NALU **SCIENTIFIC**

NaluScope 0.22.3

ASoCv3 User Manual

Marcus Luck, Mitchell Matsumori-Kelly, Alvin Yang

Contents

1. NaluScope

1.1 Overview

NaluScope is an easy-to-use application designed to control and read from Nalu hardware. Its primary purpose is to allow the user to easily view and record waveforms that are fed into the Nalu eval boards. More advanced usage of the NaluScope application includes changing the configuration and register values on the fly. *Changes to these values should **not** be changed without consulting a representative of Nalu Scientific as it can put the board into an unsafe configuration causing damage.*

1.2 Features

Our software offers an array of powerful features that streamline your data capture and processing experience while providing flexible visualization options. With its easy-to-use interface and hands-off approach, you can capture data from multiple board models and control a stand-alone headless server. The software allows you to save calibration data and board configurations for easy access, and fine-tune your hardware with total control over all the settings both in FPGA and on the ASICs.

The hands-off approach ensures that data remains unprocessed unless the user specifically wants it, making calibration and postprocessing optional features. This allows for greater flexibility in handling data according to individual preferences. The software is capable of capturing data from multiple board models using various connection types, such as UART, USB3, FTDI, and gigabit ethernet (GbE). This versatility simplifies the process of working with different hardware configurations.

In addition to the oscilloscope-like visualization, the software also provides optional 2D visualization with both grid and pixel visualization of the channels. This enables users to choose the most suitable visualization method for their specific requirements. The built-in calibration feature ensures accurate data capture and allows users to save calibration data and board configurations along with the captured data for future reference.

By controlling a stand-alone headless server, users can capture and store data on different hardware platforms, providing more flexibility and scalability. The software also allows for fine-tuning of the hardware settings, granting users total control over their equipment for optimal performance.

1.2.1 Key Features

- **Easy-to-use interface:** Intuitive design for seamless navigation.
- **Hands-off approach to data processing:** Data remains unprocessed unless required by the user.
- **Optional calibration and postprocessing:** Choose when and if to calibrate and post-process data.
- **Data capture from multiple board models:** One software solution for various board types.
- **Supports multiple connection types:** UART, USB3, FTDI, and GbE connections available.
- **Oscilloscope-like visualization:** Familiar and easy-to-read data display.
- **Optional 2D grid and pixel visualization for channels:** Customize visualization according to preference.
- **Calibration capabilities:** Perform calibration when needed.
- **Control of stand-alone headless server:** Manage remote data capture and storage.
- **Store data on different hardware platforms:** Flexibility in choosing where to save data.
- **Save calibration data and board configurations:** Keep important settings together with captured data.

• **Fine-tune hardware settings with total control:** Optimize hardware performance as needed.

1.3 System Requirements

1.3.1 Hardware requirements

REQUIREMENT	MINIMUM	RECOMMENDED
OS	Windows 7 Ubuntu 18.04 CentOS 8 macOS (10.13 High Sierra and later) Fedora	Windows 10
CPU	Intel i5 4th gen (Hashwell)	AMD Ryzen 7 3rd gen.
Memory	2 GB	8 GB
Disk	1 GB (installation only)	
Disk Speed	50 MB/s sustained write (for gigabit ethernet)	

1.3.2 Supported operating systems

Our software is designed to be compatible with a wide range of operating systems, ensuring performance for users across various platforms. It is fully supported on Windows (7, 8, and 10), macOS (10.13 High Sierra and later), and all major Linux distributions, including Ubuntu, Fedora, and Debian. This broad compatibility allows users to enjoy the software's features and functionality regardless of their preferred operating system.

To ensure optimal performance, it is recommended that users keep their systems up-to-date with the latest OS updates and security patches.

1.4 License Information

NaluScope is licensed under the *GNU Lesser General Public License version 3* (LGPLv3). It is important to familiarize yourself with the terms and conditions outlined in this license.

Please visit https://www.gnu.org/licenses/lgpl-3.0.en.html for the full LGPLv3 license text.

2. Installation

2.1 NaluScope Installation

1. To download NaluScope please visit the downloads page.
2. Download the latest version of the software for your operating system.
3. Optionally, you may wish to download the standalone server as well. Please read the section below for more information.

Installation of the software is done using the distributed installer on Windows or using the compressed package on Linux.

> **Note**
>
> On Windows, the software is distributed as an installer executable. The installer is not signed and your antivirus program might issue a warning. This warning can safely be ignored.

It is recommended that the application is installed as a subfolder to the root directory on windows (or `$HOME` on Linux). This makes it easy to upgrade to a newer version and to remove the application when it is no longer needed. The default folder on Windows is `C:\Nalu_Scientific\NaluScope-X.Y.Z`

2.2 Standalone Server Setup

The standalone server is a separate standalone executable written in Rust which can be run separately from NaluScope. In combination with NaluScope it can be used as a DAQ for the hardware, while NaluScope acts as the user interface. Using the standalone server may be useful for some hardware setups, as it allows for interfacing with NaluScope over a network connection.

There is no installer for the server; it may be placed in any suitable folder. For convenience, it is recommended to download the executable to a folder where acquisitions will be stored. For information on how to integrate the server with your own scripts, please see the Appendix.

2.3 Third-Party Dependencies

If UART over USB is used or USB3 (UPAC96), the appropriate drivers should be installed from the FTDI website:

- USB3 Drivers
- UART over USB

Although the drivers may already be installed, it is recommended to download and update the drivers anyway to ensure the hardware works as expected.

3. Starting the Application

When starting the application the user is faced with the Startup dialog. The dialog is divided into two halves: the left side handles projects and the right side handles the connection with the hardware.
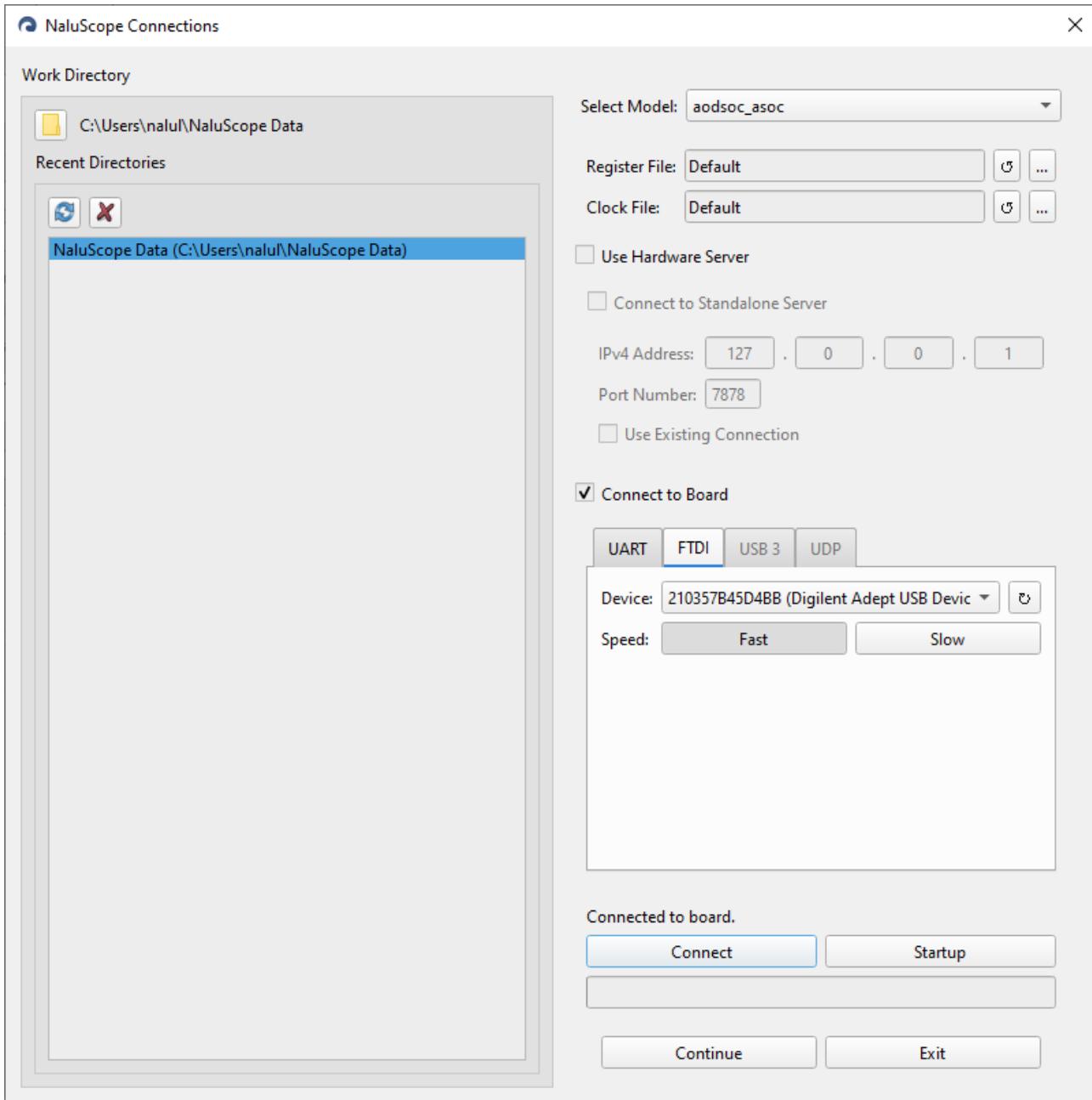

Figure 1. Startup dialog when opening NaluScope. The work directory, model, and device are examples and will not be the same for all users.

3.1 Select a Work Directory

On the left side are options to select a work directory. A work directory is essentially a workspace folder where NaluScope will default to saving and loading data. When the *Use Hardware Server* option is selected without *Connect to Standalone Server*, acquisitions will be saved in the selected work directory.

A work directory may be selected by clicking the folder icon. Previously-used work directories may be selected from the list. The default work directory is the last-used directory.

3.2 Connect to the Hardware

A board must be connected for most operations, including data capture. Previously-captured acquisitions may be viewed without a connection; this step may be skipped for this purpose.

> **Note**
>
> On Linux the port permissions may need to be modified to be used by the application. Visit the support website for more information.

1. First, select the type of hardware from the model dropdown list.
2. The default register and clock specification files will be loaded in, but custom files can be loaded in with the […] button. To reset the files back to the defaults, click the reload button.
3. If you wish to connect to a board over gigabit Ethernet (UDP), you will need to select the *Use Hardware Server* option. Depending on your network setup, you may wish to connect to a *standalone server* (see Section 3.4) running on a different machine.
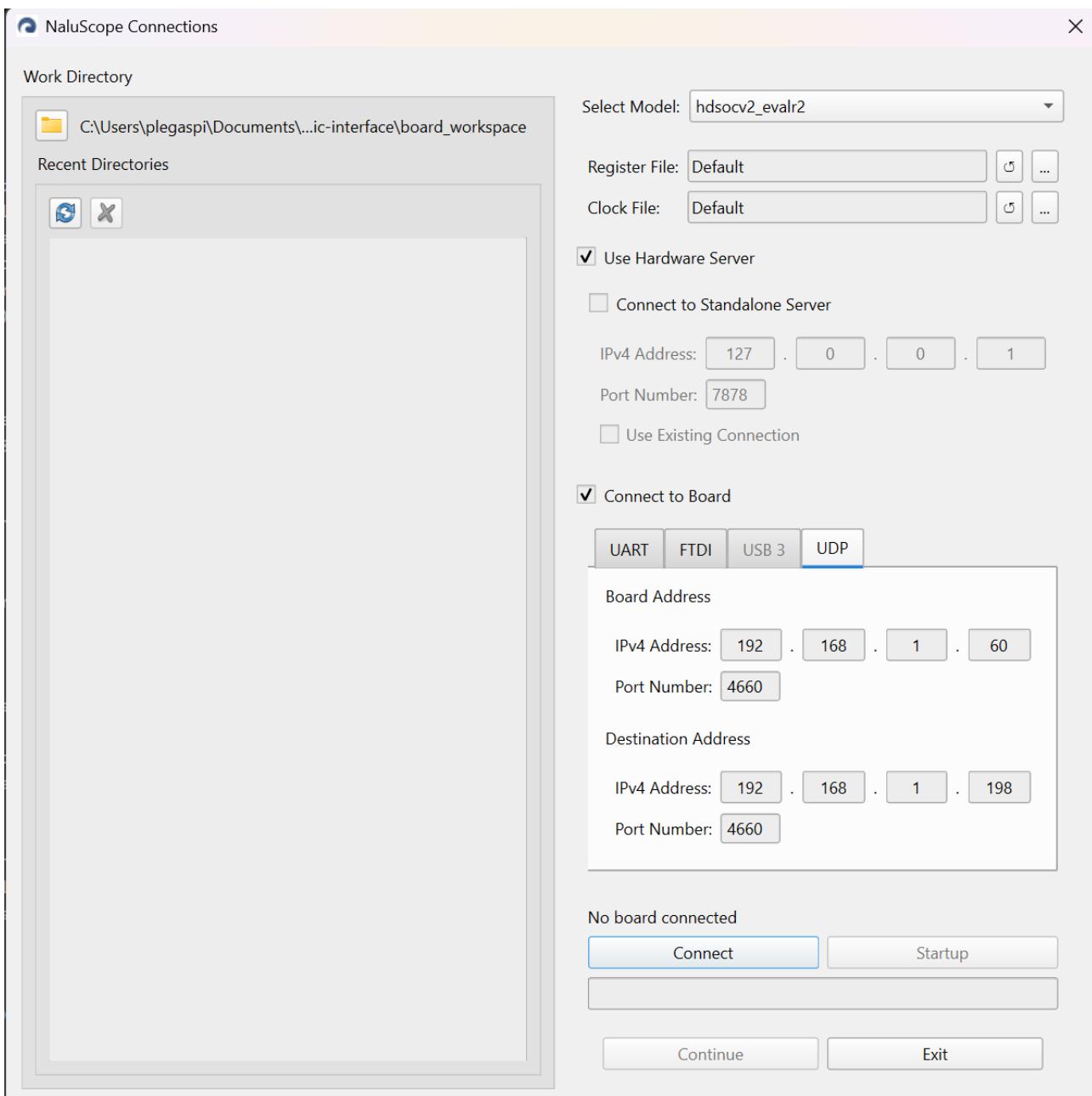
Figure 2. The Board and Destination IPv4 addresses must be provided to interface with your board.

- If connecting to a standalone server, check the *Use Standalone Server* option and enter the IP address and port of the server. The default port is 7878. If a stand-alone server is not used, a server will be started on the local machine on address `127.0.0.1` on an random available port.

- The *Destination Address* will typically be your computer's IP address. *See *Networking* in Section 8 to determine the aprropriate destination address.

Note

When connecting to an instance of the standalone server, you may select the *Use Existing Connection* option to restore the application state from a previous run. This allows you to exit NaluScope during capture and reconnect at any point. Note that for some board models, you may need to reinitialize the board when using this option.

1. Select the connection type and configuration to use for connecting to the hardware. If using UART or FTDI, the *Fast* option is typically preferred. You may also choose to skip connecting to a board to open the application for simply viewing data instead; see the section below for more information

2. Click *Connect* to create a connection to the board. Once a connection has been established, the board needs to be started. Press *Startup* to run the sequence. **The startup sequence can take as long as 1 minute on some boards.**

Troubleshooting

There are a few reasons why a connection can fail, please visit the support website for solutions. If a connection to the board is successful but the startup process fails, use the default register and clock files. If startup continues to fail, power down the board and try reconnecting with the default register and clock file.

Firewall

The first time NaluScope is started on Windows based machines, Windows Security will ask if you give NaluScope permission to communicate on the network. This is important since the application uses network sockets to communicate with the hardware.
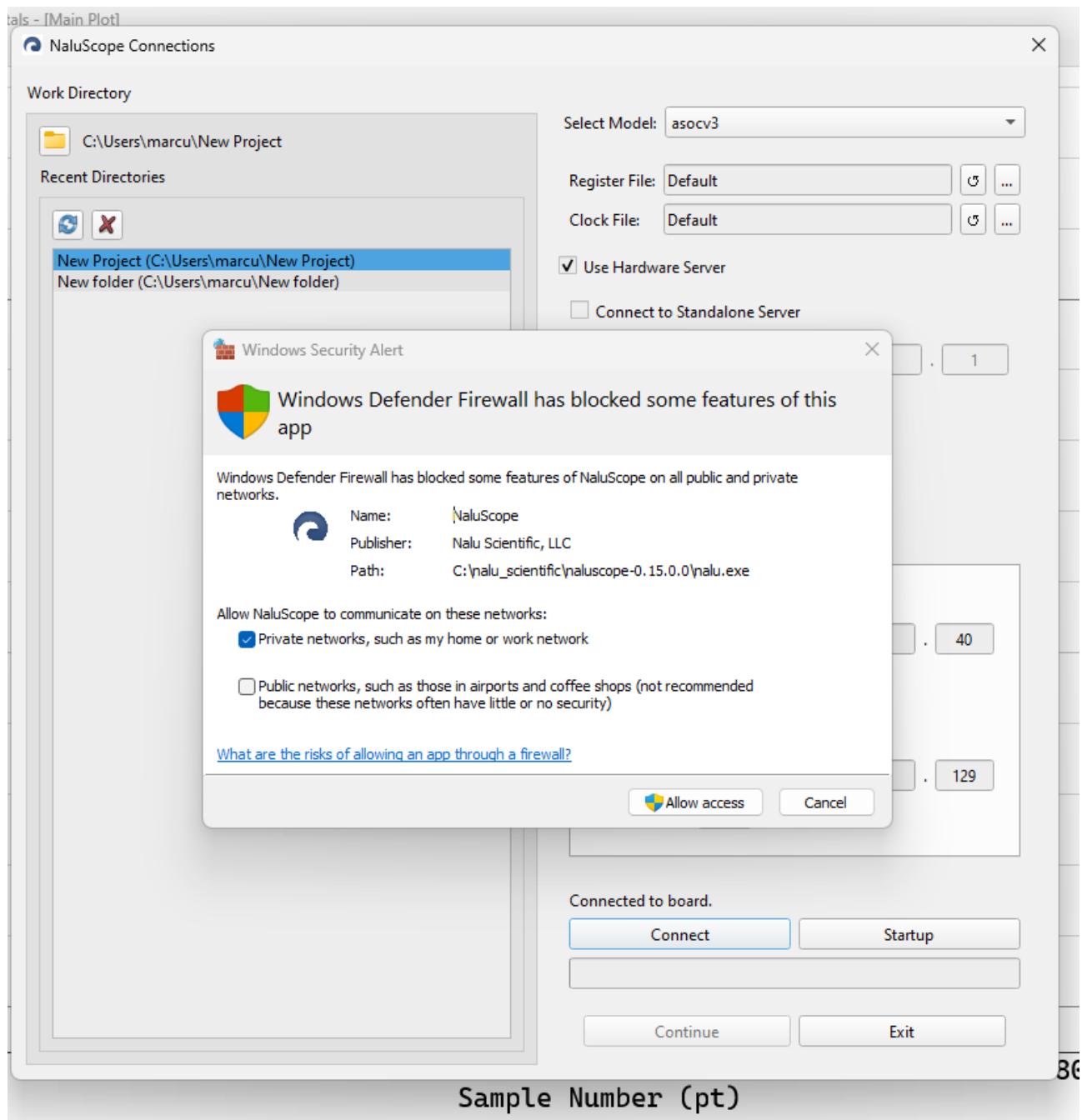
Figure 3. The Windows Firewall dialog, asking if you allow NaluScope access to the network..

3.3 Offline Mode
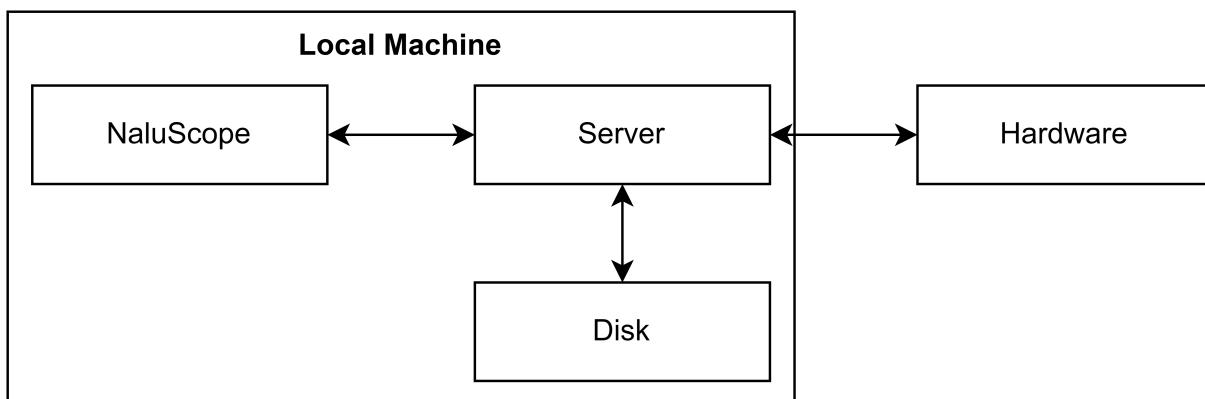
It is possible to start the application without a connection to view old data. To start the application in offline mode:

1. Select the work directory you would like to use.
2. Uncheck the *Connect to Board* checkbox. If you wish to view acquisitions captured while using the hardware server, you must also choose how you wish to connect to the server.
3. Select *Continue*

3.4 Standalone Server

The standalone server serves as an intermediary executable, facilitating performance-critical hardware operations and bridging the gap between the hardware and NaluScope. It is essential for utilizing gigabit ethernet connections and can also be employed for UART and FTDI (D2XX) connections. Moreover, the standalone server can be executed on either the same machine as NaluScope or a separate machine connected to the network, enabling remote hardware operation. The diagram below illustrates the use cases for the standalone server.

NaluScope Built-In Server


Standalone Server on Remote Machine
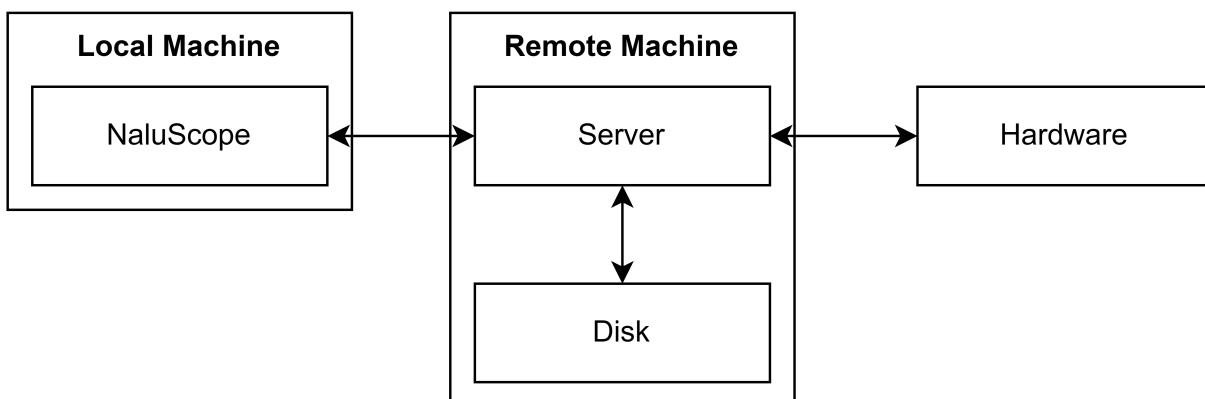
Figure 4. Server use cases: *(top)* starting the server locally through NaluScope and *(bottom)* running the standalone server on a remote machine.

3.4.1 Running the Server

The standalone server may be started either by double-clicking the executable or executing it from a terminal. If the executable is started without any arguments, it will start a server on the default address: `127.0.0.1` at port `7878` . The default output directory is the current working directory of the process. To run the application with additional arguments:

```
server_{version}.exe [OPTIONS]
```

Options:

```
 -a, --addr <ADDR>        The address to host the server at. Expected format is "{HOST}:{PORT}" [default:
 -o, --output <OUTPUT>    The output directory for acquisitions
 -d, --debug              Show debug messages
     --api                Open an interactive API in the system browser
     --log-dir <LOG_DIR>  The directory to store log files in
 -h, --help               Print help information
 -V, --version            Print version information
```

> **Note**
>
> The IP address `127.0.0.1` is the loopback address. If this address is used the server will only be visible to the local machine. To allow connections from other machines, the server must be started with the `--addr` option using an interface visible to the machine running NaluScope.

3.4.2 Data storage

When connected to a standalone or local server instance, data is saved on the fly to the output directory on disk. Each acquisition is stored in a subdirectory named with the start timestamp of the readout. See **Section 8.3** for more information on the data format.

> **Warning**
>
> Currently an acquisition created on a big-endian system will not be readable on a little-endian system, and vice versa. It is recommended to use a little-endian machine for the highest portability.

4. Application Overview

In the main window, you can capture and view events. The main parts are the toolbox on the left and the plot on the right.
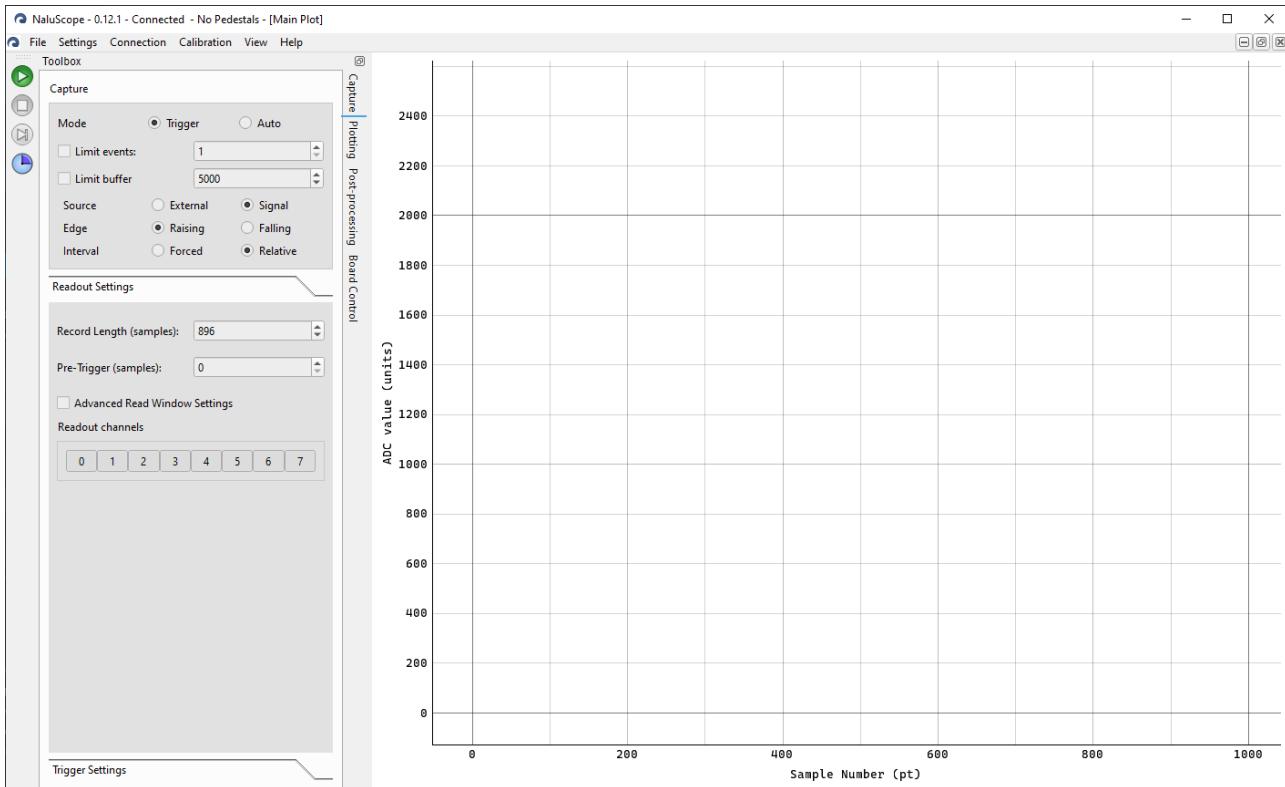

Figure 5. The main window, containing *(left)* the toolbox, and *(right)* the plot. The contents of the toolbox shown varies hardware models.

> **Note**
>
> The toolbox can be detached and moved, allowing the plot to stretch the entire window. Individual tabs from the toolbox can also be detached.

4.1 Menu Options

The menu options on the top left of the window provide additional features and configuration options. The following is a list of the options available under each menu.

- **File**
    - **Change Work Directory or Connection:** Reopens the startup dialog, allowing the user to change the working directory or connection settings
    - **View Work Directory:** Opens the work directory using the system file explorer
    - **Quit:** Exit the application
- **Settings**
    - **Reset:** Resets the board. This option is useful if the board locks up
    - **Reinitialize:** Runs the initialization sequence again. This may help if data cannot be captured
    - **Settings:** Opens the settings dialog
- **Connection**
    - **Reconnect Board:** Attempts to reconnect to the board. This option is only available when the board is disconnected
    - **Disconnect Board:** Disconnects the board. This option is only available when the board is currently connected
    - **Reconnect Server:** Attempts to reconnect to the hardware server specified in the startup dialog. This option is only available when disconnected from the server, and when "Use Hardware Server" was specified in the startup dialog
    - **Disconnect Server:** Disconnects from the hardware server. This option is only available when the server is currently connected
- **Calibration**
    - **Pedestals:** Contains options for generating and loading/saving pedestals calibration data
        - **Generate:** Shows a dialog for generating pedestals calibration
        - **Reset:** Remove the current pedestals calibration, if any
        - **Load → (format):** Loads pedestals calibration from a previous save file
        - **Save → (format):** Saves the current pedestals calibration in one of the available formats
    - **Threshold Scan:** Generates a trigger baseline by sweeping the input for noise
- **Help**
    - **Open Log Folder:** Opens the folder containing the log files using the system file explorer. Please use this if you are reporting a bug
    - **About:** Shows the about dialog

4.2 Quick Actions Side Panel

The side panel on the left-most side of the application includes several buttons which can be used to quickly perform common tasks, described below. The panel can be detached and moved around to a more convenient location if desired. The behavior of the actions is related to the selected triggering mode. See **Section 5.1** for more information.

4.2.1 Acquisitions

The green "play" button is used to start an acquisition. The acquisition will run using the current settings until the user stops it by clicking the stop acquisition (red "stop" button) button.

If using the standalone or local server options, the acquisition will immediately appear highlighted green in the *Acquisitions* tab upon starting a readout and will be updated in real-time. Otherwise, the acquisition will appear under the *Plotting* tab only after stopping the acquisition.

4.2.2 Triggering

There are two methods of issuing trigger events to the board when using the "External" trigger mode. The *Manual Trigger* (blue "forward/skip") button can be used to immediately send a trigger event to the board while an acquisition is running. The *Auto Trigger* (timer icon) button opens a dialog that can be used to automatically send trigger events at a fixed interval.

4.3 Keyboard Shortcuts

SHORTCUT	ACTION
Ctrl + Q	Quit the application
Ctrl + 1	Focus the main plot
Ctrl + 2	Focus the pixel view
Ctrl + 3	Focus the intersection visualizer

5. Tool Box

The toolbox contains several tabs, each of which contains a set of tools including capturing data, controlling the board, or manipulating and visualizing the data.

5.1 Capture Tab

In the Capture tab in the Toolbox section, there are options for event acquisition. The first choice to make is the Acquisition mode: Auto or Trigger.

Auto (Continuous mode) is used to capture data without a trigger. It will read out data in a stream and is an excellent way to capture background data and noise. Trigger mode will only capture an event when the trigger conditions are met. This is the primary mode to capture events. It requires the user to set up the trigger conditions correctly, or no data will be captured.
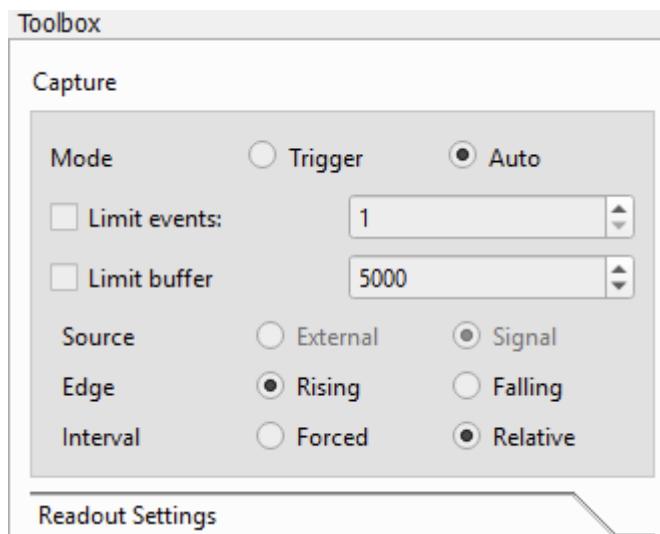

Figure 6. Capture events.

Limit events is an option to limit the number of events captured in an acquisition. In the current version of the software, all captured events are stored in memory until saved to disk.

Limit buffer makes the memory buffer circular, overwriting the older events with newer ones when the buffer is full. This is useful if there is limited memory on the computer or when running as an oscilloscope.

> **Note**
>
> A standard PC can capture millions of events before RAM is an issue

Start/Stop Acquisition to start and stop capturing data. When acquisition starts, it is impossible to change specific settings on the board, and those options are disabled. This feature will help the user to avoid accidentally changing the experiment parameters during the acquisition.
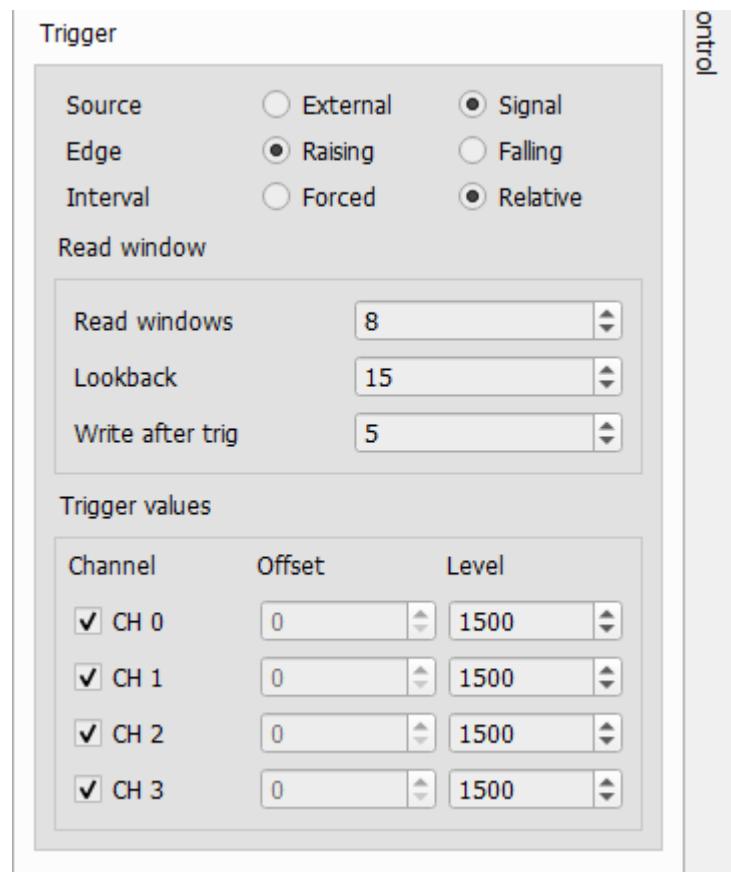
Figure 7. Trigger settings.

There are two types of sources for the trigger:

- External uses a separate connector on the board to receive a trigger and triggers when a pulse is received.
- Signal trigger on the received signal when it reaches a certain threshold and reads an event.

The trigger can be set to trigger on either the Rising or Falling edge of the signal.

The interval can be either:

- **Forced:** specific segments of the chip's internal memory are sampled after digitizing all samples.
- **Relative:** read out the event relative to the trigger.

> **Note**
>
> It is highly recommended to set the interval to *Relative* with this version of the board.

5.1.1 Trigger Settings

Figure 8. Set the trigger values.

It is possible to toggle which channel to trigger on and at what level. Offset changes the 0 point for the trigger. It is useful with a noisy signal to move the offset close to the signal value; the level will then be set relative to the offset. The level is the ADC value.

5.1.2 Read Window Settings

The read window specifies the region of interest to read out from the board. The settings may be configured to define a specific set of samples to read relative in time to the trigger event. The sampling array is a circular buffer, after reaching the end of the sampling array it will continue sampling starting at the beginning of the array.

The read window is set differently depending on whether the **Interval** setting is set to "Relative" or "Forced". More information on the difference between these modes can be found above in Section 5.1. Note that this mode is not recommended for general use as it is possible to get an event with a mixture of old and new data, but this mode may be useful for debugging purposes.

> **Note**
>
> A window (set of 64 samples for ASoCv3) is the smallest region that can be digitized. It is normal to see the trigger edge move within a window.

Normal Mode (Trigger-Relative)

Figure 9. Read window controls in trigger-relative mode.

In normal mode when using the "Relative" interval setting, two settings control where the read window is situated relative to the trigger event:

- **Record Length** - The number of samples to read out in total; this is the "width" of the read window.
- **Pre-Trigger** - The number of samples before the trigger event to begin recording at.

There are several notable uses for these settings:

- **Trigger event in the middle of the read window:** the pre-trigger value should be roughly half of the record length.
- **Observe signals before the trigger event:** the pre-trigger value must be larger than the record length.
- **Start the read window on the trigger event:** set the pre-trigger value to 0.

Starting the read window after the trigger event is not possible without using Advanced mode.

Normal Mode (Forced)
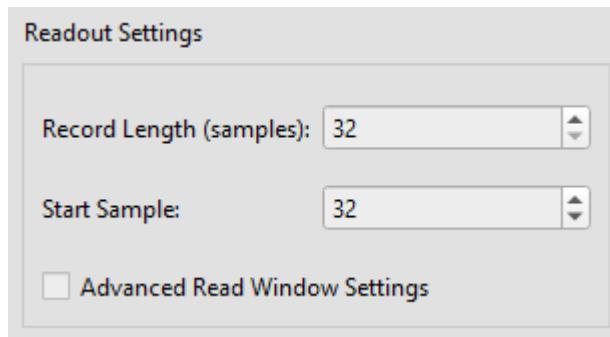
Figure 10. Read window controls in forced mode.

In normal mode when using the "Forced" interval setting, two settings control where in the sampling array the data is read from:

- **Record Length** - The number of samples to read out in total; this is the "width" of the read window.
- **Start Sample** - The index of the first sample to read out.

Advanced Mode (Trigger-Relative)
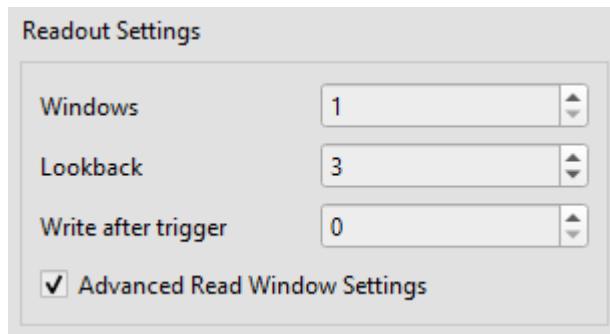
Figure 11. Advanced mode when using "Relative" interval.

Advanced mode provides full control over the read window settings.

> **Note**
>
> All settings are in terms of windows, rather than samples. One window is equivalent to 64 for ASoCv3 samples.

The following diagram explains the relation between the windows, lookback, and write after trigger settings relative to the trigger event:
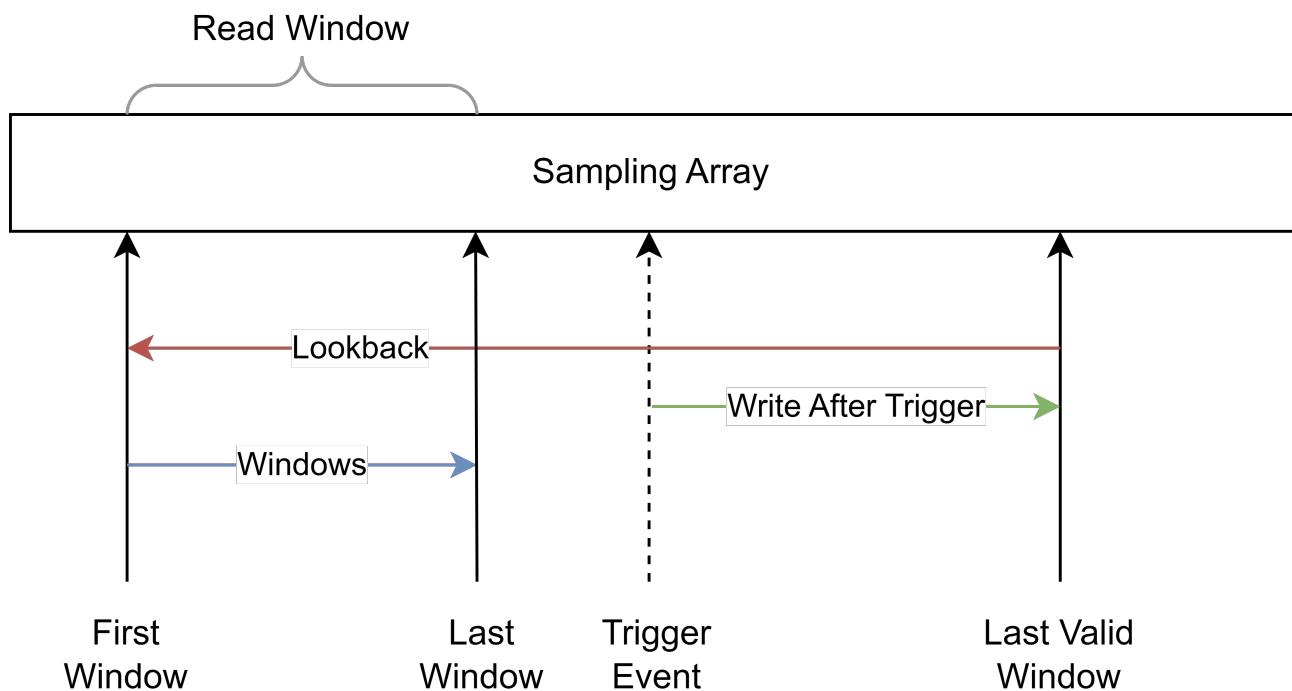
Figure 12. Advanced read window controls in trigger-relative mode.

The following settings are available:

- **Windows** - the number of windows to read.
- **Write After Trigger** - the number of windows to continue to digitize following the trigger event.
- **Lookback** - the number of windows to "look back" after digitizing "write after trigger" more windows.

Together, these settings define a start and stop point for the read window within the sampling array relative to the trigger event.

Advanced Mode (Forced)

In forced mode, *Lookback* becomes the start window relative to the beginning of the sampling array, and the write after trigger value cannot be configured.
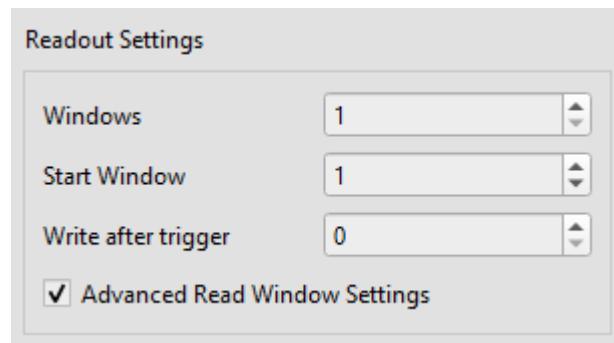
Figure 13. Advanced read window controls in forced mode.

5.2 Board Control Tab

The board control tab contains miscellaneous board-specific settings. For the ASoCv3, the only setting included is the ability to set the external DACs to bias the input circuitry.

The figure below shows the controls for the external DACs. The channels listed correspond to each of the input channels on the evaluation board. The values for each channel may be set individually or can be set all at once by highlighting multiple rows and using the right-click menu. The button at the bottom of the window allows all channels to be set at once.
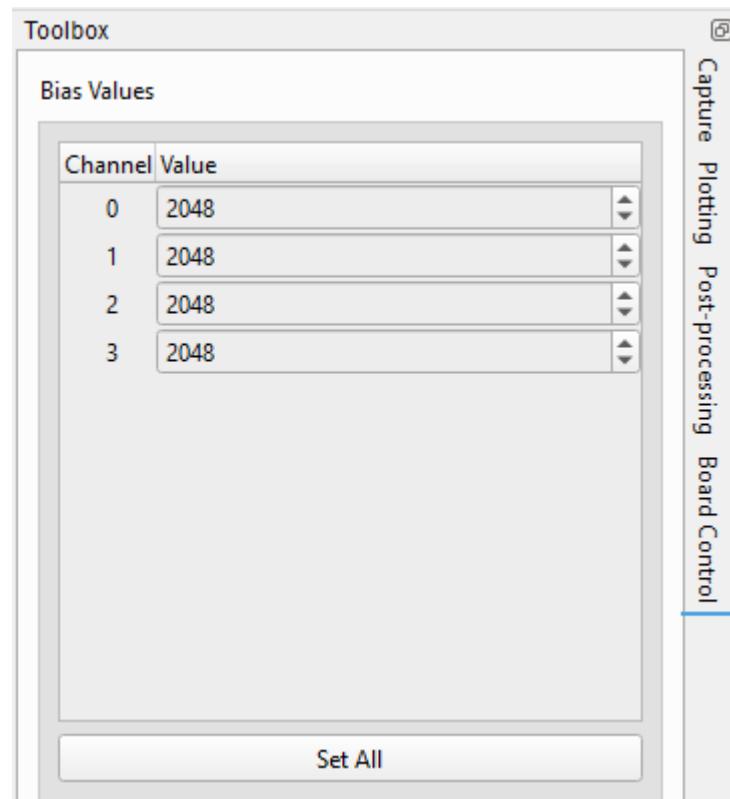
Figure 14. Controls for the external DACs. The number of channels shown varies between board models.

The range of allowed values for the external DACs on the ASoCv3 is from 0 counts (0 mV) to 4095 counts (2500 mV).

5.3 Plotting Tab

5.3.1 Plot Options

The box at the top of the tab provides options for how the plot is displayed.
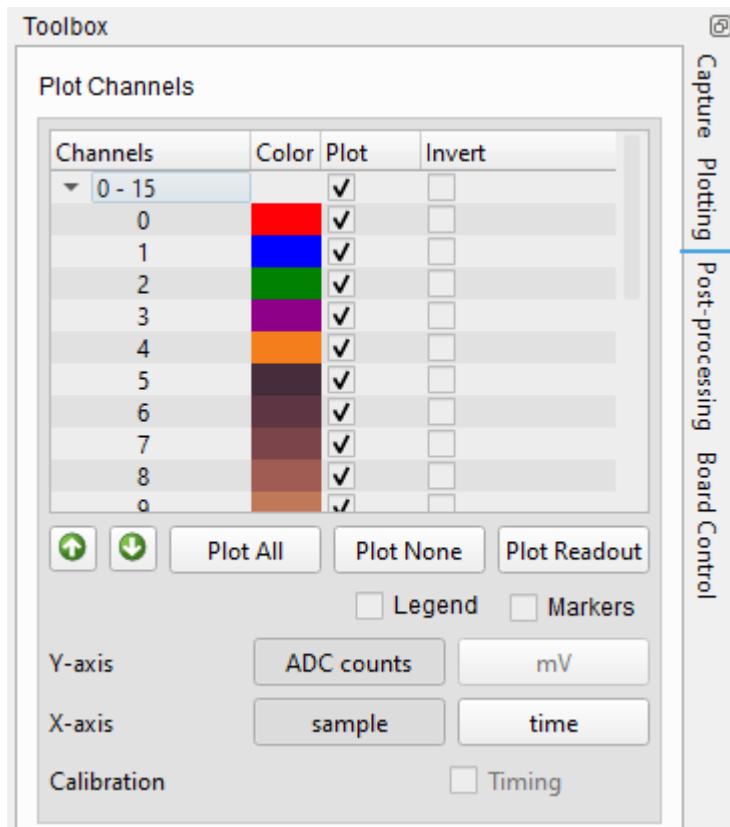
Figure 15. Toggle the channels to plot.

Channels can be enabled or disabled in the plot by checking or unchecking the checkboxes in the "Plot" column. When pedestals are loaded, specific channels may be inverted about the x-axis by selecting the "Invert" option. To toggle many channels at once, you may toggle the top-level row (labeled as "X - Y") or select multiple rows then toggle one of the selected checkboxes. Colors for individual channels may be modified by double-clicking a color under the "Color" column.

Sample markers and the legend can be toggled on or off using the respective checkboxes.

> **Warning**
>
> During a live plot sample markers may cause the application to slow down on boards with many channels. It is possible to encounter a higher rate of corrupt events in this situation unless using the standalone or local server, so it is recommended to disable sample markers when plotting live data.

Units for the *x* and *y* axis may be toggled using the respective buttons. Note that switching to mV requires *ADC to mV* calibration, and enabling timing correction are both beta features and are disabled by default.

5.3.2 Math Channel

The math channel allows the user to perform mathematical operations on the currently-selected event by evaluating an expression in terms of channels. The output of the expression is displayed as a new channel in the main plot window.
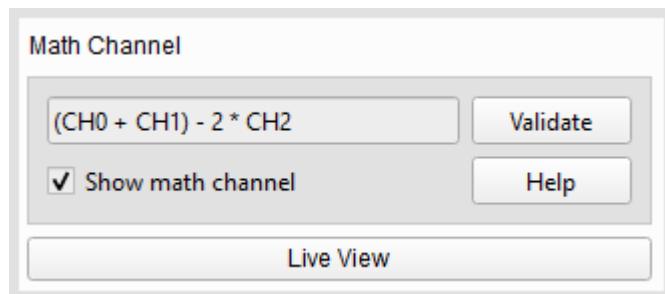

Figure 16. Math channel displayed with an example expression.

To use the math channel:

1. Enter an expression in terms of CH0 through CH3. Expressions support basic arithmetic & various functions and allow for the use of parentheses to group operations. For a full list of supported functions, click the Help button to display the help dialog.
2. Click the *Validate* button. If the expression is invalid, an error message will be displayed.
3. Enable the *Show math channel* checkbox. The math channel will be displayed in the plot window.

5.3.3 Live View

When capturing data the live view is turned on by default, meaning that the plot will update as new events are captured. This behavior can be turned off by unchecking the *Live View* checkbox. Turning off the live view may help to improve application performance in some cases.

5.3.4 Project Tree

> **Note**
>
> Legacy acquisitions are deprecated and will be removed in a future release. Please use the new acquisition format.

The project tree displays acquisitions captured in the legacy acquisition format. When an acquisition is complete, the captured events will be displayed in the tree. All acquisitions are stored in RAM, so to avoid potential data loss it is recommended to save the acquisitions to disk after capturing an acquisition.
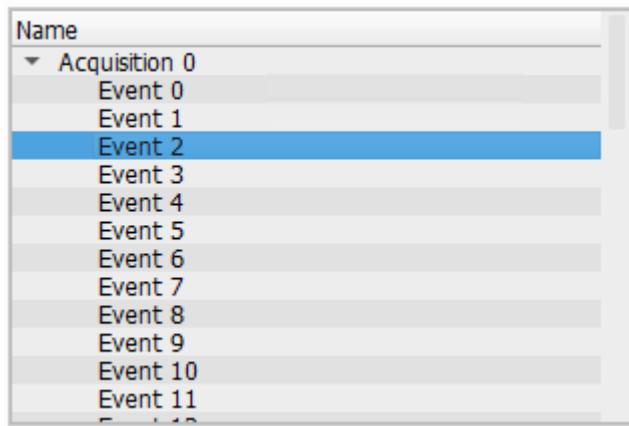
Figure 17. The project tree.

One event at a time may be plotted by left-clicking on it in the tree. To plot multiple events, select multiple events by holding the `Ctrl` or `Shift` keys and left-clicking on the events you wish to plot. Note that plotting many events will cause the application to freeze until rendering is complete.

When right-clicking an acquisition in the project tree, the following options are available:

- **Edit info:** opens a dialog allowing the user to view acquisition parameters and add notes to individual acquisitions.
- **Export selection:** export the selected acquisitions to various formats.
- **Load acquisition:** load a previously-saved acquisition.
- **Delete acquisition:** delete the selected acquisitions from the project tree, but not from the disk. If the acquisition was not saved to disk, it will be lost.
- **Export settings:** save the acquisition settings to a YAML file.
- **Pedestals:** contains options for plotting and saving the pedestals used when capturing the acquisition. This option is only available if pedestals were used when capturing the acquisition.

When right-clicking one or more events instead, a different menu is displayed:

- **Export selection:** export the selected events to various formats.
- **Load acquisition:** load a previously-saved acquisition.
- **Delete events:** delete the selected events from the acquisition. This will not modify the acquisition on disk until it is saved again.
- **Plot events:** plot the selected events.

Export Dialog

When exporting events, an export dialog will open and give the options to export in the following formats:

- Any number of events:
- `.acq` : Gzipped Python pickle containing information about the acquisition and event data.
- `.csv` : A CSV file with acquisition number, event number, window number, time, and data headers.
- Only single events:
- `.evt` : Gzipped Python pickle containing acquisition information and the parsed event.
- `.raw` : Gzipped Python pickle containing acquisition information and the raw event data.
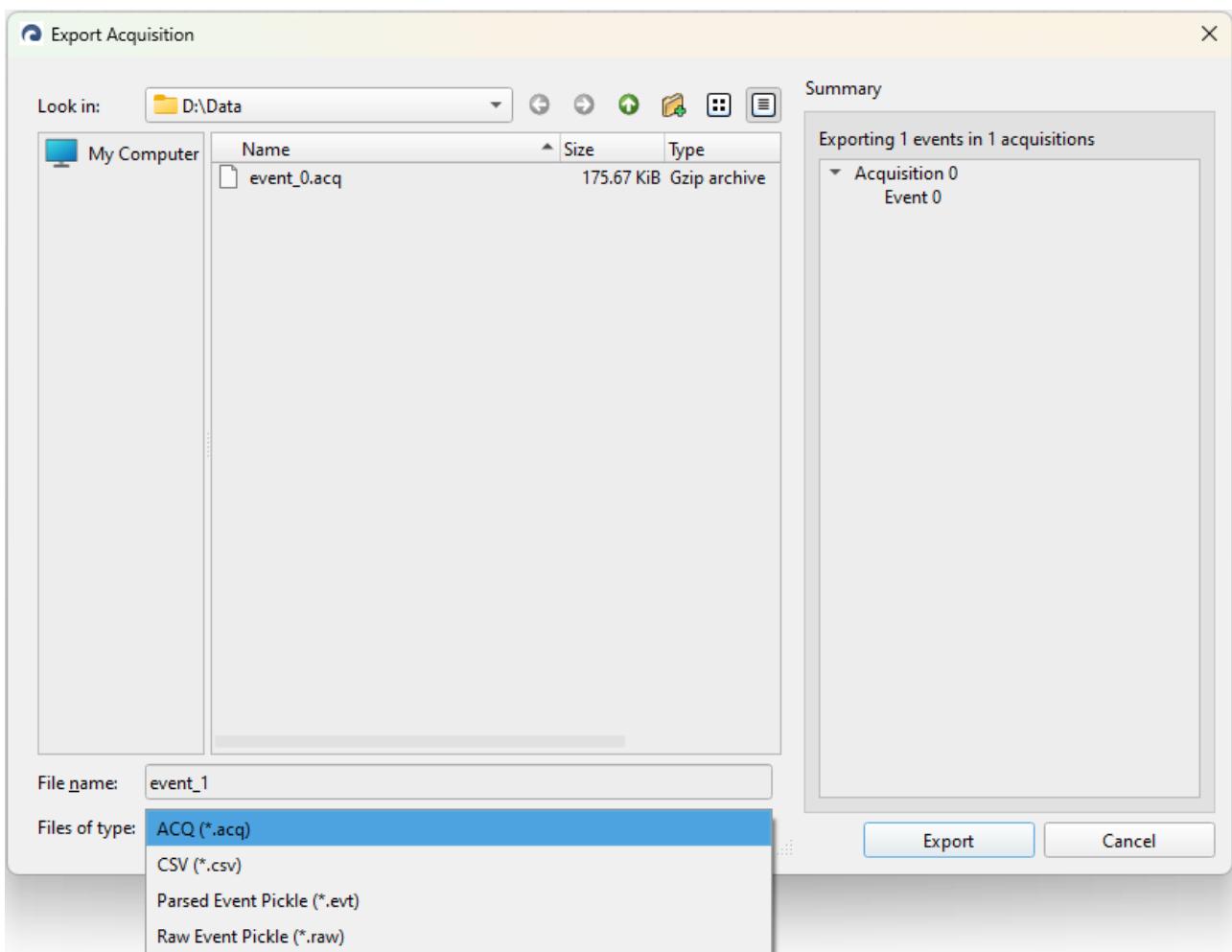
Figure 18. Export Dialog.

5.4 Acquisitions Tab

When using the application with a hardware server, the user has a new tab available labeled "Acquisitions". This tab contains all the acquisitions previously captured which are available to the server.

If the application was started with a local (non-standalone) server, all acquisitions in the project directory will be listed. For remote server connections, the working directory of the server process will typically serve as the "project directory" instead. There is no need to load acquisitions manually; the acquisition tab will automatically detect and load acquisitions in the project and will update itself when the project changes.

> **Note**
>
> Some features are currently restricted in this tab. For example, the options to export acquisitions or plot events are currently unavailable. Many of these missing features will be implemented in future releases.
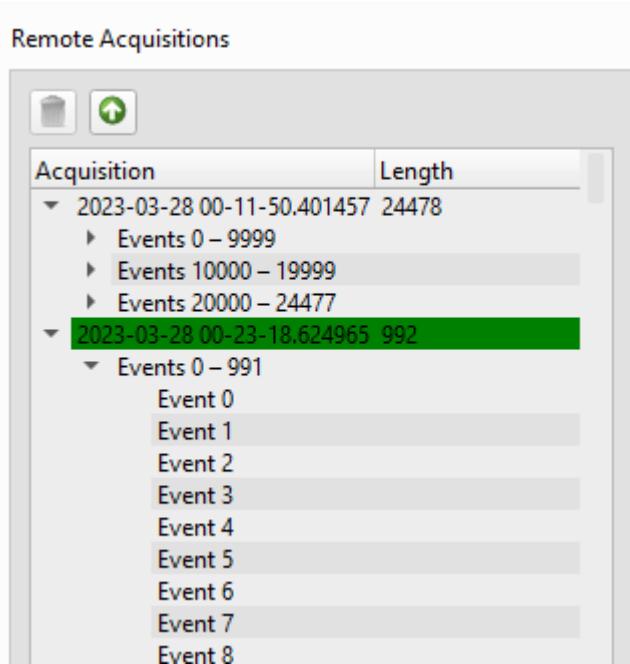
Figure 19. Acquisitions tab. The acquisition currently being captured, if any, is highlighted in green.

5.4.1 Plotting

Each acquisition has its events split up into chunks of 10,000 events for ease of navigation, as acquisitions captured using gigabit Ethernet can quickly reach hundreds of thousands or millions of events. To view an event, the user can navigate to a specific event and click on it. Multiple events may be plotted at once by selecting multiple events (currently limited to 10 events).

To view previously-captured events while capturing data, the user must first navigate to the "Plotting" tab and deselect the "Live View" option. The user may then navigate to any event of their choosing, including previous events in the current acquisition (highlighted green).

5.4.2 Manipulating Acquisitions

Data within an acquisition is read-only and may not be edited after capture. However, any acquisition except the current acquisition (highlighted green) may be renamed or deleted at any time. To rename an acquisition, the user may either use the right-click menu or use the F2 keyboard shortcut. Note that not all names are valid, as the name refers to the physical location of the acquisition on disk. To delete an acquisition, the user may either use the right-click menu or click the button with the trashcan icon.

5.4.3 Viewing Acquisition Information

Metadata regarding an acquisition may be viewed by right-clicking on an acquisition and selecting "View Details". The information which is presented includes:

- An overview of important information about the acquisition and the board
- A set of parameters holding information about the board
- A copy of the registers at the time the acquisition was first started
- A list of which types of calibration data are embedded in the acquisition

All the information presented in this dialog is *read-only*.

5.5 Post-Processing

The Post-Processing tab contains various functions and features which can be applied to data that has been captured.
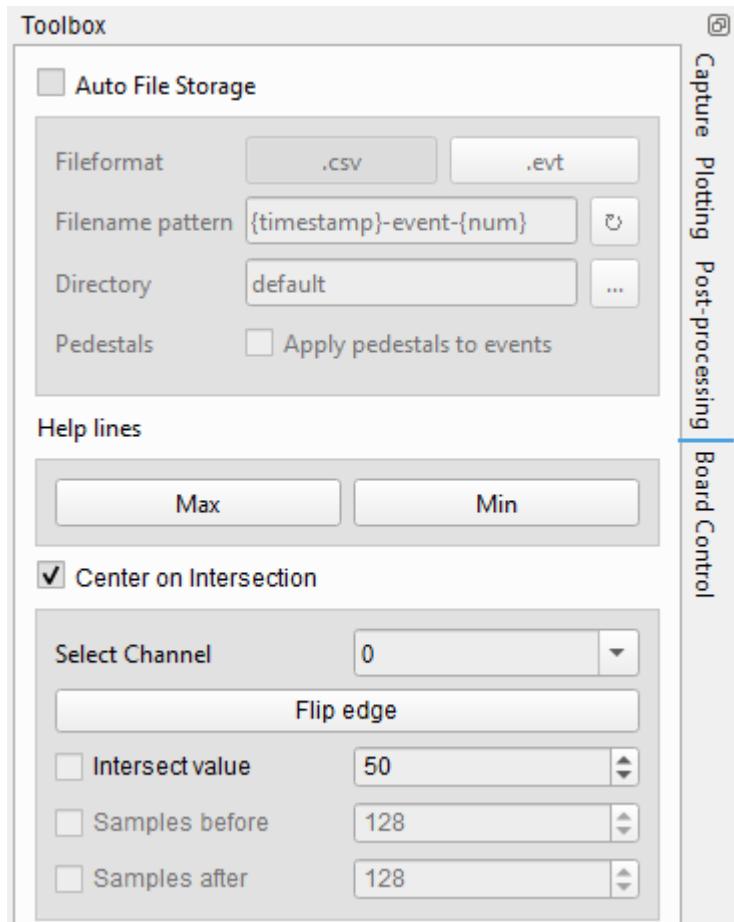
Figure 20. Post-processing options.

5.5.1 Auto File Storage

Enabling this option will automatically save captured events to separate CSV or EVT (Python pickle) files in the selected directory. The default directory, if not specified, is the current work directory selected when starting the software. To save pedestals-corrected events instead, enable the *Apply pedestals to events* option.

> **Note**
>
> This option is not available when using the standalone server.

> **Warning**
>
> This feature is unstable, and may not work as expected for all hardware models. If you encounter any issues, please report them using the method shown on the support site.

5.5.2 Help Lines

Additional help lines may be enabled in the main plot which display the minimum and maximum values of the data. These lines are useful for quickly determining the range of the data.

To enable or disable these lines, select or deselect the corresponding buttons.

5.5.3 Center on Intersection

Center on Intersection is a post-processing tool that shifts data horizontally along the time axis to align time *zero* with some condition in the data. This is useful for aligning data to a trigger signal, for example.

To configure the tool:

1. Select the channel to center on using the dropdown menu.
2. To only consider falling edges, enable the *Flip edge* option. Otherwise, leave the option unchecked.
3. Select a value to center on using the *Intersect value* input. This value is the value the function will use to check for intersection; this value can be approximated by examining the *y*-axis in the main plot.

The *Samples Before* and *Samples After* options are disabled in this release.

> **Note**
>
> If using this tool with previously captured data (not during live capture), you may need to change your currently selected event to see the effect of the tool.

6. Plot Area

The plot area is the right-hand side of the application which is used to display data in various ways. Each type of plot will appear as a sub-window in the plot area. The default plot is the main plot, which displays the digitized waveforms as a simple line plot.

6.1 Visualization Methods

In addition to the main plot, used for displaying simple waveforms, there are two additional visualization tools. The available visualizers are summarized in the table below and are explained in more depth in the following sections.

VISUALIZER	USE	SHORTCUT
Main Plot	Simple waveform plot.	Ctrl + 1
Pixel View	Easily identifying channels with low or high signal levels.	Ctrl + 2
Intersection Visualizer	Easily identifying channels with correlated signals.	Ctrl + 3

6.1.1 Pixel View

Pixel view represents events as a grid of channels in which each channel is a single "pixel." This visualizer is useful for quickly identifying channels with low or high signal levels.

© 2024 Nalu Scientific
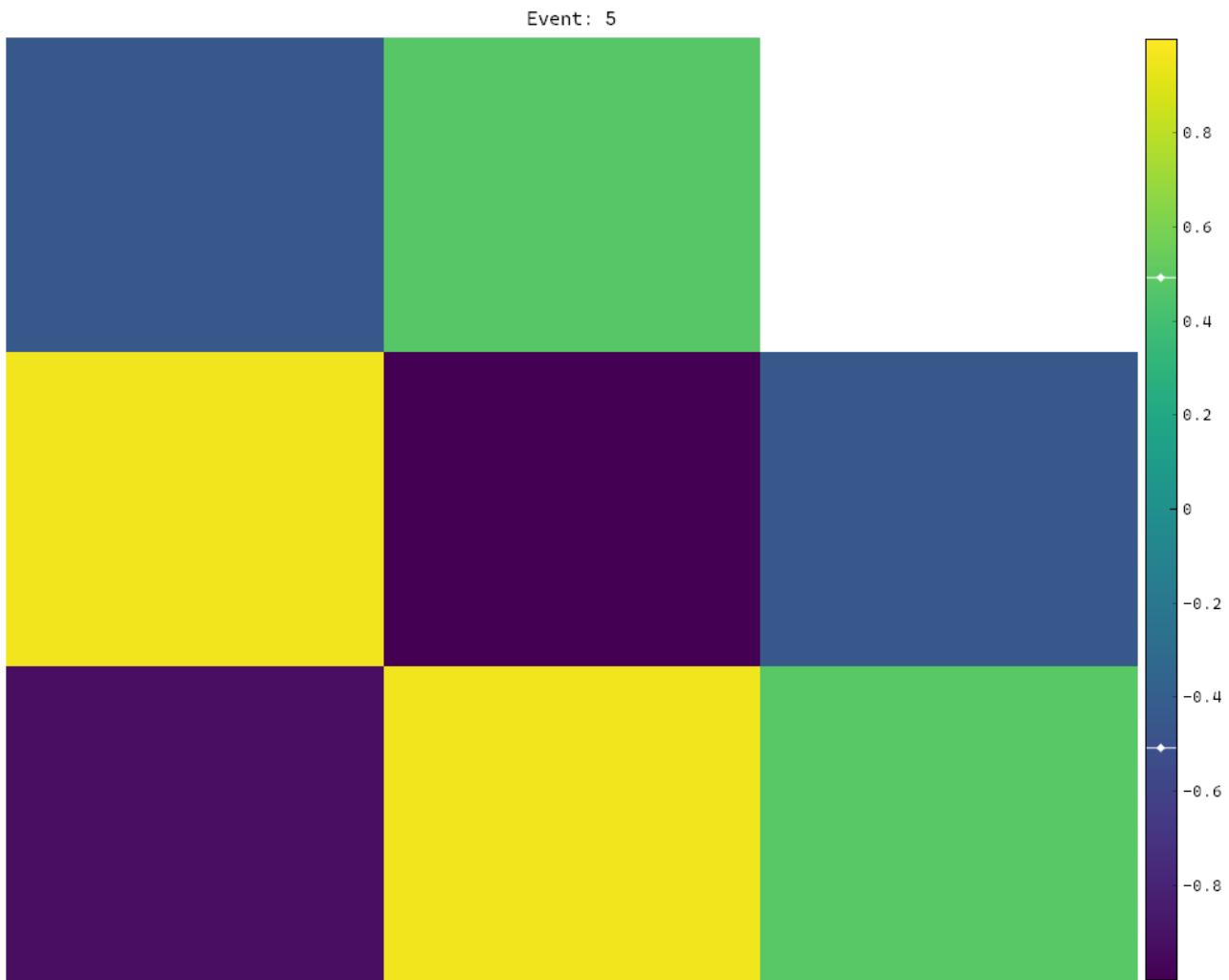
Figure 21. Pixel view displaying example data. The number of channels shown in the plot varies by board model.

See **Section 8.4** for more information on the algorithms used to generate the data displayed in the visualizers.

6.1.2 Intersection Visualizer

The intersection visualizer represents events as a grid of channels. Each channel is placed on either the *x*- or *y*-axis, and the intersections between each channel are colored according to the RMS values of the signals at that intersection. This visualizer is useful for quickly identifying channels with correlated signals.
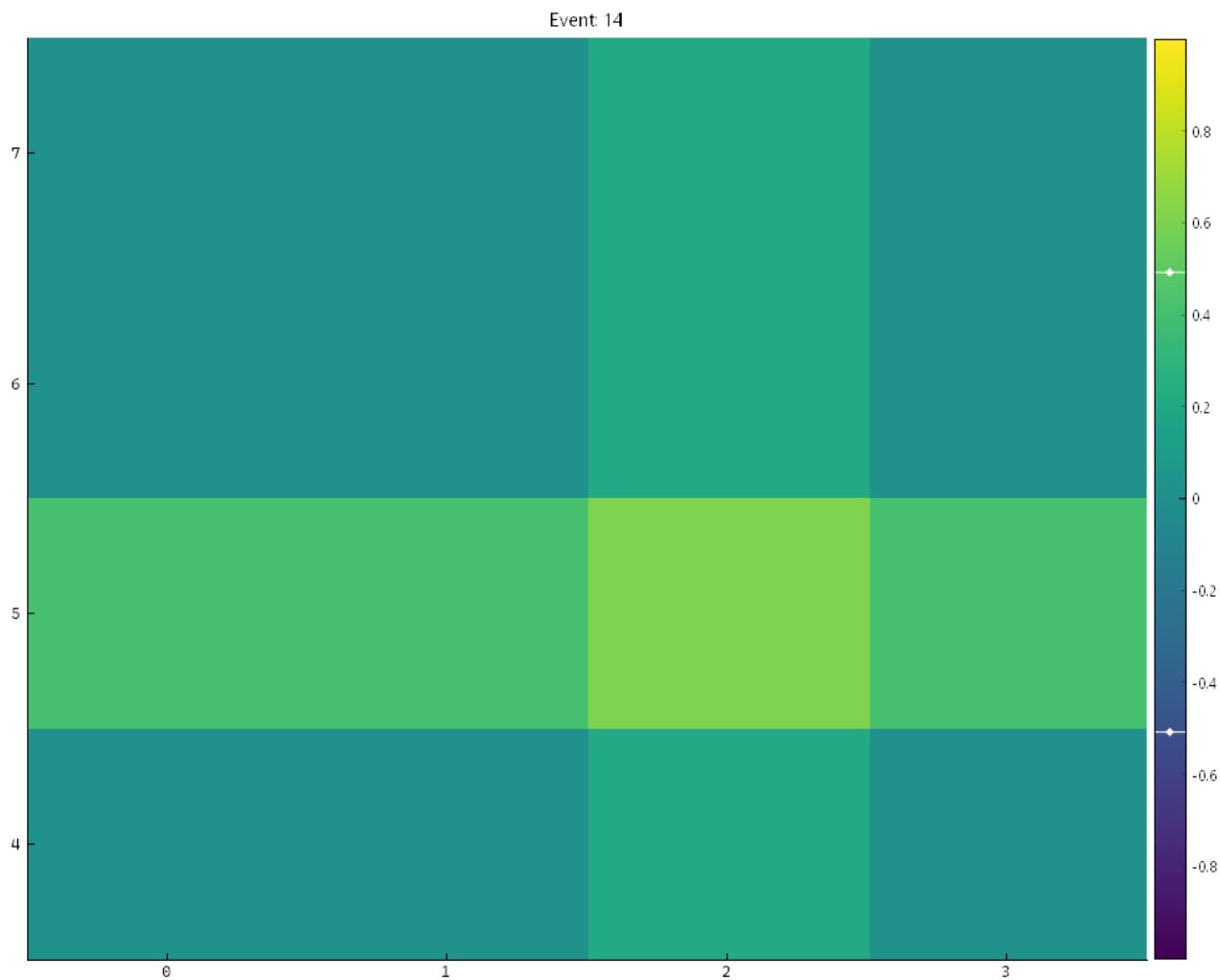
Figure 22. Intersection visualizer displaying example data. The number of channels shown in the plot varies by board model.

The channels displayed on either axis should be tailored to the experiment setup. To configure the channels:

1. Navigate to the *Settings* menu
2. Switch to the *Plotting* tab
3. Click the *Select 2D Visualizer Channels* button
4. Select or deselect the required channels, as shown in the example screenshot below.
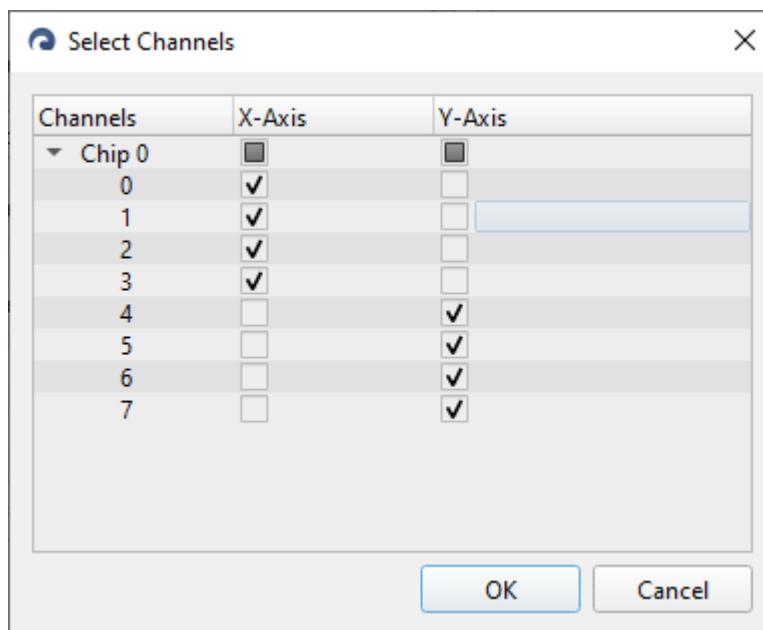
Figure 23. Example channel settings for the intersection visualizer. The number of channels available in this dialog varies by board model.

See **Section 8.4** for more information on the algorithms used to generate the data displayed in the visualizers.

6.2 Pedestal Plots

It can be useful to examine the generated pedestal calibration data to gain insight into data quality, which is not often apparent just by looking at waveforms. The *Pedestals Plots* feature allows the user to plot various aspects of the pedestals for this purpose. To use this tool, select the *Pedestals Plots* option under the *View* menu after generating pedestal calibration.
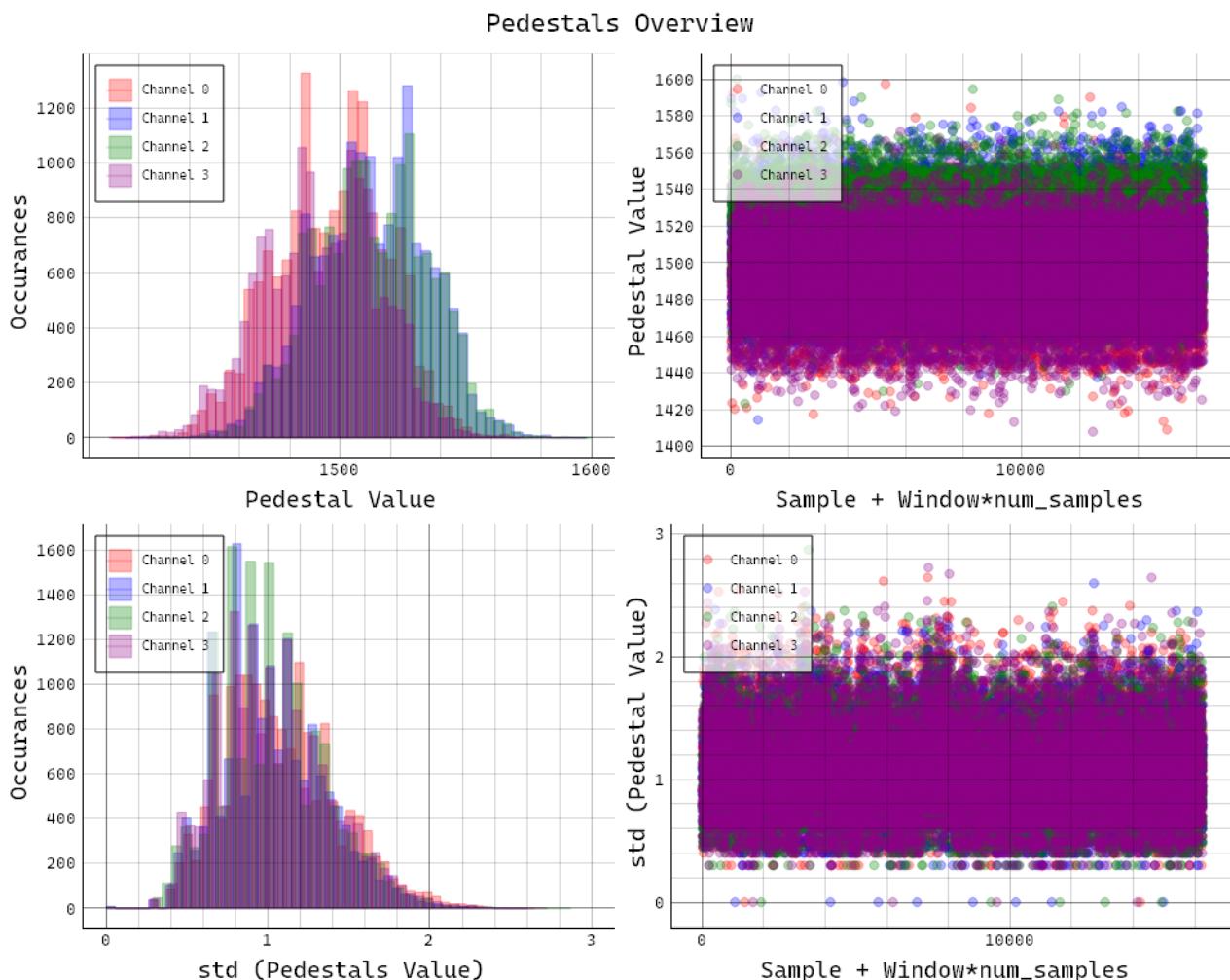
Figure 24. Pedestal plots using example data.

The pedestal plots window consists of four separate plots:

LOCATION	DESCRIPTION
Top left	Histogram of the pedestal values.
Top right	Pedestal values plotted according to location in the sampling array.
Bottom left	Histogram of the standard deviation of each sample.
Bottom right	Standard deviation of each sample in the sampling array

There are a few important things to consider when examining these plots.

- The pedestal values vary depending on the external bias, temperature, and humidity.
- The standard deviation of the samples should be as low as possible for quality data. Using too few warmup events, or forgetting to disconnect external signals when generating pedestal calibration can result in a higher standard deviation. The typical standard deviation is 1-2 ADC counts.
- Samples with zero standard deviation are most likely dead sampling capacitors. This is common and there is no reason for concern unless a large number of samples have zero standard deviation.

Settings for the pedestal plots, including outlier filtering, included channels, and plot appearance, can be accessed by right-clicking the plot area and selecting the *Settings* options. See **Section 8.1** for more information.

> **Note**
>
> The pedestal plots filter outliers using a modified Z-score algorithm. Samples with an absolute score past a certain threshold are removed from the data. Using the *Settings* window, the user may disable outlier filtering or adjust the threshold.

6.3 Mouse Controls and Plot Options

6.3.1 Zooming and Panning

There are two "mouse modes" which can be selected by right-clicking on the plot as shown below.
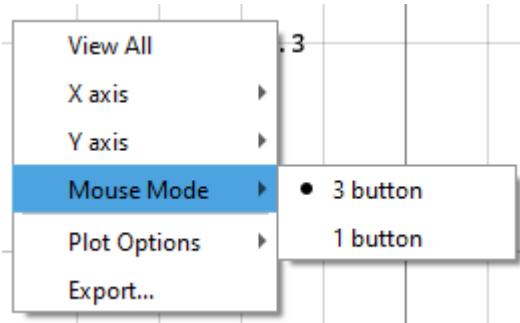
Figure 25. Mouse mode options.

In *3 button* mode (default), the user can pan the plot by holding the left mouse button and dragging the plot in the desired direction. Zooming may be done either by using the scroll wheel or by holding the right mouse button and dragging the mouse in the desired direction.

In *1 button* mode, the user can zoom and pan by holding the left mouse button and dragging a box around the area of interest. The plot will zoom to the selected region once the mouse button is released. The scroll wheel may also be used to zoom out uniformly.
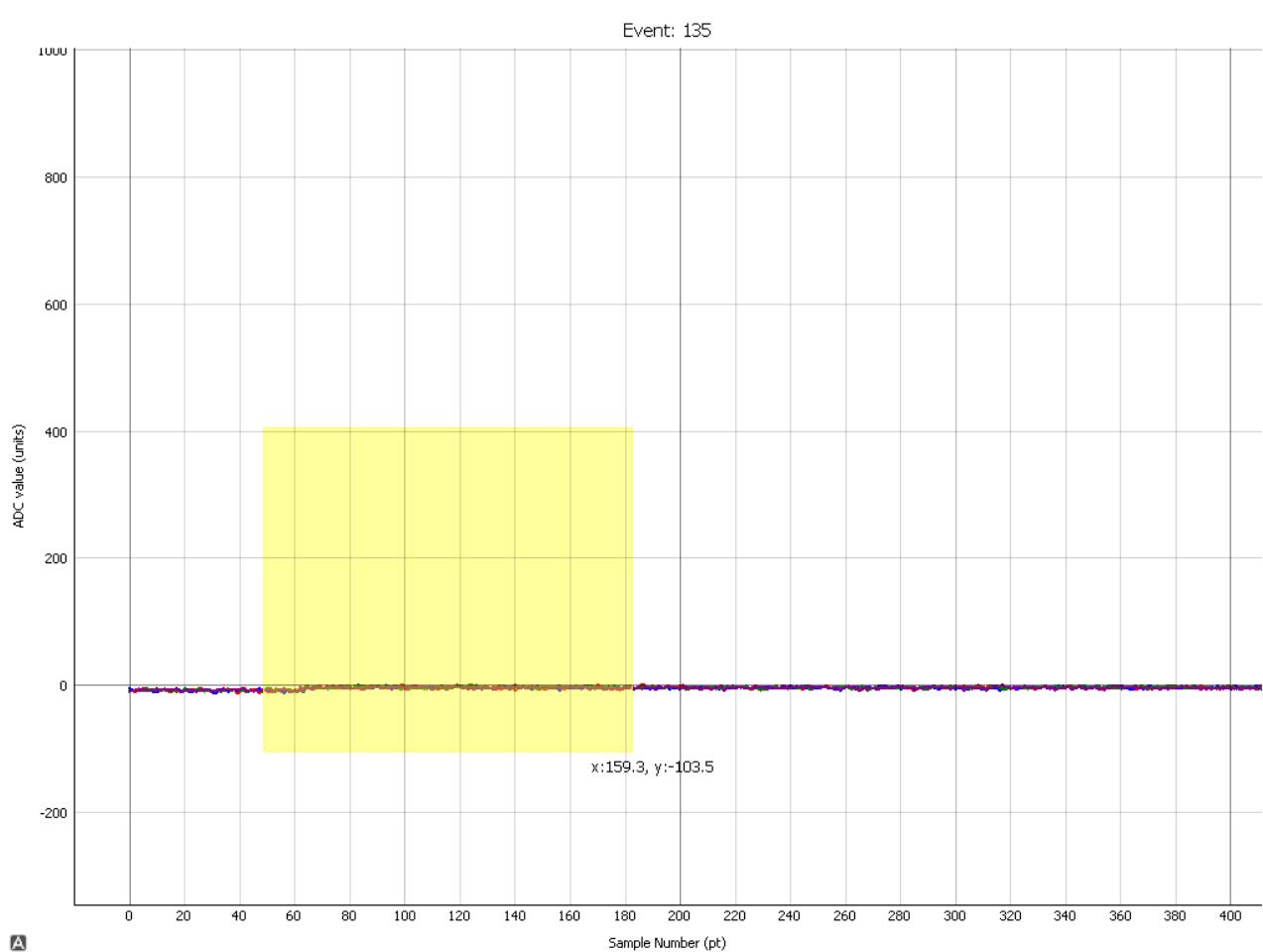
Figure 26. One button mouse mode zoom.

The axis options in the right-click menu offer additional methods of fine-tuning the axis scales and also allow the user to invert the axis direction, or disable mouse interaction to "lock" the plot in place along an axis.
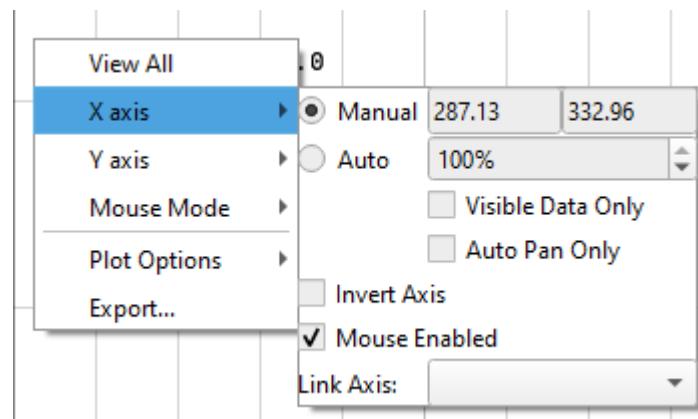
Figure 27. Axis options.

6.3.2 Plot Appearance

The *Plot Options* menu allows the user to change the plot type and configure the plot appearance.
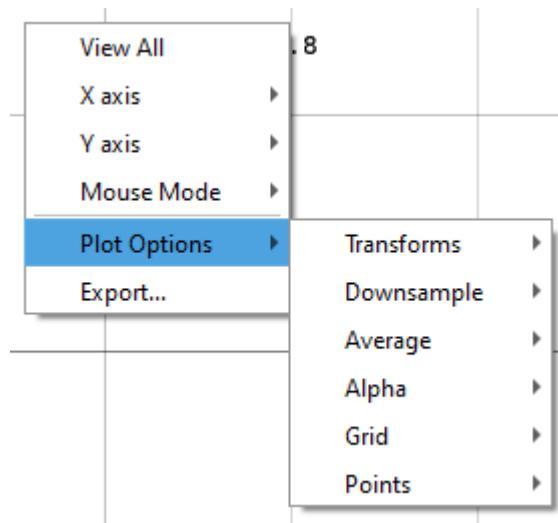
Figure 28. Plot options.

Note

The options to apply transformations to the data are currently not functional.

6.3.3 Exporting Plots

A plot may be exported to disk as either an image, `matplotlib` window, or vector graphics by using the *Export* option at the bottom of the right-click menu.
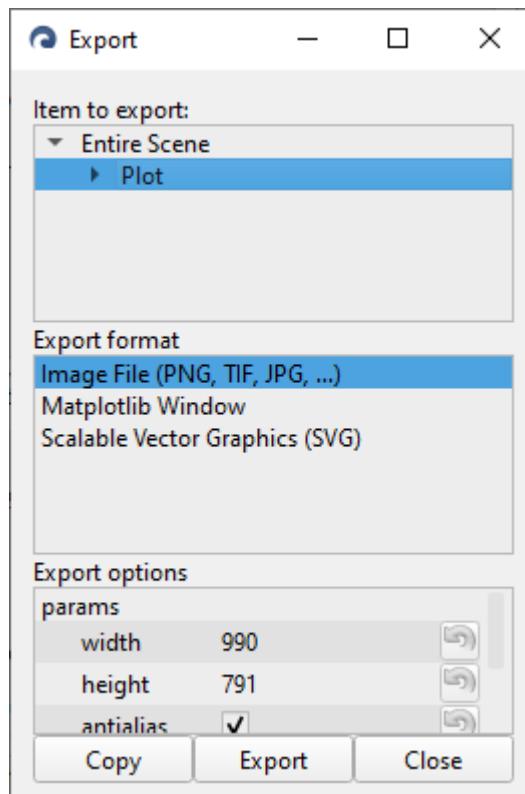
Figure 29. Export options.

© 2024 Nalu Scientific

The export dialog allows for the configuration of the exported region, image size, and background color.

Tip

It can be very handy to use the *Copy* button in this dialog to copy the plot to the clipboard, which can then be pasted into a document or presentation. The dialog can be left open if it needs to be used frequently.

7. Calibration

NaluScope provides several methods of calibration to correct and convert data. The various types of calibrations may be generated, loaded, and saved from the *Calibration* menu at the top left. Note that calibration methods that are currently in beta are not described in this manual.

7.1 Pedestals

Pedestals calibration is used to subtract the built-in bias levels of the sampling array. During the generation of the pedestal calibration data, many events are captured and the average value of each sample is computed. Parameters regarding how the pedestals are generated may be configured in the dialog shown in the figure below.
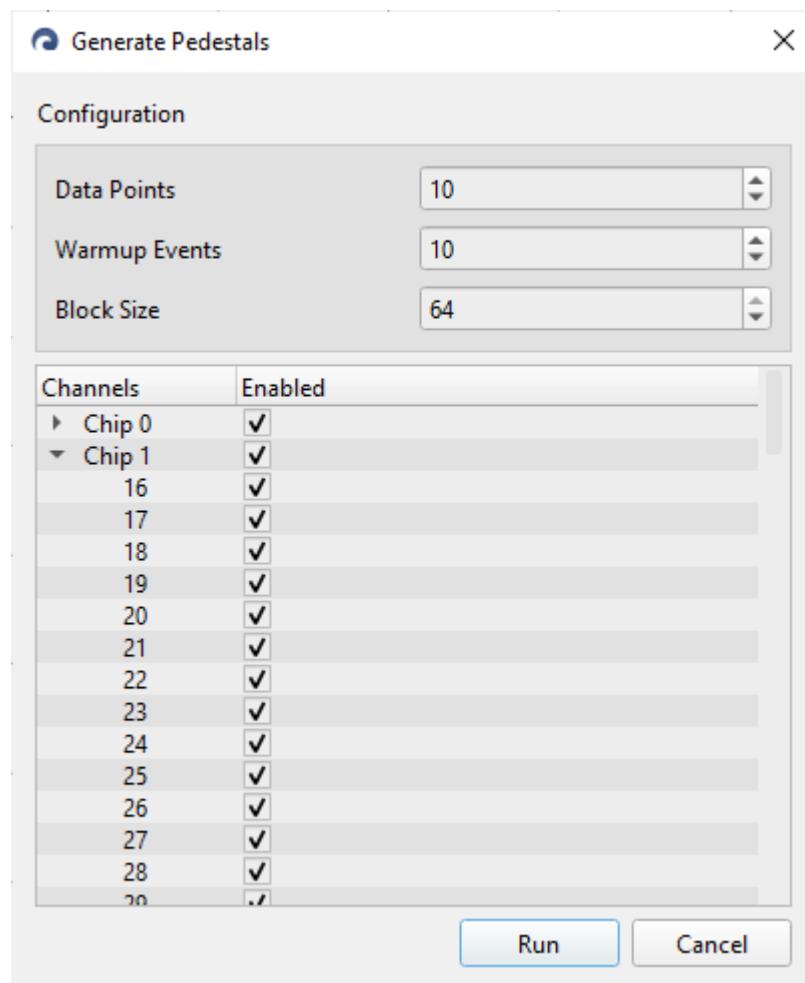
Figure 30. Pedestals calibration dialog.

Settings:

- **Data Points:** controls the number of events to capture per sample. A larger number of data points will result in a more accurate calibration but will take longer to generate.
- **Warmup Events:** controls the number of events to capture before starting to capture data points. All warm-up events are discarded and do not contribute to the calibration data. This is useful to allow the board to warm up and stabilize and reduce temperature-related effects.
- **Block Size:** controls the number of events to capture at a time. For some boards, the full sampling array cannot be reliably read out all at once, in which case the data must be read out in blocks. This parameter controls the size of each block. The block size has some impact on the time it takes to generate the calibration data.

Individual channels may be included or excluded from pedestal generation by selecting the checkboxes for each channel. Note that some board models, such as the UPAC96, do not allow individual channel control.

7.2 Threshold Scan

The threshold scan provides a method of determining the optimal threshold level for use with the signal (aka "self") trigger mode. After running a threshold scan, a plot will appear showing the number of trigger events at each threshold level. The optimal threshold level is the one which results in the highest number of trigger events.
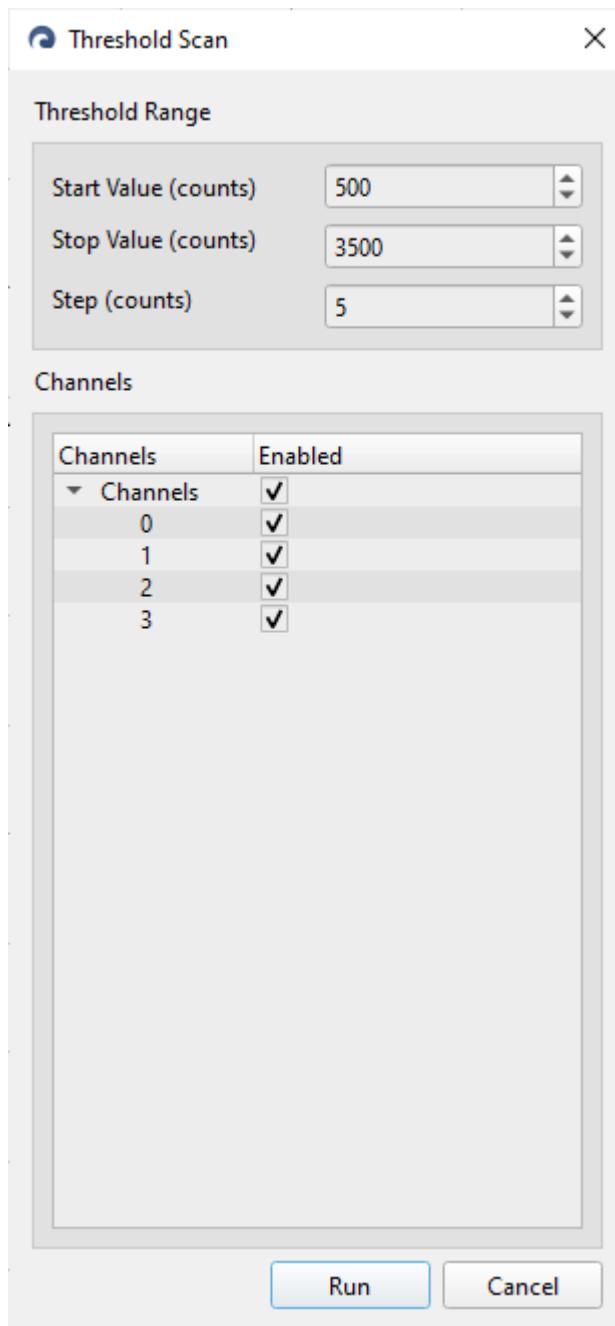
Figure 31. Threshold scan dialog.

Settings:

- **Start Value:** the value in counts that the threshold scan will start at.
- **Stop Value:** the value in counts that the threshold scan will stop at.
- **Step:** the value in counts that the threshold scan will increment by on each step. A larger step size will result in a faster scan but can miss the optimal threshold value. It is recommended that the step size be set to no larger than 5 counts on most boards.

Individual channels may be included or excluded from the threshold scan. Fewer channels will result in a faster scan.

Note

Due to temperature and humidity effects, a threshold scan should also be run if the environment changes or if previously valid trigger settings no longer yield the desired outcome.

8. Advanced Usage

This section discusses the advanced usage of the application and explains the methods used to achieve certain results.

8.1 Settings Window

The settings window, accessed through the Settings menu at the top of the main window, contains options to configure plotting and access the advanced functionality of the board.

8.1.1 Register Settings

> **Warning**
>
> Using these settings incorrectly can cause irreversible damage to the hardware. Please consult a representative of Nalu Scientific before using these settings.
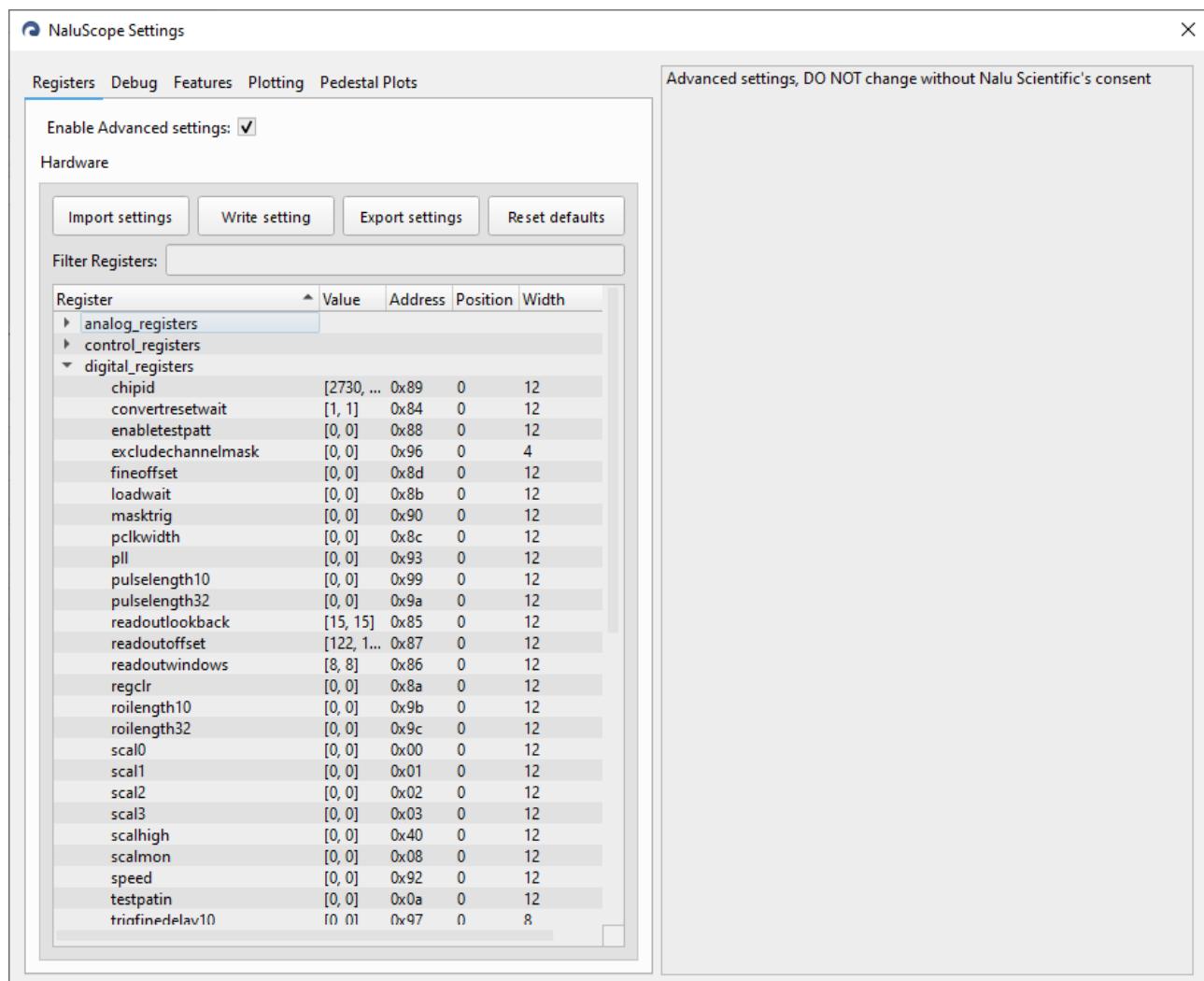
Figure 32. Register Settings.

The *Registers* tab provides methods to manipulate the hardware registers in both the FPGA and the chip.

- To write a register, double-click on the value next to the register name and enter the value in decimal you wish to write to the register.
- To read a register, right-click on one or more registers and select the *Read Register(s)* option.

There are also several buttons to help with manipulating the registers:

- **Import Settings:** allows the user to load registers from a YAML file.
- **Write Settings:** allows the user to re-write all registers on the board.
- **Export Settings:** saves the current register values to a YAML file.
- **Reset Defaults:** restores the default values for all registers.

> **Warning**
>
> Use of the *Load Settings*, *Write Settings*, or *Reset Defaults* buttons may require the board to be reinitialized. This can be done from the *Debug* tab or through the *Settings* menu.

8.1.2 Debug Settings

The *Debug* tab allows the user to read debug information from the board and perform operations typically not necessary to the user.
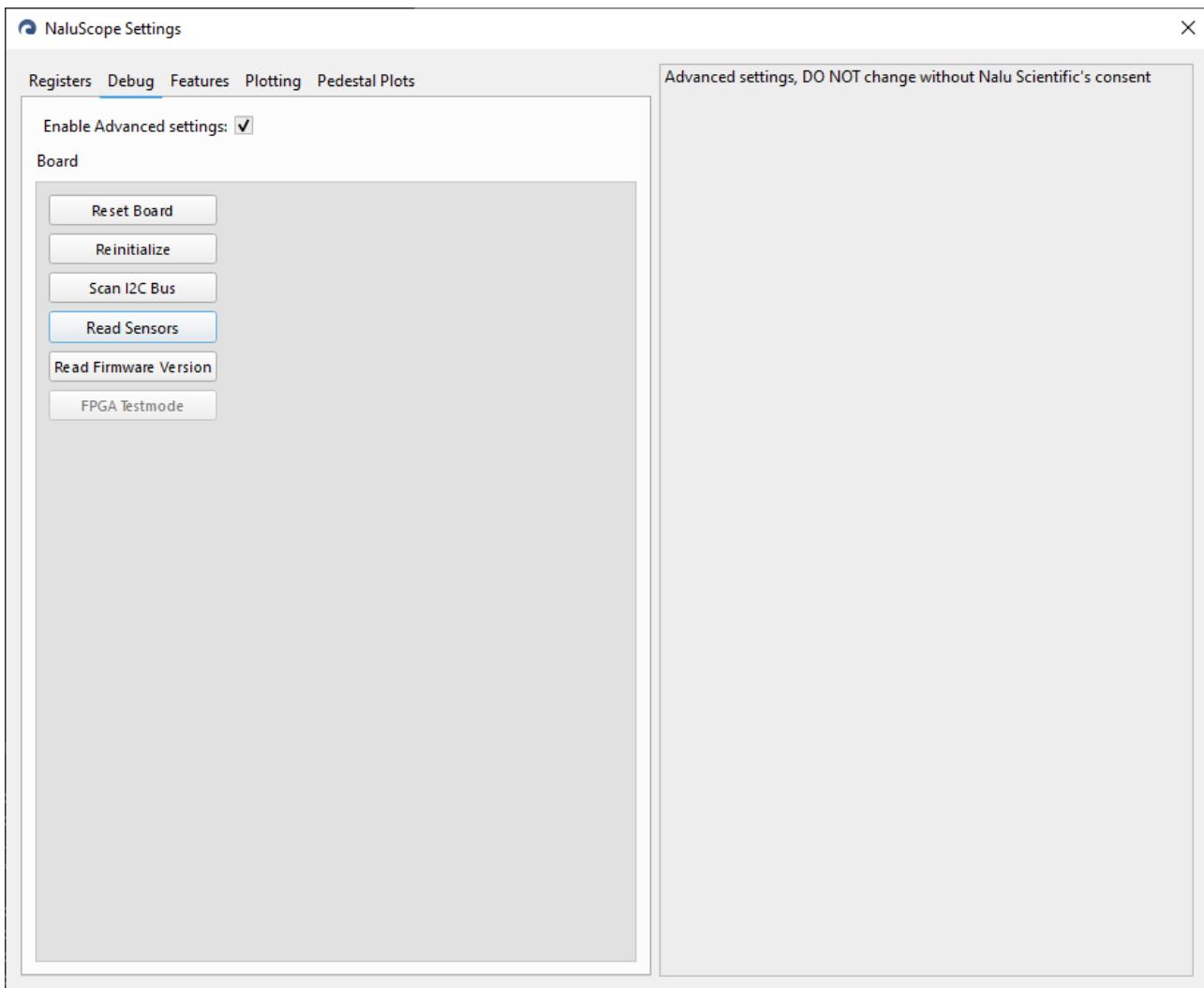
Figure 33. Debug settings.

Although you must first enable advanced settings to use these buttons, they are generally not destructive actions. The buttons are described below:

- **Reset Board:** can be used to get the firmware out of a bad state. This should be followed by a reinitialization.
- **Reinitialize:** runs the initialization sequence again. This may help if data cannot be captured.
- **Scan I2C Bus:** scans the onboard I^2C bus for devices. This is useful for debugging whether one of the onboard devices is not working.
- **Read Sensors:** reads the various temperature and voltage monitors on the board. The availability of certain sensors depends on the board model.
- **Read Firmware Version:** Reads the firmware version. This is useful for checking whether the firmware is up to date.

8.1.3 Feature Settings
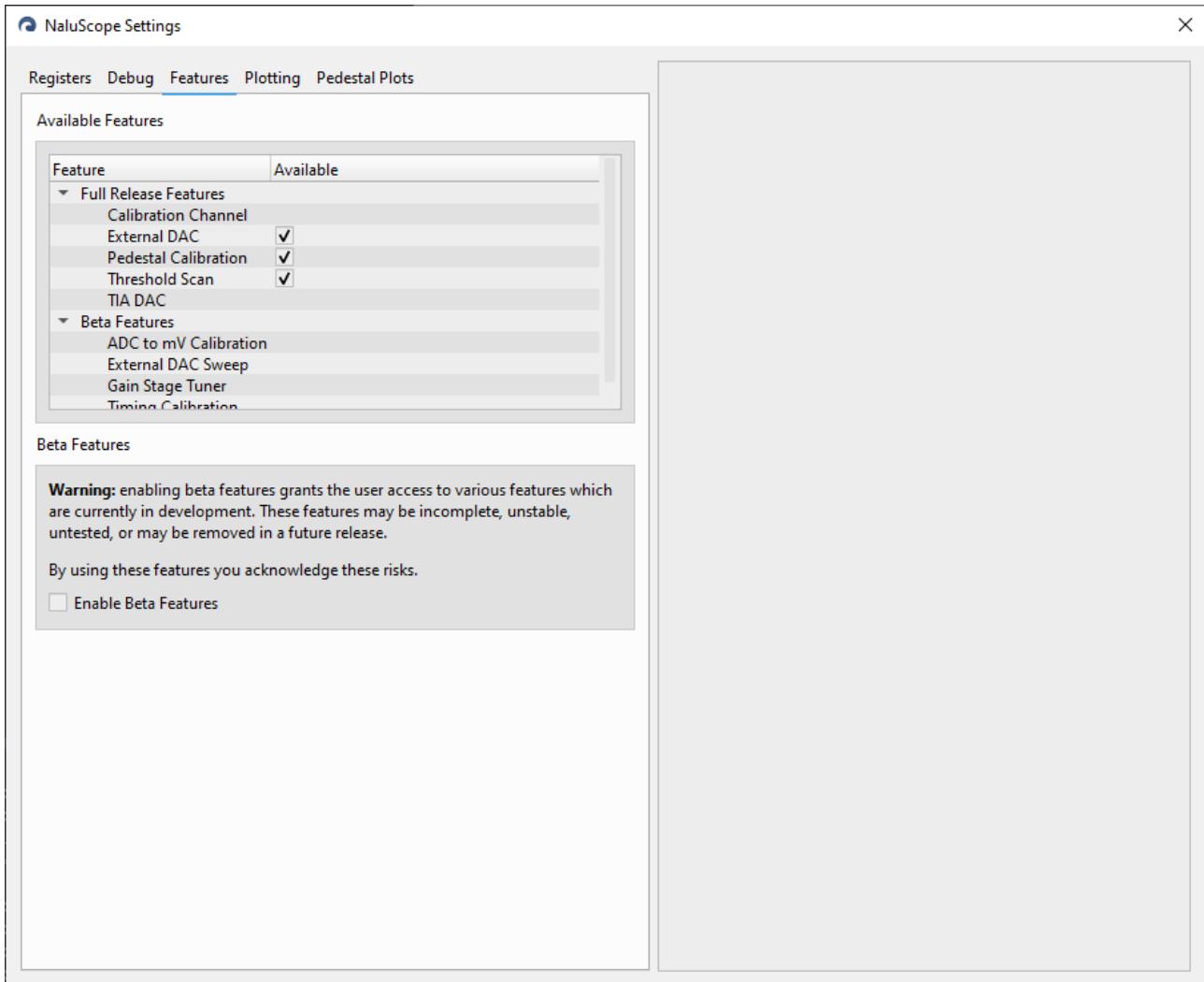
Figure 34. Feature settings.

The *Features* tab displays all available features for the current board. Features that are currently in beta are not enabled by default. To use these features, click the *Enable Beta Features* checkbox. Once the features are enabled, they can be disabled by unchecking the *Enable Beta Features* checkbox.

> **Warning**
>
> As shown in the warning message, beta features are considered experimental and may be unstable. Please consult a representative of Nalu Scientific before using any beta features.

8.1.4 Plotting Settings

The *Plotting* tab allows for the customization of the plot appearance, including font family, and size of the text used in the plot. This tab also contains the settings for the intersection visualizer (labeled as *2D Visualizer* in the screenshot) which control how the board channels are distributed between the *x*- and *y*-axes.
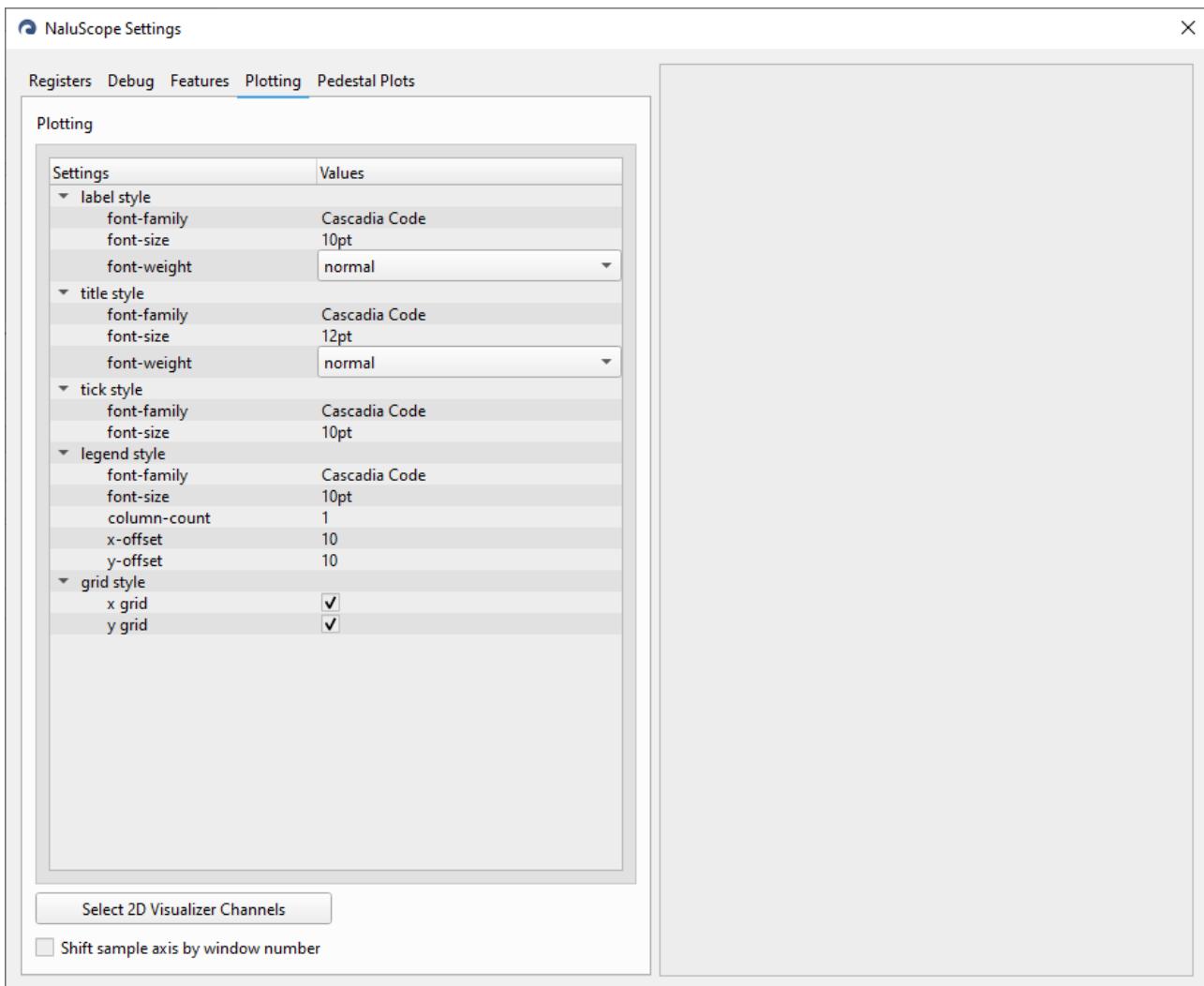
Figure 35. Plotting settings.

8.1.5 Pedestal Plots Settings

The *Pedestal Plots* tab allows for the configuration of outlier filtering and the appearance of the plots.
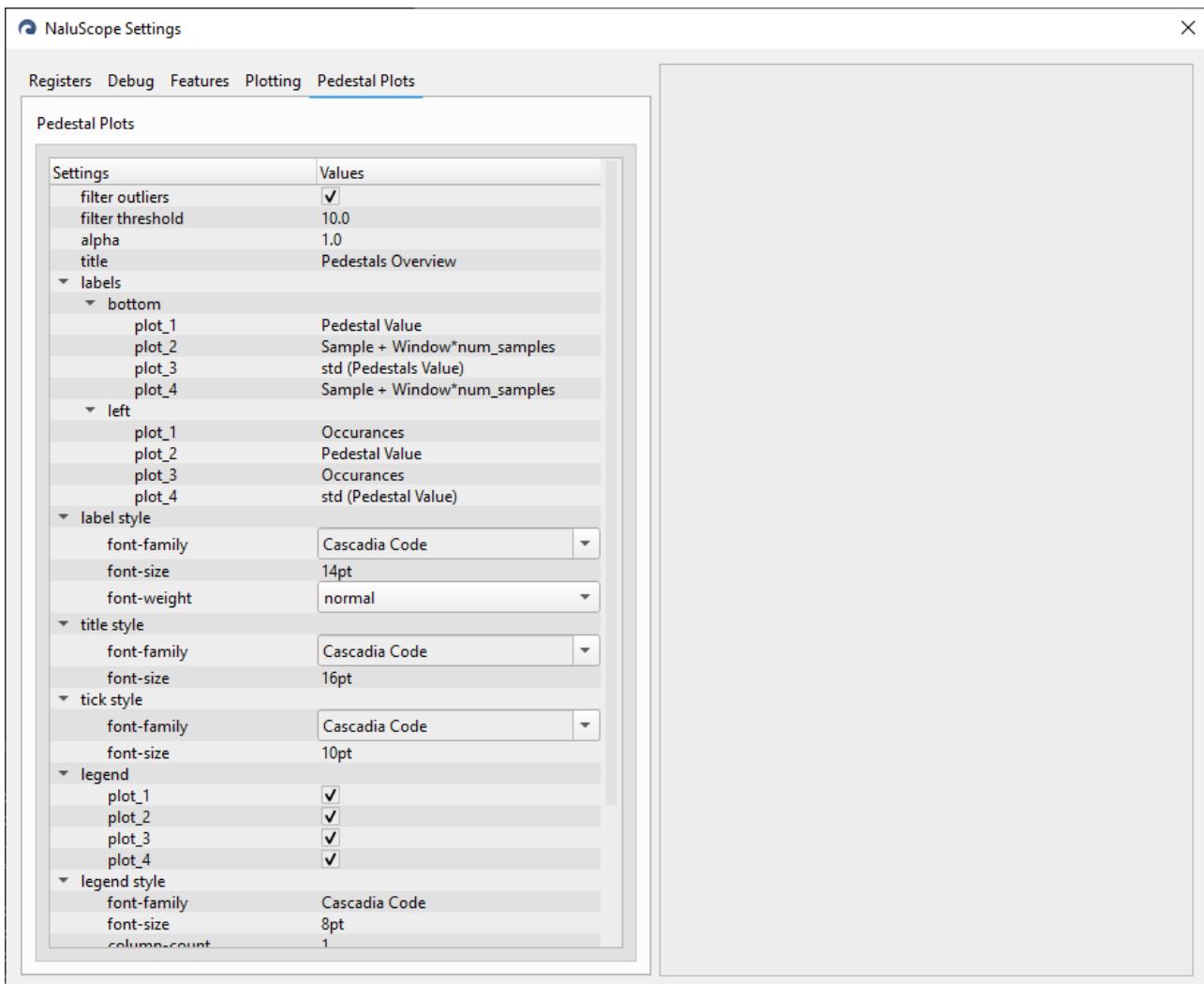
Figure 36. Pedestal plots settings.

The outlier-filtering algorithm uses a modified Z-score to determine whether a sample is an outlier. The threshold corresponds to the score at which a sample is considered an outlier.

8.2 Calibration

This section discusses methods used to generate and/or apply various types of calibration to events.

This section does not discuss calibration methods which currently in beta, such as the "Time Calibration" feature, as the methods are still undergoing development and are subject to change.

8.2.1 Pedestals Generation & Correction

Pedestals are the average value of the baseline of a channel. Knowledge of the baseline of the channels allows the application to subtract the baselines from the samples in an event, which is useful for removing both the built-in bias of the sampling capacitors and the DC offset of the signal through simple subtraction; this is known as "pedestals correction."

Pedestals calibration data is generated by averaging the entire sampling array of a channel over many captured events; the precise number is configurable from the generation dialog. When capturing data, the temperature of the hardware generally increases, causing the baseline to drift. A configurable number of "warmup" events may be

discarded from the beginning of each readout to allow the hardware temperature to stabilize. Furthermore, due to these temperature effects, it is *highly recommended* to generate new pedestal calibration data if the temperature of the hardware or environment changes significantly.

Due to limitations internal to the firmware, reading out the full sampling array reliably is not possible. Therefore, in generating pedestal calibration data, the sampling array is read out in smaller pieces called "blocks" which are stitched together to form the full sampling array. The size of these blocks is configurable from the generation dialog. The last block may be smaller if the block size does not evenly divide the sampling array. Note that since the hardware temperature can fluctuate between block readouts, warmup events are discarded from the beginning of each block rather than the beginning of the first block.

The algorithm used to generate pedestal calibration data is shown in the pseudocode below. Note that while the pedestal generation dialog expects the block size to be in terms of windows, the algorithm below uses the block size in terms of samples.

```
INPUT: regular_events # number of events to average in the pedestals
INPUT: warmup_events # number of events in each block to discard
SET sample_count = {{ board_configs["params"]["samples"] }} # sampling array size
SET channel_count = {{ board_configs["params"]["channels"] }} # number of board channels

# Capture and store events into buffer
SET buffer = array of length regular_events
SEND start readout command
FOR i = 0 to warmup_events + regular_events - 1:
    SEND firmware-simulated external trigger
    PAUSE for a short time
    IF i >= warmup_events:
        SET block_buffer[block, i] = event

# Shuffle event data in buffer into a new array for averaging
SET unaveraged_data = array of size (channels, sample_count, regular_events)
FOR event_index = 0 to regular_events - 1:
    SET event = buffer[event_index]
    FOR channel = 0 to channel_count - 1:
        FOR sample = 0 to sample_count - 1:
            # Each event in buffer is an array of size (num_channels, windows, samples)
            SET unaveraged_data[channel, sample, event_index] = event[channel, sample]

OUTPUT averaged_data = average along last axis of unaveraged_data
```

The output of this algorithm, along with information about the raw data, some parameters, and sensor readings are included in the final calibration data. This data may be saved by the user for later use. The file is saved as a Gzipped Python pickle file.

The calibration data generated from the above algorithm is used to correct the pedestals from events. Correction is much simpler than the generation of the data. It involves subtracting the relevant portion of the calibration data from the samples in an event.

8.2.2 Threshold Scan

The threshold scan is used to find the ideal threshold to set the trigger to when using the *Self Trigger* (also called *Signal*) mode. The result of the threshold scan changes with the external biasing, so it is recommended to run a threshold scan whenever the biasing is changed.

The threshold scan relies on a feature of the chip known as *scalars*, which are a count of the number of times the trigger circuit has fired within a short span. The higher the value, the closer the signal level is to the ideal trigger level. In the threshold scan, the trigger level is swept along some range in fixed steps, with the scalar values for each channel read at each point. The scalar values are then plotted against the trigger level, and the ideal trigger level is chosen by the user as the point where the scalar values are highest.

8.3 Acquisition Formats

This section discusses the storage formats of both new acquisitions and legacy acquisitions. Examples can be found on the examples repository detailing how to open both types of acquisitions using Python.

> **Warning**
>
> Support for legacy acquisitions will be removed in a future release. It is recommended to begin using the new acquisition format as soon as possible.

8.3.1 New Acquisitions

Acquisitions captured using the standalone or local server use a different data format than legacy acquisitions to support higher readout rates and additional failsafe mechanisms. A single acquisition is a directory containing the following files:

```
acquisition/
├── metadata.yml - Stores board parameters and register values at time of capture
├── {n}.bin - One or more "chunks" containing raw events stored back-to-back in chronological order. See b
├── {n}.idx - Contains information about where in the `.bin` files each event is located. See below for mo
├── (pedestals_calibration) - Pedestals calibration data, stored as a Python pickle
├── (timing_calibration) - Timing calibration data, stored as a Python pickle
└── (adc2mv_calibration) - ADC/mV calibration data, stored as a Python pickle
```

Modification of the acquisition components should be avoided, as this may cause the acquisition to become unreadable.

> **Note**
>
> Currently the endianness of `.bin` and `.idx` files is determined by the endianness of the system on which the acquisition was captured. This means that if an acquisition is captured on a big-endian system, it will not be readable on a little-endian system, and vice versa. Therefore it is recommended to use a little-endian machine for the highest portability.

8.3.2 Bin File Format

Each `.bin` file is referred to as a "chunk." Each chunk is limited to 500 MB and begins with the following header, represented in Rust syntax:

```rust
struct Header {
    /// Format revision number
    version: u16,
    /// Reserved for future use
    _reserved: u16,
    /// Length of the metadata sector
    metadata_sector_length: u32,
}
```

The metadata sector follows the header and contains a copy of the `metadata.yml` file.

8.3.3 Idx File Format

Each `.idx` file is referred to as an "index file," and contains a long list of "entries" holding information about where each event is located in the chunk file with the corresponding name. Each entry has the following format:

```rust
struct IndexEntry {
    /// Offset in bytes of the event in the chunk.
    offset: u32,
    /// Length of the event
    length: u32,
}
```

8.3.4 Legacy Acquisitions

A legacy acquisition is captured without using the standalone or local server and has the `.acq` file extension. These files are gzipped Python pickles, and while they can be opened using only the Python standard library, it is recommended to use the NaluDAQ package for convenience.

The pickle for a legacy acquisition is a large nested dictionary. The dictionary structure most notably includes the following keys:

```
Acquisition
├── `"calibration"`
│   ├── `"pedestals"` - Pedestals calibration
│   ├── `"caldata"` - ADC/mV calibration
│   └── `"timingcal"` - Timing calibration
├── `"events"` - Python `deque` containing all raw event data
└── `"metadata"`
    ├── `"info"` - Information about how the readout was started
    ├── `"naludaq_version"` - Version of NaluDAQ used to capture the acquisition
    ├── `"naluscope_version"` - Version of NaluScope used to capture the acquisition
    └── `"settings"` - Board parameters and registers at time of capture
```

8.4 Visualizer Algorithms

The visualizers use the following algorithms to compute the data displayed in the plots.

The accompanying Python code in the following sections is a simplified equivalent of the internal implementation. These code snippets cannot be run as-is; their only purpose is to provide a clearer picture of the algorithms.

8.4.1 Pixel View

The color of each pixel in the pixel view is determined by the following algorithm:

1. The event data is normalized so the data for all channels is in the range [0, 1].
2. A 2D array is built where each element is the arithmetic mean of the corresponding channel.
3. The array is displayed as an image.

The Python code below is a simplified equivalent of the internal implementation:

```python
import math
import numpy as np

# normalize the data
event: np.ndarray = ... # 2D data array with shape (channel count, sample count)
event /= np.max(event)

# compute coordinates for each channel
channels = event.shape[0]
grid_size = math.ceil(math.sqrt(channels))
coordinates = {c: (c % grid_size, c // grid_size) for c in range(channels)}

# populate the output array
output = np.full((grid_size, grid_size), fill_value=np.nan)
for channel, (x, y) in coordinates.items():
    output[x, y] = np.mean(event[channel])
```

8.4.2 Intersection Visualizer

The color of each pixel in the intersection is determined by the following algorithm:

1. The RMS of the signal is computed for each channel.
2. A matrix is built from the sum of each RMS value for each pair of channels on the *x*- and *y*-axes.
3. The matrix is linearly normalized to the range [0, 1].
4. The matrix is displayed as a heatmap.

The Python code below is a simplified equivalent of the internal implementation:

```python
from itertools import product
import numpy as np

# configurable channel parameters
x_channels = [0, 1, 2, 3]
y_channels = [4, 5, 6, 7]

# compute RMS values
event: np.ndarray = ... # 2-D data array with shape (channel count, sample count)
x_axis_data = event[x_channels]
y_axis_data = event[y_channels]
all_x_rms = np.sqrt(np.mean(x_axis_data ** 2, axis=1))
all_y_rms = np.sqrt(np.mean(y_axis_data ** 2, axis=1))

# compute the intersections
combinations = product(enumerate(all_x_rms), enumerate(all_y_rms))

# populate the output array
output = np.zeros((len(x_channels), len(y_channels)), dtype=np.float32)
for (x, x_rms), (y, y_rms) in combinations:
    output[x, y] = x_rms + y_rms

# normalize the output
output /= np.linalg.norm(output)
```

8.5 Ethernet

IP Addresses

An IP address (Internet Protocol address) is a unique set of numbers assigned to every device connected to a network that uses the Internet Protocol (IP) for communication. It functions as a home address, allowing data to find the right destination when traveling across the internet or a local network. There are two main types: IPv4 (e.g., 192.168.1.1) and IPv6 (e.g., 2001:0db8::1), with IPv6 designed to handle the growing number of connected devices. As of writing this manual, Nalu Scientific does not support using IPv6 addresses.

8.5.1 Modes of Operation

There are two methods that can be used to interface with your board over a gigabit Ethernet connection: 1) a peer-to-peer connection between your computer and board or 2) connecting your the board to your network.

Connecting the Board Over a Peer-to-Peer Connection

Determine if the board will be connected directly to one of the gigabit Ethernet interfaces on your computer (peer-to-peer) or if an adapter will be used. If your computer only has one Ethernet port and does not support Wi-Fi, it will be unable to access the Internet while connected to the board and your computer's network settings may need to be re-configured each time a board is connected. It is recommended that you use an Ethernet adapter or another network interface card for a more seamless experience.

Windows

Click on the Start (Windows icon) button in the taskbar and type "settings". Select the Settings app.
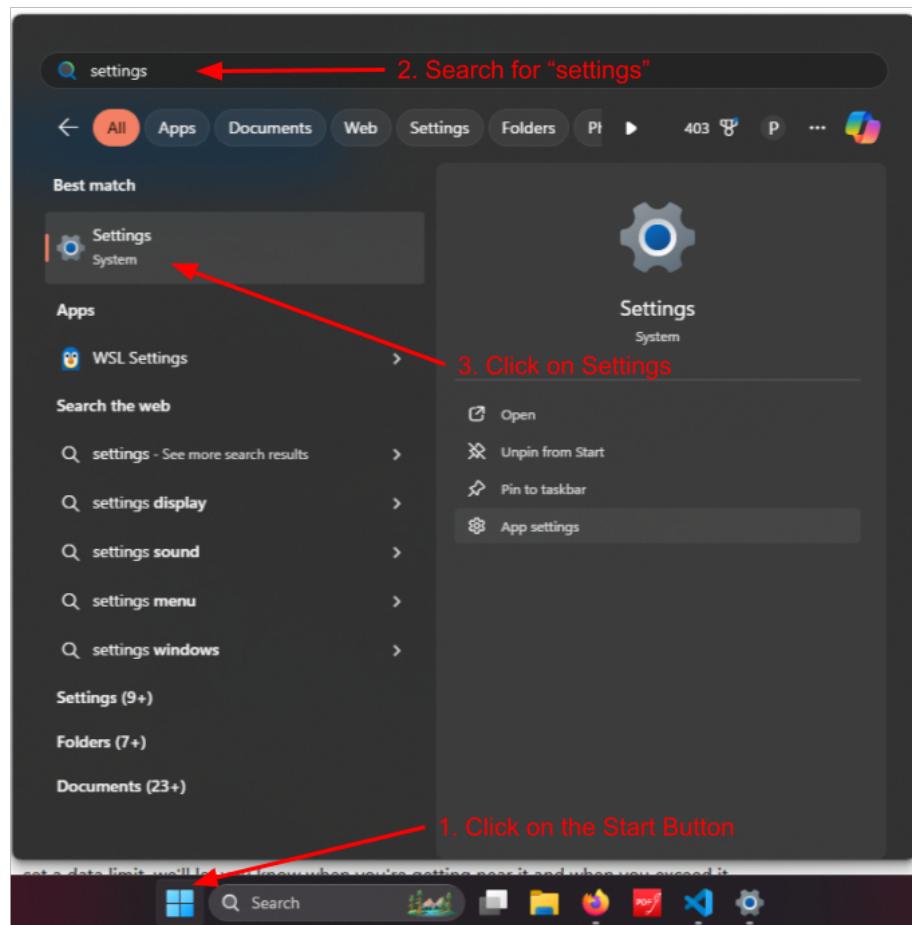
Figure 37. Launching settings with the Start button and search.

Click on *Network & internet*. If you are using Windows 10, the Settings app may have a slightly different layout. Locate the Ethernet adapter that will be used for the peer-to-peer connection to the board and click on *Properties*.
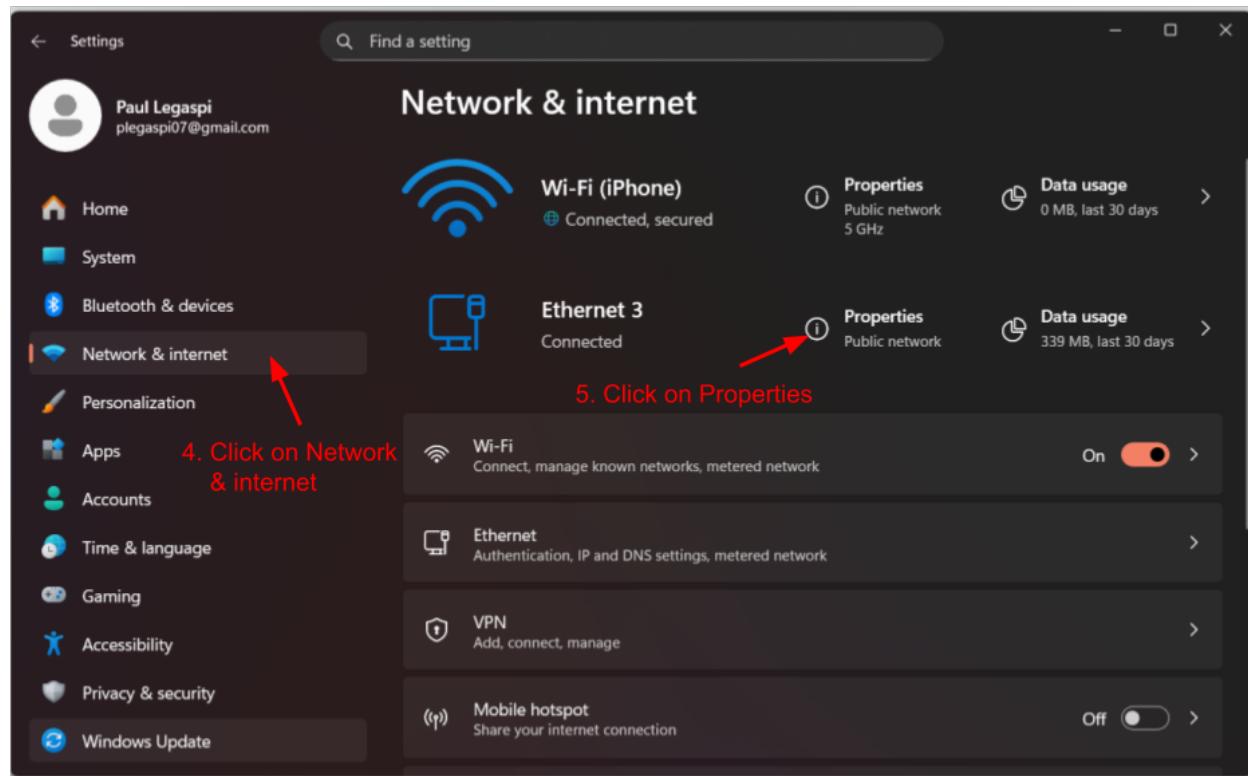
Figure 38. All network interfaces can be located and configured under *Network & internet*.

Next to *IP assignment*, click on *Edit*.
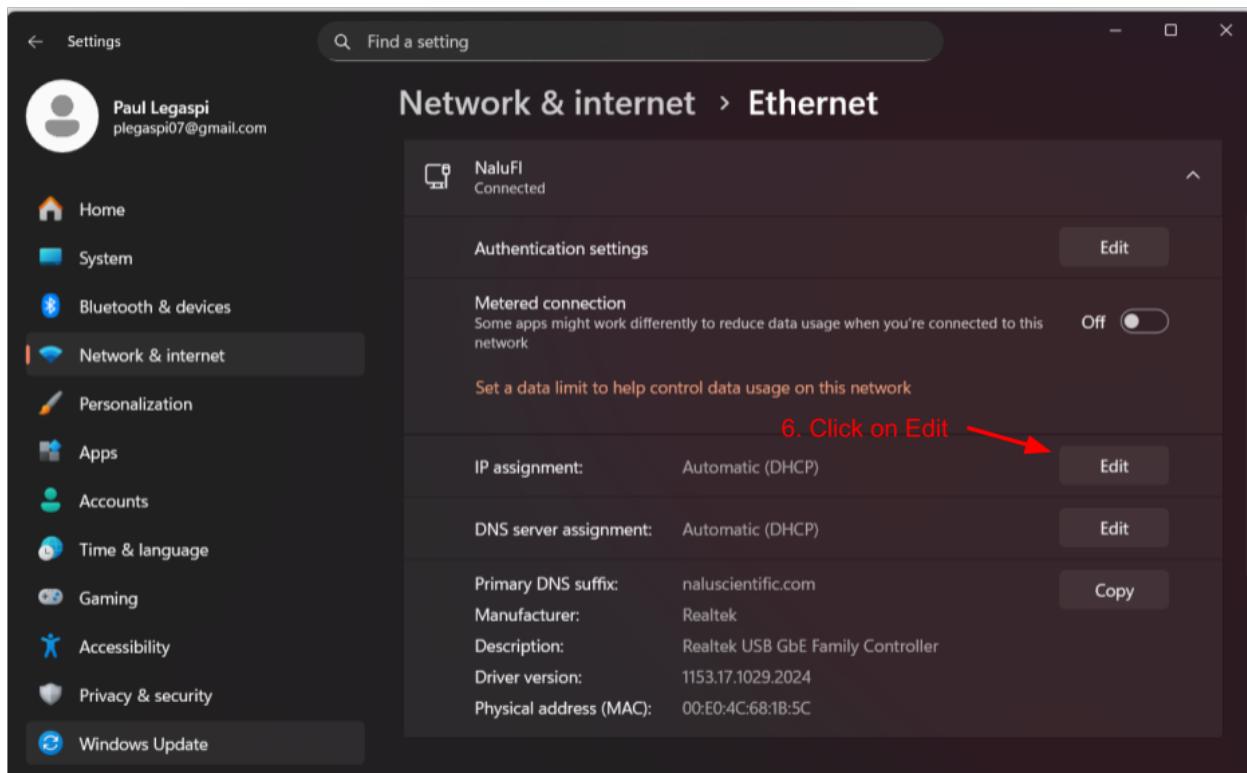
Figure 39. *IP assignment* is currently *Automatic (DHCP)*. After a successful configuration, it should read *Manual* and display the configuration information for the intended IPv4 address.

Complete the following fields as instructed below.

IP Address: It should follow the format of *192.168.1.XXX* where XXX is a number from 1-256. This number should be *different* from the IP address of the board. Please confirm that this IP address is not already assigned to another device. If you can ping the IP address *before* applying the changes in the dialog box, the IP address is already assigned to another device on the network and should not be re-used. Please use a different IP address in that scenario.

Subnet Mask: *255.255.255.0*

Default Gateway: Leave this field blank.
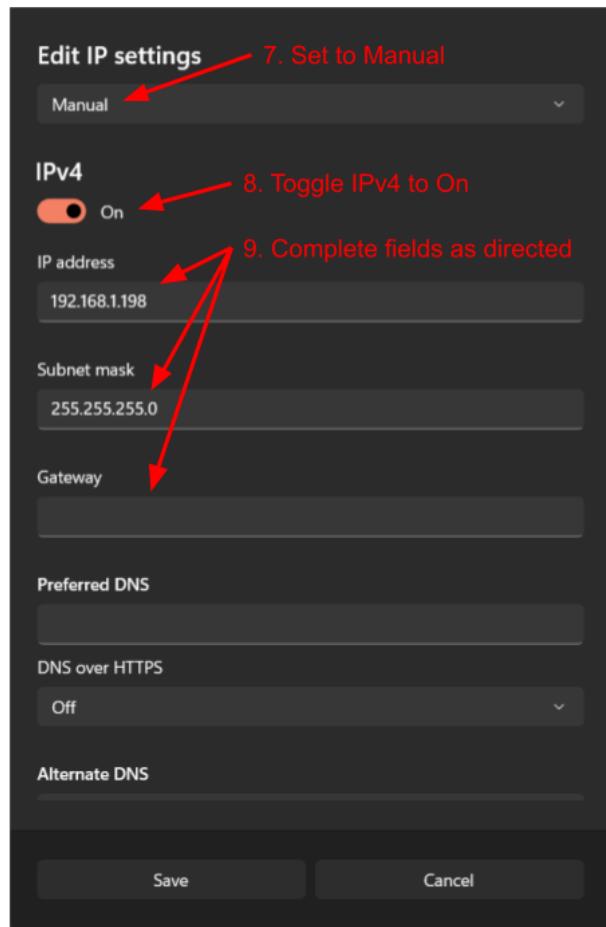
Figure 40. The IPv4 settings dialog box.

Click *Save* to apply your changes. If you receive any warnings or errors while saving your changes, it is likely that the IP address you selected was already assigned to another device and cannot be used. Try different IP addresses until there is no error message.

> **Note**
>
> Record the IP address that you assigned to your network interface. This IP corresponds to the Destination Address that you will be entering in the UDP connection information of NaluScope.

Linux

Instructions for changing the IP address, subnetmask, and default gateway of a Linux device differ based on the distribution of choice. Guides for each distribution are readily available on the Internet. The same configuration options listed in Step #4 of the Windows setup process can be used.
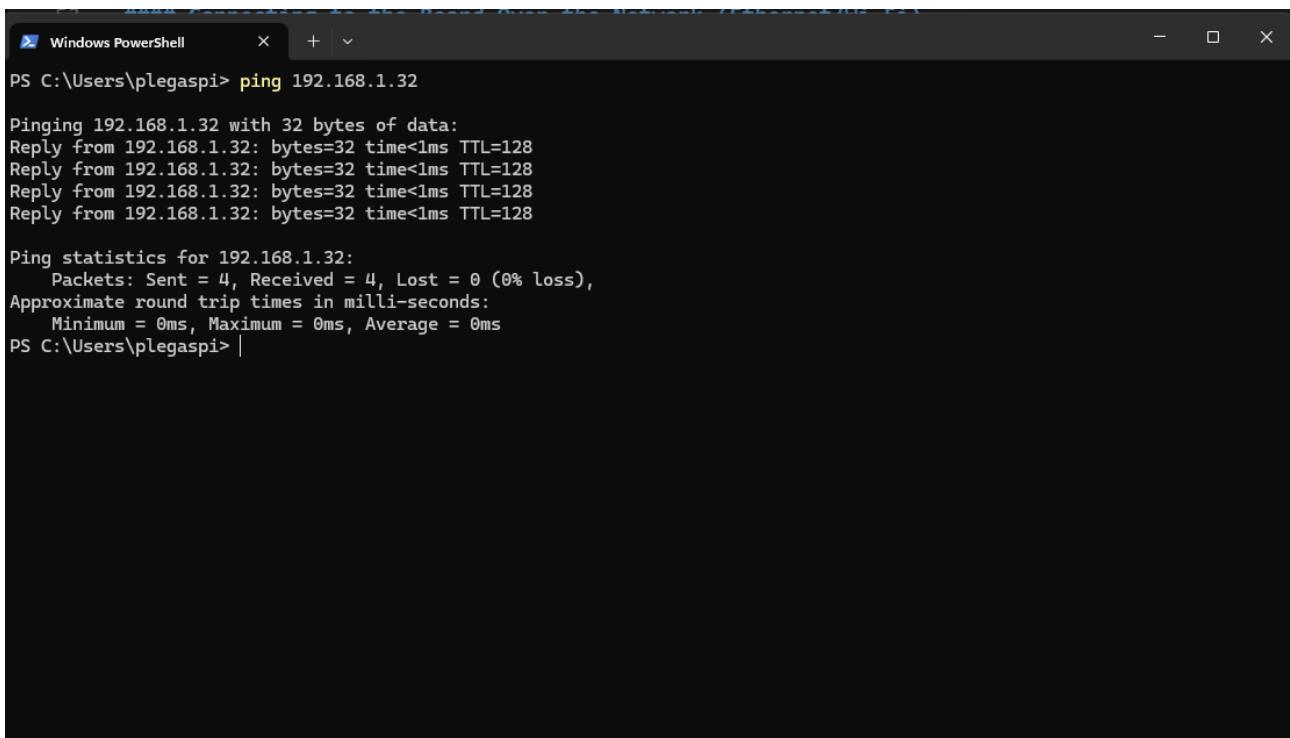
Connecting to the Board Over the Network

> **Warning**
>
> Connecting to the board over Wi-Fi is NOT recommended. Boards are capable of flooding Wi-Fi networks with packets at a rate that a client computer's network interface will be unable to send a stop command to stop data capture.

It is also possible to interface with the board without a physical connection to your computer provided that both devices are on the same network. If the network uses a different subnet that differs from 192.168.1.XXX (e.g. 192.168.2.XXX) and your router is not configured to route traffic between different subnets, NaluScope will be unable to connect to the board. If you are using a different subnet, you can confirm that Naluscope can reach the board by pinging the IP address of your board using the following command:

```
ping 192.168.XXX.XXX
```


Figure 41. Example of successful ping.

> **Note**
>
> IP assignment via DHCP is not currently supported.

8.5.2 Changing the Board IP Address

To change the IP address of the board, refer to the notebooks in Naluscientific/naluexamples. Examples for configuring the IP address are provided for UDP and UART-over-USB (FTDI) connections. In cases where the board's default IP address is already taken and/or you do not have control over how IP addresses are assigned on your network, UDP

configuration will not work and you will need to use UART-over-USB (FTDI) (or another connection method) to configure the IP address. The IP address set with these methods is temporary and will be reset to the default board IP address on reboot.

MAC addresses cannot be configured via software. Please contact support@naluscientific.com if you have multiple boards of the same model or have a MAC address conflict with an existing device to request firmware with a new MAC address. You can also request to have a different default IP address programmed into the firmware. An IP address configured with this method will continue to keep this IP address after rebooting.

8.5.3 Troubleshooting

> **Note**
>
> Connection to the board may also be unsuccessful if multiple devices on the same network have the same IP address or MAC address. You can determine if the IP address of the board is in use by pinging the IP before connecting the board to your network. It is unlikely that multiple devices will have the same MAC address as the board unless there are multiple boards of the same model connected to your network. If you would like to see the MAC addresses of the devices connected to your network, it is recommended that you connect to your router's web management interface or contact your network administrator. If multiple boards of the same model will be connected to the same network, you will need to change the IP and MAC addresses of the boards.

1. Turn the board on and off. If the board's registers become misconfigured during operation, Ethernet transmission may stop working or the board may not behave as expected.

2. Use a different Ethernet cable or confirm that the one connected to the board is not damaged. To test a cable, connect it between two known working devices and verify the connection. For example, connect a computer to a router or switch and try pinging another device on the network. If the router or switch has Internet access and your computer's Wi-Fi is turned off, successfully loading a website is also a good indicator that the cable works.

3. Verify that the Ethernet cable is securely attached to the Ethernet ports of the board and computer/network. The board should also be connected to the Ethernet interface that was configured during the setup process. Configurations applied to individual network interfaces are not applied globally, and connecting the board to a different network interface can lead to Naluscope being unable to detect the board. If you are using a different network interface to connect to the board, repeat the setup process for the intended mode of operation.For peer-to-peer connections, it is recommended that you set a different IP address for the current network interface to avoid conflicts with configurations for previous network interfaces. Update the Destination Host in Naluscope to the IP address corresponding to the current network interface.

4. Confirm that the board can be reached via its assigned IP address by launching Command Prompt on Windows or Terminal on Linux and running the following command:

```
ping 192.168.1.XXX
```

If you see "Request timed out" or "Destination host unreachable", this can indicate several possibliities. If you are using a peer-to-peer connection between your computer and the board, verify that the appropriate settings are applied for IP Address, Subnet Mask, and Default Gateway.

5. Check that your destination address which is typically your computer's IP address matches. You can determine this by launching Command Prompt on Windows and examining the output of `ipconfig`. Identify the network adapter used to connect to the board. You should then be prompted with the following:

```
Wireless LAN adapter Wi-Fi:
   Connection-specific DNS Suffix  . : *********
   Link-local IPv6 Address . . . . . : fe80::5666:c56e:670e:e5d7%16
   IPv4 Address. . . . . . . . . . . : 192.168.1.193
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . : 192.168.1.1
```

Confirm that the listed IPV4 address for the network adapter matches what is entered for the destination address in Naluscope. If you have multiple network adapters, verify that no other adapter has the same IP address. For Linux environments, `ifconfig` will produce a similar readout for network adapters but you may need to install `net-tools` if the command cannot be found.

9. Troubleshooting and Support

For questions, feedback, bug reports, feature requests, or additional help please visit the Contact page on our support website and follow the instructions for submitting a ticket.

9.1 Common Issues

For a complete list of common troubleshooting solutions please visit the FAQ page on our support website.

9.1.1 Board Not Detected

If the board does not appear in the list of available ports in the startup dialog, here are some things to try:

- Click the *Refresh* button in the dialog to reload the list of available ports.
- Make sure the board is powered on and connected with a working cable.
- Make sure the board is programmed with the latest firmware.
- Check the Windows Device Manager for driver issues.

If the application is used on Linux, the port's permissions may need to be modified. First, find which `/dev/ttyUSB<Port Number>` device the board appears as. This can be done by running `ls /dev/ttyUSB*` before and after connecting the board. The new device that appears is the correct port. Then, run the following command to modify the permissions:

```
sudo chmod 777 /dev/ttyUSB<Port Number>
```

9.1.2 Cannot Startup the Board

Problem: The board does not start up from the connection dialog after connecting successfully.

Solutions: Here are some things to try:

- Power cycle the board.
- Make sure the correct port and board model are selected in the startup dialog.
- Make sure the board is powered on and connected with a working cable.
- Make sure the software and firmware are up to date.

9.1.3 Cannot Capture Events

Problem: The board suddenly stops sending back data after previously capturing events properly.

Solutions: Here are some things to try:

- Make sure the board still has power and hasn't accidentally been disconnected.
- In the Settings menu, select Reset and then Reinitialize.
- Close and reopen NaluScope, then start up the board again.

These steps may also help if calibrations cannot be generated.

9.1.4 Black Screen on Startup (Linux)

Problem: NaluScope starts, but the screen is black.

Solution: This may be due to incompatible or improper links to the graphics driver between Linux distributions. This can be solved by deleting `libstdc++.so.6` in the `nalu` folder, allowing the system to fall back on its version of `libstdc++.so` .

9.1.5 FTDI Devices Not Detected (Linux)

Problem: The device is showing up as only UART and not FTDI.

This occurs when a service is converting the FTDI device into a serial port (UART).

Solution: Running the command: `sudo modprobe -r ftdi_sio` will stop the conversion of the FTDI device. You will have to run this command each time you plug the device/restart the computer. Consider adding a rule to prevent `ftdi_sio` from running.

9.1.6 Error: DEVICE_NOT_FOUND (Linux)

Problem: Naluscope won't startup with error:

```
File "/build/naluscope/build/venvs/gui_build/lib/python3.9/site-packages/ftd2xx/ftd2xx.py", line 133, in c
ftd2xx.ftd2xx.DeviceError: DEVICE_NOT_FOUND
```

This error occurs when NaluScope does not have permission to the FTDI device.

Solution: The permissions can be fixed by:

Determine where the board's USB is mounted to. `lsusb` can be helpful. `lsusb` will output in the form:

```
Bus 002 Device 001: ID xxxx:xxxx (FTDI_DEVICE)
```

Use the Bus & Device number to locate where the device is mounted in the directory:

`/dev/bus/usb/002/001` (Replace 002 & 001 with your corresponding Bus and Device numbers)

Set permissions for the files with `sudo chmod 777 /dev/bus/usb/path_to_your_usb`

9.1.7 Error: Settings schema (Linux)

Problem: Naluscope crashes with the following error when trying to select a project directory.

```
Settings schema 'org.gtk.Settings.FileChooser' does not contain a key named 'show-type-column'
```

Solution:

This error occurs when you have an older version of GTK installed on your system, and the older schema does not have the keys needed by FileChooser.

Download the newer [schema file](#) and move the file to

`/usr/share/glib-2.0/schemas/org.gtk.Settings.FileChooser.gschema.xml`

Run `glib-compile-schemas .` in the `/usr/share/glib-2.0/schemas/` directory.

Restart the shell and run NaluScope again. For GNOME users, you may press `Alt`+`F2` and run the `r` command to restart gnome-shell

10. Resources

Example code making use of the NaluDAQ and NaluDAQ_rs Python packages can be found in the examples repository. Included in the examples are notebooks for capturing data and opening acquisitions from disk.

Firmware, hardware documentation, and this manual can be found on the board-specific downloads page.

NaluScope and the standalone server executable may be found on the software downloads page

11. Appendix

The terms used in this manual are defined below.

11.1 Glossary

TERM	DEFINITION
Acquisition	A continuous collection of events captured over a period of time. This term also refers to the files/directories of an acquisition saved to disk.
ASIC	Application-Specific Integrated Circuit. Usually used in reference to ASICs designed by Nalu Scientific.
Baseline	The average value of a sample capacitor with no input signal. This often will be used in reference to *Pedestals Calibration*, which can be used to remove this offset voltage from digitized data.
Board	Typically refers to the combination of the FPGA and evaluation board, but may also refer just to the evaluation board.
Channel	A single input channel on a board.
Chip	A single instance of a Nalu Scientific ASIC on a board.
Digitizer	A device which converts an analog signal into a digital signal.
Digitizer Head	A pointer to the current sample in the sampling array
Evaluation Board	The PCB which holds the Nalu Scientific ASICs.
Event	A single capture of some portion of the sampling array.
FPGA	Field-Programmable Gate Array. Used by the software to communicate with the Nalu Scientific ASICs. Must be programmed with the appropriate firmware.
Pedestals	The bias voltage of a sample capacitor with no input signal. This often will be used in reference to *Pedestals Calibration*, which can be used to remove this offset voltage from digitized data. This term is used interchangeably with "baseline".
Sampling Array	The array of capacitors whose voltages are sampled during capture.
Trigger Event	A signal which indicates to the chip that an event should be read out.

11.1.1 Connection Types

TERM	DEFINITION
FTDI (D2XX)	A connection method using the FTDI D2XX driver. This works similarly to UART and uses the same physical cable, but offers higher reliability at high speeds.
Gigabit Ethernet (GbE)	Uses a network connection over ethernet to communicate with the board. This method offers the highest readout speed and is recommended for most applications.
UART	Universal Asynchronous Receiver/Transmitter. This connection method is discouraged due to its fragility when using high baud rates.
USB 3.0 (D3XX)	A connection method using the FTDI D3XX driver. This connection offers much higher speed and reliability than UART or FTDI (D2XX), but is not available on all hardware models.

11.2 Hardware Constants

Below is a list of constants that describe the hardware and cannot be changed.

CONSTANT	VALUE
Channel Count	4
External DAC Range	0 V (0 counts) to 2.500 V (4095 counts)
Resolution	12 bits
Sampling Array Length	254 windows (16256 samples)
Sampling Rate	3.200 GHz
Window Length	64

11.3 Board Capabilities

CONNECTION TYPE	AVAILABLE?
FTDI (D2XX)	Yes
GbE	Yes
UART	Yes
USB 3.0 (D3XX)	No

CALIBRATION TYPE	AVAILABLE?
Pedestals	Yes
Threshold Scan	Yes
ADC to mV	Yes (Beta)
Timing	Yes (Beta)

DISPLAYED SPEED	BAUD RATE
Slow	115200
Fast	2000000

11.4 Standalone Server REST API

The server uses HTTP for communication, allowing integration with any software language. The latest API specification and interactive controls are available at `http://SERVER_IP:SERVER_PORT/api` while the server is running. For example `http://127.0.0.1:7878/api` .

The REST API to control the server is as follows:

METHOD	ADDRESS	DESCRIPTION
POST	/acq	Create a new acquisition.
DELETE	/acq	Delete an acquisition.
GET	/acq/event	Fetch an event from an acquisition.
GET	/acq/list	List all acquisitions.
GET	/acq/misc_data	Get misc data from an acquisition
PUT	/acq/misc_data	Set misc data for an acquisition
PUT	/acq/move	Move acquisition from one place to another.
GET	/acq/output	Get the output acquisition.
PUT	/acq/output	Set the output acquisition.
GET	/acq/show	Fetch acquisition details.
GET	/board/raw	Read register(s) on the board
PUT	/board/raw	Send non-read command(s) to the board.
PUT	/connection/clear	Clear input and output buffers for the current connection.
PUT	/connection/d2xx	Connect to a board through D2XX.
PUT	/connection/disconnect	Disconnect any open connection.
GET	/connection/info	Get information about the current connection.
GET	/connection/list	List all devices connected to the system.
PUT	/connection/serial	Connect to a board over a serial port.
PUT	/connection/udp	Connect to a board over UDP.
PUT	/server/data-format	Set board model and data stopwords.
PUT	/server/shutdown	Shut down the server.