








lys_instr: A Python Package for Automating Scientific Measurements

Ziqian Wang^{1,2}[✉], Hidenori Tsuji², Toshiya Shiratori³, 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](https://doi.org/10.xxxxxx/draft)

Software

- [Review](#) 
- [Repository](#) 
- [Archive](#) 

Editor: [Open Journals](#) 

Reviewers:

- [@openjournals](#)

Submitted: 01 January 1970

Published: unpublished

License

Authors of papers retain copyright and release the work under a

Creative Commons Attribution 4.0 International License ([CC BY 4.0](#))

Summary

Modern experiments increasingly demand automation frameworks capable of coordinating diverse scientific instruments while remaining flexible and easy to customize. Existing solutions, however, often require substantial adaptation or manual handling of low-level communication and threading. 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, offering 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 fully 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 now 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, specific instrument methods tightly coupled to diverse communication protocols (e.g., TCP/IP, VISA, serial, etc.) limit interchangeability and flexibility across systems. At the high level, coordinating workflows involving conditional logic, iterative processes, and advanced algorithms from different libraries can lead to redundant implementations of similar functionality across different contexts. Moreover, designing graphical user interfaces (GUIs) for these low- and high-level functionalities typically involves complex multithreading, which requires familiarity with GUI libraries such as Qt and the underlying operating system (OS) event-handling mechanisms. Existing frameworks such as QCoDeS (Nielsen & others, 2025), PyMeasure (developers, 2025), PyLabControl (Steiner & LISE-B26, 2024), LabVIEW (LabVIEW, 2024), and MATLAB's Instrument Control Toolbox (MATLAB Instrument Control Toolbox, 2024) provide powerful ecosystems for instrument control and measurement scripting, but require users to handle low-level communications and high-level workflow logic themselves. These challenges impose substantial overhead on researchers designing custom measurement systems.

To address these issues, we introduce `lys_instr`—an object-oriented framework that abstracts common control patterns from experiment-specific implementations, reducing coding and

42 design costs while enabling flexible and efficient automation.

43 Design philosophy

44 `lys_instr` adopts a three-layer architecture based on levels of abstraction, separating individual
45 instrument control, workflow coordination, and control system assembly. Each layer applies
46 object-oriented design patterns in GoF (Gamma et al., 1994), suited to its role, enhancing
47 flexibility, modularity, and ease of use. The framework builds on the `lys` platform, leveraging
48 its powerful multidimensional data visualization capabilities.

49 1. Base Layer: Individual Instrument Control

50 This layer provides low-level abstractions that standardizes core instrument functionalities—pri-
51 marily motors and detectors. Measurement systems typically involve two types of controllers:
52 *motors*, which adjust experimental parameters such as external fields, temperature, or physi-
53 cal positions, and *detectors*, which record experimental data, e.g., cameras, spectrometers.
54 Concrete device implementations are separated from these interfaces, following the *Template*
55 *Method* design pattern. Independent automatic threading allows each instrument to operate
56 asynchronously, ensuring responsive operation without blocking other instruments or the GUI.
57 Users can create standardized objects for integration into `lys_instr` with minimal device-
58 specific coding. Additionally, every Base Layer component includes a GUI by default, removing
59 the need for users to create one manually.

60 2. Intermediate Layer: Workflow Coordination

61 This layer provides high-level abstractions, including the GUI, to coordinate Base Layer
62 instruments for common experimental tasks. Many measurements share similar workflows. A
63 representative example would be a *scan* process, recording experimental data sequentially while
64 scanning parameters such as fields, temperature, or positions. Owing to the standardization of
65 the instrument in the base layer, these similar workflows can be implemented with common
66 algorithms by *Bridge* and *Composite* design patterns. `lys_instr` provides a set of such
67 high-level functionalities commonly required across experiments. Moreover, the GUI interacts
68 with components via signals, following the *Observer* design pattern, ensuring low coupling and
69 high extensibility.

70 3. Top Layer: Control-System Assembly

71 This layer enables flexible assembly of components from the Base and Intermediate Layers
72 into a complete control system, on both the character user interface (CUI) and GUI levels.
73 Following the *Mediator* design pattern, it manages connections among abstract devices (and,
74 through the Base Layer, the corresponding real hardware) to enable automatic data flow, and
75 organizes the GUI for user interaction. This grants users maximum freedom to construct
76 tailored control systems without managing complex aspects such as inter-device communication
77 or multi-threading. Several typical GUI templates are also provided for quick hands-on use.

78 Overall, `lys_instr` provides prebuilt support for standard instruments, common workflows,
79 and GUI components, so users generally need to implement only device-specific interfaces for
80 hardware communication for automating their measurement workflows.

81 Example of Constructed GUI

82 With `lys_instr`, users can easily construct a GUI like the one shown in Figure 1. The
83 `lys_instr` window is embedded in the `lys` subwindow, with Sector A for storage, Sector B
84 for detector, and the Motor tab in Sector C. Multi-dimensional, nested scan sequences can
85 be defined via the visual interface in the Scan tab in Sector C. `lys` tools in the outer window
86 tabs allow customization of data display, enabling advanced, on-the-fly customization of data
87 visualization.

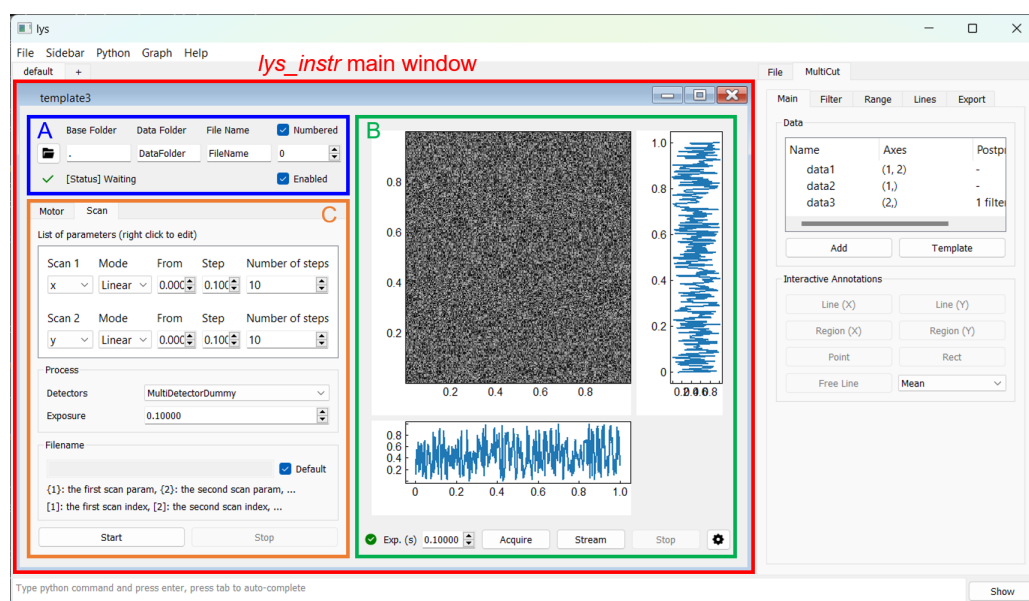


Figure 1: Example GUI of `lys_instr`. The main window, embedded in the `lys` window, is organized into three sectors: Storage panel (A), Detector panel (B), and Motor and Scan tabs (C). The Scan tab enables dynamic configuration of multi-dimensional, nested experimental workflows.

Projects using the software

`lys_instr` has been deployed in complex, real-world scientific instruments, supporting multiple peer-reviewed publications. It automates Ultrafast Transmission Electron Microscopy (UTEM) at the RIKEN Center for Emergent Matter Science, 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 achieve without `lys_instr` (Hayashi et al., 2025; Shiratori et al., 2024).

Acknowledgements

We acknowledge valuable comments from Takahiro Shimojima and Kyoko Ishizaka. This work was partially supported by a Grant-in-Aid for Scientific Research (KAKENHI) (Grant No.).

References

- developers, P. (2025). *PyMeasure*. <https://github.com/pymasure/pymasure>
- Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1994). *Design patterns: Elements of reusable object-oriented software*. Addison-Wesley.
- Hayashi, S., Han, D., Tsuji, H., Ishizaka, K., & Nakamura, A. (2025). Development of precession lorentz transmission electron microscopy. *arXiv Preprint arXiv:2505.05790*. <https://arxiv.org/abs/2505.05790>
- Koga, J., Chiashi, Y., Nakamura, A., Akiba, T., Takahashi, H., Shimojima, T., Ishiwata, S., & Ishizaka, K. (2024). Unusual photoinduced crystal structure dynamics in TaTe₂ with double zigzag chain superstructure. *Applied Physics Express*, 17(4), 042007. <https://doi.org/10.35848/1882-0786/ad3b61>

- 111 *LabVIEW*. (2024). [Computer software]. National Instruments Corporation. <https://www.ni.com/en-us/shop/labview.html>
- 112
- 113 *MATLAB instrument control toolbox*. (2024). [Computer software]. The MathWorks, Inc. <https://www.mathworks.com/products/instrument.html>
- 114
- 115 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>
- 116
- 117
- 118 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>
- 119
- 120
- 121 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>
- 122
- 123
- 124 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>
- 125
- 126
- 127 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>
- 128
- 129
- 130 Nielsen, J. H., & others. (2025). *QCoDeS* (Version 0.52.0). <https://doi.org/10.5281/zenodo.15144297>
- 131
- 132 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>
- 133
- 134
- 135 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>
- 136
- 137
- 138 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>
- 139
- 140
- 141 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.ultramicro.2024.114064>
- 142
- 143
- 144 Steiner, J. F., & LISE-B26. (2024). *PyLabControl: A scientific instrument control library in python*. <https://github.com/LISE-B26/pylabcontrol>.
- 145