electronics

MDPI

Article

Software Components for Smart Industry Based on Microservices: A Case Study in pH Control Process for the Beverage Industry

Héctor Serrano-Magaña [1], Apolinar González-Potes [2,*], Vrani Ibarra-Junquera [2], Patricia Balbastre [3], Diego Martínez-Castro [4] and José Simó [3]

[1] Instituto Tecnológico Nacional de México, Campus Tlajomulco, Jalisco 45640, Mexico; serranoh239@gmail.com
[2] Laboratorio de Bioingeniería, Universidad de Colima, Km. 9 Carretera Colima-Coquimatlán, Colima 28400, Mexico; vij@ucol.mx
[3] Instituto de Automática e Informática Industrial (AI2), Universitat Politècnica de València, 46022 Valencia, Spain; patricia@ai2.upv.es (P.B.); jsimo@disca.upv.es (J.S.)
[4] Facultad de Ingeniería, Universidad Autónoma de Occidente, Calle 25 No. 115-85 Km. 2 Vía Cali-Jamundí, 760030 Cali, Colombia; dmartinez@uao.edu.co
* Correspondence: apogon@ucol.mx

Abstract: Modern industries require constant adaptation to new trends. Thus, they seek greater flexibility and agility to cope with disruptions, as well as to solve needs or meet the demand for growth. Therefore, smart industrial applications require a lot of flexibility to be able to react more quickly to continuous market changes, offer more personalized products, increase operational efficiency, and achieve optimum operating points that integrate the entire value chain of a process. This requires the capture of new data that are subsequently processed at different levels of the hierarchy of automation processes, with requirements and technologies according to each level. The result is a new challenge related to the addition of new functionalities in the processes and the interoperability between them. This paper proposes a distributed computational component-based framework that integrates communication, computation, and storage resources and real-time capabilities through container technology, microservices, and the publish/subscribe paradigm, as well as contributing to the development and implementation of industrial automation applications by bridging the gap between generic architectures and physical realizations. The main idea is to enable plug-and-play software components, from predefined components with their interrelationships, to achieve industrial applications without losing or degrading the robustness from previous developments. This paper presents the process of design and implementation with the proposed framework through the implementation of a complex pH control process, ranging from the simulation part to its scaling and implementation to an industrial level, showing the plug-and-play assembly from a definition of components with their relationships to the implementation process with the respective technologies involved. The effectiveness of the proposed framework was experimentally verified in a real production process, showing that the results scaled to an industrial scale comply with the simulated design process. A qualitative comparison with traditional industrial implementations, based on the implementation requirements, was carried out. The implementation was developed in the beverage production plant “Punta Delicia”, located in Colima, Mexico. Finally, the results showed that the platform provided a high-fidelity design, analysis, and testing environment for cyber information flow and their effect on the physical operation of the pH control.

Keywords: edge computing; software components; distributed industrial automation systems; Industry 4.0; industrial cyber physical systems

check for updates

Citation: Serrano-Magaña, H.; González-Potes, A.; Ibarra-Junquera, V.; Balbastre, P.; Martínez-Castro, D.; Simó, J. Software Components for Smart Industry Based on Microservices: A Case Study in pH Control Process for Beverages Industry. *Electronics* **2021**, *10*, 763. https://doi.org/10.3390/electronics10070763

Academic Editor: Juan M. Corchado

Received: 21 January 2021
Accepted: 18 March 2021
Published: 24 March 2021

1. Introduction

The ANSI/ISA-95 standard was established to facilitate the integration of business functions and control systems in productive enterprises. This establishes a hierarchical model to indicate systematically, and through a defined structure, the exchange of information between the control systems and the management of a plant. A typical CPS architecture comprises two main layers: a physical layer and a cyber layer; see Figure 1. The physical layer contains a network of sensors and actuators, which collect information from the environment and actuate on a given physical system, whereas the cyber layer represents the decision-making framework and the communication infrastructure [1].

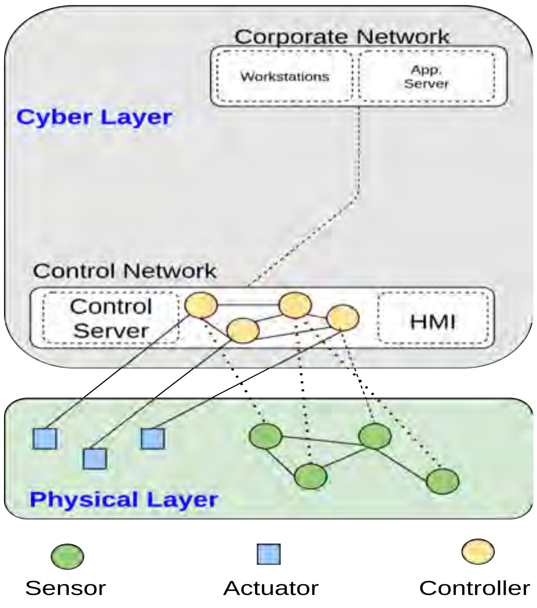


Figure 1. A typical cyber-physical system.

The applications of the fourth industrial revolution require great flexibility on the part of industrial cyber-physical systems (ICPS) to achieve optimal operating points that integrate the entire value chain of a process. This requires the capture of new data that are subsequently processed at different levels of the hierarchy of automation processes, with requirements and technologies according to each level [2–5]. This presents challenges related to the incorporation of new functionalities in the processes and the interoperability between them with the collaboration between distributed nodes. This is easier to achieve with flat architectures based on services that allow horizontal interaction between components based on information.

On the other hand, addressing these solutions based on conventional criteria and adding new hardware nodes when new functions are required, demands higher implementation costs, the reconfiguration of systems to support the new data transmitted through the network without affecting the fulfilment of real-time requirements, and difficulties in the exchange of information between the platforms of different manufacturers. This context requires a more flexible approach than adding new nodes, and that is the reconfiguration of existing nodes according to the requirements of new functionalities in the system. Then, as a consequence of the increase in resources in the hardware nodes, technologies such as virtualization make the inclusion of new components more flexible at very low costs, which is very convenient in terms of the flexibility and exchange of information between components. Obviously, as in traditional approaches, it is necessary to verify the fulfilment of deadlines in real-time systems.

Small software components have recently emerged as a great possibility to attack complex problems with an interconnection of small functionalities. This is how microservices-based software architectures have recently been used in a meaningful way to solve a vast variety of software complexity problems that need to adjust to the changing requirements. A microservice, in terms of [6], is considered as a small application that can be deployed, scaled, and tested independently, with a single responsibility. The large applications developed with the monolithic approach can be separated into smaller isolated mini-applications, providing a specific service, giving the advantage or facility of modifying the services that are required without affecting other services. The microservice can therefore replace the monolithic architecture, allowing the development of distributed, lightweight, decoupled, and independent service applications working together. Then, because the services in this architecture can be separately deployed in different nodes or processes there may be an overhead of communication between them, affecting the performance of the distributed system, because of the high latency network communication, it is therefore necessary to provide an effective protocol of network communication.

In this work, we propose a component-based microservice framework which enables a series of interconnected components to support flexibility, interoperability, and robustness in the demanding and flexible applications of Industry 4.0. To maintain a fast and flexible system reconfiguration and keep the functionality of plug-and-play enabled, the components must support reconfiguration and online addition or removal, without losing or degrading the robustness from previous developments. To achieve the challenges mentioned above, component-based microservices integrate communication, computation, storage resources, as well as real-time capabilities, by using container technologies and the implementation of microservices, as well as the computational isolation of the microservices between them, and middleware based on the publish/subscribe pattern, which allows the decoupling of each microservice and defines a constructive pattern of the software to be developed. An important aspect to take into account is that enabling the requirements set out above, some strong points are included, such as it allows the development of applications with code that are easier to maintain because of the separation of the services, it can be updated and scaled in different programming languages and even using diverse middleware stacks and data tiers for different services [6] with the use of a bridge that integrates the data flow to the data distribution service (DDS) middleware, which is the communication backbone of this architecture. As a case study, a pH control approach robust to model uncertainties including the scaling-up of the process is presented. The pH control is crucial in many chemical and biological processes. The pH automatic control is mainly implemented by manipulating the flow rate of the titration streams. However such processes present complicated dynamics with inherent non-linearities, and their modeling, parameter estimation, and control are challenging tasks.

The main contributions are summarized as follows:

- A holistic framework for automation in the bioprocess industry, with ease of integration and scalability of software and hardware components by utilizing a data-centric communication backbone to manage information exchange and reliably orchestrate system components together.
- A proposed component-based software design pattern that allows associating microservices with containerization technologies and the data-centric communication model. The main focus here is to scale applications to industrial implementations.
- As an application design process, a set of components interconnected with each other are proposed to follow a plug-and-play technique. The connection between components supports the data-centric model of the framework. The software components are mainly oriented to the bioprocess industry; however, it can be a starting point for other industries.
- Scalability to a wide range of applications in-process monitoring and control, analysis, visualization, etc., mainly oriented to the bioprocess industry, as demonstrated in a case study.

Finally, from a general aspect, the work contributes to the development and implementation of industrial automation applications by bridging the gap between generic architectures and physical realizations through the use of container technologies, the concepts of microservices, and the decoupling of each microservice with a middleware based on the publish/subscribe pattern.

The outline of this paper is as follows. Section 2 presents an overview of related works, such as other performances and comparative studies. In Section 3 a detailed design of the architecture, with different components and microservices, is exposed. Section 4 shows the process of design and implementation with the proposed framework through the implementation of a complex pH control process, ranging from the simulation part to its scaling and implementation to an industrial level. The validation, results, and discussions of its implementation at the juice production plant are carried out in Section 5, and finally, Section 6 concludes this work.

2. Related Works

ICPS is considered an enabling technology for Industry 4.0 [7,8]. Many works about software architectures have drawn attention to the design and implementation of ICPS. In [9], a software architecture for the industrial internet of things (IIoT) composed of four layers (Sensing/THINGS Layer, Data Provider Layer, Fog/Edge Computing Layer, and Applications/Services Layer) is proposed. It is a conceptual model with a low level of abstraction, oriented to the design and development of real-time applications for IIoT, which directly relates to ICPS. A work that contributes to the architectures for industry 4.0 and that reduces the gap between designs and physical implementations is presented in [10], where a platform that supports the integration of dynamic data-based decision support systems into real-time mass customization manufacturing environments is discussed. One of the challenges discussed in this paper is related to data acquisition and its personalized management; this is achieved by providing interoperability through a middleware between field devices, enterprise databases, and data-based decision support systems. Other important works that have been highly considered in the literature [11–13], which include different aspects of ICPS, such as Cloud, Fog, and Mist computing platforms for production systems, where the interrelation between the different layers are studied. Other works focus more on data analysis, knowledge representation and supervisory models [14–17], however, they mainly reference architectures and do not present details of implementation or integration with physical systems. To relate our work to the state of the art, there are three main challenges for the proposed architecture, and they have also been considered in the literature: scalability, flexibility, and robustness to design, develop and implement industrial automation applications. To gather these aspects, we focus mainly on container technology, microservices concepts, and publish/subscribe based on Data Distribution Services (DDS) middleware.

Container technology has also enabled flexibility and scalability. In [18–20], the authors focused their work on demonstrating the feasibility from a time constraints point of view and implementing controllers directly as programmable logic controllers (PLC), also supporting time constraints and general features for real-time applications. Other works [21–24] present its great utility on the cloud computing side, showing its applicability for deployed applications in microservice architecture.

The concept of microservices-based architectures for industrial edges has also been widely considered in recent years [25–29]. The decomposition of functionalities into small services has allowed the flexibility and scalability of industrial software [30,31]. Many of these works present proposals for the integration of microservices with the industrial world, generally including different layers of abstraction. Another important aspect in microservices taken into account is implementation using container technology.

Different publish/subscribe system approaches have been addressed depending on the application domain; one of them is the Data Distribution Service (DDS) specification, which is provided by the Object Management Group (OMG) [32], and allows developing distributed applications centered in the data, with asynchronous and decoupled communications requirements through the publish/subscribe architectural pattern. Some other works include the publish/subscribe paradigm via the Message Queue Telemetry Transport (MQTT) protocol [33,34], which allows to gather security and scalability; however, performance in some time constraints is difficult to make work with MQTT.

Online new components, without the need for reconfiguring the system, keep temporal and functional restrictions in industrial automation systems. Many of the works listed above have some characteristics presented in this paper; however, they are not gathered in an integrated way. Each of the software components presented, that meet the concept of microservices, are implemented with container technology and integrated through the publish/subscribe pattern. Generally, each component or microservice is unknown to each other, with it being very useful to develop loosely coupled distributed applications, enabling timely data dissemination from publishers to subscribers, delivering information, and processing data in diverse application scenarios with specific requirements.

3. Component-Based Microservices for Industrial Automation

Currently, products are becoming more and more personalized, forcing many industrial sectors to produce fewer quantities, making processes much more varied and complex and compelling production to be modular, flexible, and scalable without losing robustness. To deal with this, we propose a component-based microservices framework, implemented in lightweight software containers and interconnected among them through a middleware oriented to the publication and subscription of messages. The main idea of the proposed framework is to use components-based microservices, allowing easy implementation, scalability, and fast maintenance, without losing or degrading the robustness of other processes or even previous developments. The challenge is to maintain the previous conditions in the face of modified or new developments, this can be possible, with a system based on microservices, components, containers, and a strong and secure communication pattern.

The Component-Based Data Distribution Service (CBDDS) is a commercial term for the comprehensive, integrated suite of the following seven OMG open standards and supports architecture development at a higher level of abstraction. We have adopted CBDDS to represent our architecture with a superior level of detail because CBDDS addresses five architectural tenets: Open Architecture (OA), Model Driven Architecture (MDA), Component-Based Architecture (CBA), Service Oriented Architecture (SOA), and Event-Driven Architecture (DOA). Figure 2 represents container communication and the connection of different components between them, using the publish/subscribe pattern supported through DDS. Five different message structures have been considered to represent communication between microservices (see Table 1).

Table 1. General data model.

topic1.idl	**topic2.idl**	**topic3.idl**	**topic4.idl**	**topic5.idl**
struct topic1 {string id; bool onoff; int32 data;}	struct topic2 {string id; bool onoff;}	struct topic3 {string id; string time; float64 data;}	struct topic4 {string id; bool onoff; int32 data1; float64 data2;}	struct topic5 {string id; int32 data1; int32 data2;}

In Figure 2 and Table 1, the organization of the components is focused on the general requirements in the bioprocess industry; however, in a conceptual way it is a good starting point for many other possible industries, the constituent elements serve as guidelines for the development of control and automation systems. Clearly, from a plug-and-play concept, all sensor components (analog and digital) communicate with the controller components, HMI, alarms, and Data Base (BD), the controller must receive HMI data (references set by users) although they may come from elsewhere, sensor data and publish control actions.

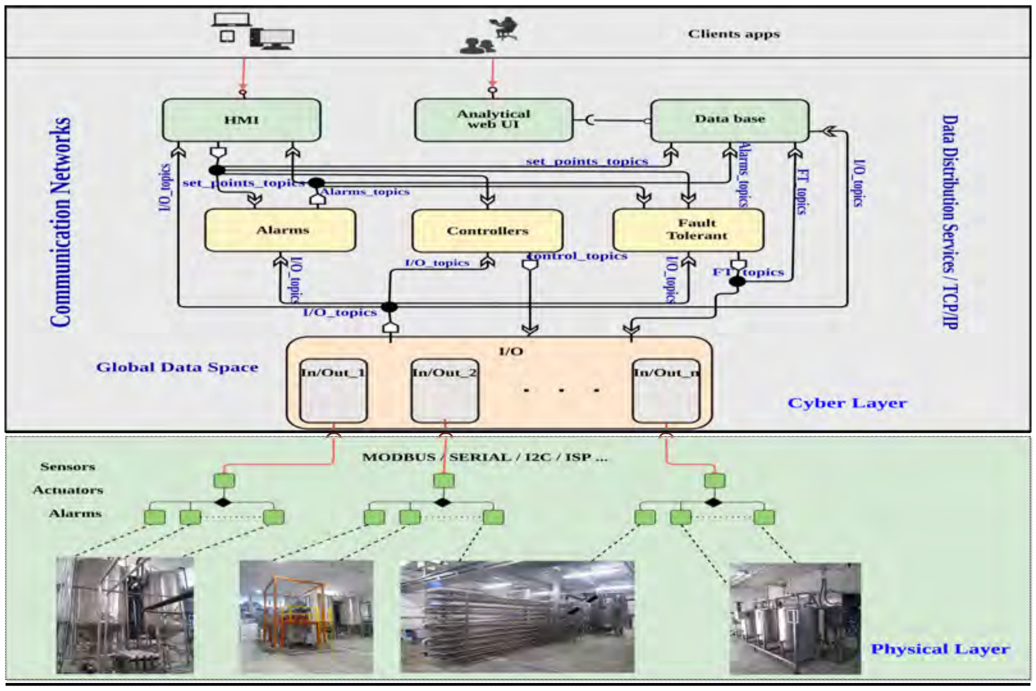


Figure 2. General component-based DDS architecture.

4. Procedure for Developing the Software Components

Recently, research has been carried out to standardize implementations for Industry 4.0. However, if, on the one hand, conceptual architectures lack a physical implementation, special implementations often cannot be generalized to other systems, and it still lacks a physical realization of a generic architecture. Thus, generic proposals focus on reference architectures and specific ones on use cases. To show the process of designing an application with this framework, initially a presentation of a case study is made to contextualize the problem and the need for development. In terms of the control process for the beverages industry, the simulation model needed to be created first. The implementation of an industrial process must be linked to the requirements of the system—in this case, compliance with Industry 4.0 generalities and particular requirements of the control process are presented. As a design process to follow a plug-and-play based technique of the proposed components in this framework, the following steps must be followed:

1. Application must be decomposed into microservices with the associated data model.
2. Identify the containers that will host the services.
3. Associate in each container whether the service is offered or required with a publication or subscription.
4. Identify non-functional requirements.

In this step, it is important to identify the nature of the data and identify it with the data model supported by the framework. Once the above steps are completed, a process design can be generated according to the design pattern based on the components presented in previous sections. Thus, a mapping from microservices to DDS can be established—i.e., an offered microservice is mapped to a publication and required microservice to a subscription. With the design achieved, the implementation characteristics for its scalability must be presented. Among others, hardware, programming languages, containers, etc. must be specified. The process scaled to the industrial level can be verified with simulation-level design. A comparison with the traditional standardized model for industrial implementations based on compliance with system design and implementation requirements is performed.

4.1. pH Control for Beverages Industry as a Case Study

At the Punta Delicia juice production plant , located in Colima, México, different types of drinks are produced. There are mainly juices such as banana, soursop, and raspberry; water in alkaline presentations, with and without gas; formulated drinks with flavors and alcohol; as well as products and formulations for third parties. Different processes are addressed in the production plant, however, to demonstrate the process of using the framework, implementation, and results, we will only focus on the part of the process, where the juice is finally stored at a certain temperature in tanks of 15,000 L until the desired conditions are obtained, then it is processed in two tanks according to a final formulation. Within the formulation processes for a final drink, the product must be stabilized at a certain pH value depending on its formulation, this is carried out in a mixing tank with a capacity of 2000 L. This means that the pH reference values can range from 4 to 9 and will depend on the future product characteristics. From an industrial point of view, these considerations demand and require a very robust controller and implementation.

4.2. pH Control Approach

For the final formulation process of any of the products made at Punta Delicia and mentioned above, pH control is crucial and each of the products to be processed has its own requirements, characteristics, and considerations. In addition, such processes present complicated dynamics with inherent non-linearities, and their modeling, parameter estimation, and control are challenging and delicate tasks. Thus, the design of high-performance model-based control algorithms for bioprocesses is frequently hampered by its complexity, which can include poorly understood nonlinear functions, due to the limited process knowledge, nonlinearities, unmodeled dynamics, unknown internal and external noises, environmental influences, and time-varying parameters. Hence, the problem considered in this work was an easily scalable pH controller that forces the process measurements to follow a desired time-varying reference, despite uncertainties like scaling-up processes. The control algorithm implemented is based on a master-slave synchronization, where the real process is the slave, while the master is generated by the real-time simulation of the closed-loop mathematical model of the process. Then the objective is the pH control with minimal process information, considering a time-varying reference ranging between the basic and acid region. This algorithm forces the process measurements to follow a desired time varying reference, despite uncertainties like scaling-up processes. For this purpose, a master–slave synchronization scheme is used derived by [35,36]. The design of a high-performance model-based control algorithms is frequently hampered by its complexity. Moreover, the poor understanding of nonlinear functions, due to the limited process knowledge, nonlinearities, environmental influences, and time-varying parameters, lead to hardly controlled problems that certainly can be tackled by data-based alternatives, such as intelligent ones [37–39]. However, the need of easily implemented solutions, robust to unmodeled dynamics, unknown internal and external noise, could be tackled by simple robust nonlinear controller that can be easily implemented. The advantages of this concept are that the syntonisation can easily be performed for a simplified model and the controller can be designed such that the real process is synchronized with the master in spite of the bounded unknown dynamics and perturbations in the real plant.

$$\dot{X} = f(X) + \sum_{i=1}^{m} g_i(X)u_i \tag{1}$$

$$\dot{z} = f(z) + \Delta f(z) + (\sum_{i=1}^{m} g_i(z) + \sum_{i=1}^{m} \Delta g_i(z))u \tag{2}$$

$$y = h(z) \tag{3}$$

where x represents the states of the system of Equation (1) and takes values of $x \in \mathbb{R}^n$; in the same way, z represents the states of the system of Equation (2) and takes values of

$z \in \mathbb{R}^n$. u and u_i are the control inputs and take values of $u \in \mathbb{R}^m$. y represents the system output and takes values of $y \in \mathbb{R}^m$. $f(\cdot)$ $\Delta f(\cdot)$, $g_i(\cdot)$, $\Delta g_i(\cdot)$ are smooth vector fields on an open set $U \in \mathbb{R}^n$.

The initial model was taken from Ali nejati et al. [40], in which a pH neutralization process is considered, where the flow rate of the acid and the buffer, base, and effluent flow rates are given by q_1, q_2, q_3, and q_4 respectively. It is assumed that the mixing of the process is perfect, the volume (V) of the tank is constant and that there is complete solubility of the ions. The dynamic equation is given by:

$$\dot{x_i} = \frac{q_1}{V}(w_{1i} - x_i) + \frac{q_2}{V}(w_{2i} - x_i) + \frac{q_3}{V}(\alpha_i - x_i) \qquad i = 1, 2, 3 \tag{4}$$

where q_1 and q_3, are the control inputs. The model parameter identification is developed in Ali Nejati et al. [40].

Using a single control input, Equation (4) can be represented by:

For the pH control system, Equation (1) can be represented as follows:

$$\begin{aligned}
\dot{x_1} &= \frac{q_1}{V}(w_{11} - x_1) + \frac{q_2}{V}(w_{21} - x_1) + \left(\frac{\alpha_1 - x_1}{V}\right)u \\
\dot{x_2} &= \frac{q_1}{V}(w_{12} - x_2) + \frac{q_2}{V}(w_{22} - x_2) + \left(\frac{\alpha_2 - x_2}{V}\right)u \\
\dot{x_3} &= \frac{q_1}{V}(w_{13} - x_3) + \frac{q_3}{V}(w_{23} - x_3) + \left(\frac{\alpha_3 - x_3}{V}\right)u
\end{aligned} \tag{5}$$

Similarly, Equation (2) is rewritten as follows:

$$\begin{aligned}
\dot{x_1} &= \frac{q_2}{V}(w_{21} - x_1) + \frac{\alpha_1 - x_1}{V}u_1 + \frac{w_{11} - x_1}{V}u_2 \\
\dot{x_2} &= \frac{q_2}{V}(w_{22} - x_2) + \frac{\alpha_2 - x_2}{V}u_1 + \frac{w_{12} - x_2}{V}u_2 \\
\dot{x_3} &= \frac{q_2}{V}(w_{23} - x_3) + \frac{\alpha_3 - x_3}{V}u_1 + \frac{w_{13} - x_3}{V}u_2
\end{aligned} \tag{6}$$

Finally, Equation (3) represents the output of the system; for the pH control, the output is the pH value and for this case it is represented in the following way [40]:

$$h(x, y) = -x_1 + x_2 - x_3 C_{x3} + 10^{-y} - 10^{y - pKw} = 0 \tag{7}$$

where C_{x3} is a function of pH and the dissociation constants for the ith species and for anions of diprotic weak acid (H2A) are described as follows [41]:

$$C_{x3} = \frac{2 + 10^{pK_2 - y}}{1 + 10^{pK_2 - y} + 10^{pK_1 + pK_2 - 2y}} \tag{8}$$

$y = pH$, pKw is the dissociation constant of water, 10^{-14}, pK_1 and pK_2 are the equilibrium constants for the chemical reactions in the system.

4.3. Controller Design

Consider the nonlinear affine system described by:

$$\dot{x} = f(x) + g(x)u, \qquad y = h(x) \tag{9}$$

Here, the feedback linearization method is used to transform the nonlinear pH systems shown in Equations (5) and (6) into linear systems in order to apply linear control techniques such as PD, PI, PID, and others. The objective is that by means of differential geometry we can generate a differential equation that relates the output y to a new input v. The system has relative degree of r if the following conditions are satisfied:

$$L_g L_f^i y = 0 \qquad \leq i \leq r-2 \tag{10}$$
$$L_g L_f^i y \neq 0$$

where L_f is the Lie derivate in direction of f. Assume that the above system is linearizable and has the relative degree of r. The result of the input transformation:

$$u = \frac{v - L_f^r y}{L_g L_f^{r-1} y} \tag{11}$$

is a linear relation between y and v given by:

$$y^r = v \tag{12}$$

The Lie derivatives of Equation (5) with respect to Equation (7) and Equation (6) with respect to Equation (7) are given by:

$$L_g y = \frac{dy}{dx} \cdot g(x) = -\frac{\partial h/\partial x}{\partial h/\partial y} \cdot g(x) = -\frac{h_x}{h_y} \cdot g(x)$$
$$L_{g_1} y = \frac{-10^{-y} + 10^{y-pK_w} + \alpha_1 - \alpha_2 + C_{x3}\alpha_3}{V \cdot hy} \tag{13}$$
$$L_{g_2} y = \frac{-10^{-y} + 10^{y-p_{kw}} + w_{11} - w_{12} + w_{13}C_{x3}}{V \cdot hy} \tag{14}$$

and

$$L_f y = \frac{\partial y}{\partial x} \cdot f(x) = \left[\frac{hx}{V \cdot hy}\right][f(x)]$$
$$QW = q_1 \cdot w_{11} + q_2 \cdot w_{21} - q_1 \cdot w_{12} - q_2 \cdot w_{22}$$
$$L_{f_1} y = \frac{(q_1 + q_2)\left(-10^{-y} + 10^{y-pk_w}\right) + QW + q_1 w_{13} \cdot C_{x3} + q_2 w_{23} \cdot C_{x3}}{h_y V} \tag{15}$$
$$L_{f_2} y = \frac{q_2\left(-10^{-y} + 10^{y-p_{kw}}\right) + q_2 w_{21} - q_2 w_{22} + q_2 w_{23} C_{x3}}{V \cdot hy} \tag{16}$$

It can be shown that $L_g y \neq 0$ if $0 \leq y \leq 14$, therefore the relative order ir one and the linearization relation is as (10).

Now, if v is output of a PI controller, then it can be written as:

$$v = K_C e + K_I \int e + \dot{y}_d \tag{17}$$

where $e = yd - y$, K_C and K_I are constants.

Figures 3 and 4 show the control structure for both cases: simulated and with a real plant.

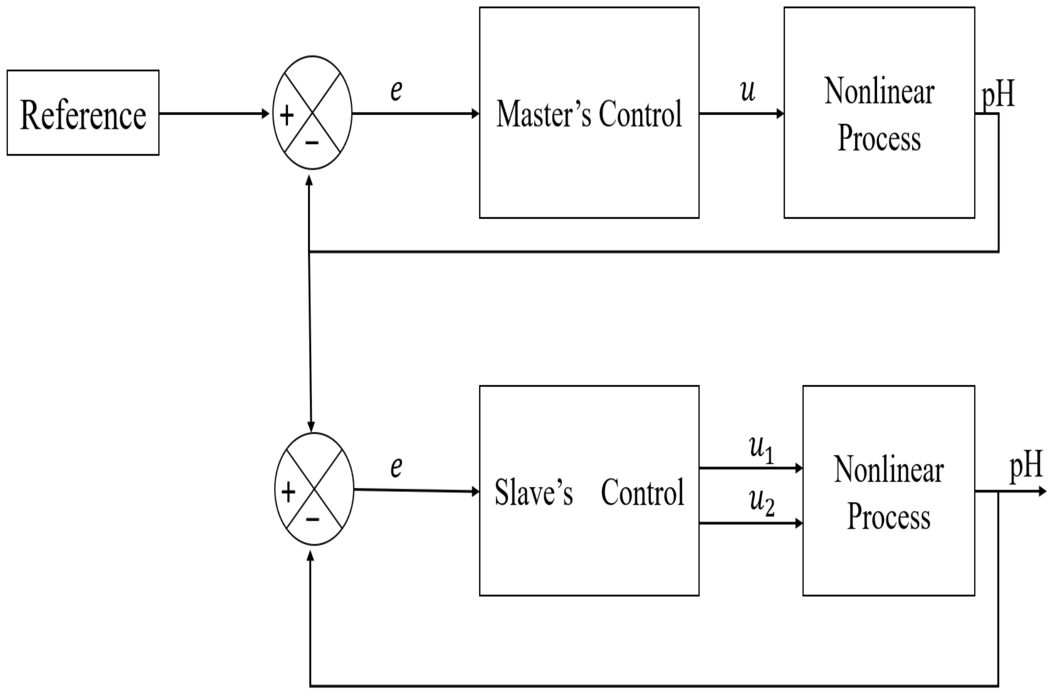


Figure 3. Control structure: master–slave synchronization in a simulation.

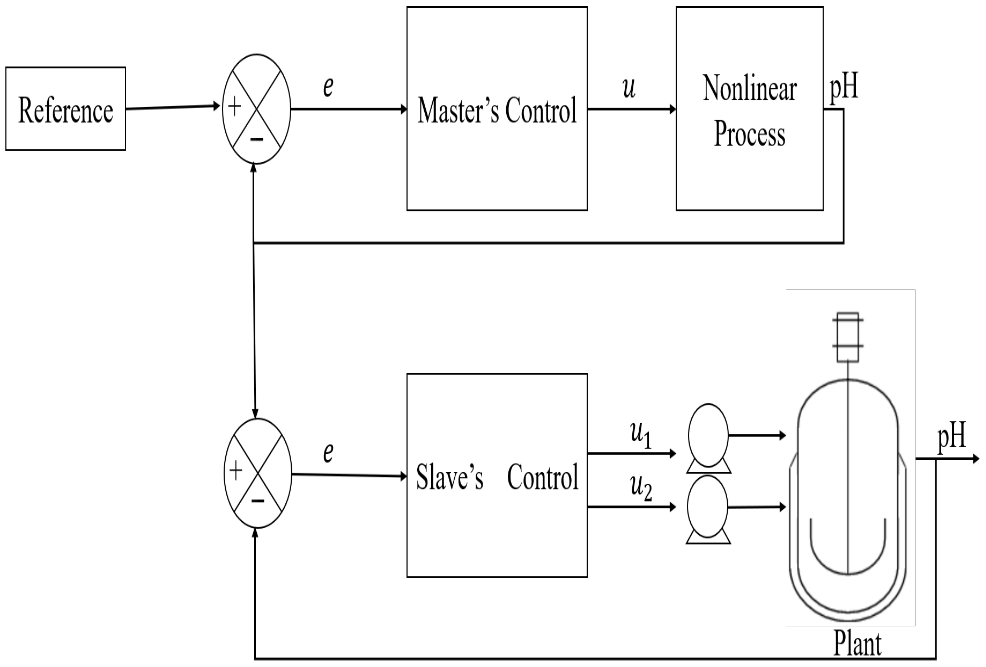


Figure 4. Control structure: master–slave synchronization in a real plant.

4.4. Requirements for pH Control Process Implementation

Industrial bioprocesses require advanced modularity, flexibility, and scalability of production and must be able to maintain the reliability of many interconnected elements and devices. In addition to the requirements imposed by the pH control approach and complying with the above, requirements from the implementation should be added as follows:

1. The architecture of components should be designed to promote scalability. (scale-out)
2. Features should be completely isolated from each other in time and space.

3. Innovation should not be constrained concerning supporting new input kinds, new target platforms, new visualization, new strategies, etc. Additionally, functionalities should be implemented in the most effective programming language.
4. The design should not be limited to new types of inputs, new destination platforms, data visualization techniques, new strategies, new drivers, etc.
5. The platform should be as modular as possible to facilitate the individual functionalities updates and upgrades. Additionally, adding new features should be as transparent as possible for the currently running system.
6. The minimum required execution in the fastest processes must be at least 1 s of performance.
7. All the data such as sensors, control values, references, etc. must be stored in real time.

The fulfilment of these requirements will allow the scale up of the pH control process to an industrial level. The basic framework presented in the previous section allows guiding the design from the defined components. Thus, making a functional identification of the components and the different services offered and required by each component, with their respective functionality, allows obtaining a model of the implementation.

The Table 2 lists the different containers needed for the implementation, the different services offered and required, the type of service (non-functional), and the data model to be used among the different ones provided by the framework.

Table 2. Relationship of implementation features with containers, topics, services, and their data structure.

Container (id)	Topic	Type	Service	Data Model
Recipe_Sensors	error	Event	Offer	topic2.idl
	rec_pHm	Periodic (1 s)	Offer	topic3.idl
Peri_pump1	error	Event	Offer	topic2.idl
	acid_pH	Event	Requires	topic1.idl
Peri_pump2	error	Event	Offer	topic2.idl
	basic_pH	Event	Requires	topic1.idl
PH_Control	rec_pHm	Event	Requires	topic3.idl
	acid_pH	Periodic (1 s)	Offer	topic1.idl
	basic_pH	Periodic (1 s)	Offer	topic1.idl
pH_visualization	rec_pHm	Event	Requires	topic3.idl
	error_type	Event	Requires	topic1.idl
	ref_alarm	Event	Offer	topic1.idl
	set_points	Periodic (1 s)	Offer	topic3.idl
Alarms	error	Event	Requires	topic2.idl
	error_type	Event	Offer	topic1.idl
	ref_alarm	Event	Requires	topic1.idl
DB_Recipes	rec_pHm	Event	Requires	topic3.idl
	error_type	Event	Requires	topic1.idl
	ref_alarm	Event	Requires	topic1.idl
	set_points	Event	Requires	topic3.idl

4.5. Component-Based Microservices for pH Control Process Implementation

In large industrial control systems, the cyber layer is typically composed of a Supervisory Control and Data Acquisition (SCADA) system [42]. The architecture of the case study implemented is presented in Figure 5, where one can see the assembly of the different components, which are integrated through the different connections between them. The I/O block is made up of three containers, Recipe_sensors, publishes an error_topic in case there is a reading error in the pH sensor and also publishes the topic rec_pHm periodically, the components Peri_pump1 y Peri_pump2 publishes error topics in the case of problems with Modbus network connections, acid_pH, and basic_pH are data subscriptions with the control values and allow us to act directly on the respective peristaltic valves. Similarly, the controllers block in this case is built with a single container subscribing to the services of rec_pHm (pH sensor) and set_points (the reference). DataBase (DB) implements a container with the necessary resources to store the data of this application, thus, each application can implement its DB distributed and independent. In Table 2, we can see the relationship of the different type topics, with their data structure, the topics used, and implementation requirements, relating the application with the general structure of components presented in Figure 5.

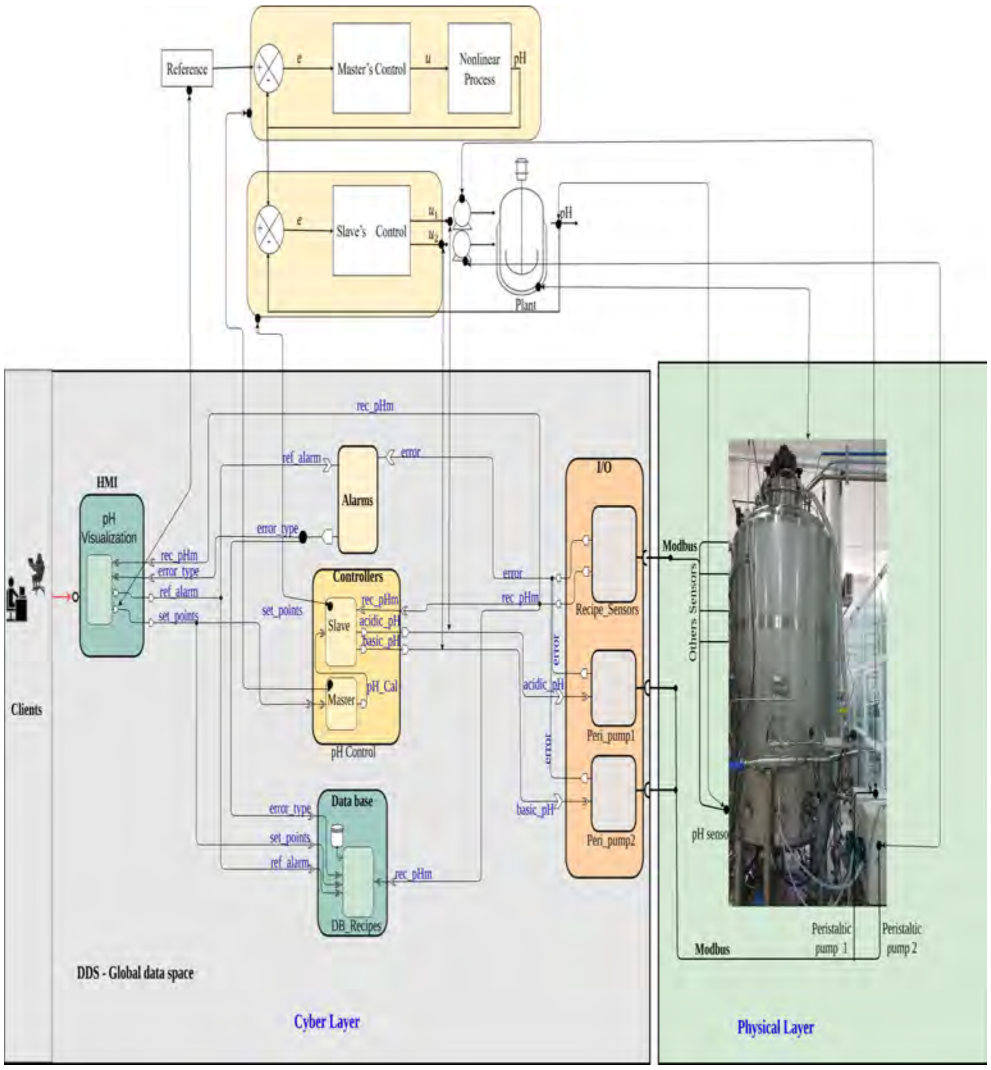


Figure 5. Component-Based DDS architecture design for pH control.

4.6. Implementation Details

The implementation begins with hardware architecture, mainly with the selection and location of equipment and interconnection between them. Wireless and wired equipment is installed, computers for sensing and actuation are wired for the greater certainty of communication, and other equipment such as displays and manual commands can be implemented in both wired and wireless ways. In the embedded-PC platform, we set up devices with the following characteristics: Intel Celeron J3060 1.60 GHz, 2 GB in RAM and 32 GB of emmc, 2 PC's with web services, controllers and monitoring functions Intel Xeon E3-1225V5 3.30 GHz, 32 GB DDR4, and a server with Intel Xeon C620, 32× 284 DDR4 and

15.36 TB to store the different databases. Additionally Access Point, switches and routers for network management, 4 display monitors for HMI functions located in the control room, and finally a series of mobile devices that allow the visualization and execution of commands from any space in the industrial plant. Figure 6 shows the distribution of the hardware used so far for the control and current operation of processes implemented in the juice and beverage production plant at Punta Delicia.

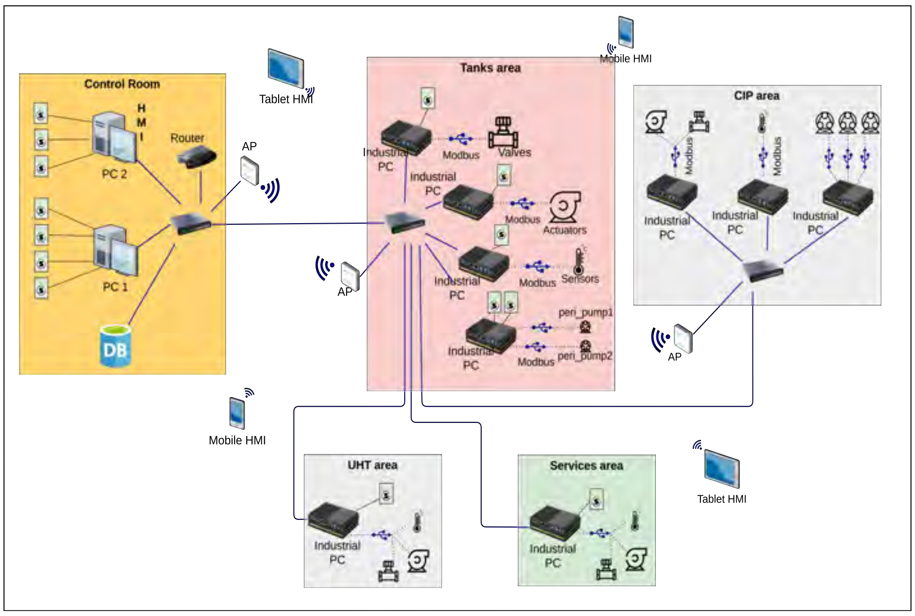


Figure 6. Embedded Personal Computer (PC) and devices platform hardware for industrial automation process in Punta Delicia Plant.

In particular, for the implementation of pH control and all the functionalities specified in Table 2, only the tank area and the control room are required. Singular containershave been used for this implementation, and other containers can be implemented. However, from the point of view of size and ease of migration, we have adopted singular ones instead of others. As shown in the Figure 7, Recipe_Sensors is implemented in industrial PC3, Peri_pump1, and Peri_pump2 on industrial PC4, pH_Control and Alarms on PC1, HMI on PC2, and finally, DB_Recipes on the DB server, understanding, that the containers can be copied and executed like any other file. In the same way that is justified in [43], open-source software presents a great opportunity and minimal costs for the development of industrial platforms. The programming languages used are C and C++ for access to physical drivers (Modbus, serials, and others), low footprint, DB writing, and general operations management. Python was used for the controller implementation because of its mathematical benefits. Python is suitable for mathematical expressions, plus it has a large developer community for a wide range of libraries and data science analysis, and others such as javascript for WEB HMI. Finally, in Figure 8, we can see the real tank and KKTS peristaltic pumps with a stepper motor and a capacity of up to 1600 mL/min as actuators.

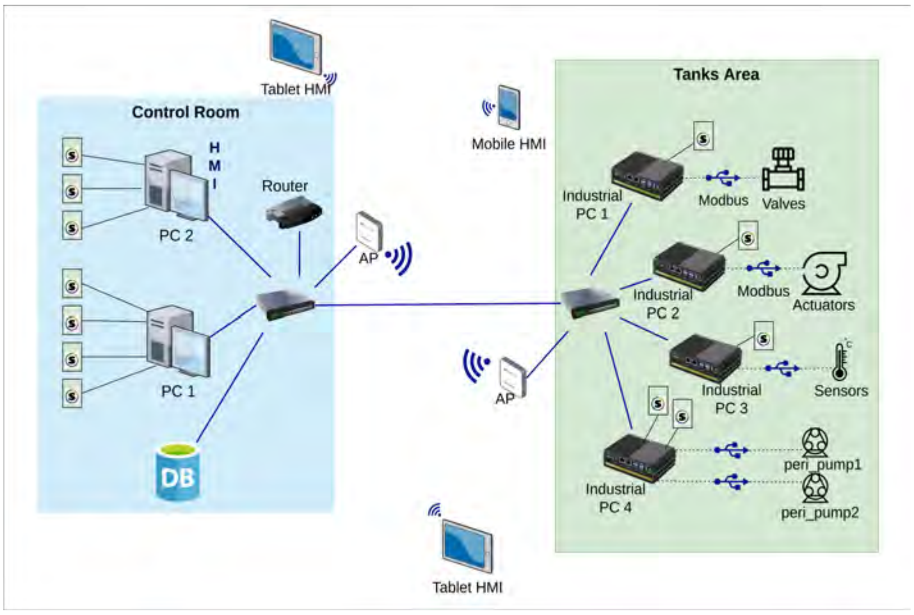


Figure 7. Embedded PC and devices platform hardware for control pH process.

Figure 8. Mixing tank for preparation of formulations in the Punta Delicia plant.

5. Results and Discussions

In this section, we present the verification of the case study developed with the proposed framework and a qualitative comparison with traditional implementations. It is worth mentioning that the focus in this case study was not on the robustness and optimization of the control algorithm but on the functionality of the framework for testing the system under study, as well as compliance with design and implementation requirements.

As can be seen from the results of this case study below, the developed framework succeeded in providing a consistent integration between a hardware/software infrastructure and an industrial-scale control process. Through this integration, the effect of the control logic, which was implemented in Python drivers, and access to the Database in C++, visualization tools in HTML and Javascript, was tested and the response of the industrial process with the simulated process control logic was analyzed and compared.

The parameters, values, and answers for this test can be as follows. In Tables 3 and 4, the parameters and values used in the tests of the simulated control process and the scaling to the industrial level can be observed.

Table 3. Parameters for simulation tests.

Parameter	Value	Parameter	Value
$Volume$	1000 mL	α_i	0.0255
K_C	0.0510	K_I	0.05165
pK_w	10^{-14}	q_1	0.966 mL/s
q_2	0	w_{1i}	0.15
w_{2i}	0	$pK1$	6.34
$pK2$	10.25		

Table 4. Parameters for industrial tests.

Parameter	Value	Parameter	Value
$Volume$	1000–2000 L	α_i	0.0255
K_C	0.0510	K_I	0.05165
pK_w	10^{-14}	q_1	0.966 mL/s
q_2	0	w_{1i}	0.15
w_{2i}	0	$pK1$	6.34
$pK2$	10.25		

Table 5 shows the different concentrations of acid and base in each test, as well as the volume of product used in tests carried out in a real agitated tank.

Table 5. Concentrations and volume for tests in agitated tank.

Test	Acid	Base	Volume
1	Citric acid at 0.19 M	Sodium hydroxide at 0.9 M	1500 L
2	Citric acid at 0.19 M	Sodium hydroxide at 0.9 M	1500 L
3	Citric acid at 0.15 M	Sodium hydroxide at 0.7 M	1000 L

A set of references in a container publishing the current value every second have been additionally implemented for comparative purposes with simulation in the different tests, a step every 10 min in growth and then in decrease, with references ranging from pH 5 to 9 was established, according to the usual operating region of the processes in the production plant. Both, simulated pH process control and scaled industrial process then have the same reference.

Figure 9 shows the tracking of a trajectory where the reference value changes every 10 min. As result, it can be seen how the slave (in this case the real tank) reaches the references and stabilizes the solution in the tank to the reference value, showing the robustness of the controller for an industrial-scale pH control problem. It is important to mention that the industrial process presents the signals with some noise that can be eliminated with filters, however, it is not the objective of this work to address that aspect. Figure 10 consisted of maintaining the value of the reference the whole time, also showing that the slave is synchronized with the master, regardless of some operating conditions.

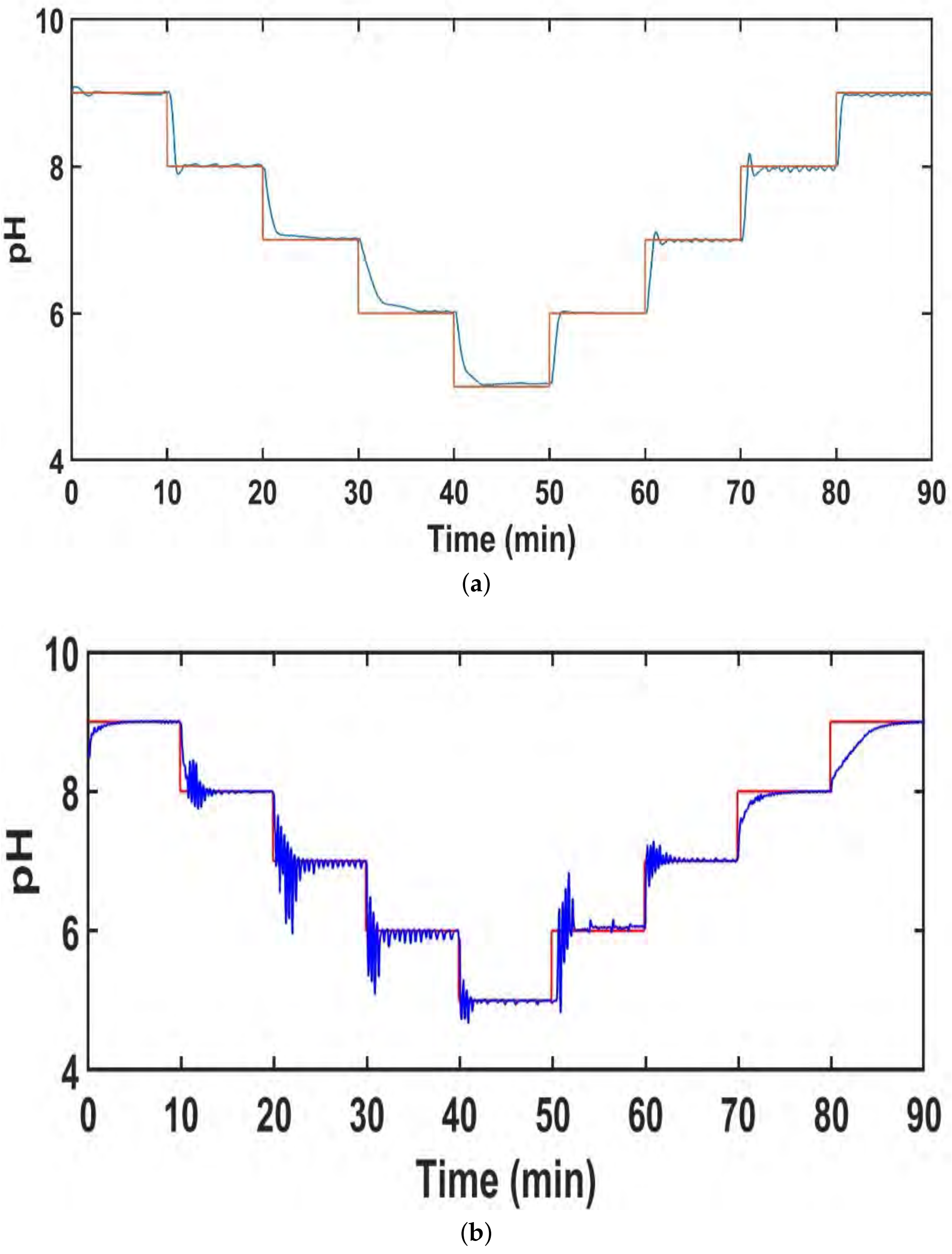
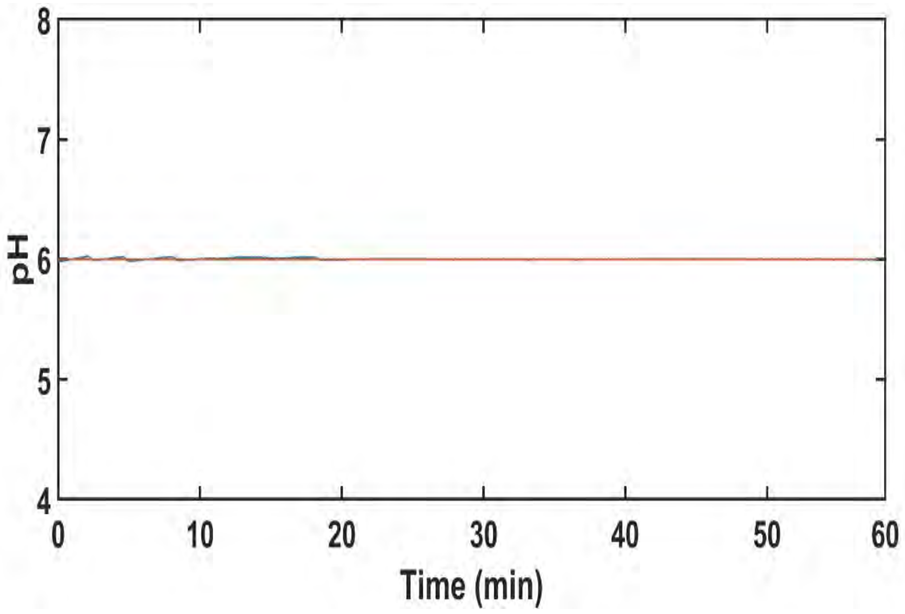
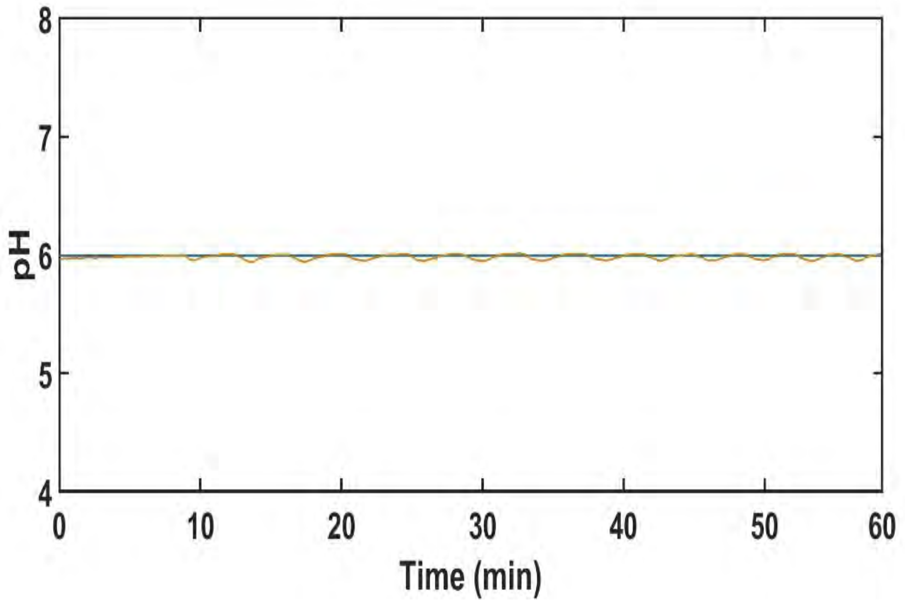


Figure 9. The figure shows the result of the pH control for simulated and real processes, respectively, by master–slave synchronization, in a stirred tank with a volume of 1500 L for scaled industrial process. (**a**) pH reference and pH measured for simulation, (**b**) pH reference and pH measured for industrial-scaled process.

Comparing the Figure 9 of the simulated process with of the scaled process at the industrial level, it is observed that while the simulated process reaches the reference on average between 2 and 4 min, the industrial scale-up process takes approximately 4 to 6 min on average. Tables 3 and 4 show the parameters for both processes and the principal difference is the Volume of the tank, while for the simulated case it is 1 L, for the industrial case, the Volume is between 1000 and 2000 L, which is evidence that this controller can be scaled up to industrial level with a minimum effort.



(**a**)



(**b**)

Figure 10. The figure shows the result of the pH control for the simulated and real processes, respectively, maintaining the value of the reference the whole time, with a reference value of pH = 6 for simulation and pH = 7 in a stirred tank with a volume of 1500 L for scaled-industrial process. (**a**) pH reference and pH measured for simulation, (**b**) pH reference and pH measured for industrial-scale process.

In [44], a review of control system applications in industrial processes and [45] a review of pH neutralization process control, where most of the bioreactors are supplied with traditional Programmable Logic Controller (PLC)-based automation implementations and most of the PLC supports only PID controllers, are presented. Now, we present a qualitative comparison with traditional implementations, based on requirements from the pH control implementation that indicate the grade of flexibility and scalability of the framework. The next requirements are used for this comparison:

1. The implementation should promote scalability (scale-out).

2. The application must be decentralized and the different elements must be able to communicate directly with each other.
3. Features should be completely isolated from each other in time and space.
4. pH controller should be compatible between different platforms.
5. Innovation should support new target platforms, new visualization, new strategies, etc.
6. Implementation must be able to be performed in the most effective programming language.
7. The design should not be limited to new types of inputs, new strategies, new drivers, etc.
8. The platform should be as modular as possible to facilitate the individual functionalities updates and upgrades.
9. New features should be as transparent as possible for the currently running system.
10. The minimum required execution in the fastest processes must be at least 1 s of performance.
11. All data, such as sensors, control values, references, etc., must be stored in real-time.

In Table 6, it can be seen that traditional implementations do not meet many requirements of modern industry and can be a complex task with high development costs. However, with the application construction model presented in this paper, commissioning the service of control is fully feasible, without altering previous developments or those that are currently operating. Finally, rgw experimental results correspond to the formal results obtained from the simulation model, which indicates that the platform implemented fulfilled the requirements of the control system and advanced modularity, flexibility, and scalability of production.

Table 6. Comparison with traditional implementations based on the requirements of design.

Requeriment	This Work	Traditional Implementations	Comments
1	Complies	Partial	Traditional implementations usually require additional hardware
2	Complies	Does not comply	Traditional implementations use the client-server model.
3	Complies	Does no comply	Traditional implementations are monolithic
4	Complies	Does no comply	Specialized and robust controllers are still platform-dependent
5	Complies	Partial	Traditional implementations usually require additional hardware
6	Complies	Does no comply	Traditional implementations use well-defined programming languages
7	Complies	Partial	Traditional implementations have their own drivers
8	Complies	Does not comply	Traditional implementations are monolithic
9	Complies	Partial	Traditional implementations are private
10	Complies	Complies	
11	Complies	Complies	

6. Conclusions

The development and implementation of processing plants have particular relevance due to their intensification, since they help to avoid the high costs associated with automation and production processes. To achieve coordination between distributed industrial devices, plug-and-play software components based on the microservice architecture are adopted by the industry. In this work, we have presented a flexible, scalable, and robust framework based on software components, container technology, microservices, and the publish/subscribe paradigm.

This paper contributes to the development and implementation of industrial automation applications, closing the gap between generic architectures and physical realizations through the use of container technologies, the concepts of microservices, and the decoupling of each microservice with a middleware based on the publish/subscribe pattern. To demonstrate the applicability of the architecture proposed, a case study is developed and implemented at the juice production plant, Punta Delicia, located in Colima, Mexico. We have shown the implementation of a complex process, such as pH control process, ranging from the simulation part to its scaling and implementation on an industrial scale, showing the plug-and-play assembly from a definition of components with their relationships, to the implementation process with technologies involved. Finally, the experimental results correspond with the formal results obtained from the simulation model, which indicates that the platform implemented fulfilled the requirements of the control system. The validation of this process allows demonstrating that each development carried out can be treated independently until the processes are scaled up to their ideal point, reducing the development and final application costs.

Author Contributions: Conceptualization, A.G.-P., V.I.-J.; methodology, A.G.-P.; software, A.G.-P.; validation, H.S.-M., and A.G.-P.; formal analysis, H.S.-M., A.G.-P., V.I.-J., D.M.-C., P.B.; investigation, H.S.-M., A.G.-P., V.I.-J., D.M.-C., P.B., J.S.; resources, A.G.-P., P.B., J.S.; All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Acknowledgments: This work has been supported by for research cooperation between Universidad de Colima (Mexico), Universidad Autónoma de Occidente (Colombia), Universitat Politècnica de València (Spain) and the juice production plant Punta Delicia located in Colima, Mexico.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

q_1	flow rate of acid, mL/s
q_2	flow rate of buffer, mL/s
q_3	flow rate of base, mL/s
V	volume of the mixing tank, mL
y	value of pH
yd	value of pH reference
u	flow rate of base stream, mL/s
u_1	flow rate of base stream, mL/s
u_2	flow rate of acid stream, mL/s
pK_w	dissociation constant of water
K_C	controller gain
K_I	controller gain
v	linearization input
w_1i	concentration of the ith species in the acid stream, mol/L
w_2i	concentration of the ith species in the buffer stream, mol/L
α_i	concentration of the ith species in the base stream, mol/L
x_i	reaction invariant of ith species, mol/L

References

1. Rajkumar, R.R.; Lee, I.; Sha, L.; Stankovic, J. Cyber-physical systems: The next computing revolution. In Proceedings of the 47th Design Automation Conference (DAC), ACM, Anaheim, CA, USA, 13–18 June 2010; pp. 731–736. [CrossRef]
2. Khan, W.Z.; Ahmed, E.; Hakak, S.; Yaqoob, I.; Ahmed, A. Edge computing: A survey. *Future Gener. Comput. Syst.* **2019**, *97*, 219–235. [CrossRef]
3. Jaskó, S.; Skrop, A.; Holczinger, T.; Chován, T.; Abonyi, J. Development of manufacturing execution systems in accordance with Industry 4.0 requirements: A review of standard- and ontology-based methodologies and tools. *Comput. Ind.* **2020**, *123*. [CrossRef]
4. Belman-Lopez, C.E.; Jiménez-García, J.A.; Hernández-González, S. Análisis exhaustivo de los principios de diseño en el contexto de Industria 4.0. *Rev. Iberoam. Autom. Inform. Ind.* **2020**, *17*, 432–447. [CrossRef]
5. Hizam-Hanafiah, M.; Soomro, M.A. The Situation of Technology Companies in Industry 4.0 and the Open Innovation. *J. Open Innov. Technol. Mark. Complex.* **2021**, *7*, 34. [CrossRef]
6. Pahl, C.; Jamshidi, P.; Zimmermann, O. Microservices and Containers. In *Software Engineering 2020*; Felderer, M., Hasselbring, W., Rabiser, R., Jung, R., Eds.; Gesellschaft fur Informatik: Bonn, Germany, 2020. [CrossRef]
7. Januário, F.; Cardoso, A.; Gil, P. A Distributed Multi-Agent Framework for Resilience Enhancement in Cyber-Physical Systems. *IEEE Access* **2019**, *7*, 31342–31357. [CrossRef]
8. El Hariri, M.; Youssef, T.; Saleh, M.; Faddel, S.; Habib, H.; Mohammed, O.A. A Framework for Analyzing and Testing Cyber–Physical Interactions for Smart Grid Applications. *Electronics* **2019**, *8*, 1455. [CrossRef]
9. Ungurean, I.; Gaitan, N.C. A software architecture for the industrial internet of things—A conceptual model. *Sensors* **2020**, *20*, 5603. [CrossRef] [PubMed]
10. Coito, T.; Martins, M.S.; Viegas, J.L.; Firme, B.; Figueiredo, J.; Vieira, S.M.; Sousa, J.M. A Middleware Platform for Intelligent Automation: An Industrial Prototype Implementation. *Comput. Ind.* **2020**, *123*, 103329. [CrossRef]
11. Beregi, R.; Pedone, G.; Mezgár, I. A novel fluid architecture for cyber-physical production systems. *Int. J. Comput. Integr. Manuf.* **2019**, *32*, 340–351. [CrossRef]
12. Chen, G.; Wang, P.; Feng, B.; Li, Y.; Liu, D. The framework design of smart factory in discrete manufacturing industry based on cyber-physical system. *Int. J. Comput. Integr. Manuf.* **2020**, *33*, 79–101. [CrossRef]
13. Merdan, M.; Hoebert, T.; List, E.; Lepuschitz, W. Knowledge-based cyber-physical systems for assembly automation. *Prod. Manuf. Res.* **2019**, *7*, 223–254. [CrossRef]
14. Sanin, C.; Haoxi, Z.; Shafiq, I.; Waris, M.M.; Silva de Oliveira, C.; Szczerbicki, E. Experience based knowledge representation for Internet of Things and Cyber Physical Systems with case studies. *Future Gener. Comput. Syst.* **2019**, *92*, 604–616. [CrossRef]
15. Peres, R.S.; Dionisio Rocha, A.; Leitao, P.; Barata, J. IDARTS – Towards intelligent data analysis and real-time supervision for industry 4.0. *Comput. Ind.* **2018**, *101*, 138–146. [CrossRef]
16. Lass, S.; Gronau, N. A factory operating system for extending existing factories to Industry 4.0. *Comput. Ind.* **2020**, *115*, 103128. [CrossRef]
17. Boyes, H.; Hallaq, B.; Cunningham, J.; Watson, T. The industrial internet of things (IIoT): An analysis framework. *Comput. Ind.* **2018**, *101*, 1–12. [CrossRef]
18. Goldschmidt, T.; Hauck-Stattelmann, S.; Malakuti, S.; Grüner, S. Container-based architecture for flexible industrial control applications. *J. Syst. Archit.* **2018**, *84*, 28–36. [CrossRef]
19. Hofer, F.; Sehr, M.; Sangiovanni-Vincentelli, A.; Russo, B. Industrial Control via Application Containers: Maintaining determinism in IAAS. *arxiv* **2020**, arXiv:2005.01890v1.
20. González-Nalda, P.; Etxeberria-Agiriano, I.; Calvo, I.; Otero, M.C. modular CPS architecture design based on ROS and Docker. *Int. J. Interact. Des. Manuf.* **2017**, *11*, 949–955. [CrossRef]
21. Wan, X.; Guan, X.; Wang, T.; Bai, G.; Choi, B.Y. Application deployment using Microservice and Docker containers: Framework and optimization. *J. Netw. Comput. Appl.* **2018**, *119*, 97–109. [CrossRef]
22. Abeni, L.; Balsini, A.; Cucinotta, T. Container-Based Real-Time Scheduling in the Linux Kernel. *ACM SIGBED Rev.* **2019**, *16*. [CrossRef]
23. Anjali, F.N.U.; Caraza-Harter, T.; Swift, M.M.*Blending Containers and Virtual Machines: A Study of Firecracker and GVisor*; Association for Computing Machinery: New York, NY, USA, 2020. [CrossRef]
24. Kozhirbayev, Z.; Sinnott, R.O. A performance comparison of container-based technologies for the Cloud. *Future Gener. Comput. Syst.* **2017**, *68*, 175–182. [CrossRef]
25. Aheleroff, S.; Xu, X.; Lu, Y.; Aristizabal, M.; Pablo Velásquez, J.; Joa, B.; Valencia, Y. IoT-enabled smart appliances under industry 4.0: A case study. *Adv. Eng. Inf.* **2020**, *43*, 101043. [CrossRef]
26. Chen, B.; Wan, J.; Shu, L.; Li, P.; Mukherjee, M.; Yin, B. Smart Factory of Industry 4.0: Key Technologies, Application Case, and Challenges. *IEEE Access* **2017**, *6*, 6505–6519. [CrossRef]
27. Dai, W.; Wang, P.; Sun, W.; Wu, X.; Zhang, H.; Vyatkin, V.; Yang, G. Semantic Integration of Plug-and-Play Software Components for Industrial Edges Based on Microservices. *IEEE Access* **2019**, *7*, 125882–125892. [CrossRef]
28. Alam, M.; Rufino, J.; Ferreira, J.; Ahmed, S.H.; Shah, N.; Chen, Y. Orchestration of Microservices for IoT Using Docker and Edge Computing. *IEEE Commun. Mag.* **2018**, *56*, 118–123. [CrossRef]

29. Pontarolli, R.P.; Bigheti, J.A.; Fernandes, M.M.; Domingues, F.O.; Risso, S.L.; Godoy, E.P. Microservice Orchestration for Process Control in Industry 4.0. In Proceedings of the 2020 IEEE International Workshop on Metrology for Industry 4.0 and IoT, MetroInd 4.0 and IoT 2020—Proceedings, Roma, Italy, 3–5 June 2020; pp. 245–249. [CrossRef]
30. Benayache, A.; Bilami, A.; Barkat, S.; Lorenz, P.; Taleb, H. MsM: A microservice middleware for smart WSN-based IoT application. *J. Netw. Comput. Appl.* **2019**, *144*, 138–154. [CrossRef]
31. Krämer, M.; Frese, S.; Kuijper, A. Implementing secure applications in smart city clouds using microservices. *Future Gener. Comput. Syst.* **2019**, *99*, 308–320. [CrossRef]
32. Ren, H.L.; Jiao, Y.P. Study on the Distributed Real-Time and Embedded System Middleware Based on the DDS. In *Advanced Materials Research*; Materials Science and Information Technology; Trans Tech Publications Ltd.: Bach, Switzerland, 2012; Volume 433, pp. 7522–7525.
33. Amoretti, M.; Pecori, R.; Protskaya, Y.; Veltri, L.; Zanichelli, F. A Scalable and Secure Publish/Subscribe-based Framework for Industrial IoT. *IEEE Trans. Ind. Inf.* **2020**, *17*, 3815–3825. [CrossRef]
34. Calabretta, M.; Pecori, R.; Vecchio, M.; Veltri, L. MQTT-Auth: A Token-based Solution to Endow MQTT with Authentication and Authorization Capabilities. *J. Commun. Softw. Syst.* **2018**, *14*, 320–331. [CrossRef]
35. Ibarra-Junquera, V.; Jørgensen, S.; Virgen-Ortíz, J.; Escalante-Minakata, P.; Osuna-Castro, J. Following an optimal batch bioreactor operations model. *Chem. Eng. Process. Process Intensif.* **2012**, *62*, 114–128. [CrossRef]
36. González-Potes, A.; Mata-López, W.A.; Ibarra-Junquera, V.; Ochoa-Brust, A.M.; Martínez-Castro, D.; Crespo, A. Distributed multi-agent architecture for real-time wireless control networks of multiple plants. *Eng. Appl. Artif. Intell.* **2016**, *56*, 142–156. [CrossRef]
37. Kwan, C.; Lewis, F.; Yeung, K. Adaptive control of induction motors without flux measurements. *Automatica* **1996**, *32*, 903–908. [CrossRef]
38. Kwan, C.; Lewis, F.L. Robust backstepping control of nonlinear systems using neural networks. *IEEE Trans. Syst. Man Cybern. Part A Syst. Hum.* **2000**, *30*, 753–766. [CrossRef]
39. Polycarpou, M.; Zhang, X.; Xu, R.; Yang, Y.; Kwan, C. A neural network based approach to adaptive fault tolerant flight control. In Proceedings of the 2004 IEEE International Symposium on Intelligent Control, Taipei, Taiwan, 2–4 September 2004; pp. 61–66. [CrossRef]
40. Nejati, A.; Shahrokhi, M.; Mehrabani, A. Comparison between backstepping and input–output linearization techniques for pH process control. *J. Process Control* **2012**, *22*, 263–271. [CrossRef]
41. Wright, R.A.; Kravaris, C. On-line identification and nonlinear control of an industrial pH process. *J. Process Control* **2001**, *11*, 361–374. [CrossRef]
42. Ali, S.; Qaisar, S.; Saeed, H.; Khan, M.; Naeem, M.; Anpalagan, A. Network challenges for cyber physical systems with tiny wireless devices: A case study on reliable pipeline condition monitoring. *Sensors* **2015**, *15*, 7172–7205. [CrossRef] [PubMed]
43. Nguyen, T.; Chidambara, V.A.; Andreasen, S.Z.; Golabi, M.; Huynh, V.N.; Linh, Q.T.; Bang, D.D.; Wolff, A. Point-of-care devices for pathogen detections: The three most important factors to realise towards commercialization. *TrAC Trends Anal. Chem.* **2020**, *131*, 116004. [CrossRef]
44. Juneja, P.K.; Sunori, S.K.; Sharma, A.; Sharma, A.; Pathak, H.; Joshi, V.; Bhasin, P. A Review on Control System Applications in Industrial Processes. *IOP Conf. Ser. Mater. Sci. Eng.* **2021**, *1022*, 012010. [CrossRef]
45. Abdullah, N.H.S.; Karsiti, M.N.; Ibrahim, R. A review of pH neutralization process control. In Proceedings of the 2012 4th International Conference on Intelligent and Advanced Systems (ICIAS2012), Kuala Lumpur, Malaysia, 12–14 June 2012; Volume 2, pp. 594–598. [CrossRef]