Willow GUI User Guide

March 20, 2015

1 Introduction

This is the user guide for the Willow GUI — a desktop application for interacting with Willow, a 1024-channel electrophysiology system developed by LeafLabs (www.leaflabs.com) in collaboration with MIT's Synthetic Neurobiology Group (SNG). This document, which is contained in the source code repository in `docs/user_guide/`, contains information for end-users. Developers should check out the developer's guide, which can be found in `docs/dev_guide/`.

2 System Overview

Willow is based on a Spartan 6 FPGA which collects and routes data from up to 32 amplifier/digitizer chips (Intan RHD2132), each chip sampling 32 channels of electrophysiology signals. The primary purpose of the hardware is to record all 1024 channels of data directly to an SSD, which is connected via SATA interface to the Willow board. A photograph of the bare hardware, connected and ready to record from a single headstage, is shown in Figure 1.

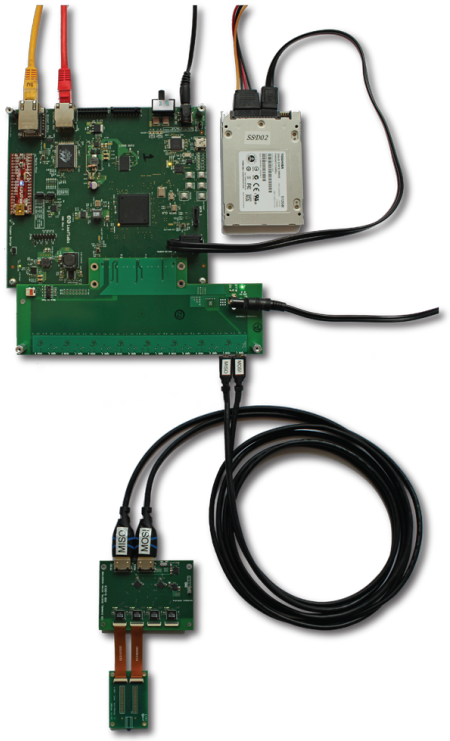

Figure 1: Willow hardware with a single headstage connected.

The Willow board communicates over TCP and UDP connections with a workstation computer running a daemon server called `leafysd`. All interactions with the hardware go through the daemon, which exposes an interface based on Google's Protocol Buffers (protobuf). The daemon repository includes command-line utilities for sending and receiving protobuf messages to and from the daemon. The GUI was designed to replace the command-line workflow with a more accessible and streamlined graphical approach.

There are many layers to the full Willow system; to learn more about them in detail, see `wired-leaf-docs/user_gui`
For the end-user, however, it is sufficient to be familiar with the basic architecture of the system. Namely, that the only robust mode of data acquisition is recording, which saves data directly to the SSD, and not to the workstation. To tighten the feedback look for data validation, there are other modes of acquisition (namely streaming and snapshots) which route data directly to the workstation, but these are not robust to data loss. For scientific experiments, the workflow is: record to SATA disk, then transfer over to a workstation for analysis. This is further explained in Section 4.

3 Setup

This setup guide assumes you have already installed the daemon and configured your network; if not, refer to the Willow user guide in `wired-leaf-docs/user_guide`.

3.1 Obtaining the GUI

To access the GUI repository, you'll need to be given permissions on the server — to request access, email `neuro@leaflabs.com`. Once you have permissions, you can clone the repo like so

```
$ git clone git@git.leaflabs.com:sng-gui
```

3.2 Dependencies

Before running the GUI, install these mandatory dependencies:

```
$ sudo apt-get install python-numpy python-matplotlib python-qt4 python-h5py \
          python-progressbar
```

3.3 Configuration

The GUI uses a configuration file, `src/parameters.py`, which stores variables related to the daemon installation, data storage, and other parameters. Before running the GUI, open `src/parameters.py` in an editor and modify the `DAEMON_DIR` and `DATA_DIR` variables to point to the locations of the `sng-daemon` repo and the preferred top-level data directory on your system.

4 Usage

To run the GUI:

```
$ cd src/
$ ./main.py
```

The GUI will present its main window (Figure 2), which consists of the status bar, the button panel, and the message log. The daemon is automatically started in the background, with `stderr` and `stdout` redirected to `log/eFile` and `log/oFile`, respectively. The message log should display a line indicating that the daemon was started successfully.

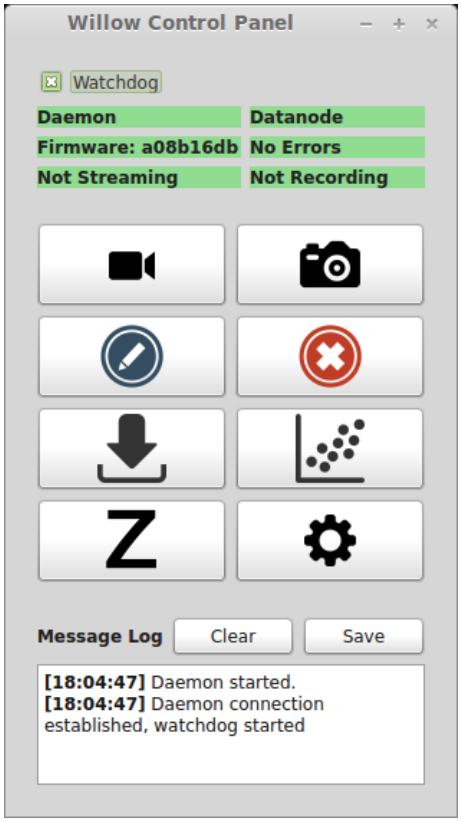

Figure 2: GUI main window, immediately after startup, showing the status bar at the top, the button panel in the middle, and the message log at the bottom.

4.1 Status Bar

The status bar is an array of labels at the top of the main window, which indicates the status of the daemon and the hardware. The status bar is updated by a watchdog process which pings the datanode once every second, and evaluates the response to determine daemon status, datanode status, firmware version, error conditions, streaming state, and recording state. The general color code for the labels is: green for "good/normal state", orange for "bad state, something is wrong", and gray for "unknown state". For example, if the datanode is unresponsive, its label will turn orange and the labels below it will turn gray to indicate an unknown state. Similarly, if the daemon is unresponsive, its label will turn orange and all labels below it will turn gray. These conditions are shown in Figure 4.

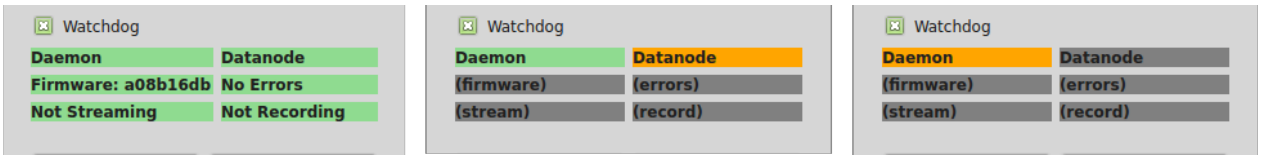

Figure 3: Status bar in three states: Daemon up and connected to Datanode, showing firmware version, error state, and streaming/recording states (*left*); Daemon up but datanode unresponsive (*center*); Daemon down or unreachable, hardware state unknown (*right*).

The firmware label, when green, will show the firmware version as a 32-bit hexadecimal number. The error label beside it indicates whether an error condition is present on the datanode. If so, this label will turn orange and display the value of the top-level error register, which is a bitmask of the error states of

the sub-modules on the FPGA. If an error condition comes up on the datanode, you'll need to reboot it to clear the errors. Hardware-related bug reports can be sent to `info@leaflabs.com`, and should include both the firmware version and the error state bitmask as context.

The "Stream" label indicates the streaming state of the datanode: green for idle, and blue for streaming. Similarly, the "Record" label beside it is green for idle, but turns red while recording and additionally indicates the current disk usage as a percentage in real time. These states — streaming and recording — are determined from the hardware registers on the datanode, and are therefore robust to daemon and/or GUI crashes.

The Watchdog process can be disabled by un-checking the "Watchdog" box at the top of the status bar. This should rarely be necessary, but may free up some bandwidth during streaming or transfer operations on slow workstations. The status bar will no longer be updated while the Watchdog is disabled.

4.2 Message Log

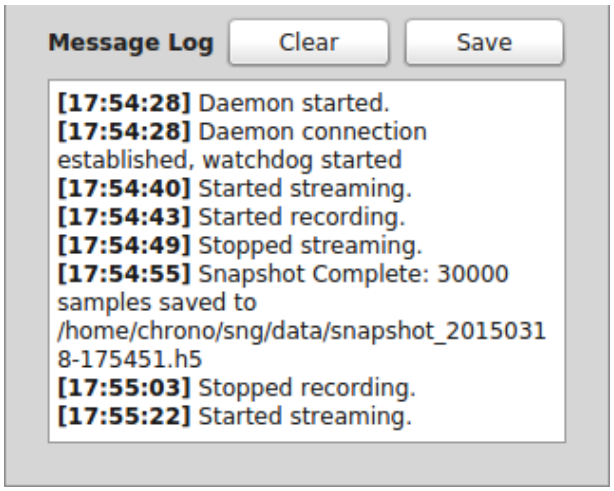

Figure 4: The message log, showing a list of actions taken, with timestamps.

At the bottom of the main window is the message log, which displays and logs messages from the GUI. Examples of messages include success/failure reports from requested actions, status updates from background threads, and general errors. Each message is prepended with a timestamp of the form `HH:mm:ss`. The message log can be cleared, or saved to a log file using the buttons at the top-right. Message logs are saved to the `log/` subdirectory by default, but can be saved anywhere by navigating the presented file dialog.

The message log is intended to serve as a supplement to the experimentalist's lab notebook, logging the actions taken — and their results — to help contextualize the data for future analysis.

4.3 Button Panel

At the center of the main GUI window is the button panel, containing 8 buttons with symbolic icons. These buttons are your starting point for all the capabilities offered by the GUI. Mousing over a button will display a tooltip which describes its action.

4.3.1 Streaming

Streaming is the real-time visualization of data from the Intan chips. When streaming, the data acquired from the ADC's is routed directly to the GUI, without being saved to SATA. Currently, users can view one channel at a time, as a scrolling waveform in a stream window.

To launch a stream window, click the "Launch Stream Window" button in the Button Panel. You will be presented with a dialog to configure the stream parameters. Fill in the channel number you wish to view, the plotting range, and the refresh rate — or, accept the default values and click "OK". The stream window will appear (Figure 5) in an idle state. To start/stop the stream, use the "Start" and "Stop" buttons. You can also navigate the plot, even while streaming, using the Navigation Toolbar at the bottom of the stream window. To undo/redo navigation changes, use the back and forward arrows; to return to the original view, click the home icon.

Live streaming in matplotlib is reasonably CPU intensive, so you may want to pause the stream while running other tasks within the application. Adjusting the "refresh rate" parameter in the stream dialog may help alleviate performance issues.

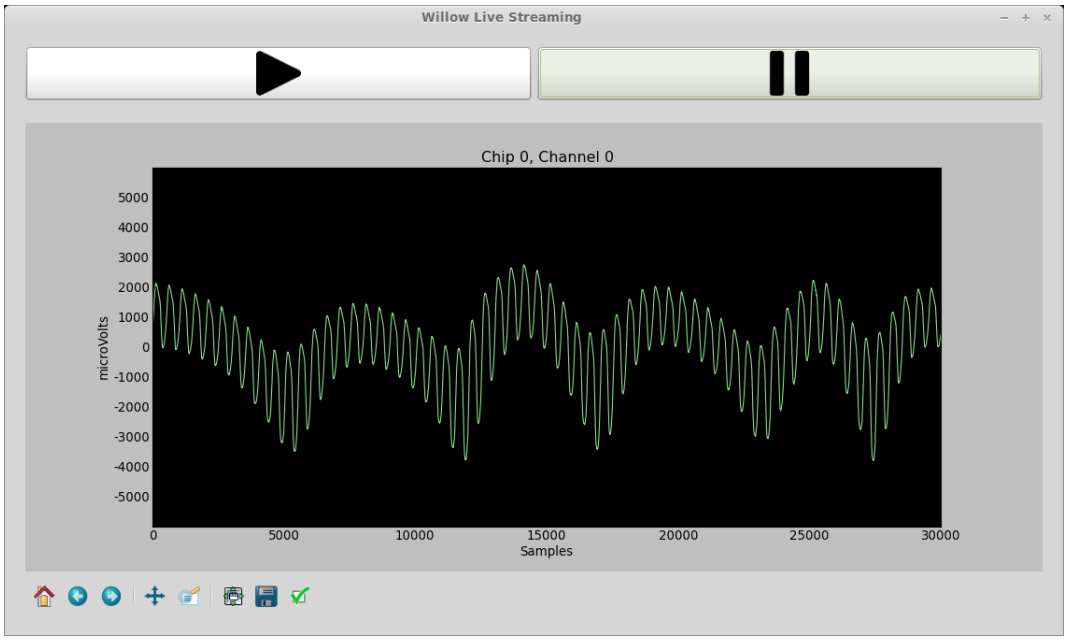

Figure 5: Stream window showing 60 Hz noise from a bare Intan chip.

4.3.2 Snapshots

A snapshot is a short (< 10 second) acquisition of board samples (all 1024 channels) which is saved directly to the workstation. It is essentially a stream which gets saved to a file (HDF5 format by default) to be analyzed offline. Like streaming, taking a snapshot does not involve the SATA drive, so you can take snapshots without worrying about overwriting your experiment on disk.

To take a snapshot, first make sure that any Stream Windows are stopped; then click the "Take Snapshot" button in the Button Panel. You will be presented with a dialog to configure the snapshot. Fill in the number of seconds you wish to acquire, and the filename you want to save to, and click "OK". The default filename is generated from the "Data Directory" parameter in your settings (see Section 4.3.7) and a timestamp taken at the moment you clicked "Take Snapshot" — the format is `dataDir/snapshot_YYYYMMDD-hhmmss.h5`, so the filename should be unique up to 1 second of granularity.

The snapshot dialog also gives you the option to "Plot when finished" by checking the box in the lower-left corner. If this is selected, then after the snapshot is saved, the GUI will automatically import the file and present it in a Plot Window (refer to Section **??** for information on using the Plot Window).

4.3.3 Recording

Recording is the acquisition of data directly to the SATA disk. It is the only acquisition mode that is guaranteed to be robust, and is intended for scientific experiments. To start a recording, click the "Start Recording" button in the Button Panel. The corresponding label in the status bar will turn from green to red, and the label will begin displaying the disk usage. Accurate reporting of disk usage is depending on the "Datanode Storage Capacity" parameter in your settings (see Section 4.3.7). To stop recording, click the "Stop Recording" button.

IMPORTANT: Every time you click "Start", the system will start recording from the beginning of the disk, over-writing any previous experiments you may have recorded. In this sense, the disk acts as a temporary (non-volatile) buffer for experimental data. If you've just recorded an experiment and want to preserve the data, it is recommended that you transfer the data over to your workstation immediately after stopping recording (see Section 4.3.4). If storage space is an issue on your workstation, an alternative is to remove the SATA disk and label it, thus repurposing the disk buffer as an archival system.

4.3.4 Transferring Experiments

The transfer function is used to transfer an experiment from the SATA disk to the workstation's file system. An experiment must be transferred before it can be analyzed — but even if analysis is not immediately necessary, it is wise to transfer important experiments over as soon as they are recorded, because (as explained in Section 4.3.3) experiments on disk are in danger of being over-written by subsequent recordings.

To perform a transfer, click the "Transfer Experiment" button in the button panel. You will be presented with a dialog to specify how much of the experiment you wish to transfer (either the entire experiment, or a subset), and the target filename. Click "OK" to begin the transfer. An infinite progress bar will appear until the transfer is complete — once it is, a success message will appear in the message box.

A couple of things to consider before initiating a transfer: First, transferred experiment files can be quite large (as large as the datanode storage capacity, i.e. 1 TB), so make sure you have the space on your workstation to store these files. Second, transfer times are approximately equal to recording times; if your recording lasted n minutes, you can expect to wait $\sim n$ minutes for a transfer to complete. In order to avoid daemon contention, you will be locked out of any GUI interactions until the transfer is complete.

4.3.5 Plotting Data

In addition to its primary purpose as the driver software for the Willow hardware, the GUI also offers some basic plotting utilities for exploring datasets. To plot a dataset (either a snapshot or a recording), click the "Launch Plot Window" button in the button panel. In the file dialog that appears, select the file you wish to import, and click "OK". Then enter how much data to import: either the entire file, or a subset. Be aware of the time and memory requirements before initiating an import — 1 second of data can take nearly 10 seconds to import, and large datasets can quickly consume memory resources. To prevent excessive memory consumption, the GUI will refuse to import datasets larger than the "Dataset Import Limit" parameter in the settings (see Section 4.3.7.

The dataset will be imported in a background thread, with a progress dialog indicating the status of the operation. Once complete, a Plot Window will appear (Figure 6), showing the data as an array of 1-dimensional waveforms. A control panel at the top allows you to adjust how many channels are shown at once, which bank of channels is currently being displayed, and the range of the plotting axes. A "bank" here represents a contiguous, ordered set of n channels, where n can range from 1 to 16. To page through banks, use the spinbox in the control panel — when activated, it can be controlled using the arrow keys on your keyboard.

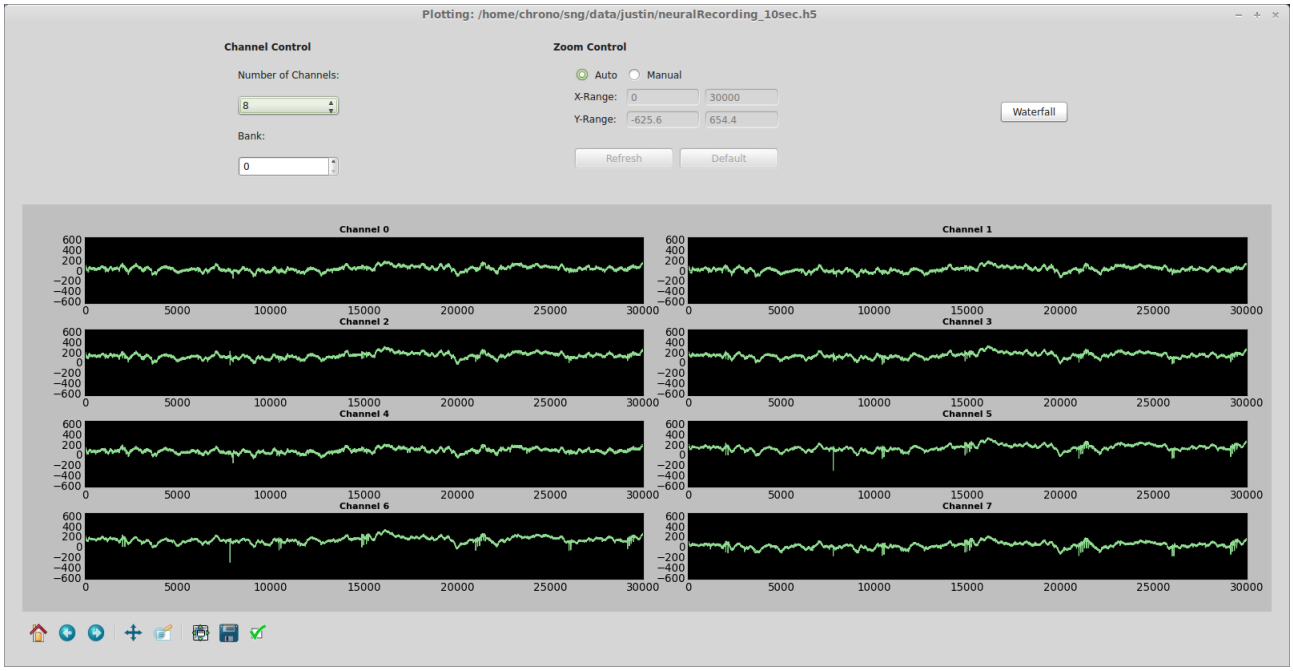

Figure 6: The Plot Window.

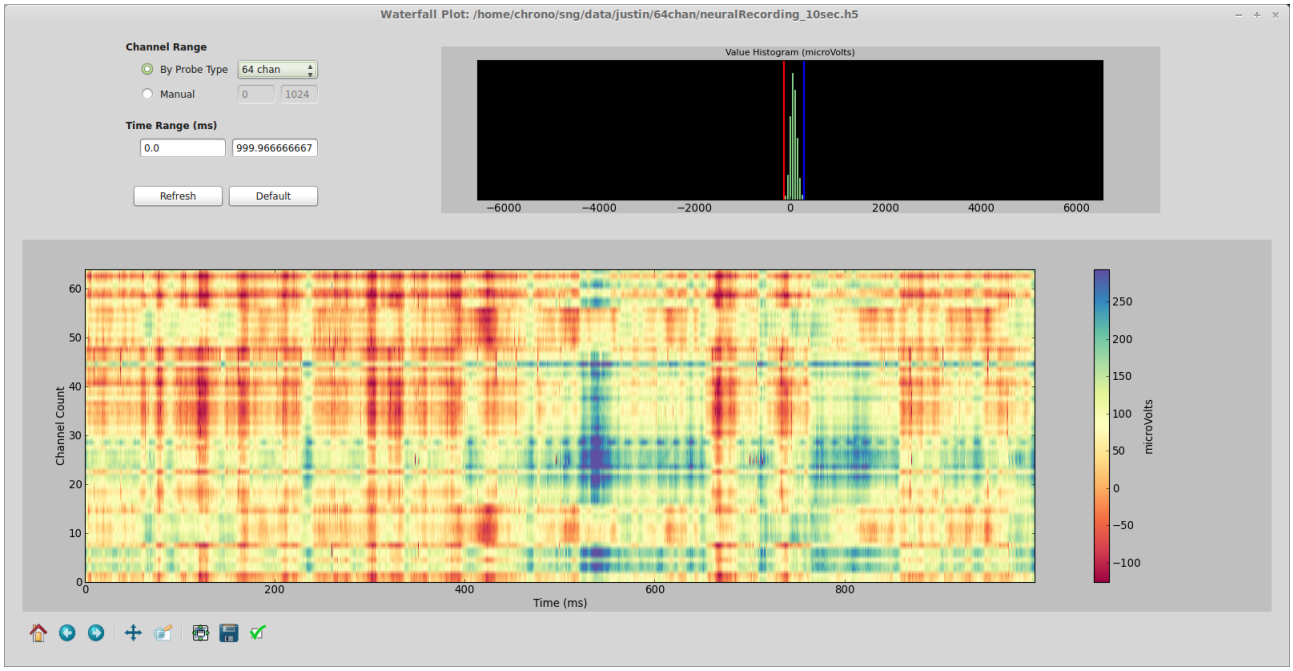

Figure 7: Waterfall Plot Window.

The "Zoom Control" section allows you to set the plotting range of **all** plots in the window simultaneously. By default, the zoom mode is "Auto", which looks at the current bank of channels and chooses a y-range based on its min and max; the auto-range updates every time you advance banks, or change the number of channels displayed. In "Manual" mode, you can set exactly the x-range and y-range of all the plots in the array — just enter the extrema and click "Refresh". Another way to adjust the plots is by using the Navigation Toolbar at the bottom of the window. This allows you to zoom in on one plot in

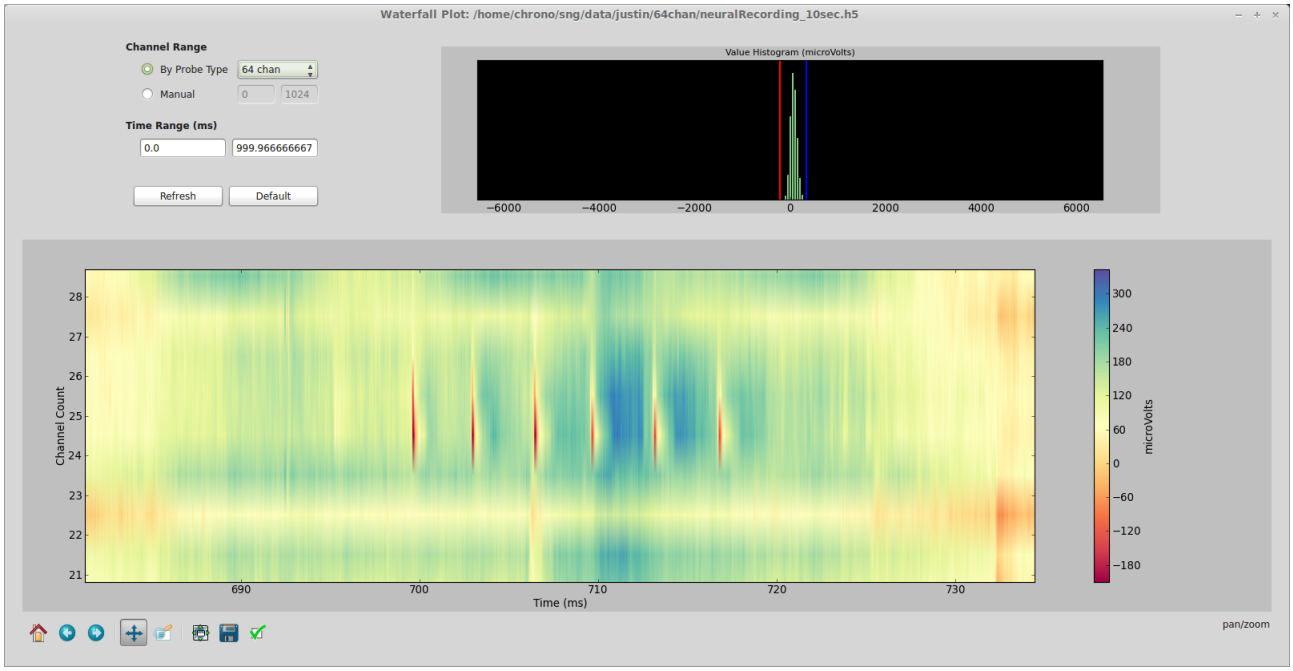

Figure 8: A train of 6 spikes, shown on several channels in the Waterfall Plot.

particular, leaving the others unchanged. To get back to the zoom settings defined by the "Zoom Control" group, just click the home icon.

Near the top-right corner of the Plot Window, you'll find a button labeled "Waterfall". Click here to launch the Waterfall Plot Window. The Waterfall Plot Window (Figure 7) shows **all** the data from a given import, in a 2-dimensional color plot. The x-axis is time (with units in sample number), the y-axis is channel number, and color represents signal (voltage), with values indicated by the colorbar on the right. When you first open the Waterfall Plot Window, it will show the entire 1024-channel dataset. If any of the channels were not connected to a live Intan chip, they will show up as zeros, which will appear as a large red band on the Waterfall Plot. To zoom in on the channels of interest, adjust the "Channel Range" parameters in the control panel at the top — this can be done manually, or automatically by probe type. To zoom in on a window of time, adjust the "Time Range" parameters. As usual, you can also zoom into subregions of the plot by using the Navigation Toolbar at the bottom. Finally, to adjust the limits of the color mapping, an interactive value histogram is provided at the top-right of the Waterfall Plot Window. The red and blue vertical lines represent the lower and upper limits of the color map, and these can be adjusted by left-clicking and right-clicking, respectively, on the histogram. In Figure 8, we have used these techniques to zoom in on a spike train that registered across several channels.

4.3.6 Impedance Testing

The Intan headstages offer impedance testing functionality which can be accessed from the GUI. Click the "Run Impedance Test" button on the button panel — this will present a dialog where you can choose to measure the impedance of a single channel, or of all channels in sequence. If you choose "Single Channel", the process will take about 10 seconds, and the result will be posted on the message log. If you choose "All Channels", the loop will run for about 3 minutes, and the result will be saved in a time-stamped `.npy` file, also reported in the message log. In "All Channels" mode, you also have the option to "Plot When Finished", which will bring up a plot window showing the impedance results across all channels (see Figure 9). All impedance measurements are taken at $f = 1$ kHz.

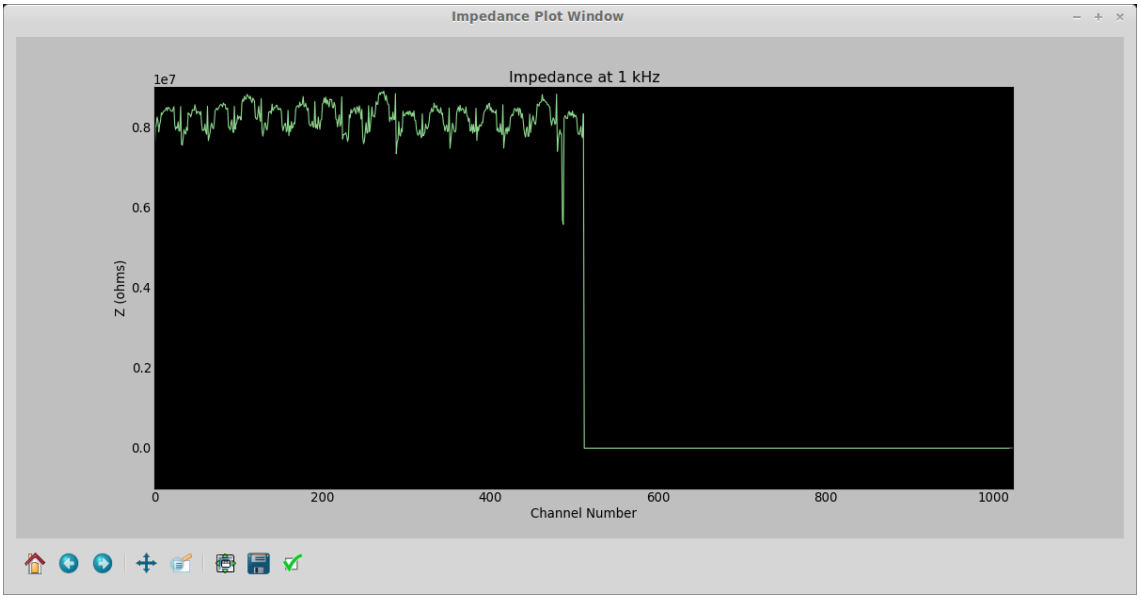

Figure 9: Impedance results from a measurement taken with 4 of 8 headstages bare headstages connected. As expected, channels 0-511 show the 32-channel periodicity from the PCB traces, and channels 512-1023 are dead.

4.3.7 Settings

Blah blah settings.