Computer assisted electroretinogram data acquisition and analysis

Nick Balge, Andrew Dias, Whitney Johnson, Dan Jonovic

Advisor: Professor Mitch Tyler

Client: Dr. Bikash Pattnaik

12/11/2008

Contents

Table of Figures

1. Abstract

The electroretinogram (ERG) records voltage responses of the retina to light stimuli. This project's goal was to design an interface in LabVIEW that would monitor, control, and record values from the ERG corresponding to liquid delivery, light intensity, flash duration, flash frequency, and voltage responses. We integrated all of these controls while updating the graphical user interface (GUI) so that it was easy to navigate in a dark room under a red filter. Up to eight different solutions can be delivered, and only one solution is delivered at a time. Light intensity is controlled by setting a number on a filter wheel through the interface. An option to export data to Microsoft Excel is included along with real-time graphs. All relevant functionality present in a previous version of the interface was preserved, and input points were organized into tabs so that real-time graphs could be enlarged and the interface did not look cluttered. The new interface is functional on the client's setup and provides the capabilities to properly record and analyze the ERG.

2. Background

An electroretinogram (ERG) is used to measure the electrical responses to light of various cell layers in the retina (Medline, 2007). This response is gathered by placing an electrode either on the retina or the skin surrounding the eye, connecting the other electrode to ground, and stimulating the retina with light; the intensity and wavelength of the light stimulus is often varied.

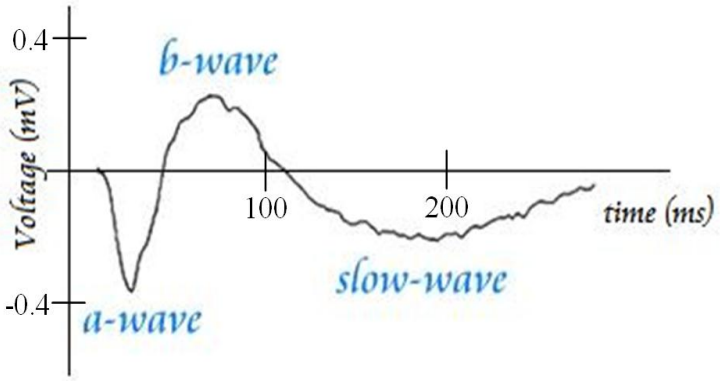

***Figure 1**: Example of ERG wave [Creel, 2008]. The a-wave is negative, the b-wave is positive, and the slow-wave is negative.*

The result of the light stimulus is a triphasic wave that consists of a, b, and slow-waves (Figure 1). These different components represent separate layers of cells in the retina. The a-wave is called the "late

receptor potential" and reflects the physiological health of photoreceptors in the outer retina. The b-wave reflects the health of the outer cell layers of the retina and the slow-wave provides information on the bipolar and ganglion cells (Creel, 2008; Ye and Goo, 2007). Measurements taken from the wave include the amplitude of the a-wave from time zero, the amplitude of the b-wave with respect to the a-wave, the time between the flash and trough of the a-wave, and the time between the flash and the peak of the b-wave (Figure 2). The measurement of the time from flash onset to wave peak is called the implicit time. The b/a wave ratio is a good indicator of ischaemia, or insufficient blood flow in a central retinal vein obstruction (Matsui, Y *et al*, 1994).

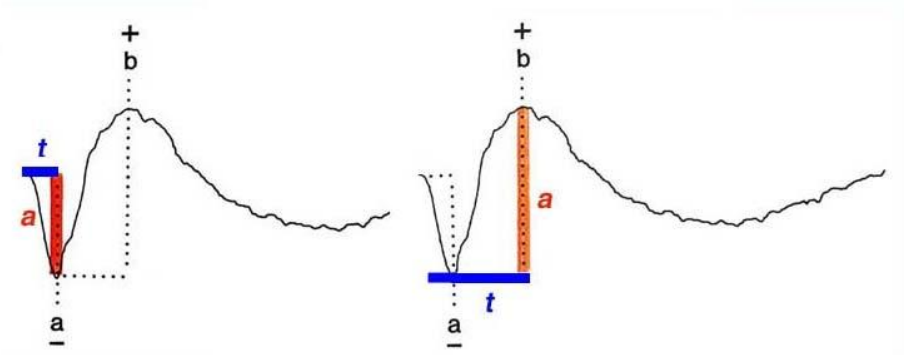

Figure 2: *Measuring the amplitudes and times of waves*

There are a few ways to record the ERG. The environment this is performed in is illuminated with red light, since it has the lowest amount of energy in the visible spectrum. The most common way to perform the ERG is to dark-adapt the retina in this environment, attach electrodes, and record the ERG with a combination of blue, red, and white flashes. Dark adaptation involves exposing the retina to darkness, resulting in a greater sensitivity to luminance. The test would be repeated with moderate background illumination so that the retina is light-adapted (Creel, 2008). The use of log filters to reduce light intensity limits the ERG to recording rod activity because they are much more sensitive than cones.

Different solutions are known to block the activation of various cell types in the retina. For example, the pharmacological agent L – AP5 has been shown to significantly reduce the b-wave amplitude in ERGs produced from flashes above the intensity 100 q/rod (Green and Kapousta-Bruneau, 2006). Other chemicals reduce cone concentrations, thereby limiting the cell types responding to the stimulus. This type of research helps to isolate different cells layers, making it easier to distinguish problem areas of retina.

3. Applications/Motivation:

ERGs have been used to diagnose a number of eye diseases. Retinal physiology can also reflect systemic disorders such as liver and kidney disease (Creel, 2008). These techniques have been in use for a number of years. Currently, there is a need for many new drugs for eye disease to be developed and tested. An ERG provides an optimal research environment because tests can be conducted on an isolated retina, which allows various solutions to be in contact with the sample. Although systems exist for the measurement of ERGs in animals, it is important to create a computer program that will help integrate components of a given test design and allow researchers to visualize and analyze their data.

4. Problem Statement:

A computer interface will be designed to manipulate ERG data acquisition and analysis variables. These aspects include regulating liquid exchange through a valve controller, incorporating a logarithmic light wheel, and exporting data in a simple and organized format. See Appendix A for more details.

5. Lab layout:

Our design constraints follow the setup of our client's lab (Figure 3).

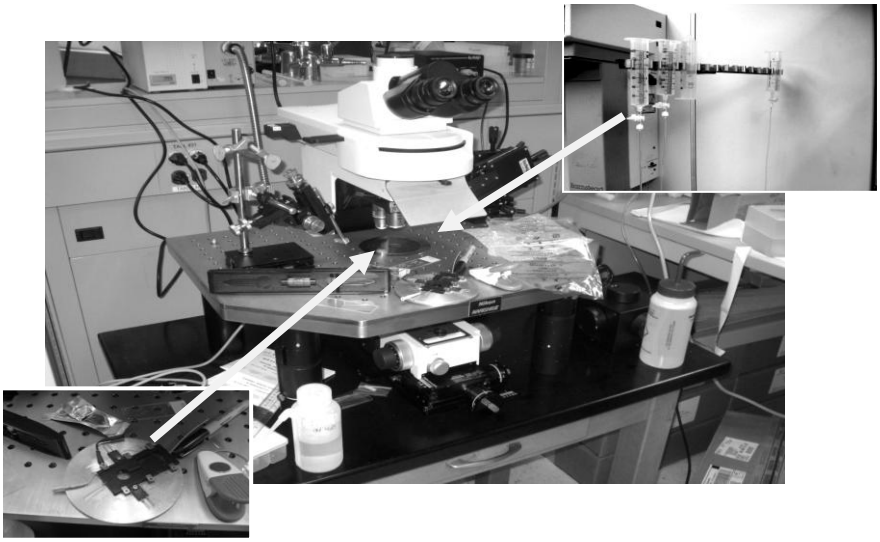

Figure 3: *Organization of our client's lab. A rat retina is placed on a slide (bottom left) under the microscope to record the ERG. The retina is perfused with drugs or solution from the liquid delivery system (top right).*

Our client places electrodes on the retina and records the ERG. Light intensities are set via a light wheel and liquids are delivered via the solution delivery system. Up to eight solutions can be prepared, but the retina will only be perfused with one solution at a time (Figures 3, 4). The light wheel, solution delivery system, and ERG are all connected to a computer running Microsoft Windows XP. All have their own data acquisition solutions (DAQs) that allow interfacing with LabVIEW.

Figure 4*: A solution valve opener allows controlled flow of a chemical (Audie Technology, 2008).*

6. Design constraints:

The interface should help perform retinal research, and constraints are defined by the need to acquire suitable ERGs under different conditions. The functional requirements of the interface can be summarized as follows:

- Display of accurate waveforms in real-time
- Adjustment of light intensity and duration
- Incorporation of solution (drug) delivery – only one at a time
- Exporting of data in a format readable by Microsoft Excel

The waveform of the ERG depends on whether the retina is dark or light adapted, so testing will be performed at different intensities of unidirectional light. This system must be able to change light intensities over the retina. Additionally, different drugs must be perfused over the retina one at a time. Our interface must be able to manage the solution delivery system and switch between drugs. It must also deliver only one drug at a time to minimize the possibility of the retina running dry.

Further, real-time graphical display and the ability to export raw data must be supported. It is desirable to view the ERG waveform as it is being recorded to verify that data is being acquired. Raw data should be exported into another format for further analysis because LabVIEW is not the optimal environment to perform rigorous calculations. Although these features are required additions, functionality from the original interface should not be removed.

Performance requirements must meet the following criteria:

- Usable with and without a red filter
- Readable
- Simple and intuitive

The client runs experimental trials in a dark room because the retina should be stimulated as little as possible from external light. The monitor is normally one such light source, so a red filter is in place over it during testing so that emission of light is minimized. The interface must be usable in ambient room light without a red filter and in the dark testing environment with the red filter. All text and graphics must be large enough to read and select buttons easily. Large graphics, buttons, and text would make it easy to learn how to use the interface. For similar reasons, the interface should also be simple and intuitive. A simple interface would also increase efficiency in testing because it would be more difficult to make mistakes.

7. The current interface

The user of the interface interacts with the graphical user interface (GUI), also known as the front end of the system (Figure 5).

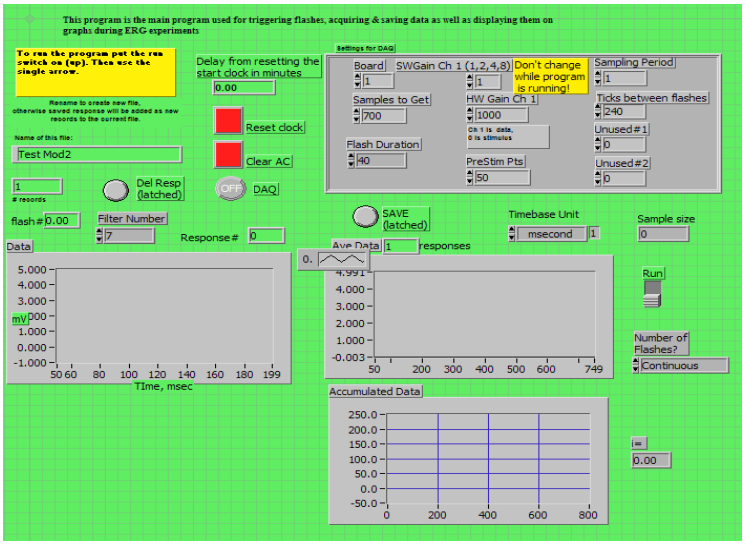

Figure 5: *GUI for the original interface*

The main components of the GUI are the functionality and the usability. This interface displays graphs and allows the user to change some settings like sample duration and the number of consecutive samples – it has some basic functionalities.

This interface lacks:

1) The ability to change the light intensity
2) Options to set and change the solution that will be delivered
3) An option to export data into another format

To make the GUI more usable, the graphs, buttons, and forms may need to be reorganized. The interface looks cluttered because there are three graphs that take up a lot of room and because there are many options that have to be changed by keyboard entry. If fewer boxes were used and alternative methods of input were considered, the GUI may be easier to use. Providing tabs may be a way to reduce the number of boxes the user views on one screen and would be a way to make boxes and font sizes larger, increasing usability. Tradeoffs between allowing the user to manipulate many features on one screen and size of boxes and forms need to be considered and optimized.

8. Design 1: Tweak Existing Interface

The first design is based on keeping and slightly modifying the existing backend of the program that the client and his associates have developed (Figure 6). Because the graphical programming has already been written and functions properly, this would save time. Also, programming in LabVIEW is advantageous because it efficiently integrates hardware and software, thereby promoting data collection.

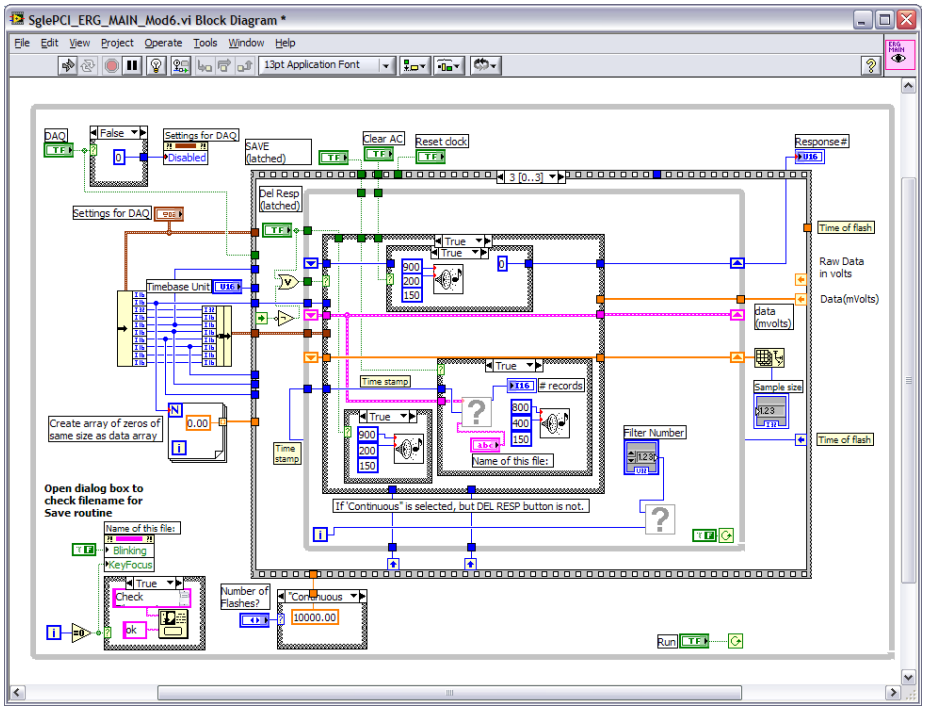

Figure 6: *The existing backend*

A. Functionality

The most important part of each individual design plan is to add the necessary functionality to the client's system. This includes all of the issues mentioned in the PDS such as solution valve integration, light wheel control, and data storage. In this design, these tasks would be accomplished by rewriting some of the LabVIEW code. As mentioned above, the majority of the code would remain the same, and only enough would be rewritten as is necessary to include all of the functionality.

B. GUI Modification

The aspect of this design that makes it most unique involves the modification of the graphical user interface. In this plan, the GUI would undergo only minor modifications. This would allow for the correction of some of the simpler features that could use improvement. Again, this option would also save time, which would be advantageous if integration of the necessary functionalities takes longer than expected (Hobart).

The first design component would be to intuitively rearrange all of the controls. This process would involve grouping related commands together. Given a logical arrangement of controls, the user would be able to find the target button with the greatest possible speed and least possible frustration. Rearranging could also aid in clearly defining which items are displays and which are user controls. These modifications could be easily implemented, as LabVIEW allows the dragging of objects to their necessary positions.

Another simple improvement would be to change the color scheme of the interface. Because the system operates from behind a red filter and in the dark, it should have a sufficient level of contrast. While the current display does have adequate contrast behind the filter, there is still room for improvement in this regard. Additionally, when the red filter is removed the color scheme could potentially create frustration or annoyance if the user is staring at it for too long. For this reason, an improvement of color harmony would be beneficial. In modifying the color scheme, steps would be taken to decrease the overwhelming feeling while making sure not to create a color scheme that fails to fully capture the user's attention in the first place (Morton, 2008). In redesigning the color of the GUI care would be taken to find an adequately contrasting scheme to meet the client's needs, while adhering to the previously mentioned principles.

Finally another fairly simple fix would involve changing the existing font sizes. Overall, font size would be increased because the most important design aspect of a visible element is readability. In some cases this would involve a minor rewording of the labels so that they don't take up too much space. After making sure that all of the words can be accurately read, font sizes could be altered to convey

their importance to the execution of the experiment. This would direct the user's eyes to the buttons which he should be using the most; further increasing ease and speed of execution.

In this design, the speed of development would be maximized, allowing more time for testing and the resolution of bugs. Although a rapid solution, a design involving only a slight modification of the GUI might limit the potential for improvement in that area. With this design, further modification of this program could be very complex unless programming structures are simplified.

9. Design 2: Complete GUI Overhaul

The second design plan incorporates some of the same features as the first. This design would again build off of the existing graphical programming that the client has written in LabVIEW. All of the necessary functionalities would again be added to this program with few other changes to the code.

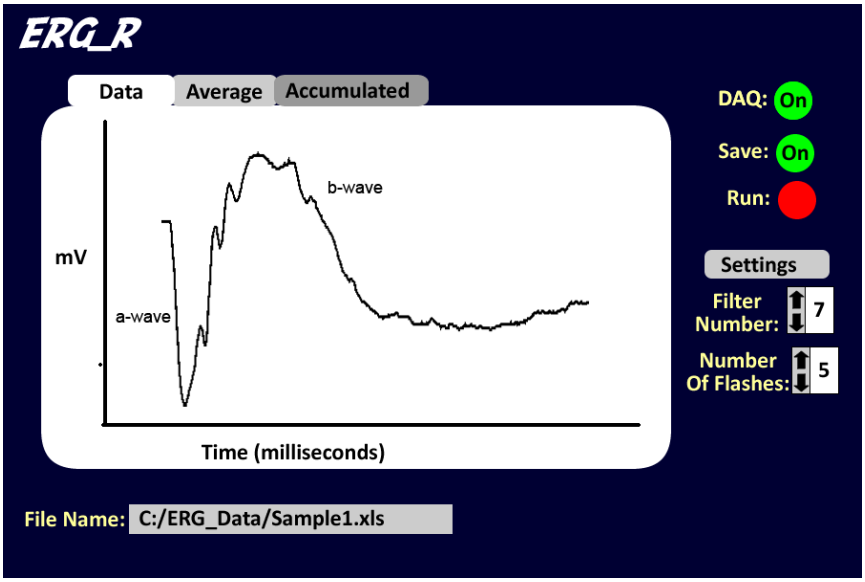

Figure 7: *Possible GUI Overhaul*

A. GUI Plans

As with design one, design two is unique in its approach to the GUI. In this case, however, the GUI would undergo a complete overhaul (Figure 7). This process would allow for a fresh start and the opportunity to include all of the modifications from the first design while adhering to various GUI design principles that would otherwise be difficult to incorporate if only minor changes were possible. This plan would be more time consuming but would open the door to creating a user friendly interface.

B. GUI Design

A staple characteristic of good GUIs is simplicity. The user should not feel overwhelmed by the number of visible buttons on program start. Also, navigating through a large number of controls will usually take longer than desired. To avoid this, the number of items on the main GUI would be greatly reduced. Only the most frequently used features would be included so that the user's eyes fall directly on these items (Hobart).

Along with simplicity, the GUI must also have functionality. Completely removing some of the features from the old GUI would not be beneficial and might even break the requirements for the design. Therefore, rather than getting rid of controls to increase simplicity, various controls would be organized into labeled, tabbed menus. Using menus, even more controls could be added while still encouraging a strong feeling of simplicity. Another benefit of adding menus would be to foster the organization of the GUI as a whole, by again grouping related commands together as mentioned for the first design. A related change would also involve moving displayed graphs to tabbed windows rather than trying to display them all at once. This would greatly increase their possible size and visibility.

Another equally important quality of a successful GUI is consistency. As various windows and menus are incorporated, care must be taken to make sure that the layout of each screen resembles the layout of

the others. Another important point is to ensure consistency of labels between windows. Even slight mislabeling could cause the user to become confused about a value that is inputted and run the experiment in a way other than is intended.

Finally, familiarity is a quality that aids a user in operating any GUI. Concerning the GUI as a whole, its layout and appearance should be similar to frequently used programs. However, when incorporating new ideas, care should be taken not to create something entirely unique, even if it works. Familiarity at the level of single controls is also important and would be improved by changing the way that certain commands are inputted. For example, a box that previously required the user to type in a value manually or adjust the value with an arrow might be better suited to be a drop down menu. The advantage of such menus is that the user is only able to set valid values, adding to the efficiency of program navigation.

Aside from these encompassing concepts of GUI design, creating one from scratch would permit the addition of other optional features are deemed necessary as the project progresses. In these ways, the second design option increases the potential for a satisfactory user interface.

A complete overhaul of the GUI would allow for much more freedom in design that would result in a more intuitive user interface with simplicity, consistency, and familiarity. This would help meet the standards of good GUI design. An overhaul of the GUI would take more time than a tweak, though minimal time would be spent programming LabVIEW code. Once again, further modification of this program could be very complex unless programming structures are simplified.

10. Design 3: Start over from scratch

The third design option encompasses an entire redesign of the project and involves the comparison of different solutions available as well as other programming languages. This establishes which computing environment would best address the problem statement.

A. Existing Solutions

Electro-Diagnostic Imaging offers an ERG measuring solution, though it does not allow the capability of interfacing with additional hardware. While it performs similar measurement operations, it is designed for clinical settings and not a research environment (Electro-Diagnostic).

AD Research has both hardware and software systems developed for data analysis and acquisition (AD Instruments). This is similar to LabVIEW, though their LabChart software already has measurement capabilities and data analysis built in. While there would not be software programming involved in this system, it does not allow the addition of capabilities that are required, including solution delivery and light wheel control.

B. Programming Languages

Using a Unix-based environment would allow better control over timings of actions that need to be performed in milliseconds. While the flash duration and other settings are on the order of 40ms, the Windows time management meets the needs of this project even though it is less precise than Unix-based operating systems. Also, the responses of hardware to software commands is not a concern in the current Windows-based system, which does not justify the extra time and resources that would be required to switch to a different operating system. In addition, there has been a stated preference for a Windows-based system since many of the research labs doing similar work operate in a Windows environment.

Visual Basic and C++ are two alternative programming languages that would allow much tighter control over the code through manually implementing loops and data analysis techniques. Through this management, it would also be possible to reduce the amount of resources this program would use compared to a similar LabVIEW program.

LabVIEW allows for both the hardware and software interactions needed to produce the functionality needed by the client. The programming interface is more intuitive than that of C++ or Visual Basic, although it doesn't offer the same level of control. In addition, many functional tasks exist in LabVIEW prepared as functions known as virtual instruments (VIs). Each of these VIs can be incorporated into another program and are then called SubVIs. More complex analysis tasks can be performed with significantly less development time using LabVIEW because of the ability to use SubVIs.

C. The Design

This design builds off the second design option (Figure 7) and involves using LabVIEW in a Windows environment for the platform. It would encompass starting with a new program, recreating the basic functionality of receiving signals, displaying the graphs, averaging multiple tests together, and taking input from the user. Afterwards, the requested functionality would be added along with the implementation of the new user interface. The primary disadvantage of this design is that it would require a significant investment of time into the project to rewrite portions of a program that work well as-is.

The advantage of this design is that starting over provides for complete control over the code, resulting in a modular system that can be more easily imported onto other platforms. It is also possible that some commands could be simplified, making the program more efficient. These benefits do not directly

impact the user of the program, but will be noticed if future modification occurs. Modifications could include the changing of a DAQ or addition of another feature – though neither of these possibilities are likely in the near future.

D. Pros and Cons

Starting over from scratch allows the greatest flexibility in design because any feature can be incorporated. It would also allow for a large simplification of the program. As it is, there are dozens of SubVIs incorporated into the main one, and more SubVIs within each of them. While this can help to organize information, some of the files date back to 1998 and LabVIEW version 5.0. This causes a compatibility issue since these older files cannot be opened to edit in the client's version, 7.1, or the latest, 8.6. On the other hand, this redesign would involve a significant amount of work, including the reprogramming of the interface between the software and hardware. Redesigning the interface would not noticeably impact the performance of the final program, producing a poor result in comparison to the work invested.

11. Design Matrix

After establishing three distinct design options, different criteria were established and assigned weights according to their importance and relevance within the project (Figure 8). The criteria are: functionality, ease of use, complexity, aesthetic appeal, and modularity.

A. Definition of Criteria

- **Functionality:** How well does the final product address the needs expressed by the client? This includes the addition of valve control, light wheel control, and data storage.
- **Ease of Use:** How simple is the user interface? Is it intuitive and readable with a high level of contrast in both light and dark environments?

- **Development Time:** What is the level of difficulty involved with developing and implementing this solution? Is it an optimal use of available resources, providing the greatest return on investment?
- **Modularity:** This programming technique addresses the need to separate and group functional elements of the program together (Lesiecki). The resulting program is more easily modifiable and adaptable since modules require one change instead of individually addressing every location in the program where that module is used.

Criteria (weight)	Tweak	Overhaul	Start over
Functionality (30)	30	**30**	30
Ease of Use (15)	8	**15**	15
Development Time (15)	15	**12**	1
Modularity (10)	2	**4**	10
Total (70)	55	**61**	56

Figure 8: *The Design Matrix*

The weights used in the matrix are relative since categories such as ease of use and development time have a subjective component and cannot realistically obtain a perfect score. The 'Overhaul' and 'Start Over' design options are very similar, as the front end is identical. The difference is a better underlying program (back end), which also results in a much longer development time. For the time constraints of this course, a long development time is not feasible. For this reason, Design 2, the GUI overhaul has been chosen as the route to proceed.

12. Results

One of the main features added to the program is the ability to save the data that has been recorded. Saving allows further analysis in external programs and was accomplished by replacing a preexisting

save function stored in a SubVI. This component was developed concurrently with the main program because the changes within it do not have an impact on the main user interface. The data was originally saved in a proprietary LabVIEW format. The format was changed to a tab separated spreadsheet, writing two rows of header data with the names and values of each user adjustable setting (Figure 9). This is followed by two columns, time and voltage, with the information collected throughout the experiment. The time is calculated based on the flash duration and number of samples taken to obtain the value in milliseconds from start.

	A	B	C	D	E	F	G	H	I	J
1	Comment	This is our test of the save feature during the ECB								
2	# of resp	Flash Dura	SWGain Cl	HW Gain,	Prestim P	Sampling	Ticks betv	date string	time string	
3	0	3	1	1	3	3	240	12/4/2008	10:46.0	
4										
5	Time (ms)	Value (mV)								
6	0	0								
7	1	-0.07499								
8	2	-0.14993								
9	3	-0.22476								
10	4	-0.29944								
11	5	-0.37391								
12	6	-0.44811								
13	7	-0.522								
14	8	-0.59553								
15	9	-0.66864								
16	10	-0.74128								

***Figure 9**: Sample output file*

We also wrote a portion of LabVIEW code that controls the solution delivery process during testing. There are two controls included in the “Solution Delivery” tab on the front end: a valve on/off switch and a drop-down menu to select a valve number. The code behind these controls is designed to send an on/off signal to the DAQ, initiating or terminating power to the system of valves. After the system has been activated, an array of true/false signals is sent to the DAQ that will open the desired valve and close all other valves.

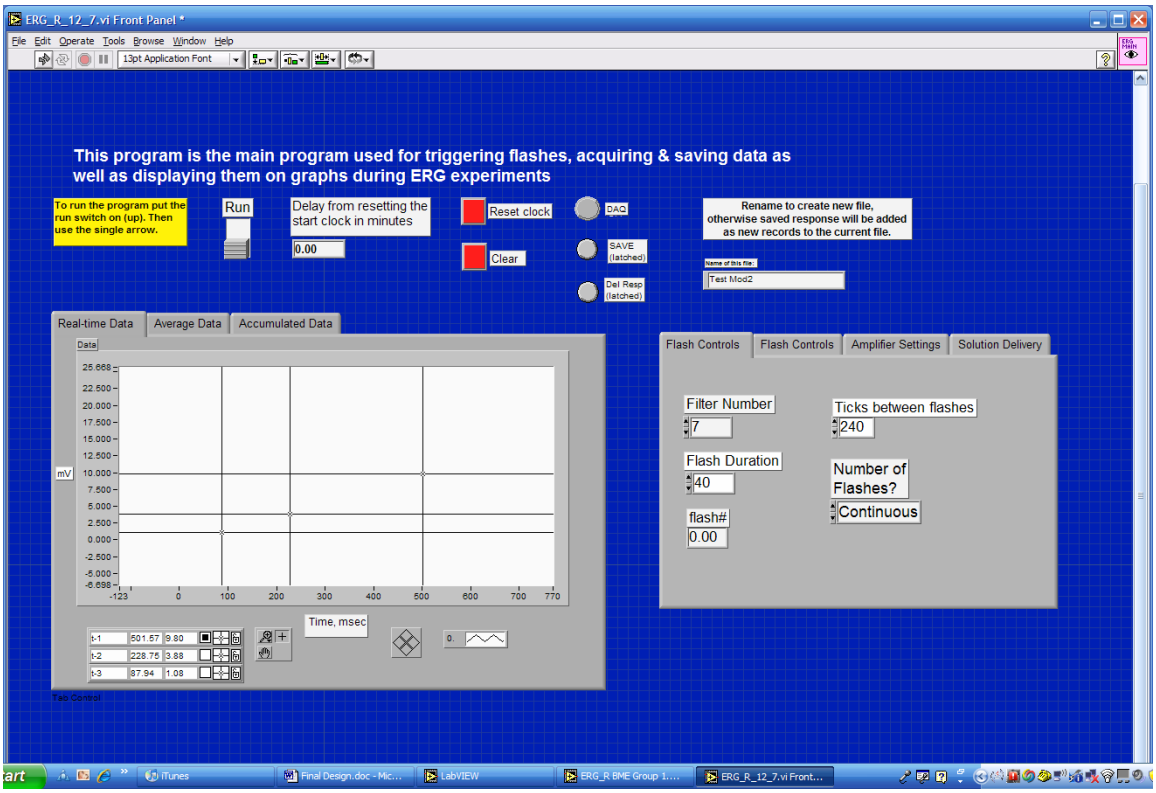

Figure 10: *Final GUI*

The main modifications of the GUI include the addition of cursors to the graphs and the organization of graphs and controls into tabbed windows. In the original interface, there was no way to check values from the waveform displayed on the graphs. In the final design, the user has cursors available that can be placed at any point on the graph; the corresponding x- and y-values are displayed directly below (Figure 10). These cursors were expressed as a feature desired by the client later on in the semester. This addition is useful to view the exact values of peaks and troughs of the waveform, and can allow the user distinguish the accuracy of the data.

Tabbed controls and graphs create a more organized and aesthetically pleasing interface. Putting the Real Time Data, Average Data, and Accumulated Data graphs into tabs increased free space enough to enlarge each graph. An increase in size allows the user to more easily view the waveform and to more accurately place cursors. By placing various controls in tabbed windows, we reduced clutter and

allowed for better navigation. Instead of having over 20 buttons and text boxes visible at once, we placed a number of controls into four categories: flash control, sampling period, amplifier settings, and solution delivery. The final design has a reduced number of controls visible at all times, making it less overwhelming to find specific buttons and inputs. Moreover, hiding certain controls in tabs prevents the user from modifying settings that should not be changed once the program is running.

13. Testing

Our client operates using LabVIEW 7.1, so it was necessary that the changes we made were functional in his lab environment. We upgraded to LabVIEW 8.6, the latest version at this time, but certain components of his existing program were not forwards compatible. As a result we tested our changes in both versions 7.1 and 8.6 to verify that they currently work and would support an upgrade.

To test the save feature, a wave similar to that of the ERG was simulated, along with a sample rate of 300Hz over 300ms. The data file, stored as plain text, was then opened in Microsoft Excel and the results stored within were checked against the simulated values to verify their accuracy. This component was designed in 7.1 and can operate independently of the main program. It also runs in version 8.6 without any further modifications.

The user interface was first redesigned in 8.6, but had to be converted to the client's version. We employed the use of functions and elements that were available in both versions 7.1 and 8.6 to reduce the chance of problems arising during a software update.

Due to the late arrival of the light wheel, which was ordered at the beginning of the semester, we were unable to fully integrate this component. However, there is a variable passed throughout the program

containing the filter wheel position that is connected to a SubVI that was designed for an older version of the light wheel. The data is transmitted using binary code through a parallel port, and as a result we were unable to test this output to verify that the value being sent was correct.

Another group worked on the development of a system for valve control with a different application purpose. Taking the specifications, a binary array of on / off commands for each valve, a menu was added to select a valve. This component was tested through the use of an LED in LabVIEW for each valve, and verification that only one LED would be lit up at a time, indicating that that would be the active valve.

14. Difficulties Encountered and Actions Taken

At the beginning of the semester, we expected the late arrival of some of the hardware to pose a problem during testing. What we did not expect was that the light wheel ordered the first week of the semester would arrive one week before the final product was delivered. We tried to see if the light wheel could be interfaced, but some drivers were lacking. The client later informed us that he had acquired the correct drivers, but as this was the day before the final presentation, there was not adequate time to integrate this feature. The light wheel DAQ takes a filter number and a binary code for setting the device. Our interface includes a setting to change the filter number and the specific binary code. It was impractical to debug these features because the information is in binary code, sent through a parallel port. Despite these difficulties, we feel that adding the light wheel to our interface should not pose a large problem.

We also anticipated additional functionalities conflicting with structures in the current code, requiring a backend overhaul. Instead, we encountered a problem of a different nature that could not be fixed

within the timeframe of this semester. Our client's interface is run in LabVIEW version 7.1, while the newest version is 8.6. Anticipating that our client would update to LabVIEW 8.6 early in the semester, we began developing additional features and working with the current interface in version 8.5, which is compatible with 8.6 and the latest available version through the university. We were unable to test the compatibility of the program as we developed features due its complex nature. Testing would have required additional hardware and software only available connected to the client's computer.

The system was upgraded to version 8.6 towards the end of the timeframe of this project, and the client's software ceased to function. It was determined that the libraries of functions in LabVIEW had changed, and National Instruments suggesting rewriting the entirety of the code due to the complexity of these issues. As a result, we converted the save feature to version 7.1. We also revised the user interface in 7.1 to account for this difficulty. We did not encounter any further problems making all interfaces compatible.

15. Future Work

With the save SubVI added to the program and code written for both solution valve control as well as light wheel filter selection, little work remains in the area of LabVIEW programming. Once the light wheel and solution valve hardware are completely interfaced, final tests can be performed on the system. Because testing has already been carried out on the individual parts of the project, it is unlikely that many bugs will be encountered in this process. If the client chooses to officially upgrade his version of LabVIEW, it would be necessary to update the libraries so that they can be fully accessed and referenced correctly. In addition, the code will need to be maintained for further modification of hardware or functionality.

16. Conclusion

The selected design addresses the requirements while improving ergonomics and the user experience over the previous version. Functionality has been added in the form of a working save feature, solution valve control within the main program window, light wheel filter selection, and cursors for the ERG wave graphs. A redesigned GUI has also been implemented which will improve ease of use for the client and will be helpful in making the system usable to others who may be less familiar with it. Overall, the redesigned ERG recorder program successfully provides the capabilities needed to record and analyze the electroretinogram wave. This will be beneficial to the client's research, potentially leading to new solutions for debilitating eye disease.

17. References

AD Instruments. "ADInstruments Products". 2008. <http://www.adinstruments.com/products/research>

Audie Technology. 2008. "Flow Pro Automatic Valve Opener". <http://www.audiotech.com/flow-pro/valve-opener-flow-pro.html>

Creel, D. 2008. "Clinical Electrophysiology". John Moran Eye Center, University of Utah. <http://webvision.med.utah.edu/ClinicalERG.html>

Green and Kapousta-Bruneau. 2006. "Evidence that L-APS and D,L-APH can preferentially block cone signals in the rat retina." *Visual Neuroscience*. 24(1):9-15.

Electro-Diagnostic Imaging, Inc. 2008. "VERIS™ Science Software." <http://www.veris-edi.com>

Hobart, J. "Principles of good GUI Design." < http://www.iie.org.mx/Monitor/v01n03/ar_ihc2.htm>

Kazaryan AA. 2007. "Functional characteristics of the retina in Glaucoma." *Acta Opthalmologica Scandinavia* 85(240).

Lesiecki, N. 01 Jan 2002. "Improve modularity with aspect-oriented programming." IBM. http://www.ibm.com/developerworks/java/library/j-aspectj

Matsui, Y *et al*. 1994. Electroretinogram b/a wave ratio improvement in central retinal vein obstruction. *British Journal of Ophthamology*. 78(3):191-8.

Medline Plus. 2008. "Electroretinography." *US National Library of Medicine*. <http://www.nlm.nih.gov/medlineplus/ency/article/003388.htm>

Morton, JL. 2008. "Color Theory." < http://www.colormatters.com/colortheory.html>

Ye JH, Goo YS. 2007. "The slow wave component of retinal activity in rd/rd mice recorded with a multi-electrode array." *Physiol Meas* 28:1079-1088.

18. Appendix A: Product Design Specifications

Team Members: **Dan Jonovic**
Andrew Dias
Nick Balge
Whitney Johnson

Client: **Dr. Bikash Pattnaik**
Advisor: **Prof. Mitch Tyler**

Function: Electroretinogram (ERG) is the measurement of voltage across the retina in response to light flash. This project's goal is to create a system controlled through LabVIEW to monitor, control, and record values from an ERG corresponding to liquid exchange, light intensity, flash duration and frequency, as well as the waveforms.

Client requirements:

- Filter number, corresponds to the light intensity
- Flash duration (in ms)
- Sample duration (in ms)
- Liquid delivery through valve control of up to 8 different solutions
- Cursors in graphs for viewing time and voltage
- Export data to Excel for further analysis
- Animal testing may be performed
- Recording of a-waves, b-waves, and slow-waves

Design requirements:

1. Physical and Operational Characteristics

a. Performance requirements: Used daily on a Windows platform. It will likely be used by itself so RAM and processing power is not an issue. Delivery of one, and only one, liquid into the testing environment at a time.

c. Accuracy and Reliability: The interface is as accurate and precise as the selected programming environment is on the chosen operating system.

d. Life in Service: The software program should run without modification until the data acquisition hardware is changed.

e: Operating environment: The system should be functional in a Windows environment.

f: Ergonomics: The user interface on the computer should be intuitive: Items displayed should be easily understood with consistent formatting and a clean interface.

g. *Materials*: Windows-based computer with LabVIEW 7.6 installed.

h. *Aesthetics*, *Appearance, and Finish*: The program should be easy to use and functional with the following considerations:

- Text easily readable
- Contrasting Colors
- Intuitive Organization
- As simple as possible without dropping features

2. Production Characteristics

a. *Quantity*: 1

b. *Target Product Cost*: Software and hardware have already been purchased so no additional costs should be incurred.

3. Miscellaneous

a. *Standards and Specifications*: This project will not need to conform to FDA standards, as human testing is not going to be performed. It is necessary, however, to follow the RARC (Research Animal Resources Center) for animal testing.

b. *Customer*: The client would like to use a system in a Windows environment.

c. *Competition*: Our system is intended for research and there are no patents covering the scope of this project. There are some software solutions that have similar capabilities:

- CSH Protocols has developed a protocol for progressive ERG measurement during the development of larval zebrafish noninvasively. <http://cshprotocols.cshlp.org/cgi/content/abstract/2008/4/pdb.prot4973>
- VERIS EDI software is Macintosh based and has protocols for mfERG and transient ERG recording. Their software provides data analysis and storage capabilities but does not interface with the additional components (valves, etc.) that the client desires. <http://www.veris-edi.com/>
- AD Instruments LabChart Software is a data acquisition solution that, when combinied with LabScope, offers powerful data analysis techniques. This software does not provide the hardware interface to communicate with valves or record other settings such as the light wheel. <http://www.adinstruments.com/products/software/research/LabChart-Software>