Using LabVIEW to Automate the ALEPH High Intensity Titanium Sapphire Laser

Sunny Wong, Reed Hollinger, Jorge Rocca
sw758@cornell.edu
Colorado State University REU Summer 2020

Project Overview

1 Examples of experiments that could benefit from automation

2. A basic data acquisition loop

3. Interfacing with Newport Power Meters for energy measurements

4. Interfacing with stepper motors to control the camera and target system

5. Interfacing with the laser firing system.

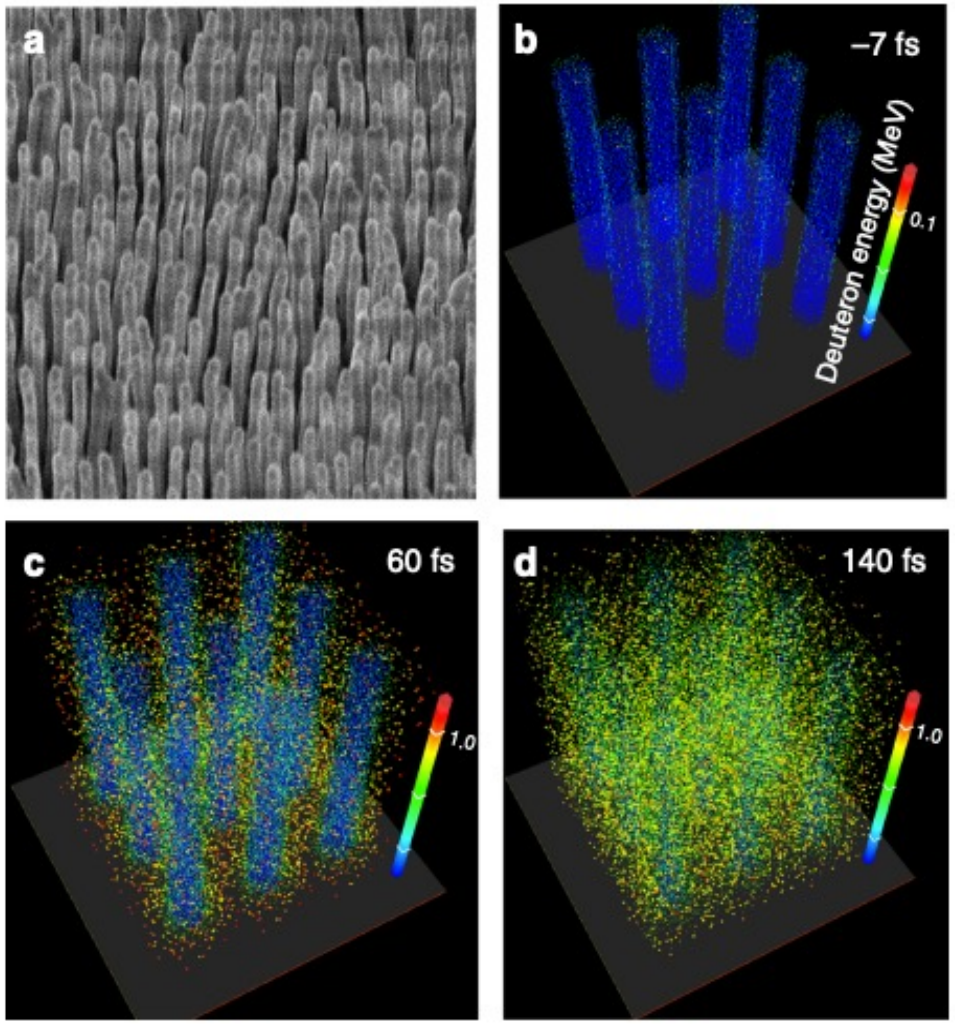
Benefits of Automation

- Current shot rates: one shot per 1-5 minutes
- The 0.85-petawatt Ti:Sa Laser can run at 3.3 hertz
- With enough data points through intensity scans, diagnostics behave as a function of energy

	Time 1	Shot Count	Energy 1 (mJ)	Energy 2 (mJ)	Target	Comments	Post-Comments	Prepulse Energy	Prepulse Delay	Prepulse Duration
7/21/2020	6:42:43 PM	1	0.393	19.99558	25 um Ni	Starting experimental scan on Ni foil observing K shell		~10 mJ	0.5 ns	60 fs
7/21/2020	6:49:57 PM	2	0.361	19.71816	25 um Ni	Starting experimental scan on Ni foil observing K shell	K-a still > He-a	~10 mJ	0.25 ns	60 fs
7/21/2020	6:54:43 PM	3	0.35	19.07796	25 um Ni	Starting experimental scan on Ni foil observing K shell		~10 mJ	0.125 ns	60 fs
7/21/2020	7:00:27 PM	4	0.391	19.16332	25 um Ni	Starting experimental scan on Ni foil observing K shell		~10 mJ	62.5 ps	60 fs
7/21/2020	7:06:55 PM	5	0.388	19.16332	25 um Ni	Starting experimental scan on Ni foil observing K shell		~10 mJ	30 ps	60 fs
7/21/2020	7:11:33 PM	6	0.441	18.6725	25 um Ni	Starting experimental scan on Ni foil observing K shell		~10 mJ	single pulse no prepulse	60 fs
7/21/2020	7:20:37 PM	7	0.368	18.3524	25 um Ni	Starting experimental scan on Ni foil observing K shell		~10 mJ	15 ps	60 fs
7/21/2020	7:26:46 PM	8	0.37	18.43776	25 um Ni	Starting experimental scan on Ni foil observing K shell		~10 mJ	trying for 0 ps	60 fs
7/22/2020	12:31:03 PM	9	0.218	10.3499	CH 4 micron	Espec/Proton Wedge/ Bubble Detectors/ NTOF (shoul		10 mJ	1 ns prepulse	60 fs
7/22/2020	12:34:15 PM	10	0	11.54494	CH 4 micron	test energy shot		10 mJ	1 ns prepulse	60 fs
7/22/2020	12:44:00 PM	11	0	8.77074	CH 4 micron	test energy shot		10 mJ	1 ns prepulse	60 fs
7/22/2020	12:58:17 PM	12	0.406	16.30376	CH 4 micron	Shooting again with 1 ns delay	Shot 12 prepulse may have be	10 mJ	1 ns prepulse	60 fs
7/22/2020	1:01:44 PM	13	0.404	18.43776	CH 4 micron	blocked prepulse		10 mJ	no prepulse	60 fs
7/22/2020	1:07:56 PM	14	0.383	17.8189	CH 4 micron	1 ns prepulse again (may have been misaligned last sh		10 mJ	1 ns	60 fs

Example: Neutron Generation

- High intensity laser irradiates an array of deuterated polyethylene (CD_2) nanowires to drive a D-D fusion reaction
- D-D fusion reaction generates a neutron burst
- The maximum number of neutrons per shot was 3.6 X 10^6 with 60 femtosecond laser pulses at 1.64 J
- Limited to shot/min, but high repetition rates are desirable for fusion science and high energy density physics



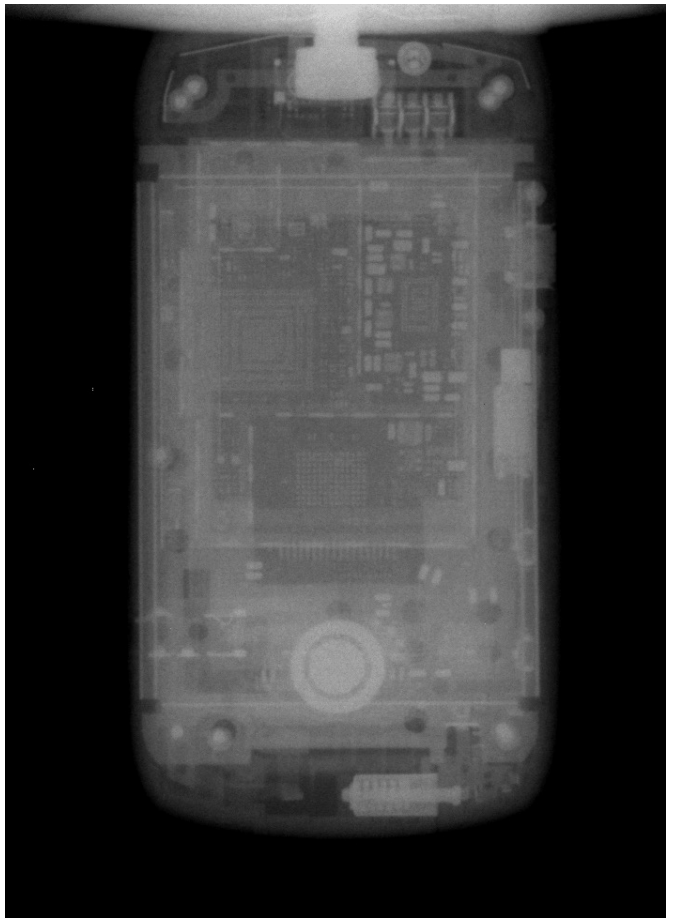
Curtis, A., *et al. Nature Communication* **9,** 1077 (2018).

Example: X-Ray Tomography

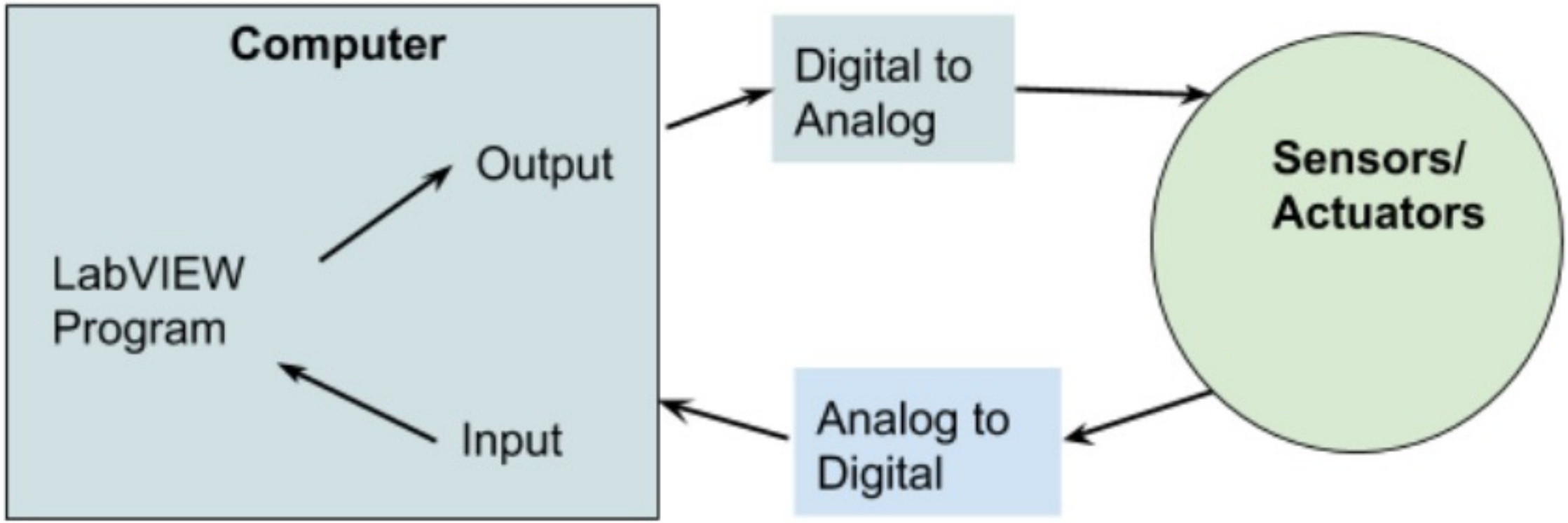
- Using X-Ray sources (created by irradiating a tantalum foil with high intensity laser) to generate tomographic images of electronics

- The GIF to the left consists of 72 shots (taken at about a shot per minute)

- An automated laser system running at 3.3 Hz can complete the same scan in about 20 seconds.



Data Acquisition Control Loop

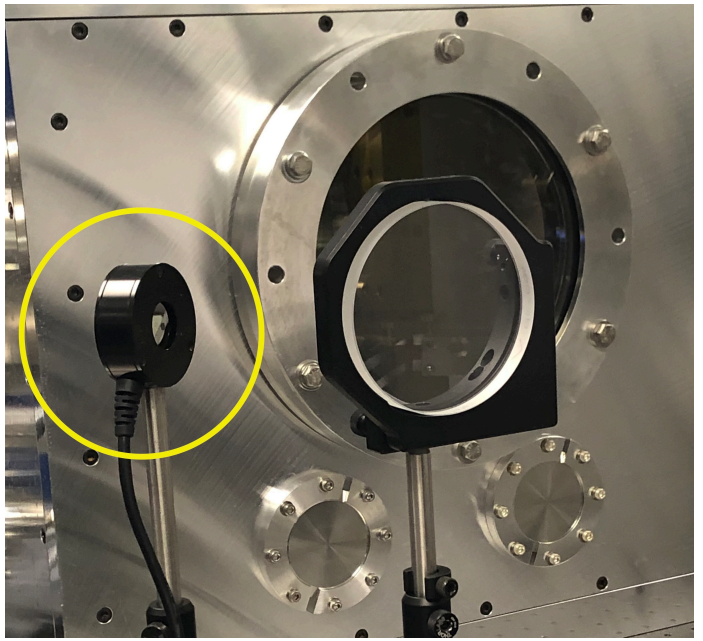


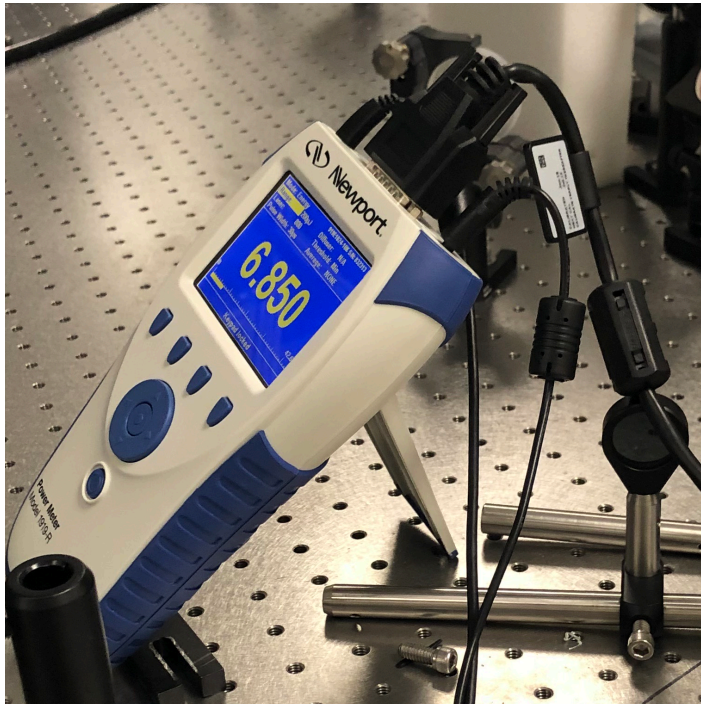
Newport 1919-R Power Meters

1. Program receives measurements from both power meters

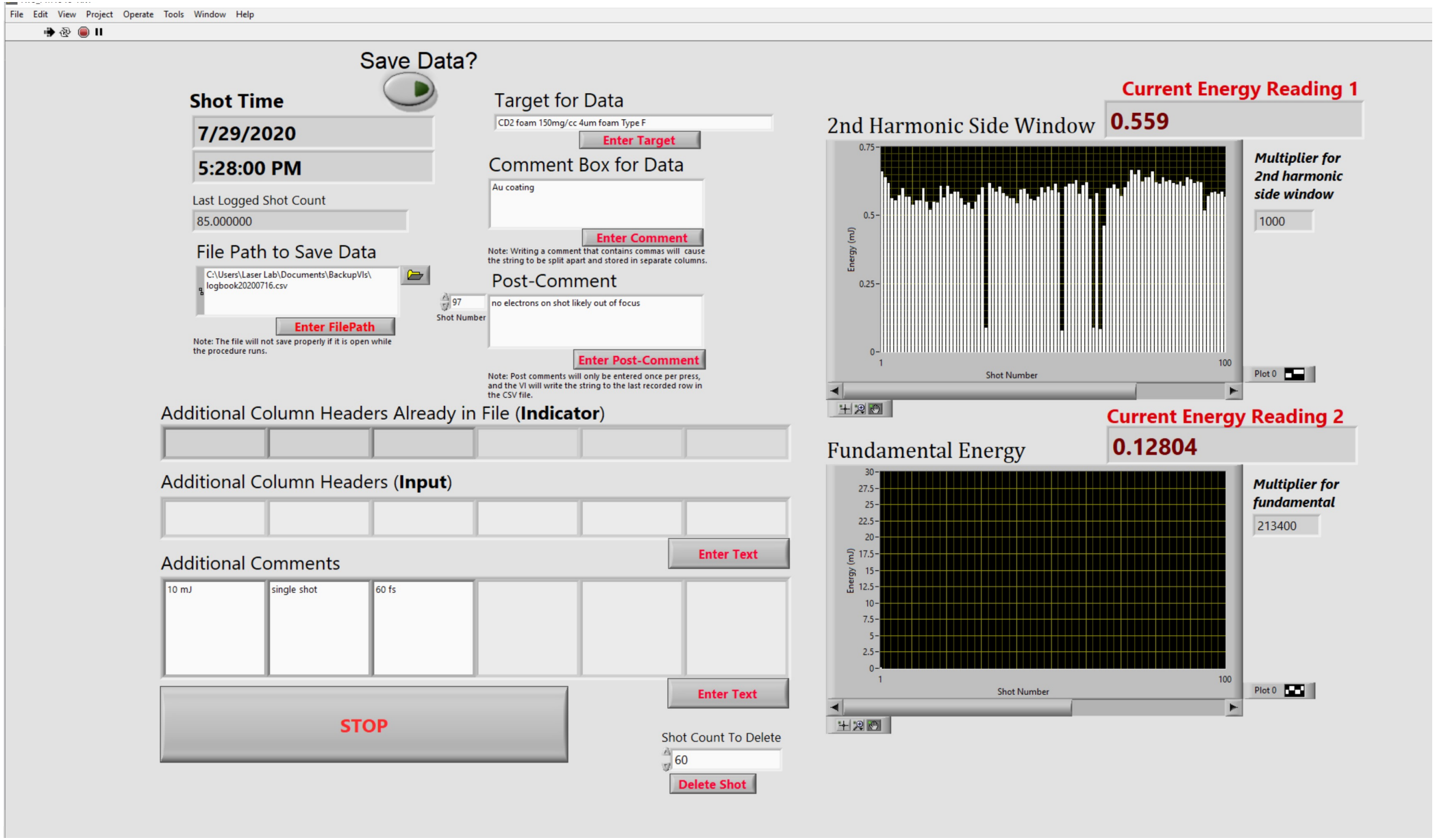
2. Plots measurements and monitors
Fundamental And Second Harmonic Energy
- Uses bar graphs and arrays to plot
information in real time

3. Logs Data as a CSV file
- Allows user to add sub-headers,
add comments, delete shots,
and to change file-path

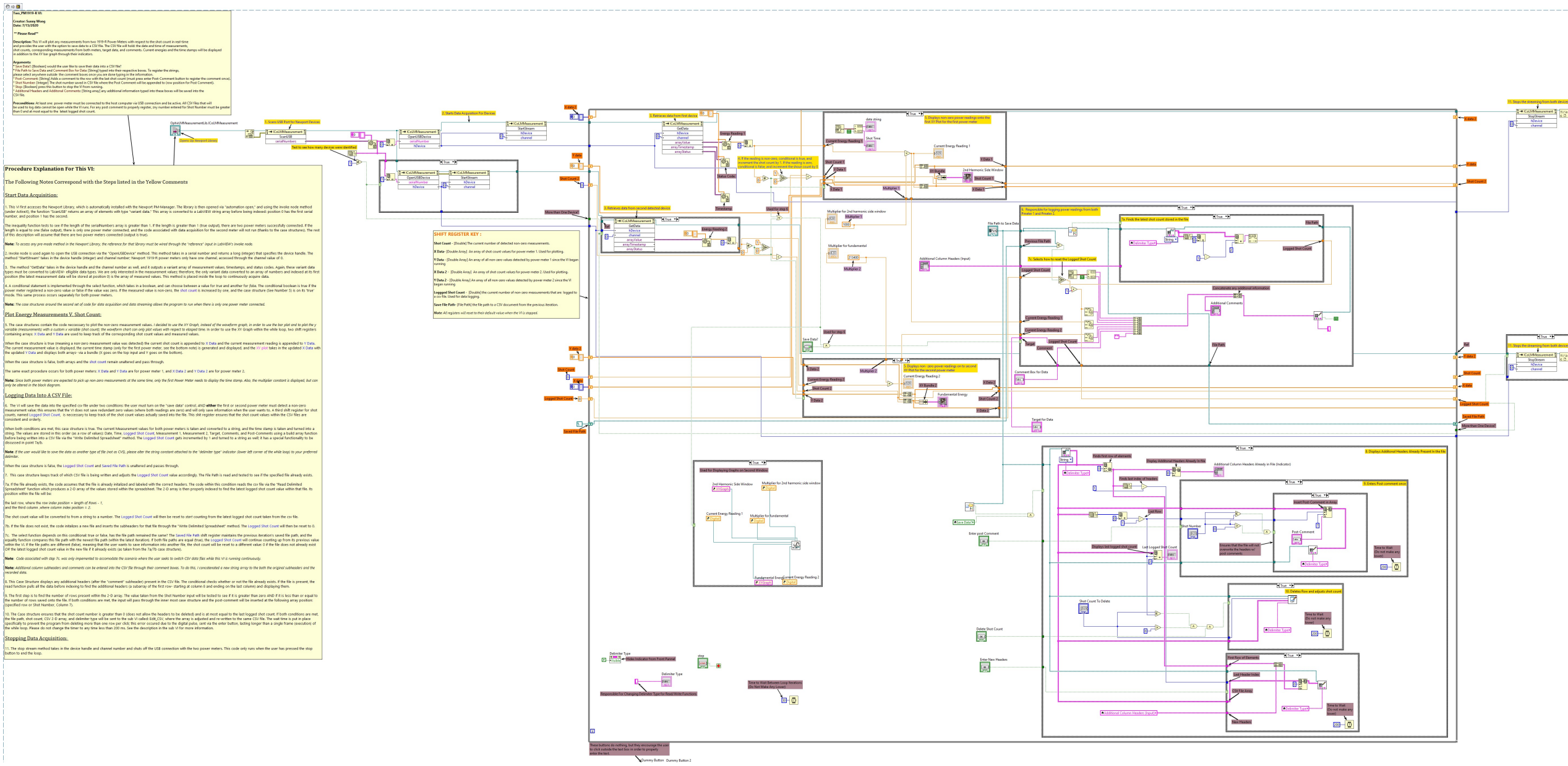




Graphical Interface



Code

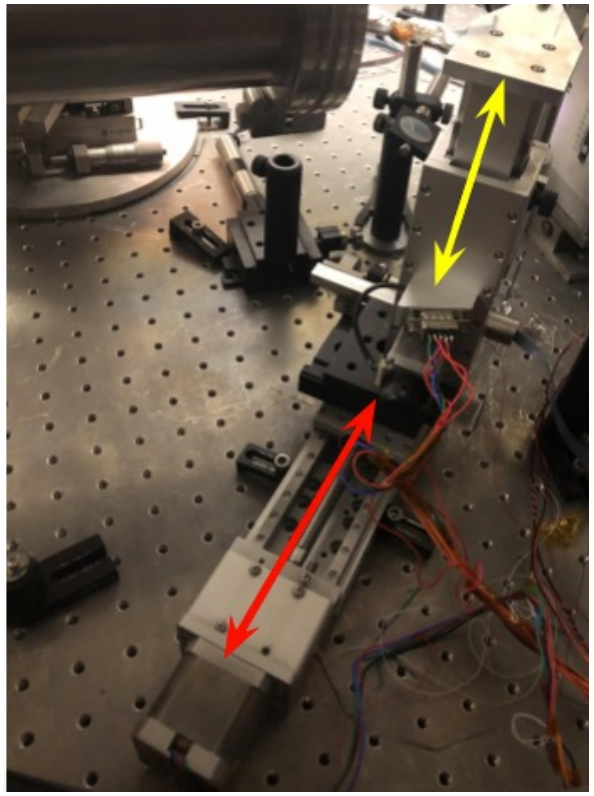

Two_PM1919-R VI:
Creator: Sunny Wong
Date: 7/13/2020
** Please Read**
Description: This VI will plot any measurements from two 1919-R Power Meters with respect to the shot count in real-time and provides the user with the option to save data to a CSV file. The CSV file will hold the data and time of measurements, shot counts, corresponding measurements from both meters, target data, and comments. Current energies and the time stamps will be displayed in addition to the XY live graph through their indicators.
Arguments:
* Save Data?: [Boolean] would the user like to save their data into a CSV file?
* File Path to Save Data and Comment Box for Data: [String] typed into their respective boxes. To register the strings, please select anywhere outside the comment boxes once you are done typing in the information.
* Post-Comment: [String] Adds a comment to the row with the last shot count (must press enter Post-Comment button to register the comment once).
* Shot Number: [Integer] The shot number saved in CSV file where the Post Comment will be appended to (row position for Post Comment).
* Stop: [Boolean] press this button to stop the VI from running.
* Additional Headers and Additional Comments: [String array] any additional information typed into these boxes will be saved into the CSV file.
Preconditions: At least one power meter must be connected to the host computer via USB connection and be active. All CSV files that will be used to log data cannot be open while the VI runs. For any post comment to properly register, any number entered for Shot Number must be greater than 0 and at most equal to the latest logged shot count.
OpenUSBMeasurementLib.lvclsVIMeasurement
Opens the Newport Library
1. Scans USB Port for Newport Devices
Test to see how many devices were identified
2. Starts Data Acquisition For Devices
X data 2
Y data
Shot Count 2
More than One Device
Y data 2
Read Count
X data
Logged Shot Count
Saved File Path
1. Retrieves data from first device
Energy Reading 1
Status Code
Timestamp
Used for step 4
3. Retrieves data from second detected device
Energy Reading 2
4. If the reading is non-zero, conditional is true, and increment the shot count by 1. If the reading is zero, conditional is false, and increment the shot count by 0
Shot Count
X Data 1
Multiplier 1
Multiplier for 2nd harmonic side window
Multiplier for fundamental
Multiplier 2
Current Energy Reading 1
2nd Harmonic Side Window
X Data 2
Current Energy Reading 2
Fundamental Energy
Shot Count 2
Displays non-zero power readings onto the first VI Plot for the first power meter
date string
Shot Time
Displays non-zero power readings onto the second VI Plot for the second power meter
6. Responsible for logging power readings from both P meter 1 and Primary 2.
File Path to Save Data
Reference File Path
Additional Column Headers (Input)
7a. Finds the latest shot count stored in the file
Delimiter Type
Logged Shot Count
7c. Selects how to reset the Logged Shot Count
Concatenate any additional information
Additional Comments
Comment Box for Data
Target for Data
Used for Displaying Graphs on Second Window
Used for step 5
Save Data?
Enter post Comment
8. Displays Additional Headers Already Present in the file
Finds first array of elements
Finds last index of headers
Additional Column Headers Already in File (Indicator)
9. Enters Post comment once
Insert Post-Comment in Array
Post-Comment
Shot Number
Displays last logged shot count
Last Logged Shot Count
Ensures that the file will not overwrite the headers w/ post comments
Time to Wait (Do not make any issues)
10. Deletes Row and adjusts shot count
Shot Count To Delete
Delete Shot Count
Enter New Headers
First Row of Elements
Last Header Index
EOF File Array
New Headers
Additional Column Headers (Input)
Hides Indicator From Front Panel
stop
Responsible For Changing Delimiter Type for Read/Write Functions
Time to Wait Between Loop Iterations (Do Not Make Any Issues)
These buttons do nothing, but they encourage the user to click outside the text box in order to properly enter the text.
Dummy Button
Dummy Button 2
11. Stops the streaming from both devices
Ref
Procedure Explanation For This VI:
The Following Notes Correspond with the Steps listed in the Yellow Comments
Start Data Acquisition:
1. This VI first accesses the Newport Library, which is automatically installed with the Newport PM-Manager. The library is then opened via "automation open," and using the invoke node method (under Activex), the function "ScanUSB" returns an array of elements with type "variant data." This array is converted to a LabVIEW string array before being indexed: position 0 has the first serial number, and position 1 has the second.
The inequality function tests to see if the length of the serialNumbers array is greater than 1. If the length is greater than 1 (true output), there are two power meters successfully connected. If the length is equal to one (false output), there is only one power meter connected, and the code associated with data acquisition for the second meter will not run (thanks to the case structures). The rest of this description will assume that there are two power meters connected (output is true).
Note: To access any pre-made method in the Newport Library, the reference for that library must be wired through the "reference" input in LabVIEW's invoke node.
2. Invoke node is used again to open the USB connection via the "OpenUSBDevice" method. This method takes in a serial number and returns a long (integer) that specifies the device handle. The method "StartStream" takes in the device handle (integer) and channel number; Newport 1919-R power meters only have one channel, accessed through the channel value of 0.
3. The method "GetData" takes in the device handle and the channel number as well, and it outputs a variant array of measurement values, timestamps, and status codes. Again, these variant data types must be converted to LabVIEW- eligible data types. We are only interested in the measurement values; therefore, the only variant data converted to an array of numbers and indexed at its first position (the latest measurement data will be stored at position 0) is the array of measured values. This method is placed inside the loop to continuously acquire data.
4. A conditional statement is implemented through the select function, which takes in a boolean, and can choose between a value for true and another for false. The conditional boolean is true if the power meter registered a non-zero value or false if the value was zero. If the measured value is non-zero, the shot count is increased by one, and the case structure (See Number 5) is on its 'true' mode. This same process occurs separately for both power meters.
Note: The case structures around the second set of code for data acquisition and data streaming allows the program to run when there is only one power meter connected.
Plot Energy Measurements V. Shot Count:
5. The case structures contain the code necessary to plot the non-zero measurement values. I decided to use the XY Graph, instead of the waveform graph, in order to use the bar plot and to plot the y variable (measurements) with a custom x variable (shot count); the waveform chart can only plot values with respect to elapsed time. In order to use the XY Graph within the while loop, two shift registers containing arrays: X Data and Y Data are used to keep track of the corresponding shot count values and measured values.
When the case structure is true (meaning a non-zero measurement value was detected) the current shot count is appended to X Data and the current measurement reading is appended to Y Data. The current measurement value is displayed, the current time stamp (only for the first power meter, see the bottom note) is generated and displayed, and the XY plot takes in the updated X Data with the updated Y Data and displays both arrays- via a bundle (X goes on the top input and Y goes on the bottom).
When the case structure is false, both arrays and the shot count remain unaltered and pass through.
The same exact procedure occurs for both power meters: X Data and Y Data are for power meter 1, and X Data 2 and Y Data 2 are for power meter 2.
Note: Since both power meters are expected to pick up non-zero measurements at the same time, only the first Power Meter needs to display the time stamp. Also, the multiplier constant is displayed, but can only be altered in the block diagram.
Logging Data Into A CSV File:
6. The VI will save the data into the specified csv file under two conditions: the user must turn on the "save data" control, AND either the first or second power meter must detect a non-zero measurement value; this ensures that the VI does not save redundant zero values (where both readings are zero) and will only save information when the user wants to. A third shift register for shot counts, named Logged Shot Count, is necessary to keep track of the shot count values actually saved into the file. This shift register ensures that the shot count values within the CSV files are consistent and orderly.
When both conditions are met, this case structure is true. The current Measurement values for both power meters is taken and converted to a string, and the time stamp is taken and turned into a string. The values are stored in this order (as a row of values): Date, Time, Logged Shot Count, Measurement 1, Measurement 2, Target, Comments, and Post-Comments using a build array function before being written into a CSV file via the "Write Delimited Spreadsheet" method. The Logged Shot Count gets incremented by 1 and turned to a string as well; it has a special functionality to be discussed in point 7a/b.
Note: If the user would like to save the data as another type of file (not as CVS), please alter the string constant attached to the 'delimiter type' indicator (lower left corner of the while loop) to your preferred delimiter.
When the case structure is false, the Logged Shot Count and Saved File Path is unaltered and passes through.
7. This case structure keeps track of which CSV file is being written and adjusts the Logged Shot Count value accordingly. The File Path is read and tested to see if the specified file already exists.
7a. If the file already exists, the code assumes that the file is already initialized and labeled with the correct headers. The code within this condition reads the csv file via the "Read Delimited Spreadsheet" function which produces a 2-D array of the values stored within the spreadsheet. The 2-D array is then properly indexed to find the latest logged shot count value within that file. Its position within the file will be:
the last row, where the row index position = length of Rows - 1,
and the third column ,where column index position = 2.
The shot count value will be converted to from a string to a number. The Logged Shot Count will then be reset to start counting from the latest logged shot count taken from the csv file.
7b. If the file does not exist, the code initalizes a new file and inserts the subheaders for that file through the "Write Delimited Spreadsheet" method. The Logged Shot Count will then be reset to 0.
7c. The select function depends on this conditional: true or false, has the file path remained the same? The Saved File Path shift register maintains the previous iteration's saved file path, and the equality function compares this file path with the newest file path (within the latest iteration). If both file paths are equal (true), the Logged Shot Count will continue counting up from its previous value within the VI. If the file paths are different (false), meaning that the user wants to save information into another file, the shot count will be reset to a different value: 0 if the file does not already exist OR the latest logged shot count value in the new file if it already exists (as taken from the 7a/7b case structure).
Note: Code associated with step 7c, was only impemented to accomodate the scenario where the user seeks to switch CSV data files while this VI is running continuously.
Note: Additional column subheaders and comments can be entered into the CSV file through their comment boxes. To do this, I concatenated a new string array to the both the original subheaders and the recorded data.
8. This Case Structure displays any additional headers (after the "comment" subheader) present in the CSV file. The conditional checks whether or not the file already exists. If the file is present, the read function pulls all the data before indexing to find the additional headers (a subarray of the first row- starting at column 6 and ending on the last column) and displaying them.
9. The first step is to find the number of rows present within the 2-D array. The value taken from the Shot Number input will be tested to see if it is greater than zero AND if it is less than or equal to the number of rows saved onto the file. If both conditions are met, the input will pass through the inner most case structure and the post-comment will be inserted at the following array position: (specified row or Shot Number, Column 7).
10. The Case structure ensures that the shot count number is greater than 0 (does not allow the headers to be deleted) and is at most equal to the last logged shot count. If both conditions are met, the file path, shot count, CSV 2-D array, and delimiter type will be sent to the sub VI called Edit_CSV, where the array is adjusted and re-written to the same CSV file. The wait time is put in place specifically to prevent the program from deleting more than one row per click; this error occured due to the digital pulse, sent via the enter button, lasting longer than a single frame (execution) of the while loop. Please do not change the timer to any time less than 200 ms. See the description in the sub VI for more information.
Stopping Data Acquisition:
11. The stop stream method takes in the device handle and channel number and shuts off the USB connection with the two power meters. This code only runs when the user has pressed the stop button to end the loop.
SHIFT REGISTER KEY :
Shot Count - [Double] The current number of detected non-zero measurements.
X Data- [Double Array]. An array of shot count values for power meter 1. Used for plotting.
Y Data - [Double Array] An array of all non-zero values detected by power meter 1 since the VI began running.
X Data 2 - [Double Array]. An array of shot count values for power meter 2. Used for plotting.
Y Data 2 - [Double Array] An array of all non-zero values detected by power meter 2 since the VI began running.
Logged Shot Count - [Double] the current number of non-zero measurements that are logged to a csv file. Used for data logging.
Save File Path- [File Path] the file path to a CSV document from the previous iteration.
Note: All registers will reset to their default value when the VI is stopped.

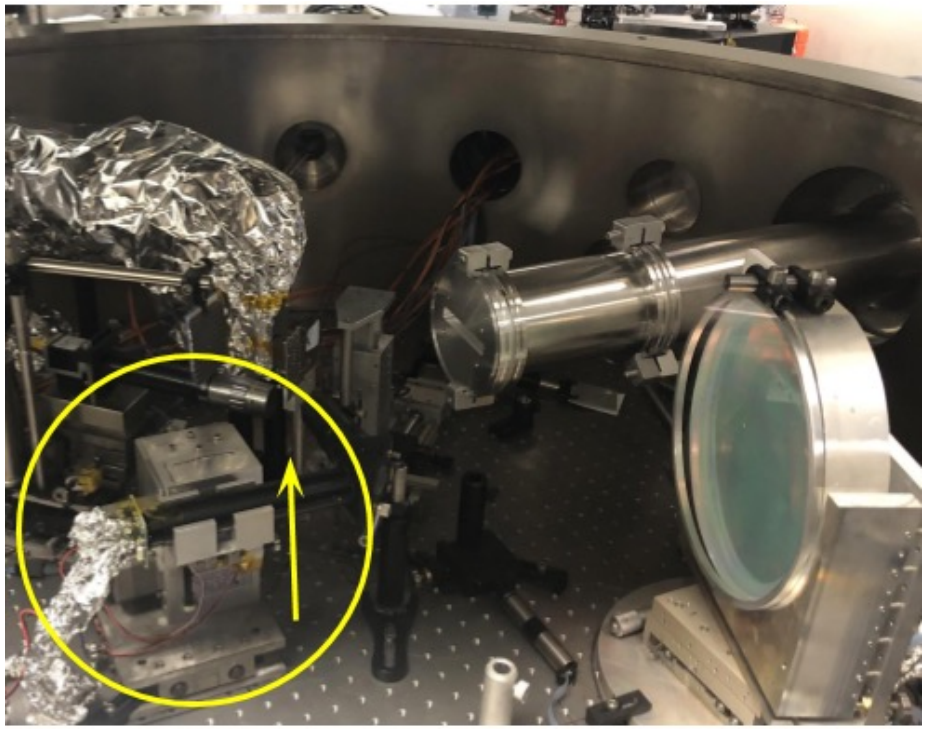
Maneuvering the Camera System and Target Mount

1. LabVIEW program sends TTL signals to relay switches via NI CompactDAQ Digital I/O board

2. Relay Switches are wired in parallel to the physical stepper motor system built by CSU

3. Program receives and displays response signal to confirm that the motor controls are moving





Firing the Laser

1. Check if cameras are down

2. If the above condition is satisfied, the laser will fire.

In the future, an interlocking camera system will help protect equipment.

More functionality will be added later.

Summary

The LabView program…

1. Displays and logs data from the two power meters
2. Can maneuver the camera and target system
3. Can fire the laser after checking if the camera is down

Email: Sw758@cornell.edu