STUDENT SESSION

SOFTWARE SYSTEMS FOR LABORATORY INSTRUMENTS IN LIFE SCIENCE: DESIGN PRINCIPLES AND CHALLENGES

Ana Stančić[1,2]*,
[0000-0001-6535-2152]

Svetlana Anđelić[3]
[0009-0001-3069-6702]

[1]HTEC Group,
Belgrade, Serbia

[2]Student,
Singidunum University,
Belgrade, Serbia

[3]Singidunum University,
Belgrade, Serbia

Abstract:

The digital transformation irrevocably changed the way life science research is done. Software used in life science has gradually shifted from a supporting tool to being a central part in laboratory instrumentalization and workflows. In modern laboratories, software is part of every step of the scientific process, from experiment planning and real-time instrument control to data acquisition, visualization, analysis, and results reporting. Besides other categories of software used in life science, laboratory instrument control software (LICS) has a particularly critical role. It must ensure reliable communication with hardware components that enable precise instrument control and data acquisition while maintaining a high level of usability and user-friendliness, which is expected of modern applications. Based on the author's combined experience in life science research and instrument software development, this paper reviews key design principles and common challenges in developing LICS.

The paper explores several design principles, including human-centric design that minimizes the cognitive load and aims to prevent errors, and modular, layered architectures that support reusability of software components and facilitate maintainability. Additionally, it covers hardware-software synchronization, interoperability based on FAIR (findability, accessibility, interoperability, and reusability) principles, and data integrity and safety mechanisms that are required for regulatory compliance.

Furthermore, this paper describes some of the common challenges in the field of LICS development, such as balancing between real-time performance and usability, and managing legacy systems and backward compatibility. It also explores the complexities of regulatory compliance, the difficulty of achieving maintainability across different data standards, and addresses security concerns.

Keywords:

Laboratory Instrument Control Software, Instrument Control, Life Science.

INTRODUCTION

The third industrial revolution (the digital revolution) has fundamentally transformed virtually everything, from daily life to industry and science. In the life sciences, software has evolved from being a supporting data storage tool into a central component of the research process, enabling more advanced discoveries in biology, medicine, and related fields. The umbrella term “life science” covers a broad range of disciplines focused on studying the living system, from molecular biology and genetics to applied fields such as pharmacology and epidemiology [1].

Correspondence:
Ana Stančić

e-mail:
ana.stancic.23@singimail.rs

DOI: 10.15308/Sinteza-2026-599-606

Contrary to physics or chemistry, where universal laws provide a stable theoretical framework, life sciences deal with inherent biological complexity and variability, and some authors even describe them as "borderline chaotic" [2], [3]. The ongoing digitalization process in life sciences helped tame some of that chaos.

Over time, software used in life science laboratories has evolved from simple data recording tools to complex, specialized systems. Early software solutions in the 1980s focused on digitizing instrument output and replacing paper records [4], while the 1990s and 2000s brought more structured software platforms shaped by increasing regulatory requirements, such as the Food and Drug Administration (FDA) Code of Federal Regulations (CFR) Part 11 guidelines for electronic records integrity [5]. Today, software supports every step of the scientific workflow, from experiment planning and real-time data acquisition to analysis and reporting. The scale of this transformation is demonstrated in market trends: according to Fortune Business Insights, the global life science software market size was evaluated at USD 17.69 billion in 2025 and is projected to grow from USD 19.48 billion in 2026 to USD 43.19 billion by 2034, exhibiting a CAGR of 10.50% during the forecast period [6].

1.1. CLASSIFICATION OF LIFE SCIENCE SOFTWARE SYSTEMS

Modern life science software can be classified into three functional categories: Direct Instrument Interaction, Data Management and Analysis and Operational Oversight.

Direct Instrument Interaction includes software that communicates directly with physical hardware. This group included Laboratory Instrument Control Software (LICS) - specialized applications that are usually provided by hardware manufacturers that operate machines such as chromatographs, mass spectrometers, or gene sequencers by translating user-defined methods into machine actions and capturing raw data [7]. A closely related sub-category is Scheduling and Automation Software that controls robotic components such as liquid handlers, robotic arms, plate movers, and others, to increase efficiency of these assays and minimize the human error factor [8].

The Data Management and Analysis category covers systems for storing, processing, and interpreting experimental data. This category includes Scientific Data Management Systems for archiving unstructured (raw) data from various laboratory instruments [9], [7], Life Science Analytical Software for transforming raw signals into meaningful visualizations (charts or graphs) [7], Bioinformatics tools for large scale biological datasets used in genomics, proteomics and other type of "omics" [10] and Simulation and Modelling Software which are extensively used in in silico drug discovery research [11].

Operational Oversight systems manage high-level laboratory organization. Key examples include Laboratory Information Management Systems (LIMS) for managing end-to-end laboratory operations [7], [12], Laboratory Instrument Systems (LIM) clinical and diagnostic reporting [12], Electronic Lab Notebooks (ELN) digital record keeping and collaboration [7] an Quality Management Systems (QMS) for managing regulatory compliance [13].

Among these, LICS plays a central role since it directly connects mechanical engineering, data processing, and regulatory compliance. Designing these systems is particularly challenging due to the need to balance reliability, real-time performance, and flexibility required by the unpredictable nature of life science research. This paper focuses on the main design principles and practical challenges involved in developing such systems, based on the author's experience in laboratory research and LICS development, as well as a review of relevant literature.

2. DESIGN PRINCIPLES

The transition from isolated analogue instruments to integrated digital systems required a shift in software design. Life science research is characterized by inherent variability and complexity [3], and LICS must reflect this by addressing user needs, modular architecture, and deterministic hardware interactions, while ensuring data integrity and safety.

2.1. HUMAN-CENTRIC DESIGN

The primary users of LICS are domain-expert scientists and laboratory technicians, not software engineers. Consequently, LICS design must prioritize usability and the reduction of cognitive load [14]. The user interface (UI) should show only the information relevant to the current user task. For example, when a chemist defines an analytical method for a chromatographic run, only the parameters for the selected column and detector should be displayed on the UI, rather than the entire

set of all possible instrument parameters. This approach minimizes the possibility of human error in high-stakes environments where a minor error (such as a misconfigured parameter in a method) could lead to the loss of a valuable sample, unnecessary chemical consumption, and waste of time [14].

Modern LICS includes what is known in literature as "defensive design" - proactive mechanisms such as pre-run validations, confirmation prompts of critical actions, pre-run checks if hardware is in the correct state to execute an action, and real-time notifications of hardware malfunctioning during execution. These mechanisms are especially important given the fact that laboratory personnel often in challenging conditions: wearing protective glasses and gloves, managing multiple instruments simultaneously, or handling time-sensitive samples and reagents. Human-centric design also includes support for learning and error recovery, which could be achieved with clear error messages and logging mechanisms that would allow users to trace, understand the issue, and get the information on how to resolve it [15]. To build the intuitive and accessible (often called "user-friendly") software, it is best to include end users in the early phases of the software development lifecycle (SDLC) [14], which could be achieved through iterative prototyping and usability testing, which in this domain is not just a good practice, but rather a necessity.

2.2. MODULARITY AND LAYERED ARCHITECTURE

LICS must coordinate hardware control, data acquisition, post-processing, user interactions, and results displaying, while also functioning as a part within larger automated systems. This demands a shift from monolithic to modular architecture - a decomposed system with smaller, well-defined modules with clear responsibilities [16].

A typical layered architecture for LICS includes several well-defined layers:

- Hardware Abstraction Layer (HLA) - for low-level communication with hardware through various protocols (USB, Ethernet, TCP-IP, etc.);
- Instrument Control Layer - manages command sequencing and state machine logic;
- Data acquisition and processing layer - handles real-time signal collection, filtering, and buffering;
- Application or Business Logic Layer - implements the scientific methods, calculations, validation rules, and
- The Presentation Layer - provides the graphical user interface (GUI) [17].

The key benefit of this approach is the separation of concerns. By isolating hardware-specific communication in the HLA, the same upper-layer software can be reused across different instrument types. For example, the same data visualization and reporting modules can serve to display output of a UV-Vis spectrophotometer, FTIR spectrometer, or liquid chromatograph, with only the HLA layer and parts of the control layer being replaced [17]. This modularity also enables easier introduction of new modules later in the development of a specific software (for example, new detector types or new analytical methods), transition to modern communication protocols, and independent testing of each layer [18]. Furthermore, a shift towards service-oriented and microservices-based architecture in LICS in recent years enables better interoperability and flexible deployment, especially in modern, cloud-connected laboratories [19].

2.3. DETERMINISM AND HARDWARE-SOFTWARE SYNCHRONIZATION

Since LICS must manage physical processes in real-time, determinism - the guarantee that software will respond to hardware events within predictable, bounded time intervals, is one of the core design principles. This is particularly critical during data acquisition, where instruments such as mass spectrometers generate thousands of spectra per second, and any failure to capture data results in permanent data loss [17]. To achieve this, real-time instrument software typically divides tasks into two categories. Hard real-time operations, where missing a deadline can lead to data loss or hardware damage, are handled by firmware running on embedded systems. These components manage critical tasks like triggering detectors, signal sampling at precise intervals, and executing emergency stops. The second category are soft real-time requirements, where occasional delays are tolerable but undesirable, and are typically handled by the desktop application layer. These include updating the UI, refreshing instrument status indicators, and handling user interactions [20].

A critical aspect of this principle is fault tolerance and hardware safety. The software must be designed to handle hardware-level interruptions gracefully and safely. This is often implemented through state machine architecture, where the instrument is always in a well-defined state (e.g. Idle, Initializing, Running, Error…) and transitions between states follow strict safety rules.

The primary goal of this principle is to prevent a potential software error from damaging or destroying expensive equipment or valuable samples in case it happens. To maintain performance, LICS usually uses dedicated threads and buffering to separate high-speed data collection from slower requirements such as results display and storage [17], [20].

2.4. INTEROPERABILITY AND THE FAIR DATA PRINCIPLES

Early laboratory systems often created "data silos" - isolated repositories of experimental results stored in proprietary file formats that could not be accessed or interpreted by other systems. This directly impeded reproducibility and collaboration. Modern design addresses this problem by focusing on interoperability, the ability of LICS to seamlessly exchange data with other laboratory systems, including LIMS, SDMS, and ELN.

The key approach to achieve this is FAIR (Findable, Accessible, Interoperable, and Reusable) Data Principles, first published by Wilkinson et al. in 2016 [9]. In practice, this means that LICS should use standard formats such as Analytical Information Markup Language (AnIML) or JSON-LD, well-documented application programming interfaces (APIs), commonly REST-based, for integration with other systems and include metadata like instrument settings, timestamps, and operator identity [1] [9].

Besides FAIR principles, standardization initiatives such as the Standardization in Lab Automation (SiLA) consortium define open communication protocols between laboratory devices and software systems [21]. The SiLA 2 standard, based on HTTP/2 and Protocol Buffers, defines a service-oriented architecture that enables plug-and-play connectivity between instruments from different vendors, thus directly addressing the vendor lock-in problem. These initiatives together represent the "open by design" philosophy that is essential for creating integrated, data-driven workflows required by modern research [21].

2.5. DATA INTEGRITY, TRACEABILITY, AND REGULATORY COMPLIANCE

For LICS used in pharmaceutical, clinical, or other regulated settings, data integrity and regulatory compliance are mandatory requirements that must be embedded from its earliest system design phases [22], [23]. The primary framework for data integrity is the ALCOA+ set of principles given by the FDA in the 1990s and adopted globally by regulatory agencies, including the European Medicines Agency (EMA) and the World Health Organization (WHO) [24]. ALCOA+ requires data to be Attributable (traceable to the person or the system that generated it), Legible (readable and permanent), Contemporaneous (recorded at the time the activity was performed), Original (the primary record or a verified true copy), and Accurate (free from errors and correctly reflecting the observed result). The "plus" in the name extends these requirements to include Completeness, Consistency, Endurance (long-term) preservation, and Availability (retrievable when needed for the audit or inspection) [24].

At the regulatory level, the most influential standard is the FDA's 21 CFR Part 11, which establishes the criteria under which electronic records and signatures are considered equivalent to paper ones [5]. Its European counterpart is the EU GMP Annex 11 that imposes similar requirements [25]. Compliance with these regulatory standards requires secure authentication, role-based access, and immutable audit trails that record every event with a timestamp, user ID, and a reason for change. These features ensure that electronic data is as legally binding as traditional paper records. Importantly, while software can be "CFR-ready" by including the necessary technical requirements, true compliance is only achieved when the entire system (hardware, procedures, and personnel) is validated on-site, in its specific operational environment [5], [25]. This principle emphasizes that regulatory compliance is not a property of software alone, but rather the entire system in which it operates.

3. CHALLENGES IN LIFE SCIENCE SOFTWARE DEVELOPMENT

While design principles outlined in the previous section provide a blueprint for effective LICS development, their practical implementation is constrained by a set of technical, organizational, and regulatory challenges. These challenges originate from the characteristics of the domain: physical nature of the controlled processes, long laboratory hardware lifecycle, regulatory stringency, fragmented data standards, and growing cybersecurity threats.

3.1. BALANCING REAL-TIME PERFORMANCE WITH USABILITY

One of the major challenges in LICS is balancing between two conflicting needs: capturing real-time high-speed data without any loss and providing a smooth, responsive GUI simultaneously. Modern instruments can generate thousands of data points per second [17], and every data point matters. At the same time, users expect smooth, real-time visualization, responsive controls for monitoring the ongoing process, and to intervene if necessary.

This conflict originates from competing for system resources between time-critical data acquisition and computationally expensive GUI operations. On standard desktop operating systems, not designed for hard real-time guarantees, this can lead to data loss [17]. The common architectural solution to this issue is the separation of concerns across multiple threads or processes, where the data acquisition loop runs at a higher priority and captures data into a buffer, while the GUI thread reads from this buffer at a lower, visually acceptable refresh rate. In more sophisticated solutions, data acquisitions are delegated to dedicated embedded hardware, reducing the load on the main application [26]. Nevertheless, achieving this balance remains difficult. The buffering mechanisms must be able to handle data spikes, inter-thread communication must be designed to avoid deadlocks and race conditions, and the overall system must be tested under worst-case load conditions that are not the typical ones. As instrument capabilities continue to advance and data throughput increases, it is expected that this challenge will intensify rather than diminish [17], [26] .

3.2. LEGACY SYSTEMS AND BACKWARD COMPATIBILITY

Laboratory instruments usually represent capital investments, and they stay in use for a couple of decades [3]. It is not uncommon that modern research laboratories use hardware ranging from 20+ year- old to brand-new instruments, creating a need for backwards compatibility and a massive maintenance burden. This challenge appears in several forms. At the communication level, the software must support legacy protocols that are no longer standards or physical interfaces that require specialized adapters. At data level, older instruments may produce data in outdated or proprietary formats that require a tool (dedicated parsers and converters) to integrate with modern workflows and pipelines. At the operating system level, older instrument drivers may only work on outdated operating systems, which creates conflict with modern IT security rules [9], [27].

Vendor lock-in is one additional challenge in this context. Historically, many instrument manufacturers have developed closed, proprietary software ecosystems in which control software, data acquisition, data analysis, and data storage are tightly coupled and incompatible with third-party tools [9], [28]. Therefore, a laboratory using instruments from different vendors often needs to manage separate software platforms, each with its own specific requirements. In this constellation, replacing even one instrument requires expensive software migrations and revalidation efforts. While open standards like SiLA 2 aim to address this problem [21], their adoption is slow, and legacy systems will remain in use for many years to come.

3.3. REGULATORY VALIDATION BURDEN

In regulated industries, any software that affects product quality, patient safety, or data integrity must be formally validated. This process, called Computer System Validation (CSV), ensures that the system works as intended and comprises creating extensive documentation, including requirements, design specifications, and different testing protocols along with traceability matrices linking each requirement to its corresponding test [29], [13]. This process slows software updates significantly. While in non-regulated environments, development teams can release software updates frequently based on user feedback, in regulated environments, even small changes require and trigger a revalidation process [13]. Each change must go through formal review and approval, which creates a conflict with agile development practices that focus on fast, continuous improvements and favour waterfall-like practices with emphasis on upfront specification and sequential validation phases. Due to this, regulated laboratories often continue to use outdated software since updating can be too expensive and risky.

Recent efforts tend to improve this situation. New approaches, such as the FDA's Computer Software Assurance guidance and the ISPE GAMP 5 Second Edition (2022), promote a risk-based approach instead of heavy documentation [30], [29]. They support more flexible and iterative development methods. Nevertheless, the transition from traditional CSV to these modern approaches is still ongoing across the industry, and many organizations still follow old validation practices, which makes it difficult to adopt newer technologies.

3.4. INTEROPERABILITY AND DATA STANDARDS

Despite years of effort, laboratory data remain trapped in silos due to proprietary vendor formats that usually require the vendor-specific software to read. This makes it difficult to combine data from different systems, repeat experiments reliably, and manage the scientific findings across the whole laboratory [9].

Several standardization initiatives have attempted to address this problem. For example, the AnIML provides an open, XML-based format for storing analytical data, and as mentioned, SiLA 2 focuses on standardizing communication between devices [9], [21]. However, these standards are facing slow and incomplete industry adoption. The main reason is business-related: vendors have little motivation to support an open standard that would make it easier for customers to switch to competitors' solutions. Furthermore, even when standards are nominally supported, implementation inconsistencies across vendors can still prevent true interoperability. As a result, many laboratories rely on workarounds like custom file converters, manual data entry, or expensive middleware to connect different systems. This increases costs, risks, or errors, and makes it harder to build automated digital workflows that are expected in modern laboratories and required by regulations [9].

3.5. SECURITY CONCERNS

As laboratory systems connect to networks and cloud systems, cybersecurity risks are growing. Historically, instruments were isolated and protected mainly by physical access control. Today, they are connected for remote monitoring, data sharing, and integration with systems like LIMS, which increases exposure to cyber threats [31].

The laboratory environment is especially vulnerable due to several reasons. Many instruments run on outdated operating systems that no longer receive security updates and cannot easily be upgraded due to validation and compatibility issues. If such a device were connected to networks, it would represent a weak entry point that could be exploited to jeopardize the entire laboratory [31]. The consequences of cyberattacks in laboratories have broader concerns than data theft, such as ransomware attacks that block access to critical data, cause weeks of downtime, and potential loss of valuable data. Additionally, unauthorized manipulation of instrument settings and parameters could compromise the scientific results or affect patient safety [24].

Addressing these risks requires balancing between security and usability. Common measures include separating instrument networks, controlling user access, encrypting data and data transfer, and regular checks for vulnerabilities. New emerging standards are being applied to the laboratory context, and the SiLA consortium has established a dedicated working group on laboratory cybersecurity [21]. However, many organizations are still struggling to shift from treating instruments as isolated tools to managing them as a part of a connected IT system.

4. CONCLUSION

This paper reviews the key design principles and challenges in developing software systems for laboratory instruments in life sciences. As discussed, effective LICS must balance between human-centric design, deterministic hardware control, use modular and layered architecture to handle the inherent complexity of this domain, and include data integrity and regulatory compliance from the beginning. However, the practical implementation of these principles remains challenging. Development teams must deal with trade-offs between performance, usability, long lifespan of legacy systems, validation requirements, fragmented data standards, and increasing cybersecurity risks. Standardization efforts like FAIR and SiLA 2 help represent progress in the right direction, but their adoption is slow.

Looking ahead, the most significant change is the growing role of Artificial Intelligence (AI), which is no longer a theoretical prospect but a present reality. In 2026, agentic AI systems that can plan and run experiments are already starting to change the way laboratories work [32]. However, the effectiveness of AI in this domain still resides in the principles described in this paper and well-designed underlying software systems. Even the most advanced machine learning models cannot scale beyond pilot implementations without structured metadata, automated audit trails, interoperable communication protocols, and modular architecture. The ultimate goal for a laboratory instrument in this sense is the ability to disappear into the background, to become an invisible yet robust tool that will empower scientists to focus on their next discoveries while leaving the repetitive tasks to AI.

REFERENCES

[1] D. Romain, et al., ""Be sustainable": EOSC-Life recommendations for implementation of FAIR principles in life science data handling," *The EMBO Journal,* vol. 42, no. 23, p. e115008, 2023.

[2] F. Crick, What Mad Pursuit: A Personal View of Scientific Discovery, New York: Basic Books, 1988.

[3] S. Killcoyne and J. Boyle , "Managing Chaos: Lessons Learned Developing Software in the Life Sciences," *Computing in Science & Engineering*, vol. 11, no. 6, p. 20–29, 2009.

[4] L. Shi, M. Wang and X.-J. Wang, "Application of artificial intelligence in life science: Historical review and future perspectives," *Fundamental Research*, vol. 6, no. 1, pp. 20-27, 2024.

[5] U.S. Food and Drug Administration, "Electronic Records; Electronic Signatures. 21 CFR Part 11," U.S. FDA, Silver Spring, MD, 1997.

[6] Fortune Business Insights, "Life Science Software Market Size, Share & Industry Analysis," Fortune Business Insights, Pune, Maharashtra, India, 2024.

[7] H. K. Machina and D. J. Wild, "Laboratory Informatics Tools Integration Strategies for Drug Discovery: Integration of LIMS, ELN, CDS, and SDMS," 2013, vol. 18, no. 2, pp. 126-136.

[8] S. Mukherjee, et al., "Collaborative Robotics, Mobile Platforms, and Total Laboratory Automation in Clinical Diagnostics," *Diagnostics*, vol. 16, no. 4, p. 518, 2026.

[9] M. D. Wilkinson, et al., "The FAIR Guiding Principles for scientific data management and stewardship," *Scientific Data*, vol. 3, p. 160018, 2016.

[10] X.-K. Ma, et al., "Bioinformatics software development: Principles and future directions," *The Innovation Life*, vol. 2, no. 3, p. 100083, 2024.

[11] O. M. Kadioglu and T. Efferth, "Computational approaches streamlining drug discovery," *Nature*, vol. 616, pp. 673-685, 2023.

[12] N. H. Y. Yuen, C.-H. Li, K. C.-M. Kwok and R. Leung, "Laboratory Information Management System (LIMS) 4.0: A model to solve the current pain points for commercial electrical and electronic laboratories," *SAGE* Open, 2025.

[13] U.S. Food and Drug Administration, "General Principles of Software Validation; Final Guidance for Industry and FDA Staff," FDA, Rockville, MD, 2002.

[14] J. W. Pylvänäinen, G. Jacquemet and S. Marcotti, "Practical recommendations for developing software for life science applications," *Journal of Cell Science,* vol. 138, no. 5, p. jcs263711, 2025.

[15] S. Pelayo and M. S. Ong, "Human Factors and Ergonomics in the Design of Health Information Technology: Trends and Progress in 2014," *Yearbook of Medical Informatics*, vol. 10, no. 1, p. 75–78, 2015.

[16] M. Moussa, B. Abdelhalim, B. Ismail and A. B. Abderrahmane, "An Implementation of Microservices Based Architecture for Remote Laboratories," in *Cross Reality and Data Science in Engineering*, Springer, 2021, pp. 154-161.

[17] H. Turkmanovic, M. Karličić, V. Rajovic and I. Popovic, "High Performance Software Architectures for Remote High-Speed Data Acquisition," *Electronics*, vol. 12, no. 20, p. 4206, 2023.

[18] U. Kanewala and J. M. Bieman, "Testing Scientific Software: A Systematic Literature Review," *Information and Software Technology*, vol. 56, no. 10, pp. 1219-1232, 2014.

[19] J. Nau and M. Soll, "An Extendable Microservice Architecture for Remotely Coupled Online Laboratories," in *REV 2023: Open Science in Engineering, Thessaloniki*, Greece, 2023.

[20] K. Salamun, I. Pavić, H. Džapo and I. Čuljak, "Weakly Hard Real-Time Model for Control Systems: A Survey," *Sensors*, vol. 23, no. 10, p. 4652, 2023.

[21] D. Juchli, "SiLA 2: The Next Generation Lab Automation Standard," *Advances in Biochemical Engineering/Biotechnology*, vol. 182, pp. 147-174, 2022.

[22] D. Gokulakrishnan and S. Venkataraman, "Ensuring data integrity: Best practices and strategies in pharmaceutical industry," *Intelligent Pharmacy,* vol. 3, no. 4, pp. 296-303, 2025.

[23] B. Prasanna, P. Kothapalli and M. Vasanthan, "The Role of Quality Assurance in Clinical Trials: Safeguarding Data Integrity and Compliance," *Cureus*, vol. 16, no. 8, p. e67573, 2024.

[24] U.S. Food and Drug Administration, "Data Integrity and Compliance With Drug CGMP: Questions and Answers Guidance for Industry," FDA, Silver Spring, MD, 2018.

[25] European Medicines Agency, "Guideline on computerised systems and electronic data in clinical trials," EMA, Amsterdam, 2023.

[26] J. Camacho-Olarte and D. A. Tibaduiza Burgos, "Real Application For A Data Acquisition System From Sensors Based On Embedded Systems," *Engineering Proceedings*, vol. 2, no. 1, p. 25, 2020.

[27] L. F. Fávero, N. R. de Almeida and F. J. Affonso, "A Systematic Mapping Study on the Modernization of Legacy Systems to Microservice Architecture," *Applied System Innovation*, vol. 8, no. 4, p. 86, 2025.

[28] N. Rupp, R. Ries, R. Wienbruch and T. Zuchner, "Can I benefit from laboratory automation? A decision aid for the successful introduction of laboratory automation," *Analytical and Bioanalytical Chemistry,* vol. 416, no. 1, pp. 5-19, 2024.

[29] International Society for Pharmaceutical Engineering (ISPE), GAMP 5: A Risk-Based Approach to Compliant GxP Computerized Systems, Tampa, FL: ISPE, 2022.

[30] U.S. Food and Drug Administration, "Computer Software Assurance for Production and Quality System Software — Draft Guidance for Industry and Food and Drug Administration Staff," FDA, Silver Spring, MD, 2022.

[31] K. Tsiknas, D. Taketzis, K. Demertzis and C. Skianis, "Cyber Threats to Industrial IoT: A Survey on Attacks and Countermeasures," *IoT*, vol. 2, no. 1, pp. 163-186, 2021.

[32] H. Aboulsoud, M. K. Hassan, Y. M. Youssef and I. I. Nassef, "Artificial Intelligence in Laboratories: A Systematic Review of Existing Applications, Advantages, and Implementation Difficulties," *Advances in Artificial Intelligence and Machine Learning*, vol. 5, no. 2, pp. 3703-3713, 2025.