BULLETIN of Cherkasy State Technological University

Journal homepage: https://bulletin-chstu.com.ua/en Vol. 31 No. 3. 2026

UDC 004.67:004.8
DOI: 10.62660/bcstu/3.2026.11

ISSN: 2306-4412
E-ISSN: 2708-6070

Architecture of hybrid Edge Lakehouse for environmental monitoring on Raspberry Pi

Ivan Borodii*
Postgraduate Student
Ternopil Ivan Puluj National Technical University
46001, 56 Ruska Str., Ternopil, Ukraine
https://orcid.org/0009-0005-4986-4429
Halyna Osukhivska
PhD in Technical Sciences, Associate Professor
Ternopil Ivan Puluj National Technical University
46001, 56 Ruska Str., Ternopil, Ukraine
https://orcid.org/0000-0003-0132-1378

Abstract. In this article, a hybrid Edge Lakehouse architecture for environmental monitoring is presented, which has been implemented and experimentally tested on a Raspberry Pi 5 edge device. The relevance of the research is driven by the need for a robust edge computing architecture that supports local collection, processing and transactional storage of sensor data, followed by synchronisation with a cloud-based analytics environment, within the limitations of hardware resources. The main objective of this work was to develop the Edge Lakehouse architecture as a local component of an autonomous monitoring system and to evaluate its performance. In this study, experimental modelling of the hardware-software system was conducted, along with measurements of execution time, memory usage and Central Processing Unit load, as well as an assessment of resilience to forced power shutdowns and temporary network outages. The proposed hybrid Edge Lakehouse architecture maintains data integrity and operational continuity under emergency and unstable conditions, as confirmed by experimental studies. In particular, the results showed that the average processing time for a single micro-batch of sensor data was 25.51 seconds, of which 20.04 seconds were spent on initialising the processing environment, 1.18 seconds on writing to the local table and 3.94 seconds on synchronising with the cloud environment. The system demonstrated correct recovery without loss of confirmed data and the 48-hour replication window allowed for autonomous synchronisation recovery without duplicate records. The obtained data also confirmed the architecture's suitability for long-term autonomous functioning. The proposed solutions have practical value as a reliable local layer for autonomous environmental monitoring systems

Keywords: internet of things; data acquisition; data handling; edge computing; artificial intelligence; sensor systems and applications; monitoring system

INTRODUCTION

Continuous monitoring of microclimate parameters, including atmospheric gas composition, temperature, humidity and pressure, is critical for maintaining environmental safety and effective condition assessment. In contemporary Internet of Things (IoT) systems, substantial volumes of sensor data are typically transmitted to remote cloud services. This approach increases dependence on network quality, introduces processing delays and impedes rapid response during periods of unstable connectivity. Furthermore, the use of simple local buffers, such as those based on JavaScript Object Notation (JSON) or Comma-Separated Values (CSV)

Article's History: Received: 04.05.2026; Revised: 26.08.2026; Accepted: 21.09.2026; Published: 07.10.2026

Suggested Citation:
Borodii, I., & Osukhivska, H., (2026). Architecture of hybrid Edge Lakehouse for environmental monitoring on Raspberry Pi. *Bulletin of Cherkasy State Technological University*, 31(3), 11-21. doi: 10.62660/bcstu/3.2026.11.

*Corresponding author (ivanborodii@tntu.edu.ua)

Copyright © The Author(s). This is an open access article distributed under the terms of the Creative Commons Attribution License 4.0 (https://creativecommons.org/licenses/by/4.0/)

formats, does not ensure data integrity following power failures. Consequently, the significance of this study is driven by the need to ensure reliable local collection, micro-batch processing and transactional storage of sensor data under limited hardware resources, unstable network connectivity and the necessity for subsequent synchronisation with a cloud analytical environment.

Recent studies on edge analytical systems show that the effectiveness of local monitoring depends on several factors. These include the placement of computations close to the data source, hardware limitations, energy usage, data storage and recovery mechanisms. In particular, P. Cruz *et al.* (2022) examined practical challenges of deploying Multi-access Edge Computing. They found that edge infrastructures are challenging to implement due to operational, structural and network issues. However, they mainly focused on deployment and communication, not on reliable data storage and recovery in a resource-limited edge node. H. Hua *et al.* (2023) examined Edge Computing from a Machine Learning viewpoint. They concluded that local analytical methods need to be adapted to hardware constraints, latency needs and executing tasks across different locations. In addition, their research paid limited attention to persistent local storage and data integrity when power or network failures occur. O. Jouini *et al.* (2024) organised Machine Learning techniques, frameworks, applications and research challenges for edge environments. They identified resource limitations as a key barrier to effective edge intelligence. However, the problem of creating a fault-tolerant local data layer for autonomous monitoring systems was not addressed in their study. S. Alam *et al.* (2024) reviewed both available commercial and experimental deep learning accelerators for edge and emerging computing, classifying them by architecture, which includes dataflow, neuromorphic and processing-in-memory approaches. The authors found that effective edge intelligence relies heavily on hardware architecture, energy efficiency, model compression, data accuracy and supported programming frameworks. However, their work concentrated on acceleration hardware, not on the reliability of local sensor data storage, transactional persistence or recovery mechanisms in autonomous monitoring systems.

The data management side of this problem has been addressed in studies on Data Lake and Lakehouse architectures. S. Azzabi *et al.* (2024) reviewed Data Lake concepts, architectural models and implementation challenges. They concluded that flexible storage architectures are vital for managing different types of data. However, their work focused on general data platforms instead of resource-limited edge devices. A.A. Harby & F. Zulkernine (2025) compared Data Lake, Data Warehouse and Lakehouse approaches. They showed the benefits of Lakehouse systems for managing analytical data. Still, their experiments did not cover autonomous sensor systems, local recovery or deployment on compact computing devices. R.J. Atwal *et al.* (2024) introduced MotherDuck as a hybrid analytical environment supporting query execution both locally and in the cloud. However, this work did not address sensor data collection, transactional persistence or recovery after failures at the edge node level. I. Borodii & H. Osukhivska (2025) studied the use of Apache Spark and Lakehouse storage formats for data loading and storage. They confirmed that this approach is suitable for analytical systems. Nevertheless, their study did not explore integrating local sensor data collection, transactional storage, fault-tolerant recovery and controlled cloud synchronisation within a single monitoring framework based on Raspberry Pi 5. The paper aimed to develop and experimentally validate a hybrid Edge Lakehouse architecture for an environmental monitoring system using Raspberry Pi 5. The research included implementing local data collection from SCD41 and BME688 sensors, micro-batch processing with the Apache PySpark framework, transactional storage in Delta Lake format, incremental synchronisation with the MotherDuck cloud environment, performance evaluation of the primary pipeline stages, verification of system resilience to forced power shutdowns and temporary network loss and assessment of the suitability of Small Language Models (SLMs) as a variable component for local analysis of sensor micro-batches.

MATERIALS AND METHODS

The environmental monitoring system was based on a hybrid Edge Lakehouse architecture implemented using the Apache PySpark 4.0.0 framework. To ensure transactional integrity at the edge node level, the Delta Lake 4.0.0 data storage format was selected. Apache PySpark was chosen for its comprehensive support of the Delta Lake format, including a transactional log mechanism and reliable integration with Apache Spark data processing workflows. The processing pipeline was developed in Python version 3.11, enabling effective integration of Big Data analysis tools with local Artificial Intelligence (AI) capabilities. The hardware component of the study was based on a Raspberry Pi 5 microcomputer with 8 GB of Random Access Memory (RAM) and a Broadcom BCM2712 processor. The device was equipped with a 128 GB memory card. The software environment operated on the Linux kernel version 6.12.62. The selection of Raspberry Pi as the hardware platform was consistent with recent survey evidence indicating that Raspberry Pi remains one of the most widely used low-cost microcomputers for prototyping, automation, monitoring, data acquisition and embedded systems used for research purposes across multiple domains (Mathe *et al.*, 2024).

The system collected primary environmental data using SCD41 and BME688 sensors connected to a Raspberry Pi 5 through an Inter-Integrated Circuit (I2C) interface. Data acquisition followed a micro-batch

processing approach: readings of carbon dioxide (CO_2), pressure, temperature and humidity were taken every 5 minutes and temporarily stored in a JSON buffer file on the disk drive. Each sensor micro-batch comprised 10 consecutive measurements of environmental parameters. Once a batch of 10 records was accumulated, the records were loaded into the Delta Lake table and the JSON file was cleared. This micro-batch approach enabled the analysis of short sequences of interrelated measurements rather than isolated values. The execution time of the main processing stages was measured within the Python implementation using high-resolution timers placed at the boundaries of each operation. The PySpark initialisation time was measured as a cold start, in which a new SparkSession was created before writing a sensor micro-batch to the Delta Lake table. The measured initialisation time included start-up of the PySpark runtime environment, loading of the required Delta Lake configuration and creation of the processing session. Sensor polling, JSON buffer writing, Delta Lake persistence and MotherDuck synchronisation were measured as separate stages and were not included in the PySpark initialisation time. A hot start scenario based on reusing an already initialised SparkSession was not evaluated in this experiment and was considered a direction for further optimisation.

The software implementation was based on modular decomposition, with components responsible for sensor data acquisition, micro-batch processing, writing to Delta Lake tables, logging and subsequent intelligent analysis of generated batches. Continuous system operation was ensured by an operating system service that automatically launched the software suite during device boot. Data synchronisation with MotherDuck was implemented as a cron task on the Raspberry Pi 5. The synchronisation logic was defined as a Structured Query Language (SQL) script that selected local Delta Lake records and inserted or refreshed them in the corresponding MotherDuck cloud tables. The cron task was executed once per day. During each execution, records with timestamps from the last two days were selected. This 48-hour interval was used as a technical recovery window, not as a statistical observation period and was intended to cover short-term power interruptions or temporary household network outages. If connectivity was unavailable during one execution, locally committed records could still be synchronised during the next daily run. To avoid duplicate records, the synchronisation task refreshed the corresponding 48-hour data range in MotherDuck by overwriting records that already existed in this interval.

The Delta Lake tables 'bme' and 'scd' were structured to store both raw sensor measurement data and reference indicator values for subsequent data analysis. Specifically, the anomaly_batch_gt and health_risk_gt fields were incorporated as ground truth for evaluating the results of intelligent sensor data analysis. The reliability of the local transactional layer was evaluated by forcibly interrupting power during the active phase of writing to the Delta Lake table. After automatic system restart, state recovery was achieved without loss of committed data or damage to the file structure of local storage. The anomaly and health-risk statistics were calculated for a 30-day experimental observation period. During this period, only measurements successfully collected from the sensors and written to the corresponding local Delta Lake tables were included in the analysis. The anomaly_batch_gt and health_risk_gt indicators were generated after data collection according to the deterministic threshold rules subsequently specified for abnormal changes and potential health risks. Deterministic threshold rules were used as ground truth, reflecting formalised indoor air quality standards and physically interpretable threshold changes in sensor readings. This approach enabled evaluation of the ability of SLMs to accurately reproduce formalised rules under limited hardware resources, which was essential for verifying their practical suitability on edge devices. Therefore, the study assessed not only the general anomaly detection capability of SLMs but also the broader criterion of how precisely the models reproduced specified deterministic rules under hardware limitations.

The anomaly_batch_gt indicator was determined by identifying sudden changes in sensor readings within a single micro-batch of measurements. For each micro-batch, the maximum absolute changes in temperature, relative humidity, atmospheric pressure and gas sensor resistance were calculated. If any of these indicators exceeded the defined threshold, all records in the corresponding micro-batch were marked as abnormal (anomaly_batch_gt = true). Additionally, the health_risk_gt indicator was generated for each measurement based on whether safe threshold values for CO_2, temperature or relative humidity were exceeded. If any parameter fell outside the acceptable range, the corresponding record was assigned a potential health risk label (health_risk_gt = true). Threshold values were established according to widely accepted recommendations for indoor air quality. The specific threshold values used to identify abnormal changes and potential health risks are presented in Table 1.

Table 1. Threshold values for identifying abnormal changes and potential health risks

Parameter	Anomaly Condition	Anomaly Description
Threshold values for detecting abnormal changes		
ΔT (temperature)	> 1.5°C	Sudden change in temperature
ΔRH (humidity)	> 5%	Significant change in relative humidity

Continued Table 1.

Parameter	Anomaly Condition	Anomaly Description
Threshold values for detecting abnormal changes		
ΔP (pressure)	> 1 hPa	Unusual change in atmospheric pressure
Gas drop (gas sensor resistance)	> 5%	Sudden drop in gas sensor resistance
Threshold values for identifying potential health risks		
CO_2	> 1000 ppm	Deterioration of indoor air quality
Temperature	< 18 and > 30 °C	Potentially hazardous temperature conditions
Relative humidity	< 30 and > 70%	Excessively dry or excessively humid air

Source: developed by the authors

SLMs were used in this study for the intelligent analysis of sensor data. These models were deployed locally on an edge device using the Ollama platform (Ollama's documentation, n.d.). Deploying SLMs on the edge device allowed analysis to occur without transmitting data to external cloud services, thereby reducing network latency and enhancing the autonomy of the monitoring system. The AI component of the architecture comprised five compact models, each with up to approximately 4 billion parameters and representing different architectural types: TinyLLaMA, SmolLM2, Qwen2.5, DeepSeek-R1 and Phi-3 Mini. These models were chosen based on the limited computational resources of the peripheral device and their capacity for local execution. The main features of the SLMs are presented in Table 2.

Table 2. Features of the SLMs

Model	Number of parameters (billion)	Memory capacity (GB)
TinyLLaMA	1.1	0.638
SmolLM2	1.7	1.8
Qwen2.5	3	1.9
DeepSeek-R1	1.5	1.1
Phi-3 Mini	3.8	2.2

Source: developed by the authors

In this experiment, each generated micro-batch of sensor measurements was transmitted to the language model as a standardised text instruction (prompt) that included a description of sensor parameter values and formalised interpretation rules. To maintain evaluation accuracy, all models received the same prompt. The instruction was structured to address two primary tasks: identifying abnormal states within the batch and detecting potential health risks. It provided formalised rules for determining abnormal states in micro-batches and for assessing health risks based on threshold values for temperature, humidity and CO_2. Additionally, the instruction required a structured response that included binary classification and a confidence score. The prompt was constructed as a rule-driven sequence that transformed each input sensor micro-batch into a standardised analytical request for each SLM. Initially, the system identified the sensor type and selected the corresponding rule set, distinguishing between SCD41 and BME688 data. The prompt was subsequently assembled in a unified format comprising four components: the model role, the context of a single micro-batch, the records contained within that batch and two explicitly defined analytical tasks.

The first task addressed batch-level anomaly detection by evaluating whether the variation of key parameters within the micro-batch exceed predefined thresholds: specifically, temperature, humidity and CO_2 for SCD41 and temperature, humidity, pressure and gas resistance for BME688. The second task addressed record-level health risk detection by assessing whether individual measurements indicate unsafe environmental conditions, such as elevated CO_2 or abnormal temperature and humidity values. To ensure consistency of interpretation, the prompt also embedded decision rules specific to each sensor and required the model to return results in a strict JSON structure containing the anomaly flag, the batch health-risk flag, a brief analytical status, a concise health-risk summary and the confidence score. This design rendered the prompt generation process transparent, reproducible and directly comparable across different SLMs.

The evaluation of SLMs was based on an experimental approach focused on verifying the models' capacity to reproduce formalised classification rules within an Edge Computing environment. The analysis results for sensor batches processed by each language model were stored in distinct Delta Lake tables, enabling assessment of both model accuracy and computational performance on the edge device. The effectiveness of the language models was assessed by comparing their outputs to the ground truth labels anomaly_batch_gt and health_risk_gt, which were generated using threshold-based rules. Standard binary classification metrics, including precision, recall and F1-score, were used to evaluate model performance. Furthermore, the execution

time of each model was measured to determine its suitability for deployment on edge devices with constrained hardware resources. Beyond the language analysis component, the proposed architecture utilised a two-tier data processing model. The local edge node tier managed sensor data collection, micro-batch formation, transactional record writing to a Delta Lake table and ensured data consistency in case of system failure. The cloud-based MotherDuck layer provided long-term storage and supported advanced analytics, with synchronisation between local and remote layers achieved through a scheduled SQL task. This design separated the robust local data collection and storage layer from the remote analytics layer. The hybrid Edge Lakehouse architecture of the environmental monitoring system is illustrated in Figure 1.

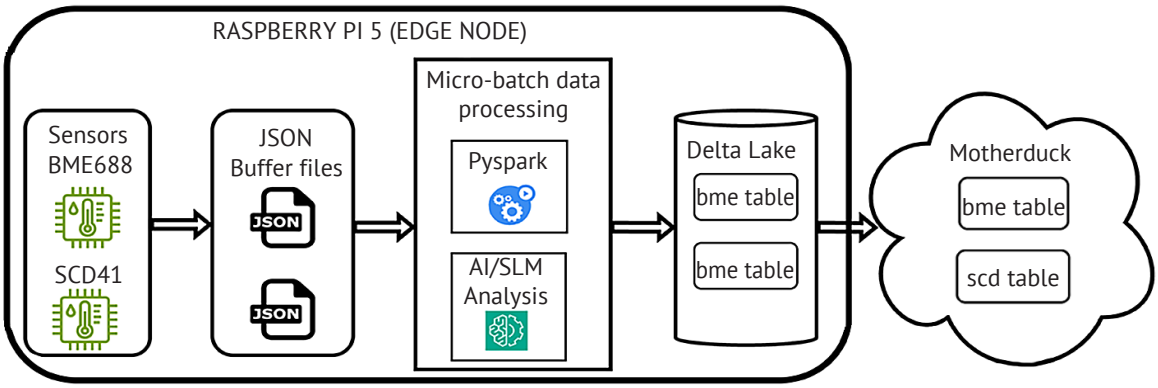


Figure 1. Hybrid Edge Lakehouse architecture for an environmental monitoring system
Source: developed by the authors

RESULTS

The proposed hybrid Edge Lakehouse architecture was experimentally validated under real operating conditions using a Raspberry Pi 5 node. The results were analysed across three complementary dimensions: the structure of the collected experimental data, the performance of the local processing pipeline and the reliability of local storage, synchronisation and auxiliary analysis using language models. This framework enabled assessment of both the temporal properties of the system and its practical suitability as a reliable local layer within an autonomous monitoring architecture. Two sensor datasets with distinct event structures were generated during the experiment. The data in Table 3 demonstrate that the two streams differ in total observation volume as well as in the frequency of abnormal micro-batches and potentially risky records. These findings confirmed that separate processing of sensor streams was methodologically justified, as each dataset had a distinct distribution of abnormal micro-batches and potentially risky records.

Table 3. Statistics on the experimental collection of sensor data

Table name	Parameter	Value
bme	Total number of sensor measurements	6,308
	Number of micro-batches generated	829
	Number of micro-batches with anomalies (anomaly_batch_gt = true)	51
	Number of records with health risks (health_risk_gt = true)	5,141
scd	Total number of sensor measurements	5,391
	Number of micro-batches generated	563
	Number of micro-batches with anomalies (anomaly_batch_gt = true)	30
	Number of records with health risks (health_risk_gt = true)	808

Source: compiled by the authors

The results in Table 4 demonstrate that the temporal profile of the local pipeline was primarily influenced by the initialisation stage of the processing environment, rather than by Delta Lake write or synchronisation operations. Since PySpark initialisation dominated the processing cycle, the main opportunity for optimisation was reducing initialisation overhead. Methodologically, the obtained data confirmed the technical feasibility of using Apache PySpark and Delta Lake on a constrained edge node, while identifying the lifecycle of the local processing environment as the principal performance bottleneck.

Table 4. Edge node performance metrics

Processing stage	Average execution time, s	RAM usage, MB	CPU load, %
Data collection and writing to a JSON buffer	0.35	36.99	22.14
Initialising a PySpark session	20.04	63.90	38.84
Writing micro-batch to a Delta Lake table	1.18	36.95	99.39
Synchronising with MotherDuck	3.94	244.75	33
Total for 1 data micro-batch	25.51 (sum)	95.65 (average)	48.34 (average)

Source: developed by the authors

The measurements summarised in Table 4 showed that the principal temporal bottleneck was associated with initialisation of the processing environment on the edge device rather than with the Delta Lake write operation itself. Although the highest CPU load was recorded at the stage of writing data to the Delta Lake table, where CPU utilisation reached 99.39%, the stage lasted only 1.18 seconds on average. In the conducted experiment, this short interval of increased processor utilisation was not associated with increased memory consumption, loss of committed records or corruption of the local Delta Lake table. Therefore, the observed CPU load indicated a brief resource peak of the transactional write stage rather than sustained processor saturation during the entire processing cycle. The comparative evaluation in Table 5 demonstrates that the performance of SLMs was dependent on the specific task, which limited their suitability as universal components for analysis in real time in the tested configuration. The results indicated that specific models performed better in certain tasks while showing lower effectiveness in others. Consequently, within the proposed architecture, the language component was interpreted as an auxiliary analytical layer rather than the primary mechanism for operational local decision support.

Table 5. Evaluating the suitability of SLMs as an AI component of the architecture

Indicator \ Model	TinyLLaMA	SmolLM2	Qwen2.5	DeepSeek-R1	Phi-3 Mini
Precision (anomaly)	0.30	0.43	0.43	0.15	0.43
Recall (anomaly)	0.24	0.84	1	0.04	1
F1-score (anomaly)	0.27	0.57	0.60	0.07	0.60
Precision (health risk)	0.43	0.31	0.36	0.40	0.50
Recall (health risk)	0.73	0.13	0.55	0.42	0.11
F1-score (health risk)	0.54	0.19	0.44	0.41	0.18
Average execution time, s	191.94	309.76	319.16	190.06	748.89

Source: developed by the authors

The results presented in Table 5 indicate that the most notable deviation was observed for Phi-3 Mini. Although this model had the largest number of parameters among the tested SLMs and the longest average execution time of 748.89 seconds, its F1 score for health risk classification reached only 0.18. This value was mainly determined by the low recall of 0.11, whereas precision was 0.50. Thus, the main source of error was not the frequent assignment of false positive health risk labels, but the failure to identify positive health risk cases. Additional testing of the synchronisation mechanism with MotherDuck demonstrated that a 48-hour replication window enabled self-recovery of the dataset following simulated network connection loss, without introducing duplicate records. For anomaly detection, Qwen2.5 and Phi-3 Mini achieved the highest F1-score (0.60), primarily because both models reached perfect recall (1.00), indicating that all true anomalies were successfully detected. However, this performance was accompanied by relatively low precision (0.43), meaning that approximately 57% of detected anomalies were false positives. While such behaviour may be acceptable in safety-critical monitoring systems where missing an anomaly is more costly than issuing false alarms, it would substantially increase the number of unnecessary alerts during routine operation. In contrast, DeepSeek-R1 demonstrated extremely poor anomaly detection performance (F1 = 0.07; recall = 0.04), failing to identify the overwhelming majority of true anomalies.

A different ranking emerged for health risk classification. TinyLLaMA achieved the highest overall performance (F1 = 0.54) by maintaining the most balanced combination of precision (0.43) and recall (0.73). Conversely, the larger SmolLM2 (F1 = 0.19) and Phi-3 Mini (F1 = 0.18) exhibited critically low recall values (0.13 and 0.11, respectively), indicating that they failed to recognise most genuine health risk cases. These findings suggest that increasing model size does not necessarily improve task-specific reasoning under edge-computing constraints and may even reduce classification effectiveness after quantisation. The results also reveal several notable performance paradoxes. Despite its substantially smaller parameter count,

TinyLLaMA clearly outperformed the considerably larger Phi-3 Mini in health risk classification while completing inference almost four times faster (191.94 s versus 748.89 s). This indicates that model scaling alone is not a reliable predictor of downstream performance on resource-constrained hardware and that compact architectures may provide superior efficiency-quality trade-offs for specialised classification tasks. Similarly, DeepSeek-R1 produced the shortest inference time (190.06 s), despite belonging to a reasoning-oriented model family that would normally be expected to incur longer processing times. Its combination of exceptionally high speed and extremely poor anomaly detection performance suggests that the model may have been affected by implementation-related constraints, such as prompt formatting, token limits, or inference truncation, resulting in incomplete reasoning or invalid classification outputs. This possibility should be considered when interpreting the observed results.

Inference latency is also a critical consideration for deployment on Raspberry Pi 5. Phi-3 Mini required 748.89 s (approximately 12.5 minutes) to process a single dataset, making it unsuitable for near real-time monitoring scenarios. Given the previously reported processor utilisation approaching full capacity during data acquisition, such prolonged execution would likely increase the risk of thermal throttling and reduce system responsiveness. In contrast, Qwen2.5 provides the most favourable balance between predictive performance and computational efficiency, achieving the highest anomaly detection accuracy (F1 = 0.60), competitive health risk classification performance (F1 = 0.44), and a substantially shorter inference time of 319.16 s. Overall, these findings indicate that Qwen2.5 represents the most suitable single-model solution for the proposed edge lakehouse architecture, whereas TinyLLaMA may be preferable for health risk assessment if a task-specific multi-model deployment strategy is considered. These findings indicated that the local layer of the architecture operated as a robust transactional component of an autonomous monitoring system, rather than functioning solely as a temporary data buffer.

DISCUSSION

The findings confirmed that an autonomous monitoring system requires not only local computation, but also data integrity, recovery after failures and controlled synchronisation with a cloud analytical environment. The obtained results support P. Cruz *et al.* (2022), who emphasised that Multi access Edge Computing depends on operational, infrastructure and network conditions. Similar limitations were observed because the Raspberry Pi 5 node operated with limited resources and without guaranteed continuous connectivity. The conclusions of H. Hua *et al.* (2023) and O. Jouini *et al.* (2024) are also confirmed, since the local pipeline and analytical components had to be adapted to hardware limitations. However, unlike these studies, this research also focused on transactional storage and recovery of sensor data. S. Alam *et al.* (2024) showed that efficient intelligence at the edge depends on hardware architecture, energy usage, numerical precision, model compression and programming support. The present experiment confirmed this view, because the tested configuration was sufficient for local data collection and storage, but not for efficient real time execution of SLMs.

The findings correspond to P. Grzesik & D. Mrozek (2024), S.A. Cajas Ordóñez *et al.* (2025) and K.S. Velaga *et al.* (2025), who considered the combination of Machine Learning and Edge Computing important for systems with low latency and limited resources. These studies support this research direction, but showed that local intelligence alone is not sufficient for autonomous monitoring. A reliable local data layer is required before analytical models can be applied. The proposed architecture is related to monitoring systems discussed by I. Ficili *et al.* (2025), E. Illueca Fernández *et al.* (2024), M. Al-Okby *et al.* (2025). These authors showed that modern monitoring solutions combine acquisition, communication, storage, analytics, visualisation, alerts and decision support. A. Garcia *et al.* (2025) also confirmed the relevance of integrated air quality monitoring architectures that combine sensing technologies with intelligent analytical components. The present study supports this direction, but differs because it examined local storage reliability rather than prediction or visualisation. N.A. Kulykovska *et al.* (2024) presented an automated IoT system for indoor air quality monitoring, which confirmed the practical relevance of sensor systems for continuous environmental observation. However, their work focused mainly on the organisation of monitoring functions, whereas the present study examined the reliability of local transactional storage, recovery after failures and synchronisation with a cloud analytical layer.

S. Abimannan *et al.* (2023), H.Q. Flayyih *et al.* (2024) and R. Sserujongi *et al.* (2025) demonstrated intelligent and distributed approaches for air quality and environmental analysis. T. Miller *et al.* (2025) reviewed the use of federated learning in environmental monitoring and showed that distributed intelligent methods can improve privacy, scalability and collaborative model training. The present architecture complemented these studies by showing how sensor data can be preserved and synchronised before advanced analysis. The results also support T.V. Quang *et al.* (2025), who highlighted local processing for privacy preserving indoor air quality platforms. In this study, local processing reduced dependence on external services, while transactional storage maintained data integrity during failures. A.G. Tkachuk *et al.* (2023) confirmed the usefulness of Edge Computing in environmental science and the present research extended this direction by combining local environmental measurements with transactional

storage and cloud synchronisation. The performance results are consistent with S.M. Raza *et al.* (2021), who showed that CPU utilisation, memory consumption and system overhead influence local services on limited IoT gateways. In this study, Apache PySpark initialisation required 20.04 seconds, whereas writing one micro-batch to Delta Lake required only 1.18 seconds. Thus, the main bottleneck was the processing environment lifecycle.

The findings complement V. Bonar *et al.* (2025), N. Kuchuk *et al.* (2024) and A.E. Edje *et al.* (2023), who addressed local analytics, adaptive sampling, compression, operational transactions, heterogeneous streams, redundancy, outliers and resource balancing in infrastructures combining IoT, edge and cloud layers. The present study agrees with these conclusions, but showed that transmission optimisation should be supplemented by transactional persistence and recovery. L. Palazzetti *et al.* (2025) and F.A. Delgado-Rajó *et al.* (2025) considered low cost monitoring systems using Raspberry Pi or similar compact computers. This study extended their conclusions by showing that Raspberry Pi 5 can function as a local transactional storage component. The data storage results can also be interpreted in the context of Lakehouse research. In particular, M. Armbrust *et al.* (2021) introduced the Lakehouse concept as a model combining open data formats with analytical capabilities, while A.A. Harby & F. Zulkernine (2025) demonstrated the advantages of Delta Lakehouse for ingestion, metadata management, compaction and deduplication. The present study confirmed that the architectural principle can be adapted to Raspberry Pi 5. R.J. Atwal *et al.* (2024) presented MotherDuck as a hybrid analytical environment combining local and cloud execution, which is supported here by synchronisation with MotherDuck and recovery within a 48 hour replication window. I. Borodii & H. Osukhivska (2025) previously confirmed the applicability of Apache Spark and Lakehouse formats for analytical data loading and storage, while the present research transfers this approach to an autonomous monitoring system.

The language model results should be interpreted in relation to M. Jin *et al.* (2024), F. Corradini *et al.* (2025) and E.J. Husom *et al.* (2025). M. Jin *et al.* (2024) showed that language models can be applied to time series analysis. However, the present results demonstrated that such feasibility did not automatically imply suitability for real time processing on Raspberry Pi 5. F. Corradini *et al.* (2025) emphasised the need to balance quality, latency and memory requirements in SLMs, while E.J. Husom *et al.* (2025) showed trade-offs between energy efficiency, inference latency and output accuracy for quantised models at the edge. The experiment confirmed these conclusions, since model execution time ranged from 190.06 to 748.89 seconds and classification quality varied across tasks. The low F1 score of Phi-3 Mini for health risk classification was mainly associated with its recall value of only 0.11, despite a precision value of 0.50. This indicated that the model assigned the positive health risk label less frequently than required by the deterministic ground truth rules. As a result, a larger number of parameters and longer execution time did not necessarily correspond to better reproduction of deterministic threshold rules under the tested Raspberry Pi 5 configuration. Overall, previous studies confirmed the importance of local computation, hybrid synchronisation, intelligent analysis and modern storage architectures. The present research differs by combining these directions with transactional persistence, recovery after forced power interruption and controlled cloud synchronisation on a single Raspberry Pi 5 device.

CONCLUSIONS

This study proposed and experimentally validated a hybrid Edge Lakehouse architecture for environmental monitoring, using the Raspberry Pi 5. The system integrated local sensor data collection, micro-batch processing with Apache PySpark, transactional storage in Delta Lake format and incremental synchronisation with MotherDuck. Experimental results confirmed that the proposed approach supports both the operability of the local pipeline on the edge node and its effectiveness as an independent transactional layer within an autonomous monitoring system. The average processing time for a single sensor micro-batch was 25.51 seconds. The majority of this duration was attributed to PySpark session initialisation, which required 20.04 seconds and constituted 78.6% of the total processing time. In contrast, writing the micro-batch to Delta Lake and synchronising with MotherDuck were substantially faster, requiring 1.18 seconds and 3.94 seconds, respectively. These findings indicated that the primary time-consuming factor in the proposed architecture is the initialisation of the processing environment, rather than the data operations.

Evaluation of SLMs as an analytical component demonstrated limited suitability for local micro-batch analysis. For anomaly detection, F1-scores ranged from 0.07 to 0.60, while for health risk assessment, scores ranged from 0.18 to 0.54. Additionally, the average execution time per micro-batch ranged from 190.06 to 748.89 seconds, which is substantially longer than the base data processing pipeline. Therefore, under the current Raspberry Pi 5 configuration, SLMs are more appropriate as auxiliary interpretation modules rather than as primary tools for real time analysis. An experimental study has shown that the scientific contribution of this work is that a peripheral device can integrate local transactional sensor data storage, micro-batch processing, fault-tolerant recovery, controlled synchronisation with a cloud analytical environment and an analysis component using language models. Future research should focus on reducing the overhead

associated with launching the processing environment, particularly by optimising the Spark session lifecycle or adopting lighter-weight local processing mechanisms for specific operations. Further investigation into hardware acceleration for local language analysis on edge nodes, as well as testing the architecture under extended autonomous operation and complex network instability, is warranted

ACKNOWLEDGEMENTS

None.

FUNDING

None.

CONFLICT OF INTEREST

None.

REFERENCES

[1] Abimannan, S., El-Alfy, E.-S.M., Hussain, S., Chang, Y.-S., Shukla, S., Satheesh, D., & Breslin, J.G. (2023). Towards federated learning and multi-access edge computing for air quality monitoring: Literature review and assessment. *Sustainability*, 15(18), article number 13951. doi: 10.3390/su151813951.

[2] Alam, S., Yakopcic, C., Wu, Q., Barnell, M., Khan, S., & Taha, T.M. (2024). Survey of deep learning accelerators for edge and emerging computing. *Electronics*, 13, article number 2988. doi: 10.3390/electronics13152988.

[3] Al-Okby, M.F.R., Junginger, S., Roddelkopf, T., & Thurow, K. (2025). RTIMS: Real-time indoor monitoring systems: A comprehensive review. *Applied Sciences*, 15(24), article number 13217. doi: 10.3390/app152413217.

[4] Armbrust, M., Ghodsi, A., Xin, R., & Zaharia, M. (2021). Lakehouse: A new generation of open platforms that unify data warehousing and advanced analytics. In *Proceedings of the 11th conference on innovative data systems research*. Moultonborough: CIDR.

[5] Atwal, R.J., *et al.* (2024). MotherDuck: DuckDB in the cloud and in the client. In *Proceedings of the 14th conference on innovative data systems research*. Chaminade: CIDR.

[6] Azzabi, S., Alfughi, Z., & Ouda, A. (2024). Data lakes: A survey of concepts and architectures. *Computers*, 13(7), article number 183. doi: 10.3390/computers13070183.

[7] Bonar, V., Hotovych, V., & Matiichuk, L. (2025). Edge computing and data optimisation for remote diagnostics of vehicles. *Scientific Journal of the Ternopil National Technical University*, 4(120), 39-45. doi: 10.33108/visnyk_tntu2025.04.039.

[8] Borodii, I., & Osukhivska, H. (2025). Research on the efficiency of data loading and storage in data lakehouse architectures for the formation of analytical data systems. *Information Technology: Computer Science, Software Engineering and Cyber Security*, 4, 28-36. doi: 10.32782/IT/2025-4-4.

[9] Cajas Ordóñez, S.A., Samanta, J., Suárez-Cetrulo, A.L., & Carbajo, R.S. (2025). Intelligent edge computing and machine learning: A survey of optimisation and applications. *Future Internet*, 17(9), article number 417. doi: 10.3390/fi17090417.

[10] Corradini, F., Leonesi, M., & Piangerelli, M. (2025). State of the art and future directions of small language models: A systematic review. *Big Data and Cognitive Computing*, 9(7), article number 189. doi: 10.3390/bdcc9070189.

[11] Cruz, P., Achir, N., & Carneiro, V.A. (2022). On the edge of the deployment: A survey on multi-access edge computing. *ACM Computing Surveys*, 55(5), article number 99. doi: 10.1145/3529758.

[12] Delgado-Rajó, F.A., & Travieso-González, C.M. (2025). Flexible hybrid edge computing IoT architecture for low-cost bird songs detection system. *Ecological Informatics*, 90, article number 103231. doi: 10.1016/j.ecoinf.2025.103231.

[13] Edje, A.E., Abd Latiff, M.S., & Chan, W.H. (2023). IoT data analytic algorithms on edge-cloud infrastructure: A review. *Digital Communications and Networks*, 9, 1486-1515. doi: 10.1016/j.dcan.2023.10.002.

[14] Ficili, I., Giacobbe, M., Tricomi, G., & Puliafito, A. (2025). From sensors to data intelligence: Leveraging IoT, cloud, and edge computing with AI. *Sensors*, 25(6), article number 1763. doi: 10.3390/s25061763.

[15] Flayyih, H.Q., Waleed, J., & Ibrahim, A.M. (2024). IoT-based AI methods for indoor air quality monitoring systems: A systematic review. *International Journal of Computing and Digital Systems*, 16(1), 813-826. doi: 10.12785/ijcds/160159.

[16] Garcia, A., Saez, Y., Harris, I., Huang, X., & Collado, E. (2025). Advancements in air quality monitoring: A systematic review of IoT-based air quality monitoring and AI technologies. *Artificial Intelligence Review*, 58, article number 275. doi: 10.1007/s10462-025-11277-9.

[17] Grzesik, P., & Mrozek, D. (2024). Combining machine learning and edge computing: Opportunities, challenges, platforms, frameworks, and use cases. *Electronics*, 13(3), article number 640. doi: 10.3390/electronics13030640.

[18] Harby, A.A., & Zulkernine, F. (2025). Data Lakehouse: A survey and experimental study. *Information Systems*, 127, article number 102460. doi: 10.1016/j.is.2024.102460.

[19] Hua, H., Li, Y., Wang, T., Dong, N., Li, W., & Cao, J. (2023). Edge computing with artificial intelligence: A machine learning perspective. *ACM Computing Surveys*, 55(9), article number 184. doi: 10.1145/3555802.

[20] Husom, E.J., Goknil, A., Astekin, M., Shar, L.K., Kåsen, A., Sen, S., Mithassel, B.A., & Soylu, A. (2025). Sustainable LLM inference for edge AI: Evaluating quantized LLMs for energy efficiency, output accuracy, and inference latency. *ACM Transactions on Internet of Things*, 6(4), article number 28. doi: 10.1145/3767742.
[21] Illueca Fernández, E., Jara Valera, A.J., & Fernández Breis, J.T. (2024). Embedded machine learning of IoT streams to promote early detection of unsafe environments. *Internet of Things*, 25, article number 101128. doi: 10.1016/j.iot.2024.101128.
[22] Jin, M., *et al*. (2024). Time-LLM: Time series forecasting by reprogramming large language models. *ArXiv*. doi: 10.48550/arXiv.2310.01728.
[23] Jouini, O., Sethom, K., Namoun, A., Aljohani, N., Alanazi, M.H., & Alanazi, M.N. (2024). A survey of machine learning in edge computing: Techniques, frameworks, applications, issues, and research directions. *Technologies*, 12(6), article number 81. doi: 10.3390/technologies12060081.
[24] Kuchuk, N., Kashkevich, S., Radchenko, V., Andrusenko, Y., & Kuchuk, H. (2024). Applying edge computing in the execution IoT operative transactions. *Advanced Information Systems*, 8(4), 49-59. doi: 10.20998/2522-9052.2024.4.07.
[25] Kulykovska, N.A., Timenko, A.V., Hrushko, S.S., & Shkarupylo, V.V. (2024). Automated Internet of Things system for monitoring indoor air quality. In *Proceedings of the 4th edge computing workshop* (pp. 97-104). Zhytomyr: CEUR.
[26] Mathe, S.E., Kondaveeti, H.K., Vappangi, S., Vanambathina, S.D., & Kumaravelu, N.K. (2024). A comprehensive review on applications of Raspberry Pi. *Computer Science Review*, 52, article number 100636. doi: 10.1016/j.cosrev.2024.100636.
[27] Miller, T., Durlik, I., Kostecka, E., & Puszkarek, A. (2025). Federated learning for environmental monitoring: A review of applications, challenges, and future directions. *Applied Sciences*, 15, article number 12685. doi: 10.3390/app152312685.
[28] Ollama's documentation. (n.d.). Retrieved from https://docs.ollama.com/.
[29] Palazzetti, L., Giannetti, D., Verolino, A., Grasso, D.A., Pinotti, C.M., & Betti Sorbelli, F. (2025). AntPi: A raspberry Pi based edge-cloud system for real-time ant species detection using YOLO. *Ecological Informatics*, 91, article number 103383. doi: 10.1016/j.ecoinf.2025.103383.
[30] Quang, T.V., Doan, D.T., Ngarambe, J., Ghaffarianhoseini, A., Ghaffarianhoseini, A., & Zhang, T. (2025). AI management platform for privacy-preserving indoor air quality control: Review and future directions. *Journal of Building Engineering*, 100, article number 111712. doi: 10.1016/j.jobe.2024.111712.
[31] Raza, S.M., Jeong, J., Kim, M., Kang, B., & Choo, H. (2021). Empirical performance and energy consumption evaluation of container solutions on resource constrained IoT gateways. *Sensors*, 21(4), article number 1378. doi: 10.3390/s21041378.
[32] Sserujongi, R., Ogenrwot, D., Niwamanya, N., Nsimbe, N., Bbaale, M., Ssempala, B., Mutabazi, N., Wabinyai, R.F., Okure, D., & Bainomugisha, E. (2025). Design and evaluation of a scalable data pipeline for AI-driven air quality monitoring in low-resource settings. *ArXiv*. doi: 10.48550/arXiv.2508.14451.
[33] Tkachuk, A.G., Hrynevych, M.S., Vakaliuk, T.A., Chernysh, O.A., & Medvediev, M.G. (2023). Edge computing in environmental science: Automated intelligent robotic platform for water quality assessment. *Journal of Edge Computing*, 2(2), 163-174. doi: 10.55056/jec.633.
[34] Velaga, K.S., Guo, Y., & Yu, W. (2025). Edge AI for smart cities: Foundations, challenges, and opportunities. *Smart Cities*, 8(6), article number 211. doi: 10.3390/smartcities8060211.

Архітектура гібридного Edge Lakehouse для екологічного моніторингу на Raspberry Pi

Іван Бородій

Аспірант
Тернопільський національний технічний університет імені Івана Пулюя
46001, вул. Руська, 56, м. Тернопіль, Україна
https://orcid.org/0009-0005-4986-4429
Галина Осухівська

Кандидат технічних наук, доцент
Тернопільський національний технічний університет імені Івана Пулюя
46001, вул. Руська, 56, м. Тернопіль, Україна
https://orcid.org/0000-0003-0132-1378

Анотація. У цій статті представлено гібридну архітектуру Edge Lakehouse для моніторингу навколишнього середовища, реалізовану та експериментально перевірену на периферійному пристрої Raspberry Pi 5. Актуальність дослідження зумовлена потребою в надійній архітектурі периферійного обчислення, яка забезпечує локальний збір, обробку та транзакційне збереження сенсорних даних із подальшою синхронізацією з хмарним аналітичним середовищем в умовах обмежених апаратних ресурсів. Основною метою роботи була розробка архітектури Edge Lakehouse як локального компонента автономної системи моніторингу та оцінка її продуктивності. В цьому дослідженні проведено експериментальне моделювання апаратно-програмної системи, вимірювання часу виконання, використання пам'яті та навантаження процесора, а також оцінку стійкості до примусового відключення живлення та тимчасових перебоїв у роботі мережі. Запропонована гібридна Edge Lakehouse архітектура підтримує цілісність даних та безперебійність роботи в екстрених та нестабільних умовах, що підтверджено експериментальними дослідженнями. Зокрема, результати показали, що середній цикл обробки одного мікропакета даних сенсора становив 25,51 секунди, з яких 20,04 секунди припадало на ініціалізацію середовища обробки, 1,18 секунди на запис у локальну таблицю та 3,94 секунди на синхронізацію з хмарним середовищем. Система продемонструвала коректне відновлення без втрати підтверджених даних, а 48-годинне вікно реплікації дозволило самостійно відновити синхронізацію без дублювання записів. Отримані дані також підтвердили придатність архітектури до тривалого автономного функціонування. Запропоновані рішення мають практичну цінність як надійний локальний рівень для автономних систем моніторингу навколишнього середовища

Ключові слова: пікове співвідношення сигналу до шуму; «сіль і перець»; гаусівський шум; мультиплікативний шум; «перетин»; Matlab