




# NOMAD CAMELS: Configurable Application for Measurements, Experiments and Laboratory Systems

Alexander D. Fuchs <sup>1,2\*</sup>, Johannes A. F. Lehmeyer <sup>1,2\*</sup>, Heinz Junkes <sup>3</sup>, Heiko B. Weber <sup>1</sup>, and Michael Krieger <sup>1¶</sup>

<sup>1</sup> Lehrstuhl für Angewandte Physik, Department Physik, Friedrich-Alexander Universität Erlangen-Nürnberg, Germany. <sup>2</sup> Physics Department and CSMB, Humboldt-Universität zu Berlin, Berlin, Germany. <sup>3</sup> Fritz-Haber-Institut, Berlin, Germany. ¶ Corresponding author \* These authors contributed equally.

DOI: [10.21105/joss.06371](https://doi.org/10.21105/joss.06371)

## Software

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

Editor: [Sarath Menon](#) 

## Reviewers:

- [@NicolasCARPi](#)
- [@ktahar](#)

Submitted: 08 December 2023

Published: 07 March 2024

## License

Authors of papers retain copyright and release the work under a Creative Commons Attribution 4.0 International License ([CC BY 4.0](#)).

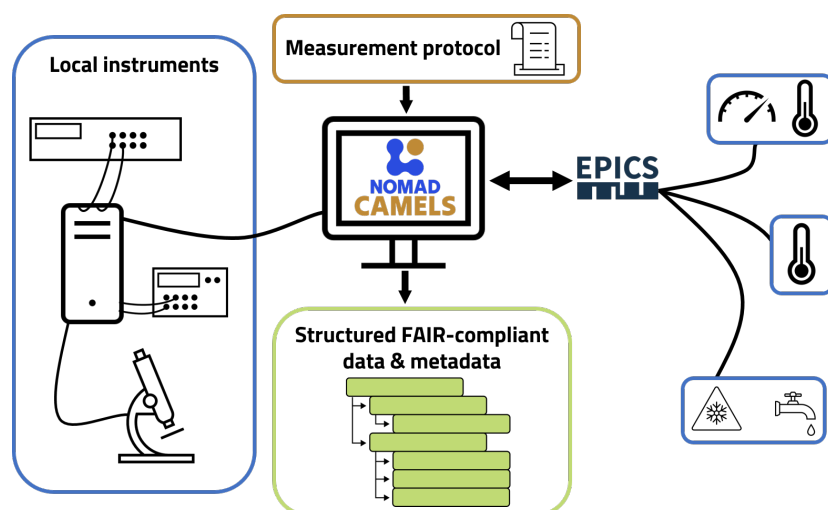
## Summary

NOMAD CAMELS (short: CAMELS) is a configurable, open-source measurement software that records fully self-describing experimental data. It has its origins in the field of experimental physics where a wide variety of measurement instruments are used in frequently changing experimental setups and measurement protocols. CAMELS provides a graphical user interface (GUI) which allows the user to configure experiments without the need of programming skills or deep understanding of instrument communication. CAMELS translates user-defined measurement protocols into stand-alone executable Python code for full transparency of the actual measurement sequences. Existing large-scale, distributed control systems (e.g. EPICS ([EPICS, 2023](#))) can be natively implemented. CAMELS is designed with a focus on full recording of data and metadata. When shared with others, data produced with CAMELS allow full understanding of the measurement and the resulting data in accordance with the FAIR (Findable, Accessible, Interoperable and Re-usable) principles ([Wilkinson et al., 2016](#)).

## Statement of need

Research data management has piqued greater and greater interest in recent years. Today, research funding agencies demand sustainable research data strategies. The key criterion is to create research data following the FAIR principles and thereby improve world-wide data-driven research ([DFG Positionspapier, 2018](#)). While one ingredient, electronic lab notebooks (ELNs), are an important step towards FAIR data, it is equally important to record the measurement data and related metadata along FAIR principles as early as possible in the research workflow.

In experimental physics many custom-built measurement setups are controlled by very specific software written by individual researchers. This results in a heterogeneous landscape of software fragments for measurements written in many different languages and with often incomplete documentation, making it almost impossible for other researchers to extend existing code. The degree to which the stored raw data is understandable varies greatly but is often unintelligible even for researchers from the same lab. Important metadata such as instrument settings or the actual measurement steps performed to obtain the final raw data are rarely recorded, making it virtually impossible to reproduce experiments. Therefore, the documentation of experiments is incomplete, preventing FAIR research data. Although there are some tools available (e.g. *SweepMe!* ([SweepMe, 2023](#)), *iC* ([Pernstich, 2012](#)), PyMoDAQ ([PyMoDAQ, 2023](#))) to realise control of arbitrary measurement instruments, they are frequently not open-source or their data output is not compliant with the FAIR principles.



**Figure 1:** Visualization of CAMELS functionality and workflow. CAMELS connects directly with local instruments and/or large-scale lab infrastructure running network protocols, e.g. EPICS. Customizable measurements protocols are translated into Python code and executed. The output is FAIR-compliant measurement data.

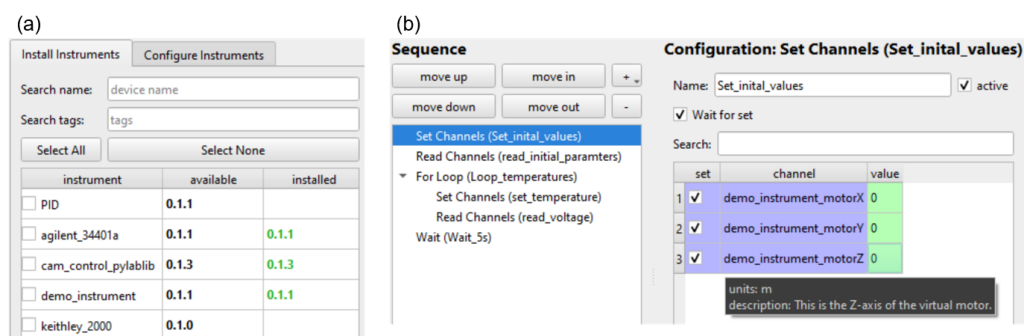
## NOMAD CAMELS

CAMELS is an open-source tool that automatically collects all computer-accessible experimental metadata. It features a user-friendly graphical interface that enables the creation and customization of measurements without the need for programming knowledge. By default, the data is stored in a structured HDF5 file format that closely resembles the structure of the NeXus standard ([FAIRmat NeXus Proposal, 2023](#); [Könnecke et al., 2015](#)). The final HDF5 file contains both the actual measurement data and metadata in a single file, compliant to the FAIR principles.

CAMELS is a stand-alone desktop application. It allows for direct access to the *NOMAD* repository ([Scheidgen, Brückner, et al., 2023](#); [Scheidgen, Himanen, et al., 2023](#)) or its on-premise installation called *NOMAD Oasis* enabling direct linking to electronic lab notebook entries. The user can for example connect measurements to previous experiment workflows documented in *NOMAD* ELNs. CAMELS can subsequently upload measurement results directly into the ELN providing a simple and stream-lined data workflow.

CAMELS builds on *bluesky* ([Allan et al., 2019](#); [Bluesky, 2023](#)) initially developed to control instruments at large-scale research institutions using EPICS ([EPICS, 2023](#); [Knott et al., 1994](#)). In CAMELS, *bluesky* is employed to orchestrate the instrument communication either directly (e.g. via PyVISA ([PyVISA, 2023](#))) or via using network protocols. Existing lab infrastructure controlled by EPICS is therefore immediately accessible. A schematic overview of the functionality of CAMELS is displayed in [Figure 1](#).

CAMELS provides a comprehensive set of functionalities that can be split into three primary components: instrument management, measurement protocols, and manual controls.



**Figure 2:** (a) The instrument manager allows to install and configure instrument drivers from the curated instrument driver library or the user's hard drive. (b) The measurement protocol editor allows users to configure arbitrary measurement sequences.

## Instrument Management

Scientific instruments can be added to CAMELS in two ways: The first involves the *instrument manager* (c.f. Figure 2a) to add instruments from the official curated driver repository (*CAMELS-drivers*, 2023). These drivers are installed into the Python environment via *pip* (*Pip*, 2023) with each driver being packaged individually.

The second way is to add self-built drivers by creating the necessary files locally and placing these in the directory specified in the CAMELS settings. To facilitate this process CAMELS provides a *driver builder* that automatically generates the essential structure and boilerplate code. As CAMELS is an open-source project developed by and for the community, users are encouraged to contribute to the driver library by creating pull requests for new drivers on the GitHub repository (*CAMELS-drivers*, 2023).

In general, a CAMELS driver comprises two files: One containing the hardware interface communication, the other one defining the available instrument settings. Data communication to instruments is handled via *channels* that can be set and/or read; they correspond to an instrument's individual functionality or physical property.

## Measurement protocols

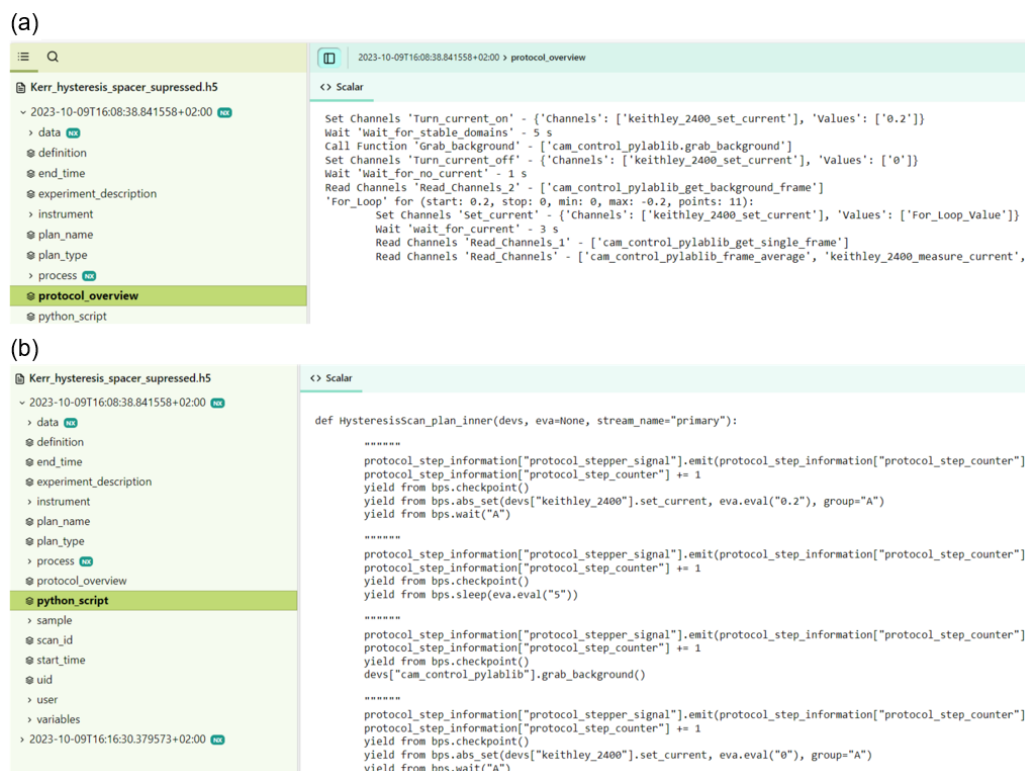
In CAMELS a *measurement protocol* is a distinctive sequence of individual steps including setting and reading instrument channels (see Figure 2b), loops, conditional execution, running sub-protocols, PID control, etc. This yields a measurement in a 'recipe-style' format, where the next step is usually executed after the successful completion of the preceding step. Asynchronous data acquisition is supported.

CAMELS translates the protocol created in the GUI into a Python script, which is then executed. The script can be viewed, run independently of CAMELS, and modified if required. CAMELS protocols and settings can be stored and shared with colleagues enabling easy repeatability of experiments.

## Manual controls

Certain scientific instruments require manual control before starting predefined measurement routines, e.g. adjusting stages, controlling temperature, valves, pumps, etc. In CAMELS this is achieved through the *manual controls* functionality which can be applied to any writable instrument channel.

## Data output



**Figure 3:** CAMELS stores measurement data together with rich metadata collected automatically into a structured HDF5 file by default. This includes (a) a human readable measurement protocol summary and (b) the executable Python script that was used to actually record the data. This allows others to understand the data acquisition and to reproduce the experiment.

After executing the measurement protocol, the time-stamped data is by default saved to an HDF5 file with a structure similar to the NeXus standard (Könnecke et al., 2015). Data can also be exported in CSV format with the metadata exported in JSON.

The stored data can be divided into distinct sections:

- Time-stamped raw data obtained during the execution of the measurement protocol.
- Instrument settings.
- Human-readable summary of the measurement protocol information (see Figure 3a).
- Complete Python script that recorded the data (see Figure 3b) as well as information on the Python environment, i.e. a list of used packages and versions.
- User-defined metadata, e.g. sample and user information.

## Documentation

In-depth documentation and guides for installing, using and troubleshooting can be found on the [CAMELS documentation webpage](#) (Fuchs & Lehmeyer, 2023).

## Acknowledgements

We thank Patrick Oppermann (Fritz-Haber-Institut der Max-Planck-Gesellschaft) for valuable discussions.

NOMAD CAMELS is being developed within the NFDI consortium *FAIRmat* funded by the Deutsche Forschungsgemeinschaft “DFG, German Research Foundation”, project 460197019.

## References

- Allan, D., Caswell, T., Campbell, S., & Rakitin, M. (2019). Bluesky's Ahead: A Multi-Facility Collaboration for an a la Carte Software Project for Data Acquisition and Management. *Synchrotron Radiat. News*, 32(3), 19–22. <https://doi.org/10.1080/08940886.2019.1608121>
- Bluesky Project. (2023). <https://blueskyproject.io/>
- CAMELS\_drivers: instrument implementation for CAMELS. (2023). [https://github.com/FAU-LAP/CAMELS\\_drivers](https://github.com/FAU-LAP/CAMELS_drivers)
- DFG Positionspapier: Förderung von Informationsinfrastrukturen für die Wissenschaft. (2018). [https://www.dfg.de/download/pdf/foerderung/programme/lis/positionspapier\\_informationsinfrastrukturen.pdf](https://www.dfg.de/download/pdf/foerderung/programme/lis/positionspapier_informationsinfrastrukturen.pdf)
- EPICS - Experimental Physics and Industrial Control System. (2023). <https://epics-controls.org/>
- FAIRmat NeXus proposal. (2023). <https://fairmat-nfdi.github.io/nexus-fairmat-proposal/9636feecb79bb32b828b1a9804269573256d7696/fairmat-cover.html>
- Fuchs, A. D., & Lehmeier, J. A. F. (2023). *NOMAD CAMELS documentation*. <https://fau-lap.github.io/NOMAD-CAMELS/>
- Knott, M., Gurd, D., Lewis, S., & Thuot, M. (1994). EPICS: A control system software co-development success story. *Nucl. Inst. Methods Phys. Res. A*, 352(1-2), 486–491. [https://doi.org/10.1016/0168-9002\(94\)91577-6](https://doi.org/10.1016/0168-9002(94)91577-6)
- Könnecke, M., Akeroyd, F. A., Bernstein, H. J., Brewster, A. S., Campbell, S. I., Clausen, B., Cottrell, S., Hoffmann, J. U., Jemian, P. R., Männicke, D., Osborn, R., Peterson, P. F., Richter, T., Suzuki, J., Watts, B., Wintersberger, E., & Wuttke, J. (2015). The NeXus data format. *J. Appl. Crystallogr.*, 48(1), 301–305. <https://doi.org/10.1107/S1600576714027575>
- Pernstich, K. P. (2012). Instrument Control (iC) – An Open-Source Software to Automate Test Equipment. *Journal of Research of the National Institute of Standards and Technology*, 117, 176–184. <https://doi.org/10.6028/jres.117.010>
- pip documentation V23.3.1. (2023). <https://pip.pypa.io/>
- PyMoDAQ. (2023). <https://pymodaq.cnrs.fr/>
- PyVISA. (2023). <https://pyvisa.readthedocs.io/>
- Scheidgen, M., Brückner, S., Brockhauser, S., Ghiringhelli, L. M., Dietrich, F., Mansour, A. E., Márquez, J. A., Albrecht, M., Weber, H. B., Botti, S., Aeschlimann, M., & Draxl, C. (2023). FAIR Research Data With NOMAD: FAIRmat's Distributed, Schema-based Research-data Infrastructure to Harmonize RDM in Materials Science. *Proceedings of the Conference on Research Data Infrastructure*, 1. <https://doi.org/10.52825/cordi.v1i.376>
- Scheidgen, M., Himanen, L., Ladines, A. N., Sikter, D., Nakhaee, M., Fekete, Á., Chang, T., Golparvar, A., Márquez, J. A., Brockhauser, S., Brückner, S., Ghiringhelli, L. M., Dietrich, F., Lehmeier, D., Denell, T., Albino, A., Näsström, H., Shabih, S., Dobener, F., ... Draxl, C. (2023). NOMAD: A distributed web-based platform for managing materials science research data. *Journal of Open Source Software*, 8(90), 5388. <https://doi.org/10.21105/joss.05388>
- SweepMe! - A multi-tool measurement software. (2023). <https://sweep-me.net/>

Wilkinson, M. D., Dumontier, M., Aalbersberg, Ij. J., Appleton, G., Axton, M., Baak, A., Blomberg, N., Boiten, J. W., da Silva Santos, L. B., Bourne, P. E., Bouwman, J., Brookes, A. J., Clark, T., Crosas, M., Dillo, I., Dumon, O., Edmunds, S., Evelo, C. T., Finkers, R., ... Mons, B. (2016). The FAIR Guiding Principles for scientific data management and stewardship. *Sci. Data*, 3(1), 1–9. <https://doi.org/10.1038/sdata.2016.18>