lys_instr: A Python Package for Automating Scientific Measurements

Ziqian Wang [1,2¶], **Hidenori Tsuji**[2], **Toshiya Shiratori** [3], and **Asuka Nakamura** [2,3]

1 Research Institute for Quantum and Chemical Innovation, Institutes of Innovation for Future Society, Nagoya University, Japan **2** RIKEN Center for Emergent Matter Science, Japan **3** Department of Applied Physics, The University of Tokyo, Japan ¶ Corresponding author

DOI: 10.xxxxxx/draft

Software

- Review
- Repository
- Archive

Editor: Open Journals

Reviewers:

- @openjournals

Submitted: 01 January 1970
Published: unpublished

Summary

Modern experiments increasingly demand automation frameworks that coordinate diverse scientific instruments while remaining flexible and customizable. Existing solutions, however, often require explicit management of low-level communication and concurrency, resulting in substantial development overhead. We present `lys_instr`, a Python package that addresses these challenges through an object-oriented, multi-layered architecture for instrument control, workflow coordination, and GUI construction. It enables researchers to rapidly build responsive, asynchronous measurement systems with minimal coding effort. Seamlessly integrated with the `lys` platform (Nakamura, 2023), `lys_instr` unifies experiment control, data acquisition, and visualization, providing an efficient foundation for next-generation automation-driven experimental research.

Statement of need

Modern scientific research increasingly relies on comprehensive measurements across wide parameter spaces to understand physical phenomena. As experiments grow in complexity—with longer measurement times and a greater diversity of instruments—efficient automation has become essential. Measurement automation is evolving beyond simple parameter scans toward informatics-driven, condition-based optimization, paving the way for AI-assisted experimental workflow management. This progress demands robust software infrastructure capable of high integration and flexible logic control.

However, building such a system remains nontrivial for researchers. At the low level, instrument methods tightly coupled to diverse communication protocols (e.g., TCP/IP, VISA, serial, etc.) limit interchangeability and cross-system flexibility. At the high level, coordinating workflows that combine conditional logic, iterative processes, and advanced algorithms across multiple libraries often leads to redundant implementations across contexts, reducing efficiency. Moreover, designing graphical user interfaces (GUIs) for these low- and high-level functionalities typically involves complex multithreading, which requires familiarity with GUI libraries and underlying operating system (OS) event-handling mechanisms. These challenges impose significant substantial development overhead, underscoring the need for a control framework that balances architectural flexibility with reduced implementation complexity.

State of the field

Commercial platforms such as `LabVIEW` (National Instruments Corporation, 2024) and `MATLAB`'s Instrument Control Toolbox (The MathWorks, Inc., 2024) provide mature solutions for instrument communication and measurement execution, but their proprietary environments limit integration with the open-source Python ecosystem and its evolving scientific libraries. Python frameworks including `QCoDeS` (Nielsen et al., 2025), `PyMeasure` (PyMeasure Developers, 2024), and `PyOpticon` (Randall & Majumdar, 2025) offer strong ecosystem compatibility and provide instrument drivers, experiment routines, and GUI components. However, efficient workflow orchestration, particularly for conditional logic, nested procedures, and multithreading, still requires substantial user implementation. Without a dedicated workflow abstraction layer, workflow logic often resides alongside instrument control and GUI code, limiting modular reuse across experiments. Thus, although existing frameworks address specific aspects of instrument control, ecosystem integration, or GUI construction, none unifies hardware abstraction, reusable workflow definition, encapsulated concurrency management, and low-code experiment composition within a single architecture.

To address this gap, `lys_instr` introduces a layered architecture spanning low-level instrument interfaces to high-level GUI integration. In particular, it defines a dedicated workflow layer that abstracts common measurement patterns from device-specific implementations. By separating instrument control, workflow coordination, and GUI components, `lys_instr` enables reusable workflows, built-in hidden multithreading, and low-code construction of complex experiments. This architectural shift—rather than incremental extension of existing frameworks—was necessary to achieve extensibility, coding efficiency, and usability within a coherent system.

Software design

`lys_instr` adopts a three-layer architecture organized by functional separation (Figure 1). Each layer applies object-oriented design patterns from GoF (Gamma et al., 1994), according to its responsibilities, enhancing flexibility, modularity, and usability.

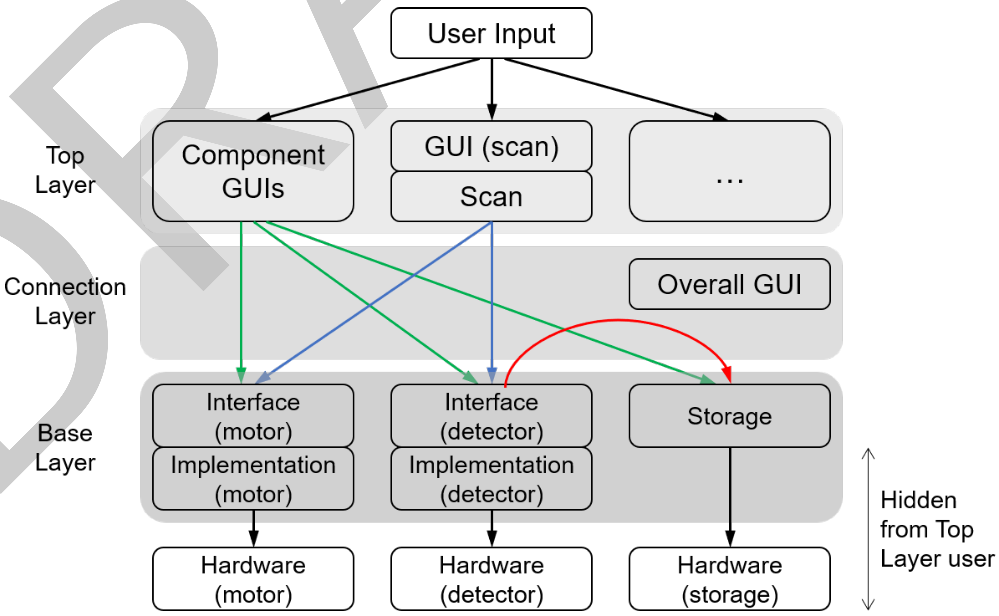

Figure 1: Schematic of the code architecture of `lys_instr`.

1. Base Layer: Device Controller Abstraction

This layer defines abstract interfaces that standardize core instrument controllers. The interfaces encapsulate the concrete implementations, following the *Template Method* design pattern. Typically, most measurement systems include two types of components: *controllers*, which adjust experimental parameters such as external fields, temperature, or physical positions, and *detectors*, which record experimental data, e.g., cameras, spectrometers. Accordingly, `lys_instr` provides standardized *controller* and *detector* interfaces that unify instrument behavior, allowing higher layers to operate on different devices uniformly through common interfaces. Users only need to provide device-specific subclasses that inherit from these interfaces to handle communication with their respective hardware devices. Moreover, each interface manages its own thread(s), ensuring responsiveness and asynchronous operation without blocking other controllers or the GUIs in higher layers. This structure enables users to create controller objects that can be readily integrated into higher-level workflows with minimal device-specific coding.

2. Top Layer: Workflow Coordination

This layer implements workflows common across many setups. Most measurements share similar procedural structures, such as a *scan* process in which data are sequentially recorded while parameters like fields, temperature, or positions are varied. These workflows are standardized using the abstract interfaces defined in the Base Layer, independent of any specific hardware devices, following the *Bridge* and *Composite* design patterns. For example, `lys_instr` provides a standardized *scan* routine that calls *controller* and *detector* interface methods without requiring knowledge of the underlying concrete implementations. This abstraction allows such workflows to be reused across different hardware configurations, greatly improving coding efficiency. In addition, `lys_instr` includes prebuilt GUI components corresponding to each Base Layer component, enabling direct GUI-based control through the same abstract methods. This design cleanly separates workflow logic from device-specific details, simplifying extension to complex measurement systems. Moreover, the GUI communicates with Base Layer interfaces via event-driven messaging, following the *Observer* design pattern to ensure low coupling and high extensibility. With this layer, users can design measurement workflows from scratch without manually creating GUI components.

3. Connection Layer: Control-System Assembly

This layer enables flexible assembly of components from the Base and Top Layers into a complete control system by managing connections within and across layers. Following the *Mediator* design pattern, it connects abstract Base Layer interfaces (and the corresponding hardware devices) to enable automatic data flow, and links GUI components to their respective interfaces, fully hiding device-specific implementations from this layer and above. It also organizes the GUI components into a cohesive application for user interaction. This design grants users maximum freedom to construct tailored control systems without handling low-level tasks such as inter-device communication or multi-threading. Several prebuilt GUI templates for common scenarios are provided for quick hands-on use.

Overall, `lys_instr` provides prebuilt support for standard device controllers, common experimental workflows, and GUI components and assemblies, so users generally need to implement only device-specific subclasses to handle communication with their hardware. This enables rapid integration of new instruments into automated measurement workflows with minimal coding and design effort.

Example of constructed GUI

With `lys_instr`, users can easily construct a GUI like the one shown in Figure 2. In this example, the `lys_instr` window is embedded in the `lys` platform, with Sector A for storage, Sector B for detector, and Sector C for controllers. Multi-dimensional, nested scan sequences can be defined via the visual interface in the `Scan` tab in Sector C. `lys` tools in the outer window tabs allow customization of data display, enabling advanced, on-the-fly customization of data visualization.

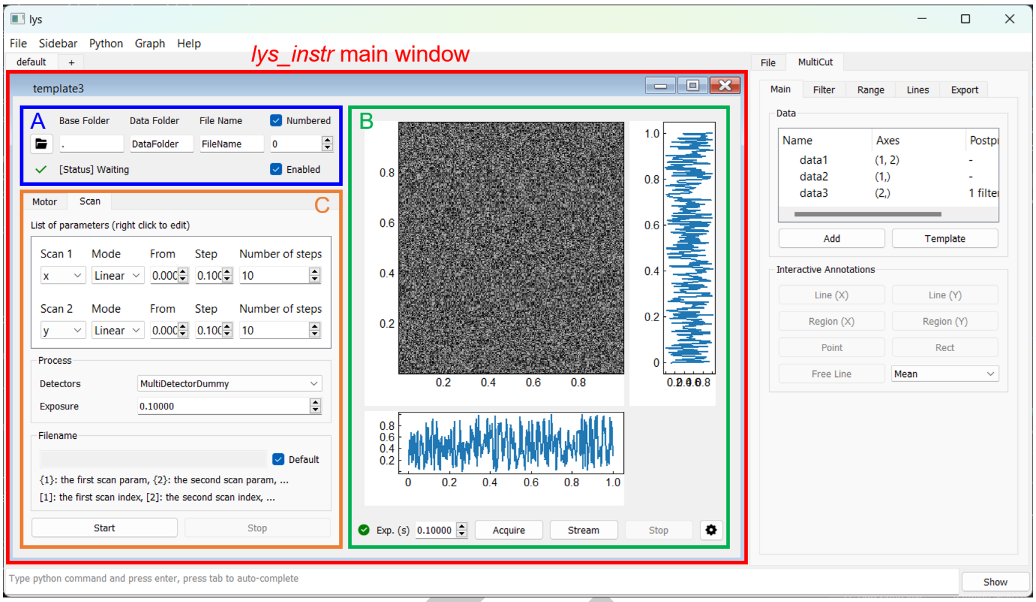

Figure 2: Example GUI of `lys_instr`. The main window, embedded in the `lys` window, contains three sectors: Storage panel (A), Detector panel (B), and controller panel (C). The Scan tab in (C) enables dynamic configuration of multi-dimensional, nested experimental workflows.

Research impact statement

`lys_instr` has been deployed in complex, real-world scientific experiments and has supported multiple peer-reviewed publications. It automates ultrafast electron diffraction (UED) and ultrafast transmission electron microscopy (UTEM) systems, coordinating ultrafast laser excitation and pulsed electron beam detection in pump–probe experiments (Koga et al., 2024; Nakamura et al., 2020, 2021, 2022, 2023; Shimojima et al., 2021, 2023a, 2023b). It enables precise control of electromagnetic lenses and electron deflectors for advanced microscopy involving electron-beam precession, a capability that would be difficult to implement without `lys_instr` (Hayashi et al., 2026; Shiratori et al., 2024).

The software has demonstrated seamless control of transmission electron microscopes from multiple manufacturers across different institutes, including RIKEN Center for Emergent Matter Science and Nagoya University, illustrating reproducible performance and hardware-independent workflow management. Through integration with sister packages in the `lys` family, including `lys_em` (lys_em Developers, 2026) and `lys_fem` (lys_fem Developers, 2026), `lys_instr` supports complex multi-instrument automation within a research-driven ecosystem, enabling efficient deployment of advanced workflows while preserving modularity and extensibility.

AI usage disclosure

Generative AI tools were used to provide debugging suggestions during the final stages of software development. All code was implemented, reviewed, and verified on real hardware, with functionality confirmed through unit tests and experimental validation.

Acknowledgements

We acknowledge valuable comments from Takahiro Shimojima and Kyoko Ishizaka. This work was partially supported by Grant-in-Aid for Scientific Research (KAKENHI) Grants No. 21K13889 and No. 25K00057, and JST PRESTO Grant No. JPMJPR24JA.

The Journal of Open Source Software

References

Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1994). *Design patterns: Elements of reusable object-oriented software*. Addison-Wesley. ISBN: 978-0201633610

Hayashi, S., Han, D., Tsuji, H., Ishizaka, K., & Nakamura, A. (2026). Development of precession lorentz transmission electron microscopy. *Ultramicroscopy*, *280*, 114276. https://doi.org/10.1016/j.ultramic.2025.114276

Koga, J., Chiashi, Y., Nakamura, A., Akiba, T., Takahashi, H., Shimojima, T., Ishiwata, S., & Ishizaka, K. (2024). Unusual photoinduced crystal structure dynamics in $TaTe_2$ with double zigzag chain superstructure. *Applied Physics Express*, *17*(4), 042007. https://doi.org/10.35848/1882-0786/ad3b61

lys_em Developers. (2026). *Lys_em: Python package for electron microscopy control and data analysis*. https://github.com/a-tock/lys_em

lys_fem Developers. (2026). *Lys_fem: Python package for finite-element modeling and experiment integration*. https://github.com/lys-devel/lys_fem

Nakamura, A. (2023). lys: Interactive Multi-Dimensional Data Analysis and Visualization Platform. *Journal of Open Source Software*, *8*(92), 5869. https://doi.org/10.21105/joss.05869

Nakamura, A., Shimojima, T., Chiashi, Y., Kamitani, M., Sakai, H., Ishiwata, S., Li, H., & Ishizaka, K. (2020). Nanoscale Imaging of Unusual Photoacoustic Waves in Thin Flake VTe2. *Nano Letters*, *20*(7), 4932–4938. https://doi.org/10.1021/acs.nanolett.0c01006

Nakamura, A., Shimojima, T., & Ishizaka, K. (2021). Finite-element Simulation of Photoinduced Strain Dynamics in Silicon Thin Plates. *Structural Dynamics*, *8*(2), 024103. https://doi.org/10.1063/4.0000059

Nakamura, A., Shimojima, T., & Ishizaka, K. (2022). Visualizing Optically-Induced Strains by Five-Dimensional Ultrafast Electron Microscopy. *Faraday Discussions*, *237*, 27–39. https://doi.org/10.1039/D2FD00062H

Nakamura, A., Shimojima, T., & Ishizaka, K. (2023). Characterizing an Optically Induced Sub-micrometer Gigahertz Acoustic Wave in a Silicon Thin Plate. *Nano Letters*, *23*(7), 2490–2495. https://doi.org/10.1021/acs.nanolett.2c03938

National Instruments Corporation. (2024). *LabVIEW*. National Instruments Corporation. https://www.ni.com/en-us/shop/labview.html

Nielsen, J. H., Astafev, M., Nielsen, W. H. P., & others. (2025). *QCoDeS: Modular data acquisition framework* (Version v0.54.2). https://doi.org/10.5281/zenodo.17459861

PyMeasure Developers. (2024). *PyMeasure: Scientific measurement library for instruments, experiments, and live-plotting.* (Version 0.14.0). Zenodo. https://doi.org/10.5281/zenodo.11241567

Randall, R., & Majumdar, A. (2025). PyOpticon: An open-source python package for laboratory control, automation, and visualization. *Chemistry of Materials*, *37*(13), 4585–4592. https://doi.org/10.1021/acs.chemmater.5c00644

Shimojima, T., Nakamura, A., & Ishizaka, K. (2023a). Development of Five-Dimensional Scanning Transmission Electron Microscopy. *Review of Scientific Instruments*, *94*(2), 023705. https://doi.org/10.1063/5.0106517

Shimojima, T., Nakamura, A., & Ishizaka, K. (2023b). Development and Applications of Ultrafast Transmission Electron Microscopy. *Microscopy*, *72*(4), 287–298. https://doi.org/10.1093/jmicro/dfad021

Shimojima, T., Nakamura, A., Yu, X., Karube, K., Taguchi, Y., Tokura, Y., & Ishizaka, K. (2021). Nano-to-Micro Spatiotemporal Imaging of Magnetic Skyrmion's Life Cycle. *Science Advances*, *7*(25), eabg1322. https://doi.org/10.1126/sciadv.abg1322

Shiratori, T., Koga, J., Shimojima, T., Ishizaka, K., & Nakamura, A. (2024). Development of ultrafast four-dimensional precession electron diffraction. *Ultramicroscopy*, *267*, 114064. https://doi.org/10.1016/j.ultramic.2024.114064

The MathWorks, Inc. (2024). *MATLAB instrument control toolbox*. The MathWorks, Inc. https://www.mathworks.com/products/instrument.html