A graphical user interface platform to support power electronic converters and integrated systems

Benjamin Dean, Michael Starke, Steven Campbell, Radha Sree Krishna Moorthy
Grid Systems Architecture (GSA) Group
Oak Ridge National Laboratory
{deanbr, starkemr, campbellsl, krishnamoorr}@ornl.gov

***Abstract*— Power Electronic Systems have grown in size and complexity over the past years. Therefore, users require interfaces to properly configure, develop, and deploy these systems. This work presents a Graphical User Interface that dynamically adapts to the system and allows the user to control and monitor its operation. This GUI uses networked communication and open-source Python libraries to allow the user to interact with the system at its different hierarchical layers, as well as visualize the interconnection of the full system. This is demonstrated by showing the user interface applied to a megawatt class charging station that consists of various energy resources.**

Keywords—HTML, Power Electronics, User Interface

I. INTRODUCTION

The drive for developing the ideal user experience for interfacing products is significant. Estimates for global market investments in graphical user interfaces (GUI) approximate to $1.596 billion for 2021 [1]. Forecasts suggest that these investments will only grow in the coming years. Well-constructed GUIs are recognized as a valuable commodity as these tools can improve productivity and reduce learning time and operational errors [2].

Developing a high-quality GUI requires many considerations. As presented in [3]-[5], there are many attributes to supporting a GUI: consistency, universality, feedback, closure, error prevention, reversibility, control, automatic adoption, screen scalability, and memory maintenance to name a few. These are critical traits in ensuring that GUIs become a production tool and not a burden.

While GUI development has been significant in many industries, the adoption of GUIs to support wide implementation of power electronic systems (PES) has been limited. GUIs to support single PES modeling and interfacing examples have been presented in [6]-[9]. In [6]-[7], GUIs have been created for modeling and evaluating single power electronic converters in Matlab/Simulink simulation. In [8]-[9] GUIs have been developed to support digital signal processor (DSP) interfacing using National Instruments LabView and software package called *jscomm*.

Wind, solar, microgrid, and industrial plant operational control and data have also observed some developments in GUIs [10]-[17]. In [10] a solar plant GUI has been constructed that uses Modbus to connect to the solar inverters to retrieve data. A web portal and trend graphs are available for basic information on solar irradiance and converter information. A GUI has been constructed as part of a tool for performance comparison of PV plants in [11]. Optimization of PV plant performance in a GUI is presented in [12]. In [13], a GUI has been constructed for displaying wind plants connectivity and operation status, gauge meters, and provides plots on output. In [14], a GUI has been developed to interface a platform for wind energy technology evaluation. In [15], a GUI has been constructed for Microgrid design and performance evaluation in Matlab/Simulink considering power factor and harmonics. In [16], a Supervisory Control and Data Acquisition (SCADA) system is introduced for industrial systems that can tie the different features necessary including fire protection, ventilation, generation, and other systems into a common platform. In [17], a mixed integer linear programming (MILP) output is put into tables within a GUI.

In many of the works presented, GUIs are constructed by the user or developer to support the informational and control needs of a specific domain (resource, plant, etc). This work looks to present a framework of GUIs that are automatically generated based on information content provided by the resources and system models. These created GUIs allow for the operator to both control the PES and integrated resource, as well as control plant level system interactions. This adopts many of the attributes of consistency, universality, and automatic adoption. These GUIs have also been developed to provide real-time data from the devices in the field and manual control opportunities based on user needs. This gives the user an overview of the system's operation, as well as individual system components. In the next section, the GUI architecture to support a PES system of integrated resources is presented followed by an example use case of a system.

II. Architecture to Support Dynamic GUIs

With the growth and increasing complexity of integrated PES, GUIs have become of increasing interest. Observability and controllability represent two critical components required to ensure that the usefulness of a GUI is maximized. Having this capability throughout an architecture to support debugging and over-ride controls is a important to constructing larger power electronic systems. For a system to be observable, components must have the ability to communicate information to other systems. Furthermore, for a system to be controllable, the system must be able to accept communication controls signals from external devices.

An architecture to support communications and controls for PES considering an agent integration framework is presented in [18]. This system has been developed to support features that include integrated safety, plug-and-play capability, unit commissioning and operation, and DER peer-to-peer coordination. Key to these features includes the ability to provide user direct visibility and control in real-time considering the different layers of the architecture. Another objective is to utilize as many open-source libraries as possible to ensure open frameworks. A representation of the architecture including linkages to the various systems for GUIs is shown in Fig. 1.

The architecture is hierarchal with three distinct layers: 1) resource management controller (RMC), the resource integration controller (RIC), and the converter controller (CC). The CC is at the lowest stage in the hierarchal framework. The CC performs the fast computation and control calculations that are necessary for power electronic converter closed loop controls (~1us). The CC pushes data and receives control request from the RIC through user datagram protocol (UDP) based communication. The RIC is responsible for coupling the PES with resources such as energy storage systems or electric vehicle chargers. The coupling is associated with how systems interact in terms of safety and other factors discussed in [18]. Finally, the RMC incorporates all the integrated technologies and performs various optimization strategies to orchestrate behavior from the systems. For the communication a message queuing telemetry transport protocol (MQTT) is used to support plug-and-play integration of the RICs.

In this framework, three different GUI based systems have been constructed to provide user feedback and control: electrical network and central control GUIs that reside in the RMC and a device control GUI that is in the RIC. These systems are described in further detail in the following subsections.

A. Central Controller Graphical User Interface

The central controller GUI has similarly been constructed to be able to adopt different electrical systems and configurations. In this case, information is pulled from a message queuing system and pushed to an application programing interface (API) server in JavaScript Object Notation (JSON). This server utilizes *Flask*, a python library that allows Python 3 code to link and interact with HyperText Markup Language (HTML) code. By launching an internet browser, the user is able to control the various functions of the system. This browser can be either set to display only on the

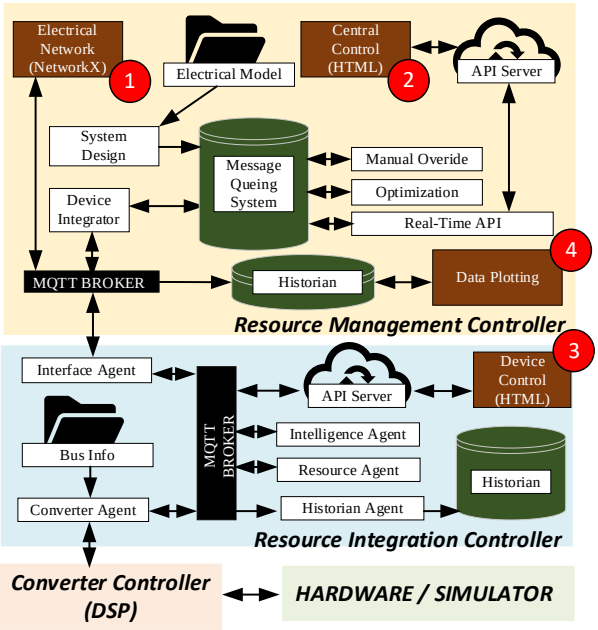

Figure 1. Software and communications framework to support power electronic converters in various system topologies. Integration of graphical user interfaces is also shown for resource management controller an\d resource integration controllers.

computer's internal IP address, *localhost*, or utilize the computer external IP address to support outside device interactions. The use of HTML provides users the ability to utilize lightweight computational platforms such as a phone, or single board computer (SBC) for access. Scalability is also easily adaptable as HTML has automatic adoptability.

The HTML GUI, shown in Fig. 2, is hierarchal in nature to provide universality, consistency, and reversibility. The API data includes device measurements, configuration information including bus interconnections, and status data of the devices. This information is utilized to develop a natural construct for displaying the hierarchy. The initial HTML GUI contains buttons pertaining to the different bus systems upon which power electronic and resource integrated systems are connected. Selection of any bus button results in the HTML portal address that links to the specific bus with the corresponding devices. Selection of the devices creates another web portal that provides a control window of input windows to interject setpoings, buttons for starting and shutting down the corresponding resource, button for clearing any faults or performing an emergency stop, and a button to either close or return to the previous window. Any information entered or buttons pushed result in an API message that is sent to the message queuing system and ultimately converted to a MQTT message for distribution to the devices. Another button has been provided to provide direct access to the RIC HTML which bypasses the central controller MQTT system. This allows the user to interact with the devices in a lower level of the layered communication hierarchy.

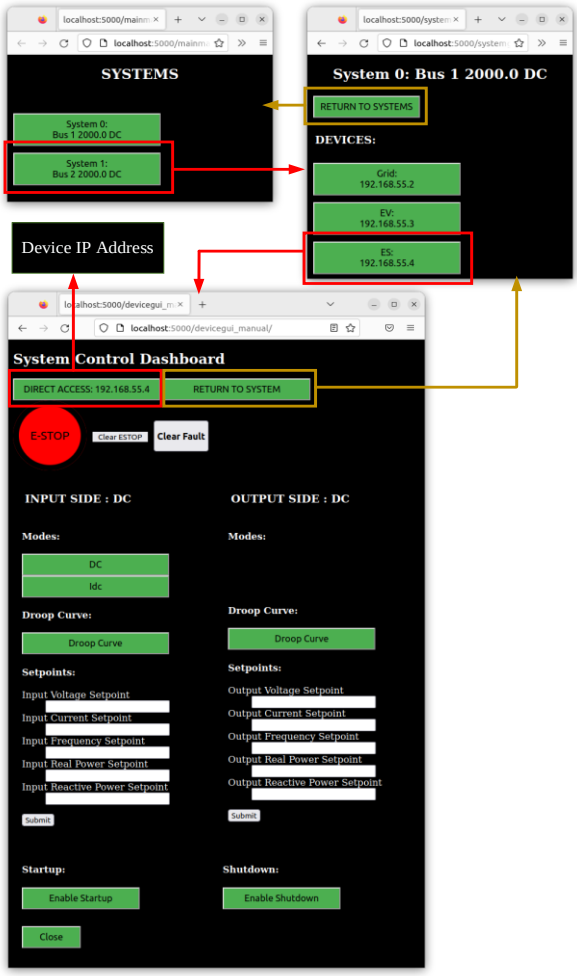

Figure 2. Image of HTML layers and the interconnection on the central controller.

B. *Electrical Networks Graphical User Interface*

The electrical network GUI has been constructed with plug-and-play adaptability considerations while also adopting consistency and universality. This GUI uses python libraries Matplotlib, Tkinter, and NetworkX to automatically generate an electrical representation of the distribution network in real-time. The Matplob and Tkinkter libraries provide the basis for the background GUI to operate. An empty plot provides the template for the NetworkX to structure a set of graphics for the system. The core program can extract the necessary system data by subscripting to the necessary MQTT topics and extracting messages pertaining to the devices (resource + converter) and bus interconnections. The electrical model is provided directly in a text file format while any devices interconnecting the system provide bus interconnection information via the MQTT communications. This information is used to create the necessary linkages between the components. The NetworkX library and functions use this information along with routing information to create the plot with images automatically. As new information is incorporated (messages showing system changes such as outages or new devices added to the system), the NetworkX automatically updates the layout of the system without any user inputs.

Each converter and resource image is interconnected to an IP address that will automatically launch a internet browser to connect to the independent device hosted server HTML. Details on the device HMTL interface is discussed in the next section.

C. *Device Controller Graphical User Interface*

Similar to the central controller GUI, the device controller GUI uses a flask API server to host the HTML interfaces, as shown in Fig. 3. The server and HTML is also hierarchal and interconnects different aspects of a single resource for user viewing and control. Upon the launch of the server, a main HTML browser is shown that provides buttons to different informational components to the system operation: converter specifications, converter real-time dashboard, error codes, fault codes, and system real-time control. Selection of any of these buttons will result in separate HTML browser windows with the respective information displayed and buttons and entries for control.

The converter specification window provides the converter information required for operation. This information is stored in the converter controller and is communicated up to the RIC and then to the RMC. This information includes contactor and pre-charge circuit information, operation limits, such as minimum and maximum voltage, available control modes, such as power control, voltage regulation, and current control. By providing this information, a user can verify the capabilities of the hardware in the system. During system operation, converter configuration information is subject to change, such as in the event of a converter derating. If this occurs, the information is updated, and by refreshing the page, the updated information on the converter specifications can be observed.

The converter real-time dashboard provides the current setpoints and measurements of the device in the system. This offers a visualization of the commands given from either the central controller or a user alongside the measurements provided by the converter. The status of the converter's contactors, precharge circuits, and chosen control mode are also visible. The converter's current state is also included with error and fault codes having dedicated pages. This provides additional information on the nature of the error or fault beyond just a state machine update.

The system real-time control allows the user to control the resource without the presence of the RMC on the network. The server code directly sends commands to the RIC level MQTT broker. Typical system operation in the architecture presented in [18] relies on the RMC to optimize and send setpoints to the RIC and be relayed on to the converter and resource. However, during system debugging and testing, there is an advantageous to operate without the RMC to better control and understand the operation of the resource. This can be done through the system real-time control GUI. A user can clear faults/errors, send startup/shutdown commands, set control modes, and send setpoints to the converter. When done with multiple converters across the system, the system operator can observe different conditions that the RMC would not typically operate the system

at, such as maximum power operation across all system converters, or specific subsystem activation/deactivation.

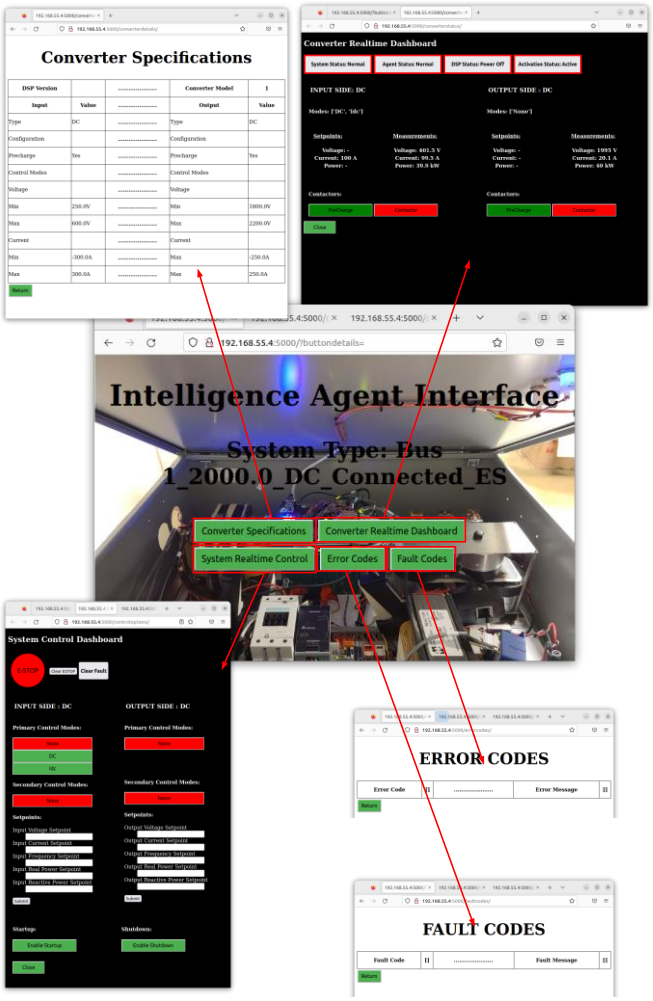

Figure 3. Image of HTML layers and how they interconnect on the device controller and linkage

III. Megawatt Class Charging Implementation Example

For demonstration of the architecture and integrated GUI capability, the controller hardware in the loop system for a megawatt class charging system proposed in [19],[20] is used. This system consists of 2 DC multiports connected to an AC electrical system. Each DC multiport supports multiple DC/DC conversion stages that are used to integrate energy storage and an EV charger. An AC/DC converter composed of cascaded H bridges interfaces each multiport system with the power grid.

For this architecture, two multiports were used to make up the megawatt class charging system. A typical multiport contains up to 3 EV chargers, an energy storage system, and a grid-tied inverter.

For this example, the point of common coupling, where all grid-tied inverter attach to the power grid, is designated as “Bus 0”. The RMC electrical model specifies the voltage level of this bus as 13.8kV. The grid-tied inverter for multiport 1 connects Bus 0 to Bus 1, which is the DC-DC bus for multiport 1. This bus has a nominal value of 2kV. The DC-DC converters that tie the resources to the bus specify Bus 1 as their connection point. The point where the resource ties to the DC-DC converter does not have a bus number specified. For multiport 2, the grid-toed inverter species Bus 0 and Bus 2, and the multiport’s resources also specify Bus 2.

Based on the interconnection provided by the converters, along with the bus configuration in the RMC, the entire system topology can be visualized and displayed, as shown in Fig. 4. The two multiports are shown with one medium-duty EV charging and one energy storage system in each multiport.

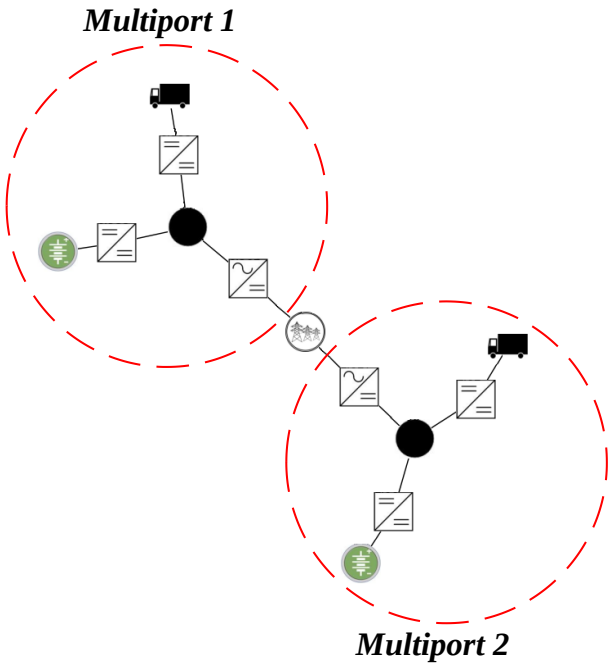

Figure 4. Automatic generation of electrical models using NetworkX python libraries and pre-loaded images for converters, resources (EVs, batteries), and buses.

IV. Conclusions

This work presents a framework for supporting both distributed and centralized user interfaces. This provides users the ability to interact with varying integrated power electronic systems and to perform control and testing. Addressed in this work are qualities required for an effective graphical user interface to be used with a power electronic system, namely scalability, adaptability, and real-time feedback. Also incorporated into this system is a SQLite data historian and data plotting tool, which allows the user to see the data that was logged on the HTML interface.

This work also demonstrates the advantage gained when information about the system is available at different system levels (including the device level). Plug-and-play adoption of the various GUI systems provides a flexible and dynamic capability to auto populate information, with different systems and subsystems populating the HTML interface.

Future work will look at the more advanced implications of augmented reality, virtual reality, voice and gesture type implications proposed in more advanced UI technologies [21]. Further future work will also be including dashboard integration to allow the user to see how measurements, control, and the system status change over time.

REFERENCES

[1] Business Research Insights, User Interface (UI) Design Market Size, Share, Growth, and Industyr Analsyis, By Type (User Experience (UX) Desing, Interactive Design(ID), Visual & Graphical Design and Others), By Application (Software and APP, Web Page, Game, TV Interfaces and Others. Regional Forecast from 2022 to 2028), BRI102500, Dec. 2022. https://www.businessresearchinsights.com/market-reports/user-interface-ui-design-market-102500

[2] R. E. Opaluch and Y. Tsao, "Ten ways to improve usability engineering — Designing user interfaces for ease of use," in AT&T Technical Journal, vol. 72, no. 3, pp. 75-88, May-June 1993.

[3] K. Y. Zamri and N. N. Al Subhi, "10 user interface elements for mobile learning application development," 2015 International Conference on Interactive Mobile Communication Technologies and Learning (IMCL), Thessaloniki, Greece, 2015, pp. 44-50.

[4] B. Shneiderman and C. Plaisant, Designing the user interface: strategis for effective human-computer interaction, 5th ed. USA: Addison Wesley, 2009.

[5] J. Gong and P. Tarasewich, “Guidelines for handheld mobile device interface design,” in Proceedings of DSI 2004 Annual Meeting, 2004, pp. 3751–3756.

[6] R. Višnić, V. Šunde and I. Mrčela, "Matlab/GUI interface for simulation of power electronic converters," 2011 Proceedings of the 34th International Convention MIPRO, Opatija, Croatia, 2011, pp. 136-140.

[7] S. Doolla, S. S. Bhat, T. S. Bhatti and M. Veerachary, "A GUI based simulation of power electronic converters and reactive power compensators using MATLAB/SIMULINK," 2004 International Conference on Power System Technology, 2004. PowerCon 2004., Singapore, 2004, pp. 1710-1715 Vol.2.

[8] A. Marquez, J. I. Leon, L. G. Franquelo and S. Vazquez, "Educational hardware/software interface for power electronic applications," 2012 6th IEEE International Conference on E-Learning in Industrial Electronics (ICELIE), Montreal, QC, Canada, 2012, pp. 47-51.

[9] J. A. Sanchez et al., "Educational software interface for power electronic applications," IEEE EDUCON 2010 Conference, Madrid, Spain, 2010, pp. 1165-1170.

[10] E. Irmak, M. Ersan and O. Guney, "Software and Hardware Implementation of a Graphical User Interface for Solar Power Plants," 2019 1st Global Power, Energy and Communication Conference (GPECOM), Nevsehir, Turkey, 2019, pp. 312-315.

[11] M. S. Jaferi, M. N. Mohd Hussain, N. '. Mohamad Zakaria, M. M. Hussain, K. Daud and S. S. Mat Isa, "Performance Evaluation GUI for 50 MW Large Scale Solar PV System," 2022 11th Electrical Power, Electronics, Communications, Controls and Informatics Seminar (EECCIS), Malang, Indonesia, 2022, pp. 17-21.

[12] G. Adinolfi, R. Ciavarella, V. Palladino, M. Valenti and G. Graditi, "A multi-objective optimization design tool for Smart Converters in photovoltaic applications," 2018 International Symposium on Power Electronics, Electrical Drives, Automation and Motion (SPEEDAM), Amalfi, Italy, 2018, pp. 793-798.

[13] T. Kamel, Y. Biletskiy and L. Chang, "Data communication to monitor power electronic converters," 2015 IEEE 24th International Symposium on Industrial Electronics (ISIE), Buzios, Brazil, 2015, pp. 992-997.

[14] L. Wang, H. Zhao and B. Li, "The design and implementation of graphic user interface for real-time wind farm simulation platform," The 10th Renewable Power Generation Conference (RPG 2021), Online Conference, 2021, pp. 445-450.

[15] R. Shrivastava, J. Kodnani, R. Manikandan and R. R. Singh, "GUI based Power Factor and Harmonics Computation for Microgrid Central Controller," 2021 Innovations in Power and Advanced Computing Technologies (i-PACT), Kuala Lumpur, Malaysia, 2021, pp. 1-6.

[16] I. C. Hoarcă, N. Bizon and F. M. Enescu, "The design of the graphical interface for the SCADA system on an industrial platform," 2020 12th International Conference on Electronics, Computers and Artificial Intelligence (ECAI), Bucharest, Romania, 2020, pp. 1-6.

[17] E. Lázár, D. Petreuș, R. Etz and T. Pătărău, "Minimization of operational cost for an Islanded Microgrid using a real coded Genetic Algorithm and a Mixed Integer linear Programming method," 2017 International Conference on Optimization of Electrical and Electronic Equipment (OPTIM) & 2017 Intl Aegean Conference on Electrical Machines and Power Electronics (ACEMP), Brasov, Romania, 2017, pp. 693-698.

[18] M. Starke et al., "Agent-Based Distributed Energy Resources for Supporting Intelligence at the Grid Edge," in IEEE Journal of Emerging and Selected Topics in Industrial Electronics, vol. 3, no. 1, pp. 69-78, Jan. 2022.

[19] M. Starke et al., "A MW scale charging architecture for supporting extreme fast charging of heavy-duty electric vehicles," 2022 IEEE Transportation Electrification Conference & Expo (ITEC), 2022, pp. 485-490.

[20] R. S. K. Moorthy, M. Starke, B. Dean, A. Adib, S. Campbell and M. Chinthavali, "Megawatt Scale Charging System Architecture," 2022 IEEE Energy Conversion Congress and Exposition (ECCE), 2022, pp. 1-8.

[21] Christopher Holloway, Designing for Tomorrow: Addressing UI Challenges in Emerging Interfaces (With Infographic)