CENTRALIZED DATABASE AND DISTRIBUTED COMMUNICATION IN A SOFTWARE SYSTEM FOR HIGH-PRECISION INDUSTRIAL MACHINE CONTROL

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/JIT2601020M UDC: 004.6:[004.9:004.451.9

Abstract: High precision in industrial processes depends not only on the mechanical and control components of the system, but also on the way data related to machine operation and operator activity are organized, stored, and processed. This paper examines the application of the relational data model and distributed communication architecture in an industrial software system intended for controlling machines in which deformation, forming, and material processing are directly conditioned by precise positioning and stable process supervision. The implemented solution includes a centralized application layer developed in the Dart programming language using the Flutter framework, a distributed microcontroller layer implemented in C++, and a centralized PostgreSQL database deployed in a Docker environment on a Proxmox server. Communication between the computer and the main control node is established via a USB-UART connection, while remote actuator and sensor modules are interconnected through a CAN bus. The system supports management of operator accounts, position presets, work sequences, worker-specific tasks, event logs, and remote access via Tailscale VPN infrastructure. The paper analyzes the database structure, relationships between tables, integrity constraints, data export and import organization, as well as security mechanisms based on PIN hashing and software license protection using asymmetric cryptography. The results show that the integration of a centralized database layer and a distributed communication architecture represents a functionally and technically justified solution for this class of industrial systems.

Keywords: PostgreSQL, RDBMS, Flutter, Dart, ESP32-S3, CAN bus, Tailscale VPN

INTRODUCTION

Modern industrial production increasingly relies on software systems that in addition to process control, provide reliable organization, storage, and processing of data generated during operation. Their importance is particularly evident in environments where process quality depends on positioning accuracy, repeatability of work sequences, event logging, and access control. Under such conditions, the database is not merely a supporting application component, but one of the key elements of the overall software architecture [1].

The need for systematic data management is especially pronounced in machines used for material deformation, forming, and processing, where even minor deviations can affect product quality and operational efficiency. In addition to process parameters, it is necessary to ensure the storage of position presets, work sequences, user settings, operator accounts, access rights, sessions, machine parameters, events, and audit trails. Such requirements point to a strictly structured and consistent data model suitable for multi-machine and multi-location environments.

This paper considers an industrial software system based on a distributed architecture in which the central application layer interacts with a main microcontroller node and a set of remote actuator and sen-

sor modules. The application layer is implemented in the Dart/Flutter environment, while the control firmware is developed in C++. Since the system is based on clearly defined entities with stable relationships, the relational model represents a natural choice for data organization [2]. The main application interface, which provides access to command functions, positioning controls, and predefined position registers, is shown in Fig. 1.

Figure 1. *Main interface of the implemented software system for industrial machine control*

The technical contribution of this paper lies in the definition, implementation, and evaluation of an integrated model that combines a centralized relational persistence layer, a distributed communication architecture, multi-client access, and access control within a precision-control system intended for industrial environments. The proposed model is not limited to data storage alone, but also encompasses the structural organization of operator, machine, process, and event-related information, as well as its interaction with the distributed control layer. In this way, the paper addresses both the informational and communication aspects of industrial software systems in which reliability, consistency, and operational continuity are of critical importance. Within this context, the following research questions are addressed:

1. Does a centralized relational data model provide a reliable foundation for managing operator, configuration, and process data in high-precision industrial systems?
2. To what extent does PostgreSQL, as a centralized RDBMS, satisfy the functional and architectural requirements of a distributed system compared with NoSQL approaches?
3. How do relational structure, integrity constraints, access control, audit trails, and distributed communication contribute to the sustainability and practical applicability of the implemented solution?
4. How does the proposed architecture contribute to operational continuity, supervision, and coordinated management in multi-machine industrial environments?

Methods and Materials

This section presents the architecture of the implemented system, the organization of the communication layer, the structure and logic of the data model, and the technological environment in which the solution was developed. Special attention is given to the distribution of functions between the application, control, and sensor-actuator layers, centralized data persistence, remote access mechanisms, and system security aspects. In this way, the methodological and technical framework underlying the analysis presented in the remainder of the paper is defined.

System Architecture

The considered system was implemented as a multi-layer architecture in which supervision, control, acquisition, and persistence functions are distributed across several interconnected components. At the highest level, there is a central application layer developed in the Dart programming language using the Flutter framework [3]. This layer is available through desktop and mobile applications and serves as the main user, supervisory, and administrative interface of the system. Through it, operator login, worker account management, password reset, user creation and deletion, task definition and assignment, access to position presets, creation and modification of work sequences, as well as review of logs and system events are performed. The administrative interface for managing operator accounts and access-related actions is shown in Fig. 2.

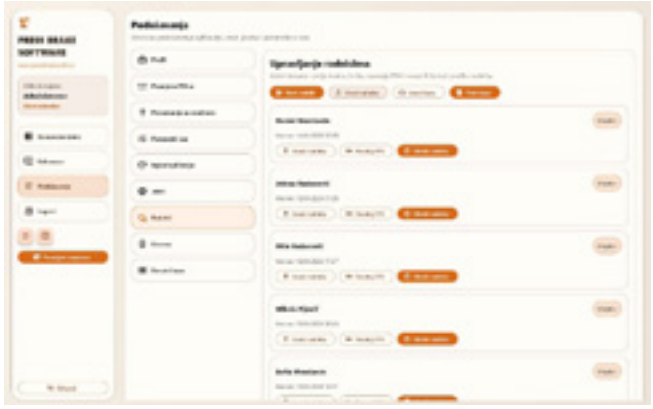


Figure 2. *Administrative interface for managing operator accounts and access-related actions*

At the control level, the system consists of multiple microcontroller nodes. The main control node is implemented on an ESP32-S3 microcontroller and communicates with the computer via a USB-UART connection. It acts as a bridge between the application layer and the remote microcontroller modules, receiving commands, forwarding them to actuator and sensor nodes, and returning status information to the application. The internal hardware layout of the main ESP32-S3 control node is shown in Fig. 3.

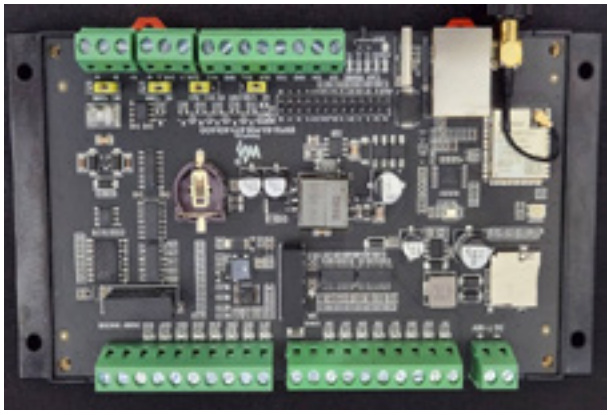


Figure 3. *Internal hardware layout of the main ESP32-S3 control node*

The firmware of the main node was developed in the C++ programming language and includes communication logic, operating mode management, coordination of execution modules, and data acquisition from the distributed layer. Remote execution nodes are responsible for actuator control, particularly hybrid stepper motor drivers with encoder feedback. Their function includes generation of control signals, supervision of axis states, execution of motion commands, and system calibration through the machine-zero referencing procedure. In addition to the execution modules, the system includes sensor nodes for collecting information from various types of sensors, including industrial PNP and NPN metal detectors, limit switches, laser presence and position sensors, as well as high-pressure probes in the hydraulic system. In this way, a functional separation is achieved between actuator control and process-signal acquisition, which improves modularity, reduces local signal-routing complexity, and supports a clearer distribution of responsibilities within the distributed control layer. At the application level, this architecture is complemented by dedicated supervisory interfaces through which machine-specific parameters can be reviewed, modified, and validated. These interfaces provide controlled access to axis parameters such as steps per millimeter, speed, acceleration, homing settings, backlash compensation, and working limits, as shown in Fig. 4.

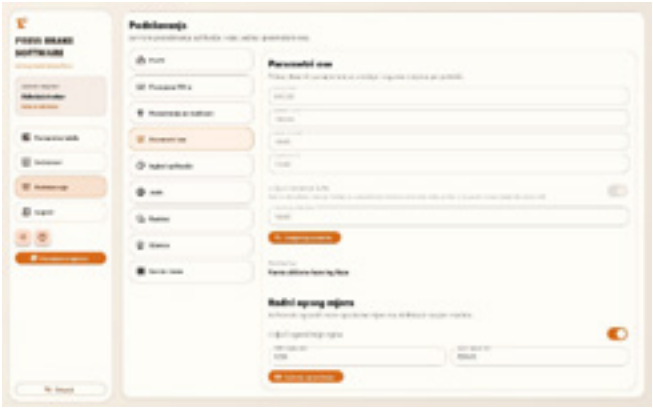


Figure 4. *Interface for axis parameter, homing, and working-range configuration*

A particular value of the solution lies in its cross-platform nature. The application can run on Windows, Linux, and Android operating systems, enabling flexible use in office, production, and field environments. At the same time, access to centralized data is organized through a central server layer within the Tailscale VPN environment [4], [5], thereby logically separating desktop and mobile clients from the database itself. In this way, direct exposure of the infrastructure is avoided, while controlled access, clearer architecture, and easier system maintenance are ensured.

Communication Architecture

The communication architecture of the implemented system is organized hierarchically. At the first level, there is a USB-UART connection between the computer and the main ESP32-S3 node, through which commands, status messages, and process data are exchanged between the supervisory software and the control layer. At the second level, the main control node communicates with remote execution and sensor modules via a CAN bus [6]. The selection of the CAN bus is based on the requirements of the industrial environment, in which high-power electric motors, contactors, relay assemblies, variable-frequency drives, and power lines represent significant sources of electromagnetic interference. The hardware platform of the main ESP32-S3 control node, including the CAN and RS485 interfaces used within the communication layer, is shown in Fig. 5.

Figure 5. *Hardware platform of the main ESP32-S3 control node with CAN and RS485 interfaces*

Under such conditions, differential signal transmission increases the resistance of the communication layer to electromagnetic disturbances, which makes the CAN bus an important element of the overall robustness of the architecture.

An additional advantage of this organization is that execution modules can be physically positioned close to the actuator assemblies they control, while sensor modules remain close to the sources of process information. This reduces the need for long and sensitive sensor and actuator wiring, while communication toward the main node remains unified through the bus infrastructure.

Robustness to Network and Server Interruptions

In the implemented architecture, the central database and VPN layer are not part of the real-time motion-control path. Critical machine-control operations are handled locally by the ESP32-S3 main control node and the distributed execution modules connected through the CAN bus. Therefore, interruption of the connection to the central server primarily affects remote supervision, synchronization of logs, retrieval of new tasks, and access to centralized historical data, while already validated local machine-control functions remain available at the controller level.

In the event of communication loss between the supervisory application and the central service, the architecture is designed so that unverified remote commands are not executed until the connection is restored and the machine state is re-synchronized. During such an interruption, the central server primarily loses its role in remote supervision, data synchronization, and access to historical or administrative information, while the local control layer remains responsible for previously validated machine-control routines. Safety-related signals, such as limit switches, homing state, end-limit supervision, emergency conditions, and local actuator status, remain under the responsibility of the ESP32-S3 control node and the distributed execution modules. This means that critical control behavior is not directly dependent on continuous availability of the central database or VPN connection. Such separation between the supervisory layer and the local control layer reduces dependence on the central server for critical machine movements, limits the risk of unsafe command execution during communication interruptions, and improves operational continuity in industrial environments.

Centralized Database Architecture

The implemented solution uses a centralized PostgreSQL database. The database is installed in a Docker container, while the complete server environment runs on a Proxmox server [7], [8] within the company infrastructure. This approach was introduced because of the need for multi-layer access to the system, including desktop and mobile clients, simultaneous operation of multiple machines, and access from multiple locations.

Relational Data Model

The data model was designed as a multi-context relational system that explicitly models the organizational structure, access levels, machine context, and process data. At the top of the hierarchy are the companies and locations entities, which define the organizational and spatial context of the system. The machines entity represents the central connection between the physical infrastructure and the process layer, with each machine being linked to a company and location, and further associated with controllers, process parameters, sessions, events, and operator access. The *machine_controllers* entity is used to record the identity and status of control units, including controller type, firmware version, and operational data relevant for supervision. The structure of the centralized relational data model and the relationships between its main entities are shown in Fig. 6.

The operator layer is modeled through the operators entity, along with the additional relations *operator_location_access* and *operator_machine_access*, thereby establishing multi-level access control. In this way, it is possible to precisely define which operator is allowed to work at particular locations and on particular machines. The *operator_sessions* entity enables the recording of logins, logouts, and session statuses, thereby introducing additional traceability of operator activities.

The process layer is modeled through the entities *presets, sequence_articles*, and *sequence_steps.* The presets entity includes predefined position presets, while the *sequence_articles* and *sequence_steps* entities model work sequences through logical grouping and the elaboration of individual steps. Such an organization enables process patterns to be stored, shared, reused, and adapted to different operators, machines, or locations. The *machine_parameters* entity is used to store machine configuration parameters, such as steps per millimeter, speeds, and accelerations, while *machine_state_snapshots* models current or periodically recorded machine states, including position, calibration status, reference mode, and end-limit states. In this way, the database stores not only static configuration information, but also dynamic operational data. The event and revision layer is modeled through the entities *machine_events* and *audit_logs*. The *machine_events* entity contains process and system events related to a specific machine and operator, while *audit_logs* provides a generic trace of administrative and entity-level changes in the system.

Long-Term Data Management and Database Scalability

Since the machine_state_snapshots table may

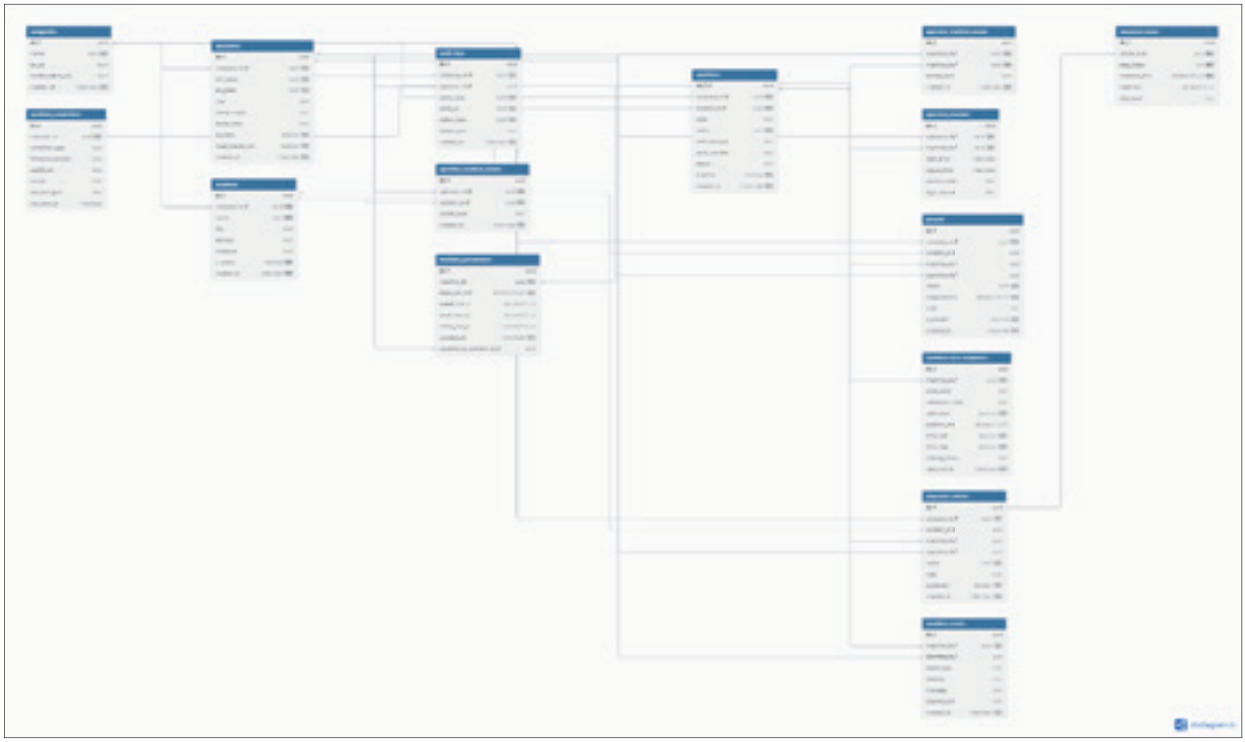


Figure 6. *Entity-relationship schema of the centralized relational database*

grow rapidly in a multi-machine environment, long-term data management must be considered as part of the database architecture. In the implemented model, current machine states and historical state records are conceptually separated. Recent records are used for supervision, diagnostics, and synchronization, while older records can be archived or aggregated depending on maintenance and reporting requirements.

For larger deployments, the table can be optimized using indexes over machine_id and timestamp fields, time-based partitioning, and retention policies that keep high-resolution state data only for a defined operational period. Older snapshot data may be transferred to historical tables or summarized into aggregated diagnostic records. In this way, the system preserves traceability without allowing high-frequency operational data to reduce database performance over time.

Integrity Constraints and Transaction Management

The quality of the model is reflected not only in the number of entities, but also in the way constraints are defined over the data. Primary keys ensure the unique identity of each record, while foreign keys prevent illogical associations between entities. Mandatory fields are defined as NOT NULL, and appropriate relations apply cascading deletion or NULL assignment rules depending on the semantics of the data. In this way, the database actively contributes to preserving system integrity.

In more complex operations, such as creating a work sequence with multiple steps, assigning access to an operator, or changing machine configuration, transactional consistency plays a key role. Such operations represent a set of mutually related writes that must be treated as a single logical unit. The PostgreSQL transaction mechanism ensures that such units are either fully committed or completely rolled back in the event of an error, thereby preventing partial writes and preserving the logical correctness of the system [9].

Multi-Client Access and Remote Availability

The centralized database allows an operator to retain access to personal settings, work measures, sequences, tasks, and work history regardless of the machine being used, provided that the machine belongs to the same system environment. In this way, a unified user and process context is achieved across multiple machines, which is particularly important in production systems where operators work on the same or similar machines distributed across multiple workstations or locations. Access to the database is organized through the application layer, which separates business logic from the query layer itself, thereby achieving clearer code organization, easier testing, and greater maintainability of the system. In the implemented architecture, desktop and mobile clients do not access the PostgreSQL database directly, instead, all interactions with centralized data are mediated through the application service layer within the Tailscale VPN environment, which enforces access control and isolates the database from direct client exposure.

The use of a centralized PostgreSQL solution, instead of a local database tied to a single workstation, naturally supports multi-client operation, central administration, and the availability of the same data across multiple devices.

The complete work process can be monitored and administered remotely using a dedicated Flutter mobile application [10] and desktop client, with access established through Tailscale VPN infrastructure. In the implemented solution, Tailscale enables secure connection of devices within a closed virtual network, without the need for public exposure of services or complex configuration of traditional VPN solutions. In this way, the mobile application, remote computer, and the server environment hosting the PostgreSQL database remain available through controlled, identity-based access, which supports secure supervision, centralized administration, and reliable access to operational and configuration data from different locations. The administrative interface for remote monitoring of service and database status is shown in Fig. 7.

Figure 7. *Administrative interface for remote monitoring of service and database status*

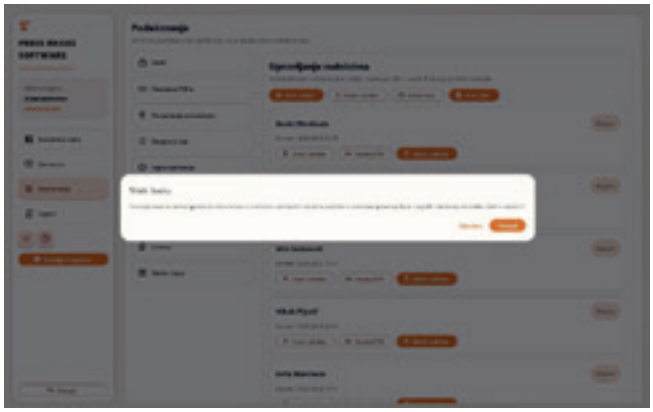


Figure 8. *Interface for restoring the database from a backup file*

The practical value of this architecture lies in the fact that all centrally defined data related to a specific worker are available through the VPN, including their settings, predefined measures, work tasks, and activity history. In this way, the system goes beyond the framework of a local control application tied to a single machine and becomes a centralized industrial software environment supporting multiple users, multiple devices, and multiple locations.

Data Portability and Security Mechanisms

An additional operational value of the system lies in its support for data export and import. The implemented approach enables backup creation, data migration, and restoration of the system state whenever servicing, infrastructure replacement, or transfer of configuration to another controlled environment is required. In this way, the system supports not only routine administrative procedures, but also recovery scenarios in which continuity of operation must be preserved. Export and import mechanisms [11] make it possible to transfer operator data, presets, work sequences, and related configuration elements without the need for repeated manual setup. This reduces downtime, simplifies maintenance procedures, and improves portability of the software solution across different workstations and deployment contexts. Consequently, these mechanisms do not represent merely auxiliary administrative functions, but an important element of system continuity, maintainability, and long-term operational reliability. An example of the interface used for database restoration from a backup file is shown in Fig. 8.

The security aspect of the system does not relate only to the protection of user access to the application, but also to the protection of the software solution itself against unauthorized use. For this reason, two mutually complementary protection mechanisms have been applied. The first relates to operator authentication, where login is performed using a PIN whose value is stored in the database in hashed form, thereby reducing the risk associated with direct exposure of authentication data and improving protection of user credentials. In addition, this approach supports personalized access to operator-specific settings, presets, and work sequences while ensuring that unauthorized users cannot access protected system functions. The operator login interface used for authentication and loading of personalized user settings is shown in Fig. 9.

Figure 9. *Operator login interface of the implemented industrial control application*

The second relates to software license protection using asymmetric cryptography [12]. In this model, the private key remains with the system author and is used for digitally signing the license, while the public key is embedded in the application and is used to verify its authenticity. Such an approach ensures that valid licenses can only be issued within the controlled service environment and that the verification process can be performed locally within the application itself. In this way, both user-level access protection and software-level authorization are addressed as integral parts of the overall system security model.

The implemented licensing subsystem also includes a dedicated internal License Manager application intended exclusively for the software author or authorized service personnel. The tool processes exported request files generated by the main application and enables license generation, renewal, and activation-code export. The licensing workflow supports both perpetual and time-limited licenses and is fully based on the same asymmetric cryptographic model, in which the private signing key is retained exclusively within the internal service environment. The administrative interface used for license verification and activation is shown in Fig. 10.

Figure 10. *Administrative interface for license verification and activation*

This architecture prevents local generation or modification of a valid license by the end user, while license verification remains possible without relying on a constant Internet connection or a remote license server.

Results

The performed validation confirmed that the implemented system provides stable and consistent operation of the centralized data model, distributed communication layer, and remote multi-client access architecture. The obtained results indicate that the proposed software-hardware integration satisfies both the functional requirements of precise industrial control and the organizational requirements of centralized data management in multi-machine environments.

At the application and database level, the centralized relational model enabled reliable management of operator accounts, position presets, work sequences, assigned tasks, and system events within a unified information environment. The average operator login time, including authentication, access-right verification, and loading of the user context, was 142 ms, while loading of position presets and work sequences required 31 ms and 56 ms, respectively. These values indicate that centralized persistence does not introduce delays that would negatively affect routine industrial use.

At the communication level, validation showed that the hierarchical organization based on the main control node and remote execution and sensor modules enables reliable exchange of commands and status information under industrial operating conditions. The average response time between the application layer and the main ESP32-S3 node over the USB-UART connection was 48 ms, while the average exchange time between the main node and remote modules over the CAN bus was 14 ms. No communication loss was observed during the test scenarios, and the functional separation between execution and sensing layers was preserved in all cases. These results confirm that the selected communication architecture provides both responsiveness and robustness in environments exposed to electromagnetic disturbances.

Remote access through the Tailscale VPN infrastructure was also successfully validated. Desktop and mobile clients were able to access centralized data and administrative functions with an average remote response time of 187 ms under normal operating conditions. This result confirms that the developed architecture supports centralized supervision and coordinated management of multiple users,

machines, and locations within a single software environment, without compromising practical usability.

At the database consistency level, the correctness of relations between the organizational, machine, operator, process, and revision layers was verified through complex transactional operations. Work-sequence modification, access-right assignment, and machine-parameter updates were executed with a 100% success rate in the validation scenarios, without integrity violations or partial writes. Export and import of configuration and operational data required 3.2 s and 4.7 s, respectively, confirming that backup, migration, and restoration procedures can be performed efficiently within maintenance and recovery workflows.

From the industrial perspective, the implemented solution improved operational consistency, reduced dependence on machine-local data, and increased the availability of process-related information. Centralized storage of operator settings, work measures, and task assignments enabled the same operator to continue work on different machines without repeated manual reconfiguration, thereby reducing setup effort and improving workflow continuity. In addition, the availability of logs, audit records, and machine-event history improved process traceability, diagnostics, supervision, and accountability in production conditions.

The obtained results also show that the distributed architecture contributes to maintainability and scalability of the overall system. By separating execution and sensing functions across dedicated remote modules and interconnecting them through the CAN bus, the implemented solution reduced sensitivity to long signal lines, improved communication robustness, and supported modular deployment across multiple machines and locations. In that sense, the proposed architecture provides not only a technically valid integration of software, database, and communication layers, but also measurable operational benefits relevant to industrial practice.

DISCUSSION

The obtained results confirm the justification for selecting a centralized relational data model for the analyzed type of industrial software system. The nature of the data within the system clearly corresponds to the relational approach, since the entities are predefined, their relationships are stable, and preserving the integrity of those relationships is essential for the correct operation of the system. Under such conditions, a relational database management system represents the most suitable architectural and functional solution.

The choice of PostgreSQL proved justified due to the centralized nature of the system. Unlike local databases tied to a single machine or workstation, a centralized RDBMS allows the same operator to access their data from multiple machines and different client platforms while preserving a unified operator and process context. Such an approach is particularly important in systems with multiple workstations, multiple users, and multiple locations, where local persistence would not be able to provide the required level of consistency, availability, and administrative transparency.

An additional value of the implemented solution lies in the breadth of the data model, which includes companies, locations, machines, controllers, access rights, sessions, machine parameters, states, events, and audit trails. In this way, the organizational, machine, operator, process, and revision layers of the system are integrated within a single database, enabling not only reliable persistence but also improved traceability, supervision, and process management.

From the communication perspective, the distributed architecture based on the main control node and remote execution and sensor modules proved suitable for the industrial environment. The choice of the CAN bus is particularly significant, since differential signal transmission provides greater resistance to electromagnetic interference typical of systems with electric motors, contactors, variable-frequency drives, and power lines. In this way, the communication layer does not represent only a transmission mechanism, but also an important element of the overall robustness and maintainability of the system. Remote access via the Tailscale VPN infrastructure further extends the functional scope of the solution by enabling centralized administration, log review, task assignment, and access to user settings from different locations.

The comparison with NoSQL approaches further confirms the appropriateness of the selected solution. A NoSQL approach may be useful in systems dominated by unstructured telemetry, high-volume

sensor streams, or document-oriented data storage. However, the analyzed industrial control system is based on clearly defined entities and stable relationships between operators, machines, locations, access rights, process parameters, work sequences, sessions, events, and audit records. In such a context, relational integrity is more important than flexible data representation.

For example, when an operator is granted access to a specific machine at a specific location, this relationship must remain consistent with the corresponding company, location, machine, session, and audit records. In a document-oriented NoSQL model, such information would often have to be duplicated or coordinated across multiple documents, which increases the risk of inconsistent access rights, outdated copies of security-related data, and more complex validation logic at the application level. In contrast, PostgreSQL enforces these relationships through foreign keys, constraints, transactions, and rollback mechanisms.

Therefore, NoSQL systems could be considered as an auxiliary layer for high-volume telemetry, analytical storage, or non-critical log aggregation, but they are less appropriate as the primary source of truth for this type of safety- and integrity-sensitive industrial application. For the implemented system, the centralized relational model provides clearer consistency rules, stronger auditability, and more reliable management of operator access, machine parameters, and work-sequence modifications.

Based on this, it can be concluded that the main contribution of the paper does not lie only in the application of individual technologies, but in their architectural integration. The centralized database layer, distributed communication architecture, multi-client access, and security mechanisms are integrated into a unified solution tailored to industrial precision-control systems. It is precisely in this integration that the key value of the proposed model can be recognized.

Conclusion

This paper examined the application of a centralized relational data model and distributed communication architecture in an industrial software system intended for high-precision machine control. The analysis showed that such systems require not only reliable control of process-related functions, but also structured management of operator accounts, position presets, work sequences, access rights, system events, and audit-related data. In that context, the relational model proved to be a suitable foundation for organizing and maintaining consistency of industrial operational data.

The proposed architecture combines a centralized PostgreSQL persistence layer, a distributed microcontroller-based communication structure, and multi-client remote access within a unified software environment. Such integration enables a clear separation between application, control, sensing, and data-management layers, while also supporting centralized supervision, improved traceability, and coordinated operation across multiple machines and locations. The obtained findings confirm that this approach is functionally, technically, and architecturally appropriate for industrial environments in which precision, consistency, and operational continuity are critical.

An additional contribution of the paper lies in showing that the selected architecture provides not only technical feasibility, but also practical industrial value through improved maintainability, centralized administration, controlled access, and support for backup, recovery, and licensing mechanisms. At the same time, the study indicates that the developed model provides a solid basis for further extension toward advanced supervision, analytics, production-resource coordination, and broader integration of multiple industrial machines into a shared information environment.

References

[5] H. Desamsetti, “Relational Database Management Systems in Business and Organization Strategies,” Global Disclosure of Economics and Business, vol. 9, no. 2, pp. 151–162, Jul. 2020, doi: 10.18034/gdeb.v9i2.700.

[6] N. M. Ahmed and G. M. Haji, “TRADITIONAL RDBMS TO NOSQL DATABASE: NEW ERA OF DATABASES FOR BIG DATA,” International Journal of Scientific & Technology Research, vol. 10, no. 9, pp. 101–106, Sep. 2021.

[7] Google, “Flutter architectural overview,” Flutter Documentation, official documentation. Available: https://docs.flutter.dev/resources/architectural-overview

[8] Tailscale Inc., “What is Tailscale?,” Tailscale Documentation, official documentation. Accessed: May 12, 2026. [Online]. Available: https://tailscale.com/docs/concepts/what-is-tailscale

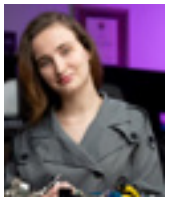
[9] D.-F. Hrițcan and D. Balan, “Using Tailscale and PfSense for Security and Anonymity of IoT Environments,” in 2024 International Conference on Development and Application Systems (DAS), 2024, doi: 10.1109/DAS61944.2024.10541192.

[10] Texas Instruments, "Introduction to the Controller Area Network (CAN)," Application Report, Rev. B.
[11] V. P. Oleksiuk and O. R. Oleksiuk, "The practice of developing the academic cloud using the Proxmox VE platform," Educational Technology Quarterly, vol. 2021, no. 4, pp. 605–616, Dec. 2021, doi: 10.55056/etq.36.
[12] Proxmox Server Solutions GmbH, "Proxmox VE Administration Guide," Proxmox Virtual Environment Documentation, official documentation. Accessed: May 12, 2026. [Online]. Available: https://pve.proxmox.com/pve-docs/pve-admin-guide.html
[13] The PostgreSQL Global Development Group, "Transactions," PostgreSQL Documentation, official documentation. Accessed: May 12, 2026. [Online]. Available: https://www.postgresql.org/docs/current/tutorial-transactions.html
[14] K. C. Panda, "Application Development Using Flutter and React Native: Cross Platform Development," Journal of Research in Science and Engineering, vol. 7, no. 1, pp. 6–8, Jan. 2025, doi: 10.53469/jrse.2025.07(01).02.
[15] D. Bordoloi, "Import and Export Database Management System," Mathematical Statistician and Engineering Applications, vol. 70, no. 1, pp. 182–189, Jan. 2021, doi: 10.17762/msea.v70i1.2298.
[1] C. Stohrer and T. Lugrin, "Asymmetric Encryption," in Trends in Data Protection and Encryption Technologies, V. Mulder et al., Eds. Cham, Switzerland: Springer, 2023, pp. 11–14, doi: 10.1007/978-3-031-33386-6_3.

Received: April 17, 2026
Accepted: May 5, 2026

ABOUT THE AUTHORS

Daniel Menićanin is a student at the Faculty of Information Technology, Pan-European University Apeiron, specializing in Programming and Software Engineering. His work is focused on robotics, industrial automation, and advanced software solutions for CNC and hydraulic systems. Throughout his academic career, he has participated in numerous innovation projects, including the development of adaptive gaming controllers and industrial control software. For his achievements, he received several awards at innovation and technology conferences in Bosnia and Herzegovina. His current research interests include embedded systems, industrial communication, and intelligent automation technologies.



Jelena Radanović is a Programming and Software Engineering student at Pan-European University Apeiron. She has been actively involved in software development and innovation projects related to industrial applications and modern programming technologies. Her work combines technical problem-solving with creative software design, particularly in automation and user-oriented systems. Jelena has received recognition for her contributions at regional innovation events and continues to expand her expertise in software engineering, application development, and emerging technologies.

Dražen Marinković received his M.Sc. degree in 2015 and Ph.D. degree in 2020 from the Faculty of Information Technology at Pan-European University Apeiron in Banja Luka. He currently works as an associate professor at the same institution. His academic and research activities are focused on computer networks, data science, distributed systems, and modern computing technologies. He has participated in various scientific and educational projects related to information technologies and software systems.

FOR CITATION

Daniel Menićanin, Jelena Radanović, Dražen Marinković, Centralized RDBMS and Distributed Communication in a Software System for High-Precision Industrial Machine Control, *JITA – Journal of Information Technology and Applications, Banja Luka*, Pan-Europien University APEIRON, Banja Luka, Republika Srpska, Bosna i Hercegovina, JITA 16(2026)1:20-30, (UDC: 004.6:[004.9:004.451.9), (DOI: 10.7251/JIT2601020M), Volume 16, Number 1, Banja Luka, June (1-76), ISSN 2232-9625 (print), ISSN 2233-0194 (online), UDC 004