SAPA CAPSULE PROJECT

ELECTRONIC STETHOSCOPE

PROJECT TEAM

Sude Nur GÖREZ
Yusuf YILDIZ
Yusuf Taha GÜMÜŞ

Table of Contents

Chapter 1

Introduction

1.1 Objective

The aim of this project is to obtain the sound coming from the heart and lung system, convert the analog sound signal to a digital signal using ARDUINO, analyze the behavior of the respiratory rate (number of breaths per minute) and heart rate (number of heart beats per minute), determine abnormal conditions by observing the graph of the signal, which is the heart and respiratory sound, and create an electronic stethoscope system used to amplify and filter the signals. In this way, the heart and respiratory sounds will be heard clearly and distinctly, and ease of use and correct resolution will be provided for modern clinicians.

1.2 Background

Stethoscopes have been used to listen for bodily noises and diagnose illnesses for more than 200 years. [1] The initial purpose of stethoscopes was to listen to a patient's chest sounds, assess the respiratory and cardiovascular systems, and assess the trachea and bronchial tree's ability to operate as an airway. [2] Sounds from the heart and lungs vibrate the diaphragm as it comes into touch with the patient's chest wall. These sounds then travel through the cavity and up the hollow tube to the earpiece as sound waves. With the introduction of digital stethoscopes, doctors could now remotely monitor their patients' cardiorespiratory health and do digital remote auscultations to evaluate them outside of the clinic. Characterizing the frequency response of various digital stethoscope setups is crucial to boosting the acceptance of a hybrid diagnosis between in-person and distant auscultations. [3] Conversely, digital stethoscopes give physicians the ability to perform a more thorough study by precisely recording and processing cardiac sounds. This technique detects even minute changes in heart sounds, making it an essential tool for early disease identification. Using this comprehensive information, physicians treat patients with more individualized and efficient techniques. [2] The limits of the earlier stethoscope possibilities have been overcome by incorporating new medical engineering solutions into hectic clinical and educational contexts, as healthcare technology continues to advance over time. It is hoped that as current technology advances, more precise and efficient medical equipment will be developed that can do auscultation to acquire a diagnosis, which will be advantageous to patients and medical professionals alike.

Chapter 2

Circuit Design

In this project, in the circuit established to capture pure heart and lung sounds, 22 kΩ (2), 10 nF (2) components were used for the high pass filter 30 Hz and 1K, 4.7 kΩ, 1 µF, 10 µF components were used for the low pass filter 720 Hz. A single filter between 30-720 Hz was created by taking the common frequency ranges of heart and lung sounds. (1 kΩ (2), 18 kΩ (2)) resistors were used to amplify the incoming signal and the total gain was equal to (18 times). Therefore, all filter topologies were designed based on Salley Key Filter and filter circuits were drawn in LTSpice simulation software. In addition, Arduino UNO was used to convert the analog signal coming from the circuit output to digital signal and to transmit this data to MATLAB software. Since there are negative values in the received analog signal, the voltage range that the Arduino's analog pin can read is between 0-5V, so the voltage range is regulated by giving -2.15 V DC voltage to the Arduino's ground line. Finally, thanks to the data transferred with Arduino, a GUI was designed in MATLAB's AppDesigner application where the cut-off frequencies can be adjusted, and the graphs of the captured signal, the filtered signal and their FFTs were drawn.

2.1 Filters

In order to extract the appropriate frequency bands, the system uses different topologies of feedback filters, and delivering, at the same time, good signal conditioning. Filters are categorized as follows:

2.1.1 Low Pass Filter

The Sallen-Key active low pass filter is a low cost, low noise filter design suitable for a wide range of applications. This filter plays an important role in noise processing and filtering in electronic circuits.

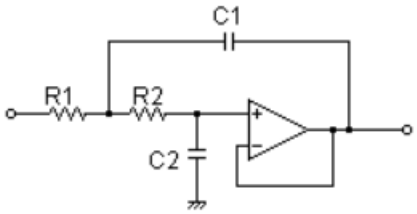

Figure 2.1.1: Sallen Key Active Low Pass Filter Design

- Cutoff Frequency: (f_c = 720 Hz)
- Components: Resistors (22 kΩ), Capacitors (10 nF)
- Purpose: Removes high-frequency noise while preserving critical acoustic signals.

Transfer function:

$$\frac{V_{out}(s)}{V_{in}(s)} = \frac{\frac{1}{R_1 C_1 R_2 C_2}}{s^2 + s\left(\frac{1}{R_2 C_1} + \frac{1}{R_1 C_1}\right) + \frac{1}{R_1 C_1 R_2 C_2}}$$

Cut off frequency formula for the Sallen key filter:

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

The Sallen-Key filter provides a second order Butterworth type regulation. This circuit contains 10nF (2) capacitors and 22kΩ (2) ohm resistors and a LM358 Op-amp. The cut-off frequency is calculated as 720 Hz.

2.1.2 High Pass Filter

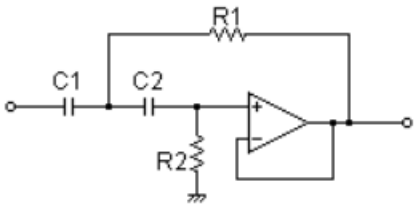

Figure 2.1.2: Sallen Key Active High Pass Filter Design

This circuit contains 1 µF, 10 µF capacitors and 4.7 kΩ, 1 kΩ ohm resistors and a LM358 Op-amp. The cut-off frequency is calculated as 30 Hz.
- Cutoff Frequency: (f_c = 30 Hz)
- Components: Resistors (4.7 kΩ), Capacitors (10 µF)
- Purpose: Eliminates low-frequency interference, such as environmental hums.

Each filter is coupled by buffers so that the signal transitions are not attenuated.
Transfer function:

$$\frac{V_{out}(s)}{V_{in}(s)} = \frac{s^2}{s^2 + s\left(\frac{1}{R_2C_1} + \frac{1}{R_2C_2}\right) + \frac{1}{R_1C_1R_2C_2}}$$

Cut off frequency formula for the Sallen key filter:

$$f_c = \frac{1}{2\pi\sqrt{R_1C_1R_2C_2}}$$

2.2 Inverting Amplifier

The inverting amplifier stage is an important stage for signal amplification. We used the LM358 op-amp due to its advantages over noise-handling and operational characteristics. Resistor values of R_1 and R_2 gain:

$$Gain = -\frac{R_2}{R_1}$$

With this setup, it is possible to amplify weak signals from the microphone sufficiently to allow processing.

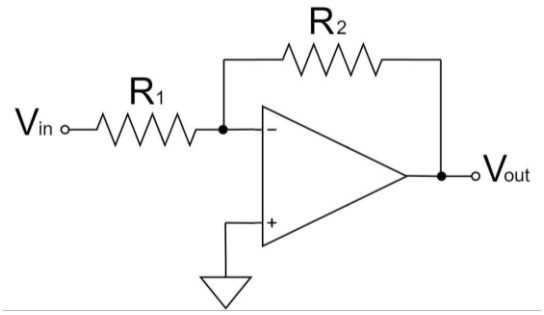

Figure 2.2.1: Inverting Amplifier Design

Chapter 3

Simulation Part

3.1 Filters Simulation

Using LTSpice, simulations verified the effectiveness of the designed circuits in amplifying, filtering, and reducing noise. Results guaranteed the circuit's ability to properly filter out signals over the targeted frequency interval (20 Hz to 1000 Hz), a main requirement for the detection of heart and lung sounds.

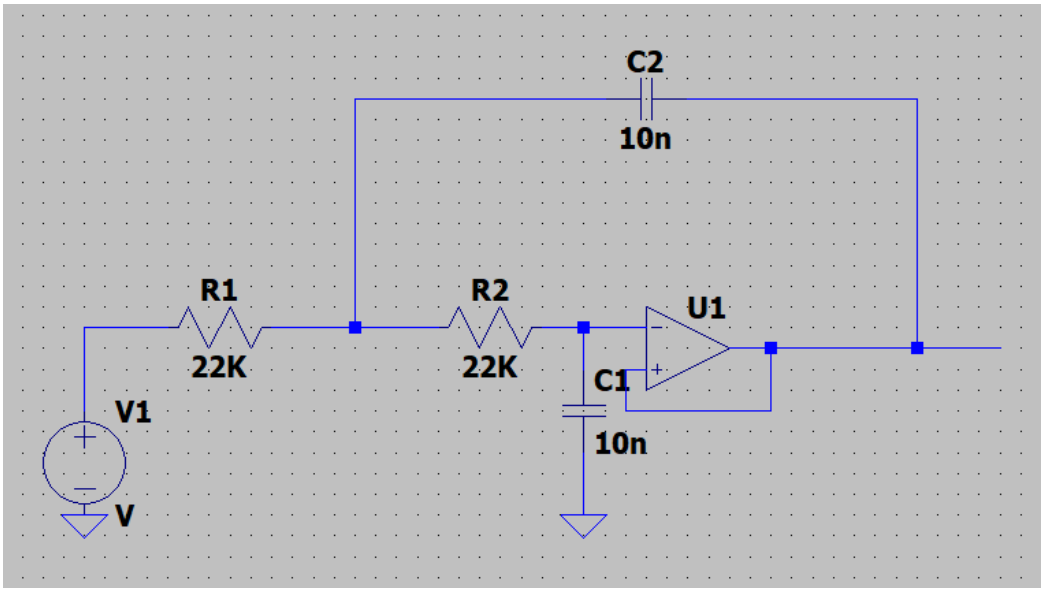

Figure 3.1.1: Low pass filter that has 720 Hz cut off frequency.

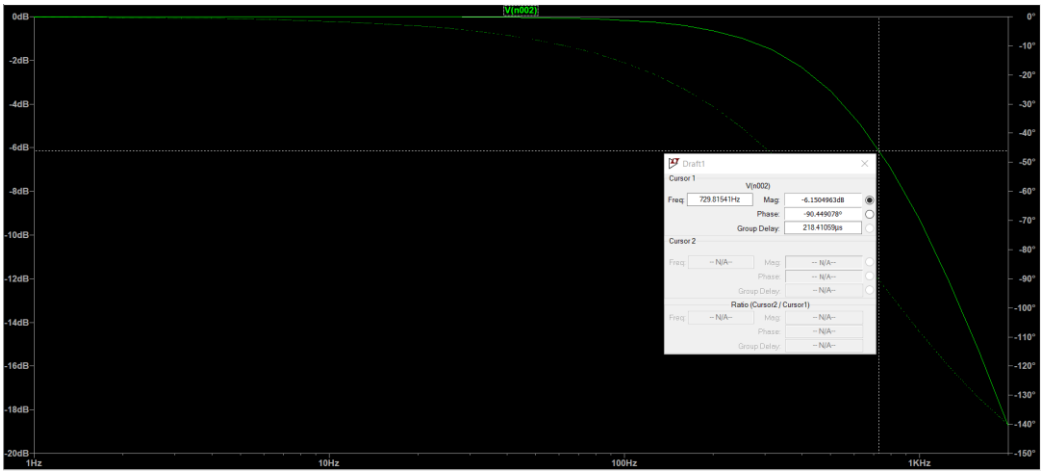

Figure 3.1.2: Bode plot off 730 Hz low pass filter.

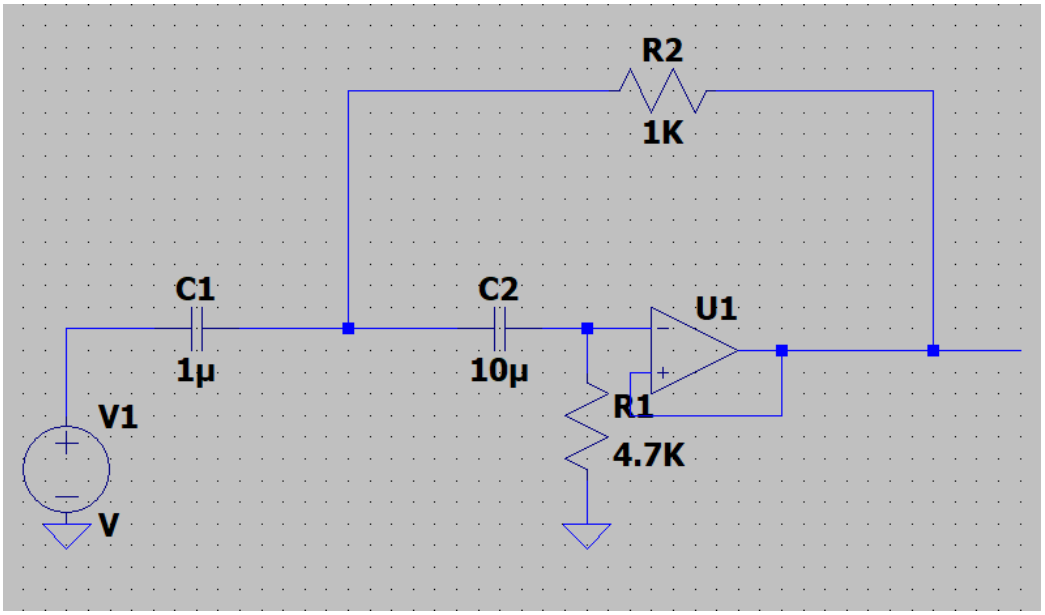

Figure 3.1.3: High pass filter that has 30 Hz cut off frequency.

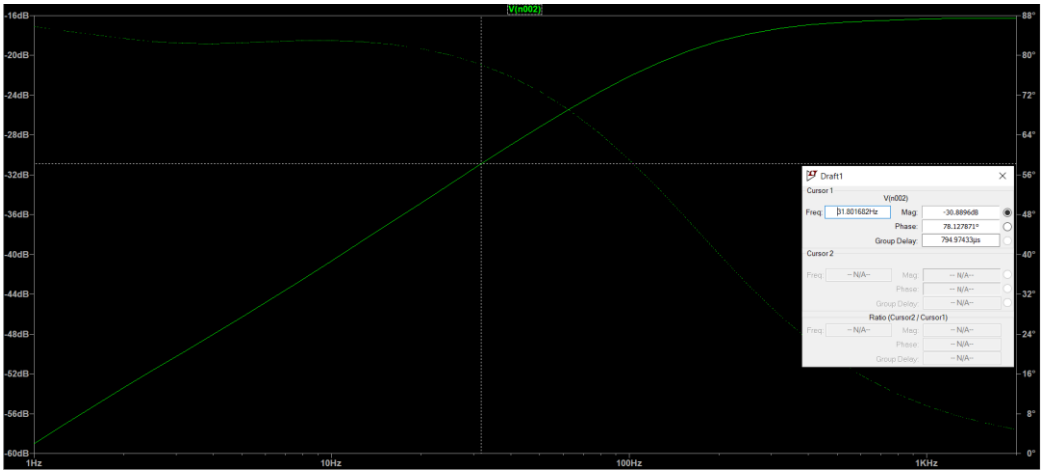

Figure 3.1.4: Bode plot off 30 Hz high pass filter.

Using cursors in LTSpice, the frequency values were checked for correctness. This was an important indicator for the optimized operation of this circuit and especially for the measurement of heart and lung sounds with noise reduction.

3.2 Microphone Circuit Simulation

In the microphone circuit design process, the type of microphone to be used was considered as an electret microphone and depending on the application requirements, one end was considered as anode and the other end as cathode. As in the schematic in the figure, the circuit is established by considering the factors required by the circuit, sensitivity, frequency response, noise levels. Then it is connected to the filters with jumper wires.

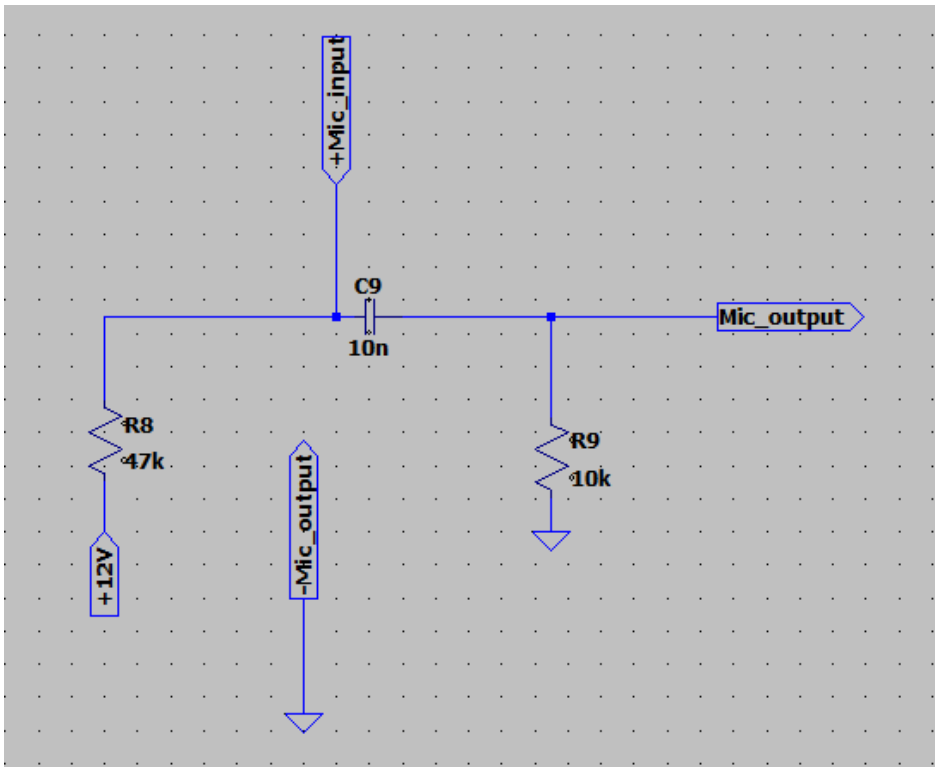

Figure 3.2.1: Microphone circuit diagram

Chapter 4

CIRCUIT IMPLEMENTATION

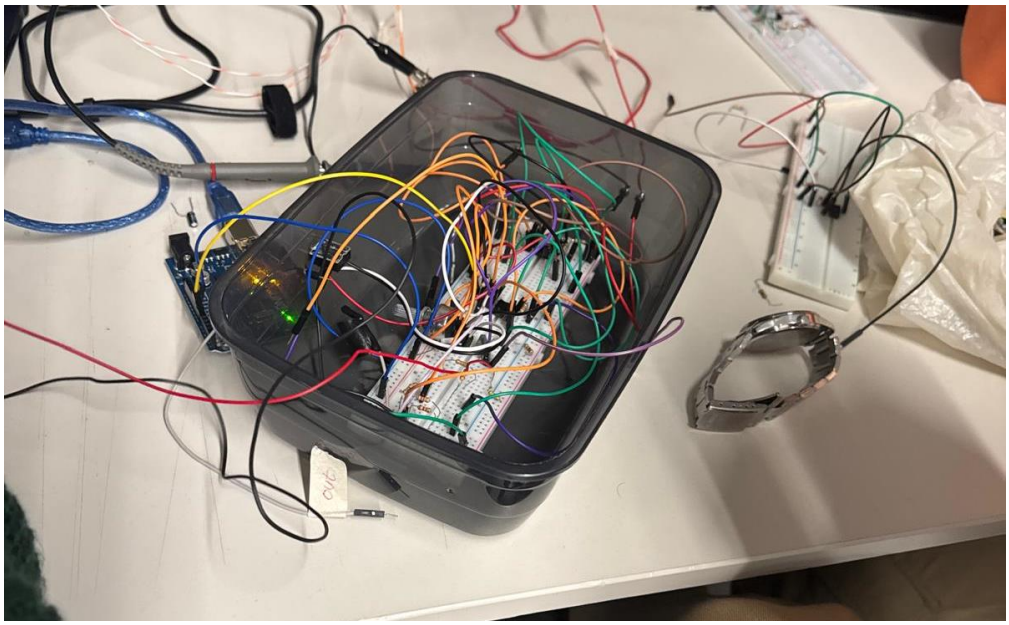

Figure 4.1: Inside of the box model of the stethoscope.

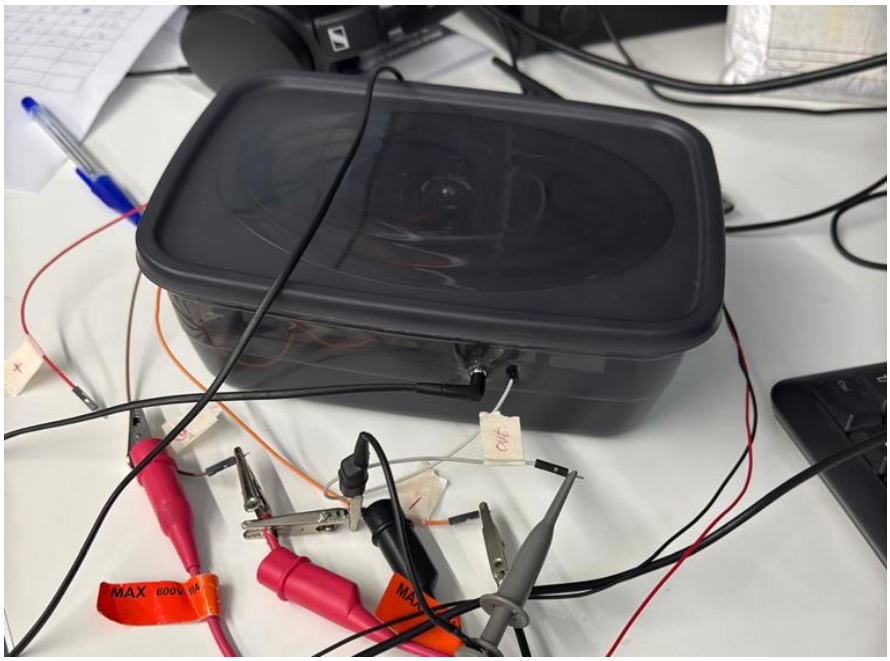

Figure 4.2: Outside of the box model of the stethoscope.

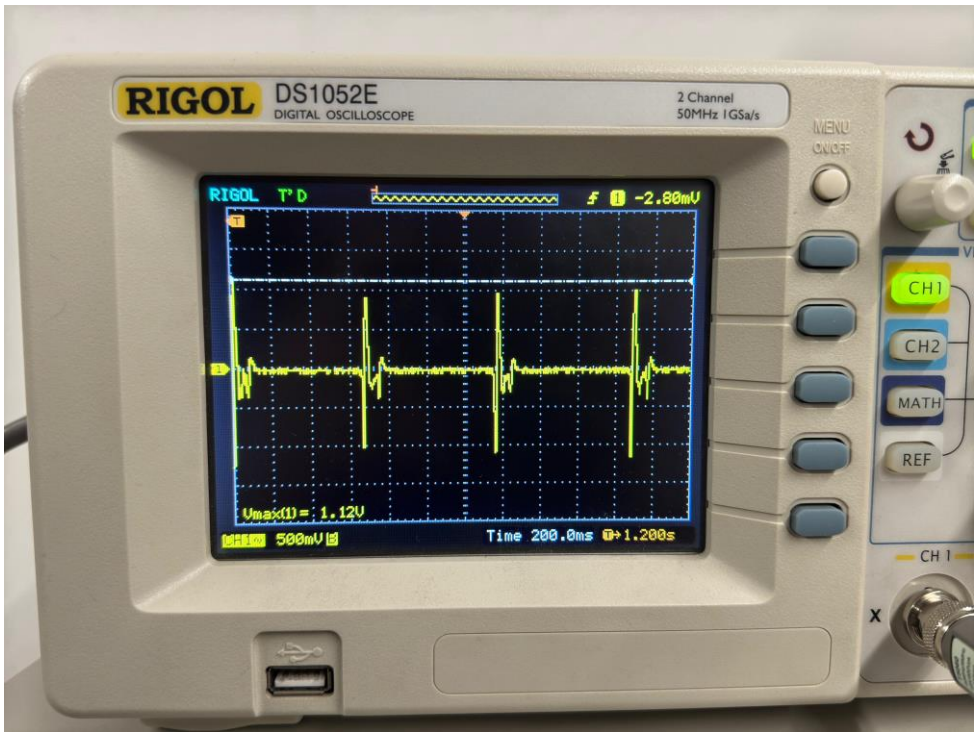

Figure 4.3: The signal of hearth beats on oscilloscope.

Chapter 5

Graphical User Interface (GUI) in MATLAB

The project used MATLAB for its powerful signal processing and GUI design capabilities. MATLAB provided efficient signal filtering, FFT analysis, and dynamic visualization with customizable cutoff ranges. The Application Designer improved usability and interaction with analyzed signals by facilitating interface development.

To facilitate analog voltage readings, the team integrated Arduino into the circuitry. Arduino reliably acquired the raw signal data and seamlessly transferred it to MATLAB for further processing. This integration increased the efficiency of the circuit's data acquisition and signal analysis. The collaboration between MATLAB and Arduino provided a comprehensive approach to evaluating respiratory signals.

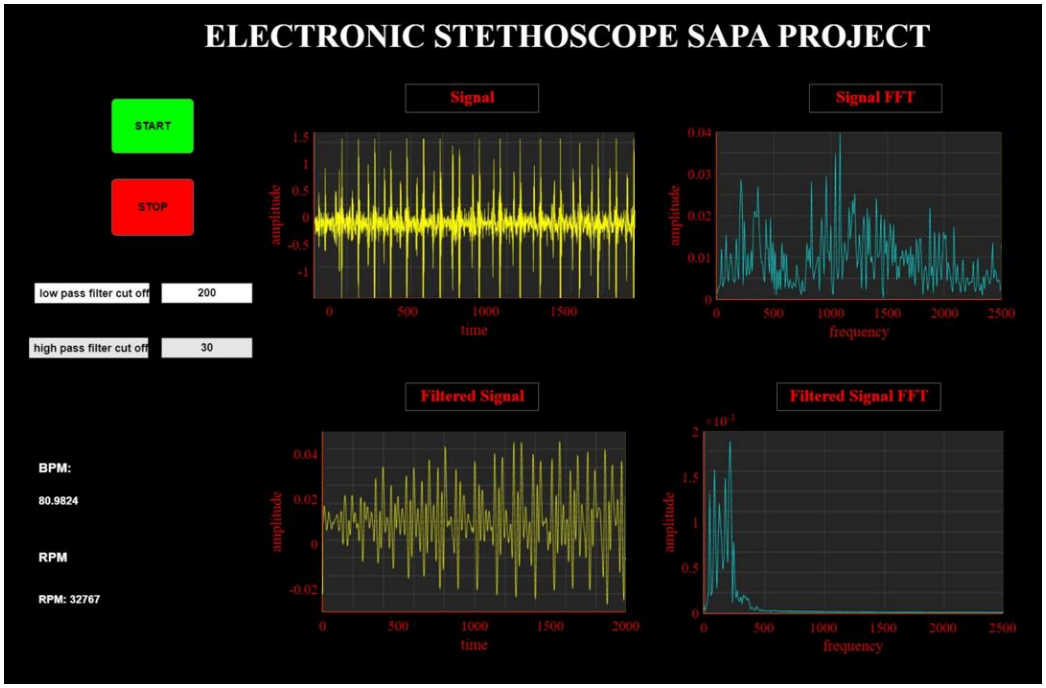

Figure 5.1: Observing the graph of Heartbeat signals by using MATLAB GUI.

GUI Codes

This MATLAB script provides a GUI to visualize, filter, and analyze real-time data captured from an electronic stethoscope. It includes signal processing functions such as high-pass and low-pass filtering, and Fourier Transform analysis to display the frequency components of the signal. The GUI also provides basic diagnostics for medical or mechanical use by calculating BPM (Beats Per Minute) and RPM (Revolutions Per Minute).

```
classdef GUI < matlab.apps.AppBase

    % Properties that correspond to app components
    properties (Access = public)
        UIFigure                        matlab.ui.Figure
        TextArea_4                      matlab.ui.control.TextArea
        TextArea_3                      matlab.ui.control.TextArea
        TextArea_2                      matlab.ui.control.TextArea
        TextArea                        matlab.ui.control.TextArea
        TextAreaLabel                   matlab.ui.control.Label
        SAPAProjectElectronicStethoscopeLabel  matlab.ui.control.Label
        Label2                          matlab.ui.control.Label
        RPMLabel                        matlab.ui.control.Label
        BPMLabel                        matlab.ui.control.Label
        LableLabel                      matlab.ui.control.Label
        lowpassfiltercutoffEditField
matlab.ui.control.NumericEditField
        lowpassfiltercutoffEditFieldLabel  matlab.ui.control.Label
        highpassfiltercutoffEditField
matlab.ui.control.NumericEditField
        highpassfiltercutoffEditFieldLabel  matlab.ui.control.Label
        STOPButton                      matlab.ui.control.Button
        STARTButton                     matlab.ui.control.Button
        UIAxes_4                        matlab.ui.control.UIAxes
        UIAxes_3                        matlab.ui.control.UIAxes
        UIAxes_2                        matlab.ui.control.UIAxes
        UIAxes                          matlab.ui.control.UIAxes
    end


     properties (Access = private)
        value
        f
        f1
        cf1
        cf2
        X1
        X2
        X3
        X4
        serialport
        Fs = 8000;
        fs = 5000;
        fc1 = 30;
        fc2 = 200;
```

```
        T = 1/5000;
        m = 1;
        plotWindow = 2000;
        IsRunning = false;
        Timer
        Data
        ffData
        fdata
        coefficient_1
        coefficient_2
        fc3=35
        bpm
        rpm
        coefficient_3
        coefficient_4
        orderOfFilter=4;
        end

    methods (Access = private)

        function timerCallback = func(app)
            if ~app.IsRunning
            stop(app.Timer);
        end
    end
end

    % Callbacks that handle component events
    methods (Access = private)

        % Code that executes after component creation
        function startupFcn(app)
            app.IsRunning=true;
            app.Timer=timer;
            app.Timer.Period=0.1;
            app.Timer.ExecutionMode='fixedRate';
            app.Timer.TimerFcn=@(~,~)app.timerCallback;
            start(app.Timer);


        end

        % Button pushed function: STARTButton
        function STARTButtonPushed(app, event)
            app.IsRunning = true;
            app.value = serialport("COM3", 9600);
            configureTerminator(app.value, "CR/LF");
            flush(app.value);
            app.value.UserData = struct("Data", [], "Order", 1,
"fData", [],"ffData",[]);
            app.IsRunning = true;

            while app.value.UserData.Order < 100000
                    app.Data = readline(app.value);
```

```
                    app.value.UserData.Data(end+1) =
str2double(app.Data);
                    app.value.UserData.Order =
app.value.UserData.Order + 1;
                if mod(app.value.UserData.Order, 10) == 0
                    configureCallback(app.value, "off");

                    plot(app.UIAxes,
app.value.UserData.Data(max(1,end-app.plotWindow+1):end),'Color','y')
                    [app.coefficient_1, app.coefficient_2] = butter(4,
app.fc3/(app.fs/2), 'high');
                    app.value.UserData.ffData =
filter(app.coefficient_1, app.coefficient_2,
app.value.UserData.Data(1:end));
                    [app.coefficient_3, app.coefficient_4] =
butter(app.orderOfFilter, [app.fc1/(app.fs/2), app.fc2/(app.fs/2)],
'bandpass');

                    app.value.UserData.fData =
filter(app.coefficient_3, app.coefficient_4, app.value.UserData.Data);

                    plot(app.UIAxes_2, app.value.UserData.fData(max(1,
end-app.plotWindow+1):end) * 3,'Color','y');
                    drawnow;

                    configureTerminator(app.value, "CR/LF");

                end
                 if mod(app.value.UserData.Order,500)==0
                     configureCallback(app.value, "off");

                     app.cf1 = fft(app.value.UserData.ffData(end-
499+1:end));
                     app.X2 = abs(app.cf1 / 500); % Use 400 for
normalization
                     app.f = app.fs *
(0:(length(app.cf1)/2))/length(app.cf1); % Adjust frequency axis
                     app.X1 = app.X2(1:length(app.cf1)/2+1);
                     app.X1(2:end-1) = 2 * app.X1(2:end-1);
                     plot(app.UIAxes_3,app.f,app.X1,'Color','c');
                     set(gca,'ylim',[0,10]);

                     app.cf2 = fft(app.value.UserData.fData(end-
499+1:end));
                     app.X3 = abs(app.cf2 / 500); % Use 400 for
normalization
                     app.f1 = app.fs *
(0:(length(app.cf2)/2))/length(app.cf2); % Adjust frequency axis
                     app.X4 = app.X3(1:length(app.cf2)/2+1);
                     app.X4(2:end-1) = 2 * app.X4(2:end-1);
                     plot(app.UIAxes_4, app.f1,app.X4,'Color','c');
                     set(gca,'ylim',[0,15]);

                     grid on;
                     app.m=app.m+500;
```

```
                    configureTerminator(app.value,"CR/LF");
                end
                  if mod(app.value.UserData.Order, 1500) == 0
                    % Measure elapsed time
                   timeAtPointBPM = toc;
                   % Reset the timer immediately for the next
interval


                   % Detect peaks in the data
                   threshold = 0.50; % Adjust this threshold as
needed
                   [peaks, ~] =
findpeaks(app.value.UserData.Data(end-1499+1:end), 'MinPeakHeight',
threshold);
                   numPeaks = numel(peaks); % Count the number of
peaks

                   % Calculate BPM
                   app.bpm = (60 * numPeaks) / timeAtPointBPM; % BPM
calculation
                   disp(int16(app.bpm)); % Display BPM
                   tic;
                   app.LableLabel.Text = num2str(app.bpm);
                  if mod(app.value.UserData.Order, 1500) == 0
    % Measure elapsed time
    timeAtPointBPM = toc;

    % Detect peaks for RPM calculation
    thresholdRPM = 0.01; % Adjust this threshold if necessary
    [peaksRPM, ~] = findpeaks(app.value.UserData.Data(end-1499+1:end),
'MinPeakHeight', thresholdRPM);
    numPeaksRPM = numel(peaksRPM); % Burada numPeaksRPM doğru bir
şekilde hesaplanıyor.

    % Calculate RPM
    app.rpm = (60 * numPeaksRPM) / timeAtPointBPM; % Calculate RPM
based on time elapsed
    disp(int16(app.rpm)); % Display RPM
    app.Label2.Text = ['RPM: ', num2str(int16(app.rpm))]; % Update
Label2 text
    tic; % Restart the timer
end

                   end
               end
        end

        % Button pushed function: STOPButton
        function STOPButtonPushed(app, event)
            app.IsRunning=false;
            delete(app.value);
        end

        % Value changed function: lowpassfiltercutoffEditField
```

```
        function lowpassfiltercutoffEditFieldValueChanged(app, event)
            b=app.lowpassfiltercutoffEditField.Value;
            app.fc2=b;
        end

        % Value changed function: highpassfiltercutoffEditField
        function highpassfiltercutoffEditFieldValueChanged(app, event)
             c= app.highpassfiltercutoffEditField.Value;
             app.fc1=c;
        end

        % Callback function
        function ORDEROFFILTEREditFieldValueChanged(app, event)
            j = app.ORDEROFFILTEREditField.Value;
            app.orderOfFilter=j;
        end
    end

    % Component initialization
    methods (Access = private)

        % Create UIFigure and components
        function createComponents(app)

            % Create UIFigure and hide until all components are
created
            app.UIFigure = uifigure('Visible', 'off');
            app.UIFigure.Color = [0 0 0];
            app.UIFigure.Position = [100 100 1161 780];
            app.UIFigure.Name = 'MATLAB App';

            % Create UIAxes
            app.UIAxes = uiaxes(app.UIFigure);
            title(app.UIAxes, 'SIGNAL')
            xlabel(app.UIAxes, 'time')
            ylabel(app.UIAxes, 'amplitude')
            zlabel(app.UIAxes, 'Z')
            app.UIAxes.FontName = 'Times New Roman';
            app.UIAxes.GridLineWidth = 1;
            app.UIAxes.XColor = [1 0 0];
            app.UIAxes.YColor = [1 0 0];
            app.UIAxes.ZColor = [0 0 0];
            app.UIAxes.Color = [0.149 0.149 0.149];
            app.UIAxes.GridColor = [1 1 0];
            app.UIAxes.MinorGridColor = [0.8 0.8 0.8];
            app.UIAxes.XGrid = 'on';
            app.UIAxes.YGrid = 'on';
            app.UIAxes.ZGrid = 'on';
            app.UIAxes.ColorOrder = [0 0.447058823529412
0.741176470588235;0.850980392156863 0.325490196078431
0.0980392156862745;0.929411764705882 0.694117647058824
0.125490196078431;0.494117647058824 0.184313725490196
0.556862745098039;0.466666666666667 0.674509803921569
0.188235294117647;0.301960784313725 0.749019607843137
```

```
0.929411764705882;0.635294117647059 0.0784313725490196
0.184313725490196];
            app.UIAxes.FontSize = 15;
            colormap(app.UIAxes, 'turbo')
            app.UIAxes.Position = [295 398 412 251];

            % Create UIAxes_2
            app.UIAxes_2 = uiaxes(app.UIFigure);
            title(app.UIAxes_2, 'FILTERED SIGNAL')
            xlabel(app.UIAxes_2, 'time')
            ylabel(app.UIAxes_2, 'amplitude')
            zlabel(app.UIAxes_2, 'Z')
            app.UIAxes_2.FontName = 'Times New Roman';
            app.UIAxes_2.GridLineWidth = 1;
            app.UIAxes_2.XColor = [1 0 0];
            app.UIAxes_2.YColor = [1 0 0];
            app.UIAxes_2.Color = [0.149 0.149 0.149];
            app.UIAxes_2.GridColor = [1 1 0];
            app.UIAxes_2.XGrid = 'on';
            app.UIAxes_2.YGrid = 'on';
            app.UIAxes_2.ZGrid = 'on';
            app.UIAxes_2.FontSize = 15;
            app.UIAxes_2.Position = [297 54 410 267];

            % Create UIAxes_3
            app.UIAxes_3 = uiaxes(app.UIFigure);
            title(app.UIAxes_3, 'SIGNAL FFT')
            xlabel(app.UIAxes_3, 'frequency')
            ylabel(app.UIAxes_3, 'amplitude')
            zlabel(app.UIAxes_3, 'Z')
            app.UIAxes_3.FontName = 'Times New Roman';
            app.UIAxes_3.GridLineWidth = 1;
            app.UIAxes_3.XColor = [1 0 0];
            app.UIAxes_3.YColor = [1 0 0];
            app.UIAxes_3.Color = [0.149 0.149 0.149];
            app.UIAxes_3.GridColor = [1 1 0];
            app.UIAxes_3.MinorGridColor = [0.8 0.8 0.8];
            app.UIAxes_3.XGrid = 'on';
            app.UIAxes_3.YGrid = 'on';
            app.UIAxes_3.ZGrid = 'on';
            app.UIAxes_3.FontSize = 15;
            app.UIAxes_3.Position = [730 396 398 253];

            % Create UIAxes_4
            app.UIAxes_4 = uiaxes(app.UIFigure);
            title(app.UIAxes_4, 'FILTERED SIGNAL FFT')
            xlabel(app.UIAxes_4, 'frequency')
            ylabel(app.UIAxes_4, 'amplitude')
            zlabel(app.UIAxes_4, 'Z')
            app.UIAxes_4.FontName = 'Times New Roman';
            app.UIAxes_4.GridLineWidth = 1;
            app.UIAxes_4.XColor = [1 0 0];
            app.UIAxes_4.YColor = [1 0 0];
            app.UIAxes_4.Color = [0.149 0.149 0.149];
            app.UIAxes_4.GridColor = [1 1 0];
```

```
            app.UIAxes_4.XGrid = 'on';
            app.UIAxes_4.YGrid = 'on';
            app.UIAxes_4.ZGrid = 'on';
            app.UIAxes_4.FontSize = 15;
            app.UIAxes_4.Position = [731 52 398 269];

            % Create STARTButton
            app.STARTButton = uibutton(app.UIFigure, 'push');
            app.STARTButton.ButtonPushedFcn = createCallbackFcn(app,
@STARTButtonPushed, true);
            app.STARTButton.BackgroundColor = [0 1 0];
            app.STARTButton.FontWeight = 'bold';
            app.STARTButton.Position = [127 601 90 60];
            app.STARTButton.Text = 'START';

            % Create STOPButton
            app.STOPButton = uibutton(app.UIFigure, 'push');
            app.STOPButton.ButtonPushedFcn = createCallbackFcn(app,
@STOPButtonPushed, true);
            app.STOPButton.BackgroundColor = [1 0 0];
            app.STOPButton.FontWeight = 'bold';
            app.STOPButton.Position = [127 511 90 62];
            app.STOPButton.Text = 'STOP';

            % Create highpassfiltercutoffEditFieldLabel
            app.highpassfiltercutoffEditFieldLabel =
uilabel(app.UIFigure);
            app.highpassfiltercutoffEditFieldLabel.BackgroundColor =
[0.902 0.902 0.902];
            app.highpassfiltercutoffEditFieldLabel.HorizontalAlignment
= 'right';
            app.highpassfiltercutoffEditFieldLabel.FontWeight =
'bold';
            app.highpassfiltercutoffEditFieldLabel.Position = [36 377
130 22];
            app.highpassfiltercutoffEditFieldLabel.Text = 'high pass
filter cut off';

            % Create highpassfiltercutoffEditField
            app.highpassfiltercutoffEditField =
uieditfield(app.UIFigure, 'numeric');
            app.highpassfiltercutoffEditField.ValueChangedFcn =
createCallbackFcn(app, @highpassfiltercutoffEditFieldValueChanged,
true);
            app.highpassfiltercutoffEditField.HorizontalAlignment =
'center';
            app.highpassfiltercutoffEditField.FontWeight = 'bold';
            app.highpassfiltercutoffEditField.BackgroundColor = [0.902
0.902 0.902];
            app.highpassfiltercutoffEditField.Position = [181 377 100
22];
            app.highpassfiltercutoffEditField.Value = 20;

            % Create lowpassfiltercutoffEditFieldLabel
```

```
            app.lowpassfiltercutoffEditFieldLabel =
uilabel(app.UIFigure);
            app.lowpassfiltercutoffEditFieldLabel.BackgroundColor = [1
1 1];
            app.lowpassfiltercutoffEditFieldLabel.HorizontalAlignment
= 'right';
            app.lowpassfiltercutoffEditFieldLabel.FontWeight = 'bold';
            app.lowpassfiltercutoffEditFieldLabel.Position = [42 437
125 22];
            app.lowpassfiltercutoffEditFieldLabel.Text = 'low pass
filter cut off';

            % Create lowpassfiltercutoffEditField
            app.lowpassfiltercutoffEditField =
uieditfield(app.UIFigure, 'numeric');
            app.lowpassfiltercutoffEditField.ValueChangedFcn =
createCallbackFcn(app, @lowpassfiltercutoffEditFieldValueChanged,
true);
            app.lowpassfiltercutoffEditField.HorizontalAlignment =
'center';
            app.lowpassfiltercutoffEditField.FontWeight = 'bold';
            app.lowpassfiltercutoffEditField.Position = [181 437 100
22];
            app.lowpassfiltercutoffEditField.Value = 730;

            % Create LableLabel
            app.LableLabel = uilabel(app.UIFigure);
            app.LableLabel.FontName = 'AvantGarde';
            app.LableLabel.FontWeight = 'bold';
            app.LableLabel.FontColor = [1 1 1];
            app.LableLabel.Position = [46 210 67 22];
            app.LableLabel.Text = 'Lable';

            % Create BPMLabel
            app.BPMLabel = uilabel(app.UIFigure);
            app.BPMLabel.FontSize = 14;
            app.BPMLabel.FontWeight = 'bold';
            app.BPMLabel.FontColor = [1 1 1];
            app.BPMLabel.Position = [46 246 45 22];
            app.BPMLabel.Text = 'BPM: ';

            % Create RPMLabel
            app.RPMLabel = uilabel(app.UIFigure);
            app.RPMLabel.FontSize = 14;
            app.RPMLabel.FontWeight = 'bold';
            app.RPMLabel.FontColor = [1 1 1];
            app.RPMLabel.Position = [46 148 36 22];
            app.RPMLabel.Text = 'RPM';

            % Create Label2
            app.Label2 = uilabel(app.UIFigure);
            app.Label2.FontName = 'AvantGarde';
            app.Label2.FontWeight = 'bold';
            app.Label2.FontColor = [1 1 1];
            app.Label2.Position = [46 101 78 24];
```

```
            app.Label2.Text = 'Label2';

            % Create SAPAProjectElectronicStethoscopeLabel
            app.SAPAProjectElectronicStethoscopeLabel =
uilabel(app.UIFigure);

app.SAPAProjectElectronicStethoscopeLabel.HorizontalAlignment =
'center';
            app.SAPAProjectElectronicStethoscopeLabel.FontName =
'Times New Roman';
            app.SAPAProjectElectronicStethoscopeLabel.FontSize = 36;
            app.SAPAProjectElectronicStethoscopeLabel.FontWeight =
'bold';
            app.SAPAProjectElectronicStethoscopeLabel.FontColor = [1 1
1];
            app.SAPAProjectElectronicStethoscopeLabel.Position = [338
707 593 48];
            app.SAPAProjectElectronicStethoscopeLabel.Text = 'SAPA
Project Electronic Stethoscope ';

            % Create TextAreaLabel
            app.TextAreaLabel = uilabel(app.UIFigure);
            app.TextAreaLabel.HorizontalAlignment = 'right';
            app.TextAreaLabel.Position = [294 697 55 22];
            app.TextAreaLabel.Text = 'Text Area';

            % Create TextArea
            app.TextArea = uitextarea(app.UIFigure);
            app.TextArea.HorizontalAlignment = 'center';
            app.TextArea.FontName = 'Times New Roman';
            app.TextArea.FontSize = 18;
            app.TextArea.FontWeight = 'bold';
            app.TextArea.FontColor = [1 0 0];
            app.TextArea.BackgroundColor = [0 0 0];
            app.TextArea.Position = [452 646 146 31];
            app.TextArea.Value = {'Signal'};

            % Create TextArea_2
            app.TextArea_2 = uitextarea(app.UIFigure);
            app.TextArea_2.HorizontalAlignment = 'center';
            app.TextArea_2.FontName = 'Times New Roman';
            app.TextArea_2.FontSize = 18;
            app.TextArea_2.FontWeight = 'bold';
            app.TextArea_2.FontColor = [1 0 0];
            app.TextArea_2.BackgroundColor = [0 0 0];
            app.TextArea_2.Position = [898 646 146 31];
            app.TextArea_2.Value = {'Signal FFT'};

            % Create TextArea_3
            app.TextArea_3 = uitextarea(app.UIFigure);
            app.TextArea_3.HorizontalAlignment = 'center';
            app.TextArea_3.FontName = 'Times New Roman';
            app.TextArea_3.FontSize = 18;
            app.TextArea_3.FontWeight = 'bold';
            app.TextArea_3.FontColor = [1 0 0];
```

```
            app.TextArea_3.BackgroundColor = [0 0 0];
            app.TextArea_3.Position = [452 319 146 31];
            app.TextArea_3.Value = {'Filtered Signal'};

            % Create TextArea_4
            app.TextArea_4 = uitextarea(app.UIFigure);
            app.TextArea_4.HorizontalAlignment = 'center';
            app.TextArea_4.FontName = 'Times New Roman';
            app.TextArea_4.FontSize = 18;
            app.TextArea_4.FontWeight = 'bold';
            app.TextArea_4.FontColor = [1 0 0];
            app.TextArea_4.BackgroundColor = [0 0 0];
            app.TextArea_4.Position = [862 319 181 31];
            app.TextArea_4.Value = {'Filtered Signal FFT'};

            % Show the figure after all components are created
            app.UIFigure.Visible = 'on';
        end
    end

    % App creation and deletion
    methods (Access = public)

        % Construct app
        function app = GUI

            % Create UIFigure and components
            createComponents(app)

            % Register the app with App Designer
            registerApp(app, app.UIFigure)

            % Execute the startup function
            runStartupFcn(app, @startupFcn)

            if nargout == 0
                clear app
            end
        end

        % Code that executes before app deletion
        function delete(app)

            % Delete UIFigure when app is deleted
            delete(app.UIFigure)
        end
    end
end
```

https://github.com/TahGUMUS/elect/blob/main/codes

Chapter 6

Result and Conclusion

As a result, the data obtained from the electronic stethoscope was transferred to the computer with the help of Arduino. Later, after writing code in MATLAB to process these signals, a Butterworth filter was designed and implemented to obtain filtered signals. The Butterworth circuit included the use of an amplifier circuit and an electret microphone completed by the integration of low-pass and high-pass filters. The filter proved to be effective in improving the quality of the signal. After the data in MATLAB was read digitally, two signals were recorded, filtered and unfiltered. The Fourier transform of these recorded signals was performed using the FFT function. It was seen that the graph of the filtered signal depended on the order and cut-off values.

The importance of the selection of the filter order and cut-off frequency in adapting the filtering properties to the special requirements of the heart and lung signals was realized. Codes that matched these properties were designed. The cut-off and order values in the designed codes were requested from the user. Inferences were made accordingly. The versatility of the Butterworth filter in providing a smooth frequency response in the passband contributed to a more accurate representation of the heart and lung signals. This project highlights the importance of signal processing techniques, particularly Butterworth filters, in improving the capabilities of electronic stethoscopes for medical applications. The study highlights the importance of selecting appropriate parameters in Butterworth filters for electronic stethoscope applications, ultimately contributing to more accurate and insightful auscultatory signal analysis in the clinical setting.

Chapter 7

References

1- Nussbaumer, M., & Agarwal, A. (2022). Stethoscope acoustics. Journal of Sound and Vibration, 539, 117194. https://doi.org/10.1016/j.jsv.2022.117194

2- Seah, J. J., Zhao, J., Wang, D. Y., & Lee, H. P. (2023). Review on the Advancements of Stethoscope Types in chest auscultation. Diagnostics, 13(9), 1545. https://doi.org/10.3390/diagnostics13091545

3- Ang, Y. Y., Aw, L. R., Koh, V., & Tan, R. X. (2023). Characterization and cross-comparison of digital stethoscopes for telehealth remote patient auscultation. Medicine in Novel Technology and Devices, 19, 100256. https://doi.org/10.1016/j.medntd.2023.100256.