








# lys\_instr: A Python Package for Automating Scientific Measurements

Ziqian Wang<sup>1,2</sup>, Hidenori Tsuji<sup>2</sup>, Toshiya Shiratori<sup>3</sup>, and Asuka Nakamura<sup>2,3</sup>

<sup>1</sup> Research Institute for Quantum and Chemical Innovation, Institutes of Innovation for Future Society, Nagoya University, Japan  <sup>2</sup> RIKEN Center for Emergent Matter Science, Japan  <sup>3</sup> 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Published: unpublished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 GUIs for these low- and high-level functionalities typically requires complex multithreading, further increasing implementation costs. Existing frameworks such as `asQCoDeS` (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 design costs while enabling flexible and efficient automation.

## Design Philosophy

lys\_instr adopts a three-layer architecture based on multiple object-oriented design patterns to maximize flexibility, modularity and ease of use.

### 1. Base Layer: Instrument Abstraction

This layer standardizes core instrument functionalities—motors, detectors, and storage—through abstract interfaces. Concrete device implementations are separated from these interfaces, following the *Template Method* design pattern. Independent automatic multi-threading management allows each instrument to operate asynchronously, ensuring responsive operation without blocking other instruments or the GUI.

### 2. Intermediate Layer: Workflow Coordination

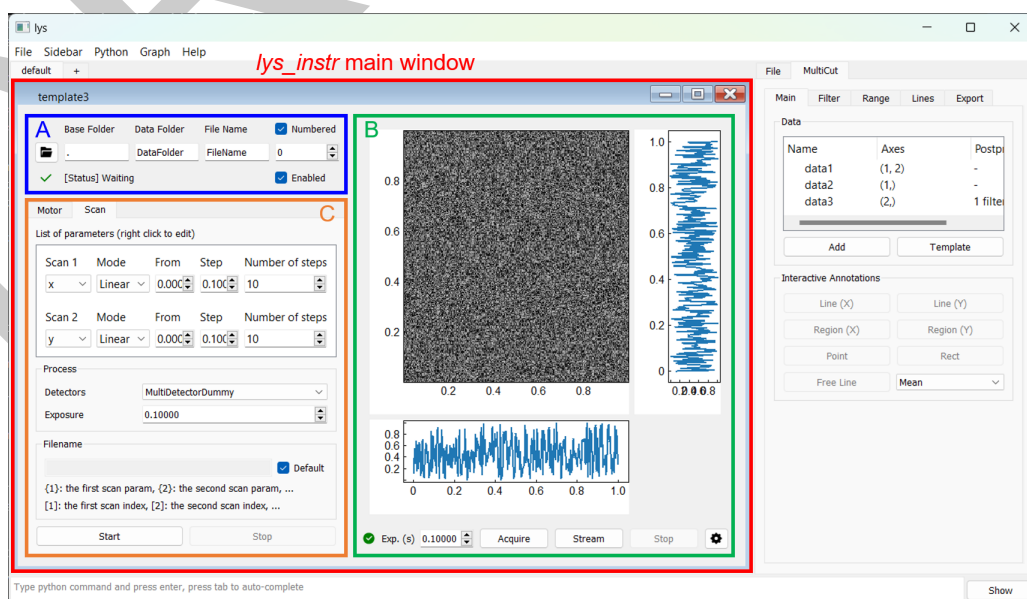
This layer provides high-level abstractions, including the GUI, to coordinate base instruments for general experimental tasks. It standardizes common operations such as *move-and-detect*, scans, and nested scan sequences without requiring knowledge of concrete instrument implementations, following the *Bridge* design pattern. The GUI interacts with components via signals, adhering to the *Observer* design pattern, ensuring low coupling and high extensibility. Concepts from the *Composite* design pattern are also employed to efficiently manage nested scan configurations.

### 3. Top Layer: Control System Assembly

This layer supports flexible assembly of the measurement GUI. Following the *Mediator* design pattern, it manages connections among abstract devices (and, through the Base Layer, the corresponding real hardware) to enable automatic data flow, and organizes the GUI for user control. This grants users maximum freedom to construct tailored control systems without managing complex aspects such as inter-device communication or multi-threading.

## Key Functionalities

lys\_instr provides a straightforward user interface—illustrated in the preliminary example in Figure 1—for integrated instrument/data management and declarative workflow control.



**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.

67 **Integrated Instrument/Data Management:** The *Base Layer* ensures asynchronous operation  
68 across all instruments (Sector A for storage, Sector B for detector, and the *Motor* tab in C),  
69 keeping each GUI component responsive during cooperative measurements.

70 **Declarative Workflow Management:** Users can define multi-dimensional, nested scan sequences  
71 (*MultiScan*) via a visual interface (the *Scan* tab in Sector C). Built on the *Intermediate*  
72 *Layer* abstraction, these workflows hierarchically coordinate motors and detectors, enabling  
73 sophisticated experiments without any low-level coding.

74 In addition, `lys_instr` supports user-defined GUI layouts through the Top Layer. It further  
75 enhances extensibility through seamless integration with the `lys` platform (the outer window  
76 with tabs), enabling advanced on-the-fly customization of data visualization.

## 77 Projects using the software

78 `lys_instr` has been deployed in complex, real-world scientific instruments, supporting multiple  
79 peer-reviewed publications. It automates Ultrafast Transmission Electron Microscopy (UTEM)  
80 at RIKEN Center for Emergent Matter Science, coordinating ultrafast laser excitation and  
81 pulsed electron beam detection in pump-probe experiments (Nakamura et al., 2020, 2021,  
82 2022, 2023; Shimojima et al., 2021, 2023a, 2023b), and it controls electromagnetic lenses and  
83 electron deflectors for advanced microscopy with electron-beam precession (?; ?).

## 84 Acknowledgements

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

87 developers, P. (2025). *PyMeasure*. <https://github.com/pymasure/pymasure>

88 *LabVIEW*. (2024). [Computer software]. National Instruments Corporation. <https://www.ni.com/en-us/shop/labview.html>

90 *MATLAB instrument control toolbox*. (2024). [Computer software]. The MathWorks, Inc.  
91 <https://www.mathworks.com/products/instrument.html>

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

95 Nakamura, A., Shimojima, T., Chiashi, Y., Kamitani, M., Sakai, H., Ishiwata, S., Li, H., &  
96 Ishizaka, K. (2020). Nanoscale Imaging of Unusual Photoacoustic Waves in Thin Flake  
97 VTe<sub>2</sub>. *Nano Letters*, 20(7), 4932–4938. <https://doi.org/10.1021/acs.nanolett.0c01006>

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

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

104 Nakamura, A., Shimojima, T., & Ishizaka, K. (2023). Characterizing an Optically Induced  
105 Sub-micrometer Gigahertz Acoustic Wave in a Silicon Thin Plate. *Nano Letters*, 23(7),  
106 2490–2495. <https://doi.org/10.1021/acs.nanolett.2c03938>

107 Nielsen, J. H., & others. (2025). *QCoDeS* (Version 0.52.0). <https://doi.org/10.5281/zenodo.15144297>

109 Shimojima, T., Nakamura, A., & Ishizaka, K. (2023a). Development of Five-Dimensional

- 110 Scanning Transmission Electron Microscopy. *Review of Scientific Instruments*, 94(2),  
111 023705. <https://doi.org/10.1063/5.0106517>
- 112 Shimojima, T., Nakamura, A., & Ishizaka, K. (2023b). Development and Applications  
113 of Ultrafast Transmission Electron Microscopy. *Microscopy*, 72(4), 287–298. <https://doi.org/10.1093/jmicro/dfad021>  
114
- 115 Shimojima, T., Nakamura, A., Yu, X., Karube, K., Taguchi, Y., Tokura, Y., & Ishizaka, K.  
116 (2021). Nano-to-Micro Spatiotemporal Imaging of Magnetic Skyrmion's Life Cycle. *Science*  
117 *Advances*, 7(25), eabg1322. <https://doi.org/10.1126/sciadv.abg1322>
- 118 Steiner, J. F., & LISE-B26. (2024). *PyLabControl: A scientific instrument control library in*  
119 *python*. <https://github.com/LISE-B26/pylabcontrol>.

DRAFT