficiency and recognise underutilisation or overload situations. Resource load and

throughput are closely attached metrics that measure task processing intensity and the system's capacity to manage incoming workloads.

Latency-related metrics are highly important in IIoT applications, especially for time-sensitive industrial processes. Response time measures the time between request submission and task completion, while latency (or delay) measures the deviations from predicted completion times. In industrial control applications, where deterministic communication is generally required, these metrics are often linked to Service Level Agreement (SLA) violation rates, which indicate whether system performance meets predefined service requirements. Further, Aslanpour et al. [20] also note that relying just on average latency can be misleading, and that tail-latency measures should also be considered in critical applications.

Alongside performance and timing, energy consumption is also a key metric, particularly in edge and IIoT environments where devices may operate under power limitations. Similarly, cost is a key indicator in cloud-integrated systems, reflecting the financial burden associated with processing, storage, connectivity, or orchestration decisions. In the literature, resource utilisation, reaction time, energy consumption, and cost are among the most frequently examined metrics.

For industrial applications, reliability, availability, and fault tolerance are also important. These metrics examine the system's capacity to retain operation during failures, node instability, or communication disruptions, which are dominant issues in distributed IIoT implementations. In multi-layer IIoT-to-Cloud designs, these metrics become increasingly essential as system complexity increases.

Also, predictive and adaptive architectures frequently utilize analysis metrics such as mean square error, root mean square error, mean absolute error and mean absolute percentage error to evaluate forecasting models used in resource management or workload prediction. In dynamic systems that leverage orchestration methods, planning and execution metrics, such as adaptation time, supplied resources, and orchestration overhead, can be further analysed to evaluate the efficiency of scaling or offloading methods.

In general, the choice of metrics depends on the application's needs and the optimisation goals. But for IIoT-to-Cloud designs, a true analysis of system performance usually requires a detailed evaluation that considers latency, resource efficiency, energy, cost, and system reliability.

3. Related Work

The need for greater efficiency and in the manufacturing industry has been increasing, requiring that automated processes run at high performance and be regularly monitored to avoid future failures that might shut down production and, possibly, an entire shopfloor. In the energy sector, remote access to production data is crucial for rapid decision-making and optimisation. With the advancement of the Internet of Things (IoT) and cloud computing, the industry gained access to scalable solutions tailored to its needs and saw results immediately. The integration of cloud platforms with Industrial Internet of Things (IIoT) devices, with secure data transmission via authentication and encryption, has allowed remote control and predictive and preventive maintenance, resulting in reduced maintenance costs and increased production efficiency.

There are plenty of studies that explore applications across various industrial sectors, and other academic papers that describe in detail the most well-known cloud platforms. However, the lack of comprehensive, well-organised review articles that integrate case studies with more complete content might pose difficulties for a reader, researcher, or developer seeking to implement similar solutions in an industrial or scientific research environment. Therefore, this state-of-the-art presents a review developed using a systematic methodology (Preferred Items for Reporting Systematic Reviews and Meta-analyses - PRISMA [21]) so that the content is reliable and complete. This section aims to provide a structured, comprehensive overview of cloud manufacturing and IoT use in industry through the analysis of case studies.

3.1. Related literature reviews

Many academic publications provide overviews, surveys or reviews of the most well-known IoT/IIoT cloud platforms, including AWS, Microsoft Azure, Google Cloud, ThingSpeak, and IBM Cloud. Nagendran et al. [22] analyse the significant rise of IoT applications throughout industrial, commercial, research, and academic fields over the years. It emphasises the challenges of implementing IoT systems due to their distributed, interdisciplinary nature, leading to the development of a service sector focused on facilitating IoT implementation. Sikarwar et al. [23] studied cloud platforms designed for IoT applications. The paper analyses the integration of IoT devices with cloud computing services, showing the architectures, features, and services among different platforms. Barros et al. [24] provide a comprehensive analysis of these platforms to help developers decide appropriate solutions for their IoT deployments. Through a systematic investigation, the authors assess 50 academic articles and industry reports to identify the most important platforms previously described. These papers generally study and compare various platforms based on their architecture, key characteristics (such as security, interoperability, and scalability), challenges, and solutions to enable the successful integration and management of IoT devices and data. However, these studies do not include case studies demonstrating industrial applications.

Morelli and Ignacio [25] present a review of cloud computing's case studies within Industry 4.0, with a particular focus on cloud manufacturing. The authors, using bibliometric analysis tools such as VOSviewer, evaluated 1420 publications from the Scopus and Web of Science databases to map this domain's academic landscape. Still, the lack of features or benefits of implementing these technologies in each case study should be taken into consideration, even though their research highlights the optimisation of manufacturing processes, collaborative networks of manufacturing resources and services, the integration of Industry 4.0 with cloud computing systems, and data reliability and cybersecurity.

With these limitations, there is a need to understand, through a state-of-the-art approach, which case studies of cloud-based IIoT applications in different industries exist and what features, benefits, and limitations these case studies have. By addressing these aspects, this research can help understand how these cloud platforms can be implemented in an industrial environment and which tools are used to make this possible.

Table 3.1 presents a comparative analysis of the previously mentioned review articles categorising them according to various characteristics: if the manuscript mentions the cloud platform used/described in the paper (MCP); whether the manuscript explains the cloud platform architecture applied (ECA); whether the manuscript presents case studies with industrial applications; and whether if the manuscript describes features, benefits and limitations of each case study.

Table 3.1 – Existing review articles on cloud services

Study	Description	MCP	ECA	CSIA	FBL
[22]	Survey on Cloud management services	✓	✓		
[23]	Literature survey on various cloud services used in integration with IOT	✓	✓		
[24]	Systematic Multivocal Mapping Study on IoT Platforms	✓	✓		
[25]	Survey on the current state of academic research related to the cloud manufacturing field	✓	✓	✓	
This Survey		✓		✓	✓

The overview of Table 3.1 reveals that all previous review articles regularly mention the cloud platforms utilized (MCP) and provide explanations of the corresponding cloud architectures (ECA), highlighting the relevance of these aspects in cloud service reviews. Although case studies with industrial applications (CSIA) are limited, appearing only in the study [25], the practical validation of theoretical findings has been underexplored. Furthermore, none of the previous papers systematically analyse the features, benefits, and limitations (FBL) of the cloud services studied, indicating a notable lack of critical evaluation and comprehensive comparative analysis in the existing literature.

In contrast, the present survey distinguishes itself by incorporating all four evaluative aspects: MCP, ECA, CSIA, and FBL. This study provides a more comprehensive and rigorous analysis by integrating technical and practical approaches and by critically evaluating the cloud services under study. Consequently, it addresses important weaknesses identified in past papers and contributes to a more complete, application-oriented understanding of cloud management and services.

3.2.Methodology

The research used a systematic approach to review cloud-based IIoT solutions. The PRISMA approach [21] ensures transparency, reproducibility, and rigorous data collection and rigorous analysis. This methodology was organised around distinct research questions meant to identify the key technologies and practices involved in the integration of cloud platforms with IIoT systems, particularly in industrial environments:

- RQ1: Which cloud-based platforms are most commonly used in Industrial IoT applications?
- RQ2: What communication protocols are employed to connect industrial systems with these cloud platforms?
- RQ3: What benefits and limitations are observed in integrating IIoT systems with cloud platforms in real-world industrial case studies?

To ensure the relevance and quality of the chosen papers, a set of keywords was carefully selected to conduct the systematic search and to be applied across the academic databases. The keywords used were: “Industrial” AND (“IIoT” OR “Cloud”) AND “Platform” AND (“IoT” OR “IIoT”). IEEE Xplore, Scopus, and SpringerLink were the academic databases used for this research due to their relevance and coverage of high-quality publications in the field.

The scope was limited to studies published in the last five years to ensure relevancy and current applicability. Additional filters were applied in the following databases to reduce the results by thematic focus:

- Scopus: Limited to IoT, IIoT, Cloud Platforms, and Engineering;
- Springer Link: Restricted to engineering disciplines, with preview-only content excluded from consideration.

Therefore, the search was conducted independently using these phrases across all digital libraries, and the duplicates were then eliminated based on the digital identifiers. Additionally, each potentially relevant paper was evaluated individually to determine whether it met the following eligibility criteria:

1. Studies with a focus on real industrial applications of cloud-based IIoT systems;
2. Studies that analyse cloud platform architectures and communication protocols;
3. Studies that include case studies with practical implementations;
4. Studies that discuss the features, benefits, and limitations of the applied technologies;

5. Original research articles published between 2019 and 2024;
6. Studies written in English and available in full text.

Also, the exclusion criteria were applied during the selection of studies for this systematic review:

1. Duplicate publications across databases;
2. Studies without full text availability;
3. Studies whose abstract or full text did not align with the research theme;
4. Review or survey papers.

These procedures aim to preserve the transparency, consistency, and reproducibility of this systematic review. Following the systematic selection and filtering procedure described, to ease the analysis and extraction of key characteristics from the chosen studies, a classification methodology was established. In this case, cloud services were classified according to the standard cloud service types (IaaS, PaaS, and SaaS), which reflect levels of user control and provider responsibility. Also, two additional categories were established: “Combination” for services that use more than one cloud service type, and “Unknown” for case studies where the current information is lacking. This classification explains the role and structure of cloud platforms used in IIoT scenarios.

Their main functions and application areas have also been classified, enabling communication protocols to be understood for the connectivity between industrial systems and cloud platforms. IoT protocols are generally used in low-power, low-bandwidth environments, while industrial protocols are designed for reliable, real-time communication in automation systems. Web protocols facilitate client-server interactions and API connectivity, while security protocols ensure data protection through confidentiality and integrity. Similarly, to the previous classification, an "Unknown" category was added for proprietary or unspecified solutions. This methodology facilitates a clear, consistent study of communication technologies across multiple scenarios.

3.3.Results

A total of 4200 research studies were collected from various databases, as shown in Figure 3.1, comprising 992 from IEEE Xplore, 2208 from Scopus, and 1,000 from Springer Link. From 4,200 research papers, 4,043 papers 8 were preserved after deleting 157 duplicates found by the Digital Object Identifier (DOI). The analysis of titles led to the removal of 339 publications. Subsequently, the remaining 3704 publications underwent an abstract and full-text evaluation, resulting in the removal of 3620 articles. Of these, 33 lacked their complete texts, 3554 did not match the search theme, 31 were classified as state-of-the-art or review papers, and 2 had not been published in English. Finally, the quantitative synthesis and meta-analysis incorporated the remaining 84 studies.

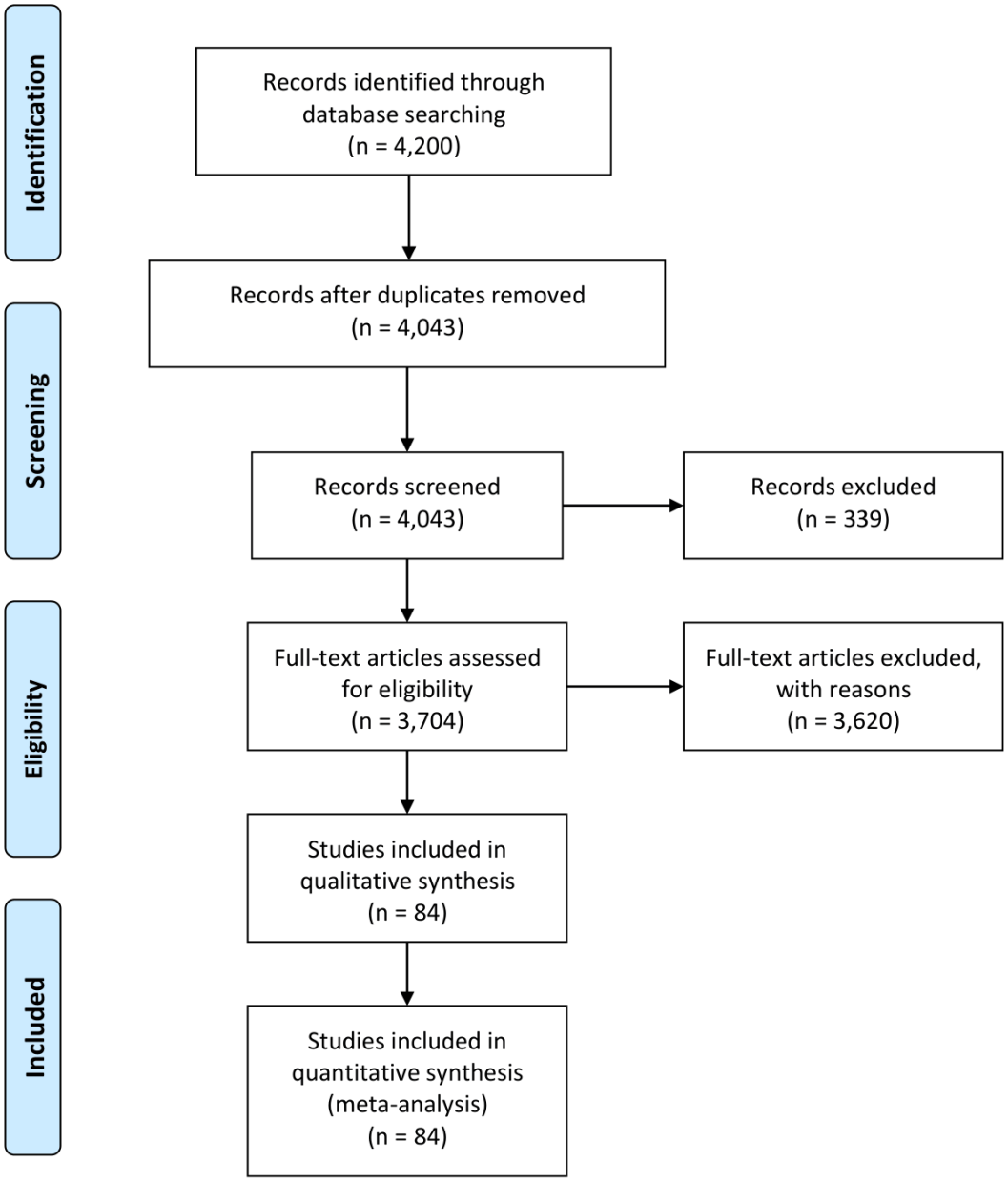

Figure 3.1 – PRISMA flow diagram

3.3.1. Energy

Academic papers in the energy sector primarily explore IoT/IIoT monitoring and control for energy assets and their infrastructure. These studies predominantly focus on real-time monitoring and fault detection in renewable energy production (particularly solar and small hydropower systems), smart-grid and microgrid energy management, and industrial energy consumption monitoring. A smaller number of these studies focus on energy-related critical infrastructure (e.g., pipeline systems) through measurement and continuous telemetry. More recent research further highlights the combination of IoT-based smart metering coupled with cloud analytics and predictive algorithms to support anomaly detection, demand estimation, and automatic control adjustments in industrial energy management [26]. Recent contributions further highlight the relevance of gateway-based IIoT deployments (e.g.,

industrial gateways integrated with data visualisation platforms) for monitoring and managing grid-tied solar integrations in high-demand facilities [27].

Across the analysed studies, sensors and telemetry pipelines collect electrical and environmental data, which are transmitted to cloud services for visualisation, analysis, and reporting. Edge and fog components are regularly integrated, with system functions distributed across end, edge, and cloud levels, delineating local acquisition and actuation from centralised storage, dashboards, and more computationally intensive analytics. Communication and integration depend on lightweight IoT messaging and industrial protocols (e.g., MQTT), with security measures described in terms of encrypted communication channels. Data management is typically implemented using scalable cloud storage and time-series backends, complemented by web- or mobile-based HMIs.

Many studies demonstrate low-cost, scalable monitoring deployments, and the use of analytics for predictive purposes is described, including machine-learning-based systems. Often, highlighted limitations include reliance on stable communication, especially in remote environments. Testing is often conducted through simulation or small-scale prototypes, with few studies describing large-scale deployments. The analysis on robustness, interoperability, and lifecycle maintenance is lacking, and variations in protocol or platform choices can be observed between implementations.

3.3.2. Manufacturing and Production

In manufacturing and production, studies frequently emphasise remote monitoring and supervision of processes and assets when implementing IIoT. The objectives in those studies include operational efficiency, product quality improvement, and safety enhancement. Across manufacturing and process industries, the most common application areas are real-time condition monitoring, supervisory control, and alarm-based interventions, with predictive maintenance services developed to reduce operational interruptions and optimise asset lifecycle management. Several studies also describe Industry 4.0 implementation through retrofitting and advanced automation methodologies. Recent findings also suggest continued integration of SCADA systems and cloud services, with security controls (e.g., VPN-based connectivity) positioned as enabling conditions for realistic IIoT deployment [28]. Also, highlighted interoperability-driven acquisition architectures that centralize variables from different shopfloor systems (e.g., PLCs and industrial robotics) using gateway-based integration with a combination of industrial and IoT communication protocols, reinforcing the emphasis on cross-vendor data collection in Industry 4.0 contexts [29].

Most architectures in the papers include sensors and instrumentation connected to PLCs and embedded gateways, with cloud-based dashboards that enable similar SCADA monitoring and control functionalities. Data transmission is commonly described using industrial and IoT protocols, including MQTT, Modbus TCP/IP, and OPC UA, with connectivity options including Zigbee, LoRaWAN, and NB-IoT, or cellular networks. Many studies focus on edge-computing components in the architecture, as well as on time-series data storage and

workflow/orchestration components to support system integration and visualisation. Advanced analytics is often described in research, which includes KPI-based monitoring to machine learning and deep learning methods for anomaly detection, prediction, and failure classification. Some studies also mention digital-twin-related monitoring applications.

Across the literature, documented trends include increased use of integrated edge-cloud architectures and the use of analytics in monitoring and maintenance processes. Most observed limitations include reliance on stable communication, interoperability challenges with heterogeneous interfaces, and insufficient documentation of large-scale deployments or long-term robustness. Other common limitations include security measures, sensor reliability and calibration, and cost or complexity issues in environments.

3.3.3. Water Management

Most studies in the water management field focus on IoT-enabled monitoring and control for wastewater treatment, sewage plant automation, flood monitoring, and water distribution systems. Commonly mentioned objectives include improving efficiency, supporting regulatory compliance, and providing immediate warning functions. The most common application focus is continuous monitoring of water quality and hydraulic parameters, with systems often providing remote monitoring, alert notifications for anomalies, and, in some cases, remote intervention in treatment operations. Recent research further highlights the importance of low-cost, IoT-based tank monitoring that integrates level sensors with basic water-quality indicators and cloud dashboards for remote monitoring [30]. Recent contributions also propose solutions that integrate long-range telemetry with IIoT gateways and PLC/SCADA supervision to enable accurate level monitoring, alarm management, and automated pump control in remote environments [31]. Additionally, recent work describes IoT-based condition monitoring in wastewater treatment facilities that combines process measurements with equipment condition monitoring for early fault detection and maintenance planning, reinforcing the crossover between monitoring and predictive maintenance in industrial contexts [32].

Across research, common implementations follow an end-to-end methodology in which different sensors are connected with PLCs and embedded gateways. Data transmission is often implemented using industrial and IoT protocols, including MQTT, Modbus, OPC UA, and TCP/IP, with wireless connectivity options such as Zigbee, LoRaWAN, NB-IoT, and cellular networks. Cloud platforms are frequently used for data storage and dashboard interfaces to facilitate remote monitoring. Edge computing components are also described in many architectures, including time-series storage and middleware elements to enable integration among system components. Advanced analytics is described in part in academic papers, including KPI-based monitoring and anomaly detection, while other studies describe predictive maintenance or process-optimisation-based applications.

The trends in this area include most integrated cloud-edge architectures and analytically supported monitoring systems. However, the dependency on internet connectivity and cloud

availability, communication reliability issues, and limited testing at scale or across different scenarios are frequently cited as limitations.

3.3.4. Prototypes, architectures and platforms development

These studies present frameworks, gateways, middleware, and reference designs intended to integrate industrial assets with cloud services. These contributions generally emphasise reusable architectural components for integrating heterogeneous industrial equipment with cloud-based monitoring, analytics, and control, with a focus on interoperability, low-latency communication, and scalable deployment across diverse industrial environments.

The deployment of industrial gateways and edge platforms that communicate with legacy industrial protocols and IoT architectures is a common technical approach between the studies. Implementations generally combine PLC-based integration with protocol translation and management components. The mentioned protocols usually include MQTT, Modbus, and OPC UA, and some research also documents vendor-specific industrial interfaces. Also, REST-based infrastructure is mentioned in some of the studies.

Edge-cloud architectures are discussed in these studies, generally supported by containerisation and modular edge frameworks. Also, many of them include time-series data storage, dashboard or visualisation components, and integration infrastructure to facilitate data ingestion and operational monitoring. Security and reliability-related features are often mentioned, including secure data transmission and procedures to support continuity and recovery during failures. Reported trends include greater use of hybrid edge-cloud deployments and scalable platform architectures. However, network stability, implementation cost, complexity, and scalability testing commonly become more challenging as the number of devices and data volumes increase.

3.4.Discussion

By analysing 84 studies, we evaluated the classification of academic papers across various industrial areas, highlighting an imbalance in research attention, as shown in Figure 3.2. Most academic papers (54%) are in the Manufacturing and Production area, indicating significant interest in this area. The Energy area represents 27% of the papers, while Prototypes, architectures and platforms development accounts for 11%. The Water Management area is the smallest, accounting for about 8% of the total. This distribution indicates a predominant interest in manufacturing-related research, in contrast to other industries, which are comparatively less significant.

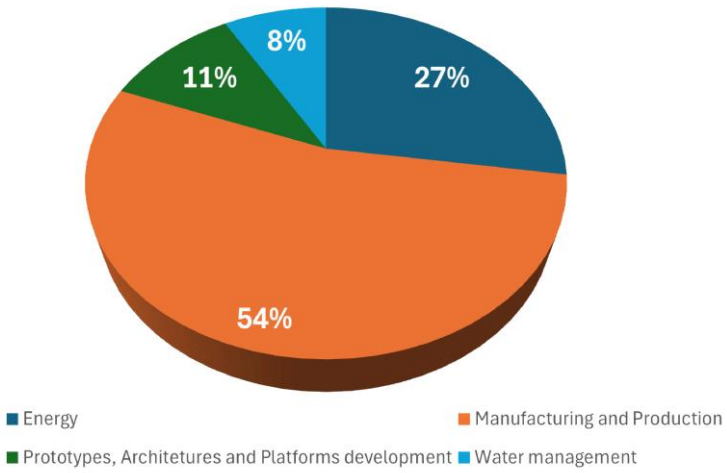

Figure 3.2 – Distribution of the papers per industrial area of application

Also, through the analysis of the papers, it was possible to see the distribution of studies by cloud service type, providing insights into the dominance of various deployment approaches within industrial IoT solutions, as shown in Figure 3.3. The findings indicate that PaaS is the most used service type, featured in 35 academic papers, due to its balance between customisation and ease of integration. IaaS is next, with 20 studies demonstrating its popularity for applications that require enhanced control over computational resources. 18 academic papers do not specify the cloud service type, indicating a common gap in documentation and reporting. The limited adoption of SaaS, reflected in only three research papers, suggests its limited application in industrial environments that regularly require better system integration and flexibility. The existence of eight papers that use a combination of service types demonstrates the need for mixed methodologies to address diverse industry requirements.

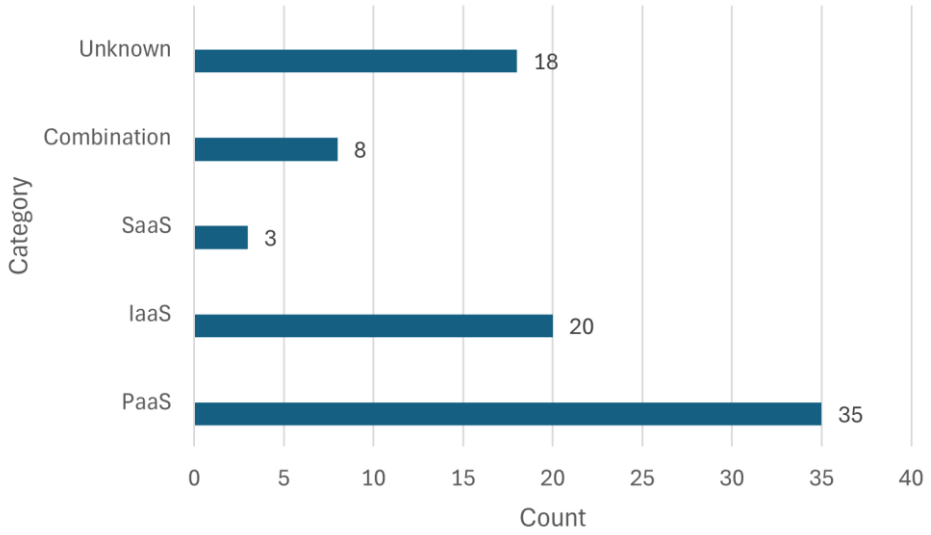

Figure 3.3 – Relationship between the types of cloud services and the number of papers

Figure 3.4 shows the distribution of communication protocols used across the analysed papers. The IoT protocols dominate the landscape, appearing in 58 research papers, reflecting their suitability for low-power, limited-bandwidth contexts typical of industrial environments. Industrial protocols and unknown categories each contribute to 10 studies. Web and security protocols are less common, appearing in 4 and 2 studies, suggesting limited use or a lack of reporting in these areas. The results indicate the predominance of IoT protocols in IIoT implementations and underscore the need for broader adoption of secure, interoperable communication standards.

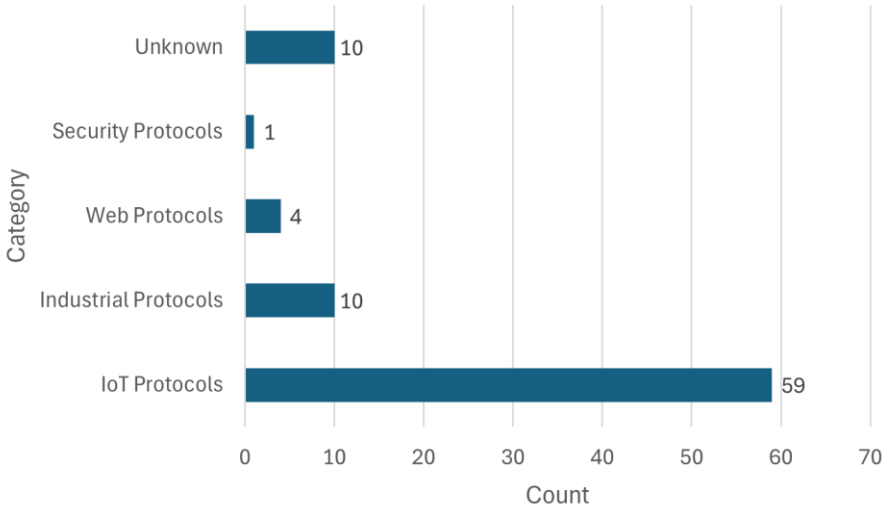

Figure 3.4 – Relationship between communication protocol categories by area of application and the number of papers

Across the analysed papers, the relationship between cloud service integration and communication protocol selection offers insights into both implementation challenges and dominant technical choices in industrial contexts. IoT protocols, especially MQTT and, to a

lesser degree, Wi-Fi, are used most frequently, indicating that they dominate communication in IIoT and cloud services, and that they are effective for real-time monitoring and control across the mentioned industrial areas. At the same time, communication to the cloud and between IIoT components via industrial protocols (mainly OPC UA) is common, with a moderate predominance, and remains crucial in environments that demand stronger security standards and higher interoperability.

Cloud-based IoT systems offer significant benefits in scalability, cost efficiency, and real-time monitoring, which are crucial for industries seeking to boost operational efficiency and reduce downtime. The integration of these platforms with remote data access enables predictive maintenance and fault detection, improving decision-making and avoiding unexpected disruptions. Moreover, their versatility enables both small and medium enterprises (SMEs) and large-scale industrial operations to use these solutions without incurring high upfront infrastructure costs. However, despite these advantages, challenges persist, most notably in areas such as stable connectivity, scalability in large-scale environments, and cybersecurity. Reliance on a stable internet connection can limit functionality in areas with poor network coverage. At the same time, scalability challenges exist as larger industrial environments inevitably push the limitations of current architectures. Also, with the increasing adoption of cloud platforms, the necessity for enhanced security mechanisms to protect sensitive industrial data becomes even more important. These limitations emphasise the need for further research and development to optimise these systems for larger industrial applications, ensuring robustness and reliability across different environments.

This section provides an overview of the results from the analysis of existing cloud-based IIoT platforms, with a focus on their integration with industrial systems, communication protocols, and case studies. Based on a systematic review of studies in this field, several observations have been made about cloud platforms, communication protocols, and the benefits and challenges faced by industry sectors that use these technologies. Regarding RQ1: “Which cloud-based platforms are most commonly used in industrial IoT applications?”, it was observed that platforms such as AWS, Microsoft Azure, and Google Cloud are the most widely used in IIoT applications. These platforms dominate the landscape due to their scalability, powerful infrastructure, and excellent security features. However, studies also noted the use of more specialised platforms such as ThingSpeak, Thingsboard, and the UoM IoT Platform, which serve specific industrial needs, especially for small and medium companies.

Considering RQ2: “What communication protocols are mostly employed to connect industrial systems with these cloud platforms?”, the review revealed that IoT protocols are the most widely used for connecting industrial systems to cloud platforms, due to their efficiency in low-bandwidth, low-power environments. Other industrial protocols, including as OPC UA and Modbus, are also applied in sectors that require higher levels of security and real-time data transmission.

Finally, regarding RQ3: “What benefits and limitations are observed in integrating IIoT systems with cloud platforms in real-world industrial case studies?”, it is concluded that cloud-based IIoT systems offer benefits, including scalability, cost efficiency, and real-time monitoring and predictive maintenance. These benefits help industries to enhance operational efficiency and reduce downtime. However, some limitations persist, such as reliance on stable internet connectivity, which might limit the use of systems in remote or poorly connected areas. Also, scalability and cybersecurity concerns remain obstacles that need to be addressed to ensure the continuous success and durability of these systems across different industrial environments. Based on these theoretical insights and the recognised technology landscape, the next section discusses the practical development and implementation of an IIoT architecture intended to fulfil these requirements.

4. Development

This section describes the development of an IIoT architecture designed to extract, process and transmit data to cloud platforms, based on a simulation of an automated machining and parts distribution system originally developed as part of an undergraduate course module. The model of this system was created using Real Games' Factory I/O software [33], which enables the performance of industrial processes to be analysed in a simulated environment. The control of this system was implemented using Siemens Programmable Logic Controllers (PLCs), configured and programmed in the TIA Portal software using Ladder Logic (commonly known simply as Ladder) and Structured Control Language (SCL). The simulated model includes various operating modes, including the production of individual or stacked parts, with the selection of machining centres and distribution stations. Parts are transported through conveyors, pop-up wheel sorters, and pneumatic actuators, whose performance depends on sensors, speed controllers, and timers integrated into the programme. This system also includes PID control of the machining centre tanks and safety mechanisms (stop and emergency buttons), as shown in Figure 4.1.

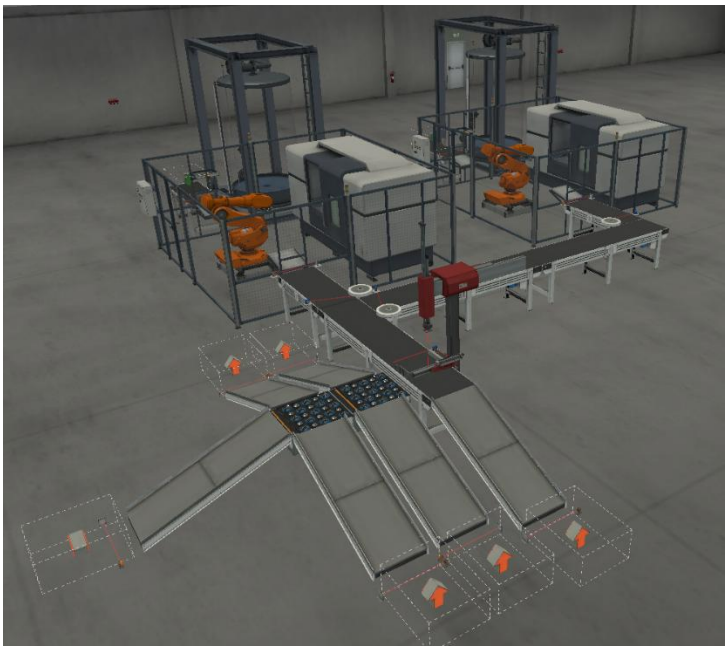

Figure 4.1 – Factory I/O simulation of the automated machining and distribution system

To make this possible, it was necessary to use an integrated set of hardware, software, and communication protocols suitable for the context of industrial automation and IIoT to support industrial communication and integration with cloud platforms, as shown in Figure 4.2. At the hardware level, two Siemens S7-1500 PLCs were used: the 1516-3 PN/DP (later simulated in the project) and the 1512C-1 PN CPU, for system simulation and communication with the IIoT gateway. This gateway, represented by the Siemens SIMATIC IOT2050 device (hereafter referred to as IOT2050), is responsible for data acquisition and processing, as well as for connecting to cloud platforms, and integrates software such as

Node-RED. The use of these devices was supplemented by a switch, a USB Wi-Fi adapter to provide the IOT2050 with internet access, and a laptop for programming, configuration and running the simulation. The simulated model includes various operating modes, including the production of individual or stacked parts, with the selection of machining centres and distribution stations. Parts are transported through conveyors, pop-up wheel sorters, and pneumatic actuators, whose performance depends on sensors, speed controllers, and timers integrated into the programme. This system also includes PID control of the machining centre tanks and safety mechanisms (stop and emergency buttons).

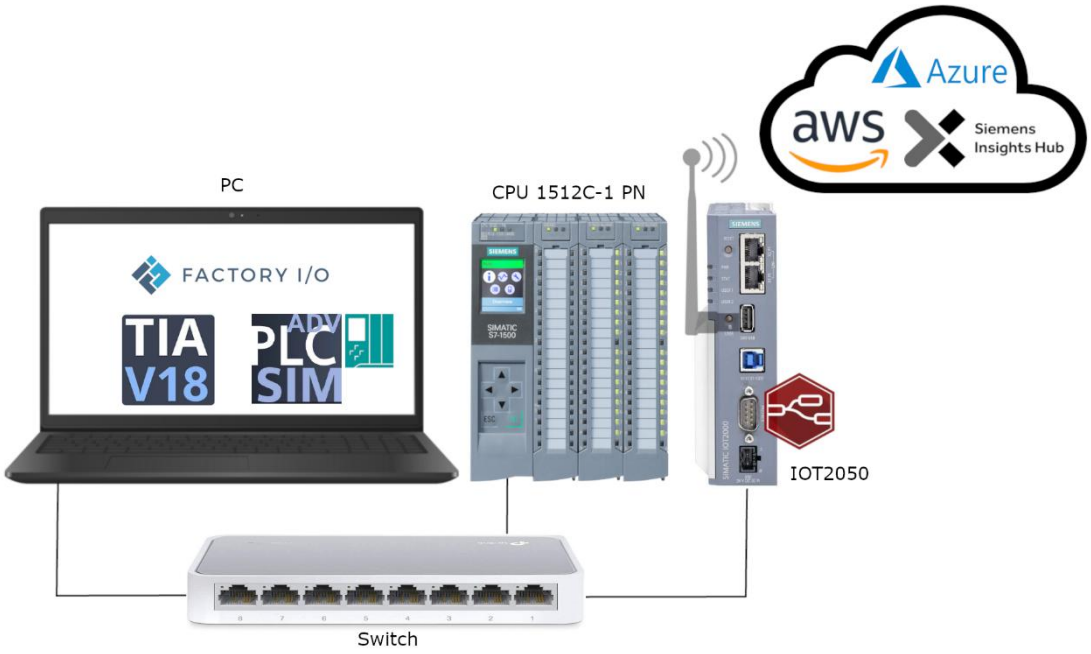

Figure 4.2 – IIoT setup used to send data to cloud-based platforms

In terms of software, Siemens STEP 7 TIA Portal V18 is used for PLC programming, Siemens S7-PLCSIM Advanced V5.0 for PLC simulation, and Factory I/O for the simulated model. Interconnection and data transmission to the cloud are established via Node-RED, while UA Expert enables the visualisation of properties/variables via OPC UA. Other software was also used, such as PuTTY for communication and configuration of the IOT2050 via SSH, and Visual Studio Code in combination with Microsoft .NET 8 SDK, Azure CLI, Azure IoT Explorer and the Azure IoT Hub Extension, which enabled the development of internal cloud telemetry functions. Communication between industrial devices (except PLC-to-PLC) was carried out via OPC UA, known for its security and interoperability, and with the cloud via the MQTT protocol. The cloud platforms used in this thesis were Siemens Insights Hub, Microsoft Azure and AWS (Amazon Web Services), utilising free or partially free services appropriate to the academic scope of the work.

During the development stage of this dissertation, it was necessary to attend online courses such as “Azure IoT - The Complete Guide”, by Memi Lavi [34] and “Learn MQTT With Node-RED“, by Rajvir Singh [35], watch technical videos and consult manuals, to understand how the platforms used work, as well as their configuration and integration methodologies. This necessity is justified by the complexity involved, particularly for those

with only basic knowledge in the field. It should be noted that several solutions, with different approaches, can yield results equivalent to those obtained in this work.

4.1. Communication between Programmable Logic Controllers (PLC) and IIoT Gateway

The simulation with Factory IO and data transmission via the IoT2050 was initially run on a physical Siemens S7-1500 PLC with a 1516-3 PN/DP CPU, a configuration commonly used in IPLeiria practical classes. This PLC supports OPC UA communication, allowing you to create an OPC UA server or client in the CPU properties and define the accessible properties. However, during the dissertation, access to this PLC was no longer available, and alternative solutions had to be found. The initial solution used the S7-PLCSIM Advanced V5.0 software on the PC to simulate the PLC in combination with the IIoT gateway. However, communication between the two is not possible as the IOT2050 cannot connect directly to a simulated PLC. It is important to note that using only the PLC simulation with Node-RED installed on the PC could have been a solution, but it does not fulfil the purpose of this work. Given this limitation, the physical Siemens SIMATIC S7-1500 CPU 1512C-1 PN PLC was used. Because it has less capacity than the original, it was not possible to transfer the developed programme directly, which would have required rewriting the entire programme. To avoid this, a solution was adopted that simulates the PLC already configured in the programme and sends the necessary properties to the physical PLC via PROFINET, a communication protocol commonly used between Siemens PLCs.

To make this possible, the Function Blocks (FB) TSEND_C (data sending) and TRCV_C (data receiving) were used in both the simulated PLC and the real PLC, as demonstrated in Figure 4.3. To facilitate the transmission of variables through these FB, Data Blocks (DBs) were also created, each containing the properties to be transmitted. In the simulated PLC, the values of the programme's properties are copied to these DBs (and vice versa) via functions written in SCL. In a real PLC, there is no such need, since the OPC UA Server configured on this device accesses the data in the DBs directly.

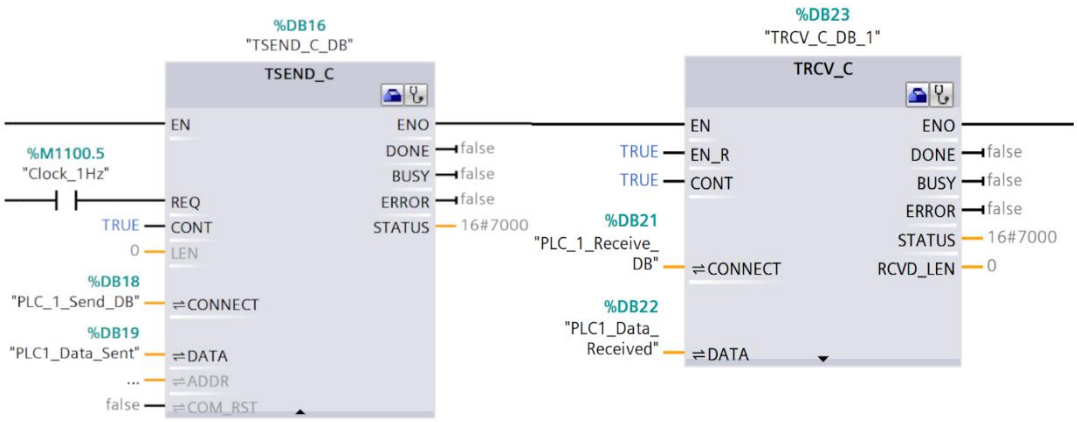

Figure 4.3 – TIA Portal's TSEND_C and TRCV_C function blocks

Siemens provides an official image for the IOT2050 to enable its operation and configuration, requiring only that it be installed from an SD Card [36]. However, this image has significant limitations: the included software is outdated, and installing or upgrading additional Node-RED palettes is necessary, meaning this device cannot reach its full potential. To solve this issue, since the IOT2050's base image is built on Debian Linux, a Docker container was created on top of the base image with the latest versions of Node-RED, Mosquito MQTT, and Node.js. With this solution, you can control, update, install, and uninstall every part of the software running in the container, including installing new palettes. To run the container, PuTTY is required to access the IOT2050's Debian Linux operating system.

The implementation in Node-RED is organised into three main flows, as shown in Figure 4.4: The first flow, "Communication PLC <-> Node-RED", ensures the collection and writing of properties between the PLC and the IOT2050, enabling bidirectional data exchange. The second flow corresponds to the dashboard, designed for operational control and real-time visualisation of system properties, performing functions characteristic of a SCADA interface. The last one, called "Communication Node-RED <-> Clouds", is responsible for integrating with cloud platforms and preprocessing data. The interconnection between flows is made through the *link in* and *link out* nodes, which are also used internally

(especially in the "Communication Node-RED <-> Clouds" flow) to divide the code into smaller modules due to its larger size and complexity.

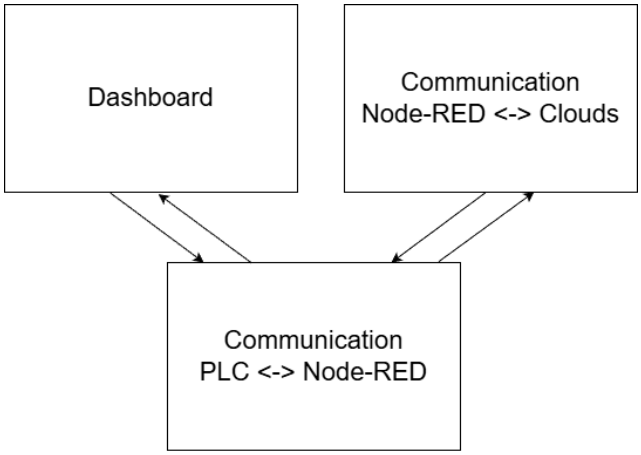

Figure 4.4 – How the Node-RED flows interact with each other

In the "Communication PLC <-> Node-RED" flow, nodes from the *@node-red-contrib-opcua* palette were used, specifically the *OpcUa-Client* node, which establishes a session with the OPC UA server and performs data read and write operations [37]. The properties to be accessed were identified using the *OpcUa-Item* node, which declares the Node IDs. These IDs correspond to the unique identifiers for each property. To read these operations automatically on a cyclical basis, the OPC UA client was configured with the function to read, an endpoint was also inserted (endpoint *Inject* nodes were used) configured to inject messages at 0.5-second intervals, enabling the periodic transmission of read requests to the server, as can be seen in Figure 4.5. Once read, the payload containing the information of each property is redirected to the dashboard and to the flow responsible for transmitting data to the cloud providers.

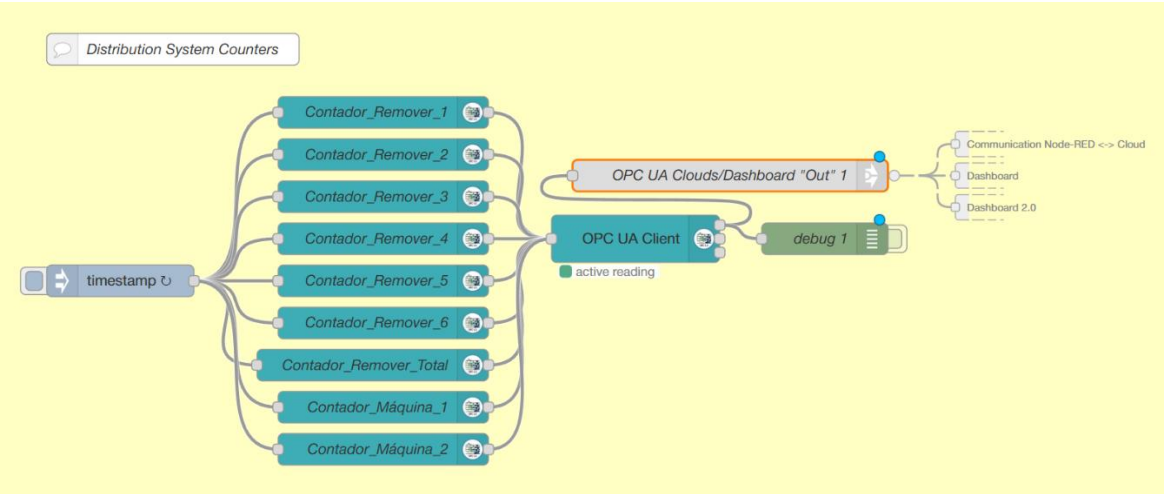

Figure 4.5 – Data reading from the *OpcUa-Client* node

In the "Dashboard" flow, which is dedicated to operational control and the visualisation of system properties, the *@flowfuse/node-red-dashboard* palette was used to build the monitoring interface [38]. It is divided into two components: Monitoring and Control. The

supervision component (depicted in Figure 4.6) includes several visualisation elements (*ui-text* node) for displaying the production counters associated with the system's six distribution points, a total counter and the two counters for the machining centres. Also, it has graphical charts to watch the tank levels and their valves. To reach these nodes, the data passes through *switch* nodes that filter the properties based on their respective Node IDs.

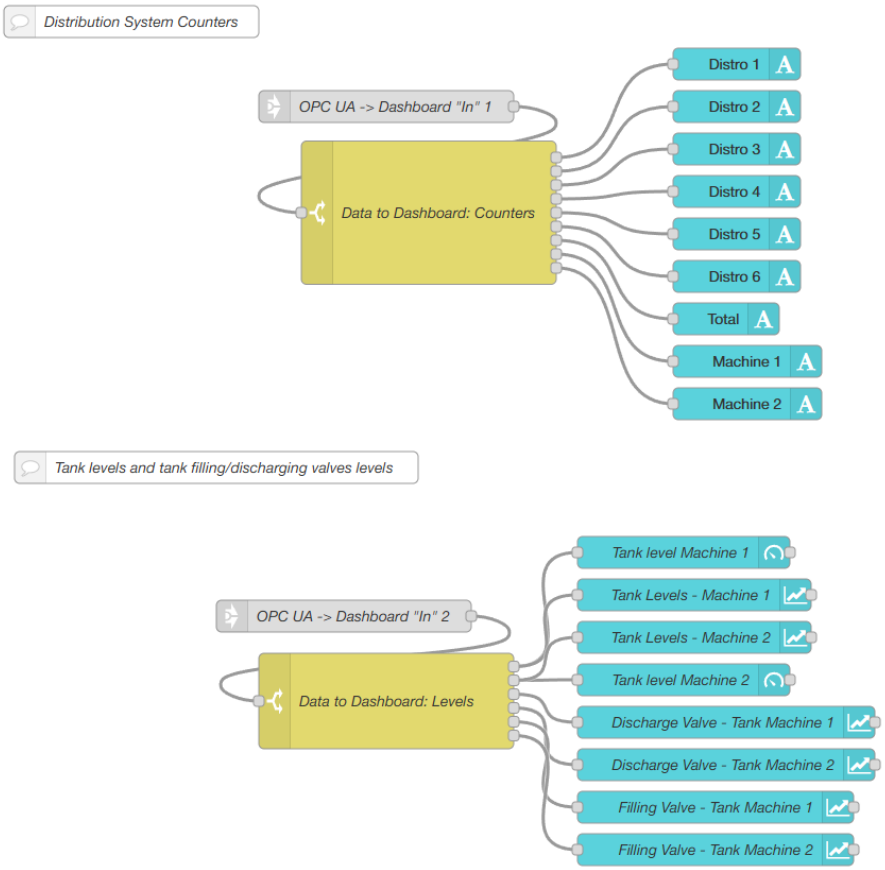

Figure 4.6 – Node-RED flow for dashboard visualisation

In the control section (see Figure 4.7), buttons (*ui-button* nodes) and switches (*ui-switch* nodes) were added to start and stop the system, activate machines, select available distribution stations, and control specific functions, such as part stacking. *Trigger* nodes were used to generate timed signals with a duration sufficient for the PLC to acquire them correctly. Alarm, stop, and emergency mechanisms were also added to the interface, ensuring the monitoring of operating conditions and a response to faults. All the controllable signals are sent back to the PLC through OPC UA. With these configurations, the IOT2050 is ready to collect data, visualise it, preprocess it, and send it to the different cloud platforms.

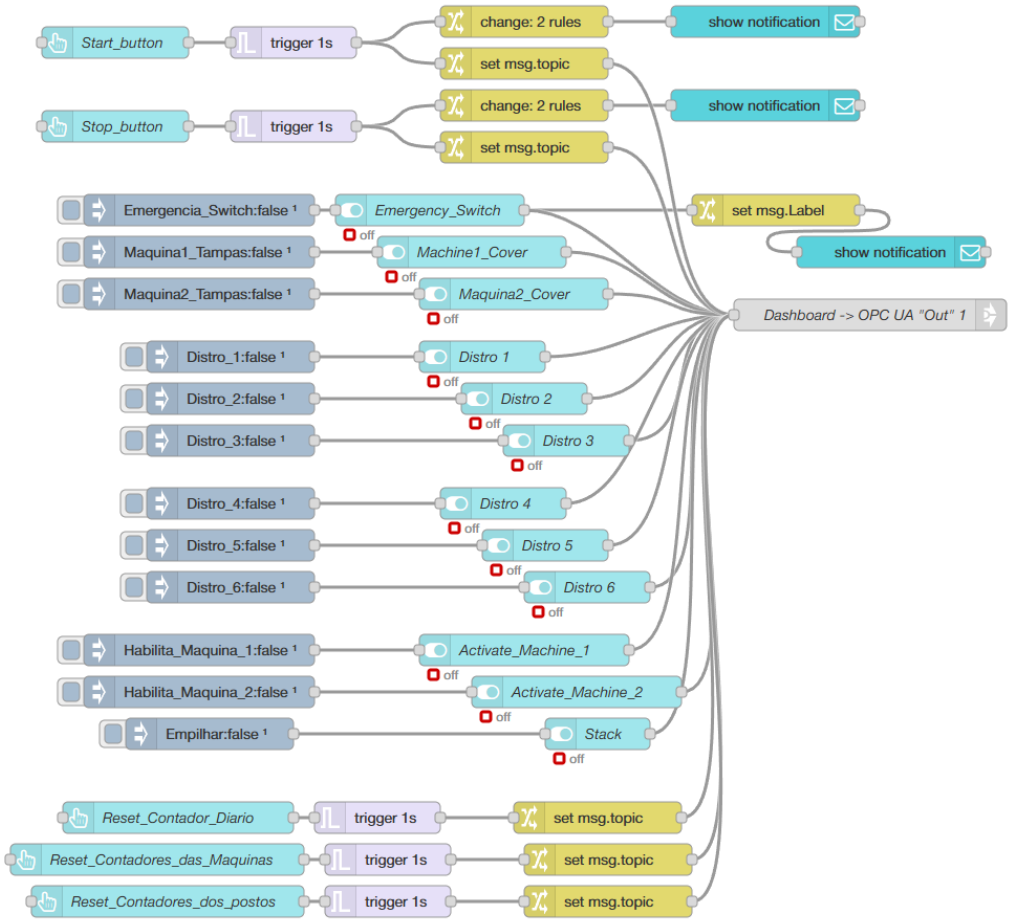

Figure 4.7 – Node-RED flow for dashboard control

4.2.Communication between IIoT Gateway and Clouds

In this section, the focus shifts from the shopfloor-side acquisition discussed previously to the cloud-side integration of the same industrial data. Following up on the case study introduced, this section explains how it works to establish communication between the IIoT gateway and multiple cloud platforms, recognising that each platform has different limitations for data ingestion, modelling, and processing.

Before transferring data to different cloud platforms, it must be pre-processed and structured according to each service's ingestion format. As demonstrated in Figure 4.8, industrial data from the OPC UA client is first organised by type (counters, levels, alerts, velocities), then joined, and finally redirected to separate outputs via *link in* and *link out* nodes.

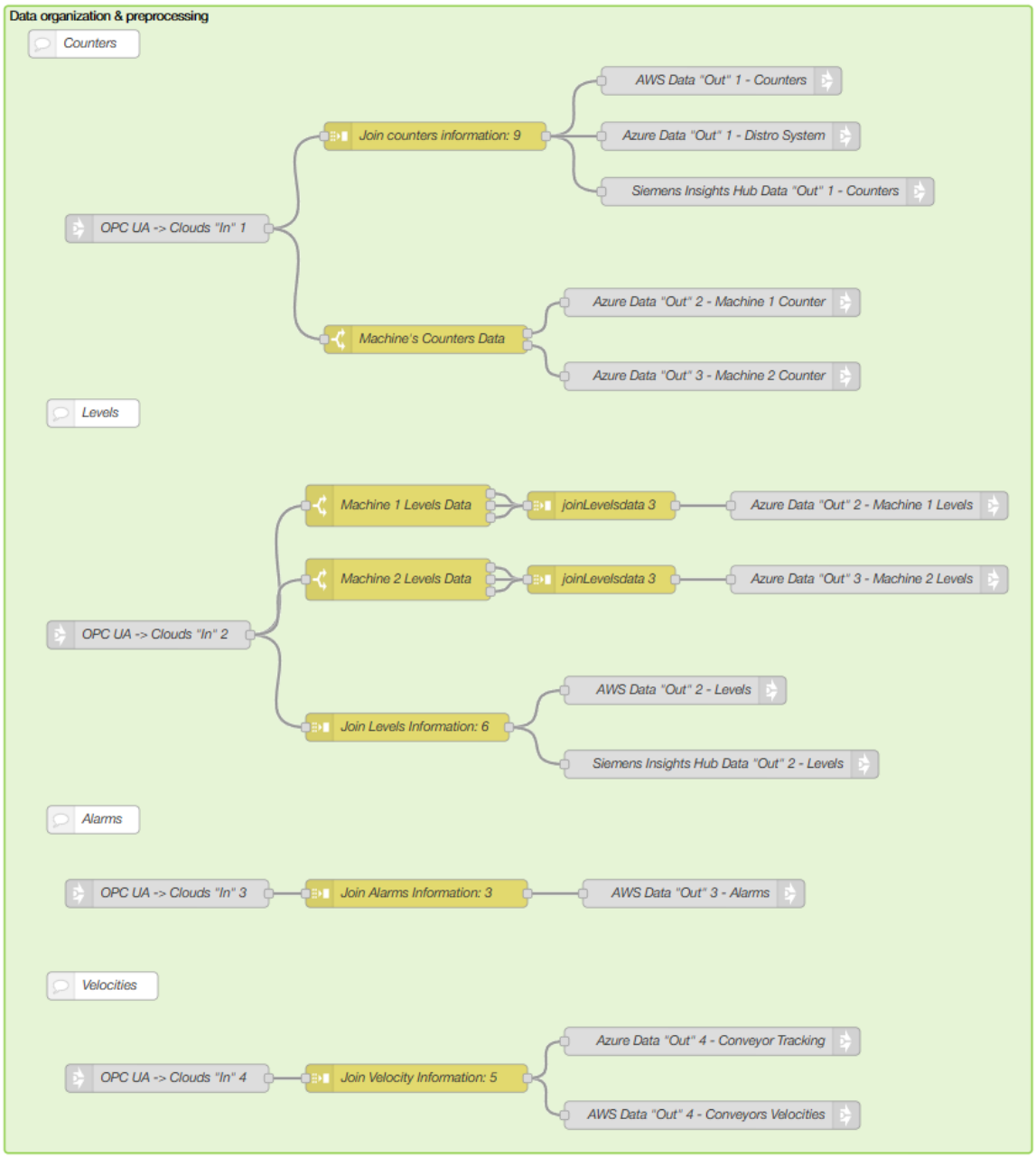

Figure 4.8 – How data is organised and pre-processed in "Communication Node-RED <-> Clouds"

Following this preparation, the section explains how communication is implemented with Siemens Insights Hub (Section 4.2.1), AWS IoT SiteWise (Section 4.2.2), and Microsoft Azure Digital Twins (Section 4.2.3), highlighting the configuration in the cloud platforms and in the communication protocols necessary to connect the IIoT Gateway to the clouds.

4.2.1. Siemens Insights Hub

Siemens Insights Hub (formerly known as Siemens MindSphere) is a cloud-based platform used to connect machines, sensors and systems to the cloud to achieve higher equipment availability, production performance, quality rate and sustainability in the shopfloor, by analysing industrial data in real time, detecting faults and creating low-code applications [39]. Despite being a PaaS platform, Insights Hub can be considered as IIoT-as-a-Service,

as it provides centralised compute and storage along with solutions, apps, and services. Siemens uses the concept of extracting data from installed data sources (e.g., a PLC or an IoT sensor with MQTT capabilities) to send industrial data either directly or via a dedicated gateway, and map to the platform data model, as depicted in Figure 4.9 [40].

.

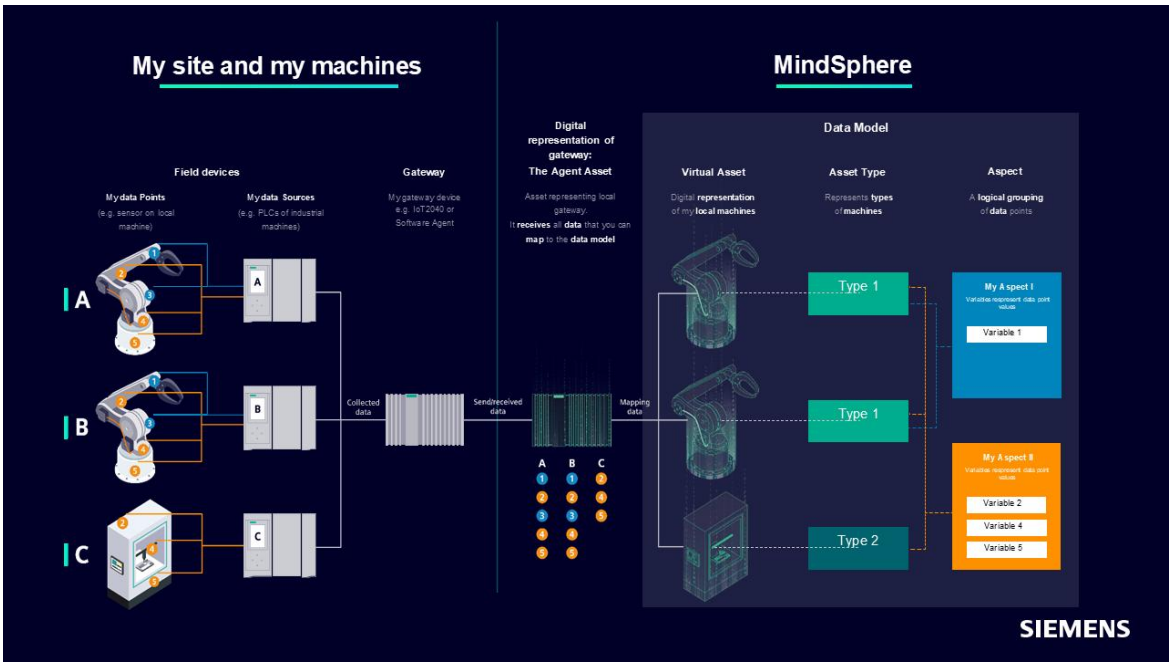

Figure 4.9 - Industrial data acquisition and data modelling architecture in Siemens Insights Hub

Unlike other cloud platforms, this cloud does not use a pay-as-you-go model; instead, it charges an annual subscription fee. The Start for Free subscription plan was used to configure and evaluate the platform. However, as the provider discontinued this plan at the end of May 2025, further experimentation and performance assessment were no longer possible without a paid subscription. This subscription offered the basic services (as can be seen in Figure 4.10) to connect devices and model the structure of a machine or industrial process allowing the organisation of industrial variables (Asset Manager), monitor production lines and machines through a common interface that integrate the most used displays and configurations (Operations Insight), design workflows by drag and drop functionality to develop graphic depiction of workflows (in a similar way that Node-RED does) to read time series data, create a dashboard or calculate KPIs (Visual Flow Creator), build web applications quickly through skeleton code (Developer Cockpit), and run low-code applications (Mendix) [42].

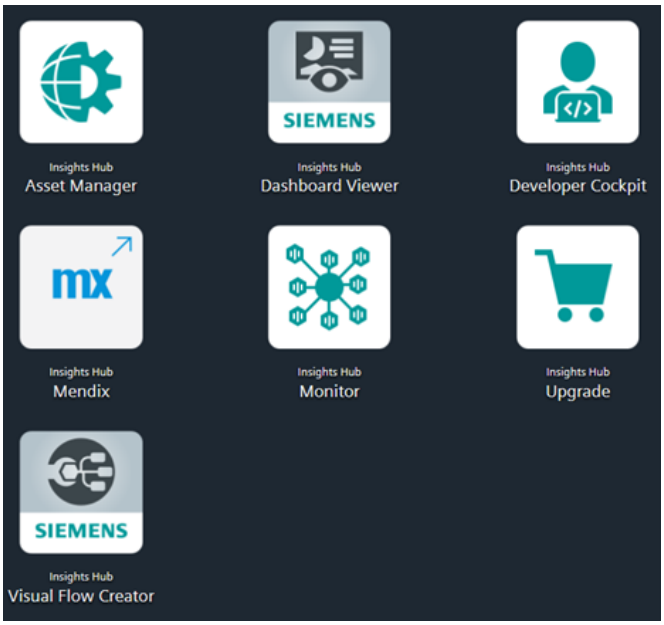

Figure 4.10 – Offered services in the Insights Hub Start for Free Subscription

To enable communication between data sources and this cloud, it is necessary to create a Digital Twin (a virtual model of a machine or automated process) in Asset Manager using Aspects, Asset Types, and Assets. Aspects are the digital representation of the equipment and a data modelling mechanism for Assets, in which data points are grouped by their logical association. These can be used to form various variables. Asset Types (or Types) are pre-configured models for an Asset, in which the asset itself assumes the properties of the Type. It is possible to define, within each Asset Type, which Aspects are integrated into the model. Finally, there are the Assets, which are digital representations of a combination of equipment or a machine with one or more automation units (e.g., a PLC) connected to the Insights Hub [41], [43]. Figure 4.11 illustrates these concepts.

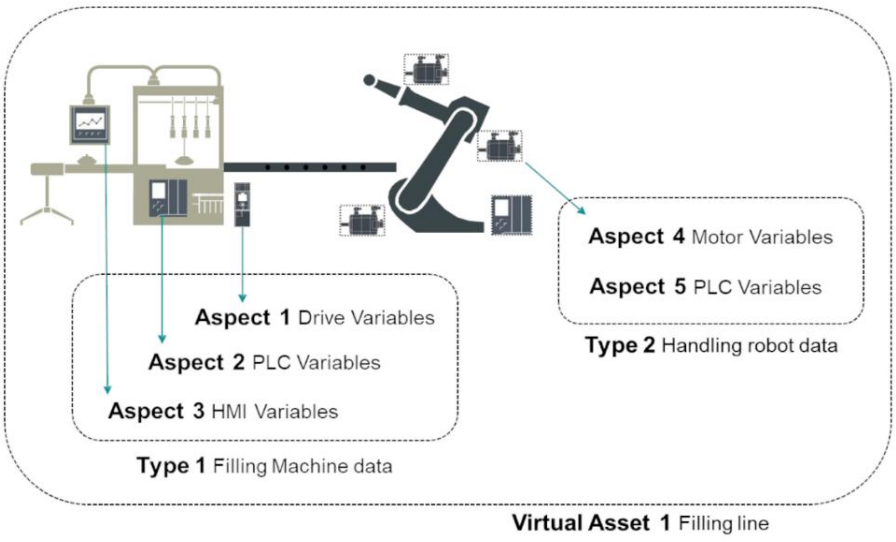

Figure 4.11- Identification of Aspects, Types and Assets in Insights Hub

The Start for Free subscription had limitations not only in the quantity of Asset, Types, or Aspects instances created (10 Assets and 3 Types), but also in data ingestion rate, data

storage volume, compute hours in some services, and other resources [42]. With these constraints, a solution was created that could overcome them, though it remained more restricted than the other cloud services presented later. In this case study, 2 Aspects (*Machine1_variables* and *Distibution_system_variables*), 1 Asset Type (*Machine_Type*) and 1 Asset (*Production_Line_Asset*) were created, as shown in Figure 4.12.

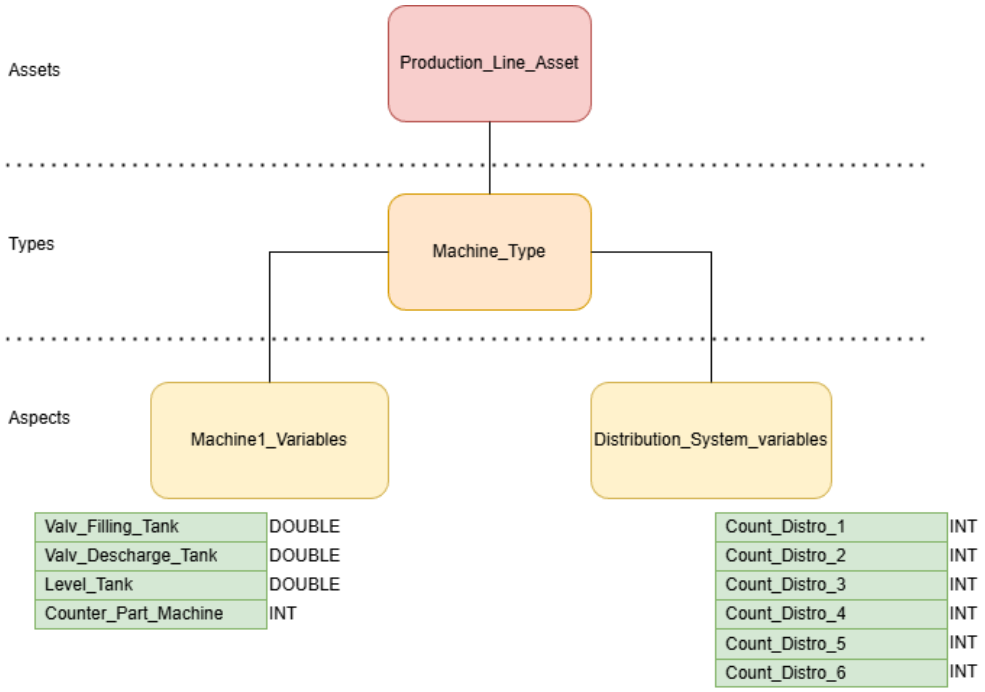

Figure 4.12 – Data model used in Siemens Insights Hub

In Insights Hub, industrial data can be sent via a variety of MindConnect hardware and software solutions, complemented by network services for secure remote access. There are dedicated IIoT gateway devices, such as MindConnect IoT2040, that allow simple onboarding and bidirectional data transfer for individual machines or small production lines. In large shopfloor environments, high-performance gateways, such as MindConnect Nano and MindConnect IoT2050, support multiple connections and data collection via standard industrial protocols to simplify data collection across devices and locations. When on the shopfloor, already have hardware (for example, an Industrial PC or a machine capable of running Windows-based software), the MindConnect Software Agent can be implemented as a gateway. At the same time, MindConnect MQTT enables the integration of devices, applications, and services that can only send industrial data through MQTT protocol. Also, the Insights hub includes the MindConnect API/LIB, which allows engineers to build custom data ingestion software using the SDK and RESTful APIs [44].

In this study case, MindConnect LIB via the *MindConnect Node-RED Agent* node (*@mindconnect/node-red-contrib-mindconnect* palette), requiring that Asset Type should be configured to use MindConnect Lib as the core type. After creating the Asset, an onboarding key must be generated to grant IoT2050 access to connect to the Insights Hub, as shown in Figure 4.13, which presents the procedure to obtain the key [45],[46].

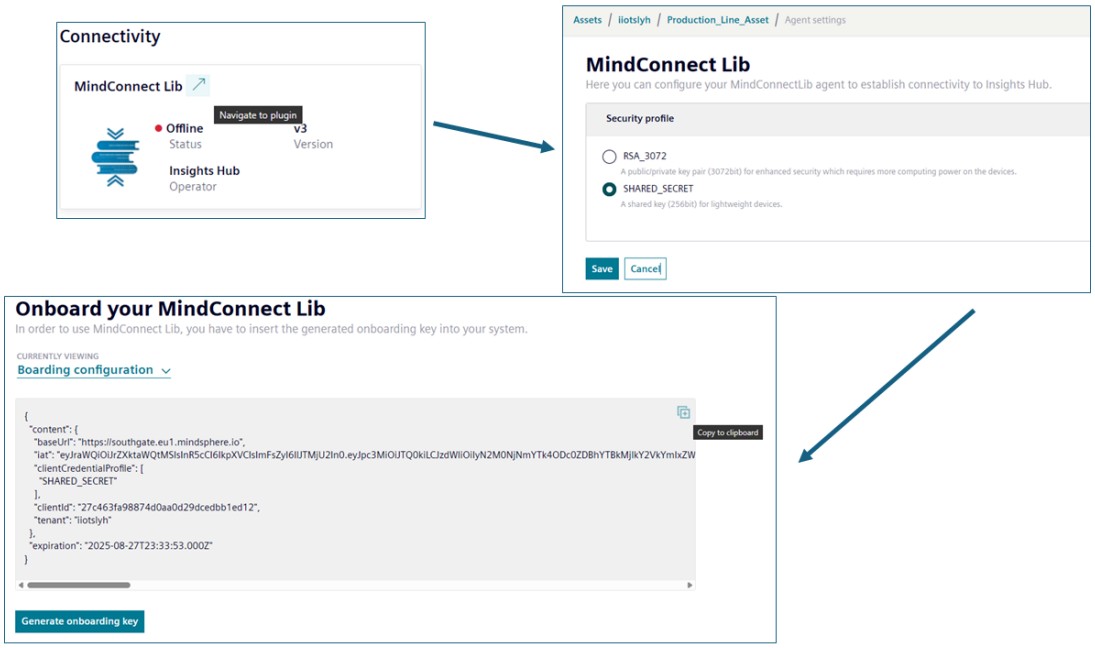

Figure 4.13 – Procedure used to generate an onboarding key in Insights Hub

The onboarding key must be copied and pasted into the MindConnect Node-RED Agent node to complete the onboarding process, so it is possible to select the desired Asset to automatically create the data source, data mapping, and a time series JSON template to be inserted into a *function* node in Node-RED. The final result of the communication between the IOT2050 and Insights Hub through Node-RED is depicted in Figure 4.14, which includes a batcher node, that collects sequences of payloads and sends them in a single message.

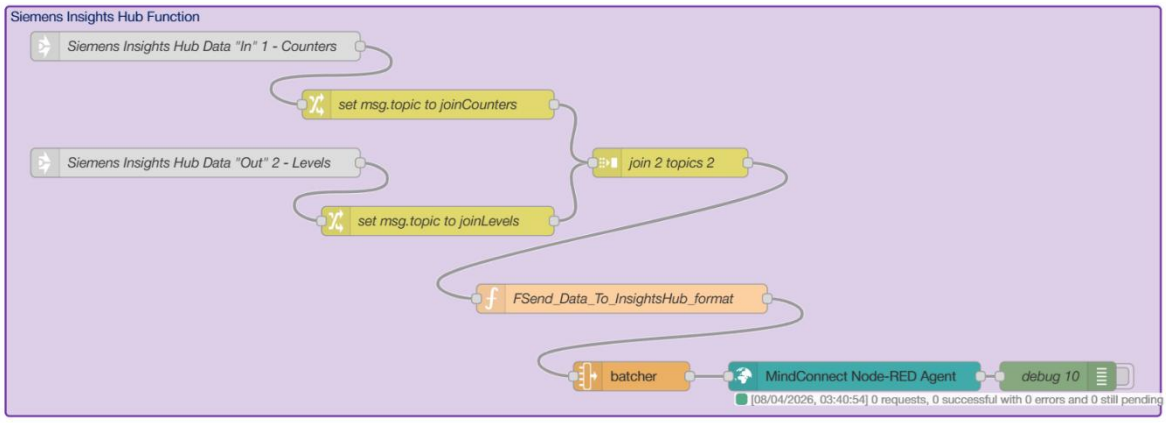

Figure 4.14 – Node-RED flows for publishing data in Insights Hub

With all configurations complete, it is possible to view industrial data through the Insights Hub Monitor (Figure 4.15), predict, create dashboards and alarms, and use other functionalities offered by this cloud platform.

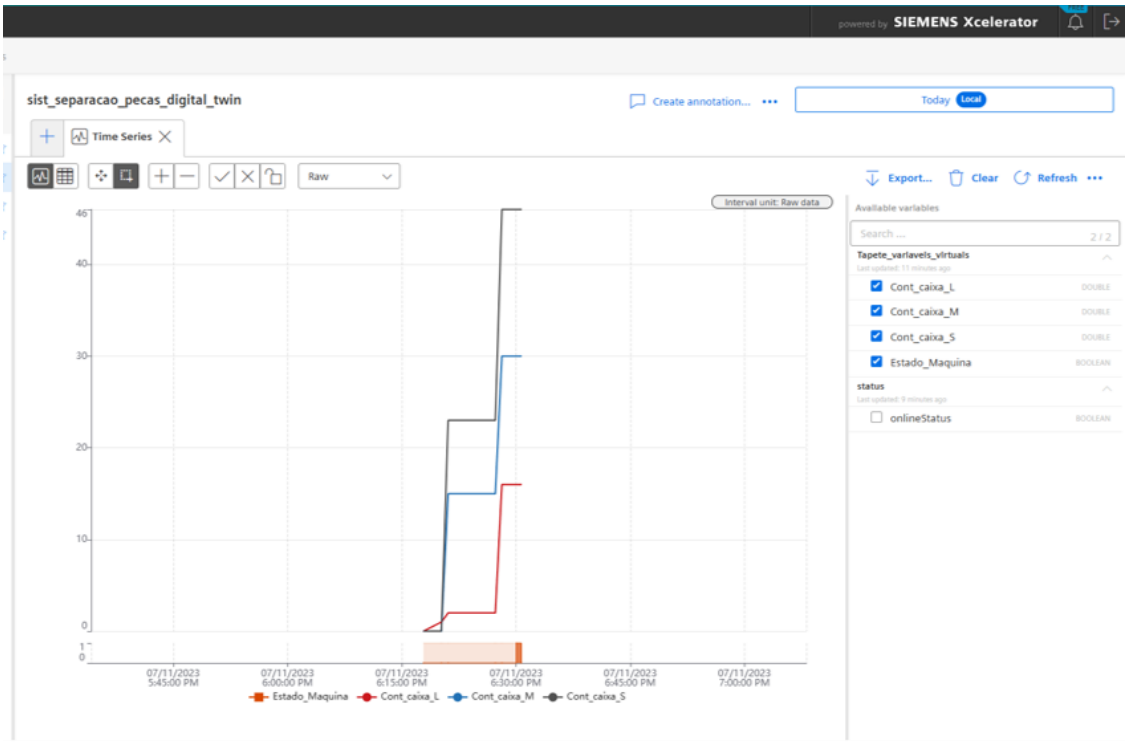

Figure 4.15 – Insights Hub Monitor graphical chart example

4.2.2. Amazon Web Services (AWS)

AWS IoT SiteWise is an IIoT service that allows users to collect, organise, store, and monitor data from industrial equipment and processes at scale, making it usable for operational monitoring and analysis and enabling easier decision-making. With this service, you can view data through visualisation tools and store it. Also, it is possible to create alerts when a process deviates from its optimal performance and to detect anomalies using machine learning [47]. Figure 4.16 shows in a resume how this cloud service works.

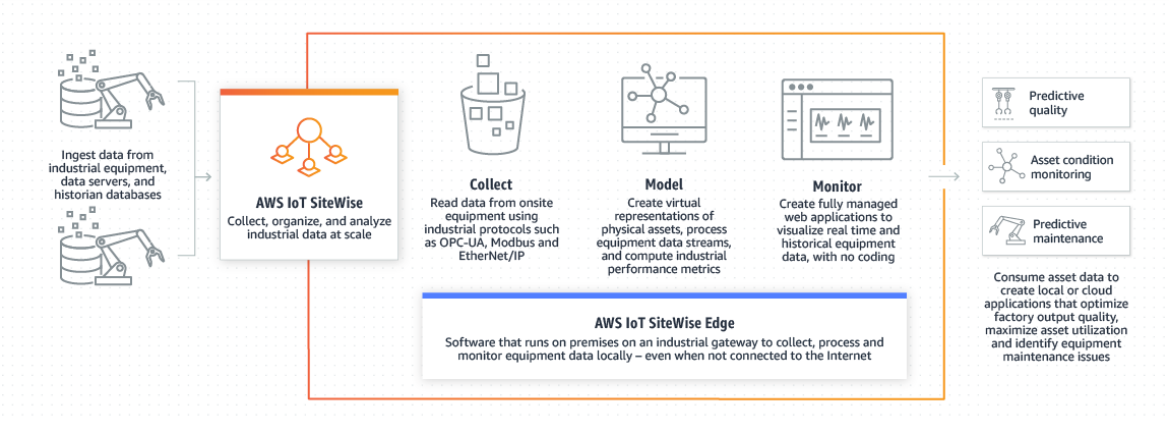

Figure 4.16 – AWS IoT SiteWise overview

To ensure industrial data is sent correctly, it is important to understand that AWS IoT SiteWise uses an asset modelling framework to create a digital representation of industrial

equipment/processes. This modelling is organised into four elements: asset models, assets, hierarchies, and properties. The asset model is defined by the type of equipment or process, and existing properties can be defined, creating a reusable "template" for multiple assets. Meanwhile, Assets are the actual instances of an asset model, representing specific pieces of equipment or processes. Hierarchies are used to describe the relationships between assets, creating structures such as “line→machine→component” and making it easier to understand how the industrial environment works. Properties are the parameters of the asset and can include time measurements, static measurements, and values derived from transformations or aggregations (Transforms and Metrics). Each property can have an alias that allows it to be identified when transferring data from or to other services, facilitating telemetry [49], [50].

Four asset models were created, resulting in five assets: *Factory_1* (represents the whole factory - Table 4.1), *Machine_1* (Table 4.2), *Machine*_2 (assets related to the two machines from the simulation), *Conveyor_Tracking_1* (asset responsible for the factory’s conveyors - Table 4.3) and finally, *Distribution_System*_1 (asset that represents the six distribution stations -

Table 4.4). Also, a hierarchy was established, with models related to machines, conveyors, and distribution stations beyond the factory model.

Table 4.1 – Properties from *Factory_1* asset

Property Name	Data Type	Property Alias
No_Conditions_toStack	Boolean	/Factory_1/No_Conditions_toStack
No_Distro_Active	Boolean	/Factory_1/No_Distro_Active
No_Machine_Active	Boolean	/Factory_1/No_Machine_Active

Table 4.2 – Properties from *Machine_1* asset

Property Name	Data Type	Property Alias
Level_Tank	Double	/Factory_1/Machine_1/Level_Tank
Valv_Discharge	Double	/Factory_1/Machine_1/Valv_Discharge
Valv_Filling	Double	/Factory_1/Machine_1/Valv_Filling
Counter_Part_Machine	Integer	/Factory_1/Machine_1/Counter_Part_Machine

Table 4.3 – Properties from Conveyor_Tracking_1 asset

Property Name	Data Type	Property Alias
vel_CT3	Double	/Factory_1/Conveyor_Tracking_1/vel_CT3
vel_CT4	Double	/Factory_1/Conveyor_Tracking_1/vel_CT4
vel_CT5	Double	/Factory_1/Conveyor_Tracking_1/vel_CT5
vel_CT6	Double	/Factory_1/Conveyor_Tracking_1/vel_CT6
vel_CT7	Double	/Factory_1/ConveyorTracking_1/vel_CT7

Table 4.4 – Properties from Distribution_System_1 asset

Property Name	Data Type	Property Alias
Count_Distro_1	Integer	/Factory_1/Distribution_System_1/Count_Distro_1
Count_Distro_2	Integer	/Factory_1/Distribution_System_1/Count_Distro_2
Count_Distro_3	Integer	/Factory_1/Distribution_System_1/Count_Distro_3
Count_Distro_4	Integer	/Factory_1/Distribution_System_1/Count_Distro_4
Count_Distro_5	Integer	/Factory_1/Distribution_System_1/Count_Distro_5
Count_Distro_6	Integer	/Factory_1/Distribution_System_1/Count_Distro_6
Count_Distro_Total	Integer	/Factory_1/Distribution_System_1/Count_Distro_Total

Industrial data can be sent to AWS IoT SiteWise using different methodologies, depending on where the data is generated and where it operates [51]. The way a solution is designed when data comes from factory machines will differ from when data is already in a cloud service. On the shopfloor, a gateway can collect information from equipment via industrial protocols such as OPC UA and send it to SiteWise. In cloud scenarios, it is also possible to send measurements directly to SiteWise through its APIs, which is helpful when applications or services already produce the data. Another common approach is to receive telemetry via AWS IoT Core (via MQTT, HTTPS, or LoRaWAN) and forward it to SiteWise via rules, allowing easy separation of data collection from its final destination and making the architecture more flexible to changes. In all cases, it is necessary to record time series for assets and properties for subsequent monitoring and metric calculation. For the case study, the last methodology was chosen because it is the easiest to implement and provides the most aggregated information in research sources, as shown in Figure 4.17.

Figure 4.17 - Architecture used to connect data to AWS IoT SiteWise with output possibilities

In AWS IoT Core, the concept of a Thing is a virtual representation of a physical device, such as a sensor or a PLC. Things can be identified by a unique name, and can have attributes to store additional relevant information, such as serial number or manufacturer, making it easier to manage and track assets. Each Thing can be associated with policies that define

what actions a device can perform, such as connecting, publishing, subscribing, and receiving messages, as presented in Figure 4.18 [52],[53].

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "iot:Connect",
        "iot:Publish",
        "iot:Subscribe",
        "iot:Receive"
      ],
      "Resource": "*"
    }
  ]
}
```

Figure 4.18 – AWS IoT Core policy for MQTT connection

Using Things, it is also possible to obtain device certificates with the corresponding private keys, which enable encryption and authentication between an IIoT device and the cloud. In Node-RED, communication was established through the *Mqtt in* and *Mqtt out* nodes, which were configured to use these credentials to connect to the AWS IoT Core. The properties were organised into functions for counters, levels, alarms, and conveyor tracking velocities, as shown in Figure 4.19. In each of them, epoch timestamps in seconds were incorporated and required by SiteWise.

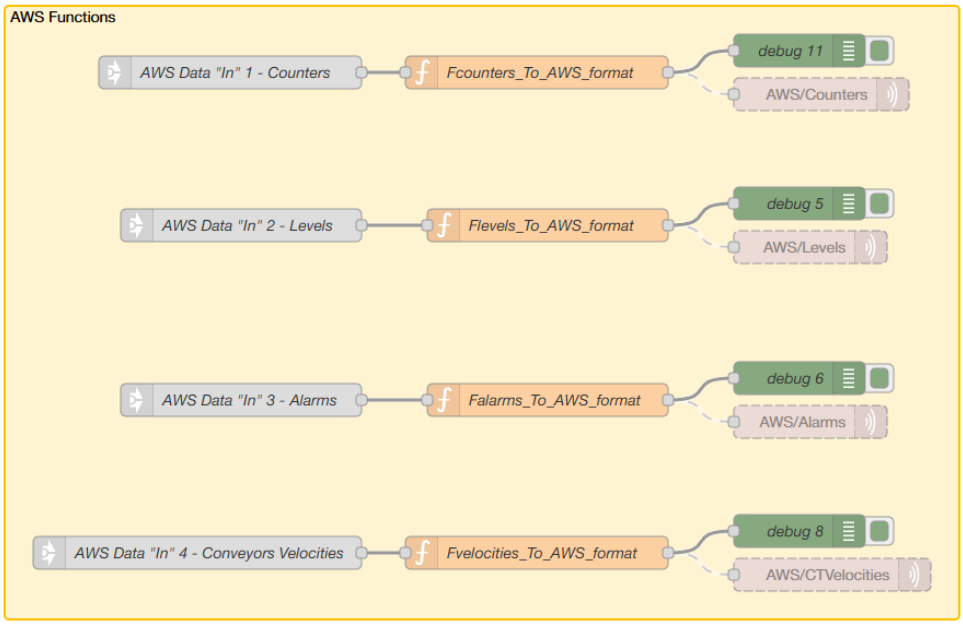

Figure 4.19 – Node-RED flows for publishing data to AWS IoT Core

The payload message sent from IOT2050 can be seen in AWS IoT Core using an MQTT test client (Figure 4.20), which allows direct interaction with the topics by writing the topic name

in the *subscribe* field. It also allows publishing messages to the selected topic, which can be very useful for sending test payloads to the IIoT gateway.

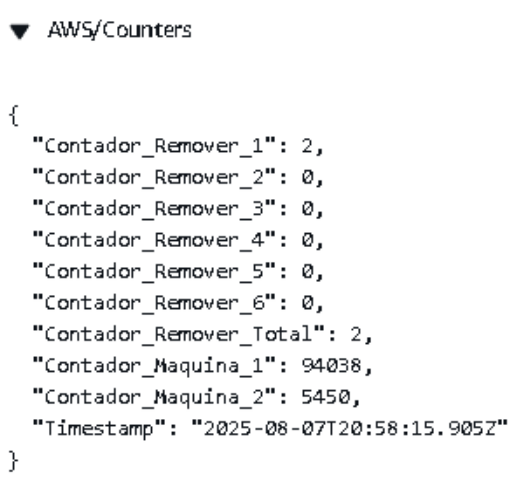

Figure 4.20 – MQTT payload example shown in the AWS IoT Core MQTT test client

Telemetry between AWS IoT Core and AWS IoT SiteWise was implemented through rules that process newly arrived MQTT messages and map the content of each topic to the desired cloud service [54]. To implement this, an SQL statement was applied in each rule to filter all messages by the desired MQTT topic. Once filtered, an action related to SiteWise is executed, in which each parameter of the filtered topic's message is sent to the corresponding asset property in SiteWise, using the property alias. To ensure the rule executes correctly, keep the order of the parameters in the topic, ensure the received values correspond to the configured properties, and use the defined data type in the SiteWise properties configuration.

All rules must have an IAM (Identity and Access Management) role with an attached policy that provides temporary access to other services without requiring permanent credentials [55]. In this case, the policy action linked to this role was *BatchPutAssetPropertyValue*, which allows the values of the topic's properties to be set in the assets' properties [56], as shown in Figure 4.21.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "iotsitewise:BatchPutAssetPropertyValue",
            "Resource": "*"
        }
    ]
}
```

Figure 4.21 – AWS IoT SiteWise policy to allow sending asset property values using *BatchPutAssetPropertyValue*

With the configurations described, the architecture is ready for operational data ingestion, storage, and visualisation in AWS IoT SiteWise, as well as for anomaly detection in industrial data. It is also possible to enable MQTT status notifications in SiteWise, which

confirm the arrival of data at the respective service on the IIoT gateway using topics generated by the service itself.

To visualise data and historical trends using charts, additional tools that provide this functionality are needed. There are several options available, including AWS IoT SiteWise Monitor, Amazon Managed Grafana and Grafana Cloud. SiteWise Monitor is designed to work only with AWS IoT SiteWise and includes managed dashboards, an asset explorer, and alarm management, making it ideal for basic monitoring scenarios. Amazon Managed Grafana is also a well-known tool for visualising industrial data, provided by AWS as a managed service, meaning the infrastructure required for Grafana and its dashboards to operate is supported and maintained by the cloud provider. This tool provides simplified integration with other AWS services, including AWS IoT SiteWise, enabling SiteWise to serve as a data source for dashboards.

Despite the ease with which these two tools integrate data, they incur usage costs. As an alternative, Grafana Cloud can be used free of charge, though with limitations. However, it offers the same tools as the others, which is sufficient for this case study.

To connect a self-managed instance of Grafana with AWS IoT SiteWise, it is necessary to install Grafana's SiteWise plug-in and use it to configure a data source that establishes communication with AWS through an Access Key ID and a Secret Access Key, generated by an IAM User (which can represent a user or an application, in this case, Grafana) and holds access keys to access other services. An IAM User also has actions defined by policies that, in this case, authorise the reading of the current asset property values (*BatchGetAssetPropertyValue*) and their historical data at a specific time interval (*BatchGetAssetPropertyValueHistory*), as shown in Figure 4.22.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iotsitewise:Describe*",
                "iotsitewise:Get*",
                "iotsitewise:List*",
                "iotsitewise:BatchGetAssetPropertyValue",
                "iotsitewise:BatchGetAssetPropertyValueHistory"
            ],
            "Resource": "*"
        }
    ]
}
```

Figure 4.22 – AWS IoT SiteWise policy to allow read-only access (Describe/Get/List) and gathering of asset property values and history for Grafana

With the configurations set, it is possible to choose various graphical charts that characterise the assets' properties over time and set alerts to automatically detect deviations from predefined thresholds, as shown in Figure 4.23.

Figure 4.23 – Grafana Cloud dashboard used in the case study

4.2.3. Microsoft Azure Digital Twins (ADT)

Azure Digital Twins (ADT) is a Platform-as-a-service (PaaS) solution that enables the creation of twin graphs from digital models of complex environments, such as buildings, industrial facilities, agricultural sites, energy infrastructure, railway systems, stadiums, and even urban areas, as depicted in Figure 4.24. These digital models can provide useful insights to improve product development, operational efficiency, and reduce costs, creating better consumer experiences [57]. Also, this service allows the import of 3D scenes of the industrial environment to monitor, diagnose, and investigate operational digital twin data.

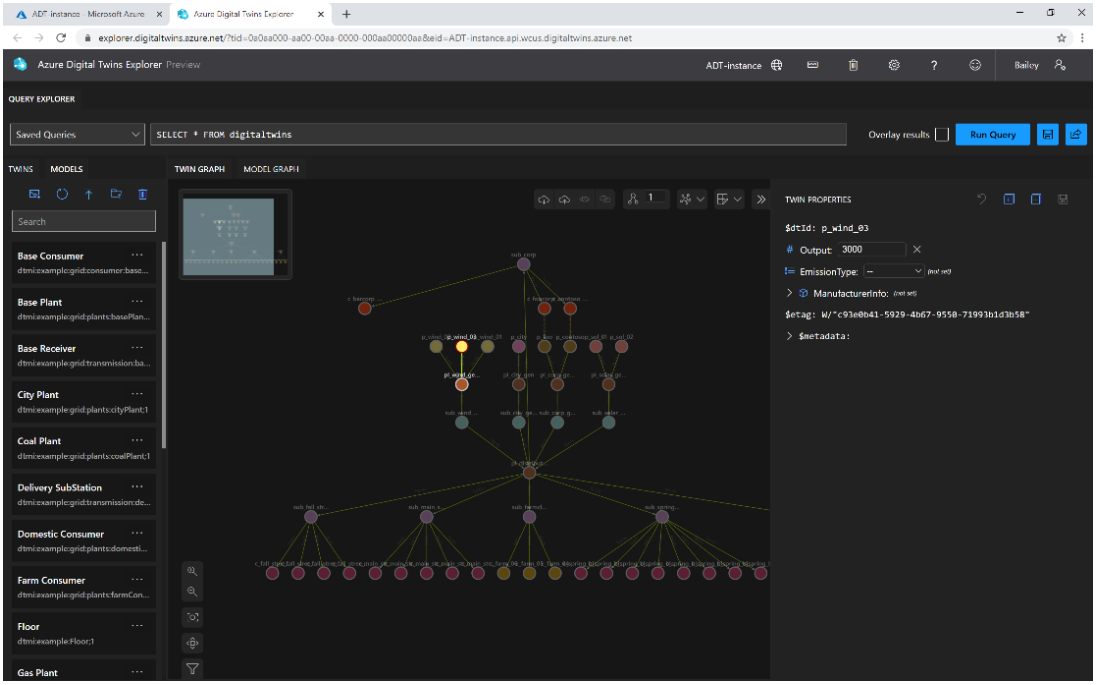

Figure 4.24 – Azure Digital Twin Explorer page

ADT uses the Digital Twin Definition Language (DTDL), a JSON-LD-based modelling language, to define digital twin models of industrial components, such as motors, pumps,

sensors, and production lines. As shown in Figure 4.25, a DTDL model is specified by an *@id* that assigns a unique identifier to the model (named Digital Twin Model Identifier), a *@type* to indicate the kind of definition that is given (e.g., an interface that represents a twin type), a *@context* that establishes the JSON-LD context that maps terms to the DTDL vocabulary and version, ensuring that the model is analysed according to a defined semantic schema, and finally, a content, that has the main information of the model given by its attributes. It can have a property (that represents the state of an entity and can be read at any time), a relationship (that represents how digital twins are connected), or a component (that allows building a model interface as an assembly for other interfaces) [58].

Figure 4.25 – Declaration of a DTDL model

Using the examples and templates provided by Microsoft Azure in [59], it was possible to create the models, with the properties and the relationships necessary for this case. It was created 5 models: *Machines* (Table 4.5), *ConveyorTracking* (Table 4.6), *DistributionSystems* (Table 4.7), *ProductionLine* (Table 4.8), *Factory*. All models have properties related to the levels of the tank's machines, the velocities of the conveyors, and the counters, except for ProductionLine and Factory, which have only contents related to Relationships to join all the models.

Table 4.5- Content from the *Machines* Digital Twin model

Name	Type	Schema (for the properties)	Target (for the relationships)
Level_Tank	Property	Float	----------
Valv_Discharge_Tank	Property	Float	----------
Valv_Filling_Tank	Property	Float	----------
Counter_Part_Machine	Property	Integer	----------
containsMachines	Relationship	----------	Dtmi:example:machines;1

Table 4.6 - Content from the *ConveyorTracking* Digital Twin model

Name	Type	Schema (for the properties)	Target (for the relationships)
vel_CT3	Property	Float	----------
vel_CT4	Property	Float	----------
vel_CT5	Property	Float	----------
vel_CT6	Property	Float	----------
vel_CT7	Property	Float	----------

Table 4.7 – Content from the *DistributionSystems* Digital Twin model

Name	Type	Schema (for the properties)	Target
Count_Distro_1	Property	Integer	----------
Count_Distro_2	Property	Integer	----------
Count_Distro_3	Property	Integer	----------
Count_Distro_4	Property	Integer	----------
Count_Distro_5	Property	Integer	----------
Count_Distro_6	Property	Integer	----------
Count_Distro_Total	Property	Integer	----------

Table 4.8 - Content from the *ProductionLine* Digital Twin model

Name	Type	Schema	Target
containsMachines	Relationship	---------	dtmi:example:Machines;1
containsConveyorTracking	Relationship	---------	dtmi:example:ConveyorTracking;1
containsDistributionSystems	Relationship	---------	dtmi:example:DistributionSystems;1

These models can be uploaded manually in ADT Explorer or through APIs, and they can also be created and deleted. Once they are available, the graph can be built in the Explorer by creating twin instances (digital representations of the industrial components) and then

setting the right relationships as configured when the models were created. Also, graphs can be imported via a spreadsheet-based workflow in Excel. The final result of this case study's graph is demonstrated in Figure 4.26.

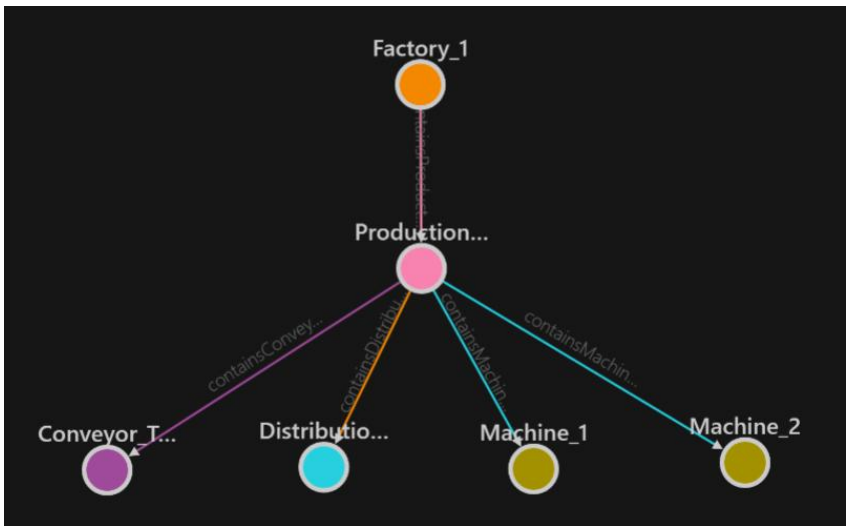

Figure 4.26 – ADT twin graph designed for the case study

There are several ways to send data to this service and also redirect it to other services, such as databases and Grafana. Figure 4.27 shows the possibilities to create an IIoT architecture based on an ADT instance, in which the digital twin graph represents the digital twin assets, relationships, and system context. Data ingestion can happen through workflow integration methods, such as Azure Logic Apps, an IoT Hub that facilitates device telemetry, or any external source capable of communicating via Client apps, which can use bidirectional communication with the service to configure models and extract insights from the twin graph. Event processing can be performed using other external computing resources, such as Azure Functions. Data output can be sent to other services for storage (e.g. Azure Data Lake), time-series analysis, and analytical tasks, enabling monitoring and decision-making processes [57].

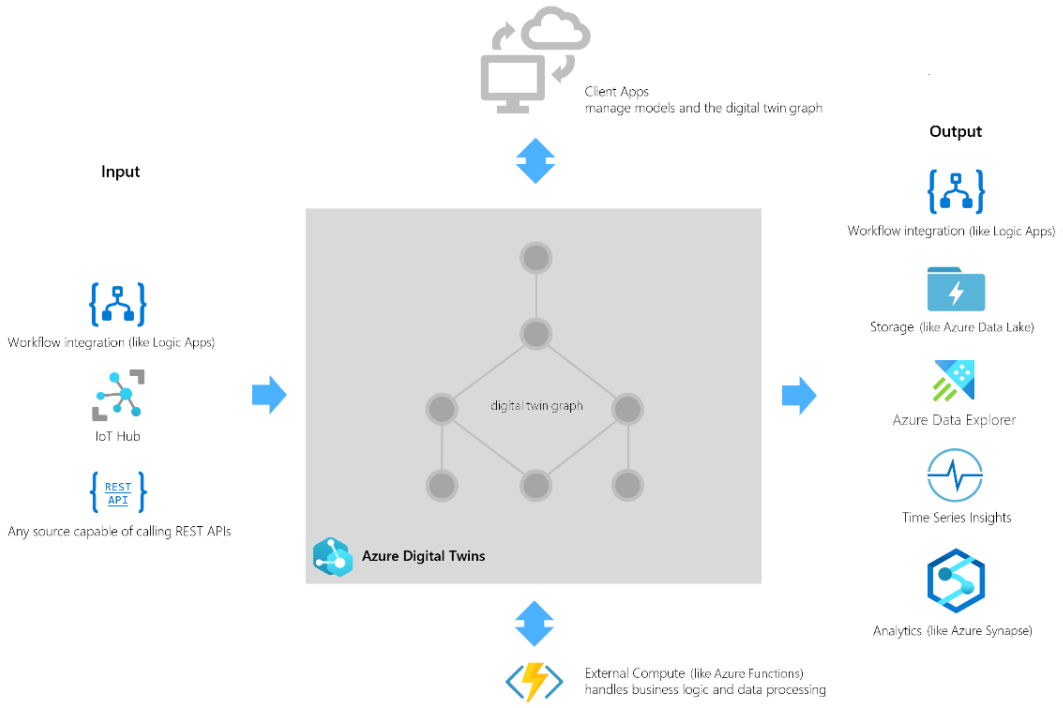

Figure 4.27 – Ways to send data to Azure Digital Twins

As described in subsection 4.1, the way to send data to ADT is through the following architecture, as demonstrated in Figure 4.28, where data from Siemens IoT2050 can be connected to Azure IoT Hub (equivalent to AWS IoT Core) using the MQTT protocol.

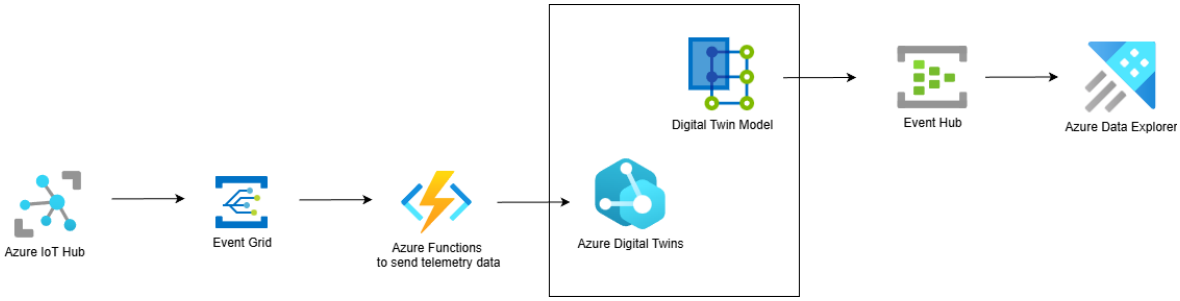

Figure 4.28 – Architecture used to connect data to Azure Digital Twins

There are two main methods to send data from Node-RED to Azure IoT Hub: Using the Azure IoT Hub node provided by the *@node-red-contrib-azure-iot-hub* palette [60], or via the *mqtt out* node [61], shown in Figure 4.29. In Node-RED, the properties were grouped by digital twin model because the telemetry process between cloud services in Microsoft Azure differs from that in AWS. Unlike AWS, which supports rule-based mechanisms for sending data to IIoT services, Azure does not offer an equivalent process and instead requires that industrial data pass through a telemetry function (explained in the following paragraphs).

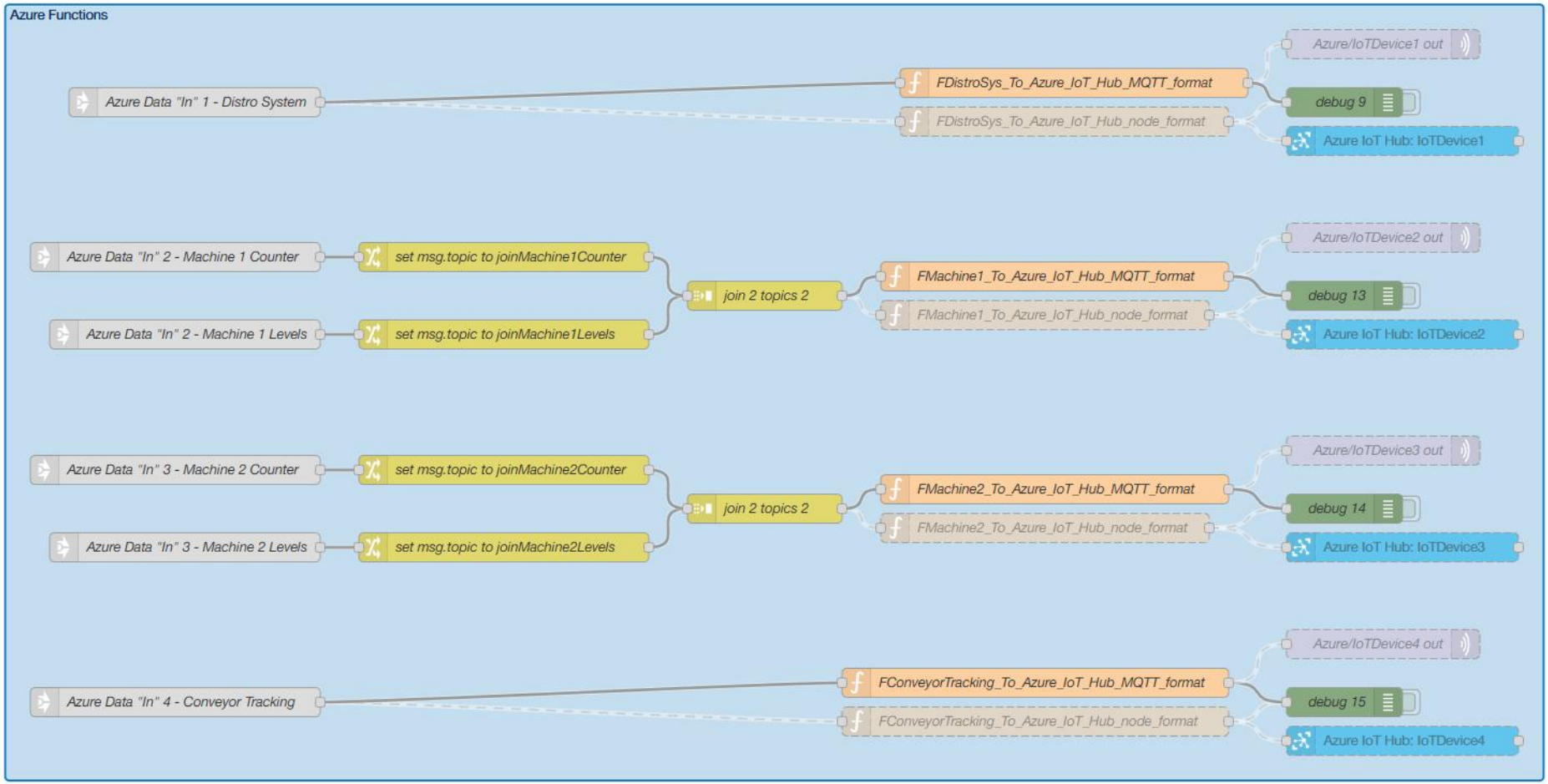

Figure 4.29 – Node-RED flow used to send data from Siemens IOT2050 to Azure IoT Hub

Both methods require registering devices in Azure, but they differ in the flexibility to manipulate the payload and in how much configuration is done automatically. Using the 'Azure IoT hub' node, only the device identification, device key, communication protocol name, and hostname are required, as demonstrated in Figure 4.30. The node will establish the connection automatically without requiring any additional configuration on the cloud side. On the other hand, the payload format is fixed, so it is not possible to fully manipulate the payload's message. If the payload does not include the expected fields (e.g. *deviceId*, *key*, *protocol* and *data*) in the function, an error will appear in Node-RED's debug.

```
var arrays = msg.payload;
var Contador_Remover_1 = arrays[0];
var Contador_Remover_2 = arrays[1];
var Contador_Remover_3 = arrays[2];
var Contador_Remover_4 = arrays[3];
var Contador_Remover_5 = arrays[4];
var Contador_Remover_6 = arrays[5];
var Contador_Remover_Total = arrays[6];
var Contador_Maquina_1 = arrays[7];
var Contador_Maquina_2 = arrays[8];


const values =
{
    "deviceId":"IoTDevice1",
    "key":"xAgCEBP4tKy8Ks5BFF5g3pgAVb40vIpjWsswcqZ3cvg=",
    "protocol":"mqtt",
    "data":{
        "Count_Distro_1": Contador_Remover_1,
        "Count_Distro_2": Contador_Remover_2,
        "Count_Distro_3": Contador_Remover_3,
        "Count_Distro_4": Contador_Remover_4,
        "Count_Distro_5": Contador_Remover_5,
        "Count_Distro_6": Contador_Remover_6,
        "Count_Distro_Total": Contador_Remover_Total
    }

}

msg.payload = values;

return msg;
```

Figure 4.30 – Function used to send data to Azure through the *Azure IoT hub* node

Using the MQTT node in Node-RED, it is possible to manipulate the message payload within the function that sends data to the Azure IoT Hub (Figure 4.31). This method provides greater control over communication settings, including the timing of the device's connection to the cloud platform. However, in this method, it is necessary to manually configure the communication parameters and generate a Shared Access Signature (SAS) token, which serves as an authentication mechanism to establish communication with the Azure IoT Hub [61]. The SAS token is a temporary credential generated from the device's security key, serves as the password to connect to Azure IoT Hub's endpoint via the MQTT protocol, and can be generated in the cloud via the Azure command line or the VS Code Azure IoT Hub extension. Therefore, although this method offers greater freedom in controlling the

payload's message and communication, it also increases the complexity of the configuration process.

```
var arrays = msg.payload;

var Contador_Remover_1 = arrays[0];
var Contador_Remover_2 = arrays[1];
var Contador_Remover_3 = arrays[2];
var Contador_Remover_4 = arrays[3];
var Contador_Remover_5 = arrays[4];
var Contador_Remover_6 = arrays[5];
var Contador_Remover_Total = arrays[6];
var Contador_Maquina_1 = arrays[7];
var Contador_Maquina_2 = arrays[8];


msg.payload = {
    "Count_Distro_1": Contador_Remover_1,
    "Count_Distro_2": Contador_Remover_2,
    "Count_Distro_3": Contador_Remover_3,
    "Count_Distro_4": Contador_Remover_4,
    "Count_Distro_5": Contador_Remover_5,
    "Count_Distro_6": Contador_Remover_6,
    "Count_Distro_Total": Contador_Remover_Total,
    "Count_Machine_1": Contador_Maquina_1,
    "Count_Machine_2": Contador_Maquina_2
};

return msg;
```

Figure 4.31 – Node-RED function used to send data to Azure through the *Mqtt out* node

Once telemetry data are sent to the cloud, the message payload can be viewed to verify the format, content, and data transfer integrity. This can be done using either Azure IoT Explorer, which provides an interface to monitor device-to-cloud messages, or the Azure Cloud Shell, by executing commands that display the received payload in a way similar to that shown in Figure 4.32. It is important to highlight that, under the Azure for Students subscription, Azure IoT Hub, Azure Event Grid, Azure Functions, and Azure Digital Twins can be used without charge [62]. However, these services have limits. Specifically, Azure IoT Hub supports up to 8,000 messages per day and a 5 KB message size limit; Event Grid supports up to 100,000 operations per month; and Azure Functions supports up to 1,000,000 requests per month.


```
{
 "event": {
  "origin": "IoTDevice1",
  "module": "",
  "interface": "",
  "component": "",
  "payload": "{\"Contador_Maquina_1\": 48,\"Contador_Maquina_2\": 48,
\"Contador_Remover_1\":12,\"Contador_Remover_2\":11,\"Contador_Remover_3\":0,
\"Contador_Remover_4\":0,\"Contador_Remover_5\":10,\"Contador_Remover_6\":10,
\"Contador_Remover_Total\":43}"
 }
}
```


Figure 4.32 – Azure IoT Hub payload's format example

After it arrives in Azure IoT Hub, the industrial data is routed through intermediary services: first Event Grid, followed by Azure Functions. Event Grid operates as an "automatic

notification system". Whenever a new message or event is detected (in this case, when telemetry arrives at the IoT Hub), it triggers the next procedure by sending a notification to the subscribed service. which, in this particular case, is an Azure Function that executes only when it is called. The Azure Function receives the event notification, reads the corresponding data, optionally cleans, validates, and transforms it into the required format, and subsequently sends it to Azure Digital Twins by invoking its APIs to update the twin's property values, ensuring the digital representation of the industrial environment stays up to date.

Despite the data flowing through this way, the configuration is done in the reverse order. First, create the Azure function in an Integrated Development Environment (IDE), deploy it, and then create and configure an Event Grid subscription to trigger the function. Microsoft Visual Studio Code (VS Code) with the Microsoft .NET SDK v8.0 was used to develop and debug the function. Azure CLI was also used to deploy the code to the respective cloud service. Four functions were created with the same structure to update data in ADT:

- ProcessHubToDTEventsConveyorTracking → Update data from the Conveyor Tracking System;
- ProcessHubToDTEventsDistroSystem → Update data from the Distribution System;
- ProcessHubToDTEventsMachine1 → Update data from Machine 1;
- ProcessHubToDTEventsMachine2 → Update data from Machine 2.

Using the Udemy course from Lavi [14], information from Doshi [63], and documentation on the template to ingest telemetry from IoT Hub [64], it was possible to build code that could send data to Azure Digital Twins. The code for all functions is divided into three parts. The first part of the code shown in Figure 4.33 demonstrates that an Event Grid event triggers the function, checks whether the *ADT_SERVICE_URL* environment variable is configured (because it contains the Azure Digital Twins endpoint), and establishes an authenticated connection to ADT using *DefaultAzureCredential* and *DigitalTwinsClient*. The identifier of the twin to be updated (*Machine_1*) is also set.

```
private static readonly string adtInstanceUrl = Environment.GetEnvironmentVariable("ADT_SERVICE_URL");
0 references
private static readonly HttpClient httpClient = new HttpClient();

[FunctionName("ProcessHubToDTEventsMachine1")]
// While async void should generally be used with caution, it's not uncommon for Azure function apps,
//  since the function app isn't awaiting the task.
0 references
public async Task Run([EventGridTrigger] EventGridEvent eventGridEvent, ILogger log)

{
    if (adtInstanceUrl == null) log.LogError("Application setting \"ADT_SERVICE_URL\" not set");

    try
    {
        // Authenticate with Digital Twins instance using DefaultAzureCredential
        var cred = new DefaultAzureCredential();
        var client = new DigitalTwinsClient(new Uri(adtInstanceUrl), cred);
        //In the line below, replace with the name of the Digital twin created in the
        // twin graph ($dtId) that you want to update
        var digitalTwinId = "Machine_1";
        log.LogInformation($"ADT service client connection created.");
```

Figure 4.33 – Verification of *ADT_SERVICE_URL*, authentication to Azure Digital Twins, and selection of the target twin

If the event has data, the second part of the code parses the JSON payload, extracts the *deviceId* value from *systemProperties*, and decodes the body field encoded in Base64 to obtain the "actual" JSON that contains the measurements. Then, extracts the telemetry variables (as depicted in Figure 4.34, the variables are *Counter_Part_Machine*, *Level_Tank*, *Valv_Discharge_Tank* and *Valv_Filling_Tank*) and shares their values in the logs.

```
if (eventGridEvent != null && eventGridEvent.Data != null)
{
    // Log the raw event data for debugging purposes
    log.LogInformation(eventGridEvent.Data.ToString());

    // Parse the event data to extract device message and device ID
    JObject deviceMessage = (JObject)JsonConvert.DeserializeObject(eventGridEvent.Data.ToString());
    string deviceId = (string)deviceMessage["systemProperties"]["iothub-connection-device-id"];

    // The body is base64-encoded JSON
    // Decode it to get the actual message
    string bodyBase64 = deviceMessage["body"]?.ToString();
    string bodyJson = System.Text.Encoding.UTF8.GetString(Convert.FromBase64String(bodyBase64));
    JObject body = JObject.Parse(bodyJson);

    // extract multiple telemetry/properties (variables from the device message)
    var counter_Part_MachineToken = body["Counter_Part_Machine"];
    var level_tankToken = body["Level_Tank"];
    var valv_Descharge_TankToken = body["Valv_Descharge_Tank"];
    var valv_Filling_TankToken = body["Valv_Filling_Tank"];

    // Convert tokens to nullable doubles
    double? counter_Part_Machine = counter_Part_MachineToken?.Value<double?>();
    double? level_tank = level_tankToken?.Value<double?>();
    double? valv_Descharge_Tank = valv_Descharge_TankToken?.Value<double?>();
    double? valv_Filling_Tank = valv_Filling_TankToken?.Value<double?>();

    // Share extracted values in logs
    log.LogInformation($"Device:{deviceId} Counter_Part_Machine:{counter_Part_Machine} " +
    $"Level_Tank:{level_tank} Valv_Descharge_Tank:{valv_Descharge_Tank} Valv_Filling_Tank:{valv_Filling_Tank}");
```

Figure 4.34 – Parsing and decoding of the event payload to extract *deviceId* and telemetry values for logging

The third part executes the twin update in ADT by creating a *JsonPatchDocument*, corresponding to a set of "replace value" actions on some of the twin properties, only when

each variable truly has a value (*HasValue*), and by using a flag (*hasUpdates*) to validate that there is at least one change to make. If it verifies that updates are needed, the function uses *UpdateDigitalTwinAsync* to send the patch to ADT, with only the updated properties on the Machine_1 twin, as shown in Figure 4.35.

```
// Update twin with multiple values
var updateTwinData = new JsonPatchDocument();
bool hasUpdates = false;

if (counter_Part_Machine.HasValue)
{
    updateTwinData.AppendReplace("/Counter_Part_Machine", counter_Part_Machine.Value);
    hasUpdates = true;
}
if (level_tank.HasValue)
{
    updateTwinData.AppendReplace("/Level_Tank", level_tank.Value);
    hasUpdates = true;
}
if (valv_Descharge_Tank.HasValue)
{
    updateTwinData.AppendReplace("/Valv_Descharge_Tank", valv_Descharge_Tank.Value);
    hasUpdates = true;
}
if (valv_Filling_Tank.HasValue)
{
    updateTwinData.AppendReplace("/Valv_Filling_Tank", valv_Filling_Tank.Value);
    hasUpdates = true;
}

// Update twin with multiple values
if (hasUpdates)
{
    await client.UpdateDigitalTwinAsync(digitalTwinId, updateTwinData);
}
```

Figure 4.35 – Building a *JsonPatchDocument* with the available telemetry values and sending the update to the target digital twin

After all the functions were deployed to the cloud using the Azure CLI, we defined and configured four Event Grid subscriptions. Each subscription was linked to one of the previously created functions, with the corresponding Azure Function declared as an event endpoint, as shown in Table 4.9.

Table 4.9 – Description of the Event Grid subscriptions with their associated functions

Event Grid Subscription	Functions Associated
dt-event-distrosys-subscription	ProcessHubToDTEventsDistroSystem
dt-event-CTrack-subscription	ProcessHubToDTEventsConveyorTracking
dt-event-Mach1-subscription	ProcessHubToDTEventsMachine1
dt-event-Mach2-subscription	ProcessHubToDTEventsMachine2

Historical data can be accessible in the Azure Digital Twins Explorer[1] [65] as demonstrated in Figure 4.36, if data history's options are configured and enabled on the ADT instance itself [66], To make it possible, it is required to create and configure an Event Hub

[1] It is important to highlight that Azure Data Explorer is not included in the Azure for Students' subscription [62], and the creation and use of the mentioned features of this service have high costs. If the reader has this subscription, the reader has access to 100 US$ of free credit that can be used for this feature. However, is recommended for the reader to use this feature wisely.

namespace and an event hub [67], along with an Azure Data Explorer (Kusto) cluster and database [68]. The Event hub receives notifications of property changes from the ADT instance and then sends the messages to the Azure Data Explorer cluster. The Kusto cluster, along with the database, allows users to receive industrial data from ADT. These data can also be visualised in Grafana, which allows the creation of alerts and other available features [69].

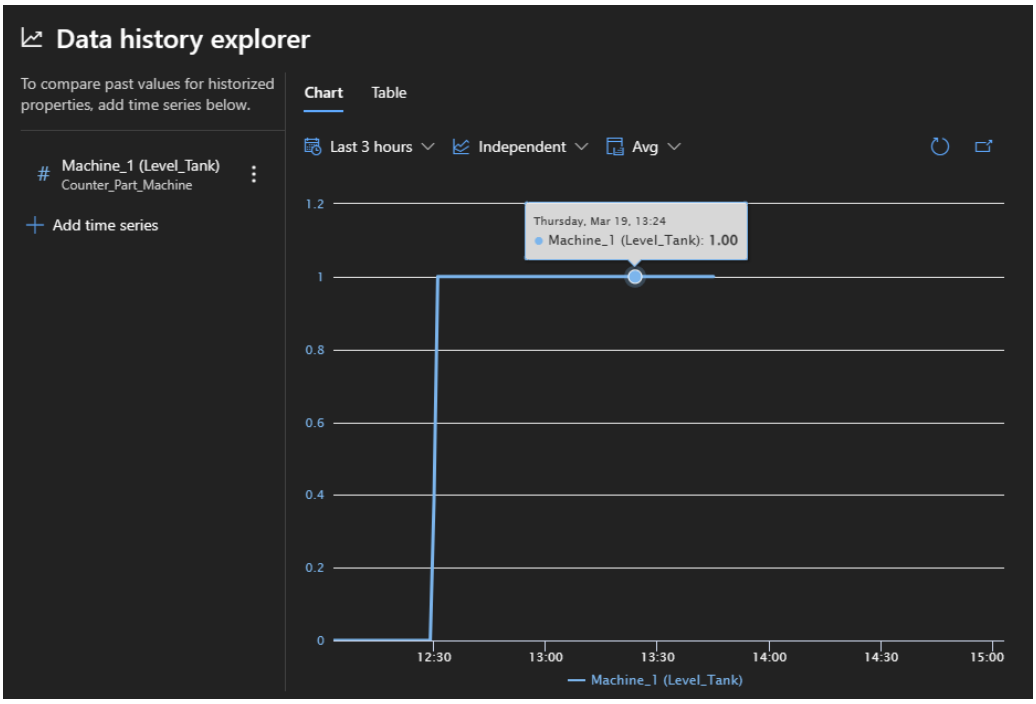

Figure 4.36 – Azure Digital Twins' data history explorer

4.3.Section Synthesis: Implemented Architecture

This section described the practical implementation of the proposed IIoT architecture in a simulated industrial environment. Factory I/O and Siemens S7-1500 PLCs were used to represent the shopfloor process, while the Siemens SIMATIC IOT2050 gateway served as the intermediate layer between the industrial network and the cloud services. Using Node-RED, the gateway collected operational data from the PLCs via OPC UA and prepared it for transmission to the selected cloud platforms.

The implementation showed how the same industrial data source could be connected to Siemens Insights Hub, AWS, and Microsoft Azure using a gateway-centred integration strategy. Each platform required specific configuration steps, but the overall architecture preserved a common structure: data generation in the simulated process, PLC-based control and data exposure, edge-level acquisition and processing, and cloud-side ingestion. This structure enabled the implementation of the three integrations within a consistent technical framework.

This chapter establishes the technical foundation required for the following evaluation, without assessing the platforms themselves. Section 5 builds on this implementation by

analysing and comparing the resulting architectures according to the selected evaluation criteria.

5. Comparative Analysis and Results

This section presents the results of deploying the proposed methods, along with a comparative analysis of the selected cloud services. Siemens Insights Hub, AWS IoT SiteWise and Azure IoT Hub can be compared through three key metrics: end-to-end latency, ease of use, and implementation complexity. To avoid subjective judgment, both ease of use and implementation complexity were evaluated based on observable implementation characteristics (e.g., number of required services, configuration steps, and dependencies) usingthe Weighted Scoring Model ,based on by Nouhas et al. [70], but with different criteria and weights. Each criterion was applied equally on all platforms. Given the limitations of the implemented system, it was not possible to capture every metric for every service, as the latency measurements only capture the time from Node-RED to the respective cloud platforms. Data acquisition from the PLC to the IIoT gateway was common across all deployments and, therefore, was not a differentiating factor.

5.1. Ease-of-use

The ease-of-use metric evaluates the overall effort required to configure and interact with the platform. It is based on the following criteria: Configuration steps (Criterion 1 - C1), number of services (C2), integration type (C3) and manual coding requirements (C4). The configuration steps outline the number of actions required to establish and operationalise the platform, from initial configuration through to successful data collection. Depending on the number of steps, the criteria score demonstrated in Table 5.1 can be high or low.

Table 5.1 – Number of steps scores (C1)

No. Steps	Score
>10	1
8-10	2
5-7	3
3-4	4
≤2	5

The number of services (Table 5.2) can be represented as the total number of cloud services that must be configured and connected to achieve the total functionality of the architecture.

Table 5.2 – Number of used services scores for ease-of-use evaluation (C2)

No. Services	Score
>5	1
4-5	2
3	3
2	4
1	5

The integration type (Table 5.3) characterises the mechanism used to connect the industrial system to the cloud platform. It can be distinguished between totally manual API (where the integration is made only by APIs, without any support from the cloud platform), API + Support[2] (which is still is used APIs, but the cloud platform offers some support through already made templates, SDKs or configuration tools, for example), and Native (in which the cloud platform already have direct mechanisms to make the integration, like giving direct support to MQTT communication).

Table 5.3 – Integration type used to connect the cloud service scores (C3)

Integration type	Score
API (totally manual)	1
API + Support	3
Native	5

To analyse the amount of code development necessary during implementation, the manual coding requirement criteria in Table 5.4 are used to describe the quantity of code written manually to work.

Table 5.4 – Quantity of manual code made scores (C4)

Manual Coding	Score
Extensive Code	1
Some Code	3
No Code	5

To score ease of use for each cloud platform, a weighted score was calculated for each criterion, reflecting their relative importance, as shown in the equation below. For manual coding requirements and integration type criteria, higher weights were given, as these factors have a more direct impact on development effort and the technical complexity of these platforms (weights: C1 = 0.15; C2 = 0.20; C3 = 0.30; C4 = 0.35). The number of steps does not necessarily reflect the effort required on these platforms, as they can be simplified through guided interfaces.

$$Ease\ of\ Use = (\mathrm{C1} \times 0.15) + (\mathrm{C2} \times 0.20) + (\mathrm{C3} \times 0.30) + (\mathrm{C4} \times 0.35)$$

[2]There was the possibility of using more than one mechanism to connect the industrial system to the cloud platform. In this case, it was chosen the most demanding mechanism technically.

Table 5.5 shows the per-criterion scores for each cloud platform, along with the classification for each criterion. Table 5.6 shows the per-criterion numeric scores (including the numeric decision of qualitative labels such as Native, API + Support, Some Code, and Extensive Code) and the following weighted total.

Table 5.5 – Ease-of-use results

Platform	No. Steps (C1)	No. Services (C2)	Integration Type (C3)	Manual Coding (C4)
Siemens Insights Hub	4	1	Native	Some Code
AWS IoT SiteWise	9	3	Native	Some Code
Azure Digital Twins	14	6	API + Support	Extensive Code

Table 5.6 – Per-criterion numeric scores and weighted total (Ease-of-use)

Platform	C1	C2	C3	C4	Score
Siemens Insights Hub	4	5	5	3	4.15
AWS IoT SiteWise	2	3	5	3	3.45
Azure Digital Twins	1	1	3	1	1.60

Table 5.6 shows that the Siemens Insights Hub has the highest ease-of-use score (4.15), indicating a lower implementation effort than other platforms. The small number of configuration steps influences this result, the use of the MindConnect Node-RED Agent node, a native integration method, and a low amount of coding to make this architecture work. AWS IoT SiteWise receives a medium score (3.45), showing an increased number of configuration steps, but still benefits from native integration (through an MQTT out node) and a low code quantity. Finally, Azure Digital Twins displays the lowest ease-of-use score (1.6), which may be explained by the higher number of required services, a greater number of configuration steps (since create a Device in IoT Hub to create a data history connection in ADT instance), and the necessity to make an extensive coding through API-based integration in Azure Functions, despite using MQTT to send data to ADT.

5.2.Implementation Complexity

Implementation complexity represents the total technical effort necessary to design, create, and implement the architecture, based on the analysis of established complexity-related criteria. It is determined by the following criteria: Configuration effort (C5), number of services (C6), dependency (C7), and concept (C8) levels. The configuration effort (Table 5.7) describes the effort required to configure the architecture, considering not only the number of steps taken but also the complexity of those steps. A low configuration effort can be applied in scenarios where the system can be set up using simple, guided procedures that require minimal parameter description and little technical knowledge. A medium

configuration effort suggests that some degree of technical understanding is necessary, which includes multiple setup stages and moderate complexity in configuring service interactions. A high configuration effort is defined as situations where the configuration process is difficult to execute, requiring a detailed setup of several interconnected services and higher levels of technical knowledge to ensure correct system performance.

Table 5.7 – Configuration effort scores (C5)

Configuration Effort	Score
Low	1
Medium	3
High	5

The number of services (Table 5.8) also appears in this evaluation, but with different scores: it increases as the quantity of services used grows.

Table 5.8 – Number of used services scores for complexity evaluation (C6)

No. Services	Score
1	1
2	2
3	3
4-5	4
>5	5

The dependency level criterion (Table 5.9) characterises the level of interdependence between system components for the normal functionality. It analyses how much each component depends on the correct operation of other components within the architecture. Higher levels of dependency are associated with architectures where data is sent to multiple interconnected and sequential services; if a failure occurs at a particular stage, the same error immediately affects the following services. On the other hand, lower dependency levels correspond to more freely connected architectures, where components can operate independently despite a failure at one of the stages, minimising the propagation of errors.

Table 5.9 – Dependency level scores (C7)

Dependency Level	Score
Independent	1
Low	2
Moderate	3
High	4
Highly Interdependent	5

The concept level (C7), shown in Table 5.10, describes the technical knowledge required for system implementation, including topics such as distributed architectures, event-driven communication, and domain-specific modelling methodologies. This criterion describes the level of understanding the user should have not only of the platform's configuration.

Table 5.10 – Concept level scores (C8)

Concept Level	Score
Basic	1
Intermediate	3
Advanced	5

To obtain the implementation complexity score, a weighted scoring (as shown in the equation below) was also applied for ease-of-use, with different weights (C5 = 0.25; C6 = 0.15; C7 = 0.30; C8 = 0.30). Higher weights were attributed to dependency and concept levels, as these two factors directly represent architectural interaction and require technical knowledge.

$$\text{Complexity} = (C5 \times 0.25) + (C6 \times 0.15) + (C7 \times 0.30) + (C8 \times 0.30)$$

Table 5.11 summarises the results of this evaluation across the three studied cloud platforms, as Table 5.12 details the per-criterion numeric scores and the resulting weighted total for implementation complexity. It is not difficult to identify clear differences between the solutions, particularly in terms of architectural structure, dependency levels, and required technical knowledge. Siemens Insights Hub has the lowest complexity score (1.65), mainly due to its low configuration effort, low number of services, and low dependency level, as the architecture relies on direct integration with limited communication between components. Also, the platform is at a basic conceptual level, as the implementation does not require deep knowledge of distributed systems or complex architectural concepts. AWS IoT SiteWise has a moderate complexity level (4), mainly due to its higher dependency level, with numerous services integrated in a sequential data flow. This increases the connection between components, as the correct operation of each stage depends on the previous one. Also, the platform is at a higher conceptual level than Insights Hub, as it requires an understanding of industrial data modelling, cloud-based data collection, and system integration procedures. In contrast, Azure Digital Twins has the highest complexity score (4.7), due to its highly interconnected architecture, involving several services that operate in a tightly coupled, sequential manner. This significantly increases the dependency level, as failures or misconfigurations in one component can propagate throughout the system. Moreover, the platform requires a deep conceptual level, as its implementation requires knowledge of advanced concepts such as event-driven architectures and digital twin modelling, which increases overall implementation complexity.

Table 5.11 – Implementation complexity results

Platform	Configuration Effort (C5)	No. Services (C6)	Dependency Level (C7)	Concept Level (C8)
Siemens Insights Hub	Low	1	Low	Basic
AWS IoT SiteWise	Medium	3	High	Advanced

Platform	Configuration Effort (C5)	No. Services (C6)	Dependency Level (C7)	Concept Level (C8)
Azure Digital Twins	High	6	High	Advanced

Table 5.12 – Per-criterion numeric scores and weighted total (Implementation complexity)

Platform	C5	C6	C7	C8	Score
Siemens Insights Hub	1	1	2	1	1.30
AWS IoT SiteWise	3	3	4	5	4.00
Azure Digital Twins	5	5	4	5	4.70

5.3. End-to-End Latency

End-to-end latency tests were conducted on the MQTT brokers of Azure IoT Hub and AWS IoT Core, following the methodology proposed by Gambit Communications Inc. [71], illustrated in Figure 5.1. In this methodology, the MQTT input (*mqtt in* node) and MQTT output (*mqtt out* node) of each cloud service were connected to the same MQTT topic. If the content initially published matched the content received, the message reception time was recorded, enabling the calculation of instantaneous latency. All information obtained from this test may be stored in an exportable Excel file, allowing the creation and analysis of charts.

Siemens Insights Hub was excluded from this analysis because communication between IOT2050 and this cloud platform was not via MQTT. Additionally, this cloud's Start For Free subscription plan was discontinued in May 2025, guaranteeing experimentation under the same conditions.

Two distinct tests were conducted in different days, each lasting approximately 30 minutes, with latency measurements collected every 10 seconds. The first test was performed with the industrial simulation was not running, and the MQTT flows used for normal industrial data transmission to the cloud platforms were disabled. In this scenario, considered the baseline measurement, only the communication associated with the latency test messages and their corresponding subscribers remained active, minimising background traffic and potential interference.

The second test was conducted while the industrial simulation was running, including the data transmission to the cloud platforms via MQTT. This allowed the evaluation of latency under real operating conditions, and the network traffic and processing loads could affect performance.

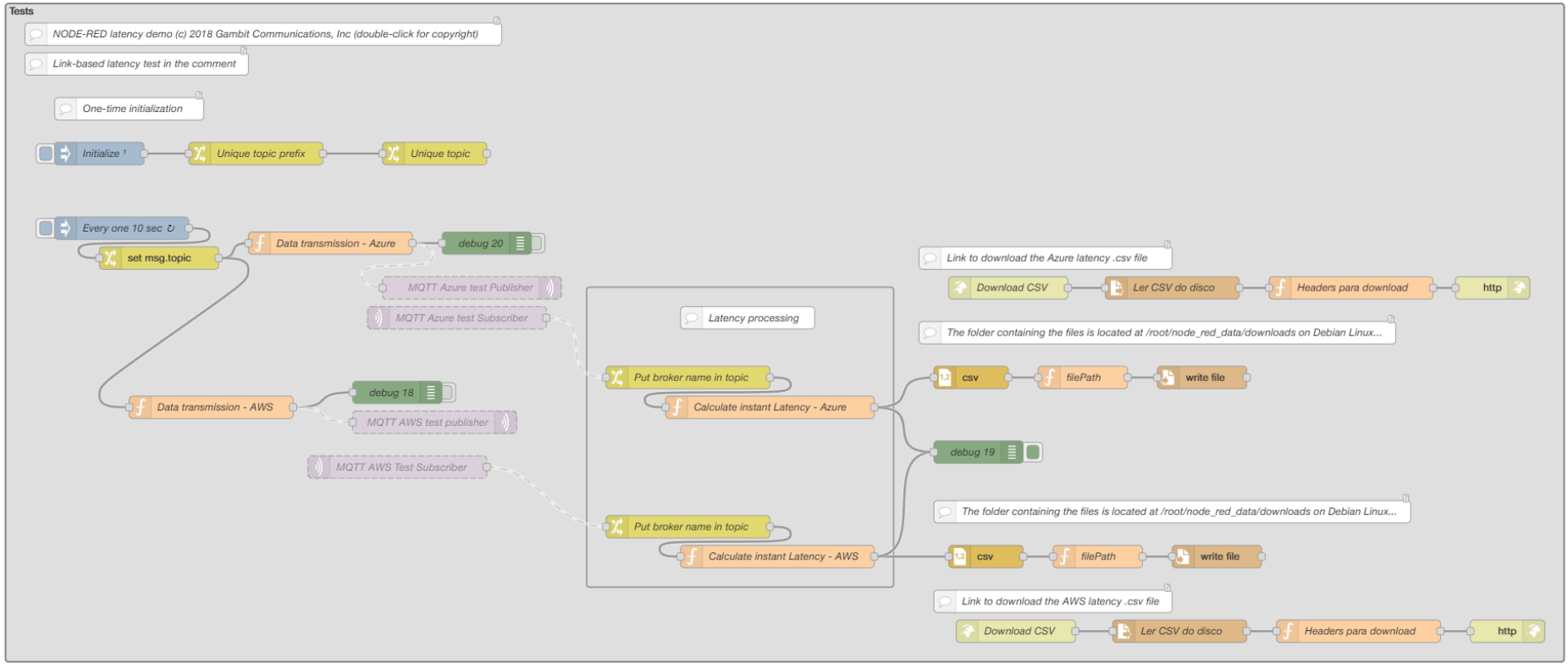

Figure 5.1 – Node-RED flow used to test the MQTT broker latencies

Measuring nominal latency alone is not enough to make conclusions of the communication between devices and clouds. It is necessary to use tail latency metrics to detect the worst case performance (or outliers) that can have a serious impact on real-time applications [72]. The tail latency percentiles calculated in this tests were p50 (measures the typical case or median, where 50% of the latency measurements are below this value), p95 (95% of the latency measurements are below this value, and helps understand how the architetures handles with worst-performing scenarios), and, finally p99 (which 99% of the measurements are below this value and reveals the rare but critical events such as outliers, peaks and extreme delays) [73]. Analysing the first test (with the industrial automation simulation off and without data being transferred to the cloud) it is clear that the Azure curve remains consistently above the AWS curve. Also, there is a very high latency outlier near to 18 s, as can be seen in Figure 5.2.

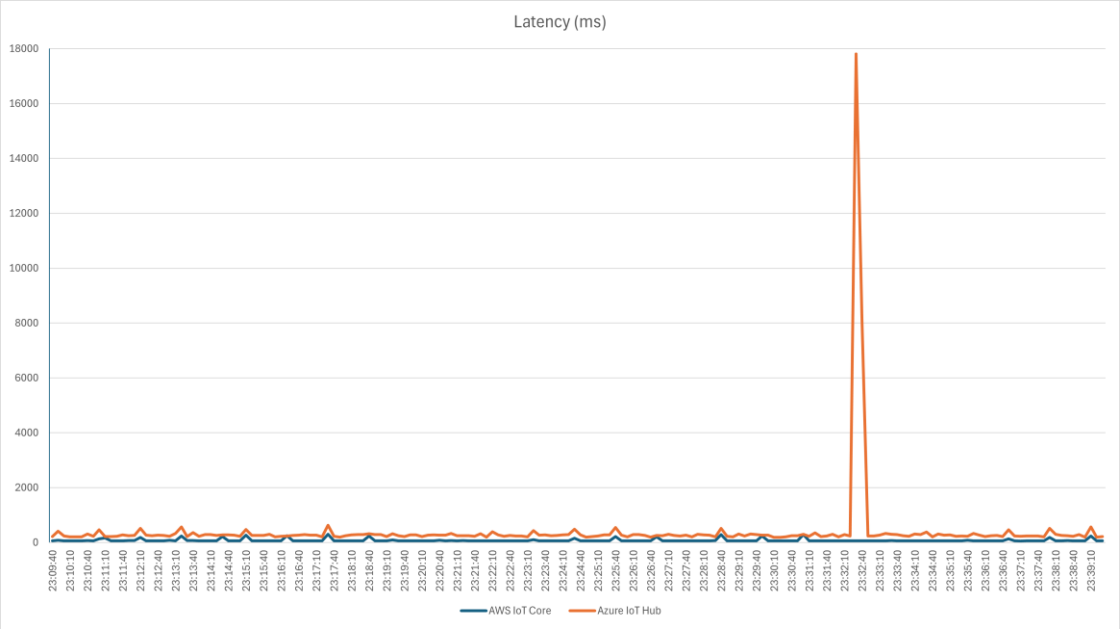

Figure 5.2 – Nominal latency before simulation: AWS IoT Core vs Azure IoT Hub

Without considering this outlier, Azure IoT Hub nominal latency values range approximately from 195 ms to 620 ms, while AWS IoT Core ranges from about 57 ms to 300 ms, as shown in Figure 5.3. The percentile analysis for Azure shows a median latency (p50) of 260 ms and e p95 is around 490 ms, indicating that most communications occur within roughly half a second. However, the p99 reaches approximately 2 s, showing the presence of rare but penalising cases that degrade performance in the worst 1% of cases. In the other hand, AWS IoT Core reveals lower percentiles: p50 of 62 ms, p95 of 215.70 ms,

and p99 of 289.72 ms. These values indicate overall faster performance and greater consistency, maintaining low latencies even in the tail of the distribution.

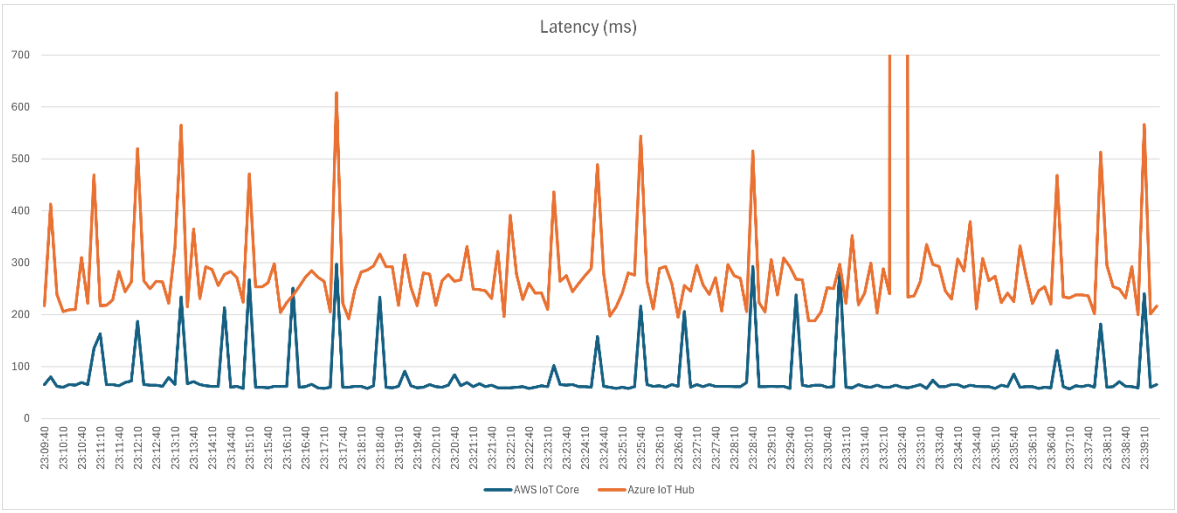

Figure 5.3 – Nominal latency before simulation: AWS IoT Core vs Azure IoT Hub (zoomed)

In the second test, with the industrial automation simulation active and continuous data transmission to the cloud, increased variability and outliers become evident for both platforms (Figure 5.4). Extreme peaks are observed, reaching values on the order of 38 s to 40 s, along with intermediate peaks between nearly 1 s and 3 s. This pattern suggests that under higher load and increased processing concurrency, the system becomes more susceptible to sporadic delays, potentially associated with queueing effects, resource contention, or network fluctuations.

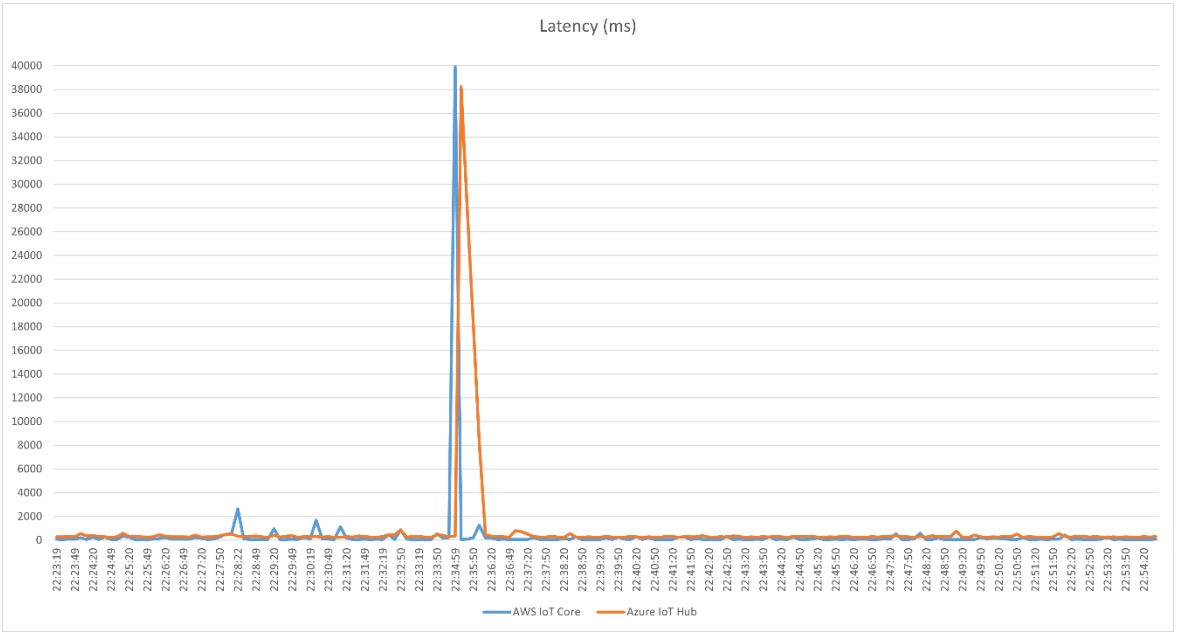

Figure 5.4 – Nominal latency during simulation: AWS IoT Core vs Azure IoT Hub

Figure 5.5 shows that by excluding outliers, the nominal latency values range from 57 to 970 ms for AWS IoT Core and from 200 ms to 860 ms for Azure IoT Hub. The percentiles obtained for AWS IoT Core are 71 ms for p50, 524 ms for p95, and 1865.46 ms for p99.

This suggests that the platform typically has a low latency and that, most of the time, it has a latency of around half a second. However, the p99 is already in the order of seconds, indicating that there are enough outliers to make the tail significant in 1% of the measurements. In the case of Azure IoT Hub, the p50 is 294 ms, the p95 is 552.6 ms and the p99 is 19,453.04 ms. The median is substantially higher than the AWS values, indicating a slower typical performance. Although the p95 is close to that of AWS, Azure's p99 is much higher. This result points to very heavy tail latency: the worst 1% of cases are extremely poor and dominated by large-magnitude outliers, something that is visible in the second graph, where one or a few isolated events stretch the vertical scale and highlight very large delays.

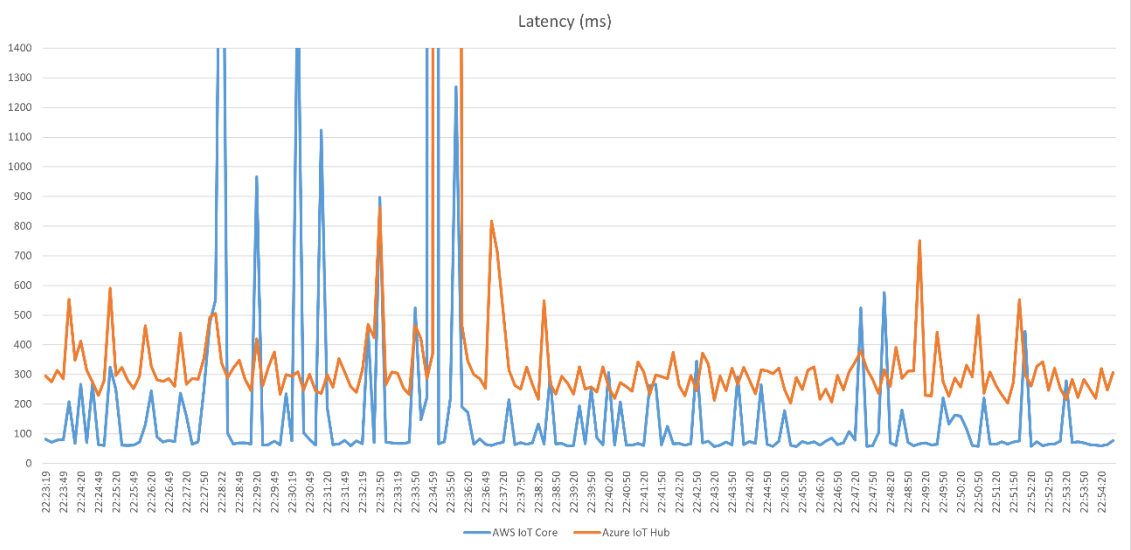

Figure 5.5 – Nominal latency during simulation: AWS IoT Core vs Azure IoT Hub (zoomed)

When comparing both tests, there is a notable increase in latency for both cloud platforms when the industrial simulation and data transmission to the cloud are in the process of running, with AWS showing greater variability than in the previous test. Although, it continues to demonstrate generally lower latency values and greater stability in the lower and middle percentiles. On the other hand, Azure maintains higher typical latencies and shows a significant deterioration at the tail of the latency distribution, with a very significant increase in the p99, which indicates the occurrence of more frequent and penalizing extreme delays. In summary, under higher load conditions, both platforms suffer performance degradation, but AWS demonstrates greater robustness in its overall behaviour, whilst Azure shows greater susceptibility to severe outliers that negatively impact worst-case communication scenarios. As a result of this work, a GitHub repository[3] was created that contains the developed program and implementation tutorials to facilitate its use in projects where Industry 4.0 integration methodologies may be necessary. In addition, an extended state-of-the-art research was submitted for publication.

[3] https://github.com/ju9da9/IISwCBP

6. Conclusions

This dissertation demonstrated that a PLC-Edge-Cloud architecture can provide a practical and reproducible approach for integrating industrial automation systems with cloud-based platforms. The results show that the choice of platform strongly influences implementation effort, modelling flexibility, architectural complexity, and communication performance. Siemens Insights Hub proved to be the most straightforward platform to implement, and AWS IoT SiteWise offered a balanced solution between configuration effort and industrial data structuring. At the same time, Azure Digital Twins provided the most advanced modelling capabilities but required greater configuration effort and service orchestration.

The latency evaluation revealed that AWS IoT Core demonstrated lower and more stable end-to-end latency than Azure IoT Hub under the tested contexts, especially in median and high-percentile measurements. Under increased load, both platforms revealed increased latency and variability. However, Azure IoT Hub exhibited heavier tail behaviour, with notable outliers affecting the worst-case performance (p99). On the other hand, AWS IoT Core maintained better robustness, despite some degradation. However, these results should be interpreted within the scope of the implemented case study, since factors such as network conditions, workload scale, geographic region, security configuration, and pricing models were not exhaustively analysed.

Overall, this work provides a practical implementation framework, a comparative analysis of cloud-based IIoT platforms, and guidance on selecting appropriate technologies for Industry 4.0 scenarios. In addition, an extended state-of-the-art manuscript titled “Cloud-Based Industrial IoT Platforms: A Systematic Review of Architectures, Protocols, and Case Studies” was submitted to WIREs Data Mining and Knowledge Discovery, with revision currently pending.

As future work, the proposed architecture should be extended by evaluating additional cloud platforms and industrial communication protocols, including Google Cloud, IBM Maximo, MQTT Sparkplug B, and advanced OPC UA PubSub configurations. This would provide a broader comparison of interoperability, configuration effort, and industrial applicability.

Further testing should also be performed in larger-scale, longer-duration scenarios involving multiple PLCs, higher message frequencies, different network conditions, and geographically distributed cloud regions. This would allow a more complete assessment of scalability, reliability, latency stability, and system robustness.

Another important direction is the inclusion of cybersecurity and cost analysis. Future studies should evaluate authentication mechanisms, certificate management, access control, data encryption, cloud pricing models, and operational costs, since these factors are critical for real industrial deployment.

Finally, the architecture could be extended with advanced analytics, anomaly detection, predictive maintenance models, and dashboard-based decision support. This would move the system beyond data acquisition and monitoring toward intelligent industrial optimisation.

Bibliography

[1] Y. Lu, 'Industry 4.0: A survey on technologies, applications and open research issues', *Journal of Industrial Information Integration*, vol. 6, pp. 1–10, Jun. 2017, doi: 10.1016/j.jii.2017.04.005.

[2] M. Aazam and E.-N. Huh, 'Fog Computing and Smart Gateway Based Communication for Cloud of Things', in *2014 International Conference on Future Internet of Things and Cloud*, Barcelona, Spain: IEEE, Aug. 2014, pp. 464–470. doi: 10.1109/FiCloud.2014.83.

[3] W. Z. Khan, M. H. Rehman, H. M. Zangoti, M. K. Afzal, N. Armi, and K. Salah, 'Industrial internet of things: Recent advances, enabling technologies and open challenges', *Computers & Electrical Engineering*, vol. 81, p. 106522, Jan. 2020, doi: 10.1016/j.compeleceng.2019.106522.

[4] A. Ioana, C. Burlacu, and A. Korodi, 'Approaching OPC UA Publish–Subscribe in the Context of UDP-Based Multi-Channel Communication and Image Transmission', *Sensors*, vol. 21, no. 4, p. 1296, Feb. 2021, doi: 10.3390/s21041296.

[5] C. Bayılmış, M. A. Ebleme, Ü. Çavuşoğlu, K. Küçük, and A. Sevin, 'A survey on communication protocols and performance evaluations for Internet of Things', *Digital Communications and Networks*, vol. 8, no. 6, pp. 1094–1104, Dec. 2022, doi: 10.1016/j.dcan.2022.03.013.

[6] B. Mishra and A. Kertesz, 'The Use of MQTT in M2M and IoT Systems: A Survey', *IEEE Access*, vol. 8, pp. 201071–201086, 2020, doi: 10.1109/ACCESS.2020.3035849.

[7] Y. Lu, C. Liu, K. I.-K. Wang, H. Huang, and X. Xu, 'Digital Twin-driven smart manufacturing: Connotation, reference model, applications and research issues', *Robotics and Computer-Integrated Manufacturing*, vol. 61, p. 101837, Feb. 2020, doi: 10.1016/j.rcim.2019.101837.

[8] C. Neves, 'Inteligent Automation & Industry 4.0 - Basic Principles', Polytechnic of Leiria, 2022.

[9] N. Y. Mulongo, 'Industry 5.0 a Novel Technological Concept', in *2024 International Conference on Smart Applications, Communications and Networking (SmartNets)*, May 2024, pp. 1–6. doi: 10.1109/SmartNets61466.2024.10577684.

[10] 'Industry 4.0 – AllAboutLean.com'. Accessed: Dec. 07, 2025. [Online]. Available: https://www.allaboutlean.com/industry-4-0/industry-4-0-2/

[11] 'Industry 4.0', FOSTEC & Company. Accessed: Sep. 15, 2025. [Online]. Available: https://www.fostec.com/en/competences/industry-4-0/

[12] W. Z. Khan, M. H. Rehman, H. M. Zangoti, M. K. Afzal, N. Armi, and K. Salah, 'Industrial internet of things: Recent advances, enabling technologies and open challenges', *Computers & Electrical Engineering*, vol. 81, p. 106522, Jan. 2020, doi: 10.1016/j.compeleceng.2019.106522.

[13] 'What Is the Industrial Internet of Things (IIoT)? - RealPars'. Accessed: Sep. 19, 2025. [Online]. Available: https://www.realpars.com/blog/iiot

[14] M. Lavi, 'Azure IoT - The Complete Guide', Udemy. Accessed: Dec. 16, 2025. [Online]. Available: https://www.udemy.com/course/az-220-microsoft-azure-iot-developer-certification-2022/

[15] J. Hurwitz, R. Bloor, M. Kaufman, and Fern Halper, *Cloud Computing for Dummies*. Wiley Publishing, Inc., 2010. Accessed: Dec. 18, 2025. [Online]. Available: https://repository.unikom.ac.id/61330/1/Cloud%20Computing%20For%20Dummies.pdf

[16] 'What is IaaS? - Infrastructure as a Service Explained - AWS', Amazon Web Services, Inc. Accessed: Dec. 18, 2025. [Online]. Available: https://aws.amazon.com/what-is/iaas/
[17] 'Types of cloud computing', Amazon Web Services, Inc. Accessed: Dec. 18, 2025. [Online]. Available: https://aws.amazon.com/types-of-cloud-computing/
[18] 'What is SaaS? - Software as a Service Explained - AWS', Amazon Web Services, Inc. Accessed: Dec. 18, 2025. [Online]. Available: https://aws.amazon.com/what-is/saas/
[19] 'What is the cloud? | Cloud definition - Cloudflare'. Accessed: Dec. 18, 2025. [Online]. Available: https://www.cloudflare.com/learning/cloud/what-is-the-cloud/
[20] M. S. Aslanpour, S. S. Gill, and A. N. Toosi, 'Performance evaluation metrics for cloud, fog and edge computing: A review, taxonomy, benchmarks and standards for future research', *Internet of Things*, vol. 12, p. 100273, Dec. 2020, doi: 10.1016/j.iot.2020.100273.
[21] D. Moher *et al.*, 'Preferred reporting items for systematic review and meta-analysis protocols (PRISMA-P) 2015 statement', *Systematic reviews*, vol. 4, pp. 1–9, 2015.
[22] G. A. Nagendran, R. J. S. Raj, C. Sharon Roji Priya, and H. Singh, 'IoT Cloud Systems: A Survey', in *2023 5th International Conference on Smart Systems and Inventive Technology (ICSSIT)*, Tirunelveli, India: IEEE, Jan. 2023, pp. 415–418. doi: 10.1109/ICSSIT55814.2023.10060983.
[23] R. Sikarwar, P. Yadav, and A. Dubey, 'A Survey on IOT enabled cloud platforms', in *2020 IEEE 9th International Conference on Communication Systems and Network Technologies (CSNT)*, Gwalior, India: IEEE, Apr. 2020, pp. 120–124. doi: 10.1109/CSNT48778.2020.9115735.
[24] T. G. F. Barros, E. F. Da Silva Neto, J. A. D. S. Neto, A. G. M. De Souza, V. B. Aquino, and E. S. Teixeira, 'The Anatomy of IoT Platforms—A Systematic Multivocal Mapping Study', *IEEE Access*, vol. 10, pp. 72758–72772, 2022, doi: 10.1109/ACCESS.2022.3189660.
[25] D. A. Morelli and P. S. D. A. Ignacio, 'Assessment of researches and case studies on Cloud Manufacturing: a bibliometric analysis', *Int J Adv Manuf Technol*, vol. 117, no. 3–4, pp. 691–705, Nov. 2021, doi: 10.1007/s00170-021-07782-0.
[26] P. Vinayagam, K. S, S. A. A, and M. A. Sithik, 'Delving into IoT-Based Smart Meters for Industrial Energy Optimization', in *2025 International Conference on Advancements in Smart, Secure and Intelligent Computing (ASSIC)*, May 2025, pp. 1–6. doi: 10.1109/ASSIC64892.2025.11158522.
[27] T. Thongtawee, T. Chuchit, T. Srirugsa, S. Kaewpoung, W. Su-hren, and A. Isaramongkolrak, 'Enhancing Energy Efficiency in Pig Farms with Grid-Tie Solar Systems and IIoT-Based Monitoring', in *2025 Seventh International Symposium on Computer, Consumer and Control (IS3C)*, Jun. 2025, pp. 1–5. doi: 10.1109/IS3C65361.2025.11131009.
[28] M. Madinov, A. Topalov, O. Klymenko, V. Dyvak, D. Kropyvnytskyi, and T. Humeniuk, 'Bluemix IIoT System for Monitoring Production Equipment', in *2025 IEEE 13th International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS)*, Sep. 2025, pp. 1335–1340. doi: 10.1109/IDAACS68557.2025.11322007.
[29] C. C. Córdova, C. Vasquez, J. Morillo, and R. Sarango, 'Integration of Heterogeneous Devices through an Industrial IoT Architecture for Data Centralization and Cloud Processing in Industry 4.0 Applications', in *2025 5th International Conference on Electrical, Computer, Communications and Mechatronics Engineering (ICECCME)*, Oct. 2025, pp. 1–6. doi: 10.1109/ICECCME64568.2025.11277735.

[30] K. K. Patro, M. Deepak, R. Dilleswararao, M. J. Rao, M. Kshyama, and E. Sravani, 'Smart IoT-Based on Real-Time Monitoring System for Overhead Tank Water Levels and Quality', in *2025 IEEE 4th World Conference on Applied Intelligence and Computing (AIC)*, Jul. 2025, pp. 975–979. doi: 10.1109/AIC66080.2025.11212028.

[31] S. Prasad.S.J, C. M. Iniyan, E. M. Barath, GM. Thangatamilan, R. Gayathri, and M. Apoorva, 'Tank Level Control Using Lora and PLC', in *2025 Second International Conference on Intelligent Technologies for Sustainable Electric and Communications Systems (iTech SECOM)*, Oct. 2025, pp. 1–6. doi: 10.1109/iTechSECOM64750.2025.11307668.

[32] M. M. Nabi and A. Kharaz, 'IoT Driven Predictive Maintenance for Enhanced Reliability of Blowers in Wastewater Treatment', in *2025 11th International Conference on Control, Decision and Information Technologies (CoDIT)*, Jul. 2025, pp. 870–875. doi: 10.1109/CoDIT66093.2025.11321348.

[33] 'Factory I/O - Next-Gen PLC Training'. Accessed: Apr. 29, 2026. [Online]. Available: https://factoryio.com/

[34] 'Azure IoT - The Complete Guide', Udemy. Accessed: Mar. 07, 2026. [Online]. Available: https://www.udemy.com/course/az-220-microsoft-azure-iot-developer-certification-2022/

[35] 'Learn MQTT With Node-RED', Product Detail Page for Learn MQTT With Node-RED. Accessed: Mar. 07, 2026. [Online]. Available: https://learn.codeandcompile.com/learn-mqtt-with-node-red

[36] *SIMATICmeetsLinux/IOT2050-Setting-Up-Example-Image*. (Jan. 27, 2026). SIMATICmeetsLinux. Accessed: Apr. 03, 2026. [Online]. Available: https://github.com/SIMATICmeetsLinux/IOT2050-Setting-Up-Example-Image

[37] 'node-red-contrib-opcua'. Accessed: Apr. 08, 2026. [Online]. Available: http://flows.nodered.org/node/node-red-contrib-opcua

[38] '@flowfuse/node-red-dashboard'. Accessed: Apr. 08, 2026. [Online]. Available: http://flows.nodered.org/node/@flowfuse/node-red-dashboard

[39] 'Insights Hub industrial IoT | Siemens Software', Siemens Digital Industries Software. Accessed: Apr. 07, 2026. [Online]. Available: https://www.siemens.com/en-us/products/insights-hub/

[40] Siemens Software, *Insights Hub Basics - Data flow from on-site to applications*, (Apr. 03, 2025). Accessed: Apr. 07, 2026. [Online Video]. Available: https://www.youtube.com/watch?v=5FrEQinqNo8

[41] Lara Simons - Siemens, 'Workshop Hands-On Mindsphere', 2023.

[42] Siemens AG Digital Industries, 'MindSphere - Start for Free System Manual'. Siemens AG, Feb. 2022.

[43] Siemens Software, *Insights Hub Basics - Data Model*, (Oct. 08, 2025). Accessed: Apr. 07, 2026. [Online Video]. Available: https://www.youtube.com/watch?v=zQF8lggk9K0

[44] 'Connectivity - Insights Hub Documentation'. Accessed: Apr. 07, 2026. [Online]. Available: https://documentation.mindsphere.io/MindSphere/connectivity/overview.html

[45] '@mindconnect/node-red-contrib-mindconnect'. Accessed: Apr. 08, 2026. [Online]. Available: http://flows.nodered.org/node/@mindconnect/node-red-contrib-mindconnect

[46] 'MindConnect Node-RED Node Overview - developer.siemens.com'. Accessed: Apr. 08, 2026. [Online]. Available: https://developer.siemens.com/industrial-iot-open-source/node-red-contrib-mindconnect/index.html

[47] 'O que é AWS IoT SiteWise Monitor? - AWS IoT SiteWise Monitor'. Accessed: Feb. 26, 2026. [Online]. Available: https://docs.aws.amazon.com/pt_br/iot-sitewise/latest/appguide/what-is-monitor-app.html
[48] 'IoT SiteWise | eu-west-1'. Accessed: Mar. 06, 2026. [Online]. Available: https://eu-west-1.console.aws.amazon.com/iotsitewise/home?region=eu-west-1#
[49] 'Model industrial assets - AWS IoT SiteWise'. Accessed: Mar. 02, 2026. [Online]. Available: https://docs.aws.amazon.com/iot-sitewise/latest/userguide/industrial-asset-models.html
[50] 'Define data properties - AWS IoT SiteWise'. Accessed: Mar. 02, 2026. [Online]. Available: https://docs.aws.amazon.com/iot-sitewise/latest/userguide/asset-properties.html
[51] 'Ingest data to AWS IoT SiteWise - AWS IoT SiteWise'. Accessed: Jan. 16, 2026. [Online]. Available: https://docs.aws.amazon.com/iot-sitewise/latest/userguide/industrial-data-ingestion.html
[52] 'Managing devices with AWS IoT - AWS IoT Core'. Accessed: Feb. 17, 2026. [Online]. Available: https://docs.aws.amazon.com/iot/latest/developerguide/iot-thing-management.html
[53] 'AWS IoT Core policy actions - AWS IoT Core'. Accessed: Mar. 21, 2026. [Online]. Available: https://docs.aws.amazon.com/iot/latest/developerguide/iot-policy-actions.html
[54] 'Rules for AWS IoT - AWS IoT Core'. Accessed: Feb. 25, 2026. [Online]. Available: https://docs.aws.amazon.com/iot/latest/developerguide/iot-rules.html
[55] 'IAM Roles', GeeksforGeeks. Accessed: Feb. 25, 2026. [Online]. Available: https://www.geeksforgeeks.org/devops/iam-roles/
[56] 'BatchPutAssetPropertyValue - AWS IoT SiteWise'. Accessed: Jan. 15, 2026. [Online]. Available: https://docs.aws.amazon.com/iot-sitewise/latest/APIReference/API_BatchPutAssetPropertyValue.html
[57] baanders, 'What is Azure Digital Twins? - Azure Digital Twins'. Accessed: Mar. 02, 2026. [Online]. Available: https://learn.microsoft.com/en-us/azure/digital-twins/overview
[58] baanders, 'DTDL models - Azure Digital Twins'. Accessed: Jan. 15, 2026. [Online]. Available: https://learn.microsoft.com/en-us/azure/digital-twins/concepts-models
[59] baanders, 'Quickstart - Get started with Azure Digital Twins Explorer - Azure Digital Twins'. Accessed: Mar. 02, 2026. [Online]. Available: https://learn.microsoft.com/en-us/azure/digital-twins/quickstart-azure-digital-twins-explorer
[60] 'node-red-contrib-azure-iot-hub'. Accessed: Jan. 15, 2026. [Online]. Available: http://flows.nodered.org/node/node-red-contrib-azure-iot-hub
[61] N. Kinkar, 'Connecting Node-Red to Azure IoT Hub using MQTT nodes.', Medium. Accessed: Dec. 28, 2025. [Online]. Available: https://medium.com/@nikhilkinkar/connecting-node-red-to-azure-iot-hub-using-mqtt-nodes-6e9160549348
[62] 'Azure for Students | Microsoft Azure'. Accessed: Mar. 19, 2026. [Online]. Available: https://azure.microsoft.com/en-us/free/students
[63] V. Doshi, 'Digital twins for renewables with a focus on solar power', Universitat Politècnica de Catalunya, 2024. Accessed: Oct. 28, 2025. [Online]. Available: https://hdl.handle.net/2117/421315
[64] baanders, 'Ingest telemetry from IoT Hub - Azure Digital Twins'. Accessed: Mar. 02, 2026. [Online]. Available: https://learn.microsoft.com/en-us/azure/digital-twins/how-to-ingest-iot-hub-data

[65] baanders, 'Data history (with Azure Data Explorer) - Azure Digital Twins'. Accessed: Mar. 05, 2026. [Online]. Available: https://learn.microsoft.com/en-us/azure/digital-twins/concepts-data-history

[66] baanders, 'Create a data history connection - Azure Digital Twins'. Accessed: Mar. 18, 2026. [Online]. Available: https://learn.microsoft.com/en-us/azure/digital-twins/how-to-create-data-history-connection

[67] spelluru, 'Create Azure Event Hubs instance Using Portal - Azure Event Hubs'. Accessed: Mar. 18, 2026. [Online]. Available: https://learn.microsoft.com/en-us/azure/event-hubs/event-hubs-create

[68] spelluru, 'Quickstart- Create an Azure Data Explorer cluster and database - Azure Data Explorer'. Accessed: Mar. 19, 2026. [Online]. Available: https://learn.microsoft.com/en-us/azure/data-explorer/create-cluster-and-database

[69] 'Creating Dashboards with Azure Digital Twins, Azure Data Explorer, and Grafana | Microsoft Community Hub', TECHCOMMUNITY.MICROSOFT.COM. Accessed: Mar. 02, 2026. [Online]. Available: https://techcommunity.microsoft.com/blog/iotblog/creating-dashboards-with-azure-digital-twins-azure-data-explorer-and-grafana/3277879

[70] H. Nouhas, A. Belangour, and M. Nassar, 'A Weighted Scoring Model Analysis of Cloud Storage Services: Comparing AWS, Azure, and Google Cloud Platform', *International Journal of Engineering Trends and Technology - IJETT*, vol. 73, 2025, doi: 10.14445/22315381/IJETT-V73I6P129.

[71] G. C. Inc, *gambitcomminc/nodered-mqtt-latency*. (Nov. 16, 2018). Accessed: Apr. 13, 2026. [Online]. Available: https://github.com/gambitcomminc/nodered-mqtt-latency

[72] 'Tail Latency in Distributed Systems Explained', simplyblock. Accessed: Apr. 30, 2026. [Online]. Available: https://simplyblock.io/glossary/what-is-tail-latency/

[73] N. Desai, 'Latency and why it's matters', Medium. Accessed: Apr. 30, 2026. [Online]. Available: https://ninad-desai.medium.com/latency-and-why-its-matters-9eeabbe6af20