Implementation of a Meshtastic Gateway System With a Local Database for Iot Applications

Daniel Menićanin[1], Jelena Radanović[2], Dražen Marinković[3]
[1]*Pan-European University Apeiron, Faculty of Information Technology, Banja Luka, Bosnia and Herzegovina, danijel.menicanin@gmail.com, 0009-0001-6311-4043*
[2]*Pan-European University Apeiron, Faculty of Information Technology, Banja Luka, Bosnia and Herzegovina, dev.radanovic@gmail.com, 0009-0001-5135-7662*
[3]*Pan-European University Apeiron, Faculty of Information Technology, Banja Luka, Bosnia and Herzegovina drazen.m.marinkovic@apeiron-edu.eu, 0009-0006-8001-2168*

Original scientific paper
https://doi.org/10.7251/JIT2502085M UDC: 004.76+004.42

Abstract: This paper presents a system for reliable collection, filtering, and processing of data from a LoRa Meshtastic decentralized network, developed for use in remote areas with weak or no mobile network coverage. The core idea stems from the need to enable efficient exchange of small data packets at intervals, without relying on expensive and often unavailable internet infrastructure. The key innovation lies in the implementation of the Meshtastic gateway concept, which provides internet access via HTTP requests, while the developed database model ensures continuity and reliability of data transmission. Data arriving in the network as unstructured messages are extracted using regular expressions, transformed into JSON format, and sent to the visualization platform Grafana, while simultaneously being stored in a local database for later queries, research, and analysis. The system's reliability is further enhanced by introducing a two-layer acknowledgment mechanism (Meshtastic ACK_APP and remote API application-level ACK), as well as an offline mode that logs undelivered messages and their causes through flags and error records. This ensures resilience to data loss and enables seamless operation continuation after connection interruptions.

Keywords: LoRa, Meshtastic, IoT, Database

Introduction

The Internet of Things (IoT) has in recent years become a key technology in areas such as industrial automation, smart cities, agriculture, and environmental monitoring. The fundamental idea of IoT systems is the collection and processing of data through a network of sensors and their transmission to remote platforms where analysis and visualization take place.

However, in practice, a major problem arises, most of these systems depend on a stable and continuous internet connection, which is not always achievable in remote or infrastructure-limited areas.

The challenge is particularly evident in applications that require only periodic transmission of small amounts of data. In such cases, maintaining a permanent mobile or fixed internet connection represents an unnecessary cost and reduces the overall cost-effectiveness of the system. This creates a gap between real-world needs and existing commercial solutions. As a response to this challenge, a system was developed that combines the LoRa [1] Meshtastic network and a multiplatform application for data collection and processing. Unlike LoRaWAN solutions, which require a centralized gateway and a connection to a provider's network, Meshtastic uses a mesh topology. This means that each node in the network functions both as an endpoint and as a repeater, achieving resilience to interruptions and eliminating dependence on a single central point. In this way, the system can operate completely offline, without the need for infrastructure support.

The role of the developed application is to receive data from the Meshtastic network, perform filtering, and transform it into a structured JSON format. Since a large number of non-system and auxiliary messages appear in the network, regular expressions (regex) are used to extract only relevant data from predefined senders. The filtered data is then written to a local database and simultaneously sent to the Grafana [2] visualization platform via HTTP API calls. This ensures dual security, data is available in real time and permanently stored locally for further analysis. Figure 1 shows the Grafana interface used for data visualization.

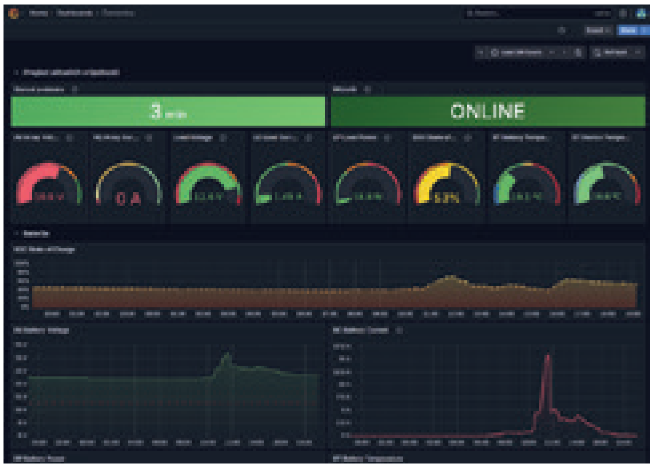


Figure 1. *Visualization of collected data using the Grafana dashboard*

The key contribution of the system lies in the implementation of reliability and redundancy mechanisms. The Meshtastic network provides delivery confirmation within the protocol itself (ACK_APP), while the application further utilizes HTTP responses from remote APIs. In case of internet disconnection, all data is automatically stored in the local database with a clearly marked status "undelivered" and an associated error reason. When the connection is restored, the system automatically performs synchronization and retransmission, eliminating the possibility of data loss.

A special quality of the developed solution lies in the fact that the application is cross-platform, developed in the Dart/Flutter environment, and runs on Windows, Linux, and Android operating systems. This provides high flexibility of use from server operation in office environments, through field monitoring on laptops, to mobile deployment on Android devices.

Based on the problem identified and the proposed solution, this work seeks to answer the following research questions:

1. How can IoT systems ensure reliable and cost-effective data transmission in environments with limited or no internet infrastructure?
2. To what extent can a decentralized LoRa-based mesh network (Meshtastic) serve as a viable alternative to traditional LoRaWAN or cellular solutions?
3. How effective is regex-based filtering in extracting relevant sensor data from a noisy, unstructured message stream?
4. What level of reliability and redundancy can be achieved through hybrid local and remote data handling, especially during intermittent connectivity?
5. How does a cross-platform gateway application contribute to the practical deployment and scalability of offline-capable IoT systems?

Methods and Materials

The architecture of the developed system is designed to integrate all the key layers required for reliable data collection, processing, and distribution in environments where internet infrastructure is unavailable or not economically viable.

At the base level of the system are sensors connected to ESP32 microcontrollers. These devices are responsible for collecting measurement values, formatting them into key–value pairs, and preparing them for transmission. The messages then enter the LoRa Meshtastic network, which represents the communication layer of the system. Thanks to its mesh topology, the network is self-sustaining and allows messages to be dynamically routed through multiple nodes until they reach their destination.

When messages reach the node connected to the application, the data processing layer begins. The application retrieves all incoming messages from the network via serial communication and filters them using regular expressions. After filtering, the data is transformed into JSON format, making it standardized and ready for integration with remote services or storage in the local database.

In the third layer, data distribution and storage take place. The data is sent to the remote Grafana service via HTTP API calls, while simultaneously being written into the local database. The database does not serve merely as passive storage, it actively participates in system functionality by enabling analytical queries and reconstruction of historical events. A special feature is the flag-based record marking mechanism: if data is not successfully delivered to the API, it remains in the database marked as "undelivered" with the corresponding error reason recorded, ensuring complete transparency. Figure 2 shows the data flow diagram of the system architecture.

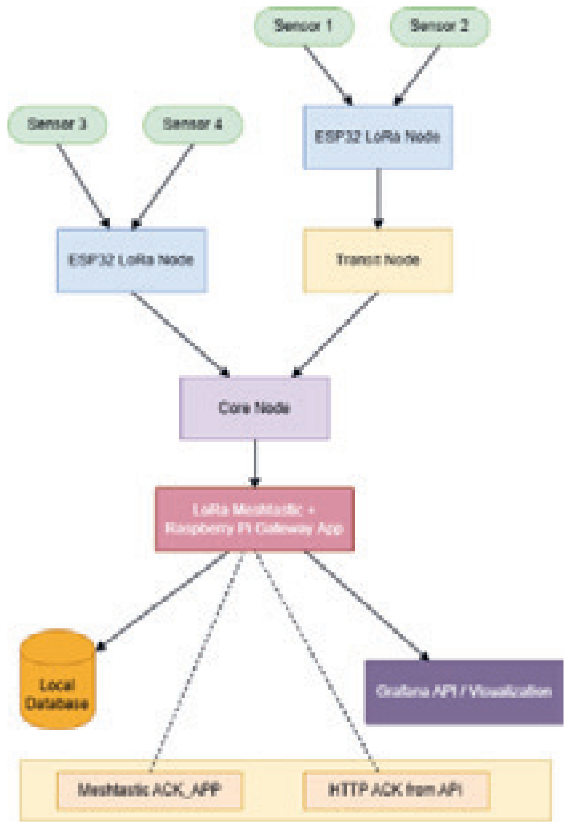


Figure 2. *LoRa Meshtastic Gateway System – Data Flow Diagram and Acknowledgment Structure*

One of the key aspects of the system architecture is the implementation of a two-layer message delivery confirmation mechanism.

At the protocol level, the Meshtastic network ensures message receipt confirmation through the ACK_APP port number. On top of this, an additional application layer is implemented, where the remote API server returns an HTTP response indicating the success of data processing. This combination allows the system to maintain a clear record at all times of whether a message has been successfully delivered and acknowledged.

If an internet connection interruption occurs, the application continues to operate smoothly in offline mode, during which all data is stored in the local database and marked with an error flag. Once the connection is restored, the application automatically retries transmission and synchronizes the database with the remote service.

Furthermore, the system architecture has been extended to include two operational modes of the application: server mode and client mode.

Figure 3 shows the application in server mode.

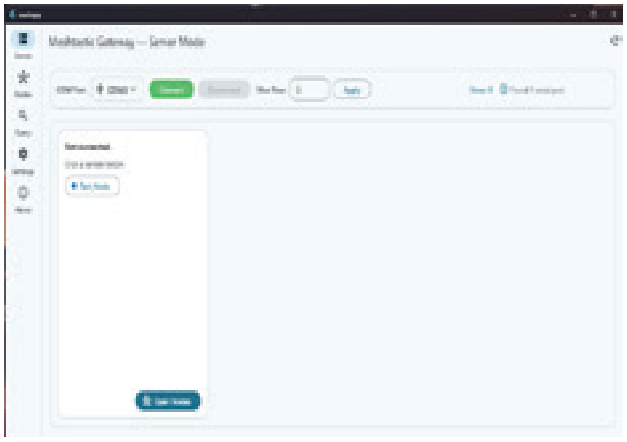


Figure 3. *Meshtastic Gateway - Server mode*

In server mode, the application assumes a central role. It serves as the point where complete data processing is performed, including filtering, transformation into JSON format, database entry, and transmission to Grafana. This mode of operation implies that the server application maintains the main instance of the database and functions as a gateway between the Meshtastic network and the internet [3]. In this way, the server mode enables integration of the entire system with external analytical tools and ensures centralized real-time monitoring of all network nodes.

In client mode, the application operates in a simplified configuration. It does not process or store data directly but instead sends queries through the Meshtastic network to the main application running in server mode and retrieves results from the database. This allows remote users, those without direct internet access, to obtain information about the system's status. The client mode extends the concept of decentralization by enabling access to data even in situations where the client does not have a connection with the server through a traditional network, but exclusively via the Meshtastic network. This functionality is particularly important for field operations, where operators or researchers can monitor data on their computers or mobile devices, while the main server remains located in a secure environment with constant power supply and internet access.

The visual and user interface component of the application has been carefully developed to ensure intuitive operation in both modes. The Dashboard provides an overview of all active nodes and key system metrics, while the configuration section allows users to enter all parameters necessary for operation (node IDs, names, API keys, and color categorization). A significant feature is the integration with the database through the application interface, which enables users to explore and query data locally without connecting to remote services. Application settings, such as theme selection or automatic connection to the first available node upon startup, further simplify usage in field conditions.

By combining server and client modes, the system architecture provides complete flexibility and achieves a balance between centralized processing and decentralized access. In this way, the developed solution gains characteristics that surpass traditional IoT implementations: the system becomes simultaneously robust, scalable, and adaptable to diverse operational conditions.

The developed application is designed to give users full control over system operations and access to all collected data, with special attention paid to interface clarity and usability. Its functionalities are organized into several modules that together form a single, intuitive tool for managing distributed IoT nodes.

At the core of the application is the Dashboard, which serves as the system's operational center. On this screen, users can view all active nodes, their basic parameters, and statuses. For each node, data such as identification number, name, and the latest received data set are displayed, along with availability indicators. Special attention has been devoted to presenting the overall "health" of the system. Users can monitor node availability, message loss rates, transmission delays, and general communication status. The Dashboard also includes the status of the local database and the remote API, allowing users to see in real time whether data is successfully transmitted to the analytical platform or held in a waiting state due to internet issues.

Visual alerts notify users of critical events such as network interruptions, API connection loss, or sudden drops in node battery voltage and capacity. This enables quick assessment of the entire system and timely response to any problems.

The second segment of the application is the Node Configuration Module, where all parameters necessary for data monitoring and processing are defined. For each node, it is possible to enter its SENDER ID, a name for easier identification, and an API key [4] that enables communication with remote services. In addition, Node ID values are assigned, which are used on the visualization platform side, along with color labels that categorize nodes within the interface to improve clarity in more complex systems.

This section provides full control over the integration of new nodes, while the flexibility of the module allows for easy addition, modification, or removal of devices from the network.

Figure 4 shows the configuration interface for the nodes.

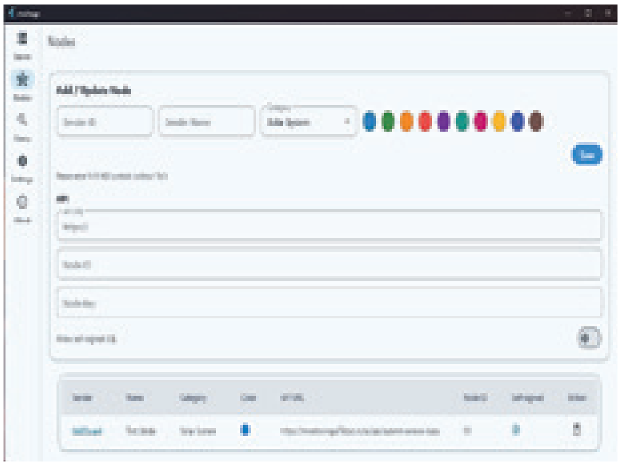


Figure 4. Configuration interface for the nodes

The third functional unit of the application relates to database operations, implemented through the Query Module. Here, users can explore and analyze historical data stored in the local database. The data can be filtered by time intervals, sender identity, or type of measurement. This functionality enables users to generate reports and perform analyses even when the internet is unavailable or when remote services are not operational. In this way, the database is not merely a passive element of the system but becomes an active tool for exploring and evaluating all collected values.

The Settings Module forms the fourth functional unit and is designed to adapt the application's operation to user needs and specific conditions. It allows for theme selection to improve visibility under various lighting environments and the activation of an auto-connect option that links to the first available node on the serial port at startup, significantly simplifying field operations. This section also provides fine control over security settings, including the acceptance of self-signed SSL certificates when working with private or test API services.

Another important part of the application is the About section, which provides a summary of essential information about the application and its authors. This section serves as an overview of the basic documentation and metadata, giving users insight into the software version and providing contact information for technical support.

All system functions are integrated into a single application that combines monitoring, configuration, and data analysis. This unified design provides users with a practical and reliable tool for managing and supervising distributed IoT systems.

In distributed IoT systems that rely on LoRa and Meshtastic networks, message loss and communication interruptions represent real challenges. Therefore, the developed solution places particular emphasis on reliability and redundancy. Mechanisms are implemented across multiple layers, from the transport layer managed by Meshtastic to the application layer and local data storage.

Figure 5 shows the Meshtastic logger, which reads the data transmitted by a node via the COM port.

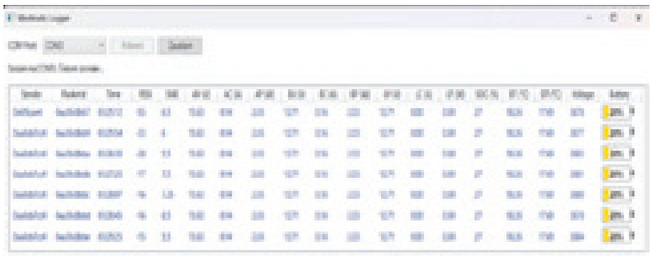


Figure 5. *Meshtastic logger*

The ACK [5] mechanism within the Meshtastic network represents the first layer of reliability. Each transmitted message can receive an acknowledgment (ACK) from the next node, ensuring that the data has been successfully forwarded within the network, even if it has not yet reached the final application. This level of confirmation is particularly important in mesh topology, where messages may be routed through multiple intermediate nodes. If an ACK is not received, the message is retransmitted, thereby reducing the likelihood of data loss due to transient interference.

Building on this, an application-level ACK is implemented, originating from the remote API after an HTTP request. While Meshtastic confirms only that the message has successfully traversed the network, the HTTP response provides information about whether the data was actually received and recorded by the external service, such as Grafana, as shown in Figure 6.

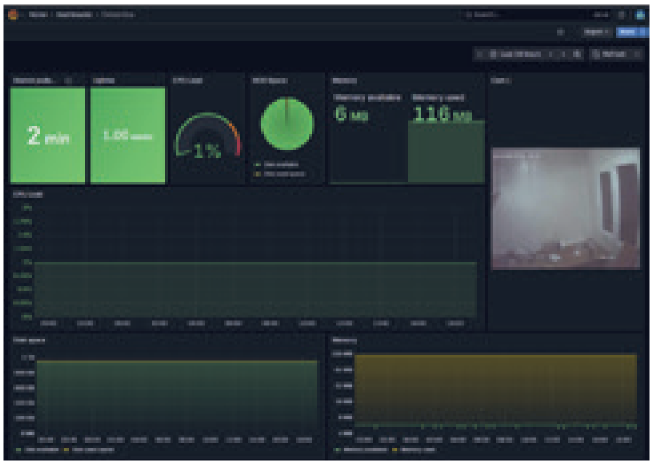


Figure 6. *Grafana*

If the remote server returns an error or fails to respond within the expected time, the application logs this event and marks the message as "undelivered."

To ensure data redundancy, every message, whether successfully sent to the API or not is recorded in the local database. If external transmission fails, the message is flagged as "undelivered," and the reason for failure (unavailable internet connection, timeout, or API response error) is stored. This provides a complete record and enables straightforward debugging and post-analysis. [6]

When the internet connection is restored, automatic synchronization is triggered. The application identifies all messages marked as "undelivered" and resends them sequentially to the remote API until a positive ACK is received. This mechanism effectively eliminates the risk of permanent data loss and allows the system to continue operating seamlessly even after prolonged network interruptions.

Such a multi-layer reliability approach makes the solution suitable for real-world scenarios where communication disruptions are frequent, such as remote rural areas or industrial environments. [7]

Results

The developed system was implemented through a combination of hardware and software components, where ESP32 microcontrollers with attached sensors formed the foundation for data collection, while the LoRa Meshtastic network provided the communication layer. The software part included an application developed in the Dart/Flutter environment, a parser for data filtering and transformation, and a local database in which all messages were stored along with their delivery statuses. This created a self-sustaining mechanism capable of operating under various network conditions.

In the first phase of testing, the system operated under stable conditions with a constant internet connection. Sensor nodes sent data at 10-second intervals, and the application processed and forwarded them to the remote API. The latency from the moment of data generation on the sensor to its display in Grafana averaged 1.2 seconds. During several hours of continuous operation, no data loss was recorded, confirming that the system functions flawlessly in ideal network conditions.

Next, the system was tested in scenarios involving internet connection interruptions. During periods lasting between five and thirty minutes without connectivity, all data were smoothly recorded in the local database, marked as "undelivered," and supplemented with metadata about the cause of the error. Once the connection was restored, the application automatically performed resynchronization and forwarded all messages to the remote API. The results showed that no data were lost, and the local database allowed complete event reconstruction and precise tracking of the time each error occurred.

The network's performance was also tested with a ten nodes. Nodes were included, periodically sending data to simulate a more complex system. The application's parser successfully filtered out around 35% of the traffic generated by Meshtastic in the form of auxiliary and non-system messages, so only relevant data from defined senders proceeded for processing. Despite the increased communication volume, the application maintained stable operation, with no network congestion or message loss observed.

The local database proved to be not only essential for maintaining system integrity but also invaluable for data analysis. Queries such as extracting all data for a specific node within the last 24 hours or searching for voltage values above a defined threshold were executed in under 100 milliseconds. This confirmed that the database serves not merely as a passive layer but as an active tool for real-time exploration of historical data.

The user interface demonstrated clarity and usability during practical testing. The Dashboard provided a clear overview of active nodes and key system metrics, while the Configuration Module offered flexibility in assigning API keys, identification numbers, and visual markers.

The Query Module proved especially valuable in offline mode, enabling report generation directly from the local database without the need for internet access. Features such as automatic connection to the first available node upon startup were particularly useful in field conditions, where speed and simplicity are crucial.

The research results confirm that the system provides stable and reliable performance. Under stable network conditions, the average latency was 1.2 seconds, no data loss occurred in any scenario, and around 35% of non-system traffic was effectively filtered out. The network successfully handled the load of ten nodes, and the local database enabled fast analytical queries with response times below 100 milliseconds. The main limitation remains the low bitrate of LoRa communication, making the system optimal for small amounts of data transmitted at periodic intervals.

Discussion

The evaluation of the implemented system indicates that the integration of a LoRa Meshtastic mesh network with a locally hosted data processing and storage application constitutes a viable and reliable solution for Internet of Things (IoT) deployments in environments with limited or intermittent internet connectivity. The system architecture, based on decentralized communication and dual-layer reliability mechanisms, demonstrated robustness in various test scenarios, including network interruptions and increased communication load.

A significant contribution of the proposed solution lies in its ability to operate autonomously in offline conditions. The introduction of a two-layer acknowledgment mechanism, protocol-level ACKs via the Meshtastic network and application-level acknowledgments from HTTP API responses, ensured data integrity across all stages of transmission. In cases of connection loss, data packets were securely stored in a local database, appropriately flagged as undelivered, and subsequently synchronized upon reconnection. This redundancy model significantly reduces the risk of permanent data loss and supports system resilience in real-world conditions.

In comparison to conventional IoT architectures, particularly those based on LoRaWAN, the presented solution eliminates the dependency on centralized gateways and service provider infrastructure. The adoption of a mesh topology ensures fault tolerance through dynamic rerouting and enables flexible deployment in areas with challenging topography or limited access to power and network infrastructure.[8]

Furthermore, the implementation of server and client operational modes enhances system scalability and decentralization, allowing field users to access critical data via the Meshtastic network without direct access to the internet.

Although effective in many cases, the LoRa technology still has several built-in constraints. The restricted data rate and narrowband communication capacity limit the applicability of the system to use cases involving low-volume and periodic data transmission. While this makes it suitable for scenarios such as environmental monitoring, agriculture, or remote infrastructure supervision, it is not applicable in contexts requiring high-throughput data exchange or multimedia support.

Another notable constraint pertains to message parsing and data filtering. Although the system effectively utilizes regular expressions to eliminate non-system and auxiliary traffic, future iterations may benefit from the implementation of more adaptive parsing methods, such as machine learning-based classification or context-aware filtering, particularly in larger and more heterogeneous networks.[9]

Potential directions for future development include:

1. Optimization of performance in high-density node environments,
2. Support for alternative or parallel visualization platforms,
3. Implementation of real-time network topology mapping,
4. Enhancements in data security, including encryption and authentication layers,
5. Remote management and over-the-air firmware updates for sensor nodes.

Overall, the system demonstrates a high level of operational stability, adaptability, and practical value in field deployments. The modular software architecture, real-time visualization integration, and effective handling of intermittent connectivity represent key advancements in the design of resilient and cost-efficient IoT systems.

These findings suggest that the proposed approach provides a sustainable foundation for the further development of decentralized sensor networks tailored to infrastructure-limited environments.[10]

Conclusion

The developed system demonstrated that the combination of a LoRa Meshtastic network and an application for filtering, transforming, and distributing data can provide a reliable and cost-effective infrastructure for IoT applications in areas with weak or no mobile network coverage. Thanks to the mesh topology and the ability of each node to function simultaneously as both a transmitter and a repeater, the network ensured resilience to interruptions and stable message transmission.

Meanwhile, the application contributed by eliminating non-system data, standardizing the content into JSON format, and enabling integration with remote services such as Grafana.

Testing confirmed that the system operates successfully even in conditions with internet connection interruptions, as the local database assumes the role of maintaining system integrity. The delivery confirmation mechanisms at both the network and application levels guaranteed that no data were lost, while later resynchronization ensured consistency between the local database and the remote API. A key advantage of the solution lies in its flexibility, the same system can be used in server mode, as a central point for processing and visualization, or in client mode, where remote users can access data through Meshtastic even without an internet connection.

The system's limitations are primarily related to the capacity of LoRa technology, which is suitable for small amounts of data transmitted periodically but not intended for applications requiring high throughput. Nevertheless, in scenarios involving intermittent and low-volume transmissions, such as environmental monitoring, agriculture, remote industrial site supervision, or smart city applications in poorly connected areas, the developed solution provides an optimal balance between reliability, simplicity, and cost-effectiveness.

REFERENCES

[1] F. Freitag, J. M. Solé, and R. Meseguer, “Position Paper: LoRa Mesh Networks for Enabling Distributed Intelligence on Tiny IoT Nodes,” 2023. [Online]. Available: https://www.researchgate.net/publication/371826243

[2] G. Y. Kusuma and U. Y. Oktiawati, “Application Performance Monitoring System Design Using OpenTelemetry and Grafana Stack,” Journal of Internet and Software Engineering, vol. 3, no. 1, 2024.

[3] R. P. Centelles, R. Meseguer, and F. Freitag, “Exploring Open Source and Proprietary LoRa Mesh Technologies: A Minimalistic Routing Protocol for LoRa Mesh Networks,” 2024. [Online]. Available: https://www.researchgate.net/publication/380253012

[4] R. Berto, P. Napoletano, and M. Savi, “A LoRa-Based Mesh Network for Peer-to-Peer Long-Range Communication,” 2021. [Online]. Available: https://www.researchgate.net/publication/352707672

[5] M. A. Khan, M. T. Islam, and A. R. Chowdhury, “Implementation of Multi-Hop Mesh Networking Using ESP32 for IoT Communication,” 2024. [Online]. Available: https://www.researchgate.net/publication/389763039

[6] T. G. Durand, “Performance Evaluation of a Mesh-Topology LoRa Network,” *Sensors*, vol. 25, no. 5, 2025.

[7] D. Arregui Almeida et al., “Gateway-Free LoRa Mesh on ESP32: Design, Self-Healing and Multi-Hop Communication,” *Sensors*, vol. 25, no. 19, 2025.

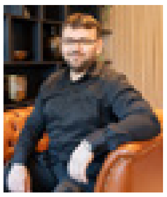
[8] P. Kietzmann, J. Alamos, D. Kutscher, T. C. Schmidt & M. Wählisch, “Long-Range ICN for the IoT: Exploring a LoRa System Design,” *Proc. 21st IFIP Networking Conference*, 2022.

[9] N. L. Giménez et al., “Embedded federated learning over a LoRa mesh network,” *Elsevier*, 2023.

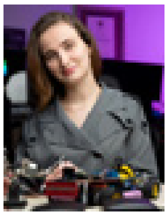
[10] J. R. Cotrim, J. H. Kleinschmidt & al., “LoRaWAN Mesh Networks: A Review and Classification of Multi-hop Communication,” *Sensors*, 2020

Received: November 3, 2025
Accepted: November 18, 2025

ABOUT THE AUTHORS



Daniel Menićanin, a student at the Faculty of Information Technology specializing in Programming and Software Engineering at Paneuropean University Apeiron, is an innovator recognized for developing groundbreaking solutions. Among his notable projects is the RC Platform Controlled via PlayStation 4 Controller, designed to support children with developmental challenges. His innovation, Software Development for CNC Hydraulic Presses, earned him a gold medal at the INN&TECH conference in Sarajevo, Bosnia and Herzegovina. In recognition of his exceptional contributions to education, science, and the arts, Daniel received the Plaque of the City of Banja Luka. He was also awarded the prestigious Butterfly Innovation Award in the Youth category, presented by the Regional Cooperation Council. Currently, Daniel’s research focuses on robotics, industrial machinery, and automation, further advancing his expertise in developing cutting-edge technological solutions.



Jelena Radanović, a dedicated and accomplished student of Programming and Software Engineering at Paneuropean University Apeiron, is widely recognized for her innovative contributions and exceptional achievements in the field. She has received the prestigious Butterfly Innovation Award and the INOST Medal for her project Press Brake Software, showcasing her expertise in software development and creative problem-solving.
Her work demonstrates a unique blend of technical proficiency and creativity, solidifying her reputation as a rising talent in programming and software engineering. Jelena continues to push boundaries in her studies and projects, positioning herself as a leader in technological advancement and innovation.

Dražen Marinković was born in 1978. He received his M.Sc. degree in 2015 and Ph.D. degree in 2020, both from the Faculty of Informatics at the Pan-European University Apeiron in Banja Luka, specializing in computer and informatics engineering. He is currently an associate professor at the Faculty of Informatics, Pan-European University Apeiron. His research interests include data science, computer networks, and related fields in modern computing technologies.

FOR CITATION

Daniel Menićanin, Jelena Radanović, Dražen Marinković, Implementation of a Meshtastic Gateway System With a Local Database for Iot Applications, *JITA – Journal of Information Technology and Applications, Banja Luka*, Pan-Europien University APEIRON, Banja Luka, Republika Srpska, Bosna i Hercegovina, JITA 15(2025)2:85-92, (UDC: 004.76+004.42), (DOI: 10.7251/JIT2502085M), Volume 15, Number 2, Banja Luka, December (81-176), ISSN 2232-9625 (print), ISSN 2233-0194 (online), UDC 004