Design and Construction of a Quad Rat Vitals Monitor

May 4, 2011

Team Members:

Matthew Bollom – Team Leader

John Renfrew – Communicator

Kenneth O. Xu – BSAC

Gabriel Bautista – BWIG

Client:

Alex Converse, Ph.D.

Advisor:

Amit Nimukar, Ph.D.

Abstract

The Quad Rat Vitals Monitor (QRVM) is needed for research to simultaneously monitor the vitals (breathing rate, heart rate, internal temperature, and blood oxygen saturation) of small animals such as rats. The team's client currently runs positron emission tomography (PET) scans on four rats concurrently, and the scans can last up to two hours. During the two hour scans, the rats are under anesthesia and the medication dose must be adjusted based on the rats' vitals. The client desires to have an accurate, reliable, and easy-to-use rat vital monitoring device to aid in this process. The current design for this monitoring device includes force-sensing resistors (FSR) for monitoring breathing rate, thermistors to monitor rectal temperatures, and pulse oximeters to monitor SpO_2 levels and heart rates. The design also includes an easy-to-read graphical user interface (GUI) that displays traces of the vitals as well as the current value of those vitals in the form of heart rate, blood oxygen saturation, temperature, and breathing rate.

Table of Contents

Introduction

Background

The design team's client currently runs positron emission tomography (PET) scans on rats to monitor the location of positron-emitting radionuclides (radiotracers) within the rats' brains. These scans can last up to two hours, and during the scans the rats are under anesthesia as shown in Figure 1. The client and his assistant must monitor the vitals of the rats during these scans to ensure that they endure no harm while under anesthesia. Currently, the client monitors the rats qualitatively. The skin color of the rats is observed and recorded to ensure that the rats are receiving enough oxygen. The rats' breathing rates are monitored simply by observation, and body temperature is monitored by touch and inexpensive thermometers from Walgreens. Currently, heart rate and blood oxygen saturation are not reliably monitored. The client would like to be able to obtain quantitative measurements of multiple vitals of each rat during the PET scans.

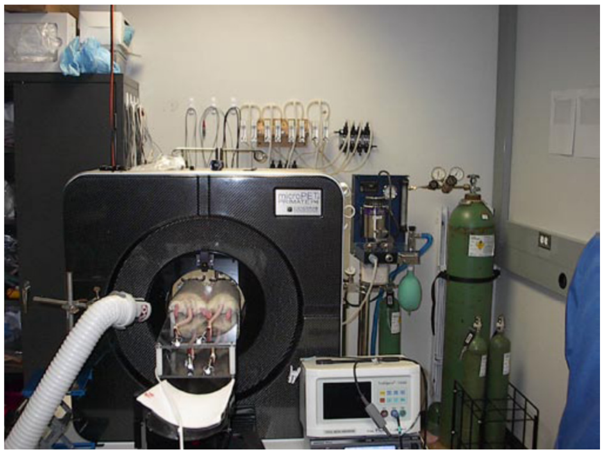


Figure 1: This picture shows four rats in the PET scanner at the client's laboratory. The rats are oriented in a two by two square [Ho, J. et al.].

Since the PET scanner is designed for monkeys instead of rats, it is large enough to simultaneously scan four rats at a time. The client always scans four rats at a time to lower costs. The radiotracers used in the rats are expensive to produce and decay relatively quickly, so producing a single set of radiotracers for four rats is more cost effective than producing four separate sets. This process requires the vital monitoring device to monitor four rats simultaneously.

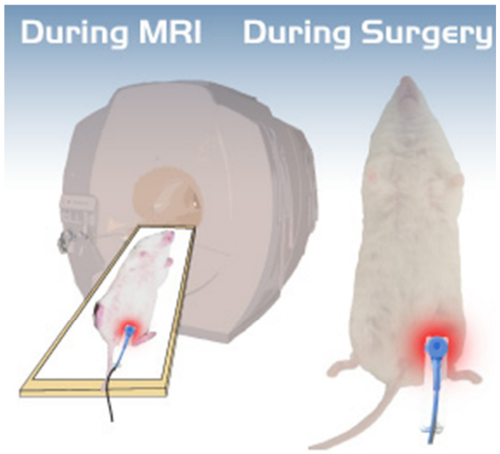


Figure 2: Pulse oximeter produced by Starr Life Sciences capable of measuring heart rate, breathing rate, and SpO_2 levels [Starr Life Sciences].

Existing Devices

There is currently one device on the market that is capable of measuring the desired vitals of rats and mice (not including temperature). This pulse oximeter designed specifically for rats and mice, called MouseOx, is produced by Starr Life Sciences as shown in Figure 2. This device is not capable of monitoring four rats simultaneously, and is priced at $7000 for one device [Starr Life Sciences]. This is not what the client is looking for because of the high cost and the inability of the device to monitor multiple rats simultaneously. A pulse oximeter, produced by Nellcor (the Nellcor N-100), was tested for blood oxygen saturation (SpO_2) level accuracy when attached to a rat's tail. When SpO_2 levels were between 75% and 95%, this pulse oximeter was capable of measuring SpO_2 levels relatively accurately. When compared to the blood sample analysis, the N-100 measured SpO_2 levels with a standard deviation of ±5.7% [Decker, MJ et al.]. This system does not measure breathing rate or temperature and is no longer produced.

The client occasionally uses a pulse oximeter designed for small animals, specifically monkeys and dogs, but is unable to consistently obtain accurate data because of the high heart rate and low blood volume of rats. The heart rates of these rats can rise above 300 beats per minute, and this pulse oximeter was not designed with a high enough sampling frequency to measure these pulses or the corresponding oxygen saturation. Therefore, the monitoring device must be able to measure vitals outside the normal ranges of small animals or humans.

Client Requirements

The vitals monitor must be able to simultaneously monitor SpO_2 levels, heart rates, respiratory rates, and rectal temperatures of four rats simultaneously. SpO_2 levels must be monitored with an accuracy of ±2%. Heart rates of up to 500 beats per minute and respiratory rates of up to 30 breaths per minute must also be monitored. Rectal temperatures of 33 to 38 degrees Celsius (93 – 100 degrees Fahrenheit) are to be monitored as well. All four vitals of all four rats must be simultaneously displayed in a user-friendly graphical user interface (GUI) on one screen. All probes used to monitor rat vitals must be non-invasive and cause no harm to the rats. Finally, no component of the device can pass the cranial end of the sternum, because the PET scans are focused around the cranial region of the rats.

Motivation

Throughout the duration of the client's experiments, the rats are under heavy doses of anesthesia, which require manual adjustments by the laboratory assistants. The development of a system that readily displays the current values of each vital sign along with the option to view the history of each vital would be extremely beneficial. The laboratory assistants must be informed if any of the vital signs enter critical ranges; this will ensure timely adjustment of the anesthetic.

Currently there is no accurate, cost-effective method to monitor and display all four vital signs simultaneously. Since our client's research is based off of a four-rat setup, the existing devices previously mentioned will not suffice. It would be impractical for him to purchase four of the existing devices that are priced at or above \$7000. The total budget of \$4000 should therefore be considered throughout the design process and while planning to manufacture the final prototype. Furthermore, the GUI displaying the four vital traces must be designed according to the type of data it will be receiving from the circuit elements and the corresponding probes designed to monitor each vital sign. The design team's goal is to produce an inexpensive, easily operable prototype that incorporates electrical monitoring systems along with a Java GUI to display the signals received from each monitoring system.

Previous Work

A few of the major benchmarks of this project are presented for perspective and understanding.

Fall 2008 & spring 2009

The project was started in fall 2008 by a different BME design team. Much of the first two semesters were spent defining the project and understanding the various issues that would be encountered. The circuits for measuring breathing rate and temperature were designed.

Breathing rate was measured using a FSR in conjunction with a voltage divider. The rat was placed on the FSR, and as it breathed a greater force would be applied to the FSR. This greater force translated into a larger resistance change and thus a voltage drop across the voltage divider. A peak detector was then used to determine the locations of the rat's breaths; the breathing rate could be calculated using this data.

Temperature was measured using a modified Walgreens thermistor and a non-inverting amplifier. In the range desired by the client, the thermistor was then shown to have a linear relationship between voltage and temperature with a high correlation coefficient (R^2=0.9983) (Ho, J. et al).

A LabVIEW interface was also created. Although this interface did not accurately represent the client's desires for the device interface, it served as a starting point for future semesters.

Fall 2009

Much of the focus for fall 2009 was designing an interface that met the client's needs and further refining the FSR and thermistor circuits. A graduate student was working on a pulse oximeter; the team integrated the FSR and thermistor circuits into the oximeter. The force sensing resistor and thermistor were extensively tested. After circuit modifications, these were shown to work effectively and meet the client's needs.

A LabVIEW interface was created as shown in Figure 3. It contains two live traces – heart rate and breathing rate – and four history traces of each vital. The history traces showed the overall trend of the vital signs over the course of the entire experiment or over a user-specified range. Data archival was also implemented in this program. Every 15 seconds, the average value of the vital over those 15 seconds would be saved to hard disk for later analysis by the client.

The new piece of the project requested by the client this semester was to provide a way to note events during the course of the experiment. A system using numbers was implemented, but it was not very user friendly because the user would have to manually note on a piece of paper what each number meant. The entire interface was integrated with the FSR and thermistor circuits, and it was extensively tested.

A custom pulse oximeter probe was designed, but it was not successful at eliminating noise due to lack of necessary shielding. Because of this, the team recommended a redesign of the probe.

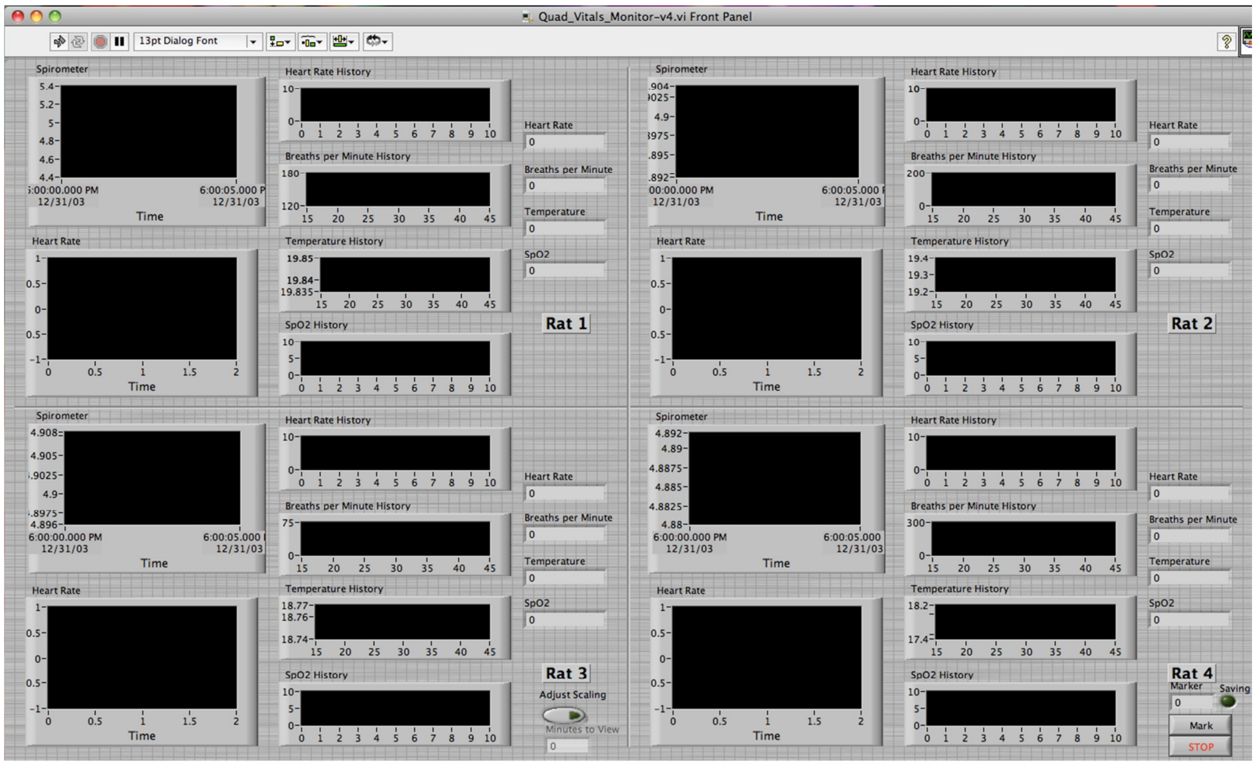


Figure 3: LabVIEW interface from Fall 2009

Spring 2010

Hardware for the project was the main focus for spring 2010. The graduate student working on the pulse oximeter had completed it early in the semester, so the team conducted some human testing to ensure the circuits and first-generation firmware was working correctly. Upon completion of subsequent rat testing, a printed circuit board was ordered. A Bill of Materials (BOM) was created, parts ordered, and one circuit board built and debugged.

The pulse oximeter probe was redesigned a second time. However, results were similar to the semester before and the team recommended commercial alternatives to be considered. Without a commercial probe, the quality of the pulse oximeter signal could not be guaranteed.

Minimal software work was conducted, but it was evident that an alternative to LabVIEW must be considered. Since the sampling rate of the rat pulse oximeter was higher than the human counterpart, LabVIEW would start having issues keeping up with the large amount of data and the concurrency needed in order to run a program of this magnitude. Several things would need to happen at the same time in this program – data acquisition, signal processing, interface updating, and data archival. LabVIEW is powerful for small applications, but the team believed it would not work well for this project.

Fall 2010

Because much of the hardware was completed, the focus for fall 2010 shifted back to software. Based on the recommendation of the previous design team, the team researched alternatives to LabVIEW. Java was chosen because it was open source, free, many of the team members had experience with Java, and it was easy to learn. Using a variety of libraries, a basic interface was created (Figure 4). Although this interface was not what the client desired, it provided the team with experience with creating interfaces, and allowed them to easily debug output from the hardware as each trace was the voltage value.

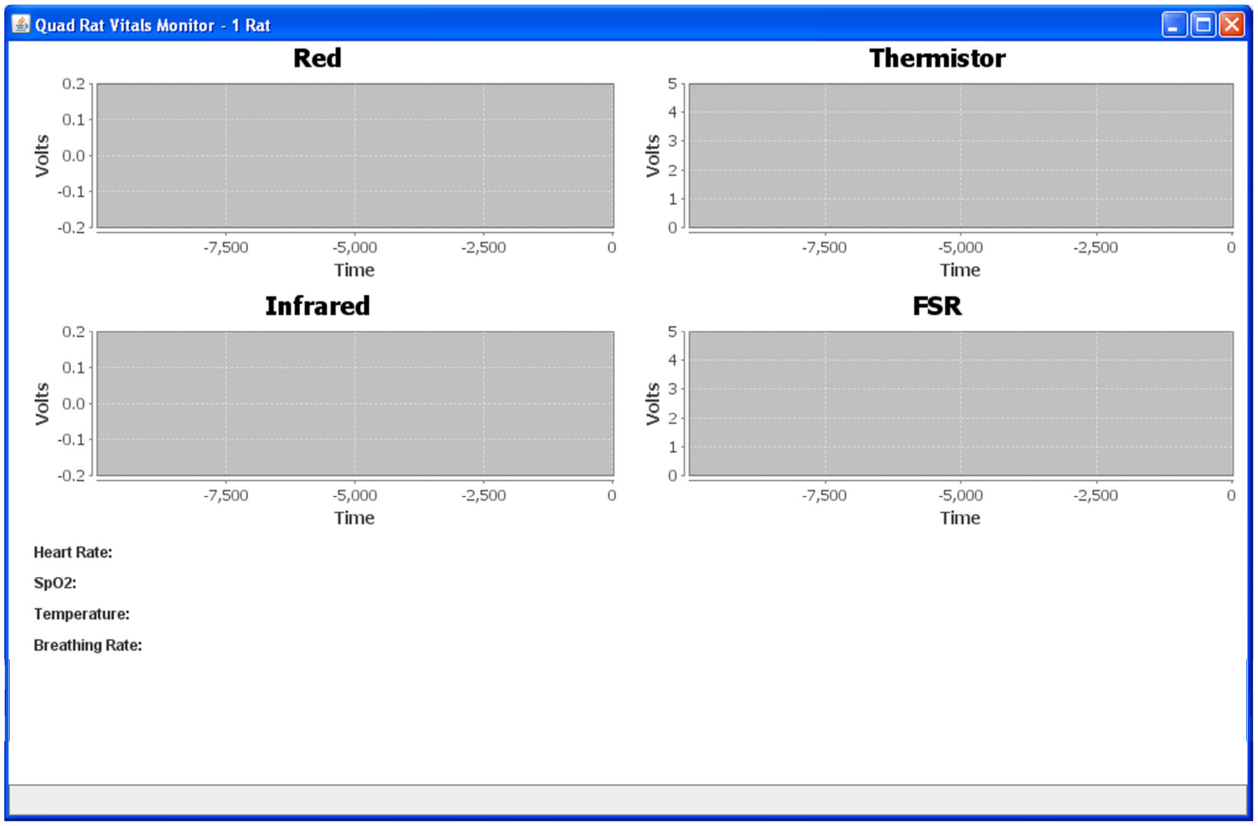


Figure 4: Fall 2010 JAVA-based GUI

Besides porting an interface to Java, signal processing was investigated. Earlier teams had signal-processing completed for the FSR and thermistor, but those algorithms were in LabVIEW and would need to be ported to a textual form. The thermistor was completed easily given the linear relation between voltage and temperature. Peak detection for the FSR was more difficult as it needed to occur real-time. Based on this observation, the team reconsidered the program design. The program was redesigned such that it had concurrency support – the program could now acquire data, process it, update the interface, and respond to user input at the same time without affecting performance of other parts of the program.

After the program redesign, the team investigated the pulse oximetry algorithms. By the end of the semester, algorithms were implemented. Although a more accurate calibration would be needed, the algorithms were working.

While software development was occurring, the team also built the three additional boards, but due to time constraints, the boards were not programmed or debugged.

Spring 2011

This semester produced a number of improvements to the graphical user interface. Additionally, the circuit boards which had been manufactured in fall 2010 have been programmed and debugged. A housing for the circuitry components has also been manufactured.

Software

Graphs

The previous interface implemented traces largely used for diagnostic purposes as shown in Figure 5A.These traces primarily displayed voltages and peaks. While these voltage values are useful to verify the circuit functionality, the values themselves are not clinically useful. The client has specified a user interface that would provide the most utility. The completed interface will suit the majority of end-users of this product. It will contain the traces in the previous LabVIEW implementation shown in Figure 5B, but it will use open-source Java programming language.

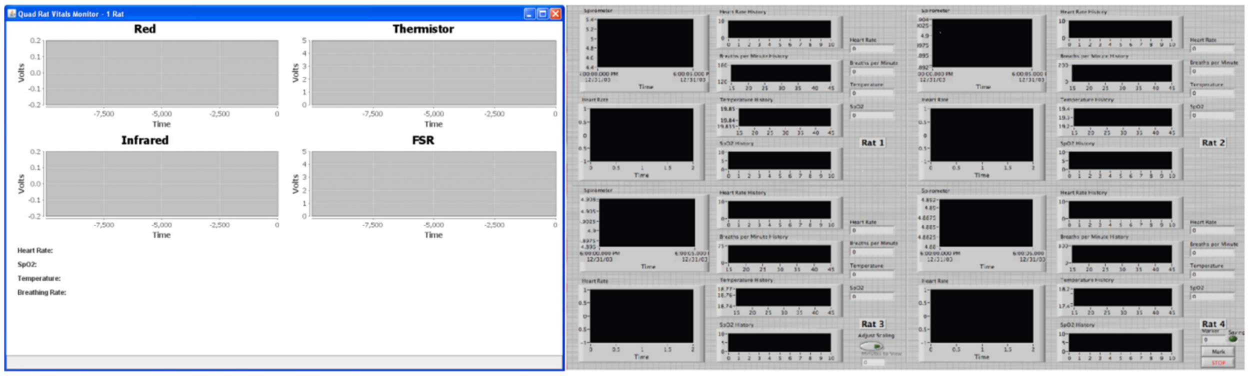


Figure 5 (A): The fall 2010 version of the user interface. No clinical graphs are present. (B): The fall 2009 implementation in LabVIEW. All the traces are present, but due to monetary and performance issues, these will be transferred to Java.

The final interface, shown in Figure 6, will have two “live traces” on the interface which display the signals from the FSR and red LED. These two graphs cover roughly 10 seconds on the domain. The live traces will be updated every data point (about 360 Hz). The interface will also include four “history traces”. These include graphs for temperature, heart rate, breathing rate, and $Sp0_2$. This will cover roughly 10 minutes on the domain, or some other user-specified value. The update interval will be every 2 seconds. These graphs will elucidate any long-term trends in the rat’s vitals. For monitoring purposes, an alarm system has been implemented in the user interface that will allow the user to set thresholds for each of the history traces by opening the ‘Settings’ menu. Should the rat’s vitals drift away from these thresholds, the corresponding graph set will turn red and the computer will sound an alarm to alert the user. The interface is set up to sound an alarm every time a value is plotted outside the thresholds, however, the graph set will remain red until the user manually presses the ‘Reset Window’ button. This way even if the rat’s vitals return to normal values, the user will know that an event outside the set threshold has occurred. Additionally, four textual displays will show the live value of breathing rate, heart rate, temperature, and $Sp0_2$.

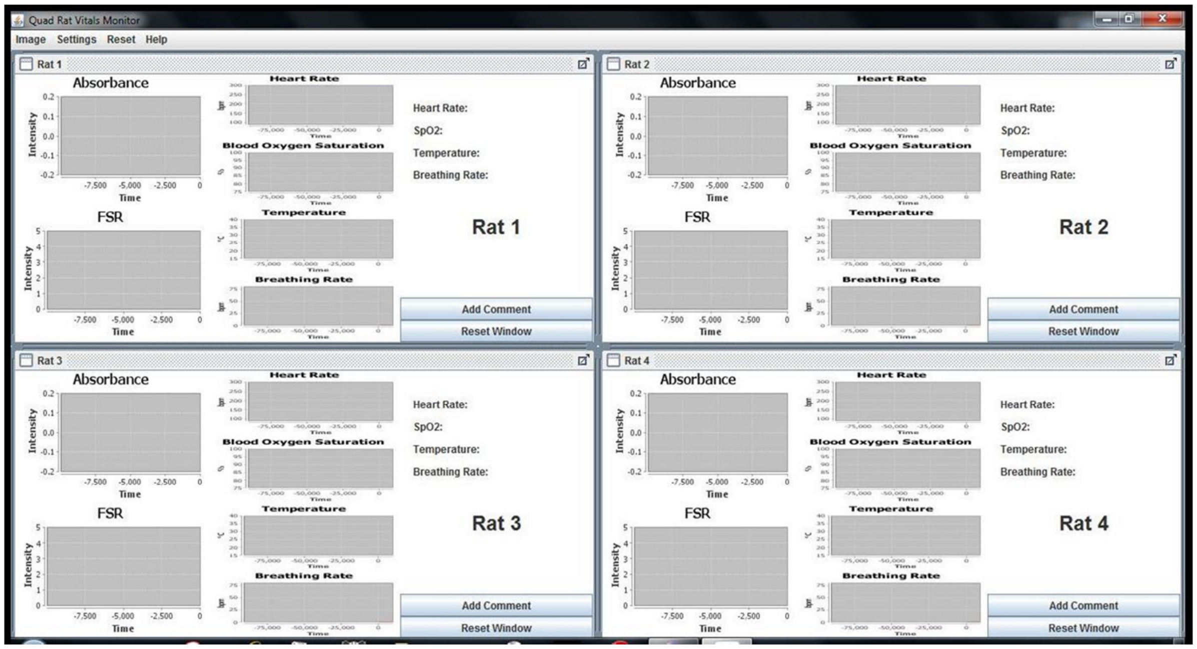
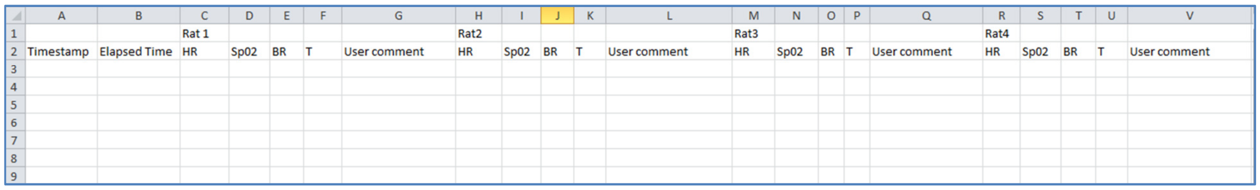


Figure 6: The final graphical user interface written in Java

Data Archival

The client has also expressed interest in saving the vitals data. This would be a very powerful tool for the device and increase its commercial viability. The location of the saved file will be prompted on program start and will not be able to change (refer to section 8 of Appendix 4) post execute. The items to be saved *per rat* include: time elapsed, heart rate, $Sp0_2$, breathing rate, temperature, and user comments. Saving each of these values would be unrealistic (~360 data points/second), so an average will be documented about every 15 seconds, depending on the computer’s processing speed. A comma-separated values (CSV) file format will be used, allowing for easy data parsing in Microsoft Excel shown in Figure 7 as per the client’s request.



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
1			Rat 1					Rat2					Rat3					Rat4				
2	Timestamp	Elapsed Time	HR	Sp02	BR	T	User comment	HR	Sp02	BR	T	User comment	HR	Sp02	BR	T	User comment	HR	Sp02	BR	T	User comment
3																						
4																						
5																						
6																						
7																						
8																						
9																						

Figure 7: The header format for the saved file created during program execution. The data for all 4 rats will be available in the same file. Auto-formatting with Excel will help the user identify specific areas of interest.

The interface will also include an “Add Comment” button which will allow the user to alter the “user comment” section of the saved file. The comment can be made global (for all rats) or local (for a specific rat). This will provide a quick way for the user to mark an important time (e.g. administering anesthetic) for later correlational analysis.

Hardware

In conjunction with the software, the team also continued to work on the hardware. Figure 8 shows the evolution of the circuits throughout the semester. The breadboard on the left contains the voltage dividers for the FSR and non-inverting amplifiers for the thermistor. It also contains a simple transimpedance amplifier to test the output from a custom-designed pulse oximeter probe.

The middle board is the prototype pulse oximeter designed by a graduate student. Adapting a human pulse oximeter would not work because of sampling rate limitations of those oximeters. Rats' heart rates can get up to 500 bpm, hence a much higher sampling rate is needed than for human oximeters. After the device was built, it underwent some human testing to ensure the circuits and first-generation firmware were functioning correctly. Finally, some preliminary rat testing was conducted.

The board on the right shows the completed printed circuit board. This board contains the circuits of the previous two boards in a smaller form. The parts for this board are relatively inexpensive (about $90.00), but it does not include any of the probes or necessary cables.

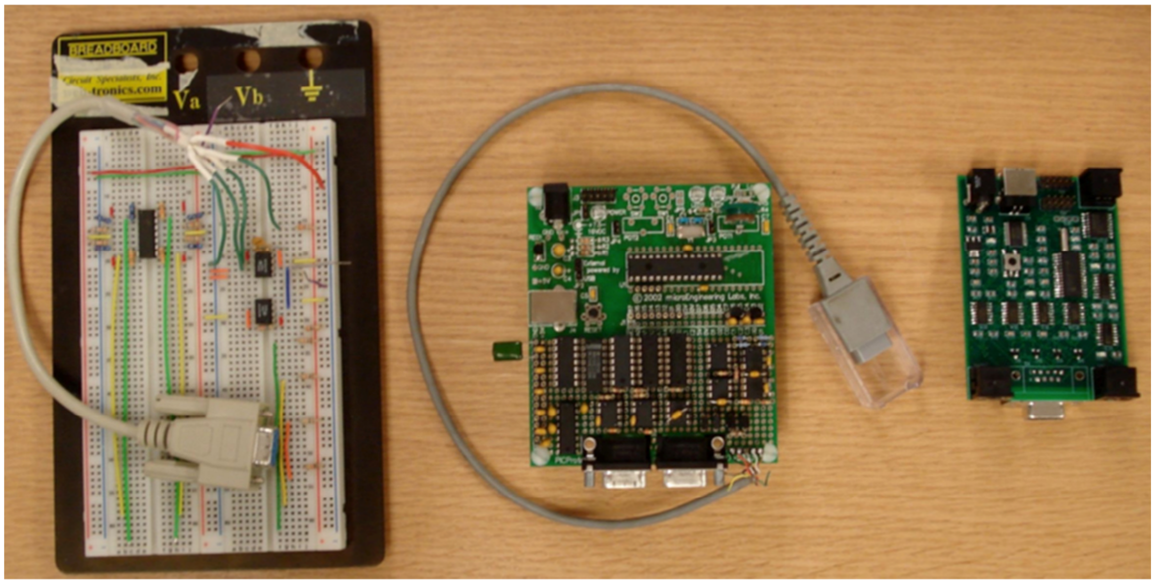


Figure 8: Circuits evolution throughout the project

Individual Boards

Prior to this semester, 4 boards were built. Only one was completely programmed and functioning as none of the prior teams had time to finish the remaining boards. This semester, the boards have been programmed and debugged. They are mostly functioning; some resistors need to be changed in order to guarantee a quality signal.

The firmware for these boards was essentially completed. An autogain algorithm was designed, which allows the pulse oximeter to change the red and infrared LED intensities to account for the different concentrations of hemoglobin in the blood and in the thickness of paws. Without this adjustment, the brightness of the LEDs for one rat might cause photodiode saturation on another rat, and the resulting signal would be incorrect.

The autogain algorithm was designed such that it calculated which LED had a greater effect on the rat's paw, and then adjusts the less strong one upwards until it is within 1% of the other. Because of the blood pulse and its effect on absorbance, the algorithm continuously samples to ensure that one of the signals will not overtake the other. When that is complete, the gain is then adjusted until it reaches a set point, which was empirically determined.

With human testing, the algorithm worked over 75% of the time when the initial current levels were at 20%. On a rat, it worked essentially all the time when initial current levels were at 50%. The rats used for this testing are comparable to those used during the client's research, so the team is confident that initial current levels of 50% should be satisfactory. To ensure this, the team recommends a modification to the algorithm – an initial current level change if something were to fail.

This initial current level change would be easy to implement. The framework is already in place. If the algorithm reaches ones of the bounds for the current levels or gain, the algorithm should be reset with initial values in the direction that would help the algorithm finish. For example, if the gain could no longer be increased to bring the signal levels to the designated setpoint, the initial current levels should be increased and the algorithm repeated. If, after several iterations of initial current level modifications, the algorithm fails to successfully perform autogain, a default setpoint should be set so the user is not prevented from acquiring data. Currently, the board will not output data if autogain fails.

In conjunction with autogain, an I^2C slave was implemented. The I^2C protocol and implementation will be discussed in more detail later. In order to effectively implement I^2C, the data transmission time would need to be significantly shortened. Previous iterations of the firmware output data in ASCII characters, which allowed for a human to understand and manually manipulate the data if necessary. However, at 115.2 kbaud (serial transmission speed), the transmission time for one data point was 840 μsec, which would prevent the I^2C bus from collecting data fast enough, thus slowing the system down. Coupled with the fact that a more than double sampling rate change would be needed, an alternative transmission protocol was very clearly needed.

The team designed a custom protocol called "3 byte transmission". This protocol encoded 3 pieces of information in 3 bytes: the rat number, the signal, and the data value. With some simple calculations, the Java end of the system could calculate the correct values and further handle movement of data more efficiently. By encoding the rat number on the board, the system controller would have less work to do and would simply gather and transmit the data, saving processing time for other matters. With 3 byte transmission coupled with the increased sampling rate of 1440 Hz and baud rate of 230.4 kbaud, the transmission time dropped to 220 µsec, a 60% reduction in time versus the ASCII equivalent. At the lower sampling rate and 115.2 kbaud, a 54% reduction in transmission time was noted. These transmission times also included a delimiter character for ease of measuring, and the final implementation will not include this delimiting character.

System Controller

Because this device will increase the amount of cables and wires in the client's laboratory, the client has requested that the device have as few cables as required. To transmit the data from boards to the user software, a USB cable for each board could be used. However, this would provide extra unnecessary cables. To help prevent this, a system controller was designed.

The system controller will consist of a power distribution system, a master I^{2}C device, data output, and status LEDs. The power distribution system will allow the user to connect one power supply to the system. The system controller would then distribute power to each board. The device would obtain the data from each board individually via I^{2}C, and then output this data.

The team used an mbed microcontroller to serve as the master microcontroller. The mbed is a departure from traditional microcontrollers as it provides an application programming interface (API)-oriented library. The mbed is relatively new (about 1.5 years), but a growing community of users supports it. It is not designed for commercial applications; rather it is designed for rapid prototyping. It supports a multitude of features, including I^{2}C, USB, and Ethernet.

Ethernet was used to output the data. The Ethernet connection is just the physical connection. TCP (transmission control protocol) was used to transmit the data on the line. TCP was chosen because it ensures that data will not be lost during transmission. If the correct data is not received at the other end, the host will resend the data. Ethernet also offers speed capabilities that will not be fully utilized because it is such high speed (i.e. 100 Mbit per second). Data collisions are not a concern with this speed.

Because TCP is network based, it allows a range of possibilities to the client for remote monitoring. Along with the local monitoring in the research laboratory, the client could monitor the rats from his office. While the team does not anticipate that being used, it is a possibility and would require minimal work to enable. Additionally, TCP allows the device to be used on a wider range of operating systems. Previous iterations were limited to Windows as the libraries were only functional on Windows. TCP allows for usage in a platform independent manner, and when coupled with the platform independent software, it gives researchers an option to utilize computers they are familiar with.

While most of the features of the system controller were functional this semester, two features were not implemented: the status LEDs and the hardware resets. The intention with the status LEDs was to light them when an alarm began on the computer. This LED would allow the researcher to see the status of a rat as they were manipulating any probes without having to walk to the computer to glance at the screen. Additionally, these LEDs could be used to show the power status of each board.

The hardware resets were a concept developed late in the semester. Its purpose was two-fold: 1) to force autogain to occur if necessary and 2) to reset the boards if they were not working as expected. To save processing time on the boards, it was decided that autogain would not be continuously running. Rather, it would run upon board reset. Since it was possible to force the hardware to reset via software, a method was designed in the user software to cause the board to reset. This would be especially useful for when the research assistant moved the pulse oximeter probe, and would like to ensure the highest quality signal.

A schematic of the system controller can be found in the Appendix on page 24.

System Communication

One issue that has been facing teams working on this project has been speed. Because of the high sampling rate needed, data output capabilities need to meet at least the sampling speed (~1440 Hz), if not faster. Failure to do so will result in data collisions. The multitude of data associated with four boards augments this concern.

To help simplify development of a communication system, the team chose to use I^2C (Inter-Integrated Chip). The I^2C was developed by Phillips Semiconductor (now NXP) in the 1990s. It is a two-wire interface – one clock line and one data line. A master device controls the clock line and dictates the speed of the communication. The data line allows the data to move between the master and any number of slaves. Multiple masters are permitted, but doing so presents multiple complexities that must be considered. Because only one master is needed for this device, this setup was not extensively researched.

The number of slaves on an I^2C bus is limited by some electrical characteristics, most notably the system capacitance of about 400 pF, and the number of available addresses. Each device on an I^2C bus is given an address and allows a master to communicate directly with it. With seven bit addresses, a maximum of 112 devices can be on an I^2C bus. 16 addresses are reserved, thus the maximum number of devices is less than one would expect.

The individual rat boards have I^2C capabilities. Other protocols such as Serial Peripheral Interface (SPI) were considered, but could not be used because the hardware is not SPI capable. SPI would have required additional hardware components and thus be more difficult to implement. I^2C simply requires 2 lines with some pull-up resistors. Software fulfills the rest of the particulars of integration.

The microcontrollers used for the individual boards permits slave operation. Figure 9 shows the procedure to implement for the PIC18F2420, the microcontrollers in use on the individual rat boards. It requires reading, clearing, and writing bits and flags at the appropriate timing as demonstrated in Figure 9.

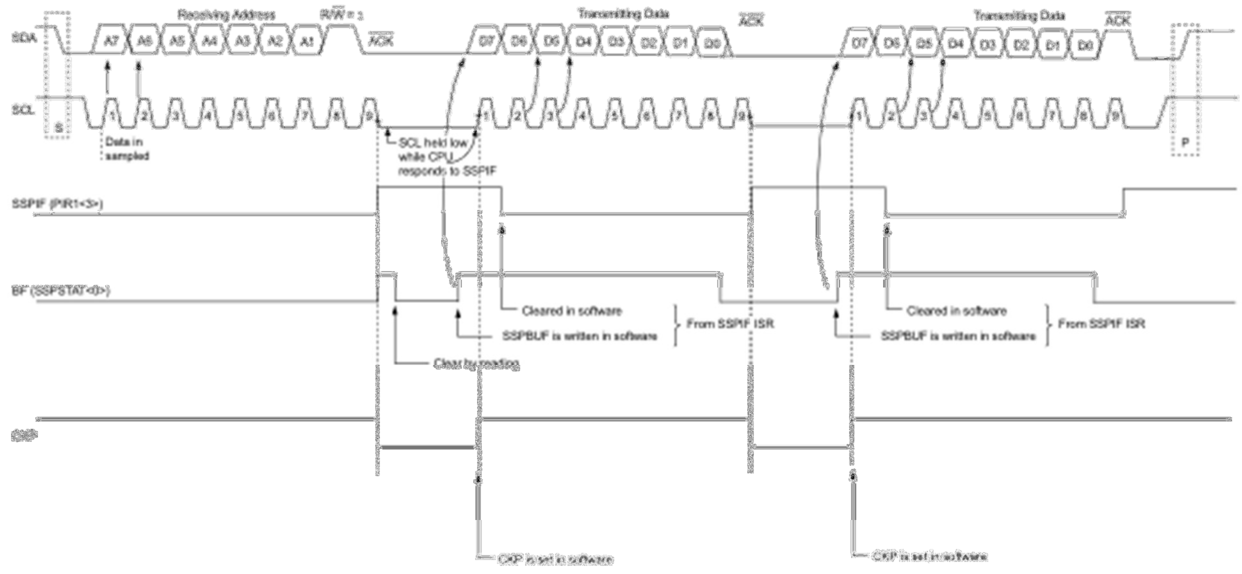


Figure 9: Implementation of I^{2}C slave in PIC18F2420, the microcontrollers in use on the individual rat boards (Microchip, p. 177)

Each of the boards has been programmed with a unique address. The master sequentially polls each of the boards for one data point, packages this into a TCP packet, and sends it to the client software if the device is paired. Initially, the delimiter value was included, but it was eventually deemed not essential as the TCP packet transmission handled delimiting by its self. The team was able to successfully operate two boards in parallel. Four was briefly attempted, but due to some timing issues, it was temporarily abandoned. A potential solution the team is considering is using the second I^{2}C pin set on the mbed for the other two boards, and then operating the two sets in parallel.

System Overview

Figure 10 shows the arrangement of data and power lines in the system. Status LEDs are not included for simplicity, but can be found in the system controller schematic in the Appendix.

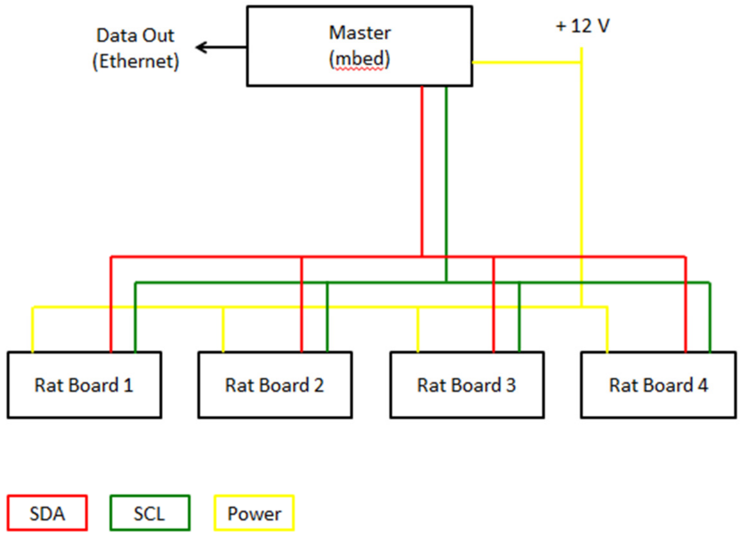


Figure 10: System overview

Housing

A housing device was made to protect the boards from the outside environment and provide a compact, safe method of transportation for the four boards and the system controller. The casing was made of a ridged copolymer plastic, acrylonitrile butadiene styrene (ABS). ABS has properties of being inexpensive, impact resistant and tough which made it an ideal candidate for our device to ensure that is protected against accidental drops as well as mass production. The case measured 8.00” x 6.00” x 3.00”. The previous ABS housing that was initially designed measured 5.85’’×6.00’’×3.00’’ and was deemed inadequate to house all the necessary components and receptacles.

Initially, the boards were originally placed in a 2 by 2 configuration with 1” spacing between the boards for wires that connected the boards to each other. The boards were to be held together by “T” beams made from the clear plastic the client provided, but this design was later proved the be flawed as it did not provide adequate spacing for the receptacles.

The final design involved the black 8.00” x 6.00” x 3.00” case (obtained from Radio Shack) and contained an aluminum L-shaped piece that held the boards together. The side that contained the main receptacles was the larger 8.00” x 3.00” side. Since each of the four boards required its own set of receptacles, the space was divided into four evenly spaced quadrants. There were screw holes on the corners of the box that ran down the sizes; this reduced some of the useable space down to roughly 37 mm by 84 mm for each quadrant. Four receptacles were used for each board. They were two mini-din receptacles (three pin and four pin), one d-subminiature connector and one LED hole. The d-sub was 11 mm tall and all other components were 12 mm tall. These four components for each quadrant were arranged in two parallel rows with two components in each row. The top row contained the d-subminiature receptacle and the LED while the bottom row contained the two mini-din connectors. Figure 11 details the arrangement of receptacles for one quadrant. The side panel to the left of the front panel contained another LED hole for the power LED and four screw holes for mounting the master board. A USB-B port for debugging, Ethernet port for connection to the computer, power input and power switch were placed at the back. Due to the proximity of these values for other components, the positions of these were determined by hand and marked with a center punch.

5	Buffer				
11	-18- Buffer	-18- D-sub	-18- Buffer	-12- LED	-18- Buffer
5	Buffer				
12	-11- Buffer	-25- Mini-din	-11- Buffer	-25- Mini-din	-12- Buffer
5	Buffer				

Figure 11: Arrangement of one quadrant of receptacles. Numbers are in mm.

In order to keep the boards upright, an aluminum L-shaped piece was made from the aluminum cover that the ABS box came with. The aluminum piece would have the appropriate holes for the four d-sub receptacles and their respective screws. Older models of the boards had boards that did not contain a lip which had 4 mm diameter screw holes, unfortunately the new boards did have a lip and the screw holes were of different sizes (2 mm diameter). Figure 12 details the aluminum L-bracket.

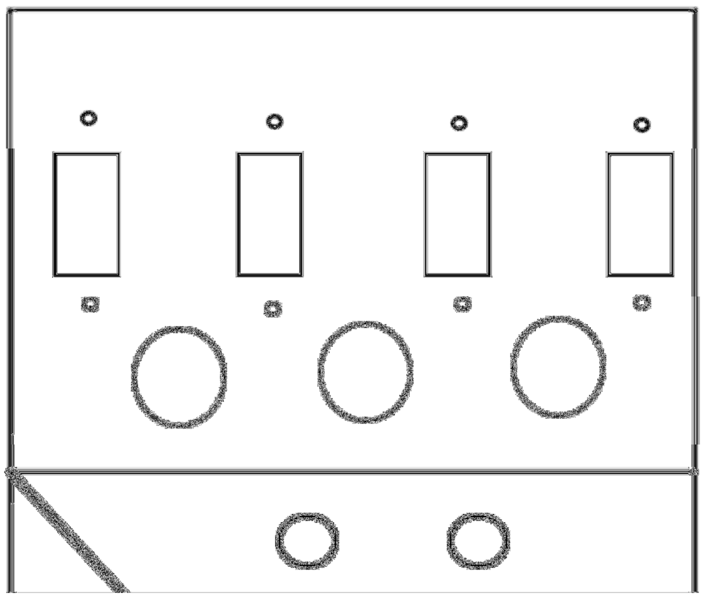


Figure 12: Aluminum L-bracket to hold boards in position in housing

The grey line in Figure 12 represents the point at which the bottom side with the two drill holes was folded up. This allowed the aluminum piece to be screwed into the bottom of the ABS case. A 45 degree cut was needed to fit in the corner as the corners had the screw wells. D-subminiature holes were drilled with the mill and dremmeled to fit the d-subminiature connectors in only one orientation. Corresponding screw holes were made to hole the boards in place. Finally, three large holes were made below the boards to account for the four pin mini-din connectors and the aluminum protective piece to fit. Only three were made because the fourth mini-din connector was off the board.

Testing

Testing was conducted on a single rat with the current build of the device. Testing of four rats simultaneously in this fashion is not feasible due the significant protocols and resources involved when putting a lab rat under anesthetic. Currently, some of the signal processing needs to be adjusting to ensure a steady, reliable heart rate reading. Additionally, occasional power cycling of the circuitry is necessary due to a bug in the software code. This will be debugged in the near future. During the actual rat testing, a signal was observed but had low amplitude. The larger amplitude available when sensing a humans pulse leads to more accurate heart rate detection. This implies that once auto-gain is fully implemented, the signal processing algorithms in place should have no problem accurately detecting the heart rate of a rat. The circuit board also does not detect a signal in some human test subjects; while seemingly disconcerting, breadth of usability on humans is in no way related to the purpose of this device. Therefore, fixing this problem will receive a notably low priority over the more significant problems stated above.

Future Work

Construction of the housing has revealed a problem with heating of the components during prolonged use. One of two steps will be taken to remedy the problem: firstly, the housing could be redesigned to accommodate more space between components (i.e. a bigger box). Secondly, a cooling system (possibly a computer fan) could be installed to maintain the components at a workable temperature. The filters associated with the software must also be changed. An extended testing session will let us deduce the correct filter coefficients to best detect the rat's high innate heart rate. Debugging of the system controller and the I^2C is also necessary to ensure all four boards are working in synchrony during experimentation. Lastly, a final lab test will be conducted. During this test, four rats will be tested. The main variables to be monitored during this test include heart rate values, data-archival ability, and possible data collision issues.

Ergonomics

During the design of the user software and hardware, human interaction with the vitals monitor was a major factor. The device must be, at a minimum, operable by someone with computer knowledge as their only experience with electronics. The Java GUI is designed to be user-friendly, and all measurement probes require minimal setup with respect to each animal, and will be simply connected into the circuit housing unit.

As seen in Figure 1 (page 4), there already is a significant amount of equipment present in the laboratory setting that our client and his research assistants work in. Therefore, the device should not add any clutter to the research environment. There will already be 14 external cords: one for each probe, one for the power supply, and one for data output. Any further wires should not be needed and thus eliminated.

Software developed and sold today is often user-friendly; hence, consumers have come to have inherent expectations of any software they use. They expect the interface to be clean and allow them to find what they are looking for with minimal effort. For software that integrates with hardware (such as this project), there should be minimal effort to establish communication between the two. These factors were taken into consideration during software design. The interface is clean, presenting all the important information well labeled. Additional information, that is not important to the user, is not displayed. The software also automatically detects the device, and if not present, a friendly message is displayed to the user. A user manual will be created to aid with any and all problems likely to be present while utilizing the prototype.

Ethical Considerations

Animal safety is the main ethical consideration in the design of a quad rat vitals monitor. The client has specified that all probes used in the final design should be non-invasive and cause no harm to the rats. During the test performed earlier this semester (and past semesters), all governmental and university standards regarding proper animal care were carefully followed [Federal Laws, Policies, Guidelines, and Other Documents]. Since none of the team members are qualified to handle animals, the research assistant was the only person who applied probes to the rat. The animals are also completely isolated from any wiring or circuit elements that could cause harm. To finish, it is important to note that the design of a quad rat vitals monitor is to serve the sole purpose of a diagnostic tool, and should not replace any standard small animal laboratory procedures. The final design will be able to accurately notify the researcher when the vital sign of any rat enters a critical range; however, it is not designed to prevent any rat from entering into such a critical range.

Cost Analysis

A sum of $2,562.90 has been allocated to the Quad Rat Vitals Monitor to date. Refer to Table 1 for specifics. This number does include various research and development costs that would not be duplicated in the fabrication of subsequent devices. This semester included the additional cost of the housing components, system controller materials, and a new power supply.

Table 1: Cost analysis from previous semesters.

Date	Vendor	Item	Cost
11/10/2008		NI Usb-6008 (2) data acquisition electronics	$320.00
12/15/2008		(approx. date) WC poster (approx. charge)	$50.00
4/1/2009		(approx. date) op amps not incl. shipping	$19.00
4/15/2009		(approx. date) +/-5 V supply not incl. shipping	$20.00
11/10/2009	Digi-Key	Electronics part (approx. charge)	$50.00
11/10/2009	Newark	Electronics part (approx. charge)	$50.00
11/13/2009	Mouser	Electronics part	$59.01
11/13/2009	Digi-Key	Electronics part	$4.50
11/17/2009	Digi-Key	Electronics part	$17.17
12/3/2009	CDW-G	Lenovo laptop computer	$659.68
4/22/2010	Sunstone	Printed circuit boards	$538.00
4/12/2010	Digi-Key	Electronics parts	$363.22
4/23/2010	Radio Shack	Electronics parts	$30.55
11/18/2010	Digi-Key	Electronics parts	$65.95
11/18/2010	Mouser	Electronics parts	$97.13
3/16/2011	Digi-Key	Housing components, System Controller parts	$201.69
4/25/2011	Newark	Power supply	$27.00
Future		Probes	$1000.00
		Total	$3562.90

Conclusion

This semester, the team has finalized four circuit boards. These circuit boards enable the measurement of breathing rate, blood oxygen saturation, temperature, and heart rate. Autogain has been implemented, as well as a custom design protocol that allows the team to transmit data in a minimal amount of time. The team has also built a system controller that communicates with the individual rat boards via I^{2}C. The master is also responsible for outputting the data via Ethernet.

The team has also working on creating user software using the open-source programming language Java. The program allows for data acquisition, signal processing (including digital filtering and peak detection), a user interface, and data archival. The software is designed to be platform independent, and is currently an executable program.

To finalize the device, a housing was constructed that enclosed all the circuit boards and wiring. This housing includes panel mount receptacles for ease of connection of probes. By enclosing the project, it is easily portable and it is protected from the environment.

References

Bjerregaard, R., Klavas, D., Collins, C., Bollom, M. *Design and Construction of a Quad Rat Vitals Monitor*. Department of Biomedical Engineering, University of Wisconsin – Madison 2009.

Bollom, M., Collins, C., Bjerregaard, R., Klavas, D. *Design and Construction of a Quad Rat Vitals Monitor*. Department of Biomedical Engineering, University of Wisconsin – Madison 2010.

Bollom, M., Xu, K., Renfrew, J., Johnson, J., Bautista, G. *Design and Construction of a Quad Rat Vitals Monitor*. Department of Biomedical Engineering, University of Wisconsin – Madison, 2010.

Decker, MJ et al. *Noninvasive oximetry in the rat.* Biomed Instrum Technol. 1989 May Jun; 23(3):222-8.

Federal Laws, Policies, Guidelines, and Other Documents. Retrieved March 3, 2011 from http://www.iacuc.org/usa.html.

Ho, J. et al. *Quad Rat Vitals Monitor*. Department of Biomedical Engineering, University of Wisconsin – Madison 2009.

I^{2}C-Bus: What's That?. Retrieved March 2, 2011 from http://www.i2c-bus.org/.

Microchip. *PIC18F2420/2520/4420/4520 Data Sheet*. Retrieved February 25, 2011 from http://ww1.microchip.com/downloads/en/DeviceDoc/39631E.pdf.

NXP (Phillips Semiconductor). *I^{2}C Manual: Application Note*, March 23, 2003. Retrieved March 2, 2011 from http://www.nxp.com/documents/application_note/AN10216.pdf.

Starr Life Sciences™ Corp. ©2009.

Using the I^{2}C Bus. Robot Electronics. Retrieved March 1, 2011 from http://www.robot-electronics.co.uk/acatalog/I2C_Tutorial.html.

Webster, J. G. (ed). *Design of Pulse Oximeters*. Philadelphia: IOP Publishing Ltd 1997.

Appendix

Product Design Specifications (v 6.2)
Quad Rat Vitals Monitor
24 Feb 2011

Team:

Matthew Bollom	Team Leader	mbollom@wisc.edu
John Renfrew	Communicator	renfew@wisc.edu
Kenny Xu	BSAC	kkxu@wisc.edu
Gabriel Bautista	BWIG	gbaustista@wisc.edu

Function: A device that is capable of recording and displaying SpO_2 levels, heart rate, rectal temperature, and breathing rate of four rats simultaneously. The purpose of this device is to help maintain appropriate anesthesia dosage on the four rats.

Client requirements: Accurately record and display SpO_2 levels, heart rates, breathing rates, and body (rectal) temperatures of four rats under anesthesia simultaneously.

Design requirements: Build a device that measures and displays the vital readings of four rats under anesthesia. The device must be able to accurately detect heart rates of up to 500 bpm and blood oxygen saturation levels to an accuracy of ±2% so that the anesthesiologist is able to determine the adequate dosage of isoflurane to keep the rats anesthetized. Device will also be designed to monitor respiratory rate (around 20 breaths/min) and rectal temperature (93-100° F).

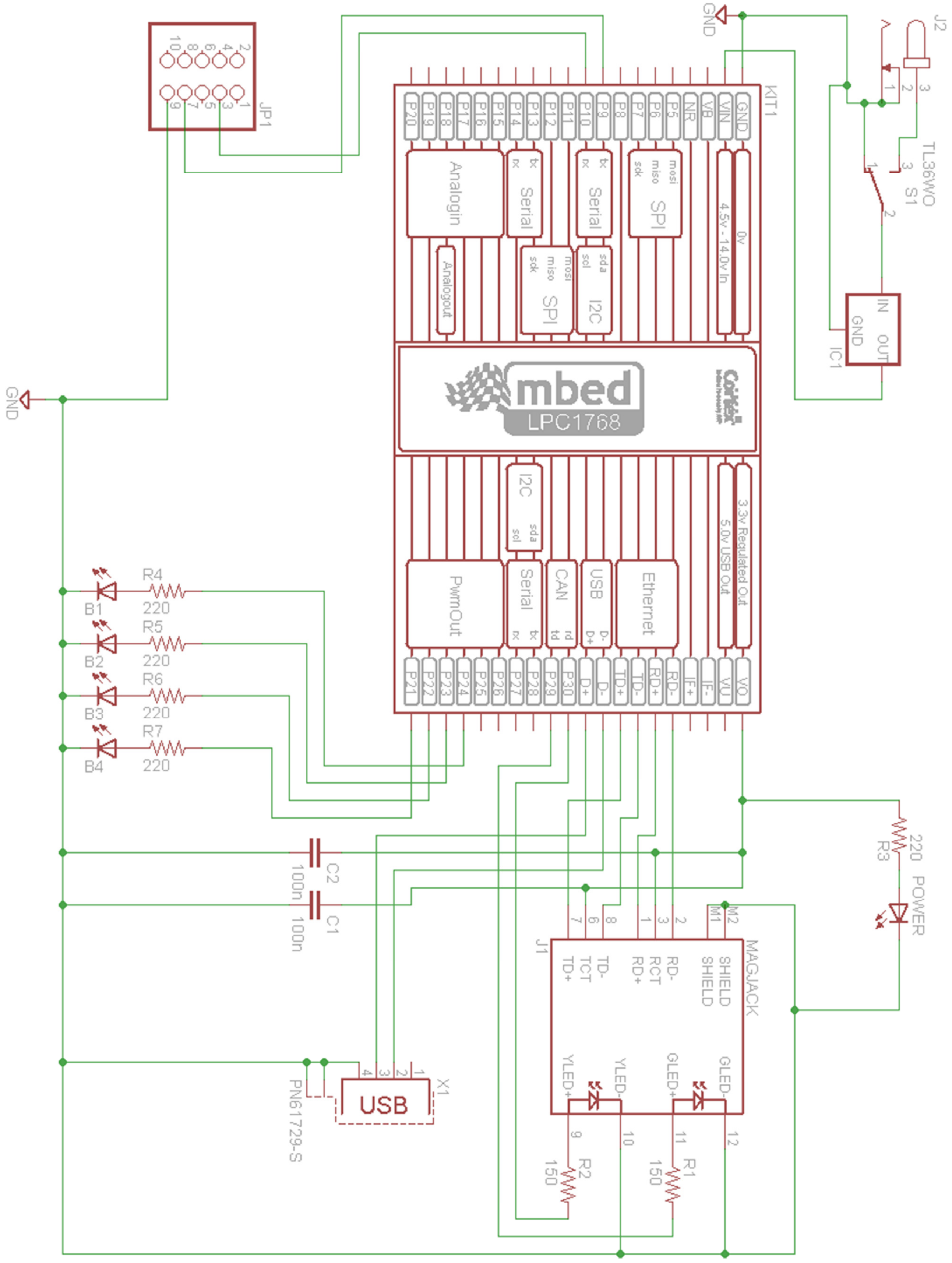
Physical and Operational Characteristics

1. *Performance requirements:* The device, at minimum, should be able to take the heart rates, breathing rates, and temperatures of four rats simultaneously and display them onscreen. It should also have running graphs showing the vitals of each rat for the duration of the experiment or for a user specified time. It should also display live traces of respiration and heart rate. The software should also record the average values of the vitals to hard disc every fifteen seconds.
2. *Safety:* The device should be safe for animal use and be consistent with the safety standards of the current rat platform.
3. *Accuracy and Reliability:* The device must be able to accurately detect heart rates of up to 500 bpm, blood oxygen saturations level accuracy of ±2%, respiratory rates of at least 30 breaths/min, and rectal temperatures of 93-100° F.
4. *Life in Service:* The device must be functional for at least 5 years, with calibration as needed. The device will be used twice a month to once a week.

5. *Shelf Life:* The device should be able to go without use for a semester and be put back into use with normal functionality.
6. *Operating Environment:* Will be used in a laboratory environment.
7. *Ergonomics:* The pulse oximeter probe should comfortably fit onto the hind paw of each rat. The probes should not be influenced by the inclusion of bubble wrap during tests. The graphical user interface must be optimized to minimize user interaction. All external probes must be user replaceable with minimal effort.
8. *Size:* Clips must be small enough so that it will not interfere with surrounding sensor and/or devices. No sensors should interfere with the PET imaging, keeping any large components inferior to the base of the heart.
9. *Weight:* The sensor system must not have a mass greater than 1 kilogram.
10. *Materials:* Commercial oximeter sensors, converted human oral thermometers, and force sensing resistors. All other materials will not be in contact with the rats.
11. *Aesthetics, Appearance, and Finish:* The circuits must be enclosed such that there are no exposed circuit elements. There must be external interfaces on the enclosure for the probes, power supply, and communication (if necessary).
12. *Software:* Please see the *Software Requirements Specification* (SRS).

- **Production Characteristics**
    1. *Quantity:* One.
    2. *Target Product Cost:* under $4,000
- **Miscellaneous**
    1. *Standards and specifications:*
        - Animal Welfare Act (7 USC, 2131 – 2159)
        - Animal Welfare Regulations
            - Code of Federal Regulations, Title 9, Chapter 1, Subchapter A – Animal Welfare
    2. *Customer:* Research organizations working with rats.
    3. *Patient-related concerns:* Currently no patient-related concerns.
    4. *Competition:* MouseOx produced by Life Starr

System Controller Schematic



Part	Digikey Part Num.	Datasheet	Size (mm)	Notes
USB receptacle	AE10158-ND	http://assmann.us/specs/a-usb-bpfs-r.pdf	20.8 x 19.4 (smaller diam)	washer
D-sub receptacle	8309-7000-ND	http://search.digikey.com/scripts/DkSearch/dksus.dll?vendor=0&keywords=8309-7000-ND	Odd shaped	
Mini-DIN receptacle - 4 pin	CP-2940-ND	http://products.cui.com/CUI_MD-30PL100_Datasheet.pdf?fileID=1762	14 (diam), 9.5 length of screw holes, 2.1 screw holes	
Mini-DIN receptacle - 3 pin	CP-2930-ND	http://products.cui.com/CUI_MD-30PL100_Datasheet.pdf?fileID=1762	14 (diam), 9.5 length of screw holes, 2.1 screw holes	
Green LED	67-1168-ND	http://www.lumex.com/specs/SSI-LXH8080GD.pdf	12 (diam)	
Red LED	67-1167-ND	http://www.lumex.com/specs/SSI-LXH8080GD.pdf	12 (diam)	
Ethernet Port *		http://www.sparkfun.com/datasheets/Prototyping/MagJack.pdf	13.6 x 15.9	
Power Switch			Square	
Power Receptacle	CP-011B-ND	http://products.cui.com/GetSpecForDigiKey.aspx?MFGNum=PJ-011B	11.6 (diam)	washer

* item was obtained from SparkFun (PRT-08534)

University of Wisconsin – Madison
BME 301

Software Requirements Specifications

Quad Rat Vitals Monitor

26 Feb 2011
(Version 1.0)

Team

Matthew Bollom	mbollom@wisc.edu	Team Leader
John Renfrew	renfrew@wisc.edu	Communicator
Kenneth Xu	kkxu@wisc.edu	BSAC
Gabriel Bautista	gbautista@wisc.edu	BWIG
Alexander Converse, Ph.D.	akconverse@wisc.edu	Client
Amit Nimunkar, Ph.D.	nimunkar@cae.wisc.edu	Advisor

Table of Contents

1. Overview

This document contains the requirements for the user software ("Software") for the Quad Rat Vitals Monitor. This Software includes data acquisition, real-time signal processing, user interface support, and data archival.

2. Document Conventions

The following are conventions used in this document:

a. The item is not open for debate
b. *Designates an item that will stored as per the requirement*
c. [Designates an item that will be replaced by the current value, as per the requirement]
d. ***Items that are open for debate***
e. ↲ designates a continued line

3. Software Support

3.1 Software implementation

The Software will be implemented using Java.

3.2 Compatibility

The Software must be compatible with Microsoft Windows XP or higher when connected to the device through USB (i.e. for debugging, Hyperterminal output).

Rationale: Java is designed to be cross-platform compatible. There are versions of RXTX available for other platforms, but they require additional work to prepare. To minimize implementation, Windows will be the main focus. Future updates may support other operating systems.

The final system will utilize data transmission via Ethernet, which much support any operating system Java runs on.

Rationale: Java provides an API, minimizing developer consideration for operating system-dependent implementations. Serial data is the exception to this.

3.3 Program execution

The Software will be an executable JAR, allowing the user to execute the program by double-clicking the program icon.

4. Hardware Support

4.1 Physical hardware

The Software will only support the hardware designed specifically for this project.

4.2 Hardware configuration

Initially, only configurations of one or four boards will be supported. Future updates may enable other configurations.

4.3 Hardware interface

The hardware will interface with the Software through an available USB port for debugging and Ethernet for system operation.

5. Data Acquisition

5.1 Overview

Data acquisition will acquire data from the system, parse it in real-time, and save it to the appropriate data structures for further processing.

5.2 Software Implementation

For USB debugging, RXTX (http://users.frii.com/jarvi/rxtx/) will be used for serial communication.

Rationale: Oracle no longer maintains the Java Communications API, so alternate solutions for serial communication will be used. RXTX has the same API as the Java Communications API, but a different namespace. It will be used because of this – all documentation will be the same between the classes.

For Ethernet integration, `TCPPacket` should be used as it provides a reliable flow of data between sources.

5.3 Data Integrity

The data acquisition classes are not permitted to modify data outside of converting bits to volts. This will help prevent fragmentation and ensure the user will get reliable data.

Software must use appropriate data structures to ensure that data cannot be accessed while it is being stored. Otherwise, data will not be consistent between different classes and may provide erroneous data. A `BlockingQueue` is suggested.

TCP will be used as the hardware system will resend the data if it is not received by the Software.

5.4 Data Parsing

Data will be provided by the hardware prefixed by a rat and signal identifier.

[rat_identifier][signal_identifier][data]

[rat_identifier] can be 1, 2, 3, or 4. *[signal_identifier]* can be R, I, F, or T.

Software will be responsible for sorting the data into appropriate data structures.

6. Signal Processing

6.1 Overview

All signal processing will be real-time. All algorithms will assume that the correct data in the correct format has been provided by the data acquisition.

6.2 Breathing Rate

The Software shall calculate the breathing rate from data provided from the hardware. Software will use peak detection to determine the times of the breaths. Software will calculate breathing rate as an average time between breaths over the ***past 10 breaths***.

6.3 Temperature

The Software shall calculate the breathing rate from data provided from the hardware. Temperature is a linear function of the data provided by hardware and can be calculated with

Temperature = -7.2988[input] + 55.636*

6.4 Pulse Oximetry

6.4.1 Overview

The principles of pulse oximetry are used to calculate heart rate and blood oxygen saturation. Two streams of data are provided by the hardware to do this – red and infrared.

6.4.2 Filtering

A 4th order FIR filter will be applied to the red and infrared data provided by the hardware. A 60 and 120Hz notch filter is applied on hardware. Experimentally determined coefficients are

b: {0.0017, 0.035, 0, -0.0035, -0.0017}
a: {1.000, -3.4648, 4.5289, -2.6477, 0.5838}

The filtering equation is given by

$$y(n) = b(1) * x(n) + b(2) * x(n-1) + \ldots + b(nb+1) * x(n-nb)$$
$$- a(2) * y(n-1) - \ldots - a(na+1) * y(n-na)$$

The filter response can be seen in Figure 1.

Figure 1: Filter response

6.4.3 Heart Rate

The Software shall calculate the breathing rate from the data provided from the hardware. Software will use peak detection on the red signal to determine the times of the peaks. Software will calculate heart rate as an average rate between peaks over the ***past 10 peaks***.

6.4.4 Blood Oxygen Saturation

The Software shall calculate the blood oxygen saturation (SpO_2) from the data provided from the hardware. Software will use peak and valley detection on the red and infrared signals to determine the values of the peaks and valleys. Software will calculate SpO_2 by first calculating the ratio between the signals

$$Ratio = \frac{\ln \frac{R_{peak}}{R_{valley}}}{\ln \frac{IR_{peak}}{IR_{valley}}}$$

and by then calculating the SpO_2

$$SpO_2 = CC * \frac{\varepsilon_{Hb}(\lambda_R) - \varepsilon_{Hb}(\lambda_{IR})Ratio}{\varepsilon_{Hb}(\lambda_R) - \varepsilon_{HbO_2}(\lambda_R) + [\varepsilon_{HbO_2}(\lambda_{IR}) - \varepsilon_{Hb}(\lambda_{IR})]Ratio}$$

where *CC* is the experimentally determined calibration coefficient and $\varepsilon(\lambda)$ is given by standard extinction coefficients in Table 1.

Table 1: Standard Extinction Coefficients

Color	Wavelength (cm)	ε	
		Hb	HbO_2
Red	660	0.81	0.08
Infrared	940	0.19	0.29

Initially, a static calibration coefficient of 0.812 will be implemented. If time permits, a variable calibration coefficient will be implemented based on values provided by the hardware.

7. User Interface Support

7.1 Overview

As the frontend to the system, an easy-to-use interface is paramount. The interface must be organized such that an individual with limited prior exposure to the program would be able to easily operate it.

7.2 Software Components

Java Swing will be used to build the graphical user interface. JFreeChart (http://www.jfree.org/jfreechart/) will be used for data graphs.

7.3 Individual Rat Display

7.3.1 Live Traces

Live traces must be displayed for FSR and either red or infrared signals.

7.3.2 Vital Values

Up-to-date vital values must be displayed for heart rate, blood oxygen saturation, breathing rate, and temperature.

7.3.3 History Graphs

There must be 4 history graphs – one for each vital. The history graphs will show the trend of the vitals. These will be updated every 15 seconds with the average of the vital value from the past 15 seconds. By default, the history graphs will show the trends over the entire experiment, but can be changed during program execution to show a specific number of minutes in the past from the current time.

7.3.4 Comments

The user must be able to enter comments for the rat. This comment will be stored in the data file as per section 8.5.

7.4 Program Display

Four individual rat displays must be displayed, each occupying a quarter of the screen. Figure 2 depicts a prototype display similar to the final display.

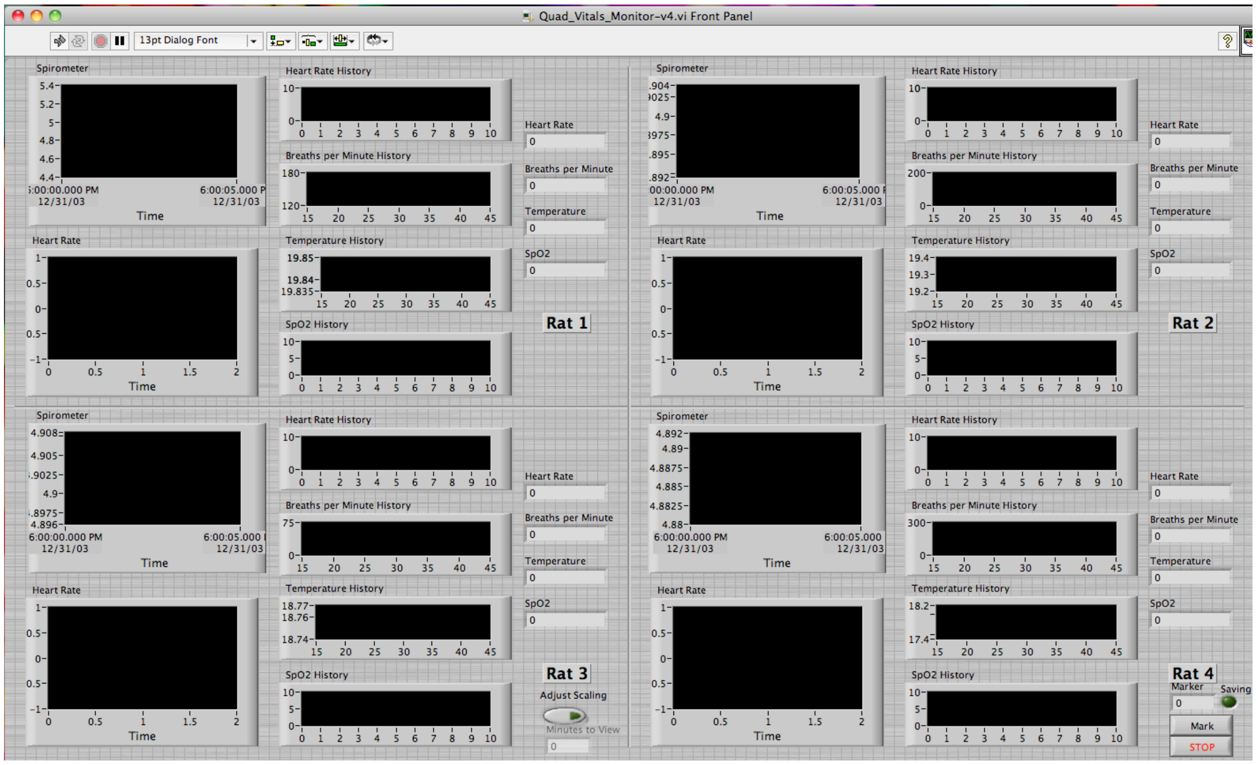


Figure 2: Sample interface as built in Fall 2009

7.5 Menu Support

The menu will allow the user easy access to a method to save a screenshot of the program, make a comment for each rat simultaneously, change program options and configuration, access help documents, and view details about the program.

8. Data Archival

8.1 Rate of data archival

Data will be archived to hard disk every 15 seconds.

8.2 File format

The archival file will be a comma-separated file.

8.3 Location of data archival

The user must be prompted for a location to archive the data at the start of program execution. The user will also be prompted to enter the name of the file. A file must be created in the specified location with the name

[user provided name].csv

Changing the archival location during program execution will not be supported.

8.4 Archival data

The archived data shall be the average value of the vital since the last archival to file and any user comments. The data will be archived as listed in Appendix 1 (section 9.1).

9. Appendix

9.1 Appendix 1 – Data Archival Format

9.1.1 Data Archival Headers

The data file shall include 2 header rows to allow the reviewer to easily differentiate the columns. These rows shall be:

“ ”,” ”,”Rat 1”,” ”,” ”,” ”,” ”,”Rat 2”,” ”,” ”,” ”,” ”,”Rat 3”,” “,” “,” “,” “,”Rat 4”,” “,” “,” “,” “

“Timestamp”,”Elapsed Time”,”HR”,”SpO2”,”BR”,”T”,”Comment”,”HR”,”SpO2”,”BR”,”T”,”Comment”, ↲
”HR”,”SpO2”,”BR”,”T”,”Comment”,”HR”,”SpO2”,”BR”,”T”,”Comment”

9.1.2 Data Archival

One row shall contain the data for one data archival. The archival file data format shall be:

“[timestamp]”,”[elapsed_time]”,”[1.HR]”,”[1.SpO2]”,”[1.BR]”,”[1.T]”,”[1.comment]”, ↲
”[2.HR]”,”[2.SpO2]”,”[2.BR]”,”[2.T]”,”[2.comment]”, ↲
”[3.HR]”,”[3.SpO2]”,”[3.BR]”,”[3.T]”,”[3.comment]”, ↲
”[4.HR]”,”[4.SpO2]”,”[4.BR]”,”[4.T]”,”[4.comment]”

[timestamp] indicates the current time as obtained from the computer clock. *[elapsed_time]* indicates the time elapsed since experiment start. _[#.*]_ indicates the rat number the item will be stored for.

9.2 Appendix 2 – Suggested Implementations

9.2.1 Generic FIR Filter

```
//variables
double[] adc;          //ADC data
double[] filt;         //filtered data
double[] b;            //b coefficients for filter
double[] a;            //a coefficients for filter
double num;            //current data point

//begin code
```

```
for(int i = adc.length - 1; i > 0; i--) {
     adc[i] = adc[i-1];
}
adc[0] = num;
for(int i = filt.length - 1; i > 0; i--) {
     filt[i] = filt[i-1];
}

double intermediate = 0;
for(int i = 0; i < b.length; i++) {
     intermediate += adc[i] * b[i];
}
for(int i = 1; i < a.length; i++) {
     intermediate -= filt[i] * a[i];
}
filt[0] = intermediate;
```

9.2.2 Absorbance Peak Detector

```
//variables
long startTime;                //program start time
double max = 3;                //max value of signal
double min = 100;              //min value of signal
double smax = 0;               //signal max
double smin = 0;               //signal min
boolean bthresh = false;       //below threshold boolean
boolean athresh = false;       //above threshold boolean
double tpeak = 0;              //temp peak
long tpeak_x = 0;              //temp peak time
double tvalley = 0;            //temp valley
long tvalley_x = 0;            //temp valley time

//begin code
long difference = (new Date()).getTime() - startTime;
if(difference > 3000) {
     if(num > tpeak) {
          tpeak = num;
```

```
            tpeak_x = difference;
            bthresh = true;
        }
        if(bthresh) {
            if(num < 0.4 * tpeak) {
                max = tpeak;
                smax = 0.875 * smax + 0.125 * max;
                //max point located at (tpeak_x, smax)
                tpeak = 0.4 * tpeak;
                bthresh = false;
                //can also calculate pulse from this data
            }
        }
        if(num < tvalley) {
            tvalley = num;
            tvalley_x = difference;
            athresh = true;
        }
        if(athresh) {
            if(num > 0.4 * tvalley) {
                mini = tvalley;
                smin = 0.875 * smin + 0.125 * mini;
                //valley located at (tvalley_x, smin)
                tvalley = 0.4 * tvalley;
                athresh = false;
            }
        }
    }
```

9.2.3 FSR Peak Detection

A running average of the last 200 data points is kept for a comparison to the current point for a determination if this is a valley in the signal.

```
//variables
int c = 0;                                      //number of points in list
double sum = 0;                                 //current sum of last 200 data points
double average = 0;                               //running average of last 200 data points
Queue<Double> list = new LinkedList<Double>();    //list of last 200 data points
double currMin = 5;                               //current min value (for comparison)
long minTime = 0;                                 //time of the min point
long lastTime = (new Date()).getTime();           //time of last min point

//begin code
//average calculation
if(c == 200) {
     average = sum / c;
     sum -= list.remove();
     c--;
}
list.add(num);
sum += num;
c++;

//peak detection
if(num < average && num < currMin) {
     //current point could be considered the min
     currMin = num;
     minTime = (new Date()).getTime();
} else {
     //current point can't be the min (because it's larger than the previous point)
     if(num < average) {
          if(minTime > lastTime + 50) {
               //old point must have been min
               //peak occurred at minTime
               lastTime = minTime;
          }
     } else {
          currMin = 5;
     }
}
```

9.2.4 Rate Calculation

This rate calculation is based on previous 10 data points, not time.

```
//variables
Queue<Long> diff = new LinkedList<Long>();  //differences list
int c = 0;                                  //number of points in list
long differencesTotal = 0;                  //current sum of differences of last 10 peaks
long differences = 0;                       //differences in point values
long lastPt = 0;                            //last point

//begin code
//get the pulse average for the last 10 peaks
if(c == 10) {
     //calculate the rate
     double rate = 1 / ((double)differencesTotal / 10) * 60000;
     //do something with the rate

     //remove the earliest peak
     differencesTotal -= diff.remove();
     c--;
}
if(lastPt == 0) {
     //first point - initialize variable
     lastPt = num;
} else {
     //store the difference in the list
     differences = num - lastPt;
     diff.add(differences);
     lastPt = num;
     differencesTotal += differences;
     c++;
}
```