

Article

A Dual Model of Computational Scaffolding and Technical Architecture for Educational Environments in Applied Sciences: An IoT-Based Architectural Instantiation

Darwin Alulema ^{1,*}  and Estanislao Montenegro ²¹ Departamento de Eléctrica, Electrónica y Telecomunicaciones, Universidad de las Fuerzas Armadas ESPE, Sangolquí 171103, Ecuador² Telecommunications Engineering Program, Universidad de las Fuerzas Armadas ESPE, Sangolquí 171103, Ecuador

* Correspondence: doalulema@espe.edu.ec

Abstract

Educational environments for applied sciences often exhibit a disconnect between the pedagogical logic that structures learning activities and the technical architecture that implements them. In response to this problem, this study proposes a dual model that distinguishes a pedagogical architecture based on computational scaffolding from a layered technical architecture, while defining explicit coordination mechanisms between the two. The model is formulated at a conceptual level as a domain-independent architectural proposal and is instantiated in an IoT-based educational scenario to examine whether this articulation can be sustained coherently in a functional implementation. The empirical analysis does not address learning outcomes or pedagogical effectiveness. Instead, it evaluates the operational feasibility of the instantiation through usability evidence, concurrent load testing, and vulnerability analysis using OWASP ZAP. The results show that the proposed articulation can be implemented consistently under the tested conditions, with stable interaction flow and identifiable operating ranges. These findings support the architectural coherence and technical-operational feasibility of the analyzed instantiation, while leaving pedagogical validation of the model for future studies.

Keywords: Internet of Things (IoT); technological scaffolding; adaptive learning; virtual learning environment; engineering education



Academic Editors: Jordi Conesa Caralt and Antonio Sarasa Cabezuolo

Received: 5 March 2026

Revised: 1 April 2026

Accepted: 3 April 2026

Published: 4 August 2026

Copyright: © 2026 by the authors.

Licensee MDPI, Basel, Switzerland.

This article is an open access article distributed under the terms and conditions of the [Creative Commons Attribution \(CC BY\)](https://creativecommons.org/licenses/by/4.0/) license.

1. Introduction

The progressive digitalization of education in engineering and applied sciences has intensified the use of virtual environments, web platforms, and remote laboratories as mechanisms to expand access to practice-oriented learning experiences. These environments have been introduced in response to structural constraints, including limited availability of physical infrastructure, restricted access to specialized laboratories, and the geographical dispersion of students, by providing greater temporal flexibility and scalability in the execution of experimental activities [1,2]. (Feisel and Rosa, 2005; Ma and Nickerson, 2006).

However, this technological expansion has not been systematically accompanied by a comparable development of pedagogical models to guide their design and use. In many cases, the adoption of digital solutions follows a primarily instrumental rationale, creating a gap between instructional decisions defined at the pedagogical level and the technical

architectures that shape the educational experience. This gap is particularly critical in engineering education, where competence development relies on the direct articulation between conceptual understanding and action on real systems [3,4], within institutional contexts marked by persistent constraints such as limited laboratory availability, heterogeneous training conditions, budgetary restrictions, and variability in students' prior preparation [2]. In this context, several studies report the use of technology-mediated educational environments as alternatives to sustain applied training practices. Nevertheless, these reports reveal variability in pedagogical coherence, alignment with learning objectives, and process traceability, suggesting the need for frameworks that integrate pedagogical decisions with their technical realization [5,6].

Within this framework, scaffolding is understood as a pedagogical resource that organizes graduated supports introduced contingently and progressively withdrawn, with the aim of assisting learners when task complexity temporarily exceeds their initial capacities and of fostering autonomous control over the activity. From this perspective, scaffolding regulates cognitive load, structures problem-solving processes, and supports the gradual development of autonomy in complex learning contexts, particularly in scientific and technical education [7,8].

In this context, computational scaffolding is proposed as a mechanism to make pedagogical support explicit and operational within technology-mediated educational environments, integrating assistance directly into the system architecture. In this approach, supports are activated based on observable evidence of student interaction and are organized according to predefined criteria for gradation and progressive withdrawal, ensuring that assistance accompanies the execution of complex tasks without replacing learner agency. In STEM education contexts, computational scaffolding enables the articulation between conceptual understanding and action on real or simulated systems, provided that its design avoids both over-assistance and premature withdrawal of support [7,9–11]. However, some studies address this approach primarily from a pedagogical perspective, without detailing how these decisions are translated into technical architectures capable of sustaining them in a consistent, observable, and replicable manner.

In parallel, the Internet of Things (IoT) has emerged as a representative domain within applied sciences, as it integrates sensors, actuators, data processing, and communication mechanisms into distributed systems operating over real physical environments. In education, these characteristics have encouraged its adoption for the development of remote laboratories, hybrid platforms, and practice-oriented environments designed to reproduce experimental scenarios beyond traditional physical constraints [12,13]. Nevertheless, as with other technology-mediated educational environments, many of these proposals have focused primarily on technical feasibility and infrastructure accessibility, devoting uneven attention to the pedagogical design that structures student interaction with the system and to the integration of learning support during task execution [14].

To analyze the convergence between scaffolding, technology-mediated education, and IoT, this study conducted a systematic mapping of the literature together with a bibliometric analysis aimed at identifying relationships among the fields of scaffolding, educational technology, IoT, and STEM education. The bibliometric network presented in Figure 1 shows that scaffolding maintains a strong association with technology-mediated education and STEM domains, consistent with its widespread use as a pedagogical strategy in digital environments [15]. In contrast, its link to IoT appears weak and mediated by general terms such as education, without a direct and structured connection to training experiences based on IoT systems. This pattern suggests that, although IoT has been widely incorporated as technological infrastructure for educational experimentation, the explicit integration of computational scaffolding as a pedagogical support mechanism remains marginal and

insufficiently systematized in this domain. In IoT-based educational experiences, emphasis is placed primarily on enabling remote or hybrid practices through connected devices, web platforms, or simulation environments. Evaluation of these experiences frequently relies on indicators of acceptance, perceived usefulness, or student satisfaction, whereas the articulation between pedagogical design, system usability, and operational behavior during interaction remains under-specified. This approach confines the analysis of the educational process to perceptual outcomes, without examining in depth how instructional decisions are translated into concrete mechanisms of interaction and learning support within the IoT environment.

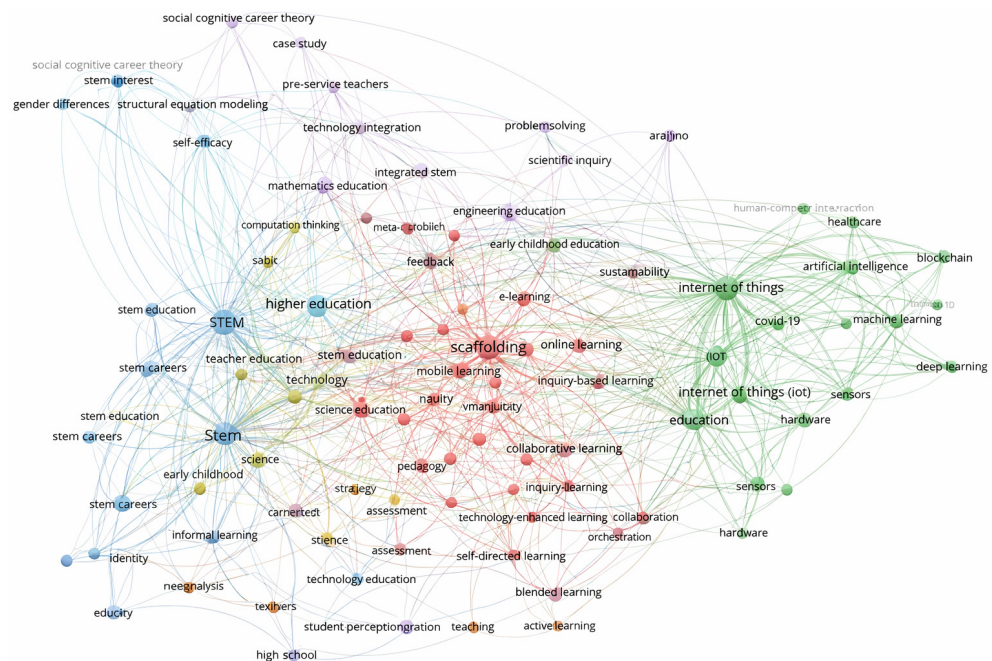


Figure 1. Bibliometric network by keywords.

Virtual and remote laboratories in engineering and applied sciences education are no longer studied merely as mechanisms for accessing experimentation. The discussion has shifted toward self-regulation, learning analytics, collaboration, and the pedagogical conditions that sustain meaningful interaction within the experimental environment [16–18]. This broadening of focus also appears in studies integrating remote laboratories with active learning strategies, where interest centers on activity coordination and the way the environment supports the student’s experimental practice [19].

The field of computational scaffolding, adaptive learning, and intelligent tutoring systems has followed a different direction. Here, the problem has concentrated on learner modeling and support personalization. The incorporation of artificial intelligence, affective adaptation, and machine learning has allowed for the adjustment of content, feedback, or difficulty based on the learner's performance, affective state, or cognitive demand [20–26]. However, in these developments, technical implementation is often treated as the system's functional support rather than as an architectural structure that is also part of the problem.

A similar situation occurs in proposals that reduce technological complexity through accessible platforms, visual programming environments, or low-barrier-to-entry tools. This decision facilitates system implementation and access but is frequently not accompanied by an explicit description of the instructional logic organizing the task sequence, the criteria regulating the provision of help, or the mechanisms through which the system responds to student actions during task execution [10]. Consequently, pedagogical support remains implicitly incorporated with limited analytical visibility, even in scenar-

ios where IoT infrastructure demonstrates technical feasibility for sustaining distributed experimental experiences.

This limitation is also observed in digital educational platforms that integrate services, interface, data persistence, and communication with external components. Although these developments resolve important functional aspects, they do not always make explicit how pedagogical sequencing, support logic, assessment, persistence, presentation, and interaction with external modules or physical devices are articulated within a single environment. This omission carries greater weight in technical and experimental scenarios, where the environment's functioning depends on the coordination among several components rather than a single adaptive mechanism or an isolated interface [6,27].

The formalization of the relationship between pedagogical regulation and technical implementation remains less explicit when both must operate coordinately within a single environment. This relationship constitutes the problem addressed by this study in applied sciences, where task sequencing, graduated support, assessment, persistence, presentation, and interaction with physical or simulated devices are part of the same operational organization. Table 1 synthesizes these limitations and delimits the position of the present research within the analyzed corpus.

The contribution of this work consists of formalizing that relationship through a dual model. The model is composed of a pedagogical architecture oriented toward task sequencing, scaffolding, and assessment, and a layered technical architecture responsible for persistence, coordination, presentation, and interaction with the physical module. The proposal does not introduce a new personalization algorithm or a variant of intelligent tutoring. It defines an architectural framework that explicitly organizes the translation of pedagogical logic into a functioning educational environment. Both architectures are defined independently and articulated through explicit coordination rules, allowing the educational environment to be analyzed as a coherent system with greater structural clarity. On this basis, the study examines whether this articulation can be sustained with architectural coherence and technical-operational consistency in a specific IoT-based instantiation.

The model is formulated as a framework independent of specific domains and technologies and is instantiated in an IoT-based educational scenario, used as a bounded case to examine its operational feasibility in an applied sciences context. This instantiation does not aim to evaluate learning outcomes or to validate the model from a pedagogical perspective. Instead, it analyzes whether the functional separation and coordination between the two architectures can be sustained consistently within a real technological setting. To this end, the following research questions are proposed:

RQ1. How can a pedagogical architecture based on computational scaffolding be explicitly articulated with a layered technical architecture for the design of educational environments in applied sciences?

RQ2. To what extent does an educational environment designed under a dual model maintain structural and operational coherence when instantiated in a specific technological scenario?

RQ3. What technical and interaction-based evidence can characterize the feasibility of an implementation derived from the dual model, without introducing domain- or technology-specific dependencies?

Methodologically, the study is structured as a design-oriented architectural investigation. Its purpose is not to test pedagogical effectiveness or learning outcomes, but to formulate a dual model, instantiate it in a concrete educational scenario, and examine whether the proposed articulation between pedagogical architecture and technical architecture can be sustained coherently under real implementation conditions. Under this logic, RQ1 is addressed through conceptual and architectural modeling, RQ2 through

the analysis of the instantiated environment as a bounded technological case, and RQ3 through technical-operational evidence derived from usability and interaction assessment, concurrent load testing, and vulnerability analysis. The IoT-based scenario was selected not as a domain-specific claim of superiority, but because it provides a demanding instantiation context for the dual model. In this type of environment, interaction with a physical module must remain coordinated with persistence mechanisms, logical regulation, interface signaling, and support activation during task execution. This makes it possible to examine whether the separation between pedagogical architecture and technical architecture can be sustained when the environment includes heterogeneous components, event-driven interactions, and state-dependent responses.

Table 1. Limitations identified in the analyzed studies and positioning of the present research.

Observed Limitations in the Analyzed Studies	Evidence in the Analyzed Corpus	Scope of the Present Research
Limited articulation between pedagogical scaffolding and educational environments supported by technical infrastructure in applied sciences	Scaffolding is widely documented in technology-mediated education and STEM domains, but its explicit integration with applied science environments that depend on more complex technical infrastructures remains partial. The relationship between pedagogical decisions and technical decisions is typically not formally structured at an architectural level [4,8,9,15].	A dual model is proposed that distinguishes between a pedagogical architecture based on computational scaffolding and a layered technical architecture. The study examines the coherence of their articulation through an instantiation in a representative applied science scenario.
Predominance of approaches centered on accessibility, connectivity, or technical feasibility	A significant portion of studies prioritizes remote access, connectivity, ease of use, or technological simplification through accessible platforms, visual programming, or low-barrier tools. However, these studies often omit an explicit description of the instructional logic that organizes task sequencing, support criteria, and system response during interaction [2,6,10,27].	The study defines an explicit pedagogical architecture that guides the design of the environment and is implemented in a differentiated manner from the technological platform used in the analyzed scenario.
Implicit treatment of pedagogical support in technically feasible environments	Even when the IoT infrastructure or the adopted platform demonstrates feasibility for sustaining distributed experimental experiences, pedagogical support is often incorporated implicitly and with limited analytical visibility. This makes it difficult to identify how support is regulated, how activity is sequenced, and how the environment responds to student actions [6,10,16,18,19].	The study makes explicit the pedagogical decisions that regulate interaction and articulates them with a technical architecture that operationalizes them through differentiated components.
Insufficiently formalized separation between pedagogical regulation and technical implementation	Works on adaptive learning, intelligent tutoring, and AI-based personalization have strengthened learner modeling and the adaptation of content, feedback, or difficulty, but technical implementation is usually treated as functional support rather than as part of an explicit architectural structure [20–22,25,26].	The study does not propose a personalization algorithm. Instead, it proposes an architectural framework that formalizes the relationship between pedagogical logic and technical implementation in a functioning educational environment.
Fragmented evaluations of technology-based educational environments in applied sciences	Evaluations tend to concentrate on perceptual metrics, acceptance, usability, or partial results, without jointly integrating the interaction experience, the system’s operational behavior, performance under load, and basic security considerations as part of the environment analysis [5,10,16,18].	The study analyzes operational behavior and interaction during the execution of the environment, including performance under load and basic security considerations, in order to examine the technical feasibility of the proposed instantiation, without yet proposing a pedagogical validation of the model.

The article is organized as follows. Section 2 develops the conceptual and methodological framework of the dual model, defining the pedagogical architecture based on computational scaffolding, the layered technical architecture, and the coordination rules between the two layers. Section 3 describes the instantiation of the model in an IoT-based educational scenario, clarifying how these architectural decisions are translated into a functional environment. Section 4 analyzes the implementation from an operational perspective, considering system behavior during interaction, performance under usage conditions, and basic security aspects, without addressing a pedagogical validation of the model. Finally, Section 5 discusses the results in relation to the proposed research questions, and Section 6 presents the conclusions and outlines directions for future work derived from the study.

2. Materials and Methods

This section presents both the conceptual formulation of the proposed dual model and the methodological basis of the study. From a research design perspective, the study is developed as a design-oriented architectural investigation. It first defines the conceptual model and then instantiates it in a bounded IoT-based educational scenario for technical-operational analysis. Accordingly, the section establishes the pedagogical architecture, the layered technical architecture, and the coordination rules between the two, which later serve as the basis for the implemented instantiation and its evaluation.

This articulation does not imply that pedagogical criteria are dissolved into the technical system. Rather, the dual model preserves an explicit distinction between the pedagogical definition of regulation and the technical mechanisms used to operationalize it within a concrete environment.

At the pedagogical level, the architecture is structured around a computational scaffolding system that organizes instructional design through graduated supports. Their activation is determined by objective evidence of performance, such as repeated errors, atypical response times, or omissions. These indicators function as inputs for regulating support and for its progressive fading as task mastery increases. This approach is grounded in Cognitive Load Theory, which posits that reducing extraneous cognitive load directs mental resources toward conceptual understanding and problem solving in applied sciences [28,29], and aligns with adaptive scaffolding frameworks that incorporate explicit activation and fading criteria without abrupt withdrawal of support [9]. From this definition, the pedagogical architecture is organized into three levels: (i) authentic tasks, which define sequences of challenges, expected products, and achievement criteria; (ii) graduated supports, including hints, feedback, and review elements with manual or automatic activation and progressive fading; and (iii) integrated assessment, which jointly considers learning, user experience, and system operability as dimensions observed during activity execution [15].

The technical architecture is defined as a structural component of the educational model that sustains its articulation with the pedagogical architecture. It is organized into four functional layers. The physical layer establishes interaction with devices and systems characteristic of applied science practices. The persistence layer manages data storage, interaction logs, and activity-related results for traceability purposes. The logical layer defines processing flows and rules that connect the pedagogical sequence with the technical execution of tasks, including scaffolding activation. Finally, the presentation layer specifies user interaction mechanisms through visualizations and feedback elements that make system states and associated supports explicit.

2.1. Pedagogical Architecture of Computational Scaffolding

The pedagogical architecture defines the instructional organization of the proposed dual model by establishing how activities are structured, how pedagogical intervention is regulated, and how evidence generated during task execution is conceptually organized in applied science environments. Its formulation does not depend on a specific domain or technology; rather, it describes a transferable instructional logic aimed at maintaining coherence among student actions, system-provided supports, and the evaluation criteria associated with each activity.

Under this definition, the pedagogical architecture consists of three coordinated layers: authentic tasks, graduated supports with activation rules, and integrated assessment. Figure 2 summarizes this organization through a functional arrangement of the layers across the learning process. At the base are the authentic tasks, conceived as a progression of activities ranging from initial exploration to integration and decision-making. Layered above this progression are the graduated supports, defined as contingent assistance mechanisms whose activation and fading are regulated according to observable evidence during interaction. At the top level, integrated assessment groups evidence associated with outcomes, processes, and execution conditions, establishing a framework for continuous and traceable observation of the learning trajectory.

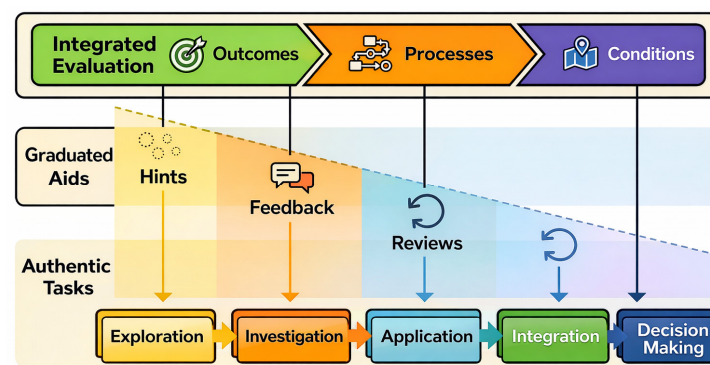


Figure 2. Pedagogical support flow in the computational scaffolding model.

2.1.1. Authentic Task Layer

The first layer corresponds to authentic tasks and constitutes the starting point of the learning process within the pedagogical architecture of the model. These tasks are conceived as activity scenarios in which students interact with verifiable phenomena, manipulate data, and produce observable results. Learning is thus situated in plausible contexts of the application domain, avoiding its reduction to abstract exercises.

From an operational perspective, an authentic task presents three essential characteristics. First, it maintains an explicit connection to real data or phenomena, either through direct measurements or through the use of reliable repositories. Second, it requires the generation of an observable and verifiable product, such as data records, visualizations, decision rules, or brief reports, that is subject to objective evaluation. Third, it incorporates explicit achievement criteria defined in advance and aligned with the learning objectives, enabling unambiguous collection of performance evidence during the activity.

The design of authentic tasks follows complementary criteria that guide their sequencing within the model. Contextual authenticity requires that activities reproduce plausible conditions of the application field. Cognitive gradation establishes a planned progression of complexity, with clear task dependencies and explicit prerequisites. Progressive transfer structures the transition from initial exploratory actions toward the integration of concepts and procedures. It defines a sequence of challenges that advances from interaction with

variables and data to the integration of components and evidence-based decision-making, while maintaining a controlled evolution of complexity.

2.1.2. Graduated Support Layer and Activation Rules

The second layer corresponds to graduated supports and defines how the pedagogical architecture regulates system intervention during task execution. Its function is to determine when, how, and to what extent the environment should provide assistance according to observable evidence of student difficulty, while avoiding both over-assistance and absence of support. Supports are therefore treated not as continuous or predetermined elements, but as contingent interventions activated in response to the student's performance during interaction.

Within this layer, three forms of intervention are distinguished. Brief hints guide immediate action without solving the task. Formative feedback makes errors explicit or confirms correct responses in order to orient correction. Guided reviews reconstruct the procedure when difficulty persists or progress remains stalled. These interventions are organized as graduated levels of support rather than as isolated responses, allowing the environment to regulate assistance progressively throughout task execution.

To make this progression explicit, the support logic can be represented through a bounded rule-based structure. Let S denote the current scaffolding level assigned to a student during a task, where $S \in \{0, 1, 2, 3\}$. In this representation, $S = 0$ indicates autonomous task execution without additional support, $S = 1$ corresponds to brief hints, $S = 2$ to guided formative feedback, and $S = 3$ to intensive support through guided review. This formulation makes explicit that support is not binary, but organized as a finite progression of pedagogically differentiated intervention states.

Support activation depends on observable thresholds derived from interaction. Let E denote the persistence of errors during task execution, T the response time relative to the expected threshold for the same task segment, and P the recent performance trend observed across attempts. Under this structure, activation occurs when the student shows persistent difficulty in task resolution. The transition from $S = 0$ to $S = 1$ is triggered when repeated errors are detected, when relevant omissions occur, or when response time exceeds the expected range without successful completion. Escalation from $S = 1$ to $S = 2$, and from $S = 2$ to $S = 3$, occurs when difficulty persists after prior support has already been provided, as evidenced by continued errors, unstable task resolution, or absence of effective progress.

This layer is further defined by parameters that delimit the conditions of intervention, including observation windows, repetition limits for the same type of support, and withdrawal criteria based on evidence of task mastery. These parameters prevent the system from reacting to each action as an isolated event and allow assistance to be regulated over sequences of interaction rather than over disconnected responses.

Progressive fading follows the same rule-based logic. Transitions from $S = 3$ to $S = 2$, from $S = 2$ to $S = 1$, and from $S = 1$ to $S = 0$ occur when performance stabilizes, errors decrease, and task execution becomes progressively more autonomous. In operational terms, the support state must remain persistent across interaction events and attempts so that the environment can regulate assistance coherently over time.

This formalization does not introduce a predictive or probabilistic learner model. Its purpose is to make explicit the rule-based logic through which the pedagogical architecture defines support progression and the technical architecture operationalizes that progression during interaction.

2.1.3. Integrated Assessment Layer

The third layer corresponds to integrated assessment and is defined as the component of the pedagogical architecture that consolidates the evidence generated throughout student interaction with the environment. This layer is formulated to observe the complete interaction process, enabling a traceable reconstruction of the learning trajectory.

Evidence is organized into three complementary dimensions. The first dimension relates to conceptual and procedural performance and includes correct solutions, decisions consistent with achievement criteria, and products aligned with the objectives of each task. The second dimension focuses on interaction and work processes, considering actions performed, corrections made, support requests, and completion of procedural steps. The third dimension groups general execution conditions of the learning environment that are necessary to interpret the work experience in terms of continuity and process stability.

Integrated assessment is defined as a scheme that collects evidence at different stages of the learning activity. It establishes prior references linked to objectives, observation points throughout the task sequence, and comparison criteria at the conclusion of the activity. Within a unified framework, it integrates learning outcomes, interaction processes, and operational conditions of the environment.

2.2. Technical System Architecture

The technical architecture is defined as the organizational structure that operationalizes the instructional logic of the dual model without relying on a specific technology. Its function is to support, at a conceptual level, the execution of authentic tasks, the activation of graduated supports, the logging of evidence, and integrated assessment, so that pedagogical decisions can be translated into observable and traceable system behaviors.

Under this definition, the technical architecture is organized into four interdependent layers (Figure 3). The physical layer establishes interaction with devices and communication mechanisms. The persistence layer defines the storage of data, events, and records required for traceability. The logical layer articulates the flows and rules that connect the pedagogical sequence with technical execution. Finally, the presentation layer defines the means through which interaction is materialized through visualizations and feedback.

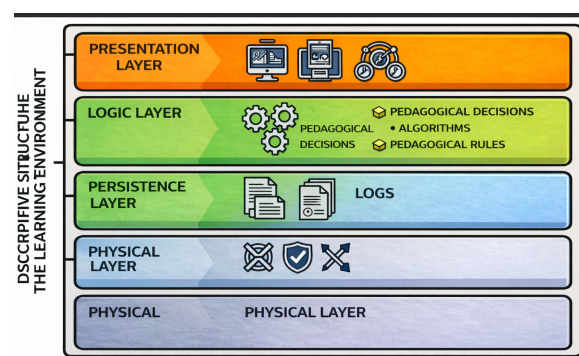


Figure 3. Physical architecture.

2.2.1. Physical Layer

The physical layer is defined as the level of the model where the means that enable student interaction with phenomena inherent to the application domain are located. This layer establishes the point of contact between the learning activity and the physical or simulated systems on which authentic tasks are built. It is not conceived as a specific set of devices, but as a functional category that encompasses any infrastructure capable of producing observable data, responding to student actions, and sustaining interaction cycles consistent with the learning objectives.

The physical layer is formulated so that students' technical actions correspond to perceptible consequences that can be interpreted as evidence within the learning process. It is also defined in direct relation to the authentic task layer, since each task presupposes the existence of a physical or simulated environment that enables data acquisition, action execution, and observation of results, while maintaining operational clarity and minimizing extraneous cognitive load associated with interaction with the environment.

2.2.2. Persistence Layer

The persistence layer is defined as the component of the technical architecture responsible for organizing and storing the data generated during interaction between the student and the system. This layer establishes the conceptual mechanisms required to support traceability of the learning process, enabling structured logging of task progress, support activation, and evidence associated with integrated assessment.

The persistence layer defines an information organization model that makes it possible to link each relevant event to its pedagogical context. Within this model, student actions, system decisions, and artifacts generated during the activity are related consistently, independently of the specific storage technology.

The persistence layer is defined in direct articulation with the graduated support layer and the integrated assessment layer described in Section 2.1. Within this articulation, each support delivery, each state transition, and each observable outcome are conceived as loggable events that enable reconstruction of the student's learning trajectory within the model.

2.2.3. Logical Layer

From an architectural perspective, the logical layer is conceived as the operational coordination core of the technical architecture. Its function is to process interaction-derived events, apply previously defined decision rules, and trigger the corresponding system responses during task execution. These rules are not pedagogically defined within the logical layer itself. Rather, they operationalize criteria previously established in the pedagogical architecture, including support activation conditions, progressive fading conditions, observation parameters, and evaluation-related thresholds.

Under this definition, the pedagogical architecture remains responsible for specifying the instructional meaning of the task sequence, the rationale for support delivery, and the criteria under which regulation occurs. The logical layer, in turn, is responsible for implementing those criteria in a reproducible and technically executable form within the environment. In this sense, the logical layer does not constitute the pedagogical architecture; it functions as the technical mechanism through which pedagogically defined decisions are enacted, recorded, and coordinated during interaction.

This distinction is essential to the dual model. The pedagogical architecture defines what kind of regulation is required and under what instructional conditions it should occur, whereas the logical layer determines how that regulation is executed within the system through flows, events, and state-dependent responses. The separation between the two is therefore not a claim of isolation, but an explicit differentiation between pedagogical definition and technical operationalization.

2.2.4. Presentation Layer

The presentation layer is defined as the component of the technical architecture that specifies how the system communicates to the student the state of the activity, the tasks in progress, and the associated supports. It is not limited to information visualization; rather, it is conceived as the means through which the pedagogical logic of the environment becomes perceptible during interaction.

Under this definition, principles are established to govern interface organization, the signaling of relevant actions, and the visibility of supports. These principles enable students to understand what is expected at each stage of the task, the rationale behind the activation of a support, and the subsequent actions required, while avoiding elements that increase extraneous cognitive load [30].

In architectural terms, this layer is explicitly linked to the logical layer, which determines the conditions under which supports are displayed, and to the pedagogical architecture, which defines the instructional meaning of each intervention. The interface is thus configured as a structural component of computational scaffolding.

2.2.5. Descriptive Structure of the Learning Environment

The technical architecture incorporates a descriptive structure of the learning environment that enables coherent organization of content, instructional supports, and evidence from the learning process. This structure is formulated as a declarative scheme that makes explicit the relationships among tasks, resources, supports, and assessment criteria within the dual model, without being conceived as a database design in a strict sense or as a specific implementation.

From an architectural perspective, this structure plays a transversal role across the physical, persistence, logical, and presentation layers. With respect to the pedagogical architecture, it enables authentic tasks, graduated supports, and integrated assessment to be expressed as organized and reusable descriptions. With respect to the technical architecture, it provides the conceptual framework required to interpret the association among content, tasks, supports, and relevant events during interaction.

The descriptive structure organizes environment elements into functional categories aligned with the dual model, such as curricular entities, learning content, pedagogical supports, and evidence generation points. These categories do not introduce system behavior; instead, they delimit which elements belong to the environment, how they are classified, and under what conditions they acquire meaning within the learning process.

Conceptually, this approach is analogous to description schemes used in other domains, such as Thing Descriptions in the Web of Things, where the declarative specification of capabilities, properties, and interactions enables decoupling system description from its concrete implementation [31,32]. Within the dual model, this principle is transferred to the pedagogical-operational layer of learning, separating the structural description of tasks, supports, and evidence from the technical mechanisms that execute them. In this way, instructional decisions can be expressed explicitly without being rigidly encoded in the system logic.

2.3. Curriculum Design and Parameterization

Curriculum design defines how the learning experience is organized across the pedagogical and technical architectures of the dual model. It is not conceived as an independent architecture, but as a parameterization mechanism that enables these architectures to be instantiated in specific learning experiences. Within this framework, the curriculum is not treated as an external system component; rather, it functions as an operational configuration that determines the learning sequence, expected products, and predefined support points while maintaining coherence with the authentic tasks, graduated supports, and integrated assessment defined in the model.

From this perspective, curriculum design fulfills a dual function. On the one hand, it establishes the learning progression that guides students throughout the applied sciences learning process. On the other hand, it defines a set of parameters that allow this progression to be interpreted by the technical architecture without requiring modifications to its internal logic. In this way, the curriculum operates as a link between pedagogical intentionality and observable system behavior.

2.3.1. Unit-Based Structure and Learning Progression

Within the dual model, the learning experience is defined through a unit-based organization that structures progression from the introduction of foundational concepts and components to the integration of subsystems and the resolution of more complex challenges. This organization does not describe a particular course; rather, it represents a structural principle of curriculum design that enables the sequencing of activities and the establishment of conditions under which students may advance in the learning process.

Each unit is characterized by the definition of concise learning objectives, observable products, and explicit dependencies on prior units. Progression is not formulated as a purely temporal sequence, but as a set of access conditions that regulate when a unit becomes approachable based on demonstrated advancement, maintaining coherence between pedagogical sequencing and the learner's operational capacity.

The unit structure is articulated with the pedagogical architecture through systematic associations among units, authentic tasks, and achievement criteria. This articulation is expressed both in the conceptual organization of the course and in the conditions under which the environment enables activities and resources, without introducing decisions external to the learning process.

2.3.2. Core Content, Resources, and Predefined Support Points

Each unit integrates core content and learning resources defined as part of the curriculum design of the model. These resources are not conceived as a static repository; rather, they are associated with states of progression and instructional need, so that their availability is linked to the student's advancement within the unit.

Within this framework, predefined support points form an explicit component of the curriculum rather than being incorporated reactively. For each unit, reinforcement resources, conceptual hints, and review materials are specified, with their activation defined according to evidence of difficulty, such as repeated errors, stagnation in progress, or persistent challenges. This anticipation enables graduated supports to be integrated coherently with the content and learning objectives of each stage.

The correspondence among units, content, and supports is defined as a principle of curriculum design, whereby supports are formulated in direct relation to the content of the current unit and to the observable evidence generated during activity execution.

2.3.3. Assessment Aligned with Activity and Support Logic

Assessment is defined in coherence with unit-based progression and the observable products generated during authentic tasks. It is not conceived as an isolated event, but as a component integrated into the activity flow and articulated with the system's support logic.

Each unit declares achievement evidence linked to learning objectives and defined tasks. This evidence enables interpretation of student performance as part of a continuous trajectory, in which outcomes are considered together with interaction conditions and supports activated during activity execution.

Within this framework, incorrect responses or incomplete performances are formulated as operational evidence that may be linked to reinforcement or review resources associated with the same unit. Assessment is thus defined as a decision mechanism expressed through observable events, coherent with the technical architecture and with the rules governing support activation and logging.

2.3.4. Curriculum Parameterization and Multi-Course Support

Curriculum parameterization is defined as the mechanism through which the formative structure of the model can be expressed and reused without modifying the system's

general logic. Conceptually, this parameterization is formulated through the definition of units, content, resources, and assessment criteria as data sets interpreted by the system during execution of the learning experience.

Under this definition, changing the course or domain is expressed as a reconfiguration of curricular parameters while maintaining constant scaffolding mechanisms, support activation logic, and evidence logging established in the model. The curriculum structure is therefore conceived as a domain-independent description that can be associated with different learning experiences.

Within the dual-model framework, curriculum parameterization establishes associations among units, resources, and evidence, ensuring that each element is linked to a specific unit and that its activation follows the defined pedagogical and technical logic. The curriculum is thus defined as a set of declarative conditions that regulate the organization of the learning experience within the system.

3. Results

3.1. Implementation of the Dual Model in an IoT Scenario

The implementation of the dual model is presented as a concrete instantiation applied to a teaching scenario in the Internet of Things (IoT). The purpose of this section is to show how the pedagogical and technical architectures are materialized in a functional system, maintaining an explicit separation between the conceptual model and the technologies employed in its implementation. IoT is adopted as the application domain due to its capacity to provide experimental contexts with observable variables. This choice does not restrict the model to this domain or to a specific set of tools. Within this delimitation, the implementation is described from two complementary perspectives: the pedagogical perspective, which specifies authentic tasks, graduated supports, and integrated assessment; and the technical perspective, which operationalizes these decisions through functional mechanisms.

The implementation is realized in a learning environment in which the tasks assigned to students rely on external experimental experiences and on data derived from those experiences. The system organizes the learning sequence, regulates access to activities and content, manages the provision of pedagogical supports, and records process evidence. In parallel, the physical module operates as a decoupled instructional resource used for task execution without forming part of the system's core. Consistent with this separation, the technical layer incorporates, where appropriate, data acquisition and validation, decision-making regarding progression, persistent event logging, and presentation of system status to the student.

3.1.1. Implementation of the Pedagogical Architecture of the Dual Model

The pedagogical architecture is implemented as a set of operational decisions that define which actions the student performs, when support is provided, and which evidence is recorded during the learning process. In this instantiation, learning is organized around authentic tasks executed on an IoT instructional module. Physical outputs and data obtained during practice constitute the direct input for evaluation and progression decisions within the platform.

In the developed use case, an authentic task is defined as an activity that requires physical interaction with the module and cannot be completed without executing real technical actions on the system. Under this criterion, the implemented tasks include:

- Reading environmental variables through physical sensors integrated into the module:
 - Temperature and humidity reading using a DHT11 sensor.
 - Motion detection using an HC-SR501 sensor.
 - Distance measurement using an HC-SR04 ultrasonic sensor.
 - Light level measurement using an LDR sensor.
- Control actions on actuators:

- Turning an LED on and off.
- Verifying state changes after action execution.
- Execution of operational sequences:
 - Performing repeated readings from the same sensor.
 - Comparing obtained values with expected ranges.
 - Observing system behavior variations under changing conditions.

Each of these tasks generates real and observable values dependent on student interaction with the module. No simulated or preloaded data are used. These tasks are integrated with an assessment process within the web platform. For each executed task, students respond to questions based on: (i) measured sensor values, (ii) observed behaviors in the physical system, and (iii) system states achieved after executing control actions. Practice and evaluative activity therefore constitute a single pedagogical unit.

Based on this structure, pedagogical support is implemented through graduated supports functioning as computational scaffolding during task execution. Three support levels are defined according to the degree of explicitness and type of resource provided:

- Level 1–Basic Support:
 - Brief orientation texts.
 - Reminders of task objectives.
 - General indications regarding procedural steps.
- Level 2–Reinforcement Support:
 - More detailed procedural explanations.
 - Additional examples related to the task.
 - Audiovisual materials reinforcing content.
- Level 3–Intensive Support:
 - More structured guidance for activity resolution.
 - Guided retries when difficulties persist.

Supports are dynamically activated based on observable indicators recorded by the platform during interaction. The implementation considers the following indicators:

- Low results or results outside predefined passing ranges.
- Extended response times in assessment activities.
- Repeated errors within the same task.
- Explicit student requests for assistance.

Progressive fading of support is implemented as an operational rule of the system. Supports are reduced in level or withdrawn when one or more of the following conditions are met:

- Sustained improvement in achieved scores.
- Decreased response times.
- Consistent task completion without repeated errors.
- Explicit student decision to discontinue support.

Assessment is implemented as a continuous process articulated with tasks and supports. The platform records interaction trajectory evidence and uses it to make predefined decisions regarding unit unlocking, support activation, and regulation of pedagogical guidance. For each student, the system records, among other data:

- Results per task and per unit.
- Achieved performance ranges.
- Progress and lock states.
- Response times.

- Platform interaction events.
- Activated support levels.

This evidence supports pedagogical regulation decisions, enabling adjustment of support levels and access to new tasks according to recorded progression.

3.1.2. Implementation of the System's Technical Architecture

The technical architecture operationalizes the dual model through the coordinated integration of physical, logical, persistence, and presentation components. It is organized as a layered, non-monolithic structure that separates functions, responsibilities, and decision levels, assigning each layer a defined role within the learning environment.

Figure 4 presents the implemented technical architecture, organized into four layers: physical, persistence, logical, and presentation. The physical layer corresponds to the IoT module and acts as experimental support for the tasks, without incorporating evaluation or automatic tracking. The persistence layer stores user states, content, progress, and assessments, enabling continuity across sessions. The logical layer centralizes progression control, evaluation, and scaffolding regulation, translating interaction events into operational decisions. The presentation layer implements the interface and is limited to capturing actions and displaying states, without embedding decision rules. It communicates with the logical layer through structured messages.

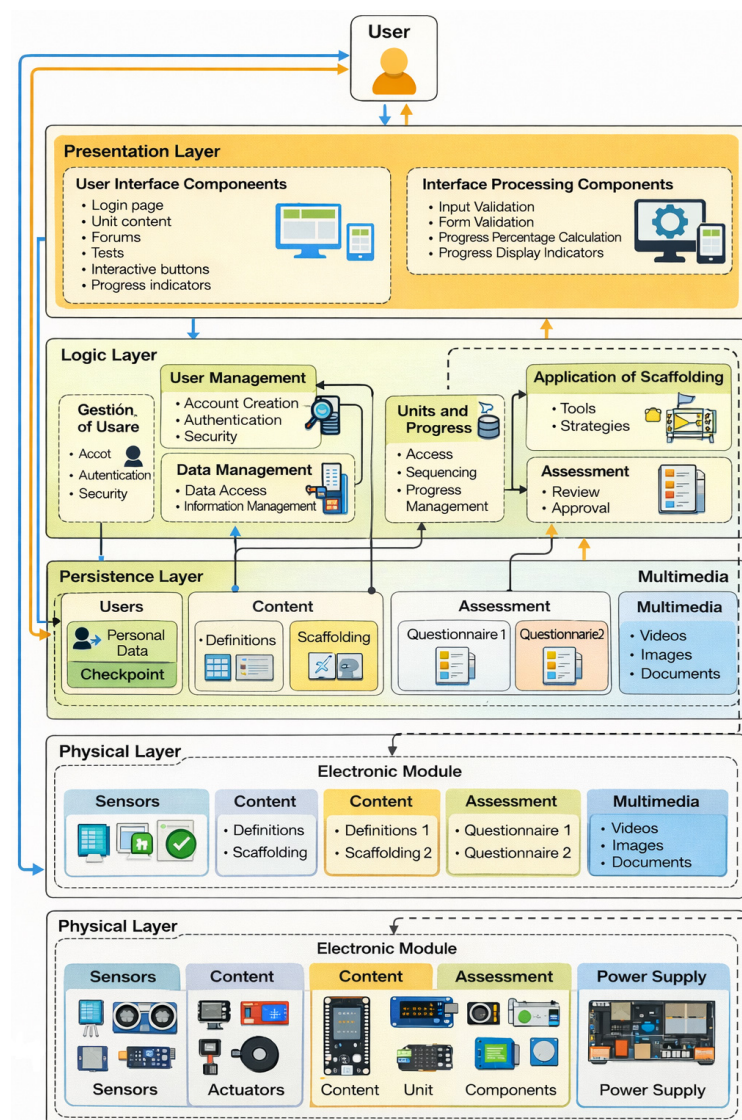


Figure 4. Implemented physical architecture for IoT.

Physical Layer

The physical layer is implemented as an instructional IoT module designed for experimental activities involving real physical variables. The module integrates sensors and actuators on a printed circuit board (PCB) and operates autonomously, decoupled from the web platform (Figure 5). It does not incorporate connectivity with the course system, does not transmit data automatically, and does not execute evaluation, progression control, or support provision. It is limited to physical-level acquisition and control. Interpretation of results and decision-making remain with the student and the upper system layers.

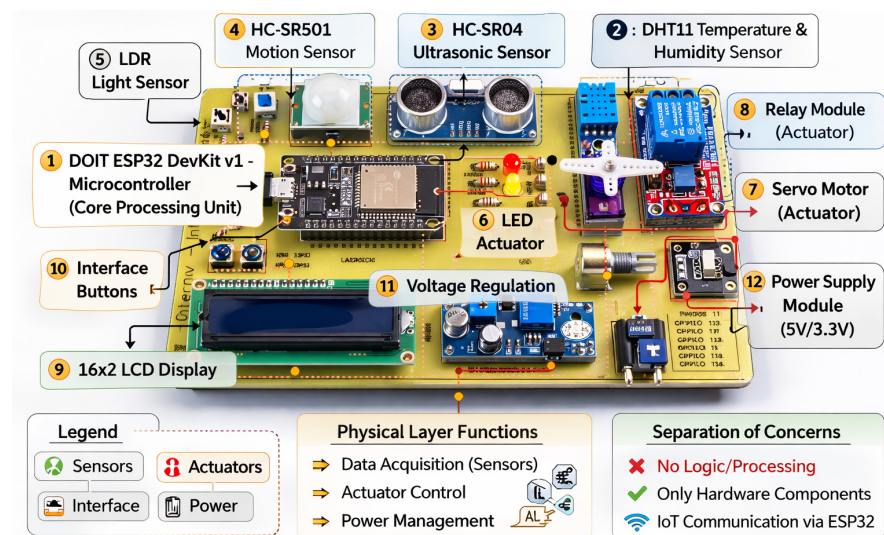


Figure 5. IoT instructional board.

The core of the module is a DOIT ESP32 DevKit v1 (Figure 5). The ESP32 functions as the acquisition and control unit, managing digital inputs, analog inputs, and digital outputs associated with the devices defined for the activities. The firmware executes signal-reading routines and output activation procedures, without pedagogical rules or progression-related decision logic.

The electronic design integrates all predefined sensors and actuators into a single unified circuit. The PCB defines fixed and permanent interconnections, without temporary wiring or variable configurations (Figure 5). This decision reduces connection errors and enables reuse of the module with a consistent configuration across sessions.

The module integrates digital and analog sensors with explicit microcontroller pin assignments (Figure 5):

- DHT11 (temperature and humidity): GPIO 4 (digital input).
- HC-SR501 (motion): GPIO 15 (digital input).
- HC-SR04 (ultrasonic): trigger on GPIO 5; echo on GPIO 18.
- LDR (light level, resistive divider): GPIO 34 (analog input, ADC).
- LED (actuator): GPIO 2 (digital output).

This allocation separates digital inputs, analog inputs, and digital outputs, avoiding electrical conflicts during simultaneous device operation.

Persistence Layer

The persistence layer is implemented using a MySQL relational database management system. This layer stores operational states generated by student interaction with the learning environment and supports reproducible decisions related to evaluation, scaffolding,

and progression control executed in the logical layer. Figure 6 summarizes the relational data model and the functional grouping of its tables.

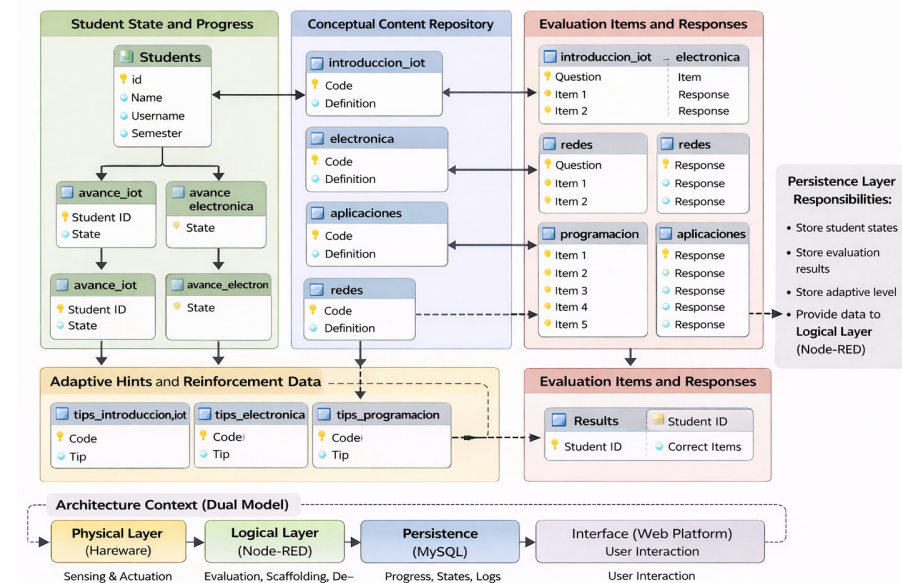


Figure 6. Relational data model of the persistence layer.

The implementation reflects the system's data model and represents entities, relationships, and states derived from execution of the dual model. The database does not store physical signals or raw values from the IoT module, since that component remains decoupled. Instead, it contains interpreted task results, student responses, evaluation states, and variables governing progression and scaffolding.

The schema is organized into structural tables aligned with four functions. First, a Students table acts as the primary entity linking all other records. Second, the conceptual repository maintains instructional units and their definitions (e.g., *introduccion_iot*, *electronica*, *aplicaciones*, *redes*). Third, evaluation tables record both unit-level items and individual responses, as well as consolidated results per student and per unit. Fourth, support tables (*tips_*) store reinforcement codes and messages, together with variables associated with support level and activation, which feed adaptive rules in the logical layer.

This table set enables reconstruction of each student's learning trajectory and maintenance of coherent states across sessions, without imposing rigid interface sequences, since activation and progression are governed through persistent states queried by the logical layer.

Logical Layer

The logical layer implements the processes that coordinate student interaction with the learning environment and execute decision rules for evaluation, scaffolding, and progression control. This logic is developed through flows in Node-RED. Each student action generates a message (msg), and system state is maintained between nodes using context variables (flow), enabling continuity across pages, attempts, and sessions without transferring decision rules to the interface. Figure 7 synthesizes this organization into three functional zones and shows the complete flow path.

In Zone 1 (Events/Interface), student interaction occurs. Responses, support requests, or errors generate events that enter the logical layer. The figure shows this direct connection to Zone 2 (Logical Layer–Node-RED), where the Scaffolding Management module is activated.

Within the logical layer, the Evaluation Engine computes the vector of correct responses from flow.responses, comparing each item with the pattern retrieved from the database (Pattern Load). This processing produces an aggregate count feeding the decision node “Exceeds Threshold?”. The rule operates using explicit numerical thresholds and classifies outcomes as approved or not approved.

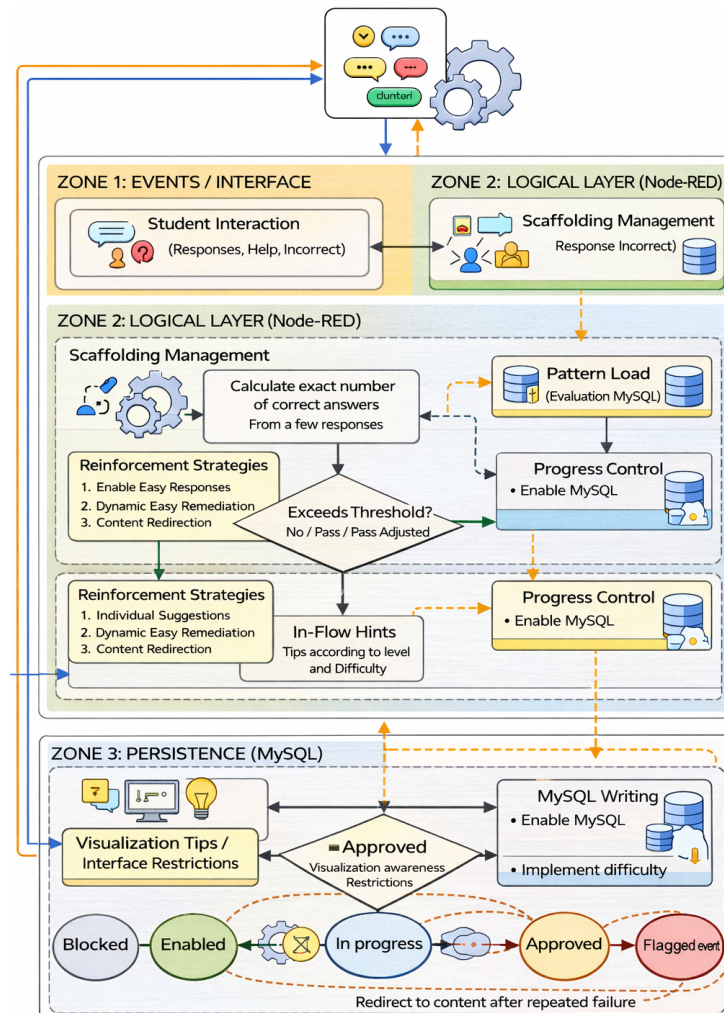


Figure 7. Operational flow of the logical layer in the scaffolding system.

If performance exceeds the threshold, the flow activates the Progress Control module, which emits enabling signals to the interface and persists the corresponding state. If not, reinforcement strategies are activated, also represented in Figure 7, including audiovisual suggestions, dynamic remediation, and content redirection. These strategies do not modify the core evaluation engine; they are integrated as conditional branches within the same flow.

Scaffolding is regulated through a discrete support-level variable (flow.level), visible in the Scaffolding Management block. This state propagates before interface rendering and conditions action availability, such as restricting navigation when the level corresponds to reduced scaffolding.

Zone 3 (Persistence–MySQL) shows the writing point for aggregated results, progression status, perceived difficulty, and support activation. The figure demonstrates that persistence does not execute decision logic; it receives states computed by Node-RED and stores them for later retrieval. The lower diagram illustrates global progression states

(Blocked, Enabled, In progress, Approved, Not Approved), which depend on signals emitted by the logical layer rather than rigid interface sequences.

The structure represented in Figure 7 separates events, processing, and persistence. Solid lines indicate direct module connections, and dashed lines indicate indirect relationships, reinforcing that process control resides in flow logic rather than presentation.

Presentation Layer

The presentation layer implements the web interface that materializes operational interaction between the student and the system. It is designed as a set of views and graphical components responsible for capturing user actions, visualizing states computed by the logical layer, and reflecting persisted progress, without incorporating evaluative processing or autonomous decision rules. Figure 8 presents a representative dashboard view, showing hierarchical organization of units, progression status, and explicit separation between presentation, logic, and persistence.



Figure 8. User interface.

The interface is developed as a dashboard in Node-RED using the node-red-dashboard library. Integration with the logical layer occurs through exchange of structured messages (msg) and flow variables (flow). As shown in Figure 8, the presentation layer fulfills three bounded responsibilities: displaying content units, visualizing progression, and capturing interactions that trigger events toward the logical layer. It does not perform evaluation or threshold control.

The visual organization is structured into hierarchical views associated with instructional units. In Figure 8, blocks corresponding to “Thematic Units,” “Progress,” “Programming,” and “Networks” are visible, each with graphical progress indicators (percentage) and access controls. These indicators do not compute progression; they represent states received from the logical layer, which in turn retrieves them from persistence.

Navigation between views is not unrestricted. Access buttons (“Go,” side controls, top navigation) generate explicit events encapsulated in msg.payload or msg.topic and sent to Node-RED flows. Enabling or blocking these controls depends on signals returned by the logical layer. The figure illustrates this relationship through arrows connecting the interface to the logical and persistence layers, differentiating responsibilities.

Each view integrates specific graphical components according to interaction requirements:

- Buttons (ui_button): initiate activities, submit responses, or enable page transitions.
- Selection components (ui_radio, ui_dropdown): capture responses or perceived difficulty levels.
- Dynamic text components (ui_text): display states, instructions, and feedback.

- Multimedia elements: render resources selected externally by the logical layer.

The interface does not validate responses or interpret results. During evaluation, it aggregates user inputs and transmits them to the logical layer for processing. Once the result is computed, it receives the corresponding state and updates visual indicators or enables new views. At the bottom of Figure 8, progression states (Blocked, Enabled, In progress, Approved, Not Approved) are shown; they are visually represented but determined outside the interface.

Visibility control constitutes a central technical mechanism. The interface applies conditions to received states—for example, support level or unit approval—to activate or deactivate buttons, show or hide supports, and restrict navigation. These conditions operate as rendering rules, not as autonomous pedagogical decisions.

The presentation layer does not access the database directly. All relevant records—response submission, navigation, support requests, and difficulty selection—are transmitted as events to the logical layer, which manages persistence. Figure 8 thus demonstrates that the interface functions as a controlled interaction component, aligned with the dual architecture and decoupled from evaluation logic and data storage.

4. Evaluation

The scope of the contribution must be interpreted with precision. At the conceptual level, the study proposes a dual articulation between pedagogical architecture and technical architecture for the structured design of educational environments in applied sciences. At the architectural level, it shows how this articulation can be translated into a coordinated implementation through a layered system. At the empirical level, however, the evidence is limited to the analyzed IoT-based instantiation and supports only its architectural coherence and technical-operational feasibility under the evaluated conditions. The study therefore does not establish pedagogical effectiveness, learning improvement, or superiority over alternative instructional approaches. Within this scope, Table 2 summarizes the main technical-operational evidence used to evaluate the analyzed instantiation from the perspectives of usability and interaction, system behavior under load, and basic technical security. This synthesis provides an overview of the results developed in the following subsections by identifying the evaluation dimensions, the evidence sources, the main findings, and their architectural implications.

Table 2. Summary of technical-operational evidence obtained from the IoT-based instantiation.

Evaluation Dimension	Evidence Source	What Was Assessed	Main Finding	Architectural Implication
Usability and interaction	SUS applied after platform interaction session with 10 participants	Perceived usability and interaction fluency	The environment was perceived as usable and operationally understandable under the evaluated conditions	Supports the viability of the presentation and interaction flow in the analyzed instantiation
Concurrent load behavior	Load testing under increasing concurrency	Request handling, response behavior, and incomplete requests under stress	Stable behavior under low/moderate concurrency and progressive degradation under higher demand	Delimits the operating range within which pedagogical coordination can be sustained without major disruption
Technical security	OWASP ZAP vulnerability assessment	Exposure to common web-related vulnerabilities	No critical vulnerabilities were identified in the evaluated configuration; hardening needs remain	Supports bounded technical viability while indicating areas for implementation-level improvement

4.1. Performance

Performance tests were designed to characterize the behavior of the web platform under conditions of increasing concurrent load, enabling the identification of stable operating ranges, progressive degradation thresholds, and saturation scenarios. The adopted approach does not merely verify functional operability; rather, it quantifies how system performance varies when concurrency levels exceed those expected in a real educational environment. This makes it possible to analyze system response under demanding conditions and to describe its operational limits. In particular, three operational dimensions were evaluated:

- The system's capacity to process an increasing volume of requests.
- The proportion of requests that fail to complete successfully under load.
- The evolution of response times as the number of simultaneous users increases.

4.1.1. Load Testing Execution Procedure

The execution of the tests was grounded in the principle of operational realism, understanding system performance as an emergent property of effective interaction between users and the platform, rather than solely as the outcome of abstract synthetic loads. From this perspective, the experimental design prioritized faithful reproduction of real usage patterns, so that the measurements obtained would reflect operating conditions comparable to those observed in a high-concurrency educational environment. Accordingly, the evaluation strategy was oriented toward preserving the sequence, timing, and structure of interactions, enabling analysis of the impact of progressively increasing concurrent users on overall system behavior.

To operationalize this approach, a sequential procedure composed of the following technical stages was followed:

- Interaction flow capture. A complete session of platform use—including login, navigation, content interaction, and assessment—was recorded. This capture was performed as a full HTTP trace, ensuring that each browser-generated request was registered in order and with real usage parameters.
- Trace conversion into a load scenario. The captured trace was exported in *.har* (HTTP Archive) format and used as input to generate an executable load-testing scenario. This scenario deterministically reproduces the sequence of HTTP requests observed in the real session, enabling simulation of multiple users executing the same interaction pattern.
- Execution across concurrency scenarios. Based on this base scenario, independent tests were executed for different concurrent population sizes: 50, 100, 250, 500, 1000, 5000, and 10,000 users. In each scenario, all simulated users execute the same interaction flow, allowing isolation of the effect of increasing concurrency on system performance.
- Recorded metrics. During each execution, the following global indicators were automatically recorded:
 - Total number of processed requests.
 - Number of failed requests.
 - Distribution of response times across three ranges:
 - Less than 800 ms.
 - Between 800 ms and 1200 ms.
 - Greater than 1200 ms.

4.1.2. Quantitative Results

The quantitative results obtained from the load tests allow systematic analysis of the relationship between concurrency level and the operational behavior of the platform.

Table 3 consolidates, for each evaluated scenario, the total volume of processed requests, the proportion of failed requests, and the distribution of response times, providing an integrated view of system performance under progressively more demanding conditions.

Table 3. Load Testing Results Under Increasing Concurrent Users.

Concurrent Users	Total Requests Processed	Failed Requests	Failure Rate (%)	<800 ms	800–1200 ms	>1200 ms
50	14,860	336	2.26%	14,524	71	496
100	29,778	942	3.16%	28,836	421	1807
250	89,486	5435	6.07%	84,051	614	6821
500	150,057	11,924	7.95%	138,133	1301	10,339
1000	293,172	34,447	11.75%	258,725	24,858	23,791
5000	935,361	577,488	61.74%	357,873	33,851	97,531
10,000	1,343,995	964,350	71.75%	379,645	32,586	111,021

Low-concurrency scenarios, corresponding to 50 and 100 simultaneous users, show clearly stable behavior. In these cases, the failure rate remains below 3.2% and the vast majority of requests are resolved in less than 800 ms. This pattern indicates that, under low loads, the system has sufficient headroom in terms of processing resources, access to internal services, and handling of concurrent requests. Even when concurrency increases to 250 users, although the failure rate rises to approximately 6%, more than 93% of the requests continue to be processed correctly, and the proportion of fast responses remains dominant. This behavior allows this range to be characterized as a stable operating regime, compatible with the expected use in educational contexts with small and medium-sized groups.

The scenario changes progressively starting from 500 concurrent users. At this point, the failure rate approaches 8% and a gradual shift in requests toward higher latency ranges begins to be observed. This phenomenon becomes more pronounced at 1000 users, where the failure rate exceeds 11% and the number of responses with times greater than 800 ms increases significantly. The coexistence of a high volume of successful requests with a sustained increase in latency suggests the emergence of bottlenecks in shared resources, such as execution threads, database access, or internal processing queues. In this range, the system continues operating, but under a regime of progressive degradation in which response capacity begins to be compromised.

The scenarios of 5000 and 10,000 concurrent users reveal a more marked qualitative change. In these cases, the failure rate exceeds 60% and reaches values close to 72% in the most extreme scenario. Although the system continues processing a non-negligible number of successful requests, a significant fraction of these present response times greater than 1200 ms, indicating a state of generalized saturation. Under these conditions, the platform clearly exceeds its intended operational capacity, and the observed performance reflects a partial collapse of the concurrent request handling mechanisms.

Taken together, the load-testing results indicate that the implemented architecture operates stably under low and moderate concurrency conditions, while showing progressive degradation as concurrent demand increases beyond that range. Based on the analysis of the indicators presented in the table, it is possible to distinguish three clearly differentiated operating zones. First, a stable zone, characterized by low failure rates and predominantly short response times, compatible with low- and medium-concurrency usage conditions. Second, a transition zone, in which the system remains operational but with progressive degradation in both reliability and latency. Finally, a saturation zone, in which the level of concurrency consistently exceeds the design capacity of the platform, and incomplete requests together with increased response times become more pronounced. This pattern is

consistent with an architecture oriented toward educational environments, where the number of real concurrent users usually falls within the stable zone or, in occasional situations, within the transition zone, but remains well below the extreme scenarios used to explore the operational limits of the system. Under this interpretation, the results do not invalidate the architectural proposal, but rather delimit the operating conditions under which the analyzed instantiation can sustain pedagogical coordination without significant disruption of the interaction flow.

From an architectural standpoint, these results must be interpreted as evidence of global system behavior rather than as a component-level performance profile. The implemented environment integrates interface processes, persistence operations, event handling, and scaffolding-related regulation within the same execution flow. Therefore, the observed degradation under higher concurrency should be understood as the cumulative effect of the instantiated architecture as a whole, rather than as a measured overhead attributable to a single module. In this sense, the scaffolding logic is part of the system load because it requires state updates, rule evaluation, and support regulation during task execution; however, the present evaluation does not isolate its computational cost from that of other technical components. A finer-grained assessment of per-component overhead remains a relevant direction for future scalability analysis.

4.2. Usability

To examine usability and interaction quality in the implemented environment, the *System Usability Scale* (SUS) was administered after a direct interaction session with the platform. The instrument was completed by 10 participants after using the environment under guided task conditions. The usability evaluation was oriented toward characterizing the users' interaction experience with the web platform, considering not only the subjective perception of ease of use but also the consistency of the navigation flow and the operational clarity of the system during the execution of the intended use tasks within the environment. This analysis starts from the assumption that a technically stable platform may still present relevant frictions if the interface, system messages, or functional organization are not understandable for the end user. In this study, SUS is not used as a measure of learning effectiveness, but as a standardized indicator of perceived usability and interaction fluency in the implemented system. Its role is therefore limited to verifying the adequacy of the system from the user's perspective within the analyzed instantiation.

4.2.1. Evaluation Instrument and Procedure

Usability was measured using the System Usability Scale (SUS), a standardized instrument widely used to evaluate the overall perception of usability of interactive systems [33]. The questionnaire consists of ten items (P1–P10) with a five-point Likert scale, alternating positive and negative statements, which allows capturing in a balanced way different aspects of interaction, such as ease of use, perceived complexity, confidence, and the need for prior learning.

The questionnaire was applied to ten participants (U1–U10) after completing a session of using the platform. The individual responses were processed following the standard SUS scoring procedure, obtaining a normalized score in the range of 0 to 100 for each participant [34]. This value summarizes the overall perception of usability and allows aggregated comparisons between users and with references widely accepted in the literature.

4.2.2. Quantitative Results Obtained

Table 4 presents the individual responses by participant for each of the ten questionnaire items, together with the resulting SUS score.

Table 4. Results of SUS scores by participant.

User	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	SUS
U1	5	1	5	2	4	2	5	1	5	2	90
U2	5	2	4	2	4	1	5	1	5	1	90
U3	5	1	5	1	4	2	5	1	5	2	92.5
U4	4	1	5	1	5	2	4	2	5	2	87.5
U5	5	1	5	2	3	1	5	1	4	1	90
U6	4	2	5	1	3	1	5	1	4	1	87.5
U7	4	2	4	2	5	1	5	1	5	1	90
U8	5	1	4	1	5	2	5	2	5	2	90
U9	4	1	4	1	4	1	4	1	5	1	90
U10	4	1	4	1	3	2	5	1	4	1	85

The results obtained show a consistently high level of usability among the evaluated participants. The average SUS score of 89.25 is clearly above the commonly accepted reference threshold of 68 points, indicating a favorable perception of the platform in terms of ease of use and overall acceptance. In addition, the dispersion of the individual values is low, with scores ranging between 85 and 92.5, which suggests a homogeneous user experience without marked differences among users.

The response pattern by item reinforces this interpretation. The questions formulated in a positive sense (for example, P1, P3, P5, P7 and P9) mostly present high values, while those formulated in a negative sense (P2, P4, P6, P8 and P10) predominantly show low values. This internal consistency indicates that participants perceive the platform as easy to use, with low unnecessary complexity and without requiring a high level of prior learning effort.

4.3. Security Testing

Security tests were conducted through dynamic analysis of the web platform using OWASP ZAP 2.16.0 (Zed Attack Proxy), with the aim of identifying weaknesses related to server configuration, HTTP header management, exposure of client-side components, and resource-sharing policies. The evaluation focused on the observable behavior of the system under a systematic automated scanning process capable of inspecting both server responses and resources loaded during normal application execution.

The procedure involved executing a full scan of the platform URL with OWASP ZAP. This made it possible to examine accessible endpoints, HTTP response headers, JavaScript files used by the application, and browser-level protection mechanisms. During the scan, the tool generated alerts classified according to risk level and number of occurrences, which provided a structured characterization of the exposure vectors present in the current system configuration.

The results of the analysis are consolidated in Table 5, which summarizes the types of alerts identified, their assigned risk levels, and the number of occurrences detected during the scan.

The alerts associated with the Content Security Policy (CSP) indicate that the application allows the loading of resources without explicit origin restrictions, either through the use of wildcard directives or due to the absence of a defined policy. From an operational perspective, this configuration implies that the browser does not receive instructions to

restrict the origin of scripts, style sheets, or other active resources, thereby expanding the exposure surface in scenarios involving content injection.

Table 5. Alerts detected during the security analysis of the platform.

Alert Type	Risk Level	Count
CSP: Wildcard Directive	Medium	3
Content Security Policy Not Configured	Medium	2
Cross-Domain Misconfiguration	Medium	12
Missing Anti-Clickjacking Header	Medium	2
Vulnerable JavaScript Library	Medium	1
Information Disclosure via HTTP Header "X-Powered-By"	Low	12
Missing X-Content-Type-Options Header	Low	11
Information Disclosure via Comments	Informational	12
Modern Web Application Detected	Informational	2
User Agent Fuzzer	Informational	24

Cross-domain misconfiguration is related to the way the server manages requests between origins. The repeated occurrence of this alert indicates that the system responds permissively to requests that do not include a restrictive policy declared at the HTTP header level, which is consistent with configurations oriented toward interoperability between components deployed within the same environment.

The absence of the Anti-Clickjacking header indicates that the server does not impose restrictions to prevent the application from being loaded within frames or iframes. Similarly, the absence of the X-Content-Type-Options header indicates that the browser may attempt to infer the content type of resources, expanding the margin of client-side interpretation for certain types of responses.

The vulnerable JavaScript library alert indicates the presence of a client-side component whose version contains known vulnerabilities, introducing an exposure vector dependent on dependency management and the updating of external components. Alerts classified as low risk and informational describe observable characteristics of the system and of the scanning process, such as the disclosure of technological information through HTTP headers or the identification of the user agent used during the analysis.

5. Discussion

The purpose of this study was to examine the feasibility of a dual model for the design of educational environments in applied sciences through the analysis of a specific implementation derived from that model. The study was not conceived as a pedagogical validation of the model, nor as an assessment of learning outcomes, but as an examination of whether the proposed articulation between pedagogical architecture and technical architecture can be operationally materialized in a real technological environment without evident functional limitations.

At the conceptual level, the study addresses RQ1 by formulating a dual model that differentiates a pedagogical architecture based on computational scaffolding from a layered technical architecture. This formulation is grounded in the analysis of previous work discussed in the article, where scaffolding is usually examined from a pedagogical perspective, while its technical materialization remains implicitly embedded in specific platforms or tools [8,9,35]. The contribution does not lie in claiming that pedagogy and technology are

distinct concerns, since that distinction is already present in the broader literature, but in making their relationship architecturally explicit through two analytically differentiated and operationally coordinated structures. In this sense, RQ1 is addressed through model formulation and architectural definition rather than through empirical validation of its educational effects.

At the architectural level, the study addresses RQ2 through the analysis of an IoT-based instantiation derived from the proposed dual model. The relevance of this case lies in the fact that it requires coordination between physical interaction, persistence mechanisms, logical regulation, and interface behavior within the same instructional sequence. Under these conditions, the instantiation makes it possible to examine whether the separation between pedagogical architecture and technical architecture can be sustained coherently in a concrete implementation. Unlike previous proposals whose evaluation relies mainly on perceptual or acceptance measures [10], the implementation analyzed here is examined from the perspective of its architectural coherence and technical-operational feasibility. This does not imply that the case is representative of all possible applied science environments, but it does function as a bounded architectural scenario in which the coordination demands of the model become observable.

At the empirical level, the study addresses RQ3 through technical and interaction evidence obtained from the implemented environment. Usability results, system behavior under increasing load, and basic security analysis make it possible to characterize the technical-operational feasibility of the analyzed instantiation under the evaluated conditions. These results indicate that the environment can operate without evident interaction breakdown under expected conditions of use, while also allowing the identification of operating ranges, progressive degradation patterns, and implementation-level hardening needs. In this sense, the empirical contribution of the study lies in showing that technical and interaction evidence can be used to assess the viability of an instantiation derived from the dual model without depending on a specific disciplinary outcome measure or on a particular technology stack beyond the analyzed case.

An important limitation of the present study must be stated explicitly. The empirical evidence reported here does not support claims about pedagogical effectiveness, learning gains, or instructional superiority of the proposed model. The evaluation was intentionally limited to examining the architectural coherence and technical-operational feasibility of the analyzed instantiation, considering usability and interaction evidence, system behavior under load, and basic security conditions. Consequently, the findings should be interpreted as evidence of implementation viability within the evaluated environment, not as evidence that the dual model improves learning outcomes in a generalizable educational sense.

The scope of the analysis remains bounded accordingly. The study does not incorporate longitudinal designs, control groups, or outcome measures oriented toward educational effects. It examines whether the proposed dual model can be implemented and operate coherently within a specific applied sciences scenario, but it does not support the attribution of pedagogical impacts to the model or comparisons among instructional approaches.

6. Conclusions

This study proposes a dual model for the design of educational environments in applied sciences that differentiates a pedagogical architecture based on computational scaffolding from a layered technical architecture, while maintaining explicit coordination mechanisms between the two. Under this approach, pedagogical decisions can be defined independently of specific implementation choices, and technical components can be organized without embedding instructional logic in an implicit or ad hoc manner.

The IoT-based instantiation analyzed in this work shows that this articulation can be implemented coherently in a concrete technological scenario. In the evaluated environment, the pedagogical architecture governs task sequencing, support activation, and integrated assessment, while the technical architecture organizes interaction with the physical module, persistence, logical coordination, and presentation. This arrangement allows the functional separation between interaction with the phenomenon and instructional coordination to be examined in the implemented system.

From the technical perspective, the results support the operational feasibility of the analyzed instantiation under the evaluated conditions. The evidence from concurrent load testing, usability assessment, and vulnerability analysis makes it possible to characterize operating ranges, identify conditions of progressive degradation, and examine whether the technical architecture can sustain the pedagogical coordination required by the environment without disrupting the interaction flow.

Its main contribution does not lie in demonstrating pedagogical effectiveness or learning improvement, but in showing that the proposed articulation between computational scaffolding and layered technical architecture can be implemented consistently in a concrete scenario. The evidence obtained supports the architectural coherence and technical-operational viability of the evaluated instantiation, rather than the educational validation of the model in a general sense.

Several future research directions are identified to extend the scope of the present work. First, the pedagogical validation of the model should be addressed through comparative or longitudinal designs incorporating learning indicators, self-regulation processes, and measures of student autonomy. Second, it is relevant to instantiate and analyze the dual model in disciplinary domains other than the IoT scenario, while keeping the pedagogical architecture constant and deliberately varying the technical architecture. This would make it possible to examine the extent to which the structural coherence observed in the current implementation is preserved when the model is applied to environments with different dynamics, tools, and constraints, such as virtual laboratories, process simulators, or cyber-physical systems. In this line of work, the aim is not to compare technologies, but to evaluate the conceptual stability of the model and the clarity of the relationship between pedagogical and architectural decisions under heterogeneous implementation conditions. It is also necessary to extend the current architecture toward tighter integration between the physical module and the platform by incorporating explicit synchronization mechanisms and event timestamping in order to distinguish network-induced latency from latency associated with student response during task execution. In addition, it is advisable to incorporate data authenticity and validation mechanisms in scenarios where the physical module transmits telemetry directly to the system, so that the persistence and decision layers can operate on verifiable input data under more integrated deployment conditions. Finally, another line of development consists of exploring more advanced support-regulation models, including probabilistic updating schemes or adaptation mechanisms based on learning curves, beyond the rule-based scaffolding logic implemented in the present study.

Author Contributions: Conceptualization, D.A.; methodology, D.A.; software, E.M.; validation, E.M.; formal analysis, D.A.; investigation, D.A. and E.M.; resources, E.M.; data curation, E.M.; visualization, D.A.; writing—original draft preparation, D.A.; writing—review and editing, D.A. and E.M.; supervision, D.A.; project administration, D.A. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Ethical review and approval were waived for this study because the research focused on the technical implementation and operational testing of a software platform and did not involve experimental evaluation of learning outcomes or interventions with human subjects. The development and testing of the system were conducted with institutional authorization from the Universidad de las Fuerzas Armadas ESPE.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data presented in this study are available on request from the corresponding author.

Acknowledgments: The authors acknowledge the institutional support provided by the Universidad de las Fuerzas Armadas ESPE during the development of this work.

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

IoT	Internet of Things
ESP32	Embedded Serial Peripheral 32-bit (Microcontroller Platform)
IDE	Integrated Development Environment
SUS	System Usability Scale
UI	User Interface
UX	User Experience
Wi-Fi	Wireless Fidelity
LED	Light Emitting Diode
JSON	JavaScript Object Notation
HTTP	Hypertext Transfer Protocol
MQTT	Message Queuing Telemetry Transport
GPIO	General Purpose Input/Output
JS	JavaScript
HTML	HyperText Markup Language
CSS	Cascading Style Sheets
LMS	Learning Management System
ESPE	Universidad de las Fuerzas Armadas ESPE
STEM	Science, Technology, Engineering, and Mathematics

References

1. Feisel, L.D.; Rosa, A.J. The role of the laboratory in undergraduate engineering education. *J. Eng. Educ.* **2005**, *94*, 121–130. [\[CrossRef\]](#)
2. Ma, J.; Nickerson, J.V. Hands-on, simulated, and remote laboratories: A comparative literature review. *ACM Comput. Surv.* **2006**, *38*, 7. [\[CrossRef\]](#)
3. Kolb, D.A. *Experiential Learning: Experience as the Source of Learning and Development*; Prentice-Hall: Upper Saddle Rive, NJ, USA, 1984.
4. Abdulwahed, M.; Nagy, Z.K. Applying Kolb's experiential learning cycle for laboratory education. *J. Eng. Educ.* **2009**, *98*, 283–294. [\[CrossRef\]](#)
5. Poo, M.; Rivera-Figueroa, A.; Figueroa-Arce, D. An evaluation of virtual laboratories for engineering education based on the Kirkpatrick model. *Educ. Sci.* **2023**, *13*, 11110. [\[CrossRef\]](#)
6. Li, J.; Liang, W. Effectiveness of virtual laboratory in engineering education: A meta-analysis. *PLoS ONE* **2024**, *19*, e0316269. [\[CrossRef\]](#) [\[PubMed\]](#)
7. Wood, D.; Bruner, J.S.; Ross, G. The role of tutoring in problem solving. *J. Child Psychol. Psychiatry* **1976**, *17*, 89–100. [\[CrossRef\]](#) [\[PubMed\]](#)
8. Belland, B.R. *Instructional Scaffolding in STEM Education: Strategies and Efficacy Evidence*; Springer: Berlin/Heidelberg, Germany, 2017. [\[CrossRef\]](#)
9. Doo, M.Y.; Bonk, C.; Heo, H. A meta-analysis of scaffolding effects in online learning in higher education. *Int. Rev. Res. Open Distrib. Learn.* **2020**, *21*, 60–80. [\[CrossRef\]](#)

10. Mostefai, B.; Boutefara, T.; Bousbia, N.; Balla, A.; Dhelim, S.; Hammia, A. Enhancing user experience in e-learning systems: A new user-centric RESTful web services approach. *Comput. Hum. Behav. Rep.* **2025**, *18*, 100643. [\[CrossRef\]](#)
11. Shao, J.; Chen, Y.; Wei, X.; Li, X.; Li, Y. Effects of regulated learning scaffolding on regulation strategies and academic performance: A meta-analysis. *Front. Psychol.* **2023**, *14*, 1110086. [\[CrossRef\]](#) [\[PubMed\]](#)
12. Al-Fuqaha, A.; Guizani, M.; Mohammadi, M.; Aledhari, M.; Ayyash, M. Internet of Things: A survey on enabling technologies, protocols, and applications. *IEEE Commun. Surv. Tutor.* **2015**, *17*, 2347–2376. [\[CrossRef\]](#)
13. Gomes, L.; Bogosyan, S. Current trends in remote laboratories. *IEEE Trans. Ind. Electron.* **2009**, *56*, 4744–4756. [\[CrossRef\]](#)
14. Hernandez-de-Menendez, M.; Guevara, A.V.; Martínez, J.C.T.; Alcántara, D.H.; Morales-Menendez, R. Active learning in engineering education: A review of fundamentals, best practices and experiences. *Int. J. Interact. Des. Manuf.* **2019**, *13*, 909–922. [\[CrossRef\]](#)
15. Belland, B.R.; Walker, A.E.; Kim, N.J.; Lefler, M. Synthesizing results from empirical research on computer-based scaffolding in STEM education: A meta-analysis. *Rev. Educ. Res.* **2017**, *87*, 309–344. [\[CrossRef\]](#) [\[PubMed\]](#)
16. Kurtz, M.; Benabbou, A.; Pons, C.; Broisin, J. Collaboration in virtual and remote laboratories for education: A systematic literature review. *Int. J. Comput.-Support. Collab. Learn.* **2025**, *20*, 549–603. [\[CrossRef\]](#)
17. Elmoazen, R.; Saqr, M.; Khalil, M.; Wasson, B. Learning analytics in virtual laboratories: A systematic literature review of empirical research. *Smart Learn. Environ.* **2023**, *10*, 23. [\[CrossRef\]](#) [\[PubMed\]](#)
18. Sapriati, A.; Suhandoko, A.D.J.; Yundayani, A.; Karim, R.A.; Kusmawan, U.; Adnan, A.H.M.; Suhandoko, A.A. The effect of virtual laboratories on improving students' SRL: An umbrella systematic review. *Educ. Sci.* **2023**, *13*, 222. [\[CrossRef\]](#)
19. Van den Beemt, A.; Groothuisen, S.; Ozkan, L.; Hendrix, W. Remote labs in higher engineering education: Engaging students with active learning pedagogy. *J. Comput. High. Educ.* **2023**, *35*, 320–340. [\[CrossRef\]](#) [\[PubMed\]](#)
20. Dong, J.; Rum, S.N.M.; Kasmiran, K.A.; Aris, T.N.M.; Mohamed, R. Artificial intelligence in adaptive and intelligent educational system: A review. *Future Internet* **2022**, *14*, 245. [\[CrossRef\]](#)
21. Lin, C.-C.; Huang, A.Y.Q.; Lu, O.H.T. Artificial intelligence in intelligent tutoring systems toward sustainable education: A systematic review. *Smart Learn. Environ.* **2023**, *10*, 41. [\[CrossRef\]](#)
22. Fernández-Herrero, J. Evaluating recent advances in affective intelligent tutoring systems: A scoping review of educational impacts and future prospects. *Educ. Sci.* **2024**, *14*, 839. [\[CrossRef\]](#)
23. Kestin, G.; Miller, K.; Klaes, A.; Milbourne, T.; Ponti, G. AI tutoring outperforms in-class active learning: An RCT introducing a novel research-based design in an authentic educational setting. *Sci. Rep.* **2025**, *15*, 17458. [\[CrossRef\]](#) [\[PubMed\]](#)
24. Lin, K.-Y.; Li, M.-H.; Lo, F.-Y.; Huang, H.-C.; Matsuno, K.; Watanabe, R. Adaptive learning with human factors and Artificial Intelligence: Associations with training effectiveness in programming education. *Int. J. Ind. Ergon.* **2025**, *110*, 103834. [\[CrossRef\]](#)
25. Merino-Campos, C. The impact of Artificial Intelligence on personalized learning in higher education: A systematic review. *Trends High. Educ.* **2025**, *4*, 17. [\[CrossRef\]](#)
26. Salazar, J.C.; Aguilar, J.; Monsalve-Pulido, J.; Montoya, E. A generic architecture of an affective recommender system for e-learning environments. *Univers. Access Inf. Soc.* **2024**, *23*, 1115–1134. [\[CrossRef\]](#)
27. Lampropoulos, G.; Fernández-Arias, P.; de Bosque, A.; Vergara, D. Virtual reality in engineering education: A scoping review. *Educ. Sci.* **2025**, *15*, 1027. [\[CrossRef\]](#)
28. Sweller, J.; Ayres, P.; Kalyuga, S. *Cognitive Load Theory*; Springer: Berlin/Heidelberg, Germany, 2011. [\[CrossRef\]](#)
29. Rathnayaka, C.M.; Ganapathi, J.; Kickbusch, S.; Dawes, L.; Brown, R. Preparative pre-laboratory online resources for effectively managing cognitive load of engineering students. *Eur. J. Eng. Educ.* **2024**, *49*, 113–138. [\[CrossRef\]](#)
30. Mayer, R.E. *Multimedia Learning*, 3rd ed.; Cambridge University Press: Cambridge, UK, 2020.
31. Kabisch, S.; Kamiya, T.; McCool, M.; Charpenay, V.; Kovatsch, M.; Matsukura, R. *Web of Things (WoT) Thing Description; W3C Recommendation; World Wide Web Consortium (W3C)*: Wakefield, MA, USA, 2020. Available online: <https://www.w3.org/TR/wot-thing-description/> (accessed on 10 June 2024).
32. Kovatsch, M.; Matsukura, R.; Lagally, M.; Kawaguchi, T.; Tsumura, K.; Kajimoto, K. *Web of Things (WoT) Architecture; W3C Working Group Note; World Wide Web Consortium (W3C)*: Wakefield, MA, USA, 2015. Available online: <https://www.w3.org/TR/wot-architecture/> (accessed on 10 June 2024).
33. Brooke, J. *SUS: A "Quick and Dirty" Usability Scale*; Jordan, E.P.W., Thomas, B., Weerdmeester, B.A., McClelland, I.L., Eds.; Usability Evaluation in Industry; Taylor & Francis: Oxfordshire, UK, 1996; pp. 189–194.
34. Sauro, J.; Lewis, J.R. *Quantifying the User Experience: Practical Statistics for User Research*, 2nd ed.; Morgan Kaufmann: Burlington, MA, USA, 2016.
35. Kim, N.J.; Belland, B.R.; Walker, A.E. Effectiveness of computer-based scaffolding in the context of problem-based learning for STEM education: Bayesian meta-analysis. *Educ. Psychol. Rev.* **2018**, *30*, 397–429. [\[CrossRef\]](#)

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.