IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 13 October 2024, accepted 9 November 2024, date of publication 26 November 2024, date of current version 9 December 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3506510

RESEARCH ARTICLE

Enhancing Agriculture Through IoT and Blockchain: A VeChain-Enhanced Sustainable Development Approach for Small-Scale Agricultural Exploitations

RENZO LEONARDO PAREJA URTECHO[1], JESÚS RODRÍGUEZ-MOLINA[1], MARGARITA MARTÍNEZ-NÚÑEZ[2], AND JUAN GARBAJOSA[3], (Life Senior Member, IEEE)

[1]Department of Telematics and Electronics Engineering, Universidad Politécnica de Madrid, 28040 Madrid, Spain
[2]Department of Organization Engineering, Business Administration and Statistics, Universidad Politécnica de Madrid, 28040 Madrid, Spain
[3]Higher Technical School of Engineering of Information Systems, Universidad Politécnica de Madrid, 28040 Madrid, Spain

Corresponding author: Jesús Rodríguez-Molina (jesus.rodriguezm@upm.es)

This work was supported in part by the Agencia Estatal de Investigación/Ministerio de Ciencia e Innovación through the project "Sustainability-Aware Internet of Things Systems Driven by Social Communities" (SIoTCom) under Grant PID2020-118969RB-I00.

ABSTRACT Small and mid-sized exploitations in the agricultural sector face several challenges, including excessive use of environmental resources, limited product visibility, and insufficient automation in farming operations. These issues result in increased operational costs, reduced crop quality, and higher risks of harvest loss, especially when weather predictions are unreliable. Addressing these challenges is crucial to improving productivity and sustainability in farming. This research work introduces a cost-effective Software-as-a-Service (SaaS) solution that leverages Internet of Things (IoT) technology and blockchain -specifically, the VeChain network- to provide real-time data on crop conditions and weather forecasts. The proposed solution enables farmers to make informed decisions by offering actionable insights and monitoring crops growth and general progress over time, thus promoting more efficient and sustainable farming practices. Furthermore, the integration of blockchain ensures transparency and immutability of data, building trust between farmers, distributors, and end customers. This increased visibility and accountability can also enhance product traceability across supply chains. By tackling these issues faced by small and mid-sized agricultural exploitations, this project aims to modernize farming practices, optimize resource utilization, and contribute to long-term sustainability, helping to secure food production in the face of global environmental and economic turmoil.

INDEX TERMS Distributed information systems, Internet of Things, sensors, cloud computing.

I. INTRODUCTION

The agricultural sector is undergoing a transformative shift driven by the integration of advanced technologies, particularly the Internet of Things (IoT) and blockchain in many different application domains, from insurance to commodities trade [1], [2]. As the global population rises, so does the demand for food production, creating the need for innovative solutions to enhance efficiency, sustainability, and productivity in farming practices [3], [4]. However, small and medium-sized agricultural enterprises often confront significant challenges that hinder their ability to adapt to or exploit these technological advancements. The research works described here explore these challenges and propose a comprehensive Software-as-a-Service (SaaS) solution that leverages IoT and blockchain technologies to address them.

One of the main issues faced by the agricultural industry is the suboptimal use of environmental resources. For example, traditional farming practices often lead to inefficient water management, resulting in increased operational costs and environmental strain. This is particularly problematic for small-scale farmers who may lack, for example, the resources

The associate editor coordinating the review of this manuscript and approving it for publication was Nurul I. Sarkar.

to implement irrigation systems [5]. The integration of IoT technologies can facilitate precise monitoring of, among other parameters, soil moisture levels and weather conditions, enabling farmers to optimize water usage and reduce waste. By providing real-time data, farmers can make informed decisions that enhance crop quality while minimizing their environmental footprint.

In addition to resource management, small agricultural enterprises frequently struggle with a significant lack of product visibility and automation in their operations [6]. Many farmers rely on manual processes for tracking crop conditions and managing resources, which can be time-consuming, inaccurate and prone to errors. The frequent absence of real-time monitoring systems further exacerbates these issues, making it difficult for farmers to respond promptly to changing conditions. The proposed SaaS solution aims to address this gap by offering an innovative distributed system that consolidates data from various sensors deployed across the farm. This system provides insights into crop health and growth, thereby reducing manual labor, monitoring the quality of the environment where crops grow and increasing the overall operational efficiency.

Another critical challenge faced by small farmers is the substantial risk of crop loss due to inaccurate meteorological predictions. Without reliable weather forecasting, farmers are often left vulnerable to unexpected weather events that can severely impact their harvests. The integration of IoT devices allows for the collection of real-time meteorological data, enabling farmers to receive timely alerts and make initiative-taking decisions to protect their crops [7]. This capability is essential for enhancing resilience in the face of climate variability and ensuring food security.

Moreover, our manuscript highlights the importance of data transparency and trust in the agricultural supply chain. Consumers are increasingly concerned about the quality and safety of the food they purchase, leading to a demand for verifiable information about production processes. By utilizing a commercial blockchain development like VeChain [8], the proposed solution ensures that all data related to crop conditions and farming practices are securely stored, traceable and immutable. This transparency not only builds trust between farmers and consumers but also empowers end customers to verify the quality of agricultural products through QR codes linked to the blockchain, which makes the implementation and data retrieval from that system more user-friendly.

A. PAPER CONTRIBUTIONS

In this manuscript there are several contributions done to the State of the Art with regards to the topics previously mentioned. The main ones are:

1) INTEGRATION OF IoT AND A COMMERCIAL, READY-TO-USE BLOCKCHAIN IN A REALISTIC AGRICULTURE USE CASE

this paper presents a novel approach by combining IoT technologies with a commercial blockchain -specifically, the VeChain network- to enhance data transparency and security in agricultural practices. This integration addresses the critical need for reliable data management in farming.

2) REAL-TIME MONITORING AND AUTOMATION

the proposed SaaS solution enables real-time monitoring of crop conditions and weather forecasts, allowing farmers to make informed decisions quickly. This capability is particularly beneficial for small and mid-sized farms that often lack access to advanced technologies. In addition to that, by focusing on a low-cost SaaS model, the project aims to democratize access to advanced agricultural technologies, making it feasible for small and medium-sized enterprises to adopt digital tools and improve their operational efficiency. Additionally, by utilizing blockchain for data storage, the project enhances trust between farmers and consumers, allowing end customers to verify product quality and growing conditions through immutable data.

When all is said and done, the present manuscript improves the working conditions of agricultural exploitations by providing a comprehensive solution that addresses the technological, economic, and environmental challenges faced by this economical sector.

B. PAPER STRUCTURE

This paper is structured as follows: an introduction on the contributions done by this manuscript and its background has already been given. Section II describes the related works that have been covered in this regard, as well as their advantages and weaknesses in the context where this paper is taking place. Section III shows the design and implementation works that have been carried out to develop a system that will cover to a significant extent the open issues that have been found in section II. Section IV displays the testing activities that have been performed to learn about the performance and the feasibility of the system. Section V includes the conclusions that have been reached and what future works can deal with.

II. RELATED WORKS

As far as blockchain and the IoT are concerned, there is a significant number of developments that have been made to maximize the outputs and productivity in agriculture. It must be mentioned, though, that a solution with such a holistic nature as the one described here, capable of integrating sensor readings, applications via QR or GUI and keeping records of the transactions in a commercial blockchain of widespread usage has not been found in the researched literature.

A. STUDY OF THE STATE OF THE ART

Zongda She [9] presents the key features of the VeChain blockchain technology. VeChain includes supply chain tracking with unique product codes for authenticity verification, so it becomes extremely useful for agricultural or industrial activities. Its ToolChain platform offers tailored services for various users, including food safety traceability and carbon footprint tracking. The immutable nature of blockchain

ensures data integrity, transparency and ease to audit any kind of transaction, enhancing trust among consumers and producers. Additionally, partnerships with major organizations like Walmart are expanding VeChain integration and effectiveness in supply chain solutions. While this piece of research does not provide detailed information about any specific use case involving VeChain, it can be used to prove that it is a useful blockchain implementation with widespread popularity, utilized in a plethora of use cases such as the one that we put forward in our manuscript.

Aliyu et al. present in [10] a framework for enhancing security in smart farming through the integration of blockchain technology and the IoT. It highlights the increasing reliance on IoT devices in agriculture and the associated security vulnerabilities. The authors propose a layered architecture that includes IoT devices, cloud processing, and blockchain for secure data management. They detail how events generated by IoT devices are monitored and processed in the cloud, with security alerts being recorded on the blockchain. In this context, the usage of smart contracts is emphasized for automating security responses. The paper also discusses the potential of sidechains to improve scalability and performance. Furthermore, the authors suggest future research directions, including the application of neural networks for attack prediction. Overall, the framework aims to provide a robust solution for securing smart agricultural practices, but it does not provide a use case where a commercial implementation of blockchain would be used.

Vieira et al. [11] provide a comprehensive survey of the integration of blockchain technology within the IoT. It begins by discussing the rapid growth of smart devices and the associated security risks, highlighting the vulnerabilities exposed by centralized security protocols. The paper emphasizes the potential of Distributed Ledger Technologies (DLTs), particularly blockchain, to enhance security through decentralization, immutability, and transparency. It outlines various IoT applications benefiting from blockchain, such as supply chain management and smart grid logistics. Additionally, the document critically examines the limitations of these technologies, including scalability and interoperability challenges. Key blockchain concepts are introduced, explaining their relevance to IoT security. The paper concludes with a summary of findings and stresses the importance of well-established blockchain projects in the IoT landscape. It serves as a valuable resource for understanding the synergies between blockchain and IoT, yet it does not provide information about any blockchain development that makes use of VeChain or other implementations.

Narasimmasubramanian et al. [12] show how the integration of Blockchain technology with the IoT can be used to enhance intelligent agriculture. It remarks the need for modernizing traditional agricultural practices to meet the growing global food demand due to population growth. Various sensors are employed to collect real-time data on environmental factors such as soil moisture, temperature, and pH levels. These data are transmitted to a decentralized blockchain system, ensuring transparency and security in agricultural operations. The proposed architecture facilitates efficient data management and monitoring, improving brand reputation and operational efficiency for farmers. The paper also outlines the roles of different sensor nodes in the IoT-Blockchain framework. Additionally, it emphasizes the potential of Artificial Intelligence (AI) to analyze data for better crop management decisions. Overall, the study advocates for the adoption of these technologies to optimize agricultural production while minimizing environmental impact. The research serves as a survey of current IoT and blockchain applications in the agriculture sector. Unfortunately, the paper focuses on the proven benefits of using blockchain in agriculture that, while widely known and accepted, do not provide information about a specific development work done that covers how a particular use case can be tackled with a blockchain iteration.

Xiao Guixia et al. discuss in [13] several blockchain-based use cases in the Food Supply Chain (FSC), showcasing their applications and impacts. One prominent example is VeChain, which is cited as being used in supply chain management across various industries, utilizing smart contracts and RFID labels to ensure product traceability and transparency. This system enhances quality control and builds trust among stakeholders. Another significant use case is CargoCoin [14], a platform that decentralizes global trade and transport, facilitating secure storage and transfer of goods through smart contracts, which streamlines documentation processes and reduces errors. Additionally, the IBM Food Trust [15] connects stakeholders in the FSC, improving transparency and food safety by providing real-time access to product data. Lastly, the Sustainable Shrimp Partnership (SSP, [16]) collaborates with shrimp farms in Ecuador to monitor product quality using IoT and blockchain technology, promoting sustainable practices and ensuring consumers can trace the origins of their seafood. These examples illustrate the transformative potential of blockchain technology in enhancing the efficiency and trustworthiness of food supply chains in diverse environments. However, the research done is more focused on providing a plethora of solutions that are already aiding in FSC (such is the case of VeChain, which has been used as the backbone of the blockchain integration of the implementation works done in our manuscript) than in providing more original work.

The Master Thesis of Iñigo Izaguirre Diaz [17] explores the development and implementation of a decentralized application (dApp) aimed at improving traceability and transparency in business processes using the VechainThor blockchain. The thesis begins with an overview of DLTs and compares various systems, including blockchain and IOTA [18]. It details the unique features of the VechainThor blockchain, such as its dual token methodology and architecture. The document outlines the requirements gathering, system architecture, and integration of the dApp with the VechainThor blockchain. Their manuscript covers the implementation phase, including

smart contract development, front-end design and testing, with a focus on security and performance. The discussion section evaluates the advantages and challenges of the implemented solution, particularly regarding security. The conclusion summarizes the key findings, highlighting the effectiveness of the dApp in enhancing delivery tracking and authentication in supply chain management. Improvements in security measures and scalability, as well as comparisons with other DLTs, are suggested as future works. In the end, the usage of VeChain in the presented environment and the development of a decentralized application is aligned with the contributions presented in our own manuscript, but the study itself is oriented towards business processes rather than a specific use case as present by us.

Jianting Xia et al. [19] describe in their own piece of research the cause study of the Lenovo electronics company, an example on how blockchain can be used for the improvement of the supply chain deployment among technology producers. The manuscript reviews existing literature, highlighting the advantages of blockchain in traceability and transparency, while noting a gap in studies addressing collaboration efficiency. Through semi-structured interviews with Lenovo's supply chain managers, the authors propose a conceptual model for blockchain-based information collaboration systems. This model integrates smart contracts, transaction authentication, and real-time data updates to enhance automation, transparency, and efficiency in supply chain management. The research emphasizes the importance of information continuity and authenticity, enabling efficient data sharing and faster supply chain responses. Additionally, it presents a three-layer architectural model consisting of a data layer, contract layer, and application layer. The findings suggest that blockchain can significantly reduce costs and improve operational efficiency. However, the paper just focuses on the fact that agriculture and traceability are the main two topics where blockchain is most popular, rather than providing their own specific usage of smart contracts or a blockchain implementation.

Xu et al. [20] offer a comprehensive survey on the convergence of Distributed Ledger Technology (DLT) and the IoT, authored by researchers from various institutions. It explores the transformative impact of these technologies across multiple sectors, including smart homes, autonomous vehicles, and smart cities. The authors develop a novel framework using a knowledge graph approach to systematically review DLT applications in the IoT. The survey highlights the limitations of traditional blockchain in IoT contexts and discusses alternative DLT prototypes with greater scalability. It also introduces a new six-layer architecture that emphasizes the integration of DLT capabilities with IoT applications. The document reviews industry-specific use cases and identifies key DLT characteristics relevant to different sectors. Additionally, it addresses challenges in DLT-IoT convergence and suggests future research directions. In summary, the survey aims to provide insights for researchers and practitioners in the field. However, they do not focus on a specific development involving blockchain in general, nor it explains a specific use case for VeChain in particular.

B. OPEN ISSUES

As can be seen in the previously reviewed literature, there is a collection of related works which have in common the search for a solution related to the usage of blockchain and/or IoT technologies in the application domain of agriculture. Unfortunately, these solutions have yet to match many of the open issues that have been put forward by the authors of this manuscript. Table 1 shows a summarized list of those proposals, along with its strengths and the weaknesses that result in several open issues that have yet to be solved.

Considering all the open issues that have been highlighted, the authors of this manuscript have developed a solution with the purpose of solving to a significant extent the most prominent problems found in this application domain.

III. PROPOSED SOLUTION

As explained previously, it becomes clear that the existing solutions present challenges and limitations that must be addressed. Such challenges are:

1. **Lack of usage of tools for large data storage** such as cloud computing:although cloud computing is mentioned as part of the infrastructure set in [10], it is overall not described in a clear manner how data are being transferred and stored in the solutions that gather it from the environment, either via sensors or any other tool used.
2. **Blockchain-based solutions** that are currently used and widespread in industrial applications:except in [9] and [17], Distributed Ledger Technologies mentioned in the studied literature tend not to cover in a thorough manner what actual blockchain implementations are being included as part of the performed development works.
3. **User Interfaces (UIs)** that can provide feedback in an amicable manner to end users:as far as the literature studied is concerned, very few details are provided on how end users will interact with the data collected.
4. **Measurable achievements** shaped as requirements and Key Performance Indicators (KPIs) in terms of system performance: specific objectives in terms of functionalities and performance would be welcomed so they can be tested and measured.

Our presented work aims to develop a Software-as-a-Service (SaaS) system based on IoT and blockchain to address the economic constraints faced by small and mid-sized exploitations. Our solution provides real-time metrics on crop conditions, weather forecasts and overall transparent, immutable data to end users. It is the authors´opinion that our proposed solution demonstrates a significant advantage in cost-effectiveness compared to existing systems, while addressing these previously identified challenges. For

TABLE 1. List of proposals with strengths and open issues.

Proposal	Strengths	Open issues
Zongda She [9]	Thorough details about VeChain and what it can offer from the blockchain point of view.	It is a description of VeChain fundamentals, it does not provide an actual development.
Aliyu et al. [10]	Framework that aims to provide a robust solution for securing smart agricultural practices.	It does not provide a specific use case where a particular or commercial implementation of blockchain will be used.
Vieira et al. [11]	Provides information for understanding the synergy between blockchain and IoT.	It does not provide information about any specific blockchain development.
Narasimmasubramanian et al. [12]	Adoption of these technologies to optimize agricultural production while minimizing environmental impact.	Only focuses on the proven benefits of using blockchain in agriculture rather than a specific use case.
Xiao Guixia et al. [13]	Describe the transformative potential of blockchain in food supply chains.	Focused on providing a plethora of solutions that are already aiding in Food Supply Chain rather than a specific use case.
Iñigo Izaguirre Diaz [17]	Use case description. Mentions specific blockchain development.	Oriented towards business processes rather than a specific use case.
Jianting Xia et al. [19]	Blockchain can significantly reduce costs and improve operational efficiency.	Does not make use of a specific, commercial blockchain. Does not comment on a particular solution.
Xu et al. [20]	Provides insights for researchers and practitioners in the field.	Does not make use of a specific, commercial blockchain. Does not comment on a particular solution.

example, targeting small and medium-sized farms, it adopts an affordable Software-as-a-Service (SaaS) model with monthly operational costs of €515.98 for a 1-hectare plantation, including fixed and variable expenses. Calculations result in these figures because a) fixed costs, at the time of writing this manuscript, include expenses for Azure Service Bus (€9/month), IoT Hub (€9.36/month), Virtual Machines (€82.01/month), Azure Kubernetes Service (€68/month), External Storage (€2.40/month), and Internet Connectivity (€342/month), totaling €512.77 and b) Service Bus usage contributes an additional €3.21 per month.

Additionally, by utilizing open-source tools and the VeChain public blockchain, the solution avoids high licensing fees, ensuring accessibility to smaller agricultural operations. In contrast, existing solutions like IBM Watson's platform start at $5,000 per month [21], excluding additional costs for sensors and cloud infrastructure, while proprietary systems such as John Deere's precision agriculture depend on expensive specialized machinery [22]. Smarteye's system, requiring sensor investments of around €12,000 [23], further emphasizes the cost-prohibitive nature of current alternatives.

From a performance perspective, the proposed solution ensures real-time monitoring and alert mechanisms, leveraging IoT technology for temperature, humidity, and pressure readings while integrating VeChain's Proof of Authority (PoA) consensus algorithm for secure, tamper-proof data storage. It also incorporates weather forecasting to aid decision-making. This holistic approach stands out from existing solutions, many of which focus solely on data collection and lack blockchain integration or real-time processing capabilities. High-performance alternatives like IBM Watson require substantial computational resources, making them less accessible for resource-constrained farming operations.

Scalability is another strength of the proposed solution, which supports up to 80 devices sending data concurrently at 30-second intervals. Its design allows horizontal scaling by adding microservice replicas on the Azure Kubernetes Service (AKS) cluster. This modular architecture ensures seamless expansion to meet growing demands. Conversely, many existing systems rely on centralized architectures that are resource-intensive and costly to scale. Proprietary systems often necessitate extensive infrastructure overhauls, further complicating scalability and increasing operational expenses.

The proposed solution also excels in sustainability by employing VeChain energy-efficient PoA blockchain, which consumes significantly less electricity than traditional blockchain technologies. In contrast, existing systems typically overlook sustainability, with some blockchain implementations consuming excessive energy and proprietary solutions contributing to higher carbon footprints due to centralized data centers.

Finally, in terms of technical integration, the proposed solution seamlessly combines IoT, blockchain, and cloud technologies into a unified workflow. IoT sensors capture data, which is immutably stored on the blockchain, and APIs and dashboards provide intuitive user interaction and data visualization. The use of open-source tools like NodeJS and Grafana, along with Infrastructure as Code (IaC) practices, enhances flexibility and automation. Existing solutions such as IBM and John Deere offer robust integration but are heavily dependent on proprietary systems, limiting adaptability and transparency. Smarteye's lack of user-facing blockchain features further highlights the technical superiority of the proposed approach.

To define what our solution would look like when having to perform design works, functional and non-functional

requirements have been set. Their main features have been described in the following subsections.

A. REQUIREMENT ANALYSIS

There are several requirements that have been defined to provide a framework for the developments carried out to create the aforementioned system. The functional requirements that were established are:

1. Feedback from weather (Functional Requirement 1, FR1). It is provided by ensuring that the following tasks are completed a) real-time weather monitoring and forecasting (the system must gather weather data from reliable sources and update users on current and predicted weather conditions when they demand it), b) weather data integration (the solution will integrate weather APIs or other data providers to deliver localized, accurate weather forecasts) and c) alerts and notifications (the system should trigger alerts based on weather changes, helping farmers make proactive decisions).

2. Distributed data and transaction history storage (FR2). This functional requirement must be satisfied with the following tools a) immutable data storage (the blockchain will securely store all historical data, ensuring that this information cannot be tampered with), b) smart contracts (they will govern how and when data gets recorded, enabling automated, trustworthy record keeping) and c) traceability (stakeholders will have access to verifiable and traceable records for transparent farming operations).

3. Existence of a connecting software layer (FR3). There will be several operations that will take place within the system that will be required for its normal performance. Consequently, a connecting layer that provides an underlying platform must be provided as well. This platform will effectively work as a middleware layer that will abstract the heterogeneity and complexity of the underlying hardware parts and will offer the illusion of a centralized, homogeneous-looking system [24]. The middleware layer will be implemented by using two developments: a) a microservices architecture (enables different system components to communicate efficiently) and b) API integration (to handle communication between sensors, blockchain, weather data systems and exposing necessary data to end users in a secure and reliable manner).

4. Usage of sensors to collect information (FR4). This will be one cornerstone of the developed system, as sensors will be required to gather data from the environment. Besides having sensor integration as a major feature, the middleware layer to be integrated will monitor sensor status and trigger alerts if sensors fail or produce data outside of the defined thresholds, helping farmers maintain optimal crop conditions.

5. Graphical User Interface (GUI)-like outputs (FR5). Data visualization will be of major importance as well, since farm workers will require some form of information feedback on the conditions and events that are taking place within the system. Therefore, there will be an interactive, user-friendly interface to display real-time and historical data on crop conditions, weather forecasts, and alerts. Users will be able to set custom thresholds for key metrics (e.g., soil moisture, temperature), with visual indicators showing when values exceed these thresholds. Farmers and other users will interact with the system through the web-based interface, which will provide intuitive access to crop data and weather forecasts and alerts, enabling data-driven decision-making.On the other hand, non-functional requirements can be listed as follows:

1. System Scalability: Support for at Least 50 Concurrent Users (Non-functional Requirement 1, NFR1): The system must be designed to handle at least 50 concurrent users simultaneously that interact with the platform without performance degradation, as it is estimated that most small and medium agricultural exploitations will have no more than 50 people participating in it (owners, hired labour etc.). This requires optimizing server capacity, database performance and network bandwidth to support concurrent user access. Load balancing should also be considered to ensure smooth operations during peak usage so that user requests are distributed evenly across available resources.

2. Mobile-Friendly Graphical User Interface (GUI, NFR2): a GUI is a must-have to correctly visualize the information gathered by the system. Its main features will be as follows: a) responsive design (the GUI must be designed to adapt to different screen sizes, especially smartphones, without compromising functionality or user experience. This ensures users can easily interact with the platform from their mobile devices in field conditions) and b) mobile optimization (the interface should be lightweight and optimized for lower bandwidth and mobile processing power. This includes optimizing images, minimizing scripts, and ensuring quick load times for mobile users).

3. QR Code Generation for Data Access (NFR3): The main features are as follows: a) secure QR code generation (the system should generate QR codes that link to specific blockchain-based records -crop conditions, sensor data-. These codes should be securely created and embedded with appropriate metadata for easy verification), b) cross-platform readability (QR codes generated by the system must be universally readable across different devices, ensuring that any smartphone with a QR code reader can scan and access the data) and c) tamper-proof data access (since the QR codes will be linked to blockchain records, the data they represent must be immutable, ensuring that no unauthorized changes can be made after the QR code is generated).

4. Cloud Computing for Middleware and Microservices (NFR4): In this case, the following characteristics should be present: a) cloud-native architecture (the system must be designed and deployed using cloud computing, ensuring high availability, automatic scaling, and fault tolerance for the middleware layer and the microservices), b) efficient resource allocation (by leveraging cloud computing, the system should dynamically allocate resources -compute, storage, and networking- as needed, ensuring that it can scale efficiently without over-provisioning or incurring in unnecessary

costs) and c) continuous delivery and deployment (the use of cloud-based infrastructure will enable Continuous Integration and Deployment (CI/CD) pipelines, allowing for rapid updates, bug fixes, and feature enhancements without causing downtime).

5. Use of a Commercial Blockchain (NFR5): the main features to be followed in this case are: a) blockchain integration for data integrity (the system will rely on a commercial blockchain solution to ensure that all crop condition data, transaction history, and sensor data are stored immutably and transparently in a widely used and supported DLT easy to work with. This will enhance trust among users and external stakeholders, such as regulators and buyers), b) cost efficiency and security (the selected blockchain platform should be cost-effective while providing the necessary security guarantees, such as encrypted data transmission, secure key management and protection from unauthorized access or tampering) and c) smart contract support (the blockchain must support smart contract functionality, allowing for automated data validation, verification, and persistence without manual intervention. These smart contracts will handle interactions between users and the system in a secure and decentralized manner).With this number of requirements in mind, a system was designed that would be able to satisfy all of them. To accomplish this, the system that was designed resulted in five subsystems that effectively fulfilled all the functional and non-functional requirements that were formulated before. These subsystems are:

1. Input Subsystem: The input subsystem is responsible for collecting data from various sources, primarily from IoT devices (sensors) and users interacting with the platform through a web browser or a mobile-based interface.

1.1. Components: for this subsystem, the following components have been listed: a) sensors (sensors integrated with the ESP32 microcontroller gather real-time data on soil moisture, temperature, humidity, and other environmental conditions in a farm), b) user inputs (farmers and other users provide input via a web or mobile interface. This could include manually entering crop details, farm operations data, or responding to system alerts) and c) data ingestion (the system ingests data from both the IoT devices and user input, ensuring that the incoming data is structured, validated, and ready for processing in the middleware subsystem).

1.2. Functions: expected functions are as follows: a) data collection and validation from sensors and users, b) triggering alerts or notifications based on abnormal conditions from sensor data and c) sending the collected data to the middleware subsystem for further processing.

2. Output Subsystem: The output subsystem delivers processed information back to users in a usable format. This includes visualizing real-time and historical data and generating verifiable records.

2.1. Components: in this case, the components to be used are a) Graphical User Interface (GUI, a web and mobile interface for farmers to view data visualizations, interact with system alerts, and make informed decisions), b) QR Code Generation (the system generates QR codes linked to blockchain-verified data records, allowing external parties - such as buyers or regulators- to scan and view information on crop conditions and farm operations and c) data visualization via Grafana (Grafana is used to visualize real-time sensor data, weather forecasts, and threshold-based alerts, providing farmers with actionable insights).

2.2. Functions: functions to be carried out are: a) displaying real-time crop condition metrics (e.g., moisture levels, temperature) and weather forecasts in an intuitive format, b) generating QR codes for verifying the authenticity and condition of farm products and c) providing alerts and notifications to users, such as warnings about weather changes or abnormal crop conditions.

3. Middleware Subsystem: The middleware subsystem acts as the core of the system, handling data processing, communication between components, and ensuring smooth integration of the input and output subsystems with the blockchain and weather services. As before, components in this subsystem and its functions are presented.

3.1. Components: the components that will be used in this subsystem are: a) microservices architecture (various microservices manage the different functions of the system, such as sensor data processing, API calls, and alert management), b) APIs (interfaces that allow the input and output subsystems to communicate with each other, as well as with external services -weather data providers, blockchain- and c) data processing engines (responsible for transforming raw sensor data and user inputs into actionable information that can be visualized or stored on the blockchain).

3.2. Functions: functions in this subsystem would be a) processing and filtering raw data from sensors, ensuring it is ready for storage or visualization, b) routing data between the input, output, blockchain, and weather subsystems and c) managing real-time data exchanges and triggering alerts when predefined thresholds are met.

4.Blockchain Subsystem: The blockchain subsystem ensures data transparency, security, and immutability by recording important data (e.g., crop conditions, transactions) on the VeChain blockchain.

4.1. Components: the components present in this subsystem are: a) VeChain Blockchain Network (the platform used to securely store crop condition data, transaction history, and sensor records in an immutable manner), b) Smart Contracts (programs that automate the process of storing and verifying data on the blockchain. These contracts govern how and when data are written to the blockchain, ensuring data integrity and transparency) and c) QR code linkage (access to data stored on the blockchain is linked to QR codes, providing verifiable access to crop conditions and farm activity records for external stakeholders).

4.2. Functions: the functionalities to be used in this subsystem are: a) ensuring that all critical farm data (such as crop conditions and transactions) are stored immutably and transparently, b) executing smart contracts that automate the

recording of sensor data and transactions on the blockchain and c) providing a secure, decentralized system that external parties can trust, especially when verifying data through QR codes.

5.Weather Subsystem: The weather subsystem gathers, processes, and displays weather data, helping farmers make data-driven decisions based on real-time and forecasted conditions.

5.1. Components: components to use would be a) Weather Data APIs (external APIs supply real-time weather data and forecasts based on the geographic location of the farm), b) alert system (monitors weather data and triggers alerts when significant weather events -storms, high winds- are predicted) and c) integration with GUI and data visualization: Weather data are integrated into the overall data visualization platform, providing farmers with critical insights that can be cross-referenced with crop information.

5.2. Functions: it is expected that the functions to be provided in this subsystem are: a) real-time and forecasted weather data -ensuring that users have up-to-date information to optimize farming activities-, b) triggering alerts to warn farmers of adverse weather conditions, allowing them to take preventive actions and c) displaying weather data alongside sensor data on the Grafana dashboard, helping farmers make holistic decisions based on both environmental conditions and forecasted trends.

These subsystems work together to create a resilient and user-friendly SaaS solution that helps small and mid-sized farms operate more efficiently by using real-time data, blockchain technology and weather insights. The way they fulfill the functional and non-functional requirements that were formulated previously has been displayed in Table 2.

TABLE 2. Relation between requirements and the system.

SUBSYSTEM	FULFILLED FUNCTIONAL REQUIREMENTS	FULFILLED NON-FUNCTIONAL REQUIREMENTS
Input Subsytem	Usage of sensors to collect information (FR4)	System Scalability: Support for at Least 50 Concurrent Users (NFR1)
Output Subsystem	Graphical User Interface (GUI)-like outputs (FR5)	Mobile-Friendly Graphical User Interface (GUI, NFR2), QR Code Generation for Data Access (NFR3)
Middleware Subsystem	Existence of a connecting software layer (FR3)	System Scalability: Support for at Least 50 Concurrent Users (NFR1), Cloud Computing for Middleware and Microservices (NFR4)
Blockchain Subsystem	Distributed data and transaction history storage (FR2)	System Scalability: Support for at Least 50 Concurrent Users (NFR1) Use of a Commercial Blockchain (NFR5)
Weather Subsystem	Feedback from weather (FR1)	Mobile-Friendly Graphical User Interface (GUI, NFR2)

B. SYSTEM DESCRIPTION AND SECURITY DESIGN

As described, the system built consists of several parts:

1) MICROSOFT AZURE CLOUD INFRASTRUCTURE [25]

It makes use of Bicep scripts [26] to establish the Microsoft Azure cloud infrastructure, thus leveraging Infrastructure-as-Code (IaC) for efficient and reproducible deployment.

2) VeCHAIN BLOCKCHAIN

It implements the VeChain blockchain network to ensure data transparency. This involves the development and use of smart contracts to interchange and store data securely and immutably.

3) SYSTEM LOGIC DEVELOPMENT

Using NodeJS [27] for both backend and frontend development. In this subsystem, the software architecture is based on microservices and APIs -which handle data storage on VeChain- sensor data processing and alert notifications, thus exposing crop conditions through a web application and generating QR codes for fast access to the information.

4) MICROCONTROLLER INTEGRATION

This subsystem uses a C code implementation to integrate the ESP32-WROOM-32 microcontroller [28] with Azure IoT Hub, generating random data for system testing.

5) DATA VISUALIZATION WITH Grafana

in this subsystem, an installation and configuration of Grafana, an open-source tool, to visualize data stored on VeChain, is used [29]. This includes setting threshold limits and displaying weather forecasts to aid farmers in making informed decisions. The holistic architecture of the developed works is represented in Figure 1. This figure provides a comprehensive understanding of the system design and its operational principles, which is vital for both design requirements fulfillment and practical implementation. From the implementation point of view, the architecture leverages a Microsoft Azure IoT hub to expose a secure endpoint using Message Queuing Telemetry Transport protocol (MQTT, [30]) secured with Transport Layer Security (TLS), ensuring safe data reception from the farm sensors.

Upon receiving the data with the correct credential parameters, they are routed to an Azure Service Bus topic for further

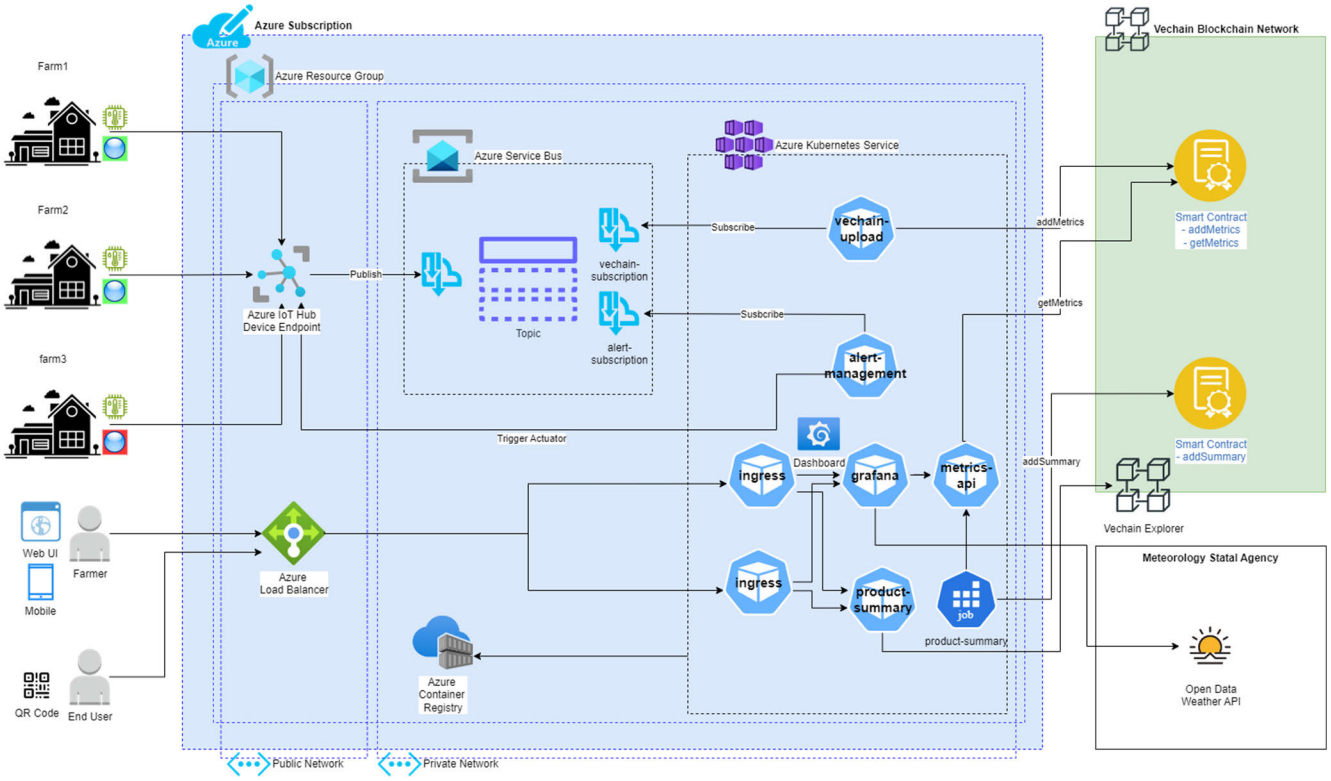


FIGURE 1. Holistic architectural implementation of the subsystems.

handling. The system makes use of Azure Kubernetes Service (AKS) to orchestrate the APIs and microservices, chosen for its health check, self-recovery capabilities and internal load balancing, all of which enhance system high availability and resiliency. Azure Container Registry (ACR, [31]) securely stores Docker images for all software components, facilitating efficient deployment and scaling within AKS. Data transparency and immutability are achieved through smart contracts published on the VeChain public test network, which is used to handle sensors data storage. The software components within AKS utilize the methods of these smart contracts. The system integrates weather forecast from the Spanish meteorology agency AEMET utilizing their open data services, which allow the dissemination of free weather information through APIs that the system uses to provide weather forecast to the farmers. The Azure Load Balancer is used to securely expose the dashboard and products information to end users and farmers. Finally, sensor data simulation is handled by an ESP32-WROOM-32 microcontroller, which generates temperature, humidity and pressure data. This device authenticates and sends the data to the Azure IoT Hub. Overall, the algorithm conceived to be used with the overall architecture will work with the following steps:

1. **Initialize Connection**: Establish a connection to the Azure IoT Hub.
2. **Receive Sensor Data**: Continuously listen for incoming data from farm sensors.
3. **Validate Data**: Check if the received data meets predefined criteria (e.g., valid temperature, humidity, etc.).
4. **Store Data on Blockchain**: If the data is valid, invoke the smart contract to store the data on the VeChain blockchain.
5. **Notify Users**: Send alerts or notifications if the data indicates abnormal conditions.
6. **Log Data**: Maintain a local log of all processed data for auditing and monitoring purposes.
7. **Visualize data**: Show the collected information to the end users.

Should this algorithm for data collection, storage and retrieval be considered as pseudocode that could be ported to different programming languages, it would look as follows:

```
function main() {
  // Step 1: Initialize connection to Azure IoT Hub
  initializeConnectionToIoTHub()

  // Step 2: Listen for incoming sensor data
  while (true) {
    sensorData = receiveSensorData()

  // Step 3: Validate the received data
  if (isValidData(sensorData)) {
    // Step 4: Store valid data on the blockchain
    storeDataOnBlockchain(sensorData)
    // Step 5: Notify users if necessary
```

```
        if (isAbnormalCondition(sensorData)) {
            sendAlertToUsers(sensorData)
        }
        // Step 6: Log the processed data
        logData(sensorData)
    } else {
        // Handle invalid data
        logInvalidData(sensorData)
    }
  }
}
function initializeConnectionToIoTHub() {
    // Code to establish connection to Azure IoT Hub
}
function receiveSensorData() {
    // Code to receive data from sensors
    return sensorData
}
function isValidData(sensorData) {
    // Code to validate sensor data
    return (sensorData.temperature >= MIN_TEMP &&
    sensorData.temperature <= MAX_TEMP)
}
function storeDataOnBlockchain(sensorData) {
    // Code to interact with smart contract and store data on
    VeChain
}
function isAbnormalCondition(sensorData) {
    // Code to check for abnormal conditions
   return(sensorData.temperature>THRESHOLD_TEMP)
}
function sendAlertToUsers(sensorData) {
    // Code to send notifications to users
}
function logData(sensorData) {
    // Code to log the processed data
}
function logInvalidData(sensorData) {
    // Code to log invalid data for auditing
}
function visualizeData (sensorData) {
    // Code to show information to end users
}
```

C. CLOUD INFRASTRUCTURE

Microsoft Azure provides infrastructure and platform services to support all the workload of the system. The cloud services provisioning is done using IaC, which offers a repeatable, proven, automated process to provision the necessary cloud infrastructure. The security policies to access the service bus topic are created using the following script, which has one policy that provides writing permissions, and another one granting read-only permissions:

In the implementation works, two subscribers are created: one to upload data to the VeChain network, and the other one for processing the data and triggering alerts. The Azure IoT Hub is created using the free tier Stock-Keeping Units (SKU), along with a routing endpoint which uses output data from the Service Bus Bicep script execution. A route is created for the service to be publicly accessible with the condition '(level=storage)', processing only data with this property from microcontrollers. Finally, the AKS deploys a Linux node to host the system software components.

Bicep Script
Resource topicAuthSendPolicy 'Microsoft.ServiceBus/namespaces/topics/ authorizationRules@2022-10-01-preview' ={ name: **topicSendPolicyName** parent: serviceBusTopic properties: {rights: [**'Send'**] }} resource topicAuthReadPolicy 'Microsoft.ServiceBus/namespaces/topics/ authorizationRules@2022-10-01-preview' ={ name: **topicReadPolicyName** parent: serviceBusTopic properties: {rights: [**'Listen'**] }}

D. BLOCKCHAIN INFRASTRUCTURE

VeChain is a blockchain ecosystem founded in 2015 that supports asynchronous operations and employs the Proof of Authority (PoA, [32]) consensus algorithm to enhance security and decentralization. Its public blockchain network is supported by the community, making it a robust and decentralized platform. The PoA 2.0 consensus algorithm attempts to balance speed and security with lower energy consumption, making VeChain one of the most energy-efficient blockchains. Two smart contracts were deployed to manage the data persistency and transparency for the system; Solidity was used to develop the smart contracts for this project [33], which is one of the most popular Object-Oriented Programming (OOP) languages for writing smart contracts. The first one, used for metrics storage, saves incoming data from sensors into the blockchain and to allow the software components to retrieve the persistent data. The saved data includes the reading date as a timestamp, temperature, humidity and pressure. The second smart contract contains the product summary. It is designed to store key data about the harvesting conditions on different crops. This contract records the harvest start and end dates, along with maximum, minimum and average values of temperature and humidity during the harvesting period. Furthermore, Sync was downloaded and installed [34]. This latter software tool can be defined as a desktop wallet application used to interact with the VeChain Thor blockchain network decentralized apps.

As for the deployment of smart contracts, Remix is a web browser Integrated Development Environment (IDE) that allows users to develop and compile Solidity-written smart contracts. To do this, the Solidity files were uploaded to the Remix platform, which provided two files. The first one was

the bytecode generated file. In this context, bytecode is a low-level machine-readable code to be stored in the blockchain. The second file was the contract Application Binary Interface (ABI), which is a JavaScript Object Notation (JSON) file that represents a standard way to interact with a specific contract. It contains important information about the contract itself, such as how functions are called and how data should be passed. To upload the smart contract to the VeChain network, the inspector decentralized App published on Sync wallet was used. The resulting output from uploading the bytecode became the contract address.

E. APIs AND MICROSERVICES

Different software components were developed to achieve the project goals, all using NodeJS as the programming language. Several dependency packages were installed and used in these components. The microservices that were executed afterwards used Kubernetes as the installation resource. Such microservices have been listed as follows:

1) VeChain UPLOAD

This microservice reads data from the Azure Service Bus Topic as a subscriber and uploads it to the VeChain blockchain using the Metric Storage smart contract. The modules installed for this service are a) VeChain dependencies, b) Azure Service Bus Dependencies and c) Web dependencies (only the ''stringify-object'' one is required). After installing and importing the dependencies, the connection variables for Azure Service Bus and VeChain are set. For VeChain the wallet address is needed, as well as a private key to sign the transactions, the contract address, a network and the ABI file related to the smart contracts. For Azure Service Bus connectivity, the connection string, subscription name and topic name must be provided. After setting up the connection variables, the connection with the external platforms is opened. As a topic subscriber, the data are taken from the Azure Service Bus topic and parsed to the correct format to be understood by the VeChain smart contract. Finally, the private key associated to the wallet address is used to sign the transaction and a gas limit is set to protect the system from costly transactions.

2) ALERT MANAGEMENT

This microservice reads data from the Azure Service Bus Topic as a subscriber, processes it to check for anomalies, and send alerts to the agricultural workers if necessary. The modules installed are a) Azure Service Bus Dependencies, b) Azure IoT Hub Dependencies and c) Web dependencies (only the ''express'' one is required). This service subscribes to the Azure Service Bus topic, evaluates the gathered temperature-related data to check whether temperature readings are within the acceptable range, and sends notifications if conditions are bad. The notifications are sent as a message -LED_ON or LED_OFF- with temperature thresholds being set at 20°C and 28°C as lower and higher limits, respectively.

3) METRICS API

This microservice reads and decodes data stored on the VeChain blockchain and exposes it though a Representational State Transfer (REST) API in JSON format for other services. The modules installed are VeChain Dependencies and Web dependencies (only the ''express'' one is needed). The method written to get the metrics from the VeChain blockchain receives the start and end timestamps as parameters and, after retrieving the data, they are parsed into a human readable format and filtered to match the requested dates. The resulting JSON is exposed via a REST API endpoint.

4) CROP CONDITION SYSTEM

This is a subsystem that is composed by two differentiated parts. The first one is referred to as Summary Job. This Job requests metrics from the Metrics API, creates a summary of the data (including start/end dates of the crop, minimum, maximum and average temperature and humidity) and stores it on the VeChain network using a Product Summary smart contract. The modules installed for this latter purpose are VeChain Dependencies and Web dependencies (only ''axios'' is needed). This Job makes a REST API call to get information about the crop conditions for a specific timeline, which returns a JSON message processed to calculate the maximum and minimum values comparing each value with the next one and replacing the variable in case a lower or greater value appears. All the records are counted, and the values are summarized to calculate the average temperature and humidity. The second differentiated part is the Web App, which publishes the crop conditions in web format. The module installed is Web Dependencies. The express module exposes a web service on ''/decode'' path receiving a transaction hash as a query parameter. Then the service retrieves encoded information from VeChain blockchain, decodes it using the ABI and ABI-decoder package, and presents the human-readable data to the end user on a web page.

FIGURE 2. Generated QR code.

F. BULK LOAD

This application reads data gathered from an olive tree field stored in a text file and sends it to the Azure IoT hub.

The modules installed are Azure IoT Hub dependencies and FileSystem dependencies. The application reads the text file content and parses the data to extract temperature, humidity and pressure variables. The resulting variables are sent to another method that a) creates a message formatted as a JSON that can be read by the Azure IoT Hub, b) sets the ''level=storage'' property and c) sends the data to Azure IoT hub for its further processing.

G. QR GENERATOR

This application generates a QR code that redirects end consumers to the summarized data of a product. The modules installed are qrCode dependencies and Canvas dependencies. The transaction URL is placed as input for the qrCode module to create the QR Code with a VeChain image placed in the center to generate trust in the end users, as represented in Figure 2. The QR code is then used to provide access to summarized information from products.

All the software components described are embedded into Docker images for deployment on the provisioned AKS cluster. The Kubernetes resources manifest files are used to deploy, configure and integrate applications into the AKS cluster. Ther are three of them: Deployments.mf, Services.mf and Ingress.mf.

Deployments.mf specifies the image name of the software component and the number of replicas for high availability, among other important parameters. Services.mf enable internal communication between different containers deployed in the same AKS cluster. Finally, Ingress.mf exposes a deployed service to the Internet, making it accessible.

In addition to these developments, the software components that do not make use of Docker images, as well as the hardware that has been used for the first round of testing abilities must be taken into account, along with the securitization framework that has been created to deploy the system in a way that can be trusted by end users.

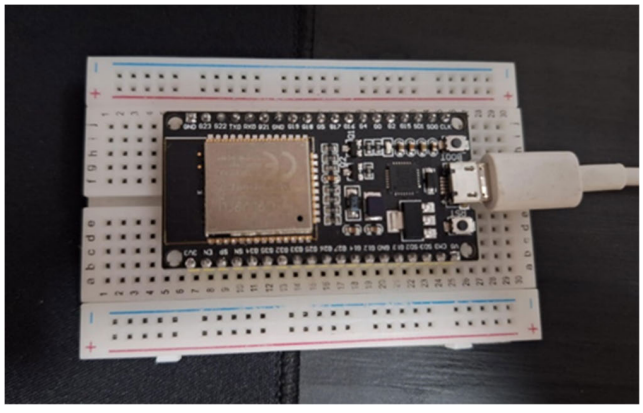


FIGURE 3. ESP32 connected device.

H. MICROCONTROLLER

For the purposes of these development works, an ESP32-WROOM-32 was used as a client microcontroller to generate and send temperature, humidity and pressure data to the system through its Wi-Fi module; it is depicted in Figure 3. To integrate the ESP32 platform with the Azure IoT hub the base code provided by Azure itself was used, as the Azure Software development Kit (SDK) provided for C libraries comes with code provided by Microsoft itself. An Arduino IDE board was configured to recognize the ESP32 platform using the ES32 Dev Module. The iot_config.h file was edited to add the system configuration parameters along with a Wi-Fi network SSID and a password. Several functions have been used for data generation. The most important one in this context is generateTelemetryPayload, which was edited to generate random temperature, humidity and pressure values, along with the previous method to get the current timestamp for further processing.

I. DATA VISUALIZATION

Grafana is an open-source visualization tool which can connect to different data sources and customize dashboard creation using Unified Query Language (UQL). Grafana is used in this system to visualize raw data from sensors in linear graphs, the sensors summary data in gauges and the weather forecast integrated in the system. Grafana has been deployed by means of AKS. To do so, a manifest file with all the needed resources was applied to the AKS cluster. The created resources are a) Persistent volume claim -used to persist dashboard, sources and local user's configurations data in an external storage, which is an Azure SSD Managed Disk previously configured- b) Deployment -used to place Grafana

deployment configurations, among other features-, c) Service -used to enable internal communication opening Grafana listening port (3000) and internal communication protocol- and d) Ingress -used to publicly expose a Grafana iteration using a private domain-. Once the administrator credentials were set on the first login into Grafana, metrics dashboards were created. It was done so by means of the Grafana web user interface. The latter was set to display temperature, humidity and pressure with a linear graph visualization. The average gauge visualization was added as well, as it will be shown in next section.

J. WEATHER FORECAST DASHBOARD

The weather forecast dashboard uses an open data API as data source to visualize the weather forecast. This API is secured with a user-key fixed token and expects a location code to deliver the weather forecast for that specific location. A section is used by Grafana to show the predicted weather values. To enable this a new visualization was added with the configuration.

K. SYSTEM SECURITY

The authors of this manuscript consider that any modern development, especially if it involves a distributed system, must make use of a securitized environment where components can be deployed and used at both the sensing and the

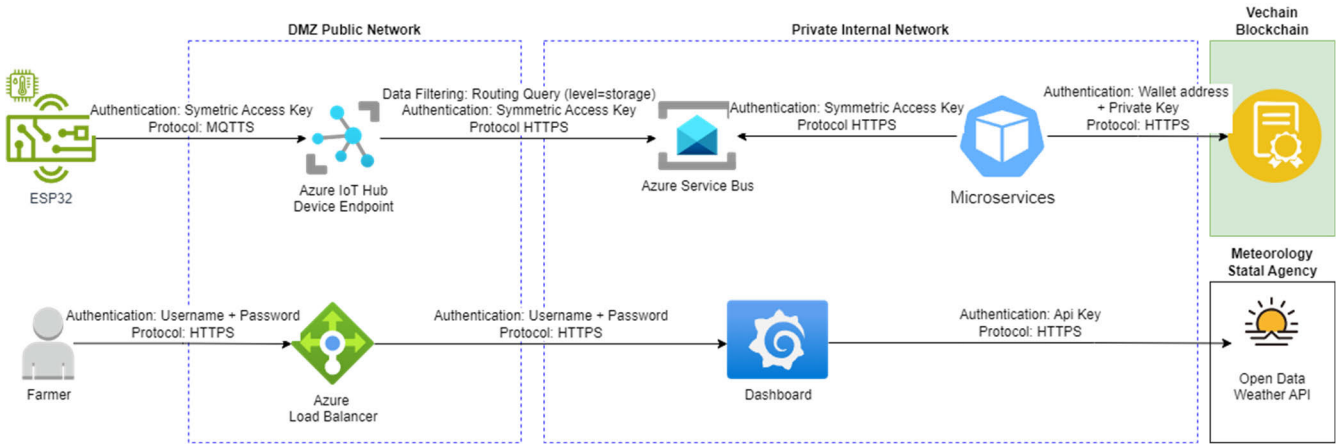


FIGURE 4. Security framework.

GUI ends. Therefore, a security framework was developed as part of the implementation works of the system.

The security of the system is designed focusing on securing the endpoints through authentication mechanisms, ensuring data encryption both in transit and at rest, and isolating critical components in an internal network. The components of the deployed security infrastructure are displayed in Figure 4. To enhance security, the system is divided into two distinct network segments. The first segment is a public network, also known as the Demilitarized Zone (DMZ), where publicly accessible endpoints are exposed. This includes an Azure IoT Hub endpoint to receive data from sensors and an endpoint from Azure Load balancer to manage and distribute requests for accessing the system dashboard. The other network segment is a private network where all the critical components, along with the dashboard, are hosted in isolation from the public network to mitigate potential security risks. Access to the Azure IoT Hub is secured using a symmetric access key and using a secured implementation of the MQTT protocol to encrypt data in transit. Only data containing specific system parameters are routed to the Azure Service Bus topic. Similarly, access to the Azure Service Bus is protected by a symmetric access key and uses the HTTPS protocol to ensure secure communication. Blockchain has also been incorporated to the defined security framework. For example, VeChain is used to store sensor data, ensuring that it remains transparent and verifiable by all stakeholders. Furthermore, smart contracts deployed on VeChain are utilized for securely storing data such as sensor readings and crop summaries; this immutability guarantees that the data cannot be tampered with after being recorded. Lastly, data transmissions with the smart contracts published on the VeChain network are restricted to a specific wallet secured with a private key.

Access to the open data weather API is secured using an API Key provided by the meteorology estate agency. Besides, all the communications with this API use the HTTPS protocol. Finally, Grafana dashboard uses a basic authentication mechanism of user and password along with the HTTPS protocol to establish a secure communication channel.

IV. TESTING AND PERFORMANCE

Once the system was designed, implemented and had its hardware and software components integrated, it was tested to assess its performance. To do so, there were several Key

Performance Indicators (KPIs) that, in addition to the Non-functional requirements, were established during this stage. Such KPIs are outlined below:

1. **Data Transmission:** The KPI for data transmission is the **reliability of data** being sent from the farm sensors to the Azure IoT Hub. This includes the percentage of successful message deliveries without loss or corruption.
2. **Data Processing:** The **time taken to process incoming data**—covering validation, transformation, and storage—is the primary KPI. This ensures that sensor readings are handled efficiently without delays that could impact downstream processes such as alerts and notifications.
3. **Blockchain Interaction:** The KPI for blockchain interaction involves **measuring the number of successful smart contract executions** and data entries onto the VeChain blockchain. Time metrics, such as latency for transactions to be confirmed on the blockchain, are also critical.
4. **User Notifications: Responsiveness** is the primary KPI. This includes the time from detecting an abnormal condition to alerting users through messages or visual indicators.
5. **Load Testing**: The KPIs for load testing include **system stability and resource usage** under various conditions, such as concurrent user loads and high-frequency data inputs. Metrics like CPU and memory utilization, as well as system throughput (requests processed per second), are monitored to validate scalability.

Additionally, the performance was measured according to all the criteria used in the Non-Functional Requirement 1, that is to say, the built system should support at Least 50 Concurrent Users without issue.

```
leona@DESKTOP-3H6N58I MINGW64 ~/Documents/tfm/3-App/product-summary
$ kubectl logs product-summary-666df5795f-txv4l -n tfm
Metrics Data:
First Record Date: 1720654270260
Last Record Date: 1720654807258
Max Temperature: 29
Min Temperature: 15
Average Temperature: 20.48
Max Humidity: 79
Min Humidity: 30
Average Humidity: 51.52
{
  txid: '0xe46b65ca504f699e20b60130708652a727b8fcea80666b5389c656f47348fbe6',
  signer: '0xf332b5f9add925725e5399b10af7a6ad6b5b0ff3'
}
```

FIGURE 5. Product summary logs uploaded to VeChain.

By using these KPIs it was ensured that the system could provide reliability, performance, and scalability while maintaining integration with blockchain technology. There are several experiments that were carried out to test the feasibility of the proposed solution.

A. SEND DATA FROM THE ESP32 MICROCONTROLLER

The system was first tested by sending data from the ESP32 microcontroller to the deployed SaaS solution, monitoring each phase of the process while measurements were collected. The ESP32, pre-loaded with the C-written code provided by Microsoft, was connected to a computer to provide power and start data generation. This computer was connected via Ethernet to the Internet, which would grant access to the deployed infrastructures. Random data were injected as feed to the system; it was considered acceptable at this point as we wanted to check on the connectivity of all the integrated subsystems. These data were transferred throughout the endpoint running with the deployed MQTT infrastructure. Upon logging into Azure Kubernetes Service (AKS) the correct status of the microservices was verified. Logs from the microservice responsible for reading from the system topic and storing data into VeChain were reviewed to verify that messages were read and stored successfully. Once logged into Grafana, data were checked to verify that ESP32 generated data were displayed correctly. Weather forecast for the following day data were also verified on the Grafana dashboard. Finally, product summary logs were generated and stored in VeChain using the Product Summary job. Summary data were made visible by consulting the transaction validator, built as a web application, with the transaction generated by the summary job. With this summarized information, a QR code was generated so it could be offered as an output for end users to visualize the status of the data transactions that had been carried out. Note that the information that the QR code contains is already verified and integrated within the VeChain network (as displayed in Figure 5, where the information that is made available via the QR code is available), so it can provide a reinforced trusted framework for end users.

B. UPLOAD AND VISUALIZE REAL DATA FROM AN OLIVE TREE FIELD

Once it was made clear that laboratory tests with sensing loads were successful, field data were collected from an olive tree field consisting of 20 olive trees located in an area close to the center of the Iberian Peninsula, at the village of Maqueda in Toledo [35]. To collect the data, a WSN was set throughout the olive tree field with three nodes on each of the olive trees, so temperature (measured in Celsius degrees), humidity (measured in percentage) and pressure (measured in pascals) readings were obtained.

In every tree, one of the nodes was used as a hub to transfer data, which would be in turn sent to the Microsoft Azure and VeChain infrastructures. The nodes consist of an encased set of Printed Circuit Boards (PCBs) manufactured by Libelium [36] and referred to as Waspmote. These nodes have enough RAM and ROM memory to run the programs uploaded to them and still collect the data [37]. The first node was located at the bottom of each of the trees, whereas the other two nodes were placed in the tree branches. The sensing board had the required sensors used for this test: temperature, humidity and pressure. The appearance of each of the nodes in the WSN can be seen in Figure 6. The Libelium board is on the top left side, whereas the sensing board that collects information is on the top right. Bottom left is the battery connected to the sensor to provide power and bottom right the XBee module [38] used as the wireless interface to send and receive data.

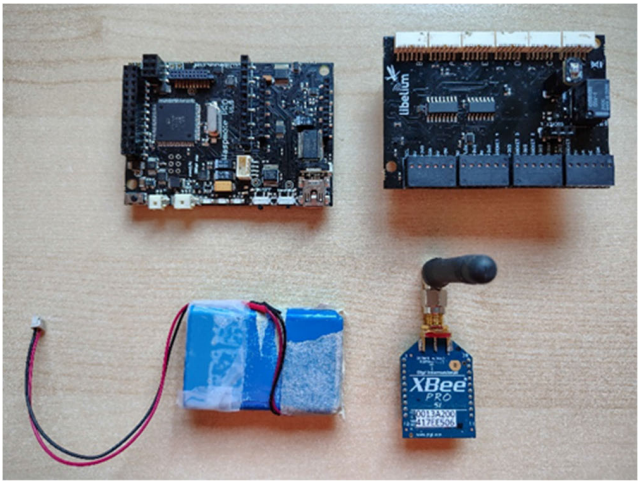
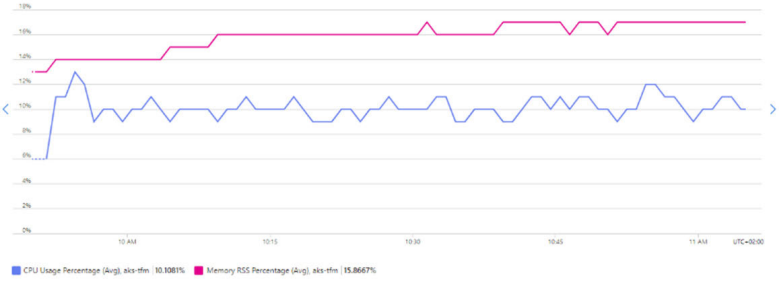


FIGURE 6. WSN node components.

The procedures followed to link that information onto the latter were like the ones used in the previous stage with laboratory randomly generated data, which proves that the system is flexible enough to adapt itself to different data sources. Even though the data were obtained from a different source, integration with all the other parts of the system was successful. Initially, recorded humidity levels were below expected thresholds, prompting manual watering

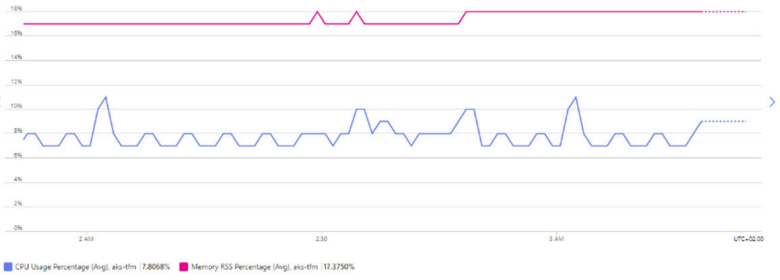
FIGURE 7. Olive tree measurements (temperature, humidity, pressure and their average values) as depicted in Grafana.

to boost humidity. This intervention succeeded in raising humidity but also led to a concerning drop in temperature, which fell to suboptimal levels. Eventually, temperature stabilized and returned to normal, although humidity levels fell down again, likely due to increased sun exposure. Meanwhile, atmospheric pressure remained consistent



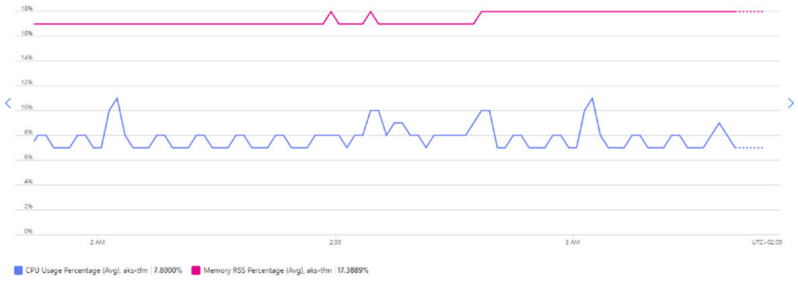
a)

# Samples	Average	Min	Max	Std. Dev.	Error %
1	267	267	267	0,00	0,00%
1	245	245	245	0,00	0,00%



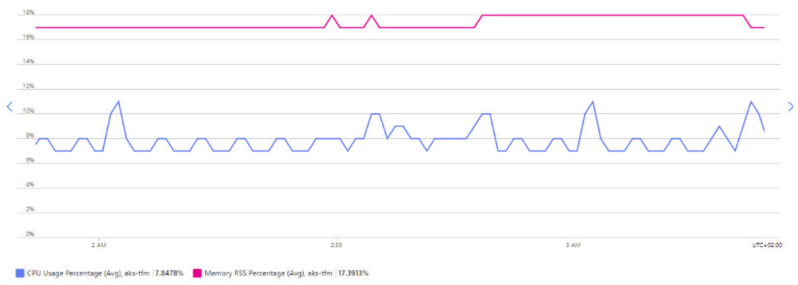
b)

# Samples	Average	Min	Max	Std. Dev.	Error %
20	287	181	402	64,33	0,00%
20	157	134	172	9,38	0,00%



c)

# Samples	Average	Min	Max	Std. Dev.	Error %
50	246	172	480	60,34	0,00%
50	157	133	317	25,28	0,00%



d)

# Samples	Average	Min	Max	Std. Dev.	Error %
100	206	169	399	36,36	0,00%
100	151	135	198	11,42	2,00%

FIGURE 8. Testing output with 1 user (8a), 20 users (8b), 50 users (8c) and 100 users (8d).

throughout the measurement period, as seen in Grafana in Figure 7.

Up until now, it can be inferred from the performed tests how the system managed to a) transmit and process information provided in second-like time intervals coming from hardware sensors and b) transmit and process information provided in seconds-like time intervals under a publish/subscribe paradigm such as the one used in MQTT data transfers. Blockchain interactions and data storage have also been tested successfully, along with user notifications via QR for end users. Therefore, and according to the KPIs that have been set for the development, load testing must be tested as well in order to get actual performance data from this aspect of the built system.

C. LOAD TESTING

In the first scenario, which involved making use of one concurrent user with a sending interval of 20 seconds, resource consumption was monitored using the Azure Monitoring Service. In this test, CPU usage increased from 13% to 17%, while memory usage rose from 6% to 11%, showing that a single load impacted the system in quite a minor manner. JMeter [39] depicted data confirmed that all requests were successfully processed, further emphasizing the effectiveness of the system under this initial testing condition. Figure 8a) shows the overall performance for this use case, as well as a JMeter output chart with two rows: the first one represents the number of attempts used to connect to the endpoint of the deployed subsystem, whereas the second one represents number of attempts used to inject olive tree data into the system. As shown, there were no errors. In the second load testing scenario, where 20 concurrent users were simulated with a ramp-up period of 10 seconds, Azure Monitoring Service provided once again major insights into resource consumption. During this test, CPU usage stabilized at 18%, while memory usage maintained an average of 9%, indicating that the system effectively managed the increased load without significant resource strain. Furthermore, JMeter data confirmed that all requests were successfully processed, as depicted in Figure 8b, demonstrating the robustness of the system under higher concurrency levels and underscoring its reliability in handling multiple simultaneous users. In the third load testing scenario, with 50 concurrent users and a 20-second ramp-up period, the system exhibited stable performance according to both Azure Monitoring Service and JMeter data. CPU usage remained consistent at 18%, while memory usage averaged 9%, indicating minimal resource strain despite the user load. JMeter results (see Figure 8c) further corroborated the system's efficiency, as all requests were successfully processed without any failures, confirming the system's ability to handle the specified load effectively.

In the fourth load testing scenario, with 100 concurrent users and a 30-second ramp-up period, the system maintained relatively stable resource usage but encountered some minor processing issues. The Azure Monitoring Service reported that CPU usage stayed at 18%, and memory usage averaged at 9%, indicating that the system was not heavily strained in terms of resources. However, JMeter results (see Figure 8d) revealed that 2% of the samples returned 502 error messages, suggesting that despite the stable resource metrics, a small portion of requests failed to process correctly under the increased load.

D. RESULTS DISCUSSION

The results of the proposed system demonstrate that the deployment of IoT sensors enabled real-time monitoring of critical parameters such as temperature, humidity, and pressure, which were securely stored on the VeChain blockchain. This ensured both the immutability of data and the transparency of farming conditions. Compared to existing solutions, the proposed architecture showed superior cost-effectiveness, scalability, and sustainability while maintaining high performance under different load scenarios. These findings align with previous research advocating for the use of IoT in improving agricultural efficiency and blockchain for ensuring transparency. However, the proposed solution goes beyond theoretical frameworks by delivering a practical implementation. Unlike the solutions discussed in prior studies, which often lacked real-time capabilities or depended heavily on proprietary platforms, this system provides an open-source, accessible alternative. Furthermore, the energy-efficient Proof of Authority (PoA) blockchain employed here addresses sustainability concerns noted in earlier works, where energy consumption was a significant drawback. However, the practical implementation of the system has boundaries to consider. The dependency on internet connectivity for real-time data transmission may pose challenges in remote areas with limited infrastructure. Additionally, the scalability of the VeChain blockchain and the overall system is currently limited to handle 80 devices at 30-second intervals, which might not suffice for larger farms or denser sensor deployments without additional computational resources. Also, the system's reliance on open-source tools and cloud infrastructure may introduce security risks that require ongoing monitoring and mitigation.

V. CONCLUSION AND FUTURE WORK

The system design, implementation and testing activities have successfully got through various critical phases, proving the system robustness in handling data flow and processing them. Data generation was effectively managed by an external microcontroller, followed by its reception at the Azure IoT Hub. These data were then published and consumed within the Azure Service Bus Topic. The system further processed and stored the data on the VeChain network, with alerts being triggered and notifications sent as required. Product summaries were also generated and stored on the VeChain network, with real-time data visualized alongside weather forecasts. Additionally, QR codes were generated to validate product summary conditions. During the load testing activities, the system experienced a minimal failure

when 100 users attempted to send messages within a 30-second period, caused by an error in the Upload VeChain microservice. Despite this, resource usage remained stable, and the auto-healing feature present in Kubernetes restarted the microservice, allowing the pending messages stored in the topic to be processed.

As for future works, there are several ideas that could be carried out. Implementing Machine Learning and AI techniques to analyse the data collected from IoT devices for predictive analytics could help in forecasting crop yields and optimizing resources usage. Exploring the integration of additional technologies such as Unmanned Aerial Vehicles (UAVs) for aerial monitoring, or robotics for automated farming tasks to further enhance operational efficiency and reduce labour costs could be considered too.

APPENDIX

The code repository for these development works that have been carried out is in GitHub, specifically in [40]. It can be accessed through the previously provided URL; implementation works can be followed and replicated at will, as it is an open-source repository where data have been made visible for any interested end user or developer.

ACKNOWLEDGMENT

As mentioned in the beginning of this paper, this work is supported by the Agencia Estatal de Investigación/Ministerio de Ciencia e Innovación through the project Sustainability-Aware IoT Systems Driven by Social Communities (SIoT-Com) under Grant PID2020-118969RB-I00.

REFERENCES

[1] Z. Xiao, Z. Li, Y. Yang, P. Chen, R. W. Liu, W. Jing, Y. Pyrloh, E. Sotthiwat, and R. S. M. Goh, "Blockchain and IoT for insurance: A case study and cyberinfrastructure solution on fine-grained transportation insurance," *IEEE Trans. Computat. Social Syst.*, vol. 7, no. 6, pp. 1409–1422, Dec. 2020, doi: 10.1109/TCSS.2020.3034106.

[2] P. M. Royo, J. Rodríguez-Molina, J. Garbajosa, and P. Castillejo, "Towards blockchain-based Internet of Things systems for energy smart contracts with constrained hardware devices and cloud infrastructure," *IEEE Access*, vol. 9, pp. 77742–77757, 2021, doi: 10.1109/ACCESS.2021.3081932.

[3] D.-Y. Jeong, S.-K. Jo, I.-B. Lee, H. Shin, and J.-G. Kim, "Digital twin application: Making a virtual pig house toward digital livestock farming," *IEEE Access*, vol. 11, pp. 121592–121602, 2023, doi: 10.1109/ACCESS.2023.3313618.

[4] K. P. Reddy, M. R. Thanka, E. B. Edwin, V. Ebenezer, S. S. Kirubakaran, and P. Joy, "Farm_era: Precision farming with GIS, AI pest management, smart irrigation, data analytics, and optimized crop planning," in *Proc. Int. Conf. Inventive Comput. Technol. (ICICT)*, Lalitpur, Nepal, Apr. 2024, pp. 1134–1140, doi: 10.1109/ICICT60155.2024.10544687.

[5] A. Shilpa, V. Muneeswaran, and D. K. D. Rathinam, "A precise and autonomous irrigation system for agriculture: IoT based self propelled center pivot irrigation system," in *Proc. 5th Int. Conf. Adv. Comput. Commun. Syst. (ICACCS)*, Coimbatore, India, Mar. 2019, pp. 533–538, doi: 10.1109/ICACCS.2019.8728550.

[6] M. P. Lamela, J. Rodríguez-Molina, M. Martínez-Núñez, and J. Garbajosa, "A blockchain-based decentralized marketplace for trustworthy trade in developing countries," *IEEE Access*, vol. 10, pp. 79100–79123, 2022, doi: 10.1109/ACCESS.2022.3194511.

[7] D. B. A. Kumar, N. Doddabasappa, B. Bairwa, C. S. A. Kumar, G. Raju, and Madhu, "IoT-based water harvesting, moisture monitoring, and crop monitoring system for precision agriculture," in *Proc. Int. Conf. Distrib. Comput. Electr. Circuits Electron. (ICDCECE)*, Ballar, India, Apr. 2023, pp. 1–6, doi: 10.1109/ICDCECE57866.2023.10150893.

[8] *VeChain—Web3 for Better. Whitepaper 3.0*. Accessed: Sep. 27, 2024. [Online]. Available: https://vechain.org/wp-content/uploads/2023/10/vechain-whitepaper-3-0.pdf

[9] Z. She. (2022). *VeChain: A Renovation of Supply Chain Management*. Accessed: Sep. 27, 2024. [Online]. Available: https://webofproceedings.org/proceedings_series/ESSP/ICEIEIS%202022/G68.pdf

[10] A. A. Aliyu and J. Liu, "Blockchain-based smart farm security framework for the Internet of Things," *Sensors*, vol. 23, no. 18, p. 7992, Sep. 2023, doi: 10.3390/s23187992.

[11] E. Vieira, J. Ferreira, and P. C. Bartolomeu, "Blockchain technologies for IoT applications: Use-cases and limitations," in *Proc. 25th IEEE Int. Conf. Emerg. Technol. Factory Autom. (ETFA)*, Vienna, Austria, Sep. 2020, pp. 1560–1567, doi: 10.1109/ETFA46521.2020.9211927.

[12] S. Narasimmasubramanian and S. A. Perumal, "A study on intelligence agriculture using blockchain with IoT," *Int. J. Eng. Res. Technol. (IJERT)*, vol. 9, no. 5, pp. 320–324, 2021.

[13] G. Xiao, N. Samian, M. F. M. Faizal, M. A. Z. M. As'ad, M. F. M. Fadzil, A. Abdullah, W. K. G. Seah, M. Ishak, and I. Hermadi, "A framework for blockchain and Internet of Things integration in improving food security in the food supply chain," *J. Adv. Res. Appl. Sci. Eng. Technol.*, vol. 34, no. 1, pp. 24–37, Nov. 2023, doi: 10.37934/araset.34.1.2437.

[14] *CargoCoin—Blockchaining Logistics*. Accessed: Sep. 27, 2024. [Online]. Available: https://thecargocoin.com/

[15] *IBM Supply Chain Intelligence Suite—Food Trust*. Accessed: Sep. 27, 2024. [Online]. Available: https://www.ibm.com/products/supply-chain-intelligence-suite/food-trust

[16] *Sustainable Shrimp Partnership: SSP*. Accessed: Sep. 27, 2024. [Online]. Available: https://sustainableshrimppartnership.org/

[17] I. I. Díaz, "Decentralized application (dApp) on the VechainThor blockchain for business processes," M. S. thesis, Dept. Faculty of Technology, Univ. South, Notodden, Norway, 2024.

[18] *IOTA: Home*. Accessed: Sep. 27, 2024. [Online]. Available: https://www.iota.org/

[19] J. Xia, H. Li, and Z. He, "The effect of blockchain technology on supply chain collaboration: A case study of lenovo," *Systems*, vol. 11, no. 6, p. 299, Jun. 2023, doi: 10.3390/systems11060299.

[20] R. Xu, Q. Lan, S. R. Pokhrel, and G. Li, "A knowledge graph-based survey on distributed ledger technology for IoT verticals," *ACM Comput. Surv.*, vol. 56, no. 2, pp. 1–36, Feb. 2024, doi: 10.1145/3609503.

[21] *IBM. Watson Discovery Pricing*. Accessed: Nov. 21, 2024. [Online]. Available: https://www.ibm.com/products/watson-discovery/pricing

[22] *Jhon Deere. Harvest Lab 3000*. Accessed: Nov. 21, 2024. [Online]. Available: https://www.deere.es/es/agricultura-de-precision/agricultura-espec%C3%ADfica-del-sitio/harvestlab-3000/

[23] *Smart Eye. SaaS Solution*. Accessed: Nov. 21, 2024. [Online]. Available: https://www.integratecnologia.es/smarteye/es

[24] J. C. Ú. Ortega, J. Rodríguez-Molina, M. Martínez-Núñez, and J. Garbajosa, "A proposal for decentralized and secured data collection from unmanned aerial vehicles in livestock monitoring with blockchain and IPFS," *Appl. Sci.*, vol. 13, no. 1, p. 471, Dec. 2022, doi: 10.3390/app13010471.

[25] *Microsoft Azure: Cloud Computing Services*. Accessed: Sep. 27, 2024. [Online]. Available: https://azure.microsoft.com/en-us

[26] *Use Deployment Scripts in Bicep—Azure Resource Manager*. Accessed: Sep. 27, 2024. [Online]. Available: https://learn.microsoft.com/en-us/azure/azure-resource-manager/bicep/deployment-script-bicep?tabs=CLI

[27] *Node. js—Run JavaScript Everywhere*. Accessed: Sep. 27, 2024. [Online]. Available: https://nodejs.org/en

[28] *ESP32-WROOM-32 (ESP-WROOM-32) Datasheet Version 2.4*. Accessed: Sep. 27, 2024. [Online]. Available: https://www.mouser.com/datasheet/2/891/esp-wroom-32_datasheet_en-1223836.pdf?srsltid=AfmBOooV0nOQkxuZ69LtkYkobNqDeoAUk9gmJm4jryIM9hLz81gBhxPO

[29] *Grafana: The Open Observability Platform*. Accessed: Oct. 8, 2024. [Online]. Available: https://grafana.com/

[30] *MQTT: The Standard for IoT Messaging*. Accessed: Sep. 27, 2024. [Online]. Available: https://mqtt.org/

[31] *Azure Container Registry*. Accessed: Sep. 27, 2024. [Online]. Available: https://azure.microsoft.com/en-us/products/container-registry

[32] P. Zhang, D. C. Schmidt, J. White, and A. Dubey, "Chapter seven—Consensus mechanisms and information security technologies," in *Advances in Computers*, vol. 115. Amsterdam, The Netherlands: Elsevier, 2019, pp. 181–209, doi: 10.1016/bs.adcom.2019.05.001.

[33] *Solidity Programming Language*. Accessed: Sep. 27, 2024. [Online]. Available: https://soliditylang.org/
[34] *VeChain Sync*. Accessed: Oct. 8, 2024. [Online]. Available: https://env.vechain.org/
[35] Diputación de Toledo, Maqueda, Spain. (2024). *Directorio de Municipios y EATIM (2024)*. Accessed: Sep. 29, 2024. [Online]. Available: https://www.diputoledo.es/global/11/50/169/dir_municipios/M/45091
[36] Libelium. *Connecting Sensors to the Cloud*. Accessed: Oct. 2, 2024. [Online]. Available: https://www.libelium.com/es/
[37] *Libelium Waspmote Datasheet*. Accessed: Oct. 2, 2024. [Online]. Available: https://www.libelium.com/wp-content/uploads/2013/02/waspmote-datasheet_eng.pdf
[38] *XBee?/XBee-PROS2CZigbee? RC Module*. Accessed: Oct. 11, 2024. [Online]. Available: https://www.digi.com/resources/documentation/digidocs/pdfs/90002002.pdf
[39] *Apache JMeter*. Accessed: Oct. 11, 2024. [Online]. Available: https://github.com/leonardoparejau/upm-iot-tfm
[40] *GitHub Code From Development Works*. [Online]. Available: https://jmeter.apache.org/

RENZO LEONARDO PAREJA URTECHO received the master's degree in Internet of Things from the Universidad Politécnica de Madrid, in July 2024, with the thesis titled "Agriculture Enhancement via Internet of Things Oriented Toward Sustainable Development Goals." He is currently the Founder and a Principal Consultant at Misof Consulting, specializing in DevOps, cloud technologies, and automation. He has experience implementing IT innovation systems across various industries, including banking, telecommunications, manufacturing, and government.

JESÚS RODRÍGUEZ-MOLINA received the Ph.D. degree (Hons.), in 2017, with the thesis titled "Contribution to the Design, Implementation and Standardization of Semantic Middleware Architectures for the Smart Grid." He is currently an Associate Professor with the Technical University of Madrid. He has performed research activities at ETH Zürich, SINTEF, NREL, CU Boulder, and TU Wien, Vienna. His research interests include distributed and cyber-physical systems, blockchain, autonomous vehicles, smart grid, and middleware.

MARGARITA MARTÍNEZ-NÚÑEZ is currently a Professor with the Department of Organization Engineering, Business Administration and Statistics, Technical University of Madrid. She has participated in several Journal Citation Reports publications with high impact and co-leads the European-level initiative for higher education called European Engineering Learning Innovation and Science Alliance (EELISA). Her research interests include digital transformation, smart grid, business and management, business models applied to sustainable development goals, and environmental economics.

JUAN GARBAJOSA (Life Senior Member, IEEE) joined Universidad Politécnica de Madrid (Technical University of Madrid, UPM), Spain, as a full-time Professor, in 1997, where he is currently a Full Professor with the Computer Systems School (ETS de Sistemas Informáticos). He is also the Deputy Vice-Rector for quality systems and competitiveness. Before that, he spent 16 years in industry and government in different engineering and management positions, ten of which at a start-up at the time, now a large multinational company. His current research interests include cyber-physical systems architecture, agile and innovation, and more recently collective intelligence.