SN Computer Science (2024) 5:802
https://doi.org/10.1007/s42979-024-03124-5

ORIGINAL RESEARCH

Reconfigurable Framework for Data Extraction Using Interoperable Brokers in Manufacturing

Nelson Freitas[1] · Andre Dionisio Rocha[1] · Fábio M-Oliveira[1] · Duarte Alemão[1] · José Barata[1]

Received: 11 October 2023 / Accepted: 9 July 2024

Abstract
Technology is an integral part of society and has undergone significant evolution across various domains, such as production and recreation, leading to the emergence of heterogeneous systems. These diverse systems often need to communicate and interact with one another to fully leverage their capabilities and resources, aiming for maximum efficiency. One effective method to achieve this is through the use of a message broker, a tool that facilitates communication between multiple heterogeneous systems. However, setting up message brokers can be complex, requiring access to multiple systems for configuration and lacking automation tools to streamline the process. The proposed solution involves creating a tool that can be instantiated on different machines to control the deployment, configuration, and usage of any message broker. Early results are promising, demonstrating enhanced data collection from industrial robots and improved connectivity between different message brokers.

Keywords Cyber-physical System · Data extraction · Interoperability · Kafka · Message broker · MQTT · Reconfigurable systems

Introduction

In today's world, a multitude of devices, being sensors, machines, or computers, continuously connect to a network, transmitting various resources and interacting with the real world based on the information of their instructions. The fusion of the physical and cyber worlds enables real-time reading and acting upon the physical aspects utilizing the computing and communication power of the cyber aspects. A system capable of using both aspects was given the name of Cyber-Physical Systems (CPS) [1, 2].

However, for some complex infrastructures (e.g. having several interconnected physical and cyber components with different capabilities and heterogeneous information flow and data management systems), a single CPS is insufficient, and a network of CPS is required. This can be true for smart buildings, industry-complex systems, smart cities, among others, where increasingly technological systems are deployed and frequently need to communicate with one another. Being a software-intensive system, the CPS presents dynamic properties that can bring new challenges or exacerbate already existing ones, such as security, safety, and reliability [3].

A common method of communication between several heterogeneous software systems that are often present in CPS are message brokers, as they provide fundamental characteristics such as scalability, easy communication between heterogeneous devices, and availability, just to name a few [4]. However, different message brokers often have different characteristics, and some are more suited for certain tasks than others (such as latency, maximum size of message allowed, security protocols, or built-in gateways) [5, 6].

✉ Nelson Freitas
n.freitas@uninova.pt

Andre Dionisio Rocha
andre.rocha@uninova.pt

Fábio M-Oliveira
fmo@uninova.pt

Duarte Alemão
d.alemao@uninova.pt

José Barata
jab@uninova.pt

[1] NOVA School of Science and Technology, Center of Technology and Systems (UNINOVA-CTS),and Associated Lab of Intelligent Systems (LASI), NOVA University Lisbon, Lisbon 2829-516, Portugal

Published online: 16 August 2024

As a result, the subsequent research question arises: "How to create an environment capable of deploying and reconfiguring message brokers for data extraction in an Industrial environment?"

The hypotheses for addressing the research question arise as follows: By employing multiple message brokers with distinct characteristics that meet the system's requirements, it is feasible to establish a framework capable of deploying and reconfiguring the message brokers within the system.

The proposed solution, therefore, creates a framework capable of handling the concerns of the CPS through the deployment, reconfiguration, and connection of any kind of message broker capable of communicating with the tools. For the implementation of the framework, a Java programming tool comprised of clients and a server, where the clients are deployed on the same machine as the message broker and the master can be deployed on any computer as long as the communication between the master and the clients exists.

The paper is structured into the following sections: Section two where the related work is introduced, giving top-down approach from a brief view of the Industry 4.0, communication of CPS, data collection, and data extraction. Section three where the framework is presented accompanied by the respective ontology, and the engineering process of the framework. Section four where the tool developed is explained as well as the demonstration case studies. The results are presented and discussed in section five. Finally, section six contains a conclusion as well as future work.

Related Work

Data-Driven Manufacturing in the Industry 4.0

In the era of Industry 4.0, a technological revolution is reshaping industries. This revolution has the main objective of satisfying the market necessities of highly customized products without compromising mass production efficiency and productivity. This puts emphasis on the hard, monotonous, and continuous work to be done by automatic interconnected machines capable of flexibly changing to accommodate customization and allow the workers to have more humane tasks, such as supervision, problem-solving, abstraction, and managing complexity [7].

The previous industrial revolutions always had a type of "fuel" that was present and allowed the processes to develop more efficiently. The first and second industrial revolutions had steam and electricity, respectively, as new types of fuel to push industrial processes forward. The third industrial revolution was a little different, as it had the digitalization of analog reads and was also marked by the usage of the first PLCs. In the fourth industrial revolution, digitalization took a step forward, allowing all the users, products, machines, and new smart sensors to send data and be interconnected, as every system has the possibility to benefit from the data of the whole process and even benefit from integration between different factories. Data became the fuel of this new industrial revolution, and this level of seamless interconnection as the collecting of sufficient information about the process allowed the creation of a cyber map of the physical processes, and with it, processes, simulations, tests, and decisions among a multitude of actions are possible in the cyber world [8].

Industry 4.0 can also be seen as a collection of concepts and technologies across several fields, from communications to computer science. Concepts such as Internet of Things (IoT), cloud computing, mobile internet, and artificial intelligence bring new insights into the industrial system as well as data generation and processing capabilities [9]. As new IoT systems start to flow into the industrial environment, a collection of ever-growing data about every single process, machine, system, product, and worker floods systems. This data is often necessary to allow systems such as CPS to flourish to their full potential, but it can bring an overwhelming amount of information to systems that were not designed to deal with it.

Cyber-Physical Systems

As the IoT grows in popularity, the CPS is the natural follower as it allows the integration of different physical (such as sensing and actuation) and cyber processes (such as processing and communication). In such a case, a method of communication is required so that IoT devices can communicate with cyberspace [10]. As a result, a CPS can be described by a set of characteristics that allow it to operate and abide by its definition. This set of characteristics is composed of [11, 12]:

- **Autonomy** – CPS are capable of learning on their own and adjusting to their surroundings, and it is the autonomy of the system that makes this possible. The capacity of the system to recover from failures or to adapt to given conditions are all examples of the autonomy of a given system and, consequently, of CPS.
- **Decentralization** - The CPS should be a self-contained system that works with autonomy and independence from all the other processes.
- **Heterogeneity** – CPS should be able to integrate multiple systems as well as meet standards for communication and information exchange. This may include different devices, such as robots, sensors, actuators, production cells, among others.

- **Integration** – This is a fundamental component of the CPS, as it by definition, allows the integration between the physical and the cyber worlds, expanding the capabilities of both.
- **Interconnection** – CPS is a synthesis of physical and cyber elements linked by wired and wireless networks with the objective of constructing intelligence. This makes the interconnection of these different elements essential to achieve the goal.
- **Interoperability** – Interoperability consists of the connection, operation, and communication between the different devices of the CPS, allowing the exchange of relevant data between all the components.
- **Modularity** – CPS are composed of modules that collaborate to adapt to changes, be they in consumer needs, product specifications, or supply chain issues, among other things.

The CPS can, therefore, be divided into three layers that together provide the characteristics mentioned above. These layers are: the physical layer, the network layer, and the application layer, which ultimately comprise the physical and cyber space. Figure 1 shows the architecture of a CPS making division between the cyber and physical space as well as each layer providing some examples of each one.

The physical layer is responsible for all the physical infrastructures as well as all the IoT and machinery, such as sensors and actuators. The network layer is responsible for communication and can have physical elements such as routers and gateways and cyber elements such as communication protocols and technologies. Lastly, the application layer is where all the intelligent systems, such as scheduling software, databases, and monitoring software are deployed.

To be able to satisfy all the previously mentioned characteristics, a CPS needs to be able to communicate between all their different devices and applications. There are numerous communication methods that can be used for communicating between heterogeneous devices over the Intranet and Internet. This can be done with the ICE (Internet Communication Engine), REST, AMQP, MQTT, JMS, among others [10, 13].

MQTT, as Message Queue Telemetry Transport, is a publish-subscribe protocol able to receive and send data between multiple publishers and subscribers. The MQTT protocol, created by IBM, quickly gained popularity as a many-to-many communication tool, where the necessity for quick and distributed communication is an essential component. As a result, systems that heavily rely on IoT can use MQTT as a useful tool for enabling them, either alone or in conjunction with other technologies, making it ideal for numerous CPS that work with IoT devices [14]. However, the usage of this protocol should not be used exclusively for IoT systems as work was made with agents regarding the usage of this technology for communication [15, 16]. The message broker is also, often chosen for communication in fog computing due to its load balance and communication characteristics [17, 18] and it can also be integrated with other important communication protocols, such OPC UA [19]. In [20] the authors also use the MQTT message broker approach to handle the security of the transmitted data, making it resilient to cyberattacks.

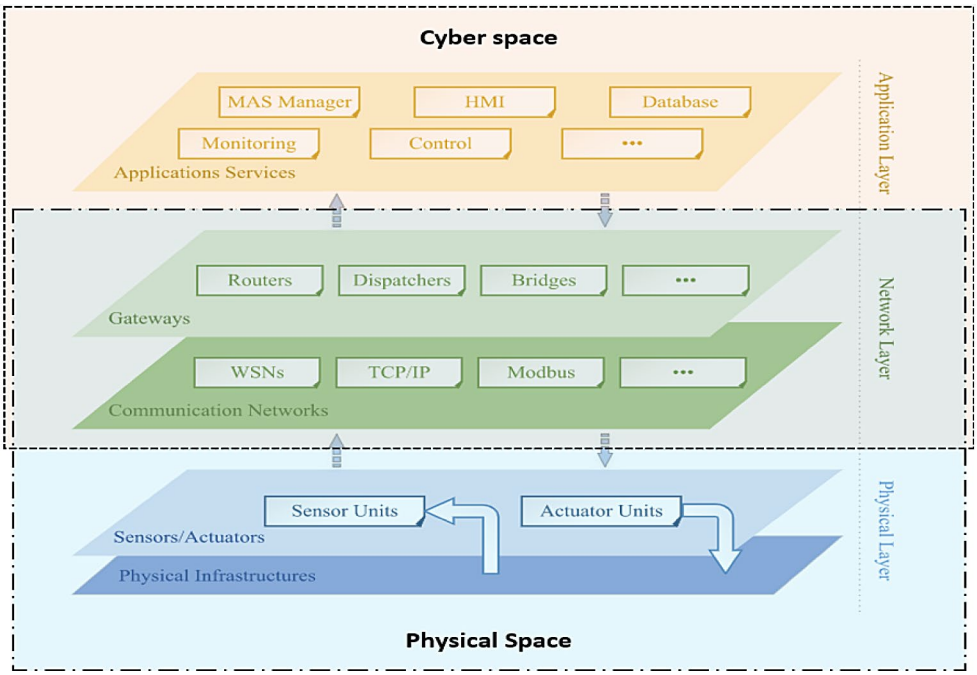


Fig. 1 Architecture of a cyber-physical system. Adapted from [1]

A message broker is often a tool chosen to act as a publish-subscribe enabler, utilizing the MQTT protocol, among others. There are numerous message brokers available, each of which is unique and specialized in a specific area [18, 21]. As demonstrated, this approach has been largely used and has many benefits for the current CPS architectures. However, a study of the usage and deployment of several message brokers in the same CPS in order to harvest the best characteristics of each message broker, or even an infrastructure of message brokers that allows the multiple CPS to utilize the best characteristics of each message broker to communicate between several components on a CPS, has yet to be made.

Data Collection

The first step regarding the gathering of enough data to support a given system, for example a CPS, is the data collection. It encompasses the process from the extraction of the data from the systems to the analysis and ultimately preprocessing of some information.

The data collection is overall a broader spectrum of the act of extracting data, as it can have tools to analyze and correct missing or wrong information, making a pre-processing of the data before being stored or sent to other systems for ingestion or analyzes [22]. Several articles focused on data collection systems that make use of architectures with middleware to implement the flow of information required for their systems. In [23] and [24] the authors used a middleware system to implement the connection between the data sources to the cloud environment, while in [25] the author claims the data collection as the whole system, ranging from the data extraction to the data storage, but also using a message broker base system to implement the seamless communication between the different data sources, the streaming service, and connection to the database.

In the vast world of big data, data collection is a crucial first step. It enables not only the extraction of information from both new and legacy systems but also some preprocessing. This preprocessing means that even in cloud-based processing environments, not all information needs to be sent, reducing bandwidth consumption. pre-analysis can be done locally, along with the discarding of irrelevant data.

Finally, an interesting approach was from the author of [26], given different view regarding the data collection with a more focused approach on machine learning and big data. The author classifies the three subtopics of data collection, data acquisition, data labelling and existing data. The author demonstrates how data collection does not need to pass by data extraction, as software can generate the data, not focusing on data sources (such as sensors). Despite this, data extraction is usually a very important component for the perception of the factory but specifically for the automatic systems or preventive maintenance software among others, being worthwhile exploring a little more about the process.

Data Extraction

Data extraction is a component that collects data from a range of sources, with sensors being the most common. However, this component is ideally not limited by any type of data or format, being able to collect any deemed relevant data. The concept of data extraction is of utmost importance, as the data needs to be extracted after being sensed so the system can analyze the information and act upon it, one example of such is the additive industry and their online feedback control [27]. With the increase in sensing of all the industrial complexes, not only the shop floor but also other areas, mainly as a consequence of the IoT development. Around 14 billion devices are calculated to be present in 2022 [28, 29] doubling this value by 2030, this can also be corroborated by an increase in published papers on the term IoT [30]. As such, it is important to have infrastructure and methodologies to be able to harvest all of this data without an overflow of the systems implemented and have solutions that can scale with the exponential increase of the devices.

An observation of the methodology regarding the data extraction can be made to quickly find a recurrent and overall design of the data extraction. One of this is that the architectures found in the state of the art such as [23– 25, 31, 32]. These architectures focus primarily on the usage of a middleware where all the sensors, or, in a broader sense, data sources, can connect and send their data, either locally or through a cloud platform. As it is possible to see Fig. 2, the architectures focus on the same idea, and with the current technological standpoint, this architecture can easily be satisfied in the requisites department. However, this insight suggests that it is indeed one of the most useful methods for connecting data sources, especially regarding data extraction.

Other authors have made several studies about the technologies used in the data extraction or ingestion points of the system. In these studies, we can observe that most technologies used are: ingestion platforms, usually in cloud environments; message brokers that are used locally and in the cloud; and ad hoc software, tailored solutions that do not utilize specific software to be made [33, 34]. The article in [34] makes a interesting survey of the different aspects of data pipelines in industry. Focusing on the ingestion part, it is found that most of the technology used is in ad hoc applications due to the specific cases of each industry scenario, such as protocols, types of.

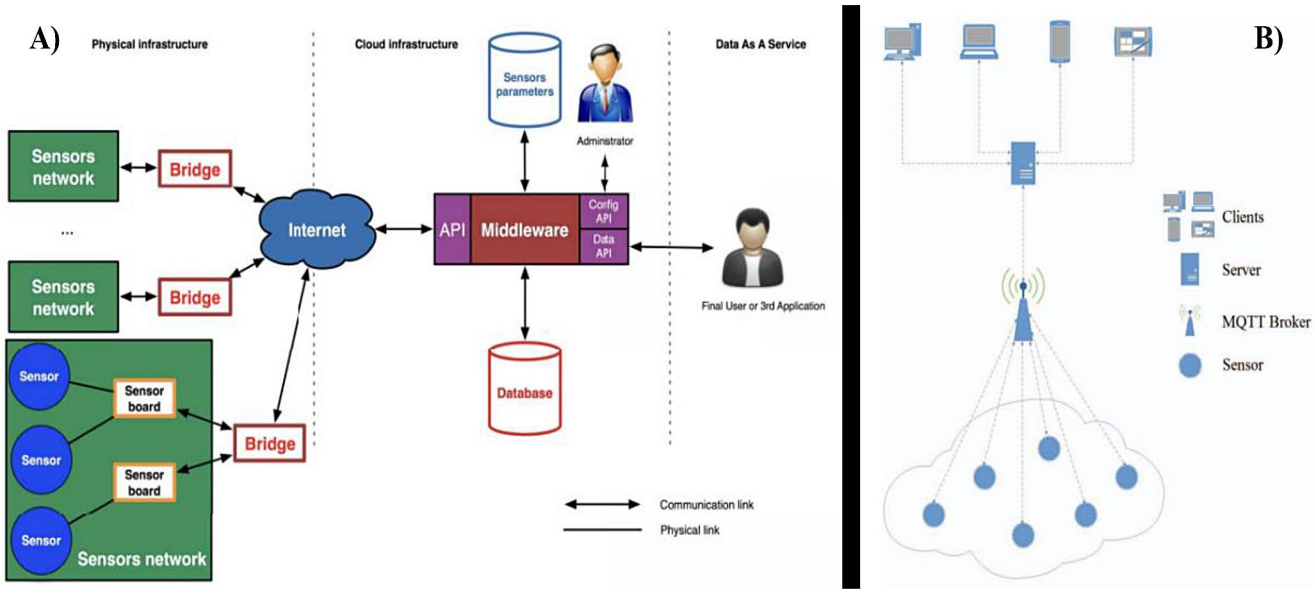


Fig. 2 Data extraction architecture, based in the same methodology **A**) [31] and **B**) [32]

communications, and standards. However, a solution to help reduce this number of ad hoc solutions should be found to help streamline the process of communication.

Here, Open Platform Communications Unified Architecture (OPC-UA) communication can also be one of the fundamental reasons, as it is becoming more and more a typical industrial practice, and as such, custom applications or servers need to be built to accommodate the protocol. Being a useful protocol that PLCs can use seamlessly and being even recommended and used as an example by Reference Architectural Model Industrie 4.0 (RAMI 4.0) [35], this protocol seems to be one of the ideal solutions regarding data extraction, at least when PLCs are involved.

When ad hoc systems are not used, a preference for ingestion and message broker software prevails; technologies such as Apache Kafka, RabbitMQ, Amazon Kinesis, Microsoft Event Hubs, and Google Pub/Sub are pointed out [33, 36]. It is important, more than relying on the technology presented in these articles, to see beyond and realize the characteristics that compose it, which are usually ingestion or message brokers/streaming capable of event-driven reasoning and having near real-time capabilities, with some software being able to be deployed locally and, in the cloud, and others being exclusively cloud services. Nevertheless, a combination of these software can also be used as a centralization of data before being sent to cloud platforms, e.g., a message broker collecting all the data from a shop floor before being sent to a cloud platform for analysis, which can be a feasible and ideal approach in several cases.

Data Extraction Framework

Proposed Framework

As previously mentioned, different CPS have different characteristics and requisites based on several factors, such as, how many and what type of devices are meant to communicate, number of data extracted and processed, among others. As such, the usage of different message brokers, that have different characteristics, in a given CPS can be the correct decision in order to save resources and make the process modular. With the deployment of several message brokers, their configuration, deployment, and reconfiguration become critical factors that can be a challenging job given the different technologies and methods used for each message broker.

The proposed framework focuses on being able to change automatically and remotely, different aspects of the message brokers used to extract data from the physical elements (Fig. 3).

In the proposed framework, three different message brokers are deployed, this can be seen in practice as different work machines in a given CPS, with different characteristics and requirements that need to be complied with. The deployed message brokers can be N, as long as the necessary resources of the machine can compel the usage of the message brokers, for demonstrations purposes three were used to exemplify the framework. The message brokers are deployed along with a client tool. This client tool is responsible for all the interactions with the message broker regarding different kinds of actions, such as, configuration, deployment, reconfiguration, and termination. All the client

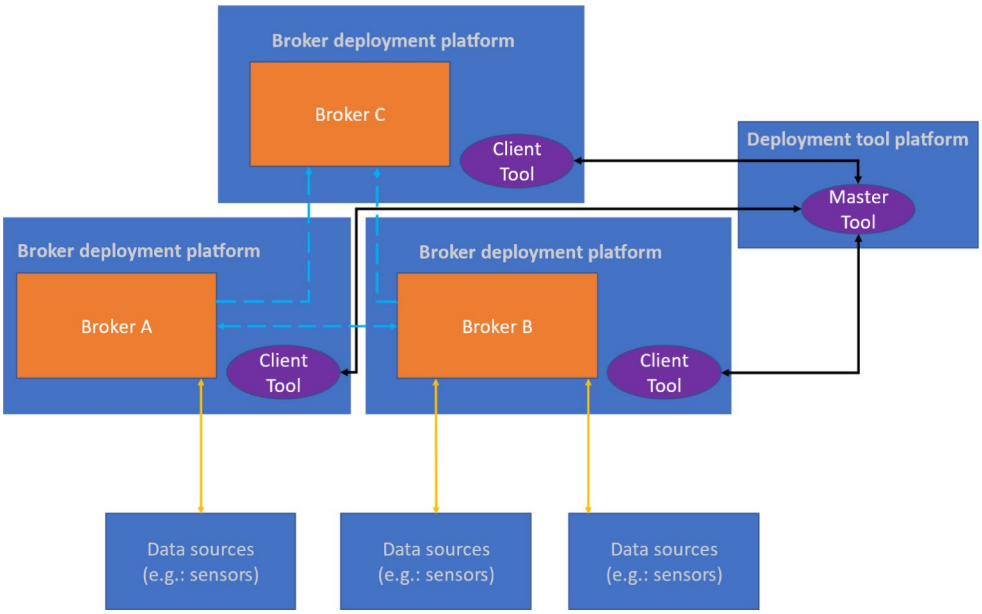


Fig. 3 Proposed framework (black lines - communication between the config tools, blue lines - communication between the broker, yellow lines - communication between data sources and brokers)

tools regarding each CPS should therefore only be accessible by one master tool, which is responsible for giving orders to each client tool to execute each action. This approach can be interpreted as a master tool dictating what should happen for each message broker, while the client tool runs different methods depending on what the master dictates.

The framework can therefore be useful to different degrees, such as load balancing, interconnection or sharing of different information between message brokers, the remote changing of configurations, or even the automatic redeployment of several instances from a unique point.

Finally, it is important to draw attention to the usage of different message brokers in different contexts, which is essential for the correct use of the framework. A simple example can be the usage of low-resource-intensive message brokers, only focused on data extraction on the shop floor (being Broker A and Broker B of Fig. 3), and a message broker with high throughput and scalability (Broker C of Fig. 3). This represents the flexibility of the framework and allows the usage of the best characteristics of each message broker, while the client tool seamlessly interconnects all the message brokers, creating an automatic and continuous flow of data. To easily implement and understand the framework an ontology and engineering process will be described, giving a clear view of the inside and implementation process.

Ontology

An ontology is a structure meant to represent in a clear form the different relationships, attributes, and hierarchy, among other important definitions, of the heterogeneous components that compose a system. These definitions are essential for a complete and understandable framework, capable of being utilized in different circumstances and harvesting their full potential. Figure 4 presents the ontology of the framework of the system.

The ontology is composed by six classes graphically represented in Fig. 5, described as such:

- **Resources – Resources** are data sources and can be any type of component or software capable of generating information. However, it is required to have some form of communication, capable of sending information to the brokers. The resources also have several objects properties that include which **Skills** the resource can **execute()**, the **is_type()** that relates the corresponding **R_type** of the **Resource**, the **properties()** that correlate directly with a corresponding **Properties** class and the **broker()** where it associates the **resource** to a specific **broker** where the information of the **resource** will be sent. Finally, the **Resources** also have several data properties that include **Brand**, **ID**, **Localization**, **Name** and **Description**.
- **R_type** – This class represents the type of **resource** that exists, associated with the **is_type()** object property from **resources**. This class data properties are **Description**, **ID**, **Name** and **Attribute**.
- **Property** – The class **property** represents characteristics of the **resource** class and are connected by the **properties** object property. This class possesses the **Description**, **ID**, **Name**, **Value**, and **Value_type** as data properties. It is important to clarify the **Value** and **Value_type** data properties. The **Value** is meant to have a designated value, being any type of data. The **Value_type** on the other hand represents the data type

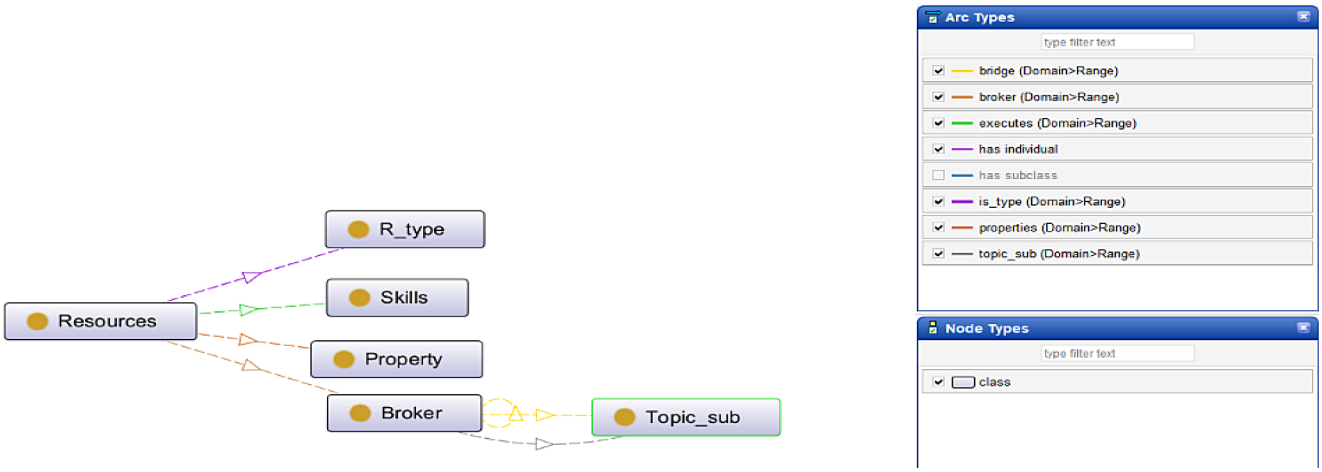
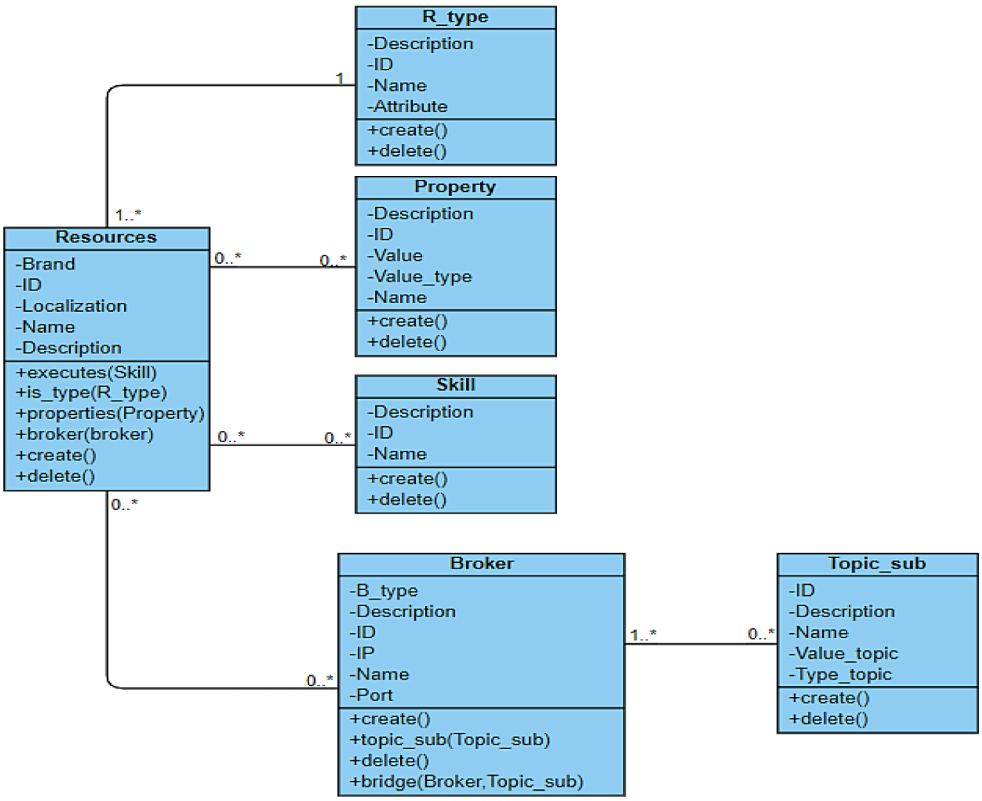


Fig. 4 Ontology graph of the framework

Fig. 5 Class diagram of ontology



associated to the **Value**, this allows the class to be extremely flexible regarding the values stored.

- **Skill** – **Skills** are actions that the resources can make. These actions can be a multitude of different kinds and can be associated with the specific **resource** by the **executes** object property. The **skill** class has the **Description**, **ID**, and **Name** as data properties.
- **Broker** – This class is responsible for the **brokers** implemented in the system. Being associated with **resource** by the **broker** object property, the **broker** class possesses the object properties **topic_sub()** meant to subscribe to a specific topic and the **bridge()** able to connect two **brokers** bound by **topic_sub**. The data properties of this class are the **B_type** (meant to specify the type of broker), **Description**, **ID**, **IP**, **Name**, and **Port**.
- **Topic_sub** – The **topic_sub** class represents the topics that exist in the system, these topics can be associated to a **broker** class by the **topic_sub()** object property. The class **topic_sub** has the data properties of **ID**, **Description**, **Name**, **Value_topic**, and **Type_topic**. The **Value_topic** and the **Type_topic** is similar to the ones from **property** class, in case of necessity they can be

used to describe the values and/or format of data, that flows in a specific topic.

Engineering Process

The engineering process is the act of transforming abstract ideas into tangible realities through a structured and systematic approach. It serves as the guiding light for engineers, laying out the steps and methodologies essential to turning concepts into innovative solutions. Figure 6 presents the flowchart of the implementation process of the framework.

The process starts by deploying the clients across all the platforms that need to deploy or configure the message brokers. Upon the completion of the client's setup, the master tool is then deployed on a designated platform, ideally a computer, that will control all the client tools. This master tool platform needs to have a network connection to the client tools so orders can flow seamlessly between the tools. When the tools are deployed and ready to take orders, it is then necessary to make a flow of information, starting with the deployment of the message brokers. As such, the question arises "Are the brokers deemed necessary for the system already created?", If not, the master tool contacts the specific client tool to deploy the broker.

When the brokers are all deployed, it is time to check the configurations and bridges between the different brokers. If anything is missing, the configurations and bridges are requested by the master tool to the client tools, updating the configuration of each broker and interconnecting them when needed.

If any topic is still not created, this is the final step of initialization. Most message brokers can automatically create topics when they are requested by the data sources. Nevertheless, some topics may not want the standard configurations, and in such cases, it is important to previously create them.

When all the initial setup is concluded, it is time to connect the data sources to the specific brokers and, consequently, the topics. Finally, after all these processes are concluded, one last question remains: the final evaluation of the flow of information in the whole system. If there is any problem or the information is not flowing as predicted, it is time to go to step one and question each part of the process, correcting any mistakes or improving the already-built infrastructure.

Giving a different perspective regarding the idea of communication within framework, a sequence diagram is presented in Fig. 7. The diagram exemplifies communication following the flowchart in Fig. 6 Figure.

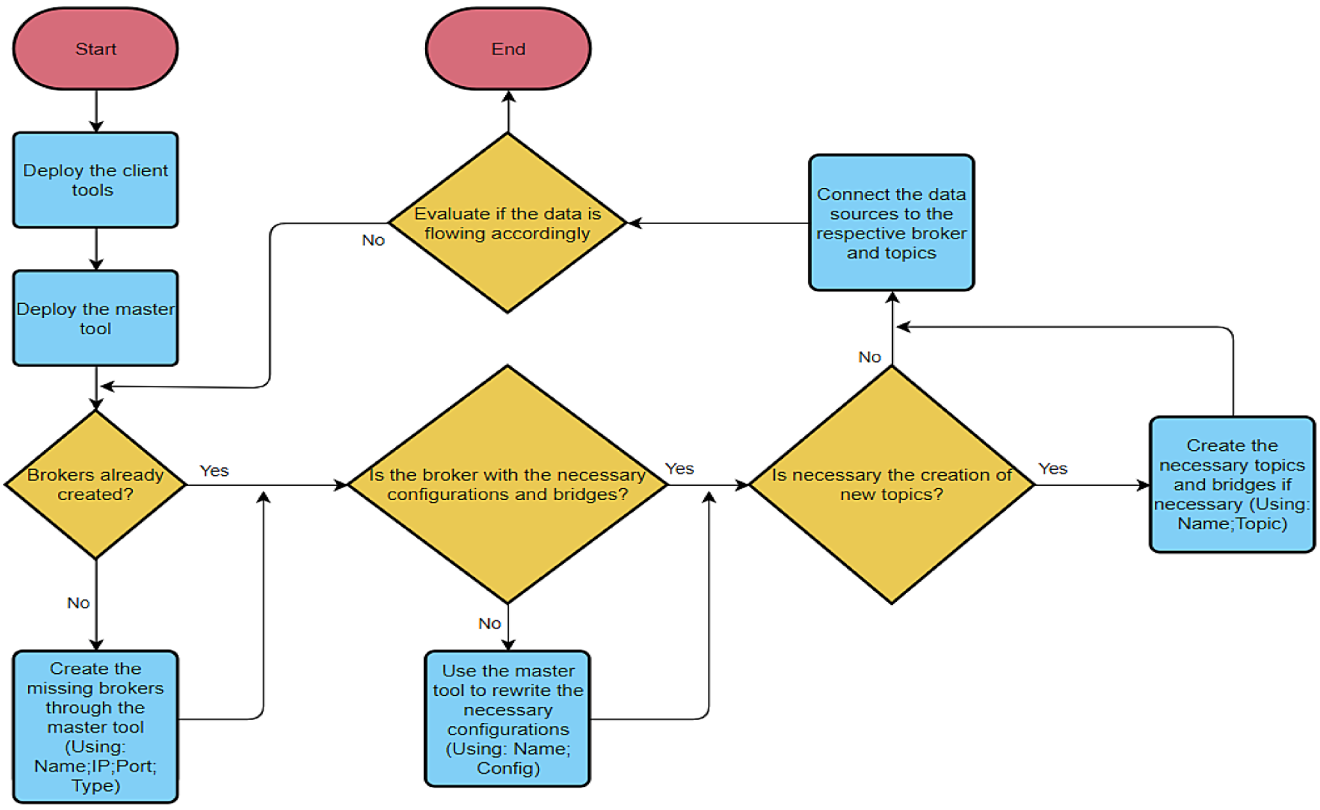


Fig. 6 Flowchart of the framework setup

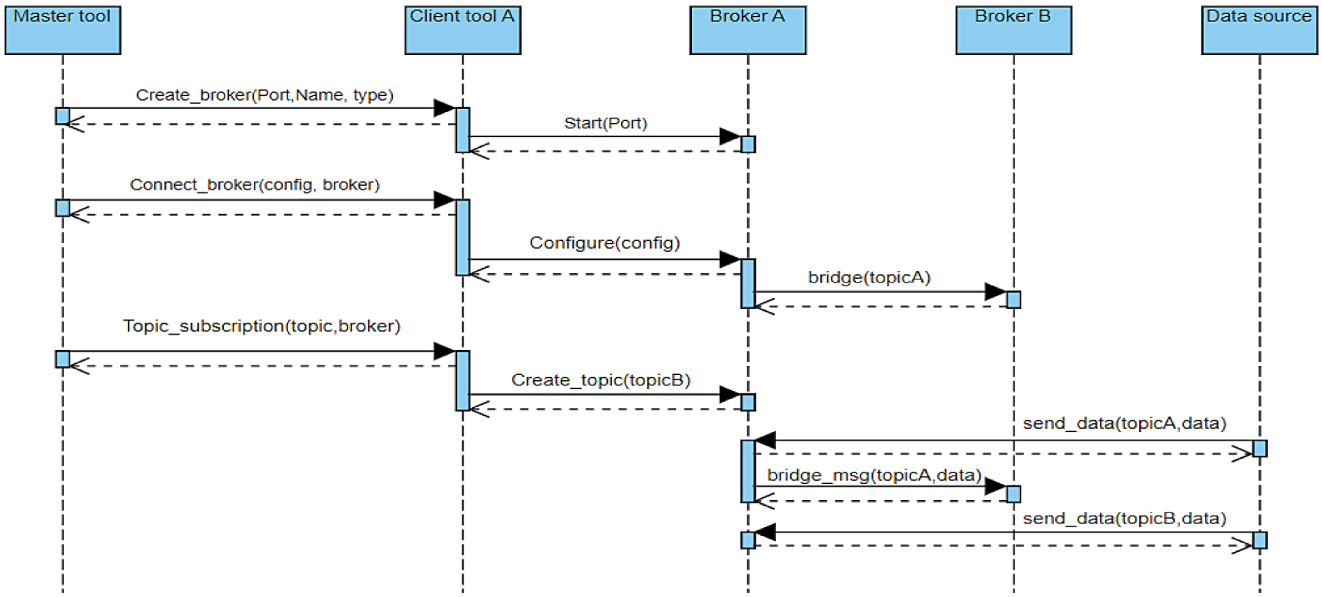


Fig. 7 Sequence diagram of the framework setup and action

The sequence diagram in Fig. 7 starts with the communication of the master tool with the client tool, issuing the creation of a message broker. Simply put, the main role of the master tool is to be the orchestrator and dictate how all the respective client tools should be created, connected, and their role. When the instruction of creation is issued, the master tool registers internally the information of the broker, like the IP, port, name, and type of broker. As the client tool receives the instruction from their master and sends an acknowledgement, it proceeds to make the necessary preparations to start the broker, in this case, Broker A. The type of broker, port, and name are specified by the master tool, and as such, the client tool only needs to start the chosen broker, specifying the port.

The second communication flow starts when a configuration or bridge connection is deemed necessary. The master tool sends a message with the configurations and a broker for the client tool. The broker is sent because several brokers can be associated with a single client. After the client tool returns the acknowledgement, it reconfigures the broker of its machine as well as makes the necessary steps for a bridge configuration with the specified broker in the configuration parameter. The bridge between two brokers is often associated with a topic that is shared between the brokers. In this specific case, it is possible to see the initial setup communication between Broker A and Broker B, where topicA is the bridge topic between the brokers.

The last communication from the master tool is regarding the creation of relevant topics. In this case, similar to the configuration of the broker, a message is sent to the client tool with the topic configuration deemed relevant and the broker for which the topic will be created. After sending the acknowledgement, the client tool communicates with the broker and creates the topic.

After all the previous steps are done, the master tool is no longer needed unless it is necessary to change the flow

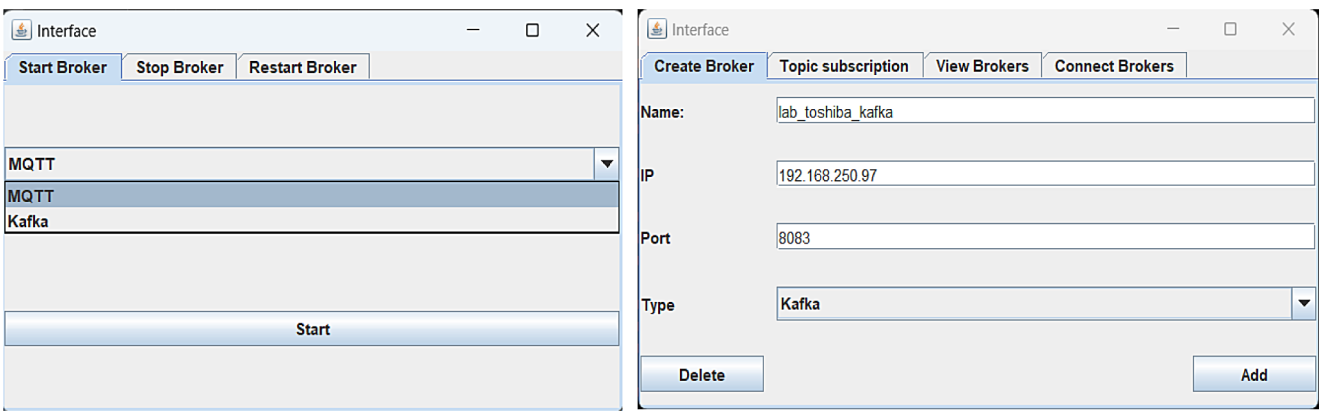


Fig. 8 Client (left) and master (right) tool

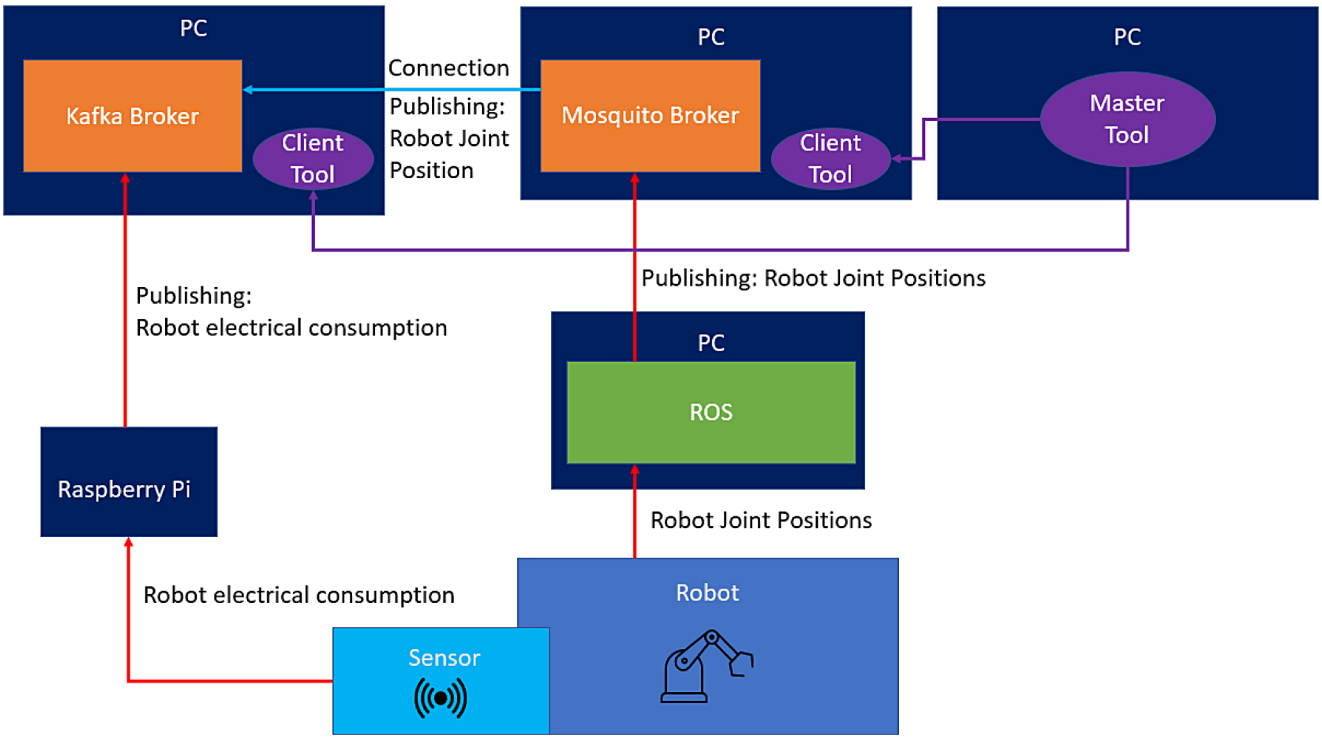


Fig. 9 Example of the utilization of the framework in a CPS

of information. As such, the following communications all depart from the data sources: If communication is possible with the broker, the data source can simply specify the topic and the data, and the data can be sent seamlessly between the two entities. One last note regarding communication refers to the topic that is making a bridge between the two brokers. In the example topicA is making this bridge and sending data to one broker in this topic, the brokers automatically communicate and share the information between them. Bridges can have innumerous configurations, but for simplification purposes, this bridge shares all the information between the two brokers.

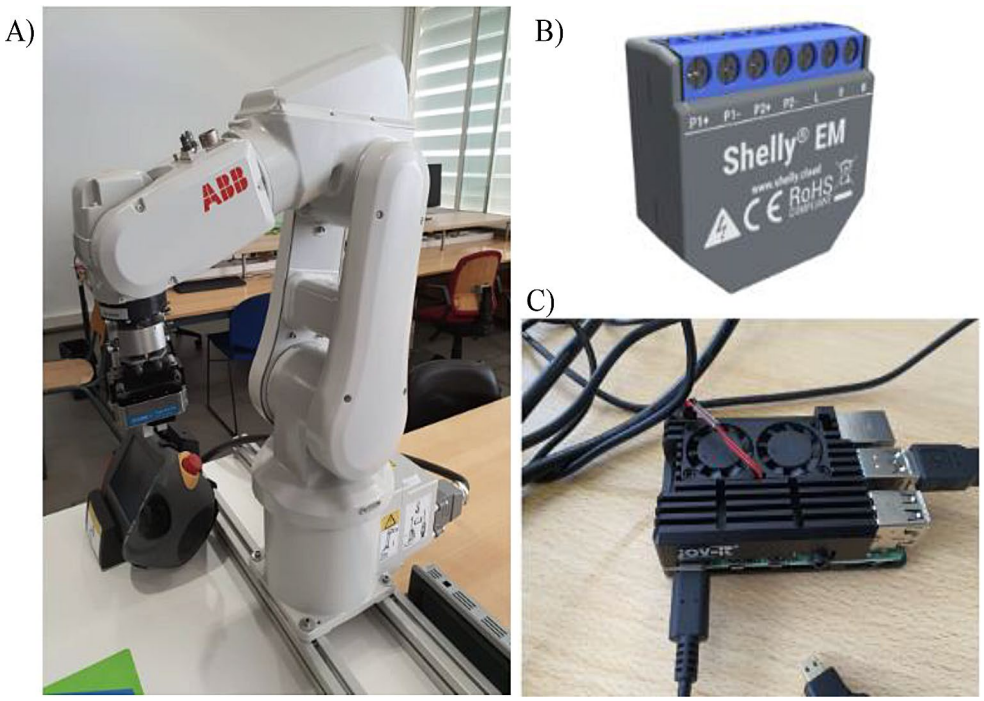


Fig. 10 Hardware used for demonstration. **A**) ABB IRB 120 Robot. **B**) Power monitoring Shelly. **C**) Rasberry PI

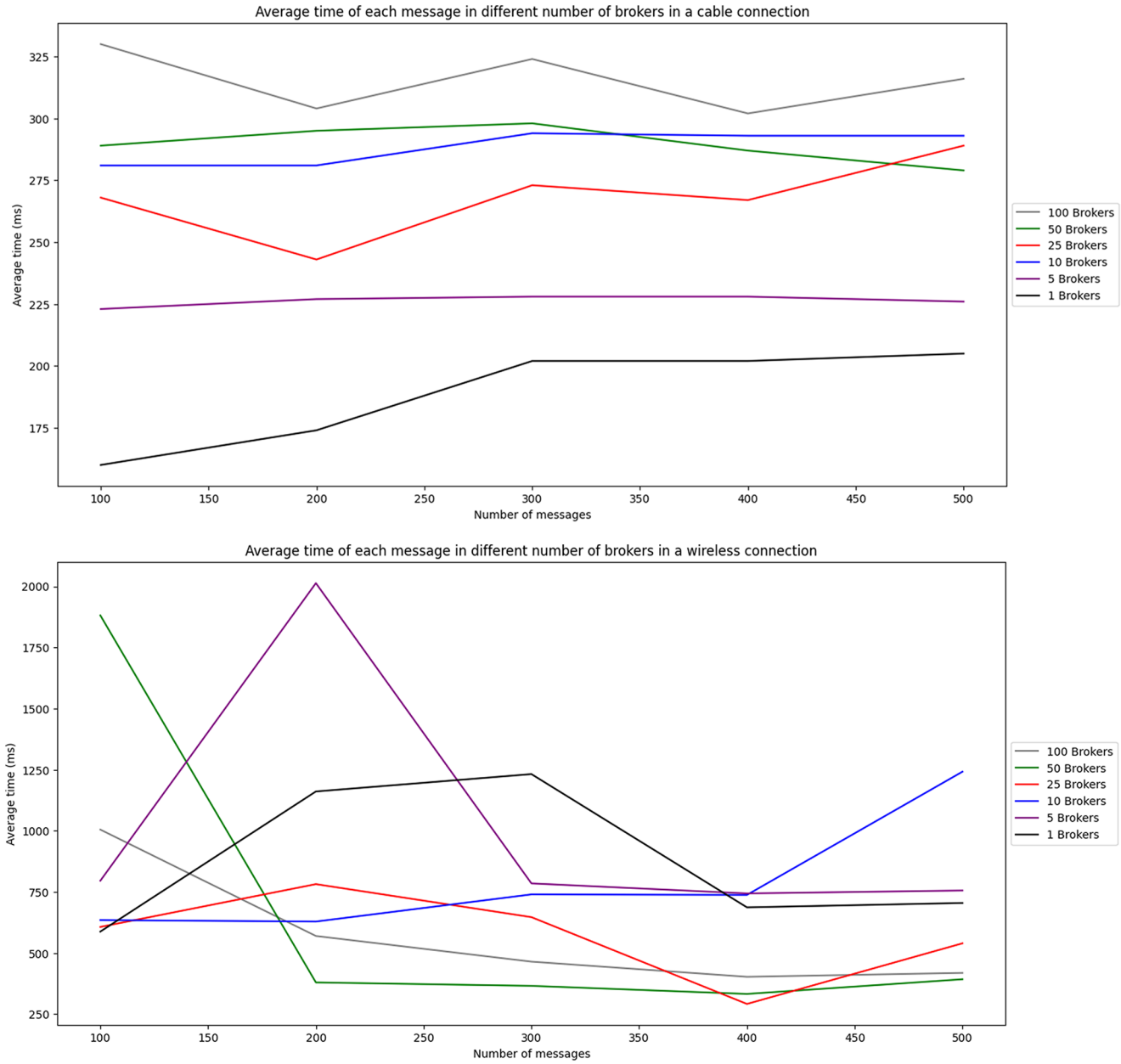


Fig. 11 Load test of Mosquitto message broker, by cable (top) and wireless (bottom)

Demonstration Scenarios

In this section, a master-client tool proposed in the framework is presented, along with two demonstration scenarios. The two demonstration scenarios have different objectives, the first one exemplifies how this connection can and should be done regarding master-client communication and hierarchy, as well as demonstrating in a real-world scenario how all the systems interact. The second demonstration scenario revolves around a deployment and load test of the MQTT Mosquitto software, to accurately see the impact of such a tool on the message broker.

Table 1 Deployment test of Mosquitto message broker

Type of test	Max delay(ms)	Minimum delay(ms)	Average(ms)	Standard deviation(ms)	Number of tests
Local	27	22	23,94	1,06	50
Tool + wireless	417	67	89,09	48,35	50
Tool + cable	88	78	80,07	2,49	50

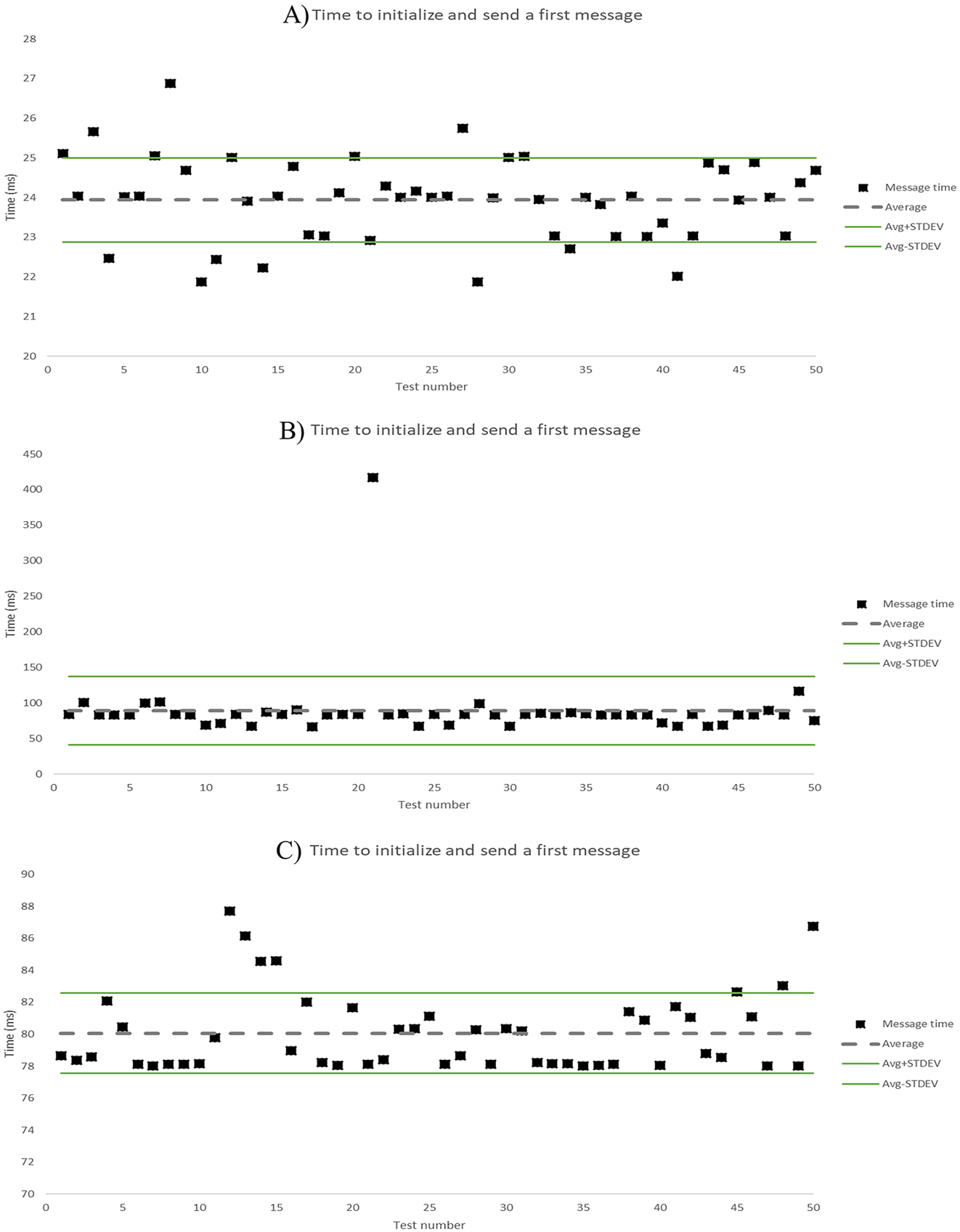


Fig. 12 Deploy tests in **a**) local without tool, **b**) with tool remote by wireless, and **c**) with tool remote by cable

Master-Client Tool

The master-client tool for the demonstration scenario is made in the Java programming language. The principal reasons for choosing such a language are that Java is one of the most popular languages in the world and can run on most operating systems without the need of changing any code, known as "write once, run anywhere". Both of these reasons allow for the facilitation of the execution of the proposed framework, as the client can be instantiated on almost any machine and the libraries are written in Java, a common programming language.

The tool is made to allow a certain degree of modularity, this means that the user only needs to install the libraries for the specific message broker that they want to communicate with in both the client and the master. This allows for the saving of resources on less capable machines and allows the user to customize the library and the tool. The libraries have five main methods:

- **Start** – Where the desired message broker, referring to the library in question, is deployed with the previous or predefined configurations.
- **Connect** – Used for automatically connecting both equal and different types of message brokers.
- **Config** – Where the configuration of the given message broker should be passed on.
- **Stop** – This method stops all the processes of the given message broker.
- **Restart** – This is a method simply for more convenience where the **Stop** and **Start** methods are combined, for a more straightforward approach.

As previously stated, the tool is made up of a client and a master (Fig. 8). Even though the client has an interface and can run simple commands, this is not necessary, as all actions are made through the master. The master, however, has different and useful tools. It can deploy the message broker, and create topics for each specific one, view all the tool's deployed message brokers, and connect two different message brokers.

The tool communicates between the master and the client with HTTP Post and Get. The master communicates with the client by the same IP as the message broker meant to be created. It assumes that if a broker is meant to be deployed by the tool, the client is already operational on the target machine.

Demonstration Scenario: CPS Example

The first demonstration scenario simulates a system where the position of an ABB IRB 120 robot and the energetic consumption of such robot are sent to two different message brokers that are then able to automatically connect and exchange information (Fig. 9). The position of the robot is processed in the Robot Operating System (ROS) and then published on another machine where a Mosquitto message broker is running. In terms of energy consumption, a sensor is linked to a Raspberry Pi, which publishes the data to a Kafka message broker (Fig. 10). The usage of the Kafka message broker is also to demonstrate that powerful event streaming/message broker platforms can be used with this approach, providing a robust and complete integration for data collection corresponding to the necessity of each CPS.

It is important to notice that the tool was able to deploy and correctly configure the message brokers so that the other systems can communicate and pass information to the respective brokers. This configuration is all done remotely, with the introduction of the parameters via the graphical interface. The developed system can be further optimized and automated to meet the needs of the user, the company, or the industrial complex.

Finally, both the message brokers were connected by the master tool order in the connection tab. This connection allows Mosquitto to send data about the robot's location directly to the Kafka message broker, bypassing the need for any intermediary software.

In summary, the best characteristics of each message broker were made possible: Mosquitto for data extraction and Kafka for aggregating and storing the information directly in a database. This was also done without any need to know how to deploy and connect the message brokers, as the client/master tool was used.

Demonstration Scenario: Deployment and Load Testes

For the second demonstration, two tests were made:

First test - The first test involves deploying a Mosquitto message broker without using the tool and trying to publish a small message (< 1 Kb). When this succeeds, the time is recorded, and the broker is terminated. This test is repeated fifty times. Subsequently, two more deployments are conducted using the tool with the same test methodology, one via a wireless network and the other via a cable network (Table 1). These tests aim to compare the time taken by both approaches (manual deployment and deployment via the tool). It is important to note that deploying through the tool inherently introduces an increased delay due to network connectivity. Nonetheless, these tests provide a reasonable indication of whether the tool can deploy the message broker remotely within a reasonable time frame.

Despite the average deployment time increasing by approximately three-fold in both tests, this can be attributed

almost entirely to the internet connection, specifically the time taken for the master tool to send the deployment order and for the first message to be published. Comparing the maximum and minimum values reveals how significantly a connection can affect the time, especially in wireless communication. For comparison, if 5ms are added to the deployment time on the local machine for each communication transaction (send the deploy message, receive acknowledgment, send publish message, receive acknowledgment), the time almost doubles. Considering this delay, the deployment time remains acceptable for a message broker, given its remote capability. This is further justified as this task is typically performed only when configuration or problem arises, not recurrently. For easier visualization, graphs of each test are provided (Fig. 11).

Second test - The second test assessed the load capacity by sending several messages (100, 200, 300, 400, and 500) with reduced size (< 1 Kb) to multiple Mosquitto brokers (1, 5, 10, 25, 50, and 100) running on the same machine. The messages were sent simultaneously or as close to simultaneously as possible. The ratio of messages to brokers was always one-to-one, but the total number of messages and brokers varied. These tests were conducted both wirelessly and via cable (Fig. 12).

In the load test, the significant influence of a wireless system on the average message time becomes evident. The small size of the messages and the quick response time mean that the noise in a wireless network greatly affects the readings. Consequently, the graph at the bottom of Fig. 12 shows excessive delays due to network connectivity, rendering the load test information unreliable. Nevertheless, this result serves as a clear demonstration of why such systems should preferably operate in a cabled environment rather than a wireless one.

The top graph of Fig. 12 reveals an interesting characteristic: the number of message brokers has less influence on the average time than the quantity of messages. However, as the number of brokers increases, this impact becomes less noticeable. The difference in performance between one broker and five brokers is nearly the same as the difference between fifty brokers and one hundred brokers. For a lower number of brokers (1 to 5), the average message time increases with the deployment of more brokers and the sending of more messages. Conversely, in the range of 10 to 50 message brokers, some anomalies occur. For instance, the deployment of 10 message brokers has a worse impact than deploying 25 or even 50 message brokers. This indicates that a larger deployment of message brokers does not always correlate linearly with a worsening of performance. An explanation for this discrepancy can be due to overheads in the communication between the message brokers. However, it is still too early to draw definitive conclusions, as more tests are needed. It is important to note that the scale of Fig. 12 is not equal, as it was adjusted to register all points even if some time discrepancies between the messages occurred.

Conclusion

The work presented in this paper proposes a framework that facilitates the deployment, configuration, and connection of message brokers in a CPS, focusing particularly on industrial complex environments. The framework allows for the development of tools capable of incorporating either third-party or custom-mad tools that facilitate all the configurations needed for the correct utilization of a message broker.

Two demonstration scenarios were presented, the first one regarding an exemplar integration in a CPS industrial environment. A tool programmed in Java was made in order not only to prove the concept but also to serve as a guideline for how to correctly instantiate the tools that would be responsible for all the actions regarding the message brokers. The demonstration also serves as a guide for how to make the program modular so that different libraries with the same standards from different users can be integrated.

The second demonstration scenario shows that despite the slowdown in the deployment of the message broker, this mostly relates to the delays in the network that now play an important role. Despite this, the time that it took still did not reach a quarter of a second and was deemed acceptable as the time it took to launch the message broker by the tool. In this second demonstration, there were also some load tests that correlated the number of messages and the number of brokers deployed with the time of sending a message. The wireless results of such a study were deemed not useful because the delays in the network overlapped any attempt to interpret such data. Regarding the cable data, it revealed some intriguing results, as the delay caused by broker deployment does not appear to be linear, implying that more brokers deployed have less impact. If this proves true, it can be a useful conclusion for some particular cases, which would benefit from having more brokers given a particular number of messages. Nevertheless, this needs further study, as the test presented does not have enough validation to make a claim, but it is an interesting subject to be studied in future work.

Overall, a framework is presented, capable of extracting the best of each message broker's technologies. However, this approach always has the limitation of utilizing a message broker and consequently the connection of each, which is limited by the advantages and disadvantages of each message broker. The load tests also show that maybe what the authors thought is the best way to use a message broker is

not always true. The deployment of several brokers and load balancing between them could be a useful technique although needs further and more intensive testing.

For future work, it is important to test the framework in a real industrial setting and even other environments, as it can be generalized for other use cases. It is important to also make the load and deployment tests with larger messages to see what impact it has and how it differs from the results gathered in this study. Also, to utilize other message brokers, as it can be interesting to see the impact of for example RabitMQ, HiveMQ or even an event streaming like Kafka. Finally, this system can be modeled to be RAMI 4.0 compliant or even other reference architecture of Industry 4.0, this can be of added value towards a smother utilization and integration with already existing Industry 4.0 industrial environments.

Author Contributions Conceptualization, N.Freitas, A.D.Rocha, F.M-Oliveira, D.Alemão and J.Barata; methodology, N.Freitas, and A.D.Rocha; software, N.Freitas., F.M-Oliveira, D.Alemão; validation, N.Freitas, F.M-Oliveira, D.Alemão; formal analysis, A.D.Rocha and J.Barata; investigation, N.Freitas and A.D.Rocha writing—original draft preparation, N.Freitas, A.D.Rocha, F.M-Oliveira, D.Alemão; writing—review and editing, N.Freitas, A.D.Rocha, F.M-Oliveira, D.Alemão and J.Barata; project administration, J.Barata. All authors have read and agreed to the published version of the manuscript.

Funding Open access funding provided by FCT|FCCN (b-on).

Data Availability The data can be requested and provided by the corresponding author.

Declarations

Research Involving Human and/or Animals Not applicable.

Informed Consent Not applicable.

Competing Interests The authors declare no conflict of interest.

References

1. Januário F, Cardoso A, Gil P. A distributed Multi-agent Framework for Resilience Enhancement in Cyber-physical systems. IEEE Access. 2019;7:31342–57. https://doi.org/10.1109/ACCESS.2019.2903629.
2. Hoffmann MW et al. Developing industrial CPS: A multi-disciplinary challenge. Sensors. 2021 Mar;21(6):1–28. https://doi.org/10.3390/s21061991
3. Tsigkanos C, Kehrer T, Ghezzi C. Modeling and verification of evolving cyber-physical spaces. In: Proceedings of the ACM SIGSOFT Symposium on the Foundations of Software Engineering; 2017 Aug; 38-48. New York: Association for Computing Machinery; 2017. https://doi.org/10.1145/3106237.3106299
4. Nguyen CN, Lee J, Hwang S, Kim JS. On the role of message broker middleware for many-task computing on a big-data platform. Cluster Comput. 2019 Jan;22:2527-40. https://doi.org/10.1007/s10586-018-2634-9.
5. Mishra B, Kertesz A. The use of MQTT in M2M and IoT systems: A survey. IEEE Access. 2020;8:201071–86. https://doi.org/10.1109/ACCESS.2020.3035849.
6. Rocha AD, Freitas N, Alemão D, Guedes M, Martins R, Barata J. Event-Driven Interoperable Manufacturing Ecosystem for Energy Consumption Monitoring. Energies (Basel). 2021 Jun;14(12):3620. https://doi.org/10.3390/en14123620.
7. Kadir BA, Broberg O, da Conceição CS. Current research and future perspectives on human factors and ergonomics in industry 4.0. Comput Ind Eng. 2019 Nov;137:106004. https://doi.org/10.1016/j.cie.2019.106004.
8. Xu X, Lu Y, Vogel-Heuser B, Wang L. Industry 4.0 and Industry 5.0—Inception, conception and perception. J Manuf Syst. 2021 Oct;61:530-5. https://doi.org/10.1016/j.jmsy.2021.10.006.
9. Tao F, Qi Q, Liu A, Kusiak A. Data-driven smart manufacturing. J Manuf Syst. 2018 Jul;48:157-69. https://doi.org/10.1016/j.jmsy.2018.01.006.
10. Pivoto DGS, de Almeida LFF, da Rosa Righi R, Rodrigues JJP, Lugli AB, Alberti AM. Cyber-physical systems architectures for industrial internet of things applications in Industry 4.0: A literature review. J Manuf Syst. 2021 Jan;58:176-92. Elsevier B.V. https://doi.org/10.1016/j.jmsy.2020.11.017
11. Tonelli F, Demartini M, Pacella M, Lala R. Cyber-physical systems (CPS) in supply chain management: From foundations to practical implementation. Procedia CIRP. 2021;98:598-603. Elsevier B.V. https://doi.org/10.1016/j.procir.2021.03.080
12. Shi J, Wan J, Yan H, Suo H. A survey of Cyber-Physical Systems. In: Proceedings of the 2011 International Conference on Wireless Communications and Signal Processing (WCSP); 2011 Nov; 1-6. IEEE; 2011. https://doi.org/10.1109/WCSP.2011.6096958
13. García-Valls M, Baldoni R. Adaptive middleware design for CPS: Considerations on the OS, resource managers, and the network run-time. In: Proceedings of the 14th Workshop on Adaptive and Reflective Middleware (ARM 2015); 2015 Dec; Collocated with ACM/IFIP/USENIX Middleware 2015. New York: Association for Computing Machinery, Inc; 2015. https://doi.org/10.1145/2834965.2834968
14. Parto M, Saldana C, Kurfess T. A novel three-layer IoT architecture for shared, private, scalable, and real-time machine learning from ubiquitous cyber-physical systems. Procedia Manufacturing. 2020;51:959-67. Elsevier B.V. https://doi.org/10.1016/j.promfg.2020.05.135.
15. Gavriluta C, Boudinet C, Kupzog F, Gomez-Exposito A, Caire R. Cyber-physical framework for emulating distributed control systems in smart grids. Int J Electr Power Energy Syst. 2020 Jan;114:105373. https://doi.org/10.1016/j.ijepes.2019.06.033.
16. Sakurada L, Barbosa J, Leitao P, Alves G, Borges AP, Botelho P. Development of agent-based CPS for smart parking systems. In: Proceedings of the IECON 2019–45th Annual Conference of the IEEE Industrial Electronics Society; 2019 Oct; 2964-9. IEEE; 2019. https://doi.org/10.1109/IECON.2019.8926653

17. Naeem RZ, Bashir S, Amjad MF, Abbas H, Afzal H. Fog computing in internet of things: Practical applications and future directions. Peer Peer Netw Appl. 2019 Sep;12(5):1236-62. https://doi.org/10.1007/s12083-019-00728-0
18. Bagaskara AE, Setyorini S, Wardana AA. Performance analysis of message broker for communication in fog computing. In: Proceedings of the 2020 12th International Conference on Information Technology and Electrical Engineering (ICITEE); 2020 Oct; 98-103. IEEE; 2020. https://doi.org/10.1109/ICITEE49829.2020.9271733
19. Hastbacka D, Kannisto P, Katkytniemi A. Interoperability of OPC UA PubSub with existing message broker integration architectures. In: Proceedings of the IECON (Industrial Electronics Conference); 2022. IEEE Computer Society; 2022. https://doi.org/10.1109/IECON49645.2022.9969039
20. Imran SA, Akhtar S. Safe and secure communication between two cyber-physical systems: a framework for security. 2021. p. 541-58. https://doi.org/10.1007/978-3-030-76632-0_19
21. Bertrand-Martinez E, Dias Feio P, de Brito Nascimento V, Kon F, Abelém A. Classification and evaluation of IoT brokers: a methodology. Int J Netw Manag. 2021 May. John Wiley and Sons Ltd. https://doi.org/10.1002/nem.2115
22. Mehmood H et al. Apr. Implementing big data lake for heterogeneous data sources. In: Proceedings of the 2019 IEEE 35th International Conference on Data Engineering Workshops (ICDEW); 2019 Apr; 37-44. IEEE; 2019. https://doi.org/10.1109/ICDEW.2019.00-37
23. Zhang L, Li F, Wang P, Su R, Chi Z. A blockchain-assisted massive IoT data collection intelligent framework. IEEE Internet Things J. 2022 Aug;9(16):14708-22. https://doi.org/10.1109/JIOT.2021.3049674.
24. Trunzer E, Prata P, Vieira S, Vogel-Heuser B. Concept and evaluation of a technology-independent data collection architecture for industrial automation. In: Proceedings of the IECON 2019–45th Annual Conference of the IEEE Industrial Electronics Society; 2019 Oct; 2830-6. IEEE; 2019. https://doi.org/10.1109/IECON.2019.8927399
25. Kuzlu M, Kalkavan H, Gueler O, Zohrabi N, Martin PJ, Abdelwahed S. An end-to-end data collection architecture for IoT devices in smart cities. In: Proceedings of the 2022 IEEE Power and Energy Society Innovative Smart Grid Technologies Conference (ISGT 2022); 2022. IEEE; 2022. https://doi.org/10.1109/ISGT50606.2022.9903049
26. A survey on data collection for machine learning: a big data-AI integration perspective. IEEE Trans Knowl Data Eng. 2021 Apr;33(4):1328-47. https://doi.org/10.1109/TKDE.2019.2946162.
27. Lupi F, Pacini A, Lanzetta M. Laser powder bed additive manufacturing: a review on the four drivers for an online control. J Manuf Process. 2023 Oct;103:413-29. https://doi.org/10.1016/j.jmapro.2023.08.022.
28. Number of Internet of Things. (IoT) connected devices worldwide, 2023. [Online]. Available: https://www.statista.com/statistics/1183457/iot-connected-devices-worldwide/1/4
29. Number of. connected IoT devices growing 16% to 16.7 billion globally, 2023, Accessed: Sep. 08, 2023. [Online]. Available: https://iot-analytics.com/number-connected-iot-devices/
30. Dachyar M, Zagloel TYM, Saragih LR. Knowledge growth and development: internet of things (IoT) research, 2006–2018, Heliyon. 2019 Aug;5(8) https://doi.org/10.1016/j.heliyon.2019.e02264
31. Mocnej J, Lojka T, Zolotova I. Using information entropy in smart sensors for decentralized data acquisition architecture. In: Proceedings of the 2016 IEEE 14th International Symposium on Applied Machine Intelligence and Informatics (SAMI); 2016 Jan; 47-50. IEEE; 2016. https://doi.org/10.1109/SAMI.2016.7422980
32. Cecchinel C, Jimenez M, Mosser S, Riveill M. An architecture to support the collection of big data in the Internet of Things. In: Proceedings of the 2014 IEEE World Congress on Services; 2014 Jun; 442-9. IEEE; 2014. https://doi.org/10.1109/SERVICES.2014.83
33. Sahal R, Breslin JG, Ali MI. Big data and stream processing platforms for Industry 4.0: requirements mapping for a predictive maintenance use case. J Manuf Syst. 2020 Jan;54:138-51. https://doi.org/10.1016/j.jmsy.2019.11.004.
34. Ismail A, Truong HL, Kastner W. Manufacturing process data analysis pipelines: a requirements analysis and survey. J Big Data. 2019 Dec;6(1):47. https://doi.org/10.1186/s40537-018-0162-3.
35. Adolphs P, Epple U. Status Report Reference Architecture Model Industrie 4.0 (RAMI4.0). [Online]. 2015. Available: www.vdi.de.
36. Lu Y, Xu X. Cloud-based manufacturing equipment and big data analytics to enable on-demand manufacturing services. Robot Comput Integr Manuf. 2019 Jun;57:92-102. https://doi.org/10.1016/j.rcim.2018.11.006.