ABDULLAH GÜL
ÜNİVERSİTESİ

Project Name:

Electronic Stethoscope

Department of

Electrical & Electronics Engineering

Abdullah Gül University

Project Report

EE2100 Signal Acquisition, Processing and Analysis Capsule

Submitted on: 02/01/2025

Submitted by: 2111011062

Enes ERDOĞAN 2111011061

Fatih Mehmet AYHAN 2111011062

Emirhan SÖNMEZ 2211011041

Table of Contents

Objective

The aim of this project is to build a digital stethoscope and use it amplifies heart sounds and estimates heart rate in beats per minute (BPM). Also listening to lung sounds and calculating RPM. Our aim is to combine the following. Traditional auscultation techniques with up-to-date digital signal processing provides a flexible tool for monitoring cardiovascular health. We designed this digital stethoscope as a real-time visualization and heart rate diagnostic device.To achieve this, the stethoscope is used as a microphone that captures sounds is then amplified through an amplifier circuit. Amplifies the sound for further processing. We plan to use a passive high permeable filter and an active low-pass filter, so that the range of sound we hear he takes care of the best.

The amplified and filtered analog signal is then sent to the Arduino Uno, where theoretically the signal is sampled and digitized using an Analog-to-Digital Converter (ADC).MATLAB is intended to use our project to allow users to observe the signal in real time and analyze heartbeat trends, and we have completed the software part

Background

Monitoring and analyzing heart rate is vital for evaluating cardiovascular health, which makes stethoscopes a key instrument in medical diagnosis. In this project, we developed a digital stethoscope that amplifies heart sounds, detects heartbeats, calculates beats per minute (BPM), and measures RPM for lung sounds.

Our system started with a real stethoscope integrated with a microphone that picked up the audible signals from the body. These are weak signals, often contaminated with noise, which thus required amplification and filtering. The microphone converted these acoustic signals into electrical signals. These were then amplified using a capacitor and then an inverting amplifier circuit. We then did a two-stage filtering of the signal to ensure clarity. To remove low-frequency noise like murmurs and respiratory sounds, a passive high-pass filter was used that allowed only frequencies above 30Hz to pass. Then, we applied an active low-pass filter that kept frequencies between 30-300Hz, which is ideal for heart sounds, filtering out sounds above 300Hz.

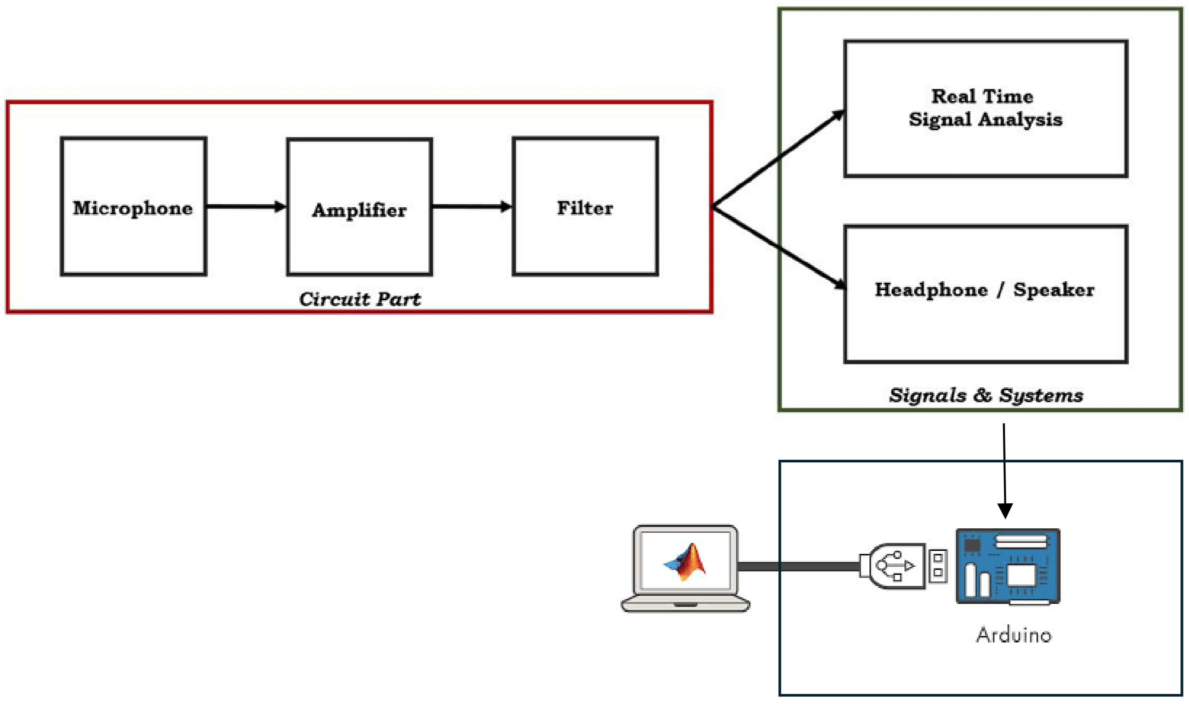
The filtered analog signal is converted to a digital signal using an Arduino Uno's Analog-to-Digital Converter. The ADC converts the continuous analog signal into a discrete digital format and samples at an appropriate rate to preserve the signal's essential characteristics. In the Arduino programming environment, a moving threshold algorithm is used to detect heartbeats by determining the peaks corresponding to systolic sounds (S1) and diastolic sounds (S2). In this theory, we simulate the digital signal we receive in matlab and see the RPM, BPM values on the screen

Introduction:

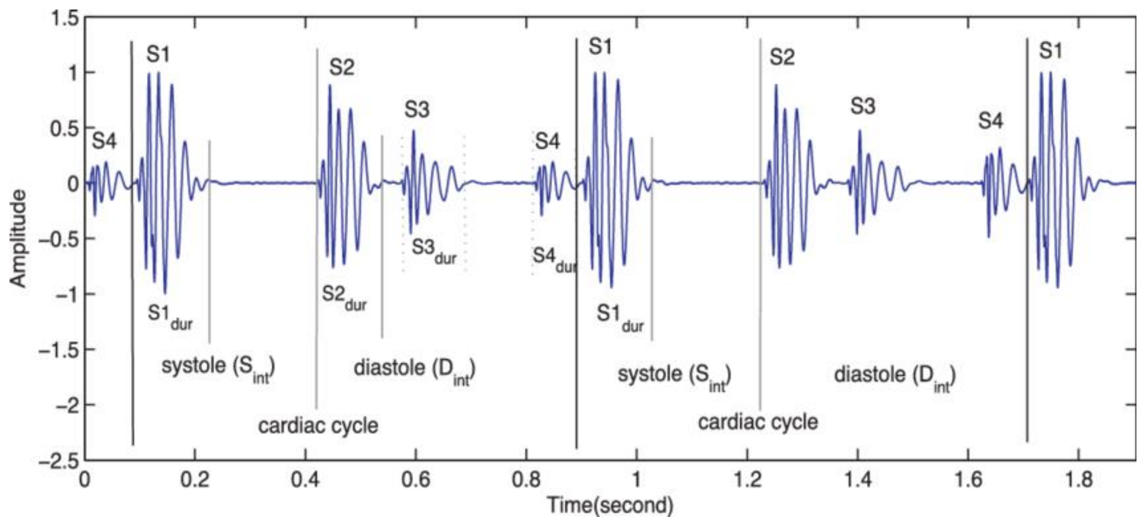
Our project is a digital stethoscope that also calculates the number of beats per minute. The design of our project includes a collection circuit, data processing in Arduino and It is designed to represent data in Matlab. The first part of the stethoscope is the acquisition unit, which consists of a stethoscope. Real stethoscope paired with a microphone and an amplifier circuit. The microphone captures

Audible signal from the body that is acoustically amplified by a stethoscope. Analog data will be independently sampled by the Arduino and capture the appropriate signal for pulse detection. Programming Environment, using a moving threshold scheme detects actual heartbeats and derives the heart rate from this. The signal is then displayed simultaneously in Matlab as a simulation, such as beats per minute.
The overall architecture of the system is as follows: microphone, amplifier, filters, buffer, Arduino Uno.

Block Diagram:



ANATOMICAL DESCRIPTION OF HEART SOUNDS



The first heart sound (1): It registers in low-frequency 10 Hz to 200 Hz last 100 minutes its recording focus is located at the tip of the heart, the phono cardio graphic analysis makes it possible to distinguish four components among which only the second and the third are audible.
This sound is contemporaneous with the closing of the Atrioventricular (AV) valves, the contraction of the ventricles, and the opening of the sigmoid valves. It seems that the two audible components of these sounds are linked to the mobilization of the AV valves. Phono-cardio graphic recordings, it is thus possible to discern two contemporary components M1 and M2 of the closings of the mitral and tricuspid valves.

The second heart sound (S2): This sound marking the end of the ventricular systole lasts less than 100 minutes and is located in high-frequency 20 Hz to 250 Hz. Its recording focus is located at the base, that is to say at the upper part of the thorax, on either side of the sternum; Most often, this second sound consists of four components, two of which are main. The first aortic and the second pulmonary. This second sound (S2) is contemporaneous with the closing of the sigmoid valves.

The third heart sound (S3): The third sound S3 from the heart occurs at the start of diastole 0.1 seconds to 0.2 seconds after the second sound, at the end of the rapid ventricular filling phase: it is usually of low-frequency 25 Hz at 75 Hz and can be heard normally in children or adults under 30 years of age. Beyond this age, its presence most often testifies to a pathological state, “proto diastolic gallop” of heart failure.

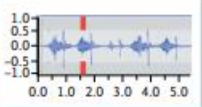
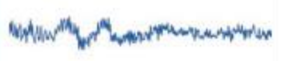
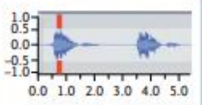
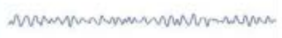
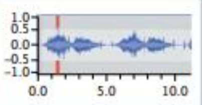
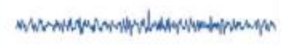
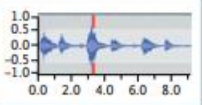
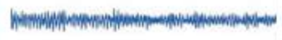
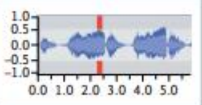
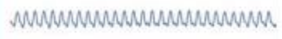
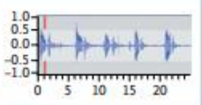
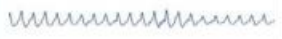
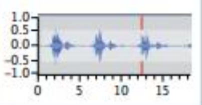
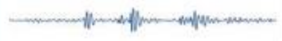
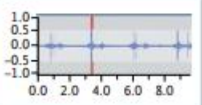
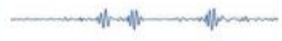
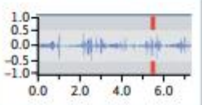
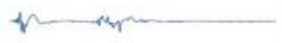
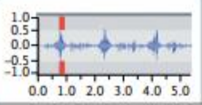
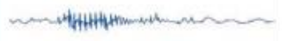
The fourth heart sound (S4): This fourth sound S4, which is never audible under normal conditions, manifests itself on the plot by small, low-frequency oscillations from 15 Hz to 75 Hz. It is due to the arrival of the systolic atrial wave in the ventricular cavity, B4 Sound is generally of lower frequency than B3 sound and shorter duration.

When we examined these values, we decided to take the frequency bandwidth of our filter in the range of 30 – 300 Hz.

ANATOMICAL DESCRIPTION OF LUNG SOUNDS

The sound of normal breathing heard over the surface of the chest is markedly influenced by the anatomical structures between the site of sound generation and the site of auscultation.

Characteristically, normal lung sounds are heard clearly during inspiration but only in the early phase of expiration. In sound analysis, the frequency range of normal lung sounds appears to be narrower than that of tracheal sounds, extending from below 100 Hz to 1500 Hz, with a sharp drop at approximately 100 to 200 Hz.7 The idea that "vesicular" sound is produced by air entering the alveoli ("vesicles") is incorrect. Indeed, modern concepts of physiology indicate that in the lung periphery gas molecules migrate by means of diffusion from parts of the lung reached through bulk flow, a silent process. Most important, studies support the idea of a double origin, with the inspiratory component generated within the lobar and segmental airways and the expiratory component coming from more central sources.

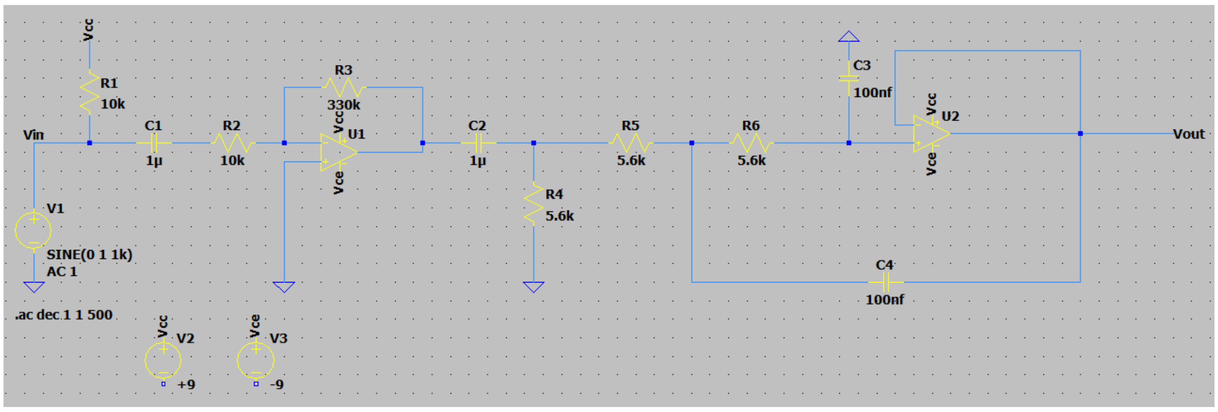
Respiratory Sound	Amplitude–Time Plot	
Acoustics	Unexpanded time	Expanded time
A Tracheal Sound White noise Typical frequency, 100–5000 Hz Drop of energy at 800 Hz		
B Normal (Vesicular) Lung Sound Low-pass-filtered noise Typical frequency, 100–1000 Hz Drop of energy at 200 Hz		
C Bronchial Breathing Strong expiratory component An intermediate sound between tracheal and normal breathing		
D Stridor Sinusoid Fundamental frequency, >500 Hz		
E Wheeze Sinusoid Typical frequency, >100–5000 Hz Typical duration, >80 msec		
F Rhonchus Sinusoid Typical frequency, about 150 Hz Typical duration, >80 msec		
G Fine Crackle Rapidly dampened wave deflection Typical frequency, about 650 Hz Typical duration, about 5 msec		
H Coarse Crackles Rapidly dampened wave deflection Typical frequency, about 350 Hz Typical duration, about 15 msec		
I Pleural Friction Rub Rhythmic succession of short sounds Typical frequency, <350hz Typical duration, >15 msec		
J Squawk Sinusoid Typical frequency, 200–300 Hz Typical duration, about 200 msec Followed or preceded by crackles		

When we examined these values, we decided to take the frequency bandwidth of our filter in the range of 150 – 1350 Hz.

ANALYTICAL AND SIMULATION PROCEDURES

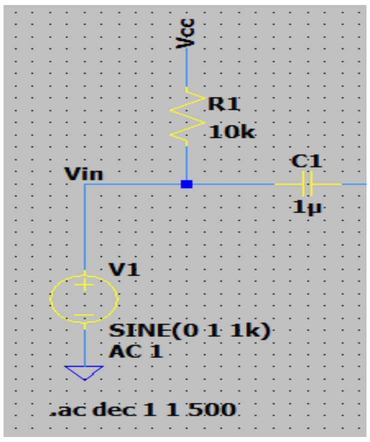
In this study, we carried out the design and testing process in three stages. First, we designed an electronic circuit that captures, amplifies and filters the signals from the heart and lungs in the appropriate frequency range. Then, we converted the obtained analog signals into digital form using Arduino. In the last stage, we processed these digital data in MATLAB program and made various analyzes in computer environment. In this process, both simulation, lab and MATLAB parts have been done carefully in order to obtain accurate and reliable results.
Let's start by examining our filter for the heart;

Filter and circuit design for heart sound filtering



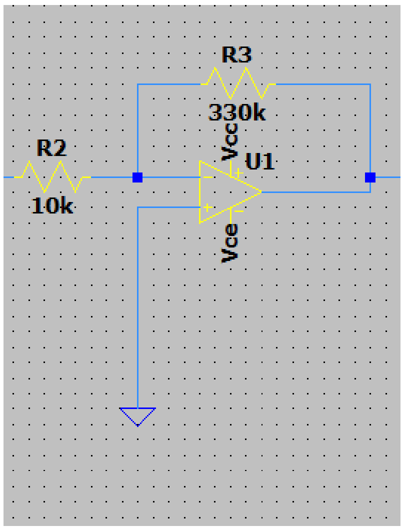
LT spice circuit filtering the heart signal

When we examine our circuit in general, we see that we receive the input to the circuit with the microphone part and then use our C1 capacitor to convert the incoming dc signal to Ac. Then we use an inverting amplifier. In the filtering part, we used a passive and an active filter. First, we filtered the sounds below a certain frequency using a high pass filter, then we filtered above a certain Hz using an active low pass filter. Thus, we have officially created a band pass and created a filter that passes the 30-300 Hz range we have determined for the heart. Let's examine the stages separately with their calculations



In this case, the incident sound wave causes a mechanical deflection of the cone and the voice coil. According to Faraday's law, as the coil moves in a magnetic field (generated by a permanent magnet), a time-varying current will be induced. This will be the input signal of our circuit. Then a C1 coded capacitor is used to convert the Dc signal to Ac signal.

Microphone Circuit



Inverting Amplifier circuit

Here we amplify our incoming signal along with the noise because our signal is so small that it would be impossible to hear a sound at our output, so we apply a gain to the circuit. Even if our noise increases, we will filter them at a later stage.

Components:

- *Operational Amplifier (U1A - NE5532)*
- *Resistors:*
  - *R2 = 10 kΩ*
  - *R3 = 330 kΩ*

The point where the negative input of the amplifier = V2
R3= Rf and R2= Ri

$$i = \frac{v_{in} - v_{out}}{Ri - R_f}$$

$$i = \frac{v_{in} - v_2}{Rin} = \frac{v_2 - v_{out}}{Rf}$$

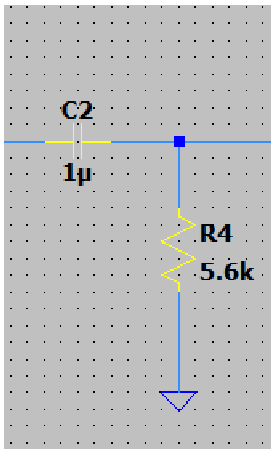
$$\frac{v_{in}}{v_{0ut}} = v_2 \left[\frac{1}{R_{in}} + \frac{1}{R_f}\right] - \frac{v_{0ut}}{R_f}$$

And as,

$$\text{i}= \frac{Vin - 0}{R_{in}} = \frac{0 - v_{out}}{Rf} \qquad so \qquad \frac{R_f}{R_{in}} = \frac{0 - v_0}{v_{in}}$$

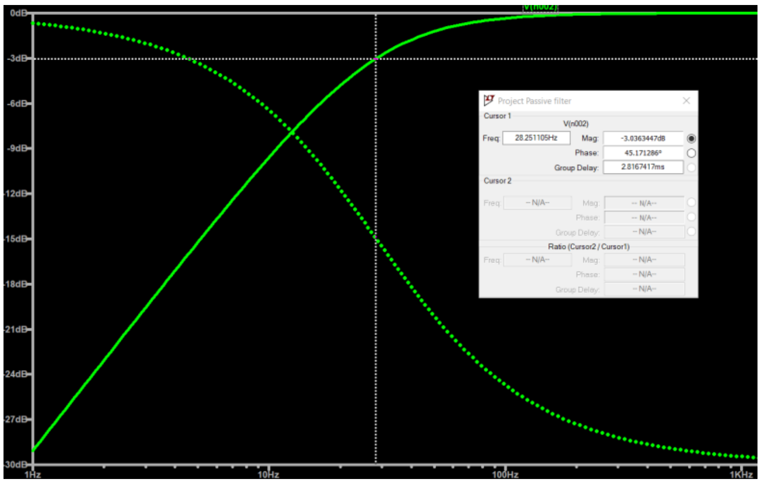
$$\text{Gain (Av)}= \frac{v_{0ut}}{v_{in}} = -\frac{Rf}{R_{in}}$$

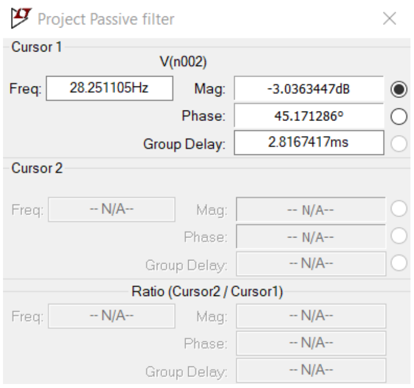
$$= -\frac{Rf}{R_{in}} = -\frac{R3}{R2} = -\frac{330K}{10k} = \text{-33}$$



Passive High-Pass Filter

A high-pass filter provides easy passage of high-frequency signals from source to load and difficult passage of low-frequency signals. We placed a capacitor in series with the load and designed a passive high-pass filter. We designed this filter to approximate the frequency onset of the heart. We want our system not to pick up the Hz range below this frequency). The cutoff frequency is equal to 0.707 or $\frac{1}{\sqrt{2}}$. So let's analyze our circuit and find our cutoff frequency.





```
Project Passive filter
Cursor 1
                    V(n002)
Freq: 28.251105Hz      Mag: -3.0363447dB
                     Phase: 45.171286°
               Group Delay: 2.8167417ms
Cursor 2
Freq: --N/A--          Mag: --N/A--
                     Phase: --N/A--
               Group Delay: --N/A--
           Ratio (Cursor2 / Cursor1)
Freq: --N/A--          Mag: --N/A--
                     Phase: --N/A--
               Group Delay: --N/A--
```

Simulation cursor data

Passive High-Pass Filter simulation

When we simulate our high-pass filter on Ltspice, a waveform is generated as seen above. - The -3 dB point is considered the cutoff frequency in both theory and practice, as it is the frequency at which the total power of the signal drops to half. So we are looking at the -3db point.

So

$$\frac{\rho_{out}}{p_{in}} = 0.5 \quad 10\log(0.5) = -3\,db$$

Theoretical Calculation of Passive High-Pass Filter

$$v_0 = v_{in}\,\frac{R}{R+\frac{1}{jwc}}$$

$$H(\omega) = \frac{v_0}{v_i} = \frac{R}{\frac{Rj\omega c+1}{j\omega c}} = \frac{Rj\omega C}{Rjwc+1}$$

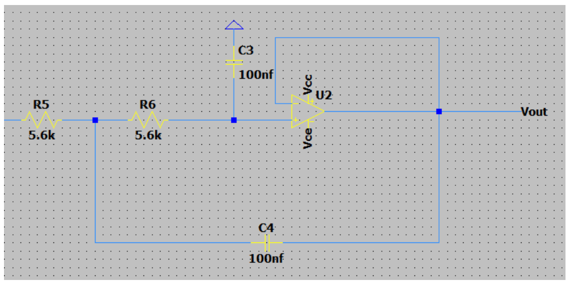
$$|\frac{Rj\omega C}{Rjwc+1}| = \frac{1}{\sqrt{2}}$$

$$\frac{\sqrt{R^2w^2C^2}}{\sqrt{R^2\omega^2C^2+1^2}} = \frac{1}{\sqrt{2}} \text{ so } \quad 2R^2\omega^2\delta^2 = R^2\omega^2C^2+1$$

$$\omega^2 = \frac{1}{R^2C^2} \quad | \quad \omega = \frac{1}{RC}$$

$$2\Pi f = \omega \quad Finally, \; f_C = \frac{1}{2\Pi RC}$$

$$f_C = \frac{1}{2\Pi RC} = \frac{1}{2\Pi 5600\times10^{-6}} = 28.4\text{Hz}$$



Active Low Pass Filter

Components

Capacitors: C3 = 100 nF, C4 = 100 nF

Resistors: R5 = 5.6 kΩ, R6 = 5.6 kΩ

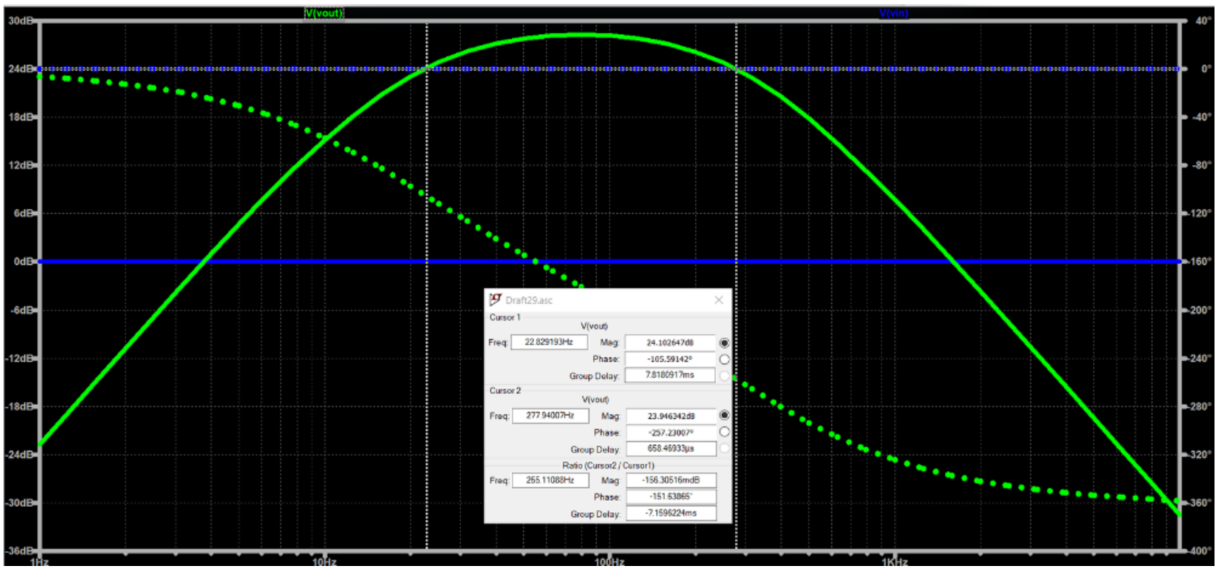
Operational Amplifier: U1B (NE5532)

A low-pass filter (LPF) blocks high-frequency signals and only passes signals below a certain cut-off frequency. The cut-off frequency determines from which frequency the filter starts blocking signals. For a second-order active low-pass filter, the cut-off frequency is calculated by the following formula:

$$f_C = \frac{1}{2\pi RC\sqrt{m \cdot n}}$$

m x n = 1

$$f_C = \frac{1}{2\pi\ 5600 \times 100 \times 10^{-9}} = 284\ \text{Hz}$$

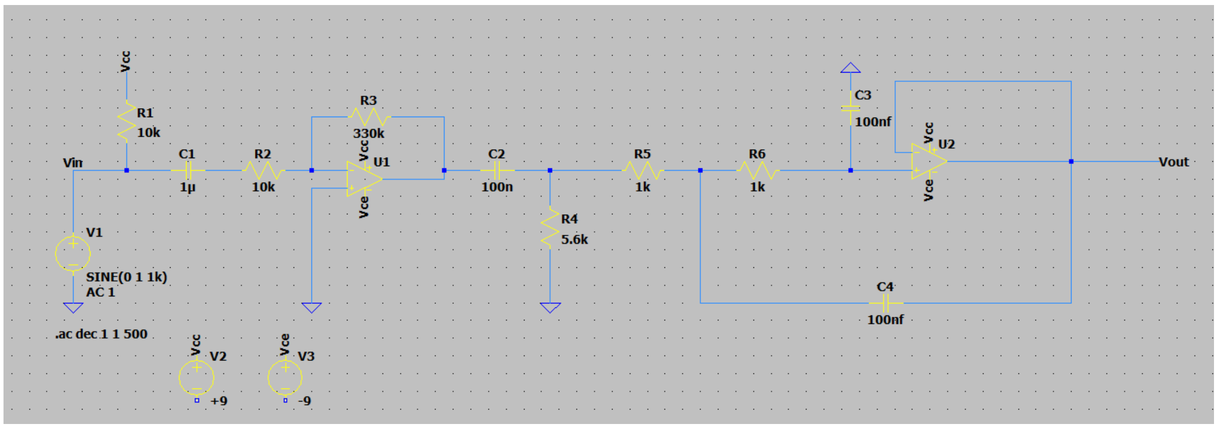


```
Cursor 1
                V(vout)
Freq: 22.829193Hz      Mag:  24.102647dB
                     Phase:  -105.59142°
               Group Delay:  7.8180917ms
Cursor 2
                V(vout)
Freq: 277.94007Hz      Mag:  23.946342dB
                     Phase:  -257.23007°
               Group Delay:  658.46933µs
          Ratio (Cursor2 / Cursor1)
Freq: 255.11088Hz      Mag:  -156.30516mdB
                     Phase:  -151.63865°
               Group Delay:  -7.1596224ms
```

Simulation data of filter design for filtering heart sound

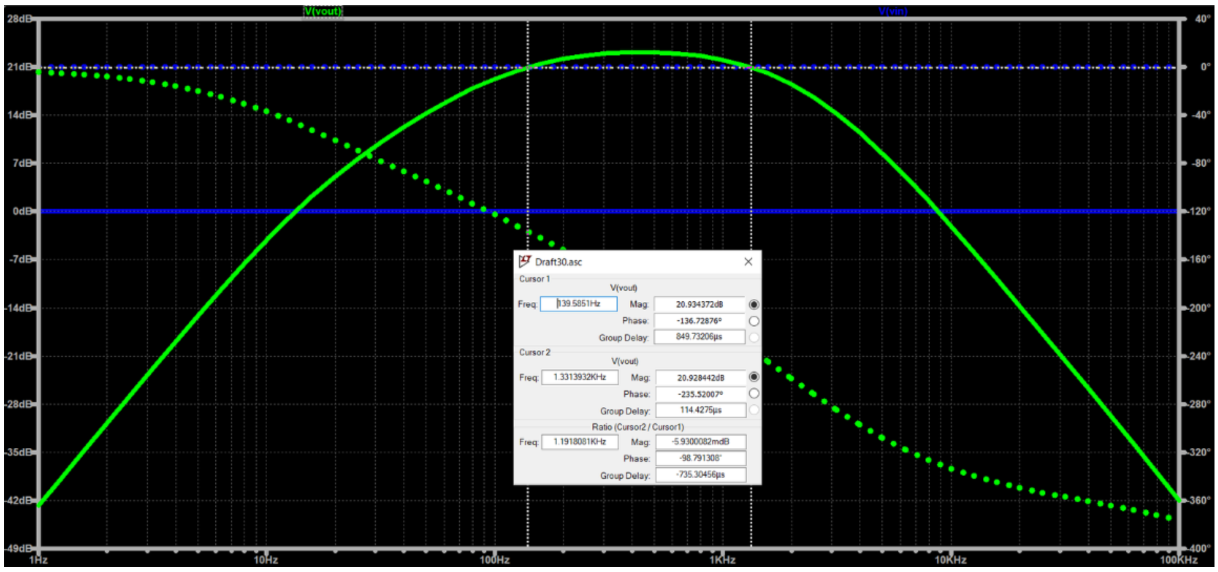
Finally, our circuit becomes a circuit that filters the heart signal (approximately 30-300 Hz). Until this stage, we have used a passive high pass filter and an active low pass circuit, and as seen in the simulation, we have actually created a band pass.

Filter and circuit design for filtering lung sound



LT spice circuit filtering the heart signal

When we examine our circuit in general, we see that we take the input to the circuit with the microphone part and then convert the incoming dc signal to Ac using our C1 capacitor. Then we use an inverting amplifier. In the filtering part, we used a passive and an active filter. First we filtered the sounds below a certain frequency using a high-pass filter, then we filtered above a certain Hz using an active low-pass filter. Thus, we have officially created a band pass and created a filter that passes the 150-1350 Hz range we have determined for the heart. In other words, using the same methods in our heart filter, we changed the frequency band we filtered by changing only some capacitor resistance data.



Cursor 1
V(vout)
Freq: 139.5851Hz Mag: 20.934372dB
Phase: -136.72876°
Group Delay: 849.73206µs

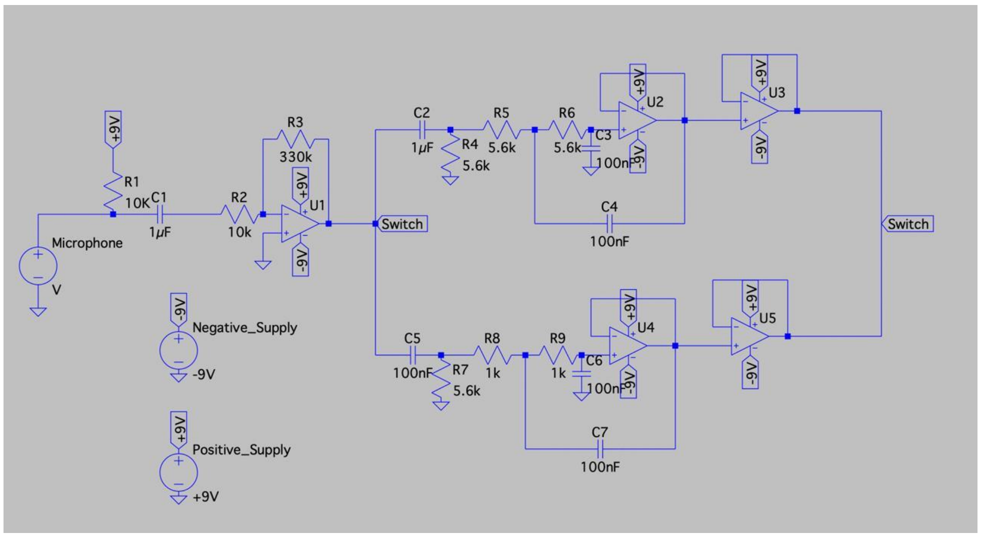
Cursor 2
V(vout)
Freq: 1.3313932KHz Mag: 20.928442dB
Phase: -235.52007°
Group Delay: 114.4275µs

Ratio (Cursor2 / Cursor1)
Freq: 1.1918081KHz Mag: -5.9300082mdB
Phase: -98.791308°
Group Delay: -735.30456µs

Simulation data of filter design for filtering lung sound

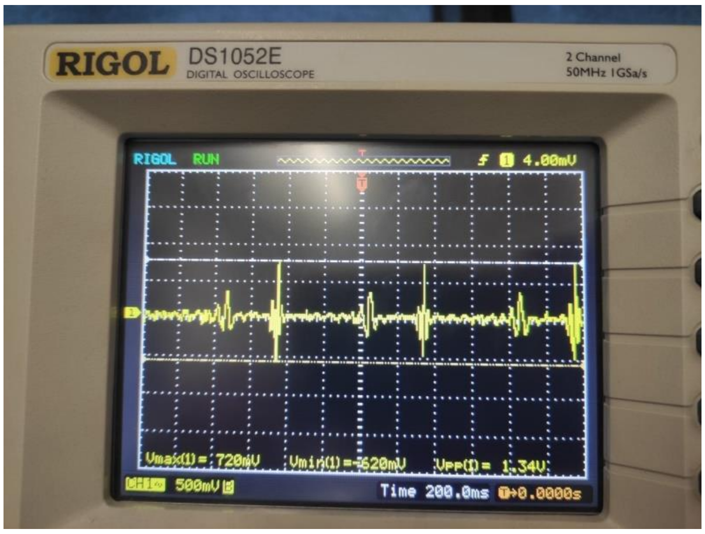
When we examine our simulation, it is seen that we put our cursors on the cutoff frequencies and when we examine the Hz range at these points, it is seen that our values are approximately between 150 - 1350Hz.

Combining Circuits and Product-Oriented Work Section



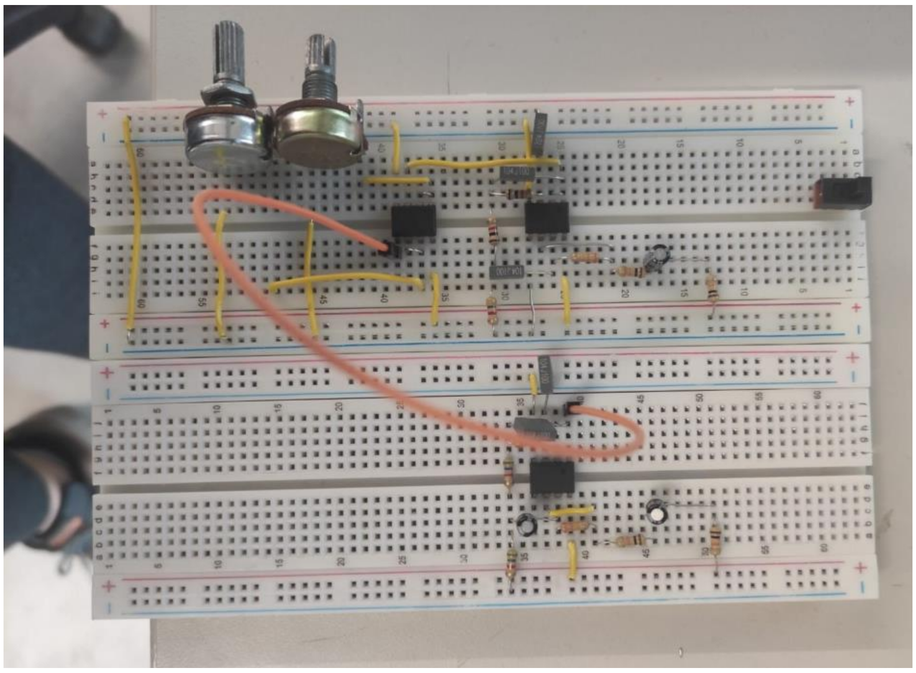
LTspice circuit design for both lung and heart

Above you can see our circuit for both the lung and the heart combined. The product design is valid for both purposes from now on. At any time, it filters the frequency range determined for the heart with the switch, and at any time it filtered the sound of the lung and allowed us to hear it.



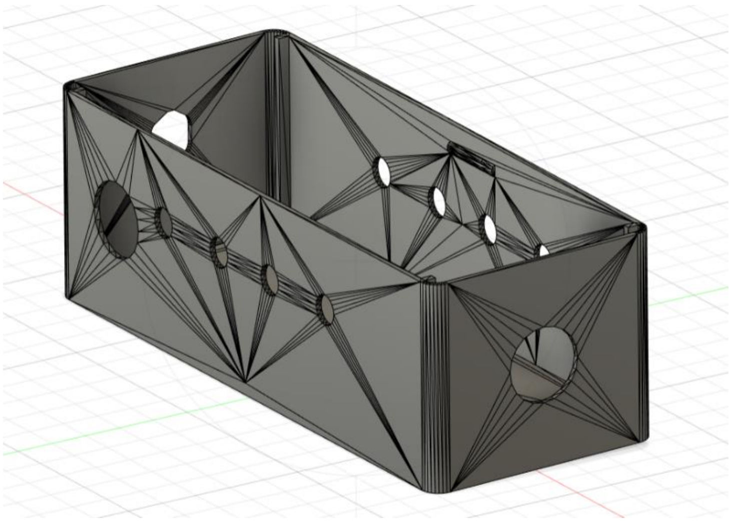
Examination of the Heart Signal on an Oscilloscope

https://www.youtube.com/watch?v=Wa070-HIovo&list=PLnEw4cW5Q_6y40JV8Ijd2BAXzhUBYx92V&index=3

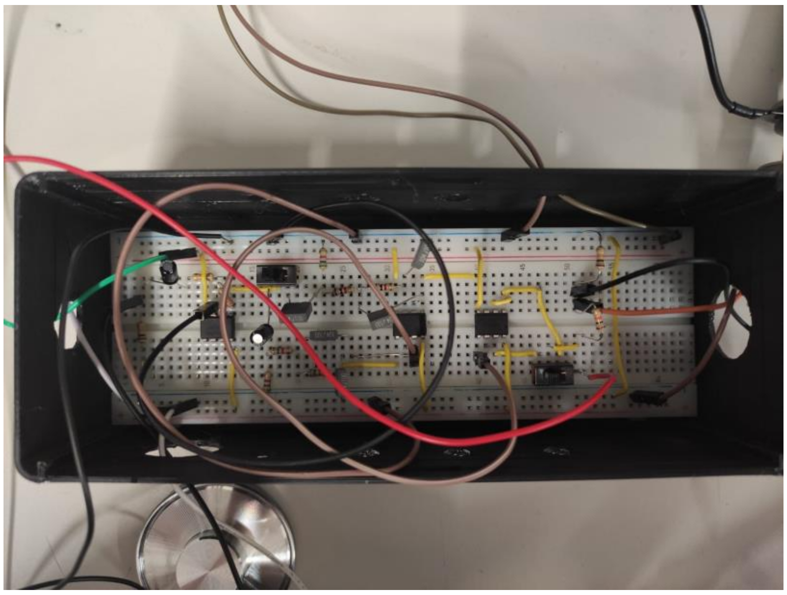


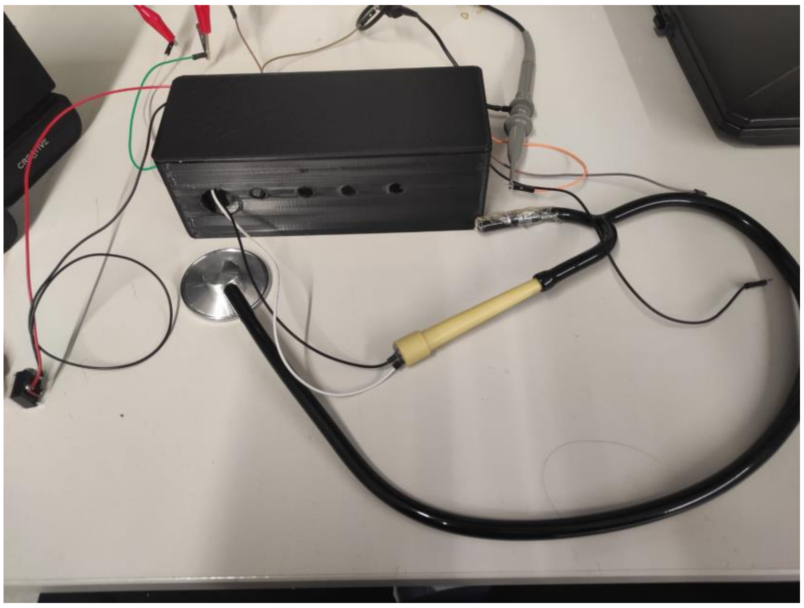
Final version of our circuit on breadboard

You can see the breadboard form above, In order to make this product more user-friendly, we have attached a jack and headphones or speaker to the end of the circuit. In this way, we can now hear the sound of the heart and lung as we can see it on the oscilloscope. This product we designed has a wide range of uses in medical applications. In fact, the work we have done so far is the simulation theoretical and scientific studies of an electronic stethoscope. By analyzing heart and lung sounds separately, it can help doctors diagnose diseases. It can also be used in tele-medicine systems for digital processing and remote transmission of biological signals. It can be used in performance monitoring of athletes, to monitor respiratory rhythms and heartbeats, or to diagnose respiratory disorders such as sleep apnea.

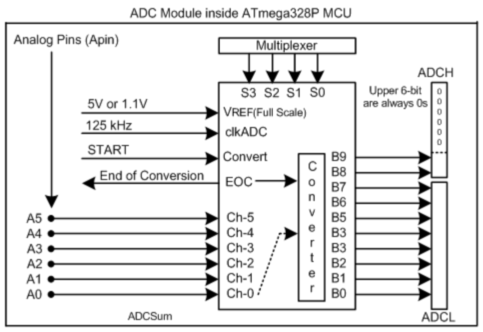


Box design for circuit design (Fusion 360)


The Final Version of our Circuit in the Box


Final Product

Software Part



ADC Module Inside Arduino Uno

In order to achieve the analog-to-digital converting process (ADC), it was necessary to decide on an appropriate board. This board was needed to get the analog signal as input and then transfer the output signal to a computer. The possible boards that are planned to be used by our team are ESP32 and Arduino boards. While deciding on which board we were going to use, the main criteria were the ADC resolution rate and the processor speed.

Figure 2-Arduino Uno Board

Figure 1-ESP32 Board

While the ADC resolution rate of the Arduino Uno board was 10-bit, the ESP32's was 12-bit. The main gap was in their processor speed rates. While an Arduino Uno board had a 16 MHz processor frequency, while this number was 240 MHz for an ESP32 board.

	ADC Resolution	**Processor Frequency**
Arduino Uno	10-bit	16 MHz
ESP32	12-bit	240 MHz

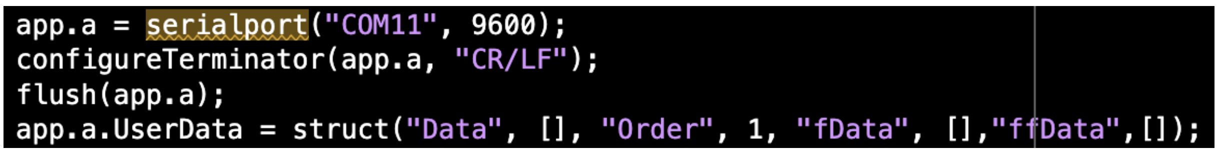
Due to the possibility of burning a board that we have chosen, even though the ESP32 boards could operate faster, the Arduino Uno board is chosen in order to minimize the cost.

The next step in choosing an appropriate board was connecting this board to the constructed circuit. One little issue with connecting the board to the circuit was the output voltage was too high (~ ±8V) for the Arduino Uno board's standard operating voltage (5V). To decrease the output voltage by half, two 10kΩ resistors are used as voltage dividers.

Another main problem was that the Arduino Uno was not able to process negative voltage levels. The only plot that can be seen on the computer would be triangles since it cannot read the negative voltage values. It is thought to increase the buffer op-amp's reference voltage level. Since the reference voltage level was zero in our circuit, the output voltage was ±2.5V - ±3.5V after the voltage divider. It was also needed to give a DC offset to the output voltage so that Arduino could read the whole values.

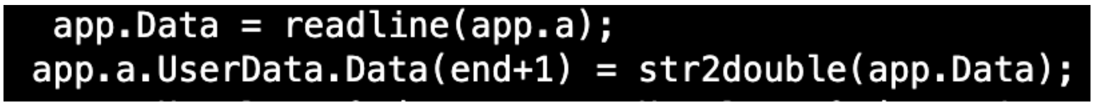
The next step was to display the digital signal on a graphical user interface (GUI). The two possible options were choosing MatLAB® or Python as the language of the signal process. Due to having a built-in app designer option, and having a lot of previous examples, it is decided to use MatLAB®.

The first step on MatLAB® was to get the information from Arduino IDE's serial port. In order to make a connection between MatLAB and Arduino IDE, several functions are created. The first step was to use serialport from the Arduino IDE. The first line of the code identifies the Arduino uno connected port as "COM11", and clarifies the baud rate that is used to get the information from the circuit.



```
app.a = serialport("COM11", 9600);
configureTerminator(app.a, "CR/LF");
flush(app.a);
app.a.UserData = struct("Data", [], "Order", 1, "fData", [],"ffData",[]);
```

Then, the information that comes from the Arduino Uno is read by the readline function.



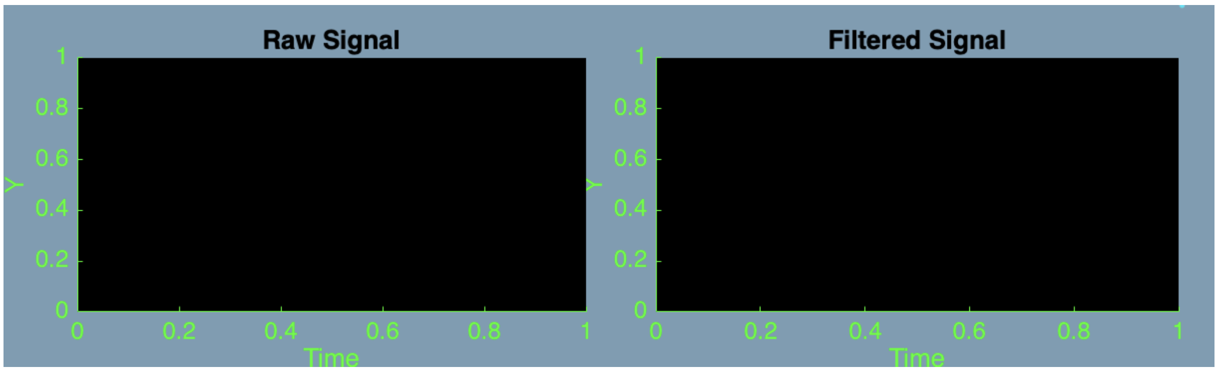
```
  app.Data = readline(app.a);
 app.a.UserData.Data(end+1) = str2double(app.Data);
```

After that, the information that is read must be stored on the app.a.UserData construct to perform the required process such as filtering, graphically visualizing, and performing BPM and RPM calculations.

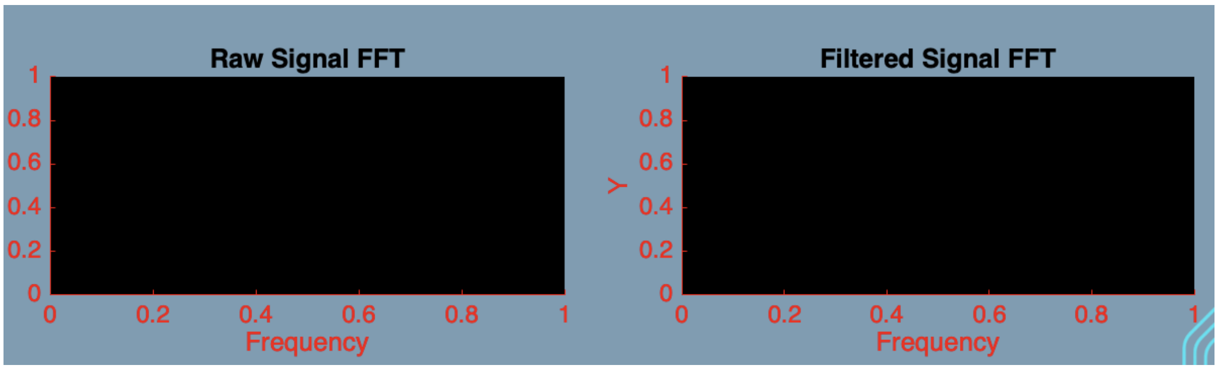
```
app.a.UserData.ffData = filter(app.coefficient_1, app.coefficient_2, app.a.UserData.Data(1:end));
```

One of the most important decisions was choosing the right sampling frequency so as not to lose any information that comes from the Arduino. According to Oktivasari et al. (2020), the frequency of the heartbeat is in the range of 20-200 Hz. This range becomes 300-1500 Hz for the lung (Reichert et al., 2008). According to the sampling theorem, the minimum sampling frequency must be equal to twice the original signal. So, the minimum sampling frequency must be 3000 Hz since the maximum original frequency is 1500 Hz (lung). However, in order to perform better sampling, it is recommended to use a sampling frequency that is eight to ten times higher than the original signal frequency. Therefore, the sampling frequency is chosen as 4000 Hz in the properties part of the MatLAB code.

After being ready to get the first signals from the Arduino, other application codes are started to be written, such as filtering and plotting the Fast Fourier Transform (FFT) graphs. In order to be able to observe both the raw and filtered signals of the circuit, two plot boxes are added to the GUI.



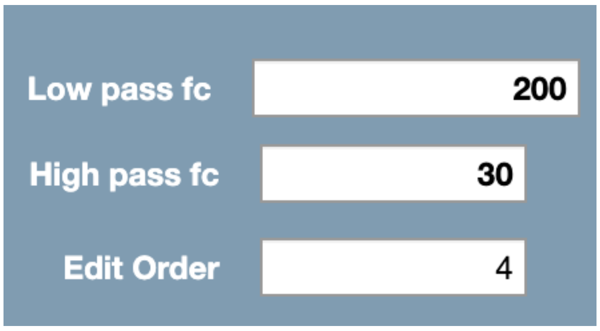
We visualized the data we received with Raw Signal on the Amplitude/Time axis. With Filtered Signa, we used it to see the data we wanted more clearly by filtering in the frequency range we wanted.



While we better analyze the characteristics of our circuit with the Raw Signal FFT graph. In addition, we have facilitated the calculation of bpm and rpm by transferring our data from the time domain to the frequency domain. However, with the Filtered Signal FFT graph, we aimed to observe the Fast Fourier transform of the desired frequency range more clearly.

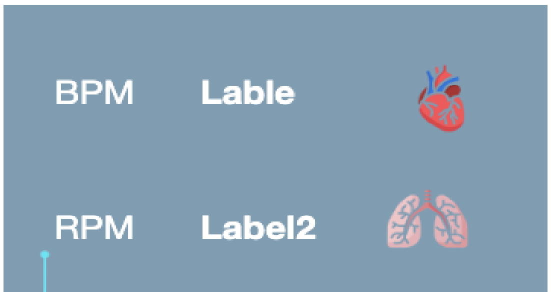
With the Start and Stop buttons, we were able to stop at any time while starting data extraction and analysis whenever we wanted. In this way, both lung and heart signals were examined in more detail, allowing the necessary diagno

sis to be made.



Low pass fc	200
High pass fc	30
Edit Order	4

With the Start and Stop buttons, we were able to stop at any time while starting data extraction and analysis whenever we wanted. In this way, both lung and heart signals were examined in more detail, allowing the necessary diagnosis to be made.



BPM	Lable
RPM	Label2

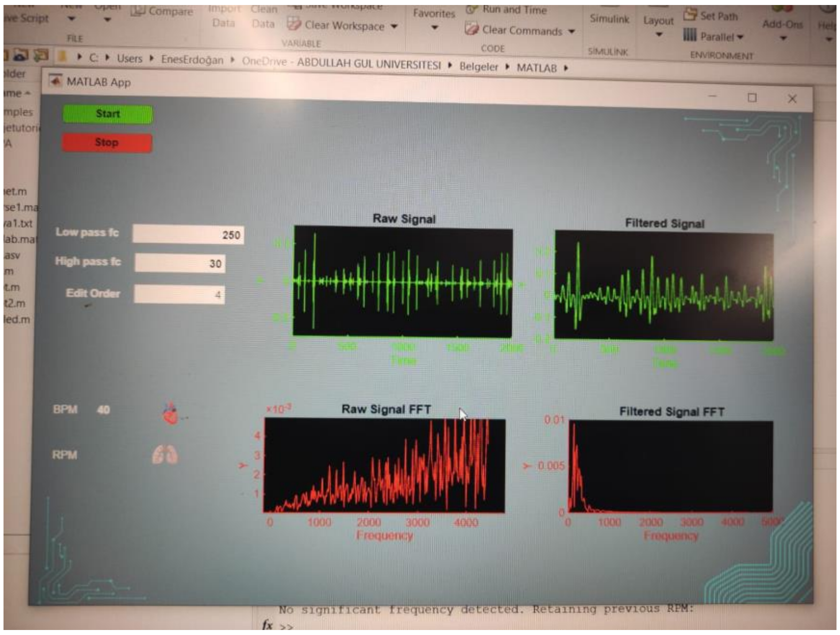
We wanted to calculate the number of heartbeats and breaths per minute with BPM (Beats per minute) and RPM (Revolutions Per Minute) calculations on FFT. I would like to point out that I have trouble determining an exact value when calculating a range with ease. This is mainly due to the fact that the sampling frequency of the program is lower than the Nyquist frequency level.

Result and Discussion

After various upgrades, we moved on to the filter stage. The filter stage consisted of a passive high pass filter and an active low pass filter; the stage was to create a band gap by cutting low frequency noise and high frequency disturbances and we got an output from our circuit

The written code was quite successful in the aspects of transforming the analog input signal to a digital signal, sampling and holding the input signal and displaying the signal as both raw and filtered to the GUI. As the stethoscope senses the heartbeat and lung sounds, the written program was able to display the signals on the added graphs with only a one-second delay rate. The GUI was able to get any value from the consumer to arrange the desired filter with the desired cutoff frequencies. The consumers are also able to choose the order of the desired filter, which allows them to see the output's behavior in different order levels.

Even though the written program was able to satisfy most of the requirements, it had several problems with the sampling frequency, which was realized during the demo section of the SAPA project. The issue was called aliasing. This aliasing problem is caused when the sampling frequency is not as high as twice the input signal. In other words, the program's sampling frequency was lower than the Nyquist frequency level.


Low pass fc 250
High pass fc 30
Edit Order 4
BPM 40
RPM

```
No significant frequency detected. Retaining previous RPM:
```

Figure 3-Aliasing Issue on the Project

Another issue was not being able to determine constant BPM and RPM values. The BPM and the RPM values were not stable enough to be observed. The BPM value was constantly changing between ~85-110. The reason for not being able to get constant values was again caused by the aliasing issue. The program was getting the wrong information between the two impulses of the heartbeat since there was a sinusoidal noise due to the aliasing problem.

References

- https://electronics.stackexchange.com/questions/684575/2nd-order-butterworth-filter-design-how-to-find-m-and-n-values
- https://web.ece.ucsb.edu/Faculty/rodwell/Classes/ece2c/labs/Lab1b-2C2007.pdf
- https://uknowledge.uky.edu/cgi/viewcontent.cgi?article=1060&context=internalmedicine_facpub
- https://www.researchgate.net/figure/Illustrates-the-PCG-signal-including-heart-sounds-S1-S2-S3-S4-1-2-Measured-average_fig2_262806530
- https://clinicalcasereportsjournal.com/article/1000046/analysis-of-the-four-heart-sounds-statistical-study-and-spectro-temporal-characteristics
- D. Ibrahim and A. Davies, "The Evolution of Digital Signal Processors," 2019 6th IEEE History of Electrotechnology Conference (HISTELCON), Glasgow, UK, 2019, pp. 25-29, doi: 10.1109/HISTELCON47851.2019.9040130.
- https://docs.arduino.cc/tutorials/uno-r4-minima/adc-resolution/
- https://en.wikipedia.org/wiki/List_of_Arduino_boards_and_compatible_systems
- https://en.wikipedia.org/wiki/ESP32
- P, O., F, H., A, H. S., R, R., & S, S. (2020). A Real-Time Heart Rate Signal Detection using an Electronic Stethoscope with Labview. Journal of biomedical physics & engineering, 10(3), 375–382. https://doi.org/10.31661/jbpe.v0i0.1183
- Reichert, S., Gass, R., Brandt, C., & Andrès, E. (2008). Analysis of respiratory sounds: state of the art. Clinical medicine. Circulatory, respiratory and pulmonary medicine, 2, 45–58. https://doi.org/10.4137/ccrpm.s530

Attaches

- https://github.com/EnesErdogans/SAPA
- https://youtube.com/playlist?list=PLnEw4cW5Q_6y40JV8Ijd2BAXzhUBYx92V&si=JCV0IcpCBlXsJw3R