EE122B Final Project Report

Rex Garland, Vickram Gidwani, Minmin Hou, Paul Lavengco, Shane Leonard
Anirudh Patel, Cuthbert Sun, Mark Swerdlow, Aria Tedjarati, Bayian Yahya
School of Electrical Engineering
Stanford University
Stanford, CA 94305

***Abstract*—We developed a cheap, open-source Arduino shield for measuring four common biological signals (ECG, cardiac ICG, respiratory ICG, and PPG). While similar shields cost around $100 and use specialized components, our design costs around $30 and uses widely accessible parts to be soldered on a through-hole PCB. This makes our project especially relevant to areas where access to expensive medical equipment is slim.**

I. Introduction

The goal for the project was to build a cheap Arduino shield that could be used in conjunction with an Arduino Uno in order to provide common biopotential measurement functionality. The three main functionalities included in the design are an electrocardiogram (ECG), an impedance cardiogram (ICG), and a photoplethysmogram (PPG). The outputs of each stage were to be then relayed to a computer via a Bluetooth module, where the data would be processed and displayed on a graphical interface.

The idea behind the project was to make the shield as inexpensive as possible while using ubiquitous parts in order to allow for its use both instructionally and practically around the world, especially in developing areas. This challenge required thinking beyond the borders of basic functionality and obligated the team to think of ways to utilize simple and cheap parts in order to achieve the same functionality as sophisticated ICs found in a standard design, a very simple example being the replacement of an instrumentation amplifier IC chip (LTC1167) with an op-amp quad-pack (LF347) and some passive components wired in an instrumentation amplifier topology. In addition to designing the shield schematic with more accessible parts, the design included all through-hole components to allow easy construction for individuals who may not have had much previous exposure to soldering.

Lastly, the code implemented for the data relay, the display, and the optional post-processing was intended to be open-source, again to allow for accessibility and possibility for use in teaching purposes.

This report will explain the details regarding the design of each stage and the subcircuits contained within them as well as steps and iterations in the design process. Also included in this report is the design and thought process behind the software and the PCB layout.

II. Electrocardiogram (ECG)

The schematic of ECG circuit is shown in Figure 1. The circuit consists of an instrumentation amplifier (IA), a second gain stage, a low pass filter, and a negative feedback op-amp for right leg drive. The IA is made of three op-amps and 5% resistors. The gain resistor is divided into two resistors, and the voltage at the node in between the two gain resistors, which is the common mode signal, is fed back into an op-amp. The output of that op-amp drives the common ground (right leg drive). In this way, the common mode signal can be suppressed. The IA and right leg drive op-amp share one LF347 chip, which has four op-amps in it.

The reason for two gain stages is that DC offset from the two electrodes could be as high as 300 mV, whereas the ECG signal is typically a millivolt in amplitude. The power rails are +/-8V. If we want to gain the ECG signal by 100x to get a reasonable signal-to-noise ratio in just one stage, then the DC offset would be 30V in the worst case, which is larger the power rails. Thus, two gain stages are needed to account for the DC offset. We designed the gain of the IA to be 20, and the gain of the second gain stage to be 5. In addition, an RC high pass filter is added in between the two gain stages so that the second gain stage would not see a DC offset. After the second gain stage, a two-pole Butterworth filter in Sallen-Key topology is used to band-limit the signal to 150Hz. A level-shifting circuit made of an op-amp and resistors (not shown in the schematic) is used to shift the bipolar signal to unipolar signal (0-5V) so that the ADC on Arduino can be used to sample the ECG signal.

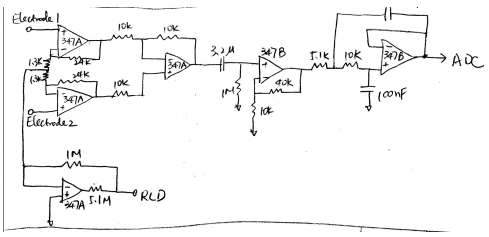

Fig. 1. ECG circuit schematic: instrumentation amplifier with right leg drive

III. Impedance Cardiogram (ICG)

The ICG circuit is composed of a 30 kHz oscillator, a voltage controlled current source, an instrumentation amplifier (IA) with DC restoration, a commutator, a low pass filter for the respiration signal, and a band pass filter for cardiac signal. The following subsections will describe the subcircuits in detail.

A. Oscillator and current source

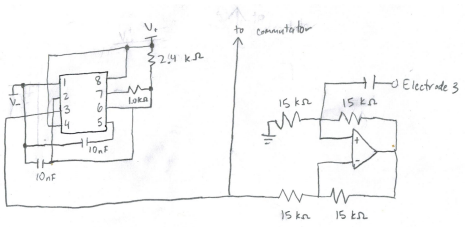

Fig. 2. Oscillator and Howland Current Source

The oscillator is made using a 555 timer. The oscillation frequency is chosen to be 30 kHz because at such a frequency the let-go current is much higher than that at 50-60 Hz so that a larger current can be injected to get a larger signal. In addition, 30 kHz is much higher than 60 Hz, which means that we can AC couple the signal to filter out the low-frequency noise without amplifying it.

We chose to create our square wave with a 555 timer because we need a rail-to-rail signal to drive the SPST switches for the commutator. We could not achieve this with a relaxation oscillator because the op-amps we are using are not rail-to-rail. c

The problems encountered with the 555 timer were 1) large overshoot at rising edges, and 2) contamination of power rails and ground with spike trains in sync with the rising and falling edges. The first problem can be minimized by optimizing the design. Between the output of the oscillator (555 timer) output and the input of the voltage controlled switches (4066) a resistor is in place. This had the effect of reducing the ringing by increasing the RC rise time, causing the rising edge to overshoot less. The second problem was not fully solved but mitigated by the inclusion of bypass capacitors. In the future, it may be useful to explore the possibilities of utilizing the Arduino in order to generate clocks rather than using on board oscillators.

The current source is a Howland current pump. The Howland current pump is used to supply a constant current output. For this application, it is crucial that the current be constant as it is meant to be input into the body in order to modulate the signal being measured. The current is limited to 1 mA for safety (below the let-go current level at 30 kHz). The Howland current pump utilizes both positive and negative feedback in order to output a steady current. By keeping the ratio of the resistors in the negative feedback circuit equal to the ratio of the resistors in the positive feedback circuit, the output current of the circuit can be kept at a constant value. For future reference, a more detailed description of the Howland current pump can be read in the TI AN-1515 application report.

B. Instrumentation amplifier (IA)

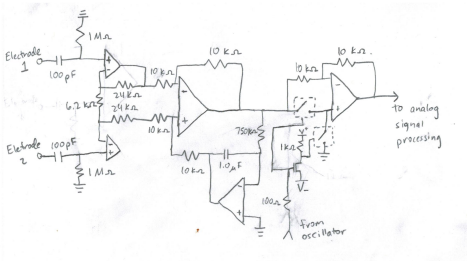

Fig. 3. Instrumentation Amplifier and Commutator

The IA design is similar with the one in ECG. Three op-amps and some 5% resistors make up the IA. The gain of the IA is designed to be 8. It was expected that the input to the IA would be a square wave, due to the modulation signal being a square wave oscillation. Originally the gain of the stage was higher, but it was found that the output of the IA lost the sharp rising and falling edges that were exhibited in the input. After studying the datasheet of the LF347 op-amps that were used to build the IA, it was found that the gain bandwidth product of the op-amp was too low to accommodate a high gain value, and was attenuating the higher harmonics of the square wave input. As a result, the IA was designed to have a smaller gain.

A DC restoration circuit is used to reduce the DC offset caused by resistor mismatch. The IA and DC restoration have four op-amps in total, which are all in one LF347 chip.

The input to the IA is a modulated 30 kHz differential signal. To eliminate the low-frequency noise such as 60 Hz noise, we added RC high pass filters (cutoff at about 1.6 kHz) at the two inputs to the IA.

C. Commutator

In order to demodulate the signal received from the body, the signal must be multiplied with the original modulation signal, therefore we simply use the same oscillator that was used to feed current into the body. In designing the ICG circuit, we found that a typical multiplying IC, such as the AD633, was too costly for the purposes of the project. Instead, a simple two switch commutator was implemented. The square wave oscillator output was used to drive two single pole single throw (SPST) switches. We implemented a simple single transistor inverter to drive the switches in 180 degree phase relative to each other. These two SPST switches act as an SPDT switch.

The switches were used to control the signal to the non-inverting input of an op-amp. The combination of the op-amp circuit and the switches replicated the effect of multiplying specifically by a square wave; this is appropriate as the original modulating signal was a square wave as well. In one position, the switches connected the non-inverting input to ground, causing the op-amp to function as an inverting amplifier. In the other position, the switches connected the output to the negative terminal of the op-amp. This eliminates any current flow through the resistors and causes the configuration to act as a voltage follower.

One problem that was seen in implementing the commutator was an issue in the output of the inverter. Originally, the drain of the transistor was connected to the positive supply voltage by a 100 kilohm resistor. As a result the output of the inverter had a slow rising edge. It was realized that internal capacitance in the transistor may be causing a slow rise time at the drain. To solve the issue, the drain resistor value was reduced in order to achieve a sharper rising edge.

D. Filters

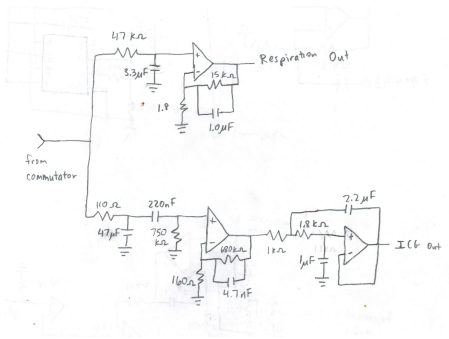

Fig. 4. Analog Filters for Respiration and ICG Signals

The bandwidth of the respiration impedance signal is 0.01 Hz - 1 Hz. Thus, an RC low pass filter with a cutoff frequency of 1 Hz is used for the respiration signal. The impedance of the chest is about 50-80 ohms. The change of the impedance due to respiration is about 50-200 milliohms. When a 1 mA peak-to-peak current is used to measure the impedance, the voltage is about 50-80 millivolts peak-to-peak. A frequency-limited (10 Hz) gain stage (8x) is added after the filter to amplify the signal (total gain of about 64).

The bandwidth of the cardiac impedance signal is 1 Hz - 30 Hz. A bandpass filter made of resistors and capacitors were used. After the band pass filter, a frequency-limited (50 Hz) gain stage is added to amplify the signal. The impedance change due to cardiac activities is on the order of 50-200 milliohmes. When using a 1 mA peak-to-peak current, the change of voltage due to cardiac activities is about 50-200 microvolts. We would need a total gain of about 10000. Then a two-pole Sallen-Key Butterworth low pass filter is used for anti-aliasing filtering (cutoff frequency is 80 Hz).

When debugging the filter and second gain stage for cardiac impedance signal, we found that the gain for the second stage that we originally chosen (4000) was too large as the signal that we were reading was the power rail and did not change with change of impedance. So we reduced the gain of the second gain stage.

IV. PhotoPlethysmogram (PPG)

A. Pulse Oximetry

We initially had wanted to include pulse oximetry functionality into our PCB board. We even got so far as to have a full theoretical design laid out, before deciding to scrap the project due to a combination of factors including board space, price considerations, and insufficient time to prototype and implement such a complex circuit. Regardless, here is the theoretical approach we wanted to take.

The first key difference between the pulse oximeter and the standard PPG is that the pulse oximeter uses both red diodes and IR diodes (in order to look at the absorption of different wavelengths of light, giving us the ratio of oxygenated to deoxygenated hemoglobin). The second key difference is that we need to record both the DC and AC waveforms for each red and IR output signals (used to properly calculate the blood oxygen level).

One major design challenge we encountered was to do the above, but to also incorporate the ambient noise rejection that we had implemented in the original lab. To do this, we needed a 2x faster clock, toggling the switches for a sample and hold circuit.

As we will see shortly, the success of the circuit entirely hinges on keeping the various clocks in phase and cooperative. Our sample and hold circuits can be simply implemented using two SPST switches running on respectively inverted clocks, with capacitors going to ground to hold the signal when the switches are open.

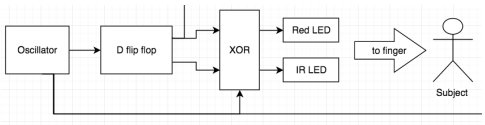

Fig. 5. Stage 1 is clock generation and manipulation

The first stage is responsible for clock generation and manipulation, then sending the appropriate clocks to the LEDs which will then pass light into the finger (to be picked up in the next stage). If we assume our oscillator outputs a square wave at some arbitrary frequency (not actually super critical what this frequency is, as long as its greater than a few hundred hertz for resolution), say 2m Hz, then the d flip flops two outputs, Q and Q will be inverses of each other, operating at m Hz.

If we then pass these two m Hz, inverted, clocks into the XOR gate where they are XORd with the original 2m Hz signal from the oscillator (not XORd with each other), the result will be two m Hz clocks running at a 25% duty cycle, where the two outputs of the XOR are 180o phase shifted. These 25% duty cycle clocks are critical later, when we are trying to do ambient noise rejection.

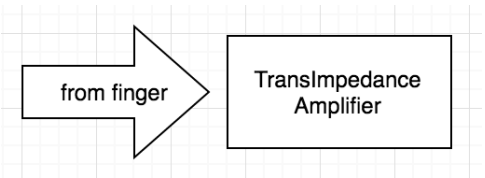

Fig. 6. Stage 2 is the trans-impedance amplifier

Stage 2 is fairly straightforward, composed of a trans-impedance amplifier that converts the current signal from the

photodiode into a voltage signal. After some experimentation, we concluded the RC values were reasonable to keep in our design.

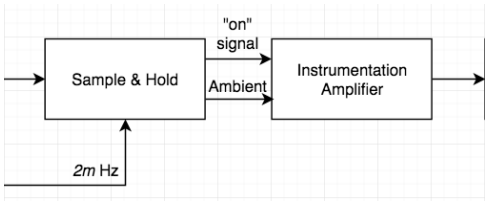

Fig. 7. Stage 3 handles ambient noise rejection

The output of the trans-impedance amplifier gets handed to Stage 3, shown in Fig 7. At this point, the signal is a mix of all of our signals (the ambient, the Red LEDs signal, and the IR LEDs signal). This smorgasbord of signals is fed into our first sample and hold stage, which is toggled by the 2m Hz clock. The on period of the clock will sample the LED on signal (takes both the Red and IR LEDs signal) and the off period will sample the LED off signal, or the ambient noise.

From here, we pass these two signals LED on and LED off into an instrumentation amplifier which will subtract out the ambient noise.

The output of the instrumentation amplifier will be a single signal, that includes both the Red LEDs and IR LEDs signal in alternating m Hz periods.

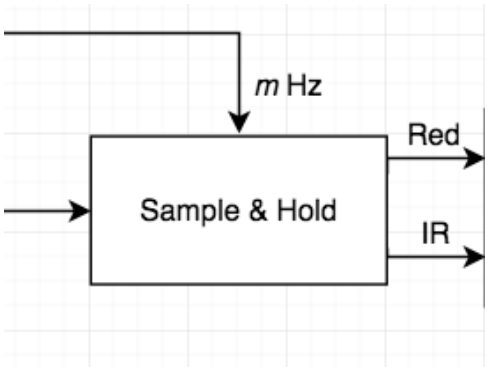

Fig. 8. Stage 4 uses a sample and hold to separate the Red and IR signals.

Now that weve successfully subtracted away the ambient noise, we need to separate the Red and IR LED signals. Luckily, we can use a simple sample and hold circuit, toggled by a m Hz clock (taken from the q output of the flip flop). This way, the high period of the clock will give us the Red LED signal, while the low period of the clock will give us the IR LED signal. The sample and hold will demultiplex our signal, and allow us to perform signal conditioning on each signal separately.

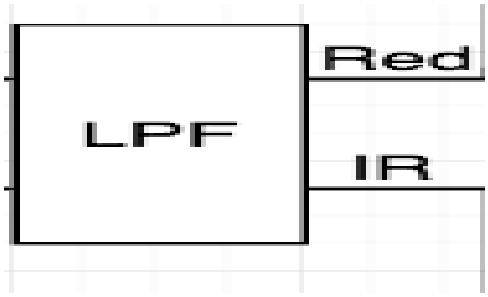

Fig. 9. Stage 5 consists of just a LPF to help filter out any residual clock noise we may have, or other high frequency, unwanted, noise.

Stage 5 is simply a LPF to filter out unwanted high frequency noise, which includes any residual clock noise and noise from the switches.

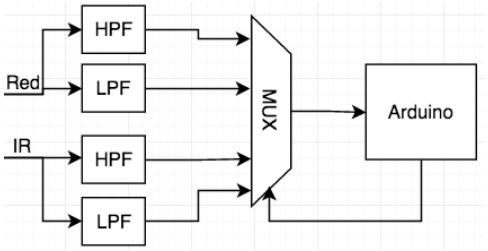

Fig. 10. Stage 6 separates the AC from the DC using filters, then passes them into the Arduino

Stage 6 is our final stage. At this point, we have our Red and IR LED signals, but need to separate them into the AC and DC components. To do so, we utilize filters, with poles placed at 0.1 Hz. Once theyre separated, they are stored in a multiplexer to the Arduino, where the Arduino controls the select pin to control which signal it wants to sample.

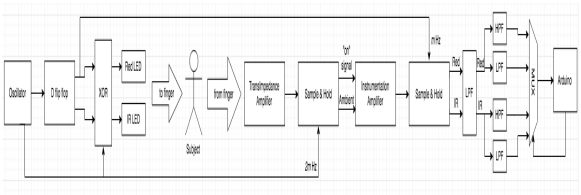

Fig. 11. Full Block Diagram of Pulse Oximeter Design

B. Design Iterations

We discussed the value of including a pulse ox versus a PPG with Laurent and Bill after realizing that our pulse oximetry design would take up prohibitive amounts of space and add a significant cost to the project. Ultimately, we settled on redesigning the PPG.

Our initial redesign was perhaps more similiar to the lab schematic than it should have been for a project of this caliber. We thought of other possible schematics but knew a fairly similiar topology would result in a reliable output signal. We planned to replace the LTC 1043 with the LTC 6943

- a switched cap building block that used $\frac{1}{5}$ of the power. Although it was slightly more expensive ($<$ \$1) it seemed worth it for our battery powered pcb. We wanted to include a variable gain option for hobbyist purposes so we placed potentiometers in series with the current limiting resistor for the LED and in series with the resistor between negative input and ground in our gain stage. We chose this resistor and not the one between the negative input and the output because we included a capacitor in parallel with that resistor for frequency limited gain and did not want to alter the time constant if the user changed the output gain.

After fully understanding the low-cost redesign specifications for this project we made significant changes to the circuit. We replaced the clock from the LTC 1043 with the output of a square wave generator which could be designed using a single op-amp and passive components (cost-cutting). The output was ± 4 (rails) at 1 kHz.

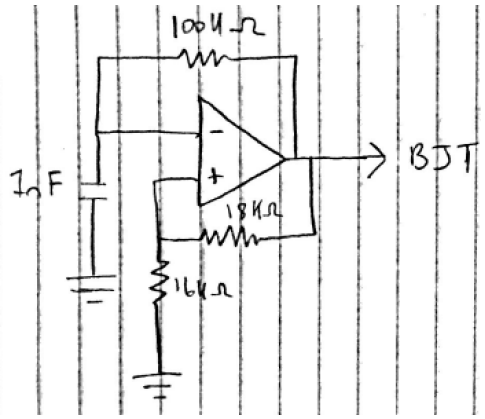

Fig. 12. Square Wave Generator used to trigger LED and as a clock input

We also redesigned the sample & hold topology with a the spst 4066 chip in mind; our plan was to turn two spst into one spdt. However, when testing our design we only had access to a spdt chip. To prototype, we turned two spdt switches into spst switches, and then turned those back into an spdt switch. The switches used input signals from the square wave generator (clock) and the drain of the BJT used to power the LED (inverted clock). These signals then fed into the positive and negative inputs of the instrumentation amplifier, respectively.

Instead of using the standard LT 1167 IA, we built an instrumentation amplifier using the standard 3 op amp topology and used 1% resistors for optimal CMRR.

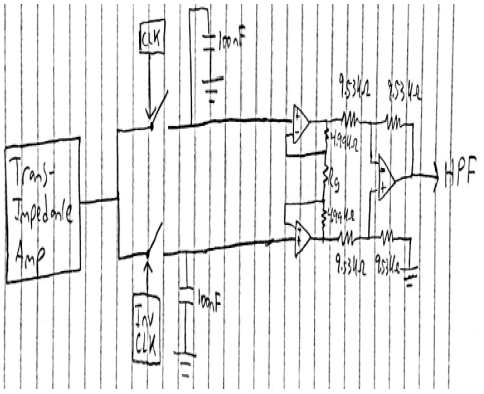

Fig. 13. S&H and instrumentation amplifier topology

The last part of our circuit was designed to extract the cleanest possible signal. The IA output fed into a high pass filter made from passive components with a cutoff of .1 Hz to eliminate the DC component of our signal. That fed into the variable gain stage mentioned above with a gain of between 84.3 and 501 (500 kΩ resistor divided by between 1 and 6 kΩ depending on user set gain). The output of the gain stage fed into a two-pole Sallen-Key with a cutoff of 25 Hz before being sent to the arduino.

A final difficulty arose after we sent this schematic in. It was time to send off for the PCBs and we were informed that instead of a square wave generator we needed to use the 555 timer which we had not had a chance to prototype; additionally we had not had a chance to prototype with the 4066 spst chip. We designed for a 555 timer with a 50% PWM, 1 kHz frequency, and rail to rail output using the design in Fig 14.

Once we characterized our 555 timer the final issue we had was getting the 4066 to do what it was built to do - switch. The 4066 requires its input signal to be at the rail amplitude, but previously the signals we used to switch were from the gate and drain of the BJT for the clock and inverted clock, respectively. Both of these were somewhat attenuated. We fixed this issue by moving the clock signal to the output of the 555 timer and connecting the emitter of the BJT to the negative rail instead of ground.

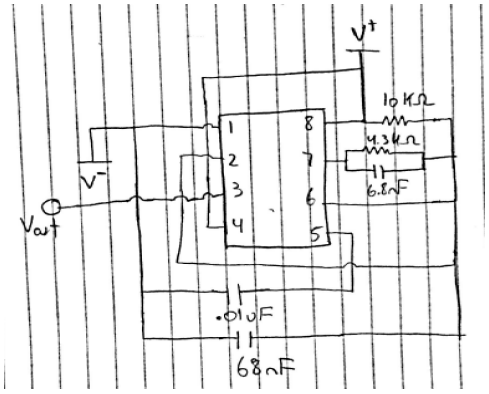

Fig. 14. This 555 timer topology produced a rail to rail signal at 1 kHz with a 50% PWM

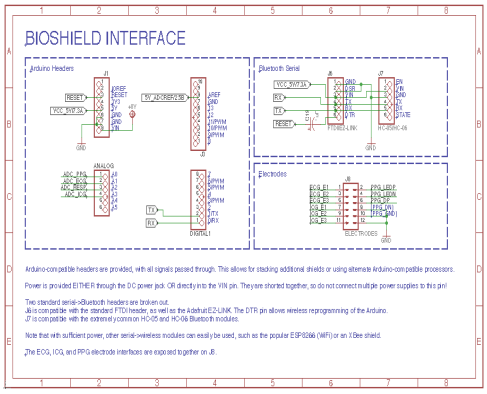

Fig. 15. The interface of the BioShield including bluetooth serial communication, electrodes, and arduino headers

With these changes we felt as if we had put significant thought into our design, change from the lab circuit, and reduced cost to an acceptable price point. Although our output signal had noticeable clock noise, it was easy to filter this out in post-processing to view a clean and accurate signal.

V. PCB Design

A. Schematic Design

With regards to the schematic files, the design of the files was fairly straightforward. We were provided schematics for the ICG, ECG, and PPG circuits from the other groups. In addition to those circuits, Shane designed a level shifter as well as circuitry for the power rails. We used eagle to design the .sch files from which the .brd (board) file would be designed from. In the .sch file, we ran into some problems while designing the schematics. The first issue we had was in regards to the PPG and ICG circuits. Eagle comes preloaded with a certain amount of libraries that include very common parts from the major electronic component producers such as TI, Linear Technologies, etc. These parts in turn are laid out in the configuration of the circuit you intend to design. The issue with the ICG and PPG circuits is the commutator chip. Originally a MAX4544 chip, it was later replaced in newer iterations of the schematic with a LTC 4066 chip. Despite having the Linear Technology library, this part was not present. We had searched for a footprint online in random libraries; however, our concern was that in the .brd file, the part would have specifications that would not match our design constraints. We created a new custom symbol for it.

For the non custom parts, we used standard library (RCL) American versions of through hole (DIP) resistors with 1% error and for capacitors, we used the same library and error as well. For the power rails, we added net labels to represent the connection to power for the ICG, ECG, and PPG. This simply makes the schematic files readability much easier. Taking advantage of the extended functionality of Eagle Pro, we were able to separate every major circuit into separate sheets which improves readability as well.

The first sheet was assigned to the Bluetooth and BioShield interface as seen in the figure above. This sheet is comprised of the headers for the Arduino, Bluetooth serial connection, and the analog input pins for the electrodes. There werent major issues in this sheet; we connected all the necessary power inputs through net labels. We did the same for TX and RX pins of the Bluetooth Serial Module, the electrodes, as well as the Arduino header pins.

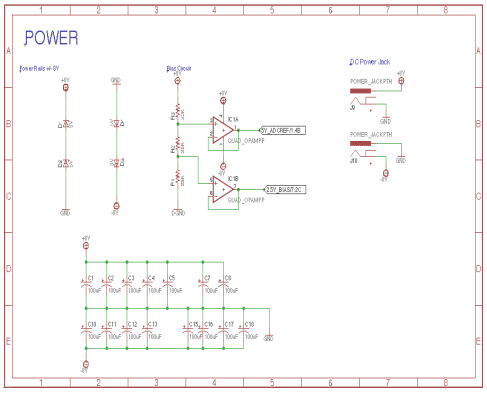

Fig. 16. The power schematic of the BioShield including power rails, decoupling capacitors, bias circuit, and DC power jacks

The Second Sheet, as seen above, was dedicated to the Power circuitry. This sheet is comprised of the power rails at +/- 8V, which is dropped as a result of 3V and 5V Zener diodes in series used as overvoltage protection, a bias circuit that generates our 5V Reference Voltage and the 2.5V bias voltage as well. This also shows the DC blocking caps that are connected to the power rails and two per chip are connected to the V+ and V- input pins. The last part of this sheet is the DC power jack. This is the connector for the 9V battery to provide power for the +/- rails.

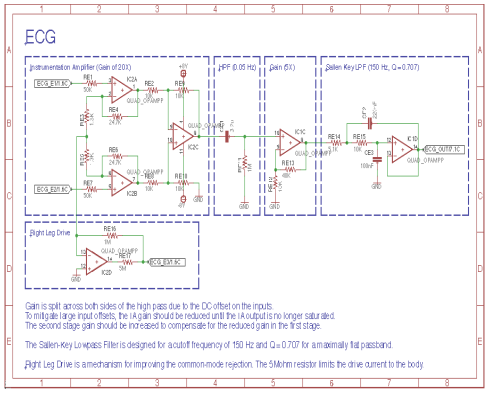

Fig. 17. This sheet is the ECG circuit which includes right leg drive for the purpose of common mode rejection

The third sheet is the ECG which includes an instrumentation amplifier, a High Pass filter at 0.5 Hz, a 5X gain amplification stage, Sallen Key low pass at 150 Hz, and a right leg drive circuit to improve common mode rejection. In this sheet, we modeled the implementation amplifier circuit by placing it in a quad 14 pin DIP generic op-amp package taken from the Linear Devices Eagle library. Since we intended to place 14 pin DIP (through hole) sockets on the actual PCB, for the design of the .sch file, there was no issues placing down a generic 14 pin DIP op amp. We placed the instrumentation amplifier and right leg drive circuits together in one package because we made a decision to place the ECG in a quad package with the power rails to save space on the board (the Sallen Key and gain amplification stages share the package with the power rails). The 20X amplification instrumentation amplifier and right leg drive circuits share one op amp package.

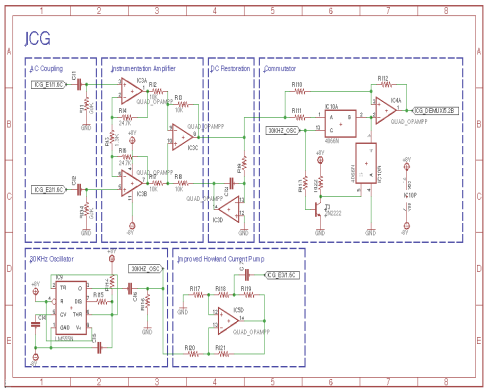

Fig. 18. This sheet contains the ICG circuit which includes a 30 kHz oscillator with 50% duty cycle and Howland current pump included

Sheets 4, as seen above, and 5, as seen below, is dedicated to the ICG circuit. The ICG circuit includes an oscillator, Howland current pump, AC coupling, instrumentation amplifier, DC restoration, the commutator circuit, channel separation, Band pass filter, gain amplification stage and low pass filter (2X), and Sallen Key low pass filter with optional gain. The challenge in this stage on the schematic was the arrangement of op amp packages due to the size constraints of the board as well as other considerations such as isolating higher frequency portions of the circuit as a result the oscillator and keeping signal crosstalk to a minimum by keeping overlapping connections to a minimum and wire lengths to a minimum as well. The oscillator is built from a LT555 timer based circuit which feeds into the Howland current pump. As a result, we decided to put one Sallen Key low pass, the Howland Current Pump, and two gain and low pass circuits in one quad package while keeping the rest of the circuits in their respective quad packages.

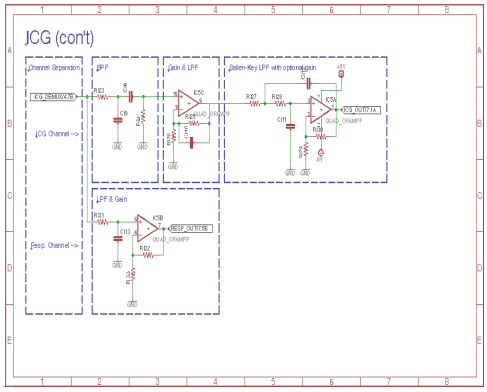

Fig. 19. This sheet is the ICG schematic continued showing the channel separation for respiratory and ICG signals

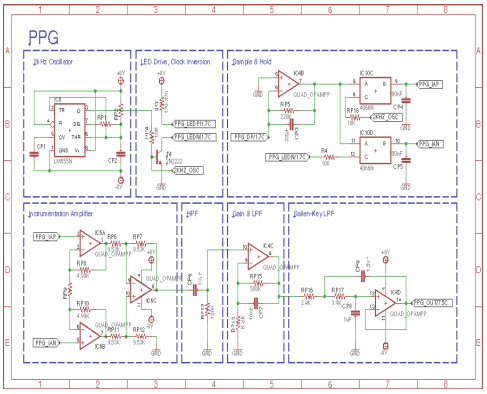

Fig. 20. This sheet is the PPG circuit which includes the 2kHz oscillator and sample and hold circuit

Sheet 6 is the PPG circuit. Designing this stage was not particularly bad however it required some maneuvering as a result of a shared LTC4066 chip with the ICG circuit. All of the packages are labeled with net labels as needed. This sheet contains a 2kHz square wave oscillator, LED drive circuit, a sample and hold circuit, instrumentation amplifier, High Pass filter, gain and low pass stage, and Sallen Key Low Pass.

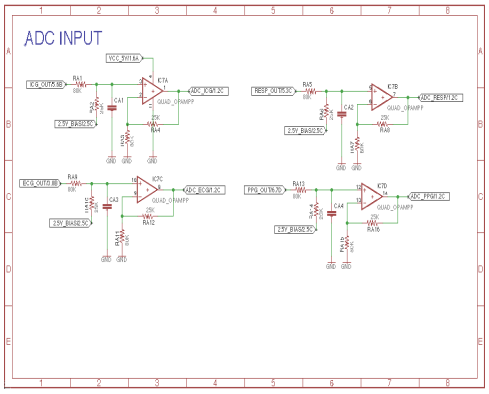

Fig. 21. The final sheet is the ADC input which combines the four signals and amplifies them and feeds them to the ADC.

B. Board file Design

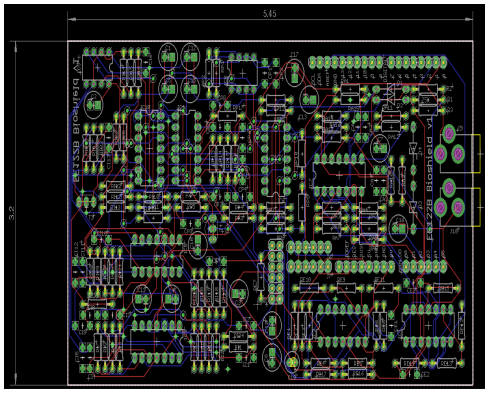

Fig. 22. The complete board file fully routed

The board file was generated directly from the schematic files. As seen above, the board is comprised of the ICG, ECG, PPG, level shifting circuit/bias circuit, the Arduino headers, Bluetooth serial module, and DC power jacks for the 9V batteries. The board was laid out in the present configuration as a result of multiple considerations. To begin the board layout, we considered how to maximize the space we had and minimize the overall size of the board. To do this, Shane and I decided to place down the Arduino headers on the outer edges of the board. We then looked at which circuits needed to be close together and which could be isolated/moved around depending on our needs. Because of the commutator chip (LTC4066) we decided to group the ICG and PPG together. We had anticipated this in the schematic design phase and grouped those op amp packages together. At this point, we had laid down the parts for the Arduino headers, whose location was finalized as soon as the ICG/PPG were grouped together. We had the freedom to move the ICG/PPG conjoined circuit wherever the rest of space not taken up by the circuit would stay the most contiguous. We placed it on the far left hand side of the board. We also took into consideration keeping the higher frequency components together and the PPG/ICG circuits both contain oscillators. As a result, conjoining the PPG/ICG circuits was ultimately beneficial more so than just spacing concerns. Keeping the high frequency circuits away from the lower frequency signals reduces possibility of signal interference as well as reducing cross talk. We decided on placing the ECG circuit close to the power rails simply because it was very isolated as far as input/output connections to other portions of the circuit go. We were able to minimize the connection length by placing the ECG closest to the power rail while maximizing distance between the PPG/ICG circuits and ECG. Then we placed the level shifter/DC bias circuit very close to the power rails. It made the most sense because the input was the DC power jacks and we had placed those closest to the edge of the board and headers to more easily supply power and keep trace lengths short. After the layout was finalized, we placed the decoupling caps, two per chip, as close as possible to the chip itself.

Routing the board was the biggest challenge simply because of the sheet number of components. We placed all of passives in the best configuration we could that would minimize the length of the trace. We used a trial and error method to route the board. There was an auto router; however, it often times routes all the possible wires it can and leaves the ones too difficult up to the designer which often times requires the designer to undo some traces the auto router placed to route the unrouted component. We used both layers of the board to route and prevent crossing lines from occurring. Any crossing lines will completely ruin the signal being carried in that wire. After completing routing for the board, the spacing between functionality was not very large nor separated; this is due to our sizing constraints. We finished the board at a final dimension of 3.2 inches by 5.45 inches.

VI. Software

A. Arduino Code and Bluetooth Module

The Arduino communicates with a computer through a Serial Port Profile, a profile for wrapping bluetooth communication with serial protocol. All of the bluetooth communication is abstracted by Adafruit's EZ-Link Bluetooth module, allowing the Arduino to send data through a standard FTDI header to the module and the computer to read data through a standard serial port (COM port for PCs and a serial file in /dev/ for Macs). The bioshield's data is sent over this serial profile in packets, the structure of which is shown in Fig. 23. The Arduino reads data from its four analog read inputs (A0-A3) and this data is sent sequentially through its serial port. In order to distinguish between the four signals present on the device, they are ordered and sent as a packet of 1 start byte and 4 data points, each 2 bytes, giving a packet length of 9 bytes. Since the Arduino's ADC gives a 10-bit value, each 10-bit data point is buffered on the left by 6 bits (100000) in the 2-byte data point. Due to this waste of bits, this method is suboptimal in speed and a method which sends data in words of size 10-bits would be faster.

Data Packet								
start byte	A0	A0	A1	A1	A2	A2	A3	A3
"00000000"	"100000xx"	"xxxxxxxx"	"100000xx"	"xxxxxxxx"	"100000xx"	"xxxxxxxx"	"100000xx"	"xxxxxxxx"

Fig. 23. The data packet structure for multiplexing signals across serial. "x" refers to data bits (most significant first).

The packet uses "100000" instead of "000000" to distinguish bytes potentially of value 0 from the start byte (i.e. a 10-bit value whose first and second most significant bits are 0). That said, the second byte in each pair of data bytes is raised by 1 to avoid sending a byte of value 0 and confusing it with a start byte. On the computer's side, the serial port is read until a byte of value 0 is read (which will now uniquely refer to the start of a packet). The only case in which this fails is if the data at the beginning of the serial port profile's transmission, which constitutes a header of some sort, is read before it ends. This header could include a 0, which would trigger the code to begin reading serial as if data was being transmitted. To avoid this issue, the serial port is read until the header ends (about 50 bits based on experience). Further investigation of the standards associated with the serial port profile's initialization protocol is necessary to make this preventative step fully robust.

B. Graphical User Interface

The GUI is written in Python and implemented with Kivy, an open-source, cross-platform library for creating Python GUIs. The code is documented with an installation guide and inline documentation to aid user customization. The screen displays the output of an Arduino analog pin (A0 - ECG by default) on a live plot. FIR filtering of the signal is shown on a plot below this plot. These plots are updated at a frame rate of 60 Hz. At each frame, all of the points that have been received since the last frame are plotted simultaneously for speed. Fig. 24 shows the a real-time screen capture of the PPG lowpass filter in action.

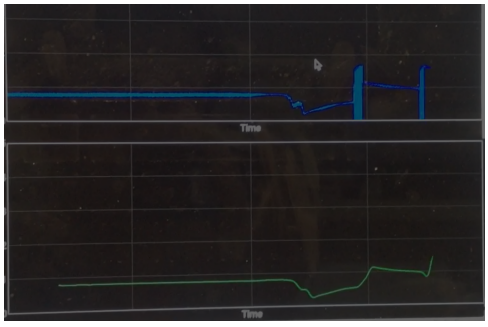

Fig. 24. A real-time screen capture of the PPG lowpass in action. The top (blue) waveform shows the unfiltered signal. The bottom (green) waveform shows the filtered signal.

C. Signal Post-Processing

In terms digital signal processing, we implemented four filters. We needed a lowpass for the ECG signal, as we were sampling at 500Hz, well above Nyquist which, for the ECG signal, was around 300 Hz. The ECG lowpass filter is shown in Fig. 25 and has a cutoff at 150Hz.

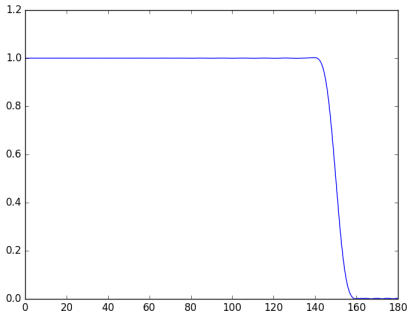

Fig. 25. The Lowpass Filter for the ECG has a cutoff frequency of 150Hz

Another filter that was used on the ECG signal was the 60Hz bandstop. It is shown in Fig. 26.

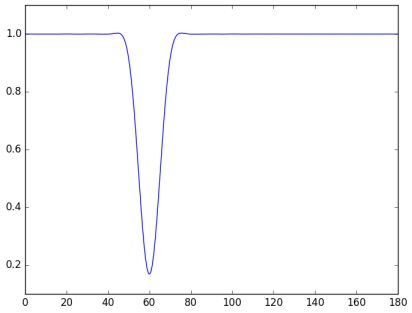

Fig. 26. The bandstop, or notch, filter at 60Hz

Furthermore, using data from the MIT-BIH database, we are able to demonstrate the effectiveness of these two filters together. This is shown in Figs. 27 and 28. As is visible in these figures, there is some smoothing in the ECG waveform that is done by the filtering. However, it is important to note that the signals from the MIT-BIH database have been filtered already with similar filters. Therefore, there is not as drastic an effect on the signal conditioning as there is for actual data that we read from the circuit.

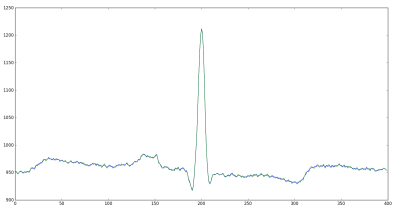

Fig. 27. The original (green) and filtered (blue) ECG waveforms from the MIT-BIH database.

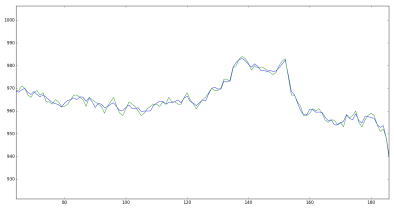

Fig. 28. The original (green) and filtered (blue) ECG waveforms from the MIT-BIH database. This image is zoomed to more clearly see the smoothing effect.

The final two filters we implemented were a 30Hz cutoff lowpass filter for the PPG (Fig. 29) and a 1Hz cutoff lowpass filter for the ICG (Fig. 30).

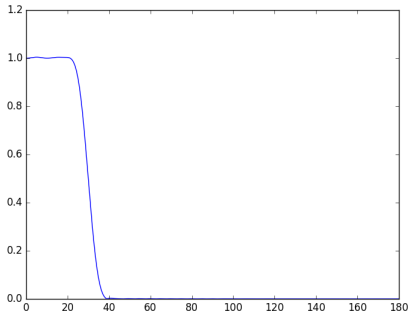

Fig. 29. The PPG lowpass filter has a cutoff frequency of 30Hz.

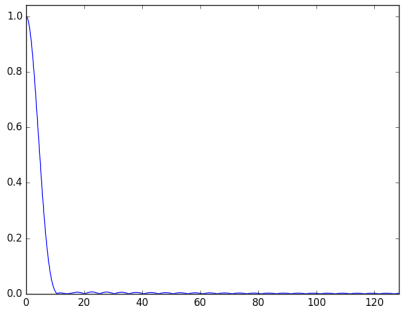

Fig. 30. The ICG lowpass filter has a cutoff frequency of 1Hz.

VII. Conclusion

In the end, the team successfully designed and built each stage of the proposed design. However, the full design of the system was not polished for the purposes of distribution. Although each stage worked individually, the full system (all three stages) has to be tested simultaneously and optimized to function concurrently. One obstacle in implementing a full functioning shield is the power consumption of the system altogether. The final design iterations of the project were not able to run at the same time with a +/-9V supply powered by two 9V batteries. One proposed issue is the current limitations imparted by the batteries. For further iterations, each stage will have to be optimized for power consumption, both to solve the issue of current draw, and to allow for the system to run longer on one battery charge. Optimizations can also be made in the software and use of the Arduino capabilities. Exploration of the Arduino's clock generation capabilities may allow on-board oscillators to phase out of the schematic, allowing for lower overall power consumption, though this will require some hacking of the Arduino MCU and manipulation of registers beyond the basic Arduino user functionality. Additionally, a final version of the PCB reflecting any changes in the circuit has to be routed.