IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 17 July 2023, accepted 10 September 2023, date of publication 15 September 2023, date of current version 21 September 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3316116

RESEARCH ARTICLE

A Smart Factory Architecture Based on Industry 4.0 Technologies: Open-Source Software Implementation

JESUS ANSELMO FORTOUL-DIAZ[1], LUIS ANTONIO CARRILLO-MARTINEZ[1], (Senior Member, IEEE), ADOLFO CENTENO-TELLEZ[1], FROYLAN CORTES-SANTACRUZ[1], IVAN OLMOS-PINEDA[2], AND ROBERTO RAFAEL FLORES-QUINTERO[1]

[1]School of Engineering and Sciences, Tecnológico de Monterrey, Monterrey 64849, Mexico
[2]Facultad de Ciencias de la Computación, Benemérita Universidad Autónoma de Puebla, Puebla 72592, Mexico

Corresponding author: Luis Antonio Carrillo-Martinez (acarrillom@tec.mx)

This work was supported in part by Novus (under Grant PEP PHHT085-22ZZNV061), Tec Labs, Tecnologico de Monterrey, Mexico; and in part by the Research Focus Group on Cyber-Physical Systems, Smart Factory Laboratory, Mechatronics Department, Puebla Campus, Puebla, Mexico. The work of Jesus Anselmo Fortoul-Diaz was supported by the Tecnologico de Monterrey Postgraduate Scholarship Program and the Mexico Science Council Consejo Nacional de Humanidades, Ciencias y Tecnologías (CONAHCYT) under Grant CVU 655053.

ABSTRACT The Smart Factory has been a concept studied during the last decade that has not been standardized yet; for this reason, the academy and industry have developed a wide variety of new architectures that describe the integration of elements for digitization and interconnection. The present research aims to introduce a new architecture proposal for migrating traditional (automation) to smart (digitization) factories, implemented through open-source software. The proposed architecture is integrated, for the first time, by the interconnection of six main elements: cyber-physical systems, edge computing, artificial intelligence, cloud computing, data analytics, and cybersecurity; the research describes in detail their definitions, sub-elements, the interconnection between elements, and the minimum requirements for implementation. The test of the proposed smart factory was done through a scale smart factory pilot testing for a pick and place process, where the assembly of wood pieces from the geometric Tangram's puzzle was required; for this reason, the pilot testing includes a six-degree-of-freedom robot arm, a conveyor, a vision system, and a storage area. The case study conducted in this research allowed the assembly of four puzzles (fish, house, rocket, and swan) that were assembled with four different batches of pieces. The implementation allowed testing flexibility and adaptability. The final assembly reports included the status of assembly, the number of pieces assembled, the number of pieces stored, the assembly sequence, and the assembly time. Similarly, the development of the SCADA system allowed asset control as well as asset monitoring. The KPIs of the assembly process measured productivity (OTD) and time tracking (ATCT and TA) of the 16 tests, founding that the interconnection and digitization of the scale manufacturing cell were fully integrated and allowed repeatability; the proposed SF architecture represents an alternative for the small and medium automated factories to achieve interconnection and digitization, and it is ready to be tested in a more complex scenario.

INDEX TERMS Architecture, Industry 4.0, open-source, pilot testing, smart factory.

I. INTRODUCTION

The concept of a Smart Factory (SF) has been studied with greater interest in the last decade because the Traditional Factory (TF) architecture does not allow flexible and autonomous tasks using the same facilities. Although the SF concept is not standardized, Radziwon et al. [1] defined an SF as a manufacturing solution that provides flexible and adaptive production processes, to resolve problems in a production facility with dynamic and changing conditions;

The associate editor coordinating the review of this manuscript and approving it for publication was Hailong Sun.

the main characteristics required in an SF are flexibility and adaptability, considering an agile and lean model with a low-cost implementation. Also, Burke t l. [2] defined the SF as a flexible system that learns from new conditions in real-time, adapts, and runs entire production processes. Herrmann [3], and Petit et al. [4] agreed that SF takes advantage of digital technologies, with the main idea of achieving asset efficiency, quality, low costs, safety, and sustainability.

In the same way, the concept of Smart Manufacturing (SM) has meant an actual topic of study, it involves a collaborative manufacturing system that responds to changing conditions of the supply network, the factory, and the customer needs [5]. The key elements of smart manufacturing include intelligent products, intelligent equipment, intelligent factories, and intelligent supply chains [6]. As it is presented in the review of Haricha et al., SM is a new topic that needs to be explored through the technical advances as well as the challenges that present, mainly in topics related to interoperability, large amounts of data, obsolete SM production lines, and SM systems complexity [7].

Specifically, the difference between smart factory and smart manufacturing lies in the fact that the first one (SF) refers to intelligent and highly digitized installation, that uses connected devices and real-time data to optimize production processes and improve efficiency [8]; it is focused on integrating technologies within the factory to develop a flexible and adaptable environment. On the other hand, the SM is related to the collaborative manufacturing systems that respond to changing conditions of the supply network, the factory, and the customer needs, so the whole manufacturing ecosystem is involved in the process (from suppliers to customers) [6], [9].In particular, it is important to understand the position where the SF takes place in the fourth industrial revolution, and according to the prior art, the smart factory can be seen as an essential part of smart manufacturing; in the same way, smart manufacturing is considered a subset of Industry 4.0 (I4.0).

The key to migrating from the Traditional Factory (rigid process production) to a Smart Factory (flexible and autonomous tasks) requires that all the connected components send the information in real-time to achieve digitization without requiring extra budget. In addition, routine tasks based on artificial intelligence (AI) control the autonomous systems to improve productivity, deal with quality issues difficult for people to detect, and incorporate made-to-order/mass-customization capabilities [4], [10].

In consequence, the update from traditional to smart factories is difficult to achieve for emerging economies, newly industrialized countries, or specialized manufacturing service countries [11], [12]; this implies that Small and Medium Enterprises (SMEs) require to invest more resources in technology to not become obsolete and unproductive. The research of Jung et al. presented the main obstacles to implementing the SF for SMEs including i) financial burden (22.4%), ii) lack of technology (21%), iii) lack of big data (14.1%), iv) lack of cooperation with related companies (14.7%), v) demand of regulatory improvement (6.5%), and vi) others (21.3%) [13]. An SF with open-source software should solve the financial burden and lack of technology (43.4%).

In brief, most SF architectures have been proposed theoretically using schemas or diagrams for possible realizations or simulations; only a few implementations with a real application and testing are reported in the literature. Firstly, some architectures were based on industrial technology standard devices (PLCs, Gateways, Servers, Sensors, Actuators, HMIs, or Smart Devices, RFIDs) [14], [15], [16], [17]. Secondly, it was based on communication protocols such as Industrial Ethernet, Profibus, OPC UA, HTTP, MQTT, AMQP, CoAP, XMPP, [18], [19], [20], [21]. Thirdly, it was based on software and platforms like Visual C#, ASP.NET, Factory IO, self-developed REST APIs, Thingworx, [5], [17], [22], [23]. Fourthly, it was based on cloud services such as Azure, IBM, AWS, or GCP, [24], [25], [26]. Finally, it was based on different architectural topologies as centralized, collaborative, connected, or distributed, [27], [28], [29].

SF proposals using open-source software represent an open opportunity for research and industrial application; as an example, Ahn et al. introduced a framework that relates the cloud and fog computing using open-source tools like OpenStack (cloud service infrastructure) to achieve data analytics and information displayed through virtual machines, [30]. Similarly, Kim et al. presented a comparison between different parameters of open-source IIoT platforms (such as Kaa, Sitewere, DeviceHive, and Fiware) like the communication protocol, language, integration, and encryption, among other parameters, [31]. In the same way, Pipan et al. studied the benefits of integrating the distributed manufacturing nodes to enable the customization of production and manufacturing processes through open-source software and IIoT SCC (Single Chip Computer), [32]. Different research has been presented where basic SCADA systems have been developed through platforms such as Node-RED, for testing systems based in open-source software like [33] and [34], or testing specific communication protocols and monitoring the interaction (Modbus and MQTT) [35]. Additionally, Li et al. presented an open-source MES (Manufacturing Execution System) framework that also integrates distributed components with the industrial standard ISA95 (integration of enterprise and control systems), [36]. Furthermore, Waters et al., integrated an open-source IIoT solution applied to the monitoring of gas waste, integrating the data from the Operational Technology (OT such as IO-Link or Raspberry Pi) and the Information Technology (IT such as Python Dashboards through Plotly library) using different communication protocols like Fieldbus, OPC-UA, and MQTT, [37].

On the other hand, the integration of digital technology has generated new patents centered only on solving a particular issue of the SF, and a full SF architecture has not been patented. As an example, Kwon and Song [38] patented a method for secure data processing in the SF using secure gateways (encryption algorithms such as AES, RSA, and DES) to control the data flow coming from PLCs and IoT devices; the decoupling of network technology requires industrial networks (Ethernet, Profinet) and industrial protocols (OPC UA) to reduce vulnerabilities in the system. Similarly, Kim and Dong [39] proposed an SF service based on 5G using an integrated server (only in the Edge) to run applications of augmented reality (AR), computer vision, robotic control, and data analysis; the collected data are sent to a local cloud through a wire connection (using Time Sensitive Network standard) to store information and carry out Machine Learning (ML) predictions. Finally, OH et al. presented an SF architecture about risk monitoring and intelligent sensing [40], to establish mutual communication with production devices; it integrates sensors (IoT modules), a network, an AI server (automatic sensor recognition), a big data server, and a manager for intelligent devices (data transmission between systems).

In counterpart, the new products for the automation industry (technology developers) are currently in the research and design phase, or the first version released, and there are basic solutions in the market to achieve an SF. For instance, Siemens offers solutions like i) Mindsphere as an operating platform to connect industrial equipment and sensors to the cloud-based IIoT, ii) industrial and intelligence edge, and iii) IoT security technologies [41], [42]. In particular, Rockwell Automation provides a variety of hardware like i) smart sensors, ii) RFIDs, iii) HMIs, and iv) intelligent controls; in the same way to accomplish digitization, they use software like FactoryTalk, with different modules such as i) 3D modeling and design, ii) digital twin, iii) analytics, iv) edge ML, and v) network manager [43], [44]. Additionally, Bosch proposed the IoT Suite software platform that interacts with modules for digitization, including i) IoT Device management, ii) IoT Remote Manager, iii) IoT Edge Agent, and IoT Edge Services; they also developed the ctrlX Automation software that includes features like i) a Linux real-time operating system, ii) open standards and a comprehensive IoT connection, iii) platform for field communication using Ethernet and Profinet protocols [45], [46]. Equally important, Eaton and T-Systems in partnership developed an IoT platform for predictive maintenance (store, visualize, analyze data, and perform linear and square analysis with predictive scenarios for machines) based on Azure [47], [48].

According to the architectures, articles, patents, and industry solutions presented above, there has not been defined or realized a full Smart Factory proposal using open-source software. The first SF architectures integrated digital tools but lacked a defined structure for industrial implementation. Next, the researchers focused on the development of SF architectures that fulfill flexibility, reliability, or digitization but the structure of the architectures was not improved. As mentioned in the previous paragraph, the automation market only offers specific technological solutions for a particular application or issue, and the technological proposals presented represent an expensive technology solution for SMEs. The most recent investigations have developed hierarchical architectures that explain the components required for industrial implementation; they focused mainly on a specific problem and propose the industrial technology required to solve the issue. Data analytics and information security have also taken relevance in the structure of their architectures, but they required to be investigated deeper.

This research is focused on presenting a new architecture proposal for the full migration technology issue of the traditional (automation) to the smart factory (digitization), in a basic form, all this with open-source software, including six main elements (Cyber-Physical Systems, Edge Computing, Artificial Intelligence, Cloud Computing, Data Analytics, and Cybersecurity). Therefore, the tools applied to traditional factories can lead to the adoption of flexibility and autonomy of the Smart Factory, applying the interconnection and digitization of all devices in the facility to be able to withstand the industrial environment.

The paper's sections are organized as follows: Section II presents the literature review and the proposed architecture of the Smart Factory with the minimum requirements for implementation. Section III presents the Smart Factory approach related to the case study and the experimental setup, including technical information. Section IV describes the results of the SF implementation (assembly logs, SCADA, and KPIs), Section V presents the discussions of the research, and Section VI the conclusions achieved in this study and the future work.

II. SMART FACTORY ARCHITECTURE PROPOSAL

The first part of this section focuses on a literature review related only to SF architecture proposals in academic papers. Subsequently, the proposed SF architecture is presented, talking about the outstanding elements that compose the architecture; a detailed description of each one is given, including its definition in the state of the art, the sub-elements in the SF architecture, the relation between the prior art and the proposed element, and finally, the minimum requirements for the implementation using open-source software to visualize the interconnection and digitization of all devices in a scale smart factory pilot testing.

A. LITERATURE REVIEW

In recent years, a diversity of SF architectures, models with different schemas, and proposed academic and industrial applications have been introduced [49]. In particular, Kemény et al. developed an SF architecture (integrated by Hardware, Components, and Software) to test different I4.0 technologies in a scale testbed facility (platform for education and research); the learning factory aimed to reinforce the concepts on students and the skill development, applying the architecture on the SF laboratory at MTA SZTAKI, which included physical (PLC, Raspberry Pi, Arduino, cameras, Kinect, conveyor, workstation, robots, RFIDs, routers, FESTO devices) and virtual (web server, web interfaces, databases, low-level services) components [22]. Moreover, Shariatzadeh et al. proposed an SF architecture (integrated by Physical Layer, IoT Platform, and Product Life Cycle Management Platform) implemented through the Thingworx Java SDK platform using a random data generator to validate the digital factory and the SF integration [23]. The previous proposals were only simulated or used for an educational environment, and the SF elements were without a detailed description of the main components.

Alternatively, the research of Kaschel and Bernal [50], mentions that flexibility is divided into process flexibility (machine, job, volume, layout) and product flexibility (sequence, operational, processing) to achieve multiple types of products. Wang et al. designed an SF architecture (integrated by Physical Resource Layer, Industrial Network Layer, Cloud Layer, Supervision, and Control Layer) to emphasize the capability of multiple routing using flexible conveying, which consists of transferring products between any machines for automatic positioning to reconfigure production routes [14]. Similarly, Jung et al. developed an SF architecture (integrated by Edge, IIoT Platform, and Enterprise Software Services) presenting an order requirement scenario between two factories (factory A with a lack of products, factory B provides the required products to factory A), to test the flexibility, the adaptation of production capacities, and sharing of resources, assets, and inventory [15].

Further investigations include the development of hierarchical architectures that detail specific components required for the SF and the industrial settings for real implementation. Specifically, Chen et al. presented an SF architecture to explain the integration of manufacturing and services; the elements that integrate the architecture are the Physical Layer (modular units, reconfiguration, interface adapter, and software adapter), Network Layer (edge computing, OPC UA interconnection, and corporate internal operation), Data application Layer (knowledge management, ontology modeling, QoS management, and information evaluation), and Terminal Layer (monitor, maintain, design, and bill management). The architecture was applied in a laboratory platform for a candy packing line and it was monitored for six months to detect the main issues and challenges in the SF architecture implementation [16]. Additionally, Wan et al. presented an SF architecture based on four layers (smart device layer, network layer, cloud layer, and application layer) that are related to the physical smart manufacturing resources, industrial wireless sensor networks, cloud platforms, and services of system applications [51]. Moreover, Illa et al. proposed an architecture that integrates three key building blocks of the Smart Factory (Smart Equipment, Seamlessly Integrated Ecosystem, and Advances Analytics), so the framework proposed included five layers (Manufacturing Applications, Enterprise Applications, IoT Platform, Data Visualization and Control, and Security); they also compared three different approaches for the smart factory including the Open Source Software, Commercial Distribution, and Platform as a service; finally, they presented a guide to implementing IoT based solution technologies and use-cases [52].

Similarly, Okeme et al. proposed an SF architecture that integrated elements from the Manufacturing Application (MES dashboard, database, and Order system), Visualization and Control (3D Monitoring, CPS controller, and CPS Simulator), IoT (OPC UA, Edge, Platforms, and Enterprise), Digital Twins (geometry, dynamics, material properties, and model update), and Cyber Security (encryption, closed and protected systems). The SF architecture was developed in a simulated environment (Factory IO) to state the benefits of the SF adoption (efficiency, cost, quality, safety, and profitability). The Factory IO platform was also linked with tools like Siemens MindSphere, PLCs, and Matlab Simulink [17], [53]. Moreover, the model proposed by Kahveci et al. is a reference architecture with features such as security, interoperability, resilience, and scalability; it is built through five layers (Control and Sensing, Data Collection, Data Integration, Data Storage and Analytics, and Data Presentation) that are tested through an assembly battery pack case study, so the architectures serve as a platform for businesses (small and medium enterprises) [54]. In the same way, Hsu et al. proposed a Smart Factory architecture that includes four layers (Physical Resource Layer, Cloud Service Layer, Terminal Layer, and Network Layer), so the infrastructure of the factory can respond to the fast demand of the market; the technologies used to implement the architecture include Edge computing, Fog computing, Cloud Computing, and Blockchain, implemented through different devices like robot arms, Raspberry Pi, microcontrollers, cameras, PLCs, sensors, among others [55].

Recently, Lee et al. investigate the application of different technologies within the Smart factory of the automotive industry applied in cellular manufacturing, finding that the most important are: digital twins, additive manufacturing, AI-based monitoring, human-robot collaboration, and advanced technology for supply chain and logistics, the research also emphasizes the importance of the five levels of a smart factory framework, including digitization, connectivity, predictability and analysis, optimization and cognitive, and self-recognition and autonomous [56]. Abdelatti et al. present a lab-scale smart factory based on the Fischertechnik kit as part of the Industry 4.0 Learning Factory, but all interconnections including hardware, software, and protocols have been replaced by open components such as Arduino, sensors, Raspberry Pi, and open-source controllers and software, using the Robot Operating System (ROS) and the MQTT protocol, integrating a Human Machine Interface (HMI) with a SCADA system [57]. Ryalat et al. presented a Smart Factory architecture based on: Physical, Network, Data Application, and Terminal layers, explaining that the implementation of the SF can be through the following pillars of Industry 4.0: cyber-physical systems, the Internet of Things (IoT), big data analytics, cloud computing, artificial intelligence, and autonomous robotics. The architecture implementation was done through a case study about a drilling process with a Kuka robot, an S7-1200 PLC, and an IoT platform to explore the real-time diagnosis, control, and prediction; the authors finally conclude that it is necessary to explore and include specific pillars of the I4.0 such as cybersecurity and artificial intelligence, and the inclusion of human-machine interaction and collaboration [58].

Summarizing the presented prior art, the SF architectures integrated digital tools into traditional factory processes and carried out tests for concepts like SF and I4.0, but lacked a defined structure for industrial implementation. Next, the researchers focused their work on the development of SF architectures that fulfill flexibility, reliability, or digitization; they improved the industrial components used for the application, but the structure of the architectures was not improved significantly (they included mainly physical components, cloud services, and edge devices). The most recent investigations have developed hierarchical architectures that explain in detail the components required for industrial implementation; they focused mainly on a specific problem and propose the tools (based on industrial technology) required to solve the issue. Data analytics and information security have also taken relevance in the structure of their architectures.

In the same way, research about adaptability in the SF environment has not been fully explored; Horbach et al. define adaptability as the ability of a production system to change actively in response to external or internal triggers [59]. The research of Komoto et al. included the study of a simulation framework based on run-time adaptability, where they could simulate the dynamic changes of the functional requirements during product development [60].

The proposed research presents an SF architecture with flexibility and adaptability, that combines the previous characteristics (physical elements, cloud services, edge devices, and data analytics) with the addition of artificial intelligence (machine learning, deep learning, imitation learning) and cybersecurity (authentication, encryption, and secure connections). The most important feature is that the SF architecture implementation is fully integrated with open-source software, becoming an alternative method within the SF, more in particular for small and medium enterprises.

B. PROPOSED ARCHITECTURE

To accomplish the SF architecture, a prior art revision, functional tests, and real implementation were completed considering:

- review of definitions, SF elements, general concepts (industry 4.0, smart manufacturing, IoT, among others), theoretical and simulated SFs, real implementations, and future trends.
- functional tests for tools and technologies to include the best devices, protocols, software, and algorithms; discarding hard-to-implement components and technology not suitable for industrial environments or no longer supported.
- interconnection between elements and the minimum requirements for the application.
- a simple implementation carried out to validate the proposed SF architecture.

The architecture includes six main elements, and a brief description of each one is presented below.

1) Cyber-Physical Systems (CPS): it is an interface between the physical and the digital environment for data transformation (physical to digital, or vice-versa) using wired/wireless protocols to communicate acquisition boards, PLCs, sensors, and actuators. The IoT components send/receive information and instructions for automated devices through protocols (OPC UA, MQTT, HTTP, CoAP, AMQP, or DDS).
2) Edge Computing (EC): it executes processes near the source data, runs local (real-time routines, communicate with the Cloud and CPS), and distributed (replication of services and information) services through Edge Nodes (divide the workload and computing between different Edge devices).
3) Artificial Intelligence (AI): it performs the decision-making process and pattern recognition, executing Deep Learning (biological process replication), Imitation Learning (human actions carried out by automated devices), or Machine Learning (data classification).
4) Cloud Computing (CC): it is a set of configurable computing resources that require minimal resource settings to allow the execution of services like the Message Queue Telemetry Transport (MQTT) Broker Server, IoT Platform, Databases, or architectural services.
5) Data Analytics (DA): it requires a data pre-processing step, to select the useful information; it allows the visualization of information through dashboards in real-time and the elaboration of statistical (Mean, Mode, Standard Deviation) or analytical (Linear Regression, Predictive Analytics) reports with a summary of current information and future trends.
6) Cybersecurity (CS): it carries out information protection and secure communications between elements through encryption/decryption algorithms and identity validation; it can be applied to all the devices and protocols within the SF.

To situate the proposed architecture in the SF's road map and the prior art, Table 1 presents a comparison of the elements included in the SF architectures of the related work, as well as the advantages and disadvantages of each one. As it is observed, the previous architectures were focused only on the physical components, cloud connections, and process digitization. In general, the two major contributions of this SF architecture proposal are i) the integration of open-source software (compatible with the automation and control hardware) as an equivalent option for the components offered by the market or industry sector, and ii) the incorporation of cybersecurity and artificial intelligence to

TABLE 1. Comparison between the Smart Factory architectures studied in the prior art with the proposed architecture.

Related work	Elements that composed the SF Architecture	Case study / Tested	Advantages	Disadvantages
[22]	Hardware, Components, and Software.	Yes	Reinforces the CPS and I4.0 concepts to students. A learning factory developed with the help of higher education students.	The architecture was tested in facilities with dimensions and capabilities not closer to a full-fledge learning factory.
[23]	Physical Layer, IoT Platform, and PLM	Yes	Addresses the issue about the digital factory and the smart factory integration. Implemented using the Thingworx platform, and the HTTP protocol as wireless communication.	Security issues are not addressed in the solution proposed for the interoperability.
[14]	Physical Resource Layer, Industrial Network Layer, Supervision and Control Layer, and Cloud Layer.	Yes	Emphasizes the capability of multiple routing and flexible conveying in the smart factory. Explains the differences between the traditional production line and the smart factory production system.	A deep analysis of enabling technologies is needed in future work.
[15]	Edge, IIoT Platform, and Enterprise Software Services	Yes	Presents a four-phase flexible adaptation for the Smart Factory Web. The tested scenario explains the order requirements between 2 factory suppliers.	Security profiles are not reviewed as part of the architecture to ensure secure communication between sites with managed user roles.
[16]	Physical Layer, Network Layer, Data application Layer, and Terminal Layer.	Yes	States the main issues and challenges that are present in the layers. The laboratory platform was tested for six months to obtain the overall effectiveness.	The compound talents and multi-field cooperation is required for the implementation of the smart factory.
[51]	Application Layer, Cloud Layer, Network Layer, and Smart Device Layer	No	Presents an integration of technologies such as IoT, Cloud computing, and Artificial Intelligence. Gives an explanation of AI applications at different layers of the architecture.	Deep research on AI and cross-discipline is needed in future work.
[52]	Manufacturing Applications, Enterprise Application, IoT Platform, Data Visualization, and Control	No	Compares three approaches: Open Source Software, commercial distribution, and Platform as a service. Presents a guide to implementing IoT-based solution technologies and use-cases.	An increase in the implementation description is required to better understand the scope of the proposed architecture.
[36]	Manufacturing Layer and Control Layer	Yes	Proposed a flexible MES architecture according to the ISA 95 standard. The architecture is based on the Odoo platform to provide an integration solution.	The scalability of the industrial ISA 95 standardized data architecture and data models are not presented.
[53]	Manufacturing Application, Visualization and Control, IoT, Digital Twin, and Cyber Security.	Yes	Presents reference architectures for IoT, Digital Twin, and Smart factory. States the benefits of smart factory adoption: efficiency, cost, quality, safety, and profitability.	Cyber secure and interoperability issues and challenges could avoid a good architecture implementation.
[54]	Control and Sensing Layer, Data Collection Layer, Data Integration Layer, Data Storage, and Analytics Layer, Data Presentation Layer.	Yes	Proposes and implement a scalable reference architecture as a platform for small and medium enterprises. The architecture is tested through an assembly battery pack for electric vehicles, at the University of Warwick.	Hibrid or cloud-based implementations are not included in the study; better data maintenance and data down-sampling methods are required to be explored.
[55]	Physical Resource Layer, Cloud Service Layer, Terminal Layer, and Network Layer.	Yes	The infrastructure of the factory can respond to the fast demand on the market. The technologies implemented include Edge computing, Fog computing, Cloud Computing, and Blockchain.	The Blockchain network architecture is not described in detail.
[56]	Digital Twins, Additive Manufacturing, AI-based Monitoring, Human-Robot Collaboration, and Advanced Technology for Supply Chain and Logistics.	No	Emphasizes the importance of the five levels of a smart factory framework, including digitization, connectivity, predictability and analysis, optimization and cognitive, and self-recognition and autonomous.	The research is aimed at the cellular manufacturing systems of the automotive industry.
[58]	Physical Layer, Network Layer, Data Application Layer, and Terminal Layer.	Yes	The architecture is implemented through a drilling process implementing industrial components such as a Kuka robot and an S7-1200 PLC.	Deep research on cybersecurity, AI and human-machine interaction and collaboration is required as future work.
This work	CPS, Edge Computing, Artificial Intelligence, Cloud Computing, Data Analytics, and Cybersecurity.	Yes	Integration of the digitization process with the CS and AI using Industry 4.0 technologies. Implemented architecture using open-source software, and compatible with the components of the industry sector tools.	The architecture is presented for small and medium enterprises, scalability tests are required for future work.

the four main components (CPS, EC, CC, and DA) used in the previous architectures. These features make the proposed SF architecture the first in integrating the six elements (CPS, EC, AI, CC, DA, and CS) and implementing them using open-source software; all the features are summarized in Fig. 1, it shows how the six elements that conformed to the proposed smart factory are interconnected, located, and related so that anyone can reproduce the smart factory using open-source software tools, with the minimum requirements.

Functional tests for technological tools (devices, protocols, software, platforms, algorithms, and services) were performed to achieve the minimum requirements to implement the proposed SF architecture. As a result, Fig. 1 presents the components and interconnection between the six elements to achieve digitization with basic requirements. The following subsections describe in detail each element of the SF architecture, as well as the sub-elements and components.

1) CYBER PHYSICAL SYSTEMS

The CPS concept was first introduced in 2006 by West and Parmer to define real systems based on a software architecture [61]. The CPS represents the intersection of the physical and the cyber environments, which means the integration of computation with physical processes. Consequently, the CPS integrates computing, communication, and storage capabilities with monitoring and controlling physical world entities [62]. The CPS can be configured as independent and autonomous; these characteristics are the basis of a Smart Factory [3], the key components of Industry 4.0, and the digitization processes [63]. According to Jamaludin and Rohani, the main CPS characteristics include the physical system, cyber and information system, heterogeneous (integration and interaction process between the cyber and physical) system, and security requirements (including real-time capability and predictability) [64].

As revealed by Fig. 1, the CPS element of the proposed SF architecture requires the integration of two main sub-elements: i) physical, that is composed of automated devices (conveyors, robots, vehicles, etc.), actuators (stepper motor, servomotors, pneumatic devices, etc.), sensors (temperature, pressure, proximity, etc.), and acquisition components (microcontrollers, PLCs, etc.), ii) IoT, that uses the MQTT protocol (defining the Quality of Service, Port, and IP Direction) to exchange information through messages (topic and payload) between clients (publisher and subscriber).

The proposed CPS element includes the four characteristics presented by [64]. The physical, cyber and information, and heterogeneous systems are developed with hardware like PLCs, microcontrollers, IoT devices, sensors, or actuators. Finally, the security requirements are achieved through secure channels (SSH and secure ports) based on the MQTT protocol to allow the connection with the EC and CC, additionally, the message payloads are encrypted/decrypted (via Advanced Encryption Standard (AES) algorithm). These components are integrated with automated devices through shields that use industrial protocols (Ethernet, CAN, Modbus) to digitize physical processes.

The application of the CPS element in Fig. 1 includes the integration of both the Physical and the IoT sub-elements; they interact in a process that starts in the Automated Devices, the process continues to the Sensors located throughout the facility, then the digital information is transferred to the IoT components (data acquisition boards, PLCs, gateways) through different wired/wireless communication protocols (serial, I2C, ethernet, SPI, Modbus, CAN, OPC UA, ZigBee, LoRa) to realize the information preprocessing and data encryption. In this step, the Message (topic and payload) is built and subsequently sent to the Cloud through the MQTT protocol. The feedback from the Edge uses different communication protocols through the Downstream connection and includes information on the control Routines (position, feedback, emergency stop) for the Automated Devices (sent directly to the Actuators or through Motor Shields).

2) EDGE COMPUTING

In the opinion of Khan et al. Edge Computing is a new paradigm that performs computing to process, analyze and store information for knowledge generation near the data source at the edge of the network [65] and closer to the devices to reduce traffic and communication bandwidth using the upstream channel (data travels from the data source to the cloud), and the downstream channel (information is sent from the cloud to the IoT devices) [66]. EC takes responsibility for specific tasks and virtualizes (generates a copy) the server's capabilities so that it can be considered an extension of the cloud [19]. In addition, the EC requires a set of autonomous devices (edge nodes hierarchically distributed as edge gateways, edge controllers, edge clouds, and edges) to execute distributed computing services and specific tasks (storage, processing, visualization) [67].

The EC element of the proposed SF architecture (see Fig. 1) requires the sub-elements: i) Edge Nodes and Edge Devices, in charge of local services (real-time computing, upstream and downstream communication) with the cloud, and downstream communication with the IoT devices, and distributed services (intelligent services, processing, storage, and data visualization), ii) Automation, to integrate automated devices (robots, conveyors, vehicles) and visual components (cameras and vision devices), and iii) Control routines (feedback, position, and emergency stop).

The EC element satisfies the characteristics described in the literature review because the computing services are executed near the data source (local and distributed). The EC implementation includes devices like Raspberry Pi, Jetson Nano, Nano Pi, among others that run open-source software and services, and the processes that require higher computing resources (automation and control) are executed in GPU environments. Distributed computing allows data replication at the Edge nodes, so it is always available from the CPS or the Cloud elements.

The implementation of the EC element in Fig. 1 combines the execution of local and distributed services. First, the Local Services execute the Real Time Computing process (implemented using Python scripts) to integrate all the information from the Routine models (algorithms to calculate the Position of the Automated Devices, receive Feedback,

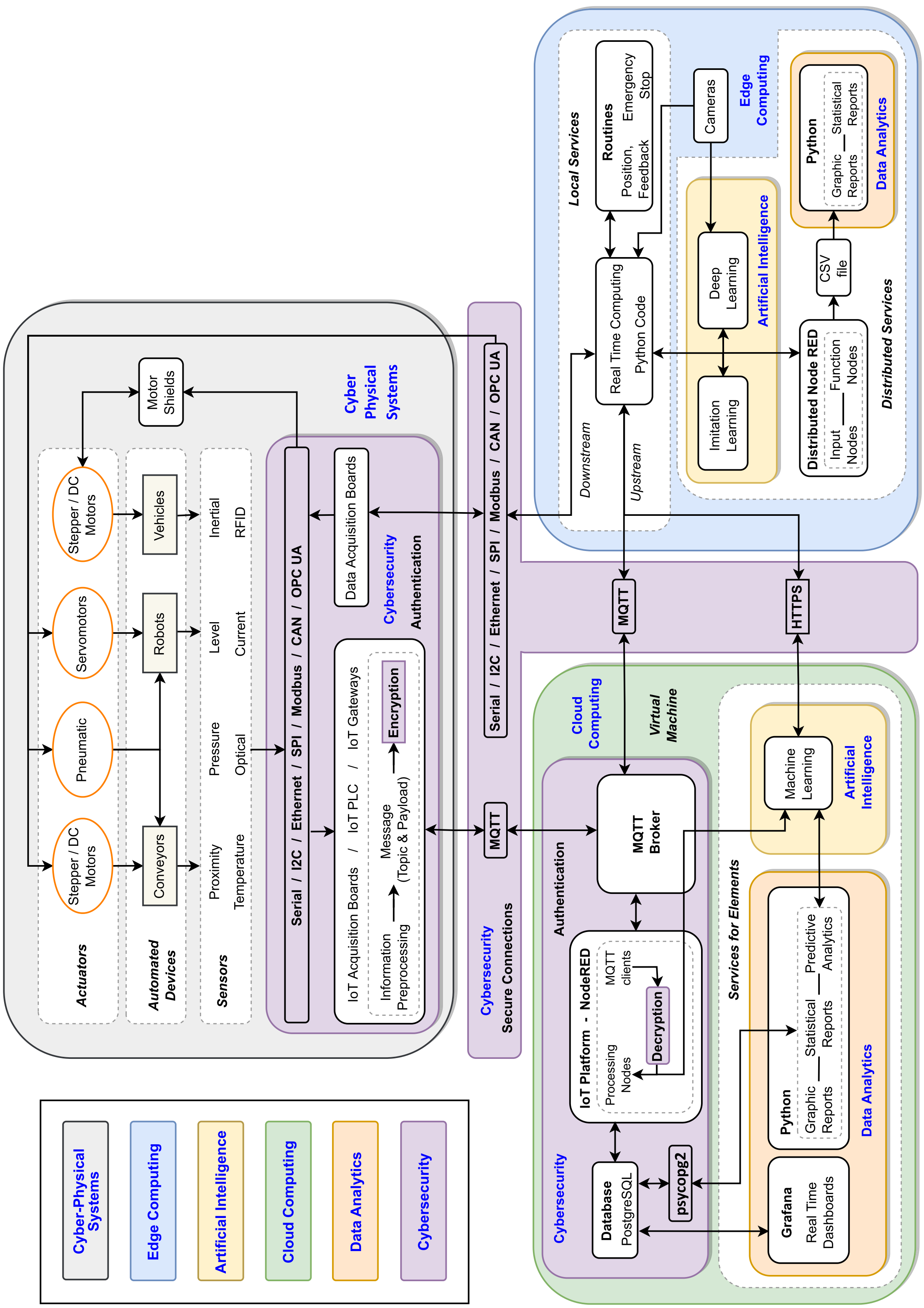


FIGURE 1. **The elements for the proposed Smart Factory architecture are indicated in different colors and the minimum requirements for implementing the proposed SF architecture are based on open-source software.**

and activate the Emergency Stop), the AI models (Deep Learning and Imitation Learning), and the Cloud; The edge devices communicate through the Downstream (exchange information with the CPS and Automated Devices) and

Upstream (exchange information with the Cloud) channels and the Distributed Services are implemented through the Distributed Node-RED (DNR) platform for data replication (storage of local information in CSV files from the AI and DA services); the Edge devices also maintain communication with Cameras to perform object detection or segmentation.

3) ARTIFICIAL INTELLIGENCE

Artificial Intelligence is a branch of computer science that researches the development of simulated human behavior like natural language processing and image or speech recognition [68]. AI studies intelligent agents that can achieve goals and perform tasks based on stipulated rules and algorithms [69]. As explained by Jakhar and Kaur, Machine Learning is the main AI subset in charge of data classification without being programmed [70]. At the same time, Deep Learning is a Machine Learning subset that develops nonlinear models to replicate human brain processes. In counterpart, Imitation Learning is a set of techniques, part of the human-AI interaction, that mimics human behavior in a given task [71], [72].

In particular, Machine Learning (ML) studies the efficiency of models that learn, adapt, and find complicated hidden patterns through iterative processes [70], [73]. According to Ashri [69] and Zohuri and Rahmani [74], the ML categories include: i) supervised learning (the training data includes the input and class), ii) unsupervised learning (a set of variables without a specific class), and iii) reinforcement learning (an agent interacts with the environment through actions, receiving a reward). Additionally, Deep Learning (DL) integrates computational models that imitate the architecture of biological neural networks through Artificial Neural Networks (ANN) [70], [74]. ML requires a massive training corpus to improve the ANN accuracy [70], and the DL model learns features to solve problems in fields like computer vision or language processing [68]. Alternatively, Imitation Learning (IL) is applied to emulate complex human behaviors [75], so the agent (something that acts) learns by observing the expert's demonstration, and the skills are generalized to unseen scenarios through methods like Behavioral Cloning (BC) or Inverse Reinforcement Learning (IRL) [76], [77].

Consequently, the AI element shown in Fig. 1 integrates the sub-elements of: i) Machine Learning techniques related to supervised and unsupervised learning to carry out basic decisions (decision trees, support vector machines, regressions), ii) Deep Learning for the replication of brain processes (visual recognition, or natural language processing), and iii) Imitation Learning to achieve complex decision-making processes where the changing environment affects the factory behavior (IRL, BC). All subsets of the AI are included in the architecture.

The proposed AI element replaces the expert's experience required in the SF process using a variety of models (ML, DL, IL) developed using open-source libraries (E.g., Python). The AI element is executed in both the Edge and the Cloud, running the code and algorithms to obtain a better performance (E.g., in GPUs, CPUs, or microcomputers, as long as the hardware allows it). To transform the automated devices into intelligent agents, it is required to train the devices with the actions and trajectory movements to replicate the expert's behavior.

The implementation of the AI element, shown in Fig. 1, requires the interaction of different models running in the Edge and the Cloud. The Edge executes two processes; first, Deep Learning implements visual recognition systems (feature extraction and classification) through trained models, and second, Imitation Learning implements models that interact with humans and the changing environment (the agent receives the states and actions of the demonstrator as training data, and then replicates the expert's actions). Both processes are implemented through Python, and the results are stored in the local devices. In comparison, the Cloud executes the Machine Learning for data classification, using relevant information received from the Edge (HTTPS) or the CPS (MQTT); once the model returns the prediction, the result is sent, as feedback, to the DA element for visualization and reporting.

4) CLOUD COMPUTING

Cloud Computing is a technology where different Information Technology (IT) services are provided by massive low-cost computing units connected by Internet Protocol (IP) networks [78]. According to Marwan et al. the National Institute of Standards and Technology (NIST) defines Cloud Computing as a model for enabling on-demand network access to a shared pool of configurable computing resources (networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction [79]. According to Birje et al., the four deployment models include public, private, hybrid, and community cloud. The most popular platforms include Amazon Web Services, IBM Blue Cloud, Microsoft Azure, and Google Cloud Platform (GCP) [24].

The CC element in Fig. 1 presents the following sub-elements for the SF: i) MQTT Broker Server (HiveMQ, CloudMQTT, Eclipse Mosquitto) that exchanges information from clients (publisher and subscriber) considering the topic message, ii) Database for information storage, it can be classified as relational (MySQL, Oracle, SQL Server, or PostgreSQL) or non-relational (Mongo DB, Cassandra, Neo4j) databases, iii) Services for elements, that host the components required to execute the DA and the AI in the cloud (Grafana, Phyton), and iv) IoT Platform, to process and visualize information (Google Cloud IoT, AWS IoT, Oracle IoT, Cisco IoT Cloud, Microsoft Azure IoT, Node-RED, Thinkspeak, or Thinger.io).

The CC element aligns with the NIST definition because it represents a rapid implementation of open-source software (servers, storage, services, network) customized to the users' requirements. It allows the easy management of new Virtual Machine instances, containers (Docker), and services, so they are always running (uninterrupted services). The CC allows the execution of open-source software through Docker as an alternative to the licensed one, so the same applications can run in the cloud to execute the required tasks. In case a service is not available, the handling containerized application (Kubernetes or Docker Swarm) can switch to the next instance to continue executing the process required.

Implementing the CC element in the pilot testing (see Fig. 1) requires the previous configuration of a Virtual Machine (VM) to host the necessary services. Once the VM is configured, the open-source software is installed, the application ports are opened, and finally, the services are initialized. The MQTT Broker Server allows communication with the Cloud and the information exchange between the IoT Platform, the CPS, Edge, and the AI elements. The IoT Platform (Node-RED, ThingSpeak, Thinger.IO) receives the data to decrypt/encrypt and process the information to subsequently send it to the Database for storage (PostgreSQL, MySQL, MongoDB, among others). The Services for AI (Machine Learning models) and DA (open-source software like Grafana, or Python scripts) elements are also running in the Cloud to execute complex processes, like Machine Learning or Predictive Analytics.

5) DATA ANALYTICS

Data analytics is the extraction of useful knowledge to discover correlations and estimations of likelihood and error; the previous steps in the DA include acquiring, preparing, and integrating new information with existing data [80]. Once the data are collected, the next step is the Exploratory Data Analysis, which involves creating data visualization to detect anomalies (duplicates, errors, or outliers) in the dataset [81]. As mentioned by Richmond, an essential step in DA is to calculate the statistical indicators (mean, median, standard deviation, or variance) to present the main data correlations and distribution [82]. Most recently, DA requires the development of statistical and computer models to create impactful predictions (predictive models) over a relevant variable [83], using software packages that include open-source libraries (like R, Python, Matlab, SAS, Orange, or Weka), [84].

According to Fig. 1, the DA requires the data collection and preprocessing as previous steps; once these steps are completed, the sub-elements of DA can be applied for: i) data consulting, through secure channels to request the information of the Database (language adapters for Python or NodeJS), ii) graphic visualization, for information display in dashboards (Python, Grafana, Node-RED), and iii) statistical reports, including central tendency measures and models of future trends (linear regressions and predictive analytics).

The proposed DA element includes the previous steps explained by Brodie, and subsequently the exploratory analysis (graphical visualizations) and tools for the report elaboration. The open-source software allows the display of information in real-time through dashboards (charts, gauges, histograms). The integration of open-source libraries in Python (Pandas, Numpy, Scypy, Matplotlib, among others) enables easy computation to obtain the central tendency measures and future trend reports through predictive models.

For instance, the DA element in the SF (see Fig. 1) requires to be executed in the Cloud and the Edge. The DA in the Cloud uses the information stored in the Database, and the connection is achieved using database adapters (psycopg2 for Python). Once the connection is created, the data is consulted by Grafana to display the real-time dashboards; Python scripts generate statistical (mean, median, standard deviation) and graphic (histograms, bar, time-series) reports, and predictive analytics (likelihood of future trends). The DA in the Edge uses information from local process (AI models and routines) and replicate the information using the DNR as a redundant system (if a device does not work suitable, a redundant device avoids data loss), storing the information in CSV files, to subsequently use the files for graphic and statistical reports.

6) CYBERSECURITY

Cybersecurity protects data centers from unauthorized access, cyber-attacks, or identity theft to maintain the integrity, availability, and security of data [85], [86]. It is also required to guarantee information confidentiality, and detect online threats and vulnerabilities [87]. As it is mentioned in the research of Halenar, industrial systems tend to be more vulnerable than information systems, for this reason, it is required to use powerful protections, and this is due to new standards implemented in automation, the heterogeneous infrastructure of modern facilities, minimal frequency updates, among others [88]. Nowadays, cybersecurity centers on the security risks of IoT assets due to the increase of objects connected to the IoT [20]. According to Lu and Xu, the mechanisms required to protect IoT assets include lightweight encryption, authentication, and access control [89]. The IoT cybersecurity technology provides device authentication, secure communications, data encryption, and secure software to prevent security issues like inadequate authentication, insufficient audit mechanisms, or low security in a protocol implementation [20].

The CS element of the proposed SF architecture (see Fig. 1) requires the sub-elements: i) authentication, implemented through private and public key definitions or by credentials (users and passwords), ii) encryption/decryption, for data codification through complex algorithms to protect the information (AES, Hash functions), and iii) secure connections, to create reliable communication channels for information exchange (SSL, TLS).

As observed, the proposed CS element includes the cybersecurity technology for IoT protection, presented by Lee. The open-source software allows secure connections between the broker, IoT platform, database, and graphical tools. The definition of strong passwords, the use of private and public keys, and data encryption are essential tools for maintaining the integrity and confidentiality of the information.

The CS element is implemented (see Fig. 1) using secure connections in the communication protocols (MQTT, HTTP) to guarantee that the information is protected when it is transferred between elements. Authentication is applied in the open-source software running in the Cloud (broker server, IoT platform, and DB), requesting the user and password for logging. Finally, the encryption in the IoT elements and the decryption in the IoT platform is implemented using the AES (Advanced Encrypted Standard) algorithm (symmetric block cipher) to code and decode messages using a secure 16-byte length key and a 16-byte length initial vector for the Cipher Block Chaining (CBC) mode to increase security.

III. THE SMART FACTORY PILOT TESTING

This section explains the case study, which consists of a Tangram puzzle assembly process achieved through pick-and-place tasks within the smart factory pilot testing. The relevance of solving the Tangram puzzle is the versatility to solve different figures (combination of geometric shapes, sizes, and colors), not necessarily the same solution each time; it offers diverse configurations (none of the pieces should remain unused, moreover, they should not overlap) to test the flexibility and adaptability of the SF. Also, solving the puzzle includes pick and place, assembling, and manipulation processes that are common in the robotics industrial environment. In addition, the experimental setup is stated, which includes a detailed explanation of the hardware and software implementation used in the scale smart factory pilot testing.

A. CASE STUDY

The implementation of the proposed SF architecture (see Fig. 1) was achieved through a scale smart factory pilot testing; this case study aimed to test the interaction between the elements of the proposed SF architecture, obtaining as deliverables i) assembly reports with information of placement sequence, parts in storage, assembly time/success, and missing pieces; ii) a basic Supervisory Control And Data Acquisition (SCADA) system for supervision and control of the SF pilot testing (processes real-time information, display logs of historical data, control automated processes, connect with remote devices); and iii) Key Performance Indicators (KPIs), that are tools for measurement, comparison, and monitoring of the state of the process with respect to a defined goal [90], some examples include the productivity (OTD) and time tracking (ATCT and TA).

The manufacturing cell included a Wlkata Mirobot arm (six-degree-of-freedom scale robot), a Wlkata conveyor, a vision system, and a storage area (see Fig. 2-a). The scale smart factory pilot testing performed a basic pick and place process; four geometric tangram puzzles (house, fish, rocket, and swan exemplified in Fig. 2-b) are assembled to test the flexibility characteristic (randomize assembly sequence) of the SF; the different locations of the pieces in the puzzles with respect to each other gives the SF the ability to make a variety of products with the same equipment (flexible), as mentioned by Lafou [91].

The shapes allowed in the assembly puzzle were five triangles (two small, one medium, and two big), one square, and one rhomboid. To test the adaptability characteristic, with the use of Deep Learning, the SF detected repeated pieces and the impostor shapes (hexagons and circles, that are not included in the tangram puzzle) included on intention in the batch, to take them out of the assembly and place on a pallet for storage in the warehouse zone by the robot.

In the Graphic User Interface (GUI), the user selected a target puzzle, and a specific batch was sent to the robotic arm through the conveyor. The cycle started when the first piece of the batch was placed on the conveyor, and the RFID sensor read the tag that indicated the beginning of the assembly; the batch information (ID, number of pieces, and shapes) was consulted in the cloud database to predict the assembly success through an ML model; if the model predicted that the assembly could not be completed, all parts were sent to storage, and the SF request a new batch to complete the solicited puzzle or solicited a new puzzle to assemble.

Subsequently, the conveyor moved the pieces to the robot's vision workspace; the vision system, integrated by a Camera and the Real-Time Computing (RTC) at the Edge, detected the piece's contour, centroid, and orientation. Simultaneously, the Deep Learning model identified the piece's shape, size, and position through a Convolutional Neural Network (CNN). The RTC ran the routine scripts (Python) for the automated devices (robot and conveyor position, visual feedback, emergency stop), the Deep Learning model (shape, size, and position), the Imitation Learning model (mapping of the piece's centroid to the robot cartesian coordinates), and the Cloud information (data exchange using MQTT and HTTPS). The result of the calculated position was sent to the robot through the downstream channel, so the piece was picked with the robot's end-effector (suction cup).

To achieve adaptability, the placing task evaluated two possible scenarios depending on the DL prediction; if the piece was required, the robot placed it in the working area for assembly; if the piece was not required (impostor or repeated), the robot placed it on a pallet for storage. No matter which scenario was achieved, once the piece was placed, the cycle ended and the process was repeated until the final piece of the batch was detected through the RFID reader and the puzzle was completed.

At the same time that the assembly process is executed, all the information from the sensors (RFID, current, inertial forces), the AI model results (Deep and Imitation Learning), and Python script results were sent to the cloud for processing and storage using the upstream communication. The IoT boards encrypted the message (topic and payload) using the AES algorithm and sent the message to the MQTT

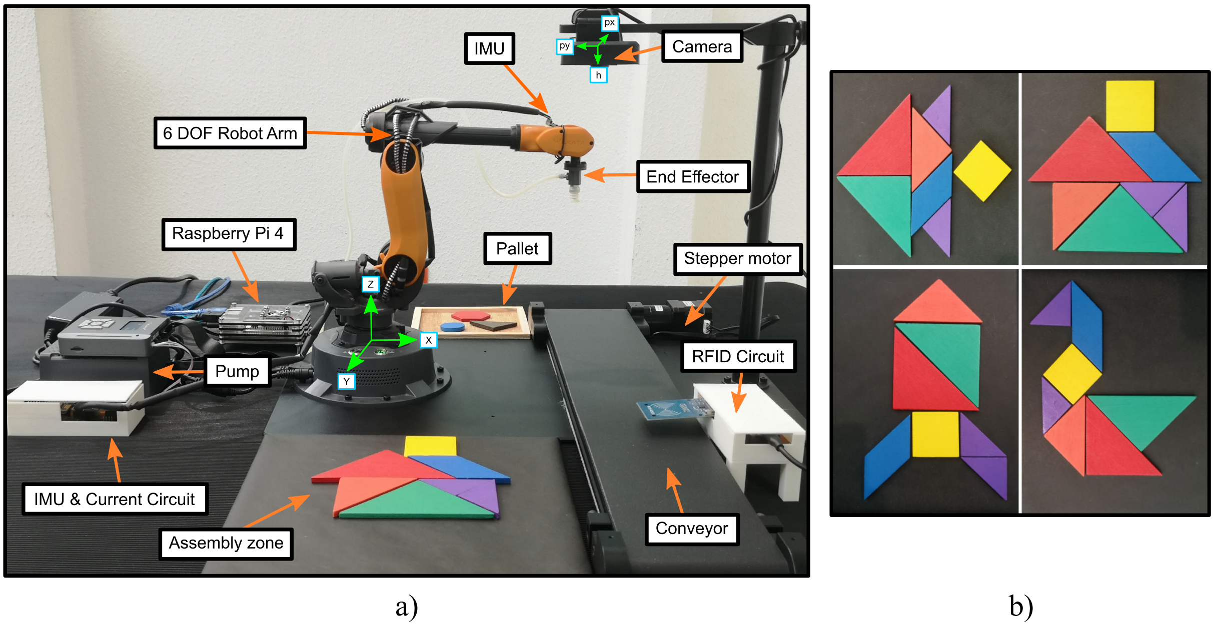


FIGURE 2. Tangram's puzzle assembly with the proposed Smart Factory. a) the manufacturing cell components that integrate the SF pilot testing; b) the four target puzzles to assemble.

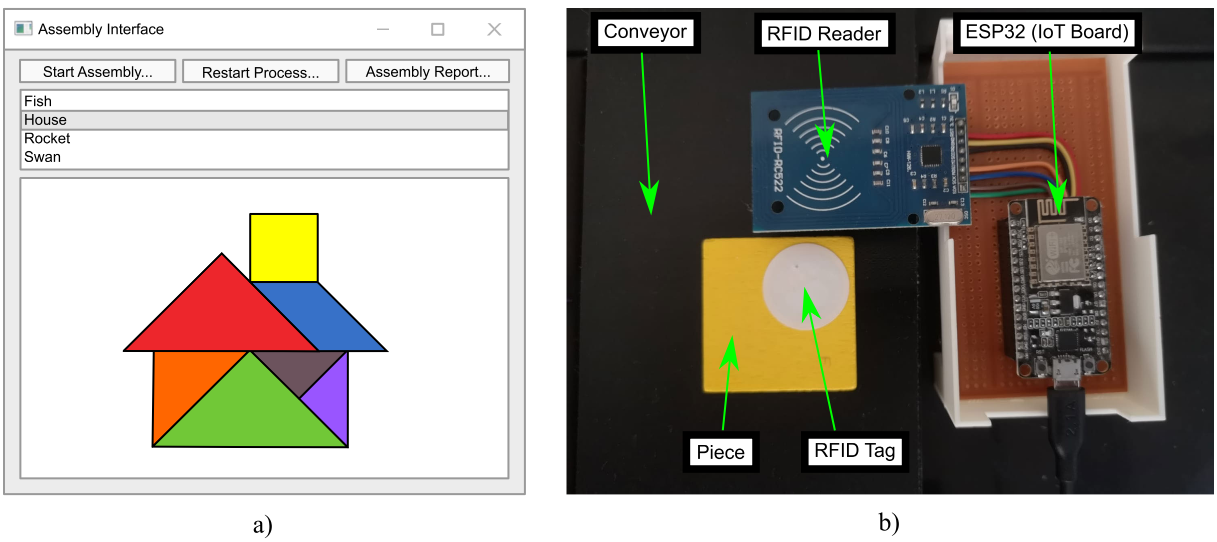


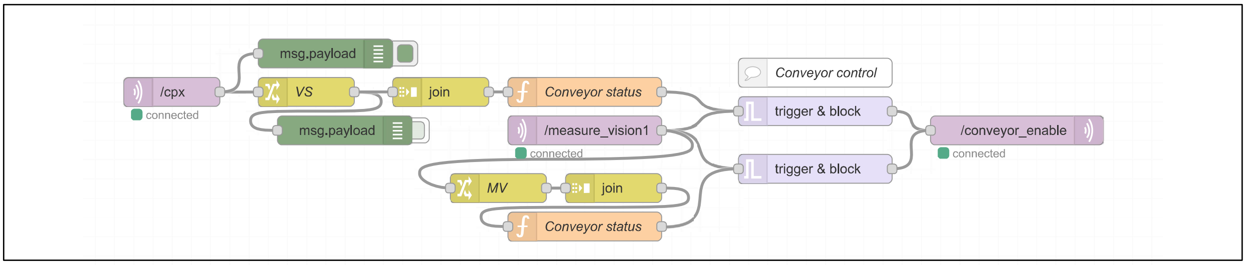
FIGURE 3. The first steps in the SF pilot testing. a) The GUI interface allows assembly selection, process start/restart, and report generation; b) The RFID circuit identifies the tag information (ID, date of manufacture, number of pieces, shapes, colors) of the batch.

Broker running in the cloud. The IoT platform received the information from the MQTT Broker, decrypted the message, and stored the data in the database.

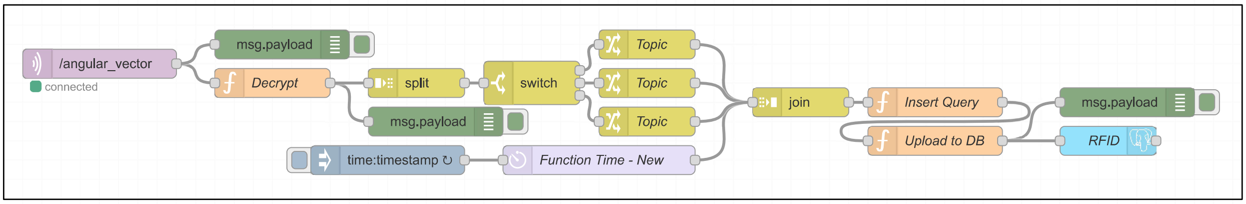
The sensors' data, the AI predictions, and the DA results were displayed in dashboards (running in the cloud) to show the assembly data in real-time. Finally, the assembly reports were generated through Python scripts at the end of the assembly process. The parameters and devices used to implement the scale smart factory pilot testing are described in detail in sub-section III-B. Additionally, the results obtained from the assembly reports are presented in Section IV.

B. EXPERIMENTAL SETUP

This section describes the parameters and configuration used to implement the pick and place process for the puzzle assembly in the SF pilot testing (see Fig. 2-a). The selection of the target puzzle was made through a Python interface (see Fig. 3-a) running in the Raspberry Pi 4 (8GB RAM, 64 bits, ARM v8 @1.5GHz, Debian OS) used as the Edge controller; once the puzzle was selected, the signal to start the calibration and home routines were sent to the Wlkata Mirobot (6 DOF robotic arm programmed through Python) and the Wlkata Conveyor, using the downstream channel (Serial Port connection). The first piece of the batch was



a)



b)

FIGURE 4. The IoT platform allows programming flows (Node-RED files) to communicate, control, and monitor the process. Node-RED flow example of a) the conveyor control includes the MQTT communication, data pre-processing, and conversion, b) sensor monitoring includes MQTT communication, data pre-processing, decryption, and database consulting.

placed on the conveyor, and the RC522 sensor detected the RFID tag, so the data was sent via Serial Peripheral Interface (SPI) to the ESP32 board (see Fig. 3-b).

The ESP32 board built a new message (topic, and payload) and encrypted the information using the AES algorithm with the CBC mode (16 bytes-length key and initialization vector); once the message was encrypted, it was sent to the Cloud (virtual machine in the GCP running Centos 7) through the MQTT protocol. The MQTT Broker (Eclipse Mosquito, version 2.0.14, TCP ports 1883/8883) that is running in the Cloud received the message and redirected it to the Node-RED platform (flow-based tool for visual programming, version 2.1, TCP port 1880) to decrypt the payload (AES-CBC mode) and process the information to build a query for requesting the data stored in the PostgreSQL database (version 14.1, TCP port 5432). Some examples of the Node-RED flows are presented in Fig. 4.

The result of the query included the main batch information that was sent via MQTT to the Machine Learning model, running in the Cloud, to predict the assembly success or abortion. The ML model is a Random Forest (RF) classifier composed of 100 estimators, that uses as input features the number of pieces included in the batch, and the class is defined as assemble/suspend according to the supervised decision if it is possible to complete the tangram with the pieces in the batch (see Fig. 5-a); the model is fitted with 100 different batch examples so the RF is able to learn when the assembly can be completed, or it is necessary to suspend the process (see Fig. 5-b). Then, the batch information, as well as the ML prediction, were sent to the Edge Device using the Upstream channel (port 443 for HTTPS over SSL and 8883 for MQTT over TLS); at the time the information was received, the conveyor was moved until it reached the vision system workspace to start or abort the assembly puzzle, taking into account the ML prediction.

In general, if the SF determined to assemble the puzzle, the vision system (Python script, with OpenCV version 4.5.4) proceeded to detect the piece's contour, centroid, and orientation through a Full HD camera (1920 × 1080 pixels) (see Fig. 6-a). On the other hand, the Deep Learning model analyzed the shape, size, and position; it differentiated between triangles, squares, rhomboids, hexagons, and circles; the DL required the libraries of Tensorflow, Keras, Numpy, OpenCV, and Matplotlib. The training and testing phases of the DL model used the 2D geometric shapes dataset [92], composed of 9 classes of geometric shapes (triangle, square, pentagon, hexagon, heptagon, octagon, nonagon, circle, and star). The dataset included 10,000 images per geometrical shape (200 × 200 pixels) and each image was randomly different from the other in background color, shape filling color, shape position in the image, shape rotation angle, and shape scale. Due to the rhomboid was not included in the 2D geometric shape dataset, it was required to design our dataset of rhomboids (10,000 images, 200 × 200 pixels), each one randomly different from the other, as the Korchi dataset [92]; some examples of the shapes used as part of the dataset for the smart factory pilot testing are shown in Fig. 6-b.

The structure of the CNN required ten layers (input, first convolution and pool, second convolution and pool, third convolution and pool, flatten, dense, and output), and it used the Adam optimizer [93], sparse categorical cross-entropy loss, and sparse categorical accuracy as parameters for the model compilation in the training phase (see Fig. 6-c). The training phase required 80% of the images (during 20 epochs), and the 20% remaining for the testing phase. The shapes used in the dataset included triangles,

```
Model_RF = RandomForestClassifier(n_estimators=100,
                                  criterion="gini",
                                  max_features="sqrt",
                                  bootstrap=True,
                                  max_samples=2/3,
                                  oob_score=True)

Model_RF.fit(BatchNo[BatchNo.columns[:-1]].values, BatchNo["Class"].values)

print(Model RF.predict([[2,2,2,2,2,2,2]]))

    ['Assemble']

print(Model_RF.predict([[0,2,2,2,2,2,2]]))

    ['Suspend']
```

1) Model parameters

2) The model is trained with: inputs target values

3) Batch with the required pieces and the estimation prediction

4) Batch without the required pieces and the estimation prediction

a)

	Inputs							Target Value
	Square	Rhomboid	Small_Triangle	Medium_Triangle	Big_Triangle	Circle	Hexagon	Class
0	2	2	2	2	2	0	0	Assemble
1	1	1	2	1	2	1	1	Assemble
2	1	2	1	1	0	1	1	Suspend
3	1	2	2	2	2	1	1	Assemble
4	2	1	3	2	2	2	0	Assemble
...	...	...	...	...	...	...	...	...
95	2	0	3	0	3	1	2	Suspend
96	0	3	0	3	1	2	0	Suspend
97	2	1	2	3	2	0	3	Assemble
98	1	2	2	2	3	2	1	Assemble
99	2	3	2	2	3	1	1	Assemble

100 rows × 8 columns

b)

FIGURE 5. **The Machine Learning model is executed in the cloud and decided to assemble or suspend. a) Random Forest model developed through the sklearn python class, defined using 100 estimators (trees); b) the training dataset (100 batch cases) includes 7 inputs (number of shapes) and the target value (assemble or suspend).**

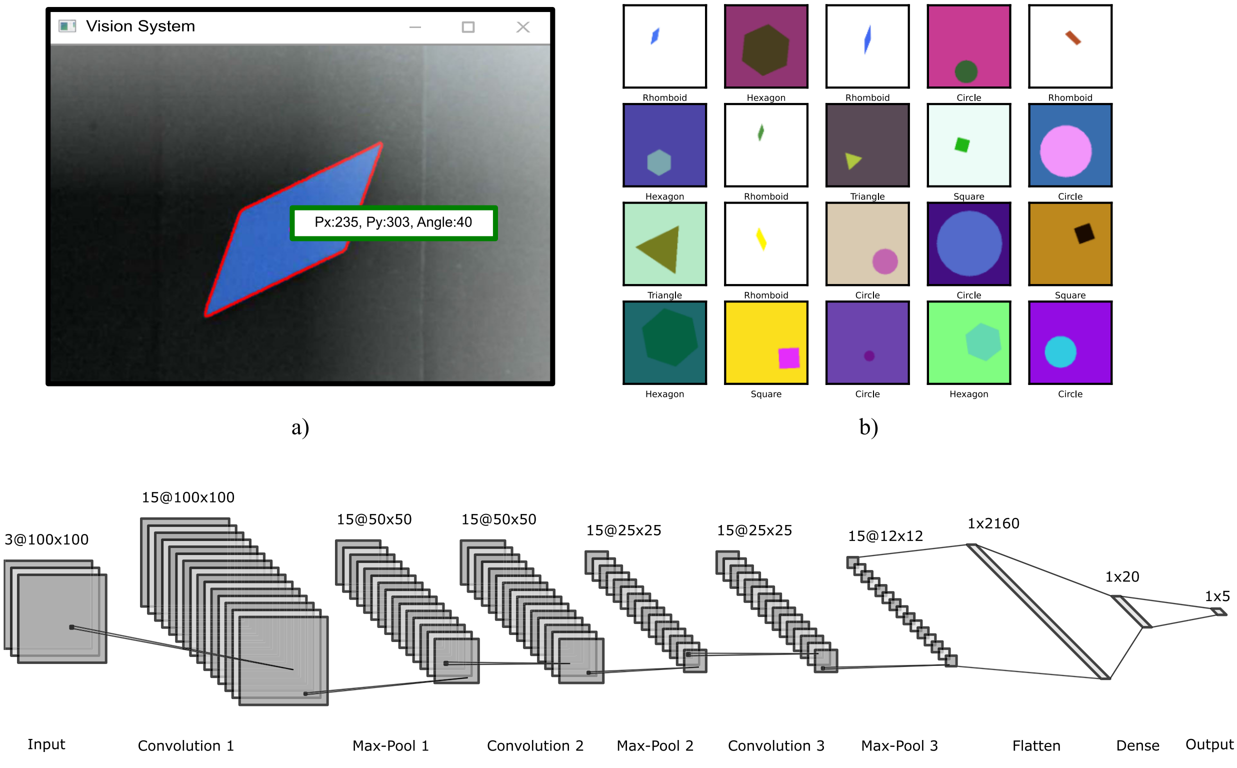


FIGURE 6. **The SF pilot testing recognizes the features of the pieces when they reach the vision workspace. a) the vision system detects the contour, the centroid, and the orientation of the piece; b) the proposed dataset used to train the CNN includes 10,000 images (with 5 different shapes, each one with different features); c) the CNN model is developed using 10 layers for the recognition of the shape by DL.**

squares, hexagons, circles, and rhomboids. The Tangram's puzzle shapes only allowed in the assembling were triangles, squares, and rhomboids; the hexagons and circles in the batch were the impostor shapes to test the adaptability. Finally, a Python Script (PS) in the Edge Device integrated the vision system results and the Deep Learning (by CNN) prediction for the five pieces.

Figure 7 presents the Behavioral Cloning algorithm followed to achieve the learning process implemented in an ANN structure. The relocation of the piece was done by the Mirobot arm, executing the pick-and-place routine through a Behavioral Cloning algorithm (BC), that maps the policy of the teacher to the agent. This learning process incorporated information about the centroids, previously identified (x, y pixels) by the vision system workspace, to map the 2D digital coordinate to the physical Cartesian coordinates into the Mirobot arm. To reach the centroid, it was necessary to place the pneumatic gripper in a defined position

Algorithm 1

1. Create an instance of the class Robot (object initiates variables and methods)
2. Define the number of iterations "i" for the learning mode
3. Learning mode (repeat for i iterations) + (aiMode = False)
    3.1. Image capture
    3.2. Centroids (x,y) location ("getState function" → contour and moment calculation)
    3.3. Manual location of the robot in the cartesian position (X,Y,Z) → ("RobotControl" function)
    3.4. Policy (π_0) learning: agent relates pixel and cartesian position ("learn function" → ANN model fit)
4. Change AI mode to run automatic (aiMode = True)
    4.1. Image capture
    4.2. Centroids (x,y) location ("getState function" → contour and moment calculation)
    4.3. Agent calculates cartesian positions using its own policy (π_a) ("act function" → ANN model predict)
    4.4. Automatic location of the robot in the position (X,Y,Z) → ("RobotControl" function)
5. Evaluation of policy and action response from AI (¿ $\pi_0 \approx \pi_a$?)
    5.1. Correct picking of the physical piece
    5.2. Correct place of the physical piece
6. Repeat steps 3-5 in case:
    6.1. $\pi_0 \neq \pi_a$
    6.2. New trajectory / task are required for the robot

FIGURE 7. Behavioral Cloning algorithm that summarizes the information to configure the agent, using the teacher's policy to control the robotic arm.

(X, Y, Z). Furthermore, the Behavioral Cloning algorithm required the definition of the agent (entity capable of perceiving its environment and making decisions to execute actions), an instance of the Robot class with its attributes and methods (Algorithm 1, Line 1). The attributes were the AI mode (learning phase / automatic behavior), the vision system, and the ANN model (ANN structure and model compilation), see (Algorithm 1, Line 3). On the other hand, the methods were the Learn function (fits the ANN), Get-State function (agent receives feedback from the workspace), and Act function (place the servomotors in the required position).

It was formed by five layers (input, three hidden layers, and output) and used the Adam optimizer, the mean squared error loss, and the accuracy metric as compilation parameters. After the ANN compilation was finished, the robot (agent) started the learning phase, where the expert (human) provided the centroids and the Cartesian coordinates as training values; the agent learned by itself the policy (mapping the location of the shape centroids (px, py), to the physical cartesian coordinates (X, Y, Z)) of the demonstrator to replicate the actions in unseen scenarios, then the Imitation Learning stage was completed (Algorithm 1, Line 4). Afterward, the PS integrated the result of the BC agent (in automatic mode) with functions like pick, place, and emergency stop through the Downstream channel.

During the pick and place process, the INA219 sensor measured the pneumatic system's current, which was sent to the ESP32 boards via Inter-Integrated Circuit protocol (I2C). On the other hand, the MPU6050 inertial measurement unit (IMU) was placed in the Mirobot link (joints four and five), to measure the angle, acceleration, and angular speed; the IMU also sent the data to the ESP32 boards via I2C. Table 2 summarizes the information of the sensors implemented in the case study. Once the IoT boards had the information, the new messages were created and encrypted in the ESP32 to send the information to the Cloud (see Fig. 8-a). Subsequently, the data was processed (Node-RED), stored in the PostgreSQL database (see Fig. 8-b), and displayed in real-time dashboards using Grafana (version 8.2.5, TCP port 3000).

TABLE 2. Specifications of the sensors implemented in the case study.

Sensor	Protocol	Variable	Supply Voltage	Location
RC522	SPI	RFID tag	3.3 v	Conveyor
INA219	I2C	Current	5 v	Pump
MPU6050	I2C	Angular position	5 v	Robot

The connections in the cloud between PostgreSQL, Python, and Grafana required configuring the IP address, port (5432), database name, and PostgreSQL authentication. Grafana displayed recent information in real-time dashboards (time-series or gauges) as it is observed in Fig. 8-c, and it required the parameter configuration of the time displayed (range) and time refresh (dashboard update). Additionally, Python stored the DB registers in DataFrames (structure with two dimensions), using the psycopg2 adapter to connect with the database. Once the process ended, the Python interface generated the assembly reports indicating the pieces that were assembled or stored, the sequence in which the pieces were placed, the status (completed / not completed), and assembly time.

Finally, to control and monitor the SF, a basic SCADA system was developed using the IoT Platform (Node-RED) running in the cloud; the modules required to develop the system were the node- red-dashboard, node-red-contrib-ui-svg, and the node-red-contrib-moment; the system included i) control nodes (buttons, switches or sliders), that required the definition of parameters such as the group name (dashboard section), label displayed in the dashboard, name

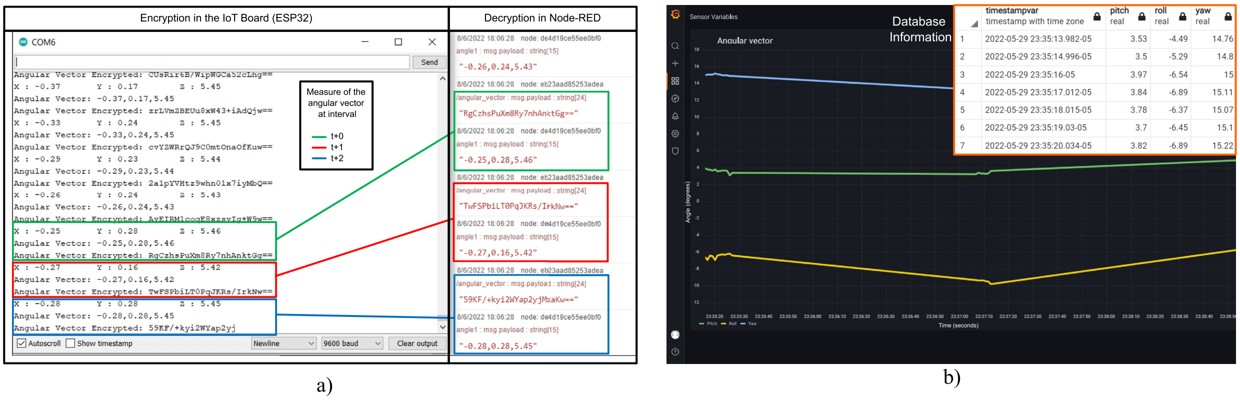


FIGURE 8. The SF pilot testing performs the collection, encryption/decryption, storage, and visualization of the data. a) The data measured (E.g. pitch, roll, yaw) by the sensors is encrypted and sent to the cloud, subsequently, the information is decrypted in the IoT platform; b) The information decrypted is stored in the PostgreSQL database and displayed in real-time dashboards with Grafana (data log display of the pitch, roll, and yaw positions).

of the node, range or value of the control node, among others; ii) monitoring nodes (texts, charts, or gauges), that required the definition of parameters such as the group name (dashboard section), label displayed in the dashboard, name of the node, type of the information displayed (indicators, time series, etc.), and units of the monitored variable. The implementation results of the SCADA system are presented in sub-section IV-B.

The SF pilot testing was provided with pieces from four different batches, each one including a different number of shapes, and was tested in 16 scenarios (four runs for each puzzle). The following section will present the results obtained from the implementation of the SF pilot testing.

IV. RESULTS

The first part of this section presents the puzzles solved log, the assemblies report, and the time assembly statistics. In the second part, the SCADA system developed through the Node-RED platform is presented (dashboard and flows), which allows the supervision and control of the SF pilot testing assets. Finally, the KPIs of the main assets are presented as indicators of efficiency during the assembly process.

A. ASSEMBLY RESULTS

1) ASSEMBLY LOGS

The scale Smart Factory pilot testing allowed the assembly of four figures (fish, house, rocket, and swan, see Fig. 2-b); each figure was integrated by one rhomboid, one square, two small-triangles, one medium-triangle, and two big-triangles.

Fig. 9 presents the steps followed by the SF pilot testing for the pick and place process to locate a piece (E.g., house's red big-triangle) in the assembly zone. The process started when the piece arrived through the conveyor, so the vision system workspace extracted the main features (see Fig. 9-a). Then, the SF calculated the coordinates to locate the pneumatic end effector above the centroids (see Fig. 9-b). The robot picked the piece, moved it to the assembly zone, and oriented the piece according to the figure required by the user (see Fig. 9-c). The robot located the piece in the position where it was required according to the figure selected (see Fig. 9-d). Finally, the pneumatic end effector released the piece (see Fig. 9-e). The process was repeated with the remaining pieces to complete the assembly, so the impostor and repeated pieces were placed in the pallet for future storage (see Fig. 9-f).

2) ASSEMBLY REPORTS

The assembly reports generated by the SF pilot testing included the status of the assembly (if it was completed or not completed), the number of pieces that were assembled, the number of pieces stored (this includes the impostor and repeated pieces in the batch), the assembly sequence (the pieces in the batch arrive in a different order each time), and the assembly time (period to build the figure required). Fig. 10 presents an example of the final assembly report delivered by the SF pilot testing.

In particular, Fig. 10 presents the information for the house puzzle with batch number one. The first page mentions the ML model prediction result (assemble), home or calibration routine timestamp (14:49.63 on June 04-2022), assembly status (completed), assemble time (11:08.18 minutes), and missing pieces (zero pieces). The second page resumes the robot's actions (10 picks, seven placed in the assembly zone, and three placed in the pallet warehouse), conveyor activation timeline (minimum 26.2 s, maximum 35.3 s, mean 30.8 s, standard deviation 3 s), conveyor deactivation timeline during the pick and place (minimum 34.2 s, maximum 37.3 s, mean 35.9 s, standard deviation 1.5 s), assembly sequence (rhomboid, medium-triangle, small-triangle, square, big-triangle, small-triangle, and big-triangle), and warehouse pallet storage sequence (hexagon, small-triangle, and circle). The last part of the second page presents features of the

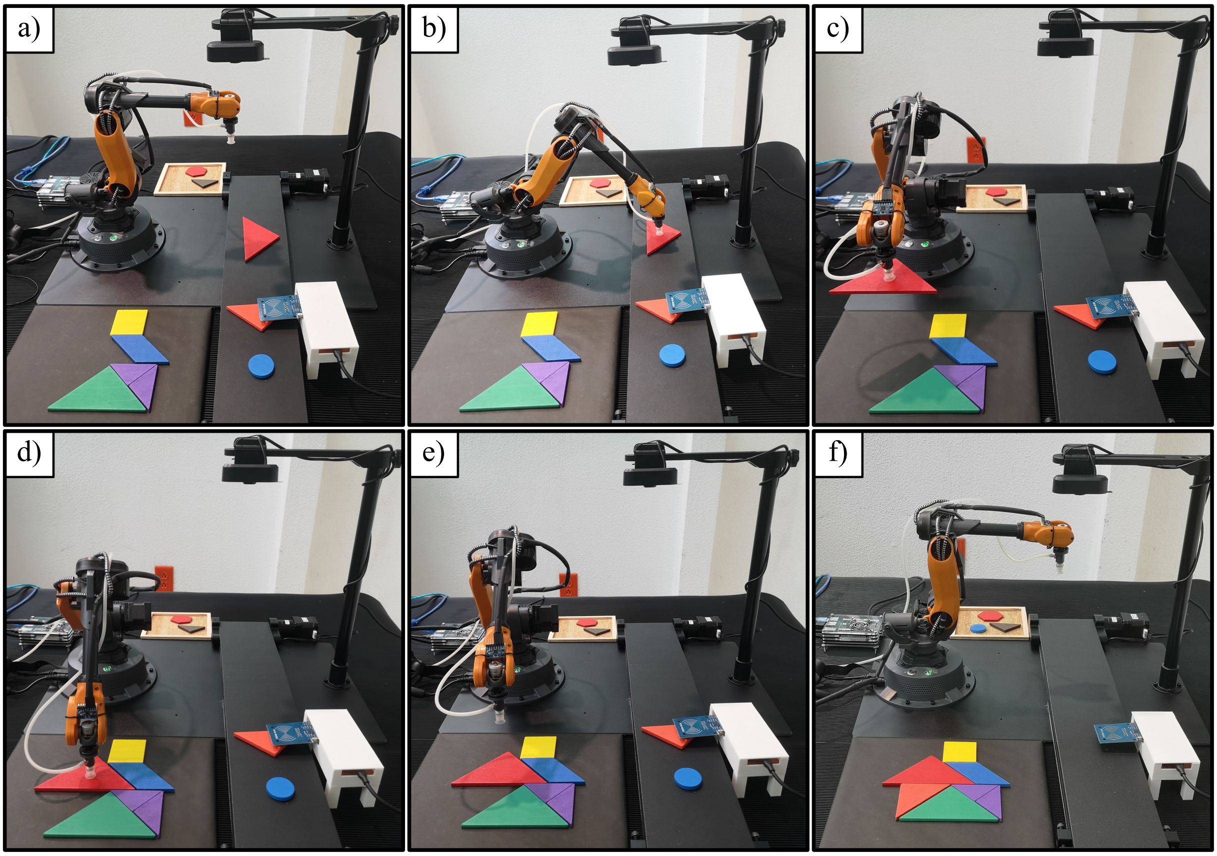


FIGURE 9. Steps in the house assembly process by the SF: a) identification of piece features, b) piece picking, c) transportation to the assembly zone, d) piece placing, e) piece releasing, and f) assembly process completion.

assembly sequence and the pieces returned to the warehouse, which includes the timestamp, shape, size, and RGB intensity.

3) ASSEMBLY STATISTICS

Fig. 11 summarizes the results for the 16 runs. The fastest assembly was nine (rocket puzzle with batch one) run with 10:30.7 minutes, and the slowest assembly was eight (house puzzle with batch four) run with 12:07.9 minutes, presenting a difference of 1:37.20 min. The mean values were calculated according to equation 1, which indicates the average value between the samples observed [94]:

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i \tag{1}$$

The standard deviation was calculated using equation 2, and it represents the squared root of the variance (variability of the data with respect to its arithmetic mean), [94]:

$$s = \sqrt{\frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{n-1}} \tag{2}$$

The mean and standard deviation of the assembly time were measured for all puzzles, and they are presented in Table 3.

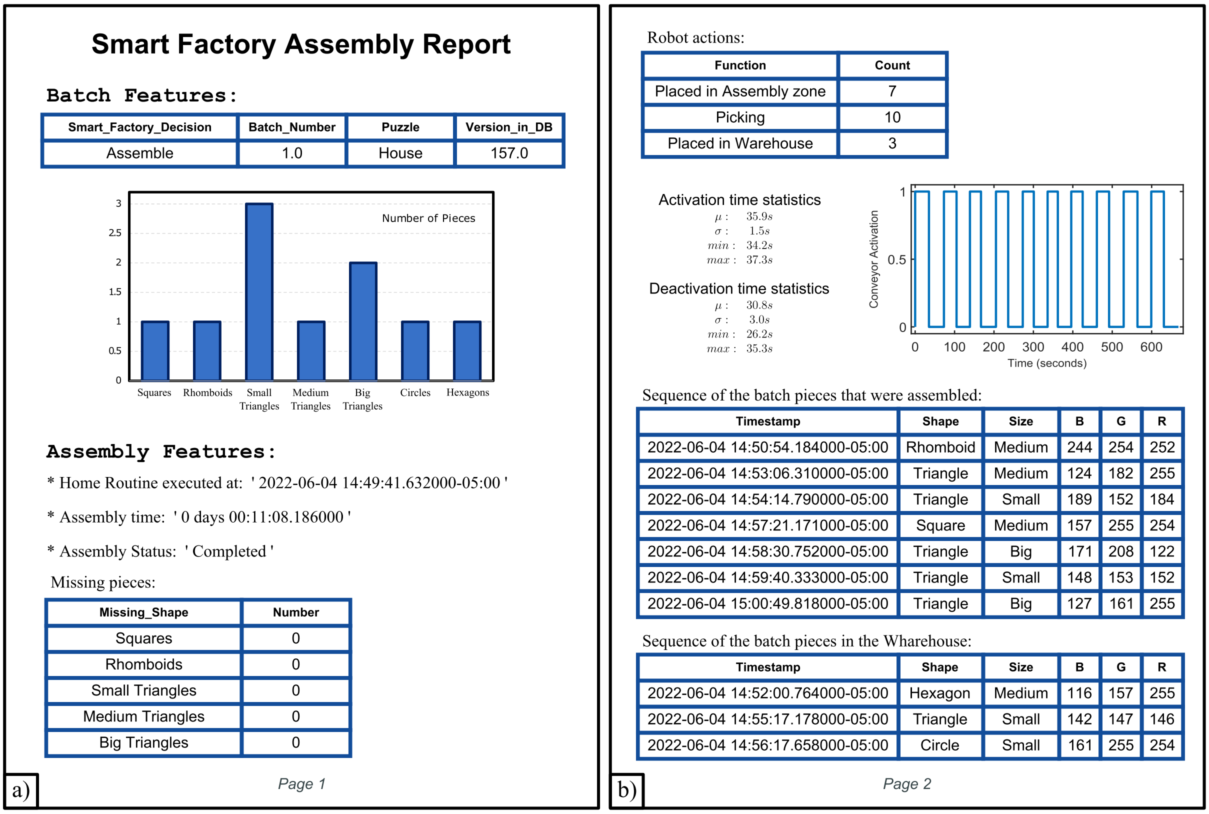
TABLE 3. Assembly time statistical information for each tangram puzzle completed.

Puzzle	Mean	Standard deviation
Fish	10:57.5 min	28.8 s
House	11:17.7 min	35.9 s
Rocket	10:44.9 min	11.0 s
Swan	11:20.8 min	4.9 s

According to Table 3, the fastest assembly puzzle was the rocket (mean 10:44.9 min.), and the slowest puzzle was the swan (mean 11:20.8 min.). The swan presented the minor standard deviation and the house presented the higher (4.9 s and 35.9 s, respectively).

B. SCADA SYSTEM APPROACH

The Supervisory Control and Data Acquisition system was implemented through the IoT Platform (Node-RED). Figure 12 displays the SCADA system developed for the asset (WLKata Mirobot), the system includes the control section that allows the movement of the robot joints (j1 to j6) by sliders, and routine execution programmed routines such as home routine to restart the process, zero position to locate the robot's joints at a value of zero degrees, and pick routine to



Smart Factory Assembly Report

Batch Features:

Smart_Factory_Decision	Batch_Number	Puzzle	Version_in_DB
Assemble	1.0	House	157.0

Assembly Features:

* Home Routine executed at: ' 2022-06-04 14:49:41.632000-05:00 '

* Assembly time: ' 0 days 00:11:08.186000 '

* Assembly Status: ' Completed '

Missing pieces:

Missing_Shape	Number
Squares	0
Rhomboids	0
Small Triangles	0
Medium Triangles	0
Big Triangles	0

a) Page 1

Robot actions:

Function	Count
Placed in Assembly zone	7
Picking	10
Placed in Warehouse	3

Activation time statistics
μ : 35.9s
σ : 1.5s
min : 34.2s
max : 37.3s

Deactivation time statistics
μ : 30.8s
σ : 3.0s
min : 26.2s
max : 35.3s

Sequence of the batch pieces that were assembled:

Timestamp	Shape	Size	B	G	R
2022-06-04 14:50:54.184000-05:00	Rhomboid	Medium	244	254	252
2022-06-04 14:53:06.310000-05:00	Triangle	Medium	124	182	255
2022-06-04 14:54:14.790000-05:00	Triangle	Small	189	152	184
2022-06-04 14:57:21.171000-05:00	Square	Medium	157	255	254
2022-06-04 14:58:30.752000-05:00	Triangle	Big	171	208	122
2022-06-04 14:59:40.333000-05:00	Triangle	Small	148	153	152
2022-06-04 15:00:49.818000-05:00	Triangle	Big	127	161	255

Sequence of the batch pieces in the Wharehouse:

Timestamp	Shape	Size	B	G	R
2022-06-04 14:52:00.764000-05:00	Hexagon	Medium	116	157	255
2022-06-04 14:55:17.178000-05:00	Triangle	Small	142	147	146
2022-06-04 14:56:17.658000-05:00	Circle	Small	161	255	254

b) Page 2

FIGURE 10. SF reports which integrate tables, figures, and plain text: a) first-page displays batch and assembly features (E.g., house puzzle assembly using batch one); b) second-page displays the actions of the asset and the assembly sequences.

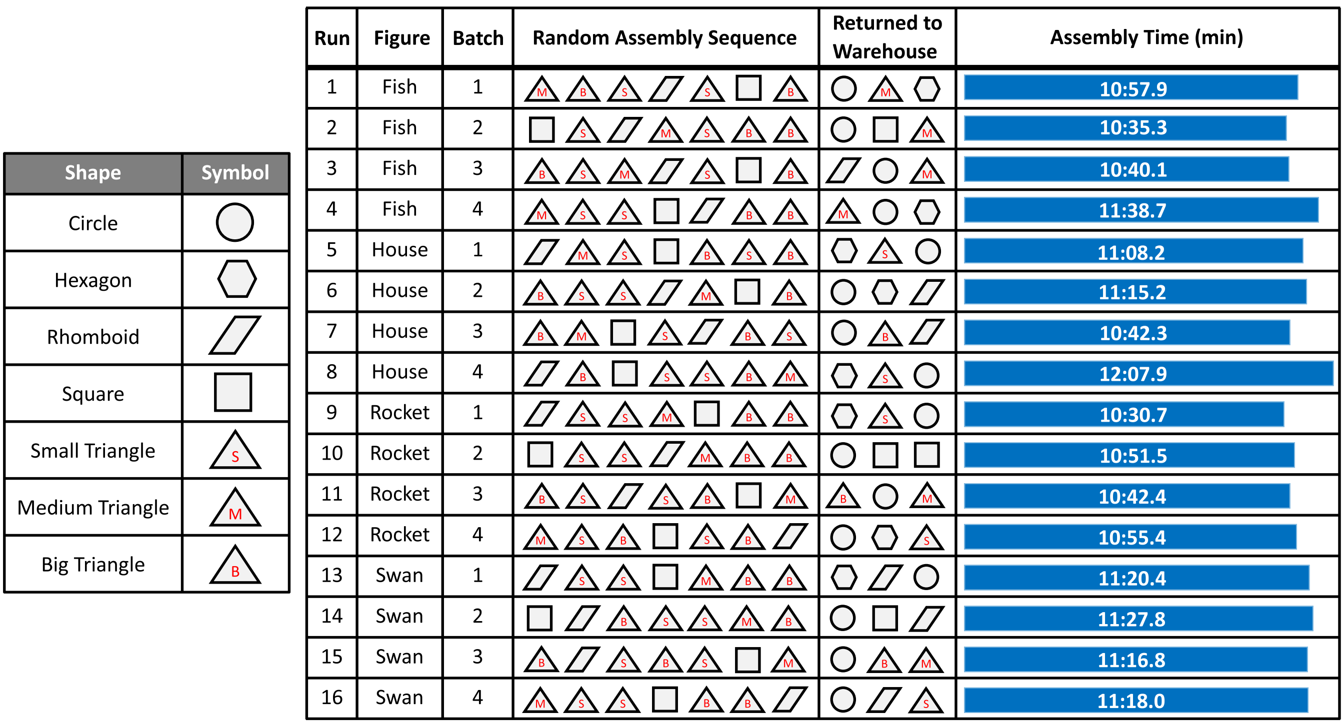


FIGURE 11. Final results obtained from 16 assembly runs, performed by the SF that combines different shapes in each batch.

1190 execute the pick and place task according to the information
1191 of the Python scripts running in the Edge.

In the same way, the monitoring section of the SCADA 1192
system displays information on the actual position of all the 1193

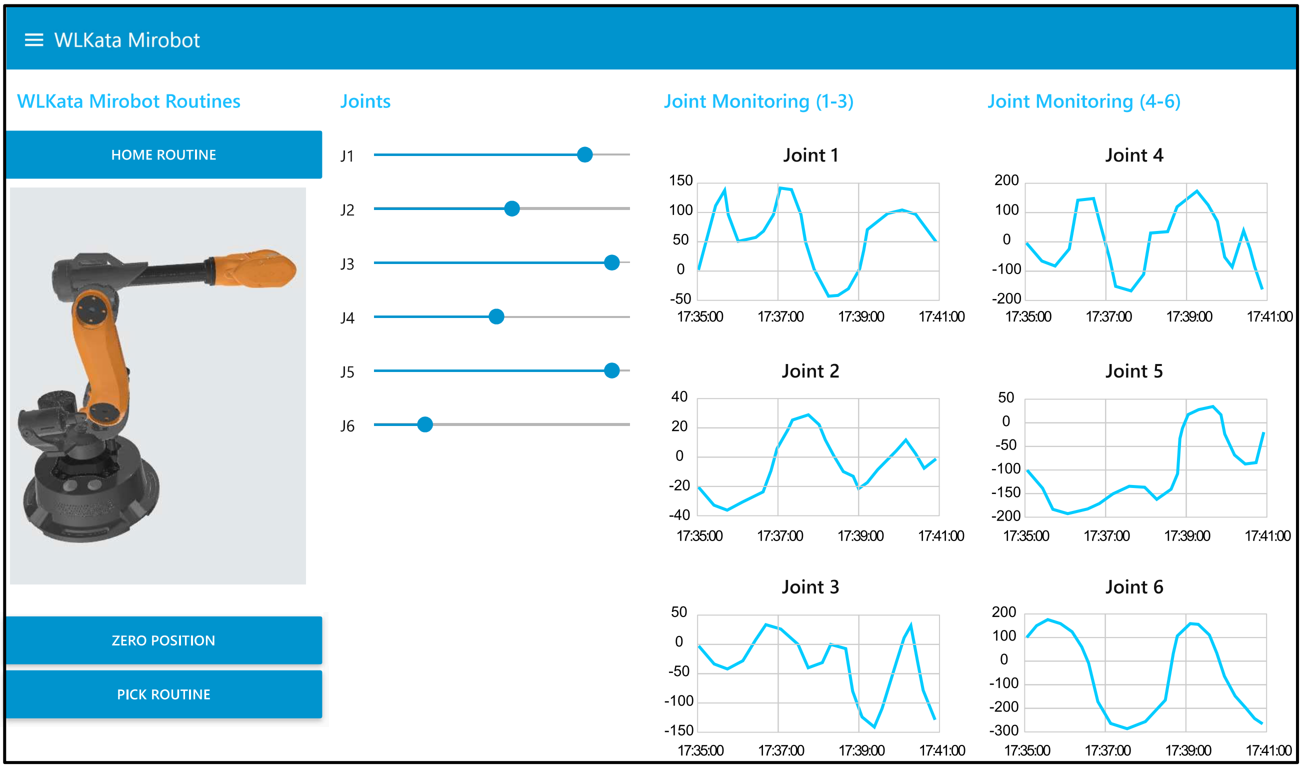
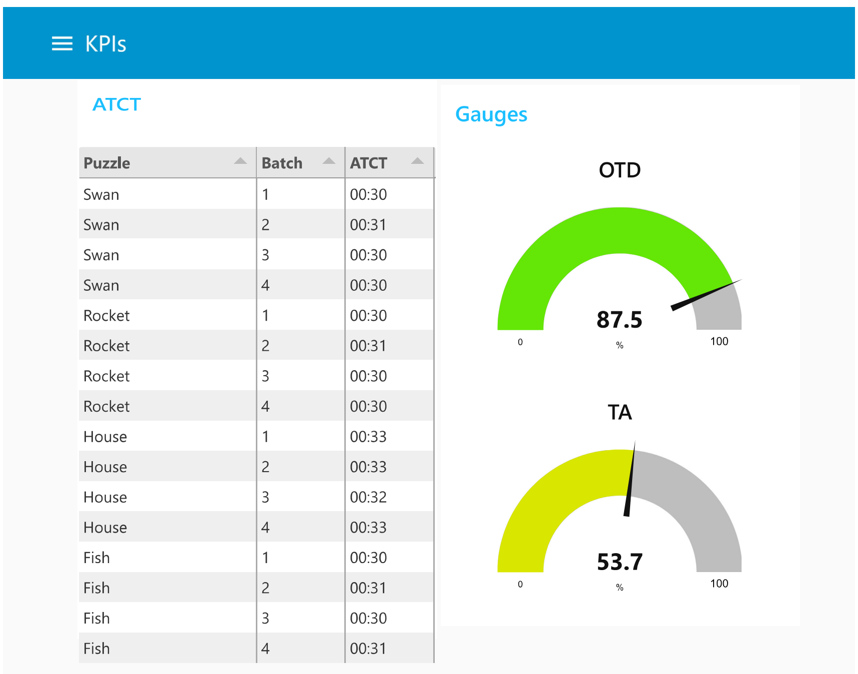


FIGURE 12. The Smart Factory pilot testing SCADA system for the WLKata Mirobot presents information on the control and monitoring of the process by dashboards.



Puzzle	Batch	ATCT
Swan	1	00:30
Swan	2	00:31
Swan	3	00:30
Swan	4	00:30
Rocket	1	00:30
Rocket	2	00:31
Rocket	3	00:30
Rocket	4	00:30
House	1	00:33
House	2	00:33
House	3	00:32
House	4	00:33
Fish	1	00:30
Fish	2	00:31
Fish	3	00:30
Fish	4	00:31

FIGURE 13. KPIs of the Smart Factory pilot testing that indicate productivity and time tracking of the 16 puzzles assembled during the case study.

joints of the robot (Joint Monitoring); the charts are updated in real-time using the actual state of the joint position, stored in the database, by the upstream channel.

C. KEY PERFORMANCE INDICATORS

The three KPIs of the assembly process were calculated through the IoT Platform, they were On-Time Delivery

(OTD), Average Task Completion Time (ATCT), and Time Activity (TA). Fig. 13 resumes the results obtained for the process of the SF pilot testing.

1) PRODUCTIVITY KPI

The OTD indicates the performance of the process to assemble the required puzzles at a specific time [95], and it was calculated according to eq. 3:

$$OTD = \frac{TD - DD}{TD} \tag{3}$$

where the total deliveries (TD) was 16, and the number of delayed deliveries (DD) was two; the threshold to detect the DD was 11:30 min. to complete the puzzle assembly, obtaining an OTD of 87.5% for the SF (see Fig. 13 - column gauges). Typically, the OTD was 50%, according to the results, the achieved OTD is higher than the minimum recommended.

2) TIME TRACKING KPIs

The ATCT was calculated to monitor the efficiency of the asset when performing repetitive tasks (pick and place process) for a specific number of times in seconds [96]; it is calculated according to eq. 4:

$$ATCT = \frac{TTCT}{NTP} \tag{4}$$

where the total time to complete a task (TTCT) was the time invested to complete the pick and place process for each puzzle; the number of times performed (NTP) was the number of pick and place routines invested to complete the puzzle, for this case was 10 times. Fig. 13 (see column ATCT) shows the values calculated for each run of the pick and place process realized in the SF pilot testing. According to the results, the pick and place process exhibits repeatability (less than 3.333% of variability), which means that the architecture is precise in its hardware and software settings.

Finally, the TA indicates the time that the assets were used within the whole process [96], and it was calculated according to eq. 5:

$$TA = \frac{AC}{PTT} \tag{5}$$

where the total time that the assets performed a task (AC) was the time that the conveyor and the robot were actively performing a task, and the time it took to complete the process (PTT) was the total time in which the SF completed all the puzzles; for the case study, the TA of the SF was 53.7%, see Fig. 13 (column gauges).

According to the results, the SF assets perform a task close to half of the total time, which represents an area of opportunity to reduce the time wasted in stopped positions.

V. DISCUSSIONS

In the most recent research, it has been detected that the SF architectures are being developed through hierarchical models to integrate specific technological solutions for particular applications or issues; the cost to upgrade a traditional factory to an SF is an impediment for the SMEs that want to migrate.

According to the Smart Factory architecture proposed, the open-source software implemented is compatible with the majority of the components of the industry; this compatibility can be applied through i) ethernet communication, 2) industrial gateways, or 3) OPCUA communication; making these changes the architecture would be working similarly as it was presented in the case study of this article, and all the information of the industrial components would be sent to the cloud and the edge without any problem.

The interaction between the six elements of the architecture required a higher level of design, programming, and definition of components to allow the SF to make independent decisions through Artificial Intelligence. The results from the SF pilot testing described the puzzles assembled, shapes, and main steps for the pieces assembled; the assembly reports included information such as the pieces in the assembly, assembly sequence, pieces in the warehouse, assembly time, assembly success, and missing pieces. Similarly, the SCADA system developed through an open-source IoT Platform allowed asset control (movement of robot joints and routine execution) as well as asset monitoring (information display).

Finally, the KPIs of the assembly process were calculated to monitor the state of the process, using the IoT Platform to measure productivity (OTD) and time tracking (ATCT and TA), different from the indicators used within the related work, which were more related with the OEE calculation, or Yield and Cost/Unit measurement, the majority of the state of the art do not realize an implementation not much less a KPI measurement. According to the results of the KPIs during the 16 runs, we found that the interconnection and digitization of the scale manufacturing cell were fully integrated and allowed repeatability; the proposed SF architecture is ready to be tested in a more complex scenario.

VI. CONCLUSION

According to the state of the art, the concept of the Smart Factory is not standardized, some research has agreed that the SF requires the digitization and interconnection of elements, to achieve the flexibility and adaptability of the factory when dynamic conditions are presented.

The proposed architecture represents an alternative to traditional factories because it combines the basic elements of the factory (cyber-physical systems, edge computing, cloud computing, and data analytics), and the new elements such as artificial intelligence and cybersecurity, to achieve the interconnection and digitization of all devices required within the factory, all of them implemented through open-source software.

Additionally, the case study presented in this research was a scale SF pilot testing, which consisted of a basic pick and place process to assemble a geometric Tangram puzzle. The implementation allowed testing features of the smart factory such as i) flexibility (randomize assembly sequence of the four geometric tangram puzzles), and ii) adaptability (DL to detect repeated pieces and the impostor shapes). Moreover, the experimental setup explained the specific technical parameters to implement the assembly process, indicating the devices, protocols, software, and algorithms used in the case study.

The proposed architecture can improve the competitiveness of the SMEs and allow them to digitize their facilities, using open-source tools, and this will allow them to invest resources in employee training, infrastructure, or new technology, so they could fulfill the norms and be able to establish relationships with companies to cooperate as suppliers or partners.

As part of future work, it would be necessary to test the proposed architecture in different processes that include assembly and manufacturing steps, variety in the periods of operation, and components implemented. Additionally, it would be necessary to perform scalability tests of the architecture, to find out the minimum changes that the architecture would require to be implemented in a process of small and medium enterprises. Some specific future tasks that are also required to study include testing different algorithms for the artificial intelligence models, encryption algorithms, 2D validation assembly, tolerance measurement, the quality of the radio frequency signals, and communication latency disconnections, or response time in the IIoT.

The present research explains the integration and definition of a new SF architecture; it required the review of SF prior art (academic papers, patents, and automation industry solutions) and previous architectures that did not integrate essential elements for the actual standards in the factory; observing this situation, maybe at some point in the future, the proposed architecture may not fit the requirements that the factories would need to implement the SF, and it would require an actualization or integration of new elements.

REFERENCES

[1] A. Radziwon, A. Bilberg, M. Bogers, and E. S. Madsen, "The smart factory: Exploring adaptive and flexible manufacturing solutions," *Proc. Eng.*, vol. 69, pp. 1184–1190, 2014.

[2] R. Burke, A. Mussomeli, L. Stephen, H. Marty, and S. Brenna, "The smart factory," in *The Smart Factory Responsive, Adaptive, Connected Manufacturing*, vol. 24. New York, NY, USA: Deloitte Univ. Press, 2017.

[3] F. Herrmann, "The smart factory and its risks," *Systems*, vol. 6, no. 4, p. 38, Oct. 2018.

[4] J.-P. Petit, P. Brosset, and P. Bagnon, "Smart factories at scale," Capgemini, Paris, France, Tech. Rep. 1, 2019.

[5] M. Moghaddam, M. N. Cadavid, C. R. Kenley, and A. V. Deshmukh, "Reference architectures for smart manufacturing: A critical review," *J. Manuf. Syst.*, vol. 49, pp. 215–225, Oct. 2018.

[6] N. Brouns, "On the road towards smart manufacturing—A framework to support the development of smart manufacturing," Ph.D. thesis, Eindhoven Univ. Technol., Eindhoven, The Netherlands, 2019.

[7] K. Haricha, A. Khiat, Y. Issaoui, A. Bahnasse, and H. Ouajji, "Recent technological progress to empower smart manufacturing: Review and potential guidelines," *IEEE Access*, vol. 11, pp. 77929–77951, 2023.

[8] M. Soori, B. Arezoo, and R. Dastres, "Internet of Things for smart factories in Industry 4.0—A review," *Internet Things Cyber-Phys. Syst.*, vol. 3, pp. 192–204, May 2023.

[9] B. N. Pasi, S. K. Mahajan, and S. B. Rane, "Redesigning of smart manufacturing system based on IoT, perspective of disruptive innovations of Industry 4.0 paradigm," *Int. J. Mech. Prod. Eng. Res. Develop.*, vol. 10, no. 3, pp. 727–746, Jun. 2020.

[10] D. Zakoldaev, A. Shukalov, I. Zharinov, and O. Zharinov, "Structure of digital and smart factories of the Industry 4.0," *IOP Conf. Ser., Mater. Sci. Eng.*, vol. 560, no. 1, 2019, Art. no. 012208.

[11] I. Guajardo, J. R. Garza, R. E. Rendón, and J. Allard, "Crafting the future: A roadmap for Industry 4.0 in Mexico: Report," Ministry of Economy, Florida, NM, USA, Tech. Rep., 2016.

[12] K. Siau, Y. Xi, and C. Zou, "Industry 4.0 challenges and opportunities in different countries," *Cutter IT J.*, vol. 2, pp. 23–34, Jul. 2019.

[13] W.-K. Jung, D.-R. Kim, H. Lee, T.-H. Lee, I. Yang, B. D. Youn, D. Zontar, M. Brockmann, C. Brecher, and S.-H. Ahn, "Appropriate smart factory for SMEs: Concept, application and perspective," *Int. J. Precis. Eng. Manuf.*, vol. 22, no. 1, pp. 201–215, Jan. 2021.

[14] S. Wang, J. Wan, D. Li, and C. Zhang, "Implementing smart factory of Industrie 4.0: An outlook," *Int. J. Distrib. Sensor Netw.*, vol. 12, no. 1, pp. 1–10, 2016.

[15] J. Jung, B. Song, K. Watson, and T. Usländer, "Design of smart factory web services based on the Industrial Internet of Things," in *Proc. 50th Hawaii Int. Conf. Syst. Sci.*, Aug. 2017, pp. 5941–5946.

[16] B. Chen, J. Wan, L. Shu, P. Li, M. Mukherjee, and B. Yin, "Smart factory of Industry 4.0: Key technologies, application case, and challenges," *IEEE Access*, vol. 6, pp. 6505–6519, 2018.

[17] O. A. Peter, S. D. Anastasia, and A. R. Muzalevskii, "The implementation of smart factory for product inspection and validation a step by step guide to the implementation of the virtual plant of a smart factory using digital twin," in *Proc. 10th Medit. Conf. Embedded Comput. (MECO)*, Jun. 2021, pp. 1–7.

[18] A. D. Pathaka and J. V. Tembhurne, "Internet of Things: A survey on IoT protocols," *SSRN Electron. J.*, vol. 3, pp. 483–487, May 2018.

[19] K. Cao, Y. Liu, G. Meng, and Q. Sun, "An overview on edge computing research," *IEEE Access*, vol. 8, pp. 85714–85728, 2020.

[20] I. Lee, "Internet of Things (IoT) cybersecurity: Literature review and IoT cyber risk management," *Future Internet*, vol. 12, no. 9, p. 157, Sep. 2020.

[21] M. Soderi, V. Kamath, J. Morgan, and J. G. Breslin, "Advanced analytics as a service in smart factories," in *Proc. IEEE 20th Jubilee World Symp. Appl. Mach. Intell. Informat. (SAMI)*, Mar. 2022, pp. 425–430.

[22] Z. Kemény, R. J. Beregi, G. Erdős, and J. Nacsa, "The MTA SZTAKI *Smart Factory*: Platform for research and project-oriented skill development in higher education," *Proc. CIRP*, vol. 54, pp. 53–58, Jan. 2016.

[23] N. Shariatzadeh, T. Lundholm, L. Lindberg, and G. Sivard, "Integration of digital factory with smart factory based on Internet of Things," *Proc. CIRP*, vol. 50, pp. 512–517, Jan. 2016.

[24] M. N. Birje, P. S. Challagidad, R. H. Goudar, and M. T. Tapale, "Cloud computing review: Concepts, technology, challenges and security," *Int. J. Cloud Comput.*, vol. 6, no. 1, pp. 32–57, 2017.

[25] J. Mocnej, A. Pekar, W. K. G. Seah, P. Papcun, E. Kajati, D. Cupkova, J. Koziorek, and I. Zolotova, "Quality-enabled decentralized IoT architecture with efficient resources utilization," *Robot. Comput.-Integr. Manuf.*, vol. 67, Feb. 2021, Art. no. 102001.

[26] D. Luo, Z. Guan, C. He, Y. Gong, and L. Yue, "Data-driven cloud simulation architecture for automated flexible production lines: Application in real smart factories," *Int. J. Prod. Res.*, vol. 60, no. 12, pp. 3751–3773, Jun. 2022.

[27] M. Saturno, V. M. Pertel, F. Deschamps, and E. de Freitas Rocha Loures, "Proposal of an automation solutions architecture for Industry 4.0," *DEStech Trans. Eng. Technol. Res.*, vol. 14, no. 2, pp. 185–195, Jul. 2017.

[28] H. Muccini and M. T. Moghaddam, "IoT architectural styles," in *Software Architecture*. Cham, Switzerland: Springer, 2018.

[29] B. S. Chohan, X. Xu, and Y. Lu, "MES dynamic interoperability for SMEs in the factory of the future perspective," *Proc. CIRP*, vol. 107, pp. 1329–1335, May 2022.

[30] D. J. Ahn, J. Jeong, and S. Lee, "A novel cloud-fog computing network architecture for big-data applications in smart factory environments," in *Computational Science and Its Applications—ICCSA 2018* (Lecture Notes in Computer Science, Lecture Notes in Artificial Intelligence, Lecture Notes in Bioinformatics), vol. 10964. Cham, Switzerland: Springer, 2018, pp. 520–530.

[31] M. Kim, J. Lee, and J. Jeong, "Open source based Industrial IoT platforms for smart factory: Concept, comparison and challenges," in *Computational Science and Its Applications—ICCSA 2019*, vol. 11624. Cham, Switzerland: Springer, 2019.
[32] M. Pipan, J. Protner, and N. Herakovič, "Integration of distributed manufacturing nodes in smart factory," in *Service Orientation in Holonic and Multi-Agent Manufacturing* (Studies in Computational Intelligence), vol. 803. Cham, Switzerland: Springer, 2019, pp. 424–435.
[33] L. O. Aghenta and M. T. Iqbal, "Development of an IoT based open source SCADA system for PV system monitoring," in *Proc. IEEE Can. Conf. Electr. Comput. Eng. (CCECE)*, May 2019, pp. 1–4.
[34] C. A. Osaretin, M. Zamanlou, M. T. Iqbal, and S. Butt, "Open source IoT-based SCADA system for remote oil facilities using node-RED and Arduino microcontrollers," in *Proc. 11th IEEE Annu. Inf. Technol., Electron. Mobile Commun. Conf. (IEMCON)*, Nov. 2020, pp. 571–575.
[35] I.-V. Niţulescu and A. Korodi, "Supervisory control and data acquisition approach in node-RED: Application and discussions," *IoT*, vol. 1, no. 1, pp. 76–91, Aug. 2020.
[36] C. Li, S. Mantravadi, and C. Møller, "AAU open source MES architecture for smart factories—Exploiting ISA 95," in *Proc. IEEE 18th Int. Conf. Ind. Informat. (INDIN)*, vol. 1, Jul. 2020, pp. 369–373.
[37] M. Waters, P. Waszczuk, R. Ayre, A. Dreze, D. McGlinchey, B. Alkali, and G. Morison, "Open source IIoT solution for gas waste monitoring in smart factory," *Sensors*, vol. 22, no. 8, p. 2972, Apr. 2022.
[38] Y.-M. Kwon and B.-H. Song, "Data security method for smart factory and gateway applying the same," Korea Intellectual Property Office, South Korea, South Korea Patent 2018 002 484 5A, 2018.
[39] J. Kim and O. Dong-Ha, "Method and system that providing smart factory service based on 5th generation communication," Korea Intellectual Property Office, South Korea, South Korea Patent 2020 006 333 9A, 2020.
[40] D. Oh, K. Choi, S. Oh, and B. An, "Risk detection smart sensing and monitoring system for conversion to smart factory in small business, and method thereof," Korea Intellectual Property Office, South Korea, South Korea Patent 20 210 107, 2020.
[41] *Accelerating Value Realization in the Smart Factory*, AWS, Siemens, Munich, Germany, 2020.
[42] *Siemens Digital Industries Software*, Siemens, Munich, Germany, 2021.
[43] *Smart Manufacturing Information to Connect and Optimize the Enterprise*, Rockwell-Automation, Milwaukee, WI, USA, 2016.
[44] Rockwell Automation. (2021). *Rockwell Automation, Inc*. [Online]. Available: https://www.rockwellautomation.com/en-ie.html
[45] *The New Freedom in Engineering: The ctrlX AUTOMATION Platform*, Bosch, Gerlingen, Germany, 2020.
[46] Bosch. (2021). *Bosch.IO GmbH*. [Online]. Available: https://bosch.io/
[47] Eaton. (2021). *Eaton—Powering Business Worldwide*.
[48] Eaton and T-Systems. (2021). *Eaton—Internet of Things*.
[49] P. Osterrieder, L. Budde, and T. Friedli, "The smart factory as a key construct of Industry 4.0: A systematic literature review," *Int. J. Prod. Econ.*, vol. 221, Mar. 2020, Art. no. 107476.
[50] H. Kaschel C. and L. M. S. Y Bernal, "Importance of flexibility in manufacturing systems," *Int. J. Comput. Commun. Control*, vol. 1, no. 2, p. 53, Apr. 2006.
[51] J. Wan, J. Yang, Z. Wang, and Q. Hua, "Artificial intelligence for cloud-assisted smart factory," *IEEE Access*, vol. 6, pp. 55419–55430, 2018.
[52] P. K. Illa and N. Padhi, "Practical guide to smart factory transition using IoT, big data and edge analytics," *IEEE Access*, vol. 6, pp. 55162–55170, 2018.
[53] P. A. Okeme, A. D. Skakun, and A. R. Muzalevskii, "Transformation of factory to smart factory," in *Proc. IEEE Conf. Russian Young Researchers Electr. Electron. Eng. (ElConRus)*, Jan. 2021, pp. 1499–1503.
[54] S. Kahveci, B. Alkan, M. H. Ahmad, B. Ahmad, and R. Harrison, "An end-to-end big data analytics platform for IoT-enabled smart factories: A case study of battery module assembly system for electric vehicles," *J. Manuf. Syst.*, vol. 63, pp. 214–223, Apr. 2022.
[55] C.-H. Hsu, S.-J. Cheng, T.-J. Chang, Y.-M. Huang, C.-P. Fung, and S.-F. Chen, "Low-cost and high-efficiency electromechanical integration for smart factories of IoT with CNN and FOPID controller design under the impact of COVID-19," *Appl. Sci.*, vol. 12, no. 7, p. 3231, Mar. 2022.
[56] J. Lee, P. C. Chua, L. Chen, P. H. N. Ng, Y. Kim, Q. Wu, S. Jeon, J. Jung, S. Chang, and S. K. Moon, "Key enabling technologies for smart factory in automotive industry: Status and applications," *Int. J. Precis. Eng. Manuf.-Smart Technol.*, vol. 1, no. 1, pp. 93–105, Jan. 2023.
[57] M. Abdelatti and M. Sodhi, "Lab-scale smart factory implementation using ROS," in *Robot Operating System (ROS)* (Studies in Computational Intelligence), vol. 1051. Cham, Switzerland: Springer, 2023, pp. 119–143.
[58] M. Ryalat, H. ElMoaqet, and M. AlFaouri, "Design of a smart factory based on cyber-physical systems and Internet of Things towards Industry 4.0," *Appl. Sci.*, vol. 13, no. 4, pp. 1–19, 2023.
[59] S. Horbach, J. Ackermann, E. Müller, and J. Schütze, "Building blocks for adaptable factory systems," *Robot. Comput.-Integr. Manuf.*, vol. 27, no. 4, pp. 735–740, Aug. 2011.
[60] H. Komoto, S. Kondoh, Y. Furukawa, and H. Sawada, "A simulation framework to analyze information flows in a smart factory with focus on run-time adaptability of machine tools," *Proc. CIRP*, vol. 81, pp. 334–339, Jan. 2019.
[61] R. West and G. Parmer, "A software architecture for next-generation cyber-physical systems," Dept. Comput. Sci., Boston Univ., Boston, MA, USA, Tech. Rep. 1, Position Paper at the NSF Cyber-Physical Systems Workshop, May 2006.
[62] T. Sanislav and L. Miclea, "Cyber-physical systems—Concept, challenges and research areas," *Control Eng. Appl. Informat.*, vol. 14, no. 2, pp. 28–33, May 2012.
[63] V. Roblek, M. Meško, and A. Krapež, "A complex view of Industry 4.0," *SAGE Open*, vol. 6, no. 2, Apr. 2016, Art. no. 215824401665398.
[64] J. Jamaludin and J. M. Rohani, "Cyber-physical system (CPS): State of the art," in *Proc. Int. Conf. Comput., Electron. Electr. Eng. (ICE Cube)*, Nov. 2018, pp. 1–5.
[65] W. Z. Khan, E. Ahmed, S. Hakak, I. Yaqoob, and A. Ahmed, "Edge computing: A survey," *Future Gener. Comput. Syst.*, vol. 97, pp. 219–235, Aug. 2019.
[66] M. Alam and I. R. Khan, "Edge computing and its impact on IoT," *Wesleyan J. Res.*, vol. 14, no. 7, pp. 211–222, Mar. 2021.
[67] J. Xing, H. Dai, and Z. Yu, "A distributed multi-level model with dynamic replacement for the storage of smart edge computing," *J. Syst. Archit.*, vol. 83, pp. 1–11, Feb. 2018.
[68] H.-D. Wehle, "Machine learning, deep learning, and AI: What's the difference?" *Data Scientist Innov. Day*, vol. 1, pp. 1–5, Jul. 2017.
[69] R. Ashri, "What is AI?" in *The AI-Powered Workplace*. Berkeley, CA, USA: Apress, 2020, pp. 15–29.
[70] D. Jakhar and I. Kaur, "Artificial intelligence, machine learning and deep learning: Definitions and differences," *Clin. Exp. Dermatol.*, vol. 45, no. 1, pp. 131–132, Jan. 2020.
[71] A. Hussein, M. M. Gaber, E. Elyan, and C. Jayne, "Imitation learning: A survey of learning methods," *ACM Comput. Surv.*, vol. 50, no. 2, pp. 1–35, Mar. 2018.
[72] T. Osa, J. Pajarinen, G. Neumann, J. A. Bagnell, P. Abbeel, and J. Peters, "An algorithmic perspective on imitation learning," *Found. Trends Robot.*, vol. 7, nos. 1–2, pp. 1–179, 2018.
[73] J. Zhang and P. S. Yu, *Machine Learning Overview*. Cham, Switzerland: Springer, 2019.
[74] B. Zohuri and F. Rahmani, "Artificial intelligence driven resiliency with machine learning and deep learning components," *Jpn. J. Res.*, vol. 1, no. 1, pp. 1–5, 2020.
[75] J. Hua, L. Zeng, G. Li, and Z. Ju, "Learning for a robot: Deep reinforcement learning, imitation learning, transfer learning," *Sensors*, vol. 21, no. 4, pp. 1–21, 2021.
[76] F. Torabi, G. Warnell, and P. Stone, "Behavioral cloning from observation," in *Proc. 27th Int. Joint Conf. Artif. Intell.*, Jul. 2018, pp. 4950–4957.
[77] Z. Cheng, L. Shen, and D. Tao, "Off-policy imitation learning from visual inputs," in *Proc. IEEE Int. Conf. Robot. Automat. (ICRA)*. Ithaca, NY, USA: Cornell Univ., Department Statistics and Applied Mathematics, 2023, pp. 1–14.
[78] S. Shilpashree, R. R. Patil, and C. Parvathi, "Cloud computing an overview," *Int. J. Eng. Technol.*, vol. 7, no. 4, pp. 2743–2746, 2018.
[79] M. Marwan, A. Kartit, and H. Ouahmane, "A secured data processing technique for effective utilization of cloud computing," *J. Data Mining Digit. Humanities*, vol. 1, pp. 1–12, Jan. 2018.
[80] M. L. Brodie, "Applied data science," in *Applied Data Science*, vol. 1. Sep. 2019, ch. 8, pp. 101–121.
[81] L. Metcalf and W. Casey, "Introduction to data analysis," in *Cybersecurity and Applied Mathematics*. Elsevier, 2016, pp. 43–65.
[82] B. Richmond, "Introduction to data analysis handbook," *Acad. Educ. Develop.*, vol. 1, no. 1, pp. 1–27, 2006.
[83] A. K. Waljee, P. D. R. Higgins, and A. G. Singal, "A primer on predictive models," *Clin. Transl. Gastroenterol.*, vol. 5, no. 1, p. e44, 2014.
[84] P. Kubben, M. Dumontier, and A. Dekker, *Fundamentals of Clinical Data Science*. Cham, Switzerland: Springer, Jan. 2018, pp. 1–219.
[85] G. N. Reddy and G. J. U. Reddy, "A study of cyber security challenges and its emerging trends on latest technologies," *Int. J. Eng. Technol.*, vol. 7, no. 11, pp. 125–128, Sep. 2014.

[86] P. Seemma, S. Nandhini, and M. Sowmiya, "Overview of cyber security," *Ijarcce*, vol. 7, no. 11, pp. 125–128, 2018.

[87] M. Hildebrandt, "Balance or trade-off? Online security technologies and fundamental rights," *Philosophy Technol.*, vol. 26, no. 4, pp. 357–379, Dec. 2013.

[88] I. Halenar, L. Halenarova, and P. Tanuska, "Communication safety of cybernetic systems in a smart factory environment," *Machines*, vol. 11, no. 3, p. 379, Mar. 2023.

[89] Y. Lu and L. D. Xu, "Internet of Things (IoT) cybersecurity research: A review of current research topics," *IEEE Internet Things J.*, vol. 6, no. 2, pp. 2103–2115, Apr. 2019.

[90] I. Setiawan and H. H. Purba, "A systematic literature review of Key Performance Indicators (KPIs) implementation," *J. Ind. Eng. Manag. Res.*, vol. 1, no. 3, pp. 200–208, 2020.

[91] M. Lafou, L. Mathieu, S. Pois, and M. Alochet, "Manufacturing system flexibility: Product flexibility assessment," *Proc. CIRP*, vol. 41, pp. 99–104, Jan. 2016.

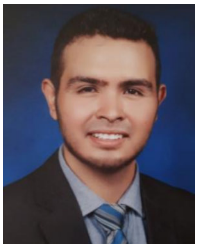
[92] A. E. Korchi and Y. Ghanou, "2D geometric shapes dataset—For machine learning and pattern recognition," *Data Brief*, vol. 32, Oct. 2020, Art. no. 106090.

[93] S. Bock and M. Weiß, "A proof of local convergence for the Adam optimizer," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Jul. 2019, pp. 1–8.

[94] D. Selvamuthu and D. Das, *Introduction to Statistical Methods, Design of Experiments and Statistical Quality Control*. Singapore: Springer, 2018.

[95] M. P. Brundage, W. Z. Bernstein, K. C. Morris, and J. A. Horst, "Using graph-based visualizations to explore key performance indicator relationships for manufacturing production systems," *Proc. CIRP*, vol. 61, pp. 451–456, Jan. 2017.

[96] J. Sauro and J. Lewis, *Quantifying the User Experience—Practical Statistics for User Research*. San Mateo, CA, USA: Morgan Kaufmann, 2012.

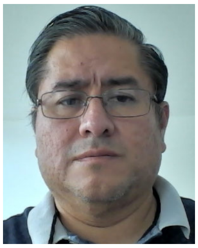


JESUS ANSELMO FORTOUL-DIAZ received the bachelor's degree in mechatronic engineering and the M.Sc. degree in intelligent systems from Tecnológico de Monterrey, in 2013 and 2016, respectively, where he is currently pursuing the Ph.D. degree with the Engineering Science Program. He was a Production Support Analyst with Audi Mexico, for three years. His research interests include smart factory, Industry 4.0, robotics, artificial intelligence, computer vision, and automation.

LUIS ANTONIO CARRILLO-MARTINEZ (Senior Member, IEEE) received the engineering degree in electronics systems from Universidad Autónoma de Tlaxcala (UATx), in 2007, and the M.Sc. and Ph.D. degrees in electronics science from Instituto Nacional de Astrofísica, Óptica y Electrónica (INAOE), in 2011 and 2016, respectively. He is currently a Research Professor with the Department of Mechatronics, Tecnológico de Monterrey. His research interests include mixed integrated circuits design, high-speed ADCs, smart factory, Industry 4.0, cyber-physical systems, the IoT, and automation and control.

ADOLFO CENTENO-TELLEZ received the engineering degree in informática from Instituto Tecnológico de Orizaba, Mexico, in 2000, and the Ph.D. degree in software engineering from Universidad Popular Autónoma del Estado de Puebla, Puebla, Mexico, in 2014. His current research interests include agile methods, applications of artificial intelligence techniques in software engineering, and cloud and quantum computing.

FROYLAN CORTES-SANTACRUZ received the B.S. degree in electronics systems engineering from Universidad Autónoma de Tlaxcala (UATx), Mexico, in 2007, and the M.S. degree in electronics systems quality from Universidad de las Américas Puebla (UDLAP), Mexico, in 2016. He is currently pursuing the Ph.D. degree in engineering sciences with Tecnológico de Monterrey, Mexico. His research interests include the application of distributed DLT systems in smart manufacturing and the development and application of artificial intelligence algorithms specifically those related to reinforcement learning and swarm intelligence for industry and cybersecurity.



IVAN OLMOS-PINEDA received the B.Sc. degree in computer science from Benemérita Universidad Autónoma de Puebla, Mexico, the M.Sc. degree in computer science from the Instituto Tecnológico y de Estudios Superiores de Monterrey, and the Ph.D. degree in computer science from Instituto Nacional de Astrofísica, Óptica y Electrónica. He is currently a full-time Professor with the Faculty of Computer Science, Benemérita Universidad Autónoma de Puebla. His research interests include computer vision, machine learning, and algorithms.

ROBERTO RAFAEL FLORES-QUINTERO was born in Puebla, Mexico. He received the B.Sc. degree in electronics from the Autonomous University of Puebla, Puebla, in 2012, and the M.Sc. and Ph.D. degrees from Instituto Nacional de Astrofísica, Óptica y Electrónica (INAOE), in 2018. He is currently a Research Professor with Tecnológico de Monterrey Puebla Campus. He is part of the national system of researchers, Mexico. His research interests include analog integrated circuit design, energy harvesting, power management systems, automation, robotics, and control systems.