




VILLASnode: An Open-Source Real-time Multi-protocol Gateway

Steffen Vogel ^{1,3}, Niklas Eiling ¹, Manuel Pitz ¹, Alexandra Bach ¹, Marija Stevic^{1,3}, and Prof. Antonello Monti ^{1,2}

¹ Institute for Automation of Complex Power Systems, RWTH Aachen University, Germany ² Fraunhofer Institute for Applied Information Technology, Aachen, Germany ³ OPAL-RT Germany GmbH

DOI: [10.21105/joss.08401](https://doi.org/10.21105/joss.08401)

Software

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

Editor: [Daniel S. Katz](#)  

Reviewers:

- [@michmx](#)
- [@Puneetha-Ramachandra](#)

Submitted: 28 May 2025

Published: 29 August 2025

License

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

Summary

VILLASnode is a multi-protocol gateway, designed to facilitate real-time data exchange between various components of geographically-distributed real-time experiments. Components can be testbeds, digital real-time simulators, software tools, and physical devices. It is designed for the co-simulation of power system and related energy applications. VILLASnode was originally designed for the co-simulation of electrical networks, but since then was further extended to cover a larger variety of use cases and domains. VILLASnode serves as the gateway that connects components across different infrastructures by providing a set of protocols and customized third-party implementations, e.g., different simulators. It enables seamless collaboration in research and testing environments while safeguarding the intellectual property of the infrastructures. The components of every infrastructure appear as a black box. The infrastructure does not need to share models or confidential information.

VILLASnode is a set of Linux command line tools and shared library. It can be installed from source or via a OCI container. It is written in C/C++ and designed in a modular and extensible way.

All components that are interfaced by the VILLASnode gateway are represented by nodes (n). These nodes act as sinks or sources for data specific to the component. Every node is an instance of a node-type. In a single VILLASnode instance, multiple instances of the same node-type can co-exist at the same time. The basic data package, common for all node-types, includes timestamped data, constituting a sample. Up to 64 values can form a sample. Samples may need modification or filtering. VILLASnode supports hooks (h) for this purpose. Hooks are simple callback functions, which are called whenever a message is processed. Paths (p) take care of the processing and define the connections and dataflows between nodes. Node-types, hooks, and paths need to be initialized in a JSON configuration file that is passed when starting VILLASnode. Figure 1 shows an example of an experiment where five different node-types are used, connected by three paths, using three hooks. It includes queues (q) and registers (r). Queues temporarily store data before data is forwarded to registers. Registers provide the possibility to (de-)multiplex data and to create new samples.

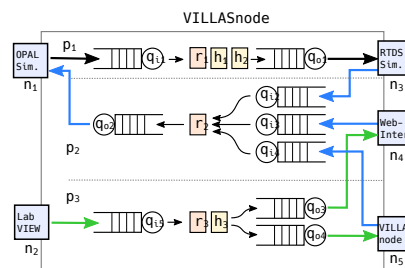


Figure 1: Example of modular experimental design with nodes, paths, and hooks (Vogel et al., 2025).

External software tools and programming languages can be interfaced with VILLASnode via several methods. Locally, they can be spawned as sub-processes and exchange data via shared-memory or standard I/O streams. Remotely, they can be interfaced via any of the supported node-types and protocols, such as MQTT or TCP sockets. Additionally, VILLASnode can be configured and controlled remotely by an HTTP REST-style Application Programming Interface (API). An OpenAPI specification of the API is provided in order to generate API bindings for a variety of different programming languages. VILLASnode is written in C++ and supports real-time execution on real-time Linux systems. To provide all necessary information, VILLASnode has detailed documentation (Vogel et al., 2025). It includes installation recommendations and best practices for development as well as example configurations and beginners guides, so-called labs.

VILLASnode can be compared with other available open-source co-simulation tools, such as Mosaik (Rohjans et al., 2013) or Helics (Hardy et al., 2024). Mosaik uses a SimAPI to communicate with other simulators based on fixed timesteps. Integrated simulators need to support the SimAPI, which requires extra implementations. Simulators cannot reuse existing supported protocols, which VILLASnode makes use of and takes care of protocol conversion. Helics is designed for large-scale testing and supports bindings for different languages. It uses a broker that manages the communication between the simulators, whereas VILLASnode has a peer-to-peer architecture.

Statement of need

VILLASnode supports the coupling of real-time simulators of different hardware and software vendors, specifications, and implementations (Monti et al., 2018). These simulators play a crucial role in both academic research and industrial applications, especially within simulation of electrical power networks. They are primarily utilized for hardware-in-the-loop (HiL) simulations. As an example, in the scope of the ENSURE project, the German electrical grid is emulated with the ability to couple dynamically simulators and physical components that can be interfaced safely via analog input and output signals (Mehlmann et al., 2023). This approach not only reduces development costs but also allows for the validation of components under scenarios in which it is difficult or unsafe to replicate real-world field scenarios. Moreover, VILLASnode supports the linking of simulators and real-time simulators from different vendors so that models do not need to be shared. It allows for communication between proprietary simulation models that cannot be shared, thus protecting intellectual property during collaboration. Local simulators and simulation infrastructure can be combined in a powerful arrangement unavailable in any individual infrastructure. This approach has the additional advantage of allowing co-simulation with heterogeneous methods, algorithms, and communication protocols. Communication with components, sensors, or actuators can also be implemented using VILLASnode.

This can be extended to geographically-distributed experiments. In this case, VILLASnode takes care of the communication between the different participants within an experiment that spans multiple infrastructures. Not only can simulators be included, but physical devices can

also be integrated ([Bach & Monti, 2025](#)). Although communication latencies constrain the dynamics of experiments for real-time applications, VILLASnode offers significant advantages such as leveraging existing infrastructure across sites and facilitating collaboration among interdisciplinary teams. Manufacturers, customers, and certifiers benefit from remote access and distributed testing while maintaining data confidentiality and intellectual property. Current work includes automation of integration of VILLASnode in practical field devices ([Pitz et al., 2024](#)).

Features

As the gateway, VILLASnode is the main component of the VILLASframework, which offers several key features.

The gateway supports a wide range of established data exchange protocols. Support for various message brokers like MQTT, AMQP, nanomsg, ZeroMQ, Redis, and Kafka provide connectivity to existing software platforms. Support for process bus protocols such as CAN, EtherCAT, Modbus, IEC 61850-8-1 (GOOSE), and IEC 61850-9-2 (Sampled Values) enable connectivity to physical devices. A range of web-based protocols like WebRTC, WebSockets, NGSI, and a HTTP REST API facilitate integration into web-based interfaces. And a set of generic protocols add support for plain UDP/TCP/Ethernet sockets, Infiniband, file-based I/O, and communication with other processes via standard I/O or shared memory regions. For custom interfaces that require hard real-time timing guarantees, the gateway supports communicating with Xilinx/AMD FPGA evaluation boards via PCIe and user space drivers implemented on top of VFIO.

Other components of the framework include VILLASweb, which provides visualization and monitoring of distributed experiments via a web browser, and VILLAScontroller, which integrates and orchestrates the execution of gateways and related hardware and software component in a joint experiment.

Acknowledgements

We like to thank several colleagues and students who supported us. Especially, we thank Leonardo Carreras, Laura Fuentes Grau, Andres Acosta, and Felix Wege. The project's support includes:

- **RESERVE**: European Unions Horizon 2020 research and innovation programme under grant agreement No. 727481.
- **VILLAS**: Funding provided by **JARA-ENERGY**. Jülich-Aachen Research Alliance (JARA) is an initiative of RWTH Aachen University and Forschungszentrum Jülich.
- **Urban Energy Lab 4.0**: Funding is provided by the **European Regional Development Fund (ERDF)**.
- **ERIGrid2.0**: European Unions Horizon 2020 research and innovation programme under grant agreement No. 870620.
- **ENSURE**: Federal Ministry of Education and Research supports the project under identification 03SFK1C0-3.
- **NFDI4Energy**: Funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – 501865131.
- **HYPERRIDE**: European Unions Horizon 2020 research and innovation programme under grant agreement No. 957788.
- **AI-EFFECT**: European Unions Horizon 2020 research and innovation programme under grant agreement No. 101172952.
- **EnerTEF**: European Unions Horizon 2020 research and innovation programme under grant agreement No. 101172887.

- [Target-X](#): European Unions Horizon 2020 research and innovation programme under grant agreement No. 101096614.

References

- Bach, A., & Monti, A. (2025). Remote real-time testing of physical components using communication setup automation. *IEEE Access*, 13, 39066–39075. <https://doi.org/10.1109/ACCESS.2025.3546311>
- Hardy, T. D., Palmintier, B., Top, P. L., Krishnamurthy, D., & Fuller, J. C. (2024). HELICS: A co-simulation framework for scalable multi-domain modeling and analysis. *IEEE Access*, 12, 24325–24347. <https://doi.org/10.1109/ACCESS.2024.3363615>
- Mehlmann, G., Kühnapfel, U., Wege, F., Winkens, A., Scheibe, C., & Vogel, S. (2023). The Kopernikus ENSURE co-demonstration platform. *IEEE Open Journal of Power Electronics*, 4, 987–1002. <https://doi.org/10.1109/OJPEL.2023.3332515>
- Monti, A., Stevic, M., Vogel, S., De Doncker, R. W., Bompard, E., Estebsari, A., Profumo, F., Hovsapien, R., Mohanpurkar, M., Flicker, J. D., Gevorgian, V., Suryanarayanan, S., Srivastava, A. K., & Benigni, A. (2018). A global real-time superlab: Enabling high penetration of power electronics in the electric grid. *IEEE Power Electronics Magazine*, 5(3), 35–44. <https://doi.org/10.1109/MPEL.2018.2850698>
- Pitz, M., Wege, F., Eiling, N., Vogel, S., Bareis, V., & Monti, A. (2024). Automated deployment of single-board computer based grid measurement and co-simulation equipment. *2024 Open Source Modelling and Simulation of Energy Systems (OSMSES)*, 1–6. <https://doi.org/10.1109/OSMSES62085.2024.10668996>
- Rohjans, S., Lehnhoff, S., Schütte, S., Scherfke, S., & Hussain, S. (2013). Mosaik - a modular platform for the evaluation of agent-based smart grid control. *IEEE PES ISGT Europe 2013*, 1–5. <https://doi.org/10.1109/ISGTEurope.2013.6695486>
- Vogel, S., Bach, A., Potter, D., Stevic, M., Pitz, M., & Mirz, M. (2025). *VILLASnode documentation*. <https://villas.fein-aachen.org/docs/node/>.