ARL-SR-0477 ● AUG 2023

DEVCOM Army Research Laboratory Visualization and Processing for Embedded Research Systems (ARL-ViPERS) User Manual

by Abigail Snellman, Zachary Drummond, David Hull, Brandon Parks, Kevin Claytor, Hugh Chung, Jeremy Hu, and Alex George

NOTICES

Disclaimers

The findings in this report are not to be construed as an official Department of the Army position unless so designated by other authorized documents.

Citation of manufacturer's or trade names does not constitute an official endorsement or approval of the use thereof.

Destroy this report when it is no longer needed. Do not return it to the originator.

ARL-SR-0477 ● AUG 2023

DEVCOM Army Research Laboratory Visualization and Processing for Embedded Research Systems (ARL-ViPERS) User Manual

Abigail Snellman, David Hull, Brandon Parks, and Kevin Claytor
DEVCOM Army Research Laboratory

Zachary Drummond, Hugh Chung, and Alex George
General Technical Services

Jeremy Hu
Fibertek, Inc

REPORT DOCUMENTATION PAGE

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED	
		START DATE	END DATE
August 2023	Special Report	Oct 2019	May 2023

4. TITLE AND SUBTITLE
DEVCOM Army Research Laboratory Visualization and Processing for Embedded Research Systems (ARL-ViPERS) User Manual

5a. CONTRACT NUMBER	5b. GRANT NUMBER	5c. PROGRAM ELEMENT NUMBER
5d. PROJECT NUMBER	**5e. TASK NUMBER**	**5f. WORK UNIT NUMBER**

6. AUTHOR(S)
Abigail Snellman, Zachary Drummond, David Hull, Brandon Parks, Kevin Claytor, Hugh Chung, Jeremy Hu, and Alex George

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
DEVCOM Army Research Laboratory
ATTN: FCDD-RLA-LB
2800 Powder Mill Road
Adelphi, MD 20783-1197

8. PERFORMING ORGANIZATION REPORT NUMBER
ARL-SR-0477

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
DISTRIBUTION STATEMENT A. Approved for public release: distribution unlimited.

13. SUPPLEMENTARY NOTES
ORCID ID: David Hall, 0000-0002-4200-7636

14. ABSTRACT
The US Army Combat Capabilities Development Command (DEVCOM) Army Research Laboratory (ARL) has developed electric-power measurement and analytics tools using high-resolution sensors, lab instrumentation, and software technology. To support the use of these sensors, an extensible suite of software modules has been created that require only a web browser for the user interface. The ARL-developed software framework and modules for "Visualizing and Processing Embedded Research Systems" is called ARL-ViPERS. This sensor-based software provides a method for configuring the sensors and interacting with and visualizing the data they produce without the need to install any software on end-user devices.

15. SUBJECT TERMS
Electromagnetic Spectrum Sciences, embedded processing, Internet of Things, IoT, live data visualization, device management, embedded web applications

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES
a. REPORT	b. ABSTRACT	C. THIS PAGE		
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UU	90

19a. NAME OF RESPONSIBLE PERSON	19b. PHONE NUMBER *(Include area code)*
Abigail Snellman	(301) 394-0425

STANDARD FORM 298 (REV. 5/2020)
Prescribed by ANSI Std. Z39.18

Contents

List of Figures

List of Tables

1. Introduction

Several prototype US Army Combat Capabilities Development Command (DEVCOM) Army Research Laboratory (ARL) sensor systems have been built on common, modular data acquisition, storage, processing, and communications hardware known as ARL's Autonomous Real-Time Electric-power Measurement and Instrumentation System (ARL-ARTEMIS). Two examples of these systems are ARL's Mobile Unattended Ground Sensor (ARL-MUGS) and Mobile Power Meter (ARL-MPM) (Fig. 1). These systems come equipped with software that can be used for sensor configuration as well as live and postprocessing analysis of data collected on electric-power (EP) systems. This ARL-developed software framework for "Visualizing and Processing on Embedded Research Systems" is called ARL-ViPERS. Hereafter, ARL-ARTEMIS and ARL-ViPERS will be referred to as ARTEMIS and ViPERS, respectively.

Fig. 1 ARL-MUGS (left) and ARL-MPM (right)

ViPERS includes both an embedded web application that can be accessed through a web browser on a user's device such as a mobile phone, tablet, or personal computer; as well as the Dataserver application, which is used to run custom processing code. Both the web application and the Dataserver run on the sensor and work together to provide user interfaces (UIs) for easy configuration of sensors as well as several data visualization tools that users may find useful for "at-the-edge" data analysis. The Dataserver's primary responsibility is to manage ongoing data processing tasks in the background, while the web server is used to provide users with the corresponding UIs. The Dataserver can be thought of as the "brain" of ViPERS whereas the web server is the "face."

All required ViPERS software runs on the ARL sensor hardware; therefore, users do not need to install any software on the user devices. ViPERS was also built with modularity in mind. It includes several base modules built for EP analysis and allows users to easily extend the software to include their own modules. Users can

also upload custom processing code and visualizations that will run in real time on the sensor; see Section 3.18 for details.

This ViPERS User Guide provides a step-by step walkthrough of the features made available to users through the web application. Section 2 provides basic information needed to connect to and use ViPERS. Section 3 covers detailed information on each module included with the ViPERS web application; and Section 4 provides information on the ViPERS Dataserver. See the companion ViPERS Implementation Guide[1] and the ViPERS Programming Manual[2] for instructions on adding new modules and available application programming interfaces (APIs).

2. ViPERS Basics

The ViPERS software framework includes the following:

1) An embedded web server that provides a UI for interaction with the sensor;
2) The Dataserver application used to run processing modules in the background on the sensor; and
3) An embedded real-time, time-series database for long-term data storage.

Users can use this section as a basic "quick-start" guide for ViPERS. This section explains how to access the ViPERS web application in the default networking configuration, navigate the application, and basic troubleshooting. Much of this information will be covered in further detail in later sections.

2.1 Connecting to ViPERS

The ViPERS web application runs an embedded web server on ARTEMIS sensors and provides several capabilities including specialized tools related to EP sensing. In the standard configuration, users can connect an ARTEMIS-based sensor to an end-user device such as a laptop, tablet, or mobile phone via an Ethernet cable. Users can then type the IP address "***10.10.0.1***" into a browser's address bar. (Note: This configuration requires the user's device to be configured for dynamic host configuration protocol [DHCP]; however, most devices use DHCP by default.) When connected to the ViPERS web server, users will be brought to the login page (Fig. 2).

Fig. 2 ViPERS login screen

On the login page, users will be prompted for a username and password. After logging in users will be directed to the home page (discussed in Section 2.2). ViPERS can also be accessed through other IP addresses (e.g., ***192.168.1.XX***) if both the sensor and the user's device are connected to a local network and the MPM/MUGS is not set up as an access point. Different networking configuration options are discussed later in Section 3.1.2.

2.2 Navigation: Home Page and Navigation Bar

The ViPERS home page contains icons for each available ViPERS module and serves as the main navigation page for the application. Clicking on any icon brings the user to the corresponding page for that module. Some of these pages are designed to display system information and system configuration, while others are focused on data display, interaction, and download. Note: The specific applications and layout included may be different, but the primary setup pages (shown in Fig. 3) are included in all units. Section 3 shows a page-by-page overview of the ViPERS modules.

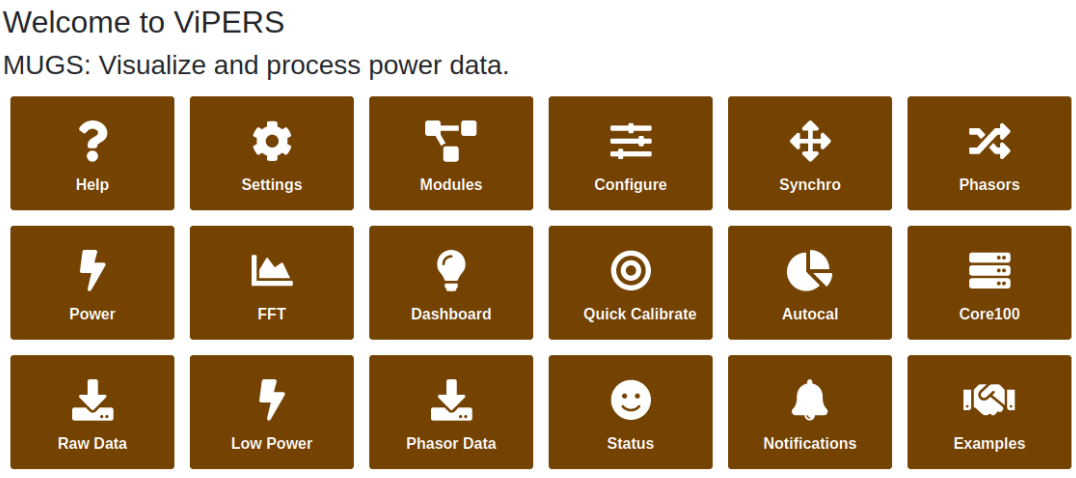


Fig. 3 ViPERS home page with links to all primary pages

The navigation bar is an important tool for users to be familiar with. The navigation bar provides users with vital system information at a quick glance and is visible at the top of every ViPERS page. At the top left of the navigation bar, there is a

DEVCOM ViPERS logo. Immediately to the right of that is the system name or "host name" (Fig. 4). By default, this is set based on the serial number (SN) of the device. For example, if the SN of an MPM is SN3007, the host name on display at the top left will be MPM-3007.

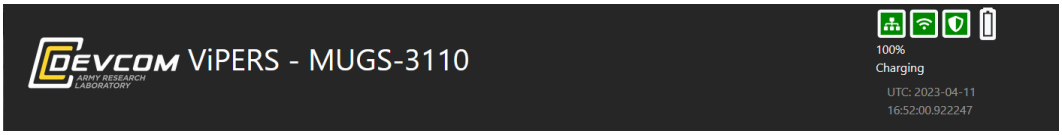


Fig. 4 ViPERS navigation bar shown at the top of all pages

Users can navigate back to the home page from any other ViPERS page by clicking either the host name or the DEVCOM ViPERS Logo at the top left of the navigation pane. To the right of the navigation bar are several icons that detail the status of the ARTEMIS MUGS/MPM battery and its network status. These icons show (Fig. 5, left to right) the unit's connectivity statuses for local area network (LAN), wide area network (WAN), virtual private network (VPN), and battery level, respectively.

Fig. 5 ViPERS navigation bar icons for LAN, WAN, and VPN connections

LAN connection is the device's connectivity to a router or direct connection with a device, WAN connection is the device's connectivity to the outside Internet, and VPN connection is the device's connectivity to the ARLPower VPN. If the corresponding icon is green, this indicates connection to the corresponding network; if it is red, this indicates that the device is not connected to the corresponding network. The battery icon displays an approximate percent of battery charge. The exact battery charge percentage as well as its charging status is displayed next to or below the battery icon. The device's current time (in coordinated universal time) is also located in the top right corner. Note: For some ARTEMIS versions, the device's time setting may be unreliable on start-up without connection to a WAN and therefore must be reconfigured manually when setting up the device. If a unit is connected to the Internet, the device's time will be configured automatically. See Section 3.1.1 for details.

2.3 Architecture and Basic Troubleshooting

The ViPERS web application works by interfacing to a "Dataserver"—an application on the ARTEMIS sensor system that runs both processing code (called

"Dataserver modules") and visualizations ("plots"). If anything is not working, it is likely because the module or plot is disabled in the Dataserver. On many plot pages there are hints to ensure the correct modules and plots are enabled. When in doubt, check the ViPERS pages: "/dataserver/modules" (Section 3.18.1) and "/dataserver/plots" (Section 3.18.2). ViPERS also interfaces with other Linux utilities for managing required sensor functions (e.g., the network state). Further information about the modules and plots that must be enabled for the default sensor configuration to work properly is available in Section 3.18.

3. Detailed Page-by-Page Guide

Figure 6 provides an overview of each of the icons on the ViPERS homepage. Each topic is numbered according to its corresponding section in this report (i.e., the numerals in brackets [ ] after section headings).

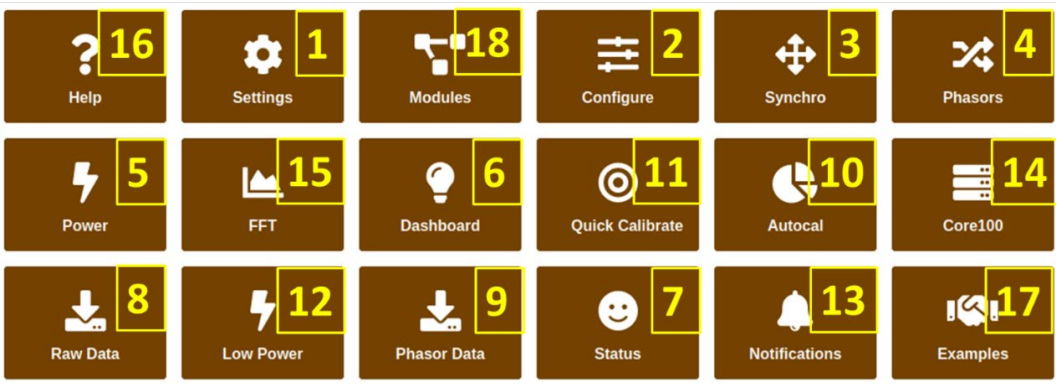


Fig. 6 Use the home page to select different UIs during the setup process

3.1 Settings [1]

The settings page is the main page used to perform system-level configurations. There are several subpages that allow users to check system information and set important parameters such as the system date and type and network operating mode. This page is separate from the Configure [2] page, which is only concerned with data-level configurations such as setting sampling parameters and creating circuits.

3.1.1 Date and Time

This page is used to set the system date and time. This step is crucial to the initial setup of an ARTEMIS device in a GPS- or network-denied area of operation. All versions of the ARTEMIS hardware before the MPMv4 (manufactured in 2023) do not have a real-time clock (RTC); therefore, they are unable to track time when the devices are not powered. In environments with external network access or GPS, the current time will be obtained through the GPS first, if available, and if not available,

time can be updated manually through the network time protocol (Fig. 7). If the current device time is too far in the past (e.g., Linux defaults to the Unix epoch, ~1970), the user may be unable to connect to the web application through the browser because of the required handshake that takes place between the browser and the device's web server. In this case, it is recommended to instead use the secure shell protocol to connect to the sensors and set the time from the Linux command prompt. Currently, two other options for setting the date and time exist if users cannot do so through the ViPERS web page:

1) A new default date can be set in "*/etc/rc.local*".
2) A fake hardware clock can be used to restore the time to the last known time. This method is implemented by ViPERS; however, it will fall behind if the system reboots.

Fig. 7 System date and time can be updated manually if external network access or GPS are unavailable

The newest versions of ARTEMIS hardware (starting with MPMv4) will have an on-board RTC, and this feature may be removed in future iterations of ViPERS as a result.

3.1.2 Network

The next settings subpage is the network page, which allows the user to enable and disable Wi-Fi and Ethernet connection modes (Fig. 8); users must enable Wi-Fi to connect to Wi-Fi networks. It is not recommended to DISABLE or use DHCP-CLIENT MODE on both adapters. Given the topology of most networks, with the router at 192.168.1.1 there will be collisions, and this may make it impossible to connect the device via its IP address.

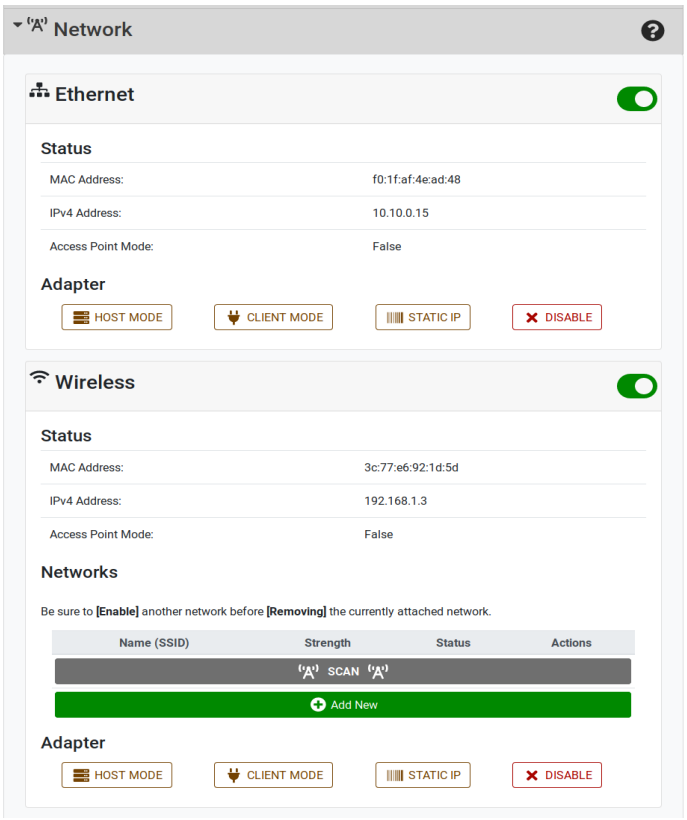


Fig. 8 The networking page allows configuration of both the Ethernet and Wi-Fi interfaces, in addition to connection to Wi-Fi networks

3.1.2.1 Ethernet

The Ethernet section of the page shows the status of the wired network (MAC address, current IPv4 address).

When in DHCP-HOST MODE, users can reach ViPERS by connecting to the Ethernet port directly and navigating to 10.10.0.1 in the browser's address bar. The four buttons at the bottom allow users to toggle between different adapter states (Table 1).

Table 1 Ethernet interface configuration buttons and their actions

Button	Effect
DHCP-HOST MODE	The interface will become a router and hand out DHCP addresses.
DHCP-CLIENT MODE	The interface will request a DHCP address from the network router.
STATIC IP	Set a static IP address and netmask for the network.
DISABLED	Disable the adapter (fully turns off the adapter).

3.1.2.2 Wireless

The wireless section of the page shows the status of the wireless network (MAC address, current IPv4 address).

When in DHCP-HOST MODE, users can reach ViPERS by connecting to the network "ARTEMIS" and navigating to 11.11.0.1 in the browser's address bar.

When in DHCP-CLIENT MODE, wireless networks are scanned and displayed in the table under Networks. Users can add a scanned network by clicking on the "*Add*" button next to its name. Once a network is added, the following additional options become available:

- **[SELECT]** will make that network the active network and **[DISABLE]** all others.
- **[ENABLE]** will allow the device to connect to that network should the currently **[SELECTED]** network become unavailable.
- **[DISABLE]** will prevent the device from connecting to that network.
- **[REMOVE]** will cause the device to forget about that wireless access point.

Additionally, the "*Add New*" button allows users to fully specify a network that was not scanned. This is useful for hidden service set identifiers, in which case the "*scan_ssid*" box must also be checked. The four buttons at the bottom allow users to toggle between different adapter states; see Table 2.

Table 2 Wi-Fi interface configuration buttons and their actions

Button	Effect
DHCP-HOST MODE	The interface will become a router and hand out DHCP addresses.
DHCP-CLIENT MODE	The interface will request a DHCP address from the network router.
STATIC IP	Set a static IP address and netmask for the network.
DISABLED	Disable the adapter (fully turns off the adapter).

3.1.3 Storage

Users can view disk usage information from the storage settings page (Fig. 9). This page displays disk usage information for all mounted data partitions on the system's secure digital (SD) card. In the default configuration of ARTEMIS, there are three mounted partitions: the root file system ("/"), the boot partition ("/boot") and the on-card data partition ("/data"). The root file system refers to all files on the Linux OS, which also includes storage of the PostgreSQL database. The boot partition contains information necessary for ARTEMIS to read on start-up and should not be edited. Finally, the /data partition contains phasor data files that have been streamed from the Zynq field-programmable gate array (FPGA) onto the main system's SD

card. This data is stored in a separate partition to avoid collisions with the OS file system and to ensure safe data storage.

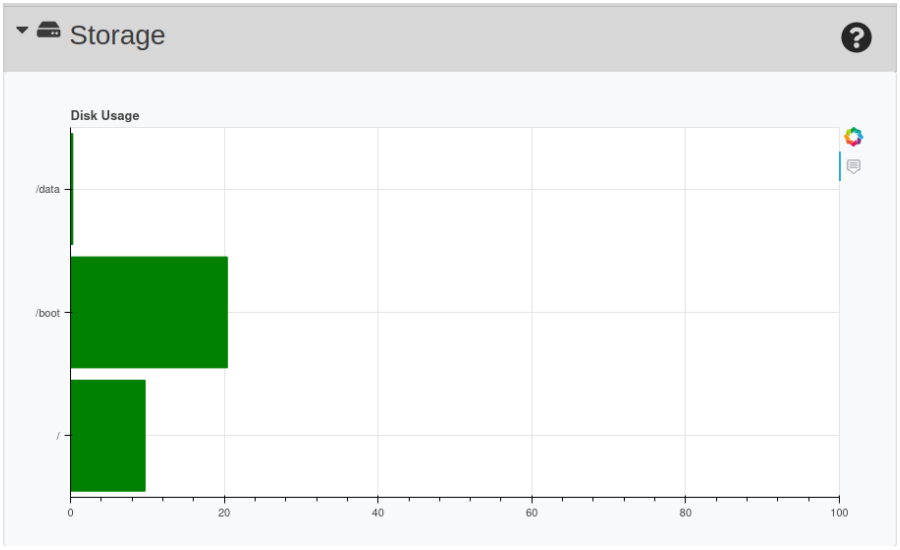
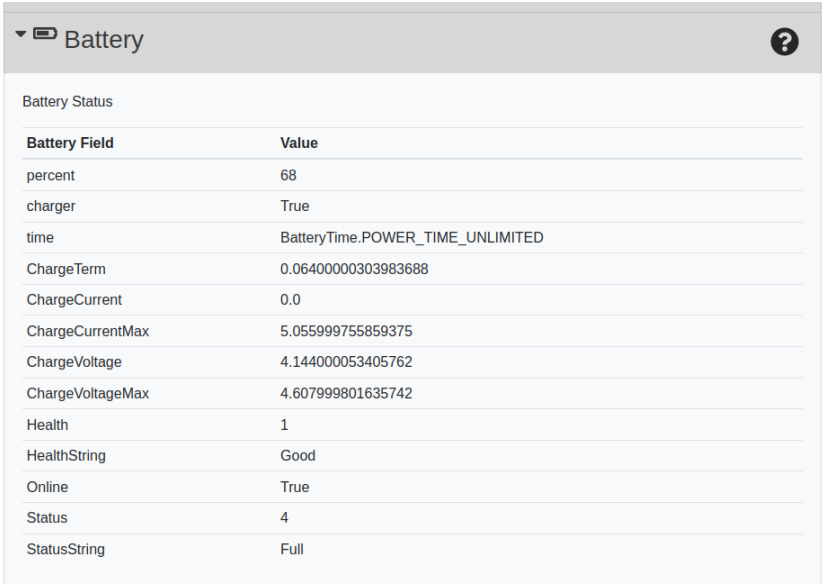


Fig. 9 Disk usage data as presented on the storage settings page

3.1.4 Battery

The battery page details several battery statistics so that users are able to assess the current battery health of their system (Fig. 10). This page includes data to approximate the battery percentage, charging status, estimated time until battery depletion in minutes, available charging statistics (e.g., charging voltage and current), and current battery health and status. For a more in-depth look at battery charge over time see Section 3.6.



Battery Field	Value
percent	68
charger	True
time	BatteryTime.POWER_TIME_UNLIMITED
ChargeTerm	0.06400000303983688
ChargeCurrent	0.0
ChargeCurrentMax	5.055999755859375
ChargeVoltage	4.144000053405762
ChargeVoltageMax	4.607999801635742
Health	1
HealthString	Good
Online	True
Status	4
StatusString	Full

Fig. 10 Battery page displays key metrics to help users assess battery health

3.1.5 System Info

The system information page is broken down into several sections that detail general system information, system version, CPU usage, memory and disk usage, and details for the top 10 running processes (Fig. 11). All resource utilization data presented from this page is instantaneous resource utilization shown from when the request was made. Users can either refresh this page for a new capture of resource utilization or visit the "Live System Performance" ViPERS dashboard page for a real-time visualization of system resources. See Section 3.6 for more information on the dashboard.

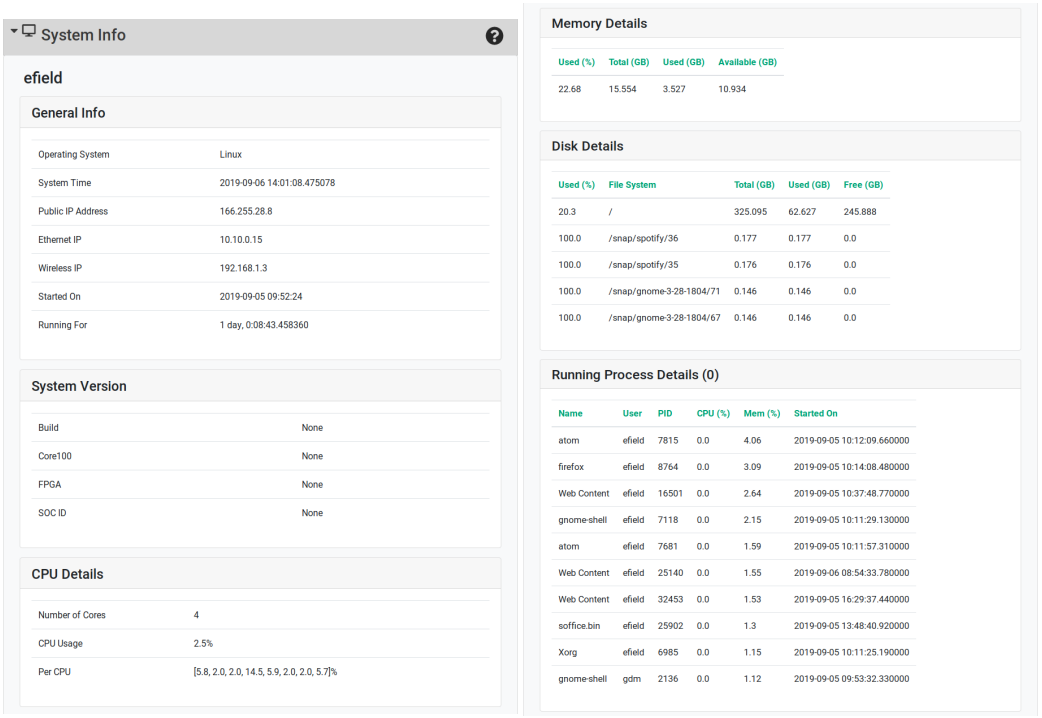


Fig. 11 The system info page is a snapshot of system status

3.1.6 Users

Administrative users can add/delete users from this page (Fig. 12). Users can be created by inputting a new username and password to the device and clicking the "Add" button. Users can be given administrative privileges when they are created by clicking the "Administrator" button.

Note: The last administrative user cannot be removed.

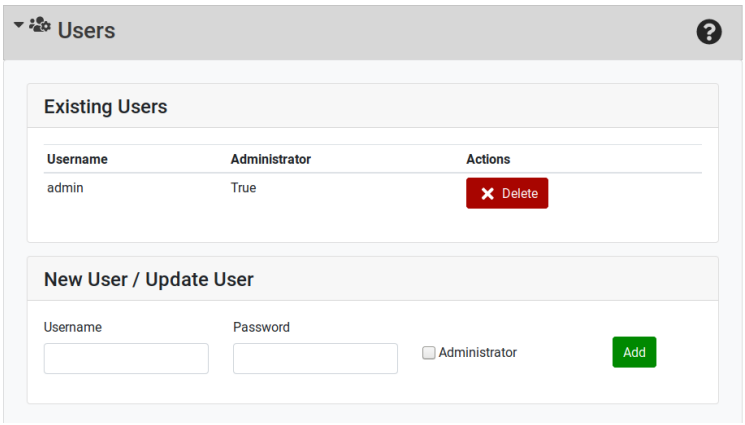


Fig. 12 Users page to add/remove additional administrative or regular users

3.2 Configure [2]

The Configure [2] page allows users to edit the data processing configuration of their system to include configuring the hardware, labeling channels, setting scale factors, and creating circuits. The following subsections provide instructions on using the Configure page to perform these various types of configurations.

3.2.1 Save and Restore

From the save and restore page (i.e., the first tab on the Configure page), users can save, load, upload, download, and delete saved configuration files. These saved configuration files pertain to the user configuration for the Dataserver power processing modules. Saved files contain the settings for the following:

- Hardware data (e.g., transport control protocol [TCP], dbus, simulator source)
- Channel data (e.g., channel names and scale factors)
- Circuit data (e.g., which channels are connected and wiring)

This information is stored by writing data to the **phasors.conf** file. To save the current configuration simply type a name in the "*Save Current Config*" box and select "*Save*". This will create a new "*.conf*" file under conf/saved/ with the chosen name. When any binary phasor data file is retrieved from the ARTEMIS unit, the **phasors.conf** file is required to retrieve information about the data such as the sampling rate and channel names.

All currently saved configurations are listed under the "*Saved Configs*" heading. Users can select one of these to make it the active configuration by clicking on "*Activate*". Clicking the saved configuration's "*Filename*" will download the file through the user's browser. Finally, the saved configuration can be deleted by

clicking "*Delete*" next to the file name. Users can also upload a configuration from their computer by clicking on the "*Choose File*" button, selecting the configuration, and then clicking "*Upload*" (Fig. 13).

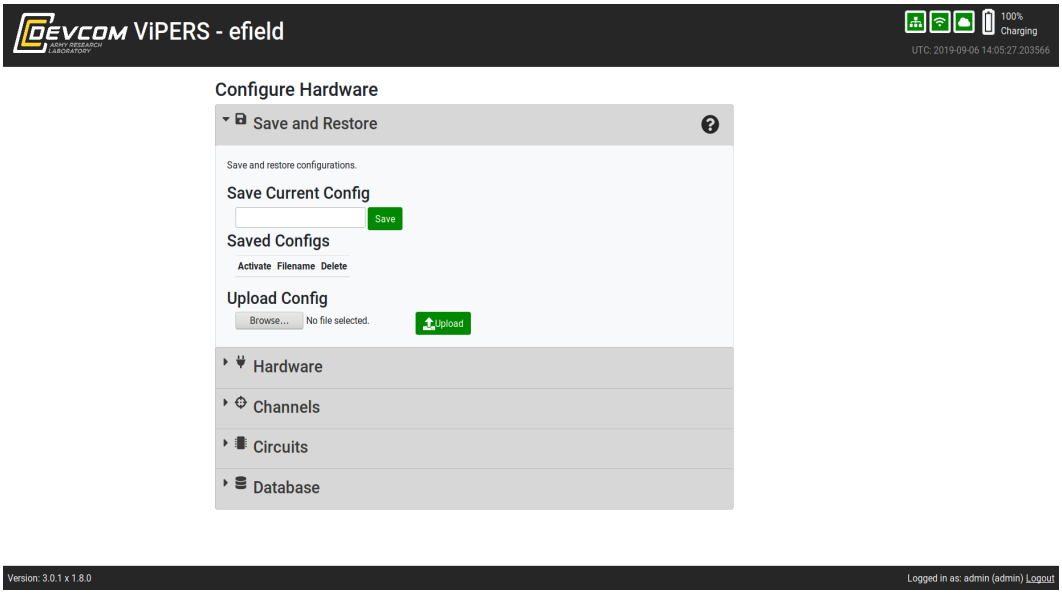


Fig. 13 Configurations for hardware, channels, and circuits can be saved and then downloaded/uploaded/restored

3.2.2 Hardware (Connect Sensors and Source Data)

The hardware page allows users to edit the hardware configuration for the fields, which are described in Table 3 and in further detail in their respective subsections.

Table 3 Form fields and their descriptions

Field	Used by	Definition
source	*synchro.py*	Selects the source of phasors for subsequent processing.
broadcast	*synchro.py, relative.py*	Enables broadcast of phasors on the LAN (via 0.0.0.0) from *synchro.py* and *relative.py*. If disabled, phasors are only broadcast to local host.
Log raw data	*hardware.py*	If checked, *hardware.py* will issue the ARLTX command to log raw data to the corresponding SD card.
bank-# (older ARTEMIS versions)	*hardware.py, relative.py*	Specify hardware BNC attenuation for a given bank of four signals. Also used by *relative.py* to determine scaling factors.
Sample rate, binimation, decimation	FPGA	Sets the appropriate sampling and decimation rates to achieve desired raw data and phasor data rates.
Frequencies	FPGA	Nominal frequencies for phasors produced by the FPGA phasor processing algorithms; default is 60, 180, and 300 Hz (first, third, and fifth harmonics, respectively).
ARLTX on start-up	*hardware.py*	This ARLTX command is issued by *hardware.py* on start-up. (This deprecated feature is still included to support legacy ARTEMIS configurations.)

3.2.2.1 Source

The source field allows users to select where to pull data from for subsequent processing. ViPERS provides the following source options to users:

- **None**: No data source
- **dbus**: Use the Core100 application dbus subscription
- **tcp**: TCP stream of phasors
- **Simulator**: Simulated phasors from a three-phase voltage source (VA, IA, IB, IC) and a cable sensor (H1, H2, H3, E1)
- **simulator-3phase**: Simulated phasors from a three-phase voltage (+ neutral) and three-phase current (+neutral) source

See Section 5.6 for further information about where this data is used in the ARTEMIS data processing chain.

3.2.2.2 Logging Raw Data and Broadcasting

Users have the option to log raw data to two internal SD cards on the ARTEMIS baseboard. By default, ARTEMIS logs raw data to these SD cards in a circular buffer where the oldest files are overwritten first; however, this behavior can be further configured from the raw data page covered in Section 3.8. The options provided on this page allow users to turn on and off raw data storage to each of the

SD cards. This configuration persists when ARTEMIS reboots. Broadcasting allows users to broadcast the binary streams for raw and phasor data on ports 2346 and 4444, respectively, so that they can be accessed from external devices and not just internally by the ARTEMIS unit.

3.2.2.3 Bank and Channel Attenuation

The BNC inputs (MUGS) accept signals in the ±1-V range when the BNC1:1 attenuator is selected, or in the ±10-V range when the BNC10:1 attenuator is selected. In ViPERS, BNC attenuation is configured by a bank, with a bank being a group of four channels. MUGS-8 units and MPMs have two banks (e.g., Bank 1 is made up of Channels 1–4 and Bank 2 is made up of Channels 5–8), whereas MUGS-16 units have four banks. Additionally, LEMO* inputs can be selected in each bank of channels for smart sensor use. **Note:** When LEMO connection is selected for a particular bank on a MUGS unit, the corresponding four BNC ports will **not** be used by the ARTEMIS, and instead inputs for that bank will be taken from the LEMO connectors on the back of the MUGS unit. The MPM connections should **always** be set to LEMO. It is left up to the ARTEMIS MUGS user to determine whether BNC1:1 versus BNC10:1 attenuation or LEMO connection is required for their sensors.

3.2.2.4 Sampling Rates, Binimation, and Decimation

ARTEMIS units can be configured to produce both raw and phasor data at different rates. Higher data rates will yield larger bandwidth and/or potentially more detail on short time-scale phenomena. However, increased data rates have an impact on system resources such as CPU, RAM, and disk storage. Most of the long-term stability testing performed by ARL has been conducted with the sampling frequency set to 8000 Hz and the binimation factor set to eight, which produces a 15.625-Hz phasor rate. Higher data rates may result in less long-term reliability, especially with 16-channel hardware variants. This is an active issue that ARL is working to address in newer hardware and firmware releases.

These page options will set the frequencies, binimation, and decimation rates for the signal processing taking place on the ARTEMIS' FPGA. The frequency parameter controls the raw sampling rate and may be set to anything at the user's discretion. The binimation factor may only take a power of two (2, 4, 8, 16, 32 . . .), and the decimation is a combination of two values. Decimation "*a*" may only be 8 or 10, whereas decimation "*b*" may be 0, 8, or 10. All three factors govern the output sample rate according to Eq. 1:

* LEMO is a registered trademark of the LEMO Group.

$$fs = \begin{cases} \frac{\text{sample\_rate}}{b \times da \times db}, & \text{if } db \neq 0 \\ \frac{\text{sample\_rate}}{b \times da}, & \text{otherwise.} \end{cases} \tag{1}$$

where b is the binimation and da, db are the decimation a and b factors, respectively. Further, the binimation sample determines the frequency cutoff,

$$F_{nyquist} = \frac{\text{sample\_rate}}{2 * b}. \tag{2}$$

Some example raw sampling rates combined with different binimation and decimation factors and their resultant output frequencies and Nyquist rates are shown in Table 4. The first row of Table 4 is the default ARTEMIS configuration.

Table 4 Phasor rates for specific filter settings and analog-to-digital converter (ADC) rates

Raw sampling frequency (Hz)	Binimation	Decimation a:b	Output frequency (Hz)	Nyquist (Hz)
8000	8	8:8	15.625	500
8000	2	8:8	62.5	2000
8000	4	10:10	20.00	1000
8000	8	10:8	12.50	500
16000	8	8:8	31.25	1000
16000	8	10:10	20.00	1000
16000	16	8:8	15.625	500

Additionally, because the decimation factors determine the bandpass filter-roll off, they affect how sharply the frequency band attenuates. In situations where excess noise or clutter signals are anticipated, larger decimation factors may be desired to reduce interference. Users are able to change the sample, binimation, and decimation rates from the ViPERS UI (Fig. 14).

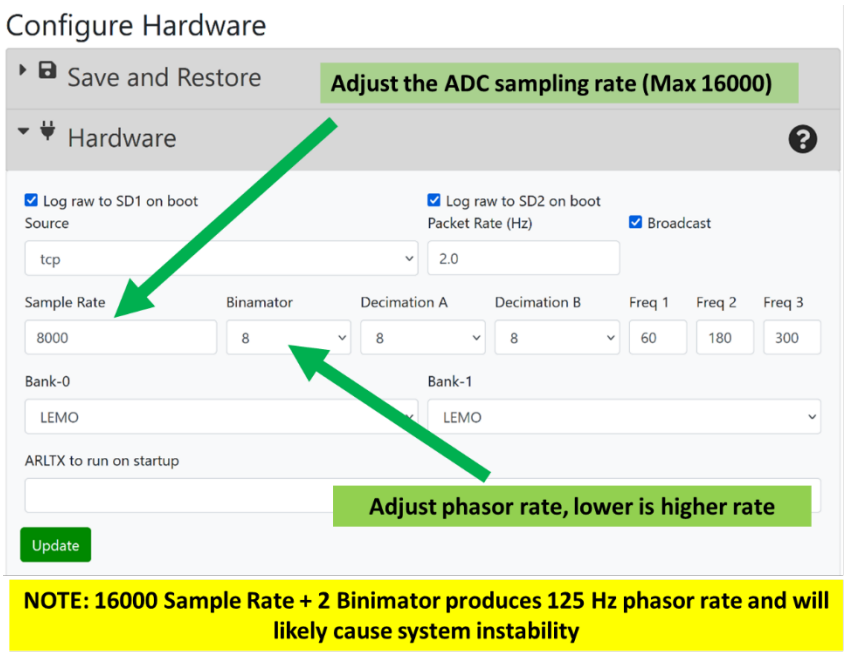


Fig. 14 Phasor rate can be adjusted on the hardware page. Ensure that the TCP source is selected when increasing the phasor rate beyond the default setting of 15.625 Hz.

If the phasor rate increased from 15.625 Hz, the TCP source **MUST BE SELECTED**. The dbus source transfers data too slowly for use at higher speeds and will cause system instability. In addition, **DO NOT SELECT** a phasor rate of 16000 when using a binimation rate of 2. This will produce a phasor rate that will compromise software that delivers data to key components of the system**.**

Increased phasor rates generally require more storage to achieve the same duration of data storage as slower data rates. For example, in Table 5, a 15.625-Hz phasor data rate will produce approximately 335 MB of binary data per day leading to approximately 690 days of storage on a 256-GB SD card. However, at a higher phasor rate of 62.5 Hz, this storage capacity is much less as 1340 MB of data will be produced daily, leading to only approximately 170 days of data storage on a 256-GB SD card. It is therefore encouraged that users desiring a higher data rate either a) more frequently download data files from their ARTEMIS units or b) upgrade their system with a larger SD card.

Table 5 Estimated storage capacity for given data rates

Phasor rate (Hz)	Primary (OS) storage	Data rate (MB/day)	Duration (230 GB)	Sample rate	Binimation
15.625	256 GB	335	~690 days	8000	8
62.5	256 GB	1340	~170 days	8000 (16000)	2 (4)

3.2.2.5 Frequencies

The frequency parameters are used by the ARTEMIS FPGA to calculate phasors for each of the provided frequencies. The defaults are 60, 180, and 300 Hz, the first, third, and fifth harmonics for a 60-Hz power system, respectively, but these values can be set to whatever is desired for the user's application.

3.2.3 Channels (Names, Scaling, and Sensitivities)

Once sensors are connected and the ARTEMIS is receiving data, the input signals need to be transformed to physically useful units. The channels page allows users to configure each input channel by providing a channel name, specifying its input type, and providing sensor-specific impedance, sensitivity, and scale factors (Fig. 15). These fields update *phasors.conf*, which is used by several ViPERS modules for configuration on start-up. Table 6 lists channel configuration options and their descriptions. Further descriptions of each form field and its options are described in their corresponding subsections.

Table 6 Channel configuration form fields and their descriptions

Field	Definition
name	A user-specified name.[a]
type	One of "voltage," "current," "efield," "hfield," "undefined." Only voltage, efield, and undefined can be used as phase references for Circuits (see Section 3.2.4).
impedance	Specifies whether the attached sensors are low-impedance (50-Ω) or high-impedance (1M-Ω) sensors.
sensitivity	The sensitivity of the physical sensor (usually in mV/V or mA/V).[b]
units	The units for the sensitivity value.
multiplier	An arbitrary additional multiplicative factor.[c]

[a] Duplicate names may cause data to be overwritten in the database and negatively impact performance.
[b] The raw values are divided by sensitivity.
[c] This is useful when using current transformers/potential transformers to step up/down the currents/voltages. It is also useful if a sensor is installed backward (use a negative factor).

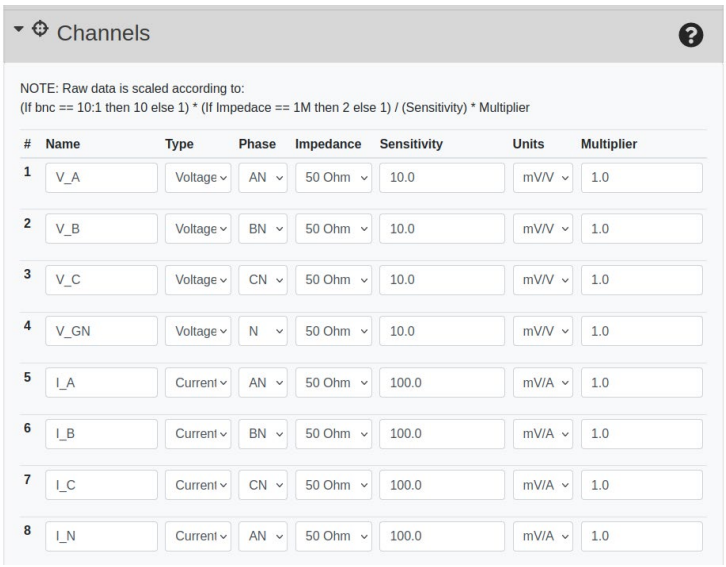


Fig. 15 Individual channel names, characteristics, and scaling can be set through the channels page

3.2.3.1 Channel Names and Types

The leftmost form field for each channel allows users to enter a name for each specified channel. The characters **[.]** and **[:]** are **NOT** allowed in channel names. Users are free to name channels however they see fit, but it is recommended to use names such as I_A or V_A so that voltages, currents, and their respective phases can be easily identified by later observers of the data. It is also recommended to use unique names for channels that will be matched within the same circuit. Channels with the same name, reference, and circuit may lead to overwritten and, therefore, lost data. However, channel names can be repeated for different circuits (e.g., users may have both an I_A and V_A in circuit Main Breaker and circuit Garage Breaker). Creation of circuits are further discussed in Section 3.2.4.

After selecting a channel name, users should select a channel type that most appropriately matches the type of signal that the sensor at that channel is *measuring*. For example, current probes should be labeled with type "current" and D-dot sensors should be labeled with type "EField."

On MPM units, channel names and types should already be configured given that they are static with respect to the cable sensor inputs. Channels 1–3 and 5–7 are typically labeled H1 through H6 and with type "HField," and Channels 4 and 8 are labeled E1 and E2 with type "EField."

3.2.3.2 Phase

The phase of the channel refers to how the measurement is performed. For instance, users can measure voltage either between the phase and neutral (AN) or between

two phases (AB). If users are measuring between phase BA, they should use phase AB with a –1 multiplier. By selecting the phase correctly here, the correct phase factors are applied at the circuit level to turn current and voltage into real and reactive power.

3.2.3.3 Channel Impedance and Sensitivity

Setting the channel impedance and sensitivity is an important step in the configuration process so that correct physical units can be reported by ARTEMIS. Channel impedance and sensitivity are two scale factors that are determined by the sensors residing on those channels. These values should be found in the sensor data sheet and the user provides these values for accurate measurement transformation. The channel impedance field allows users to compensate for high-impedance (1M-Ω) sensors, although most sensors are of 50-Ω impedance. Sensitivities and their respective units are usually listed in the sensor data sheet and are usually in the form, *mV/A* or *mV/V*. These sensitivities get applied in the *relative.py* module, which runs **after** the phasors are computed on the FPGA and pass through the *sychro.py* module, which repackages the phasor data into the dLAMP packet format. In *relative.py*, the phasor values are divided by their respective channel sensitivities. Examples of different sensitivities and their effects are shown in the following.

- **Example 1:** A voltage probe is showing $1.2V$ on the synchro phasor page and has a sensitivity of $10mV/V$. After this stage, the phasor will have a value of

$$1.2V/(10mV/V) = 1200mV/(10mV/V) = 120V\ . \tag{3}$$

- **Example 2:** A current clamp is showing $200mV$ on the synchro phasor page and has a sensitivity of $100mV/A$. After this stage, the phasor will have a value of

$$200mV/(100mV/A) = 2A\ . \tag{4}$$

When using an MPM or other smart sensors connected via LEMO cables to the MUGS, these scaling factors are likely integrated into the sensor electrically erasable programmable read-only memory and read by ARTEMIS on start-up. However, when using traditional commercial off-the-shelf current and voltage sensors with the BNC inputs, the scale factors must be input manually.

3.2.3.4 Multipliers

The final configuration field is the multiplier field. This is an arbitrary multiplier, which allows users to incorporate additional factors that cannot be accurately captured as an impedance or sensitivity factor. The multiplier field is meant to be a

general scale factor that may encompass either the physical installation condition or some other factor from the sensor data sheet. For instance, for a sensor installed backward, a multiplier of –1 can be used. For a 5:1 step-down coil, a multiplier of 5 is desired to bring the values back to what is being measured. This multiplier can also be used for channel calibration in cases where the ARTEMIS is not reporting the correct expected values for particular signals.

3.2.4 Circuits (Configuration for Power Monitoring)

ViPERS may come preconfigured for three-phase DELTA or three-phase WYE power monitoring without any additional configuration. Simply connect 10-mV/V voltage probes to Channels 1–4 (for phases A, B, C, and N, respectively), and 100-mA/V current clamps to Channels 5–8 (for phases A, B, C, and N, respectively). When using sensors with different sensitivity or another phasing scheme (e.g., monitoring two DELTA circuits), settings will have to be updated.

In general, a configuration follows the following steps:

1. Connect sensors and select signal source, data rate, and frequencies from the /configure/hardware page.
2. Check signal integrity on the /synchro page.
3. Input sensor names, sensitivities, and phase angles on the /configure/channels page.
4. Configure circuit (phase reference) on the /configure/circuits page.
5. Check phase reference on the /relative page. This page only shows the current phase adjusted to the voltage phase.
6. Check the power measurements on the /power page.
7. Log data.

ViPERS uses the circuit method to compare current and voltage phasors. By referencing a current to a voltage phasor and using a "phasing" real apparent power can be extracted. This page (Fig. 16) allows users to combine channels into a meaningful circuit. These fields update *phasors.conf* and are read from *relative.py*, which scales synchro phasors before combining them into relative phasors and power measurements.

To create a new circuit, click the "Add New" button at the bottom of the page. An existing circuit can be edited by clicking on the "Edit" button. Clicking the "Delete" button will cause the circuit to be removed. When creating or editing a circuit, a

dialog box will appear where the user can specify form fields and their descriptions (Table 7).

Table 7 Creating/editing a circuit: form fields and their descriptions

Field	Definition
name	The name of the circuit.[a]
Signal (left dropdown)	Specify up to eight signal channels as part of a circuit.
Reference (middle dropdown)	Each signal channel will have the phase of the corresponding reference subtracted off. Leaving this blank will use the default reference for that channel. The default is the first reference displayed in the dropdown box.
wiring	Can be "NONE," "SINGLE," "SPLIT," "3DELTA," or "3WYE."

[a] Duplicate circuit names will overwrite existing circuits and negatively impact performance.

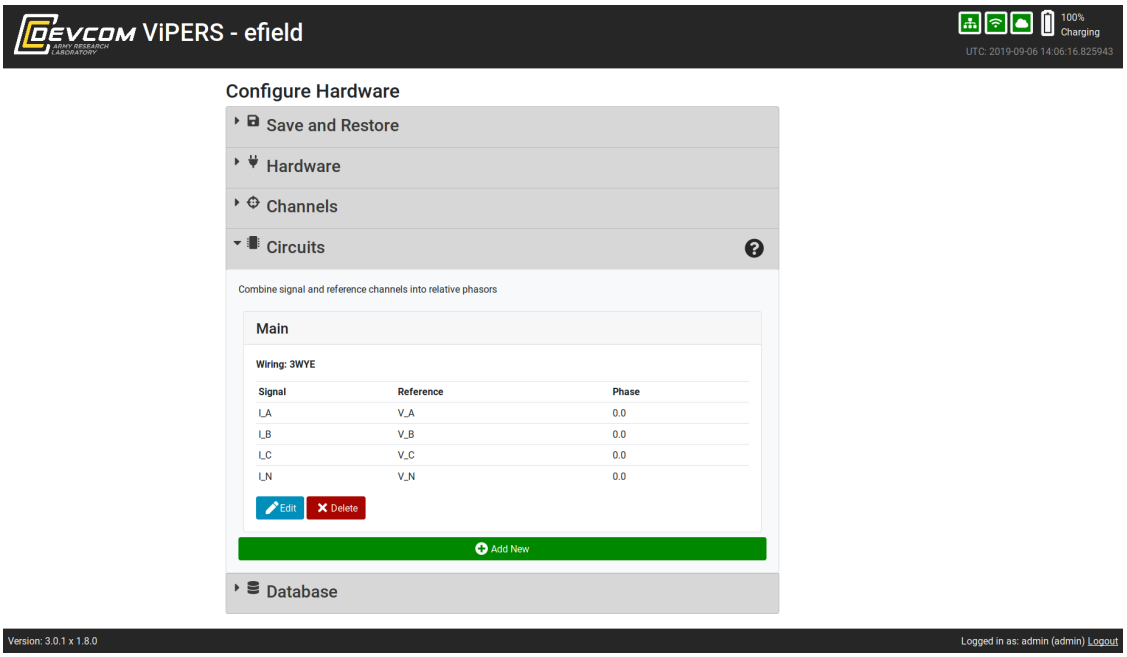


Fig. 16 Channels are referenced and combined into circuits

Each current should have a voltage reference. The easiest configuration is to reference each current to the corresponding voltage.

> **Example 1:** I_A:V_A, I_B:V_B, I_C:V_C (phasing = "NONE", "3DELTA", "3WYE").

However, the user may want to only monitor one voltage (e.g., when monitoring two three-phase circuits with one MUGS-8). In this case, currents can be referenced to a single voltage and a "phasing" scheme can be applied.

> **Example 2:** I_A:V_A, I_B:V_A, I_C:V_A (phasing = "3DELTA", "3WYE").

3.2.5 Database

The database page sets properties for *data_collector.py*, which allows users to edit database update rates (Fig. 17). These fields update the configuration file. The *data_collector.py* module is responsible for inserting circuit data into the system's PostgreSQL database only. Therefore, the configurations on this page only affect the phasor rate and the time between data pushes for the PostgreSQL database and do not affect any other data writing applications. The descriptions of form fields on the database page and are found in Table 8.

Table 8 Database page: form fields and their descriptions

Field	Definition
method	Use "series" helper (efficient inserts), "json" (safest), or "line" (fastest).
Push interval	Performs a structured query language (SQL) query to insert points into the database every *X* seconds. A value of 0 pushes every dLAMP packet received to the database as soon as it arrives.[a]
Average interval	Average data over this window is in seconds. A value of zero is no averaging.[b]

[a] A larger interval here should be less CPU-intensive but introduces delay to data insertions.
[b] Note that data is averaged in the real/imaginary plane, which can introduce artifacts if data is rapidly moving about the unit circle.

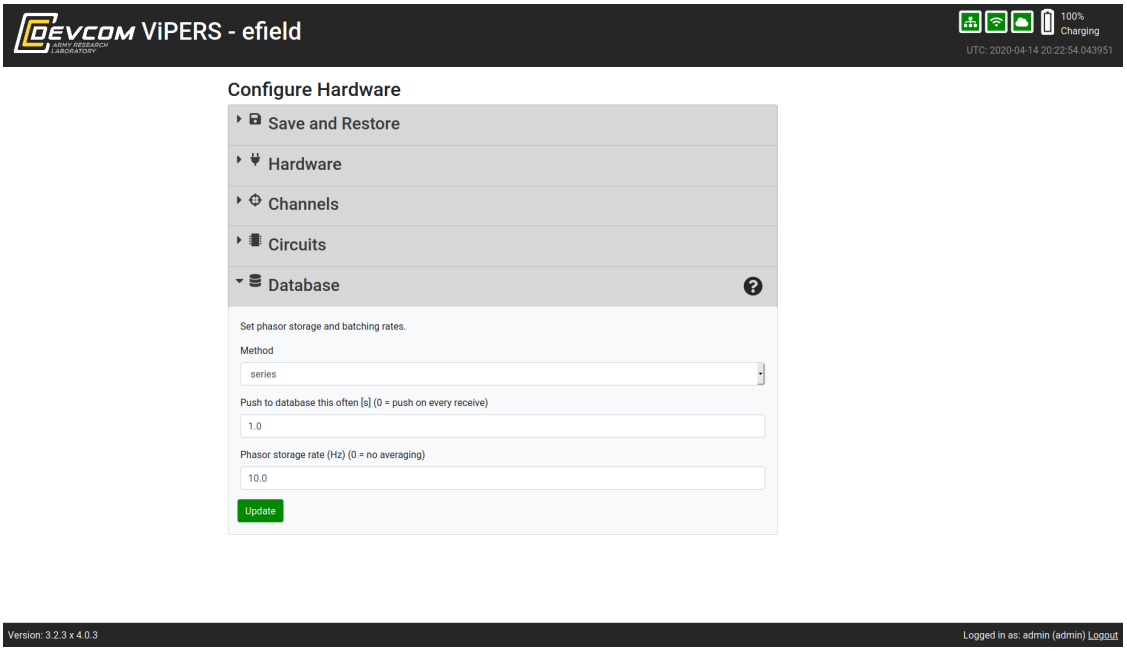


Fig. 17 Phasors can be optionally downsampled before inserting them into the database

The *data_collector.py* module can receive data from any TCP/IP stream that uses the dLAMP packet format. By default, the *data_collector.py* module takes input from *relative.py*, which streams data on port 5436. To collect data from *synchro.py* add the options "-p 5683" to the command line.

3.3 Synchro-Signal Integrity [3]

The plot shown on the Synchro [3] page displays the raw phasors (representations of sine waves) from the selected data source and helps ensure that all signals are connected correctly. These raw synchro phasors are pulled from the output of the *synchro.py* module running in the Dataserver. See Section 5.6 for further details. This page is meant to assist users in determining basic signal integrity. An example synchro phasor plot on the synchro page is shown in Fig. 18.

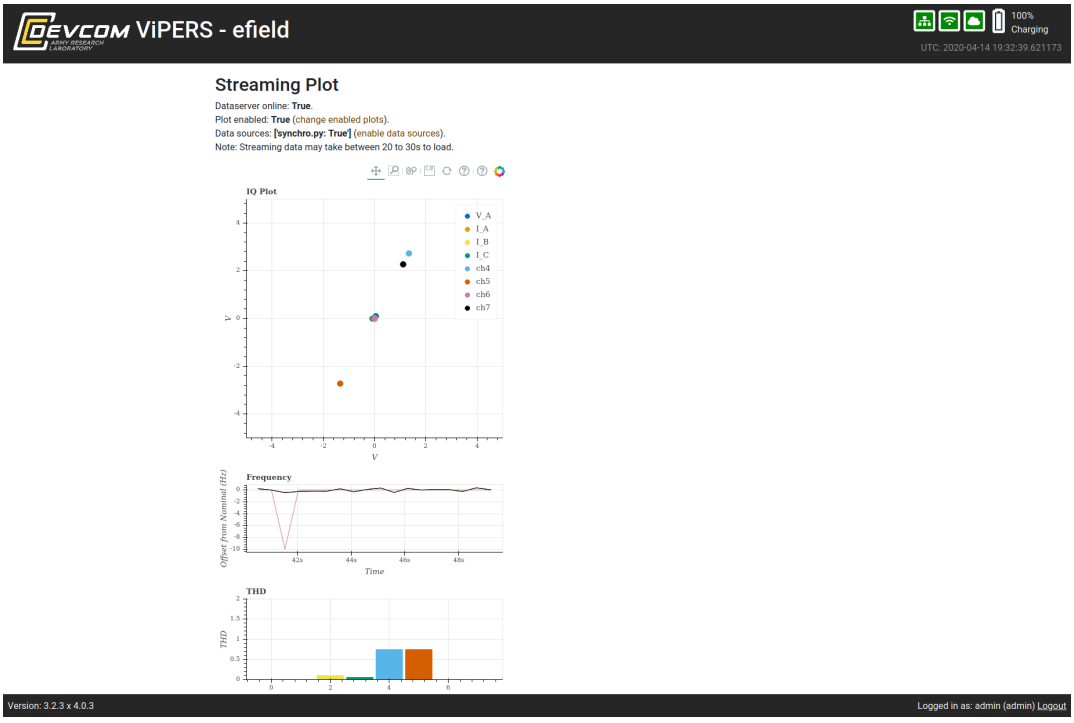


Fig. 18 Example of synchro phasor plot showing the raw data being published by Core100

Due to frequency drift, the phasors on this plot may be rotating in time. This is normal behavior as the internal ARTEMIS clock is not synchronized to the grid frequency; it is instead synchronized to a GPS pulse per second.

Common issues that can arise at this stage are seeing no data or seeing that all signals are zero. If there is no data, when measuring signals of ±10-V amplitude, zoom out on the plot. This may solve the issue. If there is still no data displayed on the plot, check that the correct data source is selected on the Configure/Hardware page (Section 3.2.2). Then check that the *synchro.py* module is enabled on the Modules page (Section 3.18). If all or some of the signals are zero (and expected to be nonzero), the connection between the sensor and the ARTEMIS may be bad. Check that the cables are connected correctly. Avoid loose cables. If the connection looks secure and the signals are still zero on the plot, try replacing the cable with

another one that has been tested and works; the original cable may have been damaged.

At this stage it is advisable to check the approximate phase angles of the voltage phasors. In a split-phase configuration, phase B should be 180° opposite phase A; and in a three-phase system, phase B should lag phase A by 120°, and phase C should lag phase B also by 120°. Relative phase angle between voltage and current phasors can also be checked here although this may be easier from the Relative page (see Section 3.4). To check relative phase angles, see that current *X* is roughly in phase with voltage *X*. Currents may be 180° out of phase if the sensor is installed backward. Signal scaling will be difficult to assess at this stage because scale factors have not yet been applied.

3.4 Relative (Signal Integrity) [4]

Once channels are named and properly scaled, they can be viewed from the Relative [4] page plot. This plot only shows data if at least one circuit has been defined in the Configure/Circuits page (Section 3.2.4). Users can also view data from any defined circuit on this page. An example relative phasor plot from a circuit labeled "Main" is shown in Fig. 19.

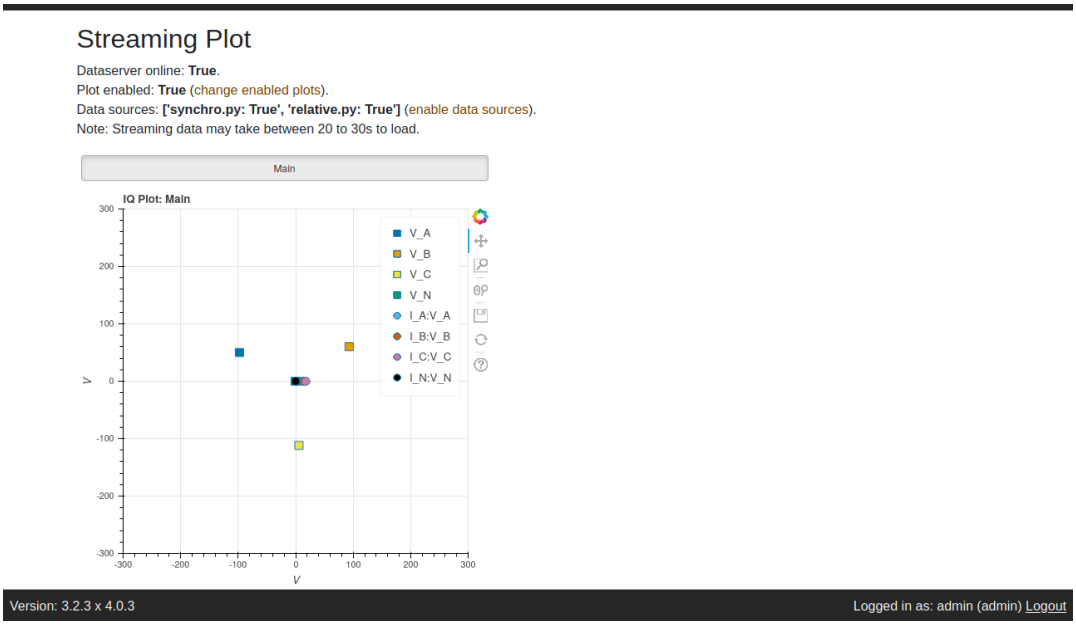


Fig. 19 Relative phasor plot showing only the current phasors

In the plot from Fig. 19, the reference phasors are scaled but their phase is otherwise unaffected. Plotting this way allows users to check that the magnitudes of the phasors are correct and that their relative phase order (ABC vs. ACB) is correct.

A common issue that may occur at this stage is seeing no data on the plot. To address this issue, ensure first that at least one circuit has been defined by visiting the Configure/Hardware page (Section 3.2.4). If there is at least one circuit present, ensure that the *relative.py* module is running on the Modules page (Section 3.18). For more information on the *relative.py* module, see Section 5.6.2.

At this stage, users should be able to easily check and assess the system for correct signal scaling and phase angles. The phasors displayed on this plot on this page are scaled, relative phasors—meaning that both the scale factors from the Configure/Channels page have been applied and current channels have been referenced to their respective voltage channels according to the configuration of the circuit on display. The magnitudes for all voltage phasors should be near expected levels (e.g., 120 or 480 V). Additionally, the phase angle of voltage phasors should be approximately correct for the circuit type. In a split phase circuit, phase B should be 180° opposite phase A. In a three-phase circuit, phase B should lag phase A by 120° and phase C should lag phase B by 120°. If currents are referenced to their respective voltages (i.e., A:A, B:B, C:C), then currents should appear mostly along the 0° axis ±~20°, depending on whether the load is inductive or capacitive. On the other hand, if currents are referenced to a common voltage (i.e., A:A, B:A, C:A), then currents should appear on the plot in phase order (i.e., A -> B -> C). Selecting the proper "phasing" on the circuit will remove this in the "power" stage. See the MUGS Installation Guide[3] for details on properly setting up phasing for a particular experiment.

Note: Voltage phasors are not phase referenced and may be drifting in time. Their relative phase angle may still be verified here.

3.5 Power (Signal Integrity and Circuit Checks) [5]

The Power [5] page attempts to replicate the user experience of a power quality meter, displaying power and reference voltages, frequencies, and total harmonic distortion of all signals. This page shows the power of a particular circuit, as configured on the Hardware/Circuits page (see Section 3.2.4), where power is defined as the product of the voltage and current. The Apparent (or total) power is the magnitude of this phasor, the Active power is the real component, and the Reactive power is the imaginary component. For more information, please review the MUGS Installation Guide.[3] When the currents, voltages, phases, and circuit phase are configured correctly, they should correspond to real and reactive power.

An example of the Power page displaying real-time EP data is shown in Fig. 20.

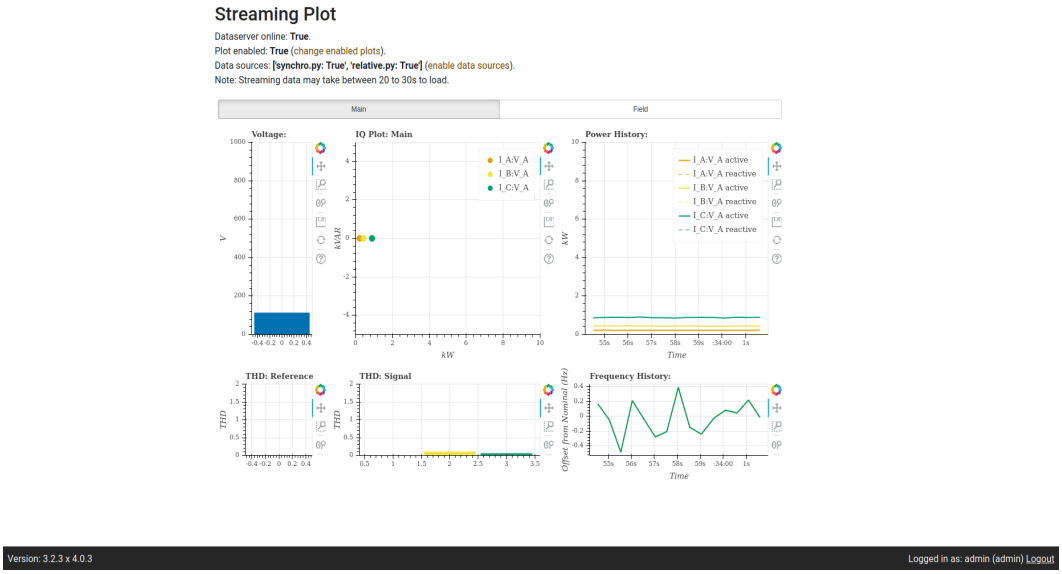


Fig. 20 Power page displays real-time EP data

There are several ways to correctly map phasing such that this page shows power correctly. Many are sensitive to the choices in phase chosen during the channels and circuits stage. When debugging, it is useful to remove these additional phases. This can be done by selecting "N" as the channel phases, and "NONE" as the circuit phase.

A common issue that may arise at this stage is no data being available on the plots. To remedy this, ensure that the *relative.py* module is enabled and running in the Modules page (Section 3.18). Further information on the *relative.py* module is available in Section 5.6.2. Additionally, check that there is at least one circuit defined on the Configure/Circuits page. Users can view data for each configured circuit, and at least one circuit must exist to view data on this page.

3.6 Dashboard [6]

Once the ARTEMIS unit has been configured with the proper sensor scaling and circuit setup, the unit will be capable of producing power measurements, along with some other information—such as frequency and total harmonic distortion (THD) estimates—for each circuit. This data can be stored in either binary files, an internal database, or both, depending on user preference. The database used in this case is TimescaleDB,[4] which is built on PostgreSQL.[5] It stores data points at a much lower resolution than is found in the binary files; therefore, it is not suitable for postprocessing algorithms requiring high-resolution data, but it does provide a great utility in that several years' worth of lower-resolution data can be stored on the OS' SD card. The data in the database can be accessed through the Dashboard [6] page. The Dashboard is built using the Grafana[6] tool, which provides a resource for

building general purpose dashboards. For ARTEMIS systems, two default dashboards exist: “Live Power Updates” (Fig. 21) and “Live System Performance” (Fig. 22).

Fig. 21 The default Live Power Updates dashboard

Fig. 22 Default Live System Performance dashboard

The Live Power Updates dashboard has several displays meant to aid the user in understanding the current data trends. In the top left corner of the dashboard there are two dials showing the current voltage and the current frequency. These dials show whether the voltage and frequency are within acceptable ranges. The voltage should be within ±10% of 120 V and the frequency should be between 59.95 and 60.05 Hz. The dial text appears green when these values are within the acceptable ranges and red when they are not. Moving to the right there are the average-power and average-power, factor-level charts. These charts show the average power and average-power factor for the window of time on display. The default display shows

data from the last 15 min, but this can be changed according to user preference using the dropdown menu on the top right. To the right of these charts is a list of all current alerts. When users create alerts, they will appear in the Alert Status panel. Each alert in this panel shows the alert's name, status, and the status duration. The Live Power Updates dashboard also contains several plots of different key metrics including magnitude of the power used on each phase, the voltage magnitude, and the frequency magnitude. This dashboard pulls data from the PostgreSQL database, which is created and maintained by the *data_collector.py* Dataserver module. See Section 4.2.1 for additional details on database structure and the *data_collector.py* module.

The Live System Performance dashboard assists users in managing sensor system resources. This dashboard has a variety of panels for viewing the real-time CPU usage, average CPU usage, main file system, hard-disk drive (HDD) storage usage, average RAM usage, and current battery voltage. These panels pull data from the *log_stats.py* Dataserver module, which inserts data on these key metrics into a separate PostgreSQL database. The "system" PostgreSQL database stores information on CPU core utilization, memory usage, disk storage space, HDD storage space, battery voltage, battery current and estimated percentage, and charger status/presence. Not all of these metrics are currently in use in the Live System Performance dashboard, but users may desire to add their own additional panels regarding these metrics.

To measure current system performance, users will find the **Real Time CPU Utilizations Per Core** and **Average Memory Used (%)** panels the most useful. When CPU utilization or memory usage is very high (>90%) for extended periods of time, users should be aware that the system may be overloaded; users should take steps to reduce the volume of programs running on the unit. The system disk and HDD storage panels are useful for understanding the storage capabilities of the MUGS/MPM unit. The HDD is currently configured to only store raw data, which has either been streamed directly to the HDD or backed up from the two raw data SD cards on the ARTEMIS board. If the HDD becomes full or nearly full, it is recommended it be replaced to avoid data loss. When the main file system storage becomes full, it is due to a combination of factors. The main file system stores the relative phasor database for each of the circuits as well as the OS and all code files. Users should monitor the file system disk space, perform occasional backups, and clear the relative phasor database to avoid data loss.

3.7 Status [7]

The ViPERS Status [7] page is used for viewing the status of important system processes as well as current alert statuses. These statuses are replicated in the System Report and Alert Report sections of the daily summary message; however, this page provides useful real-time information. Users can also use this page to diagnose and troubleshoot system problems. Status icons are displayed in three colors to represent their statuses. Processes with a green check mark "✓" are known to be "OK," while processes that may require user attention are shown with either a yellow exclamation point "!" or a red "X" depending on the severity of the issue. Example status pages are shown in Fig. 23.

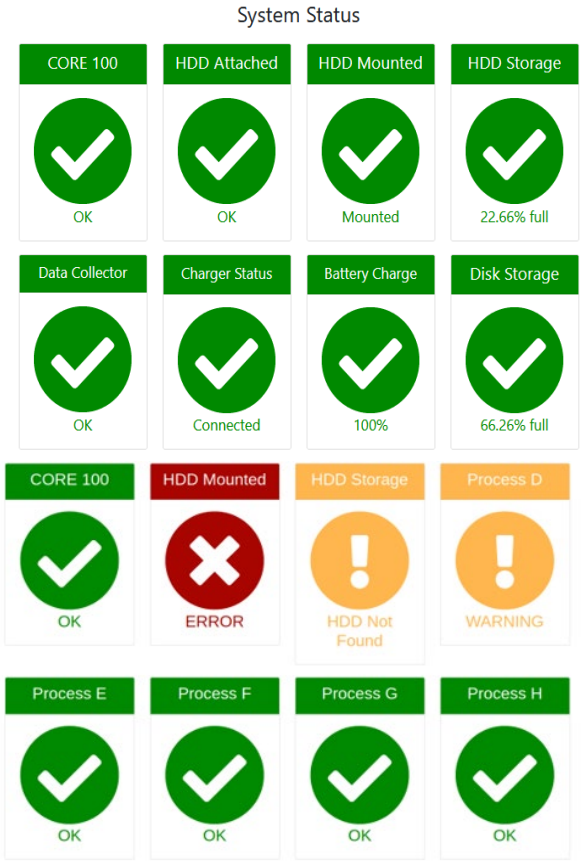


Fig. 23 Status page displays the status of the primary components of the ARTEMIS

The following are important statuses on this page:

1. Core100: System firmware for hardware interaction and lowest-level data streams.
2. HDD Attached (MUGS Only): Physical connection of external drive to ARTEMIS.

3. HDD Mounted (MUGS Only): Software connection of external drive to ARTEMIS.
4. HDD Storage (MUGS Only): Displays percent space left on external drive.
5. Data Collector: Application that inserts data in database. Green status is required for dashboard displays.
6. Charger Status: Comparable status information to the "Battery" LED. Green means battery is charged/charging.
7. Battery Charge: The estimated percentage full of the battery. Note: Battery charge percentage is an estimate and should be treated as such.
8. Disk Storage: Percent use of the primary OS file system storage. If yellow, steps should be taken to free up disk space.
9. Additional Alert Status: Applications and dashboard alerts can be set up to provide additional status information. These will be specific to the unit, so consult with the individual(s) who configured the unit regarding any additional status information.

Some status icons also provide troubleshooting steps and automated fixes. Users can access the appropriate troubleshooting guides simply by clicking the icon underneath each status name. For example, if the Core100 System status has a red "X," the process has failed. Users can restart the process by simply clicking on the icon. This will bring up a troubleshooting menu that will allow the user to restart the process from the page.

3.8 Raw Data [8]

The Raw Data [8] page (Fig. 24) allows users to view live data and download raw data files stored on their sensor systems.

Fig. 24 The Raw Data page contains multiple methods to view, download, and visualize time–domain (waveform) data

3.8.1 SD Storage

In-browser downloads are a new feature added to ViPERS. Users are now able to download their raw data and phasor data via the ViPERS web server using a convenient UI. These downloads are provided for the phasor data stored via the Linux File System as well as raw data from the SD cards or (in the case of a MUGS unit) an attached hard disk. Phasor data can be accessed from the Home > Phasor Data > Phasor Data Storage page, whereas raw data can be accessed from the Home > Raw Data page. Files from the SD card are displayed on the Raw Data > SD Storage page in a plot like the one shown in Fig. 25.

This page allows the user to scan the internal ARTEMIS SD cards for raw data files and download them to the local (OS) SD card for copy to another system. Users can select single or multiple files from "*Bundle Files for Download"* and then copy them individually or concatenate them together into a single file when copied to the OS SD card.

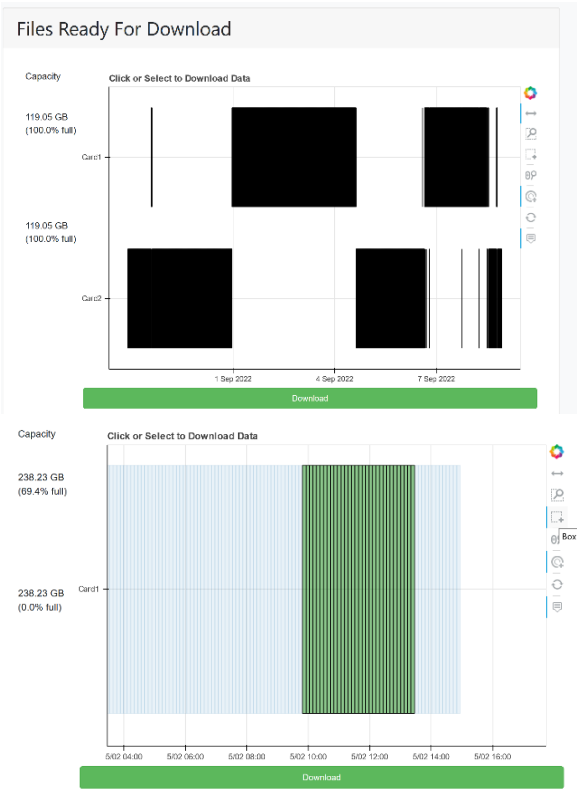


Fig. 25 Raw Data page file display of the two raw SD cards (left); the "box select" feature can be used to select data for download (right)

Files are displayed as a box according to their start and stop times and the SD card they are located on. Users can download multiple files at a time directly from the SD card. These files will be streamed into one continuous file as part of the

download. To initiate a download, simply select the desired file(s)—using the box select tool for multiple files—and click the download button (Fig. 25).

File downloads from the SD cards are typically slow (especially if the SD card is currently being written to). To assist users in requesting speedier downloads, a pop up will appear for larger downloads or ones requesting data from an SD card currently being written to. If the requested SD card is being written to, users are given the option to switch the SD card. Switching the currently writing SD card will have no effect on data storage; therefore, it is recommended to switch if given the option. Also on the SD Storage page is a form that allows users to configure how their MUGS or MPM writes data to the SD cards (Fig. 26).

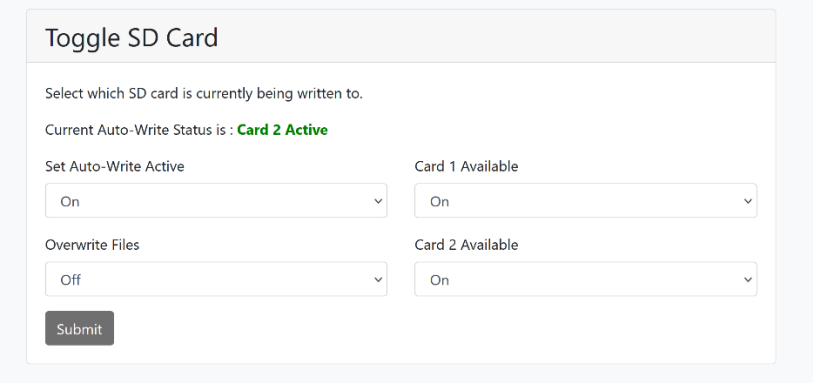


Fig. 26 Form for updating Raw SD writing modes

This form (Fig. 26) provides four options for configuring how the SD cards are written to. The "Set Auto-Write Active" option means that cards will automatically be written to when they have space available. The "Overwrite Files" option allows files to be overwritten by newer files. If this option is selected, the oldest files between each of the SD cards will be overwritten first. The "Card 1 Available" and "Card 2 Available" options mark their corresponding SD cards as available for file writing. Only cards marked as available will be written to.

3.8.2. HDD Storage

Raw data can also be streamed to an external hard drive in raw streaming binary data format.

The HDD Storage page allows the user to view and download those files via the graph under "*View and Download Raw Data on HDD*." Downloading data from the external hard disk is similar to downloading data from the SD cards and can be done on the Raw Data > HDD Storage page. Typically, downloading data from the external hard disk is faster. All external hard disks will store header files in addition to their data files (Fig. 27). These header files should be downloaded alongside any

data files on the HDD, as they contain important metadata for processing the data via the dLAMP MATLAB application. Usage of the dLAMP MATLAB application is covered in greater detail in its associated documentation.[7]

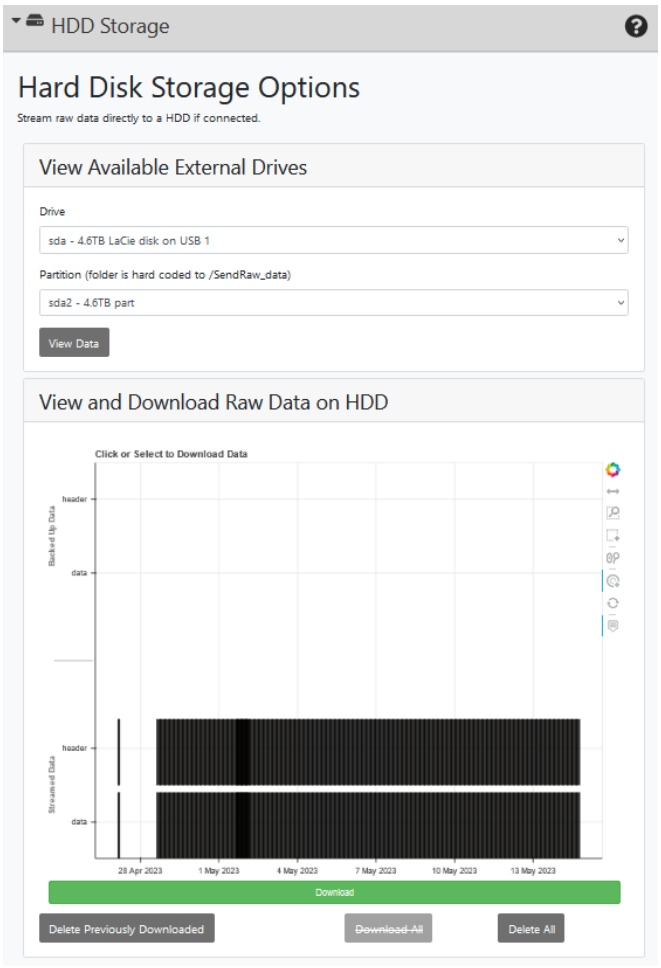


Fig. 27 The HDD Storage page contains raw data files as well as headers with sensor scale factors

The HDD Storage page also provides options for streaming raw data directly to the external hard disk and formatting the disk (Fig. 28). Before streaming or backing up data to the hard disk, users should click the “Format Disk” button to properly reformat the disk. This button will take care of wiping the data on the hard disk, creating a single exFAT partition, and creating the “raw data” and “raw SD data” folders on the HDD where streamed raw data and data copied from the SD cards will be kept, respectively. Users should also specify the duration in hours for each raw streamed file and select whether they want the raw data streaming service to restart after changes are made.

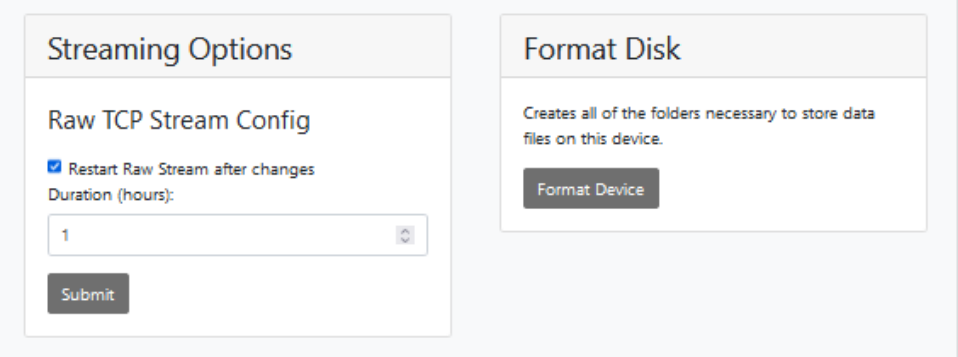


Fig. 28 Starting and stopping raw data streaming and file length (hours) can be set on this page

Users can also enable or disable the raw TCP stream under "*Streaming Options*." Finally, the attached hard drive must be formatted as *ExFAT*, which can be done by the "*Format Disk*" section.

Note: This will destroy all existing data on the external hard drive.

3.8.3 Scope Plot

The Scope Plot page (Fig. 29) runs a Bokeh application that allows the user to load and analyze raw data from any source: internal SD cards, local SD cards, or external hard drive. With the tabs the user selects the location of the data, then uses the drop-down menu to select the data file to examine. The plot will then begin to populate with the data stored in the data file, and a simple Institute of Electrical and Electronics Engineers (IEEE) p-class phasor estimator will be used to give live frequency, magnitude, and harmonic estimates of the data. If the plot does not appear, ensure that is enabled from */dataserver/plots*.

ViPERS includes an in-browser, onboard oscilloscope for viewing live raw data. The oscilloscope can be accessed from the Home > Raw Data > Scope page and provides a useful tool for users checking sensor inputs and waveforms.

All of the important basic features of a traditional oscilloscope are included making it intuitive for new users. The features provided by the oscilloscope tool are four-channel selection, per channel DC offsets and scaling, rising and falling edge triggering, downsampling at 2×, 4×, and 8× rates, and root-mean-square (RMS) amplitude display.

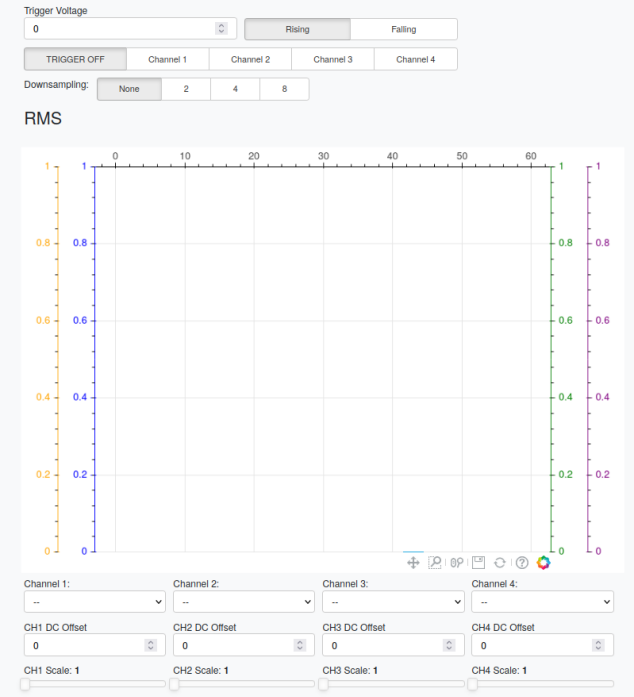


Fig. 29 The initial Scope page prior to setting channels and triggers

Using the Channel Selection drop down options found underneath the main plot, users can select from any number of inputs to MUGS or MPM. These input channels are defined using the Configuration > Hardware page where each channel can be configured and named appropriately. The oscilloscope plot has four Y-axes that are tied to the channels on display, showing users the appropriate amplitude values for each waveform. These axes are color coded and correspond to the four selectable channels at the bottom of the plot. Blue corresponds to Channel 1, yellow to Channel 2, green to Channel 3, and purple to Channel 4. When a waveform has been selected from the channel drop down options, a live plot of that waveform is displayed in the plot, and its RMS amplitude is displayed above the plot (Fig. 30).

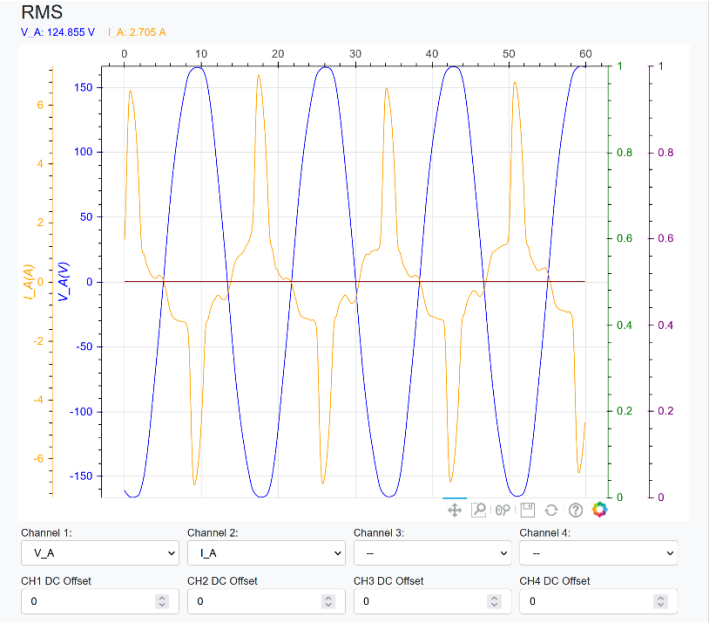


Fig. 30 Raw waveform data produced by the Scope function

Users are also able to set DC offsets and scale factors for each channel individually, and those changes will be reflected in the corresponding axis.

All trigger options for the plot are found above the center plot. The “Trigger Voltage” spinner accepts decimal input and increments by steps of 0.05. Moving to the right, the “Rising” and “Falling” radial buttons allow users to select either rising or falling edge triggering. With the current software version there is no option to trigger on both the rising and falling edges. Directly below the trigger voltage and edge controls are the trigger channel controls. The “TRIGGER OFF” button allows users to turn triggering functionality off, while the Channel selections 1–4 allow users to trigger on the waveforms shown on Channels 1–4, respectively. Triggering on a specific channel will cause that channel—and all other channels—to stabilize in the center of the plot as long as the corresponding rising or falling edge passes through the trigger voltage. In this way, triggering with the ViPERS oscilloscope works in the same way as triggering with a traditional oscilloscope.

Options for downsampling are below the trigger options. Users can plot their data without downsampling or with rates of 2×, 4×, or 8×. This option is provided solely to improve the performance of the real-time plotting capabilities and may be useful for a variety of applications. For example, in cases of intermittent network or degraded network quality it may be difficult for a large number of points to be broadcast to the user’s browser for plotting. Reduced data may help users achieve real-time operation in this case. Additionally, when all four oscilloscope channels are in use, users may experience a drop in performance for higher data rates that can be mitigated by using the downsampling options.

3.9 Phasor Data [9]

Users can download phasor from the Phasor Data [9] storage page. This page functions in the same way as the HDD and SD storage pages. Users can use the box select tool to download multiple files at once. These files will be downloaded as a zipped file additionally containing the *phasors.conf* configuration file generated by the ViPERS configuration page. This page also gives options to delete previously downloaded files, delete all phasor data files, and download all files. Previously downloaded files should be deleted every time the user downloads to conserve space in the Linux file system on the MUGS/MPM. Downloading many phasor files at a time may be time-consuming, especially if the file is large (i.e., over 2 GB). For each phasor data download, a pop-up window will warn the user of the data size and the estimated download time (Fig. 31).

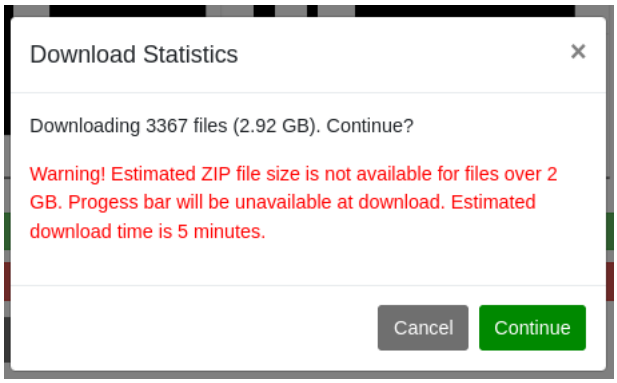


Fig. 31 Download pop-up warning

Also on the Phasor Data storage page, users have the option to enable or disable the phasor data stream and set the length of each phasor file (Fig. 32). Submitting this form immediately stops the phasor data stream and restarts it with the desired parameters, which means that the most recent phasor data file may be truncated in the process.

As files are written, they are displayed in a UI to show where data is available. Individual blocks can be selected, which enable file downloads through the web browser. The display has a few options that need to be understood to navigate the data, especially when there are a large number of files. The zoom feature can be used with a middle mouse wheel. This will likely be necessary to identify the desired download files from specific time periods as the number of files grows. An individual file can be selected, but multiple files will require the click-and-drag feature to be enabled. The plot “options” in the upper right contain a “Box Select” option. Once selected, the user can select multiple files. Hit the download button to initiate the zipping up and downloading of files through the web browser. All downloads will include the **phasor.conf** file. This file contains channel names,

sensor scaling, and circuit setups. These can be used for postprocessing. Analysis of downloaded phasor data can be performed with the dLAMP application,[7] which is discussed later in this manual.

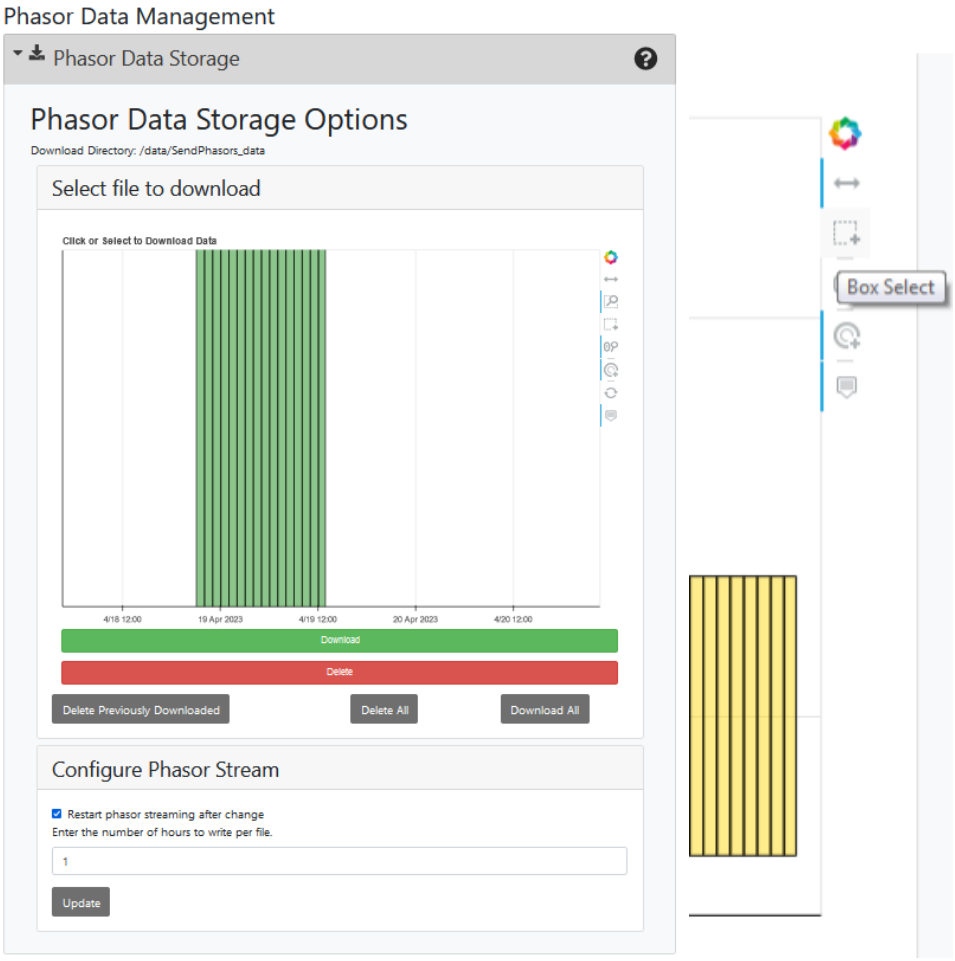


Fig. 32 Single data files are accessed with a left click; the "Box Select" option is used for multi-file selection

The dLAMP architecture is built on phasor processing, and phasor data files are available to access on the MPM/MUGS units through the "**Phasor Data**" page. The phasor data files contain GPS, accelerometer, and 8/16 channels of three phasor harmonics (typically 60, 180, and 300 Hz) data. Individual data files contain 1 h of phasors by default, but this can be changed through the "**Configure Data Stream**" option on the ViPERS page. This feature can also turn the phasor data writing on or off through the checkbox. The enable checkbox will bring up and shut down the **phs_stream.py** module upon updating the *duration* preference, which can be seen on the ViPERS Modules page.

3.10 Autocal (Model-Based Phasor Calibration) (MPM Exclusive) [10]

The MPM, unlike MUGS, only uses electric- and magnetic-field sensors that are already integrated into the sensor package. Therefore, the setup and configuration process is different, but still uses the ViPERS and dLAMP software.[11] MPMs can have their configurations changes (e.g., channel names and scalings in the configuration page); however, changing from the default state is not recommended outside of advanced users with specific use cases. The default configuration uses a 1:1 voltage output from the sensors, which is then calibrated for all sensors for each MPM at ARL using a custom UI. **This process will typically be conducted at ARL prior to deploying/delivering an MPM.**

3.10.1 Settings

The **Settings** tab contains several options for configuring the MPM for an automated "Model-based Phasor Calibration." The settings will arrive preconfigured to the correct MPM model, but the interface allows a user to change this. Changing the existing configuration is not recommended unless changes are recommended by ARL.

For model-based phasor calibration, only certain cables (namely, North Atlantic Treaty Organization [NATO] 8 conductor 200-amp and the Navy LSTGU-400) have been embedded in the MPM because these cable sensors were available for this version of the installation manual. In the **Settings** tab, there are several options for setting up the Autocal phasor calibration (Fig. 33). In most cases, the MPM will be "Production" (5000 series) and the cable twist will be –2.25 for the 200 A NATO cable and 1.00 for the Navy LSTGU-400. The nominal voltage should be set to the voltage of the cable being measured. Finally, there are several options for the "Estimator." These are algorithms used to estimate the voltages and currents on the conductors. If the user is unsure of what algorithm to apply, use the "VoltageEstimator." Once these settings are applied, the user can view the **Streaming** tab to see live estimates out of the phasor calibration. If the "data collection module" is running, the output of the autocalibration algorithm (EP estimates) will be inserted into the local database.

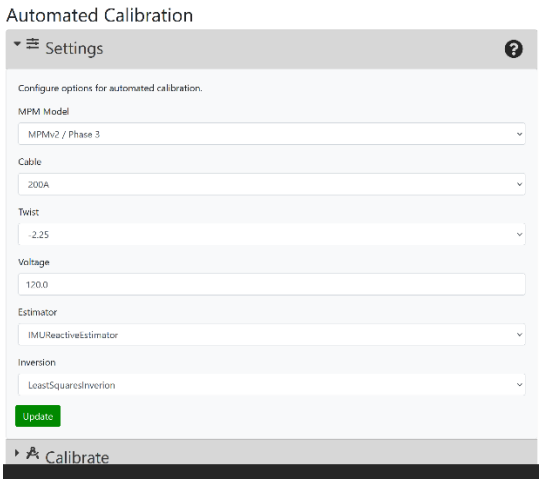


Fig. 33 The Automated Calibration Settings page can be used to set MPM model, cable type, and calibration algorithm

3.10.2 Calibrate

While a partial calibration of the magnetic field sensors can be performed in a magnetic field coil, there is interplay between the ground planes of the cable sensors and the MPM require calibration after installation in the MPM that would not be accounted for. Therefore, we have developed an app integrated into the MPM device where calibration is performed in situ by sweeping the MPM completely around the cable. In practice, it is unlikely that the user will need to perform this task (usually completed at ARL before shipping); however, for cases where it may be necessary to calibrate in situ, the calibration method has been included with the MPM as a live application to provide initial calibration and recalibration capability. This is accessible through the ViPERS Autocal page. This application running on an MPM is shown in Fig. 34. The user can select the MPM version, the load being applied, and cable features during the calibration procedure. Once set up, the user rotates the MPM around the circumference of the cable. Real-time feedback is provided in the plots and circular image, which updates red sections to green as the MPM is rotated. This UI provides an excellent method for calibration that can be understood by standard users when sensor calibration is performed. Once the calibration is complete, the sensor scale factors are saved and applied to all measurements. The procedure only needs to be performed once per unit unless hardware components are replaced.

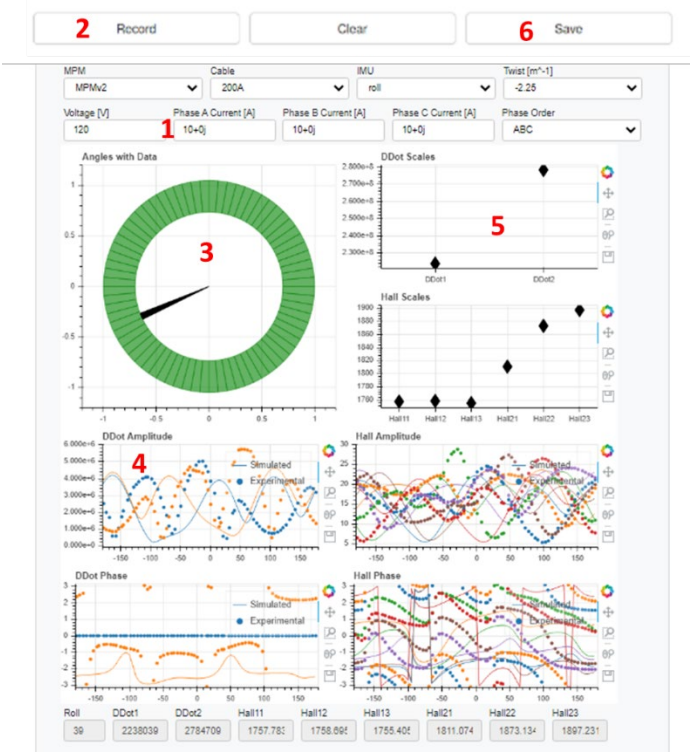


Fig. 34 The calibration web interface in the MPM

One Time (per MPM) Calibration Procedure

1) An eight-conductor 200-amp NATO cable and known loads are used during the calibration process. The known loads (complex if not fully resistive) are entered in the text boxes (1) along with the nominal voltage.

2) The MPM being calibrated is placed on the cable and secured so that it can be rotated around the cable without slipping. Putting thick zip ties in front and behind the sensor wings helps to avoid moving the MPM forward or backward on the cable while rotating. Once ready, the loads are turned on and the “Record” (2) button is pressed.

3) The angular plot (3) will start as all red, but the 5° sections will turn green as they accumulate data. The inertial measurement unit is used to measure the angular position on the cable and measurements from the sensors are binned accordingly. A full rotation of the MPM around the cable should be performed. Ideally, all sections will be green at the end of the run, but this is not required to perform the calibration.

4) There are D-Dot and Hall Amplitude and Phase plots (4) that display the expected sensor responses as the MPM is rotated around the cable. As measurements are accumulated, they will show in these plots. The final fit is used to produce calibration values for each sensor (5).

5) Once all measurements have been made, press the “Save” button to write the measurements and calibration values to file. This calibration information is used in the MPM until a new calibration is performed.

3.10.3 Live MPM Data

Once an MPM has been configured, the information is accessed by the *Automated.py* module, which broadcasts calibrated EP data. The live MPM page receives this information for display. There are plots for Voltage, Phase Current and Phasors, in addition to a plot of the MPM position and Quality estimation as determined by the MPM algorithm (Fig. 35).

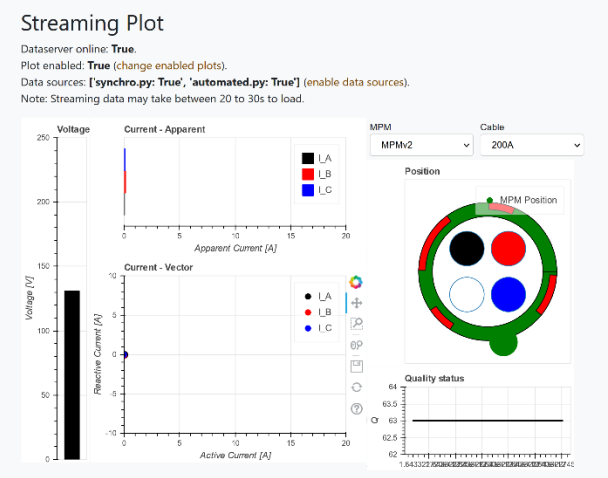


Fig. 35 Live MPM data is streamed showing MPM position and EP data

Live MPM data may also be pushed into the internal database and should be viewable in a customizable dashboard. See Section 3.6 and MPM Installation Guide[8] for more information regarding MPM installation and use.

3.11 Quick Calibrate (Load-Based Calibration) (MPM Exclusive) [11]

The Quick Calibrate [11] page is exclusive to the MPM as it is used to generate the calibration matrix to be used by the MPM to transform the Electric and Magnetic field measurements into the voltages and currents running through each conductor in the cable. This transformation is typically unnecessary on a MUGS because voltage and current can be measured directly through the use of conventional voltage and current probes. This page makes it easy for users to record a calibration sequence using symmetric loads on each phase of the cable and handles the automatic generation of the calibration matrix using the standard load-based calibration method. This application can be accessed from the Home > Quick Calibrate page, and it requires that the *quick_calibrate.py* Bokeh plot is enabled in

order to run. Users can ensure that this is running using the ViPERS Home > Modules > Plots page. See Section 3.18.2 for more information about enabling/disabling plots. In addition to this Bokeh server module, it is also necessary that the PostgreSQL database is running and consistently inserting points. After its running requirements have been met, the Quick Calibrate page will provide live instructions and a live plot to assist users in recording the calibration sequence. The UI for this page can be viewed in Fig. 36.

To begin recording the calibration sequence, the user should click the record button at the top left of the page. Afterward, the page will begin recording data and users should see the Live Data plot update every 10 s. Users will also be provided with live instructions underneath the Live Data plot, providing them with the next step in the calibration sequence. Users must follow instructions promptly as they appear. The instruction set guides users through running a known load on each phase of the cable: first on Phase A, then Phase B, and finally on Phase C. Users can press the stop button at any time to pause the recording, and then press the record button again to continue the instructions. Data will not be recorded again until the record button has been pressed.

Once the calibration sequence has been completed, users will be instructed to click the “Stop” and then “Calibrate” buttons. Before calibration, users should select a current channel to use as reference for the automatic event detection as well as a load value. For the current channel, select a channel with clear changes in magnitude when each load is on. For the value of the load input, the value of the phasor signature of the known load is a complex number. Afterward, click the “Calibrate” button and the Calibrated Current Magnitudes plot will populate with the magnitudes of the currents during the calibration sequence as shown in Fig. 36.

Fig. 36 Live data is buffered and displayed during the calibration process

If the current magnitudes are not identical to the magnitude of the known load and are not properly lined up by phase, ensure that Load value is correct in both the real and imaginary domains. If the page is unable to find a calibration matrix, an error message will be shown on the screen. This is most likely due to the calibration sequence being unable to detect three significant load events during the calibration calculation. This can happen if the resolution for inserting points into the database is not high enough. From the Configure/Database page, ensure that the phasor interval is at least 2 Hz so that the algorithm has enough points to perform the event detection properly.

3.12 Low-Power Mode (System Duty-Cycling) (MPM Exclusive) [12]

Low-power mode (LPM) is a feature exclusive to the MPM that allows users to trade some of the high-performance computing features of the MPM for longer battery life. During LPM operation, the MPM operates in a special duty-cycled mode (Fig. 37).

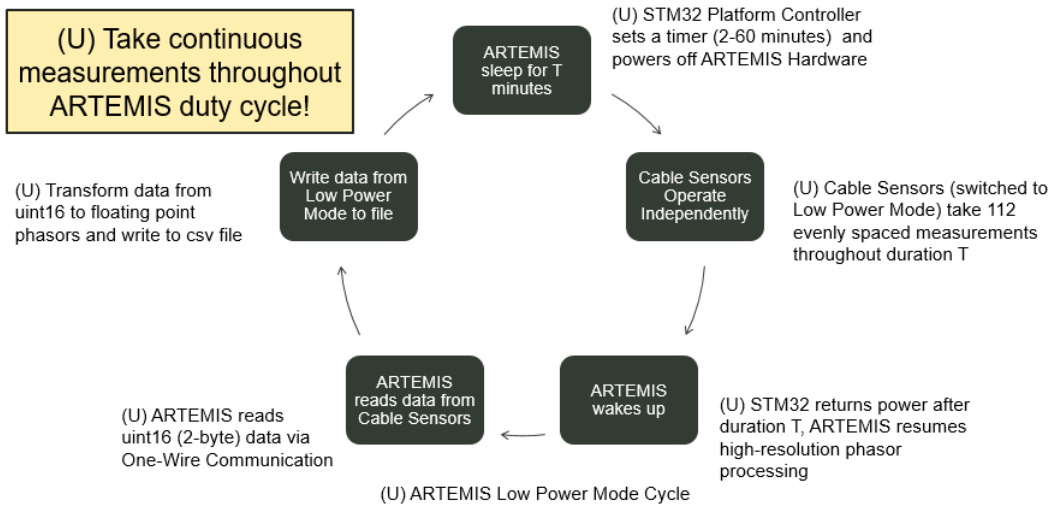


Fig. 37 LPM implements system power-cycling and data collection

During the LPM cycle, the MPM puts the main processing architecture (i.e., ARTEMIS) to sleep and conducts low-power phasor processing on the cable sensor microcontrollers. During the time that ARTEMIS is asleep, the cable sensors will collect 112 evenly spaced field phasor measurements for each E-field and H-field sensor and store them locally in RAM. When ARTEMIS wakes from sleep, it reads these field phasors and inserts them into the field phasor database. If the MPM has been calibrated using the Quick Calibrate module, the raw field phasor measurements will also be transformed into the corresponding voltages and currents that produced the E- and H-fields. These measurements will be inserted into the calibrated database.

LPM is configurable from the Low Power page, where users are able to enable or disable LPM and set time parameters for the duty cycle (Fig. 38). Users can select the amount of time ARTEMIS will sleep during the duty cycle as well as the amount of time it will stay awake after reading phasors.

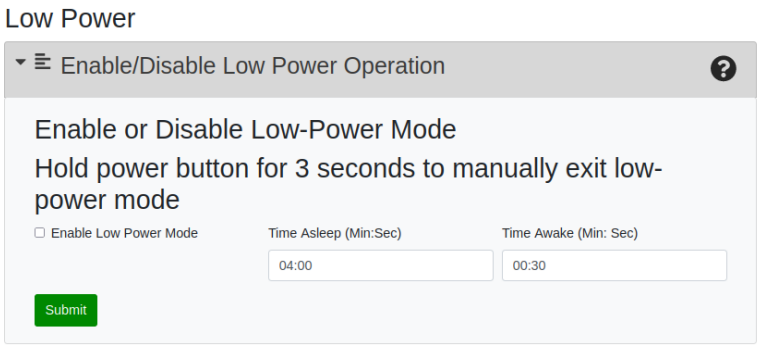


Fig. 38 LPM is initiated using the time inputs and checkbox on the Low Power page

In LPM, the cable sensors can take a single phasor measurement every 1 to 31 s, which is decided based on the ARTEMIS sleep time set by the user. Because the cable sensors can only collect 112 measurements before overwriting previous phasors, if users desire to have measurements spanning the full duration of ARTEMIS sleep, the ARTEMIS sleep time should be between 112 s and 58 min. The time awake parameter can be configured based solely on user preference for the desired amount of high-resolution data per duty cycle.

3.13 Notifications [13]

ARTEMIS is also capable of sending out notification to users via email and text. These features are only available when the ARTEMIS is connected to a network with WAN connection; therefore, some users may not find these features applicable. When they are available, such notifications provide users with very useful information regarding the health of their system and the current state of the data that they are measuring. ARTEMIS units configured for notifications can send out daily summary messages as well as smaller alert messages, depending on user preference.

3.13.1 Subscribing to Notifications

Users can subscribe to notifications from the ViPERS Alert System using the Notifications page. On this page, users enter their email, phone number, and carrier, to receive notifications from their MUGS unit. These notifications are set up on a per-unit basis; therefore, by subscribing to notifications on one MUGS unit, users will only receive notifications for **that specific** unit, not every unit in the network. Users should be aware that all text message alerts are sent through the multimedia messaging service (MMS) gateway and not through short messaging service (SMS). This is done for two reasons.

1. ARTEMIS sends all messages from "vipers.reports@gmail.com" through Gmail's* API service. Therefore, sending a message from an email address to a phone number requires the use of the MMS gateway.
2. The SMS gateway does not allow additional media content (i.e., images) to be sent with text messages.

Since daily summary and alert messages may include scalable vector graphics (SVG) or portable network graphics (PNG) images as part of the report, ARTEMIS must use the MMS gateway. A screenshot of the ViPERS notification page is shown in Fig. 39.

* Gmail is a registered trademark of Google LLC. All rights reserved.

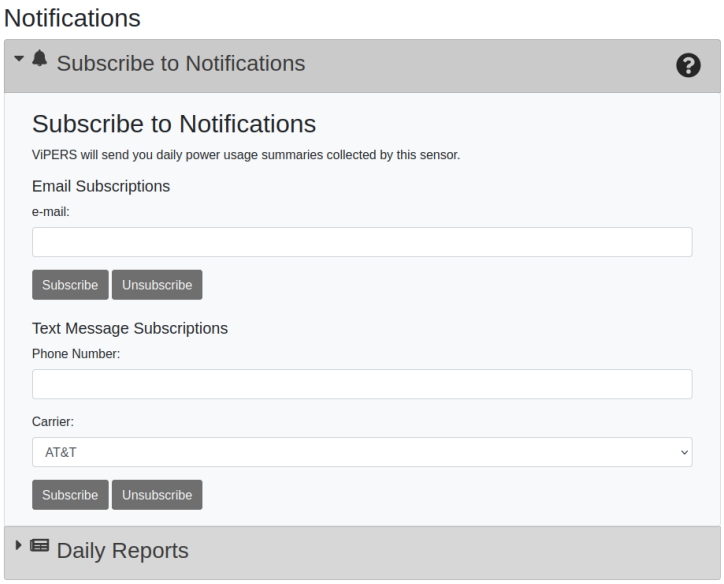


Fig. 39 Email addresses and phone numbers can be added to receive alerts and summary updates

The daily summary messenger *send_summary.py* ViPERS module, detailed in Section 5.7, sends summary emails every day at midnight to users who are subscribed to notifications on the ARTEMIS unit. These summary emails contain information regarding the power system the ARTEMIS is monitoring, information about user-defined alerts, and information about system health. Each daily summary also includes a summary plot that shows high-resolution power data collected throughout the day. The daily summary messenger script collects all data from the "TimescaleDB" database.

An example daily summary report email is shown in Figs. 40 and 41. These screenshots are from an email displayed using an iPhone* mail server. Each mail server has its own way of interpreting HTML email documents; therefore, emails may look different on different devices and email servers (e.g., Gmail vs. Outlook† may not be rendered the same.)

* iPhone is a registered trademark of Apple Inc. All rights reserved.

† Outlook is a registered trademark of Microsoft. All rights reserved.

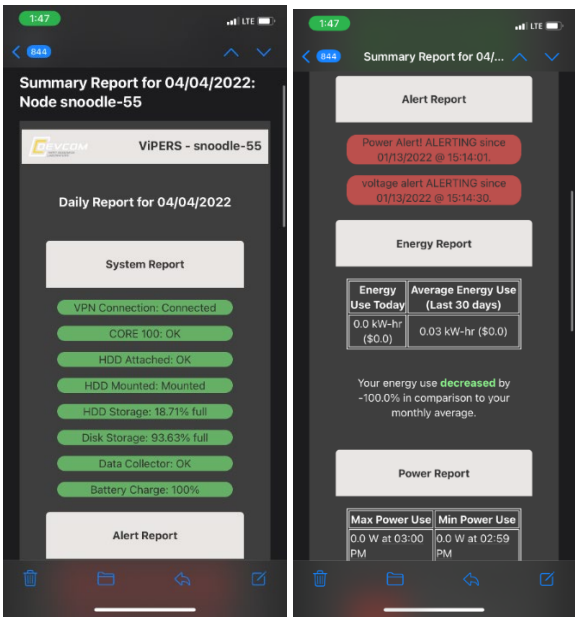


Fig. 40 The daily summary report shows standard and user-specified alerts

Fig. 41 Daily power information is also provided in the summaries

Because it is not currently possible to send HTML formatted messages via text, daily summary report messages are sent as formatted, plain-text messages instead.

By subscribing to notifications, users will receive alert messages marked with the “notify” tag in addition to the daily summary message. See Section 5 for more information about creating alerts and the ViPERS Alert Handler.

3.13.2 Daily Reports

A copy of the daily summary report can be found on the reports page of the ViPERS web application. This page also allows users to view historical daily reports by selecting a date. This page performs many of the same functions as the *send_summary.py* module.

The Daily Report and Status pages (Section 3.7) collectively display the same information as the Daily Summary Report message. The Daily Summary page allows users to access historical power quality reports and daily power usage plots for any day that the ARTEMIS unit was previously collecting data. The Status page details the status of a few important system-level processes running on the MUGS unit, as well as the statuses of all user-defined alerts through Grafana. A screenshot of the ViPERS Daily Summary Page is shown in Fig. 42.

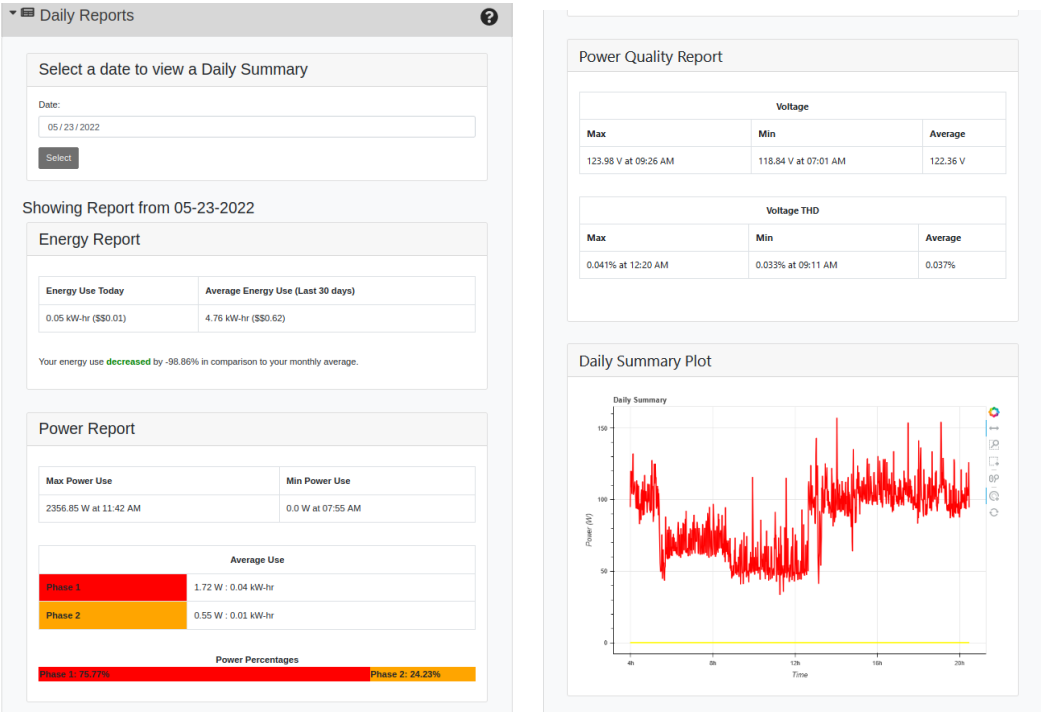


Fig. 42 The Daily Reports page shows the current day's power use (default) or a specified day in the past

This daily report includes several sections aiding users in understanding the activity for their power system for that day. The report includes an energy report section, a power report section, a power quality report section, and a daily summary plot that plots power usage on a minute scale broken for each phase in the power system being monitored. The energy report comprises a table comparing the energy use from the selected day to the average daily energy used over the previous month leading up to the selected day, as well as a statement of comparison. The power report further breaks down the energy used for a specific day and shows the

minimum, maximum, and average power used per phase, as well as their relative power percentages. The power quality report includes two tables representing the minimum, maximum, and average voltage and voltage THD.

3.14 Core100 Interface [14]

The Core100 Interface [14] page is included in ViPERS to provide programmers with easy access to the Core100 API. This page can be ignored by typical users who use ViPERS for configuration and data visualization; however, users who need to program their own applications that may rely on the Core100 firmware may find this page useful. This page contains two subpages (Fig. 43):

1. **Core100 Status**: Displays the status of the core100 service and allows a restart. The functionality of this subpage has also been duplicated on the Status [7] page.
2. **Core100 API**: Executes a range of .GetX() functions from the Core100 API. It allows programmers to view the current function names for all Core100 user-accessible functions.

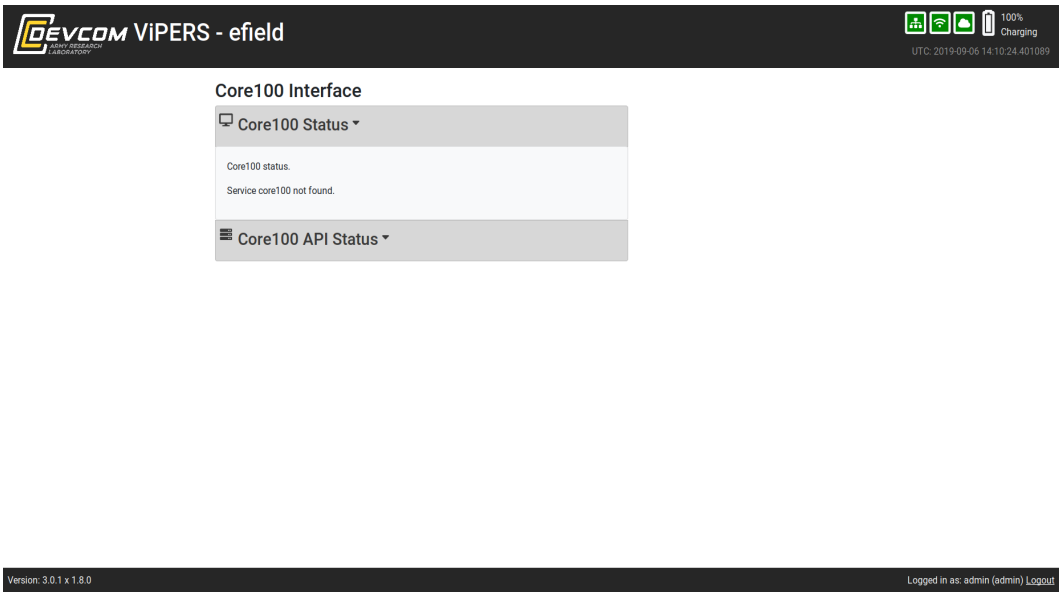


Fig. 43 Core100 and API statuses can be queried on the Core100 Interface page

3.15 Fast Fourier Transform (FFT) [15]

From the FFT [15] page, users can livestream FFT plots for all data channels, either individually or concurrently. The top of the page includes several options for users to select the size of the FFT, potential windowing, and whether to save or load data. When the user selects the "Start Capture Button" the three plots (Fig. 44), showing the raw data signal, the FFT, and an estimate of the THD, will begin updating

according to the integration time selected. Options for integration time are provided in seconds and range in powers of two from 1 to 128 s. Figure 44 shows an example of a single-channel, live-FFT plot.

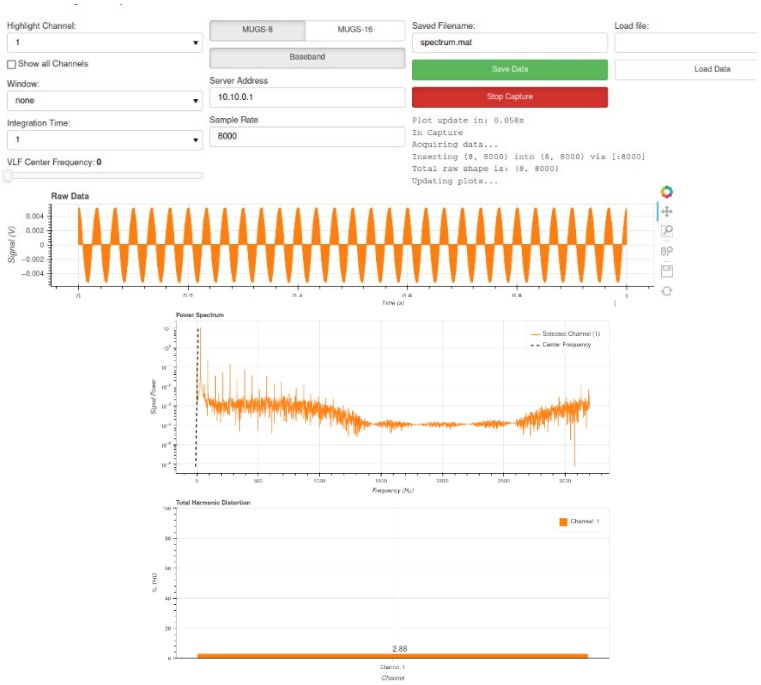


Fig. 44 The time–domain raw data can be processed (top) to frequency–domain spectral data (bottom)

3.16 Help [16]

The Help [16] page lists support topics for ViPERS pages. The list of subjects corresponds to major features and can be expanded or collapsed. Within each subject are specific help subpages, which are ordered alphabetically (Fig. 45).

Each help page can be reached by clicking on the help icon (located in the gray expansion bar) on the corresponding page.

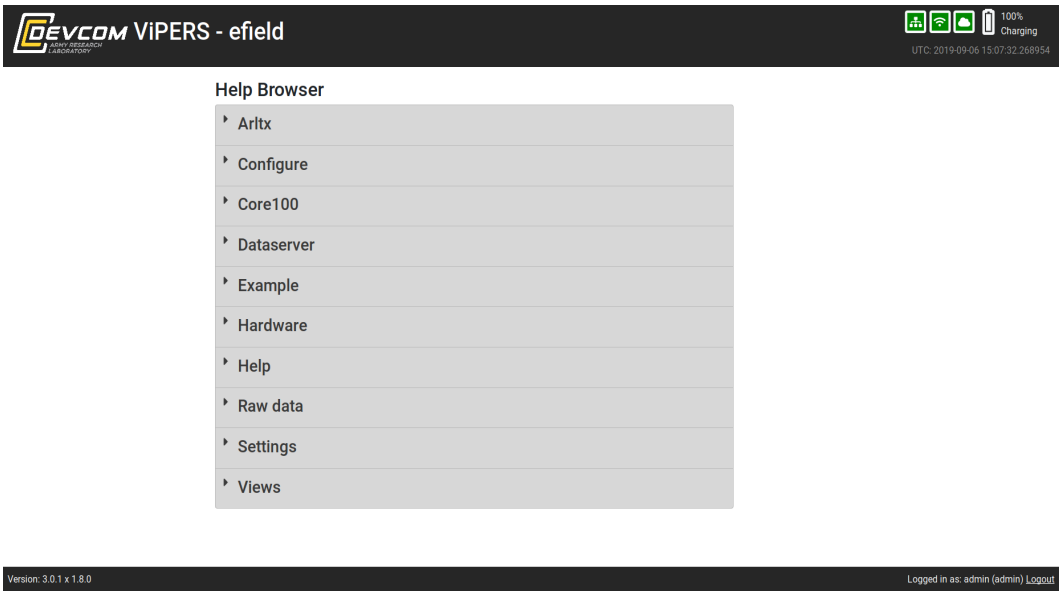


Fig. 45 Integrated Help Browser

3.17 Examples [17]

The Examples [17] page portrays basic instances of the types of pages and user inputs that can be created using ViPERS. This page is primarily for developers to reference when building UIs; therefore, standard users will not need to interact with this page. For more information about these examples, see the ViPERS Implementation Guide.[1]

3.18 Modules (Dataserver) [18]

The Modules [18] page has several subpages associated with the system configuration of the modules that have been loaded into the system. This page allows users to directly interact with the ViPERS Dataserver, which runs several key processes for the unit. It allows the user to bring specific modules up and down, restart them, and check on individual module statuses. Modules may also be brought up and down by other pages as required for operation or to control their behavior.

MPM and MUGS have slightly different modes of operation; therefore, some modules required for the MPM are not relevant to the MUGS (i.e., modules associated with Model-based Calibration).

3.18.1 Modules

The Modules page allows users to interact directly with the processes running on the ViPERS Dataserver and upload their own custom processing code. The Dataserver can run scripts in Python 3.8, Java, and Bash; and users can specify command-line arguments for their custom processing code using the UI on this page.

If a user's processing code requires additional data files or configuration files, users should upload those through the "Config Files" interface (Section 3.18.3). For additional details on implementing custom code on ARTEMIS units, see the ViPERS Implementation Guide[1] and ViPERS Programming Manual.[2] Figure 46 shows an example Modules page with several modules running. As seen, these modules are shown in a table format. The table has columns that map to convenience functions or display the status of the module as described in Table 9.

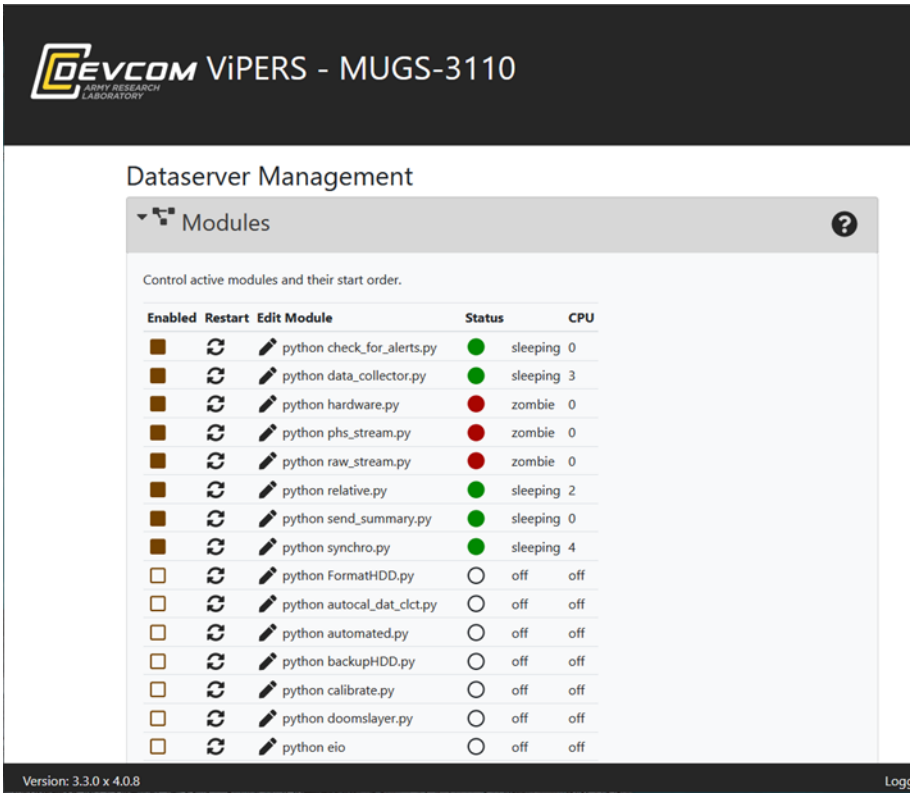


Fig. 46 The Modules tab shows the status of active and inactive modules

Table 9 Modules page: form fields and their descriptions

Column	Description
Enabled	Indicates whether this module is enabled. Enabled modules start running when they are enabled and will restart on ARTEMIS start-up. Clicking the icon changes the enabled state and starts/stops the module accordingly.
Restart	Will restart (stop then start) the module.
Edit module	Brings up a dialog box allowing the user to specify the interpreter and to provide command-line arguments.
Status	The OS classification of the module. Frequently, the status is "zombie" (module exited), "running," or "sleeping."
CPU	The current CPU utilization.

To add a new module, click "Choose File," select the file, and then click "Upload." The upload time can vary depending on network speed and file size. Individual modules can be enabled and restarted. Additionally, the interpreter or command-line arguments can be changed. This page also gives a glimpse into the CPU consumption of the data processing pipeline.

3.18.2 Plots

The ViPERS Dataserver uses the Bokeh plotting utility to serve live-updating plots. Plots can be enabled and disabled from the Modules/Plots page (Fig. 47). To enable a plot, check the box next to the plot's name and click "Update." Due to the tight integration of the Bokeh server with the web application, there is not currently a way for users to upload custom plots that use Bokeh. For custom plotting, it is instead recommended to use the ViPERS dashboard page (see Section 3.6 and associated Grafana[7] documentation for details). To create custom streaming plots with Bokeh see the developer documentation.[9]

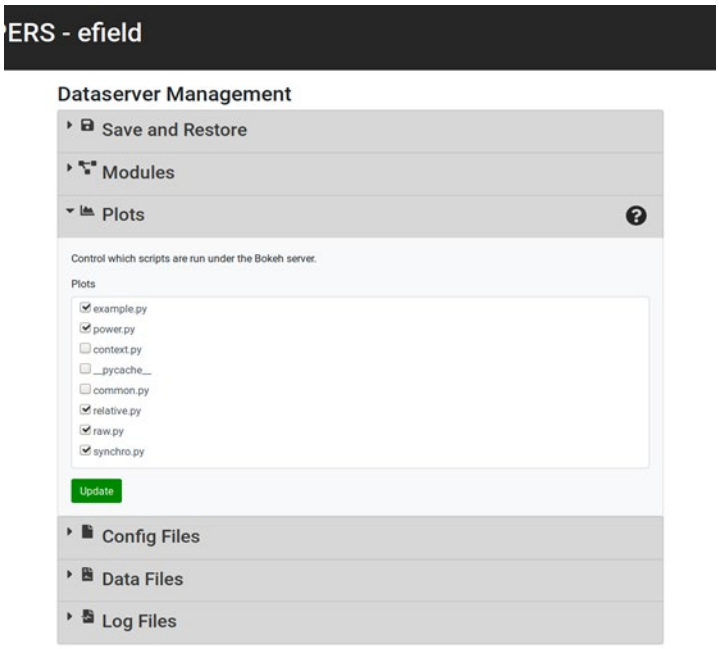


Fig. 47 Specific Bokeh applications can be enabled and disabled

Plots created through Dataserver are produced by the Bokeh Python package, which produces rich interactive plots. Each plot has a toolbar with icons that allows users to change their interaction with the plot. Table 10 shows options for plot interactions.

Table 10 Bokeh interaction icons and their descriptions

Name	Icon	Functionality
pan	cross of arrows	Pan around the figure with the mouse or a tap-drag.
box zoom	magnifying glass	Zoom into the figure with the mouse or a tap-drag.
wheel zoom	mouse and glass	Use the mouse (or pinch-zoom) to scroll in.
save	diskette	Save the figure to the local drive.
reset	circle of arrows	Reset the plot to the default (useful after zooms).
help	question in circle	Links to Bokeh documentation[9] on the toolbar.

In addition to the toolbar, streaming plots and applications may have additional widgets to allow interaction. All pages that require their respective Bokeh plot to be enabled has a Bokeh notification message at the top of the page. Pages that use this utility include Synchro, Relative, FFT, and Raw Data/Scope.

3.18.3 Config Files (Extra Configuration File)

This page allows users to manage ARTEMIS configuration files. These files may be used for both sensor and data configuration (see Section 3.2 on configuration for more details) as well as configuration files necessary for any user-provided data-

processing module. These files are stored locally in the ARTEMIS unit's OS File System. There are three actions users can take on this page:

1. Upload
2. Download
3. Delete

To upload a file, click on the "Browse" button, select a file on the local file system, and then click "Upload." To download a file, simply click on the corresponding file name. To delete a file, click the red "Delete" text next to the corresponding file name (Fig. 48).

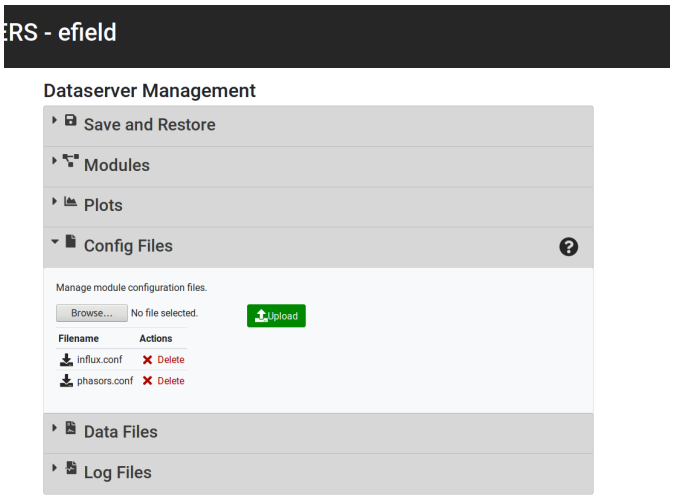


Fig. 48 The management page for module configuration files

User-defined processing modules uploaded through the Modules page that require a corresponding configuration file should reference that file in the directory, */home/snickerdoodle/vipers/dataserver/modules/conf*.

3.18.4 Log Files

This page allows users to view and delete log files generated by the modules managed by Dataserver (Fig. 49). To view the contents of a file, click on its name. A box will expand below showing the last few lines of the log file and will update every second. Once a particular log is open, the user can also download the file by clicking the "*Download Log*" button. Users may also delete the log file by clicking on "*Delete Log*."

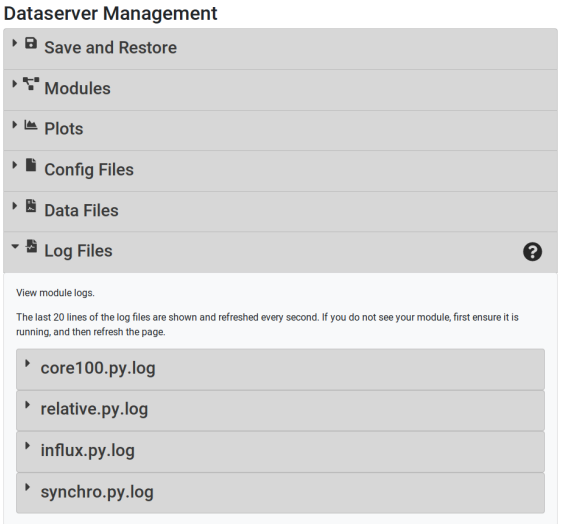


Fig. 49 View log files

4. ViPERS Dataserver Modules

ViPERS includes several stock modules that can be used to process, store, broadcast, and export information related to sensor measurements and system information. Modules are easily added and can interact with one another through socket connections or file reading/writing. Table 11 lists the default modules and their level of importance for each system in a standard configuration.

Table 11 Dataserver modules and their importance level across ARTEMIS

Module	MPM importance	MUGS importance
synchro.py	High	High
relative.py	High	High
phs_stream.py	High	High
raw_stream.py	High	High
data_collector.py	High	High
check_for_alerts.py	Moderate	Moderate
send_summary.py	Low	Low
automated.py	High	None
autocal_dat_clct.py	High	None
estimates.py	Low	Low
log_stats.py	Moderate	Moderate
lowpower.py	Moderate	None

Stock modules are divided into four categories based on functionality in the following sections: data streaming (generate, repackage, and broadcast data); data collecting (collect data from different streams and insert data points into the database); alerting (notify users of events in their system), and MPM-exclusive modules (provide required and desired features for the MPM only).

4.1 Data Streaming Modules

The largest class of ViPERS modules is the data streaming modules, which primarily inserts data from the FPGA into readily usable formats (Table 12).

Table 12 ViPERS data streaming modules and their descriptions

Module	Description
synchro.py	Selects data from a range of sources and format to a standard dLAMP packet.
relative.py	Combines synchro phasors into meaningful relative phasor and power measurements.
raw_stream.py	Reads binary raw data stream and writes it to file.
phs_stream.py	Reads binary phasor data stream and writes it to file.
estimates.py	Produces IEEE-compliant THD, RMS, frequency, and rate of change of frequency (ROCOF) estimates.
log_stats.py	Produces system metrics information and writes it to a database.

The most important data streaming modules are *synchro.py* and *relative.py*. These two modules are present in the main ViPERS phasor data processing chain and are important in providing users phasor data in the form of dLAMP packets. For more information on the dLAMP packet format see its associated documentation.[7] An overview of the main ViPERS phasor data processing chain is shown in Fig. 50.

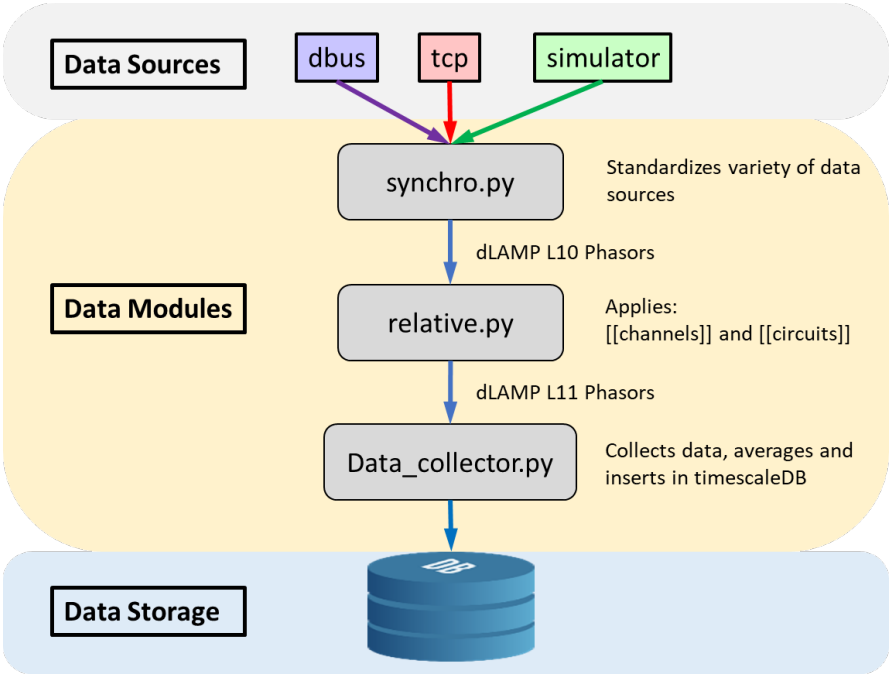


Fig. 50 Overview of the main ViPERS phasor data processing chain

4.1.1 *Synchro.py*

The *synchro.py* module standardizes the outputs of several different data sources to a single output: the dLAMP packet. As shown in Fig. 51, *synchro.py* can accept inputs from various sources (including dbus, TCP, and simulated). The data source is configurable from the ViPERS Hardware Configuration Page (Section 2.1). The *synchro.py* module receives data from any source, repackages the data it receives into the standardized dLAMP packet format, and rebroadcasts the data to port 5683. It accepts up to five clients at a time and will perform connection retries if its client or data sources disconnect.

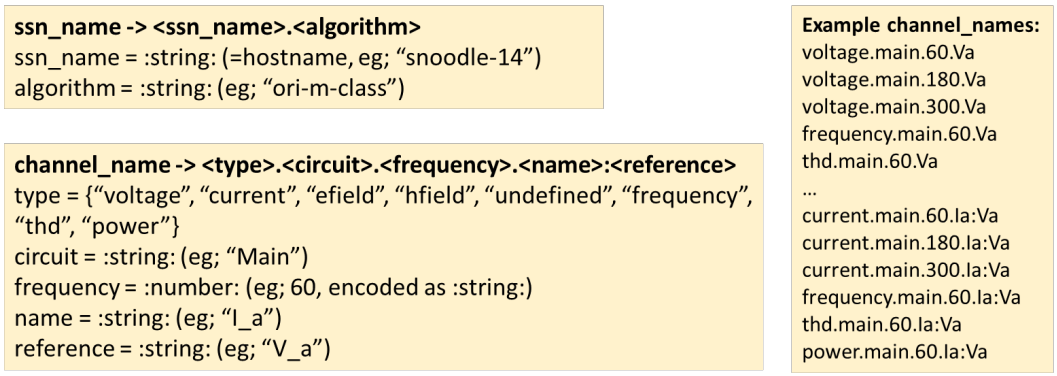


Fig. 51 Example channel names created by *synchro.py*

4.1.2 *Relative.py*

The *relative.py* module subscribes to the synchro phasors published on localhost:5683 and transforms them to relative phasors. Then, it publishes the relative phasors; their reference synchro phasor; and the corresponding power phasor, frequency, and total harmonic distortion value on port 5684. The *relative.py* script accepts and transmits data in either the dLAMP level 10 or 11 phasor packet format.

During the transformation from synchro phasors to relative phasors, channel scaling is applied to transform the data from normalized values to physical units. Channels are also reorganized into circuits as shown in Fig. 52. The channel scale factors and the circuit configuration information are found in the "phasors.conf" file and can be configured using the Configure page. The circuits are scaled according to Eq. 5:

$$scaled = \frac{(measured * impedancefactor * multiplier)}{(sensitivity * attenuation)} \quad (5)$$

Users should therefore ensure that they have entered the appropriate scale factors for each sensor on the Channel Configuration page (Section 3.2.3) to gather properly scaled data with correct physical units. Once channels are scaled, the

synchro phasor signal channel $\left(s(t)\right)$ is transformed into a relative phasor $(\rho(t))$ using a reference channel $\left(r(t)\right)$ according to Eq. 6:

$$\rho(t) = |s(t)|\, exp^{-j\left(args(t)-argr(t)\right)}\,. \tag{6}$$

Additional phases may be subtracted depending on the channel phase and the circuit wiring, as configured via the Circuit Configuration page (Section 3.2.4).

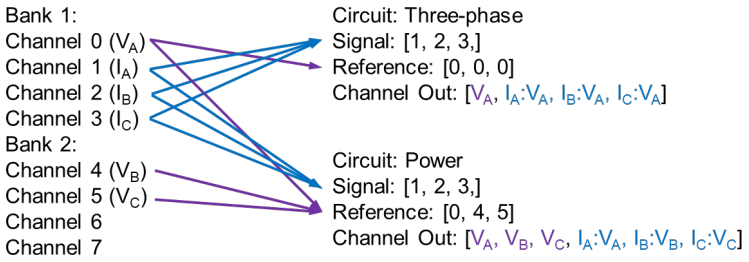


Fig. 52 Circuits are constructed by "signal" channels relative to "reference" channels

After transformation, relative phasor packets are rebroadcast on port 5684 in the dLAMP packet format. This script can accept and transmit to up to five clients at a time and performs connection retries when unexpectedly disconnected. When broadcast, phasor names will contain specific information on each phasor. This information is contained within the phasor names and is included in the metadata packet. The phasor names are sent as a comma-separated string, with one entry for each phasor. Processing levels 11 and 15 may contain additional information (detailed naming) such as the measurement type, frequency, and circuit. This is useful when measuring an electrical circuit where the data consumer needs to perform power analysis on the data provided, which may contain multiple measurements and/or electrical circuits. Phasors can also have custom (generic naming) names if no additional information is required or available. In this case, phasors will not be categorized by the main dLAMP application (Fig. 53).

Detailed Naming:

The detailed naming convention uses the following format: **Measurement_type. Circuit .Frequency.Signal_channel:Reference_channel.**

- **Measurement_type**: Designates the complex measurement the phasor represents. Common types include current, voltage, power, THD, instantaneous grid frequency (frequency), efield, and bfield.
- **Frequency**: The filtering frequency of the phasor, typically a multiple of 50/60 Hz for EP measurements.

- **Circuit**: Commonly designates a measurement as part of an electrical circuit (i.e., 82G_power_closet). There may be multiple circuits sent with data streams, which can be used to categorize the measurements by circuit.
- **Signal_channel**: The measurement channel the phasor is produced from. For a basic measurement type (e.g., current, voltage), both magnitude and phase are produced from the measurement including designated scale factors unless a reference channel is also included, in which case the phase is relative to the reference channel.
- **Reference_channel**: The measurement channel that the signal references (i.e., power measurements). There may be no reference channel if the data is an absolute phasor (or synchro phasor).

rel_phasor_name -> <type>.<circuit>.<frequency>.<name>:<reference>
type = {“voltage”, “current”, “efield”, “hfield”, “undefined”, “frequency”, “thd”, “power”}
circuit = :string: (eg; “Main”)
frequency = :number: (eg; 60, encoded as :string:)
name = :string: (eg; “I_a”)
reference = :string: (eg; “V_a”)

Fig. 53 Explanation of dLAMP name formatting used by the relative module

4.1.3 *Raw_stream.py* and *Phs_stream.py*

Phs_stream.py is an ARTEMIS-based phasor data module streamed from an internal service that broadcasts the sensor-scaled output of the FPGA phasor algorithm. With phasor-streaming Python services, users are enabled to stream raw and phasor data from the outside BNC cables to external hard drives or internal system SD cards.

Under the ViPERS modules page, activate either, *raw_stream.py* or *phs_stream.py*, to run these processes with default parameters. Both processes accept the following parameter options:

--hours=X

--device=Y

where *X* is the number of hours per file generated, and *Y* is the name of the device to stream data to. Inputs can be entered under the respective processes in the modules page. Note that we assume devices are listed under */dev/abc* and only accept the *abc* portion of the device name. For example, if the device is */dev/sda2*, input *--device=sda2*.

The latest rewrite of *phs_stream.py* and *raw_stream.py* are meant to handle two types of common interruptions: disk interrupts and socket interrupts. Upon both

types of interruptions, these processes will attempt to remount the disk automatically, or reconnect to the streaming service with a new socket. The process will fail and shut down with a logged error message if this cannot be accomplished.

Logs can be viewed under the Modules page's log section, or under */vipers/dataserver/modules/logs*.

4.1.4 *Estimates.py*

The *estimates.py* IEEE analysis module is designed to calculate IEEE Standard 519[10] compliant THD and a variety of other metrics for useful raw data analysis: the RMS amplitude, the fundamental frequency, and the frequency rate of change. The IEEE Standard 519 calculates THD using very short-time harmonic measurements from the raw data up to the 50th-order harmonic. The formula for THD and very short-time harmonic measurements from IEEE Standard 519 is reproduced in Eq. 7 (readers are encouraged to seek further documentation in IEEE Standard 519 for further details on calculating THD). This script can be configured to run in either database mode or server mode or both, depending on the user-desired functionality. If configured as a server, calculations are only performed on the raw data when at least one client is connected to save CPU cycles. Users are required to provide both a database and table name when operating in database mode or a *host ip and port* when operating in server mode. This host IP and port specify where the TCP server to emit estimates should be run. The *Estimates.py* module is configured by default to receive raw data through the local host on port 2436; however, the host and port for the incoming data stream can both be adjusted.

$$THD = \frac{\sqrt[2]{\sum_{n=2}^{50} F_{n,vs}{}^2}}{F_{1,vs}} \qquad F_{n,vs} = \sqrt[2]{\frac{1}{15}\sum_{i=1}^{15} F_{n,i}^2} \tag{7}$$

The command line arguments in Table 13 can be provided to *estimates.py* through the Modules page to configure the *estimates.py* module in different ways. If server mode is selected, users must provide both an IP and port through the –i and –p options, respectively. If database mode is selected, users must provide a database and table through the –db and –tb options, respectively.

Table 13 Options for PostgreSQL database

Option	Description
-m {server,db,server/db}, --mode {server,db,server/db}	Mode, either TCP/IP server or database
-db DATABASE, --database DATABASE	Name of PostgreSQL database
-tb TABLE, --table TABLE	Name of database table
-i IP, --ip	Name of database table
-p PORT, --port PORT	Host IP address
thd	TCP port

When running in database mode, a new PostgreSQL database and table will be created based on the user's provided command line arguments. Table 14 shows the names of each column in this new table and their descriptions.

Table 14 User-defined table stores measurements for raw data: frequency, ROCOF, RMS, and THD

Column name	Description
time	Timestamp in YYYY-MM-DD HH:MM:SS.ms
ch	Data channel (0–15 for MUGS-16, 0–7 for MUGS-8 or MPM)
frequency	Estimate for fundamental frequency using the zero-crossing method.
rocof	ROCOF, average over past 3 s
rms	RMS amplitude of the signal
thd	THD, calculated according to IEEE Standard 519[10]

4.1.5 *Log_stats.py*

The *log_stats.py* module interfaces with the OS and Core100 firmware to collect and store key system information for historical logging, identify system issues, and access apps tasked with monitoring system resources. In the default ViPERS installation, this module is configured to run via the system Linux process monitor. This is done to provide users with a more complete snapshot of their system health information so that data can be collected even when ViPERS is not running. The module collects information about system resource utilization, but also the current uptime of key features like the Core100 firmware and the Wi-Fi connection, as well as data about the current battery life. These metrics are all inserted into the "stats_db" PostgreSQL database in the "system" table. Table 15 provides an overview of each column in this database table and its description.

Table 15 The "system" table in the "stats_db" database stores information about current system health

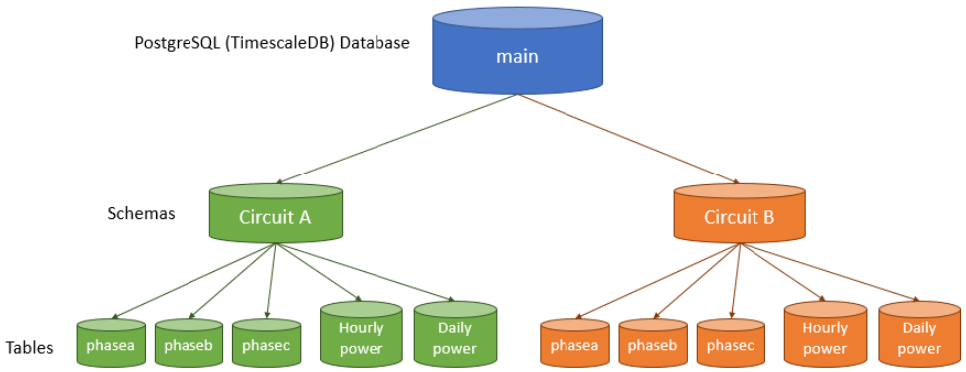
Column name	Description
time	Timestamp in YYYY-MM-DD HH:MM:SS.ms
ram_usage	Percent utilization of RAM
cpu1_usage	Percent utilization of CPU 1
cpu2_usage	Percent utilization of CPU 2
disk_storage	Percent of onboard disk storage used
hdd_storage	Percent of external hard disk used, –1 if no HDD attached
current	Battery current (A)
voltage	Battery voltage (V)
charger	Is the charger plugged in? 0 if no, 1 if yes
percent	Percent charge left in battery
core100	Is Core100 running? 0 if no, 1 if yes
network_device	Is the network device present? 0 if no, 1 if yes
wifi_connection	Is the ARTEMIS connected to the Internet? 0 if no, 1 if yes

This module is used to populate data for the "Live System Performance" dashboard (Section 3.6). For system utilization metrics, this module takes instantaneous snapshots of resource utilization every 30 s, rather than averaging resource utilization over those 30 s.

4.2 Data Collection Modules

4.2.1 *Data_collector.py*

The main PostgreSQL database stores five tables for each circuit defined by the user configurations. It stores one table for each phase of a three-phase power system, represented by tables "phasea," "phaseb," and "phasec." Each of these phasor tables stores a single point of data for each of the following categories: power, voltage, current, voltage THD, current THD, power factor, and frequency. By default, the database stores both the phase and magnitude for voltage and current for the 60, 180, and 300-Hz harmonics (first, third, and fifth, respectively). For the 60-Hz harmonic, it additionally stores the magnitude and phase of power, current voltage THD values, power factor, and frequency. The "hourlypower" and "dailypower" tables only contain power information, but have some additional processing associated—storing minimum, maximum, and total power values. The phasor tables store two points per second, while the "hourlypower" and "dailypower" tables only store one point every hour and day, respectively. A diagram of the database organization is shown in Fig. 54.



In order to access data from a particular circuit, it is necessary to include the schema name in the query. As an example, supposing there exists a circuit schema called 'circuit_a'. To access all columns in the phasea table for this circuit, the main database can then be queried as follows:

```
SELECT * FROM circuit_a.phasea;
```

Fig. 54 Database structure for basic EP data

Database queries are performed with the Structured Query Language (SQL), and users are encouraged to explore the PostgreSQL documentation[4] for more information on creating basic SQL queries. Because circuits are stored as schemas in the database, to access data from a particular circuit users should perform their query by providing the fully qualified table name in the format "schema name," "table name." For example, supposing a circuit schema called "circuit_a" exists, users can access all columns in the phasea table for this circuit using the following query:

SELECT * FROM circuit_a.phasea.

Users can access data from a single circuit named "main" without specifically naming the schema as in the following query:

SELECT * FROM phasea.

The tables "phasea," "phaseb," and "phasec" each have the same columns and store the same types of information for each phase (Table 16).

Table 16 Columns of information stored by the *data_collector.py module*

Column name	Description
time	Timestamp in YYYY-MM-DD HH:MM:SS.ms
harmonic	Harmonic order (first, third, or fifth), integer
power	Instantaneous power, complex tuple (mag, phs)
voltage	Voltage, complex tuple (mag, phs)
current	Current, complex tuple (mag, phs)
thd_voltage	THD of voltage, double precision
thd_voltage	THD of current, double precision
power_factor	Power factor, double precision
frequency	Fundamental frequency, double precision

For all complex tuple values (power, voltage, current) the user can query for the magnitude and phase individually. To do this, query the desired columns in the following format: (column name).mag for magnitude and (column name).angle for the phase.

Format for selecting metrics from PostgreSQL database

- **Example 1:** Querying for voltage magnitude on phasea.

SELECT (voltage).mag FROM phasea:

- **Example 2:** Querying for phase of the current on phaseb.

SELECT (current).angle FROM phaseb:

The "hourlypower" and "dailypower" tables summarize power information (Table 17). These tables are currently used solely for increased performance when generating the Daily Reports page and Daily Summary messages discussed in Sections 3.13.2 and 4.3.2, respectively.

Table 17 The information available in daily and hourly summary tables

Column name	Description
time	Timestamp in YYYY-MM-DD HH:MM:SS.ms
avg_total	Average total power, calculated by summing power for phasea+phaseb+phasec
min_total	Minimum total power, " "
max_total	Maximum total power, " "
avg_a, avg_b, avg_c	Average power on phase A, B, C
min_a, min_b, min_c	Minimum power on phase A, B, C
max_a, max_b, max_c	Maximum power on phase A, B, C

4.2.2 *Autocal_dat_clct.py*

A comparable module to *data_collector.py* is available for use with the data streams created by *automated.py*. The *automated.py* module applies calibration to MPM field measurements, translates them to estimates of voltage and current, and broadcasts the data using a dLAMP data stream. This module connects to the data server, unpacks the data, applies the PosgreSQL schema, and inserts the data into the *mpm_cal* database. Using the same login criteria as the "main" database, data produced by this module can be accessed from the *mpm_cal* database. MPMs may arrive with preconfigured Dashboards that pull data from this database for live-data views, which can be accessed through the ViPERS Dashboard page. Specifics on the calibration process and selectable options can be reviewed in the Calibrate page descriptions.

4.3. Alerting Modules

The Alerting Modules *send_summary.py* and *check_for_alerts.py* mainly function as part of the larger ViPERS Alerting System. This system is designed to allow users to create custom alert messages through the Dashboard page and receive messages from the ARTEMIS unit, provided it is connected to the Internet. This section functions as not only an overview of these two modules, but also the larger Alerting System as a whole.

The ViPERS Alert Handler is a ViPERS Dataserver module that utilizes the built in Grafana[5] dashboard and alerting features to allow users to create custom alert messages and receive email and text notifications. Section 4.3.1 will cover all necessary information for users to 1) understand the process an alert goes through to be received by the user, 2) create customized alerts, and 3) understand the content of their alert messages.

4.3.1 ViPERS Alerting System Overview

Figure 55 shows the process ViPERS uses to send alert messages to users. There are five steps in the alerting process. As can be seen in Fig. 55, the Grafana Dashboard is used for the alert creation and detection stages in the alerting process, while the ViPERS Alert Handler Dataserver module generates the alert messages and sends them to subscribers.

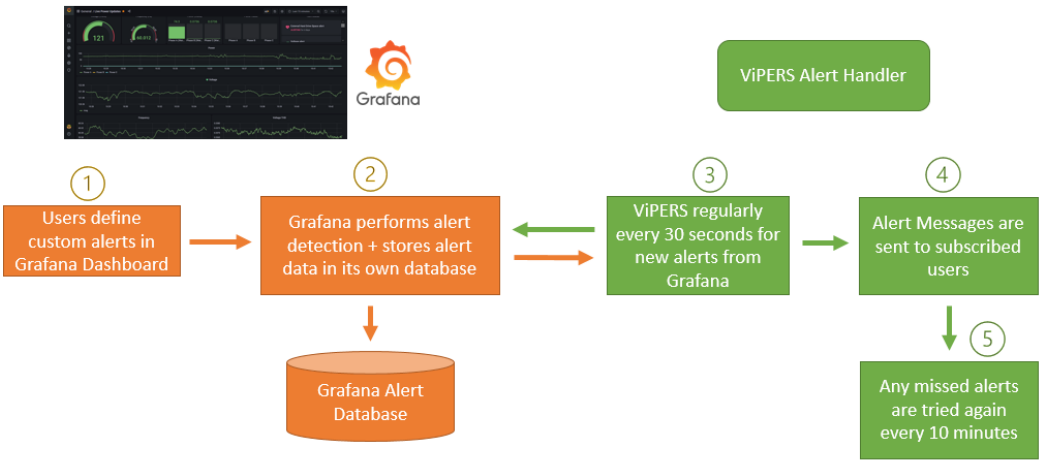


Fig. 55 The Dashboard Alerting System is integrated with ViPERS for easy, user-configured notifications

In the first step, users define custom alert messages using the Grafana Dashboard UI. Alerts are defined using an "alerting condition" that must be true for some user-defined duration before an alert is detected.

In step two, Grafana performs alert detections by periodically checking each alert's alerting condition and assigning it to one of three possible states: OK, PENDING, or ALERTING. The OK state is assigned to alerts whose alerting condition is not currently being met. The second state, PENDING, is used for alerts whose alerting conditions are currently true, but have not been met for the user-specified duration. The final state, ALERTING, is used for alerts whose alerting conditions have been true for the user-specified duration or longer. Users only receive notifications about alerts in the ALERTING state. Alerts move in the following ways only: OK $<->$ PENDING $->$ ALERTING $->$ OK (i.e., alerts can move back and forth between OK and PENDING, but an alert must move from PENDING to ALERTING before any notification will be sent to the user). Information about current and historical states for all user-defined alerts are stored internally in Grafana's Alert Database. This information includes the alert name, user-written messages associated with the alert, tags associated with the alert, and the time at which the alert changed states.

In step three, ViPERS periodically scans the Grafana Alert Database for alerts that have recently moved from the PENDING state to the ALERTING state. This scan happens every 30 s. In alert creation, the user can define tags to customize their alerts, which are used in the step to construct each alert message. ViPERS checks each alert's tag to determine whether users have elected to receive the message before generating an alert.

After they are created in accordance with user specifications, alert messages are sent out in step four. The ViPERS Alert Handler utilizes the Gmail Service API to send all alert messages from "vipers.reports@gmail.com." Alert messages can be sent either via email or text message (Section 4.4).

If an alert message fails to send to the user (e.g., due to network connectivity issues or power failures), they are queued for later delivery. Any missed alert messages are then retried every 10 min in step five. These alerts are only queued within the Alert Handler and they are not saved to file; therefore, queued alerts will not survive MUGS/MPM restart or shut down. However, if the alert is still in the ALERTING state when the MUGS/MPM starts again, the message will be sent as normal.

4.3.2 *Send_summary.py*

This module interfaces with the local database, system firmware, and OS to send out daily messages regarding status of the sensor system. This module should only be activated when an Internet connection is expected to be present. The module reads phone numbers and email addresses that have been added through the notifications page. When Internet access is available, the *main* database will be accessed and EP data will be pulled from the current day—from midnight through

23:59. In addition, statistics on power use from the last 30 days will be pulled for comparison purposes. For details on the information provided in summary reports, review Section 3.13, Notifications [13].

4.3.3 *Check_for_alerts.py*

This module interfaces with the system firmware (Core100), OS, and the Grafana alerting system to identify system issues important to ARTEMIS users. For instance, if a hard drive is disconnected or reaching capacity, this information will appear on the Status [7] page. This information can also be sent as an email or text notification if Internet access is available. Similar to the *send_summary.py* module, notifications can be added or removed from the page described in Section 3.13, Notifications [13]. Alerting messages will be sent out whenever an alert is triggered; therefore, care should be taken when configuring alerts in the Dashboard. Avoid setting a threshold close to a nominal operating condition, which may trigger numerous undesired alerts if data collected regularly rises above and below the threshold.

4.3.4 Creating Alerts

In this section, users will learn to create and configure their own alerts from the Grafana Dashboards. Users will learn to create a simple alert to notify users when a known load (i.e., a space heater on phase B with power usage 1.2 kW) has been turned on. Although this alert is rather simple, through a combination of smart SQL queries and math functions on queries, it is possible to create quite complex alerting conditions. The process of alert creation, however, remains the same for both simple and complex alerts.

In the Grafana Dashboard UI, the user can create alerts tied to specific dashboard panels. To begin the alert creation process, users right click on the title of the desired dashboard and select the “Edit” button to open editing mode for the dashboard panel (Fig. 56). From the edit view, users first click on the alert tab underneath the plot, and then click the “Create Alert” button.

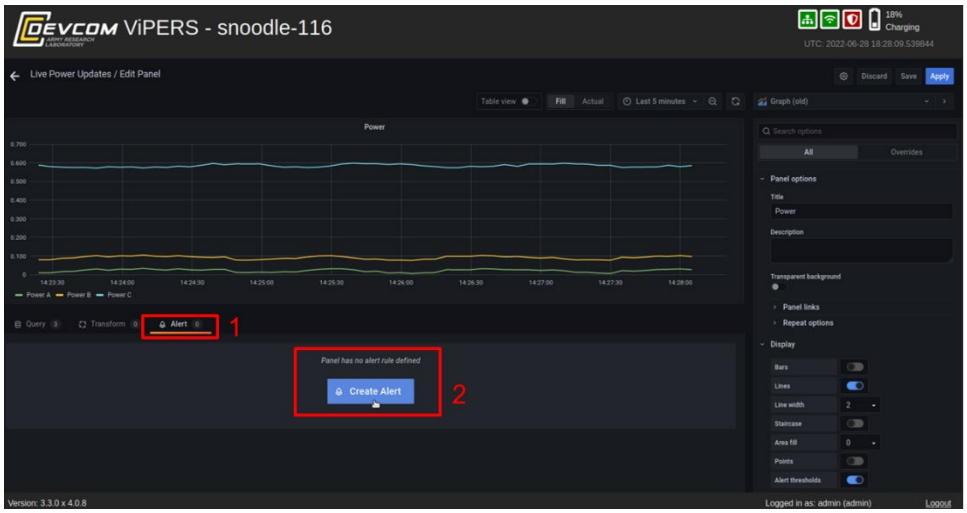


Fig. 56 Alerts are created in an individual dashboard panel

After the alert has been created, users are able to define their own alert conditions. In this step, users select a name for the alert and parameters deciding how often to evaluate the alerting conditions. The "Evaluate every" condition defines how Grafana should check to see if the alerting condition has been met, and the "For" condition defines the duration for which the alerting condition should be true before the alert is considered to be ALERTING. In other words, the "For" condition defines how long the alert should remain in the PENDING state before it switches to ALERTING. The last part of step two is to define the alerting condition. This is the condition that must be true for the alert to switch into the PENDING state. Grafana provides a variety of options for users to select their desired alerting condition. Users can choose from a variety of functions, queries, and conditional statements to define their alert (Table 18).

Table 18 Alerting functions available to users

Column name	Description
min()	Minimum value in the data set
max()	Maximum value in the data set
sum()	Sum of all points in the data set
count()	Total number of data points in the data set
last()	The last data point in the data set
median()	The median value of the data set
diff()	Maximum power on phase A, B, C
diff abs()	The difference between the first and last data points in the data set
percent diff()	The absolute difference between the first and last data points in the data set
percent diff abs()	The percentage absolute difference between the first and last data
count non null()	Same functionality as count, but ignores null values

These functions are applied to the selected query, which is defined using a desired data set as well as a time interval. The data set can be selected from any query that is plotted on the dashboard panel. Finally, users can select a conditional statement from the following list to create their condition:

1. IS ABOVE
2. IS BELOW
3. IS OUTSIDE RANGE
4. IS WITHIN RANGE
5. HAS NO VALUE

After specifying a conditional statement, users specify a threshold value or values (in the case of range options). Users can chain together multiple alerting conditions using "AND" and "OR" connectors to make even more complex alerting conditions. After defining the alerting conditions, boundary boxes should appear on the associated plot to assist the user in visualizing and further refining their query. An example threshold alarm is shown below in Fig. 57. In Fig. 57, an alert named "Space Heater" is evaluated every minute for 1 min and triggers on the condition when the last data point in the data set of query B is above 1.1 kW. Query B in Fig. 57 is the query on the plot that holds the data for phase B power. The alert therefore triggers when a load using at least 1.1 kW appears on phase B for at least 1 min.

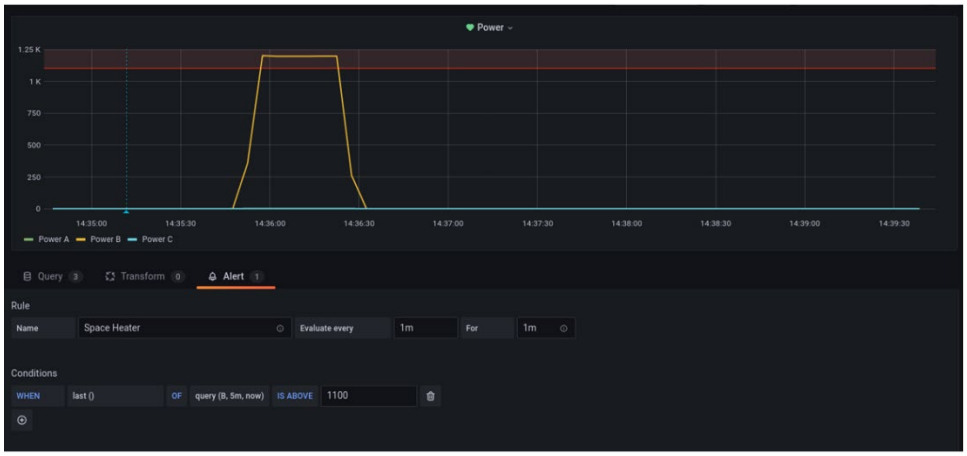


Fig. 57 The Dashboard panel displays a red threshold line where the alert has been set

In step 3, users write a custom message that describes the alert and alerting condition for anyone that may receive a notification. Under the "Notifications" header, users will find a text box labeled "Message" where they can include this custom message. This message will appear in both the email and text alerts along

with the alert name and the time the alert triggered, and therefore can be used to describe a variety of useful information to the end user. Some examples include a long-form description of the alert, instructions for remedying the underlying issue causing the alert, and links to web pages where the user can view more information about the alert.

After defining a custom alert message, users can add tags to their alerts. Tags are used by the ViPERS Alert Handler to properly route and further customize messages. ViPERS currently provides two defined tags for users to add to their messages. These are the "notify" and "include" tags. The "notify" tag is a Boolean tag taking values that are either "true" or "false." As users may or may not desire to receive individual alert messages for every alert they create, the "notify" tag allows users to toggle sending alert messages on and off. This feature may be desirable for lower-priority alerts that the user may want to keep track of through daily summary messages or via the ViPES status page, but which may not require immediate user notification. The default value for the notify tag is "false;" therefore, by default any newly created alert is not configured for ViPERS to send notifications. To turn on individual notifications for an alert, users type the tag name "notify" in the left box, and the value "true" in the right box and then click the "Add Tag" button as shown in Fig. 58.

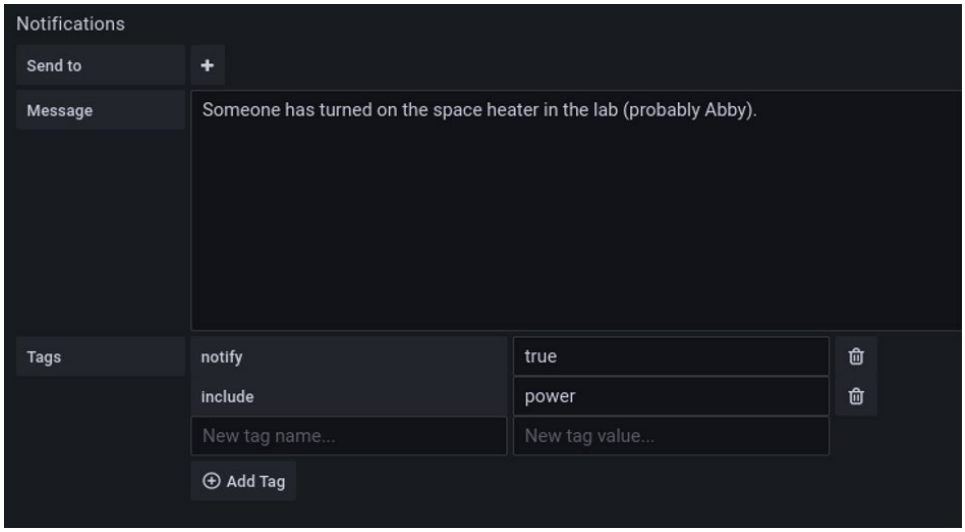


Fig. 58 An alert message can be set, in addition to tags that specify email/text and metric plot to generate

The "include" tag is slightly more complicated than the "notify" tag, as it provides more potential options to the user. The "include" tag allows users to specify the plots they would like to send along with the notifications for the alert. Users are currently able to select from any of the following key metrics tracked by the MUGS/MPM internal PostgreSQL database:

1. power
2. voltage
3. THD voltage
4. THD current
5. current
6. power factor
7. frequency

Users can select any combination of these metrics for included plots by providing the tag values in a comma-separated list. For example, to include both a plot of voltage and one of current along with the alert, the proper tag format would be "include | voltage,current." ViPERS is configured to automatically generate and include plots showing relevant data during the time of the event that caused the alert. Each alert plot therefore includes data from a short time before the alert became PENDING up until a short time after the alert started ALERTING to fully capture the event.

All alerts created through the ViPERS alert system are tied to a specific panel, and users may create only one alert per dashboard panel. To create multiple alerts for the same data query, simply duplicate the dashboard panel and create a new alert using the steps outlined previously. For users with many alerts, it may be worthwhile to construct a separate dashboard for holding all dashboard alerting panels.

4.4 MPM-Specific Modules

4.4.1 *Automated.py*

This module interfaces with the MPM configurations and calibrations created through the Autocal page detailed in Section 3.10. This module must be active to perform model-based calibration in the MPM. Changes to the MPM configuration through the Autocal page will start-up or restart this module with the configuration changes.

Once enabled, the *automated.py* module will actively determine that proper calibration for the selected cable model using the selected parameters and algorithm. The calibrated voltage and load (current) phasors will appear in the Live MPM Data subpage and can be viewed in a dashboard if a dashboard page has been built to interface with the *mpm_cal* database.

IMPORTANT: If new sensor calibrations are created through the Autocal:Calibrate page (Section 3.10.2), the user must manually restart the *automated.py* module to apply the updated sensor calibrations.

4.4.2 *Lowpower.py*

This module manages the LPM design. Users can configure this Dataserver module through the ViPERS Low Power page (Section 3.12). LPM initiates an implementation of the modular design of the MPM and independent Cable Sensor operation to balance the trade-off between energy consumption and data resolution. In LPM, the ARTEMIS processing architecture is put to sleep for some duration of time, *X*, during which the Cable Sensors continue independent, duty-cycled operation. The ARTEMIS sleep duration, *X*, additionally affects the Cable Sensor duty cycling, as the Cable Sensors are configured to take 112 evenly spaced measurements throughout the duration, *T*.

With this design comes some unique communication challenges between ARTEMIS and the Cable Sensors. To process and store the data collected in LPM, the Linux OS needs to reliably communicate with the Cable Sensors to toggle them in and out of LPM operation, read the data they collect, and adjust the Cable Sensors duty cycle to take evenly spaced measurements based on the duration, *T*. Additionally, ARTEMIS needs to manage its own wake up and judge whether to continue to execute in LPM or to switch to standard operations. There is also the challenge of synchronizing measurements between the independently operating cable sensors. Synchronous measurements between all eight data channels are necessary to produce accurate voltage and current estimates with a linear calibration matrix. This section is divided into three sub-sections to cover the following: software architecture of the LPM API on the MPM, the one-wire communication protocol used by ARTEMIS to communicate with the cable sensors, and the synchronization of measurements across cable sensors in LPM operation.

LPM has been demonstrated in laboratory testing at 5.4% total vector error compared to standard MPM ARTEMIS measurements in typical operation. The same calibration matrix can be used to transform both LPM and Standard Power Mode data, producing current estimates within 1A RMS error of the true current. During long-term testing in LPM, the MPM operated unassisted for approximately 2 weeks.

To further reduce power, a *shallow sleep* operation has been included in the MPM sensor boards. In this way, the MPM can continue to provide essential power measurements at resolutions that still exceed many power metering technologies during long-duration system sleep cycles. During the wake period, intermittently

collected high-resolution (time and dynamic range) data can be used in coordination with the lower-resolution data provided by the on-sensor processing.

These features have been developed and implemented to meet the goal of 2-week operation. To reach this goal, LPM configuration should be set so that the LPM cycle is exactly 30 min. During each cycle, ARTEMIS sleeps for 29 min and 30 s during which the Cable Sensors take a measurement every 16 s. The full system will then wake up and run for 30 s gathering high-resolution data and collecting data from the sensor boards. In total, the fully operating system is awake for approximately 1 min every hour. A plot of the battery voltage during this 2-week assessment is shown in Fig. 59.

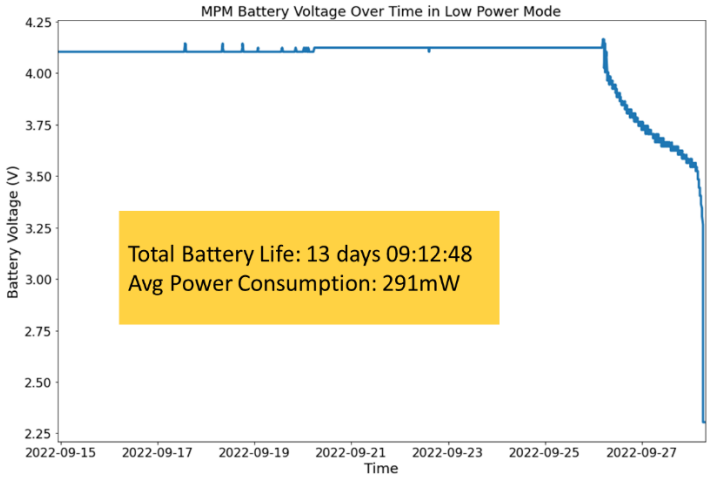


Fig. 59 MPM battery voltage slowly drops over a nearly 2-week test

In the case of more conservative LPM configurations and additional solar energy harvesting, the MPM will run unassisted in LPM for far past the 2-week specification. With the cited configuration, the MPM consumed an average of 291 mW. Assuming an average of 5 h of full sun every day, the 1.6-W solar panel can generate approximately 8 W of total power throughout the day, while the MPM consumes only 6.9 W per day. In this configuration, sustained operation may be possible even in cases where multiple days of limited sun occur. For further information on the capabilities and programming of MPM Low Power Mode, see its associated paper.[11]

5. Conclusion

ARL-ViPERS software has been developed as a multipurpose utility to provide an easy interface to ARTEMIS-based system information and control, data processing through modules running in the ViPERS Dataserver, and custom web pages allowing interaction with raw and phasor-based data. The interface can be used both with traditional sensors (e.g., current clamps) and smart sensors (e.g., cable sensors) and interfaces with system data logging for binary data files and an active time-series database. A secondary web application (Grafana) is hosted through the ViPERS dashboard page to allow for easy user dashboard customization without more time-consuming and difficult design of application-specific ViPERS pages. This capability is fully embedded with the hardware, circumventing the need for software installation, and is preloaded on ARTEMIS systems provided by ARL.

6. References

1. Claytor K, George A, Drummond Z, Snellman A. ViPERS web-based interface for Internet of Things (IoT): implementation guide. Army Research Laboratory (US): 2019 Dec. Report No.: ARL-TR-8863.

2. Parks B, Snellman A, Drummond Z, Claytor K. DEVCOM Army Research Laboratory Visualization and Processing for Embedded Research Systems (ARL-ViPERS) Programming Manual. DEVCOM Army Research Laboratory (US). Report No.: ARL-TR-XXXX.

3. Parks B, Snellman A, Hull D. Army Research Laboratory mobile unattended ground sensor (MUGS) installation guide. Army Research Laboratory (US). Report No.: ARL-TR-XXXX.

4. Timescale Inc. c2023 [accessed 2023 June 1]. https://www.timescale.com.

5. PostgreSQL. The world's most advanced open source database. c2023 [accessed 2023 June 1]. https://www.postgresql.org.

6. Grafana Labs. Grafana. c2023 [accessed 2023 June 1]. https://grafana.com.

7. Parks B, Hull D. Army Research Laboratory Live Animated Multi Phasor (ARL-LAMP) analysis software. Army Research Laboratory (US); 2015 Sep. Report No.: ARL-TR-743.

8. Parks B, Snellman A, Chung H, Hull D. Army Research Laboratory mobile power meter (MPM) installation guide. Army Research Laboratory (US). Report No.: ARL-TR-XXXX.

9. Bird S, Canvan L, Hulsey C, Paprocki M, Rudiger P, Van de Ven B. Welcome to Bokeh. Bokeh 3.2.0 documentation. [accessed 2023 July 11]. https://bokeh.org/.

10. IEEE Standard 519-2022. IEEE Standard for Harmonic Control in Electric Power Systems (Revision of IEEE Standard 519-2014). IEEE; 2022 Aug 5. doi: 10.1109/IEEESTD.2022.9848440.

11. Snellman A, Parks B, Heintzelman S, Hull D. Continuous, non-intrusive electric field & electric power monitoring using smart low power mode switching for ARTEMIS MPM. MSS BAMS, 2022 Nov.

List of Symbols, Abbreviations, and Acronyms

ADC	analog-to-digital converter
API	application programming interface
ARL	Army Research Laboratory
ARL-ARTEMIS	ARL Autonomous Real-Time Electric-power Measurement and Instrumentation System
ARL-MPM	ARL Mobile Power Meter
ARL-MUGS	ARL Mobile Unattended Ground Sensor
ARL-ViPERS	ARL-Visualizing and Processing on Embedded Research Systems
CPU	central processing unit
DC	direct current
DEVCOM	US Army Combat Capabilities Development Command
DHCP	dynamic host configuration protocol
EP	electric-power
FFT	Fast Fourier Transform
FPGA	field-programmable gate array
GPS	global positioning system
HDD	hard-disk drive
IEEE	Institute of Electrical and Electronics Engineers
IP	Internet Protocol
LAN	local area network
LED	light-emitting diode
LPM	low-power mode
MAC	Media Access Control
MMS	multimedia messaging service
NATO	North Atlantic Treaty Organization
OS	operating system
PNG	portable network graphics
RAM	Random Access Memory

RMS	root mean square
ROCOF	rate of change of frequency
RTC	real-time clock
SD	secure digital
SMS	short messaging service
SN	serial number
SQL	standard query language
SVG	scalable vector graphics
TCP	transport control protocol
THD	total harmonic distortion
UI	user interface
VPN	virtual private network
WAN	wide area network

1 (PDF)	DEFENSE TECHNICAL INFORMATION CTR DTIC OCA
1 (PDF)	DEVCOM ARL FCDD RLB CI TECH LIB
4 (PDF)	DEVCOM ARL FCDD RLA LB A SNELLMAN Z DRUMMOND D HULL B PARKS