THAYER SCHOOL OF ENGINEERING

Engg 129 Final Project Report

Pulse Oximetry & Bioimpedance

Amine Abouzaid, Donald B. Hondongwa, Yahia Imam, Graham Keggi

6/4/2012

Introduction

Oxygen is a curial element in the body. Cells use it for aerobic respiration, the process by which the body gets energy. The energy released in respiration is used to synthesize the adenosine triphosphate (ATP) to be stored. The energy stored in ATP can then be used in biosynthesis, locomotion, or transportation of molecules across cell membranes.[1] Knowing how much oxygen is flowing in the body will give us a lot of information on a person's state of health.

Red blood cells use a protein called hemoglobin to transport oxygen to the body via the circulatory system. Deoxygenated blood enters the heart where it is pumped to the lungs to be oxygenated. From there it is then pumped to the rest of the body.[1] In the lungs oxygen reacts with this protein to form oxyhemoglobin (HbO_2). Red cells with oxygenated hemoglobin circulate in the blood through the whole body. When hemoglobin comes in contact with cells it loses the oxygen to become deoxyhemoglobin (Hb) (deoxygenated hemoglobin).[1] By measuring the relative amounts of oxyhemoglobin and deoxyhemoglobin we should be able to infer how well-oxygenated the blood is.

Pulse oximetry (Pulse Ox) is the non-invasive measurement of oxygen saturation in blood (SpO_2). It determines SpO_2 by passing red and infrared light into the tissue then measuring and comparing the amount of light that makes it through of each wavelength. Because oxyhemoglobin and deoxyhemoglobin differ in light absorption, the amount of red and infrared light absorbed by blood is related to hemoglobin oxygen saturation. Oxygen saturation is defined as the measurement of the amount of oxygen dissolved in blood, based on the detection of hemoglobin and deoxyhemoglobin.[1] The measured signal will also have a pulsatile nature to it. This is because of the arterial blood being pumped through the system. The frequency of this pulsating is the frequency of the heartbeat, thus this measurement will conveniently allow us to monitor both the SpO_2 and the heart rate of the patient.

Literature has also shown that different types of cell have different impedances. This fact has been used to identify cancer cells in patients for example. This idea has been stretched a little further to say that people have different impedances. Even though the impedance may not be unique to any one patient, it is hypothesized that the impedance should be variable enough to be able to distinguish between people in a small group.

Our project was to combine the patient identification using their bio-impedance and monitor their SpO_2 and heart rate. We used a standard off the shelf LED-Light sensor set interfaced with a PSoC board to measure and process the data on the vitals. For bio-impedance,

[1] Freescale

we used an off-the shelf impedance measuring chip (the AD5933). Current systems that use this chip for measuring impedance mainly use what is known as a two-point measurement. This is where they pass a small voltage through the sample and monitor the current responses. In the project we built an analog front end to convert the 2 point measurement into a more accurate and safer four point measurement. In this scheme, we pass a limited current through the patient while monitoring the voltage response at two other points in the current path. The system is also equipped with an XBee module to enable wireless communication between the PSoC and a computer.

Design Overview

Figure 1 shows a block diagram of our system. Most of our data processing was done by the processor on the PSoC 5. In the next sections we shall go into details about each of the important components of our system.

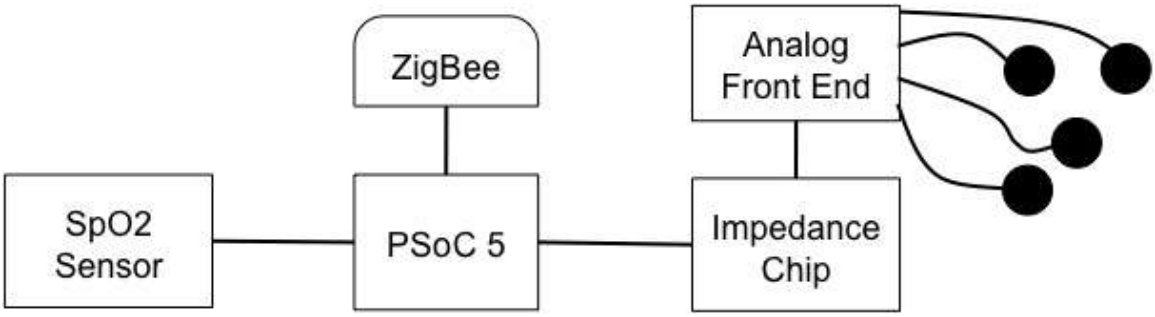

Figure 1: PSoC processing unit connected to ZigBee wireless chip via UART and the Impedance Chip via I2C and the SpO2 sensor via 4 wires as will be described later

Pulse-Oximetry

LED Driver

In order to drive the LEDs we had to address two main issues. Firstly, the red and infrared LEDs are connected together and so require a polarity shift in the driving signal. Secondly, the LEDs have different forward voltages V_f. Thus, to maintain roughly the same current through each the signal must be asymmetrical. To address these issues we decided to run the pair using a software-controlled DAC with buffer and a PWM. The DAC must be buffered since its output current is 600 µA and we require 9 mA. The opamp can supply 30 mA. The DAC is driven to a constant 2.15 V through a 60 Ω current limiting resistor. These values were chosen to make the PWM drive relatively symmetrical. This also allows a 0-5 V PWM range to drive both LEDs since with a 2.15 V bias one will always be reversed biased and one will be forward biased.

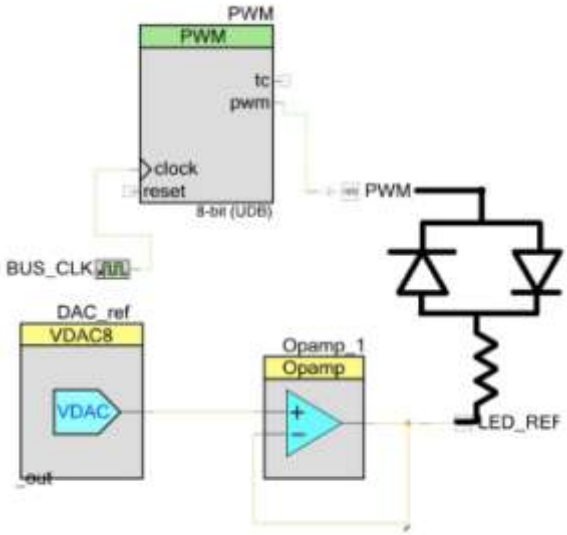

Figure 2: LED Driving Circuit

For Red we drive the PWM to 5V therefore:

$$I = \frac{V_{PWM} - V_f - V_{BIAS}}{R} = \frac{5V - 1.72V - 2.15V}{60\Omega} = 18.83mA$$

For IR we drive the PWM to 0V therefore:

$$I = \frac{V_{BIAS} - V_f - V_{PWM}}{R} = \frac{2.15V - 1.04V - 0V}{60\Omega} = 18.50mA$$

The PWM module provides the option for automatic gain control. However, we chose not to implement this feature due to a lack of time and sufficient performance without it.

Photodiode Driver

The photodiode receiver has several characteristics that we had to design for. The signal of interest is a 7 nA AC current of roughly 1-3 Hz atop a 370 nA DC offset both of which are required for correct SpO2 calculation. The first stage used a 2.5 V reference buffered out to the diode to place a voltage bias equal to the input voltage on the V- terminal of the transimpedance amplifier for greater sensitivity and stability. The next phase used a transimpedance amplifier to convert current to voltage. The PSoC limited us to a maximum resistance of 1 MΩ and 4.6 pF. Using those values we can construct an amplifier with gain of 1 MΩ and a low pass corner of 35 kHz. Following this stage we amplified the signal further using an inverting PGA referenced to 2.5 V with gain of 3 to improve the SNR. This analog signal is then sampled using a delta-sigma ADC running at 48 ksps after decimation. Finally the ADC data

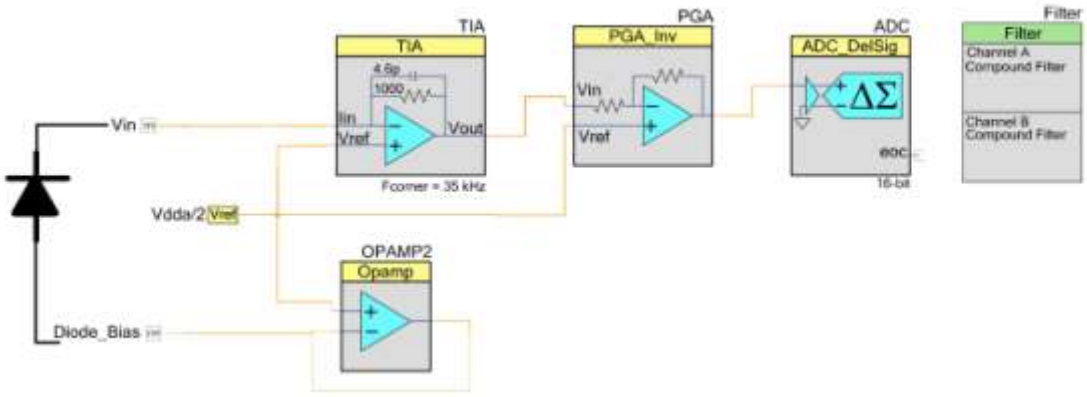

Figure 3: Photodiode Driving Circuit

is separated into red and IR data before being fed into separate high order digital low pass filters with corners at 10 Hz. These filters attenuate 60 Hz noise and any frequencies above 20 Hz by -120 dB to reduce interference with the beat detection algorithms.

The transimpedance amplifier yields: $V_O = V_{REF} - I_{IN} \cdot R_G$

The PGA yields: $V_O = V_{REF} + G \cdot \left(V_{IN} - V_{REF}\right)$ where G is a negative number

Input to the ADC: $V_{ADC} = V_{REF} + I_{IN} R_G |G|$

Since I_{IN} is 0.37uA DC offset with 7nA AC signal and G = -3, R_G = 1 MΩ the ADC sees 3.61 ≤ V_{ADC} ≤ 3.631 which seems like a very narrow range but is limited by the DC offset. If the PGA gain were increased to 7 the DC offset would place the V_{ADC} > 5 V and outside the rails. Since we are using a 16-bit ADC we get a signal that changes by 275 counts. We need to preserve the DC offset since the SpO2 calculation requires taking a log that would be disrupted by removing the offset.

Noise Considerations

Since we have a very small signal we have to maintain a high SNR to achieve accurate results. In general we attempted to maximize the early stage gain to reduce input referred noise. To this end we picked the TIA gain to be the maximum of 1MΩ, additionally the TIA generates noise spread across the spectrum so we chose the C to be the maximum of 4.6pF to produce the lowest corner frequency to limit the range of noise passed. The resistors in the PGA generate thermal noise, which we minimized by using the smallest restores available. The Delta-Sigma ADC generates both sampling noise and quantization noise which was minimized by using a very high sample rate and high number of bits.

TIA: Produces noise spectrum of roughly $e(f) = 900e^{\frac{-f}{750}} + 40 \quad \frac{nV}{\sqrt{Hz}}$ which we square and integrate over the bandwidth of 35 kHz to yield a total noise at the output of the TIA of 61nV.

PGA: Produces noise due to thermal noise of $e = 8kTR_B \quad \frac{V^2}{Hz}$ and since R_B = 60 kΩ we get a total noise of 65 pV.

ADC: Produces sampling and quantization noise, which from the datasheet given our settings produce a total noise of 445 µV.

Therefore we get a total input referred noise of: $e_{tot} = \frac{e_{ADC}}{G_{PGA} \cdot G_{TIA}} + \frac{e_{PGA}}{G_{PGA} \cdot G_{TIA}} + \frac{e_{TIA}}{G_{TIA}}$ and given G_{PGA} = 3 and GTIA=1 MΩ we get e_{tot} = 148 pA. We could improve this by using devices that allow larger TIA gains and lower corner frequencies or using a faster ADC to reduce the main source of noise.

BPM and SpO_2 Software

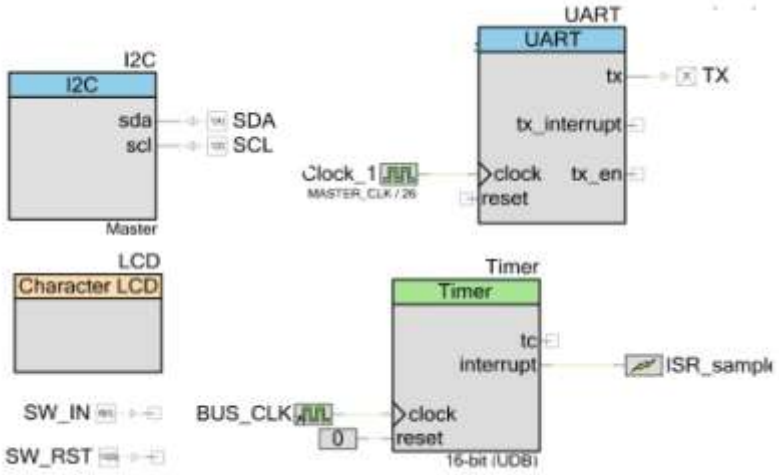

Figure 4: I2C, LDC & Xbee Modules

In addition to the hardware necessary for our signal processing chains we added communications modules and timers to run the software. The I2C module runs at 400 kHz and communicates with the bioimpedance circuits, the UART runs at 115200 baud and is used to transmit to the ZigBee wireless chip, the LCD module displays BPM and SpO2 data to the user, finally a 1 kHz timer triggers an ISR that is the core of our circuit.

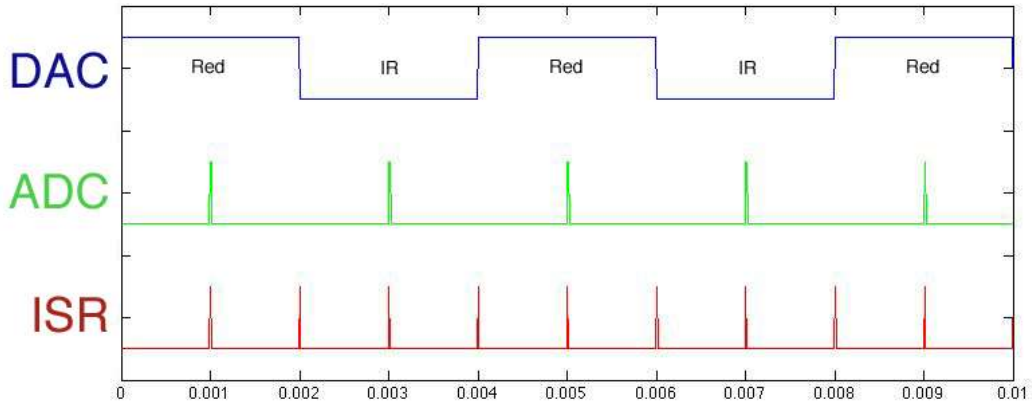

Figure 5: Internal Timing of the PSOC

The software-timing diagram is shown above. Here the ISR triggers every 1 ms, the ISR routine only sets a flag to 1 and nothing else. This is to employ the concept of top and bottom half interrupts. The top-half is the flag setting routine, this is as short as possible since the ISR routine blocks other code from executing, if the routine takes too long it will disrupt other operations. The bottom-half waits for the flag, then clears it and does the actions required and if it takes too long to complete it can be interrupted by other ISR top-halves so no actions are missed when computations slow down.

Every time the ISR triggers the code proceeds through 4 modes, 1-4. In modes 1 and 3 the PWM/DAC is toggled to enable the other LED, in modes 2 and 4 the ADC takes a sample of the photodiode and adds it to the appropriate filter channel for IR or Red. This scheme ensures that the ADC sample it taken far enough away from the edges of the LED on/off transition so that our data does not include transients in the optical intensity. This results in a sample rate of 250 Hz per channel with a very well defined separation between red and IR data (impossible to mix up since both reading and activating are controlled by the same timer and thus are always synced).

The transmission of data is done via the UART running at 115200 baud and consists of 9 bytes per 4 ms cycle. The byte string is sent as follows:

Table 1: Data Transmission Scheme

Sync	*IR 0-6*	*IR 7-12*	*IR 13-16*	*Red 0-6*	*Red 7-12*	*Red13-16*	*BPM*	*SpO2*
0xAA	*8-bits*	*8-bits*	*8-bits*	*8-bits*	*8-bits*	*8-bits*	*8-bits*	*8-bits*

The number of bits of each 16-bit IR/Red data value are kept less than 8 so that it is impossible to have the highest bit of any data byte be ‘1’ this allows that bit to indicate the sync

character. The sync character is required so that the receiver can grab the character and then know the next 3 bytes are IR data, then 3 of Red data then BPM, and finally SpO2.

Heart rate was computed using a moving average and a debounced software comparator, the data was smoothed using a 4-beat moving average as well as a timeout controlled outlier data point rejection. The function compute_bpm is called every sample and sent the latest red data point. The function returns 0 is there was no beat detected, 1 if there was a beat detected but it was deemed to be invalid, or a number that is beats per minute. The beat detection algorithm was simulated with data collected in MatLab and then implemented on the PSoC as follows:

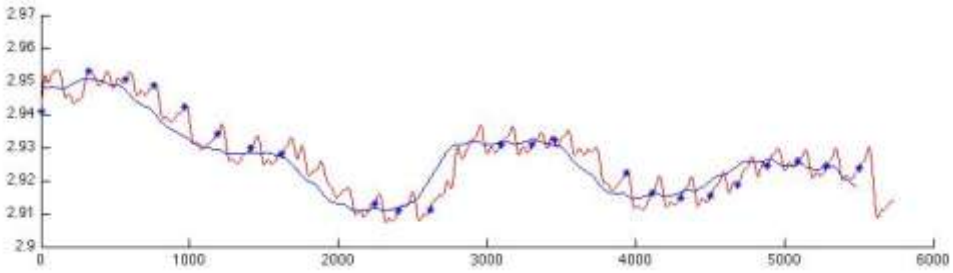

Figure 6: Red is the filtered data input to the function, blue line is the moving average calculation, and the blue asterisk are where the algorithm detects a beat

1. Replace oldest data point with new one in 128 byte vector
2. Compute average of the vector
3. If the data point is above the average detect a beat
4. Wait 100 ms
5. Repeat steps 1&2 then go to step 6
6. If the data point is below the average proceed
7. Wait 100ms
8. Go back to step 1

By using moving average we are essentially creating a simple 0.5 Hz high pass filter since we compare to that it essentially removes low frequency signals. The 100 ms delay ensures that the signal doesn't cross the mean then due to noise cross in the other direction, which would trigger a beat detection. This algorithm is fairly accurate except for the large DC jumps that occur during movement of the subject since the average lags behind the jump and doesn't intersect the data for a few beats.

The keep the data accurate we need to reject bad data in two steps, first is to identify bad data, then reject it. This is done first with a simple moving average of the last 4 beats to smooth out the data. Second we know BPM should either be stable or slowly changing so we can reject data that changes too quickly to be real. This is done by comparing the data against

the current average of the last 4 beats and ignoring it when it is more than 15 BPM above or below the average. To prevent the algorithm from rejecting sharp transitions that remain stable or changing people a 10 second timeout allows known 'bad' data to be included. This works as follows:

1. Beat detected
2. Compute average of last 4 beats
3. If |BPM-Avg|≥15 flag the data as bad
4. If it has been ≥10 seconds since we got any good data flag data good anyway
5. If the data is good add it to the 4-beat vector
6. Compute and return average

is important to send the software a signal that the beat was detected but was invalid so the SpO2 routines can reset their tracking of Red and IR min and max values that are computed on every beat.

The SpO2 routines are called on every sample. They compare the current min and max values for each channel against the incoming data and update to maintain the maximum and minimum data values for IR and Red during the most recent beat. Then the routine computes SpO2 via the extinction coefficient method described in a 1996 research paper as follows:

$$Ratio = \frac{\ln\left(R_{MAX} - V_{REF} / R_{MIN} - V_{REF}\right)}{\ln\left(IR_{MAX-V_{REF}} / IR_{MIN} - V_{REF}\right)}$$

$$SpO_2 = 100\% \cdot \frac{\left(er_2 \cdot Ratio - er_1\right)}{\left(er_2 - eo_2\right) \cdot Ratio - \left(er_1 - eo_1\right)}$$

$$er2 = 0.154 \quad er1 = 0.747$$

$$eo2 = 0.272 \quad eo1 = 0.102$$

Two push buttons control the mode of operation of the device, button A switches the device from BPM/SpO2 mode into bioimpedance mode and button B switches it back.

Bio-impedance

The AD5933

We powered the AD5933 using 3.3 V from the PSoC. To configure the chip, we wrote the start frequency, number of frequency increments, and size of the frequency increments to their respective registers. In our case these values were 1000 Hz, 180 and 50 Hz, creating a sweep from 1000 Hz to 10 kHz at 50 Hz increments, with a total of 180 data points. We then placed the AD5933 in standby mode and defined its output voltage range (2 V_{pp}), then issued a reset command followed by a start frequency command. After a delay to allow the circuit to settle, we then issued a start frequency sweep command. We polled the status register until it indicated that valid data was available. The real and imaginary components were then read from their respective registers, after which we checked the status register again to verify if the frequency sweep was complete. If not, an increment frequency command was given, and the status register was polled again until new valid data was available.

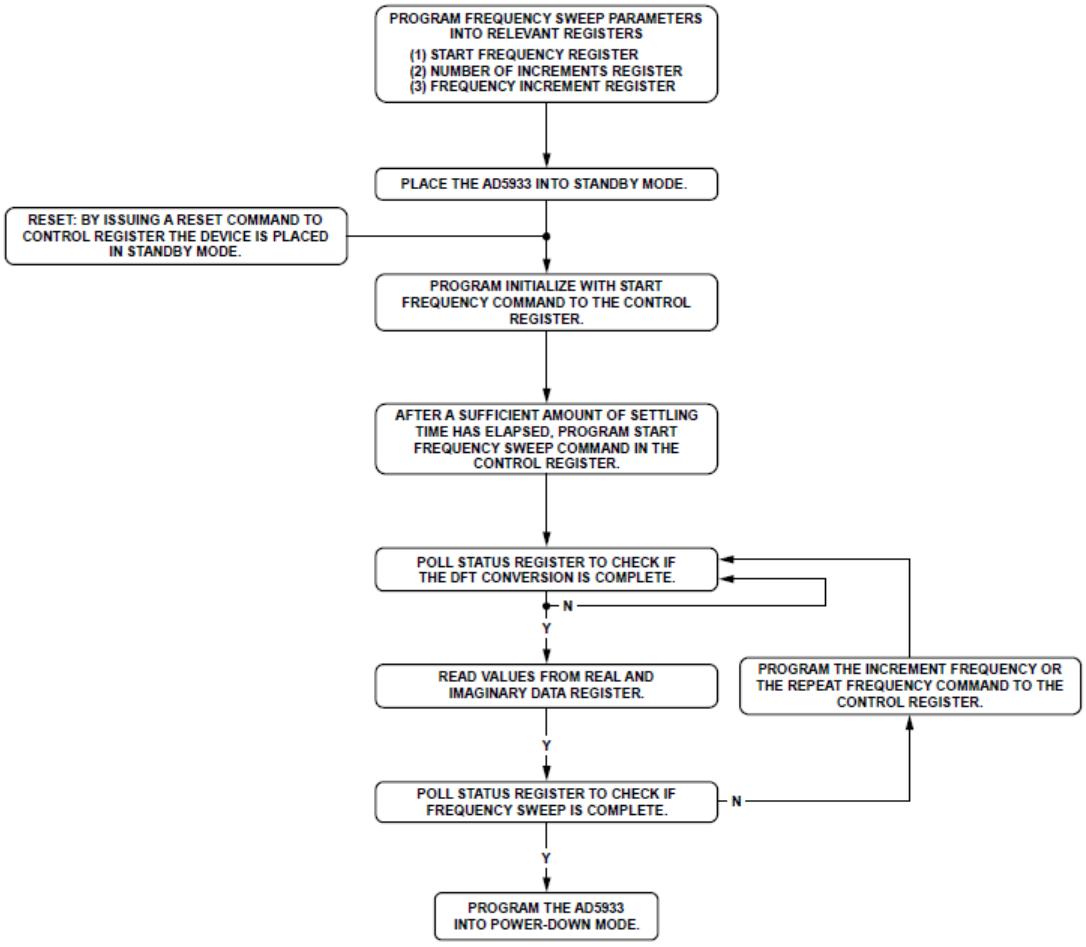

Figure 7 Frequency sweep flowchart diagram from AD5933 datasheet

Because in the 4-point impedance measurement setup the current I_{in} into the V_{in} pin is proportional to $V_{out}*Z_{load}$, rather than V_{out}/Z_{load} as in the 2-point case, I_{in} is a direct measure of the load resistance Z_{load}. The magnitude of the impedance is therefore proportional to $M = \sqrt{R^2 + I^2}$. The phase could have been computed as $\varphi = \tan^{-1}(I/R)$.

To characterize the bioimpedance of a person, we computed the impedance magnitudes at each frequency scanned by the AD5933, saved them in an array, and sent them to a PC using an XBee module. There the magnitude data could be compared to previously logged data to determine the identity of the person being scanned.

The SNR achieved by the AD5933 is typically 60 dB. It follows that this is the maximum SNR that can be achieved by our setup, and should be taken as an upper bound for our target SNR for the analog front end. The chip is rated to measure impedances in 2-pin configuration of up to 10 MΩ. This implies that the input impedance is much greater than 10 MΩ and thus significantly greater than any resistors in our AFE.

Analog Front End:

Based on the lecture given by Professor Halter, we decided to use a four point measurement system for bio-impedance. Figure 8 shows the full diagram of our circuit.

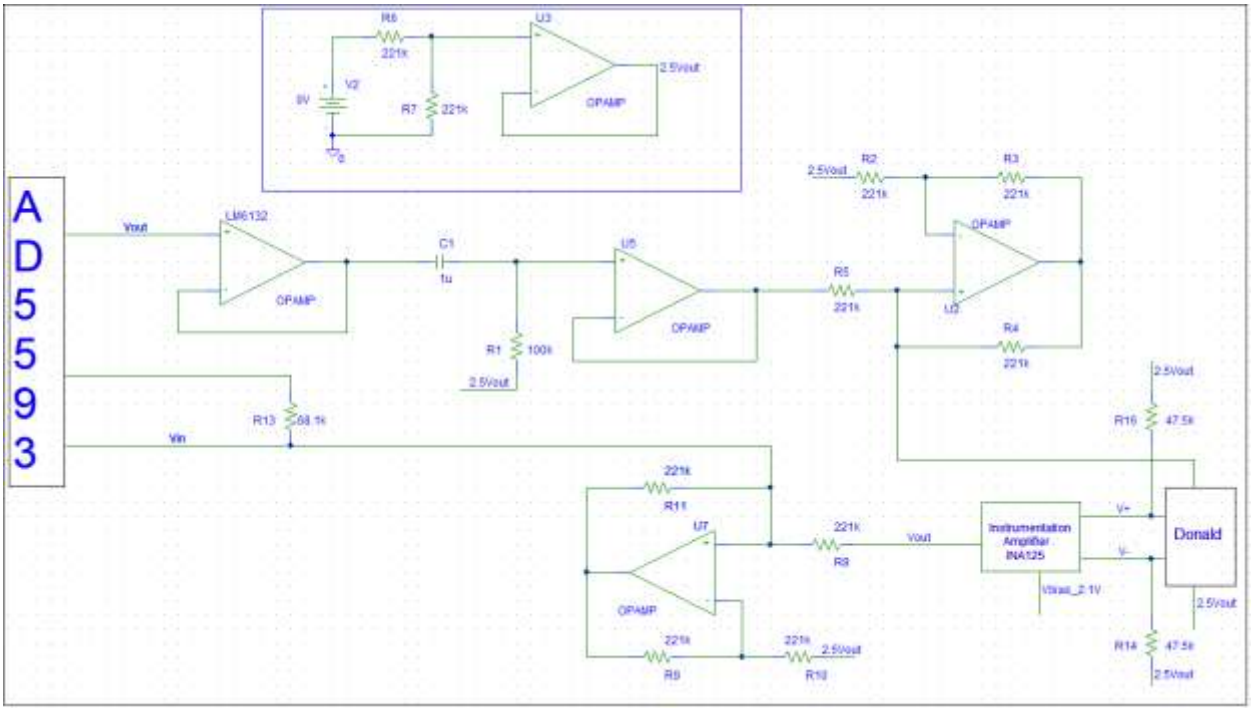

Figure 8: Analog Front End

We used the AD5933 chip to generate a 2 V_{pp} sinusoid that has a 1.48 VDC offset voltage. The signal was then buffered before passing through a high pass filter with a corner frequency of 1 Hz. Then the signal was buffered before being converted into an AC current using a Howland current source. The output of the current source was then used to drive the tissue load. We used the INA125 instrumentation amplifier chip to probe the tissue at two different locations. The I.A outputted a voltage which was then converted back to a current using a second Howland current source before going through a transimpedance amplifier in the AD5933 chip.

Going into more depth about the circuit, we concluded that the first buffering stage after the AD5933 was probably unnecessary even though we saw some distorted sinusoids from time to time. The high pass filter was biased at 2.5 V to put the signal at mid-rail to avoid clipping and to allow for more amplification later on. The cutoff frequency was set at 1 Hz to block the DC component without affecting the test signal which was at or above 1 kHz. We then buffered the signal again between the high pass filter and the Howland current source as the resistors were similar in magnitude and voltage divider effects would have appeared. It is worth mentioning that the input impedance of the LM6132 is 104MΩ which is extremely large compared to the resistors we are using and posed no problems during operation.

For the first Howland current source, we chose 221kΩ resistors. With 2 V_{pp} from the AD5933, these resistors limit the current to 9 μA_{pp} which is safe to enter the human body. We also biased the Howland at mid rail to eliminate DC current passing through the human tissue. The measuring probes from the human tissue were connected to the positive and negative terminals of the I.A. Since those pins source approximately 10nA, we had to make sure they do not saturate if the human impedance was too large. We added 47.5 kΩ resistors to 2.5 V to allow a place for that current to sink. In retrospect, this also created a relatively low impedance path through which the drive current in the TUS could flow, which possibly affected our results. In addition, the impedance of the TUS was likely significantly lower than we anticipated, thus rendering these resistors unnecessary. We chose a 681 Ω resistor to be placed across the I.A to provide a gain of 90 for the output which amplified the signal enough without causing it to saturate. We also biased the output of the I.A to 2.1 V to match the initial bias of V_{out} from the AD5933. After that, we used a second Howland current source to convert the signal back into what the AD5933 is expecting. In retrospect, we could have just used a resistor for simplicity, which would have not only reduced noise but also decreased undesired attenuation at higher frequencies.

Thermal Noise Analysis:

$$S_{out}(f) = S_{in}(f)|H(f)|^2$$

$$\overline{V^2}_{noise} = \int_0^\infty S_{out}(f)df$$

$$noise_{rms} = \sqrt{\overline{V^2}_{noise}}$$

Unity gain buffers:

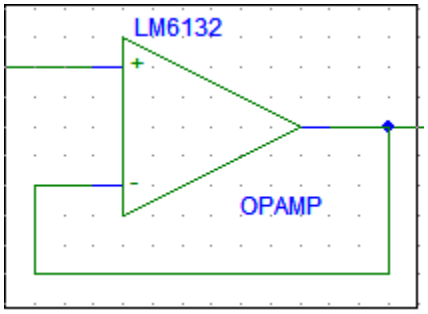

Figure 9: Unity Gain Buffer

We used the LM6132 and the input referred noise is given in the data sheet

$$S_{in}(f) = 7.29 \times 10^{-16} \frac{V^2}{\sqrt{Hz}}$$

$$S_{out}(f) = S_{in}(f)|H(f)|^2 = S_{in}(f) = 7.29 \times 10^{-16} \frac{V^2}{Hz}$$

$$\overline{V^2}_{noise} = \int_0^\infty S_{out}(f)df = \int_0^\infty S_{in}(f)df = 100\text{kHz} \times S_{in}(f)$$

$$noise_{rms} = \sqrt{\overline{V^2}_{noise}} = 8.54 \times 10^{-6}\, Vrms$$

High Pass Filter:

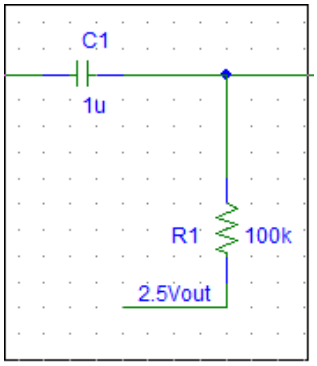

Figure 10: High Pass Filter

The input referred noise of high pass filter is the same as a low pass filter. Therefore at 296 °K

$$noise_{rms} = \sqrt{\frac{kT}{C}} = 6.203 \times 10^{-8}\, Vrms$$

Instrumentation Amplifier:

We used the INA125 as our instrumentation amplifier. Its output referred noise is given

$$S_{in}(f) = 1.6 \times 10^{-15} \frac{V^2}{\sqrt{Hz}}$$
$$S_{out}(f) = S_{in}(f)|H(f)|^2 = S_{in}(f) = 1.6 \times 10^{-15} \frac{V^2}{Hz}$$

Since we had a gain of 90 on our I.A

$$\overline{V^2}_{noise} = \int_0^\infty S_{out}(f)df = \int_0^\infty S_{in}(f)df = 100\text{kHz} \times S_{in}(f) \times 90^2$$
$$noise_{rms} = \sqrt{\overline{V^2}_{noise}} = 1.11 \times 10^{-3}\, Vrms$$

Transimpedance Amplifier:

This is the transimpedance amplifier that is at the input of the AD5933 chip with a 68.1 kΩ resistor

$$S_{in}(f) = 4kTR$$
$$S_{out}(f) = S_{in}(f)|H(f)|^2 = S_{in}(f) = 4kTR$$

$$\overline{V^2}_{noise} = \int_0^\infty S_{out}(f)df = \int_0^\infty S_{in}(f)df = 100\text{kHz} \times S_{in}(f)$$

$$noise_{rms} = \sqrt{\overline{V^2}_{noise}} = 10.6 \times 10^{-6}\, Vrms$$

Howland Current Source:

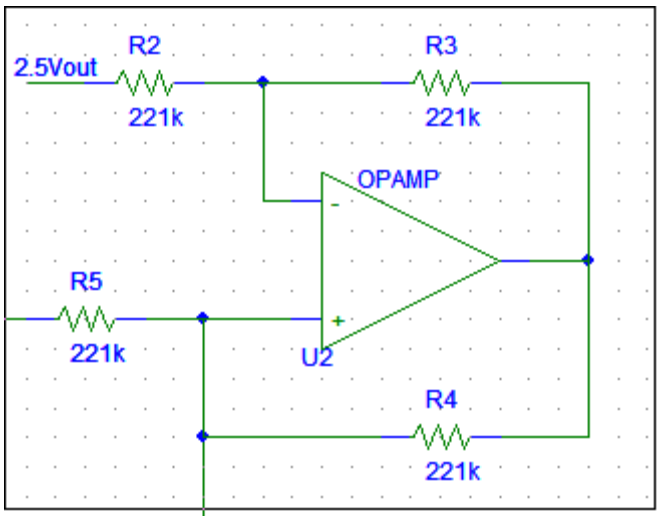

Figure 11: Howland Current Source

Each Howland current source has 4 resistors (R) which produce thermal noise as well as the load resistance R_L. Let's analyze one of the 4 resistors (R)

$$|H(f)|^2 = \frac{R_L^2}{R^2}$$

$$\overline{V^2}_{noise} = \int_0^\infty \frac{4kTR_L^2}{R}\, df$$

$$noise_{rms} = \sqrt{\overline{V^2}_{noise}} = 3.44 \times 10^{-10} \times R_L\, Vrms$$

Here is the analysis for R_L

$$|H(f)|^2 = 1$$

$$\overline{V^2}_{noise} = \int_0^\infty 4kTR_L\, df$$

$$noise_{rms} = \sqrt{\overline{V^2}_{noise}} = 4.04 \times 10^{-8} \times \sqrt{R_L}\, Vrms$$

Let's assume that the human tissue has 10 kΩ resistance for our calculations of noise in the circuit[2]

Table 2: Noise Contributions

Noise Source	Count	Noise V_{rms}
Buffers	2	1.708×10^{-5}
High Pass Filter	1	6.203×10^{-8}
Instrumentation Amplifier	1	1.11×10^{-3}
Transimpedance Amplifier	1	10.6×10^{-6}
4Howland Resistors	2	3.44×10^{-6}
Howland Load Resistor	1	4.07×10^{-6}

Now we can calculate the total noise of the system by adding up all the noise powers and taking into account the I.A gain.

$$Total\ noise = 0.00195\ Vrms$$

The AD5933 outputs a 2 V_{pp} signal which returns to the device as a 2.5 V_{pp} signal or 0.884 V_{rms}. We can calculate the SNR of the front end to be

$$SNR = 20log\left(\frac{Signal}{Noise}\right) = 53.1\ dB$$

The AD5933's typical SNR is 60 dB, so the AFE SNR is adequate, though it would ideally be higher.

Patient Identification

To identify the patient we used several Matlab algorithms. First we collect several datasets for the patient and we use the average as the calibration data. We sent the bioimpedance data to Matlab via Xbee and ran 3 different identification algorithms.

1.Total Difference:

The algorithm subtracts a known profile for a person from a newly acquired data. It then takes the absolute value and selects the person with the smallest error.

2. Root Mean Square:

The algorithm takes the root mean square difference of a known data set and a newly acquired one. The person with the least root mean square was then identified.

[2] http://www.cdc.gov/niosh/docs/98-131/overview.html

3. Rank Algorithm:

The algorithm looked at each individual point and ranked it based on how close it is to a given person's data. It would then tally all the rankings and the person with the lowest rank is then identified.

We found that total difference and the RMS algorithms worked consistently with good results while there was periodic error in the rank algorithm.

Matlab Graphical User Interface:

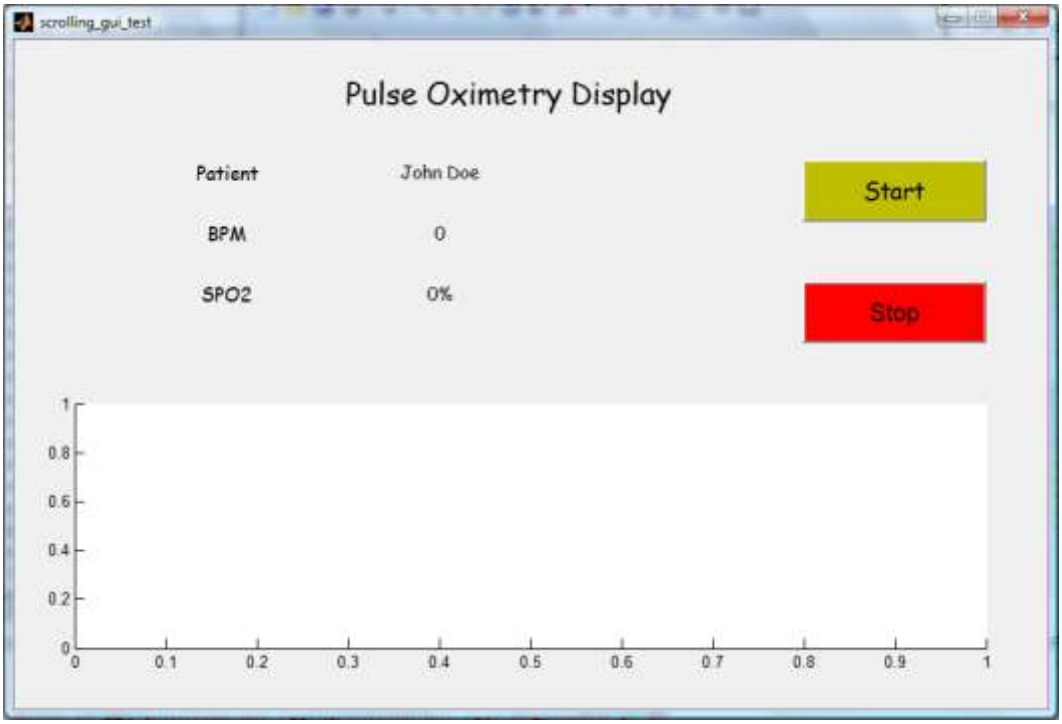

Figure 12: Matlab GUI

We created a Matlab GUI to display patient name, BPM and SPO2 levels. The GUI connects to the PSOC via XBee and receives the data in a serial fashion. The axes autoscroll and automatically zoom in on the data. Figure 12 shows the skeleton of the GUI.

Results and Discussion

The results that we obtained were quite encouraging. The values that we measured with our system matched the reading from other off-the-shelf pulse oxymetry systems fairly well most of the time. We believe we were certainly on the right track. Our system does require fine-tuning. The system seemed to be very sensitive to patient's motions. The only way we

were able to get reasonable data was by making sure that the user was perfectly still. This is not convenient if the project is to ever develop into a competitive product.

Our system was also limited by the fact that we only had the off-the-shelf coupled red and IR LEDs to work with. This meant that we could only use them in a limited fashion. One scheme that would have been interesting to apply would be to use some sort of frequency modulation scheme to light up the LEDS and then use lock in amplification system to collect that data. This scheme would effectively allow us to only monitor the frequency that we are modulating at, thus reducing the noise and improving our signal. The accuracy of our results could have also been improved if we had known the exact characterization of the LED. This would have allowed us to find more accurate absorption coefficients and constants for the SpO2 calculations.

Using the AD5933 and the front end analog circuit, we were able to identify different impedances starting with resistors between 10-100 kΩ, complex impedances of capacitors and resistors in parallel and finally distinguish between two human beings. We used frequencies between 1-10 kHz for sweeping human tissue as these signals were larger than those at higher frequencies which were difficult to distinguish from noise. Figure 14 shows 3 different sweeps done for Amine and Yahia with the electrodes fixed in the same place.

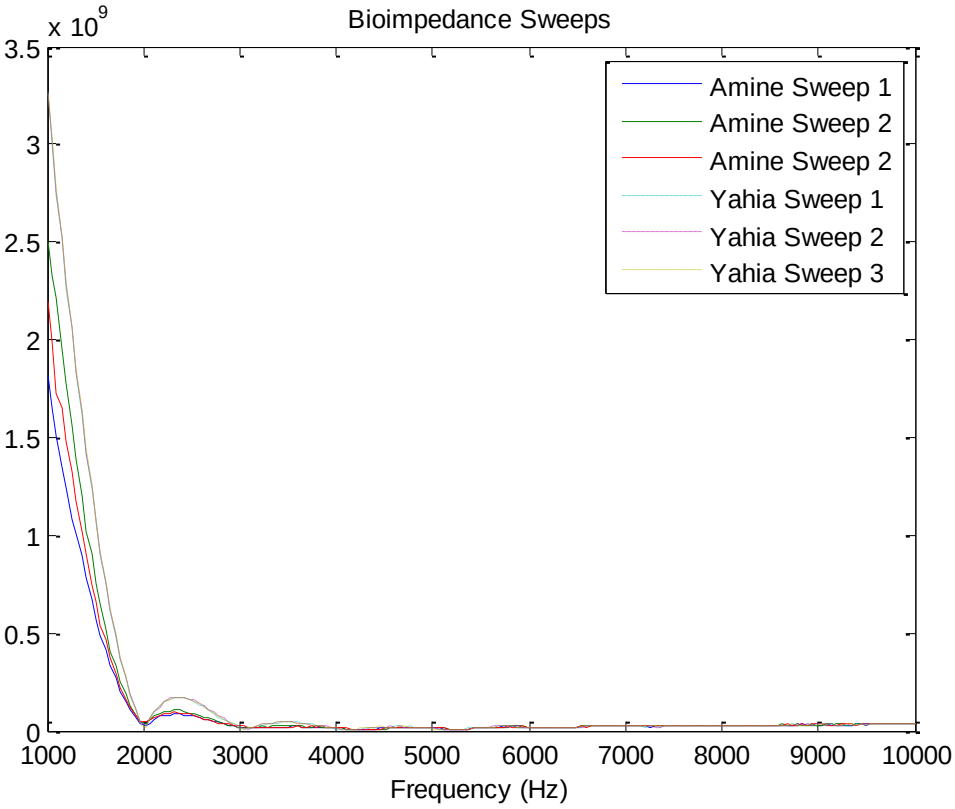

Figure 13: Bioimpedance Sweeps

One can notice that the majority of the interesting information lies between 2-4.5kHz. Figure 14 zooms in on that area of interest. It is clear to the human eye that the sets of data of the two subjects under test vary in amplitude between 2-3 kHz which can be used for identification.

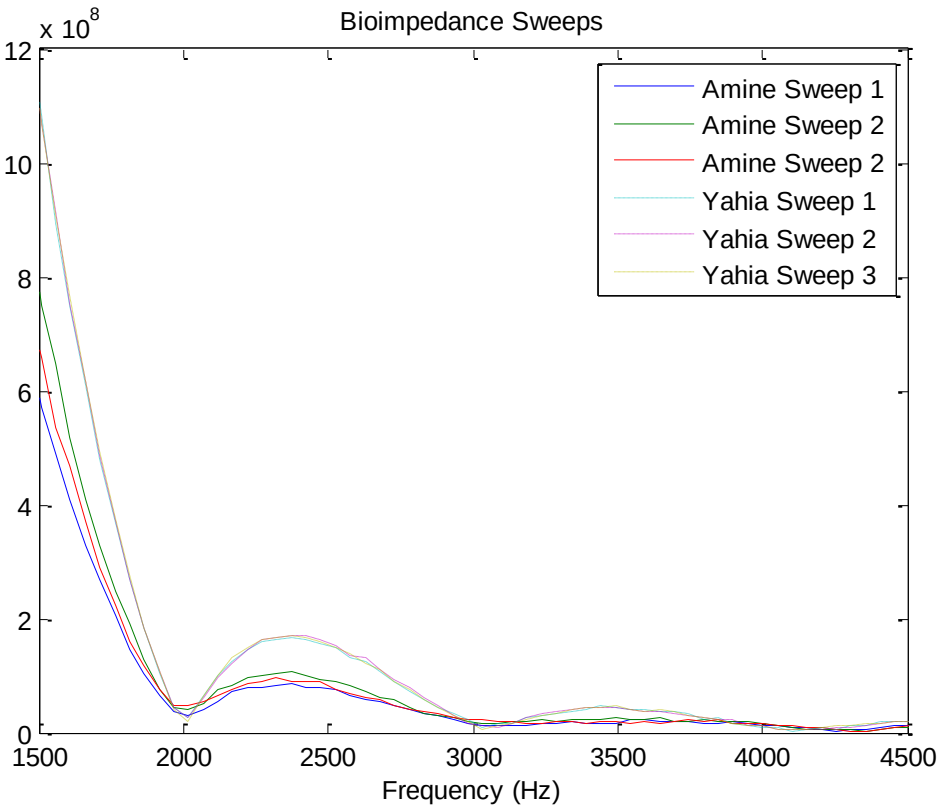

Figure 14: Bioimpedance sweeps

From the bio-impedance we were able to distinguish at least 2 people. Their impedences were clearly different and the algorithms could separate them apart. However, like with the SpO2 measurements, there is room for improvement. The AD5933 is convenient in that it provides a nice sinusoidal wave to use for the impedance calculation and also does all the calculation and data collection, however this can be a drawback. The data did not look like what we expected at low frequencies. It seems like there was a super imposed sinc function on the data even when measuring a simple resistor. This was somewhat disconcerting and warrants further investigation.

The resolution on our bio-impedance data was smaller than we would have wanted especially at higher frequencies. This was because we were trying to measure a wide range of values. We believe that the body was acting as a filter whose corner frequency was somewhere in the range of frequencies that we were using. This meant we could not raise the gain too high because we would clip the signal in the pass-band while a low gain meant a low signal to noise ratio for signals beyond that corner frequency. One idea that has come up to remedy this was to use some combination or resistor and capacitor in series or in parallel to the patient so as to move the corner frequency outside the range of interest. Another idea was to implement some

sort of variable gain to maximize the signal at all levels. Both ideas definitely warrant more investigation.

Non-prerequisite skills used

The most useful skill not covered in the prerequisite courses (ENGS 31 and 61) is familiarity with using and programming microcontrollers (ENGS 62). This includes communication protocols such as UART and XBee. Familiarity with using MATLAB for serial communication could also be helpful.