Faculty of Engineering and Technology

Department of Electrical and Computer Engineering

Digital Signals Processing

ENCS4310

Course Project

Filtering of ECG signals

Prepared by:

- Ali Shaikh Qasem ID: 1212171
- Khalid Abdalraheem ID: 1210618

Instructor: Dr. Qadri Mayyala.

Section: 2.

Date: 7/6/2024

Birzeit University

2023-2024

Introduction

Digital signals processing is crucial in many fields nowadays, especially in medical field. It plays an important part in saving people lives by analyze and classify medical information such as electrocardiogram (ECG) data. In this project, we are given actual cardiac electrical signals, and we are going to use our skills gained from this course to perform filtering, data enhancement and noise reduction so that our ECG data will be ready for peak detection, classification and medical diagnosis. Our task in this project starts by visualizing the given ECG data by reading them from Excel file and plot them. Then we are going to use python programming language to build and analyze different types of filters such as low-pass filters, high-pass filters and cascade filters and apply them to our ECG data to observe the effect of each type.

Contents

List OF Figures

1. Theory

1.1. Digital Signal Processing (DSP) and ECG Signals

DSP involves manipulating signals to improve their quality or extract useful information. In the context of ECG signals, DSP techniques aim to remove noise and artifacts while preserving the essential features of the heart's electrical activity. The key types of filters used in this project are high-pass filters, low-pass filters, and cascade filters [1].

1.2. High-Pass Filtering

High-pass filters allow signals with frequencies higher than a certain cutoff frequency to pass through while attenuating lower frequencies. This type of filtering is particularly effective in removing baseline wander, a common low-frequency noise in ECG signals caused by patient movement or respiration [2].

1.3. High-Pass Filter Theory

The transfer function of the high-pass filter used in this project is given by:

$$H(Z) = \frac{HP(z)}{X(z)} = \frac{-\frac{1}{32} + Z^{-16} - Z^{-17} + \frac{1}{32} Z^{-32}}{1 - Z^{-1}}$$

This filter belongs to the Infinite Impulse Response (IIR) family, as the output at the current sample depends on previous outputs. The frequency response of the high-pass filter is plotted to visualize how different frequencies are attenuated or passed through the filter.

1.4. Low-Pass Filtering

Low-pass filters allow signals with frequencies lower than a certain cutoff frequency to pass through while attenuating higher frequencies. This type of filtering helps in removing high-frequency noise, which can be due to muscle contractions or electrical interference [2]

1.5. Low-Pass Filter Theory

The transfer function of the low-pass filter used in this project is:

$$H(Z) = \frac{LP(z)}{X(z)} = \frac{1 - 2Z^{-6} + Z^{-12}}{1 - 2Z^{-1} + Z^{-2}}$$

Similar to the high-pass filter, this low-pass filter is also an IIR filter. The frequency response is analyzed to ensure that the desired high-frequency noise is effectively attenuated.

1.6. Cascade Filtering

In digital signal processing, a cascaded integrator–comb (CIC) is a computationally efficient class of low-pass finite impulse response (FIR) filter that chains N number of integrator and comb filter pairs (where N is the filter's order) to form a decimator or interpolator. In a decimating CIC, the input signal is first fed through N integrator stages, followed by a down-sampler, and then N comb stages. An interpolating CIC (e.g. Figure 1) has the reverse order of this architecture, but with the down-sampler replaced with a zero-stuffer (up-sampler) [3].

1.7. Cascade Filter Theory

When the high-pass and low-pass filters are applied sequentially, the overall impulse response of the system is the convolution of the individual impulse responses. Since convolution is commutative, the order of applying these filters does not affect the final outcome:

$$H_{overall}(n) = HP(n) * LP(n) = LP(n) * HP(n)$$

2. Procedure & Analysis

2.1. Data Visualization

Part 1: Insert the real-time column for each signal:

- In this part, we have inserted the columns representing the real time and the average amplitude.

Part 2: Display each of these signals (reduce the width of the display lines for better readability):

- In this part, we visualized the inserted data as clear figures as shown below:

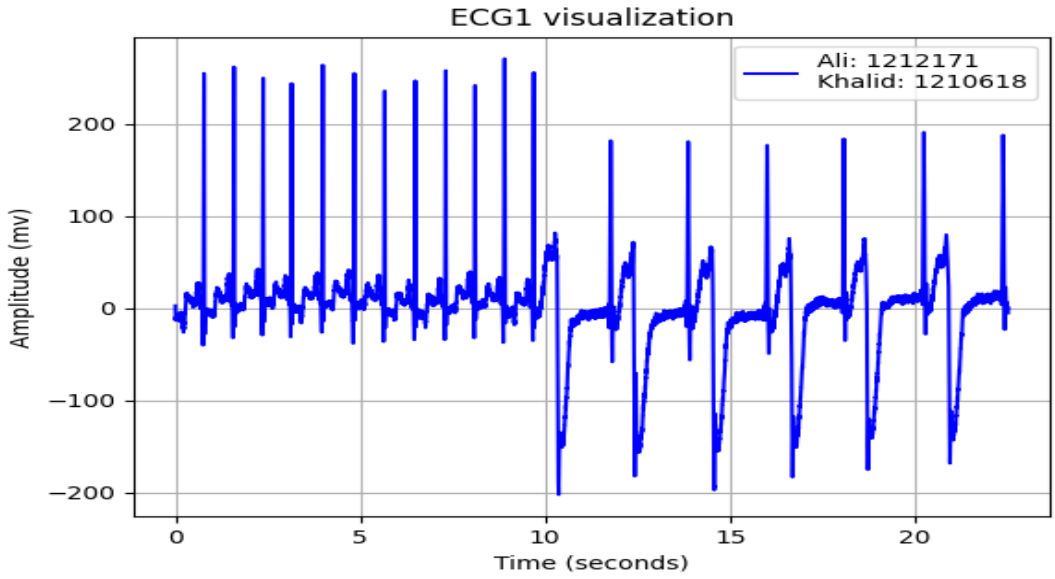

Figure 1: ECG1 visualization

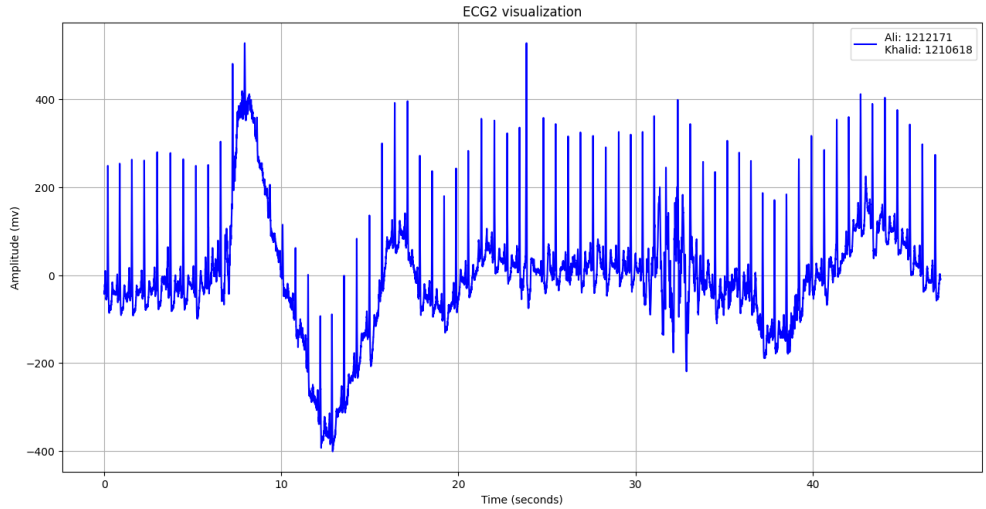

Figure 2: ECG2 visualization

Part 3: What filtering types are necessary to improve the readability of data and make automatic processing possible?

For improving data readability and enabling automatic processing, various types of filtering are crucial. Noise filtering plays a critical role in removing irrelevant or redundant information, ensuring data cleanliness and consistency. Outlier filtering aids in detecting and eliminating extreme values that could skew analysis and lead to erroneous conclusions. Data normalization becomes necessary to standardize the data range, facilitating easier comparison and processing. Moreover, duplicate filtering eliminates repetitive entries that might distort data analysis. Transformational filtering, encompassing tasks like converting data formats and structuring unstructured data, is essential for making data machine-readable and suitable for automated processing. Lastly, semantic filtering is vital to ensure data contextual relevance and accuracy, thereby enhancing its readability and usability in automated systems.

2.2. High-Pass Filtering

In this part, the following high pass filter is constructed:

$$HP(n) = HP(n-1) - 1/32\ X(n) + X(n-16) - X(n-17) + 1/32\ X(n-32)$$

Part 1: Calculate the transfer function:

$$\text{Transfer function} = H(Z) = \frac{HP(z)}{X(z)} = \frac{-\frac{1}{32} + Z^{-16} - Z^{-17} + \frac{1}{32}\ Z^{-32}}{1 - Z^{-1}}$$

Part 2: Find and plot the frequency response:

$$\text{Frequency response} = H(e^{jw}) = \frac{-\frac{1}{32} + e^{-16jw} - e^{-17jw} + \frac{1}{32}\ e^{-32jw}}{1 - e^{-jw}}$$

we represented the filter as a numerator and denominator of its coefficients, then we plotted its frequency response as shown:

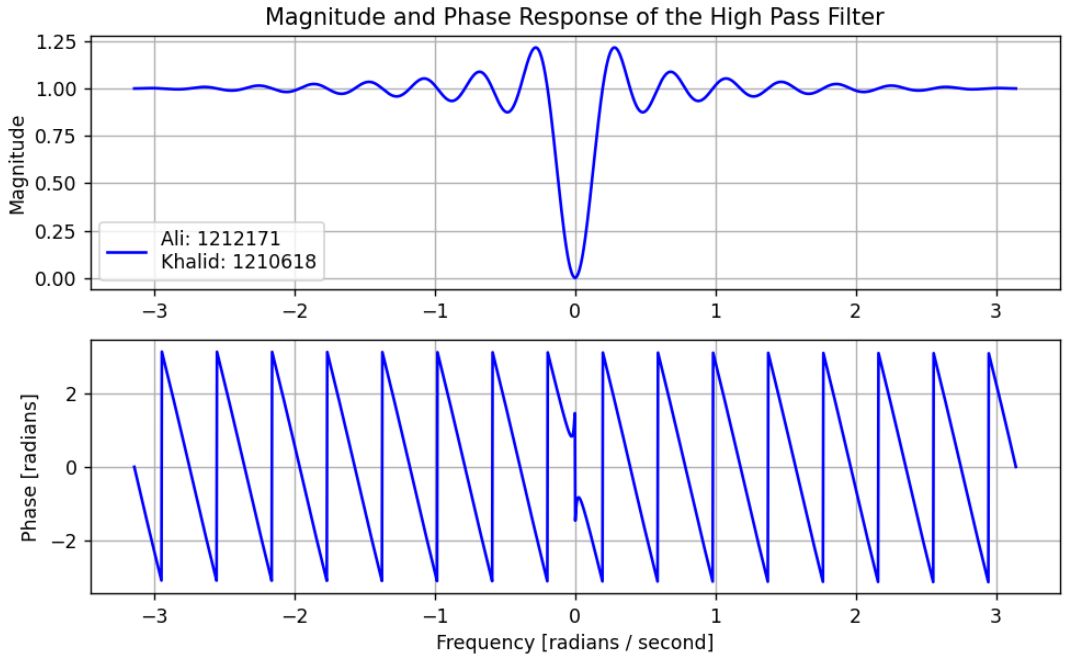

Figure 3: Frequency response of the high pass filter

Part 3: Which family does it belong to (FIR, IIR)?

From the transfer function shown above, it seems we have a pole at z = 1, but it also noticed from the numerator that there's a zero at z = 1, thus, the pole will be canceled, and all other poles will be at the origin, so the impulse response will be finite, therefore, the system belong to FIR family.

Part 4: Apply this filter to the ECG1 and then ECG2 signal:

ECG1 filtering:

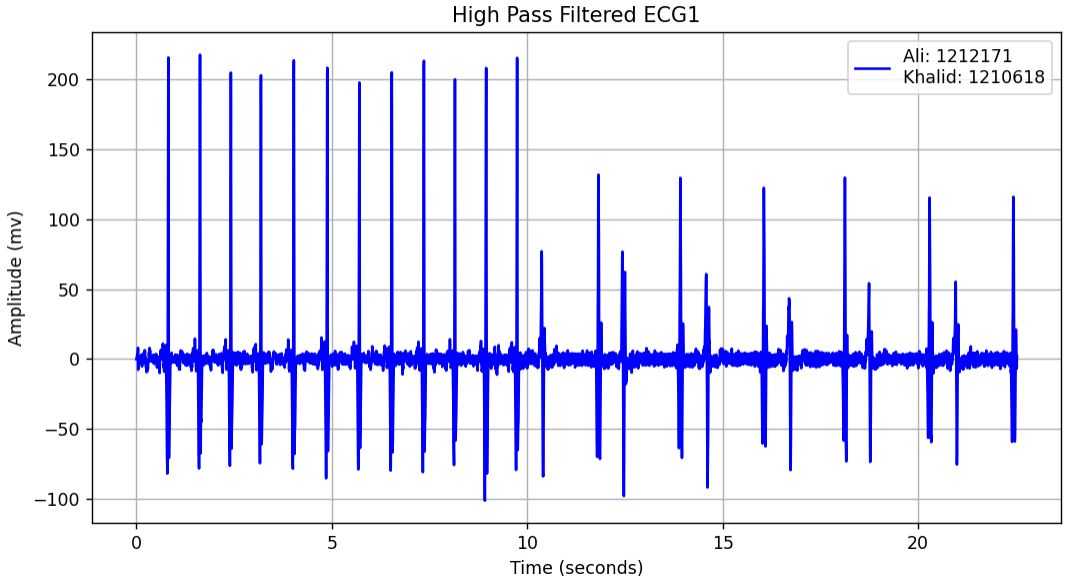

Figure 4: High pass filtered ECG1

- As shown above, the main thing noticed is that the baseline of the signal has become more stable and closed to zero, this is because high pass filter removes the low frequency noise such as baseline wander (typically below 0.5 Hz) which can be noticed from the DTFT of the signal ECG1 shown in figure.5 below. However, we still notice some high frequency noise (typically from 0.5 to 150 Hz) especially at the zero line, this is because they passed through the filter.

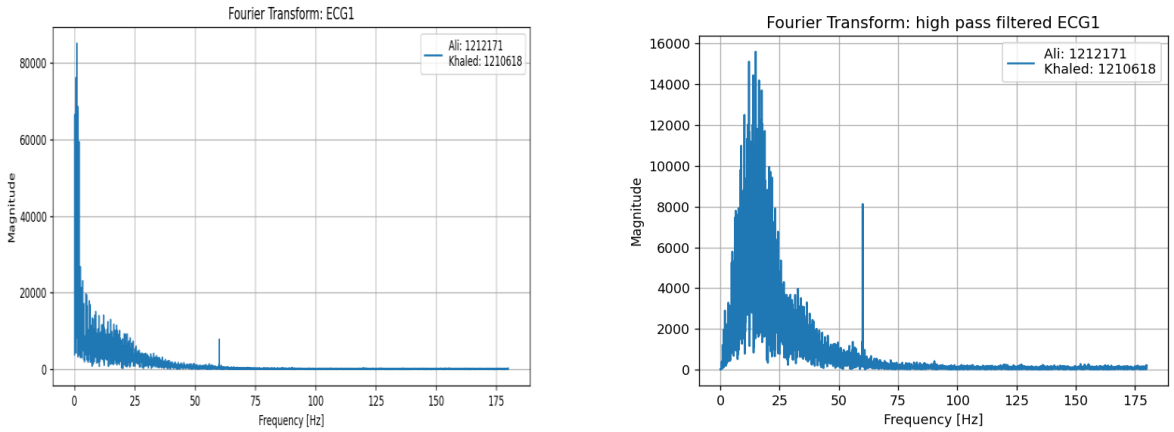

Figure 5: Fourier transform of original and filtered ECG1

ECG2 filtering:

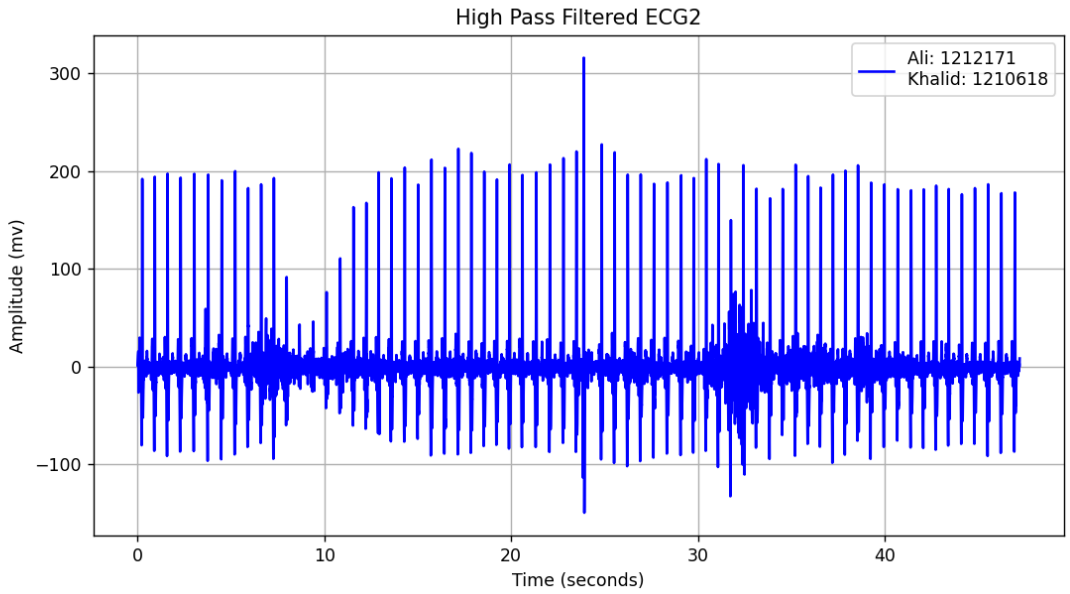

Figure 6: High pass filtered ECG2

- As shown above, we notice the same change as the previous signal, that the baseline is more stable and closed to zero due to removing low frequency baseline noise as shown in figure.7 below. But we still notice some high frequency noise.

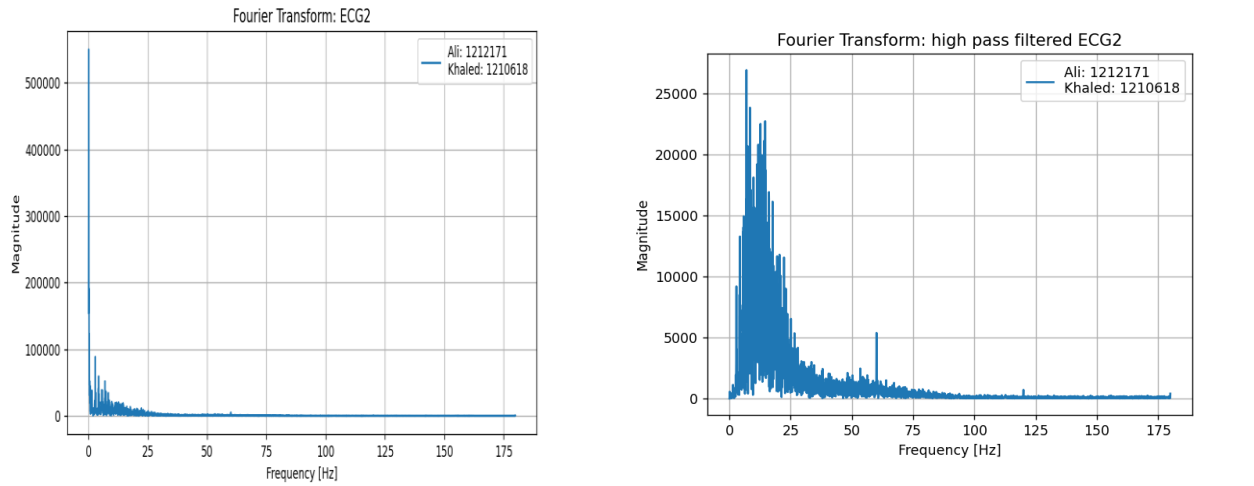

Figure 7: Fourier transform of original and filtered ECG2

Part 5: How to partially resolve the problems occurring at the start of the signal?

To address problems occurring at the start of a signal, you can use several effective techniques. Applying a smoothing filter can reduce initial noise and fluctuations, making the signal more stable. Baseline correction helps adjust the starting point of the signal to a consistent level, eliminating any initial offset or drift. Additionally, using a delay or buffer to ignore the initial part of the signal can prevent brief issues at the beginning from affecting the analysis. Calibration procedures can also ensure that the signal starts accurately and consistently each time. These methods together improve the quality of the initial signal, reducing common startup problems and making the data more reliable.

2.3. Low pass Filtering

In this part, the following low pass filter is constructed:

$$LP(n) = 2\,LP(n-1) - LP(n-2) + X(n-16) - x(n-17) + 1/32\,x(n-32)$$

Part 1: Calculate the transfer function:

Transfer function = H(Z) = $\frac{LP(z)}{X(z)} = \frac{1-2Z^{-6}+Z^{-12}}{1-2Z^{-1}+Z^{-2}}$

Part 2: Find and plot the frequency response:

Frequency response = $H(e^{jw}) = \frac{1-2\,e^{-6jw}-e^{-12jw}}{1-2e^{-jw}+e^{-2jw}}$

we represented the filter as a numerator and denominator of its coefficients, then we plotted its frequency response as shown:

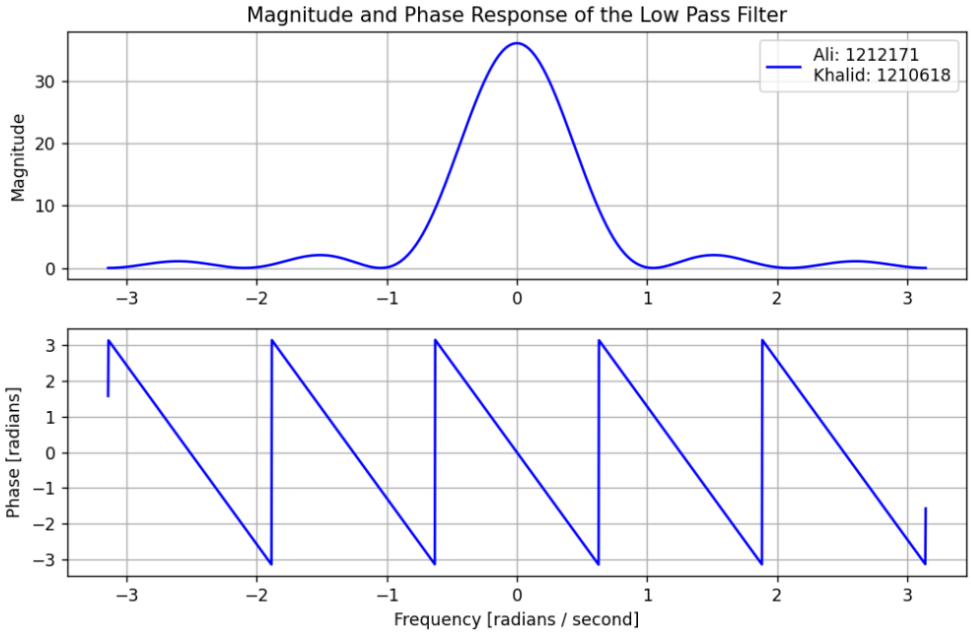

Figure 8: Frequency response of the low pass

Part 3: Which family does it belong to (FIR, IIR)?

From the transfer function shown above, it seems we have a pole at z = 1, but it also noticed from the numerator that there's a zero at z = 1, thus, the pole will be canceled, and all other poles will be at the origin, so the impulse response will be finite, therefore, the system belong to FIR family.

Part 4: Apply this filter to the ECG1 and then ECG2 signal:

ECG1 filtering:

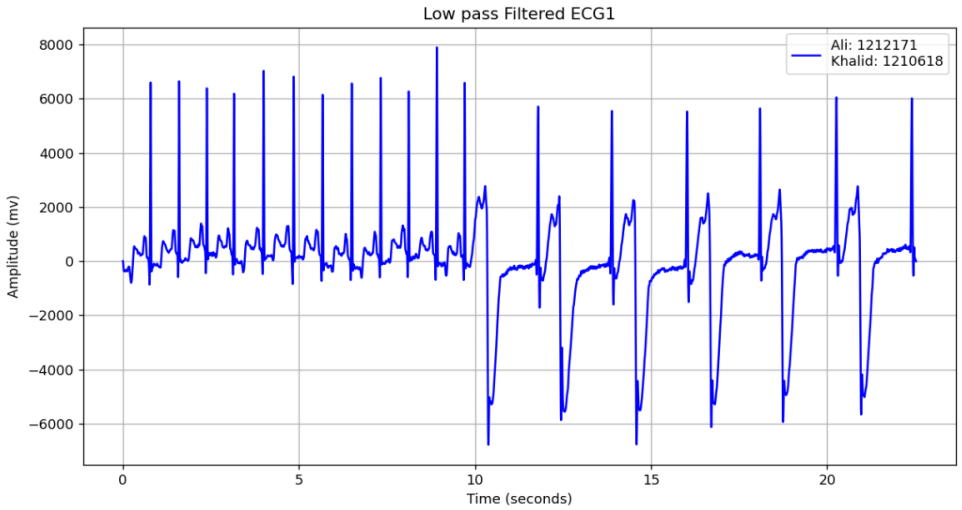

Figure 9: Low pass filtered ECG1

ECG2 filtering:

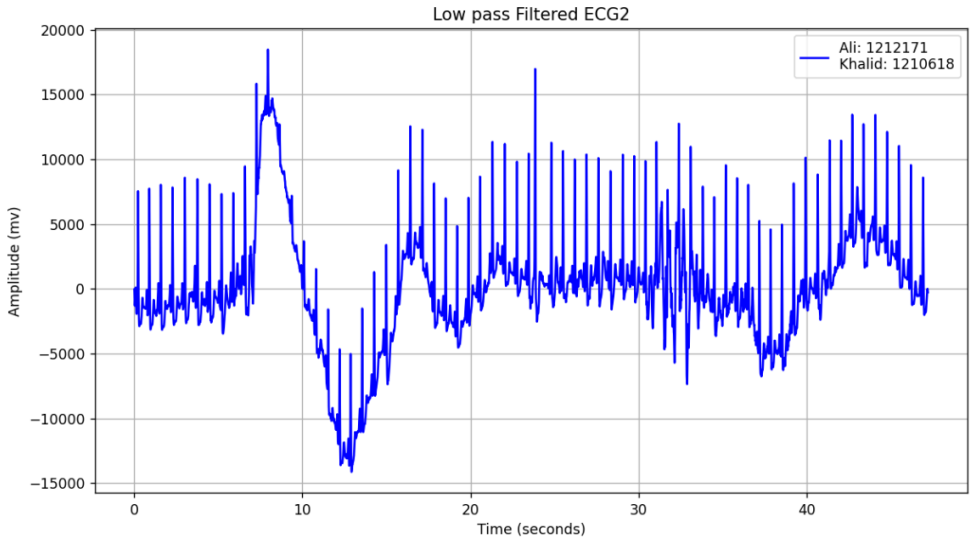

Figure 10: Low pass filtered ECG2

- In this part, we notice that both signals have become smooth and readable, this is because the low pass filter removes and attenuates high frequencies (typically from 0.5 to 100 Hz) quency components cause an undesired noise as

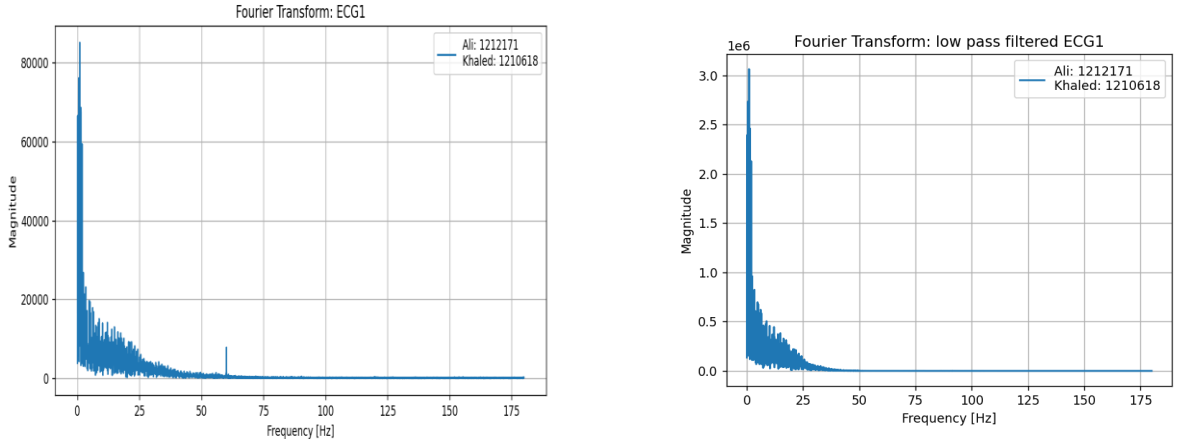

Figure 11: Fourier transform of original and filtered ECG1

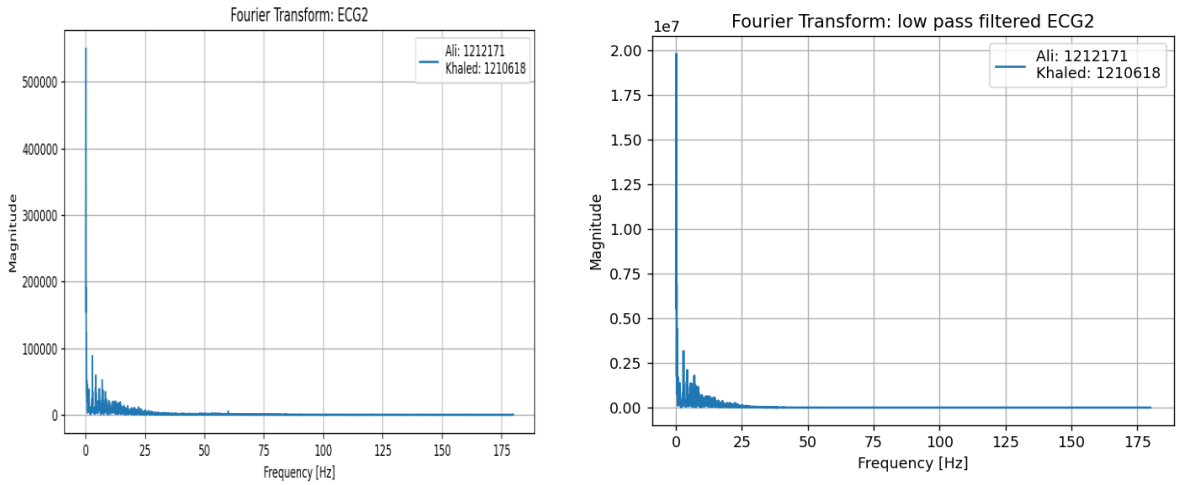

Figure 12: Fourier transform of original and filtered ECG2

2.4. Cascade Filtering

In this part we will use the output of the high pass filter as the input of the low pass filter:

Part 1: Display the result obtained for ECG1 and ECG2:

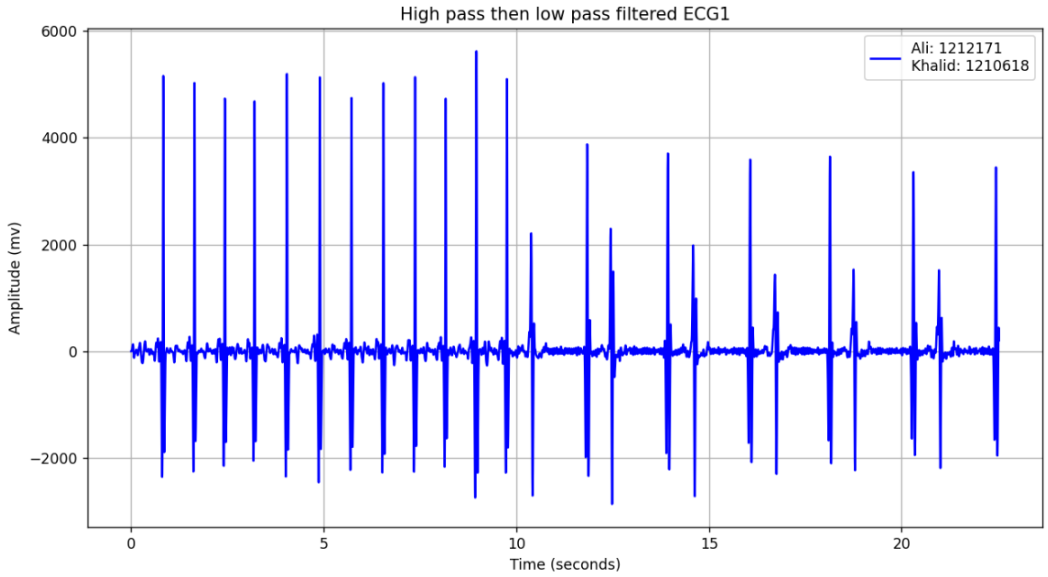

Figure 13: High pass then low pass filtered ECG1

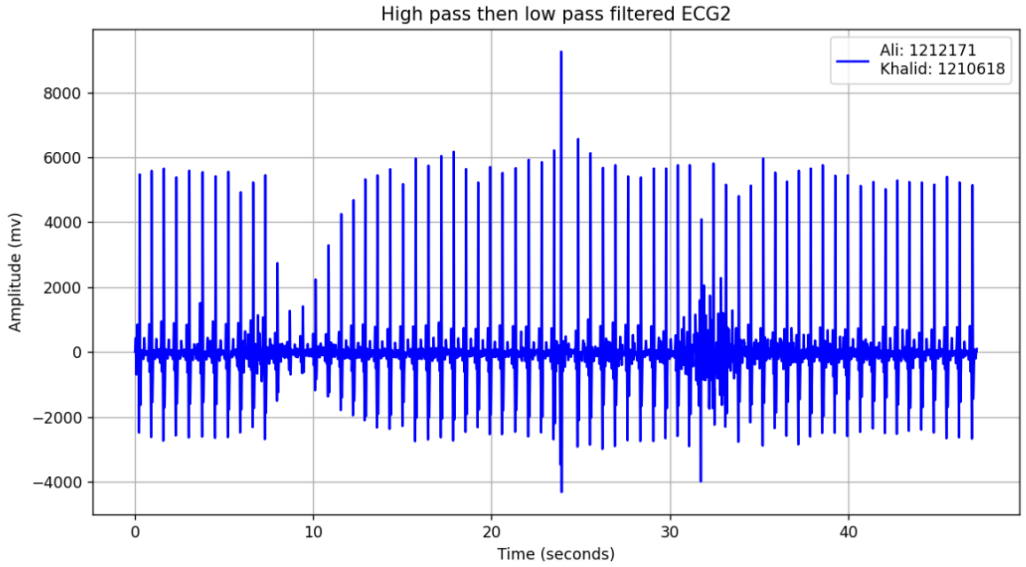

Figure 14: High pass then low pass filtered ECG2

- In this part, when we used the output of low pass filter as an input of high pass filter, this will result in a band pass filter that combines both advantages of low pass and high pass filtering. Thus, we notice that both signals shown above has a stable baseline due to removing low pass frequencies and we also notice that the high frequency noise were removed so that signals above look smooth and readable.

Part 2: Is there a difference if you reverse the filters?

when we reverse the filters, we will have the same results as shown below. This is because the overall impulse response of the system represents the convolution of both low pass and high pass filters, and since the convolution is a commutative operation, then no changes will occur.

$$\text{Overall } h(n) = h_{HP}(n) * h_{LP}(n) = h_{LP}(n) * h_{HP}(n).$$

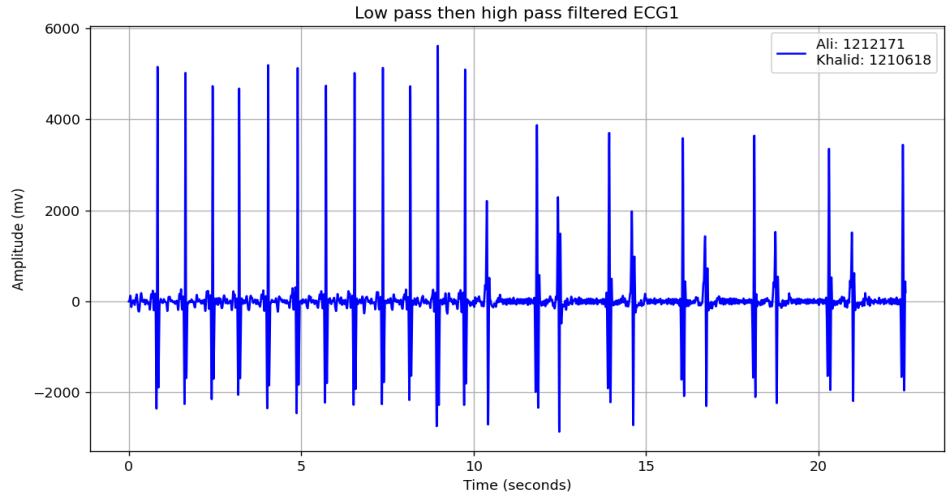

Figure 15: Reverse for ECG1

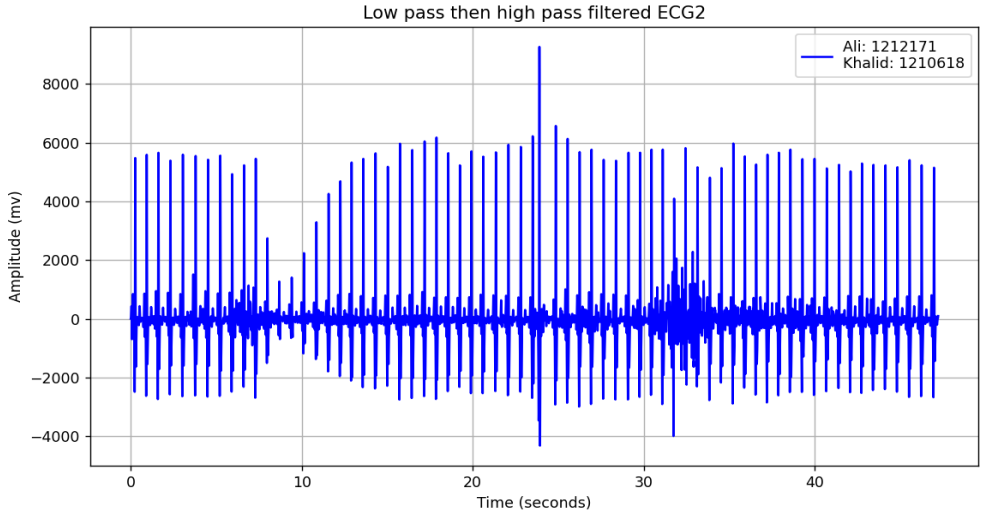

Figure 16: Reverse for ECG2

3. Conclusion

In this project, we have successfully filtered real electrocardiographic (ECG) signals using different kind of filters such as high pass, low pass and band pass filters. All these filtering techniques contributed to improve our ECG data and prepare them for subsequent peak detection and classification. We start by visualizing raw ECG data which contained significant noise and baseline wander. Then we apply these data to high pass filter which removes the baseline wander and low pass filter which removes the high frequency noise. Finally, we applied them to a band pass filter which combined features of low pass and high pass filtering. This project emphasizes the importance of digital signals processing in medical applications. Furthermore, it developed out theoretical knowledge in this course into a useful practical application.

4. References

[1]: Discrete-Time Signal Processing, Third Edition Alan V, Oppenheim Ronald W. Schafer.

[2]: https://www.geeksforgeeks.org/difference-between-low-pass-filter-and-high-pass-filter/

- Visited in 6/6/2024 at 5:18 pm.

[3]: https://en.wikipedia.org/wiki/Cascaded_integrator%E2%80%93comb_filter/

- Visited in 6/6/2024 at 5:40 pm.

5. Appendix

5.1. Python Code of the Project

```
import pandas as pd
import matplotlib.pyplot as plt
from scipy.signal import freqz, lfilter
import numpy as np
from scipy.fft import fft, fftshift

def Data_visualization (file_name :str, sheet_name: str):
    #obtain the time and amplitude
    amplitude, time = get_excel_data(file_name, sheet_name)

    #plot the data
    plt.figure()
    plt.plot(time, amplitude, 'b', label='Ali: 1212171 \nKhalid: 1210618')
    plt.xlabel('Time (seconds)')
    plt.ylabel('Amplitude (mv)')
    plt.title(sheet_name + " visualization")
    plt.legend()
    plt.grid(True)
    plt.show()

def get_excel_data (file_name :str, sheet_name: str):
    # read the excel file
    data = pd.read_excel(file_name, sheet_name, header=1) # header=1 to ignore
the first row which contains column name
    # obtain the required columns from the file
    amplitude = data.iloc[:, 2] # the amplitude is thurd column
    time = data.iloc[:, 1] # the real time is the second column
    return amplitude.tolist(), time.tolist()

def high_pass_filter (singal_name: str):
    #defning the nominator and denominator of filter transfer function
    numerator = [0] * 33
    numerator[0], numerator[16], numerator[17], numerator[32] = -1/32, 1, -1,
1/32
    denominator = [1, -1]
    #define the filter
    # detetmine the freq. respose
    w, h = freqz(numerator, denominator, worN=np.linspace(-1*np.pi, 1 * np.pi,
2048))
    # plot the frequecy response
    plt.figure()
    # Magnitude response
```

```
    plt.subplot(2, 1, 1)
    plt.plot(w, abs(h), 'b', label='Ali: 1212171 \nKhalid: 1210618')
    plt.title('Magnitude and Phase Response of the High Pass Filter')
    plt.ylabel('Magnitude')
    plt.legend()
    plt.grid()

    # Phase response
    plt.subplot(2, 1, 2)
    plt.plot(w, np.angle(h), 'b')
    plt.xlabel('Frequency [radians / second]')
    plt.ylabel('Phase [radians]')
    plt.grid()

    plt.show()

    #applying filter to the signal
    amplitude, time = get_excel_data('Data_ECG_raw.xlsx', singal_name)
    filtered_signal = lfilter(numerator, denominator, amplitude)

    #plot the filtered signal
    plt.figure()
    plt.plot(time, filtered_signal, 'b', label='Ali: 1212171 \nKhalid: 1210618')
    plt.xlabel('Time (seconds)')
    plt.ylabel('Amplitude (mv)')
    plt.title('High Pass Filtered ' + singal_name)
    plt.legend()
    plt.grid(True)
    plt.show()

def low_pass_filter (signal_name: str):
    #defning the nominator and denominator of filter transfer function
    numerator = [0] * 13
    numerator[0], numerator[6], numerator[12] = 1, -2, 1
    denominator = [1, -2, 1]

    #define the filter
    # detetmine the freq. respose
    w, h = freqz(numerator, denominator, worN=np.linspace(-1*np.pi, 1 * np.pi,
2048))
    # plot the frequecy response
    plt.figure()
    # Magnitude response
    plt.subplot(2, 1, 1)
    plt.plot(w, abs(h), 'b', label='Ali: 1212171 \nKhalid: 1210618')
```

```
    plt.title('Magnitude and Phase Response of the Low Pass Filter')
    plt.ylabel('Magnitude')
    plt.legend()
    plt.grid()

    # Phase response
    plt.subplot(2, 1, 2)
    plt.plot(w, np.angle(h), 'b')
    plt.xlabel('Frequency [radians / second]')
    plt.ylabel('Phase [radians]')
    plt.grid()

    plt.show()

    #applying filter to the signal
    amplitude, time = get_excel_data('Data_ECG_raw.xlsx', signal_name)
    filtered_signal = lfilter(numerator, denominator, amplitude)

    #plot the filtered signal
    plt.figure()
    plt.plot(time, filtered_signal, 'b', label='Ali: 1212171 \nKhalid: 1210618')
    plt.xlabel('Time (seconds)')
    plt.ylabel('Amplitude (mv)')
    plt.title('Low pass Filtered ' + signal_name)
    plt.legend()
    plt.grid(True)
    plt.show()

def cascade_filter ( signal_name: str):
    #defning the nominator and denominator of HPF
    H_numerator = [0] * 33
    H_numerator[0], H_numerator[16], H_numerator[17], H_numerator[32] = -1/32, 1,
-1, 1/32
    H_denominator = [1, -1]
    #defning the nominator and denominator of lPF
    L_numerator = [0] * 13
    L_numerator[0], L_numerator[6], L_numerator[12] = 1, -2, 1
    L_denominator = [1, -2, 1]
    # get the data array
    amplitude, time = get_excel_data('Data_ECG_raw.xlsx', signal_name)
    # using the output of HPF as input of LPF
    HPF_signal = lfilter(H_numerator, H_denominator, amplitude)
    result_signal = lfilter(L_numerator, L_denominator, HPF_signal)
    # plot the resultant result
    plt.figure()
```

```
    plt.plot(time, result_signal, 'b', label='Ali: 1212171 \nKhalid: 1210618')
    plt.xlabel('Time (seconds)')
    plt.ylabel('Amplitude (mv)')
    plt.title('High pass then low pass filtered ' + signal_name)
    plt.legend()
    plt.grid(True)
    plt.show()

def reverse_cascade_filter ( signal_name: str):
    #defning the nominator and denominator of HPF
    H_numerator = [0] * 33
    H_numerator[0], H_numerator[16], H_numerator[17], H_numerator[32] = -1/32, 1,
-1, 1/32
    H_denominator = [1, -1]
    #defning the nominator and denominator of lPF
    L_numerator = [0] * 13
    L_numerator[0], L_numerator[6], L_numerator[12] = 1, -2, 1
    L_denominator = [1, -2, 1]
    # get the data array
    amplitude, time = get_excel_data('Data_ECG_raw.xlsx', signal_name)
    # using the output of LPF as input of HPF
    LPF_signal = lfilter(L_numerator, L_denominator, amplitude)
    result_signal = lfilter(H_numerator, H_denominator, LPF_signal)
    # plot the resultant result
    plt.figure()
    plt.plot(time, result_signal, 'b', label='Ali: 1212171 \nKhalid: 1210618')
    plt.xlabel('Time (seconds)')
    plt.ylabel('Amplitude (mv)')
    plt.title('Low pass then high pass filtered ' + signal_name)
    plt.legend()
    plt.grid(True)
    plt.show()

def Fourier_transform (signal_name: str):
    amplitude, time = get_excel_data('Data_ECG_raw.xlsx', signal_name)

    # define a number of samples to read ( to aviod non numbers read from excel
file)
    N=8000
    # Compute the Fourier transform
    fft_amplitude = np.fft.fft(amplitude[:N])
    fft_freq = np.fft.fftfreq(len(amplitude[:N]), d=(time[1] - time[0]))
    # Single-sided spectrum
    single_sided_fft_amplitude = fft_amplitude[:N // 2]
    single_sided_fft_freq = fft_freq[:N // 2]
```

```
    # Plot the original signal
    plt.figure()
    # Plot the magnitude of the single-sided Fourier transform
    plt.plot(single_sided_fft_freq, np.abs(single_sided_fft_amplitude),
label='Ali: 1212171\nKhaled: 1210618')
    plt.title(f'Fourier Transform: {signal_name}')
    plt.xlabel('Frequency [Hz]')
    plt.ylabel('Magnitude')
    plt.grid()
    plt.legend()

    plt.show()


if __name__ == '_main_':
   # Data_visualization('Data_ECG_raw.xlsx', 'ECG1')
    high_pass_filter('ECG2')
    #low_pass_filter('ECG1')
    #cascade_filter('ECG2')
    #reverse_cascade_filter('ECG1')
    #Fourier_transform('ECG2')
```