Leveraging Cloud-Native IoT for Cross-Sector Smart City Intelligence

Dimitris Karadimas*, Christos Panagiotou*,Katerina Karadima*
Dimitris Amaxilatis†, Orestis Akrivopoulos†,
Christos Gkrizis‡, Konstantinos Andriopoulos‡

*PIKEI New Technologies, Patras, Greece
†Sparkworks, Patras, Greece
‡Ergon Automation, Patras, Greece

***Abstract*—The growing complexity of urban ecosystems demands scalable, interoperable, and energy-efficient solutions for real-time monitoring and decision-making. This paper presents a modular, cloud-native Internet of Things (IoT) platform that unifies multiple smart city use cases—waste collection, water management, precision agriculture, and energy monitoring—within a cohesive architecture. At its core, the system integrates a custom-designed ESP32-based edge node capable of plug-and-play sensor integration, local processing, energy-aware duty cycling, and secure wireless communication using 4G, LoRaWAN, or Zigbee. Leveraging Amazon Web Services (AWS), the platform employs IoT Core, Lambda, Timestream, and other serverless components to deliver elastic data ingestion, processing, and analytics at scale. The edge node software supports fault-tolerant telemetry via local buffering and retries, remote reconfiguration, and secure Over-The-Air (OTA) updates, ensuring operational robustness in dynamic field environments. Real-world deployments across Greek municipalities demonstrate measurable improvements in resource efficiency and system responsiveness. The proposed architecture illustrates a practical and extensible approach to cross-domain smart city intelligence, bridging heterogeneous data sources and domains through a unified, cloud-integrated infrastructure.**

***Index Terms*—Internet of Things, Smart Cities, AWS, cloud-native**

I. Introduction

As urbanization accelerates globally, cities face growing challenges related to infrastructure efficiency, environmental sustainability, and quality of life. The concept of smart cities has emerged as a comprehensive framework to address these challenges by leveraging Information and Communication Technologies (ICT), especially the Internet of Things (IoT), to improve public services through real-time monitoring, data-driven decision-making, and automation.

Over the past decade, various IoT-based applications have been introduced to solve specific urban problems. Examples include smart waste collection, water quality monitoring, precision agriculture, and energy consumption tracking. However, most of these initiatives have been implemented as vertical silos, each with its own architecture, devices, and data pipelines. This lack of interoperability and cross-domain integration increases system complexity and inhibits scalability, particularly when extending solutions across different city departments or geographic regions.

To address these limitations, cloud-native IoT architectures have gained traction. By leveraging the scalability, flexibility, and reliability of commercial cloud platforms, such as Amazon Web Services (AWS), it is now possible to create unified platforms that support multiple use cases through reusable and modular services. Serverless computing, event-driven messaging, and distributed data processing enable the rapid deployment of diverse applications without the burden of managing infrastructure.

This paper presents a cloud native and scalable AWS-based IoT architecture that supports a variety of smart city use cases within a single cohesive framework. The proposed system has been implemented and validated through four distinct but complementary deployments:

- Smart Waste Collection using weight sensors on garbage trucks;
- Water Monitoring and Leakage Detection through distributed pressure and flow sensors;
- Precision Agriculture with environmental sensing and pest detection (Dacus fly) in olive trees;
- Energy Consumption Monitoring using smart metering solutions.

These use cases span both urban and peri-urban environments, demonstrating the versatility of the architecture. The system is designed to be extensible, secure, and modular, relying on AWS services such as IoT Core, Lambda, S3, and DynamoDB to manage telemetry, execute real-time processing, and store large volumes of sensor data. A key component of the presented system is the edge computing node that has been designed and developed based on the ESP32 microcontroller, so as to support:

- Plug-and-play sensor integration, enabling flexible deployment across different environments;
- Multi-protocol wireless communication, including Wi-Fi, Bluetooth, LoRa, Zigbee, and LTE (via optional modules);
- Precision Agriculture with environmental sensing and pest detection (Dacus fly) in olive trees;
- Secure and reliable cloud connectivity, tightly integrated with AWS IoT Core.

The remainder of the paper is organized as follows: Section 2 reviews related work and background on smart city IoT platforms; Section 3 details the proposed system architecture and its AWS components; Section 4 presents the implementation of the four use cases; Section 5 discusses scalability and deployment insights; Section 6 evaluates the system through experimental results; and Section 7 concludes with lessons learned and future work directions.

II. Background

Over the last decade, various domain-specific IoT solutions have been proposed for challenges such as waste collection [1], water distribution management [2], precision agriculture [3], and energy monitoring [4]. However, most existing approaches focus on isolated use cases, lacking a unified architectural framework that supports cross-domain integration, scalability, and seamless deployment.

Several research efforts have explored generic smart city architectures. Gaur et al. [5] introduced a layered architecture for urban data collection and service integration, but lacked implementation scalability details. The EU-funded OrganiCity project [6] proposed an experimentation-as-a-service platform, focusing on citizen participation rather than infrastructure interoperability. Recent works, such as those of Syed et al. [7], emphasize the importance of using cloud and edge resources collaboratively to manage heterogeneous data streams and devices. However, end-to-end deployments covering multiple verticals on commercial cloud infrastructures remain limited.

Cloud-based IoT frameworks, such as AWS [8], Azure [9], or Google Cloud Platform (GCP) [10], offer significant advantages in elasticity, security, and service orchestration. AWS in particular provides a rich set of services, including AWS IoT Core, Lambda, and S3, enabling the construction of modular and serverless pipelines [11]. Previous studies have applied AWS-based IoT to specific domains, for example, predictive maintenance in industrial settings [12] or water quality monitoring [13]—but comprehensive platforms that address multiple domains through a shared architecture remain underexplored.

Furthermore, smart cities increasingly demand interoperability and rapid scalability. Architectures based on loosely coupled microservices, event-driven data processing, and edge-cloud synergies are seen as the future-proof approach for handling urban complexities [14]. Our work builds upon this vision, delivering a fully operational multi-domain IoT platform using AWS technologies, validated through real-world deployments in waste management, water systems, agriculture, and energy monitoring.

III. Architecture

The proposed system architecture is designed to bridge the sensing capabilities of geographically distributed, energy-constrained devices with the scalability and elasticity of modern cloud platforms. It follows a modular and cloud-native design philosophy, ensuring that diverse smart city use cases can be supported within a unified framework. At the edge, robust and lightweight nodes interface with heterogeneous sensors, process and buffer data locally, and maintain resilient wireless communication using multiprotocol connectivity including WiFi, Bluetooth, ZigBee and LoraWAN and 5G.

The edge node has been designed as a compact and modular hardware platform to support diverse environmental sensing tasks in urban and rural deployments. The PCB, designed internally by PIKEI, hosts the LILYGO® T-Call A7670 [15], a versatile ESP32-based microcontroller is chosen for its ability to combine embedded processing within a low-power footprint. The layout and placement of components are optimized to facilitate plug-and-play sensor integration, power autonomy, and reliable wireless transmission.

At the center of the design lies a dual-row socket footprint that accommodates the LILYGO® T-Call A7670 module. The PCB routes critical signals—including UART, GPIOs, I2C, and ADC-compatible lines—to nearby sensor headers and driver circuits, allowing seamless interfacing with multiple analog and digital sensors. Peripheral components are strategically arranged to support key sensing tasks in the air quality, temperature, and gas detection domains.

To support off-grid deployments, the PCB integrates a boost converter and a solar charger section. This subsystem enables operation from solar energy, charging an external LiPo battery connected via the S+ / S– terminals. The power design ensures a regulated supply to both the ESP32 core and the attached sensors, with protection and charging control managed by an onboard boost IC (likely TP4056 or similar).

The PCB is two-layered, with clearly separated analog and digital domains to reduce noise interference. Careful routing ensures short, direct paths from the sensor connectors to the MCU pins, improving analog signal fidelity and digital timing. The high-current traces from the power input to the MOSFET drivers are widened, and ground planes are used to reduce impedance and enhance electromagnetic compatibility.

The edge node's compact form factor and well-distributed connector layout facilitate installation in diverse physical environments—from waste collection vehicles and urban kiosks to rural farmlands and remote water monitoring stations.

A. Edge Node: Software

The software stack running on the ESP32-based edge node is engineered to optimize energy efficiency, ensure data reliability, and enable flexible, remotely managed operation across diverse field conditions. Its core runtime operates under FreeRTOS, enabling concurrent task execution for sensing, communication, and power management while maintaining deterministic behavior even under constrained resources.

To prolong battery life, the system employs a duty cycling mechanism that dynamically adjusts the active and sleep intervals of the node based on operational context and sensor workload. Using the ESP32's deep sleep functionality, the microcontroller enters low-power mode when idle, waking up periodically or based on event-driven triggers to perform sensing and transmission tasks. Sensors with high current draw, such as the MQ gas sensors, are power-cycled through

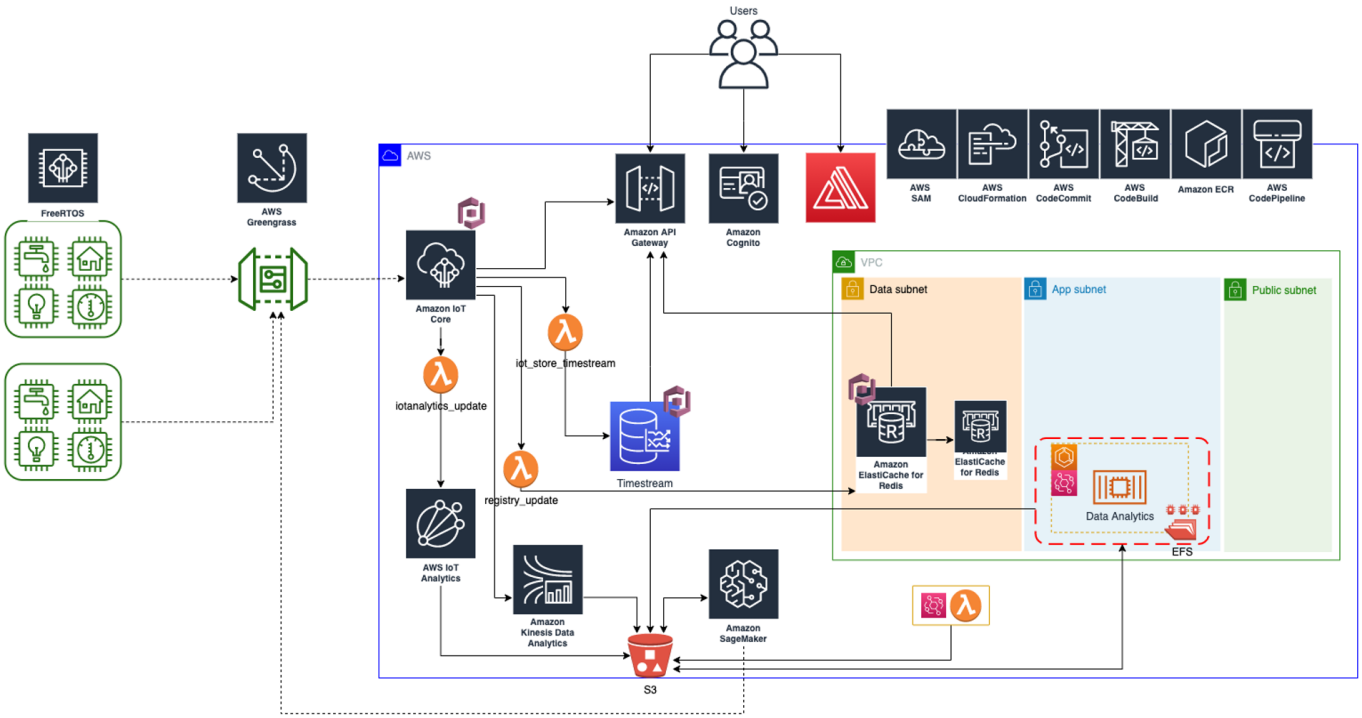

Fig. 1. AWS Cloud Architecture.

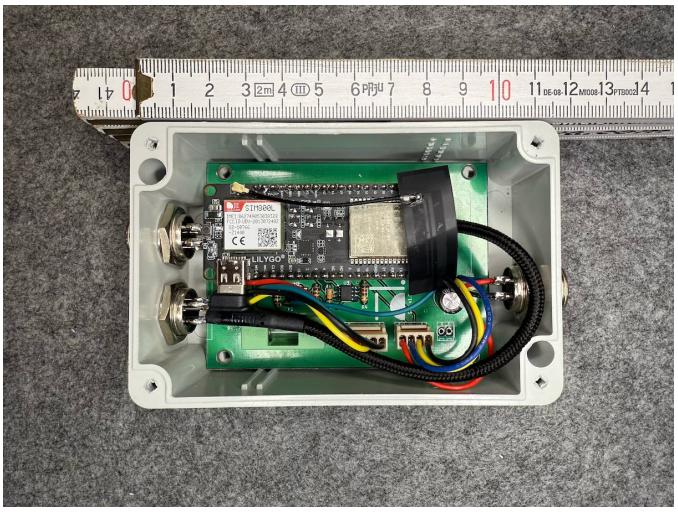

Fig. 2. Edge node in the garbage collection use case version(custom pcb with the attached ESP32 board).

software-controlled MOSFETs, and their pre-heating times are managed precisely to reduce unnecessary power use without compromising measurement accuracy.

In terms of communication, the software stack implements MQTT over LTE to publish telemetry to AWS IoT Core or equivalent brokers. The MQTT client handles session persistence, automatic reconnection, and publish retries with exponential backoff, ensuring data delivery robustness even under fluctuating network conditions. To further enhance resilience, the system incorporates a local storage buffer based on the SPIFFS or LittleFS file system. When network access is unavailable, telemetry data are stored locally in compressed JSON format. A background task continuously monitors connectivity and flushes buffered data to the cloud as soon as transmission becomes feasible again, guaranteeing eventual consistency of the data stream.

The firmware also features remote configuration capabilities, enabling parameter tuning without physical access to the device. Settings such as sampling intervals, duty cycle durations, sensor thresholds, and destination topics can be updated through MQTT control channels. Upon receiving authenticated reconfiguration messages, the node applies the new settings immediately or after the next scheduled reboot, depending on their criticality.

Overall, the software ecosystem complements the hardware's design by enabling energy-aware operation, robust data handling under intermittent connectivity, and full remote lifecycle management. It transforms the edge node into a resilient and adaptive platform capable of supporting long-term deployments in both urban and remote smart city environments.

The attached sensors, regardless the use case, feed the edge node that digitizes analog inputs via the integrated ADC, conducts local data processing, and attaches metadata (e.g. location, timestamp.

B. Cloud Services

The architecture depicted in Fig. 1 represents the scalable, cloud-native IoT infrastructure supported by Amazon Web Services that support a range of smart city use cases, including smart waste collection, water leakage detection, environmental monitoring in agriculture, and real-time energy consumption tracking. At the edge, diverse sensor devices—installed on garbage trucks, in water networks, in agricultural fields, and within buildings—act as data sources feeding telemetry into the cloud platform through AWS IoT Core.

The edge layer based on the edge node described previously, is generalized to include all sensors deployed in the field. These include hydraulic pressure sensors on waste trucks, water flow and pressure sensors in municipal pipelines, environmental sensors (e.g., temperature, humidity, pest traps) in olive tree cultivations, and smart energy meters in residential and public infrastructures. These devices connect via protocols such as MQTT or HTTP to AWS IoT Core, which act as the primary broker for device communications and state management.

Once ingested by AWS IoT Core, telemetry is routed via AWS Lambda functions to downstream services. One path sends data to Amazon Timestream, a purpose-built time series database optimized for storing and querying sensor data. Another path forwards processed telemetry to AWS IoT Analytics, which enables complex transformations, filtering, and pipeline orchestration for downstream analytics. These Lambda functions also handle data normalization, device registry operations, and user-triggered queries for historical analysis.

At the same time, AWS API Gateway exposes a set of RESTful APIs for external integration and user interaction. These APIs, secured via Amazon Cognito for identity and access management, allow authenticated users and applications to perform operations such as device registration, group management, and filtered queries. Through these APIs, alerts and notifications can be generated based on defined rules, which are delivered using Amazon SNS (Simple Notification Service) for real-time messaging (e.g., leak detection or anomaly alerts).

On the application side, the system is deployed within a Virtual Private Cloud (VPC) composed of multiple subnets:

- The Public Subnet hosts an Application Load Balancer, which routes traffic to internal services.
- The App Subnet hosts the Web UI, a dashboard interface through which municipal operators and users interact with the platform
- The Data Subnet hosts a Data Analytics engine, which aggregates, visualizes, and models data using infrastructure that includes Amazon S3 (for object storage), EFS (Elastic File System), and potentially big data frameworks for heavier analytics.

The architecture is modular and supports scaling both horizontally (more devices, more regions) and vertically (more complex analytics per use case). The serverless nature of the backend, powered by Lambda, ensures high elasticity, reduced infrastructure overhead, and fault-tolerant data flows.

In this design, each domain—waste, water, agriculture, energy—can be independently developed and deployed while still benefiting from a shared data pipeline, analytics core, and user interface, thus embodying the principles of reusability, modularity, and interoperability that are critical in modern smart city platforms.

C. Unified API Layer

To ensure a consistent and scalable way of interfacing with the collected IoT data across diverse smart city verticals—ranging from waste management and water monitoring to precision agriculture and energy efficiency—the system exposes a unified, RESTful API. This API plays a dual role: it serves as both a data integration gateway for external third-party systems and as a backend service layer for user-facing dashboards and analytics tools.

Designed with generality and reuse in mind, the API abstracts away the domain-specific details of each vertical and instead relies on generic data models and endpoints. Each data stream is associated with a logical site, device, and sensor concept, making it straightforward to query or ingest telemetry regardless of whether the data originates from a solar-powered Dacus trap, a garbage bin fill-level sensor, or a water quality node. This decoupled structure enables the same access pattern and authentication mechanisms to be reused across all deployments, reducing the complexity of both integration and maintenance.

The API supports secure access to historical and real-time telemetry data, device metadata, alerting thresholds, and semantic labels, making it suitable for programmatic integration into external platforms (e.g., municipal open data portals, environmental dashboards) as well as internal services such as the official Smart City Chalkida web UI. It is implemented using modern OpenAPI standards, with interactive documentation that facilitates adoption by external developers and integration stakeholders.

Moreover, the API also offers support for spatial queries, unit conversion, data interpolation, and role-based access control (RBAC), all of which are critical for enabling scalable, secure, and citizen-centric applications. By designing this interface to be generic, the platform ensures future-proof compatibility with new verticals and partners, reducing the cost of onboarding new use cases and fostering a true data space approach for urban intelligence.

IV. Use Cases

A. Garbage Collection Monitoring

In the Smart City Chalkida deployment, the waste collection use case integrates an edge-based sensing system into municipal garbage trucks, enabling real-time, dynamic weight estimation of waste bins during lifting. The system employs a hydraulic pressure sensor installed in the lifting actuator's oil circuit to measure the load indirectly through pressure changes. To account for the dynamics of the lifting motion, inductive

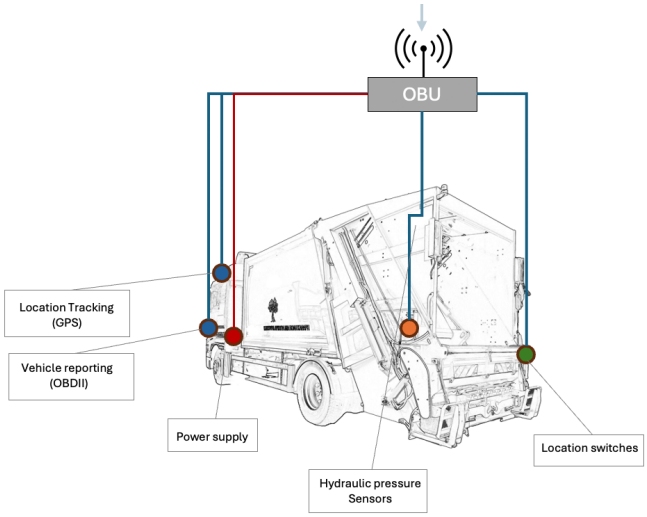

Fig. 3. Vehicle installation.

position sensors are mounted on the lifting mechanism. These sensors detect movement in real time, enabling the system to compute lifting speed and apply motion-based corrections to derive accurate weight estimations without requiring the bin to be held stationary.

This real-time processing is executed on a custom ESP32-based edge node that digitizes analog pressure signals via an on-board ADC (Analog-to-Digital Converter), processes them locally, and associates them with geospatial and technical metadata. Each vehicle also features a GNSS (Global Navigation Satellite System) receiver for positioning and an OBD-II (On-Board Diagnostics) interface for collecting engine and vehicle telemetry such as speed, fuel consumption, and operating status. All data streams are coordinated by the edge node, which handles timestamping, fusion, and transmission using available wireless technologies (Wi-Fi, NB-IoT, or 4G/5G), with offline buffering during network outages and automatic retransmission when connectivity is restored.

Once data exit the vehicle, they are routed to the cloud-based IoT Analytics Engine, where AWS services such as IoT Core and Lambda handle ingestion, processing, and integration with dashboards and fleet management systems. This lightweight yet robust vehicle-side infrastructure has been deployed across a fleet of 30 garbage trucks, enabling automated waste monitoring, operational analytics, and route optimization while maintaining low maintenance overhead and minimal vehicle intrusion.

B. Water & Energy Monitoring

The water and energy management use case integrates an edge-based smart monitoring system into municipal water distribution networks and building-level energy infrastructures, enabling near real-time telemetry collection, anomaly detection, and consumption forecasting. For water monitoring, the system employs ultrasonic flow sensors installed on main pipelines to measure flow rates accurately across the full dynamic range without requiring invasive modifications. These sensors support the detection of irregularities such as leaks, dry conditions, or tampering, and are complemented by temperature sensors to adjust for thermal effects on fluid dynamics and usage patterns.

In parallel, energy monitoring is facilitated through the deployment of digital electricity meters capable of capturing power consumption profiles at high resolution. These meters are integrated into public buildings and infrastructure hubs, enabling continuous tracking of energy usage, detection of overconsumption or abnormal load patterns, and long-term trend analysis that informs sustainability planning.

All data streams are transmitted via cellular connectivity to the cloud-hosted IoT Analytics Engine. Leveraging AWS services such as IoT Core, Lambda, and DynamoDB, the platform performs ingestion, stream processing, and integration with municipal dashboards and alerting systems. The infrastructure has been deployed across multiple monitoring points in the water networks of urban and coastal municipalities, as well as in public administrative buildings. This allows for automated leak detection, energy optimization, predictive maintenance, and the reduction of operational overheads—all without requiring significant infrastructure disruption.

C. Precision Agriculture

An added in the precision agriculture use case is the automated monitoring of Bactrocera oleae (olive fruit fly) populations using smart traps deployed across olive orchards in Western Greece. These traps are enhanced with image capture and wireless connectivity capabilities, enabling remote, low-maintenance pest surveillance. The core sensing device of the edge node is an ArduCam Mega 5MP SPI Camera Module with Autofocus Lens. This camera was selected for its compatibility with microcontrollers over the SPI interface, compact footprint, low power consumption, and ability to capture high-resolution color images (up to 2592x1944) necessary for insect detection. The autofocus capability further improves image quality in field conditions where trap distance and orientation can vary.

Images are captured once daily, typically during peak fly activity hours, and transmitted over the mobile network to the cloud. Due to the low volume of images and absence of real-time processing requirements, inference is performed entirely in the cloud. Upon arrival at the backend infrastructure, images are ingested and routed to a storage and analysis pipeline comprising Amazon S3 and AWS SageMaker. A deep learning model, based on a lightweight object detection architecture such as YOLOv5 or MobileNet-SSD, processes each image to identify and count individual flies trapped on sticky surfaces.

The ArduCam Mega 5MP module's ability to deliver consistent image quality, even in low-power environments, enables robust pest detection while preserving energy. Additionally, since inference is offloaded to the cloud, the system avoids computational load on the edge device, extending battery life and reducing hardware cost. The daily pest counts are

integrated into the farmer's dashboard, along with correlated environmental readings, allowing for spatiotemporal trend analysis and early warning mechanisms to be developed.

V. Platform Evaluation

The deployed architecture was evaluated for its ability to sustain scalable, low-latency operation under multi-domain IoT workloads, relying entirely on cloud-native services and cellular communication. As already presented, the infrastructure is built on top of AWS, leveraging IoT Core for device ingestion, Timestream for time-series persistence, Lambda functions for event routing, and SageMaker for inference tasks. Performance testing and observation revealed consistent and reliable operation under heterogeneous data streams.

The average latency from edge device to cloud ingestion via MQTT over LTE is measured below 500 ms, including TLS handshake, MQTT broker processing, and routing through AWS IoT Core Rules Engine. For data requiring post-processing via serverless functions (e.g., bin lift pairing, sensor enrichment), the end-to-end processing time remains well under 1 second, even during peak submission periods. These latency figures were derived from timestamped logs captured at both edge firmware and backend ingestion points.

The platform demonstrated horizontal scalability, with no increase in ingestion latency as new devices were onboarded during field expansion. IoT Core and Timestream services automatically scaled to handle peak data rates without requiring manual intervention, validating the serverless and autoscaling principles applied in the platform's design.

To ensure resilience under intermittent connectivity, edge nodes were tested with controlled network outages. Devices were able to buffer data for hours using a lightweight local storage queue, and successfully retransmit upon reconnection. Data integrity checks confirmed zero loss or duplication under real-world field conditions, thanks to per-message timestamping and idempotent ingestion logic in the backend.

Cloud-side inference tasks—particularly those used for Dacus image analysis—were measured under batch processing conditions. Each batch of 10 images was processed by a SageMaker endpoint using a YOLOv5 lightweight model in under 2.5 seconds, including S3 download, inference, and result serialization. This performance is more than sufficient given the low frequency of image submissions (3 per trap per day) and allows for near real-time dashboard updates post-upload.

The system also supports multi-tenant isolation and fine-grained RBAC through its API gateway. Role-specific query filtering and API token rate-limiting allow for safe concurrent access by municipal operators, farmers, and external developers without observable impact on performance.

Finally, from a maintenance perspective, the platform has achieved 99.9% uptime across all backend services since deployment, with automated logging, alerting, and cloud-based observability in place through CloudWatch and IoT Device Defender.

VI. Conclusions

The presented use cases supported by cloud-native approach showcase a transformative approach to urban sustainability through the integration of edge-based IoT systems and IoT Analytics Platform. By deploying tailored sensor networks—hydraulic pressure sensors for waste bins, ultrasonic flow and quality sensors for water pipelines, weather stations and smart Dacus traps for agriculture, and smart meters for energy grids - stakeholders achieve real-time monitoring and optimization across critical municipal functions.

Acknowledgment

Part of this work has received funding from the European Union's Interregional Innovation Investments Instrument (I3) under Grant Agreement No.101115116 and the AMBITIOUS ("Advanced CoMputing Continuum Solutions for Boosting DigITalization across European RegionS") European research project.

References

[1] I. Sosunova and J. Porras, "Iot-enabled smart waste management systems for smart cities: A systematic review," *IEEE Access*, vol. 10, pp. 73 326–73 363, 2022.

[2] S. Bera, S. Misra, and A. V. Vasilakos, "Software-defined networking for internet of things: A survey," *IEEE Internet of Things Journal*, vol. 4, no. 6, pp. 1994–2008, 2017.

[3] N. ElBeheiry and R. S. Balog, "Technologies driving the shift to smart farming: A review," *IEEE Sensors Journal*, vol. 23, no. 3, pp. 1752–1769, 2022.

[4] O.-G. Gemein, H. Niemi-Hugaerts, L. Sanchez, S. Ziegler, and S.-M. Miteva, "Smart cities," in *Springer Handbook of Internet of Things*. Springer, 2023, pp. 523–554.

[5] A. Gaur, B. Scotney, G. Parr, and S. McClean, "Smart city architecture and its applications based on iot," *Procedia computer science*, vol. 52, pp. 1089–1094, 2015.

[6] D. Amaxilatis, D. Boldt, J. Choque, L. Diez, E. Gandrille, S. Kartakis, G. Mylonas, and L. S. Vestergaard, "Advancing experimentation-as-a-service through urban iot experiments," *IEEE Internet of Things Journal*, vol. 6, no. 2, pp. 2563–2572, 2018.

[7] A. S. Syed, D. Sierra-Sosa, A. Kumar, and A. Elmaghraby, "Iot in smart cities: A survey of technologies, practices and challenges," *Smart Cities*, vol. 4, no. 2, pp. 429–475, 2021. [Online]. Available: https://www.mdpi.com/2624-6511/4/2/24

[8] Amazon Web Services, Inc., "Amazon Web Services (AWS)," https://aws.amazon.com/, 2024, accessed: 2025-06-29.

[9] Microsoft Corporation, "Microsoft Azure Cloud Computing Services," https://azure.microsoft.com/, 2024, accessed: 2025-06-29.

[10] Google LLC, "Google Cloud Platform (GCP)," https://cloud.google.com/, 2024, accessed: 2025-06-29.

[11] Amazon Web Services, Inc., "AWS IoT Core Documentation," https://docs.aws.amazon.com/iot/, 2024, accessed: 2025-06-29.

[12] A. Chakrabarti, P. K. Sadhu, and P. Pal, "Aws iot core and amazon deepar based predictive real-time monitoring framework for industrial induction heating systems," *Microsystem Technologies*, vol. 29, pp. 441–456, 2023. [Online]. Available: https://link.springer.com/article/10.1007/s00542-022-05311-x

[13] E. E.-D. Hemdan, Y. M. Essa, M. Shouman, A. El-Sayed, and A. N. Moustafa, "An efficient iot based smart water quality monitoring system," *Multimedia Tools and Applications*, vol. 82, pp. 28 827–28 851, 2023. [Online]. Available: https://link.springer.com/article/10.1007/s11042-023-14504-z

[14] J. Phuttharak and S. W. Loke, "An event-driven architectural model for integrating heterogeneous data and developing smart city applications," *Journal of Sensor and Actuator Networks*, vol. 12, no. 1, p. 12, 2023. [Online]. Available: https://doi.org/10.3390/jsan12010012

[15] LILYGO, "LILYGO T-Call V1.4 (ESP32 + SIM800)," https://lilygo.cc/products/t-call-v1-4, 2024, accessed: 2025-06-29.