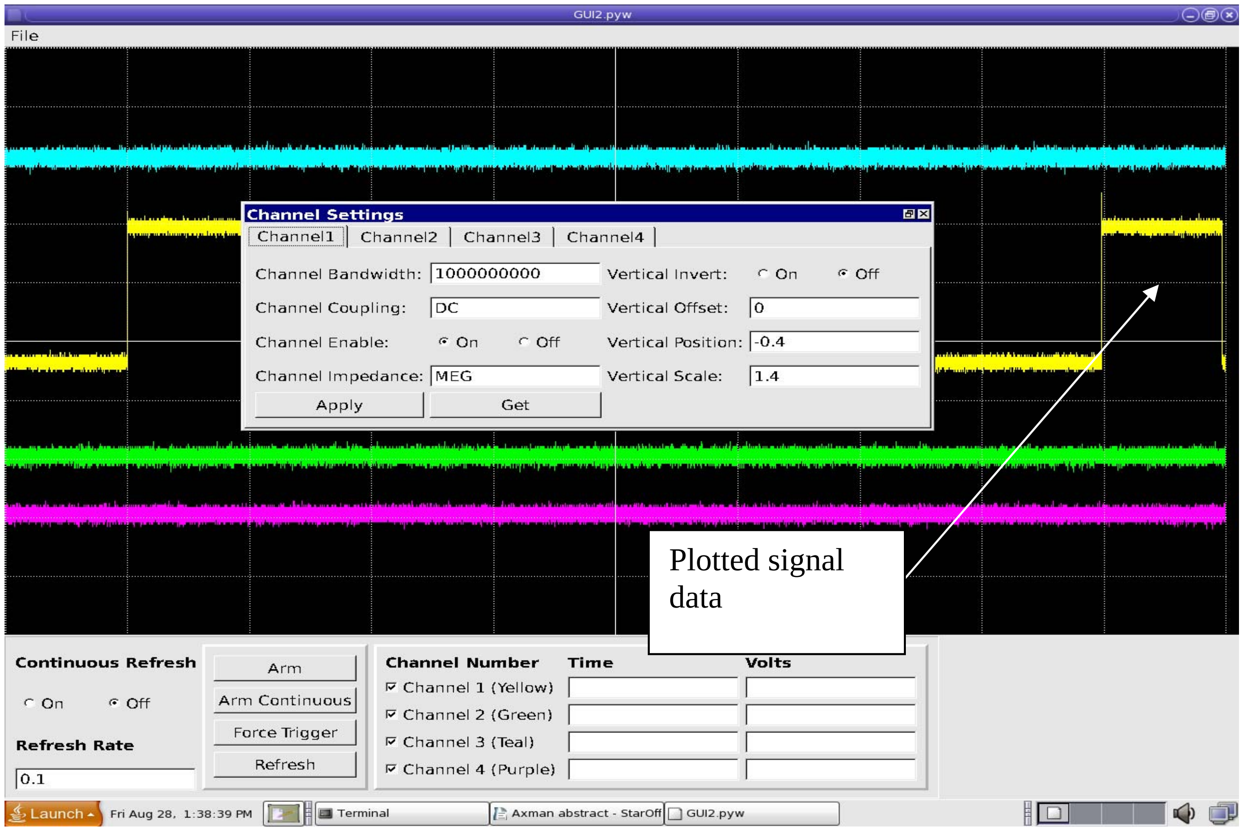
A Graphical User Interface to Oscilloscopes

D. Axman
West Irondequoit High School
Advised by Ralph Russo and Robert Rombaut

Laboratory for Laser Energetics
University of Rochester
250 East River Road
Rochester, NY 14623

Abstract

A graphical user interface (GUI) was created that allows a user to control and display data from an oscilloscope by communicating across a network. The GUI provides a convenient way to observe diagnostic signals throughout the OMEGA laser system. The GUI displays a graph of the data received from each channel of the oscilloscope, showing the signal's shape as well as other attributes. The user is able to continuously update the display with the most recent data or, at any time, acquire a single trace. The user is also able to arm the oscilloscope or force its trigger. The GUI communicates with the server program using ICE (Internet Communication Engine), a software package used for network communications. This program will allow scientists to analyze the shape of the laser pulse before and after it goes through the amplification process.



Channel Bandwidth: 1000000000
Vertical Invert: On [✓ Off]
Channel Coupling: DC
Vertical Offset: 0
Channel Enable: [✓ On] Off
Vertical Position: -0.4
Channel Impedance: MEG
Vertical Scale: 1.4

Continuous Refresh: On [✓ Off]
Refresh Rate: 0.1

✓ Channel 1 (Yellow)
✓ Channel 2 (Green)
✓ Channel 3 (Teal)
✓ Channel 4 (Purple)

Figure 1. Graphical user interface display, showing the Channel Settings dialog box.

Introduction

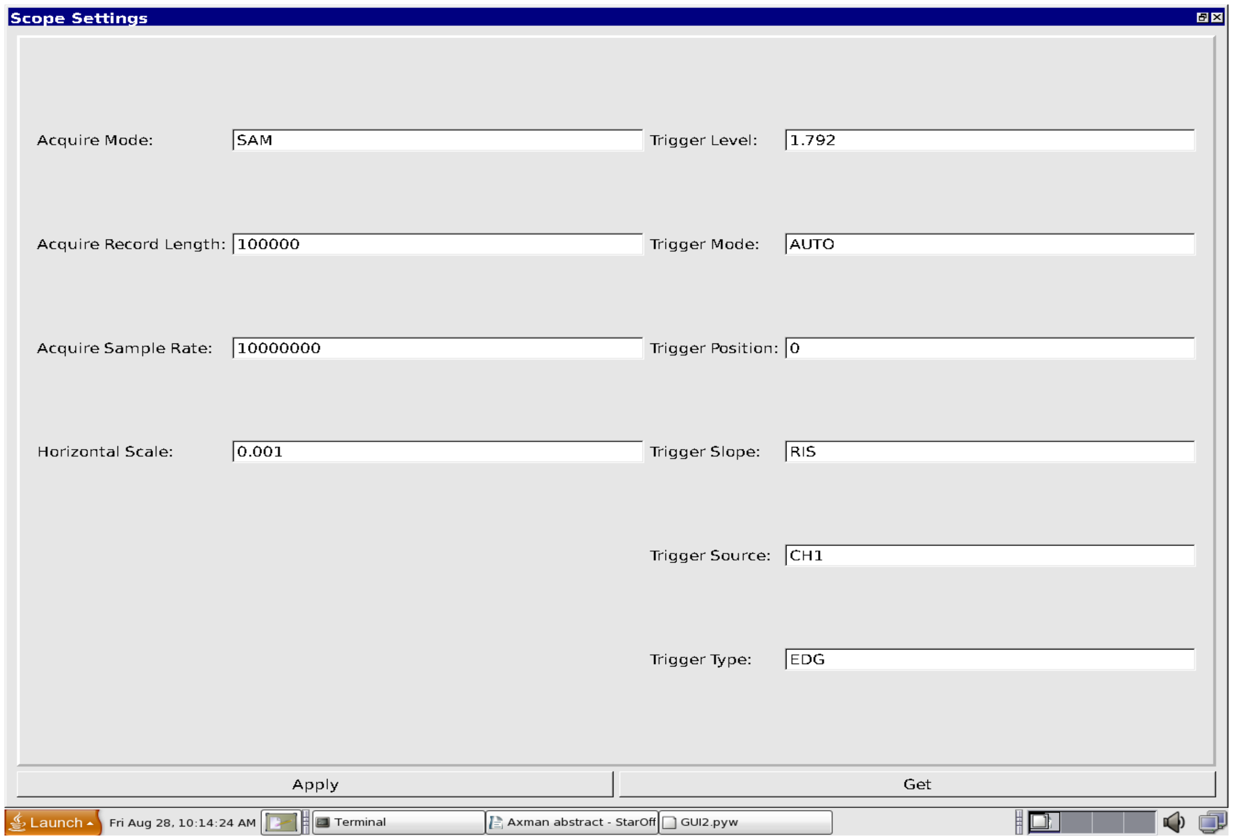
The motivation for this project was that viewing data from the OMEGA laser was not yet as streamlined as it could be. This project entailed the creation of a GUI to enhance the user's ability to view and understand the data by gathering and arranging it in the most aesthetic and understandable way possible. The main component of the GUI is the graphical display of the signal data. This data is sent to the client program (a program whose main function is the receiving of data over a network) from the server (a program whose main function is distributing data over a network) as a list of bytes, a list that is twice the size of the actual number of samples. This series of bytes is then converted into floating point numbers, which represent the data collected from the oscilloscope.

Graphical User Interface

An example of the display generated by the GUI is shown in Figure 1. The colored curves represent the signals (voltage vs. time) of channels 1 to 4. One of the primary problems encountered during the creation of this graphical display of the signal data was the fact that there are more samples and therefore more corresponding times on the horizontal axis for each of these samples than there are horizontal pixels on the screen, with approximately one thousand samples per horizontal pixel. This problem causes information to be lost due to the inability to display the difference in times of each sample's acquisition. The way that this problem was solved was that the information was first stored so that it could be easily referenced with precision and specificity in terms of time. Then, when the information was graphed, instead of simply graphing each sample by plotting it on the pixel that graphically represents the sample's time of occurrence in relation to the total time of the pulse, it was determined that the best way to save display time and memory usage was, in each group of samples per horizontal pixel, to find the largest voltage and the smallest voltage, translate them into a relative pixel coordinate and draw a vertical line between them, essentially creating a "best fit line" for the graphed points. This was done because when all one thousand points are plotted in the horizontal space of one pixel they appear as a simple line to the user regardless. Incorporated into this graphical display is a function for more specific observation of point values. This function makes it such that the user is able to drag his or her mouse on the graph and the time and voltage at that point will be displayed at the bottom of the screen for each channel that is enabled (see Figure 1).

On the bottom bar a key for the channels is displayed with a check box next to each channel. Un-checking this check box will cause the channel to be hidden on the graph and rechecking it will cause it to be shown. If the channel is disabled then the box is automatically made blank. There is also an option for enabling a continuous refresh of the graph and box in which the refresh rate can be specified in Hertz. There are also a number of functions that can be performed such as doing a manual refresh on the graph, forcing the trigger, arming single shot, and arming continuously. Any time the scope is armed it will acquire data on the next trigger. Continuous arm causes the scope to acquire data at a fixed periodic rate while a single shot arm only acquires once. By forcing the trigger the user can force the scope to capture the pulse before the conditions are met for automatic triggering.

At the top left corner of the screen is the file menu, which displays several commands: Scope Settings, Channel Settings, and Exit. Exit will close out the window cleanly and end all data acquisition from the server. Scope Settings will open up a dialog box (see Figure 2) with all of the channel independent settings for the oscilloscope. Channel Settings will open a dialog box (see Figure 1) with all of the channel specific settings for the oscilloscope. These settings will be displayed under several tabs, one for each channel, so that it is easy to change settings for individual channels. Many of these settings are self-explanatory; however, some such as the horizontal scale, the trigger level, position, and slope require explanation. The horizontal scale is the scale in seconds, or the number of seconds per interval. The trigger level is the minimum level of trigger signal required to trigger the scope. The trigger position is a position in time defined somewhere on either side of the trigger signal. The slope is a check on whether the slope of the pulse should be rising or falling to set off the trigger. When the fields for the settings are altered by the user the text turns blue so that the user knows what was changed. When the user hits the “Apply” button it sends the changes to the server program and the server sets the scope with the new attributes. The client then retrieves the attributes back from the scope and displays it in the fields as black text. The reason for this is so that if an invalid value (a value that cannot be computed such as a word where a numerical value is required or a number that is not supported by the program) is entered into a field and then set, the field will not display an incorrect value. For example, if an extremely small number were entered for the horizontal scale, the scope would only set the horizontal scale to the lowest possible value; this value would then be displayed as the actual value and not the unobtainable one the user entered. The “Get” button causes the settings to be changed to those on the scope and the text in all fields to return to the color black.



Scope Settings

Acquire Mode:	SAM	Trigger Level:	1.792
Acquire Record Length:	100000	Trigger Mode:	AUTO
Acquire Sample Rate:	10000000	Trigger Position:	0
Horizontal Scale:	0.001	Trigger Slope:	RIS
		Trigger Source:	CH1
		Trigger Type:	EDG

Figure 2. Scope Settings dialog box, used for channel-independent scope settings

As shown in Figure 1 the channel dependent settings are set up in roughly the same format as the channel independent settings with the obvious difference being that the channel settings have tabs for the settings of specific channels. This dialog box contains settings for changing quantities such as the vertical offset, vertical position, and vertical scale. This dialog box also allows the user to enable or disable certain channels. When a channel is disabled it will no longer be read by the oscilloscope, or displayed on the GUI. The same features incorporated in the scope settings dialog are also present in the channel settings dialog.

This program uses Internet Communications Engine (ICE) (see [1]) to communicate across the network with the server. ICE is a language-independent software package which means that it can be used to communicate between two programs that are written in completely different languages. ICE works by establishing a proxy and sending interpreted commands across this proxy to be reinterpreted on the other end. Since the client was created in Python (see [2,3]) and the server was created in C++, a language independent software package was essential. One limitation of the current program is that it takes up to 3.2 seconds to acquire the data from the server for all four channels and another 2 seconds to format it. Although much has been done to streamline the program as much as possible these times are still relatively long. A possible improvement to make the client more responsive would be to enable threading, which would allow for parallel execution of some functions. The client program has had every possible adjustment made so as to maximize its speed and efficiency.

Conclusion

A graphical user interface has been successfully created and will hopefully be implemented on the OMEGA laser system in the near future. This program will allow scientists to observe diagnostic signals throughout the Omega laser facility. Having a convenient program to view data from multiple oscilloscopes is a useful tool for scientists to visualize and analyze diagnostic data. The results of this effort can be extended to enable remote viewing and control of virtually any oscilloscope in the laser system.

References

1. "Manual for Ice." *Welcome to ZeroC, the Home of Ice*. July 2009. Web. 28 Aug. 2009. <http://www.zeroc.com/>.

2. Lutz, Mark, and David Ascher. *Learning Python, Second Edition*. North Mankato: O'Reilly Media, Inc., 2003. Print.

3. Summerfield, Mark. *Rapid GUI Programming with Python and Qt*. Prentice Hall, 2007. Print.