Design & Implementation of a Pulse Oximeter Datalogger

Edmund FK Hunt

April 29, 2013

Abstract

One of the largest development areas for the medical industry today is the design of simple, robust and non-invasive techniques. As a result, pulse oximetery has become an important tool in measuring heart rate and the oxygen saturation of blood, without the need for expensive blood sampling. Current uses for oximetery are mainly clinical, but there is evidence of this extending into domestic use. The goal of this project is to design and implement a fully operating pulse oximeter using the PIC18F4550, with USB communication to process and log the data using a computer. A review of current designs and research was performed and it was elected to design an oximeter which measured BOC through the relative absorption of Red and IR light. From this the appropriate electronic circuitry was designed. This included and LED driver controlling the LED brightness (controlled with PWM), a TIA with bias driven by a DAC and a second stage amplifier. The signal processing was performed as part of the software along with a GUI for logging, viewing and calibrating the data. The demonstration showed positive results, the oximeter was able to correctly measure the pulse and although the oximeter could not be calibrated, results for measuring the BOC seemed promising.

1 Introduction

1.1 Concept

The aim of this project is to devise a pulse oximeter using the PIC18F4550 as datalogger. Pulse oximetry is a non-invasive method of determining heart rate, and Blood Oxygen Concentration (BOC) by taking advantage of the differing spectral absorption characteristics of Haemoglobin (Hb) and Oxygenated Haemoglobin (HbO_2). The oximeter has a range of potential uses where low cost, non-invasive oximtery is required including home testing for sleep apnea [2] and more generally in-hospital monitoring of hypoxic episodes [9].

1.2 Theory

Hemoglobin is a globular protein (folded into a compact, nearly spherical shape) and consists of four subunits [?], known as "heme groups". Each of these "heme groups" contains a conjugated covalent bond, which gives rise to the pigmentation of the Hb molecule. Conjugation lowers the energy of the highest unoccupied orbital and raises the lowest unoccupied orbital hence a spontaneous absorption occurs at a lower wavelength. This brings absorption within the visible and IR light spectrum, hence giving the Red appearance of HbO_2 (blues, greens, yellows are adsorbed). The Fe atoms in the Hb and HbO_2 exist in different oxidation states (empirically found to be Fe^{3+} in the HbO_2 case) [8]. It is this change in oxidation state, which affects the energy levels in the (conjugated) double bonds thus showing a different absorption spectra for IR and Red light.

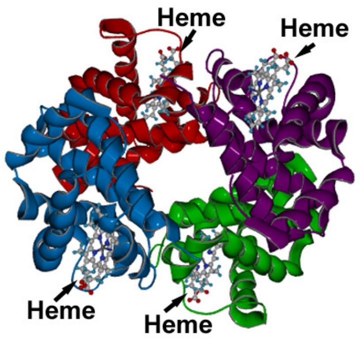


Figure 1: Molecular model of haemoglobin [3]

Figure 2. shows the different spectral absorption coefficients of HbO_2 at different wavelengths (per mol.). It is this property that can be used to calculate the arterial BOC, by comparing the optical transmission from a Red and Infrared (IR) led through a suitable body part (finger, earlobe etc).

To quantify the oxygenation of the arterial haemoglobin, SaO_2 [11] will be used where

$$SaO_2 = \frac{C_o}{C_o + C_r}$$

C_o is the concentration of HbO_2
C_r is the concentration of Hb.

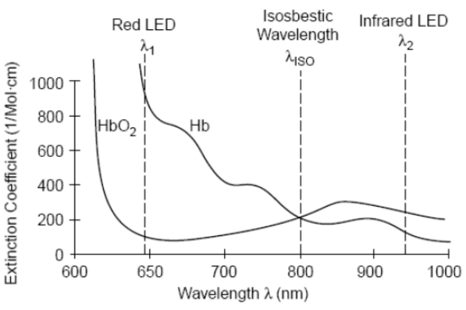


Figure 2: Absorption curves for oxygenated and oxygenated haemoglobin as a function of wavelength [12]

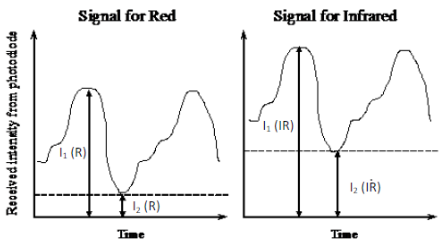


Figure 3: Definitions of I_1 and I_2 from the expected signal traces from each light source [7]

This can be calibrated with a value R_{os} which is calculated from the signal amplitude of IR and Red light, using a linear regression, calculating m and c [3]. This relationship is derived by considering the Beer-Lambert law:

$$I_{transmitted} = I_0 e^{\epsilon(\lambda)CD}$$

where $\epsilon(\lambda)$ I is the light intensity, is the absorptivity of the solute, C is the concentration, and L is the length of the light path [7]. It can be shown (Derivation in Appendix A.1) that the parameter $R_{os} = ln(\frac{\frac{I_2(\lambda_r)}{I_1(\lambda_R)}}{\frac{I_2(\lambda_r)}{I_1(\lambda_R)}}$. With the relative quantities shown in figure 2. From figure 3, it can be seen that for high S_aO_2 percentages a linear relationship exists between S_aO_2 and R$_{os}$

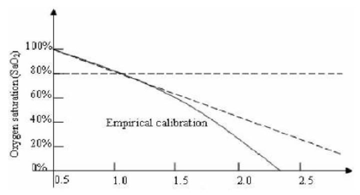


Figure 4: Calibration curve for S_aO_2 and R_{os}[3]

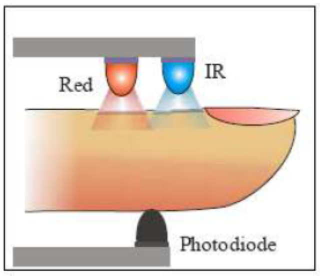


Figure 5: Basic illustration of transmissive oximeter [7].

Figure 5 shows a basic illustration of how the LEDs and photo-diode are arranged for a transmissive oximeter.

2 Design

2.1 Technical Specification

This can be found as part of the Appendix for the design report

2.2 Analogue circuit design

The following is designed to give a more in-depth discussion of the design process for the analogue circuitry, in particular component selection in addition to what has been described in the design report (see appendix). A discussion of the basics behind how each part of the circuit operates, and circuit diagrams, can hence be found there.

2.2.1 Light out circuitry

1. LEDs. The LEDs are chosen as to maximise the SNR, by selecting wavelengths which provide a large difference between the IR absorption and the Red optical absorption. From Figure 2 it can be seen that in the IR spectrum the difference in absorption is negligible, where as in the Red it is large. Hence by using IR and Red the change in Red relative to the change in IR gives the change in SaO2 Suitable wavelengths have been found to be 660nm (Red) 940nm (IR) [9]. These results have taken into account the noise introduced by the differing absorption spectra for IR and Red, for other haemoglobin compounds, namely deoxyhaemoglobin, methemoglobin, carboxygenoglobin) [12], the concentrations of which are not of interest here. A second consideration is that of the requirement that the LED should be of sufficient brightness to transmit through the finger. Too much transmission will cause saturation but the choice of LED should maximise luminosity since the LED light intensity can be calibrated down.
2. LED Driver. It has been determined that driving the LEDs will require LED drive circuitry. The PIC has the capability to sink/ source 25mA [5], and the LEDs have a maximum forward pulse current of 100mA. A number of LED drivers were considered, two circuit designed for use in pulse oximetry are compared.

   i. Polarization Circuit. The switching between red and infrared channels will be done through a square-wave between 0 and 5V using bipolar transistors.This circuit is much more complex than the AGC, but operating only requires + 5V/ 0V and hence can be controlled directly from a digital PIC output (ie. it does not need any intermediary DC voltages which must otherwise be generated from PWM or DAC.)

ii. AGC current source. Op-amp forces $v_- = v_+$ through negative feedback which fixes the voltage across R_1 and therefore the current. V_{out} follows the input voltage ensuring drain source current given by $0.5k(V_{gs} - V_t)^2$ equals the current through R_1 [11]. The circuit therefore has Automatic Gain Control (AGC) and can be used to keep the DC bias on the photodiode signal at the same point. This circuit is conceptually simple and hence easier to implement and test. To run the AGC a variable DC voltage is required from the PIC, which can either be generated through a PWM and smoothing or a DAC. This method creates an added complexity. The AGC was chosen as the preferred method, offering a simpler, more controllable and hence more robust design, but will require a DAC or PWM.

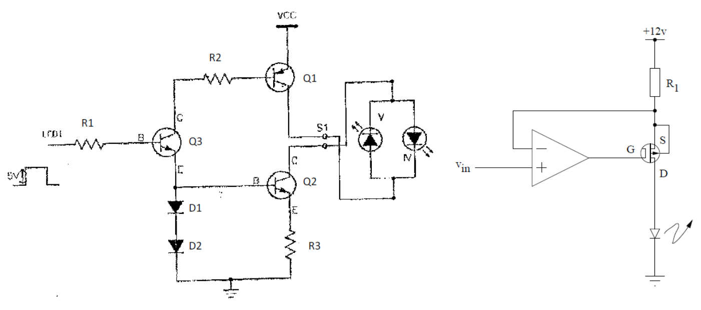


Figure 6: Candidate LED driver circuitry. One half of the polarization circuit for the LEDs(left) [7] and Op-amp driven constant current circuit for single LED (right) [11]

3. DAC vs PWM smoothing The choice of the AGC driver means that there is a requirement for the PIC to be able to generate a variable DC voltage level. This can be performed by either using the PWM facilities on the PIC or using an ADC. The PIC18F4550 has two PWM out [5], so current control would be possible by PWM (one each for red and IR). The other requirement for a controllable DC level is for setting the amplifier bias point (see below). By comparison; the DAC has a lower resolution (12 Bit) but is a smoother DC level (the PWM smoothing is not perfect). It was decided on this basis that the DAC would be more suitable for amplifier biasing where a small ripple could have a large effect on a small signal.

   It had been decided that the pulse oximeter should be able to sample at 100 samples, beat to give a good enough resolution to show a pulse trace (ie. to show an ECG style heart beat "wave"). As such, considering a maximum pulse at 240 bpm; the DAC should be capable of sampling at 400 hz. By cost benefit analysis the resolution of the DAC was chosen to be 12 bit (prices increase dramatically greater than 12bit). This gives a resolution of 1mV which is adequate for biasing an amplifier measuring a 1%, AC signal (the size of the heartbeat [9]), which for 5V DC is 50mV signal. ie. the precision of the amplifier is 2%. The next design choice was the DAC interface. One option was I^2C. By using this protocol, if required, multiple peripherals may be connected using only two general purpose input/ output pins as well as the low cost and simplicity of the bus making it suitable for the low speed sampling with multiple peripherals [6]. The alternative was the System Packet Interface (SPI). The SPI consists of a clock, control, and data lines, and lower speed FIFO buffer status lines. Again a cost benefit analysis was performed, and since multiple peripherals running from one DAC is not required a SIP DAC was chosen on the basis the same resolution can be sourced cheaper with through hole connections.

   To create a PWM DC level for the setting of LED current a low pass filter was designed with a time constant of 10ms. This was chosen to smooth the DC level as much as possible. After initial testing the response of the filter was too slow to turn the LEDs off and on correctly to ensure that IR and Red LED where not on at the same time. The time constant was therefore reduced to 0.5 μs by reducing the size of the resistor. At this much lower value it was found the PWM was smooth enough that the signal was not noticeably affected, and the time constant was easily low enough to turn on and off the LEDs in time.

2.2.2 Photodetection circuit

1. Photodiode. One photodiode will be used to measure both IR and Red transmissions. Using a single photodiode will avoid any problems arising from different performance, biasing etc. of two photodiodes imitating a change in SaO_2. This constraint requires that the photodiode be chosen with a reasonable response at both 660nm and 940nm. This requirement proved be a cause of the high cost of the photodiode (£4.84). Cheaper photodiodes tended to be very selective, and hence unsuitable for application. Figure 7 shows that both the 660 and 940nm wavelengths the response of the photodiode is reasonable, and similar.

   Other considerations are to limit the dark current caused by the reverse bias leakage current, and limit the capacitance caused by the reverse biased junction. Dark current will tend to introduce shot noise into the signal, whereas a high bias capacitance will tend to bandwidth-limit the performance of the complete TransImpeance Amplifier (TIA).

2. Transimpedance Amplifier. *A description of how a transimpedance amplifier operates is included in the design report as part of the appendix.* The op-amp for the TIA is chosen for extremely low input bias current, input current noise, and input voltage noise [1] to maximize the SNR for very small variations in the generated photo-current. The op-amp should be selected such that it can be run from the USB 0-5V supply rather than a bipolar supply, although this will introduce a DC bias that must be removed. Ideally the Op-Amp would have some in built hardening to ElectroMagnetic Interference (EMI). Typical suitable values for the op-amp design were found to be [1];

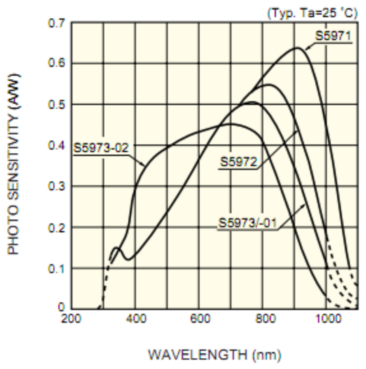


Figure 7: Spectral characterstics of chosen photodiode (S5971).

- Input Bias Current 100-1000 fA
- Input Noise Voltage 5-10 $nV\sqrt{Hz}$
- Input Noise current 0.01 $pA\sqrt{Hz}$

A suitable op-amp was found with 1pA bias current, input noise voltage 8.7 $nV\sqrt{Hz}$ Hz, input noise current 3 $fA\sqrt{Hz}$

3. AGC Circuit As described above the AGC circuit operated by setting V_- to the Op-Amp, which therfore through feedback controlled by a FET sets V_+ and ultimately the current through the photodiode (see Figure 6). The Op-amp selection was made on the basis that the input offset voltage should be minimized. This means that the voltage set by the PIC via PWM (V_+) is more exactly followed by (V_-). The op-amp should also have good noise rejection properties. An Op-amp with $500\mu V$ input offset was chosen.

   As stated, the transistor was chosen as FET. The disadvantage of using a bipolar transistor is the base current will vary with V_{ce}. A design specification for the FET is that it should have a low threshold voltage. This AGC can provide a swing from 0 to 5V. If a large V_T was chosen then a large proportion (e.g. 1/2) of this swing have not be used for controlling the current, instead it would be used to reach to the threashold voltage to turn on the transistor. This leaves only 1/2 the remaining voltage swing to control the current with in the linear region. Hence the resolution of the current control is dependent on having a low voltage threshold. A P-CHannel MOSFET with $V_T = -1.4V$ was selected as having the lowest threshold voltage whilst being suitably priced and packaged.

2.2.3 Finger Detection.

In order to determine when a finger is present there are a number of options. One option is to use the light detection circuit that already part of the oximeter. The disadvantage of this method is that it requries polling of the photodetector. This is an inefficient use of process time, power and adds complexity to the software design. An alternative is to use a micro-switch which can prompt the PIC to perform an IRQ to begin measurement. For comfort and usability a micro with a very small operating force is required. Since this may make it undetectable to the user however, indication (e.g. via LED) that the oximeter has detected a finger would be desirable for both usability and debugging. A SPST lever switch was chosen to reliably stay depressed with finger inserted. Debouncing will be performed in the software.

2.2.4 Finger Clamp

The oximeter measures heart rate by measuring the peaks and troughs of the transmitted light through the finger, which is dependent on the amount and type of material between photo-detector and LED. For this reason it is very important to ensure the finger remains still throughout the measurement. Some form of clamp is therefore required, which should also provide some force to "squeeze" the blood in the finger. It has been shown [10] that contact force has an impact on the size of the signal. As force is applied to to the artery, it begins to flatten from its original circular cross section. As the force increases this flattening of the artery gives a greater differential attenuation of the light signal since the arterial blood now covers a larger surface area in which to adsorb light. Greater than around 0.4N the applied pressure becomes greater than the internal pressure of the artery, and the artery begins to collapse restricting blood flow.To maximise the AC (pulsate) normalised to DC (background) it can be seen from figure 8 that the larges amplitudes occur within the region 0.2-0.4N, although the data shows high variability.

Since larger forces will better clamp the finger still it was decided that the clamp should have a force at the upper end of this optimum, ie. around 0.4N. Two different designs were considered, a design using a clothes peg (shown in figure 16) was trialled. No data could be found on the force applied by clothes pegs (!) so without a suitable force measuring device, it was impossible to determine the applied force. Another problem with this design is the method of holding the components in place. It was found that with the components nested in the wood too low to project and recieve a reasonable amount of light, and it was difficult to place the LEDs close enough together and lined up with the photodiode. This idea was abandoned.

Another idea was to mount the components on stripboard as shown in figure XX. The a force could then be applied using either an wrap around piece of velcro or elastic. Some elastic was sourced, and experimentally the spring constant was determined as ≈ 600 N m^{-1} Hence to achieve a force of ≈ 0.5 N the required extension is 6mm.

2.2.5 Power supply

It was considered whether the power rails would need noise removal, e.g. via capacitors across the lines or by implementing a ferrite bead between DC and AC ground. A jumper was inserted between the two grounds so this could be implemented if required.

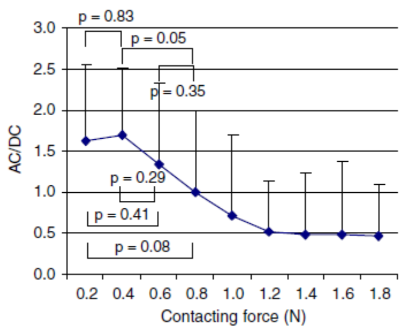


Figure 8: Ratio of AC (pulsate) to DC (baseline) signals for varying contact force, error as 1 SD, with significance levels [10]

2.3 Detailed Software and Firmware Design

2.3.1 Software Aims

1. GUI
   (a) To provide an accessible user interface such that anybody; health professionals to members of the general public are able to operate
   (b) To display a trace showing the transmitted light amplitude, such that the user can easily see a characteristic heart beat trace, over a period of few beats
   (c) To log a trace showing the SaO2 and Pulse as it changes over time, over a period of a few minutes
   (d) A menu allowing the user to easily access the other sections of the software; help, settings, calibration, log to file
   (e) Indicators; showing when a finger is present, when the logger is recording data to file, and any alarms.
   (f) To provide a "settings" user interface (away from the main GUI) where the user may select .csv output folder, the alarms etc. to keep the main GUI uncluttered.
2. File Logging
   (a) To generate .csv files recording the logged pulse rate and SaO2 over the period the user has selected to log.
   (b) A file browser such that the user can determine the location they wish the .csv file to be written to.
   (c) To record some sort of date stamp on each set of data
   (d) To save each .csv file with a unique name (e.g. of form "pulseOx_DATE_TIME.csv")
3. Calibration
   (a) To log measurements for R_{os} (as defined) and entered readings from a calibrating pulse oximeter.
   (b) To calculate a linear fit between the measured $R_{o}s$ and calibrator SaO2, using a method such as least squares.
   (c) To allow the user to enter the number of points they wish to calibrate to, and calibrate accordingly.
   (d) To display the recorded values of SaO2 and R_{os} and the calculated fit to demonstrate to the user that a calibration has been performed.
4. Signal Processing
   (a) To provide filtering of the signal, both to remove noise and to remove DC offset
   (b) To calculate the biasing required in order to control the LED brightness
   (c) To calcuate the bias required in order to bias the TIA correctly
   (d) To provide some stability in the output readings, such as bounds on pulse readings and SaO2, to prevent the output from being overly jumpy, which would result in the display being hard to read
   (e) To calculate the R_{os} values from the relative amplitudes of the transmitted Red and IR light

2.3.2 Firmware Aims

(a) To initialize the micro controller: I/O, PWM, ADC, SPI
(b) To control the LED brightness via duty ratio control of the PWM outputs through closed loop control.
(c) To generate appropriate error codes for USB communication

An example design can be found of the GUI, and software and firmware flow charts in the "design report" appendix

2.4 Filters

Two filters were required. One to removed DC component of the pulsate trace, and one to remove higher frequency components (noise) from the singal, in particular 50 Hz.

1. IIR filter. A first-order Infinite Impulse Response (IIR) filter was used for the DC removal. By placing a zero at 1 and pole at 0.992 an attenuation of the DC signal of 22dB was designed [1]. A 22dB attenuation of DC was suggested in [4], in comparison the ratio of AC to DC signal is expected to be 1% (-20dB). The bode diagram and z-plane are shown in figure 9. The selectivity that can be seen in the bode plot is the reason The transfer function for this filter is therefore

$$H(z) = \frac{1 - z^{-1}}{1 - 0.992z^{-1}}$$

By taking the inverse Z-transform of the desired transfer function, a difference equation is generated that can be used to code the filter.

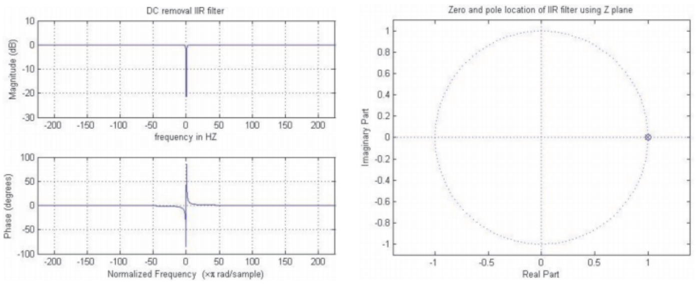


Figure 9: IIR filter Bode plot and pole-zero diagram [4]

$$y[n] = \frac{1}{a_0}\left(\sum_{i=0}^{P} b_i x[n-i] - \sum_{j=1}^{Q} a_j y[n-j]\right)$$

2. FIR filter. The Finite Impulse Response (FIR) filter was chosen for the noise filtering. Important in the design was a sharp low pass action. A filter of order 51 was used with 10 Hz cut-off, as advised in [?]. The FIR filter is computationally more demanding but is inherently stable as a result of all poles like at the origin (and therefore in the unit circle) is inherently stable. The bode plots for this filter are shown in figure 10. Again this filter is implemented using a difference equation.

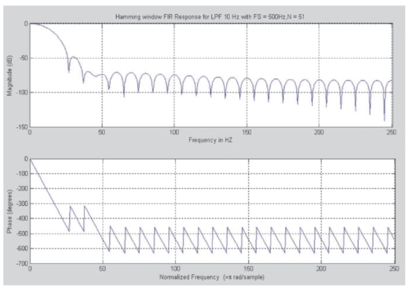


Figure 10: FIR filter plots [4]

2.5 Calibration

As discussed in the introduction there is an approximately linear relationship between R$_o s$ and S$_a$O$_2$. The calibration functions are therefore designed to derive this relationship by taking input off a calibrating oximeter and the simultaneous readings for calculated R$_o s$. This is performed by using the method of least squares for m and c;

$$m = \frac{\sum_{i=1}^{n} S_i \sum_{i=1}^{n} R_i^2 - \sum_{i=1}^{n} R_i \sum_{i=1}^{n} S_i R_i}{n \sum_{i=1}^{n} R_i^2 - (\sum_{i=1}^{n} R_i)^2}$$

$$c = \frac{n \sum_{i=1}^{n} R_i S_i - \sum_{i=1}^{n} R_i S_i - \sum_{i=1}^{n} R_i \sum_{i=1}^{n} S_i}{n \sum_{i=1}^{n} R_i^2 - (\sum_{i=1}^{n} R_i)^2}$$

where S_i is the calibrator SaO$_2$ value and R$_{os}$

[1]For a single pole/ zero as is the case here, by choosing them close together in the z-plane, yields large attenuation at that frequency, where DC lies at $\theta = 0$ (on the real axis)

3 Technical Implementation

3.1 Analogue Circuitry

The analogue circuitry was implemented as per the component selection and circuits described in section 2.2, and using the circuit diagrams that can be found in the appendix. Further points to note are; The finger clamp and associated circuitry were made detachable via header pins, the amplification circuits were also built on an additional strip board to provide more space. An (optional) cable was also introduced between circuit and clamp using multi-core wire for flexibility. It was found that the cable caused some additional noise to be introduced, but a clear signal could still be generated. The TIA and secondary amp was built as close to the photo-diode as possible in order to limit noise on the photo-diode current signal. Testing of the analogue circuitry was done in a modular style as the circuit was built. Where possible dummy signals were sent from signal generator to circuit to test.

3.2 Firmware

The firmware implemented where not the design flow charts included in the appendix. These loops were moved to the software fore ease of design. A detailed breakdown of the firmware structure can be found in the report by *Walker* who implemented the firmware code.

The firmware implementation differed from the design (see appendix) in in that the control loops were implemented in the software rather than in the firmware. The main firmware functionality was;

- Setting the PWM. The PWM in the PIC is set by first writing the period before operation, and then setting the duty ratio dynamically in order to change the DC level after smoothing.
- Setting the SPI bus. The PIC handles communications between SPI peripherals. The data bytes can therefore simply be written to the correct register `(SSPBUF)` and the PIC will control the clock and bus.
- Actioning control and read of LED brightness. A single function was written to Turn each LED ON/ OFF and to read the LED brightness (apposed to 6 if one function was used for each) which returns all 4 bytes for the LED brightness at 10 bit precision.

3.3 Software

A series of functional block diagrams have been included in the appendix to supplement this section.These give an overview of the operation and hierarchy of the functions. These show the main variable flows, and the order in which functions are called. Where important some of the contents (if statements etc.) of each function have been included. The software has been divided thusly

- *PulseOx.c* Contains the main functions for writing the main panels and functionality *Primarily written by Hunt*
- *filebrowse.c* Contains the main functions for operating the "Open Folder" file browser, used for browsing to a path to export the .csv log *Primarily written by Hunt*
- *picdriver.c* The driver for the operating the PIC from software *Primarily written by Walker*
- *POComds.c* To read and write to the PIC *Primarily written by Walker*
- *POSigProc.c* The signal processing code (includes both filters) *Primarily written by Walker*
- *PulseOx_LogSettings.c* For settings user interface and setting implementation *Primarily written by Hunt*
- *Calibrate.c* Calibration user interface, and computes linear regression using LSQ *Primarily written by Hunt*

3.3.1 Main Panel

The main panel displays all of the information which is updated on a rolling basis, ie. pulse trace, SAO2 and pulse (both logged over time and live), and indicator and alarm lights. The bulk of the main panel opperation is called from the function `int CVICALLBACK TIMER_DISPLAY` which is called by the timer at 100Hz. A loop and an if statement using a modulo 100 divider allows functions to be called at 1Hz also.

- Menu Bar. A menu bar has been implemented for a much easier to use GUI. The menu operates by calling CVI callbacks generated by *LabWindows* which perform functions this includes: "Help" for both the calibration and main panel. These functions `CVICALLBACK Menu_Help2` `CVICALLBACK Menu_Help4` open user interfaces containing annotated pictures explaining what the panels are showing and how to operate them; "Refresh" this refreshes both the pulse shape trace and the 3 minute SAO2/ Pulse log. The *LabWindows* function `ClearStripChart` is used to delete the contents of each strip chart; "Log Settings" The log settings menu calls the opening (via `LoadPanel(...)`) of the settings user interface. Here the user may choose the location to export the .csv log (detailed in "logging to file") and to turn on "Alarms" which warn the user when the pulse rate has dropped or exceeded a certain file (detailed below).
- Pulse trace plotting. The pulse trace stripchart is updated from the `CVICALLBACK TIMER_DISPLAY` control. An if statement bounds the set of function calls as an error check. The function `getPulse(&IRtrace, &RedTrace, &PulseRate, &Ros)` is called, which returns 1 for no error and 0 for error. This function exists in the signal processing c file which interfaces between the signals generated by the firmware and the software front end. The values called by reference are the IR amplitude, Red light amplitude, the pulse rate (as *double*) and the ratio of ratios, respectively. The meaning of ratio of ratios is explained in section 1.2, each trace is updated to the main panel at 100 Hz using the `Plot_Pulse` function which builds an array out of the values passed to it `(IRTrace, RedTraceas |double|)`and uses this in the *LabWindows* function `PlotStripChart` to generate the plot. The traces have 1000 points per screen, so show 10 seconds of a characteristic "heart beat" signal.

- Pulse, SAO2 numeric control updating In a similar method to the above the numeric controls showing the instantaneous pulse and SAO2 are updated. `PulseRate` is recast as an integer, since greater accuracy is unlikely to be required, and also reduces the jumpiness of the values (it is passed as a `double` to keep the precision data logging). SAO2 is fitted from R$_{os}$ using the linear regression variables (`m` and `c`) before displaying on the panel, ie `SAO2 = Ros*m + c`. `m` and `c` are declared globally as `static`, so that they are set from the calibration user interface, and can be used in `CVICALLBACK TIMER_DISPLAY`.
- Alarms There are alarms are set using the settings user interface. When the control "Set Alarms" is selected, the callback `SetAlarms` function is called which un-dims both the numerical entry for the pulse limits (in the settings panel), and the alarm in the main panel. The function `SetCtrlAttr(...)` is used for this by setting the attribute `ATTR_DIMMED` appropriately. When the user selects "OK" in the settings menu the call back `CmdOK` writes the values of the upper and lower pulse alarm to global static variables in the main C file (pulseOX.c). These are integers `PulseMinHere` and `PulseMaxHere`, and `limiton` (to indicate that the alarm has been set). These static global variables are used in the if statements controlled in `CVICALLBACK TIMER_DISPLAY`. These if statements: `limiton==1 && (intPulseRate > pulseMaxhere` — intPulseRate ¡ pulseMinhere)— contain calls to `SetCtrlVal` appropriately to illuminate the warning light. A system beep is also used when the pulse passes one of the bounds . Again it is acknowledged that in a commercial or collaborative environments defining global variables is poor practice, but for this specific application where only one or two operators administer the code it has not caused problems.
- Finger detection. Finger detection is run using the error code returned by `getpulse`. Code prevents chart updating when the error code returns 1 the finger detected light will be illuminated else not, using the `SetCtrlVal` function.

  See figure 11 for the functional diagram.

3.3.2 Calibration

A GUI is presented to the user for calibrating the oximeter. Operation is as follows;

- Two global variables are defined and initilized. `CalPoint`, to indicate the total number of calibration points has increased by one), `CalVals` an array of doubles to store the R$_{os}$ and calibrator SaO2.
- Using the function `CVICALLBACK TIMER_CLICK` (the timer callback), the current value of R$_{os}$ is polled.
- The user enters the current value of the calibrating pulse oximeter and selects "Take Value", which has the associated callback `takevalue`. This function increments `Calpoint` by one and writes the R$_{o}s$ and Calibrator SaO2 as X,Y on a plot. This is done first calling the *LabWindows* function `GetCtrlVal` to return both the current R$_{o}s$ and SaO2, to `CalVals` and then calling the *LabWindows* function `PlotXY` to plot the values on the graph control in the GUI.
- After the correct number of points have been recorded, the user then selects "Calibrate". This calls the function `CVICALLBACK CAL_CALIBRATE` calls `Calibrate` to calculate the calibration values, m and c, before writing these to the GUI controls using `SetCtrlVal`
- Selecting "Finish Calibration" calls `CVICALLBACK CAL_FINISH` which returns the user to the main panel GUI using the *Labwindows* function `HidePanel`.

Figure 12 shows the functional block diagram for the calibration GUI.

3.3.3 Logging to file

The file exporting functions allow the user to log the values of their S_aO_2 and pulse over time. The order of operation as follows:

- The user selects the location to export the .csv folder by accessing "Log Settings" from the menu bar. This loads a settings user interface, which contains a button "Choose .csv Folder", as shown in Figure XX. This calls the function `CVI CALLBACK cmdChooseCSV` which opens the file browser user interface using a call to the *LabWindows* function `LoadPane`. The user interface loads with `browse_main(void)` . This can then display a file browser and tree using a filebrowser instrument file (from *labwindows* and the function `FileBrowser_Create` which is passed the name of the file browser and the level to open as root with the string `root_level_string`. This generates the filebrowser and selecting a folder calls the function `CVICALLBACK hiddenFileBrowserCB`. The folder selected is determined using the *LabWindows* call `GetValueFromIndex(...)`. This is used to write the folder path string to a text box to indicate the path selected to the user. The filnename, which is used in a number of functions, is called by reference and is a stored as a character array is stored using dynamic memory allocation of size 200 characters with `malloc(sizeof(char) * 200)`
- The function `CVICALLBACK startlog_hit` is used to initiation the file writing. It writes a .csv file to the location selected using the `filebrowse` user interface. The name of the .csv is generated as a string ("pulse ox") appended with the current time and date. This therefore provides a unique name for each exported file. The current time and date is extracted using `CVICALLBACK startlog_hit`. This is the function called when the user selects "start log". The system time is stored in the data type `time_t` which is used specifically in C for system time values. The structure `|tm|`(as defined in `|<ctime>|`)is then used to concatenate the correct file name with `sprinf`. The file is then written and appropriate headers (name of the file, date etc.) are then written to the .csv. The file is open using `fopen` and is opened for append, so that values can be added to the log with each logging action. Each log is written using `fprintf`.
- The function `write_to_file` is used to write each value of SaO_2 and the pulse using `fprintf` and `fflush` in .csv format. It is called from the function `CVICALLBACK TIMER_DISPLAY` (ie the 100hz timer loop) inside a loop of 100 steps, to log at 1hz. This occurs when the integer `logging = 1`, as set by the `startlog` and `stoplog` functions.

This variable is defined globally since a number of functions require it, and to simplify functional calls. In a commercial piece of software liable to revision and update, global variables would be avoided, but that is not the case here.

The functional diagram is included in the appendix, figure 13.

3.3.4 Signal Processing

Detailed description of the signal processing implementation can be found in the report by *Walker*. The theory behind this implementation has been outlined in section XX. As a brief note on the implementation;

- The FIR filter coefficients (14th order) were stored in an array; where the coefficients are included in Appendix A.2
- The control of the LED brightness was implemented here. Proportional control was used in the form `RedLEDBrightness = RedLEDBrightness + Kred * (700 - RedDC)`, followed by saturation limits implemented with if statements. It was important the the proportional control was slow enough, else the LED brightness would attempt to correct itself with the pulsate signal (rather than just setting the brightness correctly for a finger insert)
- The calculation of the DC offset was also performed in *SignalProc.c*.

The pulse oximeter commands to the PIC (ie. communication between software and firmware) were stored in the file *POCmds.c*. Details can be found in the report by *Walker* who implemented this.

4 Testing

A simple test plan was made and performed, which can be found in the appendix. The testing was modular, with the LED and analogue circuitry tested stand-alone and in parallel with the software and GUI (which was supplied with a random signal). Within each part of the programme and circuitry the testing was also modular, with each element implemented and tested independent of the others, again by supplying dummy signals to the correct part of the circuit/ software.

5 Conclusions and recommendations for future work

1. Overall the circuit performed well, meeting the specifications laid out in the plan. Measurement of the light amplitude heart beat signals was very good. Clear heart beats could be seen measuring with a number of different subjects. This however was somewhat dependent on keeping the oximeter still. It was found that the 0.5N finger holder was difficult to implement well and provided too much force on the finger: the pulse signal was lost. The finished oximeter therefore had to be held in place by hand and was sensitive to wobble. Further work could be made into manufacturing a suitable finger clamp which is much more ergonomic. The elastic strap used in this design applied a force that was measured only very roughly, more time could be spent in this area, optimising the finger clamp force.
2. The measurement of heart rate and SaO2 also performed well, although less successfully. For the majority of the test the correct pulse was displayed, but was sensitive to the wobble described above. A more complex pulse counting design of software could be implemented in future work to remedy this (pulses were counted using a simple threshold in this project.) Averaging over 4 heart beats provided a good instantaneous pulse rate, but a future project may wish to build the option into the software to change this, since typically heart rate is averaged over 1 minute. The SAO2 values were un verifiable without a calibrating oximeter, but there was evidence to suggest holding ones breath caused the R_{os} value to rise (and hence the SaO2 value to fall). The calibrator itself worked completely, implemented the linear fit correctly in the test.
3. The circuitry proved fit to purpose. A pulse signal was found immediately, even without using the second stage amplifier as outlined in the design. Tests also showed that the LEDs easily saturated the photo-diode through the finger, and there was very little noise visable on the TIA. It follows therefore that this project could have been completed succesffully using cheaper components. A future project may investigate using cheaper photo-diodes and less powerful (and therfore cheaper) LEDs to cost optimise the circuit.
4. Testing of the oximeter on a cable proved to be a success, although the signal was more noisy. An alternative to the homemade twisted pair cables used here, would be to use a more noise insulated shielded cable. There is a case that this situation could be avoided with a suitably designed circuit board and surface mount components. An interesting project would be to build the entire circuit on the finger clamp, to plug directly into the computer USB using (say) a micro-USB connector.
5. The general software and GUI worked successfully under demonstration. There was a case to reduce the complexity of the main panel further, by moving certain controls into user interfaces controlled by the menu bar, such as the start and stop log. There is almost endless scope to add extra functionality to the software. As an example a more thorough calibration could be performed, if the calibration software were to calculate confidence levels of he regression fit. It could then accept or reject the calibration based on (say) a 95% confidence level. A student-t test would be suitable in this case. Moving some of the signal processing software onto the firmware would be another extension, to provide increased speed. In this project since the application is not speed limited this was not seen as necessary.
6. A more complete error checking software would be another interesting extension. In the current system the pulse and SAO2 are measured when the pulse signal falls within a reasonable bounds (ie. it ignores the transients of

the finger going into the oximeter and any major wobbles caused by this, if this creates a bpm of > 240). An alternative is to complete a cross-correlation between the heart rate signal and a "model" signal. The model signal would be an idealised heart rate signal, and the cross correlation would give the extent to which the two are correlated, thus a limit could be placed on this value to prevent recording erroneous data.

7. The part of the software which did not perform to specification is the .csv export. Although the filebrowse software generated a path, and the .csv export worked correctly, the software was unable to export to a path containing a backslash, due to the handling of backslash in c as an escape character. Some function which converted to the path to another form e.g. double backslash or forward slash may have remedied this. An unsuccessful attempt was made in this direction, although another problem such as "access denied" may have been preventing correct operation.

References

[1] Arrownac. Freescale and national contributions to pulse oximetry. Technical report.

[2] EWH Oxford Chapter. Project proposal: Pulse oximeter for apnea monitoring. Technical report, Oxford University.

[3] W. Liu Weifeng. Guowei, D. Tang. A reflectance pulse oximeter design. Technical report, IEEE/ICME.

[4] V. Markandey. Pulse oximeter implementation on the tms320c5515 dsp. Technical report, Texas Instruments.

[5] Microchip. Pic18f2455/2550/4455/4550 data sheet. Technical report, Microchip Technology Inc.

[6] D. Paret and C. Fenger. *The I^2C bus: from theory to practice.* Wiley, 1997.

[7] R. J Pereira. Projecto de um sistema digital de medida para aplicações biomédicas. Master's thesis, Departamento de Física da Faculdade de Ciências e Tecnologia da Universidade de Coimbra.

[8] B. Michalowicz A. Pin., S. Alpert. Oxygen bonding in human hemoglobin and its isolated subunits: A xanes study. Technical report, FEBS Lett.

[9] N. H. Schnapp, L.M. Cohen. Pulse oximetry: Uses and abuses. *Chest*, 98(5):1244–9, 1990.

[10] Y. T. Teng, X. F. Zhang. The effect of contacting force on photoplethysmographic signals. Technical report, IOP.

[11] N. Townsend. Pulse oximetry. Technical report, Oxford University.

[12] J.G. Webster. *Encyclopedia of medical devices and instrumentation.* Number v. 1 in Encyclopedia of Medical Devices and Instrumentation. Wiley, 1988.

The books listed here were accessed on the Internet, via *Google Books*

A Derivations and calculations

A.1 Derivation of R$_o s$

$$I_trasmitted = I_ine^{\epsilon(\lambda)CD}$$

defining

$$\alpha(\lambda) = \epsilon(\lambda)C$$

and defining the baseline transmission of a frequency as

$$I_1 = I_0e^{-\alpha_1(\lambda)D}$$

and then the transmission at peak blood flow is

$$I_2 = I_1e^{-\alpha_2(\lambda)\Delta D}$$

where ΔD is the change in transmission length due to the increased blood flow through the finger. The ratio of these two is taken

$$\Delta T = \frac{I_1}{I_2} = e^{-\alpha_2(\lambda)\delta D)}$$

To remove the δD term, which would be unweidly to measure, the singals from the two wavelegnths are considered, ie.

$$\ln(\Delta T_r) = -\alpha_2(\lambda_r)\Delta D$$

and

$$\ln(\Delta T_{IR}) = -\alpha_2(\lambda_{IR})\Delta D$$

so

$$R_{os} = \frac{\ln(\Delta T_R)}{(} \ln(\Delta T_{IR}) = \alpha(\lambda_R)$$

from Figure 2 this becomes (in terms of measured values $R_{os} = ln(\frac{\frac{I_2(\lambda_r)}{I_1(\lambda_R)}}{\frac{I_2(\lambda_r)}{I_1(\lambda_R)}}$ This value for R_{os} has a linear dependance on S_aO_2 as shown in figure 3.

A.2 FIR filter coefficients

`-0.0300398707, -0.0708376129, -0.0749721510, +0.0265132793,  +0.2664664134, +0.5946310835, +0.8841845738, +1.0000000`
+0.2664664134, +0.0265132793, — `-0.0749721510, -0.0708376129, -0.0300398707`

B Software Functional Diagrams and GUIs

Solid lines indicate functions been called, dotted lines are discrete sections of code within functions (e.g. if statements), arrows indicate files being written and variables written to the side of functions indicate that the function is setting a global variable for use by other function rather than passing. Only the main functions have been included, where a lot of similar functions e.g. `SetCtrl` are being called from the same function, ellipsis indicates this. Dots are also used to indicate that function requires a number of variables called, unimportant to the discussion.

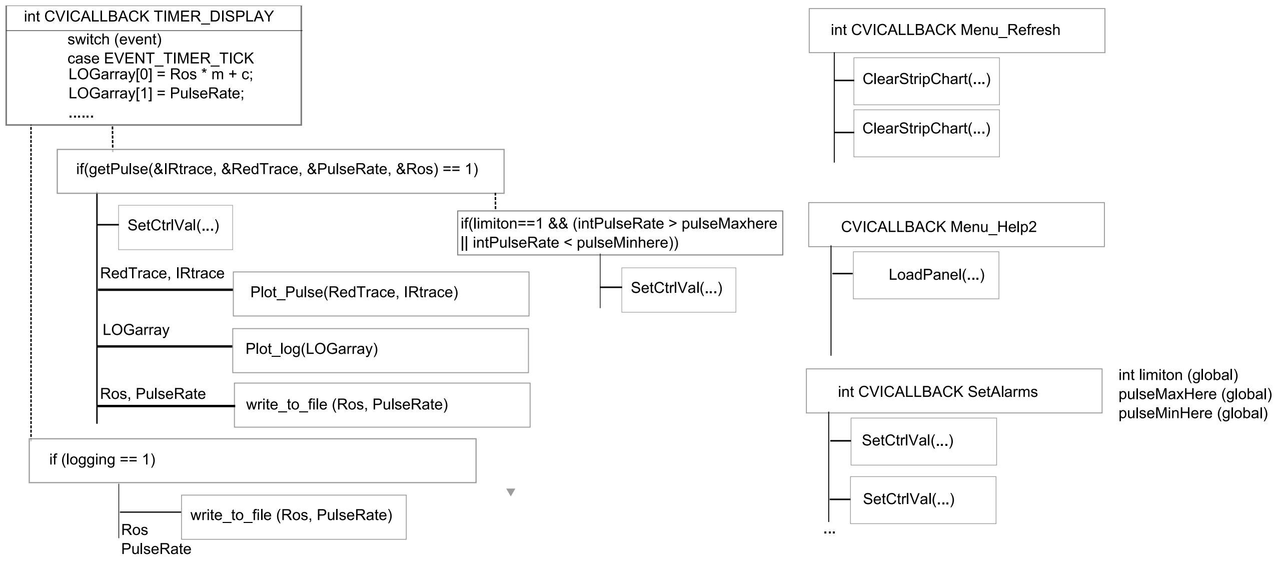
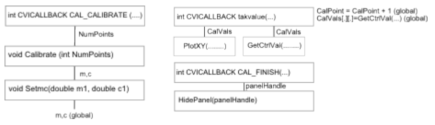
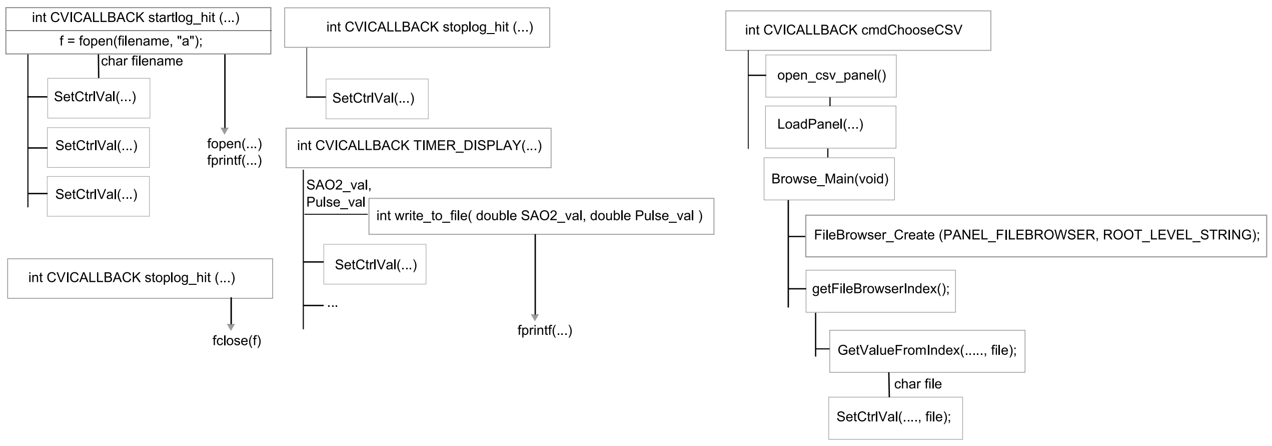


Figure 11: Functional block diagram for the main panel



int CVICALLBACK CAL_CALIBRATE (....)
NumPoints
void Calibrate (int NumPoints)
m,c
void Setmc(double m1, double c1)
m,c (global)
int CVICALLBACK takvalue(...)
CalPoint = CalPoint + 1 (global)
CalVals[.][.]=GetCtrlVal(...) (global)
CalVals
CalVals
PlotXY(.........)
GetCtrlVal(.........)
int CVICALLBACK CAL_FINISH(...)
panelHandle
HidePanel(panelHandle)

Figure 12: Functional diagram of the calibration code



int CVICALLBACK startlog_hit (...)
f = fopen(filename, "a");
char filename
SetCtrlVal(...)
SetCtrlVal(...)
SetCtrlVal(...)
fopen(...)
fprintf(...)
int CVICALLBACK stoplog_hit (...)
fclose(f)
int CVICALLBACK stoplog_hit (...)
SetCtrlVal(...)
int CVICALLBACK TIMER_DISPLAY(...)
SAO2_val,
Pulse_val
int write_to_file(double SAO2_val, double Pulse_val)
SetCtrlVal(...)
...
fprintf(...)
int CVICALLBACK cmdChooseCSV
open_csv_panel()
LoadPanel(...)
Browse_Main(void)
FileBrowser_Create (PANEL_FILEBROWSER, ROOT_LEVEL_STRING);
getFileBrowserIndex();
GetValueFromIndex(....., file);
char file
SetCtrlVal(...., file);

Figure 13: Functional diagram of the log to file coding

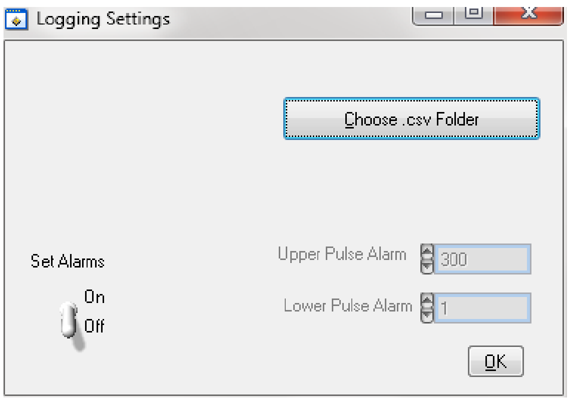


Figure 14: The logging settings panel, showing how to select a foder

C Test Method

Testing of firmware and electronics was performed by *Walker*, details of this testing can be found in that report.

Test ID	*Module*	*Date of succesful test*
1.00	Main panel load	03.06.11
1.01	Main panel quit	03.06.11
1.10	Display trace	03.06.11
1.11	Display log (SaO2, Pulse)	03.06.11
1.12	Display hidden alarm	07.06.11
1.13	Finger present test	07.06.11
2.00	Settings panel load	07.06.11
2.01	Settings panel quit	07.06.11
2.10	Pulse Alarm set	07.06.11
2.11	Pulse alarm running (ON and OFF)	07.06.11
2.22	File Browser Open	04.06.11
2.23	File Browser Close	04.06.11
2.24	File Browser Navigate to path	04.06.11
2.25	File Browser save path	04.06.11
3.01	Create export .csv file	05.06.11
3.02	Write to export .csv file	05.06.11
3.03	Write to export .csv file	05.06.11
3.04	Write to correct folder	FAIL (See report for details)
4.01	Display main help	08.06.11
4.02	Close main help	08.06.11
4.11	Display Calibrate help	08.06.11
4.12	Close Calibrate help	08.06.11
5.01	Display calibrate panel	05.06.11
5.02	Close calibrate panel	05.06.11
5.11	Record calibration values	05.06.11
5.13	Preform LSQ regression	05.06.11
5.14	Plot LSQ regression	05.06.11
5.20	Apply LSQ regression in main panel	06.06.11
6.01	Complete refresh of screen (clear all values)	08.06.11

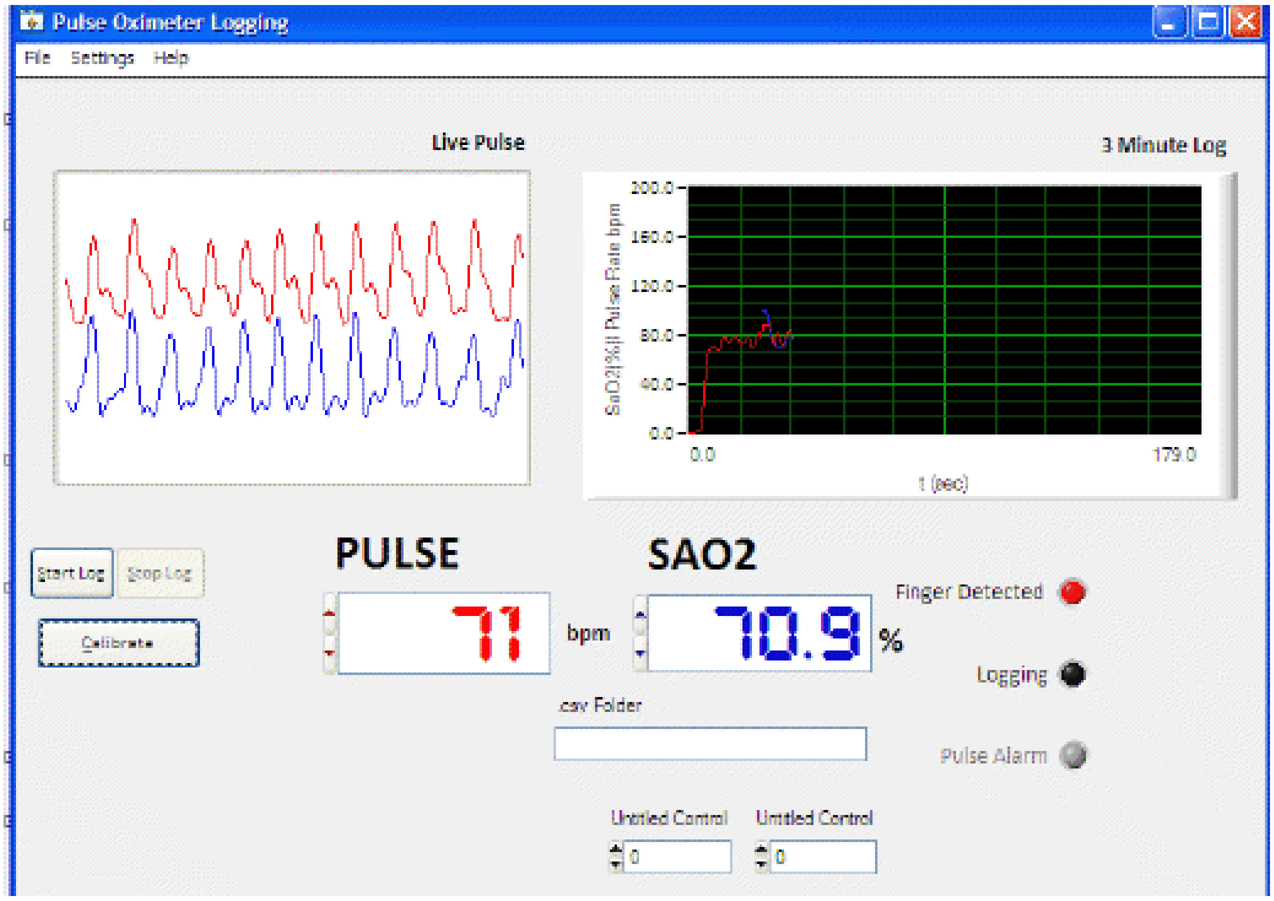


Figure 15: Screen shot from main panel GUI showing pulse trace. Note the 12 pulse over 10 seconds corresponds to 72 bpm and the read out is at 71bpm

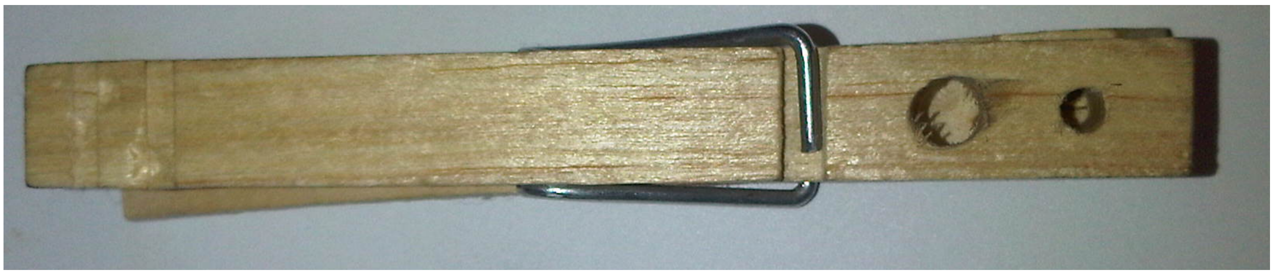


Figure 16: The attempted peg finger clamp showing holes where the IR and Red LEDs are inserted

D Project Images

E Datasheet

Pulse Oximeter

Equipment Details

Parameter	*Values*	*Units*
Interface	USB	
Light Transmission	Through Finger	
Mode of operation	Continuous	
Suitable applications	Sleep Apnea, Hospital monitoring, Sports activities	
Finger location	On board or on cable	
Degree of mobility	Immobile (requires computer	
Min Pulse	30	bpm
Max Pulse	240	bpm
Min SaO2	50	%
Max SaO2	50	%
Accuracy	±	5%

Electrical Characteristics

Parameter	*Values*	*Units*
Supply Voltage	5	V
Max Current Draw	300 (calculated)	mA
Steady State Current Draw	200 (calculated)	mA
Indicator Lights	3 LEDs (RGB)	
Measurement Wavelengths	660, 940 (nominal)	nm

Mechanical Characteristics

Parameter	*Values*	*Units*
Temperature Range	20-30 (tested), -20-80 (expected from component values)	DegC
Weight	140	g
Humidity	20-60 (tested) 10-90 (expected from component values)	%
Dimensions	12.5 x 16.0	cm

Software

Parameter	*Values*	*Units*
Operating systems	Windows 2000-Vista	
Help	Supplied within software	
Calibration	By user within software	
Display update rate	1 (SaO2, Pulse Rate), 100 (Pulse Trace)	Hz
Pulse Averaging	4	Pulse
Logging	To .csv (ascii) at 1 Hz	
Alarms	Setable, Pulse only	