Chapter 2
The Role of Open-Source Hardware for Research and Teaching

Johannes Maierhofer, Oliver M. Zobel, and Daniel J. Rixen

Abstract Open-source hardware and software are playing an increasingly important role in science and education. Open-source software for data analysis is already well-established, with Python Packages such as NumPy, SciPy and newly established machine learning packages such as scikit-learn, TensorFlow, and PyTorch.

In the field of structural dynamics Python Packages such as pyFRF, pyFBS, pyUFF, SDynPy, and pyIDI get more and more attention. However, the earlier stages of measurement – data acquisition and signal conditioning – remain heavily dependent on proprietary systems.

This poses challenges in both research and teaching. For students, commercial measurement systems often act as black boxes. The complexity and feature overload of professional software can obscure the core principles. Here, open-source acquisition tools provide a powerful alternative: by focusing on essential steps and exposing every detail of the processing chain, they enable a hands-on understanding of the underlying physics and signal processing.

From a research perspective, open-source hardware and software unlock new possibilities. For example, integrating live quality metrics directly into the measurement workflow enables immediate feedback. For example, channel consistency checks or impact repeatability for Virtual Point Transformations can be integrated into custom workflows. This prevents errors from propagating and reduces the need for costly repetitions.

However, building custom acquisition tools on top of proprietary platforms is not only technically challenging but also cost-prohibitive, especially in academic settings. A more inclusive solution is to develop open-source acquisition systems based on vendor-independent hardware. OASIS – the Open Acquisition System for IEPE Sensors – is one such initiative. It aims to create a flexible, transparent, and extensible platform suitable for both advanced research applications and introductory teaching labs.

This contribution outlines the motivation behind OASIS, highlights the benefits of open-source hardware in education and research, and invites the community to collaborate, share experiences, and help shape its future.

Keywords Testing equipment · Experimental dynamics · Open-source · Software · Hardware

Johannes Maierhofer
Chair of Applied Mechanics, TUM School of Engineering and Design,
Technical University of Munich, Boltzmannstr. 15, 85748 Garching, Germany
e-mail: j.maierhofer@tum.de

Oliver M. Zobel
Chair of Applied Mechanics, TUM School of Engineering and Design,
Technical University of Munich, Boltzmannstr. 15, 85748 Garching, Germany
e-mail: oliver.zobel@tum.de

Daniel J. Rixen
Chair of Applied Mechanics, TUM School of Engineering and Design,
Technical University of Munich, Boltzmannstr. 15, 85748 Garching, Germany
e-mail: rixen@tum.de

© The Author(s), under exclusive license to River Publishers 2026
Kristin B. Zimmerman, PhD (eds.), *Special Topics & Experimental Techniques in Structural Dynamics, 2026, Vol. 5*, Conference Proceedings of the Society for Experimental Mechanics Series, https://doi.org/10.13052/rp-9788743814160

The Success of Open-Source Software

Historical Review

The history of open-source software begins with the early days of computing, when source code was freely distributed with personal computers [1]. Unix, initially released for free by AT&T, became increasingly restricted after version 7 in 1979, as the company recognized its commercial value [2]. A major turning point followed in 1983, when licensing changes led researchers and universities to abandon Unix in favor of the emerging BSD (Berkeley Software Distribution) [2]. In the same year, a court decision established that software could be copyrighted [3], prompting BSD to remove all AT&T code [2].

Richard Stallman, concerned about the ethical implications of closed software, began promoting the term *Free Software* at MIT's AI Lab [1]. In 1983, he launched the GNU project, aiming to develop a fully open operating system. The motivation was partly practical—for example, a printer that repeatedly jammed but could not be fixed due to proprietary restrictions [4]. Importantly, *free* referred to freedom rather than zero cost [2]. The release of the GNU Compiler Collection (GCC) in 1986 provided a robust foundation for free software, and it remains widely used today.

From a Personal Project to Industrial Backbone

A look into the history of open-source projects must include the Linux project. What began in 1991 as Linus Torvalds' personal hobby—a small Unix-like kernel written for educational purposes—grew into one of the most influential open-source successes. Released under a free license, the Linux kernel developed into the operating system powering servers, supercomputers, and mobile devices worldwide. Its success also enabled new business models: companies such as Red Hat built sustainable enterprises not by selling licenses, but by providing long-term support and services [5].

On the one hand, this leads to the conclusion: open software systems can combine stability for industry with freedom for users [6]. On the other hand, the question arises: Can this be true also for open hardware systems? In this publication, we want to focus our view towards scientific measurement hardware. This is closely related to the way research (and teaching) is programmed at the time. Currently, Python is common practice in many fields.

Python in Science

During the last 10 years, Python became the standard language for data analysis. NumPy established fast array operations [7], and SciPy made advanced algorithms widely available [8]. These tools are community-driven yet form the core of daily scientific work. In vibration analysis, `pyFBS` is an example of how even advanced methods can be shared openly and reused across groups [9]. Openness here means that methods are easier to verify and teaching is closer to real research practice.

First Steps in Hardware

Compared to software, most measurement hardware is still closed and expensive. Open designs are now emerging that make instruments more accessible and adaptable [10]. They allow students to see what happens inside a system and give researchers more control over their setups. At the same time, hardware brings new responsibilities, such as ensuring safe use and compliance with regulation. The OASIS project follows this path, using Creative Commons licensing for hardware and MIT for software, and shows how open hardware can enter both teaching labs and research groups [11].

Open-Source Hardware Defined

Open-source hardware (OSH) extends the ideas of free and open software into the physical domain. The Open Source Hardware Association (OSHWA)[1] describes it as hardware whose design files are publicly available, so that anyone can study, modify, distribute, make, and sell the design or derived versions [10]. This requires that the essential artifacts of a project – schematics, layout files, bills of materials (BoM), and firmware – are openly documented.

Licensing is central to this concept. For software, permissive licenses such as MIT or Apache are widely used [7, 8]. For hardware, CERN has developed the Open Hardware License (OHL)[2] in variants that mirror software traditions: a permissive

[1] https://oshwa.org/
[2] https://cern-ohl.web.cern.ch/home

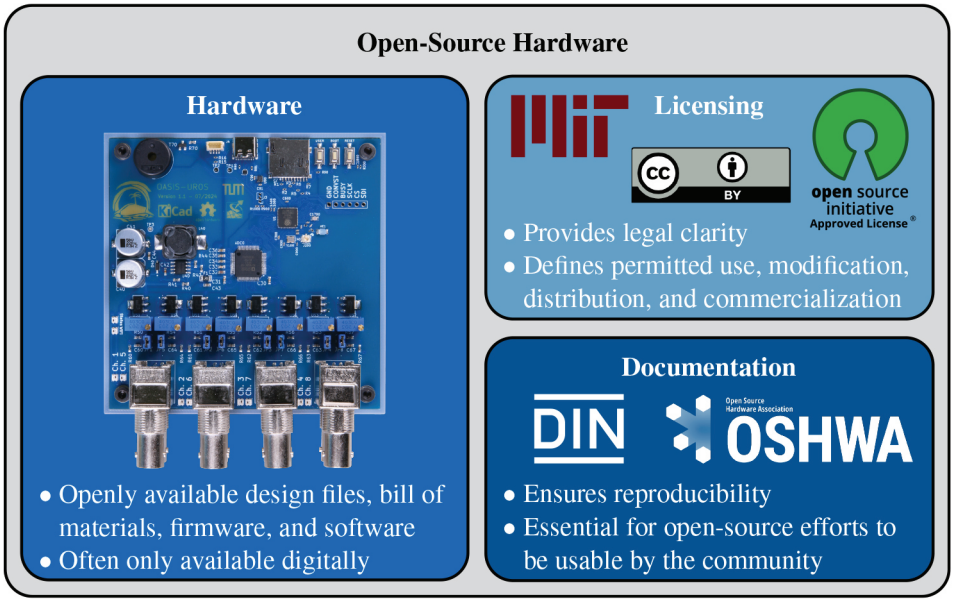


Fig. 1 Overview of open-source hardware and the required components: the hardware itself, licensing, and documentation

version (OHL-P), a weakly reciprocal version (OHL-W), and a strongly reciprocal version (OHL-S). These provide legal clarity for sharing and reusing hardware designs. In practice, projects often combine a hardware license with a separate software license. For example, the OASIS project[3] uses Creative Commons Attribution 4.0 International for hardware and the MIT license for its software components.

Standards for documentation quality are another pillar. DIN SPEC 3105 [12, 13] defines how open hardware should be documented so that others can reproduce and verify it. Certification programs such as the OSHWA certification build on these ideas and give users a simple signal that a project meets basic openness criteria. Together, licenses and standards establish the framework that allows communities to build sustainable and legally robust hardware ecosystems.

Sustainable Data and File Formats

For sustainable open-source hardware projects, the choice of data formats is as important as licensing and governance.

A first example comes from source code itself: storing code as plain UTF-8 text has become an obvious standard. Practically every editor can open the files without affecting their meaning. This universality made open-source software collaboration possible at a global scale.

For hardware design files the situation is more complex. CAD and ECAD models are central artifacts, yet commercial tools use closed formats that are rarely interoperable. Projects such as KiCAD[4] and FreeCAD[5] provide open formats, but these are not directly compatible with proprietary systems. This tension makes the decision for open formats a critical sustainability factor: once adopted, a community must ensure long-term support and backward compatibility. A format must be widely supported while not limiting the abilities of the underlying tool.

A second example concerns experimental data. Here, reproducibility requires reliable and long-term storage. HDF5[6] has established itself as a format that combines efficiency with wide software support. It allows large measurement datasets to be stored together with metadata and can be accessed from many programming environments. This makes it a robust choice for open scientific hardware projects where data must remain usable for years.

[3] https://gitlab.com/oasis-acquisition
[4] https://www.kicad.org/
[5] https://www.freecad.org/
[6] https://www.hdfgroup.org/solutions/hdf5/

Opportunities and Challenges

Open-source hardware invites new ways of building and teaching with instruments. It reduces entry barriers and makes methods visible. At the same time, it shifts effort toward documentation, validation, and governance. This section summarizes gains and pitfalls, then treats certification in depth.

Opportunities

- **Transparency.** Designs, firmware, and data paths are inspectable and auditable, which helps teaching and method verification [10, 14].
- **Adaptation.** Features can be added at the measurement stage (e.g., online quality checks) without waiting for vendor roadmaps [10].
- **Access.** Costs fall for entry setups; as an example scenario, an 8-channel EU build can be in the range of 500–1000 EUR total, whereas proprietary systems are often priced in the range of up to 1000 EUR per channel and include recurring contracts for software (illustrative only).

Challenges

- **Upkeep.** Communities must maintain designs, toolchains, and releases; this is a real cost [6].
- **Validation.** Without vendor warranties, labs must prove fitness for use.
- **Governance.** Clear versioning, issue handling, and contribution rules are needed for trust [10].

Certification and Compliance

Open status does not change regulatory obligations. Compliance attaches to the product and its intended use, not to the license.

CE (EU). For most laboratory electronics, the manufacturer issues a Declaration of Conformity after demonstrating compliance with the relevant directives and harmonised standards. Typical elements include: a technical file (schematics, layout, BoM, risk assessment), electromagnetic compatibility (EMC) and safety testing where applicable, user instructions, and product labeling. A Notified Body is only required in defined cases. Good practice for open projects: treat the repository as the living technical file; tag releases that correspond to tested hardware; archive test reports with the release.

UL/NRTL (US). In US workplaces, OSHA expects electrical equipment to be evaluated by a Nationally Recognized Testing Laboratory to the applicable standard. Listing is separate from CE and from openness. If open hardware is to be deployed in such environments, plan the enclosure, mains interface, wiring, and markings so that a later listing is feasible. Using listed external power supplies and observing creepage/clearance rules reduces rework.

Software and updates. Law increasingly treats firmware and updates as part of the product lifecycle. Open projects should define an update policy, keep change logs, and document security fixes. Release artefacts (bitstreams, binaries) should be reproducible from tagged sources.

Documentation quality. Standards and community norms help here. DIN SPEC 3105 [12, 13] stresses complete documentation for reproducibility; educational deployments benefit from that discipline [14, 10]. License choices should be explicit: hardware under CC BY 4.0 or CERN–OHL variants; software under permissive terms where appropriate [7, 8].

Quality Assurance and Data Trust

Calibration against traceable references is essential for credibility. Record calibration factors, uncertainties, and environmental conditions with the data. Maintain BoM/SBoM, firmware hashes, and versioned schematics so that past measurements remain interpretable. Prefer durable data formats (e.g., HDF5) so that large datasets and metadata travel together.

Teaching and Institutional Policies

Universities differ in lab rules; however, recurring elements exist: risk assessment, supervision, equipment checks, and written SOPs. Open hardware supports learning because students can see every step; it remains the instructor's duty to bound voltages, enclose mains circuits, and document safe operation [14].

Illustrative Example: Vibration Analysis

This section demonstrates how **pyFBS** [9] (software) and **OASIS-UROS** [11] (hardware and firmware) can serve as concrete examples that highlight both the potential and the limitations of open-source tools in vibration analysis.

pyFBS

The pyFBS project (University of Ljubljana / TUM) is a Python package for Frequency-Based Substructuring (FBS), Transfer Path Analysis, and modal identification methods, including multi-reference and modal tests [9]. It implements advanced methods such as the Virtual Point Transformation (VPT), System Equivalent Model Mixing (SEMM), Frequency Response Function (FRF) synthesis, and provides a 3D display of sensors, excitations, and modal shapes [9]. The package includes both numerical and experimental datasets ranging from simple beam-like structures to more complex assemblies, which allow users to practice realistic workflows. Licensed under MIT, pyFBS is free to use, extend, and integrate into teaching and research with minimal hurdles.

The main strength of pyFBS lies in its ability to provide researchers and students with access to advanced substructuring methods without requiring expensive software. Its open nature ensures reproducibility, as datasets and workflows are publicly available and the methods and transformations can be inspected in detail. Although the software is accessible to everyone, generating meaningful results still requires considerable expert knowledge and practical experience. In commercial solutions, the high cost can act as a natural filter, ensuring that mainly established experts use the tools with the required attention. In contrast, open-source software invites broader use, which makes the role of expertise even more central. Experimental accuracy, for example, continues to depend on high-quality data, especially in high-frequency or low-noise scenarios. Methods such as expanded VPT remain sensitive to factors like sensor placement and alignment, and overall performance is directly linked to data quality.

OASIS-UROS

OASIS-UROS is an open acquisition system for IEPE sensors (Integrated Electronics Piezoelectric), designed primarily for academic research and teaching. Building on earlier OASIS versions [15, 16], it incorporates improved design choices. Its architecture is based on an ESP32 microcontroller, combined with an 8-channel AD7606C-18 ADC, SD-card caching, adjustable voltage ranges, and an upgraded IEPE front-end. The system is capable of simultaneously acquiring eight channels with sampling rates of up to approximately 36 kHz across all channels, supporting oversampling and benefiting from the improved front-end design. Validation against commercial hardware in modal analysis has shown that in certain test cases, OASIS-UROS achieves performance close to commercial systems, though not fully identical. All hardware, firmware, and software are openly published and reproducible, with the project listed in OSHWA's certified register[7] under CC-BY-4.0 for hardware and MIT for software.

The strengths of OASIS-UROS lie in its cost-effectiveness and its pedagogical value. It enables students to study and experiment with all layers of the acquisition chain, from the analog front-end to the firmware and sampling chain, making it highly suitable for teaching and exploratory research. Its performance is sufficient for many vibration tasks, such as modal analysis and FRF estimation, particularly when extreme precision or very high bandwidths are not required. At the same time, limitations remain. Compared to high-end commercial systems, OASIS-UROS cannot match the lowest possible noise floors, the highest sampling rates, or the guarantees of certified safety and EMC compliance. Assembly, calibration, and user expertise can influence the results, and practical factors such as enclosures, input protection, power supplies, and the measurement environment remain critical for achieving a reliable data acquisition system.

[7] https://certification.oshwa.org/de000150.html

Lessons from pyFBS and OASIS-UROS

The comparison between pyFBS and OASIS-UROS illustrates broader lessons for open-source vibration analysis. Software alone is not sufficient: high-quality data acquisition hardware is essential for reliable results. The analog front end must be carefully validated for noise, gain, sampling rate, and synchronization, as otherwise software methods can yield misleading conclusions. Open licensing – MIT for software, CC-BY for hardware – and transparent publication of materials are key enablers for inspection, derivation, reuse, and building trust. At the same time, performance trade-offs are unavoidable. As requirements for channel count, sampling rates, or formal certification increase, so do cost and complexity. Consequently, open-source hardware is particularly compelling for teaching, prototyping, and non-mission-critical research, or as a supplement to commercial equipment, but does not fully replace high-end certified systems in all contexts.

Conclusion

Openness in hardware and software is not simply better or worse than proprietary systems. It shifts the balance of effort: away from procurement and vendor lock-in, and toward quality assurance, documentation, and certification. This redistribution can be highly valuable in teaching, where transparency deepens understanding, and in exploratory research, where flexibility and adaptability outweigh the need for certified performance.

The examples of pyFBS and OASIS-UROS show that advanced methods and practical hardware can both be shared under open licenses without losing relevance to the community. For long-term success, however, projects must take the next steps: align documentation with DIN SPEC 3105, define clear certification pathways (CE, UL/NRTL), and encourage reproducible community practice. By doing so, open-source hardware can establish itself as a trustworthy complement to commercial systems in research and education.

References

1. Maracke, C. “Free and Open Source Software and FRAND-based patent licenses”. In: *The Journal of World Intellectual Property* 22.3-4 (2019), pp. 78–102. ISSN: 1747-1796. DOI: 10.1111/jwip.12114.
2. Weber, S. *The Success of Open Source*. Harvard University Press, 2005. ISBN: 9780674018587.
3. Nussbaum, J. L. “Apple Computer, Inc. v. Franklin Computer Corporation Puts the Byte Back into Copyright Protection for Computer Programs”. In: *Golden Gate University Law Review* 14.2 (1984). URL: https://digitalcommons.law.ggu.edu/cgi/viewcontent.cgi?article=1344&context=ggulrev.
4. “Transcript of Richard M. Stallman’s speech “Free Software: Freedom and Cooperation””. In: (2001). URL: https://www.gnu.org/philosophy/rms-nyu-2001-transcript.txt.
5. Young, R. “How Red Hat Software Stumbled Across a New Economic Model”. In: *First Monday* (1999). DOI: 10.3998/3336451.0004.304.
6. Duparc, E. “Archetypes of open-source business models”. In: *Electronic Markets* 32 (2022), pp. 609–623. DOI: 10.1007/s12525-022-00557-9.
7. Harris, C. R., Millman, K. J., Walt, S. J. van der, Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., Kerkwijk, M. H. van, Brett, M., Wilson, A., Wilhelm, K. F., Hickey, H., Weckesser, S. J., Wilson, N. M., Millman, E., and al., et. “Array programming with NumPy”. In: *Nature* 585 (2020), pp. 357–362. DOI: 10.1038/s41586-020-2649-2.
8. Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., Walt, S. J. van der, Brett, M., Wilson, J., Millman, K. J., Mayorov, N., Nelson, A. R. J., Jones, E., Kern, R., Moore, E. W., VanderPlas, J., Laxalde, D., Perktold, J., Cimrman, R., Henriksen, I., Quintero, E. A., Harris, C. R., Archibald, A. M., Ribeiro, A. H., Pedregosa, F., Mulbregt, P. van, and SciPy 1.0 Contributors, the. “SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python”. In: *Nature Methods* 17.3 (2020), pp. 261–272. DOI: 10.1038/s41592-019-0686-2.
9. Bregar, T., Mahmoudi, A. E., Kodrič, M., Ocepek, D., Trainotti, F., Pogačar, M., Göldeli, M., Čepon, G., Boltežr, M., and Rixen, D. J. “pyFBS: A Python package for Frequency Based Substructuring”. In: *Journal of Open Source Software* 7.69 (2022), p. 3399. DOI: 10.21105/joss.03399.
10. Oellermann, M., Jolles, J. W., Ortiz, D., Seabra, R., Wenzel, T., Wilson, H., and Tanner, R. L. “Open Hardware in Science: The Benefits of Open Electronics”. In: *Integrative and Comparative Biology* 62.4 (May 2022), pp. 1061-1075. 1ssn: 15577023. DOI: 10.1093/ich/icac043.
11. Zobel, O. M., Maierhofer, J., Köstler, A., and Rixen, D. J. “OASIS-UROS: Open acquisition system for IEPE sensors upgraded, refined, and overhauled software”. In: *HardwareX* 23 (2025), e00650. issn: 2468-0672. DOI: https://doi.org/10.1016/j.ohx.2025.e00650.
12. *DIN SPEC 3105-1:2020-07, Open Source Hardware – Part 1: Requirements for technical documentation*. DOI: 10.31030/3173063.
13. *DIN SPEC 3105-2:2020-07, Open Source Hardware – Part 2: Community-based assessment*. DOI: 10.31030/3173062.
14. Heradio, R., Chacon, J., Vargas, H., Galan, D., Saenz, J., De La Torre, L., and Dormido, S. “Open-Source Hardware in Education: A Systematic Mapping Study”. In: *IEEE Access* 6 (2018), pp. 72094-72103. ISSN: 2169-3536. DOI: 10.1109/access.2018.2881929.
15. Zobel, O. M., Maierhofer, J., Köstler, A., and Rixen, D. J. “OASIS: Bridging the Open-Source Gap Between Testing and Data Processing”. In: June 2025. ISBN: 978-961-7187-17-5.
16. Zobel, O. M. and Maierhofer Johannes and Rixen, D. J. “OASIS: Open Acquisition System for IEPE Sensors: For Academic Research and Teaching Purposes”. In: *Sensors & Instrumentation and Aircraft/Aerospace Testing Techniques, Volume 8*. Ed. by Walber, C., Stefanski, M., and Seidlitz, S. Springer Nature Switzerland, 2024, pp. 79-86. ISBN: 978-3-031-34938-6. DOI: 10.1007/978-3-031-34938-6_9.