Final Report

Muhammad Athar, Jason Huang

I. Introduction

The goal of our integrated design project is to be able to measure the percentage of SpO2. SpO2 is the percentage of oxygen in a person's blood out of the maximum oxygen that a person's blood can carry. Figuring out a person's SpO2 value can help explain conditions such as sleep apnea, asthma, or Covid-19. Our ultimate goal is to be able to design a prototype circuit that would take an input signal from a SpO2 sensor and be able to process it to be able to display the heart rate signal, beats per minute of the heart (BPM), and the SpO2 percentage. The signal is processed with filtration and gain. Filtration allows us to obtain a clean signal free from noise such as a 60Hz hum from the computer and other possible unwanted frequencies. Gaining the signal allows us and the Arduino to compute these values. Utilizing our knowledge of filters and other circuit components allowed us to successfully design a prototype that met the specifications.

II. High-Level Design

The block diagram below is a high-level design of the entire system.

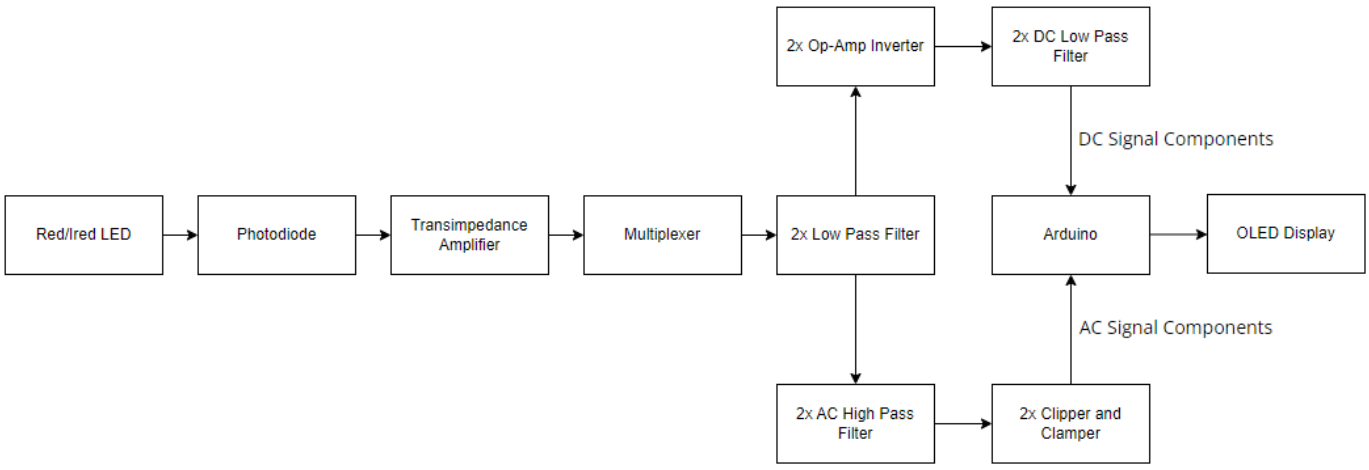


Figure 1: High-Level Block Diagram

Given Circuit Elements (LED/Photodiode)

We wanted to include these elements at the start of our high-level design because we believe that these components are key elements that drive what the circuit outputs. Although the LEDs and photodiode both come from the same SpO2 sensor, they technically operate with two separate circuits.

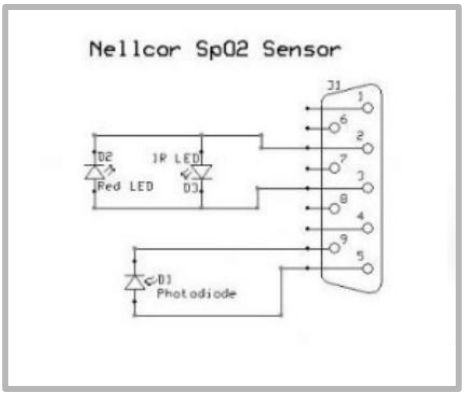


Figure 2: Nellcor Ds-100A SpO2 Sensor Pinout

As shown in the pinout of the SpO2 sensor, the two LEDs form their own section with pins 2 and 3 while the photodiode forms another section with pins 9 and 5. The relationship between these two circuits is that one of the LEDs will be forward-biased, turning it on and shining light. There is not a possible voltage setup that allows both LEDs to be turned on. This means in order to obtain both the red and ired signals, we had to implement voltage switching. This light will then be picked up by the reverse-biased photodiode which will allow a certain amount of current to flow depending on how much light is being shined into the photodiode. To get the photodiode reverse biased, a positive voltage will be applied to pin 9. Pin 5 will then be connected to the input of the circuit. This is the baseline of how the SpO2 sensor will produce a current that will be fed as an input to our circuit.

Transimpedance Amplifier

As explained in the section above, the photodiode will output a certain current depending on the light being shined into the photodiode. This means the input to the circuit will be a current source. We want to be able to work with a voltage signal instead. This is where the transimpedance amplifier helps. The transimpedance amplifier takes a current source and outputs the equal voltage value.

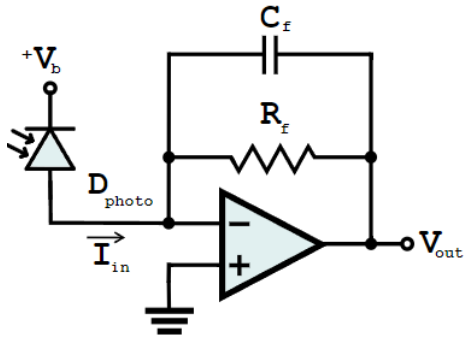


Figure 3: Transimpedance Amplifier

This picture from the Wikipedia page for transimpedance amplifier has a perfect schematic of what we used to implement this. Along with converting current to voltage, this design also allows us to easily control the gain with the feedback resistor. The reason why we want gain on our transimpedance amplifier is that we want to spread out our gains across multiple circuit schematics. By spreading the gain out, the gain for each schematic will not have to be as big which could result in some large circuit components like capacitors or resistors which we want to limit since we are building on a breadboard. The only possible downside to using the transimpedance amplifier would be that it inverts the signal. This in turn is not a real issue because obtaining the AC components is not reliant on whether the signal is flipper or not. The DC portion on the other hand is negative which can cause problems when reading it from the arduino. This can easily be fixed with another circuit block as discussed in a later section.

Multiplexer

The multiplexer is arguably one of the more complex blocks of our diagram and one of the most crucial pieces to allow us to obtain SpO2 in real-time. The reason why the multiplexer chip is important is because it allows us to be able to obtain the red and infrared signals at the same time. In reality, the multiplexer chip actually is 3 2-1 multiplexers. In our case, we only used two of those multiplexers because we only needed two branches, one for each signal. For the inputs of each multiplexer, one input was the signal from the transimpedance amplifier, and the other input was grounded. The reason why we want one of the inputs to be ground is because of the switching process. Since only one LED is one at a time, one of the multiplexers will output the signal. The other multiplexer should be controlled which in this case is ground because we don't have to be feeding the rest of the circuit a transient voltage value which could cause problems with calculations. By setting one of the inputs to ground, this eliminates that problem.

The design decision to have two separate branches rather than one branch came down to how we wanted to handle our output. By having two branches, we know which branch corresponds to what signal

and we can debug problems more easily. It becomes much harder to debug problems with one branch because the switching process happens very quickly and it can become confusing. Having two branches also allows us to develop the Arduino code which is not time reliant when measuring the signal. Having only one branch means the Arduino code must be able to parse which signal is what at a very high rate. This can cause problems for the Arduino because the operation is time sensitive and there are other operations the Arduino is doing which can cause the synchronization of the signal reading to go out of sync.

General Low-Pass Filter

Right after the multiplexer comes the general low-pass filters. There is one low-pass filter for each branch. The purpose of this low-pass filter is to pass the frequencies of the heartbeat while attenuating frequencies that can cause noise in our output. The majority of the noise that we want to attenuate comes from the 60Hz hum from the laptop as well as some 80Hz noise that comes from when the laptop is charging. Since the heartbeat frequency range (0.67-2Hz) that we want to pass is far from the 60 and 80Hz noise we want to attenuate, the filter does not have to be incredibly precise. Having more flexibility in the filter allows us to use less and a wider variety of parts. The filter we designed with analog filter wizard is a second-order filter that utilizes resistors, an op-amp, and two capacitors.

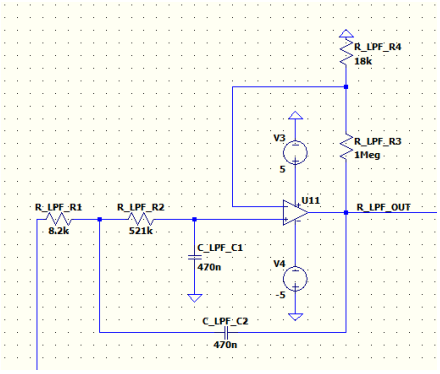


Figure 4: General Low-Pass Filter

This filter also has some gain to help increase the signal amplitude. As mentioned above, this gain is to help with keeping the part sizes small as well as spreading our gains across multiple blocks. Since we want the main signal to be gained up, that means the attenuated portion is going to be at 0 dB rather than a smaller signal. This is perfectly fine because we are amplifying the signal from peak to peak in millivolts. The noise will still keep the same amplitude as it did before going through the filter, but it will be unreadable by the Arduino while the main signal is read because it got amplified.

AC High-Pass Filter

The AC high-pass filters come after the general low-pass filters. The purpose of this filter is to get rid of the DC offset that the signal has when getting read from the SpO2 sensor. The reason why we want to get rid of the DC offset is because of consistency. Everyone has a different DC offset value when using the SpO2 sensor. By getting rid of the DC offset, all signals will be centered at zero which becomes easier to process later. Getting rid of the DC offset also will remove issues that could possibly arise when amplifying the signal. If the DC offset is of a larger magnitude than the OC component, then there is a possibility that the signal could saturate. When the AC component is being amplified, the DC component also is getting amplified and we want to remove the possibility that the signal could get saturated because of this.

The specification for this filter is based on the range of frequencies of the heartbeat. Since the range of the heartbeat is between 0.67 Hz and 2 Hz, we want our cutoff frequency to be below 0.67 Hz to ensure that the lower frequencies of the heartbeat range are not getting attenuated. Along with attenuating the DC component of the signal, the filter also has some final gain to bring the signal to a peak-to-peak of roughly 3 volts. Having the signal at this amplitude range allows the Arduino to read AC amplitudes accurately and perform reading.

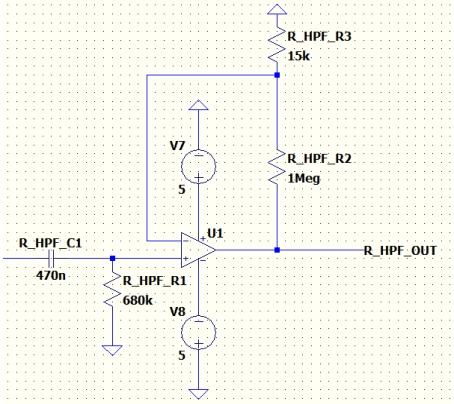


Figure 5: AC High-Pass Filter

Clipper and Clamper

The clipper and clamper circuits come after the high-pass filters. Although the signal is amplified, the range it is operating at is unusable for the Arduino. The Arduino ADC operates in a range of 0-5 volts with a 10-bit resolution. Since the DC offset was removed from the previous block, the signal is centered at 0 volts. This means part of the signal is negative and if we were to feed this signal into the Arduino, it could possibly damage the pins and the ADC. The clipper fixes as it can shift the entire signal so the entire signal is above 0 volts.

The other issue is the other side of the range of the ADC being 5 volts. Sometimes when a finger is removed and reinserted into the SpO2 sensor, it causes a large spike in voltage. This large spike in voltage can go way above the 5-volt range which can definitely damage the Arduino. To fix this issue, we implemented a clamper that would cap the voltage at 5 volts no matter what happens.

After passing through both the clipper and clamper, we are able to input the signal that corresponds to the AC components into the Arduino. Having both the clipper and clamper allows us to control the voltage range and ensure that the Arduino does not get damaged.

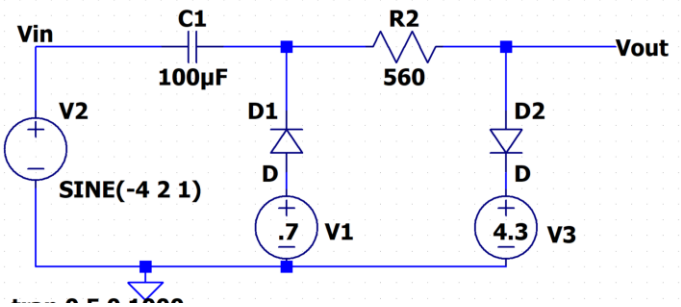


Figure 6: Clipper and Clamper

Op-Amp Inverter

The op-amp inverter is the first block that processes the signal for the DC component. This block also comes right after the general low-pass filter. The reason why we had it branching from the low-pass filter rather than the multiplexer is because the low-pass filter already filters out the 60 and 80Hz noise as well as already including some gain we can use for the DC components as well. The role of the op-amp inverter is to change the sign of the DC offset coming off the low-pass filter. When checking the DC offset, we noticed that the signals were centered at negative voltages which is problematic for the Arduino. The reason why we don't want to implement a clipper circuit to bring this offset above zero is because the clipper will destroy the value of the DC offset. We want to keep the magnitude while changing the sign and the inverting op-amp is perfect for that. Although now we have positive DC components, the ratio is hard to read because the signals are still too small for the Arduino to get a ratio. We incorporated some gain in the inverting op-amp to make the ratio more readable for the Arduino.

DC Low-Pass Filter

We implemented another set of low-pass filters after the inverting op-amps. The reason for this decision was because we wanted to attenuate the AC components of the signal. The first general low-pass filter only filters out the 60 and 80Hz noise while keeping the heartbeat signal. For the DC branch, we don't want the AC components. We designed a new low-pass filter with different specifications to filter out the AC frequency range for the heartbeat

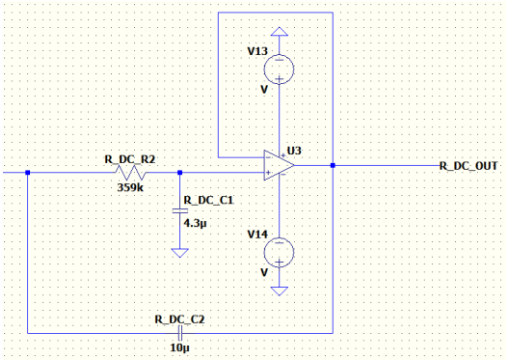


Figure 7: DC Low-Pass Filter

This filter also has some gain implemented to further improve the reading of the ratio for the Arduino. After passing the signal through these filters, they can then be read into the Arduino as the DC components of the signal.

Arduino and OLED Display

The Arduino and the OLED display are the final blocks of the high-level system. The circuit has four outputs:

- Red AC
- Infrared AC
- Red DC
- Infrared DC

With these four values being read into the Arduino, the Arduino then can compute the 'R' value which then can be used to compute SpO2. Along with the SpO2, the Arduino also can compute the frequency of the signals. We implemented the frequency by computing peak-to-peak measurements. The reason for this design choice is because the peak is one of the easier parts of the signal to find when looking at a general signal for the heart. By computing the frequency, we can compute the beats per minute of the heart. The final project specification is to display the heart signal waveform. This is simple with our current setup. Since the Red AC signal is being constantly read, all we had to do was sample the signal and plot the points on the display to show an accurate waveform. We decided on a sampling period of 20 milliseconds for multiple reasons. The first reason is because the 20 ms is accurate enough to capture the shape of the signal. The second reason is because the sampling rate is also the rate at which the signal gets plotted on the display. Having the display plot every 20 milliseconds allows the scrolling speed to allow multiple periods of the signal to be shown on the display.

The Overall System

To recap the entire system, the LEDs first shine a light into the reverse-biased photodiode which will allow current to flow into the input of our circuit. The transimpedance amplifier will take this current signal and output an equivalent voltage signal. The transimpedance output will go into 2 2-1 multiplexers which will split the circuit up into a red signal branch and an infrared signal branch. The signal then will pass through a general low-pass filter to filter out any unwanted noise. After this point, the circuit gets split again for AC and DC components resulting in four different branches. To obtain the AC components, the signal passes through an AC high-pass filter that removes the DC offset and centers the signal at 0 volts. Then the signal passes through a clipper and clamper to bind the voltage range between 0-5V for the Arduino. For the DC components, the signal passes through an inverting op-amp to get the signal to a positive value. Then the signal gets passed through the DC low-pass filter to filter out the AC components. Then the signals are then able to be passed into the Arduino. The Arduino will take these four values and calculate the SpO2, BPM, and heart waveform and display all this information on the OLED display.

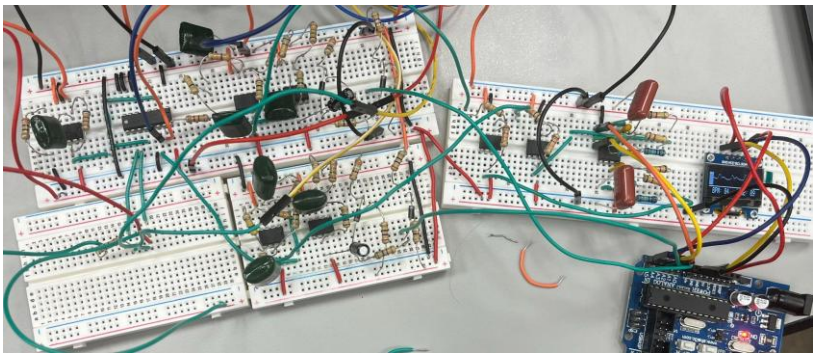


Figure 8: Physical Overall Circuit

III. Detailed Design Verification

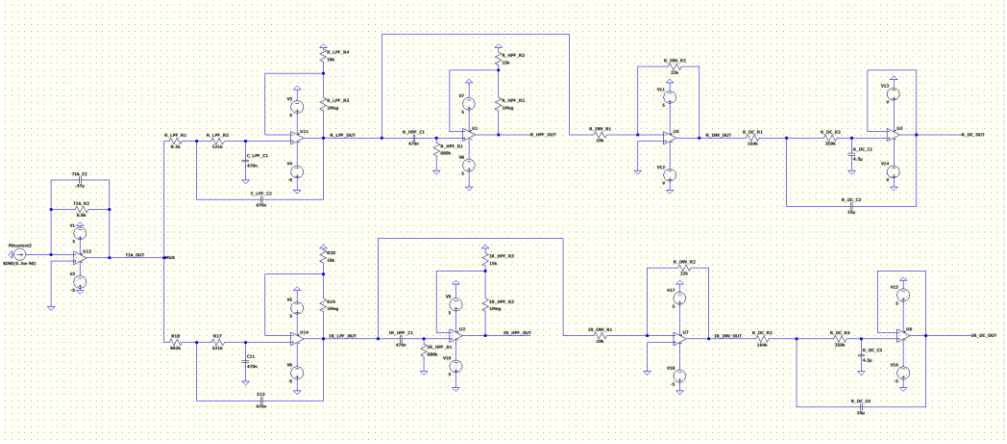


Figure 9: LTspice Schematic for Overall Circuit

As mentioned earlier, the transimpedance amplifier is responsible for converting the current produced by the photodiode into a voltage that can be further manipulated. One can think of a current-to-voltage converter as something as simple as a Norton Circuit. However, the transimpedance amplifier's input voltage bias is constant because of the op-amp configuration, while the photodiode's current is fluctuating. This is beneficial in maintaining a smooth conversion. This subsystem features a capacitor that is in parallel with a feedback resistor. The purpose of this capacitor is to take care of any parasitic capacitance that is located at the current's input terminal on the Op-amp. So, the value for this capacitor wasn't necessarily required to be some specific value. On the other hand, the feedback resistor is responsible for the actual voltage conversion. This is backed by Ohm's law, for V = IR. So, this feedback converts the incoming current into a voltage, while also gaining it by R_f. This feedback resistance is directly involved with the signal's magnitude. We tested with 6.8k ohms, but we decided to actually finalize this resistance value *after* constructing the AC filters. In the end, the resistance value we chose was, in fact, 6.8k. The transimpedance amplifier's output can be seen below.

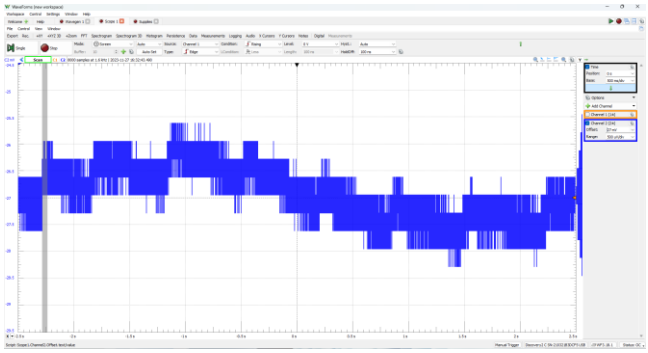


Figure 10: Transimpedance Amplifier Output

The following subsystem is an analog multiplexer IC. Therefore, its internals and components are optimized by the manufacturer, Texas Instruments. However, it has several connections that allow it to operate correctly, in that it allows a high-frequency switching behavior. Two of the select bits for the MUX were alternating. To accomplish this pins 15 and 14 are hooked up to their own inputs. These inputs are a 1) 500Hz and 5V square wave and 2) 500Hz, 5V, and 90° square wave, respectively. +5V is connected to pin 16, ground is connected to pins 6, 7, 8, 1, and 13, and the outputs of the TIA are connected to pins 2 and 12. Lastly, the outputs of the MUX are located at pins 14 and 15, where pin 14 is responsible for the red LED subsystems, and pin 15 is responsible for the infrared LED subsystems. This configuration allows for seamless switching at a high frequency to output both signals, while maintaining their shape. The truth table states that the outputs of A and B are only controlled by their respective

inputs, so alternating square waves at the inputs was definitely a good decision. In essence, when A is on, B is off, and when A is off, B is on. The MUX and its truth table is shown below:

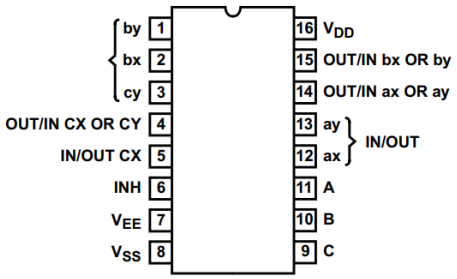


Figure 11: CD4053BE Analog Multiplexer Pinout

INPUT STATES				ON CHANNEL(S)
INHIBIT	C	B	A	
CD4051B				
0	0	0	0	0
0	0	0	1	1
0	0	1	0	2
0	0	1	1	3
0	1	0	0	4
0	1	0	1	5
0	1	1	0	6
0	1	1	1	7
1	X	X	X	None
CD4052B				
0		0	0	0x, 0y
0		0	1	1x, 1y
0		1	0	2x, 2y
0		1	1	3x, 3y
1		X	X	None
CD4053B				
0	X	X	0	ax
0	X	X	1	ay
0	X	0	X	bx
0	X	1	X	by
0	0	X	X	cx
0	1	X	X	cy
1	X	X	X	None

Figure 12: CD4053BE Analog Multiplexer Truth Table

The outputs of the MUX are inputs for an AC filtration system that is required for the signal to be read by a program.

The leading subsystem in this filtration system is a low-pass filter that is responsible for eliminating noise from the signal. The components of this filter were determined by Analog Filter Wizard. Before these components of this filter are set, the specifications for this filter must be exact. This filter's specifications are as follows: Gain = 50; Cutoff frequency = 5Hz; Stopband at 60Hz (-40dB). These values are extremely important for several reasons. Firstly, when measuring the signals' magnitude after the trans-impedance amplifier, it was noted that a 300x amplification was required to achieve a peak-to-peak voltage of approximately 2V. So, we decided it was best to split this gain in this AC filtration system. A gain of 50 on this filter would mean a gain of 60 would be the final amplification factor required to achieve the targeted peak-to-peak. The cutoff frequency of 5 Hz is perfect, for the range of frequencies required to be kept was $0.67\text{Hz} < f_B < 2\text{Hz}$. These frequencies refer to the BPM values of 40-120, which cover a vast range of individuals and scenarios. The 5Hz cutoff, however, is set because of the harmonics that are included in the signal. These harmonic frequencies are part of the signal(s), and so we do not want to eliminate them. They exist at frequencies below 5Hz, as seen by the Spectrum below:

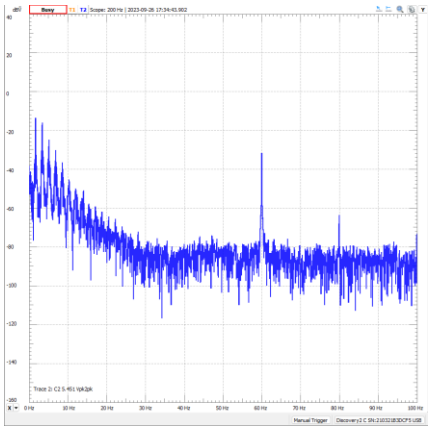


Figure 13: Spectrum before AC Filtration

After eliminating the unnecessary frequencies from the signal, there needed to be a way in which the Arduino could handle the voltage fluctuations coming from measuring the heartbeat signals. So, a high-pass filter is the second part of this AC Filtration system. This high-pass filter would reduce the DC offset of the signal to 0V. The components of this filter were also determined by Analog Filter Wizard, which uses the user's specification requirements to construct a circuit. The specifications for this circuit were as follows: Gain = 60, Cutoff frequency = 0.5Hz, Stopband at 0.1Hz (-30dB). As noted earlier, the amplification required for a readable signal after the TIA (transimpedance amplifier) would have to be 300x. So, assuming a gain of 50 from the LPF (low-pass filter), a gain of 60 was needed for this HPF (high-pass filter). Thus, the Arduino readability requirement was met. The 0.5Hz cutoff frequency is the absolute maximum possible because the majority of the AC signal is located after 0.67 Hz. The stopband at 0.1Hz is also optimized, for the signal at -30dB is attenuated significantly. At 0Hz, there is 0 gain on the signal, thus eliminating any DC offset. The spectrum for the overall system *after* the AC filtration is shown below:

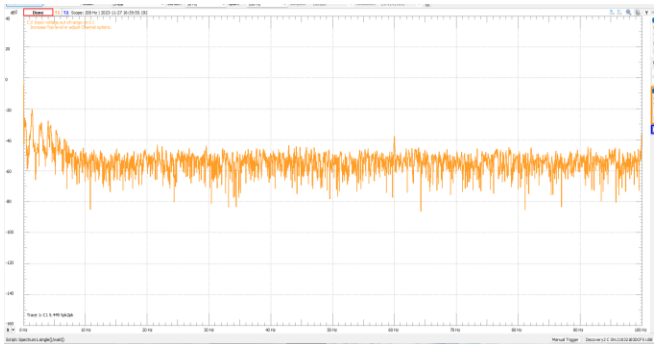


Figure 14: Spectrum after AC Filtration

Considering the fact that the HPF centers the signal at 0V, there are still some adjustments that need to be made to the signal for it to be read by the Arduino. As noted in Section II, the signal needs to fall in the range of 0-5V, or else the Arduino will flatline, so a clipper and clamper circuit is responsible

for making these adjustments. It is created by using several different components, such as capacitors, resistors, and integrated voltage sources. A capacitor is placed in series with the input voltage to ensure the signal does not lose any of its AC components, while also resetting it at 0V to allow for easier shifting/manipulating of signal. The first branch on this circuit is responsible for clamping the circuit. The reverse-biased diode allows the negative peak of the incoming signal to be clamped at the voltage of the battery. In this case, the battery can be swapped with a zener diode. The second branch is responsible for clipping the circuit at 5V. This is done by simply summing the voltage drop across the branch, which is 0.7V (forward-biased diode) + 4.3V battery. Thus, the final output of the circuit is a signal that falls between 0V and 5V, ensuring the Arduino is safe.

The AC filtration system is necessary to obtain the AC components for both signals. However, the DC components for each signal are also needed to calculate 'R'. The DC filters are preceded by inverters to ensure the signal is positive, and thus readable by the Arduino. The components for this inverter must be able to gain the signal to a reasonable voltage (<1V), for the inputs to these inverters are the low-pass filters designed in the AC filtration system. The signal must be gained approximately 2x, so the resistor values we chose were 10k and 22k, where the 22k ohm resistors were the feedback resistors. The DC filter's specifications would also be necessary to determine the values of the components, which were designed by Analog Filter Wizard. The specifications for the low-pass DC filters were as follows: Gain = 0V; Cutoff frequency = 100mHz; Stopband at 600mHz (-30dB). These specifications are optimized to reduce any signal above 0.1Hz, while effectively eliminating the AC part of the signal, which is > 670mHz. Since DC exists at 0Hz, the output was effectively a horizontal line. There is a large difference in the construction of these circuits, however. Each subsystem mentioned before the DC filters were using LF-356 op-amps. This subsystem was built using an LM358N op-amp. This op-amp, despite being different, actually has one huge benefit. It has two inputs and two outputs, so we were able to build the red and infrared circuits on the same op-amp. This was able to save us a great deal of time and breadboard space, while also making it easier to debug, if necessary.

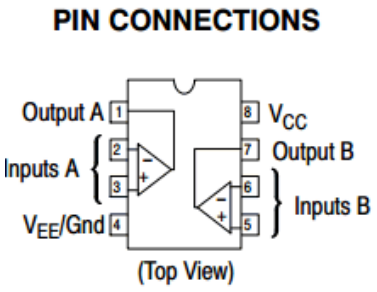


Figure 15: LM358N Pinout

Arduino Code

The Arduino code is the final step to complete the full system. The Arduino interacts with the circuit by taking the four inputs and using those values. The four outputs of the circuit are connected to the following analog pins

- AC Red: A0
- AC Infrared: A1
- DC Red: A2
- DC Infrared A3

The rest of the analog inputs (A4 and A5) are connected to the OLED display. In the Arduino code, the main signal that is being constantly read is the AC red signal. Using an FSM model, the Arduino code tries to find the peaks of the signal and notes the times and calculates the period of the signal. When the Arduino code finds a peak. The AC infrared signal, DC red signal, and DC infrared signal are all noted down as well. These values will then be used to calculate the 'R' value which then will be used to calculate the SpO2 using the formula provided.

To actually display this information. The code has interval timers to update the screen every once in a while. For instance, the code updates the SpO2 and frequency values every 2 seconds on the board. The code samples and traces the waveform every 20 milliseconds.

IV. Validation of the Overall Project

Overall, I believe that the project was a success. To reiterate, this project required that we display the heartbeat waveform, BPM, and % SpO2 of an individual on an OLED display. This is achieved by the use of a Nellcor Ds-100A SpO2 sensor, DB9 breakout board, OLED display, and CD4053BE analog multiplexer IC. In the end, we were able to do all of the above, albeit the SpO2 values, on average, fluctuate between 80-100. The BPM values calculated are *mostly* correct, for the incoming signal can "trick" the code.

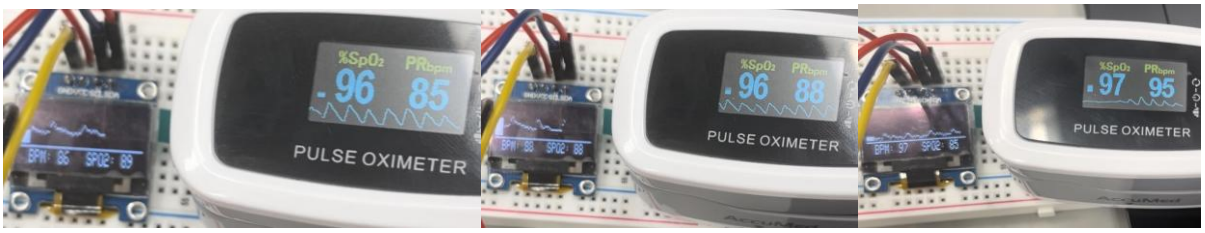
The SpO2 error stems from the fact that the infrared signal's DC component is *smaller* than the red signal's DC component. This is a problem because of the nature of the 'R' equation. This equation, shown in Figure 16, suggests that each AC component is also dependent on each DC component. After extensive testing, we were able to conclude that our AC ratio of ~0.75 was correct. So, in order to obtain an 'R' that would satisfy the requirement of a SpO2 value of 95-100%, the DC ratio would have to be between 0.533 and 0.8. This is also under the assumption that the equation for SpO2 is "%SpO2=110-25R". In our case, since the DC value of the infrared signal was larger than the red signal, our DC value was mostly > 1. This meant that our % SpO2 wasn't always 100% accurate.

The BPM calculator is mostly perfect, however, sometimes it is wrong. Both signals possessed a smaller “hump” after the beat was detected. Based on the formula we use to calculate the BPM, as well as the sampling rate we used, sometimes this BPM can be higher or lower by some small margin, usually within 10% of the correct value. This is because the code may detect that “hump” as one of the peaks it needs to compute % SpO2, rather than the correct, much larger peak. Once again, this is quite rare.

$$R = \frac{\frac{AC_{red}}{DC_{red}}}{\frac{AC_{ired}}{DC_{ired}}}$$

Figure 16: R Equation

Checking the correctness of our final product was done by comparing it to an Apple Watch SE, as well as a commercial SpO2 sensor. When comparing the BPM values of our sensor to the Apple Watch, we found that the heart rate was consistent between both devices. Most of our testing was done with the commercial sensor, so pictures are provided below this section. We found that our OLED looked similar to the commercial sensor in terms of format, but more importantly, the values across both devices were relatively consistent. Excluding the outlier values, the Nellcor Ds-100A sensor was nearly perfect across the tests conducted.



V. Conclusion

After finishing this project, we have learned a great deal about circuits, teamwork, and time management. Going into this class, we had very limited knowledge of op-amps and filters. After finishing this class we definitely have a higher level of mastery of designing filters that meet specifications as well as reading documentation to understand integrated circuits such as the multiplexer chip.

Teamwork is also another important lesson both of us have learned. We learned that communication is an important skill to have. Having clear communication prevents us from running into problems that are not directly related to the project itself. Another lesson we learned is that assigning roles helps us delegate tasks and reduce confusion as to who is responsible for what. Going into each milestone we had a clear understanding of each task we had to complete.

The last lesson we learned was time management. Integrated Design Project is a semester-long project with preset due dates for each assignment. While these due dates are set, we also had to set some internal deadlines. The milestones are really packed with deliverables and it is our responsibility to set up a schedule to complete the tasks on time. During the middle of the semester for milestone three, we definitely felt the pressure of our poor time management and definitely had to crunch for the last few days before milestone three was due.

If we were to start the project again with the knowledge we have, we definitely would do certain things differently. The first task we would do differently would be to do more critical research on what we have to do. We felt going into this project we did not do enough research early on to get a good grasp of what we had to do. Doing more proper research would allow us to come up with a more concrete plan and prototype. The second task we would do differently would be to set up better internal deadlines for each milestone. Even though we had deadlines set for each milestone, we did not always follow them. We want to be able to meet the deadlines on time the next time we are in a project like this. The third task would be to set up better communication. We felt that sometimes our communication with each other would cause a lack of progression. Setting up meetings or dedicating some time before our work session to discuss issues could help with understanding each other's problems.

Overall this project has been an invaluable experience for both of us. This project improved our understanding of circuits and working in a team environment. We hope to use this experience for future classes or possibly even our careers.