MACHINE LEARNING ENABLED WEARABLE STETHOSCOPE

By

Erlis Kllogjri

Natalia Migdal

Sam Felder

Final Report for ECE 445, Senior Design, Spring 2018

TA: Hershel Rege

2 May 2018

Project No. 24

Abstract

Our device listens to the user's heart and outputs whether there is a possible pre-existing heart condition acting up that is of concern. It does this by using microphones placed near the heart. The microphones listen to the heart. From there we pass the microphone sound through to an analog signal processing unit that filters out excess noise, convert the signal to digital, take the FFT, and finds the MFCC components. We pass those components to a K-NN algorithm that decides whether the input was similar to previous abnormalities. If it is, we send the instance to the user's doctors, notifying them that an issue has arisen. Currently we are able to detect an issue with 86 percent accuracy and send the data within 9 seconds.

Contents

1. Introduction

In the United States alone, 735,000 Americans suffer from a heart attack every year [1]. Of those 735,000 Americans, 47% suffer from the heart attack outside of the hospital. Heart attacks, however, are not the only heart condition known to doctors in the United States. 610,000 Americans die every year due to some kind of heart condition, whether it indeed is a heart attack or any condition listed in Appendix A.

We propose a way to monitor a patient's heart outside of the hospital. Figure 1 shows the physical design of our system. On the outside, it is a wearable band that a patient would wear around the chest, near the heart. On the inside, there is a system constantly picking up the patient's heart sounds through a microphone, analyzing those heart sounds, and ultimately classifying the heart signal as normal, murmur, or extrasystole. Figure 2 is the block diagram of our whole system.

We will be taking you through our design decisions, our challenges, and how we verified and tested each block of our system. Our intended goals were to (1) pick up the heartbeat sounds in frequency range 20 and 150 Hz [2], (2) classify the heartbeat sound with an accuracy of 90%, and (3) run the classification algorithm within four seconds. We successfully picked up and analyzed the heart signals with a few caveats: our accuracy reaches a maximum of 86% and the classification algorithm takes 9.556 seconds.

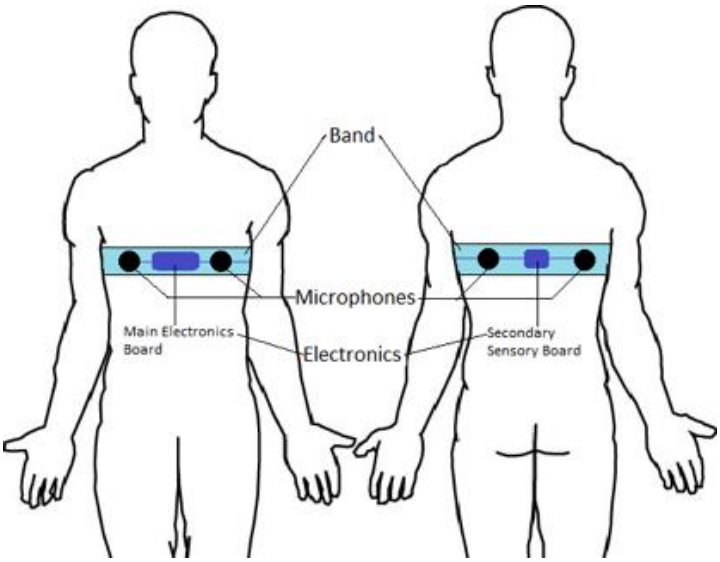


Figure 1. Physical design of the machine-learning enabled stethoscope.

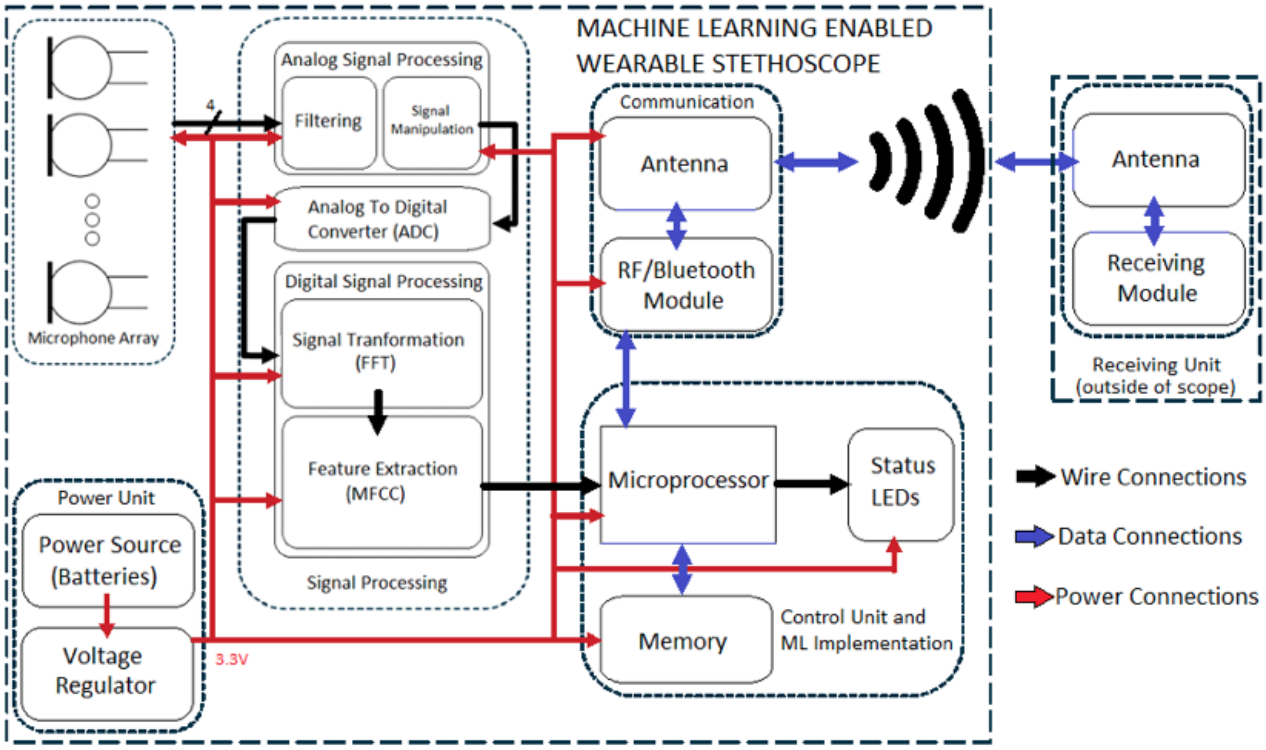
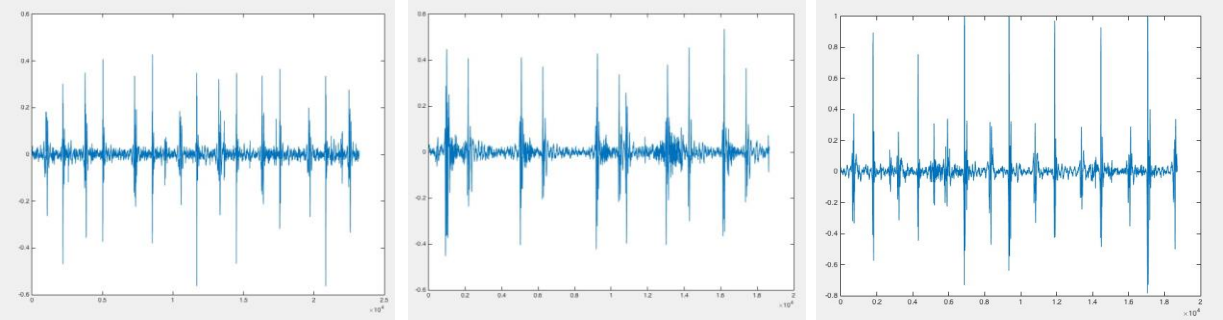


Figure 2. Block diagram of the system, containing the overall flow of the hardware and software.

1.1 Background Information

Before going into the design, it is important to clarify what we classify as normal, murmur, and extrasystole. A normal heart sound has a consistent “lub dub, lub dub” pattern. Figure 3a shows a MATLAB plot of a recorded healthy, normal heartbeat. Each peak represents a “lub” and a “dub” with little noise in between the peaks. A heartbeat with a murmur has abnormal noise between the “lub” and “dub” in the pattern. In Figure 3b, a murmur is evident through the visible noise or large amplitudes between the “lub” and the “dub.” This typically indicates abnormal blood flow near the heart and can be correlated with a serious heart condition. Finally, Figure 3c shows a heart pattern with an extrasystole. Having an extrasystole is another way of saying that the heart has an irregular rhythm, such as two “lubs” or “dubs” in a row. Figure 3c demonstrates the effect of an extrasystole with a very subtle “lub dub, dub” pattern.

The data we are using for training and classifying each incoming heart signal is from [3]. This database has all three of the classifications we used for our machine learning algorithm stored as Waveform Audio Files (WAV files).



(a) (b) (c)

Figure 3. Visual representation for (a) normal, (b) murmur, and (c) extrasystole WAV files.

2 Design

2.1 Design Procedure

2.1.1 Power Unit

For our power supply there were a couple of ways we could go. At first, we planned on creating our own regulator to drop the voltage but after doing some research, saw we could have a simple and complete circuit by using a pre-manufactured one. From here we needed to decide on the actual batteries. We needed something greater than 5 volts, which is the threshold input voltage for the voltage regulator. It also needed to be low profile since the plan was for our circuit to be wearable. We didn't want to use lithium ion batteries because of the safety issues it comes with. We settled on coin cell batteries because they allowed the voltage we needed while also being low profile and easy to replace.

2.1.2 Microphone Array

The microphones are the sensors of this circuit. They have the job of picking up the sounds produced by the heart and lungs so that we may then, through further processing, detect any issues with those organs. This means that the microphones need to be sensitive enough to pick up the sounds from the organs. The heart produces sounds in the range of 20Hz to 150Hz, and the lungs from 100Hz to 20kHz. We therefore chose microphones which had the capacity to pick up signals from this entire frequency range.

There are several microphone architectures that can be chosen here but given that this device is meant to be embedded as a wearable, we have the limitations of size and power. We chose a small condenser microphone with a 20-20kHz range and an operating voltage of 3-5V. To remove the noise below 20Hz we used the passive Resistor-Capacitor equation, Equation (2.1), to find appropriate resistor and capacitor values.

$$f_c = \frac{1}{2\pi RC} \tag{2.1}$$

Where f_c is the desired cutoff frequency, R is the resistor value and C is the capacitor value.

2.1.3 Analog Signal Processing

Now that we have a signal coming from the microphone unit we then need to further process it. This is first done by the Analog Signal Processing. Here we do 2 things; amplify the microphone input by a factor of 1.4 to reach a 3V range and separate the signal into a high pass filtered signal and a low pass filtered signal (corresponding to the heart and lung sound range). To do this, we used an active second order filter design. Here we introduce two equations, Equation (2.2) is used to calculate the values needed for the gain of the filter, and Equation (2.3) is used for the calculation of the resistor and capacitor values for the corresponding cutoff frequency in a second order filter.

$$G = 1 + \frac{R_2}{R_1} \tag{2.2}$$

Where R_1 and R_2 are the resistance values and G is the gain desired.

$$f_c = \frac{1}{2\pi R_1 C_1 R_2 C_2} \tag{2.3}$$

Where f_c is the desired cutoff frequency, R_1 and R_2 are the resistor values and C_1 and C_2 are the capacitor values. Since this is an active filter design we also need to incorporate operational amplifiers. Here we were limited from a prior design decision – we are only using one power line which runs at 3.3V. This means that the operational amplifier needed to operate at 3.3V. We chose an operational amplifier which had the lowest phase lag and power usage characteristics.

2.1.4 Microcontroller

The microcontroller choice ended up being very influential to the progress of the project. We wanted to choose a microcontroller which used little power (to meet the 12.5-hour operating time goal), 3.3V operating voltage, small physical profile, and simple to integrate into the hardware. This leaves us with 2 microcontroller architectures; ATmega microcontrollers and PIC microcontrollers. Given the goal of 4 seconds until a classification we wanted the faster microcontroller. This lead us to the PIC microcontroller. The PIC32 had sufficient processing capability as well as a fast analog to digital converter which negated the need for a separate analog to digital converter chip.

The microcontroller we chose, PIC32MX230F256B, has a 50Mhz core. From the slowest operation, the calculation of the fast Fourier transform, we find that we need at least a 4MHz processor to achieve the desired calculation time (through Equation (2.4)). The PIC32 exceeded this.

$$n = \#_{sample\,rate} * t_{sample} * \#_{operations} \qquad S_{CPU} > nlogn > 4MHz \tag{2.4}$$

Where $\#_{sample\,rate}$ is the maximum sample rate, t_{sample} is the length of each sampled audio in seconds, $\#_{operations}$ is the number of cycles it takes for each calculation and S_{CPU} is the speed of the core. When we looked for the desired analog to digital conversion speed and accuracy, we find that we needed a 10-bit digital reading, and at least 85 thousand sampled per second. The PIC32 has a capacity of up to 1 million samples per second, which far exceeds our requirements.

2.1.5 Digital Signal Processing (DSP)

To compare audio signals with one another, it is necessary to compute their Mel Frequency Cepstral Coefficients (MFCC's). We followed the flowchart in Figure 4, where we input the already-filtered audio signal and output the MFCC feature vectors to the machine learning unit.

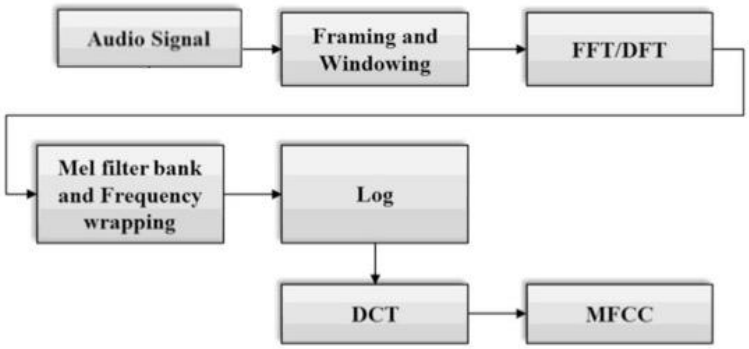
The flow starts with an incoming audio signal that gets windowed into short frames because DSP assumes that frequencies won't change much on a very short time scale. After the signal has been windowed, we calculate the Discrete Fourier Transform (DFT) of each frame,

$$S_i(k) = S_i(n)h(n)e^{-\frac{j2\pi kn}{N}} \tag{2.5}$$

where $S_i(k)$ represented the DFT of each frame *i* and $h(n)$ is an *n* sample long analysis window, also known as the Hamming window, that ensures the audio signal represented in frame *i* is a continuous signal [4]. We then proceed to derive the power spectrum from the DFT using

$$P_i(k) = \frac{1}{N}|S_i(k)|^2 \tag{2.6}$$

where $P_i(k)$ is the power spectrum of frame *i*, $S_i(k)$ is the DFT of each frame *i,* and *N* is the number of samples in frame *i.*



The resulting power spectrum gives us an idea of how of what frequencies are present in each frame. Our own ears cannot sense any difference between frequencies that are very close to each other in value. We create Mel filter banks and sum up the frequency energies in each filter bank. To create these triangular filters, we program the equation,

$$H_m(k) = \begin{cases} 0 & k < f(m-1) \\ \dfrac{k - f(m-1)}{f(m) - f(m-1)} & f(m-1) \le k \le f(m) \\ \dfrac{f(m+1) - k}{f(m+1) - f(m)} & f(m) \le k \le f(m+1) \\ 0 & k > f(m+1) \end{cases} \tag{2.7}$$

where $H_m(k)$ is the filter bank, *m* is the number of filter banks chosen by the programmer, and function *f* represents a list of *m + 2* Mel-spaced frequencies [4]. At this point, we have *m + 2* summed-up energies for each frame of the initial audio signal. The next step in the flowchart is to take the logarithm of all *m + 2* energies, again to mimic the human ear that does not pick up sound on a linear scale. Finally, we calculate the Discrete Cosine Transform (DCT) to separate the overlaps between each frame,

$$X_k = \frac{1}{2}\left(x_0 + (-1)^k x_{N-1}\right) + \sum_{n=1}^{N-2} x_n \cos\left[\frac{\pi}{N-1} nk\right] \tag{2.8}$$

where X_k is the DCT of each energy *k* and $x_{0 \ldots\ldots} x_{N-1}$ represent the *m + 2* energies [6]. This leaves us with 40 cepstral coefficients for each frame. We only take the first 13 to get the most prominent energies found in the initial audio signal [4].

2.1.6 Machine Learning

The importance of the machine learning unit is to quickly and accurately classify the incoming heart signal. After reading auscultation-based papers on classifying heartbeats, we narrowed down the machine learning algorithm search to *k*th-nearest neighbor (*k*NN), support vector machine (SVM), and multilayer perceptron (MLP). All three have their individual pros and cons. *k*NN is much simpler to implement than the latter two but is extremely sensitive to both the value of *k* and the distance metric chosen to measure the distance between the input point and the training data set points. SVM, on the other hand, is more unsusceptible to outlier points, meaning that if for some reason there is an audio sample in the training data set that does not accurately represent a heartbeat pattern, it will be ignored in the classification process. The downside to using SVM is that it works better when there are fewer data points. The larger the database, the less accurate and slower SVM becomes. The last option, MLP, is ideal for extremely large training data sets because the former two are slower to train.

[7] followed a similar research path for classifying an incoming heart signal. Figure 5 shows their results when testing the different algorithms. *Sen, Spec,* and *Acc* represent the sensitivity, specificity, and accuracy of each algorithm, respectively. Due to time constraints, the simplicity of *k*NN, and *k*NN's performance in Figure 5, we decided to move forward with *k*NN for classifying our heart signals.

Algorithm	Arrhythmia		
	Sen	*Spec*	*Acc*
k-NN	95.2%	99.7%	99.4%
SVM	89.3%	99.7%	98.9%
MLP	93.1%	99.4%	98.9%
C4.5	94.1%	99.6%	99.2%
LDA	82.0%	95.6%	94.6%

Figure 5. Accuracy of classification using different machine learning algorithms [7]

2.1.7 Memory

The memory in all the microcontrollers was insufficient for storing usage data and the database information used by the classification algorithm. The design decision was made to use a separate memory chip to augment the built-in microcontroller memory. In deciding how to implement the extra memory, we wanted to minimize the overhead from using a separate memory. An EEPROM chip met the requirements of fast read and write times, as well as minimal circuitry and easy integration. In choosing a specific memory chip, we opted for the largest possible (and reasonably priced) EEPROM memory chip – which we found to be 256kBytes.

2.2 Design Details

2.2.1 Power Unit

For the design of the power supply our biggest concerns included safety and the capability to hold a charge at maximum current for greater than 12.5 hours. We also needed to comply with the components we had already chosen, mainly the microprocessor. The processor took 3.3 volts as input so we used a voltage regulator that dropped the voltage to that amount. To power the circuit we chose 4 1.5 Volt coin cell batteries. To calculate the amount of power we needed for 12.5 hours we calculated the worst-case power consumption from each component and added them up. We chose these rather than alternatives because they provide a light profile and the voltage we need to power the circuit. We also included two capacitors to reduce the input noise of the batteries – this was recommended by the version of regulator we used. One of the capacitors was 100nF while the second was 10 microfarads.

2.2.2 Microphone Array

The microphone array is rather simple to construct. Each microphone needs only a high pass filter which acts to remove irrelevant low frequencies (less than 20Hz). Using Equation (2.1), we calculate the resistor to be 2.2k Ohms and capacitor to be 3.6 micro Farads for a cutoff frequency of 20Hz. The schematic is shown in Figure 6.

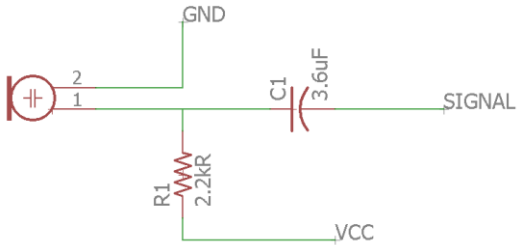


Figure 6. Microphone connection circuit with High Pass filter to remove signals below 20Hz.

2.2.3 Analog Signal Processing

Using Equations (2.2) and (2.3), we calculated the resistor and capacitor values (shown in Table 1).

Table 1 Analog Signal Processing Component Values

Component	R2	R3	R4	R5	R7	R8	C2	C3	C5	C6
Value	64.9k	1.69M	53.6k	130k	1.69M	1.69M	5.1n	5.1n	30n	30n

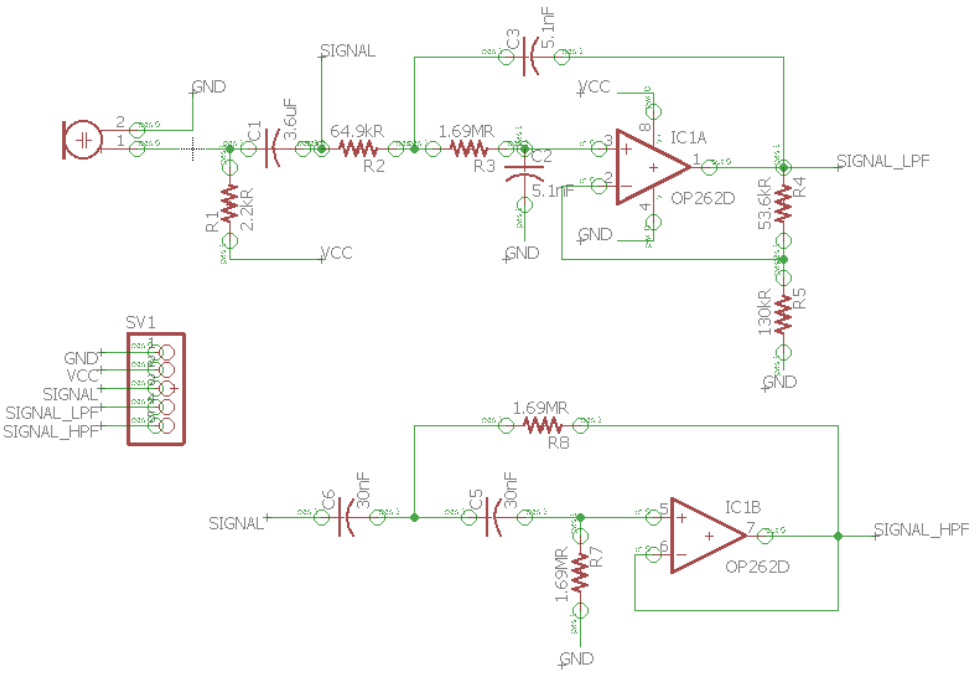


Figure 7. The circuit for the analog signal processing. The top schematic shows the microphone connected to the Low Pass Filter (LPF) and the bottom schematic shows the High Pass Filter (HPF)

2.2.4 Microcontroller

To install the microcontroller, we followed the design suggestions for the minimum required connections in the datasheet for the PIC32MX230F256B. The resulting schematic is shown in Figure 8.

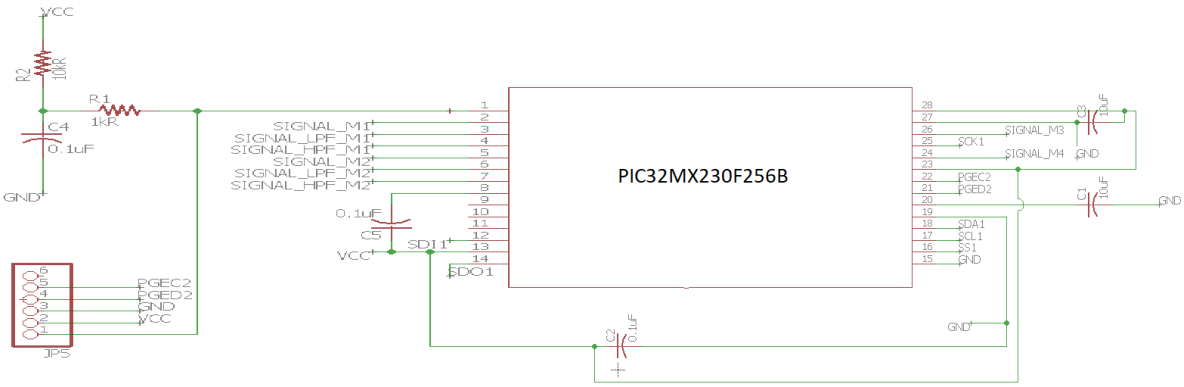


Figure 8. PIC32MX230F256B Schematic

The design decision to choose this microcontroller resulted in roadblocks in the project. Having underestimated the difficulty of finding supporting documentation for calculating MFCCs in C, as well as this microcontrollers inability to run the C++ code, we were ultimately unable to perform the classification on the microcontroller itself. We should have chosen a much more powerful microcontroller which would have had the capability of running C++ code with ease.

2.2.5 Digital Signal Processing

Using equations (2.5), (2.6), (2.7), and (2.8), we took the training data set of WAV files and converted them all to files with MFCC feature vectors so that we can use a training data set of MFCC's. In this unit, there are a few design choices we had to make in terms of how we chose the variables for calculating the cepstral coefficients. Table 2 shows these variables and the values we chose; we agreed to use the standard values for audio speech recognition, defined in [4]. In particular, we wanted to make sure the frame step was small because of how sensitive a heart signal is to a small change in frequency. With a higher overlap between frames, we captured a more accurate representation of the energies present in one sample.

Table 2 Defined variables for digital signal processing calculation

Variables	Design Choice
Window Length	25 ms
Frame Step	10 ms
No. of Filter banks	40
No. of MFCC's	13

The resulting format of each training data set file is an *m x n* matrix, where *m* is the number of frames in an audio signal, each of length 25 ms, and *n* is the number of MFCC's. Each row in the matrix represents a feature vector used to compare two sources of audio, utilized in the machine learning unit.

2.2.6 Machine Learning

*k*NN takes an input point and finds the distance between that point and all the points in the training data set. If a training data set has two classifications, "Class A" and "Class B," and the majority of the *k* points around the input point are classified as "Class B," then *k*NN classifies that input as "Class B" and adds it to the training data set so that the algorithm can "learn." In our case, our training data set points are the MFCC feature vectors of each input WAV file, our input in the incoming heart signal we just read from the patient, and our distance between points is how close the MFCC feature vectors are with one another.

*k*NN depends heavily on two metrics previously mentioned: the value of *k* and the distance metric. The value of *k* is determined through experimentation. The distance metric, however, had to be chosen ahead of time to compare MFCC features. We agreed to use a combination of dynamic time warping [8] and Euclidean distance to compare signals that vary on the time domain.

To calculate the distance between points using dynamic time warping, we followed the pseudocode in [8]. We created two arrays, one array for the input heart signal and one array for a training data set signal. Each element in the array represents a feature vector of the audio signal. We then took the Euclidean distance between all possible couplings of feature vectors and found a diagonal path through the matrix that represents the sum of the smallest Euclidean distances calculated, as seen by the blue blocks in Figure 9. This sum is the final distance metric used to find the *k* closest points to our input.

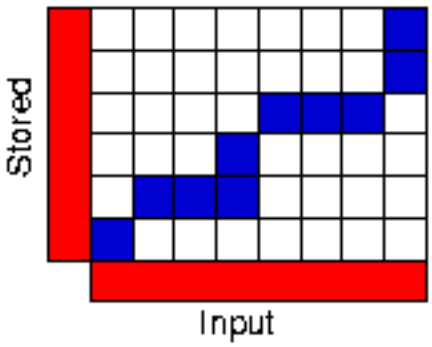


Figure 9. Dynamic time warping matrix of distances [9].

2.2.7 Memory

The training data set contains a total of 33.1 MB of data: 17.5 MB of normal WAV files, 10.2 MB of murmur WAV files, and 5.4 MB of extrasystole WAV files. After each run of the *k*NN algorithm, we add another MFCC file to memory, which is about 50 KB in size. The memory chip we initially proposed is no way capable of handling this amount of data and because of this constraint we were forced to run the algorithm on a laptop.

Integrating the EEPROM memory chip was easy due to an easy communication interface and low hardware overhead. The schematic for how the EEPROM was connected is shown below. The chip required only two connections to the microcontroller, the SCL1 pin and the SDA1 pin.

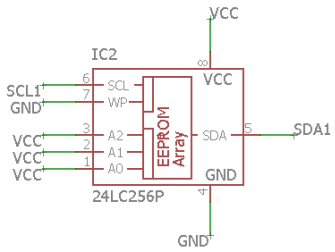


Figure 10. EEPROM chip connections

3. Design Verification

3.1 Hardware

3.1.1 Power Unit

The power block provides a single 3.3V line which powers all other components on the board. There were 3 requirements on the power unit; generate 3.3V with a deviation of 0.1V, be able to operate with currents of up to 200mA and provide 2500mAh of power (which is required for the system requirement of operating for at least 12.5 hours).

To verify the power regulator circuit, we supplied a variety of input voltages using a lab power supply – ranging from 0V to 20V and tested the output with a multimeter. The measurement was repeated across several voltage regulators. The setup looked as shown in Figure 11.

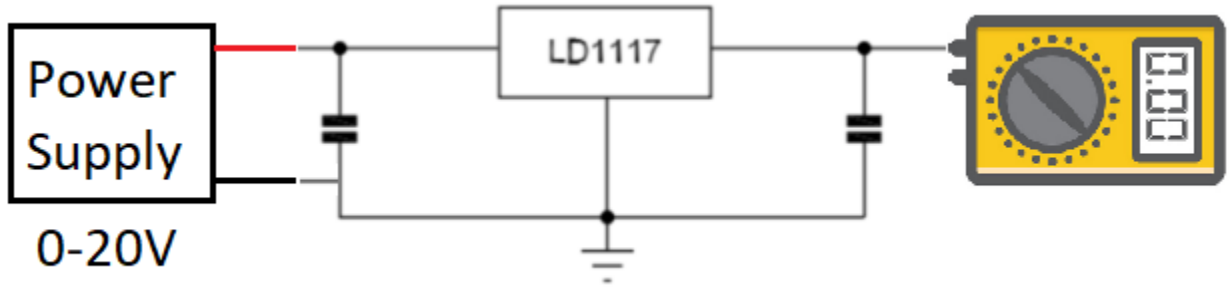


Figure 11. Power regulator verification setup. The middle unit is the power regulator circuit, with power supply on the left, and multimeter on the right.

The results from these measurements are shown in Table 3.

Table 3 Power Regulator Verification Results

Input Voltage (V)	Regulator 1 Output (V)	Regulator 2 Output (V)	Regulator 3 Output (V)
0	0	0	0
2	0.53	0.61	0.62
4	2.15	2.08	2.41
6	3.28	3.31	3.30
10	3.30	3.31	3.32
20	3.31	3.31	3.32

We required the power supply to provide at least 2500mAh of power. This was required to power the unit under worst case power usage for 12.5 hours. To verify this we connected the battery to a resistor load that produced 200mA current (our worst-case draw) and checked after the 12.5-hour period if the batteries were still providing power. Since this depletes the battery we operated this test on only two batteries (summaries in Table 4).

Table 4 Power Supply Verification Results

Battery #	Power after 12.5 hours
1	Yes
2	Yes

3.1.2 Microphone Array

The microphones are the sensors of our device and need to be able to pick up and amplify the sounds of the heart and lungs. We connected the microphones to an oscilloscope and tested their behavior. The setup is shown in Figure 12.

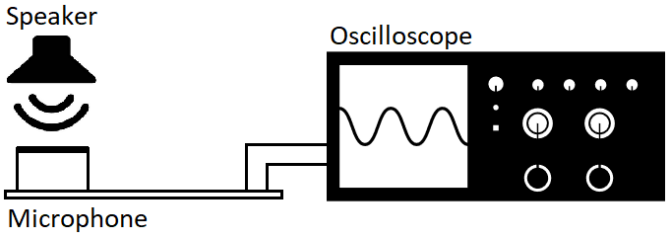


Figure 12. Microphone verification setup. The speaker outputs a known signal that is directed at the microphone. The oscilloscope picks up the signal and displays it.

Success in this test is defines as a peak-to-peak voltage of up to 3V across the relevant frequencies and from database audio as well as real time audio from a person. Two oscilloscope readings of heart sounds are shown in Figure 13, one from a database audio sample (left) and the second from a human subject (right).

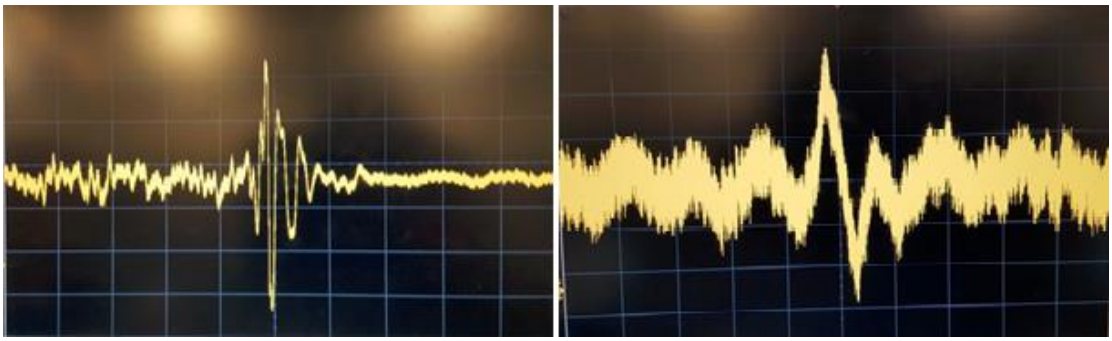


Figure 13. Two oscilloscope readings of heart sounds. The top represents a database audio sample, the bottom is recorded from a human subject.

The signal is strong enough to be used and now needs to be cleaned and separated into two frequency ranges. To verify the low pass and high pass filters is a straightforward procedure. We set up the filter circuits with a signal generator at one end and an oscilloscope at the other end. We then compare the input signal magnitude with the output signal magnitude to find the magnitude response. Experimental results are overlaid on simulated graphs in Figure 14 and Figure 15.

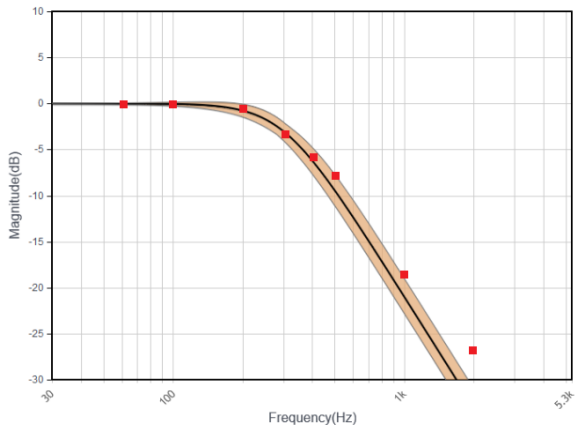


Figure 14. Experimental results (req squares) of magnitude response for the Low Pass Filter overlaid on top of simulation.

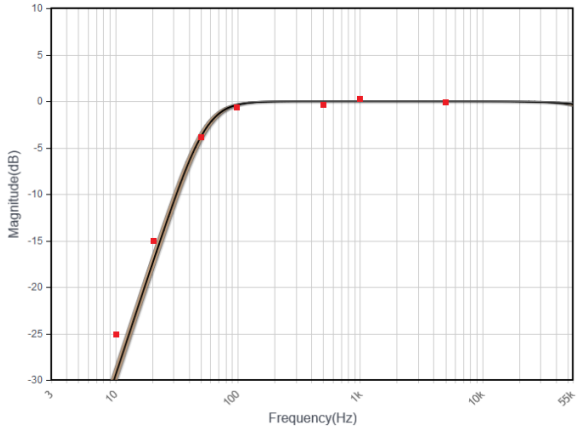


Figure 15. Experimental results (red squares) of magnitude response of the High Pass Filter overlaid on top of simulation.

3.1.3 Analog to Digital Converter

Here we needed to verify two things; that the analog to digital conversion was sampling the analog signals at the right frequency, and that the values would be in the appropriate range. To verify these, we used two methods. To verify on the breadboard, we added an LCD screen to our layout. On the LCD we printed the values as well as the number of times it had sampled within a period. On the circuit, we took advantage of the pin headers to install two LEDs which would blink every x number of samples in one experiment, and pulse width modulate the LEDs in another (resulting in brightness corresponding to the magnitude of the value read).

3.1 Software

3.2.1 Machine Learning

Once we finished programming *k*NN, we had to experiment with different *k* values to determine what value of *k* makes the classification of heart signals the most accurate. The machine learning database provided us with 412 WAV files. We took 312 of those files, converted them to MFCC feature vectors, and trained them for the training data set. The leftover 100 WAV files were then used for testing the algorithm. Figure 16 shows the probability of correctly classifying the heart signal as normal, murmur, or extrasystole, depending on what value of *k* is being used.

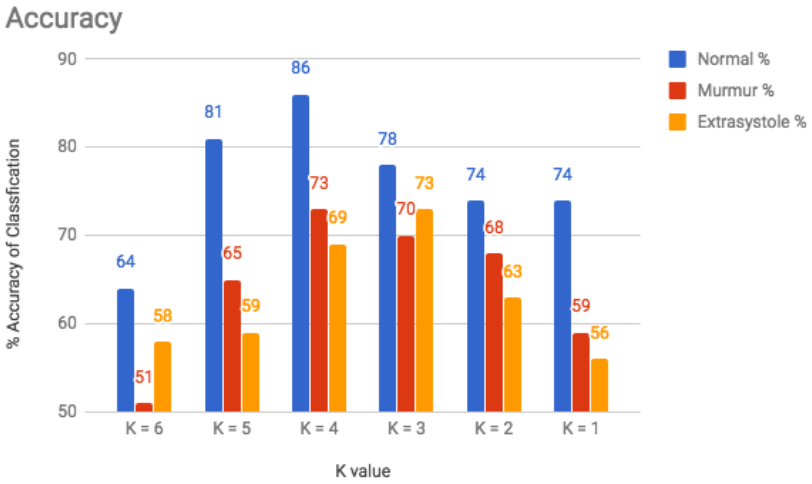


Figure 16. Probability of classifying a sound correctly depending on *k*.

Figure 16 demonstrates that *k = 4* gives us the highest accuracy of 86%. We noticed that our algorithm was significantly better at classifying a normal heartbeat, compared to classifying the other conditions. This just goes to show that because our training data set has significantly more normal WAV files, it is better at identifying a normal heartbeat pattern.

A requirement discussed in Appendix B was that the algorithm could run in under four seconds. Unfortunately, the fastest we could get our runtime to is 9.556 seconds, due to our $O(n^2)$ time complexity and large database.

4. Costs

4.1 Parts

Table 5 Parts Costs

Part	Manufacturer	Retail Cost ($) (per unit)	Bulk Purchase Cost ($) (Per 100 units)	Actual Cost ($)
P170SP1-FC15AR10K	TT_Electronics/Bl	1.05	0.775	11.16
ATTINY45-20SUR	Microchip Technology	1.1	0.917	13.32
68016-404HLF	Amphenol FCl	0.18	0.115	9.96
CF14JT27K0	Stackpole Electronics	0.1	0.016	0.72
FG24X7R1H224KNT06	TDK Corporation	0.29	0.120	2.00
RCE5C1H4R0C0DBH03A	Murata Electronics	0.45	0.210	3.18
RLB0913-121K	Bourns Inc	0.61	0.380	2.44
24LC256-E/P	Microchip Technology	0.91	0.830	3.84
CM250C32000AZFT	Citizen Finedevice Co Ltd.	1.24	0.990	10.72
ABMM-25.000MHZ-B2-T	Abracon LLC	0.8	0.580	2.40
810-10053-00050	CNC Tech	1.40	1.090	1.40
110990080	Seeed Technology Co	3.06	N/A	3.06
110990077	Seeed Technology Co	3.57	N/A	7.14
RDER71H105K2M1H03A	Murata Electronics	0.52	0.240	3.64
K104Z15Y5VF5TL2	Vishay BC Components	0.19	0.064	1.91
RCER71H106MWK1H03B	Murata Electronics	1.95	1.157	11.70
K333K15X7RF5TL2	Vishay BC Components	0.23	0.080	1.60
MFR-25FBF52-64K9	Yageo	0.1	0.031	0.82
HVR3700001694FR500	Vishay BC Components	0.54	0.201	3.87
MFR-25FRF52-53K6	Yageo	0.1	0.033	0.5

MFR-25FBF52-130K	Yageo	0.1	0.031	0.5
GRPB061VWCN-RC	Sullins Connector Solutions	0.65	0.465	6.08
OP262GSZ	Analog Devices Inc	5.59	4.749	22.36
PIC32MX230F256B-I/SP	Microchip Technology	3.94	2.89	7.88
AT25XV041B-SSHV-T	Adesto Technologies	1.10	0.869	2.20
QN9021/DY	NXP	3.93	2.900	7.86
PIC32MX230F256B-I/SS	Microchip Technology	3.66	2.69	7.32
ADF7241BCPZ	Analog Devices	4.61	3.37	9.22
Total				**158.8**

4.2 Labor

We expect to work 10 hours a week across 14 weeks. Using a salary of $40/hour, the labor cost per person comes out to,

$$salary * \ 2.5 * total\ number\ of\ hours = total \tag{4.1}$$

$$\frac{\$40}{1\ hr} * 2.5 * 14\ wks * \frac{10\ hrs}{1\ wk} = \frac{\$14{,}000}{laborer}.$$

Therefore, the total labor cost for all three of us is,

$$\frac{\$14{,}000}{laborer} * 3\ laborers = \$42{,}000$$

5. Conclusion

5.1 Accomplishments

In terms of individual blocks, we managed to implement all intended blocks except for the Bluetooth communication. We managed to set up the microphone array, have it detect the heart sounds and convert these sounds into digital readings at the required sample rates. We managed to integrate the memory chip and read from it and write to it. We also managed to implement a machine learning model on a computer which learnt from a database of heart sounds and classified issues with a peak accuracy of 86% and runtime of over 9 seconds, which fall just shy of the initial project requirements.

5.2 Ethical considerations

There are a few safety hazards to highlight with our proposed project. Starting from the simplest, there is a chance that the battery in the band can overheat, resulting in user discomfort or worse, an exploding battery. In order to avoid this, we are taking extra precaution in testing our circuitry in the safety lab and making sure the battery does not overcharge. There are a few things we can do to improve the safety of out product. For example, we can add a fire retardant chemical to the physical band material or use a material that protects the body against heat if the battery starts overheating. Both of these, however, can increase irritation, which is explained a few paragraphs below. For the scope of our project, we will do our best not to overheat the battery. However, we won't be providing any change in physical design to protect the user just yet. In order to follow IEEE Code of Ethics #1, we plan to fully disclose the safety issues with our product [7]. In the future, we hope to work on the potential battery hazards once we get the main function of the product working.

The material of the band is another important concern to the individual using the wearable stethoscope. Certain materials have the ability to induce an allergic reaction or create an uncomfortable rash on the individual where the band is placed. To avoid any kind of reaction, we plan on coating the device with a gauze bandage. Gauze bandages are prevalent in the medical world and are used specifically to protect the body. In order to decrease the chances of getting textile contact dermatitis from the gauze bandage, we are using a gauze bandage that is 100% cotton and dye-free [8].

The last safety concern we want to highlight is that our device will not be 100% effective. With our goal of 90% efficiency, there is a 10% chance that the device fails to detect a lung or heart problem for a user that may be completely dependent on the device. The initiative we are taking with this safety hazard is ensuring that, again, we follow the IEEE Code of Ethics #3 and to be transparent about the product's success rates as well as the potential flaws mentioned above [7]. The other issue with this is receiving too many false alarms. Not only would the doctor receiving the false alarm lose time trying to analyze it, but the doctor would be taken away from real-life threatening situations that could be occurring at the same time, ultimately decreasing the total quality of care. However, for the safety of the patients using our device, we prefer to output more false positives if that prevents more false negatives from occurring. In our implementation, a false negative can occur when a new symptom that perhaps isn't in the training data set yet gets fed into the k-NN algorithm where the nearest neighbors are normal heart or lung sounds. We recognize that there are ways to reduce false negatives, for example by feeding in more inputs into the microprocessor such as whether the user is sitting or jogging

at that moment to further analyze the heart or lung signal. For the scope of this project, reducing false negatives won't be a goal because as we mentioned above, the k-NN algorithm alone should get us an accuracy of at least 90% when it comes to detecting the abnormal or normal condition in the user.

Some studies [10], [11] show that heart rates are directly related to racial or gender differences. Although this is just a theory, we want to enforce equality and not segregate heartbeats or lung sounds into different groups, essentially following IEEE Code of Ethics #8 [12]. We know that machine learning training sets can create bias towards groups. So in order to prevent discrimination, we will continuously audit our algorithm and create a standard that works best for our device.

Because our medical device requires human participants, it's necessary that we go through the Institutional Review Board (IRB). It's particularly important because we need to access heartbeat and lung sound data from public records. Although there is an exception for using public data if the subjects cannot be identified [13], in order to both connect the public data to any kind of related heart or lung illness and to continuously get new data over time, we indeed need to identify the subjects. We do, however, qualify for the expedited process because our research is neither invasive and it only requires digital voice recordings [13].

5.3 Future work

The first thing we will be doing is completing the Bluetooth block of our project. Unfortunately, this part of the circuit was shorted on our PCB so we were not able to implement sending data through Bluetooth. We will then also implement the receiving end of the Bluetooth signal as well. This would allow us to integrate the product into hospitals and doctors offices.

After that we will turn to optimizing the circuit so that it can be as effective as possible to the user and the doctor. The first step would be increasing the size of memory to accommodate the training data. In order to do this we would need to take a look at other processors, finding one that has more memory but does not cause a substantial increase in power drain.

Now that we have an effective amount of training data we will be able to optimize our machine learning to have a recognition accuracy that's greater than 90 percent. There are multiple avenues we could take to achieve this accuracy. If we were to keep our K-NN algorithm, we could create more classifications, which in turn would create greater recognition accuracy. Alternatively, we could create a neural net rather than K-NN, which would serve two functions. First, dependent on the amount of data we have, it would allow us to get a greater accuracy than 90 percent. Second, it would dramatically increase the speed of our processing, allowing us to hit a run-time sufficient with our original requirements.

Once we had increased the memory and improved the speed to below 4 seconds the product would be ready for commercial use. From here we would implement the circuit as a wearable. There are multiple use cases possible. Two of them include implementing the circuit as part of an underwire in a bra, or strap in men, allowing for 24/7 access by the doctor. Another would be for post surgery use where the patient is consistently monitored by the hospital for conditions that are acting up.

References

[1] "Heart Disease Facts & Statistics | cdc.gov", Cdc.gov, 2018. [Online]. Available: https://www.cdc.gov/heartdisease/facts.htm. [Accessed: 05- Feb- 2018].

[2] American Journal of Respiratory and Critical Care Medicine. [Online]. Available: https://www.atsjournals.org/doi/full/10.1164/ajrccm.162.3.9905104. [Accessed: 07-Feb-2018].

[3] "mldata :: Repository :: :: Record of Heart Sound", *Mldata.org*, 2018. [Online]. Available: http://mldata.org/repository/data/viewslug/record-of-heart-sound/. [Accessed: 07- Mar- 2018].

[4] "Practical Cryptography", *Practicalcryptography.com*, 2018. [Online]. Available: http://practicalcryptography.com/miscellaneous/machine-learning/guide-mel-frequency-cepstral-coefficients-mfccs/. [Accessed: 15- Apr- 2018].

[5] Y. Chung, S. Oh, J. Lee, D. Park, H. H. Chang, and S. Kim, "Automatic detection and recognition of pig wasting diseases using sound data in audio surveillance systems," Sensors(Basel), 2013. [Abstract]. Available: Openi, https://openi.nlm.nih.gov/detailedresult.php?img=PMC3859042_sensors-13- 12929f1&req=4. [Accessed: 20-Feb-2018].

[6] "Discrete cosine transform", *En.wikipedia.org*, 2018. [Online]. Available: https://en.wikipedia.org/wiki/Discrete_cosine_transform. [Accessed: 14- Apr- 2018].

[7] T. Tanantong, E. Nantajeewarawat, S. Thiemjarus, "False Alarm Reduction in BSNBased Cardiac Monitoring Using Signal Quality and Activity Type Information," Sensors (Basel), vol. 15(2), Feb-2015. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4367394/#b21-sensors-15-03952. [Accessed: 20-Feb-2018].

[8] "Dynamic Time Warping", *En.wikipedia.org*, 2018. [Online]. Available: https://en.wikipedia.org/wiki/Dynamic_time_warping. [Accessed: 21- Apr- 2018].

[9] "11.2 Dynamic Time Warping", Web.science.mq.edu.au, 2018. [Online]. Available: http://web.science.mq.edu.au/~cassidy/comp449/html/ch11s02.html. [Accessed: 17- Apr- 2018].

[10] K. E. Reed, D. E. Warburton, C. L. Whitney, and H. A. Mckay, 23-Jun-2006. [Online]. Available: https://www.ncbi.nlm.nih.gov/pubmed/16770356. [Accessed: 08-Feb-2018].

[11] R. Santhanakrishnan, N. Wang, M. G. Larson, J. M. Magnani, R. S. Vasan, T. J. Wang, J. Yap, L. Feng, K. B. Yap, H. Y. Ong, T. P. Ng, A. M. Richards, C. S. P. Lam, and J. E. Ho, "Racial Differences in Electrocardiographic Characteristics and Prognostic

[12] Ieee.org, "IEEE IEEE Code of Ethics", 2016. [Online]. Available: https://www.ieee.org/about/corporate/governance/p7-8.html. [Accessed: 08-Feb-2018].

[13] A. Paulson, "Institutional Review Board (IRB)," *American Public University System (APUS)*, 09-Mar-2017. [Online]. Available: http://www.apus.edu/academic-community/research/institutional-review-board/index. [Accessed: 08-Feb-2018].

Appendix A Heart Conditions

Irregular rhythm, heart murmurs, signs of congestive heart failure, fluid in the lungs, valve leakage, aortic stenosis, pneumonia, atelectasis, pulmonary fibrosis, acute bronchitis, bronchiectasis, interstitial lung disease or post thoracotomy or metastasis ablation, hypersensitivity pneumonitis, alveolitis, asthma attacks, though it can also be a symptom of lung cancer, congestive heart failure, and certain types of heart diseases, Caused by narrowing of airways, such as in asthma, chronic obstructive pulmonary disease, foreign body. epiglottitis, foreign body, laryngeal oedema, crouppertussis (whooping cough) pneumonia, pulmonar edema, tuberculosis, bronchitis, inflammation of lung linings, lung tumors, pneumomediastinum, pneumopericardium

https://en.wikipedia.org/wiki/Crackles

https://en.wikipedia.org/wiki/Auscultation

https://en.wikipedia.org/wiki/Respiratory_sounds

https://en.wikipedia.org/wiki/Wheeze

https://en.wikipedia.org/wiki/Squawk_(sound)

http://www.who.int/gard/publications/chronic_respiratory_diseases.pdf

https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4507578/

https://www.webmd.com/lung/copd/news/20170929/respiratory-disease-death-rates-have-soared#1

https://www.healthypeople.gov/2020/topics-objectives/topic/respiratory-diseases

https://www.ieee.org/about/corporate/governance/p7-8.html

Appendix B Requirement and Verification Table

Table 6 System Requirements and Verifications

Requirement	Verification	Verification status (Y or N)
Microphone Array		
• Have a response in the frequency range between 25Hz and 25kHz • Operate on 3.3V +/- 0.1V	• Produce audio through the entire range and use an oscilloscope to plot the voltage response • Attempt to power microphones with a Vcc between 3.2V and 3.4V	Yes
Power Supply		
• Generate 3.3V +/- 0.1V • Can operate at currents 0-200mA • Batteries provide 2500mAh of power	• Measure the output voltage from the voltage regulator and verify that it stays within 3.3V +/- 0.1V • Use a constant current circuit to draw 200mA from the power supply and voltage regulator • Connect the battery to a discharging circuit. Verify that at the maximum current (200mAh), it runs for 12.5	Yes
Analog Signal Processing		
• -3dB response below 25Hz and above 300Hz for filtering out heart sounds. (Figure 1) • -3dB response below 50Hz and above 2500kHz for filtering out lung sounds. (Figure 2)	• Use signal generator to generate signals at 25Hz and below. Measure frequency response to verify -3dB below. Do the same for signals above 300Hz. • Use signal generator to generate signals at 50Hz and below. Measure frequency response to verify -3dB below. Do the same for signals above 25kHz.	Yes
Analog to Digital Converter		
• 10 Bit ADC with 60ksps • 8 Analog Input Lines for ADC	• Generate a signal with Nyquist frequency requirement >60Hz. Use ADC to look for biasing of signal. • Generate 8 independent signals. Read values converted to digital for all signals, compare to input signals.	Yes
Microprocessor		
• Can receive and transmit through UART at a rate of >10Mb/s	• Send a 100 Mb random message through the UART port. Verify signal received is the same as	Yes

• Can receive and transmit through SPI at a rate of >10Mb/s	signal sent. Verify that it took less than 10s. • Send a 100 Mb random message through the SPI port. Verify signal received is the same as signal sent. Verify that it took less than 10s.	
Memory		
• Write/Program memory at 102 kbits/s • Have a total usable memory of > 256 kbit • Operate on 3.3V +/- 0.1V	• Record the time it takes to write a large file (~2 MB). Attempt multiple times. • Try to fill the memory with > 256 Kbit • Attempt to power chip with a Vcc between 3.2V and 3.4V	Yes
Digital Signal Processing		
• Processor speed needs to be larger than 2.8MHz.	• Complete an FFT transform of a measured heartbeat (file size of 1.4Mbits) in less than half a second (which would be the processing time for a >2.8MHz processor).	Yes
Software		
• Classification accuracy of at least 90 percent • Runtime of 4 seconds	• Run the algorithm to check for speed • Run the algorithm to check accuracy against training set	No