UDC 658.512.2 **DOI: https://doi.org/10.30837/PMW.2026.T1.315**

INTEGRATION OF A STANDARDIZED BOULDERING BOARD INTO AN ARCHITECTURE FOR SIMULTANEOUS TRAINING MANAGEMENT

Guryev I.
Full-time professor, Department of Multidisciplinary Study,
Engineering Division, University of Guanajuato
ORCID: 0000-0001-6014-5912
Gurieva N.
Full-time professor, Department of Digital Art and Management,
Engineering Division, University of Guanajuato
ORCID: 0000-0002-1366-1292

Abstract. *Standardized training boards such as the MoonBoard are the most popular tool for automated indoor boulder training. However, their control systems are normally implemented as isolated devices with no integration into gym management software. In this study we have developed a low-cost, open-source architecture that based on ESP32 LED matrix controller managing a 198-hold, 18×11 bouldering board with a RESTful web application that unifies member management, attendance tracking, and training session control under a single interface accessible to multiple users simultaneously. In this study, we have introduced the real-time synchronization of training state across concurrent web clients by means of polling-based protocol using the controller's state-update requests. The board can be operated both with its own basic frontend via direct client connection within the local network, as well as by means of the gym management software via proxy integration of the board's HTTP API. This, naturally, eliminates the need for separate Bluetooth pairing or proprietary mobile applications. A lightweight JSON/HTTP polling scheme with 1.5-second period was demonstrated to be short enough to achieve nearly flowless synchronization across multiple browser sessions. The system was deployed and validated at a small indoor climbing facility over a two-month period. Results indicate that the integrated architecture reduces front-desk administrative time, improves training efficiency and, due to high accessibility, increases the usage of the training board in general.*

Keywords: *indoor climbing, IoT, ESP32, LED training board, gym management system, bouldering, MoonBoard, real-time synchronization.*

Introduction

Indoor climbing has shown remarkable growth over the past decade. The number of dedicated indoor climbing facilities in the United States alone exceeded 600 by 2023, up from approximately 400 in 2017, with similar trends observed across Europe and Latin America [1]. The professionalization of training methodology in climbing has accelerated significantly, since the inclusion of sport climbing in the 2020 Tokyo Olympic Games [2].

A central tool in the contemporary climber's training is the standardized training board, a fixed-angle overhanging panel equipped with a predefined set of holds that permits the creation and international sharing of specific problem sequences.

The MoonBoard, originally conceived by professional climber Ben Moon in the late 1980s and commercially released in 2016 with LED hold-indication technology, has become the dominant format for this category of equipment, with installations in thousands of facilities worldwide [3]. Its 2016 layout was the first commercially available LED training board, and successive versions (2017, 2019, 2024) have maintained a standardized 198-hold configuration within an 18×11 matrix.

Despite their widespread adoption, MoonBoard installations and training boards in general have historically operated in isolation from the operational software infrastructure of climbing facilities. Control is mediated through a proprietary Bluetooth-based mobile application, which requires individual device pairing and prevents simultaneous multi-user access. Route libraries are maintained in a centralized cloud database that cannot be replicated locally, creating a dependency on internet connectivity for basic training functions. No integration with membership management, check-in logging, or session duration tracking has been commercially available.

Simultaneously, the management software market for climbing facilities has grown considerably. Platforms such as Rock Gym Pro (now Onsight Pro), Approach, and Community offer cloud-based membership, point-of-sale, and waiver management, but remain entirely decoupled from training hardware [4, 5]. In result, a trainer supervising a MoonBoard session must operate two entirely independent systems, namely a mobile app for board control and a desktop or web application for member management with no data flow between them.

Aim of the investigation

This paper addresses that gap by presenting an integrated architecture in which a custom ESP32-based firmware replaces the proprietary MoonBoard control chain, exposing a local HTTP API that is proxied through a FastAPI-based gym management backend. The contribution is both technical demonstrating a viable synchronization protocol for multi-client LED state management, and operational, showing that the integration reduces workflow friction at the gym front desk while preserving full functionality of the training board for instructors.

The purpose of the research is to design, implement, and validate a low-cost, open-source IoT architecture that integrates an ESP32-based LED bouldering board controller with a web-based gym management system, enabling simultaneous multi-client training session control and member administration through a unified interface. This determined the following particular tasks:

– develop ESP32 firmware that exposes the bouldering board's LED matrix control via a local RESTful HTTP API supporting multiple training modes;

– design a FastAPI-based backend proxy layer that bridges the embedded controller with a membership management system, handling authentication, attendance tracking, and session logging;

– implement a polling-based multi-client synchronization protocol to maintain consistent training timer state across concurrent browser sessions;

– build a staff-facing web interface integrating board control and member management within a single dashboard;

– deploy and validate the integrated system in a real climbing facility, evaluating synchronization accuracy, operational impact, and overall hardware replicability.

Main part

1 State of the art

Training Load and Technology in Indoor Climbing.

Climbing's physiological demands have been a significant subject in recent literature. The early work [6] set out the basic cardiovascular and metabolic profile of the activity, and subsequent studies have narrowed the focus to the strength-endurance demands that distinguish bouldering from lead climbing. In a systematic review [7], confirmed what many coaches assume in practice: grip endurance and upper-body pulling strength dominate performance outcomes in sport climbers. A more recent contribution comes from Dindorf [8], who applied near-infrared spectroscopy (NIRS) sensors to 42 climbers performing dead-hangs. Their data showed that acute and cumulative forearm fatigue can in fact be tracked in real time, which opens the door to objective monitoring in climbing-specific settings.

Sensor networks are another active area. Rum et al. [9] built an IMU-based system to capture body-segment kinematics during wall sessions, and Breen et al. [10] went a step further, pairing wearable physiological monitors with video to produce microlocation-specific biometric profiles on a competition route. In a follow-up review, Breen [11] surveyed five sensor categories used in climbing research, namely body movement, respiration, cardiac activity, eye-gazing, and skeletal muscle monitoring. They argue that wearable data can give coaches and climbers actionable, science-grounded feedback to put alongside the more familiar subjective assessment.

Standardised Board Training and Machine Learning.

Because the MoonBoard hold layout is fixed, it has become an attractive testbed for machine learning. Dobles [12] was an early example, applying supervised classifiers to predict route difficulty directly from hold positions. Tai et al. [13] built on this with graph neural networks that capture spatial relationships between holds rather than treating positions as a flat feature vector. Stapel [14] took a different approach, proposing an automated grading and route-generation pipeline based on affordance-style hold geometry descriptors. Petashvili and Rodda [15] tested how well such models transfer between board setups, and reported that cross-setup generalization remains hard, mostly due to differences in hold geometry. A more integrated approach is given by Türedioğlu [16], who combined route planning and difficulty estimation within a single model-based framework for indoor bouldering.

All these approaches assume structured route data is available, namely hold positions, grades, and setter metadata, and that this data can be stored, retrieved, and tied back to specific climbers. Without it, training-load monitoring over time becomes

impractical. The bottleneck, then, is not the modelling itself but the lack of any integration layer between the board controller and the membership system in which climber identities and session histories are stored.

IoT and Embedded Systems in Sports Facility Management.

IoT techniques are by now well established in smart-building control and, more recently, in sport facility monitoring. The ESP32 microcontroller from Espressif Systems, a dual-core 240 MHz SoC with on-board Wi-Fi and Bluetooth, is one of the platforms most often used in this kind of project, mainly because it is cheap and capable enough for non-trivial workloads. Its mix of processing power, wireless support, and flexible GPIO suits it well to embedded equipment for sports.

LED matrix control for interactive training is not new in itself; reactive training walls and cognitive-motor training platforms have used the idea for years. What is more recent is the combination of an LED matrix with a RESTful HTTP API served directly from the microcontroller, without any intermediate gateway. This pattern has become practical thanks to the ESP32's on-chip TCP/IP stack and to readily available libraries such as the Arduino WebServer.

Gym Management Systems and Integration Gaps.

Commercial gym management platforms have become advanced and complex systems, but their scope remains administrative. Rock Gym Pro, which dominates the climbing-specific segment in the United States [5], handles memberships, point-of-sale, scheduling, and digital waivers, all without touching the training equipment itself. Anolla is something of an exception: it incorporates IoT hardware such as access control and visitor counters but does not extend to training boards. To the moment, no published work describes an architecture that puts LED-based training board control directly inside a gym membership management system.

2 System Architecture and Implementation

Embedded Controller (ESP32 Firmware).

The training board itself is a 25° inclined plywood panel of roughly 2.44 m by 3.15 m, populated with 198 standardized climbing holds in an 18-row by 11-column matrix. Behind each hold there is a WS2812B addressable RGB LED, giving independent color control for every position. The LED matrix is driven by an ESP32 microcontroller (ESP32-S3 N16R8 MCU 44 pin), running at 240 MHz with 4 MB of flash.

The firmware is written in C++ on the Arduino framework. It joins the local Wi-Fi network in station mode and runs an HTTP server on port 80, using the Arduino WebServer library. Board state is held in memory as a 198-character string in which each character encodes one of five color values, namely 'k' for black (off), 'w' for white, 'r' for red, 'g' for green, and 'b' for blue. Route data is kept on a microSD card in .mbd files.

A summary of the HTTP endpoints exposed by the firmware is given in Table 1.

Figure 1 shows the resulting four-layer architecture: a physical layer (ESP32, LED matrix and SD card), an application layer (the FastAPI backend on a Raspberry Pi), a presentation layer (the browser-based staff dashboard, member portal, and self-

registration page) and a local interface provided by the board itself via the local network connection. The local interface is indicated as a separate layer since it can be used for the initial WiFi settings as well as to access the board functions without the gym management software (Table 2, Figure 2).

Table 1 – ESP32 HTTP API endpoints exposed by the firmware

Endpoint	Method	Parameters	Description
/get_routes	GET	–	Returns JSON array of saved route names
/set_color	GET	data (198-char)	Updates LED matrix state
/states_update	GET	data (198-char)	Polls state; returns LEDs colors + training timer + training state (‘t’ or ‘0’)
/{fname}.mbd	GET	–	Returns raw route data (198 bytes)
/save_route	GET	fname, data	Saves route to SD card
/delete_route	GET	fname	Removes route from SD card
/start	GET	mode, per, dur, routes	Initiates a training session
/stop	GET	–	Terminates the active training session

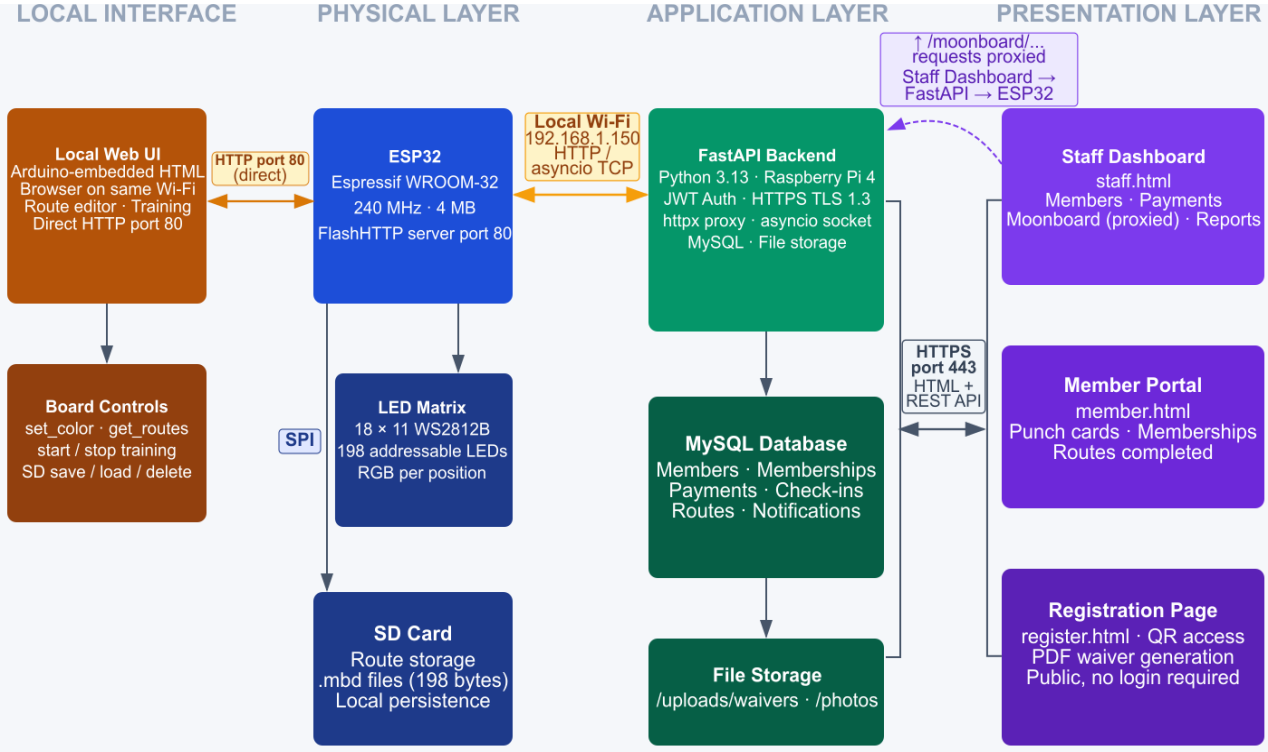

Figure 1 – Integrated IoT system architecture for climbing gym management and LED board control

Table 2 – Training modes and their behavioral descriptions

Mode	Name	Behavior
1	Memory	Sequence of holds presented for determined number of seconds; climber must reproduce it from memory
2	Free	Random single hold illuminated per period
3	Free (pairs)	Random pairs of holds of different colors are illuminated per period (for training in couples)
4	Block	Random single hold illuminated per period. After several periods, all the LEDs are illuminated red, meaning the climber should stay as close to the wall as possible (blocked)
5	Block (pairs)	Block protocol with paired holds of different colors
6	Multi-route	Cycles through a sequence of preselected set of saved routes

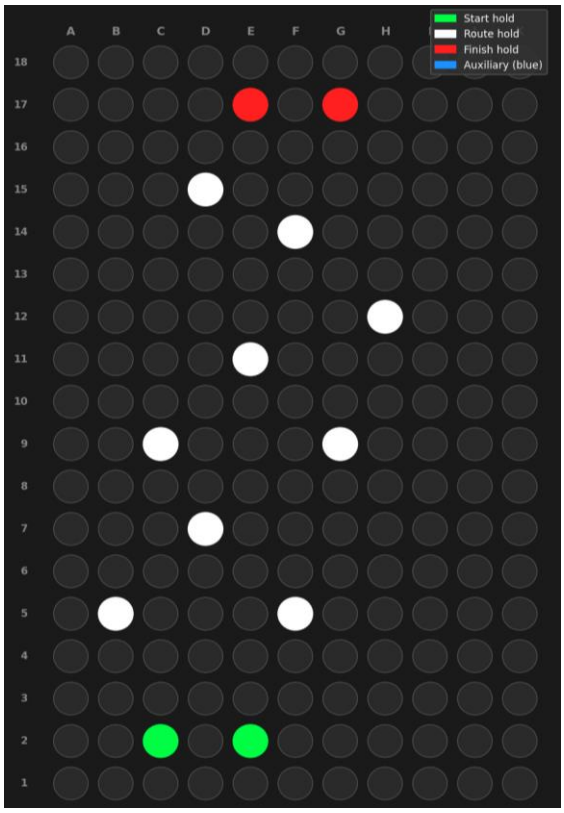

Figure 2 – Interactive LED grid interface – the 18×11 MoonBoard (198 WS2812B LEDs) with a sample route shown. Green = start holds, white = route holds, red = finish holds

Backend Proxy Layer (FastAPI).

The gym management backend is built on FastAPI (Python 3.13), with SQLAlchemy as the ORM and MySQL as the relational store. It runs on a Raspberry Pi 4 under Raspberry Pi OS and is exposed over HTTPS (TLS 1.3, Sectigo certificate) on port 443. Authentication is handled with JWT tokens signed with HS256, and role-based access control separates staff from member roles.

A set of proxy endpoints under /moonboard/... forwards requests from the web application to the ESP32, which sits at a fixed local address (192.168.1.150). For most endpoints, the proxy uses the httpx async HTTP client. Route loading, however, gave us trouble. The ESP32's WebServer library closes the TCP connection immediately after server.sendContent() returns, without sending a conformant HTTP response terminator first. Httpx, reasonably enough, raises RemoteProtocolError when it receives this. Rather than patch the firmware, we re-implemented the route-loading proxy with a raw asyncio TCP socket that reads bytes until the connection closes and then splits the HTTP header from the body manually at the \r\n\r\n boundary. This is forgiving of the embedded server's non-standard behavior and keeps the existing local HTML interface untouched.

Frontend Integration (Staff Dashboard).

The staff-facing web interface is a single-file vanilla JavaScript application, served straight from the FastAPI backend. Its Moonboard section draws an interactive 18×11 LED grid as a CSS grid of 28×28 px circular cells. A small color picker lets the trainer choose a paint color, after which clicking a cell toggles the hold. Route loading,

saving and deletion all go through the proxy API. The training controls expose all six modes, with the period and duration parameters editable from the same panel.

Multi-client synchronization is based on the states_update polling mechanism. On the client side, every browser session runs a setInterval loop at 1500 ms. On each tick, the client posts its current 198-character board state to /moonboard/states_update. If a training session is active, the response includes the remaining time in seconds, encoded as a 4-digit zero-padded string followed by a t (for example, "0247t" means 247 seconds left). The client pulls this value out and passes it to a mbSyncCountdown() function, which both updates the on-screen timer and resets a local 1-second countdown interval. The intent of this dual-timer setup is straightforward: the poll gives authoritative time, and the local countdown keeps the display smooth between polls. The net effect is that every connected client stays within roughly one polling interval of the server's view of the training timer.

Validation and Results

Deployment Context.

The system was deployed at Roca Viva, a small indoor climbing and bouldering facility in Irapuato, Guanajuato, México. The gym runs with around 60 active members, a staff of three, and a single MoonBoard installation. Before the deployment, the board was controlled exclusively through its local interface, which while permitted a multi-user connections and full control, required a connection from the local network. Member records, meanwhile, were registered in a spreadsheet.

Timer Synchronization Accuracy.

To achieve multi-client synchronization, we opened three browser sessions in parallel on three different devices (desktop, tablet and smartphone), all on the same local network. A 3-minute training session was then started from one of the clients. Timer readouts were recorded by hand every 30 seconds on all three clients, and each reading was compared against the ground-truth value returned by the ESP32's states_update response (Figure 3).

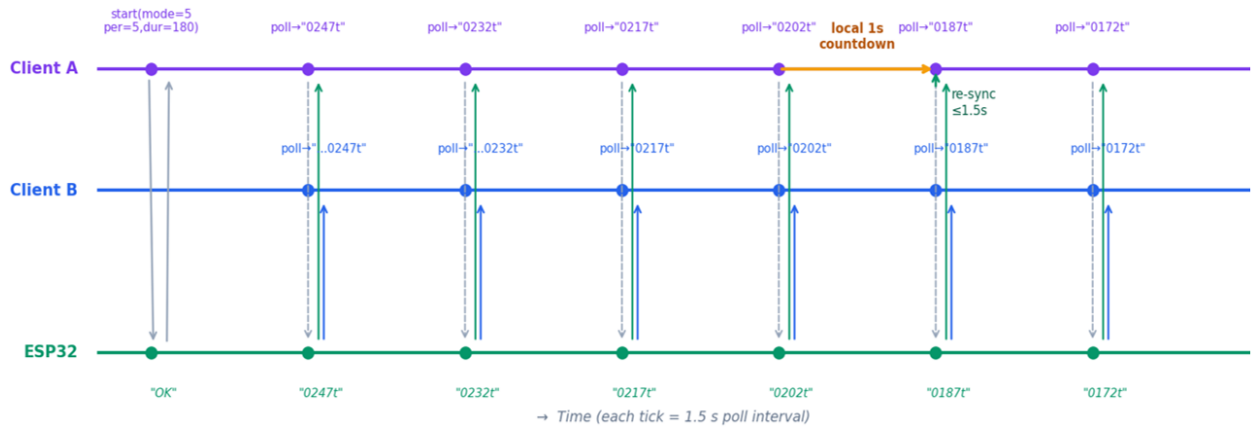

Figure 3 – The states_update polling protocol for multi-client timer synchronization. Client A initiates training; Client B joins the next poll and receives the current countdown. Both clients re-synchronize every 1.5 seconds from the ESP32's authoritative value

Across the full session, all three clients stayed within ±2 seconds of the authoritative timer. The largest divergence we registered was 1.8 seconds, and it occurred right after session started before the non-initiating clients had completed their first synchronization poll (Figure 4). After that initial offset, no drift accumulated for the rest of the session.

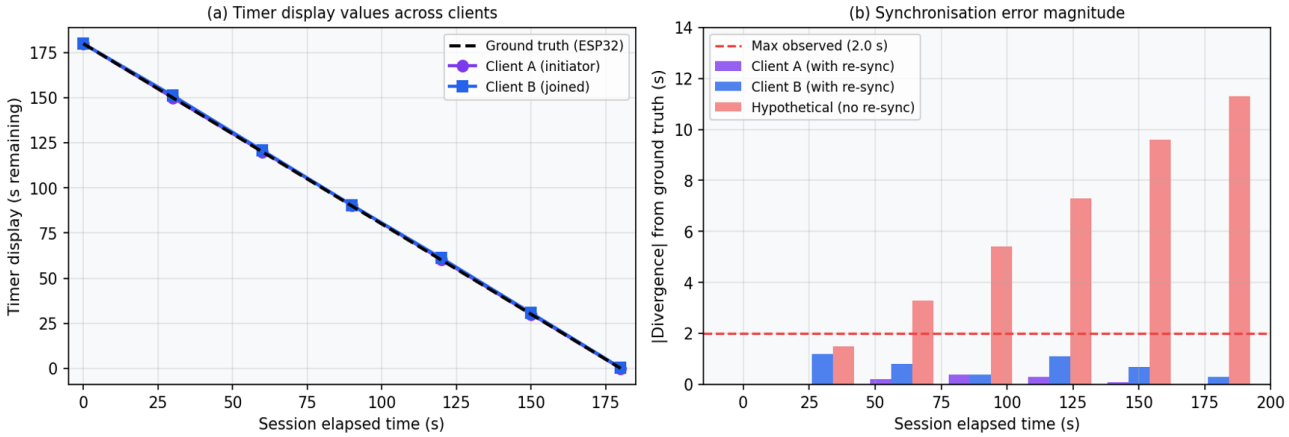

Figure 4 – Timer synchronization accuracy across a 3-minute training session.
Panel (a) shows displayed timer values.
Panel (b) shows absolute divergence from the ESP32 ground truth.
A hypothetical no-resync scenario (red bars) illustrates the drift that would accumulate without periodic correction

Operational Integration Outcomes.

Over the two-months deployment we noted several operational outcomes. First, training sessions, including start time, duration, mode and the staff member who initiated them, are now captured implicitly by the attendance and check-in system, which makes per-member longitudinal session tracking possible without any extra logging step. Second, the staff reported that having membership management and board control in the same browser tab removed the context-switching that previously forced them to move between the local board interface and the spreadsheet. Third, fourteen custom routes were stored on the board's SD card during the deployment and remained accessible across power cycles. Fourth, on two occasions the ESP32 became unreachable, in both cases the backend returned HTTP 503, and the frontend showed a connectivity warning with a retry button rather than failing silently.

System Cost and Replicability.

Total hardware cost for the embedded controller subsystem, namely the ESP32 module, the LED strip, the microSD breakout, the power supply and the housing, came to roughly USD 35. The backend server is a Raspberry Pi 4 (4 GB), which the facility was already using for other purposes. The software stack is entirely open source, under MIT and Apache-style licences. Taken together, this puts the system well within reach of small climbing facilities for which proprietary integrated solutions are simply too expensive.

Discussion

Synchronisation Protocol and Scalability.

The polling-based synchronisation used here is deliberately simple, and it keeps computational pressure on the ESP32 low. That matters because the device has to handle LED matrix updates, training-mode logic and HTTP requests concurrently. A WebSocket or Server-Sent Events (SSE) design would, in principle, give lower latency, but it would also require either a more capable embedded platform or an intermediate broker, both of which add cost and complexity. We settled on the 1.5-second interval empirically, as a compromise between synchronisation accuracy and request load.

Comparison with Commercial Alternatives.

The commercial MoonBoard ecosystem is built around Bluetooth Low Energy (BLE) for board control and a centralised cloud database for route sharing. That model has two well-known drawbacks, namely pairing with single-device and single-session, and route retrieval depends on an internet connection. Our architecture attends both constraints. Everything runs both on the local network and from the Internet, several browsers can connect at once without any pairing step, and routes live on the board's own SD card. The obvious trade-off is that we lose easy access to the global MoonBoard problem database. A reasonable middle ground would be a hybrid approach in which a background process periodically caches benchmark problem data locally, and we view this as a practical direction for future work.

Firmware Stability Considerations.

During the deployment we observed two cases in which the ESP32 became unreachable. Those were solved by first rigorously debugging the training routines. There were several situations detected when the next stone selection led to an infinite loop due to complex automatic route building procedure. The second failure was caused by a supposed error in the ESP32 core. Even though the AutoReconnect of the WiFi class was set to true, the controller did not perform a reconnection. The solution was verifying the WiFi connection status at the end of every loop function execution. In case of lost connection, the reconnection is restarted manually. These two modifications resulted in flawless board operation for more than a week without any detected failures.

Broader Applicability.

We have framed everything here in terms of the MoonBoard, but the architectural pattern itself, namely an ESP32 HTTP API proxied through a larger web backend with polling-based state synchronisation, transfers fairly directly to other LED training boards such as the Kilter Board or the Tension Board, and indeed to other interactive gym equipment. The proxy-within-backend layout also gives a natural place to add authentication, logging, and, further down the line, machine-learning-based route recommendation based on each member's session history.

Conclusion

In this paper there has been described and validated an integrated IoT architecture in which an ESP32-based LED bouldering board controller is connected to a FastAPI gym management backend, supporting multi-client real-time training sessions inside a single membership administration interface. There can be mentioned three main contributions. The first is an HTTP proxy architecture that bridges a resource-constrained embedded server with a more capable web application, while quietly absorbing the embedded server's non-standard TCP handling. The second is a differential state polling protocol that holds timer synchronisation across concurrent browser sessions to under two seconds. The third is a working deployment with hardware costs of roughly USD 35, which we ran for four weeks in a real climbing facility.

References.

1. Climbing Business Journal. (2023). Training board hold systems 2024.
2. International Federation of Sport Climbing. (2021). IFSC 2021 Annual Report.
3. Moon Climbing. (2024). MoonBoard: train hard, climb harder. moonclimbing.com/moonboard.
4. Climbing Business Journal. (2025). Evolutions in climbing gym software: how Redpoint HQ is helping usher in the future.
5. Statista. (2024). Leading climbing facility management software in the United States in 2023. Statista Inc.
6. Giles, L.V., Rhodes, E.C., & Taunton, J.E. (2006). The physiology of rock climbing. Sports Medicine, 36(6), 529-545.
7. Ginszt, M., Saito, M., Zięba, E., Majcher, P., & Kikuchi, N. (2023). Body composition, anthropometric parameters, and strength-endurance characteristics of sport climbers: a systematic review. Journal of Strength and Conditioning Research, 37(6), 1339-1348.
8. Dindorf, C., Bartaguiz, E., Dully, J., Sprenger, M., Becker, S., Fröhlich, M., & Ludwig, O. (2023). In vivo monitoring of acute and intermittent fatigue in sport climbing using near-infrared spectroscopy wearable biosensors. Sports, 11(2), 37.
9. Rum, L., Sten, O., Vendrame, E., et al. (2021). Design of a sensor network for the quantitative analysis of sport climbing. Sensors, 21(5), 1858.
10. Breen, M., Reed, T., Breen, H.M., Osborne, C.T., & Breen, M.S. (2022). Integrating wearable sensors and video to determine microlocation-specific physiologic and motion biometrics – method development for competitive climbing. Sensors, 22(16), 6271.
11. Breen, M., Reed, T., Nishitani, Y., Jones, M., Breen, H.M., & Breen, M.S. (2023). Wearable and non-invasive sensors for rock climbing applications: science-based training and performance optimization. Sensors, 23(11), 5080.
12. Dobles, A. (2017). Machine learning methods for climbing route classification. Stanford University.
13. Tai, C., Wu, A., & Hinojosa, R. (2020). Graph neural networks in classifying rock climbing difficulties. NeurIPS Workshop on Machine Learning for Sports.
14. Stapel, F. (2023). Automated Grade Classification and Route Generation with Affordances on Climbing Training Boards. University of Twente Repository. https://essay.utwente.nl/fileshare/file/94487/Stapel_MA_EEMCS.pdf.
15. Petashvili, D., & Rodda, M. (2023). Board-to-board: evaluating MoonBoard grade prediction generalization. arXiv:2311.12419.
16. Türedioğlu, M. (2023). Model-based route planning and difficulty estimation of indoor bouldering problems. Master's thesis, Middle East Technical University.