Department of Electrical and Computer Engineering

Digital Signal Processing

Filtering a real electrocardiographic signal

Prepared by:

Osaid Nur	1210733	Sec.4
Salah Dawabsheh	1210722	Sec.4
Waleed Rimawi	1211491	Sec.2

Instructor: Dr. Qadri Mayyala

Date: 4/6/2024

Table of Content

Table of Content 2
Table of figures : 3
Theory : 4
1. Data Visualization 5
1.1. Real Time Columns 5
1.2. Signals Deployment 6
Observations 7
1.3. Necessary Filtering for Data Readability and Automatic Processing 8
Types of Filtering 8
Why These Filters are Necessary 8
.2 Filtering the ECG 9
2.1. The high-pass filter 9
2.1.1. Transfer Function 9
2.1.2. Find and Plot the Frequency Response 10
2.1.3. Which Family Does the High-Pass Filter Belong to? 11
2.1.4. Apply the High-Pass Filter to ECG1 and ECG2 Signals 12
2.1.5. How to Partially Resolve Problems Occurring at the Start of the Signal 14
2.2. The low-pass filter 15
2.2.1. Transfer Function 15
2.2.2. Find and Plot the Frequency Response 15
2.2.3. Which family does it belong to? 16
2.2.4. Apply the Low-Pass Filter to ECG1 and ECG2 Signals 17
2.3. The low-pass filter 18
2.3.1. ECG1 Signal 18
2.3.2. ECG2 Signal 19
Conclusion 21
Appendix A 22
Appendix B : 30

Table of figures :

Figure 1. insert the real time values 5
Figure 2: ECG1 original signal 6
Figure 3: ECG2 original signal 7
Figure 4: Frequency Response Plot 10
Figure 5: Zero-Pole Plot 11
Figure 6: Filtered ECG1 Signal: 12
Figure 7: Filtered ECG2 Signal 13
Figure 8: Frequency Response Plot 16
Figure 9: Zero-Pole Plot 16
Figure 10: Filtered ECG1 Signal 17
Figure 11: Filtered ECG2 Signal 17
Figure 12 : ECG1 Filtered by HPF then LPF 18
Figure 13: ECG1 Filtered by LPF then HPF 18
Figure 14: ECG2 Filtered by HPF then LPF 19
Figure 15: ECG2 Filtered by LPF then HPF 19
Figure 16.Sample Run 1 30
Figure 17..Sample Run 2 30
Figure 18..Sample Run 3 31
Figure 19..Sample Run 4 31
Figure 20..Sample Run 5 31
Figure 21..Sample Run 6 31
Figure 22..Sample Run 7 31

Theory :

Digital Signal Processing (DSP)

Digital Signal Processing involves the manipulation of signals that have been converted into digital form. DSP techniques are used to improve the accuracy and reliability of digital communications by removing noise, enhancing signal quality, and extracting valuable information.

Electrocardiographic (ECG) Signals

Electrocardiographic (ECG) signals represent the electrical activity of the heart. These signals are critical for diagnosing various heart conditions. An ECG signal typically consists of a series of waves (P, QRS, T) that correspond to different phases of the heart's electrical cycle.

Sampling Frequency

The sampling frequency, or sampling rate, refers to the number of samples per second taken from a continuous signal to make it a discrete signal. In this project, the ECG signals were sampled at 360 Hz, meaning 360 data points are recorded every second. This conversion is essential for accurate digital signal analysis.

High-Pass Filter

A high-pass filter is designed to pass signals with a frequency higher than a certain cutoff frequency and attenuate signals with frequencies lower than the cutoff frequency. It is used to remove low-frequency noise, such as baseline wander, from the ECG signal.

Low-Pass Filter

A low-pass filter allows signals with a frequency lower than a certain cutoff frequency to pass and attenuates signals with frequencies higher than the cutoff frequency. It is used to remove high-frequency noise from the ECG signal.

FIR and IIR Filters

- **FIR (Finite Impulse Response) Filters:** These filters have a finite duration of response to an impulse input. They are always stable and have a linear phase response, making them ideal for phase-sensitive applications.
- **IIR (Infinite Impulse Response) Filters:** These filters have an infinite duration of response to an impulse input due to feedback. They are efficient in terms of computation but can be unstable and have a nonlinear phase response.

1. Data Visualization

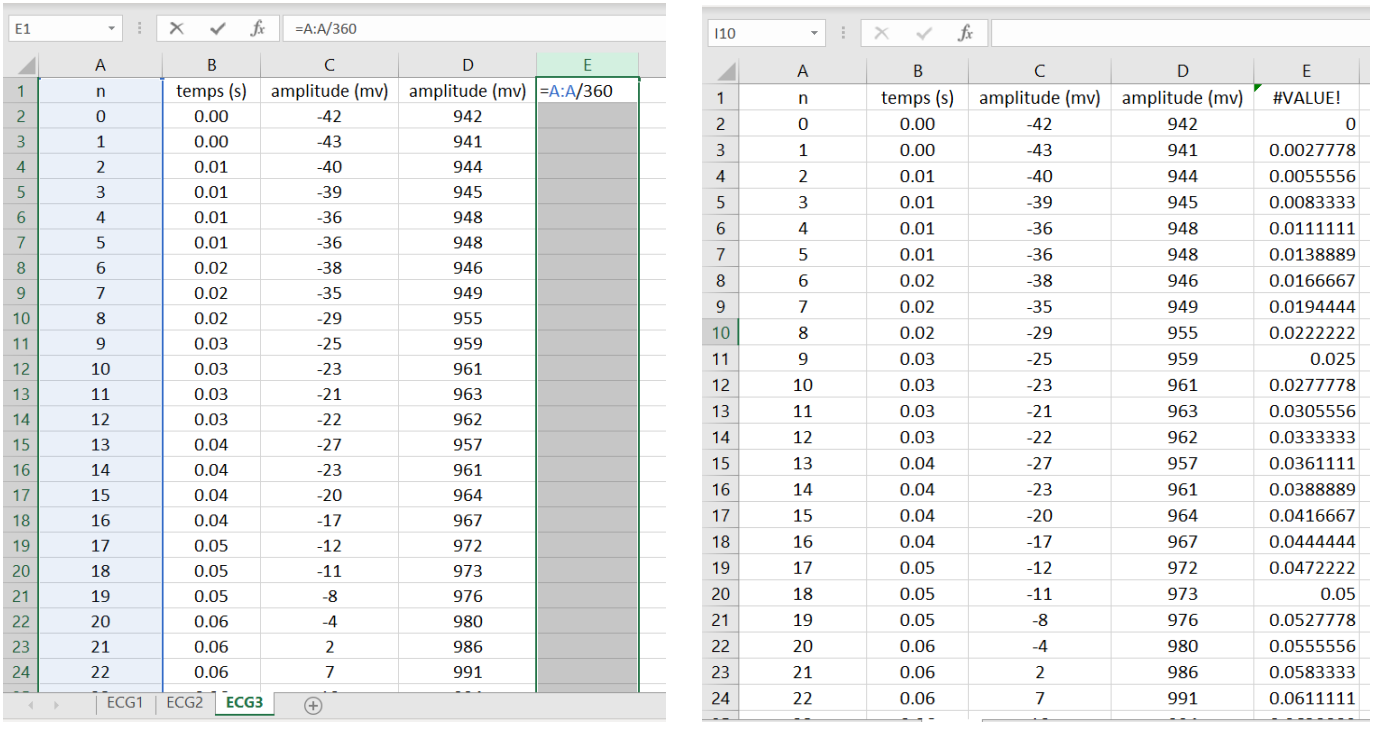
1.1. Real Time Columns

In this part, we inserted the real-time column for all signals as shown in the figures below. The real-time value t was calculated using the formula:

$$t = nT_s = \frac{n}{f_s} = \frac{n}{360\ Hz}$$

where n is the sample number, and f_s is the sampling frequency (360 Hz). This conversion from sample number to time is essential for accurate signal analysis and interpretation.

Figure 1 illustrates the inclusion of the real-time column in our dataset. This transformation is pivotal for plotting the signals against time and performing further signal processing tasks.



E1 =A:A/360

	A	B	C	D	E
1	n	temps (s)	amplitude (mv)	amplitude (mv)	=A:A/360
2	0	0.00	-42	942	
3	1	0.00	-43	941	
4	2	0.01	-40	944	
5	3	0.01	-39	945	
6	4	0.01	-36	948	
7	5	0.01	-36	948	
8	6	0.02	-38	946	
9	7	0.02	-35	949	
10	8	0.02	-29	955	
11	9	0.03	-25	959	
12	10	0.03	-23	961	
13	11	0.03	-21	963	
14	12	0.03	-22	962	
15	13	0.04	-27	957	
16	14	0.04	-23	961	
17	15	0.04	-20	964	
18	16	0.04	-17	967	
19	17	0.05	-12	972	
20	18	0.05	-11	973	
21	19	0.05	-8	976	
22	20	0.06	-4	980	
23	21	0.06	2	986	
24	22	0.06	7	991	

ECG1 | ECG2 | ECG3

I10

	A	B	C	D	E
1	n	temps (s)	amplitude (mv)	amplitude (mv)	#VALUE!
2	0	0.00	-42	942	0
3	1	0.00	-43	941	0.0027778
4	2	0.01	-40	944	0.0055556
5	3	0.01	-39	945	0.0083333
6	4	0.01	-36	948	0.0111111
7	5	0.01	-36	948	0.0138889
8	6	0.02	-38	946	0.0166667
9	7	0.02	-35	949	0.0194444
10	8	0.02	-29	955	0.0222222
11	9	0.03	-25	959	0.025
12	10	0.03	-23	961	0.0277778
13	11	0.03	-21	963	0.0305556
14	12	0.03	-22	962	0.0333333
15	13	0.04	-27	957	0.0361111
16	14	0.04	-23	961	0.0388889
17	15	0.04	-20	964	0.0416667
18	16	0.04	-17	967	0.0444444
19	17	0.05	-12	972	0.0472222
20	18	0.05	-11	973	0.05
21	19	0.05	-8	976	0.0527778
22	20	0.06	-4	980	0.0555556
23	21	0.06	2	986	0.0583333
24	22	0.06	7	991	0.0611111

Figure 1. insert the real time values

1.2. Signals Deployment

In this section, we visualize the raw electrocardiographic (ECG) signals from two datasets: ECG1 and ECG2. This visualization helps us understand the nature of the signals before applying any filtering techniques.

The following figures display the original ECG signals. These signals are sampled at 360 Hz, meaning there are 360 data points recorded every second.

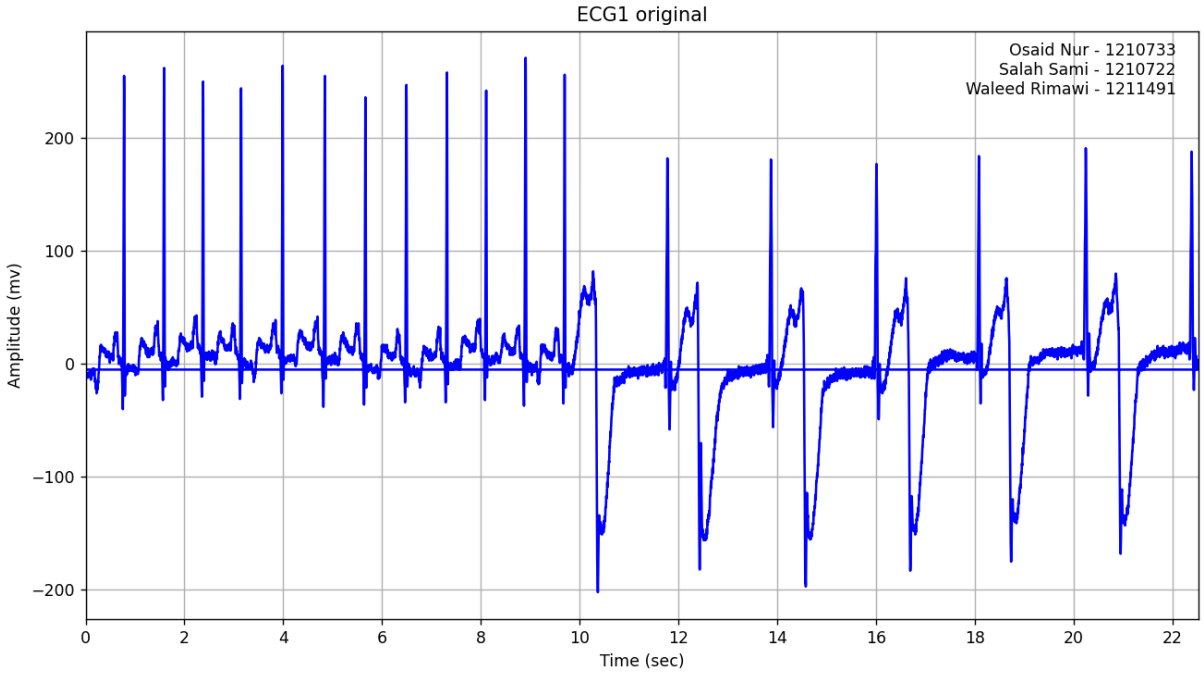


Figure 2: ECG1 original signal

Figure 2 shows the original ECG1 signal over time. The x-axis represents time in seconds, and the y-axis represents the amplitude in millivolts. The signal has periodic peaks that represent heartbeats. There is some noise in the signal that needs to be removed to make it clearer for further analysis.

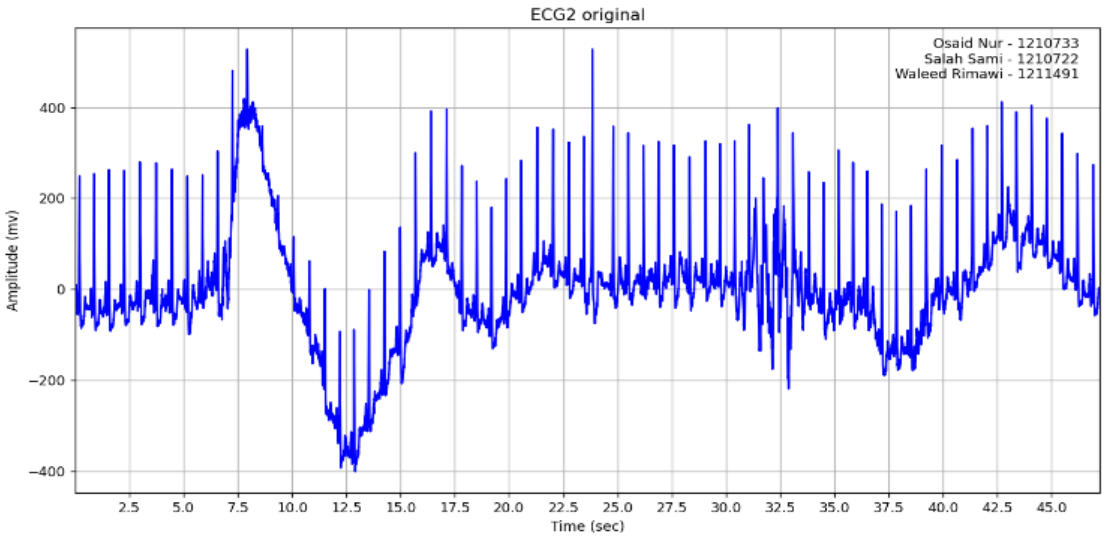


Figure 3: ECG2 original signal

Figure 3 shows the original ECG2 signal. Similar to ECG1, the x-axis is time in seconds, and the y-axis is the amplitude in millivolts. The ECG2 signal has higher peaks and more noticeable noise compared to ECG1. This noise makes it harder to see the heartbeats clearly, so filtering is necessary.

Observations

1. Noise Presence:
   - Both signals have noise that makes it hard to see the heartbeats. This noise can come from various sources, like muscle activity or electrical interference.

2. Periodic Peaks:
   - The signals have periodic peaks that indicate heartbeats. These peaks are important for measuring the heart rate and must be preserved while removing the noise.

3. Amplitude Variations:
   - The amplitude, or height, of the signals is different between ECG1 and ECG2. This difference can be due to different recording conditions or patient characteristics.

1.3. Necessary Filtering for Data Readability and Automatic Processing

In this section, we explain the types of filtering needed to make the ECG signals clearer and easier for automatic processing like peak detection and classification.

Types of Filtering

High-Pass Filtering:

- Purpose: To remove low-frequency noise that can hide important parts of the ECG signal. This noise can come from patient movement, breathing, or poor electrode contact.
- How It Works: A high-pass filter lets through frequencies higher than a certain point and reduces lower frequencies.

Low-Pass Filtering:

- Purpose: To remove high-frequency noise that makes the signal look shaky. This noise can come from electrical interference, muscle activity, or other sources.
- How It Works: A low-pass filter lets through frequencies lower than a certain point and reduces higher frequencies.

Why These Filters are Necessary

Improving Readability:

- The raw ECG signals have both low-frequency and high-frequency noise that make it hard to see the important parts. Using high-pass and low-pass filters helps reduce this noise and makes the signals clearer.
- For example, baseline wander (low-frequency noise) can make it hard to see the peaks accurately, while high-frequency noise can create false peaks.

Enabling Automatic Processing:

- Automatic processing techniques, like peak detection algorithms, need clear and accurate signals. Noise can cause these algorithms to miss peaks or detect false ones, making them less reliable.
- Filtering the signal to remove noise ensures that these algorithms can work correctly, detecting and classifying peaks accurately.

To improve the readability of the ECG signals and facilitate automatic processing, we need to apply both high-pass and low-pass filters. The high-pass filter will remove low-frequency noise, and the low-pass filter will remove high-frequency noise. This combined filtering approach will enhance the clarity of the ECG signals, making it easier to detect and classify heartbeats accurately.

2. Filtering the ECG

2.1. The high-pass filter

2.1.1. Transfer Function

We need to calculate the transfer function of the high-pass filter. This is done by converting the filter expression to the z-domain using the z-transform. We will refer to HP[n] as y[n]:

The filter expression is given by:

$$HP[n] = HP[n-1] - \frac{1}{32}x[n] + x[n-16] - x[n-17] + \frac{1}{32}x[n-32]$$

By applying the z-transform to this expression, we get:

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

Rearranging the terms, we have:

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

Thus, the transfer function $H(Z)$ is:

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

2.1.2. Find and Plot the Frequency Response

To find the frequency response $H(e^{jw})$, we substitute e^{jw} for Z in the transfer function. The resulting equation is:

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

Frequency Response Plot:

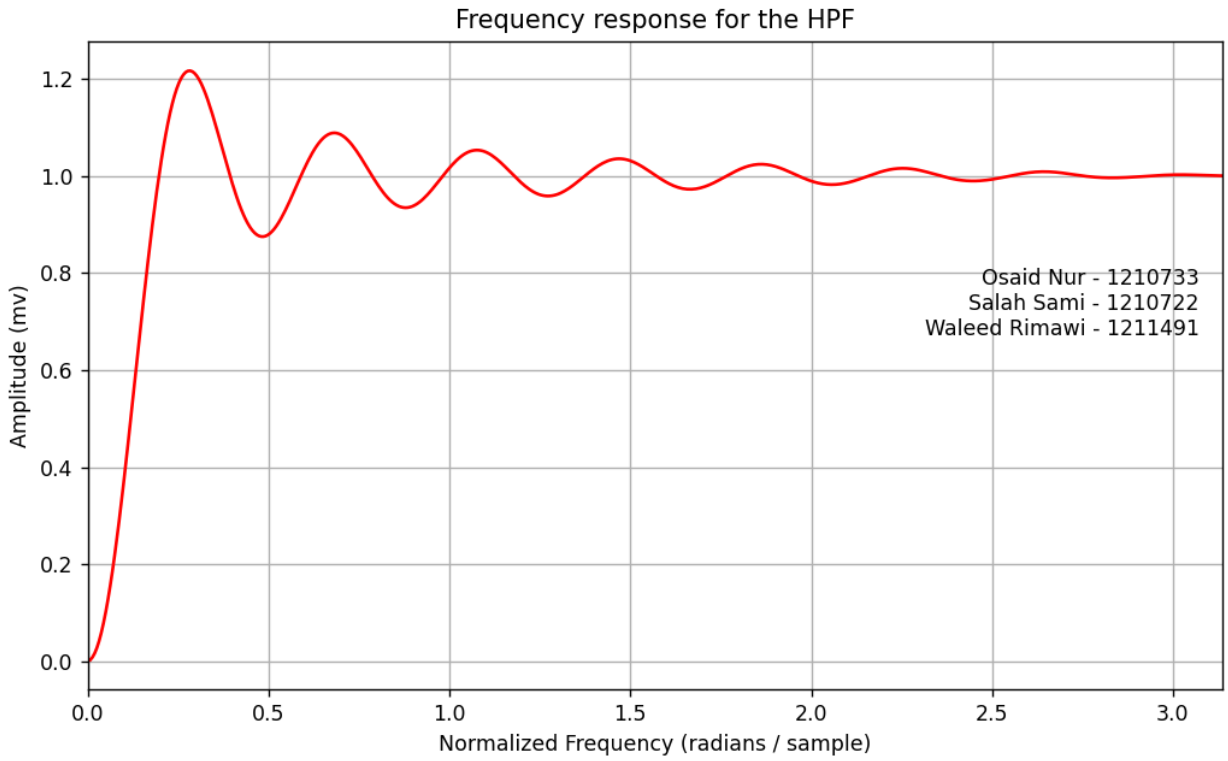


Figure 4: Frequency Response Plot

Explanation:

The frequency response plot shows that the high-pass filter effectively reduce low frequencies while allowing higher frequencies to pass. This is essential for removing low-frequency noise from the ECG signal, improving its readability for further analysis.

2.1.3. Which Family Does the High-Pass Filter Belong to?

A system is said to be an FIR (Finite Impulse Response) system if its impulse response is finite, meaning it has a limited number of non-zero terms. The transfer function H(Z) can be expressed as a polynomial that has a pole only at Z=0.

A system is said to be an IIR (Infinite Impulse Response) system if its impulse response is infinite, meaning it has an infinite number of non-zero terms. The transfer function H(Z) has poles at non-zero points, which means it can have an infinite response due to feedback.

The signal here belongs to the FIR family. The transfer function has a pole at Z=1 and at Z=0, but it also has a zero at Z=1. Therefore, the zero cancels the pole, so the transfer function will have a single pole at zero. This fulfills the conditions of an FIR filter.

Zero-Pole Plot:

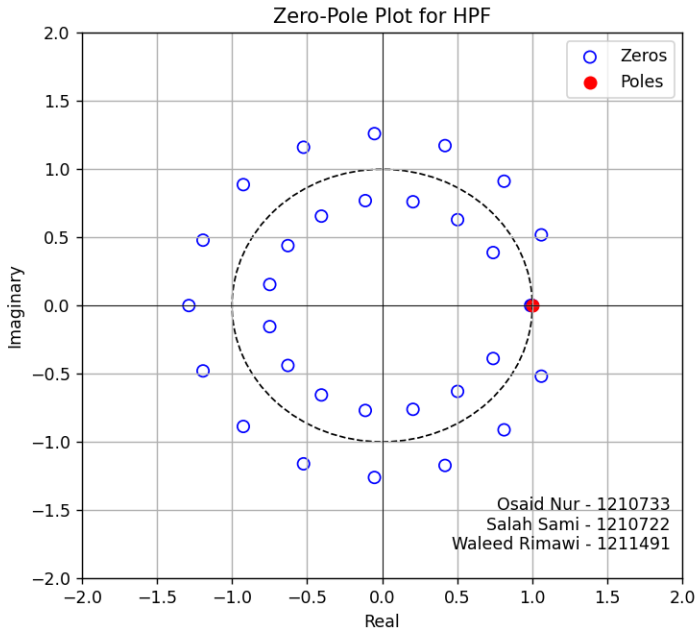


Figure 5: Zero-Pole Plot

Explanation:

- The plot shows the zeros and poles of the high-pass filter.
- Zeros are represented by blue circles, and poles are represented by red crosses.
- The presence of a zero at Z=1 cancels out the pole at the same point, leaving only a pole at Z=0, which confirms that the system is an FIR filter.

The high-pass filter belongs to the FIR family because it has a finite impulse response with a single pole at Z=0 and a zero that cancels the pole at Z=1.

2.1.4. Apply the High-Pass Filter to ECG1 and ECG2 Signals

After designing the high-pass filter, we applied it to the ECG1 and ECG2 signals. The following figures show the results.

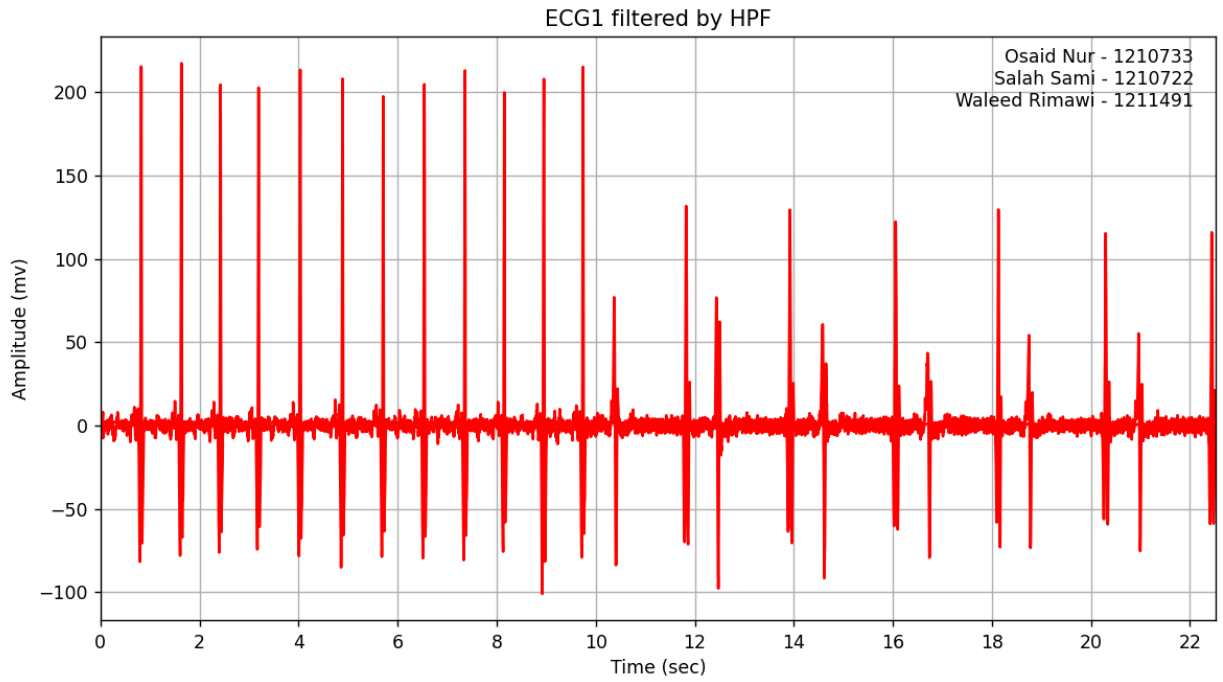


Figure 6: Filtered ECG1 Signal:

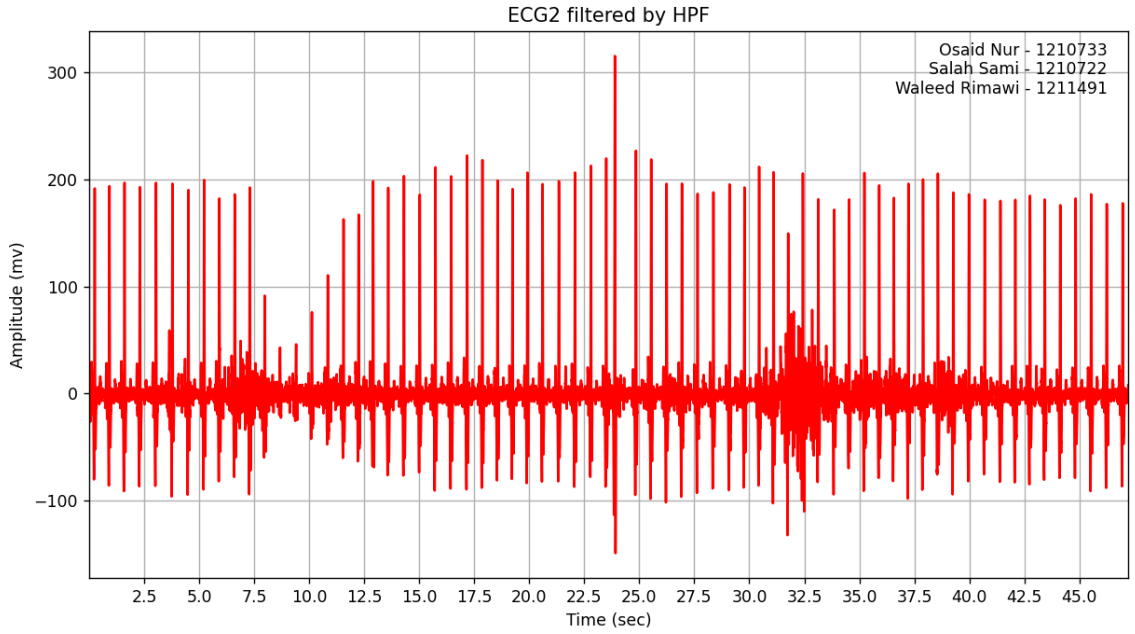


Figure 7: Filtered ECG2 Signal

Explanation:

- ECG1 Signal:
  - The filtered ECG1 signal shows the high-frequency components more prominently.
  - Low-frequency noise, such as baseline wander, is significantly reduced.
  - The periodic peaks corresponding to heartbeats are clearly visible.

- ECG2 Signal:
  - The filtered ECG2 signal also shows a significant reduction in low-frequency noise.
  - The signal is clearer, with less interference, making it easier to detect the heartbeats.

The high-pass filter effectively reduces low-frequency noise in both ECG1 and ECG2 signals. This makes the signals clearer and facilitates better detection and classification of heartbeats. The next step will involve applying the low-pass filter to further refine these signals.

2.1.5. How to Partially Resolve Problems Occurring at the Start of the Signal

When applying filters to signals, especially high-pass filters, it's common to encounter issues at the start of the signal. These issues arise because the filter requires initial conditions to start processing, and without prior data, the response at the beginning can be inaccurate or distorted.

Here are some methods to partially resolve these problems:

1. Zero-Padding:

Adding zeros to the beginning of the signal gives the filter time to settle before processing the actual signal data. This reduces the impact of initial transients.

2. Smoothing the Initial Values:

Averaging the first few samples provides a stable starting value for the filter, which helps in reducing initial distortions.

3. Use of Initial Conditions:

Using initial conditions that match the expected start of the signal can improve the filter's performance at the beginning, providing a more accurate filtered signal from the start.

These methods help in reducing the problems that occur at the start of the signal when applying a high-pass filter. By providing the filter with better initial conditions, we can achieve a more stable and accurate filtered signal.

2.2. The low-pass filter

2.2.1. Transfer Function

We calculate the transfer function of the low-pass filter by converting the filter expression to the Z domain. By applying the z-transform to the expression (we will call the output of the filter LP[n] as y[n]):

The filter expression is given by:

$$LP[n] = 2LP[n-1] - LP[n-2] + x[n] - 2x[n-6] + x[n-12]$$

Applying the z-transform, we get:

$$LP(Z) = 2LP[Z]Z^{-1} - LP[Z]Z^{-2} + x(Z) - 2x(Z)Z^{-6} + x(Z)Z^{-12}$$

Rearranging the terms, we have:

$$LP(Z)(1 - 2Z^{-1} + Z^{-2}) = x(Z)(1 - 2Z^{-6} + Z^{-12})$$

Thus, the transfer function H(Z) is:

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

2.2.2. Find and Plot the Frequency Response

To find the frequency response $H(e^{jw})$, we substitute e^{jw} for Z in the transfer function. The resulting equation is:

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

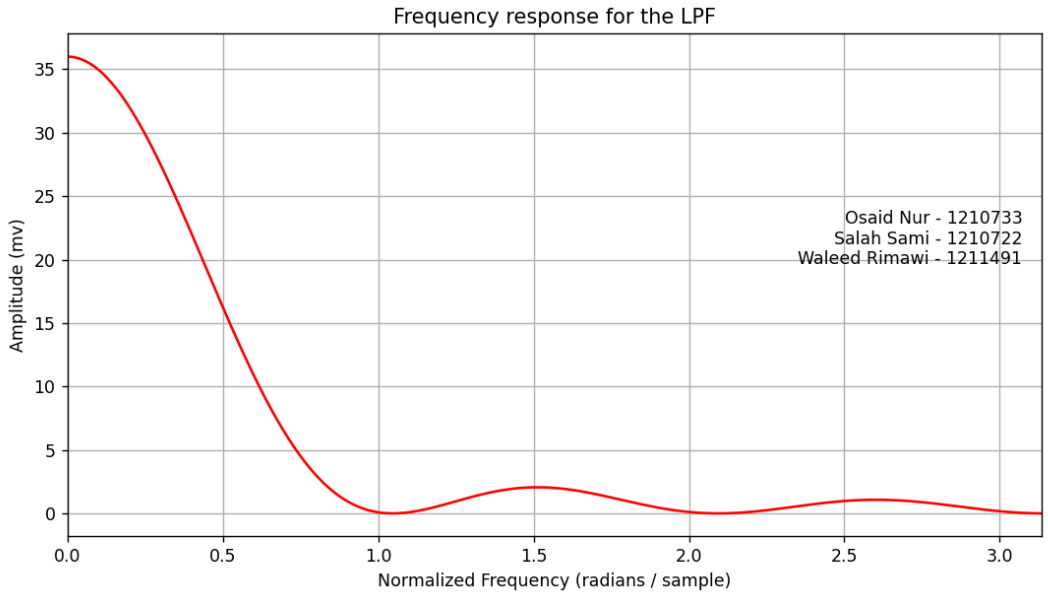


Figure 8: Frequency Response Plot

The frequency response plot shows that the low-pass filter effectively attenuates high frequencies while allowing low frequencies to pass. This is essential for removing high-frequency noise from the ECG signal, improving its readability for further analysis.

2.2.3. Which family does it belong to?

The signal here belongs to the FIR family , The transfer function has a pole at z=1 and at z=0, but it also has a zero at Z=1 , so the zero cancels the pole , so that the transfer function will have a single pole at zero so this fulfills the conditions of FIR .

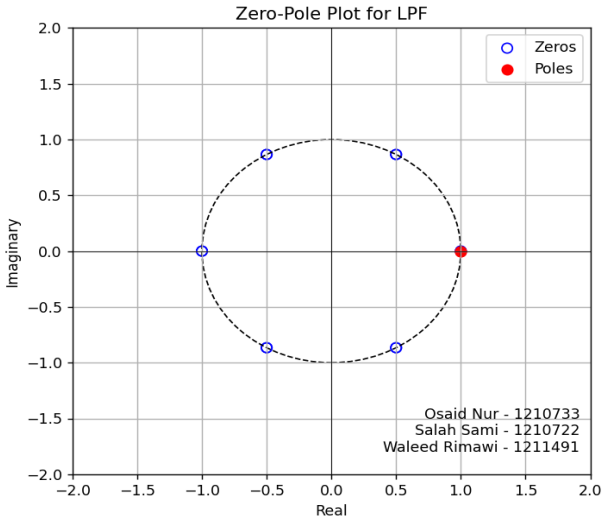


Figure 9: Zero-Pole Plot

Explanation:

- The plot shows the zeros and poles of the low-pass filter.
- Zeros are represented by blue circles, and poles are represented by red crosses.
- The presence of a zero at Z=1 cancels out the pole at the same point, leaving only a pole at Z=0, which confirms that the system is an FIR filter.

The low-pass filter belongs to the FIR family because it has a finite impulse response with a single pole at Z=0 and a zero that cancels the pole at Z=1.

2.2.4. Apply the Low-Pass Filter to ECG1 and ECG2 Signals

After designing the low-pass filter, we applied it to the ECG1 and ECG2 signals. The following figures show the results.

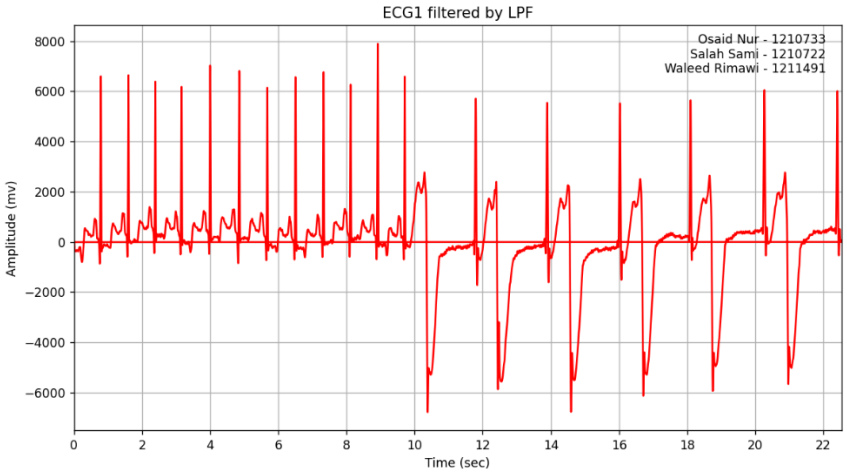


Figure 10: Filtered ECG1 Signal

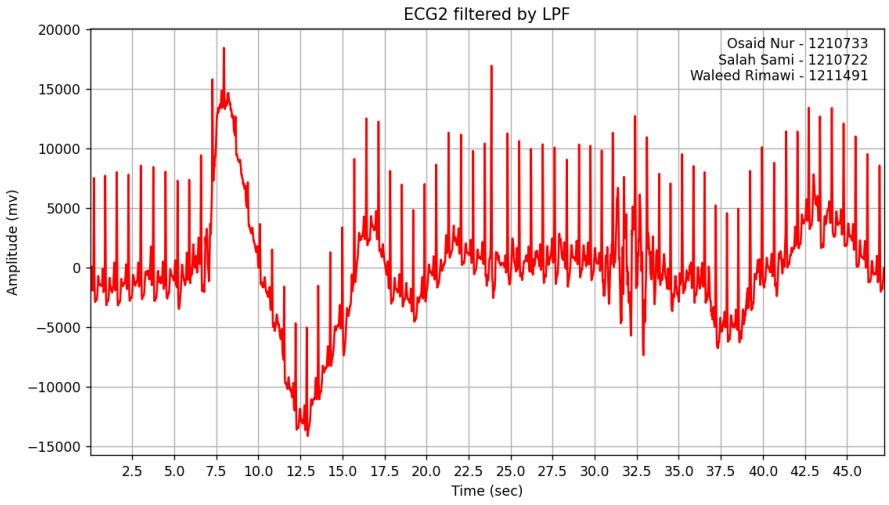


Figure 11: Filtered ECG2 Signal

Explanation:

The low-pass filter effectively reduces high-frequency noise in both ECG1 and ECG2 signals. This makes the signals clearer and facilitates better detection and classification of heartbeats. The next step will involve using the output of the high-pass filter as the input to the low-pass filter and comparing the results.

2.3. The low-pass filter

In this section, we use the output of the high-pass filter as the input for the low-pass filter and vice versa. We then display the results for ECG1 and ECG2 signals and analyze if there is any difference when the order of the filters is reversed.

2.3.1. ECG1 Signal

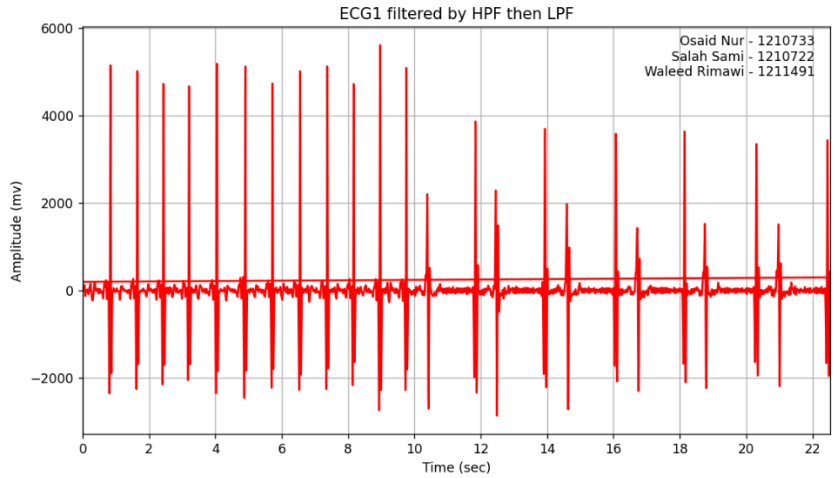


Figure 12 : ECG1 Filtered by HPF then LPF

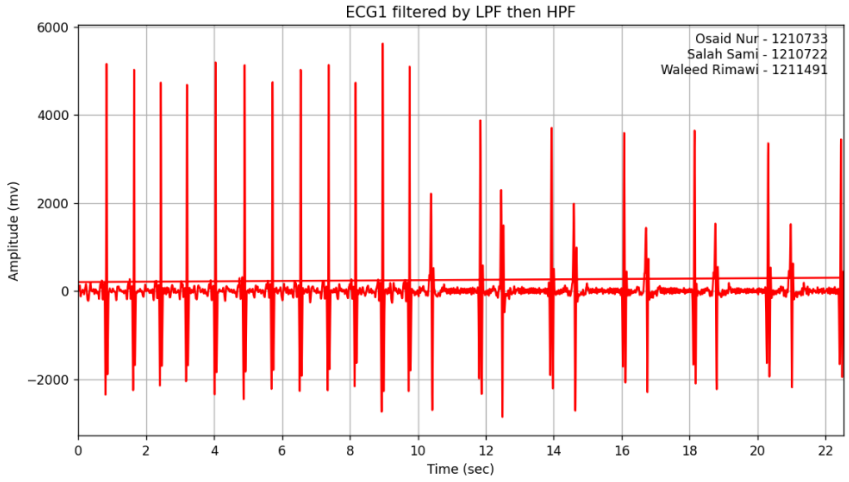


Figure 13: ECG1 Filtered by LPF then HPF

Explanation:

- **HPF then LPF:** The signal is first passed through the high-pass filter to remove low-frequency noise, followed by the low-pass filter to remove high-frequency noise.
- **LPF then HPF:** The signal is first passed through the low-pass filter to remove high-frequency noise, followed by the high-pass filter to remove low-frequency noise.

In both cases, the filtered signals show reduced noise compared to the original signal. However, the overall shape of the signal remains consistent, indicating that the order of applying the filters does not significantly affect the final result.

2.3.2. ECG2 Signal

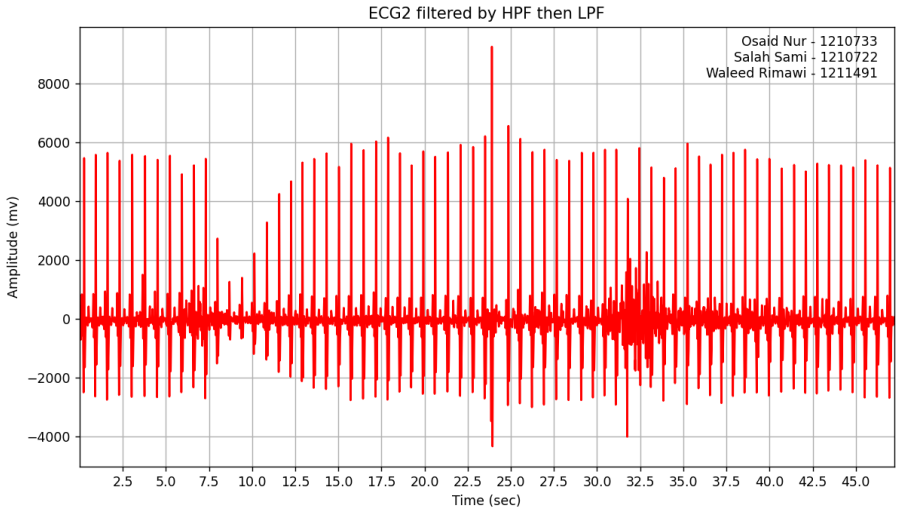


Figure 14: ECG2 Filtered by HPF then LPF

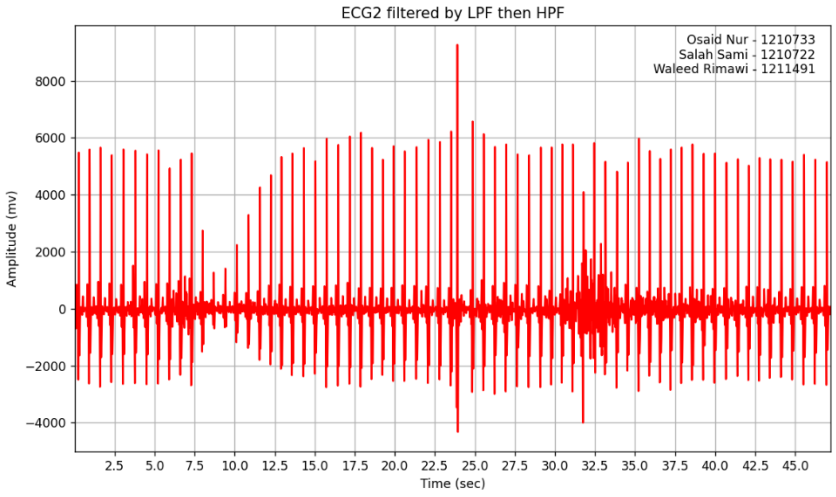


Figure 15: ECG2 Filtered by LPF then HPF

Explanation:

- **HPF then LPF:** Similar to ECG1, the signal is first processed to remove low-frequency noise, followed by high-frequency noise reduction.
- **LPF then HPF:** The signal is first processed to remove high-frequency noise, followed by low-frequency noise reduction.

For ECG2, both filtering orders produce a similar result with clear heartbeat peaks and reduced noise. This consistency further supports the observation that the order of applying the filters does not have a significant impact on the final outcome.

The results indicate that applying the high-pass filter followed by the low-pass filter, or vice versa, produces similar outcomes. This suggests that the order of applying the filters does not significantly impact the final filtered signals for both ECG1 and ECG2.

Conclusion

The application of high-pass and low-pass filters significantly improved the clarity of ECG signals by effectively reducing noise. This enhanced the visibility of important features like heartbeats, making the signals more suitable for automatic processing techniques such as peak detection. The combined filtering approach, regardless of the order, consistently produced clear and noise-free signals, demonstrating the robustness of the filtering techniques used in this project.

This project illustrates the importance and effectiveness of digital signal processing in medical signal analysis, providing a foundation for further advancements in automated health monitoring systems.

Appendix A

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from scipy.signal import freqz, lfilter,tf2zpk

# display the original two signals ECG1 and ECG2
def displayData(sheet_name):

    if sheet_name in sheets_dict:

        # get the series of x and y from the sheet
        x,y = getValues(sheet_name)

        # create the plot
        fig, ax = plt.subplots()

        # make the x axis limits fits the data
        plt.xlim(min(x), max(x))

        # set the step amount on the x axis
        plt.locator_params(axis='x', nbins=20)

        # plot the data
        ax.plot(x, y, marker=' ', linestyle='-', color='r')

        # configure the plot
        plt.title(f'{sheet_name} original')
        ax.set_xlabel('Time (sec)')
        ax.set_ylabel('Amplitude (mv)')
        plt.text(0.98, 0.98, "Osaid Nur - 1210733\nSalah Sami - 1210722\nWaleed
Rimawi - 1211491", ha='right', va='top', transform=plt.gca().transAxes)
        ax.grid(True)
        plt.tight_layout()
        plt.show()

    else:
        print(f"Sheet '{sheet_name}' not found in the Excel file !")

# get the series of x and y from the sheet
def getValues(sheet_name):
    if sheet_name in sheets_dict:
        data = sheets_dict[sheet_name]
```

```
        # convert the relevant columns to numeric values, forcing errors to NaN
        data[data.columns[2]] = pd.to_numeric(data[data.columns[2]],
errors='coerce')
        data[data.columns[4]] = pd.to_numeric(data[data.columns[4]],
errors='coerce')

        # Drop rows with NaN values in the relevant columns
        data = data.dropna(subset=[data.columns[2], data.columns[4]])

        # define the x axis and y axis , take the 4th and 2nd columns including
the normalize amplitude
        x = data[data.columns[4]]
        y = data[data.columns[2]]
        return x, y
    else:
        print(f"Sheet '{sheet_name}' not found in the Excel file !")
    return None, None

# plot the filtered data
def plotFilteredData(x, y, sheet_name, filter_type):

    # create the plot
    fig, ax = plt.subplots()

    # make the x axis limits fits the data
    plt.xlim(min(x), max(x))

    # set the step amount on the x axis
    plt.locator_params(axis='x', nbins=20)

    # plot the data
    ax.plot(x, y, marker=' ', linestyle='-', color='r')

    # configure the plot
    plt.title(f'{sheet_name} filtered by {filter_type}')
    ax.set_xlabel('Time (sec)')
    ax.set_ylabel('Amplitude (mv)')
    plt.text(0.98, 0.98, "Osaid Nur - 1210733\nSalah Sami - 1210722\nWaleed
Rimawi - 1211491", ha='right', va='top', transform=plt.gca().transAxes)
    ax.grid(True)
    plt.tight_layout()
    plt.show()

# plot the frequency response of the filter
def plotFreqResponse(b, a, filter_type):
    # get the frequency response of the filter using the built-in function
```

```
    x_values, y_values = freqz(b, a)

    # plot the frequency response
    plt.plot(x_values,abs(y_values), 'r')

    # set the x axis limits to fit the data
    plt.xlim(min(x_values), max(x_values))

    # configure the plot
    plt.text(0.98, 0.65, "Osaid Nur - 1210733\nSalah Sami - 1210722\nWaleed
Rimawi - 1211491", ha='right', va='top', transform=plt.gca().transAxes)
    plt.title(f'Frequency response for the {filter_type}')
    plt.xlabel('Normalized Frequency (radians / sample)')
    plt.ylabel('Amplitude (mv)')
    plt.grid()
    plt.show()

# plot the zero-pole plot of the filter
def plotZeroPole(b, a,filter_type):
    # Calculate zeros, poles, and gain
    zeros, poles, gain = tf2zpk(b, a)

    # Plot the zeros and poles
    plt.figure(figsize=(8, 8))

    # Plot zeros
    plt.scatter(np.real(zeros), np.imag(zeros), s=50, facecolors='none',
edgecolors='b', label='Zeros')

    # Plot poles
    plt.scatter(np.real(poles), np.imag(poles), s=50, facecolors='r',
edgecolors='r', label='Poles')

    # Plot unit circle for reference
    unit_circle = plt.Circle((0, 0), 1, color='black', fill=False,
linestyle='dashed')
    plt.gca().add_artist(unit_circle)

    # Set plot limits
    plt.xlim([-2, 2])
    plt.ylim([-2, 2])

    # Add labels, title, and legend
    plt.axhline(0, color='black',linewidth=0.5)
    plt.axvline(0, color='black',linewidth=0.5)
    plt.title(f'Zero-Pole Plot for {filter_type}')
```

```
    plt.text(0.98, 0.15, "Osaid Nur - 1210733\nSalah Sami - 1210722\nWaleed
Rimawi - 1211491", ha='right', va='top', transform=plt.gca().transAxes)
    plt.xlabel('Real')
    plt.ylabel('Imaginary')
    plt.grid()
    plt.legend()
    plt.show()

# apply the high-pass filter
def HPF(x):
    # define the filter coefficients
    b = [-1/32, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, -1, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1/32]
    a = [1, -1]

    # convert the input signal to an array of float64
    x = np.asarray(x, dtype=np.float64)

    # calculate the filtered signal using the coefficients and the input signal
    y = lfilter(b, a, x)
    return y

# apply the low-pass filter
def LPF(x):
    # define the filter coefficients
    b = [1, 0, 0, 0, 0, 0, -2, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0]
    a = [1, -2, 1]

    # convert the input signal to an array of float64
    x = np.asarray(x, dtype=np.float64)

    # calculate the filtered signal using the coefficients and the input signal
    y = lfilter(b, a, x)
    return y

# apply a single filter
def applyFilter(sheet_name, filter_type):
    # get the series of x and y from the sheet
    x, y = getValues(sheet_name)

    # apply the filter depending on the filter type
    if filter_type == "HPF":
        y_filtered = HPF(y)
    else :
        y_filtered = LPF(y)
```

```
    # plot the filtered data (the output of the filter)
    plotFilteredData(x, y_filtered, sheet_name, filter_type)

# apply two filters consecutively
def apply2Filters(sheet_name, filter1, filter2):
    # get the series of x and y from the sheet
    x, y = getValues(sheet_name)

    # apply the first filter
    if filter1 == "HPF":
        y_filtered = HPF(y)
    else :
        y_filtered = LPF(y)

    # apply the second filter
    if filter2 == "HPF":
        y_filtered = HPF(y_filtered)
    else :
        y_filtered = LPF(y_filtered)

    # plot the filtered data (the output of the filter)
    plotFilteredData(x, y_filtered, sheet_name, f'{filter1} then {filter2}')

# global dictionary to store the sheets
sheets_dict={}

# main function that read the file and display the main menu
def main():
    global sheets_dict
    # Read all sheets into a dictionary of DataFrames
    try:
        sheets_dict = pd.read_excel('Data_ECG_raw.xlsx', sheet_name=None)
        print("File read successfully!")
    except FileNotFoundError:
        print("The file Data_ECG_raw.xlsx was not found !")
        exit()

    # display the main menu
    while(1):
        print("Choose what you want to do:")
        print("1- Display the original signals")
        print("2- Display the frequency response of the filters")
        print("3- Display the zero-pole plot of the filters")
        print("4- Apply a single filter")
        print("5- Apply two filters consecutively")
```

```
        print("6- Exit")
        selection = input(">> ")

        # display the original signals
        if selection == "1":
            print("choose the signal you want to display:")
            print("1- ECG1              2- ECG2")
            num = input(">> ")
            if num=="1":
                displayData('ECG1')
            elif num=="2":
                displayData('ECG2')
            else:
                print("Invalid selection ! try again ...")

        # display the frequency response of the filters
        elif selection == "2":
            b_hpf = [-1/32, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, -1,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1/32]
            a_hpf = [1, -1]
            b_lpf = [1, 0, 0, 0, 0, 0, -2, 0, 0, 0, 0, 0, 1]
            a_lpf = [1, -2, 1]

            print("choose the filter :")
            print("1- HPF                2- LPF")
            num = input(">> ")
            if num=="1":
                plotFreqResponse(b_hpf, a_hpf, "HPF")
            elif num=="2":
                plotFreqResponse(b_lpf, a_lpf, "LPF")
            else:
                print("Invalid selection ! try again ...")

        elif selection == "3":
            b_hpf = [-1/32, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, -1,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1/32]
            a_hpf = [1, -1]
            b_lpf = [1, 0, 0, 0, 0, 0, -2, 0, 0, 0, 0, 0, 1]
            a_lpf = [1, -2, 1]
            print("choose the filter :")
            print("1- HPF                2- LPF")
            num = input(">> ")
            if num=="1":
                plotZeroPole(b_hpf, a_hpf,"HPF")
            elif num=="2":
                plotZeroPole(b_lpf, a_lpf,"LPF")
```

```
            else:
                print("Invalid selection ! try again ...")


        # apply a single filter
        elif selection == "4":
            print("Choose one of the following options:")
            print ("1- apply HPF on ECG1")
            print ("2- apply HPF on ECG2")
            print ("3- apply LPF on ECG1")
            print ("4- apply LPF on ECG2")
            sel = input(">> ")
            if sel=="1":
                applyFilter('ECG1', 'HPF')
            elif sel=="2":
                applyFilter('ECG2', 'HPF')
            elif sel=="3":
                applyFilter('ECG1', 'LPF')
            elif sel=="4":
                applyFilter('ECG2', 'LPF')
            else:
                print("Invalid selection ! try again ...")

        # apply two filters consecutively
        elif selection == "5":
            print("Choose the signal you want to display:")
            print("1- ECG1              2- ECG2")
            num = input(">> ")
            if num=="1":
                print("Choose one of the following operations:")
                print("1- apply HPF then LPF")
                print("2- apply LPF then HPF")
                sel = input(">> ")
                if sel=="1":
                    apply2Filters('ECG1', 'HPF', 'LPF')
                elif sel=="2":
                    apply2Filters('ECG1', 'LPF', 'HPF')
                else:
                    print("Invalid selection ! try again ...")
            elif num=="2":
                print("Choose one of the following operations:")
                print("1- apply HPF then LPF")
                print("2- apply LPF then HPF")
                sel = input(">> ")
                if sel=="1":
                    apply2Filters('ECG2', 'HPF', 'LPF')
```

```
                elif sel=="2":
                    apply2Filters('ECG2', 'LPF', 'HPF')
                else:
                    print("Invalid selection ! try again ...")
            else:
                print("Invalid selection ! try again ...")

        # exit the program
        elif selection == "6":
            break

        # invalid selection
        else:
            print("Invalid selection ! try again ...")

if __name__ == "__main__":
    main()
```

Appendix B :

Sample Run :

In our code , we made a main menu that display all operations required in the project , and we added an additional option to display the zero pole plot for every filter , this will help us determine the family that each filter belong to .

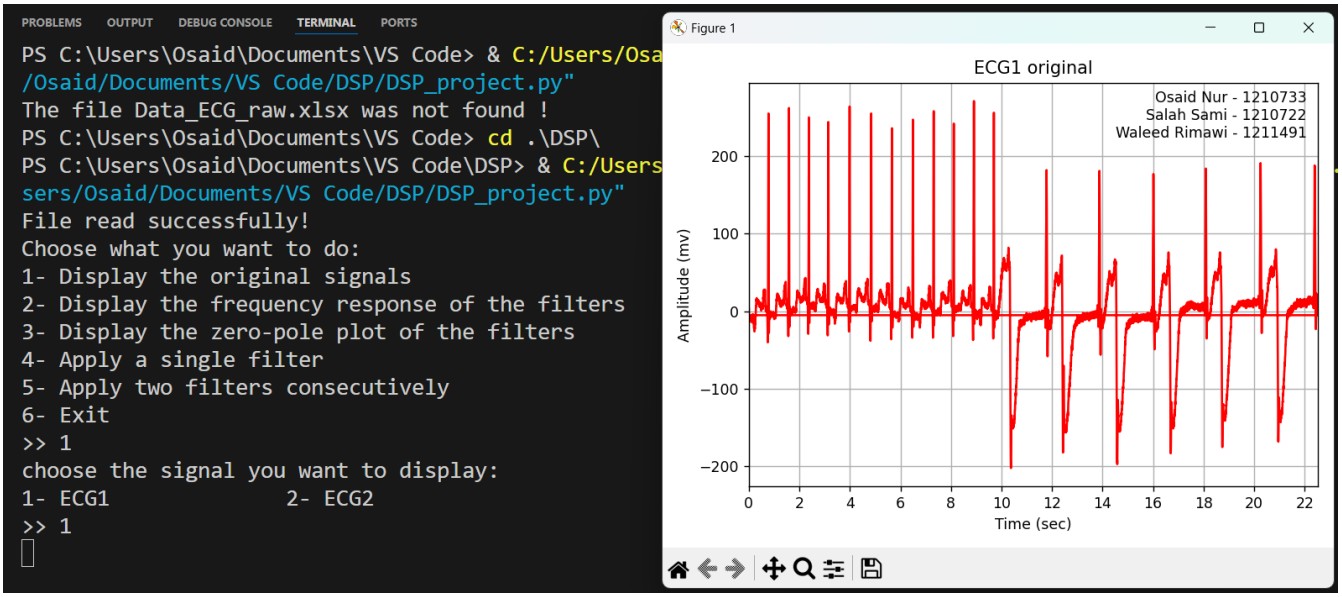


Figure 16.Sample Run 1

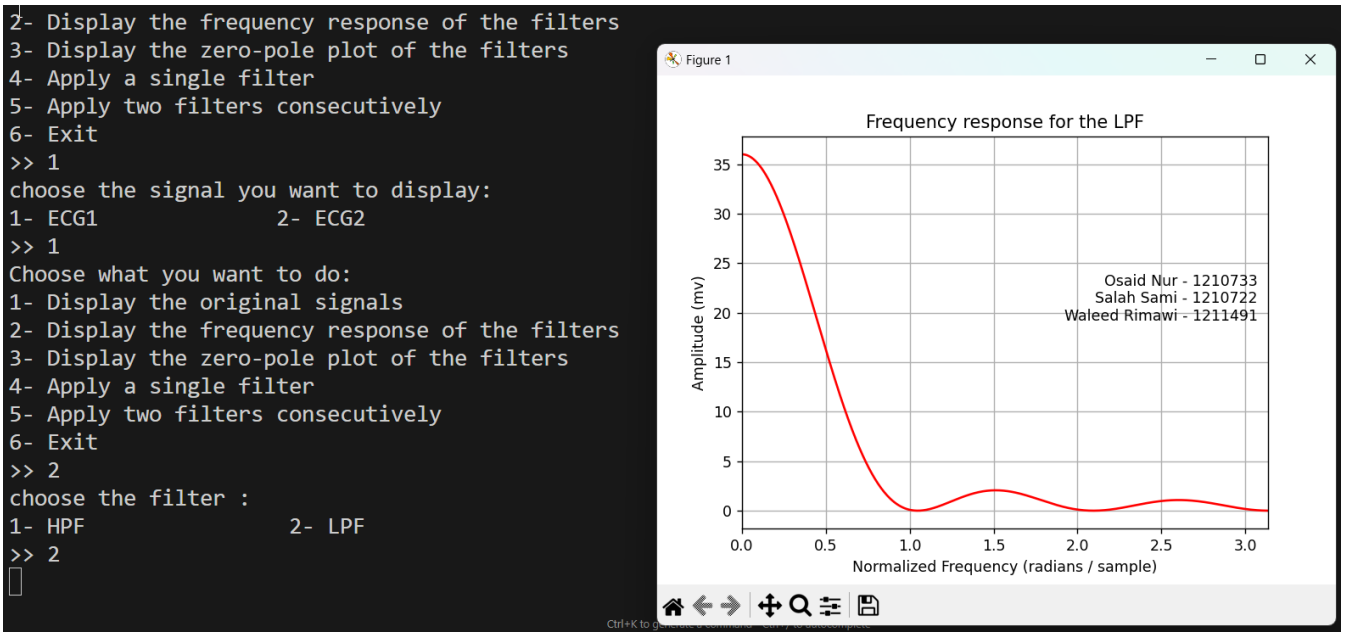


Figure 17..Sample Run 2

```
2- Display the frequency response of the filters
3- Display the zero-pole plot of the filters
4- Apply a single filter
5- Apply two filters consecutively
6- Exit
>> 2
choose the filter :
1- HPF                2- LPF
>> 2
Choose what you want to do:
1- Display the original signals
2- Display the frequency response of the filters
3- Display the zero-pole plot of the filters
4- Apply a single filter
5- Apply two filters consecutively
6- Exit
>> 3
choose the filter :
1- HPF                2- LPF
>> 1
```

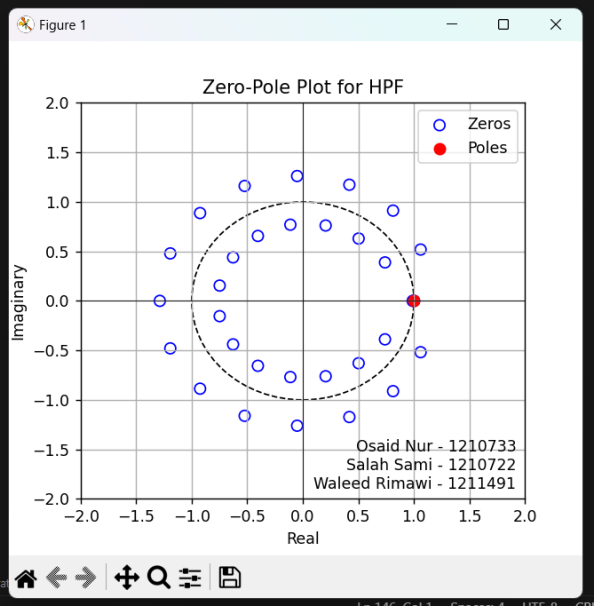


Figure 18..Sample Run 3

```
5- Apply two filters consecutively
6- Exit
>> 3
choose the filter :
1- HPF                2- LPF
>> 1
Choose what you want to do:
1- Display the original signals
2- Display the frequency response of the filters
3- Display the zero-pole plot of the filters
4- Apply a single filter
5- Apply two filters consecutively
6- Exit
>> 4
Choose one of the following options:
1- apply HPF on ECG1
2- apply HPF on ECG2
3- apply LPF on ECG1
4- apply LPF on ECG2
>> 1
```

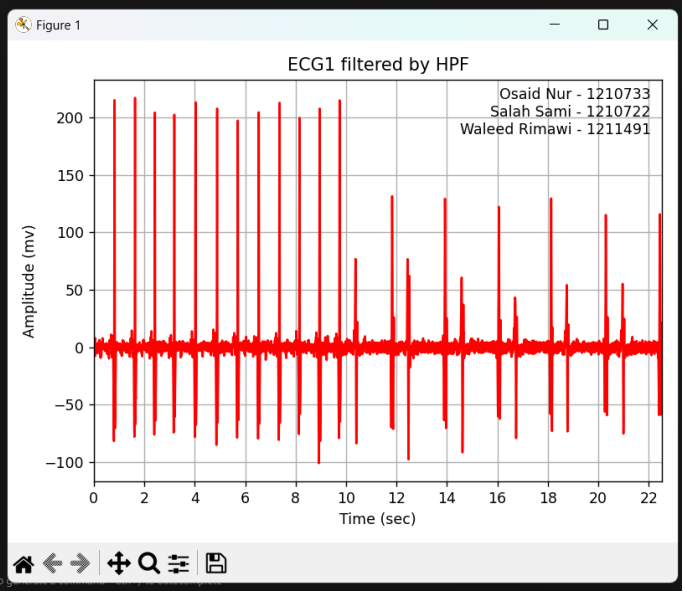


Figure 19..Sample Run 4

```
Choose one of the following options:
1- apply HPF on ECG1
2- apply HPF on ECG2
3- apply LPF on ECG1
4- apply LPF on ECG2
>> 1
Choose what you want to do:
1- Display the original signals
2- Display the frequency response of the filters
3- Display the zero-pole plot of the filters
4- Apply a single filter
5- Apply two filters consecutively
6- Exit
>> 4
Choose one of the following options:
1- apply HPF on ECG1
2- apply HPF on ECG2
3- apply LPF on ECG1
4- apply LPF on ECG2
>> 3
```

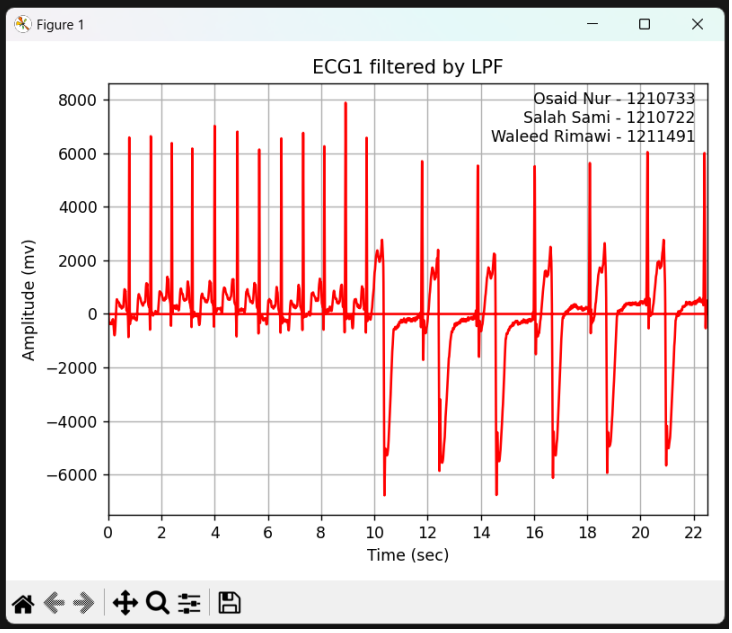


Figure 20..Sample Run 5

```
1- apply HPF on ECG1
2- apply HPF on ECG2
3- apply LPF on ECG1
4- apply LPF on ECG2
>> 3
Choose what you want to do:
1- Display the original signals
2- Display the frequency response of the filters
3- Display the zero-pole plot of the filters
4- Apply a single filter
5- Apply two filters consecutively
6- Exit
>> 5
Choose the signal you want to display:
1- ECG1              2- ECG2
>> 1
Choose one of the following operations:
1- apply HPF then LPF
2- apply LPF then HPF
>> 1
```

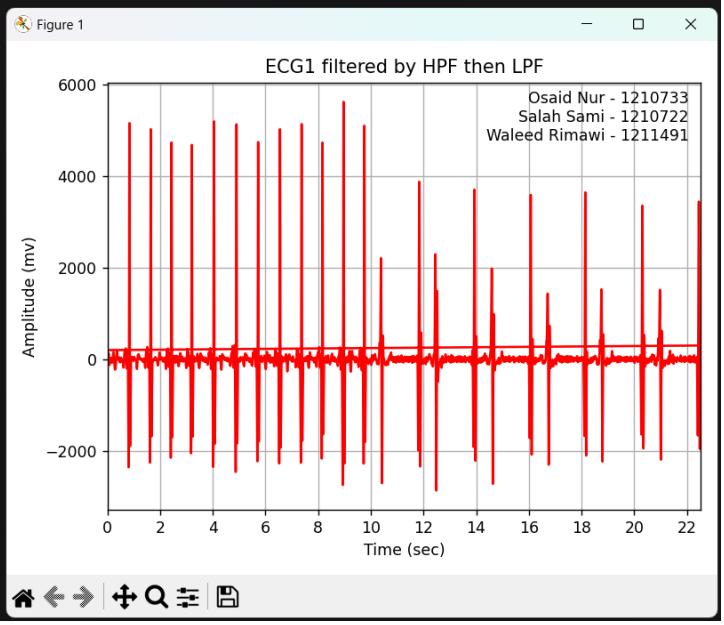


Figure 21..Sample Run 6

```
>> 1
Choose one of the following operations:
1- apply HPF then LPF
2- apply LPF then HPF
>> 1
Choose what you want to do:
1- Display the original signals
2- Display the frequency response of the filters
3- Display the zero-pole plot of the filters
4- Apply a single filter
5- Apply two filters consecutively
6- Exit
>> 5
Choose the signal you want to display:
1- ECG1              2- ECG2
>> 1
Choose one of the following operations:
1- apply HPF then LPF
2- apply LPF then HPF
>> 2
```

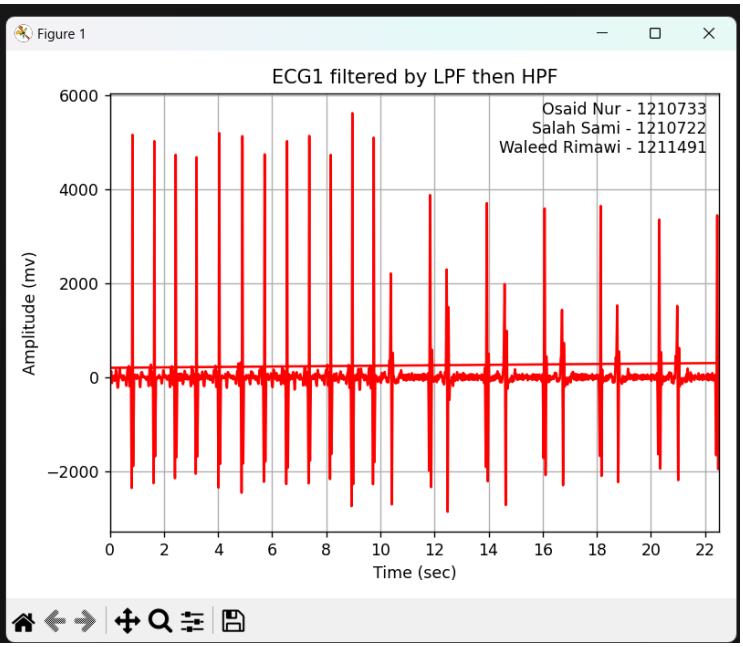


Figure 22..Sample Run 7