computers

MDPI

Article

Realization of a Gateway Device for Photovoltaic Application Using Open-Source Tools in a Virtualized Environment †

Emmanuel Luwaca * and Senthil Krishnamurthy *

Center for Intelligent Systems and Emerging Technologies, Department of Electrical, Electronic, and Computer Engineering, Cape Peninsula University of Technology, Bellville Campus, Cape Town 7535, South Africa

* Correspondence: e.luwaca@gmail.com (E.L.); krishnamurthys@cput.ac.za (S.K.)

† This article is an extended version of a paper entitled [A practical evaluation of an IEC 61499 standard-based Low-Cost Gateway for photovoltaic power plants], which was presented at [SAUPAC, Centurion South Africa, 29–30 January 2025].

Abstract

Electronic communication and industrial protocols are critical to the reliable operation of modern electrical grids and Distributed Energy Resources (DERs). Communication loss between devices in renewable power plants can lead to significant revenue losses and jeopardize operational safety. While current control and automation systems for renewable plants are primarily based on the IEC 61131-3 standard, it lacks defined communication frameworks, leading most deployments to depend on Original Equipment Manufacturer (OEM)-specific protocols. The IEC 61499 standard, in contrast, offers a reference model for distributed automation systems, introducing Service Interface Function Blocks (SIFBs) and high-level communication abstractions that enable hardware-independent integration. This study proposes adopting the IEC 61499 standard for DER automation systems to enhance interoperability and flexibility among plant components. A photovoltaic power plant gateway is developed on a virtualized platform using open-source tools and libraries, including Python version 3, libmodbus version 3.1.7, and open62541 version 1 The implemented gateway successfully interfaces with industry-validated software applications, including UAExpert and Matrikon OPC Unified Architecture (OPC UA) clients, demonstrating the feasibility and effectiveness of IEC 61499-based integration in DER environments.

Keywords: automation; distributed control systems; distributed energy resource; IEC 61131-3; IEC 61499; IEC 62541; Modbus; OPCUA; virtualization

Academic Editor: George Angelos Papadopoulos

Received: 19 September 2025
Revised: 24 October 2025
Accepted: 28 November 2025
Published: 1 December 2025

Citation: Luwaca, E.; Krishnamurthy, S. Realization of a Gateway Device for Photovoltaic Application Using Open-Source Tools in a Virtualized Environment. *Computers* **2025**, *14*, 524. https://doi.org/10.3390/computers14120524

1. Introduction

Southern Africa's energy demand is on the rise and is expected to continue increasing. A study predicts that demand in the Southern African region might increase by eight to fourteen times between 2010 and 2070 [1]. In recent years, there has been a growing drive to deploy renewable energy systems as an ecologically acceptable energy generation strategy.

Over the past two decades, South Africa has embraced renewable energy as one of the means to rapidly increase its generating capacity and combat the rolling blackouts (load shedding) it has experienced. As part of South Africa's Integrated Resource Plan (IRP) of 2019, the government plans to roll out renewable power (adopt clean energy) and reduce its carbon emissions by decommissioning old coal power plants in phases. Approximately 5400 MW of coal-fired generation capacity is earmarked for decommissioning. This number will increase to 10,500 MW by 2030 and 35,000 MW by 2050 (South Africa Department of Energy, 2019). South Africa's Integrated Resource Plan (IRP) of 2023 further emphasizes

South Africa's commitment to a diverse energy generation mix, with a short-term action framework that aims to deliver an additional 3615 MW of Photovoltaic (PV), 4468 MW of wind, 3743 MW of Battery Energy Storage System (BESS), etc., of renewable power by the year 2030.

However, the veritable nature of renewable energy resources as part of a developing country's energy mix presents technical and economic challenges. Renewable energy resources pose challenges to power systems in terms of stability, dependability, and interoperability. However, Distributed Energy Resources (DER) systems are among the ways being explored globally to sustain rising energy demand whilst reducing carbon emissions. DERs also have social, economic, and environmental benefits. The objectives of an intelligent grid include improving energy efficiency, reducing the carbon footprint, and providing an open framework for information exchange between power system devices [2]. Given the dispersed nature of devices in a DER power plant, communication is vital for controlling them. The currently installed power plant utilizes algorithms based on the well-established IEC 61131-3 standard [3], featuring a central Power Plant Controller (PPC) that employs proprietary or basic protocols, such as Modbus. There are no defined standardized interfaces, protocols, and services. Using legacy communication protocols has led plant owners to rely heavily on original equipment manufacturers throughout all project phases.

In contrast to IEC 61131-3, the IEC 61499 standard proposes a distributed conceptual reference architecture for industrial automation applications. The IEC 61499 standard architecture has stimulated the development of new engineering technologies that offer portability, reusability, and interoperability for distributed control and automation applications [4]. IEC 61499 open-source tools, such as the 4DIAC integrated development platform, also support communication protocols such as TCP/IP, Modbus, and IEC 62541- OPC Unified Architecture (OPC UA) using open-source libraries. This study examines how existing open-source tools can be used to monitor and control a solar power plant. A solar plant is simulated using MATLAB, and the various measurements that are key in monitoring and controlling a solar power plant, such as irradiance, temperature, power (active and reactive), and metering, are monitored using third-party tools. The research work applies IEC 61499 to distributed energy resources. It begins with an introduction in Section 1, followed by a literature review. Section 3 discusses MATLAB TCP communication setup in Section 2. Section 3 covers tests conducted and the addition of communication protocols to the 4Diac Forte runtime, and Section 4 provides the implementation of a Modbus-to-OPC gateway using IEC 61499 Service Interface Function Blocks (SIFBs) in 4Diac. Section 4 also discusses the use of openPLC61850 to achieve IEC 61850 communication. Finally, Section 5 of the paper concludes with a summary of the results achieved and discusses ongoing research activities.

2. Literature and Review on the IEC 61499 Standard

The IEC 61499 standard provides an open reference architecture for industrial automation applications, with key features such as object-oriented modeling using function blocks as basic elements and event-driven execution [5]. In the programming domain, a function block is a class with both attributes and methods. The inputs to a function block are parameters or attributes passed to the class during instantiation. The IEC 61499 standard introduces advanced software technologies as one of its means to provide a reference framework that enables compatibility between distributed control system applications from different vendors [6]. The framework introduces the use of software functions, including encapsulation of functionality, Component-based design, and Event-driven execution. The IEC 61499 standard defines several basic functional and structural entities for modeling and implementing distributed control and automation applications. The main functional

entities are function block types, resource types, device types, and applications built from function block instances [7], as shown in Figure 1.

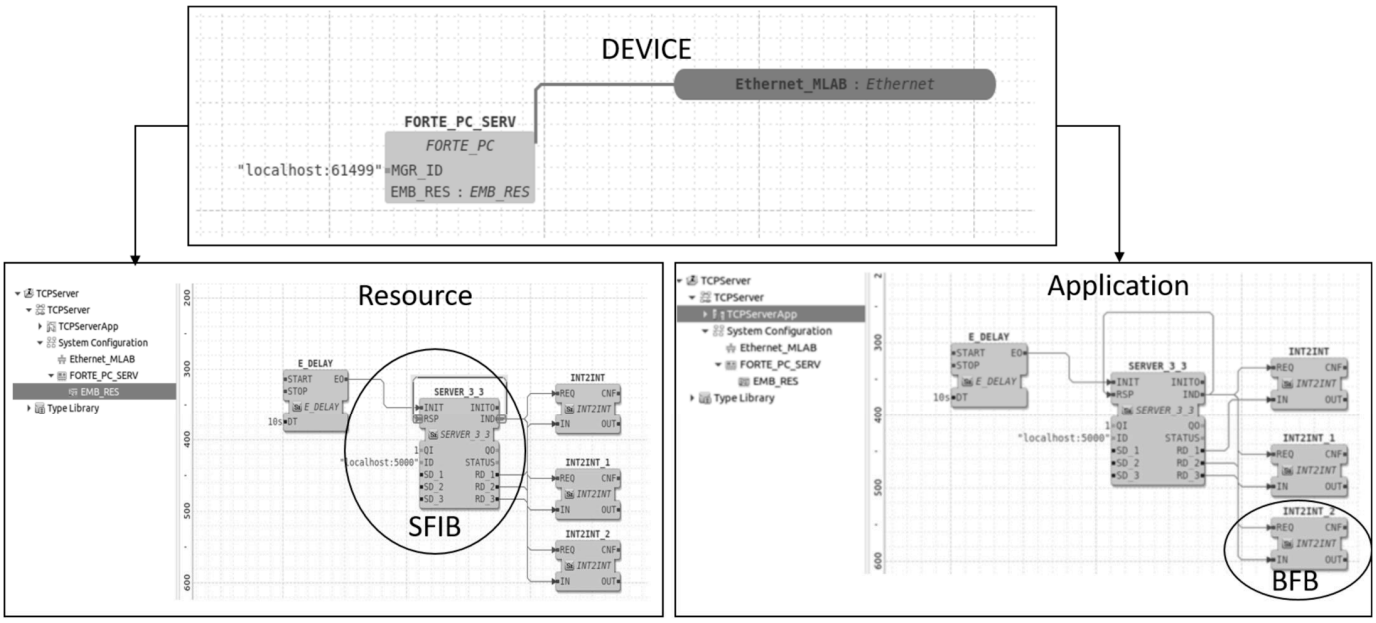


Figure 1. Function model of the IEC 61499.

The basic function block model (orange box in Figure 1) is the basis for the IEC 61499. A Function Block generally provides an Interface for Event Inputs/Outputs and Data Inputs/Outputs and encapsulates several algorithms. The standard defines four function block types: Simple Function Block, Basic Function Block, Composite Function Block, and Service Interface Function Block (SIFB). Using IEC 61499-compliant software tools, users can encapsulate algorithms into basic function blocks using IEC 61131-3 programming languages or any other high-level language supported by an Integrated Development Environment (IDE). The function blocks are event-driven, and events trigger the algorithm's internal processing. The association between event inputs, outputs, and the execution of the algorithms is specified using an Execution Control Chart (ECC), which is a state machine. Another essential function block introduced in the IEC 61499 standard is the Service Interface Function Block (SIFB) type [8]. The SIFB represents the interface to the low-level services provided by the operating system or hardware. By utilizing SIFB function blocks, the IEC 61499 runtime can support various communication protocols. Figure 2 shows some of the interfaces and services that can be enabled and supported using SIFBs [8]. These interfaces and services in IEC 61499-compliant applications can connect to hardwire input/output modules, and wrapping interfaces can connect to legacy systems.

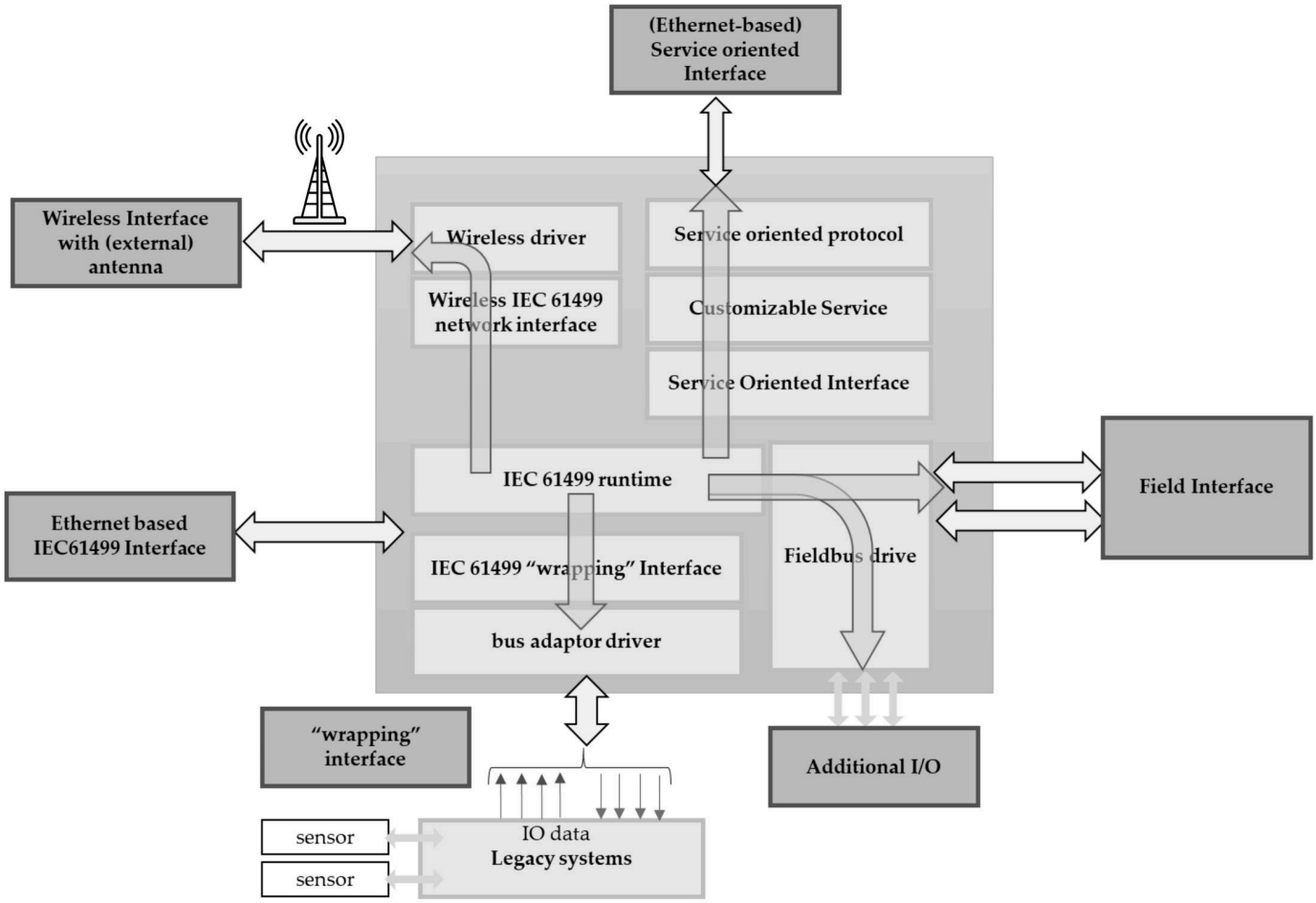


Figure 2. Functional model of an automation system based on IEC-61499.

In an IEC 61499 application, resources are the backbone that facilitates the integration of different components into a distributed application. However, SIFBs are the inputs and outputs of systems that provide access to the physical world [9]. The IEC 61499 standard introduces a new and innovative paradigm for industrial automation. However, it is not widely supported or used in the industry yet. The IEC 61499 standard version 1 has been in existence since the early 2000s. A review of the literature following the publication of the IEC 61499 version standard primarily focused on explaining the concepts it presented. An overview of the concepts and ideas of the IEC 61499 standard, along with enhancements between editions 1 and 2, is discussed in [9]. However, the evaluation of a new standard at the time of publication did not convince all automation researchers to adopt it. The paper [10] surveys the research conducted on IEC 61499 at the time of its publication and highlights deficiencies in the IEC 61499 standard from a software perspective. The same author in [11] compares IEC 61131-3 and IEC 61499 and argues that academics are promoting IEC 61499 based on unsubstantiated claims about its main features. In contrast to the allegations made in [10–12] presents methods for integrating IEC 61131-3 and IEC 61499 to exploit the benefits of both standards.

Although industry support for IEC 61499 remains limited, it can serve as a foundation for developing Industry 4.0 applications, as detailed in [12]. The study presented in [13] utilized a commercial third-party communication stack (SISCO MMS stack ease lite) to propose a flexible intelligent electronic device model based on the IEC 61850 standard. The study in [14] explored the use of open standards such as IEC 61499 for smart grid applications. The paper [15] discusses the mapping between IEC 61850 and IEC 61499. To make IEC 61499 more applicable to industry. Three main challenges in its transition

were identified in [4], including industrial concerns and technical issues, with ongoing research on transforming modeling techniques and implementation tools. Paper [16] discusses the concept of an Intelligent Logical Node (iLN) as a possible use case. Meanwhile, ref. [17] discusses a practical application of the IEC 61499 standard for structuring Cyber-Physical Systems. In summary, the IEC 61499 standard provides tools to future-proof automation systems. However, due to the industry's slow adoption of the IEC 61499 standard, its benefits have not yet been fully realized. This manuscript highlights the practical application of IEC 61499 in PV plants. Table 1 summarizes the use-case scenarios for IEC 61499.

Table 1. IEC 61499 evaluation.

Criteria	Evaluation of IEC 61499 to Support Criteria	Use Case
Standardization	Standardized models using function blocks.	DCS
Modular, Portable, and Reusable FBs	Supports the use of modular, reusable, and portable FBs	PLCs, DCS, IoT
Interoperability	Limited. runtime interoperability is not detailed	Multi-vendor integration
Scalability	Supports both small and large applications	PLC and DCS
Tools	Limited—both open-source and commercial	PLC and DCS
Certification	Lacks a strong certification ecosystem.	

3. MATLAB TCP Communication

This section describes the TCP/IP communication blocks in the MATLAB/Simulink environment. This study examines the Simulink TCP client-server communication block, specifically how MATLAB encodes and decodes various data types transmitted over TCP/IP. Simulink can communicate with remote applications using the standard MATLAB Transport Communication Protocol (TCP) over Internet Protocol (IP) communication blocks. The base element of the block was an S-function block that used a C MEX file [18]. The client block enables the transmission of live data from the Simulink model to an application in this study. The remote application is a Python code that runs on a Linux Ubuntu Virtual Machine (VM) and an IEC 61499 TCP Server. RFC 1180 provides a detailed TCP/IP communication model. Figure 3 shows an example of TCP/IP communication between a client and a server device.

3.1. Simulink PV Plant Simulation with TCP Client Implementation

To provide a more accurate depiction of a real-world plant, this section discusses the simulation of a PV plant in MATLAB (https://www.researchgate.net/publication/351733228_Design_and_Simulation_of_100_MW_Photovoltaic_Power_Plant_Using_Matlab_Simulink, accessed on 31 May 2023). Maximum Power Point Tracking (MPPT) is a well-known control method used in PV systems to provide the maximum available active power from the PV array to the grid. However, to prevent grid instability caused by energy, active power must be controllable. Active power control should be considered alongside grid frequency variation and voltage stability, as in traditional power plants [19]. Maximum power point tracking ensures the maximum power is delivered to the grid.

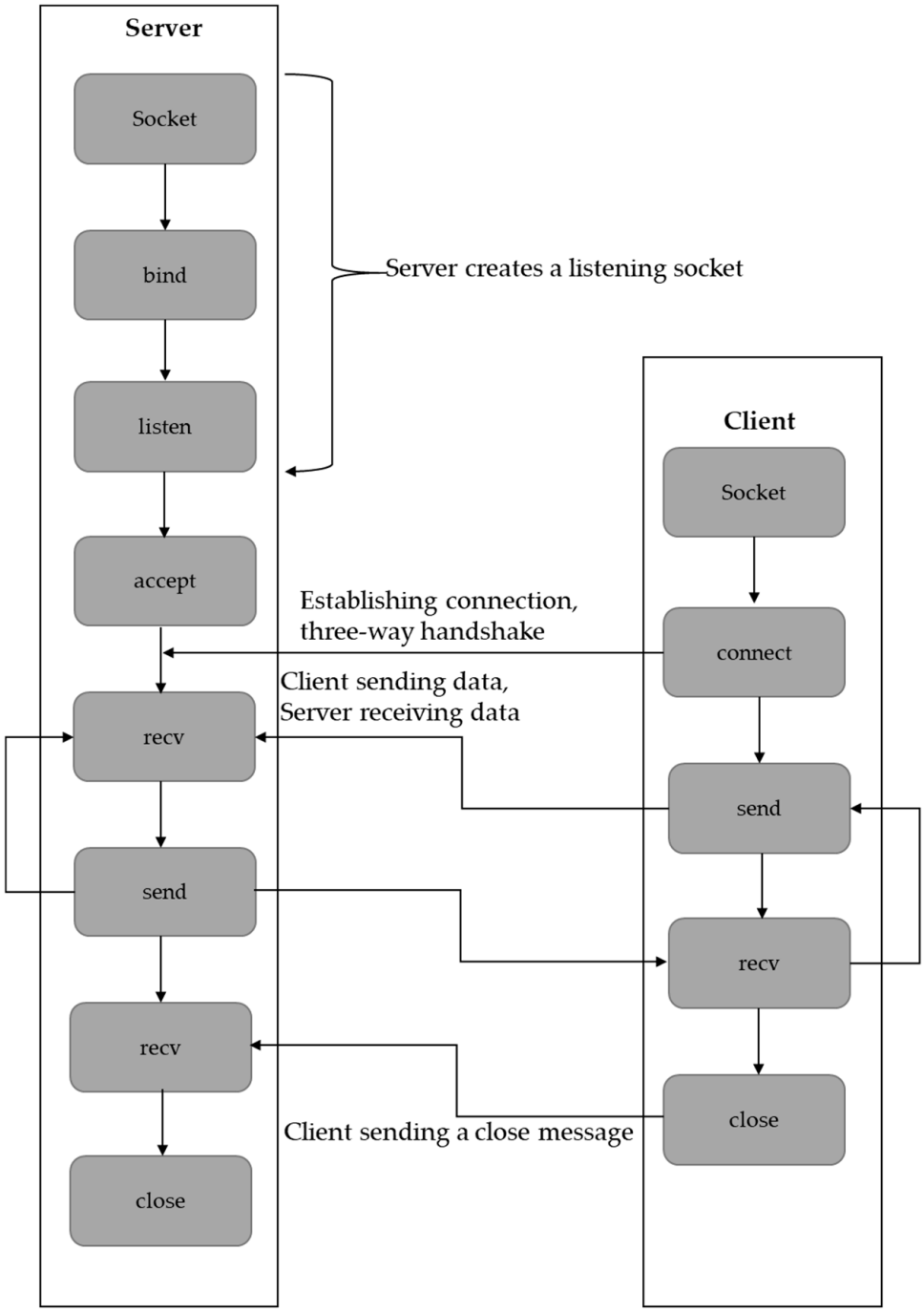


Figure 3. TCP/IP connection establishment and message exchange between client and server.

The PV system in Figure 4 is simulated using MATLAB, and the MATLAB array is built of parallel string modules, each consisting of multiple string modules connected in series [20]. Figure 5 shows the maximum power point tracking implementation using the Incremental Conductance (IC) method for the controller. We have added communication functionality to the PV plant simulation, highlighted in orange.

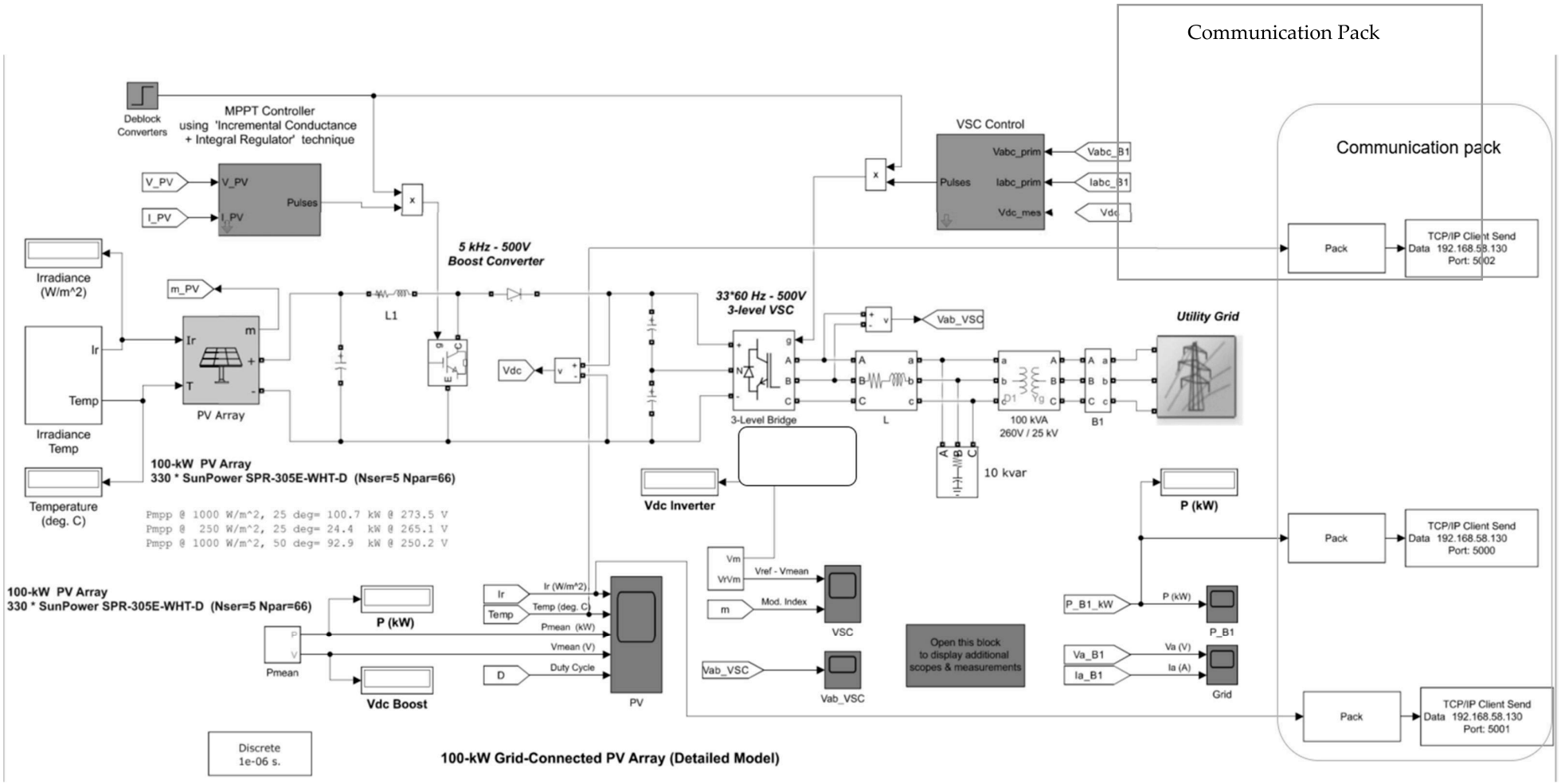


Figure 4. 100 kW Grid-Connected PV Plant. NOTE: 330 * refers to the maximum power at standard testing conditions for the SunPower SPR-305E panels in the model.

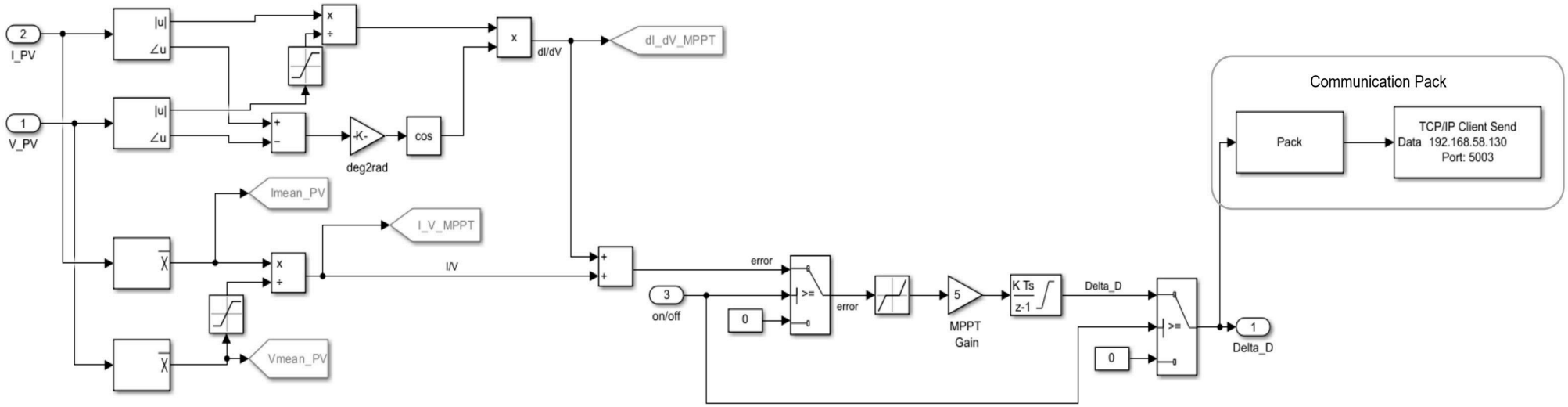


Figure 5. Maximum power point tracking by the incremental conductance method.

The modeled PV plant in Figure 4, has Maximum Power Point Tracking (MPPT) implemented using the 'Perturb & Observe' technique. The MPPT controller in this model is based on the IC + IR (Incremental Conductance + Integral Regulator) technique, as shown in Figure 5 [21]. Incremental conductance looks to increasingly contrast the ratio of the derivative of conductance with the instantaneous conductance as shown in Figure 6, and is based on the fact that the power slope of a photovoltaic is null at the maximum power point (dP/dV = 0), and can be found in terms of using (1) and (2) [21].

$$\frac{\delta P_{PV}}{\delta V_{PV}} = \frac{\delta(V_{PV} \times I_{PV})}{V_{PV}} = V_{PV} \times \frac{\delta I_{PV}}{\delta V_{PV}} + I_{PV} = 0 \quad (1)$$

where the change in photovoltaic current is ∂IPV, the change in photovoltaic voltage is ∂VPV, and ∂PPV is the change in photovoltaic power, respectively. Equation (1) can be written as:

$$\frac{\delta I_{PV}}{\delta V_{PV}} = -\frac{I_{PV}}{V_{PV}} \approx \frac{\Delta I}{\Delta V} \quad (2)$$

where photovoltaic current and voltage ΔI and ΔV are the increments, respectively.

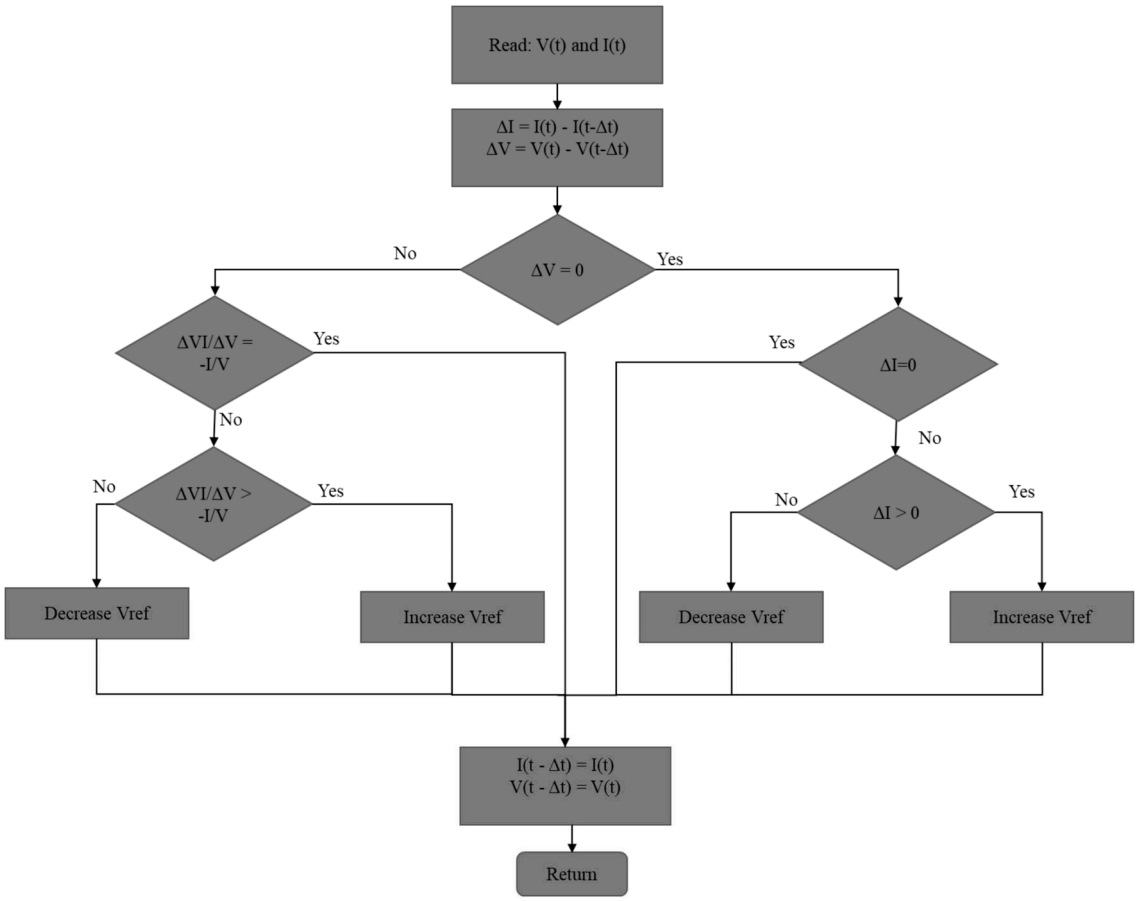


Figure 6. Flowchart of the IC Algorithm.

The IC+IR technique depicted in Figure 5, adjusts (increases and decreases) Vref and Iref until I(t − Δt) = I(t) and V(t − Δt) = V(t) so that the modeled PV array delivers a maximum power of 100 kW at 1000 W/m^2 sun irradiance and 25 °C temperature. The measurements obtained from the simulated PV plant are shown in Figure 7. A sample of the dataset used in the simulation is shown in Table 2.

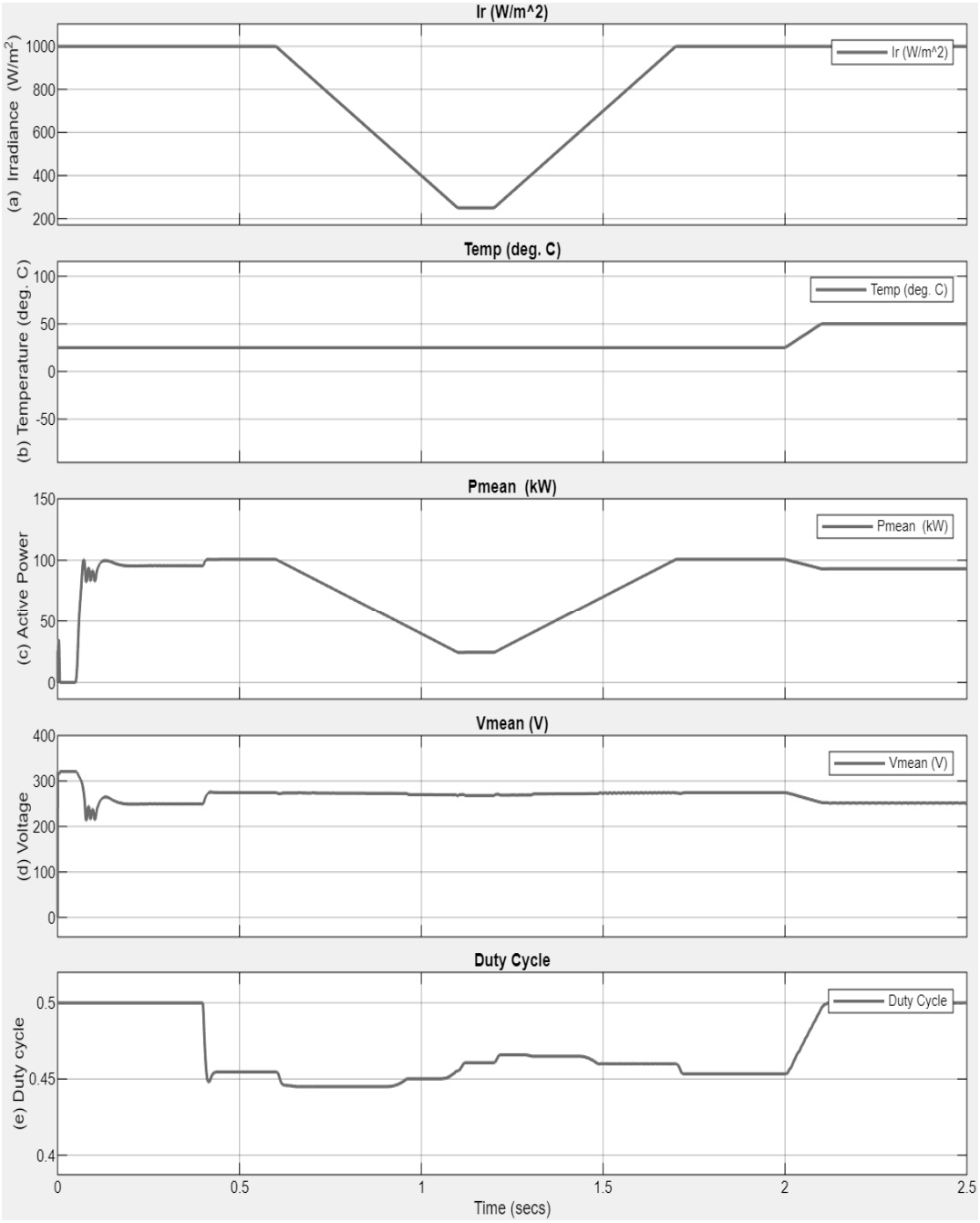


Figure 7. Output Waveform of Simulated PV Plant. (**a**) Irradiance, (**b**) Temperature, (**c**) Active power, (**d**) Voltage and (**e**) Duty cycle respectively.

Table 2. Extract of the irradiance and temperature dataset used in the Matlab simulation.

Time	Irradiance	Temp
0	1000	25
1.1	250	25
1.2	250	25
1.7	1000	25
2.1	1000	50

The Simulink model of the PV plant, shown in Figure 5, enabled us to observe its behavior. The model allows us to test the stability of the simulated PV plant and understand the I-V and PV output characteristics under various conditions. In Figure 7, the datasets of

irradiance (a) and temperature (b) were simulated to observe the behavior of the PV system. Irradiance and temperature are signals typically measured using field instruments such as pyranometers and Resistance Temperature Detectors (RTD). The simulation allowed us to test the PV system using simulated data for these measurements. The simulation started with a high irradiance of 1000 W/m^2 and a steady temperature of 25 °C between 0 and 0.4 s. Because there was good irradiance and a stable panel temperature, the PV plant was started with the inverter thyristors pulsed at a 50% duty cycle (c). Active power (c) was ramped up to almost the full load of 100 kW, and the voltage stabilized at approximately 250 V. Between 0.4 and 0.6 s, the system's response to the IC algorithm optimization for maximum power point tracking was observed. During this period, the duty cycle (e) varied, resulting in a slight increase in the output power (c) and voltage (d). The impact of shading was simulated for 0.6–1.7 s. The output active power (c) was reduced to less than 25% of the rated maximum output at the point of maximum shading. Between 2 and 2.4 s, the impact of the temperature increase (b) became apparent. An increase in temperature results in an increased duty cycle (e), which results in a decrease in the active power produced (c) and reduced voltage (d), even though there is no decrease in the available irradiance (a). In using a simulation study. We enabled communication between Simulink and external devices via TCP/IP. A Python and IEC 61499 4Diac TCP server was used to receive the transmitted data for external consumption and could be used by other applications. Different data types can be packed for transmission, such as "uint," "double," and "bool," bool. The application is interested in "Real" floating-point numbers; therefore, the data are transmitted as doubles.

3.2. Testing Simulink Communication with a TCP Server Implemented Using a Python Script

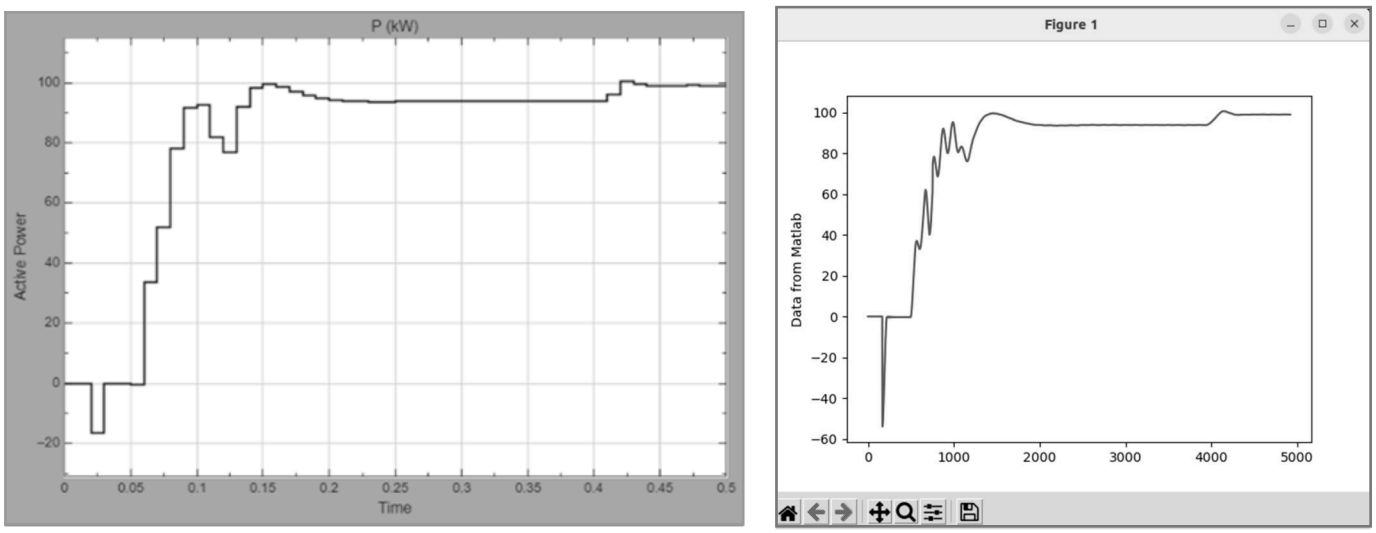
Python provides two levels of access to network services, and through the "socket" library, it provides the essential socket support in the underlying operating system, facilitating the implementation of client(s) or server(s) connections. Python supports creating custom messages by interacting with the operating system kernel via sockets, such as the Socket class. SOC_STREAM. These sockets can create a TCP Server that listens for connections on a specified communication port. These sockets can be used to create a TCP Server that listens for connections on a specified communication port. In Figure 8, the TCP server listens for connections on TCP port 5000 and allows five simultaneous connections. Once the server was started, the MATLAB client was used to connect and send data to the TCP server.

```
def server_program():
    # get the hostname
    'host = socket.gethostname()'
    host = '192.168.58.130'
    port = 5000  # initiate port
    server_socket = socket.socket()
    server_socket.bind((host, port))
    server_socket.listen(5)
```

Figure 8. Python TCP Server.

Matching the encoding mechanism used by the sending device when decoding the packet is essential, as using an incorrect decoding mechanism results in an inaccurate representation of the data sent by the sending device. The Python 'struct library' was used during this test, a module that converts between Python values and C structs represented as Python bytes objects. The library can use the first character of the format string to indicate the byte order, size, and alignment of packed data.

The right-hand side of Figure 9 shows the hard-coded payload size and challenges with retransmitted packets. The challenge with retransmission stems from unacknowledged transmissions caused by variable, unpredictable payload lengths. In this investigation, Python was used to understand how MATLAB publishes TCP data and how payload data is encoded/decoded. The developed understanding is then used to implement the TCP server using IEC 61499, as shown in Figure 10. Using the developed IEC 61499 function blocks and data exchange, the PV plant output was controlled by varying the duty cycle of the Insulated-Gate Bipolar Transistor (IGBT) in the power inverter. Section 4 discusses how the data exchanged by the 4Diac application is transferred to upstream systems, such as the Supervisory and Data Acquisition (SCADA) system.



(**a**) (**b**)

Figure 9. Active power output vs. data captured by the python script. (**a**) Active power sent out from Matlab; (**b**) Data captured by the Python script.

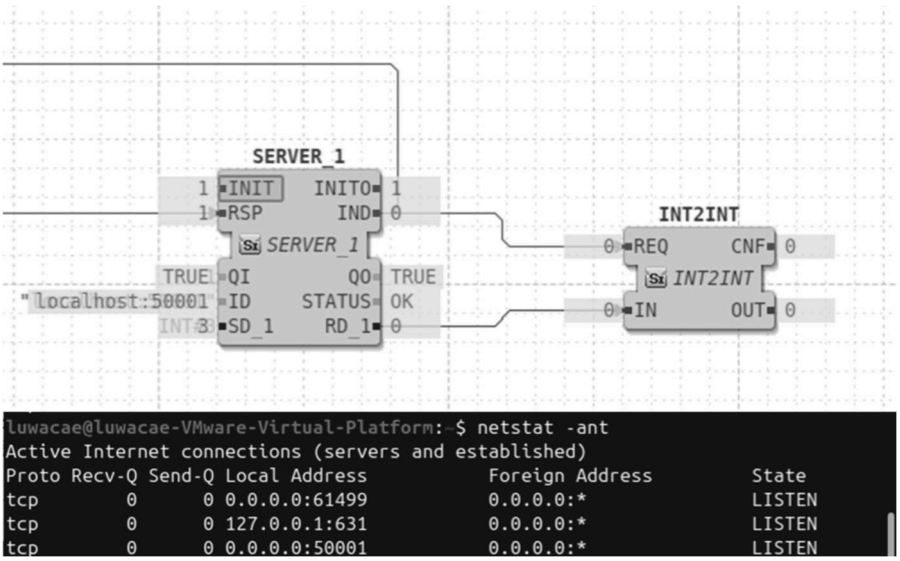


Figure 10. TCP Server implemented using IEC 61499.

4. Testing and Implementation

4.1. Testing Communication Protocols on 4DIAC

4DIAC is an open-source project that includes an Integrated Development Environment (IDE) for developing IEC 61499-compliant distributed control applications [22]. The

4DIAC IDE includes standard elements for IEC 61499 projects, including System, Application, Device, and Function Blocks (Simple, Basic, and SIFB). The 4diac runtime (FORTE) is available in two versions: (1) a prebuilt runtime environment for both Windows and Linux operating systems. (2) The source code of the runtime is also available for developing Custom Function Blocks and building additional functionality onto the 4diac (Forte) runtime. Using the source code version, users can also add communication libraries to the 4diac runtime, enabling communication functionality without having to interface with them programmatically. The Client/Server and Publisher/Subscriber SIFB are included in the Forte runtime available on the Eclipse website. Modbus communication and other fieldbus protocols require additional libraries to be compiled into the Forte runtime, specifically the libmodbus library, to function with 4Diac. Figure 11 shows how the Forte library is compiled with additional functionality for custom function blocks and communication libraries using CMake. NOTE: This is after the libraries have been installed on the machine.

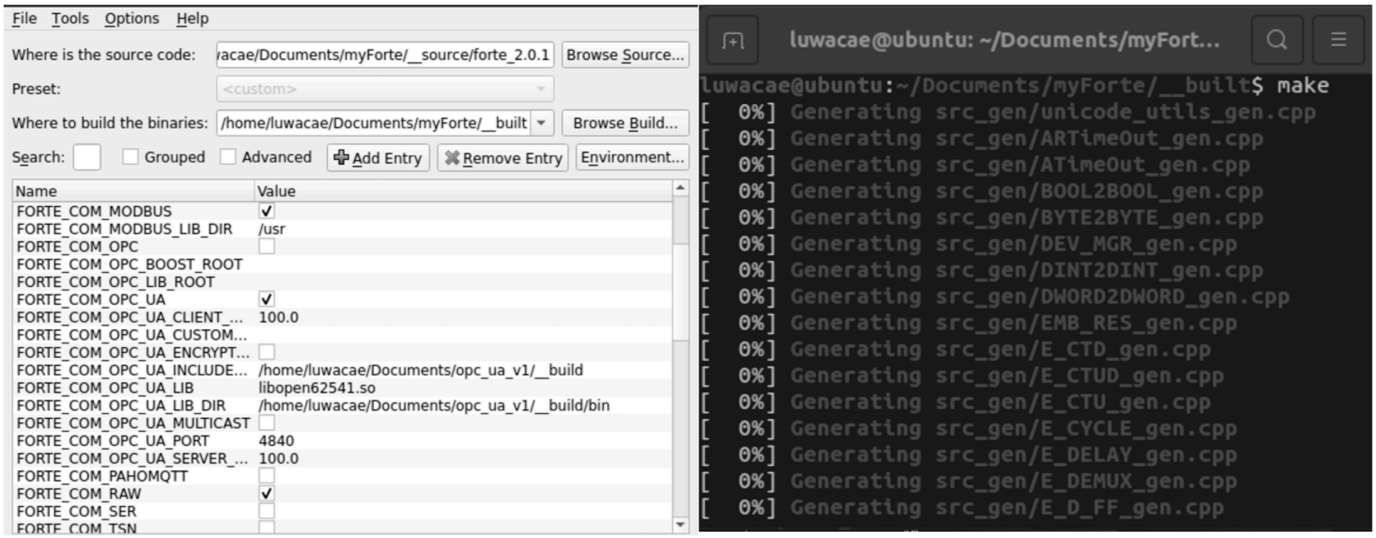


Figure 11. Building custom function blocks and additional communication protocols of the Forte runtime.

Once the libmodbus library is compiled into the 4Forte runtime, the IEC 61499-defined generic CLIENT/SERVER SIFB for bidirectional data exchange and the PUBLISH/SUBSCRIBER SIFB for unicast data exchange are used to facilitate communication with external devices. The standard CLIENT/SERVER SIFB is defined in Extensible Markup Language (XML) and is easily customizable within the application or using an XML editor, as shown in Figure 12. The updated SIFB interface function block is imported into the 4Diac environment. The newly imported SIFB in Figure 13 is used in Figure 14 to read several Modbus registers from the Modbus server.

Figure 14, illustrates the Modbus TCP communication between the 4Diac IDE and a substation Intelligent Electronic Device (IED). The Client SFIB block in this illustration was modified by editing the XML file to receive and send eight tags of the "ANY" data type. The "ANY" data is a placeholder for defining the data objects. This C++ feature provides a type-safe container for storing a single value of any type. The "ANY" data are associated at runtime based on the data type of the input or output. In Figure 14, the SIFB connects to the SEL751A relays and reads the Modbus values for row 33, which correspond to the pushbutton LEDs on the relays. The returned values are represented as a structured data type {FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, TRUE}, corresponding to the relay LED status. The Modbus response packet for the input register poll is shown in Figure 15.

Figure 12. Editing the server.fbt inputs and outputs using an XML editor.

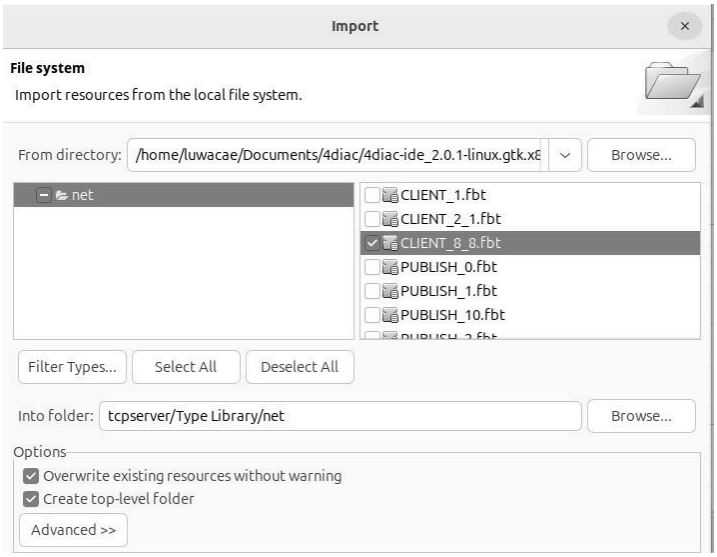


Figure 13. Importing of the updated SIFB with addition inputs.

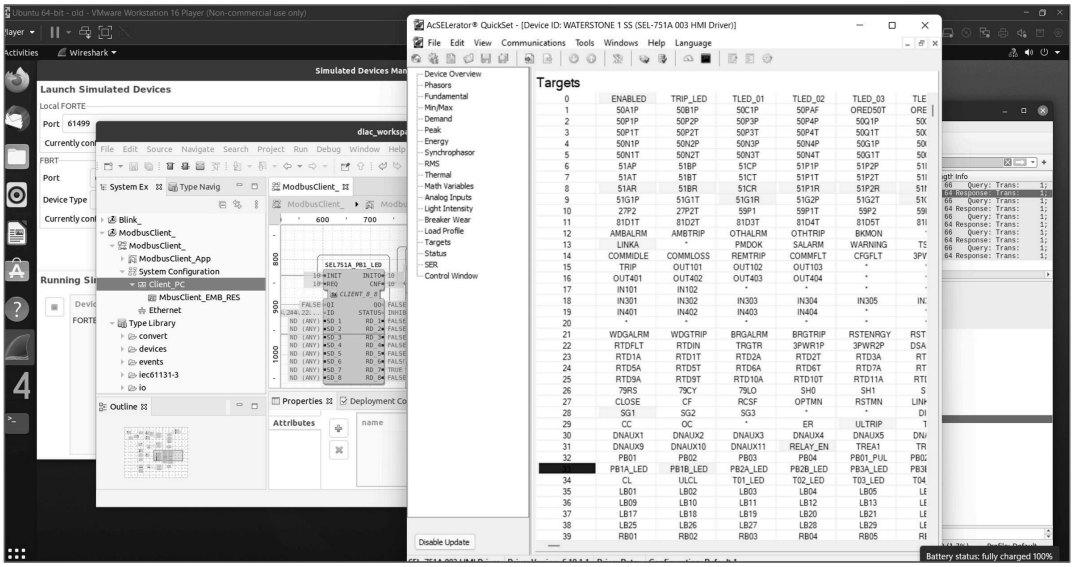


Figure 14. Using the 4diac client SIFB to communicate with the SEL751A using Modbus protocol.

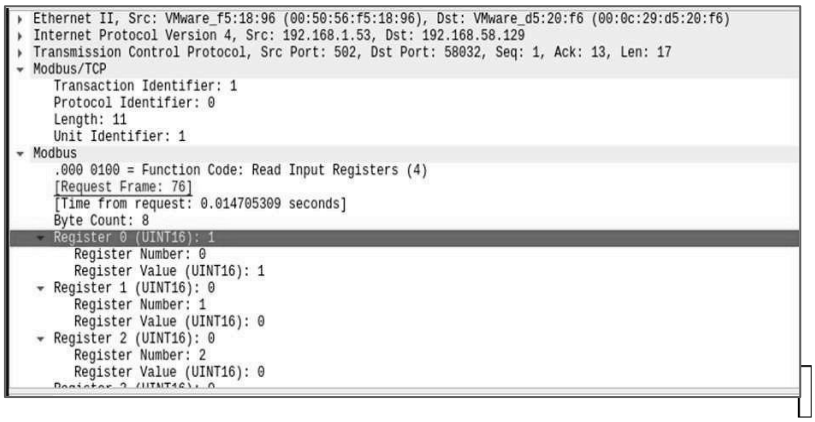


Figure 15. Modbus TCP Input register response.

4.2. Implementing a Low-Cost Gateway Application

4.2.1. Using 4DIAC—OPC UA

The simulation of a PV plant using MATLAB Simulink, which allows access to other Photovoltaic (PV) plant data. We also discussed TCP/IP and Modbus communications. Section 4.1 details how to build communication libraries on 4diac. Modbus TCP communication is used to read the slave devices Modbus registers. The collected values can be served to the OPC client such SCADA or third party protocol tools such as UA Expert.

OPC UA can be included in the FORTE runtime, similar to the Modbus protocol described in Section 4.1. Another alternative for adding OPC UA support discussed in papers is to have SIFBs that provide at least part of the functionality, or to develop an OPC wrapper that can interact with the runtime to provide protocol support [23]. The OPC UA open-source library (open6254) can be compiled into the Forte runtime in 4diac and used as a standard service interface function block, as defined in IEC 61499. OPCUA offers built-in cybersecurity features, including authentication and encryption. Users can use X.509 certificates to authenticate applications, encrypt, and sign messages. A prototype gateway device was developed on Linux Ubuntu using 4Diac, as shown in Figure 14. The data were sourced from MATLAB using TCP/IP client/server communication and a Modbus server. This data is then available to upstream streams, such as the SCADA master, using OPC UA as depicted in Figure 16.

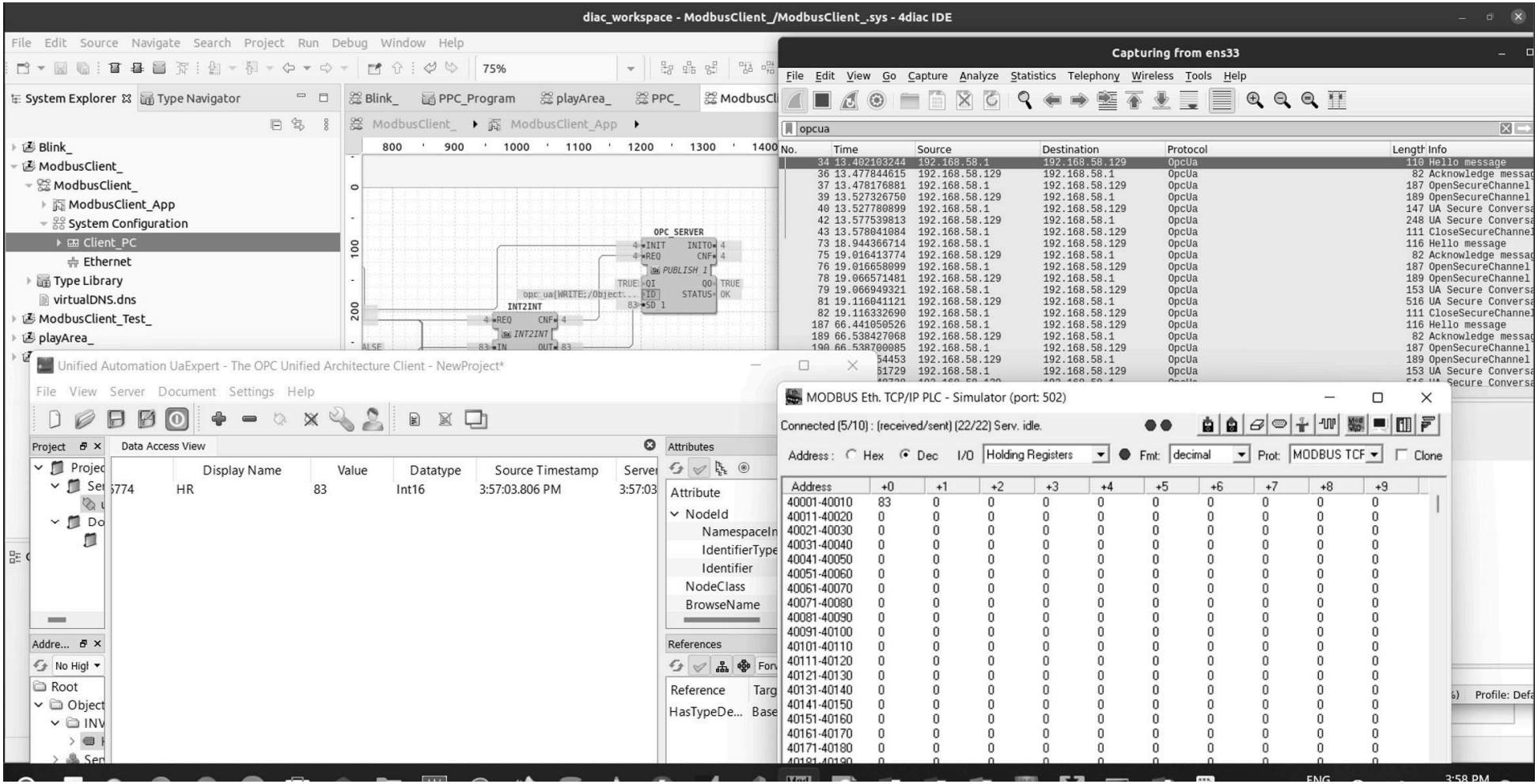


Figure 16. Gateway implementation using Modbus and OPCUA.

The latency tests were conducted using Wireshark, and the OPC UA interface was measured at approximately 75.7 ms, within the expected range for low-latency industrial communication. The sampling interval of the communication framework highlights that the tested framework can support update rates on the order of tens to hundreds of milliseconds, depending on the computational platform and protocol overhead. The research work acknowledges that OPC UA can achieve refresh rates of around 100 ms in typical low-latency networks. In contrast, full OPC UA stacks on low-performance embedded devices (e.g., Raspberry Pi) may introduce additional latency, which should be considered in real-time deployments. Future work is proposed to conduct comprehensive performance tests, including latency and throughput measurements, to validate operational suitability for time-critical distributed control applications. These revisions provide concrete evidence of communication speed and clarify the operational constraints and feasibility of the proposed framework.

Lightweight security mechanisms suitable for low-performance CPUs (e.g., message-level encryption using Python's cryptography library and token-based authentication) can be integrated into the proposed framework. OPC UA secure communication features, such as Secure Channels and User Token Policies, can be incorporated through the Service Interface Function Block (SIFB) abstraction layer, and Trade-offs between computational overhead and security robustness are considered for deployment on embedded controllers such as Raspberry Pi. These additions clarify the framework's potential to adopt OPC UA-based secure communication in future implementations while maintaining efficiency and scalability.

Another standard gaining momentum for renewable power plants is IEC 61850. IEC 61850-7-420 defines the IEC 61850 information models for DER applications and the exchange of information within DER systems [24]. Supplementing and facilitating understanding of the IEC 61850 standard are technical reports, such as IEC TR 61850-90-11. The IEC TR 61850-90-11 technical report describes the functional view of logic based on existing logical nodes for generic process automation, along with the logic's operational modes. Furthermore, the technical report provides a standardized methodology for describing the logic that can be developed to realize object-oriented modeling of logical nodes. Figure 17 shows the functional view of the IEC 61850 logical node.

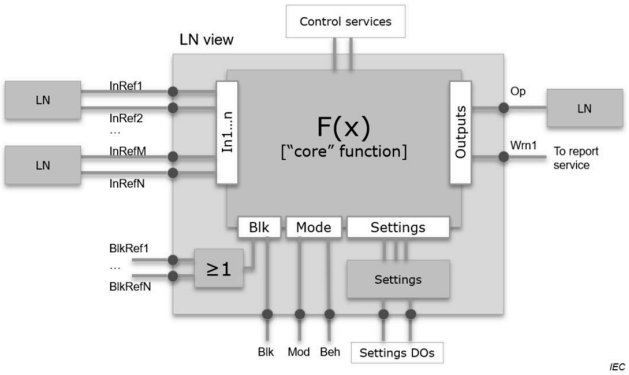
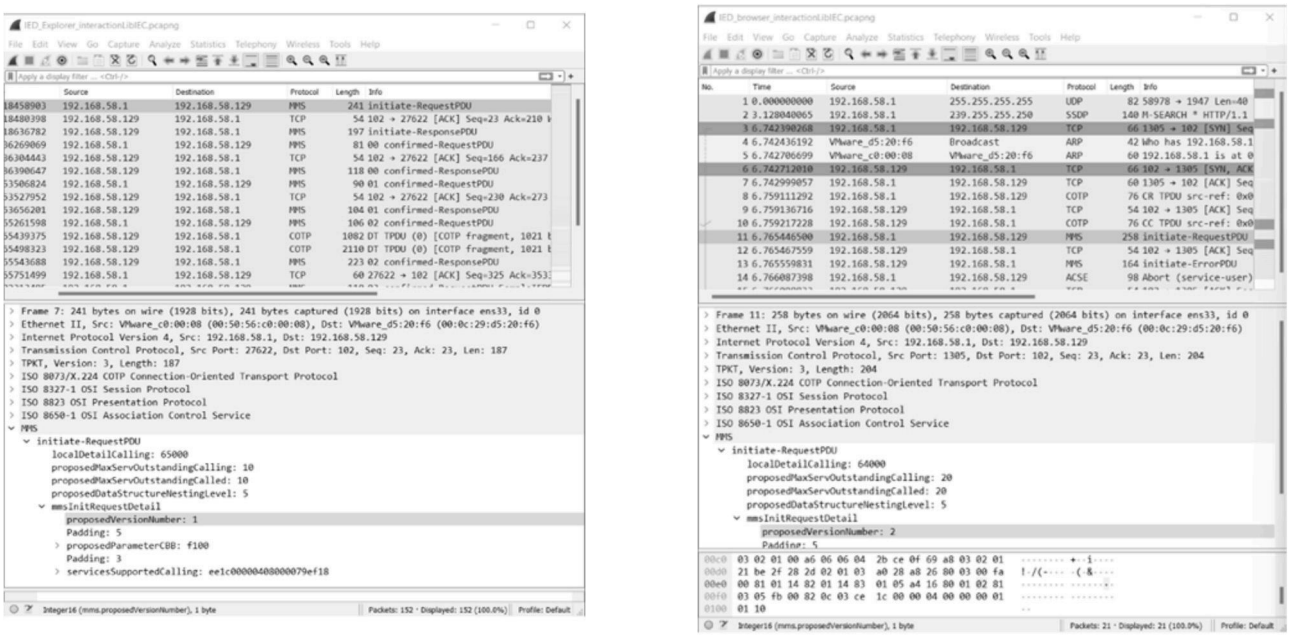


Figure 17. Functional view of a logical node [25].

In Figure 17, the "core" function has input and output data. The function itself could have a blocking input (Blk), implemented using OR gates, i.e., a close block on protection of Intelligent Electronic Devices (IEDs). The function can be implemented using IEC 61131-3 or IEC 61499 Programmable Logic Controllers (PLCs). An open-source library identified through the literature search for implementing IEC 61850 communication is LibIEC61850. The LibIEC61850 supports the Generic Object-Oriented Substation events (GOOSE), Sampled Values (SV), and Manufacturing Message Specification (MMS) protocols

defined in IEC 61850. The LibIEC61850 library is demonstrated in [26] for a metering application. The LibIEC61850 library supports both versions 1 and 2 of IEC 61850 as shown in Figure 18. Version 2 of the IEC 61850 standard provides cybersecurity features such as: (1) Authentication, which can be implemented using X.509 certificates; (2) Security and confidentiality using Transport Layer Security (TLS); (3) Role-based access control (RBAC). The technical report document IEC 61850-90-19 applies this method to IEC 61850.

The libIEC61850 library, much like the open62541 and LibModbus libraries discussed earlier in this section, can be compiled and built into integrated development environments such as 4diac, leveraging the existing logic engines within these environments. Although the libIEC61850 library can be compiled into the 4Diac runtime application, licensing issues under the Eclipse Public License (EPL) prevent its use. To test the libIEC61850 within an automation IDE, the following section discusses a gateway application in Section 4.2.2 using OpenPLC61850.

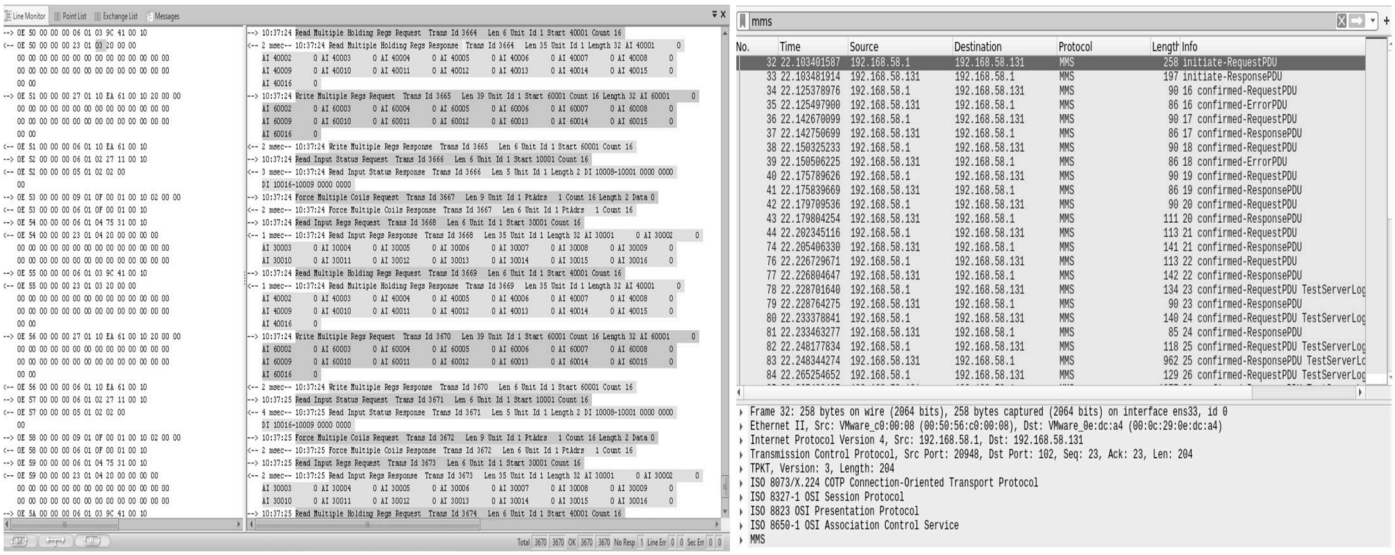


(**a**) (**b**)

Figure 18. libIEC61850 MMS testing. (**a**) Using an IEC 61850 version 1 client (iedExplorer); (**b**) using an IEC 61850 version 2 client (Siemens IEC browser).

4.2.2. IEC 61850-USING OpenPLC61850

OpenPLC61850 is an integrated development environment developed by the authors in [27], expanding on the initial OpenPLC developed by [28]. OpenPLC61850 integrates the LibIEC61850 library on the openPLC web-based application developed by the author in [28]. The openPLC61850 application uses several open-source libraries, such as libmodbus, openDNP3, and LibIEC61850, to support communication protocols, enabling the development of a gateway application compliant with IEC 61131-3. Figure 19a shows the Modbus data exchange between the ASE Test set, acting as the Modbus TCP server, and OpenPLC61850, acting as the Modbus client. OpenPLC61850 then serves this data to upstream devices as an IEC 61850 MMS Server. Figure 19b shows the message exchange between the IEC 61850 client and server device (OpenPLC61850).



(**a**) (**b**)

Figure 19. Modbus and MMS message exchange. (**a**) ASE and OpenPLC61850 Modbus data exchange); (**b**) MMS data exchange between OpenPLC61850 Server and IEC 61850 Client.

All the software components used to implement the gateway application using openPLC61850 are shown in Figure 20. The ASE Test set software (purple border), a commercially licensed protocol-testing device, is used to simulate a Modbus TCP server. The data served by this application is read by the OpenPLC61850 application, which performs the gateway function and serves it to the MMS client (orange border). The data received from the slave device to the PLC address shown in Table 3, allowing logic to be developed based on the measured values and for mapping onto MMS logical nodes using the mapping wrapper on OpenPLC61850.

Table 3. Slave device IO address mapping on OpenPLC61850.

Device Name	Device Type	DI	DO	AI	AO
ASE	TCP	%IX100.0-to-%IX100.7	%DX100.0-to-%QX101.7	%IW100-to-%IW11	%QW100 to %QW107

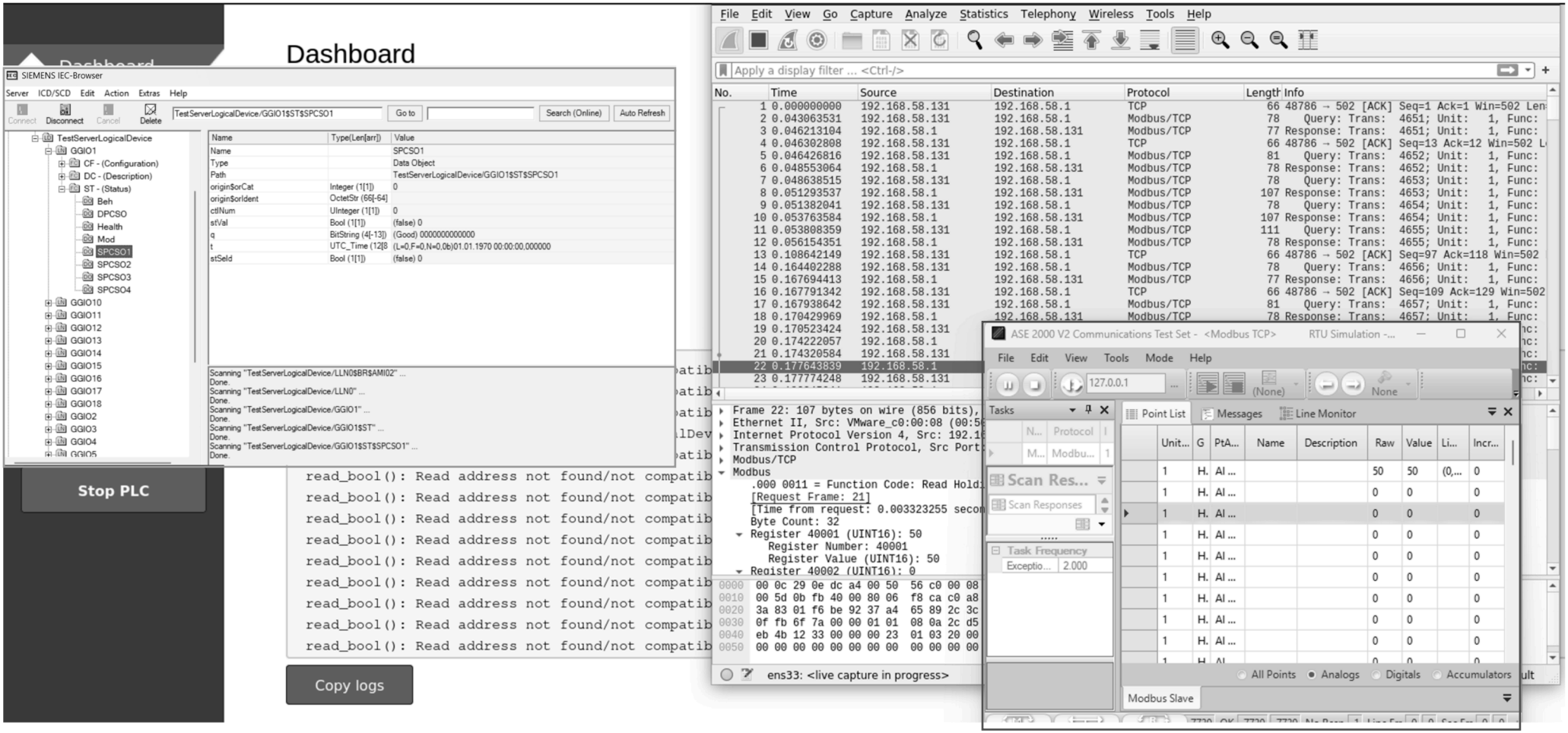


Figure 20. Gateway implementation using Modbus and IEC 61850.

5. Conclusions

The IEC 61499 standard offers a robust foundation for distributed automation architectures; however, its industrial adoption remains limited due to a lack of practical implementations. This work demonstrates the design and validation of a low-cost, standards-based communication gateway for photovoltaic (PV) plants, addressing system integration challenges using open and interoperable tools. By leveraging the 4Diac framework, the developed gateway extends runtime functionality to support key industrial communication protocols, thereby enabling seamless interoperability among heterogeneous systems. Simulation results conducted in a virtualized environment confirm the feasibility and scalability of the proposed solution. Furthermore, the findings indicate that the gateway can be effectively deployed on compact embedded platforms such as Raspberry Pi or Linux-based controllers. The proposed approach provides a cost-effective pathway toward flexible, vendor-independent DER integration and lays the groundwork for future field validation and large-scale deployment of IEC 61499-based automation systems. At present, the 4diac application does not natively support libIEC61850. Based on the evaluation of the available tools in this document, it has been proven that DER plants requiring IEC61850 communication can still achieve the required functionality using low-cost or open-source tools.

Author Contributions: Conceptualization, S.K.; Investigation, Method and Validation, E.L.; Resources, S.K.; Writing—original draft, E.L.; Review, Editing and Supervision, S.K.; Funding acquisition, S.K. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported in part by the National Research Foundation (NRF) under Thuthuka Grant 138177, in part by the Eskom Tertiary Education Support Program (TESP) through a research grant, and in part by the Eskom Power Plant Engineering Institute (EPPEI).

Data Availability Statement: The original contributions presented in this study are included in the article. Further inquiries can be directed to the corresponding authors.

Acknowledgments: The researchers acknowledge the support and assistance of the Center for Intelligent Systems and Emerging Technologies within the Department of Electrical, Electronic, and Computer Engineering at the Cape Peninsula University of Technology, Bellville, Cape Town 7535, South Africa, for their financial and material support in executing this research project. The opinions presented in this paper are those of the authors and not the funders.

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

4Diac	4diac™ IDE is an IEC 61499. Compliant Development Environment
AC	Alternating Current
DER	Distributed Energy Resource
ECC	Execution Control Chart
EPL	Eclipse Public License
FORTE	4Diac runtime
GOOSE	Generic Object-Oriented Substation Event
IC	Incremental conductance
IDE	Integrated Development Environment
IEC 61131-3	IEC61131 is an IEC standard for programmable controllers
IEC 61499	IEC61499 is an IEC Standard that provides a generic model for distributed systems
IEC 62541	An IEC standard that defines the Information Model of the OPC Unified Architecture
IED	Intelligent Electronic Device
IGBT	Insulated-Gate Bipolar Transistor

iLN	Intelligent Logical Node
IR	Integral Regulator
MMS	Manufacture Message Specification
Modbus	Modbus is an open, client-server application protocol for industrial communication.
MPPT	Maximum Power Point Tracking
OPCUA	Open Platform Communications Unified Architecture
PLC	Programmable Logic Controller
PO	Perturb & Observe
PV	Photovoltaic
RBAC	Role-based access control
SCADA	Supervisory Control and Data Acquisition
SEL 751A	Feeder Protection Relay is for industrial and utility feeder protection
SFIB	Service Interface Function Block
SV	Sampled Values
TCP/IP	Transmission Control Protocol/Internet Protocol,
TLS	Transport Layer Security
TR	Technical Report
uint	Unsigned Integer
VM	Virtual Machine
XML	Extensible Markup Language

References

1. Spalding-Fecher, R.; Senatla, M.; Yamba, F.; Lukwesa, B.; Himunzowa, G.; Heaps, C.; Chapman, A.; Mahumane, G.; Tembo, B.; Nyambe, I. Electricity supply and demand scenarios for the Southern African power pool. *Energy Policy* **2017**, *101*, 403–414. [CrossRef]
2. Kumar, R.; Srinivasan, S.; Indumathi, G.; Dzitac, S. A Communication Viewpoints of Distributed Energy Resource. In Proceedings of the International Workshop Soft Computing Applications, Arad, Romania, 24–26 August 2016; pp. 107–117. [CrossRef]
3. *IEC 61131-3:2025*; Programmable Controllers—Part 3: Programming Languages (4th ed.). International Electrotechnical Commission (IEC): Geneva, Switzerland, 2002.
4. Vyatkin, V. IEC 61499 as enabler of distributed and intelligent automation: State-of-the-art review. *IEEE Trans. Ind. Inform.* **2011**, *7*, 768–781. [CrossRef]
5. Lyu, G.; Brennan, R.W. Towards IEC 61499-based distributed intelligent automation: A literature review. *IEEE Trans. Ind. Inform.* **2020**, *17*, 2295–2306. [CrossRef]
6. Vyatkin, V. *IEC 61499 Function Blocks for Embedded and Distributed Control Systems Design*; ISA: Gurugram, India, 2007.
7. Zhu, L.; Shi, D.; Duan, X. Standard function blocks for flexible IED in IEC 61850-based substation automation. *IEEE Trans. Power Deliv.* **2010**, *26*, 1101–1110. [CrossRef]
8. Cavadini, F.A.; Montalbano, G.; Kollegger, G.; Mayer, H.; Vytakin, V. IEC-61499 distributed automation for the next generation of manufacturing systems. In *The Digital Shopfloor-Industrial Automation in the Industry 4.0 Era*; Soldatos, J., Lazaro, O., Cavadini, F., Eds.; River Publishers: Gistrup, Denmark, 2022; pp. 103–127.
9. Christensen, J.H.; Strasser, T.; Valentini, A.; Vyatkin, V.; Zoitl, A. The IEC 61499 function block standard: Overview of the second edition. *ISA Autom. Week* **2012**, *6*, 6–7.
10. Thramboulidis, K. IEC 61499 in factory automation. In *Advances in Computer, Information, and Systems Sciences, and Engineering, Proceedings of the IETA 05, TENE 05 and EIAE 05*; Springer: Berlin/Heidelberg, Germany, 2006; pp. 115–124.
11. Thramboulidis, K. IEC 61499 vs. 61131: A comparison based on misperceptions. *arXiv* **2013**, arXiv:1303.4761. [CrossRef]
12. Campanelli, S.; Foglia, P.; Prete, C.A. Integration of existing IEC 61131-3 systems in an IEC 61499 distributed solution. In Proceedings of the 2012 IEEE 17th International Conference on Emerging Technologies & Factory Automation, Krakow, Poland, 17–21 September 2012; pp. 1–8. [CrossRef]
13. Andren, F.; Strasser, T.; Zoitl, A.; Hegny, I. A reconfigurable communication gateway for distributed embedded control systems. In Proceedings of the IECON 2012-38th Annual Conference on IEEE Industrial Electronics Society, Montreal, QC, Canada, 25–28 October 2012; pp. 3720–3726. [CrossRef]
14. Strasser, T.; Stifter, M.; Andrén, F.; de Castro, D.B.; Hribernik, W. Applying open standards and open source software for smart grid applications: Simulation of distributed intelligent control of power systems. In Proceedings of the 2011 IEEE Power & Energy Society General Meeting, Detroit, MI, USA, 24–28 July 2011; pp. 1–8. [CrossRef]

15. Strasser, T.; Andrén, F.; Vyatkin, V.; Zhabelova, G.; Yang, C.-W. Towards an IEC 61499 compliance profile for smart grids review and analysis of possibilities. In Proceedings of the IECON 2012-38th Annual Conference on IEEE Industrial Electronics Society, Montreal, QC, Canada, 25–28 October 2012; pp. 3750–3757. [CrossRef]
16. Vlad, V.; Popa, C.D.; Turcu, C.O.; Buzduga, C. A solution for applying IEC 61499 function blocks in the development of substation automation systems. *arXiv* **2015**, arXiv:1508.06435. [CrossRef]
17. Cruz, E.M.; Carrillo, L.R.G.; Salazar, L.A.C. Structuring Cyber-Physical Systems for Distributed Control with IEC 61499 Standard. *IEEE Lat. Am. Trans.* **2023**, *21*, 251–259. [CrossRef]
18. Sysel, M. Matlab/Simulink TCP/IP communication. In Proceedings of the 15th WSEAS International Conference on Computers, Corfu Island, Greece, 15–17 July 2011; pp. 71–75.
19. Krishnamurthy, S.; Mohlwini, E.X. Voltage stability index method for optimal placement of capacitor banks in a radial network using real-time digital simulator. In Proceedings of the 2016 International Conference on the Domestic Use of Energy (DUE), Cape Town, South Africa, 30–31 March 2016; pp. 1–8. [CrossRef]
20. MATLAB Help Center. Simscape Electrical. Available online: https://ch.mathworks.com/help/sps/ug/detailed-model-of-a-100kw-grid-connected-pv-array.html (accessed on 31 May 2023).
21. Dighore, V.; Welankiwar, A. Design and Simulation of 100 MW Photovoltaic Power Plant Using Matlab Simulink. 2021. Available online: https://www.researchgate.net/publication/351733228_Design_and_Simulation_of_100_MW_Photovoltaic_Power_Plant_Using_Matlab_Simulink (accessed on 31 May 2023).
22. Zoitl, A.; Strasser, T.; Ebenhofer, G. Developing modular reusable IEC 61499 control applications with 4DIAC. In Proceedings of the 2013 IEEE 11th International Conference on Industrial Informatics (INDIN), Bochum, Germany, 29–31 July 2013; pp. 358–363. [CrossRef]
23. Seilonen, I.; Vyatkin, V.; Atmojo, U.D. OPC UA information model and a wrapper for IEC 61499 runtimes. In Proceedings of the 2019 IEEE 17th International Conference on Industrial Informatics (INDIN), Helsinki, Finland, 22–25 July 2019; pp. 1008–1013. [CrossRef]
24. *IEC 61850-7-420*; Communication Networks and Systems for Power Utility Automation—Part 7-420: Basic Communication Structure—Distributed Energy Resources Logical Nodes. IEC: Geneva, Switzerland, 2009. Available online: https://cdn.standards.iteh.ai/samples/13405/92c4628f823549998e9f9d88013e5b37/IEC-61850-7-420-2009.pdf (accessed on 31 May 2023).
25. *IEC TR 61850-90-11*; Part 90-11: Methodologies for Modelling of Logics for IEC 61850 Based Applications. IEC: Geneva, Switzerland, 2020. Available online: https://webstore.iec.ch/en/publication/32080 (accessed on 31 May 2023).
26. Hara, M.; Kriger, C. Implementation of an IEC 61850-Based Metering Device Using Open-Source Software. *Univers. J. Electr. Electron. Eng.* **2021**, *8*, 17–34. [CrossRef]
27. Roomi, M.M.; Ong, W.S.; Mashima, D.; Hussain, S.S. OpenPLC61850: An IEC 61850 MMS compatible open source PLC for smart grid research. *SoftwareX* **2022**, *17*, 100917. [CrossRef]
28. Alves, T.R.; Buratto, M.; De Souza, F.M.; Rodrigues, T.V. OpenPLC: An open source alternative to automation. In Proceedings of the IEEE Global Humanitarian Technology Conference (GHTC 2014), San Jose, CA, USA, 10–13 October 2014; pp. 585–589.