

# Jena: Combining Deterministic Rules and LLM Reasoning for Grounded EEG Acquisition Support

Martina Berto<sup>1,\*</sup>, Linas Raičinskis<sup>1</sup> and Aleksandras Voicikas<sup>1,2</sup>

<sup>1</sup>Neurotechnology UAB, Vilnius, Lithuania

<sup>2</sup>Institute of Biosciences, Life Sciences Center, Vilnius University, Lithuania

## Abstract

Large language models (LLMs) offer a promising interface for complex physiological recording systems, but their use in EEG acquisition requires grounding in live device state, curated domain knowledge, and deterministic safeguards. We present Jena, a rule-guided LLM assistant architecture integrated into EEG acquisition software for device setup, troubleshooting, live signal quality monitoring, and results interpretation. The system follows a split design: a local plugin captures runtime context and prepares query-specific payloads, while a server-side backend combines authentication, curated knowledge retrieval, EEG quality evaluation, rule-based controls, and LLM reasoning. Instead of conventional vector-based retrieval, the backend uses LLM-guided navigation over an allowlisted knowledge map, with the aim of making source selection transparent and auditable. We describe the architecture and a signal quality monitoring workflow, illustrating how deterministic rules and constrained LLM reasoning can be combined to keep EEG assistance grounded, bounded, and supportable.

## Keywords

Rules and reasoning, Retrieval-augmented generation, EEG, Signal quality monitoring, Auditable AI

## 1. Introduction and motivation

The possibility of combining conversational AI with real-time biosensor data has gained traction in recent years, with Electroencephalography (EEG) and brain-computer interfaces receiving particular attention [1, 2, 3, 4]. Beyond adapting to neural signals in real time, LLMs have also shown strong performance on related technical tasks — from predicting experimental outcomes [5] to code generation and software engineering [6, 7] — positioning them as versatile collaborators across the full research cycle. Such applications could transform the research and healthcare industry by fundamentally reshaping how scientists and clinicians approach experimental workflows. Yet reliable deployment remains non-trivial: building a trustworthy LLM-based companion requires integrating heterogeneous data sources, from real-time physiological signals, to device specifications, software state, and communication protocols, alongside appropriate data representations and deep domain knowledge. Retrieval-augmented generation (RAG) architectures have begun to address these challenges [8], but reliable deployment also demands a coherent reasoning framework in which rules and constraints are embedded directly into the system architecture, governing what the AI may know, inspect, claim, and expose to users.

Here, we present Jena, an AI-assisted component integrated into EEG acquisition platforms, demonstrated here with BrainAccess Board [9] developed by Neurotechnology UAB. Jena provides real-time signal inspection, data quality feedback, and technical support. Rather than a general-purpose chatbot, the system acts as a constrained acquisition copilot: grounded in curated manufacturer knowledge, reactive to live runtime state, and bounded by deterministic rules.

The novelty of Jena lies in integrating live EEG acquisition state, query-specific EEG payload construction, backend-computed quality measurements, allowlisted manufacturer documentation, and deterministic control rules into a grounded LLM support architecture for EEG acquisition.

---

*RuleML+RR'26: Companion Proceedings of the 10th International Joint Conference on Rules and Reasoning, August 24–26, 2026, Vilnius, Lithuania*

\*Corresponding author.

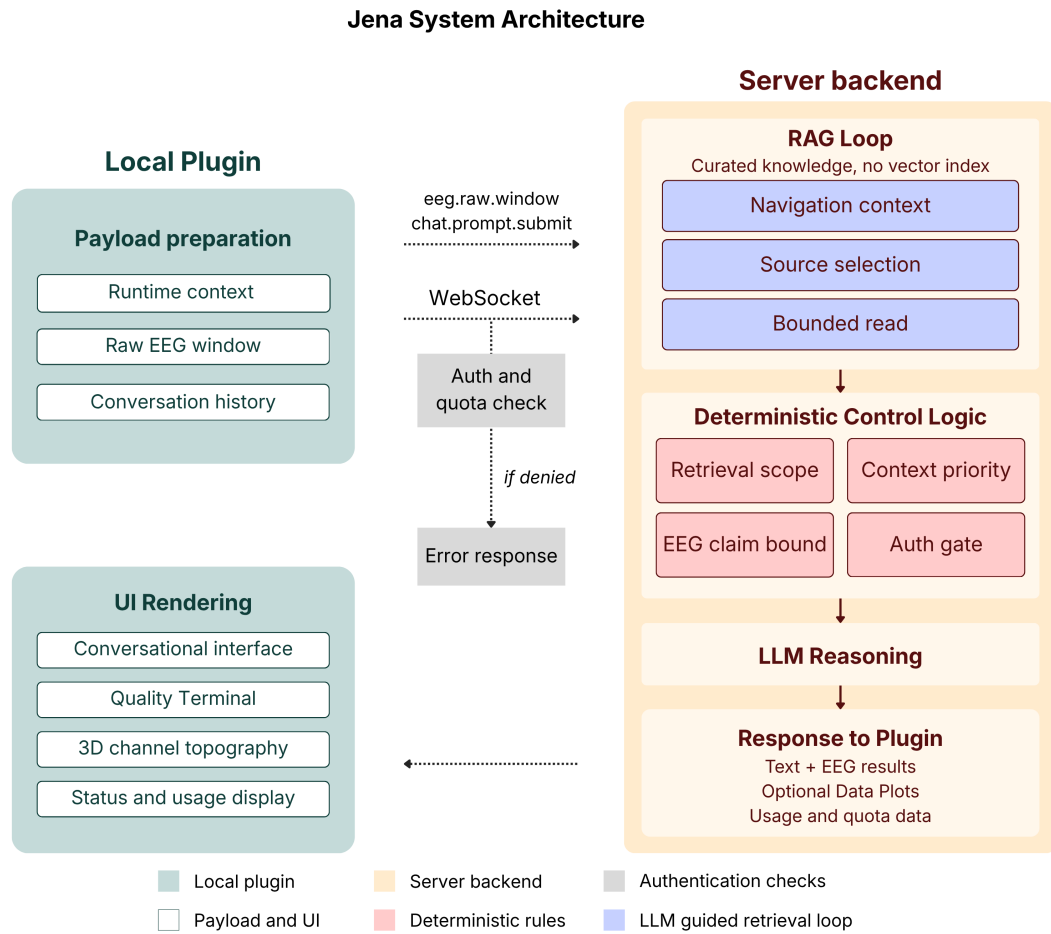
✉ martina@neurotechnology.com (M. Berto); linasr@neurotechnology.com (L. Raičinskis); aleksandrasv@neurotechnology.com (A. Voicikas)



© 2026 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

## 2. System Implementation

Jena follows a split architecture. A local plugin runs close to the desktop acquisition environment, while a server-side backend centralizes reasoning, retrieval, authentication, and administrative functions (Figure 1). This separation reduces the attack surface by centralizing credentials and policy logic server-side, and allows backend behavior to evolve independently of client installations. It also offloads computationally intensive inference to the server, improving resource efficiency of the system [10]. The two components communicate through a structured context payload capturing the current EEG acquisition state. The backend receives and combines this live context with curated documentation provided by the software and hardware manufacturer, stored conversation history, rule-like control logic, EEG quality evaluation, and LLM reasoning to produce grounded operator assistance. Responses are returned to the plugin as text, with optional citations, data quality plots, and server-computed EEG-based summaries.



**Figure 1: Jena system architecture.** The local plugin assembles runtime context, recent EEG data, and conversation history into a structured payload, which is sent to the server backend over a WebSocket connection after authentication and quota checks. The backend uses an LLM-guided retrieval loop to navigate a curated knowledge map and read bounded sections from allowlisted sources, replacing conventional vector similarity search with model-driven source selection. A set of deterministic software rules constrain retrieval scope, context priority, EEG-related claims, and access policy. The grounded response is then returned to the plugin and rendered in the Jena interface with EEG summaries and optional data plots.

## 2.1. Plugin Architecture and Server-Side Reasoning

At interaction time, the local plugin refreshes the runtime context, including connected devices, recording state, impedance mode, active related processes, and warnings. This state determines the query-specific payload. If no EEG device is connected, the model is informed of the device state but no signal payload is attached. If an EEG device is connected and live signal context is required, the plugin may retrieve a recent EEG window, preserve contact and impedance information, and prepare additional acquisition-derived fields depending on the query type and device mode.

The plugin therefore performs both context acquisition and rule-based payload preparation. Live data are filtered to EEG-quality channels, ordered, converted to microvolts, and bounded through device/channel caps, compact binary encoding, and decimation when needed. The resulting payload, comprising runtime context, selected EEG-derived data, and conversation history, is forwarded to the server over a WebSocket connection.

The server combines this payload with curated manufacturer documentation, authentication and quota policies, backend-computed EEG quality information, and LLM reasoning to generate grounded operator assistance. Importantly, the retrieval strategy differs from conventional vector-based RAG pipelines, which typically retrieve top-k embedding-matched chunks before generation [8, 11]. Jena does not maintain a vector index. Instead, the backend exposes a curated knowledge map: an allowlisted inventory of source groups and bounded documents. During inference, the model selects relevant sources from this map and reads bounded sections as evidence. This makes source selection explicit and auditable, avoids index maintenance, and is well suited to structured technical documentation such as manuals, protocol guides, and troubleshooting procedures. However, at larger repository scales, hybrid designs combining curated maps with indexed retrieval may still be preferable. Ongoing work is evaluating this design systematically against indexed RAG baselines, including retrieval quality, latency, and robustness.

## 2.2. Deterministic Control Logic and Safeguards

A central design feature of Jena is the integration of deterministic software rules with probabilistic LLM reasoning. In this work, deterministic software rules refer to explicit non-LLM control logic implemented in the plugin and backend, including routing patterns, access-control policies, source allowlists, quota gates, payload limits, and signal-processing decision points (Table 1). They are deterministic in the practical sense that the same runtime state, configuration, and query classification produce the same system action. The current implementation does not use a standalone declarative rule engine or formal rule language; instead, these rules are implemented as auditable procedural logic around the LLM. We distinguish them from prompt-level constraints, which are server-side instructions that bound how the LLM interprets context, phrases answers, and avoids unsupported claims (Table 1).

On the plugin side, deterministic rules inspect device state, recording state, impedance mode, and available quality measurements to route each query and prepare only the required payload, such as a recent EEG window, impedance values, quality indices, or optional signal plots.

On the server side, deterministic safeguards determine what the LLM may access and use. Authentication, subscription state, and quota checks determine whether a request is permitted; retrieval is restricted to allowlisted sources; and live runtime facts are prioritized over static documentation for device-state questions. EEG-related claims are bounded by data availability and backend-computed measurements, so the model does not independently infer connectivity, signal quality, access rights, or data availability. Prompt-level constraints then shape the LLM response within these software-defined boundaries. They require concise, human-readable answers, prevent claims that disabled UI or device actions were performed, refuse medical interpretation, and instruct the model to acknowledge missing documentation or unavailable measurements rather than guessing.

Together, these mechanisms form a lightweight neuro-symbolic control structure: deterministic software rules define the available data, permitted sources, and claim boundaries, while server-side prompt

**Table 1**  
Representative rule-based constraints implemented around the LLM.

| Category                            | Condition                                              | Action or constraint                                                                                 |
|-------------------------------------|--------------------------------------------------------|------------------------------------------------------------------------------------------------------|
| <i>Deterministic software rules</i> |                                                        |                                                                                                      |
| Device state                        | No EEG device is connected                             | Include device state only; attach no EEG signal payload.                                             |
| EEG payload                         | Live signal context is required                        | Attach a recent EEG window, impedance values, or quality indices depending on device state and mode. |
| Retrieval scope                     | Documentation evidence is required                     | Expose only allowlisted knowledge sources.                                                           |
| Context priority                    | Query concerns current device or recording state       | Prioritize runtime facts over documentation or web evidence.                                         |
| Access/quota                        | User lacks permission or exceeds quota                 | Reject or limit the request before LLM reasoning.                                                    |
| <i>Prompt-level constraints</i>     |                                                        |                                                                                                      |
| Action boundary                     | User asks Jena to perform a UI or device action        | Explain manual steps instead of claiming that an action was performed.                               |
| Medical boundary                    | User requests medical interpretation or diagnosis      | Refuse medical advice and state that the devices are not medically certified.                        |
| Claim boundary                      | Required measurements or documentation are unavailable | Avoid signal-quality or feature claims and state what information is missing.                        |

constraints guide LLM reasoning and response generation within those boundaries. The final response is returned to the plugin and rendered in the Jena interface with usage and quota information, EEG quality results, practical suggestions, and optional data plots.

This design encourages safety, auditability, and verifiable decision support by reducing hallucinated EEG interpretations or unsupported operational claims. In the current implementation, signal quality monitoring is the only fully operational analysis pipeline, but the same rule-guided architecture can be extended to spectral analysis, signal classification, event-related potentials (ERPs), steady-state visual evoked potentials (SSVEPs), and other EEG workflows.

### 2.3. Signal Quality Monitoring as an Illustrative Example

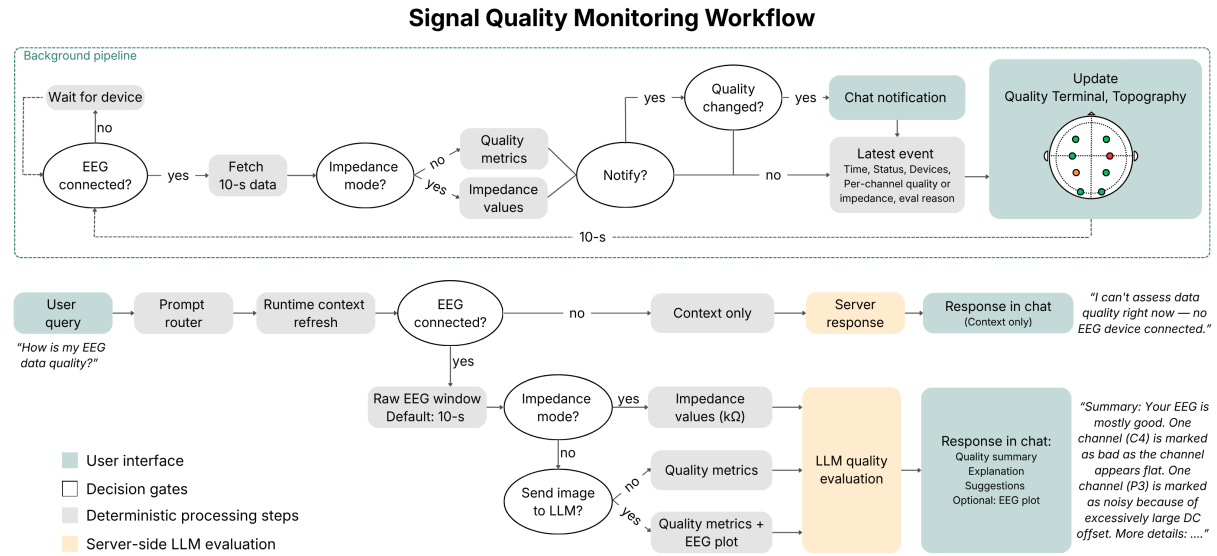
Signal quality monitoring illustrates how Jena combines rule-based control logic, deterministic signal processing, and LLM-based explanation. This workflow first establishes the acquisition state: whether an EEG device is connected, whether impedance mode is active, and which quality measurements are available. These checks determine what information can be included before the model generates a response.

During acquisition, data quality can be affected by poor electrode contact, high impedance, excessive noise, flat channels, and movement artifacts [12, 13]. Jena monitors these factors through a deterministic pipeline running alongside the conversational interface. At regular intervals, the plugin retrieves a recent EEG window, applies the acquisition software’s quality assessment algorithm, and updates the Quality Terminal with the current status, affected channels, impedance-mode state, and a short explanation. Channel-level results are also rendered on a 3D topographic map, where electrodes are color-coded by quality state. If the overall quality changes, the system can notify the user in the chat.

From the operator’s perspective, this appears as an interactive troubleshooting exchange rather than a raw signal-processing report. For example, if the Quality Terminal or topographic map indicates poor contact on one or more electrodes, the user may ask: “Why is my EEG signal bad?”. The prompt router classifies the request as quality-related and triggers a dedicated handling path. If no EEG device is connected, Jena reports the connection state and provides setup guidance without attaching signal data. If a device is connected, the plugin retrieves a recent EEG window, currently 10 seconds by default. In

impedance mode, the payload includes per-channel impedance values; otherwise, it includes software-derived signal-quality indices. When enabled, an EEG plot may also be attached as optional visual context for artifact interpretation [14].

The server then generates a quality explanation from this bounded context. The model does not assess signal quality from unconstrained conversation alone: its response is grounded in device state, impedance-mode status, backend-computed impedance values, software-derived quality indices, optional EEG plots, and the routing decision. The returned response may include a quality summary, likely causes, practical corrective suggestions, and optional visual output. This allows flexible natural-language interaction while keeping EEG-related claims tied to available measurements and explicit system state.



**Figure 2: Signal quality monitoring workflow.** A deterministic background pipeline evaluates EEG quality every 10 seconds, updates the Quality Terminal, and renders channel-level results on a topographic map. For quality-related user queries, the prompt router triggers an on-demand path in which runtime state, impedance mode, quality indices, and optional EEG plots determine the payload sent to the server. The user-facing output is a concise chat response with explanation, suggestions, optional visual evidence, and channel-level quality/topography feedback.

### 3. Discussion and Conclusion

Jena illustrates an architecture for combining LLM reasoning with structured runtime context, curated technical knowledge, and deterministic control logic in EEG acquisition software. Rather than acting as an unconstrained assistant, it is grounded in manufacturer documentation, aware of live device state, and bounded by explicit procedural rules for retrieval scope, access policy, context priority, and EEG-related claims. Although these rules are not implemented as a separate formal rule engine, their conditions and effects can be inspected, tested, and audited.

The signal-quality workflow demonstrates this principle in practice: query intent determines the handling path, only relevant data are included, and EEG-related responses are grounded in available measurements rather than inferred from conversation alone. The LLM-guided retrieval strategy follows the same design goal by replacing precomputed vector search with navigation over an allowlisted knowledge map, making source selection more transparent for structured technical documentation.

Future work will extend the same architecture to richer session-level reports, additional EEG analyses such as power spectra, ERPs, SSVEPs, P300, and frequency-band activity, and other physiological modalities.

Finally, the implementation highlights practical challenges for industrial deployment. Model selection requires balancing answer quality, latency, and cost, while context construction must avoid both excessive irrelevant detail and insufficient evidence. Scaling to many users also requires authentication, quota enforcement, telemetry, caching, and administrative monitoring, making operational infrastructure part of the assistant architecture itself.

**Declaration on Generative AI.** During the preparation of this work, the authors used generative AI tools for coding, grammar, and editing assistance. The authors reviewed and edited the content as needed and take full responsibility for the publication's content.

**Conflict of interest note.** Due to the Track Chair's affiliation with Neurotechnology UAB, this paper was handled independently, with the Chair excluded from its review and decision.

## References

- [1] R. Rajkumar, V. Ganapathy, Bio-inspiring learning style chatbot inventory using brain computing interface to increase the efficiency of e-learning, *IEEE Access* 8 (2020) 67377–67395.
- [2] D. Baradari, N. Kosmyrna, O. Petrov, R. Kaplun, P. Maes, Neurochat: A neuroadaptive ai chatbot for customizing learning experiences, in: *Proceedings of the 7th ACM Conference on Conversational User Interfaces*, 2025, pp. 1–21.
- [3] D. E. Gherman, T. O. Zander, Towards neuroadaptive chatbots: a feasibility study, *Frontiers in Neuroergonomics* 6 (2025) 1589734.
- [4] S. Chandrasekharan, J. E. Jacob, Bridging neuroscience and ai: a survey on large language models for neurological signal interpretation, *Frontiers in Neuroinformatics* 19 (2025) 1561401.
- [5] X. Luo, A. Rechardt, G. Sun, K. K. Nejad, F. Yáñez, B. Yilmaz, K. Lee, A. O. Cohen, V. Borghesani, A. Pashkov, et al., Large language models surpass human experts in predicting neuroscience results, *Nature human behaviour* 9 (2025) 305–315.
- [6] J. Jiang, F. Wang, J. Shen, S. Kim, S. Kim, A survey on large language models for code generation, *ACM Transactions on Software Engineering and Methodology* 35 (2026) 1–72.
- [7] S. Ren, C. Xie, P. Jian, Z. Ren, C. Leng, J. Zhang, Towards scientific intelligence: A survey of llm-based scientific agents, *arXiv preprint arXiv:2503.24047* (2025).
- [8] Y. Gao, Y. Xiong, X. Gao, K. Jia, J. Pan, Y. Bi, Y. Dai, J. Sun, H. Wang, H. Wang, et al., Retrieval-augmented generation for large language models: A survey, *arXiv preprint arXiv:2312.10997* 2 (2023) 32.
- [9] Brainaccess board, <https://www.brainaccess.ai/software/brainaccess-board/>, 2026.
- [10] A. Banitalebi-Dehkordi, N. Vedula, J. Pei, F. Xia, L. Wang, Y. Zhang, Auto-split: A general framework of collaborative edge-cloud ai, in: *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, 2021, pp. 2543–2553.
- [11] G. P. Rusum, S. Anasuri, Vector databases in modern applications: Real-time search, recommendations, and retrieval-augmented generation (rag), *International Journal of AI, BigData, Computational and Management Studies* 5 (2024) 124–136.
- [12] E. S. Kappenman, S. J. Luck, The effects of electrode impedance on data quality and statistical significance in erp recordings, *Psychophysiology* 47 (2010) 888–904.
- [13] X. Han, F. Amiel, C. Wang, W. Hu, Z. Wang, X. Zhang, Impact of data quality on eeg-based biomedical systems, in: *2025 6th International Conference on Bio-engineering for Smart Technologies (BioSMART)*, IEEE, 2025, pp. 1–4.
- [14] P. Arpaia, M. De Luca, L. Di Marino, D. Duran, L. Gargiulo, P. Lanteri, N. Moccaldi, M. Nalin, M. Picciafuoco, R. Robbio, et al., A systematic review of techniques for artifact detection and artifact category identification in electroencephalography from wearable devices, *Sensors* 25 (2025) 5770.