engineering proceedings

MDPI

Proceeding Paper

A Fault-Tolerant Edge Computing Architecture for Low-Latency Actuation in Agricultural IoT †

Huynh Quy Khang [1], Pham Chi Thinh [2], Nguyen Dai Nghia [3] and Truong Minh Thai [1,*]

[1] College of Communication and Information Technology, Can Tho University, Can Tho 90000, Vietnam; hqkhang@ctu.edu.vn
[2] College of Engineering, Can Tho University, Can Tho 90000, Vietnam; thinhb2304673@student.ctu.edu.vn
[3] Can Tho University Software Center, Can Tho University, Can Tho 90000, Vietnam; ndnghia@ctu.edu.vn
* Correspondence: tmthai@ctu.edu.vn
† Presented at the 13th International Electronic Conference on Sensors and Applications, 18–20 November 2026; Available online: https://sciforum.net/event/ECSA-13.

Abstract

Latency and connectivity problems in traditional cloud computing platforms regularly hinder the use of Internet of Things (IoT) systems for smart agriculture, especially in remote rural areas. Network quality is an essential element of centralized architectures since they require continuous telemetry transmission to support real-time control decisions. This sensitivity leads to delayed control responses and possible over-control of high-capacity farm machinery, raising operating expenses. To solve these problems, this paper proposes a lightweight edge computing architecture based on embedded microservices for fast, reliable, and automated control. The thresholds for different environmental conditions are set directly on the ESP32 microcontroller, and a wireless sensor network is used to create a fully local control loop. This configuration enables edge devices to autonomously and accurately recognize anomalies in environmental conditions at the monitoring site, eliminating the need to await cloud server processing. The experimental system was built to demonstrate the benefits of on-premises processing over cloud-based processing by carefully measuring trigger latency and fault tolerance under different network conditions in rural areas. The results indicate that the Edge architecture reliably lowers network round-trip latency, reducing overall trigger latency by 40.09% compared to the cloud platform. Without network-related trigger delays, the system lets local hardware make control decisions quickly and reliably, even if the network goes down completely. Additionally, the architecture substantially decreases unnecessary energy consumption by heavy agricultural actuators, such as aeration pumps used in aquaculture farms. In summary, this decoupled architecture improves real-time response and energy efficiency. It also makes it easier to add fog computing and distributed artificial intelligence to smart agriculture systems.

Keywords: edge computing; agricultural IoT; dockerized microservices; autonomous agents; deterministic fault tolerance; smart agriculture

Check for updates

Academic Editor: Stefano Mariani

Published: 4 November 2026

Copyright: © 2026 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license.

1. Introduction

The Internet of Things (IoT) is of great importance in the field of precision agriculture, especially in climate-sensitive areas like the Mekong Delta, where it is necessary to carry out constant monitoring of the environment if healthy crops and aquatic life are to be maintained [1]. Cloud systems based on a microservice approach have improved scalability

and data tracking in agricultural IoT applications [2]. Yet the centralized nature of these architectures still faces basic infrastructure limitations. In a large number of remote areas, less than half of the people in rural regions have access to reliable internet connectivity [3]. As a result, the reliance on steady telemetry connections often leads to network delays. In emergency situations, unpredictable network latency can cause high-power devices, for example, 1500 W aeration pumps, to run for longer than they should. This problem reduces the accuracy of environmental controls and causes a great deal of energy to be wasted.

By bringing data processing closer to its source, edge computing helps to reduce the latency caused by the cloud [4]. Yet, a number of current edge solutions depend on artificial intelligence models that are resource-heavy for anomaly detection, making them impractical and too costly to use on microcontrollers, which have limited resources [5]. In order to overcome these drawbacks, this study presents a lightweight edge computing architecture that uses microservices to achieve reliable and fast autonomous control. The proposed system creates a local control loop by applying statistical median filtering directly on the LoRa-enabled ESP32 gateways, thus ensuring that the operation remains within the hardware's power limitations.

The study makes three main contributions: First, it develops a decentralized edge framework that separates the fast local control loop from the regular telemetry uploads, thus allowing for immediate action even when there is no internet connection; second, it incorporates a simple median filtering method on the edge node in order to tell the difference between real environmental changes and temporary hardware anomalies, thus making it unnecessary to use complex AI processing; and third, it shows that the proposed system reduces average actuation delay by 40.09% and cuts jitter by 96.12% as compared to cloud-based systems, leading to more reliable control and significant energy savings for large-scale agricultural machinery.

2. Related Work

The combination of Wireless Sensor Networks (WSNs) with cloud computing is highly effective for collecting large amounts of agricultural data [6,7], but its heavy dependence on continuous telemetry uplinks for real-time control is hindered by unpredictable network round-trip times and rural jitter [8]. The need for centralized remote control of high-capacity actuators necessarily causes communication delays, leading to a physical over-control of heavy machinery, which in turn reduces operational efficiency and increases energy waste. To reduce cloud latency, edge computing brings computational tasks closer to their sources [9]; yet most current research uses resource-intensive machine learning (ML) algorithms for anomaly detection [10]. Running such demanding models on constrained embedded devices such as ESP32 microcontrollers uses up the limited RAM and Flash storage [5], making them essentially unsuitable for large-scale and low-cost applications.

As a result, current IoT solutions are either hindered by vulnerabilities in cloud connectivity or by the high computational demand of edge AI, with very little attention paid to the industrial effects of network-induced over-actuation. To address this issue, the paper proposes a lightweight Edge Agent that incorporates microservices. Unlike the use of complex machine learning, a deterministic median filtering algorithm is implemented directly on microcontrollers enabled with LoRa to identify transient faults. Since the local control loop is entirely separated from the periodic uploads to the cloud, the design strictly limits actuation latency. The system has been verified through actual network deployments and not by means of simulation, and it thus ensures deterministic responsiveness, eliminates over-actuation induced by the network, and guarantees complete operational autonomy during internet outages with almost no computational overhead.

https://doi.org/10.3390/engproc2026163077

3. Proposed System Architecture & Methodology

3.1. Layered System Architecture

To overcome the inherent problems of latency, connectivity, and power consumption associated with centralized cloud computing in the field of smart agriculture, the present study suggests the use of a lightweight edge computing architecture based on microservices. By moving deterministic decision-making functions from the cloud to the network edge, the approach creates a control loop that is characterized by low latency, fault tolerance, and power optimization.

The architecture that has been proposed includes three layers: the Hardware Perception Layer, the Edge Gateway Layer, and the Cloud Microservices Backend (as illustrated in Figure 1).

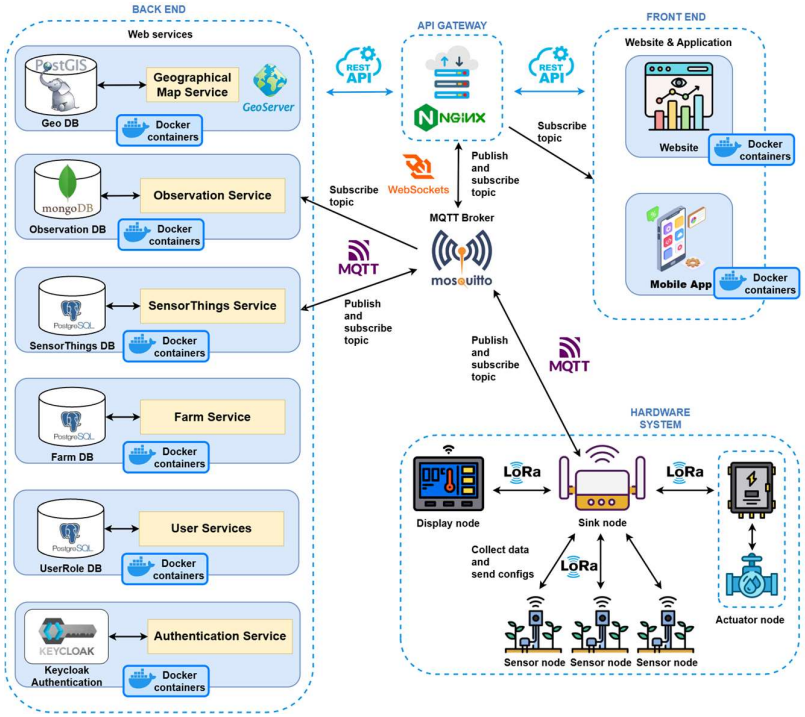


Figure 1. Suggested system architecture for edge computing.

The Hardware Perception Layer: Physically deployed in the field to execute localized farming tasks, this edge-level layer comprises sensor, actuator, and display nodes. To overcome the severe range and power limitations of Wi-Fi in expansive rural areas, these end nodes utilize the Long Range (LoRa) protocol to establish a localized star topology.

Edge Gateway Layer: The heart of the proposed architecture is the Sink node, since it acts as a bridge for edge computing by gathering LoRa packets from a number of sensor nodes and then connecting to the external internet using MQTT.

Cloud Microservices Backend: The cloud backend is made up of a central cloud infrastructure that makes use of Docker containers in order to achieve high scalability and to provide resource isolation. The MQTT telemetry messages are handled by the Mosquitto Broker and are then sent via the Nginx API Gateway. The backend contains a number of individual microservices, including GeoServer (Geo DB) for spatial mapping, the SensorThings Service, which handles the agronomic logic, and dedicated databases for Observation, Farm Management, and Keycloak Authentication.

https://doi.org/10.3390/engproc2026163077

3.2. Microservice-Embedded Autonomous Edge Agent

In conventional architectures, the Sink node functions simply as a 'dumb' forwarder, sending the LoRa sensor data directly to the Mosquitto broker and placing total reliance on the SensorThings Service to assess environmental thresholds and send actuation commands. As a result, a highly vulnerable and network-dependent control loop is produced.

To achieve this, we incorporate an Autonomous Edge Agent directly into the microcontroller of the Sink node. With the deterministic decision-making logic having been transferred from the Cloud to the Edge Gateway, the Sink node can then act as a local orchestrator. It takes in the LoRa sensor data, carries out real-time analytics, and is able to send LoRa control commands directly to the Actuator node (for example, water pumps or valves) without having to wait for a consensus from the Cloud backend.

3.3. Deterministic Fault Tolerance and Mathematical Models for System Optimization

To guarantee the reliable and efficient functioning of the autonomous Edge Agent, the proposed architecture includes an anomaly filtering mechanism at the Sink node and also sets up mathematical models to measure actuation latency and excessive energy consumption.

Agricultural sensors can give rise to abnormal readings because of electromagnetic interference, unstable power supplies, or temporary hardware disturbances, which might cause unnecessary control actions. In order to address this problem with as little computational demand as possible on the ESP32, a sliding-window median filter is used to eliminate impulsive noise and transient abnormalities while at the same time retaining genuine changes in the environmental conditions. The sensor readings after filtering are then employed to check for threshold violations before the Edge Agent carries out its local control decision.

When an actual environmental breach has been confirmed by the median filter, the system's responsiveness is determined by its latency model. In a traditional cloud-centric architecture, the decision-making process is wholly based on the central SensorThings service (1). The total cloud actuation latency (L_{Cloud}) is assumed to be the following:

$$L_{Cloud} = \Delta t_{actuator-sink} + \Delta t_{up} + \Delta t_{server-compute} + \Delta t_{down} + \Delta t_{sink-actuator} + \Delta t_{relay-trigger} \quad (1)$$

The latencies t_{up} and t_{down} refer to the network transmission delays, and tserver-compute stands for the amount of time taken for internal processing at the cloud server (1). On the other hand, in the edge-centric architecture proposed, the Sink node functions as the autonomous decision-maker (2). Once the threshold has been verified by means of the median filter, the actuation command is carried out immediately through the local LoRa network, without waiting for a response from the cloud (2). The total edge actuation latency (L_{Edge}) is therefore mathematically reduced to

$$L_{Edge} = \Delta t_{actuator-sink} + \Delta t_{agent-compute} + \Delta t_{sink-actuator} + \Delta t_{relay-trigger} \quad (2)$$

In theory, the system can function independently of cloud-based services when operating within the specified conditions, which in turn guarantees that the essential equipment will keep running (2).

The high latency associated with the traditional cloud-centric architecture can also lead to unnecessary energy consumption, especially in the case of systems that use high-power actuators. Since the latency of cloud-centric setups is highly dependent on network conditions, and since these conditions may be unstable in real-world situations, such latency can cause actuator deactivation commands to be executed with a delay (1), making

https://doi.org/10.3390/engproc2026163077

the actuators remain active when they should not. The operational power (E_{waste}) that is wasted as a result of this delay is given by

$$E_{waste} = P_{actuator} \times t_{waste} \quad (3)$$

where $P_{actuator}$ denotes the rated power of the agricultural machinery, and t is the excess operational time strictly caused by network latency (3). By carrying out the deactivation of the actuator locally without waiting for a response from the cloud, unnecessary energy consumption is reduced, especially in the case of large-scale deployments (3).

4. Experiment and Results

To assess the proposed edge computing architecture compared with the cloud computing architecture, a hardware-based experimental testbed was set up and tested under real-world network conditions. The experiments were intended to measure end-to-end actuation latency, the excess energy consumption caused by the network, and the system's resilience when operating under degraded or disconnected network conditions.

4.1. Hardware Testbed and Software Configuration

The testbed consisted of sensor nodes equipped with simulated probes, an Actuator node, which used a solid-state relay to simulate a high-capacity 1500 W aeration pump, and a central Sink node that coordinated a local LoRa star topology. The remote cloud backend, which was hosted externally, was fully based on Docker containers and included Eclipse Mosquitto (as an MQTT broker), an Nginx API Gateway, and a Quarkus-based SensorThings service.

The SX1278 LoRa transceivers used for communication between the Sink and Actuator nodes worked at a frequency of 434 MHz. To deliberately reduce the transmission latency of the control loop, the settings were optimized to a Spreading Factor of 7 (SF7), a bandwidth of 250 kHz, a coding rate of 4/8, a transmit power of 20 dBm, and a 12-symbol preamble.

To simulate the kind of internet variability found in rural areas, the Sink node was connected to the remote server through a 2.4 GHz Wi-Fi link provided by a commercial ISP. Two different network conditions were tested: a baseline scenario (with a jitter of 2.57 ms) and a congested scenario (with a jitter of 30.29 ms, this congestion being caused by heavy concurrent downloads). To ensure statistical reliability, 200 independent actuation samples were collected for each architecture. The latency was accurately recorded using the micros() function available on the ESP32 to detect hardware-level threshold violations and relay activations, together with the System.nanoTime() method of the Quarkus server in order to clearly separate out the computational overhead in the cloud.

4.2. End-to-End Actuation Latency and Stability Analysis

The experiment looked at the end-to-end actuation latency of both the cloud-centric and the edge-centric architectures under strong and weak network conditions, taking 200 measurements for each setup. Figure 2 shows the total processing time over the 200 iterations, and Table 1 gives a summary of the main statistical figures.

The findings indicate that edge-centric maintains considerably more stable processing times in both types of network conditions. In the case of strong network conditions, edge-centric attains an average latency of 0.314 s, which is 35.92% less than the 0.491 s seen with cloud-centric. When the network conditions worsen, edge-centric stays almost the same at 0.315 s, while cloud-centric rises to 0.526 s. Therefore, under unreliable rural network conditions, edge-centric reduces the mean actuation latency by 40.09% as compared to cloud-centric.

https://doi.org/10.3390/engproc2026163077

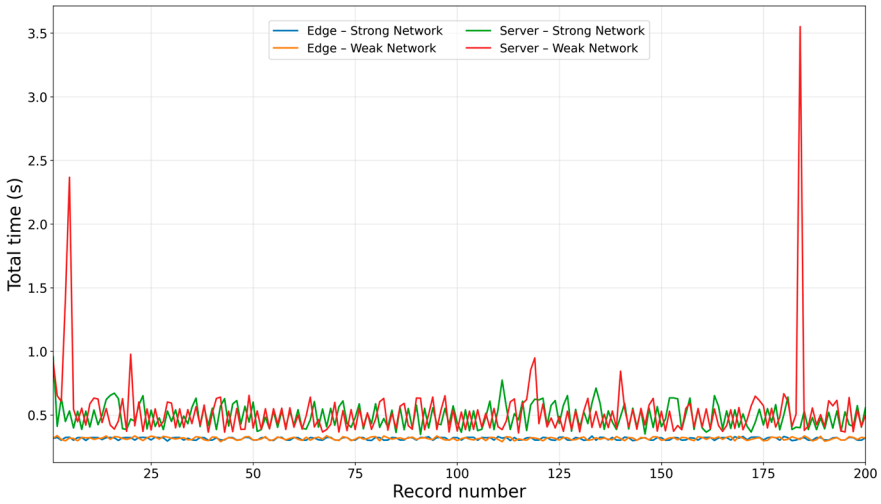


Figure 2. Comparison of total processing time under different network conditions.

Table 1. Statistical summary of total processing time.

Configuration	Mean (s)	Median (s)	Max (s)	SD (s)
Edge—Strong network	0.314	0.321	0.334	0.010
Edge—Weak network	0.315	0.317	0.338	0.011
Cloud—Strong network	0.491	0.480	0.958	0.097
Cloud—Weak network	0.526	0.504	3.551	0.284

The difference in stability is just as clear. While the standard deviation for edge-centric is only 0.010–0.011 s, that of cloud-centric is 0.097 s in the case of strong-network conditions and 0.284 s when the network is weak. Practical tests conducted under weak-network conditions therefore show an improvement in system jitter of 96.12%. It is worth noting that cloud-centric achieves a maximum latency of 3.551 s, whereas edge-centric has a maximum latency of just 0.338 s, which is a reduction of 90.48% in peak latency. These findings show that the proposed edge-centric architecture offers considerably lower and more predictable actuation latency, especially in situations involving unreliable rural networks.

Figure 3a,b show that communication overhead is the main factor causing end-to-end latency in both architectures, while computation time is negligible. In the case of the edge paradigm, the local LoRa transmissions are the most significant, with the Sink-to-Actuator and Actuator-to-Sink links accounting for about 49% and 41% of the latency, respectively. On the other hand, in the cloud scenario, the latency is mainly due to the internet uplink/downlink delays, which make up 41.51% under baseline conditions and rise to 45.41% when there is congestion.

Taken together, the latency caused by communication results in a considerable amount of energy being wasted. When a 1500 W pump carries out 100 cycles each day, the cloud-based method wastes about 7.4 Wh per day (2.70 kWh per year) under normal connectivity conditions and 8.8 Wh per day (3.21 kWh per year) during congestion. On the other hand, the edge-based architecture keeps the latency at a steady and deterministic level of about 0.31 s, thus eliminating the power inefficiencies associated with the internet.

In the end, a stress test that involved simulating a full network disconnection (100% packet loss) confirmed the system's strong fault tolerance. Although the cloud-dependent baseline experienced a complete critical failure as a result of the MQTT links being severed, the Edge Agent managed a 100% tasking success rate. The autonomous local control loop continued to function without interruption, thus ensuring the resilience of the mission-critical agricultural operations during severe internet outages.

https://doi.org/10.3390/engproc2026163077



Figure 3. The composition of the end-to-end actuation latency components for the edge and cloud architecture: (**a**) Contribution of each latency component under good network conditions. (**b**) Contribution of each latency component under poor network conditions.

5. Discussion

The empirical results show that moving the control loop to a local edge architecture is effective in separating actuation from variations in internet connectivity. Compared with the cloud-based baseline, the proposed system achieved a 36.05% reduction in mean actuation latency under normal conditions and a 40.11% reduction under network congestion. Most importantly, during severe congestion, it reduced the latency standard deviation (jitter) by 96.13% and the maximum latency by 90.48%. In the case of a 1500 W agricultural pump carrying out 100 cycles each day, eliminating this network-caused delay results in energy savings of between 0.074 and 0.088 Wh per cycle, which amounts to total annual operational energy savings of 2.70 to 3.21 kWh.

Moreover, the architecture ensures a 100% success rate for tasking and full operational independence even when the internet is completely out. When disconnected, edge nodes carry on with the tasks by using the configurations that are stored locally and at the same time hold back the actuation logs. As soon as connectivity is restored, a bidirectional synchronization takes place: the edge node retrieves the updated cloud thresholds (these take priority) and sends up its buffered execution logs to the server as definitive ground truth.

To attain fault tolerance for slowly varying environmental conditions (for example, dissolved oxygen or pH) without having to use the computational resources of current AI/ML edge models [5,11], the system includes a lightweight median filter directly on the ESP32. We do recognize that our present experiments have mainly been concerned with communication performance; it therefore remains a limitation that the exact degree of denoising achieved by the filter and the extra delay it introduces have not been quantified through fault-injection tests, a point that will be addressed in future work. In the end, this kind of resource-efficient approach proves to be practically essential for maintaining mission-critical agricultural operations in areas with limited infrastructure.

6. Conclusions and Future Work

The study offered an empirical verification of a lightweight edge computing architecture based on microservice features, which had been specially designed to overcome the problems of latency, network instability, and inadequate operational power efficiency that are present in conventional cloud-based agricultural IoT systems. This was done by decentralizing deterministic decision-making through the use of an autonomous agent

that carries out median filtering at the local LoRa gateway, thereby effectively separating the microsecond-scale actuation loop from the unstable telemetry uplink. Practical tests conducted on unreliable rural networks resulted in a 40.09% reduction in mean actuation latency and a 96.12% improvement in system jitter, the peak latency being brought down to 0.338 s. Building on this deterministic responsiveness, future work will focus on empirically implementing and validating the new feature at the edge, which is the elimination of network-induced "over-actuation" to achieve significant operational power savings for high-capacity agricultural machinery, while verifying a complete tasking success rate even when the internet is entirely unavailable. Because of the decentralized and resource-efficient character of this architecture, it forms a firm foundation for future work in fog computing, as an intermediate fog layer could be used to coordinate a number of autonomous edge nodes in order to achieve large-scale, federated agricultural automation while at the same time maintaining the resilience of local control.

Author Contributions: Conceptualization, T.M.T.; methodology, T.M.T.; software, H.Q.K.; hardware design and implementation, P.C.T. and N.D.N.; validation, T.M.T.; formal analysis, H.Q.K., P.C.T. and N.D.N.; investigation, H.Q.K., P.C.T. and N.D.N.; data curation, H.Q.K., P.C.T. and N.D.N.; visualization, H.Q.K.; writing—original draft preparation, H.Q.K., P.C.T. and N.D.N.; writing—review and editing, T.M.T.; supervision, T.M.T.; project administration, T.M.T.; funding acquisition, T.M.T. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by Model 11, "Application of Industry 4.0 to Monitor Agriculture and Aquaculture Environment", under the Technical Cooperation Project Phase 2 (TC2), funded by the Japan International Cooperation Agency (JICA).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data presented in this study are available from the corresponding author upon reasonable request.

Acknowledgments: The authors gratefully acknowledge the financial and technical support provided by the Japan International Cooperation Agency (JICA) through the Technical Cooperation Project Phase 2 (TC2). The authors also acknowledge Can Tho University for its support in the implementation and evaluation of the project activities related to this research.

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

References

1. Falcon, W.P.; Naylor, R.L.; Shankar, N.D. Rethinking Global Food Demand for 2050. *Popul. Dev. Rev.* **2022**, *48*, 921–957. [CrossRef]
2. Truong, T.M.; Huynh, K.Q.; Ngo, T.T.T.; Luu, C.; Le, B.M. The Microservice Based Architecture for IoT Systems in Agriculture and Aquaculture. In *Proceedings of the 2024 14th International Conference on Information Communication and Management, Paris, France, 6–8 November 2024*; Association for Computing Machinery: New York, NY, USA, 2025; pp. 22–28. [CrossRef]
3. International Telecommunication Union (ITU). *Measuring Digital Development: Facts and Figures 2023*; International Telecommunication Union (ITU): Geneva, Switzerland, 2023.
4. Shi, W.; Cao, J.; Zhang, Q.; Li, Y.; Xu, L. Edge Computing: Vision and Challenges. *IEEE Internet Things J.* **2016**, *3*, 637–646. [CrossRef]
5. Zhou, Z.; Chen, X.; Li, E.; Zeng, L.; Luo, K.; Zhang, J. Edge Intelligence: Paving the Last Mile of Artificial Intelligence with Edge Computing. *Proc. IEEE* **2019**, *107*, 1738–1762. [CrossRef]
6. Khanna, A.; Kaur, S. Evolution of Internet of Things (IoT) and its significant impact in the field of Precision Agriculture. *Comput. Electron. Agric.* **2019**, *157*, 218–231. [CrossRef]
7. Ray, P.P. Internet of things for smart agriculture: Technologies, practices and future direction. *J. Ambient Intell. Smart Environ.* **2017**, *9*, 395–420. [CrossRef]
8. Aazam, M.; Zeadally, S.; Harras, K.A. Deploying Fog Computing in Industrial Internet of Things and Industry 4.0. *IEEE Trans. Ind. Inform.* **2018**, *14*, 4674–4682. [CrossRef]

9. Khan, W.Z.; Ahmed, E.; Hakak, S.; Yaqoob, I.; Ahmed, A. Edge computing: A survey. *Future Gener. Comput. Syst.* **2019**, *97*, 219–235. [CrossRef]
10. Kamilaris, A.; Prenafeta-Boldú, F.X. Deep learning in agriculture: A survey. *Comput. Electron. Agric.* **2018**, *147*, 70–90. [CrossRef]
11. Ray, P.P. A review on TinyML: State-of-the-art and prospects. *J. King Saud Univ. Inf. Sci.* **2022**, *34*, 1595–1623. [CrossRef]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.