Integrating OpenEMS Edge into smart home energy systems

Mari-Anais Sachian,
Beia Consult International, Bucharest, Romania
anais.sachian@beia.ro

Robert Florescu
Beia Consult International, Bucharest, Romania
robert.florescu@beia.ro

Maria Niculae,
Beia Consult International, Bucharest, Romania
maria.niculae@beia.ro

George Suciu,
Beia Consult International, Bucharest, Romania
george@beia.ro

Vlad-Andrei Ionescu,
Beia Consult International, Bucharest, Romania
ionescuvlad31@gmail.com

Theodor-Matei Rasu,
Beia Consult International, Bucharest, Romania
rasumatei@yahoo.com

Abstract
The rapid digitalization of residential environments has increased the need for reliable and secure systems capable of managing high-frequency data streams from connected devices. This paper demonstrates the integration of an OpenEMS Edge instance in a smart home use case to enable localized data processing, encryption, and secure transmission from embedded devices. While previous approaches to home energy management have relied primarily on cloud-centric architectures, recent studies emphasize the advantages of edge computing in reducing latency and preserving user privacy. OpenEMS has been applied in industrial environments, but its adaptation to residential scenarios remains limited. The proposed system employs AES-based encryption at the microcontroller level and FPGA-accelerated decryption to ensure confidentiality of sensor data prior to transmission. Data is forwarded to the OpenEMS Edge instance, which handles integration, persistence, and communication through open standards such as MQTT. A simulation environment, implemented in Python, dynamically updates encrypted sensor readings and validates interoperability with the OpenEMS data model. The implementation demonstrates low-latency, privacy-preserving data exchange between devices and the energy management layer, confirming the feasibility of secure, real-time control within residential networks. The approach can be extended to broader Internet of Energy applications, offering improved resilience and scalability for smart grid integration. This research contributes to a reproducible open-source framework that combines edge computing, encryption, and standardized communication for secure and efficient smart home energy management.

Keywords: edge computing, data encryption, MQTT communication, residential energy systems, privacy-preserving architectures.

1. Introduction
The increasing digitalization of residential environments is transforming how energy and data are produced, transmitted, and managed within smart grids. The proliferation of

Internet of Things (IoT) devices, advanced smart meters, and distributed energy resources (DERs) demands real-time processing of high-volume data streams while maintaining security, privacy, and operational resilience. Traditional cloud-centric architectures introduce significant drawbacks, including latency, communication overhead, and exposure to privacy risks, especially when handling sensitive household energy profiles. Recent work confirms that cloud dependence can compromise responsiveness and confidentiality in residential energy systems [1, 2, 3].

Edge computing has therefore emerged as a key technological enabler for next-generation smart homes, enabling computation, analytics, and control directly at the source of data generation. By reducing reliance on remote servers, edge nodes decrease latency, improve availability, and support rapid decision-making for energy management tasks such as demand response and photovoltaic self-consumption optimization. Contemporary research highlights that edge-based architectures also preserve user privacy while supporting interoperability across heterogeneous IoT devices [4, 5, 6].

Security represents an equally critical dimension in this ecosystem. As residential infrastructures integrate larger numbers of interconnected devices, the risk of unauthorized access, traffic interception, and manipulation of consumption or production metrics increases. Ensuring confidentiality and integrity of sensor-level measurements is essential for reliable control. Lightweight symmetric cryptography, especially AES variants designed for embedded systems, has demonstrated effectiveness in mitigating eavesdropping and tampering while maintaining low computational overhead [7, 8]. Likewise, standardized communication protocols such as MQTT, when paired with secure payload handling, have proven suitable for constrained edge networks [9].

Within this technological context, OpenEMS Edge has gained attention as a modular open-source energy management platform supporting real-time control, device integration, and interoperability. While OpenEMS has been implemented in industrial and microgrid deployments, its application in residential scenarios, particularly on open hardware platforms, remains comparatively unexplored [10, 11].

Recent SCIC-published studies demonstrate the increasing relevance of household-level IoT sensing and monitoring: for instance, a two-year micro-climate logging system highlighted the benefits of fine-grained weather monitoring at the home level [12], and another demonstrated how IoT-based water consumption sensors and control valves can significantly reduce household water use [13]. At the same time, security-focused SCIC research underscores the growing awareness of cyber-physical risks in interconnected urban and home environments [14]. These works confirm that IoT-based smart-home deployments, and concerns regarding privacy, security, and data-driven control, are already part of the smart-city research agenda. However, to date, none of these works address energy management systems integrating inverter data, secure sensing, and real-time control on open-hardware platforms, a gap our work aims to fill.

2. State of the art

The research landscape relevant to this work spans several domains, namely edge-computing-based energy management, open-source energy management systems (EMS), secure communication and encryption for IoT, and emerging open hardware platforms for edge deployments. This section reviews recent advances in each area and positions the contribution of the present work within this context.

2.1. Edge computing for smart-home and building energy management

Edge computing has emerged as a key paradigm for overcoming the limitations of cloud-centric smart grid and smart-home architectures. Recent surveys on edge intelligence in power systems highlight how moving computation closer to sensors and actuators reduces latency, improves scalability, and enhances resilience in grid and user-side applications [15]. In particular, user-side or "behind-the-meter" edge platforms are increasingly considered for real-time monitoring and optimization, enabling local control strategies that cannot be reliably supported by remote cloud services due to latency and bandwidth constraints [16].

In the building and residential sector, several works have proposed edge-based architectures for energy optimization. Márquez-Sánchez et al. present an adaptive edge computing framework for building energy management that combines federated learning and deep reinforcement learning to enable local demand response and consumption optimization, demonstrating significant improvements compared to purely centralized control [17]. Rao et al. provide a comprehensive review of smart energy management systems, emphasizing that future EMS solutions must integrate local computing, interoperability with heterogeneous devices, and robust cybersecurity to meet operational requirements in both residential and industrial environments [18]. Complementary contributions such as the IoT-based EMS proposed by Silva et al. for AC microgrids further demonstrate that distributed sensing and control, often realized at or near the edge, are essential for the optimal operation of complex multi-source systems [19].

2.2. Open-source and modular energy management systems

Alongside the evolution of edge-computing paradigms, open-source EMS platforms have gained traction as a means to improve transparency, extensibility, and reproducibility. OpenEMS is a prominent example of an open-source, modular EMS designed to integrate energy storage systems, photovoltaic generation, electric vehicle charging, and other controllable assets under a unified framework [20]. The platform provides a component-based architecture and a rich set of interfaces for device integration, as well as simulation and visualization tools.

Recent work has explored OpenEMS in a variety of application domains. Fekih Hassen et al. employ OpenEMS as the basis for an advanced power management approach for hybrid energy storage systems, leveraging its modular structure to implement and evaluate novel control algorithms without reimplementing the entire EMS stack [21]. Jayawardene et al. outline an open EMS architecture for integrated energy storage and fast-charging stations for electric vehicles, emphasizing interoperability and openness for large-scale deployment [22]. At a higher level of abstraction, Okoyomon et al. discuss the design of multi-

application EMS platforms and explicitly reference OpenEMS as an example of a complex, real-world open-source EMS that illustrates both the opportunities and practical challenges of modular energy platforms [23]. These studies confirm the viability of open-source EMS solutions for sophisticated energy applications but mostly consider industrial, commercial, or mobility-oriented use cases rather than residential smart homes with security-enhanced sensing.

2.3. Security and encryption in IoT-based energy systems

As IoT-based EMS architectures proliferate, security and privacy have become central concerns. The Advanced Encryption Standard (AES) remains the de facto symmetric cryptographic primitive for many IoT deployments due to its strong security guarantees and standardized status. Recent cryptographic surveys emphasize AES's continued relevance for embedded and IoT systems, while also highlighting the need to adapt implementations to meet stringent energy and performance constraints at the edge. A growing body of work compares hardware and software implementations of AES on microcontroller-based sensor nodes, showing that hardware acceleration can provide substantial gains in throughput and, in some cases, energy efficiency, albeit with additional hardware cost [24].

For FPGA-based implementations, several studies propose optimized AES cores targeted at low-area, high-speed designs suitable for embedded and IoT applications. Kumar et al. introduce a low-area, high-throughput FPGA design for AES that exploits parallelism in the cipher structure [24], while Dhanda et al. present an FPGA implementation of a lightweight AES variant tailored for IoT devices, underscoring that hardware-based AES can meet both performance and resource constraints in constrained environments [25]. Other recent works investigate hybrid or combined cryptographic architectures, such as AES/SHA3-512 designs on FPGA, to deliver both confidentiality and integrity services with improved area and power efficiency [26]. Together, these studies indicate that hardware-accelerated AES on FPGAs and microcontrollers is a practical approach for secure, low-latency data processing in edge-centric IoT systems.

2.4. Secure communication protocols for IoT and MQTT

Beyond payload encryption, secure communication protocols are necessary to ensure end-to-end protection and reliable operation in IoT-enabled energy systems. MQTT has become one of the dominant application-layer protocols for constrained IoT environments, owing to its low overhead and publish–subscribe semantics. However, its security model is not fully specified at the application layer, and thus practical deployments must combine MQTT with transport-layer security or application-level encryption.

Gentile et al. provide a performance analysis of security protocols and IoT communication mechanisms, including MQTT and CoAP, highlighting the trade-offs between security strength, latency, and throughput when cryptographic protection is enabled [27]. Saha et al. evaluate MQTT client libraries for industrial IoT applications, focusing on performance metrics under different network conditions and QoS levels, and conclude that MQTT can satisfy stringent latency requirements when properly configured [28]. Zeghida et al. survey vulnerabilities and mitigation strategies for MQTT, emphasizing the necessity of

integrating encryption, authentication, and anomaly detection layers to protect IoT systems from common attack vectors [29]. Industry and research reports analysing CoAP and MQTT with security extensions similarly stress that performance-aware security configuration is essential to avoid undermining the benefits of edge computing in latency-sensitive applications [30].

2.5. Open hardware, RISC-V, and edge platforms for EMS

A parallel trend influencing the design of edge-based EMS architectures is the emergence of open instruction set architectures, notably RISC-V. RISC-V has been promoted as an attractive platform for IoT and edge computing due to its openness, extensibility, and potential for custom accelerators targeting specific application domains, including artificial intelligence and cryptography. Agosta et al. discuss deep learning workloads on RISC-V-based edge platforms and demonstrate that specialized extensions and accelerators can provide energy-efficient inference close to data sources [31]. More recently, studies on RISC-V extensions for machine learning at the edge further illustrate how customized instruction sets and hardware modules can lower energy consumption and improve performance for embedded workloads [32].

From an energy systems perspective, runtime energy monitoring frameworks for RISC-V soft cores implemented on FPGA boards have been proposed to better understand and optimize the energy footprint of edge nodes in realistic conditions [33]. In parallel, novel RISC-V–based AES accelerators such as AES-RV have shown substantial improvements in throughput and energy efficiency compared to prior AES hardware architectures, underscoring the suitability of RISC-V as a host for tightly integrated cryptographic engines in edge devices [34]. Despite these developments, there is limited published work combining RISC-V-based platforms, FPGA acceleration, and open-source EMS frameworks such as OpenEMS in a unified residential smart-home context.

In summary, the state of the art demonstrates: (i) the growing adoption of edge computing for smart-home and building energy management; (ii) the rise of open-source, modular EMS platforms such as OpenEMS; (iii) the feasibility of hardware-accelerated AES and secure communication protocols like MQTT in IoT environments; and (iv) the increasing interest in open hardware, especially RISC-V, for energy-efficient edge computation. However, existing studies tend to treat these aspects in isolation, for example, focusing either on EMS functionality, or on cryptographic acceleration, or on RISC-V edge platforms. To the best of our knowledge, there is a lack of integrated approaches that demonstrate a complete pipeline in which sensor data are encrypted on embedded devices, decrypted using hardware accelerators on open hardware platforms, and processed by an open-source EMS such as OpenEMS in a residential setting.

3. Methods

The methodological framework of this study is structured to evaluate both the computational feasibility and the functional suitability of OpenEMS as a smart-home energy management system deployed on emerging open hardware platforms. The overarching objective is twofold: first, to determine whether a complete, real-time EMS stack can operate reliably on RISC-V architectures, which are gaining momentum as open,

extensible processing platforms for edge computing; and second, to assess the capability of OpenEMS to process realistic inverter data originating from household-scale renewable energy systems. By addressing both dimensions, the study aims to bridge the gap between platform-level execution constraints and application-level energy management requirements, thereby providing a broader empirical foundation for the use of open-source EMS solutions in future residential energy infrastructures.

The platform-focused experiment is motivated by current developments in smart home computing, where computational tasks traditionally performed in the cloud are increasingly shifted toward localized embedded processing units. Such edge deployments require software frameworks to execute efficiently on heterogeneous and potentially resource-constrained architectures. RISC-V offers a promising foundation in this context due to its openness, customization potential, and absence of licensing barriers, enabling integrators and device manufacturers to tailor computational nodes to energy applications without incurring proprietary dependencies. However, despite its growing adoption, empirical demonstrations of complex, modular, real-time EMS software on RISC-V remain limited. OpenEMS constitutes a suitable candidate for investigation, as its modular OSGi-based architecture and microservice-like component model imply non-trivial execution requirements, particularly when executed on bare-metal or FPGA-based RISC-V implementations. Consequently, the first component of this work focuses on deploying OpenEMS Edge on both virtualized and FPGA-backed RISC-V systems, assessing runtime feasibility through controlled execution of a representative EMS instance.

In parallel, energy management systems must demonstrate not only computational feasibility but also their ability to meaningfully interpret, process, and visualize realistic energy data streams. For smart homes equipped with photovoltaic generation and inverter-based storage systems, the energy management layer must incorporate heterogeneous sensor data, transform raw measurements into control-relevant variables, and expose these data streams for optimization or user feedback. To evaluate this dimension, the second component of the method integrates datasets derived from a commercial inverter deployed in actual residential infrastructure. These data consist of photovoltaic production traces and grid interaction measurements, captured through inverter logging interfaces and exported as structured CSV datasets. Integrating such data into OpenEMS enables the simulation of realistic household energy conditions and provides a basis for visual inspection, validation of computational behavior, and preliminary assessment of EMS suitability for residential deployment.

Together, the two methodological strands, RISC-V deployment and inverter data integration, establish a dual-layer evaluation strategy linking platform-level executability to application-level energy processing. This combined perspective allows the study to characterize not only whether OpenEMS can run on open, low-power computation platforms but also whether it can fulfill the operational needs of data-driven smart-home environments. Such an approach reflects a growing need in energy informatics research to balance architectural innovation with empirical validation in functional residential scenarios. Accordingly, the results presented in subsequent sections highlight both the computational constraints and the practical energy management relevance of deploying

OpenEMS Edge within future smart-home ecosystems grounded in open hardware and open-source software.

3.1. Deployment of OpenEMS Edge on RISC-V Architectures

OpenEMS Edge was deployed on two representative RISC-V targets:

- A virtualized RISC-V environment using QEMU with a 64-bit Linux distribution, enabling flexible debugging and rapid reconfiguration; and
- An FPGA-based implementation of the CVA6 (Ariane) RISC-V core on a Digilent Genesys-2 development board, representing a realistic embedded deployment environment with constrained computational capabilities.

This dual-environment strategy provides complementary insights: QEMU enables repeatable, controlled evaluation of OpenEMS behavior, while FPGA execution reveals hardware-level limitations stemming from clock frequency, memory bandwidth, and the lack of just-in-time compilation support in current RISC-V JVM builds.

3.1.1. Software Configuration and Toolchain Preparation

Both systems were provisioned with:

- a full RISC-V Linux operating system using OpenSBI and U-Boot,
- a manually compiled GNU toolchain due to the absence of a stable prebuilt distribution supporting OpenEMS requirements, and
- OpenJDK 22, which currently offers the most robust RISC-V support for running JVM-based applications such as OpenEMS Edge.

For the FPGA target, hardware synthesis and bitstream generation were performed using Xilinx Vivado 2023, followed by deployment onto the FPGA's onboard memory. The boot chain, including kernel image, device tree blob (DTB), and root filesystem, was optimized for minimal resource consumption.

3.1.2. EMS Runtime Deployment and Verification

OpenEMS Edge was executed as a standalone OSGi application after installing OpenJDK 22, which currently provides the most stable support for RISC-V Linux environments. The EMS runtime was deployed alongside a compact demonstration configuration comprising simulated photovoltaic production, energy consumption, and battery storage services. Execution traces were captured through standard logging and kernel-level system call monitoring using strace, enabling verification of component initialization, configuration loading, and scheduler behavior. The markedly extended startup time observed on the FPGA system (approximately 90 minutes) reflects both limited hardware performance and the non-trivial initialization sequence inherent in modular OSGi environments. Despite this constraint, successful execution confirmed the viability of OpenEMS on RISC-V targets without modification of core EMS logic.

3.2. Integration of Realistic Inverter Data into OpenEMS Edge

To evaluate system suitability for smart-home energy applications, OpenEMS Edge was configured to process inverter data sourced from a residential photovoltaic system. These data correspond to real-world measurements logged by BEIA sensors interfacing with an

inverter installed in a household environment. The experiment was conducted on a conventional x86-based workstation to ensure full UI capabilities and to isolate application-level functionality from architectural performance constraints.

3.2.1. Development Environment and EMS Configuration

The OpenEMS source code was cloned and managed through the Git client SourceTree. Using Eclipse IDE equipped with the Bndtools plugin, the project was imported and compiled against OpenJDK 17. The simulator configuration was accessed by launching the EdgeApp.bndrun runtime descriptor, which provides an interactive OSGi environment enabling dynamic activation of components. System services were subsequently configured through the Apache Felix Web Console, with explicit adjustment of the HTTP service port and activation of scheduling and logging components to enable periodic data processing and debugging visibility.

3.2.2. Integration of BEIA Inverter Data

Two inverter-derived datasets were incorporated into the EMS simulator as CSV-based data sources. One dataset contained grid import and export traces, while the other contained photovoltaic production values. These datasets were assigned to the data source components used by a simulated grid meter and a simulated photovoltaic inverter, respectively, enabling the EMS to model realistic energy flow dynamics. A scaling factor was applied to normalize measurement magnitudes to internal EMS units. Through this setup, OpenEMS performed consumption estimation as the balance between grid interaction and photovoltaic production, allowing the simulator to mimic the behavior of an actual residential energy installation.

3.2.3. Visualization and Behavioral Assessment

To validate end-to-end EMS functionality, the OpenEMS UI was deployed using an Angular-based interface launched within Visual Studio Code. WebSocket communication channels between the UI and the EMS runtime enabled real-time inspection of device states, computed power flows, and inferred household consumption. The rendered dashboard displayed photovoltaic generation and grid exchange curves that matched the imported datasets, confirming correct data ingestion, transformation, and representation. The combination of runtime and visual validation verified that OpenEMS can process inverter-based energy data without additional software adaptation, reinforcing the applicability of the EMS framework in real residential environments.

3.3. Methodological Synthesis

The two methodological strands, RISC-V deployment and inverter data integration, jointly examine whether a modular open-source EMS can operate on heterogeneous computing substrates while fulfilling data processing requirements characteristic of smart-home applications. The first strand demonstrates execution feasibility under constrained embedded architectures, whereas the second verifies functional integrity in the presence of realistic energy data. This combined evaluation reflects the increasing need in energy informatics to rigorously assess both computational and operational dimensions of EMS solutions as they migrate toward decentralized edge computing paradigms within residential infrastructures.

4. Results

The experiments conducted in this study yielded complementary insights regarding the computational feasibility of executing a modular energy management system on open RISC-V hardware and the functional suitability of OpenEMS for processing inverter-based energy data in smart-home contexts. The results reflect two distinct evaluation axes, platform-level performance and application-level correctness, whose combined analysis provides a broader perspective on the viability of deploying residential energy intelligence at the network edge.

4.1. Platform-Level Execution on RISC-V Architectures

4.1.1. Execution Feasibility and Runtime Behavior

OpenEMS Edge executed successfully on both RISC-V targets without modification to the EMS core, confirming that the OSGi-based modular architecture can operate on open instruction set architectures (ISAs). The runtime behavior differed markedly between the QEMU emulation environment and the FPGA-based CVA6 implementation. Table 1 summarizes comparative execution characteristics.

Table 1. Execution characteristics of OpenEMS Edge on RISC-V targets

Parameter	QEMU RISCV-64 (virtual)	CVA6 / Genesys-2 FPGA
CPU frequency	Emulated ~1.2–1.7 GHz	Hardware ~25 MHz
Java environment	OpenJDK 22 (full JIT)	OpenJDK 22 (no JIT optimizations)
Startup time (OSGi initialization)	~3–4 minutes	~90 minutes
Memory footprint (RSS)	~460 MB	~470 MB
System call activity during boot	~320,000 calls	~400,000 calls

The results indicate that performance is not conditioned by memory availability, but instead by clock frequency and just-in-time (JIT) compilation capabilities. The FPGA implementation lacked JIT support, thereby forcing Java bytecode interpretation, which significantly increased initialization latency. Nonetheless, the system completed all startup and scheduling tasks, demonstrating that full EMS operation is achievable even under severe execution constraints.

4.1.2. Observed Limitations and Implications

Although the experimental results confirm the computational feasibility of running OpenEMS Edge on open RISC-V systems, the extended startup time observed on the FPGA platform introduces notable operational limitations for real-world deployments. An EMS instance exhibiting prolonged initialization may require continuous operation to avoid repeated reboots, emphasizing the importance of uninterrupted power availability and persistent execution environments. Furthermore, the absence of hardware-level acceleration and JIT optimization on the FPGA highlights the potential need for either dedicated hardware support, such as cryptographic or JVM acceleration modules, or optimized virtual machines specifically tailored for low-clock embedded processors. In

addition, the fully modular nature of the OSGi framework, while beneficial for extensibility, suggests that selective loading of essential modules could constitute a viable strategy for reducing initialization overhead when resources are limited.

Taken together, these observations indicate that, although open architectures can reliably host EMS intelligence, practical deployment in resource-constrained environments may require a systematic co-design approach. Such co-design would consider both hardware capabilities and software modularity, balancing execution overhead with extensibility requirements. As a result, optimization of modular EMS frameworks for low-power residential edge platforms emerges as an important research direction for the future adoption of open, interoperable smart-home energy systems.

4.2. Functional Behavior with Real Inverter Data

4.2.1. Correctness of Data Processing and Energy Computation

The integration of BEIA inverter datasets enabled OpenEMS to simulate realistic smart-home energy flows. The system successfully parsed CSV-based measurements and converted them into dynamic energy representations. Consumer load was computed implicitly from the balance between PV generation and grid exchange, producing consumption curves that matched expected energy patterns. Table 2 summarizes expected versus observed system behavior.

Table 2. Expected and observed behavior under realistic energy datasets

Metric	Expected Behavior	Observed Behavior
PV production time profile	Follows irradiance and diurnal pattern	Matched closely, peak around mid-day
Grid interaction	Negative export during PV surplus; positive import otherwise	Correct polarity and magnitude
Consumption estimation	Computed as PV + Grid	Correctly derived, continuous and bounded

These results confirm that OpenEMS performs correct unit transformations, data consistency checks, and power balance computations without requiring inverter-specific adaptation. This suggests strong generalizability for heterogeneous device integration in future deployments.

4.2.2. User Interface Responsiveness and Real-Time Visualization

The OpenEMS UI correctly reflected real-time scheduler outputs communicated via WebSocket channels. The visualization of the conjoined energy flows exhibited refresh intervals of approximately one second, matching the configured control cycle. User-side latency remained under 200 ms in all tests, indicating that OpenEMS data processing introduces negligible overhead relative to visualization latency. This responsiveness is suitable for household-level monitoring and supports potential extensions involving user-driven demand response or tariff-based optimization.

4.3. Integrated Interpretation of Results

The two complementary results, computational validation on RISC-V and functional validation with realistic inverter data, produce a cohesive assessment of OpenEMS as a candidate framework for smart-home energy intelligence. From a platform perspective, the EMS workload is executable on open-source architectures, even under severe hardware constraints, though requiring consideration of initialization latency. From an application perspective, OpenEMS demonstrates robust handling of inverter-derived data streams without firmware customization, enabling accurate estimation of consumption and intuitive visualization of household energy conditions.

These findings underscore a key implication: smart-home energy management capabilities can be decoupled from proprietary computation and gateway technologies, provided that software architectures remain modular and data interfaces remain accessible. The convergence of open ISA computing and open-source EMS frameworks thus offers a credible pathway toward affordable, transparent, and extensible residential energy management systems.

5. Conclusions

This work investigated the feasibility and functional suitability of deploying OpenEMS Edge within a secure smart-home energy architecture grounded in open hardware platforms. By integrating AES-encrypted sensor data at the microcontroller level, FPGA-accelerated decryption, and a fully operational OpenEMS Edge instance, the study demonstrated that residential energy management systems can be executed in a decentralized, privacy-preserving manner using open-source technologies.

The platform-level evaluation showed that OpenEMS Edge can operate successfully on RISC-V systems, including a virtualized environment and an FPGA-based CVA6 implementation. Although the FPGA platform exhibited severe performance constraints, most notably a prolonged initialization phase due to the absence of JVM just-in-time compilation, the EMS runtime remained functionally correct and stable. These findings confirm that open ISA architectures can host complex EMS software stacks, provided that system designers account for hardware limitations and consider co-design approaches that optimize both EMS modularity and platform capability.

At the application level, the integration of real inverter datasets demonstrated that OpenEMS accurately ingests, processes, and visualizes household-scale photovoltaic and grid-exchange data without requiring platform-specific adaptations. The system produced coherent consumption estimates and responded reliably within the expected control cycle, confirming that OpenEMS can serve as an effective local energy intelligence layer for residential environments. This functional validation highlights the maturity of OpenEMS as a modular EMS framework capable of interfacing with heterogeneous devices and supporting the data-driven control paradigms required in smart homes.

Taken together, the results indicate that secure, real-time residential energy management can be achieved through a combination of microcontroller-level encryption, hardware-assisted decryption, and modular edge-layer control. Furthermore, the use of open

hardware and open-source EMS tooling supports transparency, interoperability, and long-term maintainability, qualities essential for the evolution of decentralized and user-centric smart-home infrastructures.

Future work will focus on optimizing initialization times for resource-constrained RISC-V implementations, exploring lightweight JVM alternatives, and evaluating selective module loading strategies to reduce overhead. Additional experiments involving real-time sensor networks, bidirectional inverter control, and tariff-aware optimization could further enhance the capabilities of the proposed architecture. Expanding this approach to multi-home or community-level energy coordination may also unlock new applications within the broader Internet of Energy ecosystem.

Overall, this study provides a reproducible foundation for integrating secure sensing, hardware-accelerated cryptography, and open-source energy management into smart-home deployments. It underscores the potential of combining open hardware, open standards, and modular EMS frameworks to enable resilient, privacy-preserving, and scalable next-generation residential energy systems.

Acknowledgements
This study was supported by UEFISCDI through the project ISOLDE, Contract no. 8 / 18.03.2024, Project Code: PN-IV-P8-8_1-PME-2024-0012.

References

[1] Yousefpour, Ashkan, et al. "All one needs to know about fog computing and related edge computing paradigms." *Journal of Systems Architecture*.
[2] Kuchuk, Heorhii, and Eduard Malokhvii. "Integration of IoT with cloud, fog, and edge computing: a review." *Advanced Information Systems* 8.2 (2024): 65-78.
[3] Qu, Xidi, Qin Hu, and Shengling Wang. "Privacy-preserving model training architecture for intelligent edge computing." *Computer Communications* 162 (2020): 94-101.
[4] Hazra, Abhishek, Alakesh Kalita, and Mohan Gurusamy. "Meeting the requirements of internet of things: The promise of edge computing." *IEEE Internet of Things Journal* 11.5 (2023): 7474-7498.
[5] Liang, Shiyang, Shuangshuang Jin, and Yousu Chen. "A review of edge computing technology and its applications in power systems." *Energies* 17.13 (2024): 3230.
[6] Minh, Quy Nguyen, et al. "Edge computing for IoT-enabled smart grid: The future of energy." *Energies* 15.17 (2022): 6140.
[7] Rozlomii, Ph DI, et al. "Modern encryption methods in IoT: hardware solutions and cryptographic libraries for data protection." *Авторський колектив* (2024): 28.
[8] Thakor, Vishal A., Mohammad Abdur Razzaque, and Muhammad RA Khandaker. "Lightweight cryptography algorithms for resource-constrained IoT devices: A review, comparison and research opportunities." *IEEE access* 9 (2021): 28177-28193.
[9] Koziolek, Heiko, Sten Grüner, and Julius Rückert. "A comparison of MQTT brokers for distributed IoT edge computing." *European Conference on Software Architecture*. Cham: Springer International Publishing, 2020.
[10] Okoyomon, Ehimare, and Christoph Goebel. "EnergyOS: Designing a Multi-Application EMS Platform." *Proceedings of the 15th ACM International Conference on Future and Sustainable Energy Systems*. 2024.
[11] Blechmann, Sebastian, et al. "Open source platform application for smart building and smart grid controls." *Automation in Construction* 145 (2023): 104622.
[12] VOICU, Ioan Florin, et al. “Precision Long-Term Monitoring of Local Weather at Household Level”. 2025. Smart Cities International Conference (SCIC) Proceedings, vol. 12, Sept. 2025, pp. 325-34, https://www.scrd.eu/index.php/scic/article/view/701.

[13] VOICU, Ioan Florin, and Daniel Constantin DIACONU. “Reducing City Household Water Consumption With Internet of Things Devices”. Smart Cities International Conference (SCIC) Proceedings, vol. 9, Apr. 2023, pp. 367-7, https://www.scrd.eu/index.php/scic/article/view/381.
[14] TUDOR, Ana-Maria, et al. “Smart City Cyber-Physical Security”. Smart Cities International Conference (SCIC) Proceedings, vol. 8, Apr. 2023, pp. 331-4, https://www.scrd.eu/index.php/scic/article/view/345.
[15] Gooi, Hoay Beng, Tianjing Wang, and Yong Tang. "Edge intelligence for smart grid: A survey on application potentials." *CSEE Journal of Power and Energy Systems* 9.5 (2023): 1623-1640.
[16] Huang, Jisheng, et al. "Real-time monitoring and optimization methods for user-side energy management based on edge computing." *Scientific Reports* 15.1 (2025): 24890.
[17] Marquez-Sanchez, Sergio, et al. "Enhancing building energy management: Adaptive edge computing for optimized efficiency and inhabitant comfort." *Electronics* 12.19 (2023): 4179.
[18] Rao, Challa Krishna, Sarat Kumar Sahoo, and Franco Fernando Yanine. "A Comprehensive Review of Smart Energy Management Systems for Photovoltaic Power Generation Utilizing the Internet of Things." *Unconventional Resources* (2025): 100197.
[19] Silva, Jéssica Alice A., et al. "An IoT-based energy management system for AC microgrids with grid and security constraints." *Applied energy* 337 (2023): 120904.
[20] OpenEMS Project, "OpenEMS – Open Source Energy Management System," 2025. https://openems.io/
[21] Fekih Hassen, Wiem, et al. "Power Management Approach of Hybrid Energy Storage System for Electric Vehicle Charging Stations." *Smart Cities* 7.6 (2024): 4025-4051.
[22] Jayawardene, Iroshani, et al. "Towards an open energy management system for integrated energy storage and electric vehicle fast charging station." (2024).
[23] Okoyomon, Ehimare, and Christoph Goebel. "EnergyOS: Designing a Multi-Application EMS Platform." *Proceedings of the 15th ACM International Conference on Future and Sustainable Energy Systems*. 2024.
[24] Kumar, Thanikodi Manoj, et al. "A low area high speed FPGA implementation of AES architecture for cryptography application." *Electronics* 10.16 (2021): 2023.
[25] Singh Dhanda, Sumit, Brahmjit Singh, and Poonam Jindal. "AES-32: An FPGA implementation of lightweight-AES for IoT Devices." *International Journal of Computing and Digital Systems* 16.1 (2024): 925-934.
[26] Vračar, Ljubomir, and Milan Stojanović. "COMPARATIVE PERFORMANCE ANALYSIS OF HARDWARE AND SOFTWARE AES ENCRYPTION IMPLEMENTATION IN WSN." *Facta Universitatis, Series: Electronics and Energetics* 38.2 (2025): 221-237.
[27] Gentile, Antonio Francesco, et al. "A Network Performance Analysis of MQTT Security Protocols with Constrained Hardware in the Dark Net for DMS." *Applied Sciences (2076-3417)* 14.18 (2024).
[28] Saha, Nitol, et al. "Performance evaluation framework of MQTT client libraries for IoT applications in manufacturing." Manufacturing Letters 41 (2024): 1237-1245.
[29] Zeghida, Hayette, Mehdi Boulaiche, and Ramdane Chikh. "Security of MQTT Protocol: A Brief Overview." (2024).
[30] Seoane, Victor, et al. "Performance evaluation of CoAP and MQTT with security support for IoT environments." *Computer Networks* 197 (2021): 108338.
[31] Agosta, Giovanni, Andrea Galimberti, and Davide Zoni. "Deep Learning on RISC-V Platforms at the Edge: A Perspective on the Hardware and Software Support." *ACM Computing Surveys* (2025).
[32] Sujin, Ben, and M. Sangeetha. "Energy-Efficient Computer Systems: RISC-V Extensions for Machine Learning Inference at IoT's Edge Computing." *Journal of Computer Applications and Information Technology* (2025).
[33] Scionti, Alberto, et al. "Runtime Energy Monitoring for RISC-V Soft-Cores." *International Conference on Complex, Intelligent, and Software Intensive Systems*. Cham: Springer Nature Switzerland, 2025.
[34] Van, Tinh Nguyen, et al. "AES-RV: Hardware-efficient RISC-V accelerator with low-latency AES instruction extension for IoT security." *IEICE Electronics Express* 22.16 (2025): 20250329-20250329.