ISBN: 978-3-95450-231-8 ISSN: 2673-5490 doi: 10.18429/JACoW-IPAC2023-THPL098

OPENDIGITIZER: DIGITIZER MODERNISATION USING OPENCMW AND GNU RADIO 4.0 FOR FAIR

Ralph J. Steinhagen, Alexander Krimm, David Ondreka
GSI Helmholtzzentrum, Darmstadt, Germany
Frank Osterfeld, Ivan Čukić, Björn Balzs, Giulio Camuffo, KDAB, Berlin, Germany

Abstract

OpenDigitizer is an open-source project modernizing FAIR's digitizer infrastructure and GUI using OpenCMW, WebAssembly, and GNU Radio 4.0 frameworks. It provides monitoring, diagnostics, and supports development of advanced measurement and control loops. Utilizing directed signal flow graphs for efficient post-processing and feedback control, the core is accessible to domain experts with minimal programming experience. The WebAssembly-compatible UI enables native deployment on mobile and browser platforms, facilitating flexible use during commissioning, troubleshooting, and control room operations.

INTRODUCTION

OpenDigitizer is an open-source project aimed at modernizing the digitizer infrastructure and user interfaces at the Facility for Antiproton and Ion Research (FAIR) [1, 2]. The project embraces modern C++20 development standards and leverages OpenCMW, GNU Radio 4.0, ImGUI, and WebAssembly ecosystems(Fig. 1) [3–7]. The primary applications of OpenDigitizer include:

- First-line diagnostics and fault identification, serving as a distributed diagnostic tool for accelerator equipment with nanosecond-level synchronization, offering functionalities similar to oscilloscopes, software-defined radios, spectrum analyzers, VNAs, and other hardware monitoring equipment.
- Providing building blocks for higher-level diagnostics and monitoring tools, assisting equipment experts, operators, and FAIR users in developing basic to advanced top-level diagnostics and feedback control loops.
- Supporting a rapid prototyping R&D environment for quick adaptation, testing, and integration of solutions developed on lab test stands or during machine studies into the 24/7 operation of the facility.

The existing FESA-based system has been deployed for over 200 data acquisition systems [8, 9]. Around 350 systems directly connected to hardware and an additional 300 post-processing services are expected for FAIR. All these systems will share the same OpenCMW, GNU Radio, OpenDigitizer, and UI/UX software stack.

The motivation for the OpenDigitizer reimplementation stems from the issues encountered with the FESA-based prototype, which include [10]:

- Heavy reliance on proprietary, largely undocumented code developed for another accelerator facility, not entirely aligned with FAIR's requirements.
- The need for flexible handling of signal post-processing on both front-end systems directly connected to digitizers and middle-tier services providing high-level diagnostics monitoring and feedback control loops based on the same data.
- Difficulties with evolving the system towards the critical needs for commissioning and operation of FAIR accelerators due to departure of many original developers and the accompanying loss of expertise.

This paper provides an overview of OpenDigitizer's objectives, main components, and applications.

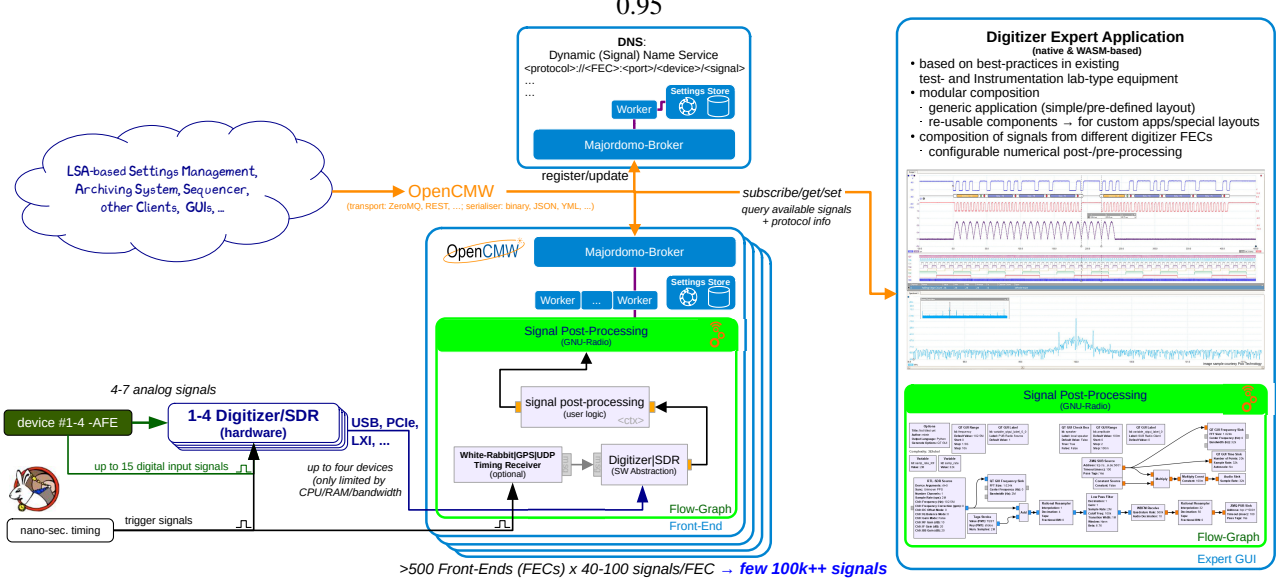


Figure 1: Architecture of the opencmw digitizer implementation.

ISBN: 978-3-95450-231-8 ISSN: 2673-5490 doi: 10.18429/JACoW-IPAC2023-THPL098

OPENDIGITIZER GOALS

OpenDigitizer focuses on the following key objectives:

1. Streamline the codebase by eliminating unnecessary complexities and features of the legacy systems, while retaining their tried-and-tested functionalities, making it more adaptable, flexible, and maintainable.
2. Use and contribute to existing standards such as GNU Radio and OpenCMW to avoid redundancy and promote interoperability [11]. Key improvements include:
   (a) Improved timing integration within GNU Radio, supporting ms- to ns-level industrial and in-house timing system standards (e.g., software-based, GPS, White Rabbit [12]).
   (b) Advanced GNU Radio integration for both continuous and indexed chunked data (e.g., event-driven data for processes like injection, ramp, slow or fast-extraction, and transient recording).
   (c) Bridging the gap between RSE and accelerator domain expertise through the use of graphs for general signal processing, diagnostics, semi-automation, and feedback loops.
   (d) Enabling transfer of processing algorithms from ad-hoc experiments to routine operation by utilising the same framework in services and UI.
3. Improve accessibility for new contributors by reducing cognitive complexity and flattening the learning curve, allowing academics, industrial partners, and students to participate and contribute more effectively.
4. Use a distributed micro-service architecture reducing dependencies on centralised services to provide redundancy and avoid restriction to preconfigured environments.
5. Promote collaboration and knowledge exchange across industries, academic institutions, and government organizations, enabling the sharing of ideas and expertise by contributing to a shared ecosystem.

By focusing on these objectives, OpenDigitizer aims to facilitate innovation, adaptation, and collaboration to address FAIR's complex requirements effectively. By committing to modern C++, RSE standards, and the FAIR principles (Findability, Accessability, …), OpenDigitizer encourages collaboration within a shared cross-domain ecosystem, significantly increasing the pool of developers contributing. This approach ensures a safe, well tested, and maintainable codebase, reduced total cost of ownership, and minimized continuous maintenance effort.

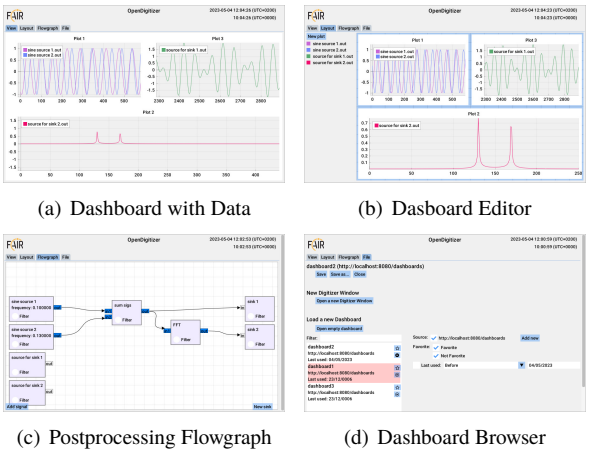
KEY COMPONENTS OF OPENDIGITIZER

The OpenDigitizer system builds on several essential components that contribute to its flexibility and ease of use as shown in Fig. 1. The two key components are: OpenCMW, an open-source middleware solution developed at GSI and FAIR providing flexible data transport, efficient data serialisation, and intuitive domain objects based on compile-time reflection [3], and the GNU Radio framework (version 4.0), a powerful software toolkit designed for signal processing and software-defined radio [7, 11, 13, 14]. GNU Radio uses directed signal flow graphs for efficient expression of post-processing and feedback control loop logic. This feature makes it easy for domain experts with minimal programming experience to inspect and reconfigure existing systems.

ImGUI and WebAssembly (WASM, through Emscripten) are used for the user interfaces, enabling cross-platform compatibility, and native deployment on mobile as well as other browser-based platforms supporting flexible use during commissioning and troubleshooting [15, 16].

User Interface

The importance of a simple, yet powerful user interface became evident from the experience with the existing system [1]. A comprehensive user interface facilitates quick demonstrations, error analysis, and prototyping. It helps shortening development cycles and avoides expensive parallel developments by keeping users and developers synchronised. UX and UI are thus considered core aspects during the OpenDigitizer implementation. An overview of the prototype elements can be seen in Fig. 2.



(a) Dashboard with Data (b) Dasboard Editor

(c) Postprocessing Flowgraph (d) Dashboard Browser

Figure 2: Screenshots of the UI of the system.

The UI is implemented in C++ to ensure algorithms can be moved freely between the different system layers. This provides some flexibility regarding refactoring and addressing implementations and potentially bottlenecks at the most suitable location of the overall vertical stack, as well as also addresses Java-related performance issues of the existing system. The service incorporates n HTTP server that, in addition to a high-performance ZeroMQ-based transport interface, exposes the data via a generic REST interface as well as the UI client. The client compiles into a WebAssembly binary of less than 2 MB including all dependencies [17].

The primary UI consists of an dashboard displaying a set of charts and a processing flow graph. Both views can be configured, modified, stored and retrieved locally or on the service at the user preferences. This provides a greater control over data processing without the immediate need for programming expertise and enhances the overall user

ISBN: 978-3-95450-231-8 ISSN: 2673-5490 doi: 10.18429/JACoW-IPAC2023-THPL098

experience. This store/retrieve feature facilitates collaboration among all users and helps to maintain an organised commissioning and operation of FAIR.

OPEN SOURCE & AGILE APPROACH

OpenDigitizer reduces dependency on specialized hardware systems and proprietary or in-house software, encouraging contributions from external partners and streamlining the onboarding process. Emphasizing open-source principles, continuous improvement, usability, and maintainability, OpenDigitizer fosters collaboration and knowledge sharing. Through the adoption of an open-source strategy and an agile approach, OpenDigitizer:

- Facilitates collaboration and the exchange of ideas
- Promotes better abstractions and clean code principles
- Enhances collective ownership and mitigates the risks associated with critical knowledge being held by a small group, which is commonly known as small insufficient 'bus factor'
- Aligns with the Public Money, Public Code campaign to leverage taxpayer-funded research for the benefit of the broader public, promote high-quality software development, and avoid costly vendor lock-in, feature creep, and technical debt [18].

OpenDigitizer's commitment to open-source principles extends to embracing modern software engineering practices, ensuring sustainability and maintainability, and fostering public-private partnerships with industry partners like, for example, KDAB.

Adhering to FAIR principles, OpenDigitizer minimizes the total cost of ownership and supports continuous improvement during hardware and beam commissioning. By utilizing free and open-source solutions, OpenDigitizer contributes to other domains and benefits the general public.

Through its open-source strategy and agile approach, as well as its alignment with the Public Money, Public Code campaign, OpenDigitizer reduces costs, fosters collaboration, promotes innovation, and contributes to the broader scientific community. We hope that the commitment to high-quality software development and sustainability sets a positive example for others to follow.

FUTURE DEVELOPMENTS

As OpenDigitizer continues to develop and adapt, its future enhancements will address the evolving requirements of the FAIR facility as well as other core users and further expand its capabilities. Currently, OpenDigitizer achieved a Minimum Viable Product (MVP) state with a vertical stack implementation that covers most critical core features but not necessarily all in their full horizontal breadth. Key areas of focus for the next development steps include:

- Migrating the existing GNU Radio post-processing blocks from version 3.X to version 4.0. This transition will occur within the context of the GNU Radio ecosystem and its large user community.
- Refining the user experience (UX) and enhancing common features: OpenDigitizer will continue iterating on its UX design and functionality, ensuring that users have access to intuitive, powerful tools that cater to their needs.
- Strengthening integration with other systems: OpenDigitizer aims to boost its interoperability with other instrumentation and control platforms to facilitate seamless data exchange and collaboration among different FAIR facility components.
- Enhancing scalability, performance, and efficiency: OpenDigitizer will work on improving its ability to manage increased data volumes and support more devices while optimizing its software stack, data processing algorithms, and hardware compatibility.
- Investigating mobile and augmented reality (AR) solutions: OpenDigitizer may explore mobile platforms and hands-free AR solutions during the FAIR facility commissioning to improve efficiency and offer innovative ways to visualize and interact with data.
- Upholding open-source principles: OpenDigitizer will remain dedicated to open-source principles, fostering collaboration, and promoting high-quality, reusable, and maintainable code development while seeking partnerships with other organizations and research institutions.

By focusing on these aspects and consistently refining its capabilities, OpenDigitizer aspires to stay at the forefront of data acquisition, processing, and visualization within the FAIR facility and beyond. Its emphasis on open-source principles and collaboration ensures that OpenDigitizer will keep adapting and growing in the ever-changing landscape of accelerator science and technology.

CONCLUSION

The OpenDigitizer project introduces a modern, open-source approach to distributed synchronised digitizer infrastructure. While initially designed for the FAIR facility, the expressed intent is to be highly adaptable also for use by other similar research facilities, industry, academia, as well as private users. By leveraging the power of OpenCMW, WebAssembly, and GNU Radio 4.0 frameworks, OpenDigitizer simplifies the codebase, enhances integration with existing standards, lowers cognitive complexity for new contributors, and fosters collaboration and knowledge sharing.

Interested readers are invited to explore the OpenDigitizer project website, try out the system, and consider adopting and contributing to its development. Your participation can help shape the future of digitizer infrastructure, not just at FAIR, but across various domains. By collaborating with fellow users and contributors, you can benefit from shared knowledge and expertise, fostering innovation and advancing digitizer modernization in diverse industries and disciplines.

ISBN: 978-3-95450-231-8 ISSN: 2673-5490 doi: 10.18429/JACoW-IPAC2023-THPL098

REFERENCES

[1] R. J. Steinhagen *et al.*, "Generic Digitization of Analog Signals at FAIR – First Prototype Results at GSI," in *Proc. IPAC'19*, Melbourne, Australia, May 2019, pp. 2514–2517. doi:10.18429/JACoW-IPAC2019-WEPGW021

[2] A. Krimm and R. Steinhagen, "FAIR Common Specification - On the Digitization of Analog Signals in the FAIR Accelerator Complex," FAIR, Tech. Rep., 2020. https://edms.cern.ch/document/1823376

[3] R. Steinhagen *et al.*, "OpenCMW - A Modular Open Common Middle-Ware Library for Equipment- and Beam-Based Control Systems at FAIR," in *Proc. ICALEPCS'21*, Shanghai, China, 2022, paper TUPV009, pp. 392–399. doi:10.18429/JACoW-ICALEPCS2021-TUPV009

[4] A. Krimm and R. Steinhagen, "FAIR Common Specification - Modular Open Common Middle-Ware Library for Equipment- and Beam-Based Control Systems of the FAIR Accelerators," FAIR, Tech. Rep., 2020. https://edms.cern.ch/document/2444348

[5] FAIR, *Opencmw c++ repository*, https://github.com/fair-acc/opencmw-cpp, Accessed: May 3, 2023, 2023.

[6] FAIR, *Opendigitizer java repository*, https://github.com/fair-acc/opencmw-java, Accessed: May 3, 2023, 2023.

[7] GNU Radio Development Team, *Gnu radio*, https://www.gnuradio.org, Accessed: May 3, 2023, 2001.

[8] L. Fernandez *et al.*, "Front-End Software Architecture," in *Proc. ICALEPCS'07*, Oak Ridge, TN, USA, Oct. 2007, pp. 310–312. https://jacow.org/ica07/papers/WOPA04.pdf

[9] T. Hoffmann, "FESA - The Front-End Software Architecture at FAIR," in *Proc. PCaPAC'08*, Ljbuljana, Slovenia, Oct. 2008, pp. 183–185. https://jacow.org/pc08/papers/WEP007.pdf

[10] FAIR, *Opendigitizer*, https://github.com/fair-acc/opendigitizer, Accessed: May 3, 2023, 2023.

[11] R. J. Steinhagen *et al.*, "GNU Radio 4.0 for Real-Time Signal-Processing and Feedback Applications at FAIR," in *Proc. IPAC'23*, Venice, Italy, pp. 2514–2517. http://jacow.org/ipac2023/papers/thpl099.pdf

[12] P. Moreira, J. Serrano, T. Wlostowski, P. Loschmidt, and G. Gaderer, "White rabbit: Sub-nanosecond timing distribution over ethernet," in *2009 International Symposium on Precision Clock Synchronization for Measurement, Control and Communication*, 2009, pp. 1–5. doi:10.1109/ISPCS.2009.5340196

[13] GNU Radio Development Team, *GNU Radio GitHub repo*, https://github.com/gnuradio/gnuradio, 2001.

[14] FAIR & GNU Radio Development Team, *GNU Radio 4.0 – graph-prototype*, https://github.com/fair-acc/graph-prototype, 2022.

[15] W3C WebAssembly Community Group, *Webassembly specifications*, https://webassembly.org/specs, Accessed: May 3, 2023, 2023.

[16] O. Cornut, *Dear imgui*, 2015-2021. https://github.com/ocornut/imgui

[17] FAIR, *Opendigitizer online demo*, https://fair-acc.github.io/opendigitizer, Accessed: May 3, 2023, 2023.

[18] Free Software Foundation Europe, *Public money? public code!* https://publiccode.eu/en/, Accessed: May 3, 2023, 2017.