FINAL PROJECT INDIVIDUAL REPORT

Respiratory Rate and Vasomotor Activity Extraction of Photoplethysmography (PPG) Signal using DWT Method

Name	: Jeremia Christ Immanuel Manalu
NRP	: 5023231017
Course	: Non-Stationary Signal Analysis
Class	: A
Lecturer	: Nada Fitrieyatul Hikmah, S.T., M.T.
Department	: Biomedical Engineering

FACULTY OF INTELLIGENT ELECTRICAL AND INFORMATICS TECHNOLOGY

INSTITUT TEKNOLOGI SEPULUH NOPEMBER

2025

CHAPTER I. FUNDAMENTAL THEORY

1.1. PPG Signal

Photoplethysmography (PPG) is a low-cost, non-invasive optical technique used to detect volumetric changes in blood in the peripheral circulation. The fundamental principle relies on the absorption and reflection of light by biological tissues, described by the Beer-Lambert law. When a light source illuminates the skin, the intensity of the reflected or transmitted light varies according to the blood volume within the region. The resulting waveform consists of two primary components, a quasi-static DC component, which corresponds to the optical properties of the tissues, venous blood, and steady blood volume; and a pulsatile AC component, which is synchronous with the cardiac cycle.

The AC component is of primary interest in signal analysis as it reflects the systolic and diastolic phases of the heart. During systole, blood volume increases, leading to higher light absorption, while diastole results in lower absorption. The morphology of the PPG signal contains the systolic peak, the dicrotic notch, and the diastolic peak, which provide valuable information regarding arterial stiffness and vascular compliance. In digital signal processing, the raw PPG signal is often contaminated by motion artifacts, power line interference, and baseline wander, necessitating robust preprocessing techniques to ensure the accuracy of subsequent physiological parameter extraction. The components of the PPG signal can bee seen in **Figure 1.1** below.

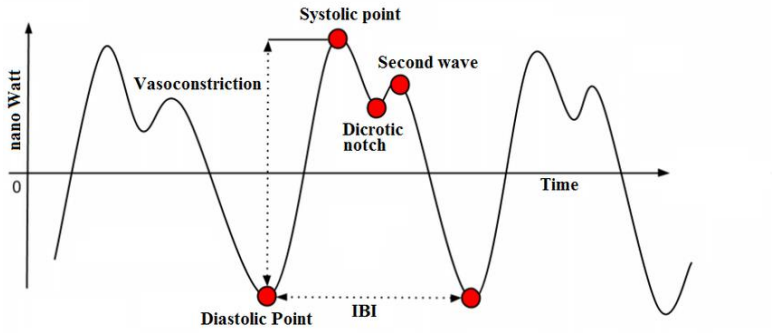


Figure 1.1. *An example of PPG Signal and Its Components*

1.2. Respiratory Rate

Respiration significantly modulates the photoplethysmogram through three primary mechanisms: baseline wander induced by intrathoracic pressure changes, amplitude modulation (AM) caused by a reduction in cardiac output during inspiration, and frequency modulation (FM) known as Respiratory Sinus Arrhythmia (RSA). To extract the Respiratory Rate (RR) from the PPG signal, analysis is focused on the High-Frequency (HF) band. According to physiological standards, the respiratory frequency in resting adults typically falls within the range of 0.15 Hz to 0.4 Hz (corresponding to 9 to 24 breaths per minute).

In the context of non-stationary signal analysis, extracting the respiratory rate involves isolating this specific frequency band from the composite PPG signal. By applying band-limiting filters or spectral decomposition methods, the dominant frequency component within the 0.15–0.4 Hz range can be identified. This frequency peak represents the fundamental respiratory rate. The accuracy of this extraction is heavily dependent on the

signal quality and the effective removal of the lower-frequency vasomotor components and higher-frequency cardiac harmonics that lie outside this specific spectral window.

1.3. Vasomotor Activity

Vasomotor activity refers to the constriction and dilation of blood vessels, regulated by the sympathetic nervous system and local factors to manage blood pressure and thermoregulation. This activity manifests in the PPG signal as low-frequency oscillations, distinct from the cardiac and respiratory rhythms. Analysis of vasomotor activity is concentrated in the 0.004 Hz to 0.15 Hz frequency range. This range encapsulates both the Very Low Frequency (VLF) band (0.004–0.04 Hz), often associated with thermoregulation and the renin-angiotensin system, and the Low Frequency (LF) band (0.04–0.15 Hz), which is linked to the baroreflex feedback loop and Mayer waves.

Analyzing the spectral power within this 0.004–0.15 Hz band allows for the assessment of sympathetic autonomic modulation. In signal processing terms, extracting the highest frequency or the dominant power peak within this range provides a quantitative metric of vasomotor tone. Unlike the respiratory signal which is parasympathetically mediated, the vasomotor components are inherently non-stationary and slower, requiring sufficiently long recording durations and methods capable of resolving low-frequency dynamics, such as the Discrete Wavelet Transform or Power Spectral Density estimation, to accurately characterize the highest frequency of vasomotor oscillations.s

1.4. Discrete Wavelet Transform (DWT)

The Discrete Wavelet Transform (DWT) is a powerful mathematical tool for analyzing non-stationary signals, providing a time-frequency representation that overcomes the fixed resolution limitations of the traditional Fourier Transform. The DWT decomposes a signal $x[n]$ into a set of basis functions, which are dilated and translated versions of a single prototype function called the mother wavelet, $\psi(t)$. In this analysis, the Morlet wavelet is often cited as a theoretical basis due to its sinusoidal shape modulated by a Gaussian envelope, making it highly suitable for capturing biological oscillations.

The practical implementation of the DWT is typically achieved using the Mallat Algorithm, which utilizes a computationally efficient filter bank structure. The algorithm decomposes the signal through a series of high-pass filters ($g[n]$) and low-pass filters ($h[n]$). At each level of decomposition j, the signal is convolved with these filters to produce Detail Coefficients (D_j), representing high-frequency information, and Approximation Coefficients (A_j), representing low-frequency trends. The recursive decomposition equation for the approximation coefficients $a_{j+1}[k]$and detail coefficients $d_{j+1}[k]$ is given by:

$$a_{j+1}[k] = \sum_n h[n-2k]a_j[n]$$

$$d_{j+1}[k] = \sum_n g[n-2k]a_j[n]$$

In the specific context of analyzing physiological signals, the DWT allows for the separation of signal components based on their frequency content. The coefficients obtained from the filter bank (specifically the detail coefficients at different scales 2^j) correspond to specific frequency bands. By examining the energy distribution or the time-domain waveform of these coefficients, one can isolate features such as the dicrotic notch

or specific frequency rhythms associated with autonomic function. The frequency response of the filter bank at level j effectively acts as a bandpass filter, isolating signal structures that correlate with the scale of the wavelet at that level. The DWT filter coefficients (until Q8) for Morlet (as Mother Wavelet) can be seen in **Figure 1.2** below.

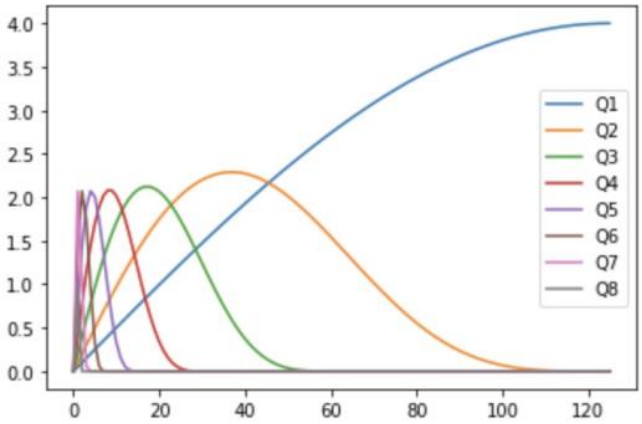


Figure 1.3. *An Example of the DWT Frequency Response using Morlet as Mother Wavelet*

1.5. Signal Preprocessing

Signal preprocessing is a critical initial stage in digital signal processing to enhance the signal-to-noise ratio (SNR) before feature extraction. The raw PPG signal often contains a significant DC offset and baseline drift caused by respiration or subject movement. To address this, a baseline wander removal technique is employed, often utilizing a Moving Average filter. This process estimates the low-frequency trend of the signal by averaging neighboring samples within a defined window and subtracting this trend from the original signal to yield a zero-mean AC signal.

Following baseline removal, a bandpass filtering stage is essential to isolate the physiological frequencies of interest (typically between 0.5 Hz and 8.0 Hz for the cardiac component). This is achieved using Infinite Impulse Response (IIR) filters, specifically a cascade of a Low-Pass Filter (LPF) and a High-Pass Filter (HPF). Additionally, Linear Detrending is applied to remove any remaining linear trends or slopes in the signal. If a signal segment $x[n]$ exhibits a linear trend $y = mx + c$, parameters m (slope) and c (intercept) are estimated using the least-squares method:

$$E = \sum_{n=0}^{N-1} (x[n] - (mn + c))^2$$

By minimizing the error E, the trend is identified and subtracted, resulting in a stationary signal suitable for spectral analysis and peak detection.

1.6. Peak Detection

Peak detection is the algorithmic process of identifying local maxima (systolic peaks), minima (diastolic troughs), and zero-crossings within the preprocessed signal. The fundamental approach relies on derivative-based logic or Zero-Crossing Analysis. A zero-crossing occurs when the signal amplitude transitions from positive to negative (or vice versa) relative to a zero reference line. Once a zero-crossing is detected, the algorithm searches for the global maximum or minimum within the interval between two crossings.
To ensure the physiological validity of the detected peaks, a refractory period is implemented. This logic mimics the biological refractory period of the heart muscle,

during which a second contraction cannot occur. Mathematically, if a tentative peak is detected at time t_i, any subsequent peak detected at time t_j is rejected if $(t_j - t_i) < T_{ref}$, where T_{ref} is a minimum time threshold (e.g., 300 ms). This prevents the algorithm from falsely identifying dicrotic notches or noise artifacts as true systolic peaks, thereby ensuring accurate Heart Rate Variability (HRV) calculation.

1.7. Fast Fourier Transform (FFT)

The Discrete Fourier Transform (DFT) transforms a discrete time-domain signal $x[n]$ of length N into its frequency-domain representation $X[k]$. The standard DFT equation is defined as:

$$X[k] = \sum_{n=0}^{N-1} x[n] W_N^{kn}, \text{ where } W_N = e^{-j\frac{2\pi}{N}}$$

Here, W_N is known as the Twiddle Factor. A direct computation of this summation for all k requires N^2 complex multiplications, which is computationally prohibitive for real-time applications. To address this, the Cooley-Tukey Radix-2 Decimation-in-Time (DIT) algorithm is employed. This algorithm requires that the signal length N be a power of 2 (i.e., $N = 2^M$).

The core principle of the Radix-2 FFT is the "divide and conquer" strategy. The summation in the DFT equation is split into two parts: one for the even-indexed samples ($2r$) and one for the odd-indexed samples ($2r + 1$):

$$X[k] = \sum_{r=0}^{\left(\frac{N}{2}\right)-1} x[2r] W_N^{2rk} + \sum_{r=0}^{\left(\frac{N}{2}\right)-1} x[2r+1] W_N^{(2r+1)k}$$

By utilizing the property $W_N^{2rk} = W_{\frac{N}{2}}^{rk}$, the equation can be rewritten as a combination of two smaller DFTs of size $\frac{N}{2}$:

$$X[k] = \underbrace{\sum_{r=0}^{\left(\frac{N}{2}\right)-1} x[2r] W_{\frac{N}{2}}^{rk}}_{E[k]} + W_N^k \underbrace{\sum_{r=0}^{\left(\frac{N}{2}\right)-1} x[2r+1] W_{\frac{N}{2}}^{rk}}_{O[k]}$$

$$X[k] = E[k] + W_N^k O[k]$$

Where $E[k]$ is the DFT of the even-indexed part and $O[k]$ is the DFT of the odd-indexed part. Due to the periodicity of the DFT ($E\left[k + \frac{N}{2}\right] = E[k]$ and $O\left[k + \frac{N}{2}\right] = O[k]$) and the symmetry of the Twiddle factor ($W_N^{k+\frac{N}{2}} = -W_N^k$), the computation for the second half of the frequencies can be derived without recalculating the sums:

$$X\left[k + \frac{N}{2}\right] = E[k] - W_N^k O[k]$$

The pair of equations above constitutes the Butterfly Operation, which is the fundamental computational unit of the FFT. By recursively applying this structure, the complexity is reduced from $O(N^2)$ to $O(N \log_2 N)$. In the software implementation, the input data array is first reordered using Bit-Reversal Permutation. This reordering swaps elements based on the binary reversal of their indices (for example, index 001_2 swaps with 100_2), allowing the FFT to be computed "in-place" with $O(1)$ auxiliary storage, effectively optimizing memory usage during signal processing.

1.8. Power Spectral Density (PSD)

While the FFT provides the magnitude spectrum, the Power Spectral Density (PSD) describes how the power of a signal is distributed over frequency. For non-stationary biological signals, Welch's Method is preferred over the standard periodogram to reduce the variance of the spectral estimate. Welch's method divides the signal into overlapping segments, applies a window function (such as a Hanning window) to each segment to reduce spectral leakage, calculates the periodogram for each segment, and finally averages them. The modified periodogram for a segment $x_m[n]$ is given by:

$$P_m(f) = \frac{1}{NU}\left|\sum_{n=0}^{N-1} x_m[n]w[n]e^{-j2\pi fn}\right|^2$$

Where $w[n]$ is the window function and U is a normalization factor for the power of the window. The final Welch PSD estimate is the average of these periodograms:

$$\hat{P}_{Welch}(f) = \frac{1}{K}\sum_{m=0}^{K-1} P_m(f)$$

This averaging process smooths the noise in the spectrum, making the peaks corresponding to the Vasomotor (LF/VLF) and Respiratory (HF) activities distinct and measurable.

CHAPTER II. RESULT AND ANALYSIS

2.1. Problems Statement

Find the following parameters extraction value of the PPG signal using DWT algorithm (preferably Mallat Filter Bank):

1. Finding the Respiratory Rate of the given PPG signal, with the frequency range of Respiratory Rate is in high frequency (HF) or 0,15 – 0,40 Hz.
2. Finding the Vasomotor Activity peak frequency value of the given PPG signal, with the frequency range of Vasomotor Activity is in low frequency (LF) or 0,04 – 0,15 Hz.

[IMPORTANT NOTES BEFORE CONTINUING TO THE NEXT SECTIONS]

*Because I developed the program on **Delphi 12** (from group project), and this program also have the **Group Final Project Assignment** in it, the next code explanation and the program results will have the **same exact GUI** as the one in the Group Final Project Assignment.*

*The dataset that I will used in this Chapter are my own PPG data, named **jeremia-data.csv**, that have the column format: **Index** as the first column and **Amplitude (0-4096)** as the second column. The format of that dataset can be seen below:*

Index,Amplitude (0-4096)

0,1089

1,1328

2,1520

3,1666

...

14999,929

2.2. Code Explanation

The implementation for this project is developed using the Delphi 12, designed to process non-stationary PPG signals for the precise extraction of physiological parameters. The system is architected as a sequential pipeline that transforms raw time-series data into frequency-domain features, specifically targeting the Respiratory Rate and Vasomotor Activity. The steps integrates standard Digital Signal Processing (DSP) techniques with the DWT to handle the inherent non-stationarity of biological signals. The processing chain is divided into four distinct stages:

1. **Data Acquisition and Time Standardization.** The pipeline begins by ingesting raw PPG data from CSV files. A critical initial step is the standardization of the time domain. Regardless of the sampling intervals present in the raw file, the system enforces a fixed sampling frequency (F_s) of 50 Hz. This uniform resampling is mathematically essential to ensure the validity of subsequent Discrete Fourier Transform (FFT) operations and Wavelet decompositions, effectively treating the input as a discrete signal $x[n]$ with a constant time step Δt.

2. **Signal Preprocessing and Conditioning**. Before analysis, the signal undergoes rigorous conditioning to remove artifacts. The system applies a Baseline Wander Removal algorithm using a Moving Average convolution to eliminate low-frequency DC drift caused by subject movement or breathing mechanics. Subsequently, a custom Infinite Impulse Response (IIR) Bandpass Filter (0.01 Hz – 8.0 Hz) is applied. This stage isolates the physiological AC components of the PPG signal while attenuating high-frequency noise and sensor artifacts, resulting in a clean "Preprocessed Signal."

3. **Discrete Wavelet Decomposition (DWT)**. To analyze the signal at different resolutions, the preprocessed signal is decomposed using the Discrete Wavelet Transform. The system implements a filter bank structure (based on the Mallat algorithm) to separate the signal into eight distinct levels of coefficients (Q1 to Q8). Each level corresponds to a specific frequency band. This multi-resolution analysis allows the system to isolate the specific signal components relevant to respiration (typically found in mid-range levels) and vasomotor activity (found in deeper, lower-frequency levels) for visualization and further analysis.

4. **Spectral Analysis and Rate Extraction**. The final stage involves quantifying the physiological rates. The system employs Welch's Method to estimate the Power Spectral Density (PSD) of the processed signal. By analyzing specific frequency bands defined by physiological standards, 0.15–0.4 Hz for Respiratory Rate and 0.04–0.15 Hz for Vasomotor Activity, the algorithm identifies the dominant spectral peaks. A peak refinement technique using parabolic interpolation is then used to determine the exact frequency of these peaks, providing a precise numerical value for the respiratory rate (in breaths per minute) and the highest vasomotor frequency (in Hz).

a. Data Loading and Time Vector Generation

- ***Procedure:*** *TForm1.LoadCSVData*
- ***Location:*** *Unit1.pas*

```
1.  function TForm1.LoadCSVData(const FileName, ColumnName: string):
    Boolean;
2.  var
3.    SL: TStringList; i, colIndex, timeColIndex, startIndex: Integer;
4.    line, colName: string; parts: TArray<string>;
5.    tempSignal, tempTime: TList<Double>;
6.    hasHeader: Boolean; val: Double;
7.  begin
8.    Result := False; colIndex := -1; timeColIndex := -1; hasHeader := True;
9.    SL := TStringList.Create;
10.   tempSignal := TList<Double>.Create;
11.   tempTime := TList<Double>.Create;
12.   try
13.     SL.LoadFromFile(FileName);
14.     if SL.Count < 2 then raise Exception.Create('File is empty or has only a
    header.');
15.     line := SL[0];
16.     parts := line.Split([',', ';']);
17.     if not TryStrToFloat(Trim(parts[0]), val) then
```

```
18.    begin
19.      hasHeader := True;
20.      for i := 0 to High(parts) do
21.      begin
22.        colName := AnsiLowerCase(Trim(StringReplace(parts[i], "", ",
    [rfReplaceAll])));
23.        if (timeColIndex = -1) and (Pos('index', colName) > 0) then timeColIndex
    := i;
24.        if (colIndex = -1) and (colName = AnsiLowerCase(Trim(ColumnName)))
    then colIndex := i;
25.      end;
26.    end
27.    else
28.    begin
29.      hasHeader := False;
30.      Log('CSV file detected without a header. Assuming Column 0 = Time,
    Column 1 = Signal.');
31.      timeColIndex := 0; colIndex := 1;
32.    end;
33.    if colIndex = -1 then colIndex := High(parts);
34.    if timeColIndex = -1 then timeColIndex := 0;
35.
36.    startIndex := IfThen(hasHeader, 1, 0);
37.    for i := startIndex to SL.Count - 1 do
38.    begin
39.      line := SL[i];
40.      parts := line.Split([',', ';']);
41.      if (High(parts) >= colIndex) then
42.      begin
43.        if
    TryStrToFloat(StringReplace(parts[colIndex],'.',FormatSettings.DecimalSepara
    tor,[]), val) then
44.          tempSignal.Add(val);
45.      end;
46.    end;
47.
48.    if tempSignal.Count < 10 then raise Exception.Create('Not enough valid data
    points found.');
49.
50.    // FORCE TIME GENERATION BASED ON 50 Hz
51.    // This fixes the "Index vs Seconds" issue
52.    FOriginalFs := 50.0;
53.    FSignalRaw := tempSignal.ToArray;
54.    SetLength(FTimeRaw, Length(FSignalRaw));
55.    for i := 0 to High(FTimeRaw) do
56.      FTimeRaw[i] := i / FOriginalFs;
```

```
57.
58.    Result := True;
59.  except
60.    on E: Exception do
61.    begin
62.      Log('ERROR: ' + E.Message);
63.      Result := False;
64.    end;
65.  end;
66.  SL.Free; tempSignal.Free; tempTime.Free;
67. end;
```

This function is responsible for importing the raw PPG data from a CSV file. The logic first iterates through the file lines, parsing the specific column containing the PPG amplitude values. A crucial step occurs in the second block of the code: the generation of the Time Vector (*FTimeRaw*). Instead of relying on potentially irregular timestamps from the file, the system enforces a fixed sampling frequency (F_s) of **50.0 Hz**. The time for each sample i is calculated using the linear equation $t[i] = i/F_s$. This ensures that the signal is treated as uniformly sampled, which is a mandatory prerequisite for the subsequent Fourier Transform (FFT) and Wavelet (DWT) analysis.

b. Signal Preprocessing (Filtering)

- ***Function:*** *TPPGAnalyzer.PreprocessSignal (Calls Filter_CustomRef)*
- ***Location:*** *PPGAnalyzer.pas*

```
1.  function TPPGAnalyzer.PreprocessSignal(const Signal: TSignalArray; Fs:
    Double): TSignalArray;
2.  var
3.    i: Integer;
4.    signalNoDC, filteredSignal, normalizedSignal, baseline, window:
    TSignalArray;
5.    windowSize: Integer;
6.    std: Double;
7.  begin
8.    // 1. Baseline Wander Removal (Moving Average)
9.    windowSize := Round(1.0 * Fs);
10.  if (windowSize mod 2) = 0 then Inc(windowSize);
11.
12.  if Length(Signal) > windowSize then
13.  begin
14.    SetLength(window, windowSize);
15.    for i := 0 to windowSize-1 do window[i] := 1.0 / windowSize;
16.
17.    baseline := Convolve(Signal, window);
18.    SetLength(signalNoDC, Length(Signal));
19.    for i := 0 to High(Signal) do
20.      signalNoDC[i] := Signal[i] - baseline[i];
21.  end
22.  else
```

```
23.    signalNoDC := Copy(Signal);
24.
25.  // 2. Apply Custom Filter Baru (0.5 Hz - 8.0 Hz)
26.  // Ini memanggil fungsi Filter_CustomRef yang baru dibuat
27.  filteredSignal := Filter_CustomRef(signalNoDC, Fs, 0.5, 8.0);
28.
29.  // 3. Normalisasi Amplitudo
30.  std := StdDev(filteredSignal);
31.  if std > 0 then
32.  begin
33.    SetLength(normalizedSignal, Length(filteredSignal));
34.    for i := 0 to High(filteredSignal) do
35.      normalizedSignal[i] := filteredSignal[i] / std;
36.    Result := normalizedSignal;
37.  end
38.  else
39.    Result := filteredSignal;
40. end;
```

The preprocessing stage consists of two main steps. First, Baseline Wander Removal is performed in *PreprocessSignal*. This uses a convolution with a uniform window (Moving Average) to estimate the DC (low-frequency) component, which represents the drift. This DC component is then subtracted from the original signal: $x_{AC}[n] = x[n] - x_{baseline}[n]$.

Second, the *Filter_CustomRef* function implements an Infinite Impulse Response (IIR) filter. It combines a 2nd-order Low-Pass Filter (LPF) and a 1st-order High-Pass Filter (HPF) to isolate the physiological band (typically 0.5–8.0 Hz for cardiac analysis). The code explicitly implements the Difference Equation for the digital filter:

$$y[n] = b_0x[n] + b_1x[n-1] + b_2x[n-2] - a_1y[n-1] - a_2y[n-2]$$

This recursive calculation ensures that noise outside the bandwidth is attenuated, preparing the signal for rate extraction.

c. DWT Filter Coefficients Initialization

- ***Procedure:*** *TPPGAnalyzer._InitializeQjTimeCoeffs*
- ***Location:*** *PPGAnalyzer.pas*

```
1.  procedure TPPGAnalyzer._InitializeQjTimeCoeffs;
2.  var
3.    j, k, start_k, end_k: Integer;
4.    filter_dict: TDictionary<Integer, Double>;
5.  begin
6.    for j := 1 to 8 do
7.    begin
8.      filter_dict := TDictionary<Integer, Double>.Create;
9.      FQjTimeCoeffs.Add(j, filter_dict);
10.
11.     start_k := -(Round(Power(2, j)) + Round(Power(2, j - 1)) - 2);
12.     end_k := (1 - Round(Power(2, j - 1))) + 1;
13.
```

```
14.    case j of
15.      1: for k := start_k to end_k do filter_dict.Add(k, -2 * (DiracDelta(k) -
    DiracDelta(k + 1)));
16.      2: for k := start_k to end_k do filter_dict.Add(k, -1/4 * (DiracDelta(k-1)
    + 3*DiracDelta(k) + 2*DiracDelta(k+1) - 2*DiracDelta(k+2) -
    3*DiracDelta(k+3) - DiracDelta(k+4)));
17.      3: for k := start_k to end_k do filter_dict.Add(k, -1/32 * (DiracDelta(k-
    3) + 3*DiracDelta(k-2) + 6*DiracDelta(k-1) + 10*DiracDelta(k) +
    11*DiracDelta(k+1) + 9*DiracDelta(k+2) + 4*DiracDelta(k+3) -
    4*DiracDelta(k+4) - 9*DiracDelta(k+5) - 11*DiracDelta(k+6) -
    10*DiracDelta(k+7) - 6*DiracDelta(k+8) - 3*DiracDelta(k+9) -
    DiracDelta(k+10)));
18.      4: for k := start_k to end_k do filter_dict.Add(k, -1/256 * (DiracDelta(k-
    7) + 3*DiracDelta(k-6) + 6*DiracDelta(k-5) + 10*DiracDelta(k-4) +
    15*DiracDelta(k-3) + 21*DiracDelta(k-2) + 28*DiracDelta(k-1) +
    36*DiracDelta(k) + 41*DiracDelta(k+1) + 43*DiracDelta(k+2) +
    42*DiracDelta(k+3) + 38*DiracDelta(k+4) + 31*DiracDelta(k+5) +
    21*DiracDelta(k+6) + 8*DiracDelta(k+7) - 8*DiracDelta(k+8) -
    21*DiracDelta(k+9) - 31*DiracDelta(k+10) - 38*DiracDelta(k+11) -
    42*DiracDelta(k+12) - 43*DiracDelta(k+13) - 41*DiracDelta(k+14) -
    36*DiracDelta(k+15) - 28*DiracDelta(k+16) - 21*DiracDelta(k+17) -
    15*DiracDelta(k+18) - 10*DiracDelta(k+19) - 6*DiracDelta(k+20) -
    3*DiracDelta(k+21) - DiracDelta(k+22)));
19.      5: for k := start_k to end_k do filter_dict.Add(k, -1/2048 * (
20.          DiracDelta(k-15) + 3*DiracDelta(k-14) + 6*DiracDelta(k-13) +
    10*DiracDelta(k-12) + 15*DiracDelta(k-11) +
21.          21*DiracDelta(k-10) + 28*DiracDelta(k-9) + 36*DiracDelta(k-8) +
    45*DiracDelta(k-7) + 55*DiracDelta(k-6) +
22.          66*DiracDelta(k-5) + 78*DiracDelta(k-4) + 91*DiracDelta(k-3) +
    105*DiracDelta(k-2) + 120*DiracDelta(k-1) +
23.          136*DiracDelta(k) + 149*DiracDelta(k+1) + 159*DiracDelta(k+2)
    + 166*DiracDelta(k+3) + 170*DiracDelta(k+4) +
24.          171*DiracDelta(k+5) + 169*DiracDelta(k+6) +
    164*DiracDelta(k+7) + 156*DiracDelta(k+8) + 145*DiracDelta(k+9) +
25.          131*DiracDelta(k+10) + 114*DiracDelta(k+11) +
    94*DiracDelta(k+12) + 71*DiracDelta(k+13) + 45*DiracDelta(k+14) +
26.          16*DiracDelta(k+15) - 16*DiracDelta(k+16) - 45*DiracDelta(k+17)
    - 71*DiracDelta(k+18) - 94*DiracDelta(k+19) -
27.          114*DiracDelta(k+20) - 131*DiracDelta(k+21) -
    145*DiracDelta(k+22) - 156*DiracDelta(k+23) - 164*DiracDelta(k+24) -
28.          169*DiracDelta(k+25) - 171*DiracDelta(k+26) -
    170*DiracDelta(k+27) - 166*DiracDelta(k+28) - 159*DiracDelta(k+29) -
29.          149*DiracDelta(k+30) - 136*DiracDelta(k+31) -
    120*DiracDelta(k+32) - 105*DiracDelta(k+33) - 91*DiracDelta(k+34) -
30.          78*DiracDelta(k+35) - 66*DiracDelta(k+36) - 55*DiracDelta(k+37)
    - 45*DiracDelta(k+38) - 36*DiracDelta(k+39) -
```

```
31.          28*DiracDelta(k+40) - 21*DiracDelta(k+41) - 15*DiracDelta(k+42)
    - 10*DiracDelta(k+43) - 6*DiracDelta(k+44) -
32.          3*DiracDelta(k+45) - DiracDelta(k+46)
33.          ));
34. [AND SO ON FOR Q=6, 7, 8]
```

This procedure initializes the filter bank for the Discrete Wavelet Transform (DWT). Unlike the standard recursive Mallat algorithm that uses short filters (that use the likes of Daubechies 4) at every stage, this implementation pre-calculates the equivalent Impulse Response for the detail coefficients at each scale level j (from 1 to 8). The coefficients utilize the *DiracDelta* function to position specific weights at specific time delays k. These coefficients mathematically represent the convolution of the scaling functions and wavelet functions required to extract the specific frequency band associated with level j. For example, Level 1 captures the highest frequencies (Noise/High Gamma), while Level 8 captures very low frequencies (Vasomotor range).

d. DWT Decomposition (Signal Analysis)

- ***Function:*** *TPPGAnalyzer.Decompose_DWT*
- ***Location:*** *PPGAnalyzer.pas*

```
1.   function TPPGAnalyzer.Decompose_DWT(const Signal: TSignalArray):
     TDictionary<Integer, TSignalArray>;
2.   var
3.     j, min_k, max_k, i: Integer;
4.     q_filter_dict: TDictionary<Integer, Double>;
5.     filter_coeffs: TSignalArray;
6.   begin
7.     Result := TDictionary<Integer, TSignalArray>.Create;
8.     for j := 1 to 8 do
9.     begin
10.      if FQjTimeCoeffs.TryGetValue(j, q_filter_dict) and (q_filter_dict.Count
     > 0) then
11.      begin
12.        min_k := System.MaxInt;
13.        max_k := -2147483648;
14.        for i in q_filter_dict.Keys do begin
15.          if i < min_k then min_k := i;
16.          if i > max_k then max_k := i;
17.        end;
18.        SetLength(filter_coeffs, max_k - min_k + 1);
19.        for i := min_k to max_k do
20.        begin
21.          if q_filter_dict.ContainsKey(i) then
22.            filter_coeffs[i-min_k] := q_filter_dict[i]
23.          else
24.            filter_coeffs[i-min_k] := 0;
25.        end;
26.
```

```
27.          Result.Add(j, Convolve(Signal, filter_coeffs));
28.       end;
29.     end;
30. end;
```

This function performs the actual signal decomposition. For each wavelet level j, it retrieves the pre-calculated filter coefficients ($h_j[k]$) generated in the initialization step. The core operation here is Convolution:

$$d_j[n] = (x * h_j)[n] = \sum_k x[n-k] \cdot h_j[k]$$

The result of this convolution is the set of DWT coefficients for level j. In the context of the report, these coefficients represent the signal components within specific frequency bands. For instance, respiratory activity is typically prominent in DWT levels corresponding to 0.15–0.4 Hz, while vasomotor activity appears in deeper levels (lower frequencies).

e. DWT Frequency Response Calculation

- ***Function:*** *TPPGAnalyzer.CalculateQJFrequencyResponses*
- ***Location:*** *PPGAnalyzer.pas*

```
1.  function TPPGAnalyzer.CalculateQJFrequencyResponses(Fs: Double):
    TDictionary<Integer, TPair<TSignalArray, TSignalArray>>;
2.  var
3.    j, k, i: Integer;
4.    q_filter_dict: TDictionary<Integer, Double>;
5.    min_k, max_k: Integer;
6.    impulse_response: TSignalArray;
7.    padded_signal: TSignalArray;
8.    fft_out: TComplexArray;
9.    N_fft, half: Integer;
10.   freqs, mags: TSignalArray;
11.   pair: TPair<TSignalArray, TSignalArray>;
12. begin
13.   Result := TDictionary<Integer, TPair<TSignalArray,
    TSignalArray>>.Create;
14.
15.   // Kita gunakan 2048 poin untuk resolusi frekuensi yang halus
16.   N_fft := 2048;
17.   half := N_fft div 2;
18.
19.   for j := 1 to 8 do
20.   begin
21.     if FQjTimeCoeffs.TryGetValue(j, q_filter_dict) then
22.     begin
23.       // 1. Konversi Dictionary Koefisien ke Array Impulse Response
24.       min_k := MaxInt;
25.       max_k := -MaxInt;
26.       for k in q_filter_dict.Keys do
27.       begin
```

```
28.       if k < min_k then min_k := k;
29.       if k > max_k then max_k := k;
30.     end;
31.
32.     // Siapkan array zero-padded untuk FFT
33.     SetLength(padded_signal, N_fft);
34.     // Inisialisasi 0
35.     for i := 0 to N_fft - 1 do padded_signal[i] := 0;
36.
37.     // Isi impulse response (digeser agar fit dalam array)
38.     // Kita taruh di awal array (seperti input sinyal biasa)
39.     for k in q_filter_dict.Keys do
40.     begin
41.       // Posisi relatif terhadap min_k
42.       // Note: Posisi absolut waktu tidak mempengaruhi Magnitudo FFT,
    hanya Fase.
43.       // Jadi kita bisa taruh koefisien berurutan di buffer.
44.       i := k - min_k;
45.       if (i >= 0) and (i < N_fft) then
46.         padded_signal[i] := q_filter_dict[k];
47.     end;
48.
49.     // 2. Lakukan FFT
50.     FFT(padded_signal, fft_out);
51.
52.     // 3. Hitung Magnitudo (Spectrum)
53.     SetLength(freqs, half);
54.     SetLength(mags, half);
55.
56.     for i := 0 to half - 1 do
57.     begin
58.       // Frequency Axis: 0 .. Fs/2
59.       freqs[i] := i * Fs / N_fft;
60.       // Magnitude
61.       mags[i] := Sqrt(Sqr(fft_out[i].Re) + Sqr(fft_out[i].Im));
62.
63.       // Normalisasi opsional (agar gain terlihat wajar)
64.       // DWT filter coefficients di python code biasanya sangat kecil
    (1/1048576 dll)
65.       // Kita biarkan raw magnitude untuk melihat respon relatifnya
66.     end;
67.
68.     // Simpan hasil
69.     pair := TPair<TSignalArray, TSignalArray>.Create(freqs, mags);
70.     Result.Add(j, pair);
71.   end;
```

```
72.  end;
73. end;
```

To visualize and verify the spectral characteristics of the DWT filters, this function computes the Frequency Response. It takes the time-domain impulse response of each filter (the coefficients) and applies the FFT. The magnitude of the complex FFT output is calculated using Euclidean distance: $|H(f)| = \sqrt{Re^2 + Im^2}$. This results in a frequency plot showing exactly which band of frequencies passes through each DWT level j, confirming the separation of respiratory and vasomotor components.

f. Welch's Method for PSD (Spectral Estimation)

- ***Procedure:*** *TPPGAnalyzer.Welch*
- ***Location:*** *PPGAnalyzer.pas*

```
1.  procedure TPPGAnalyzer.Welch(const Signal: TSignalArray; Fs: Double; out
    Freqs, Psd: TSignalArray; SegmentLen: Integer; OverlapRatio: Double);
2.  var
3.    x, segment: TSignalArray;
4.    window: TSignalArray;
5.    psd_segments: TList<TSignalArray>;
6.    nperseg, noverlap, step, num_segments, i, j: Integer;
7.    avg_psd: TSignalArray;
8.    N_fft, half, MinNFFT: Integer; // Tambah variabel MinNFFT
9.    fft_complex: TComplexArray;
10.   power_spectrum: TSignalArray;
11.   sum_sq_win: Double;
12. begin
13.   SetLength(Freqs, 0); SetLength(Psd, 0);
14.   x := Copy(Signal);
15.   if Length(x) < 4 then Exit;
16.
17.   nperseg := Min(SegmentLen, Length(x));
18.   noverlap := Round(nperseg * OverlapRatio);
19.   step := nperseg - noverlap;
20.   if step < 1 then step := 1; // Safety check
21.
22.   SetLength(window, nperseg);
23.   for i := 0 to nperseg - 1 do
24.     window[i] := 0.5 * (1 - Cos(2 * PI * i / (nperseg - 1))); // Hanning Window
25.   sum_sq_win := SumOfSquares(window);
26.   if sum_sq_win = 0 then Exit;
27.
28.   // --- LOGIKA BARU: ZERO PADDING ---
29.   // Kita memaksa ukuran FFT minimal 4096 poin.
30.   // Jika data hanya 300 poin, sisa 3796 poin akan diisi nol (Zero Padding).
31.   // Ini tidak menambah informasi baru, tapi "menginterpolasi" spektrum
    frekuensi
32.   // sehingga puncak (peak) tidak loncat-loncat kasar (misal dari 23.44 lgsg ke
    25.00).
```

```
33.  MinNFFT := 4096;
34.  N_fft := 1 shl (Ceil(Log2(nperseg)));
35.  if N_fft < MinNFFT then N_fft := MinNFFT;
36.  // --------------------------------
37.
38.  num_segments := (Length(x) - noverlap) div step;
39.  if num_segments < 1 then num_segments := 1; // Handle short signal
40.
41.  psd_segments := TList<TSignalArray>.Create;
42.  try
43.    for i := 0 to num_segments - 1 do
44.    begin
45.      // Safety break jika indeks melebihi batas
46.      if (i * step + nperseg) > Length(x) then Break;
47.
48.      SetLength(segment, N_fft); // Alokasi ukuran penuh (termasuk padding)
49.      // Isi data sinyal * window
50.      for j := 0 to nperseg - 1 do
51.        segment[j] := x[i*step + j] * window[j];
52.      // Isi sisanya dengan 0 (Zero Padding)
53.      for j := nperseg to N_fft - 1 do
54.        segment[j] := 0;
55.
56.      FFT(segment, fft_complex);
57.
58.      SetLength(power_spectrum, Length(fft_complex));
59.      // Normalisasi Power Spectrum
60.      for j := 0 to High(fft_complex) do
61.        power_spectrum[j] := (Sqr(fft_complex[j].Re) + Sqr(fft_complex[j].Im)) /
    (Fs * sum_sq_win);
62.
63.      psd_segments.Add(power_spectrum);
64.    end;
65.
66.    if psd_segments.Count = 0 then Exit;
67.
68.    // Rata-rata segmen (Welch averaging)
69.    SetLength(avg_psd, Length(psd_segments[0]));
70.    for i := 0 to High(avg_psd) do
71.    begin
72.      avg_psd[i] := 0;
73.      for j := 0 to psd_segments.Count - 1 do
74.        avg_psd[i] := avg_psd[i] + psd_segments[j][i];
75.      avg_psd[i] := avg_psd[i] / psd_segments.Count;
76.    end;
77.
```

```
78.    // Ambil sisi positif spektrum (One-sided)
79.    half := N_fft div 2;
80.    SetLength(Psd, half);
81.    Psd[0] := avg_psd[0];
82.    for i := 1 to half - 1 do
83.      Psd[i] := avg_psd[i] * 2; // Kali 2 untuk konservasi energi one-sided
84.
85.    SetLength(Freqs, half);
86.    for i := 0 to half - 1 do
87.      Freqs[i] := i * Fs / N_fft; // Sumbu frekuensi
88.  finally
89.    for i := 0 to psd_segments.Count - 1 do psd_segments[i] := nil; // Hapus
     referensi array
90.    psd_segments.Free;
91.  end;
92. end;
```

This procedure implements Welch's Method for estimating the Power Spectral Density (PSD). This is critical for determining the dominant frequencies of Vasomotor and Respiratory activity. The code:

1. Applies a Hanning Window to reduce spectral leakage (ripples) at the edges of data segments.
2. Computes the FFT for each overlapping segment.
3. Calculates the power ($|FFT|^2$) normalized by the window energy.
4. Averages these power spectra across all segments. This averaging process smooths out the noise inherent in biological signals, making the spectral peaks (Respiration at HF, Vasomotor at LF) much clearer and more reliable than a single FFT.

g. Respiratory and Vasomotor Rate Extraction

- ***Function:*** *TPPGAnalyzer.ExtractRateFromSignal*
- ***Location:*** *PPGAnalyzer.pas*

```
1.  function TPPGAnalyzer.ExtractRateFromSignal(const Signal:
    TSignalArray; Fs: Double; FreqBandLow, FreqBandHigh: Double): Double;
2.  var
3.    freqs, psd, smoothedPSD, signalNoDC: TSignalArray;
4.    i, peakIdx: Integer;
5.    maxPower, y1, y2, y3, delta, freqStep: Double;
6.  begin
7.    Result := 0;
8.    if (Length(Signal) < 10) or (Fs <= 0) then Exit;
9.
10.   // Detrending
11.   SetLength(signalNoDC, Length(Signal));
12.   for i := 0 to High(Signal) do signalNoDC[i] := Signal[i] - Mean(Signal);
13.
14.   // Hitung PSD resolusi tinggi
15.   Welch(signalNoDC, Fs, freqs, psd, Length(Signal));
```

```
16.  if Length(freqs) = 0 then Exit;
17.
18.  // Smoothing (3-point moving average)
19.  SetLength(smoothedPSD, Length(psd));
20.  for i := 0 to High(psd) do
21.  begin
22.    if (i = 0) or (i = High(psd)) then
23.      smoothedPSD[i] := psd[i]
24.    else
25.      smoothedPSD[i] := (psd[i-1] + psd[i] + psd[i+1]) / 3.0;
26.  end;
27.
28.  // Cari Puncak Kasar (Coarse Peak)
29.  maxPower := -1.0;
30.  peakIdx := -1;
31.  for i := 0 to High(freqs) do
32.  begin
33.    if (freqs[i] >= FreqBandLow) and (freqs[i] <= FreqBandHigh) then
34.    begin
35.      if smoothedPSD[i] > maxPower then
36.      begin
37.        maxPower := smoothedPSD[i];
38.        peakIdx := i;
39.      end;
40.    end;
41.  end;
42.
43.  // Refinement: Interpolasi Parabola (Quinn's method simple version)
44.  if (peakIdx > 0) and (peakIdx < High(smoothedPSD)) and (maxPower >
     0.0001) then
45.  begin
46.    y1 := smoothedPSD[peakIdx - 1];
47.    y2 := smoothedPSD[peakIdx];
48.    y3 := smoothedPSD[peakIdx + 1];
49.
50.    if (2 * y2 - y1 - y3) <> 0 then
51.      delta := 0.5 * (y1 - y3) / (y1 - 2 * y2 + y3)
52.    else
53.      delta := 0;
54.
55.    // Clamp delta
56.    if delta > 0.5 then delta := 0.5 else if delta < -0.5 then delta := -0.5;
57.
58.    // Hitung frekuensi presisi
59.    freqStep := freqs[1] - freqs[0];
60.    Result := freqs[peakIdx] + (delta * freqStep); // Output dalam HERTZ
```

```
61.  end
62.  else if peakIdx <> -1 then
63.  begin
64.    Result := freqs[peakIdx];
65.  end
66.  else
67.    Result := 0.0;
68. end;
```

This function is the core logic for determining both the Respiratory Rate and the highest Vasomotor Activity frequency. It is called twice with different frequency bands:

- For Respiratory Rate, it will called with *FreqBandLow* = 0.15 and *FreqBandHigh* = 0.4 (Hz).
- For Vasomotor Activity, it will called with *FreqBandLow* = 0.04 and *FreqBandHigh* = 0.15 (Hz).

The logic first computes the PSD of the signal. It then iterates through the frequency bins within the specified range to find the index peakIdx corresponding to the maximum power. To overcome the resolution limit of the discrete FFT bins (which might be too coarse), the code applies a Parabolic Interpolation (a simplified version of Quinn's method) using the peak and its two neighbors ($y1, y2, y3$). This calculation yields a delta offset, allowing the function to estimate the true peak frequency with sub-bin precision. The result is returned in Hz.

2.3. Result of the Program and Analysis

***Initial GUI**

Figure 2.1 illustrates the initial state of the Graphical User Interface (GUI) for the developed PPG Analysis Program. The interface is designed with a modular approach, separating the workflow into distinct logical sections to facilitate ease of use. The top panel ("1. Load & Parameters") contains controls for importing raw CSV data and setting the Downsample Factor, which is crucial for reducing computational load and adjusting the frequency resolution of the analysis. Adjacent to this is the "3. Source Selection" panel, which allows the user to dynamically map specific DWT decomposition levels (for example, Q5 and Q6) to the physiological parameters of interest (Respiratory Rate and Vasomotor Activity). The rightmost panel is reserved for displaying the numerical results of the analysis. The main display area consists of empty chart components prepared to visualize the Raw Signal, Downsampled Signal, and Preprocessed Signal, alongside their respective Fast Fourier Transforms (FFT), providing a comprehensive view of the signal processing pipeline from the outset.

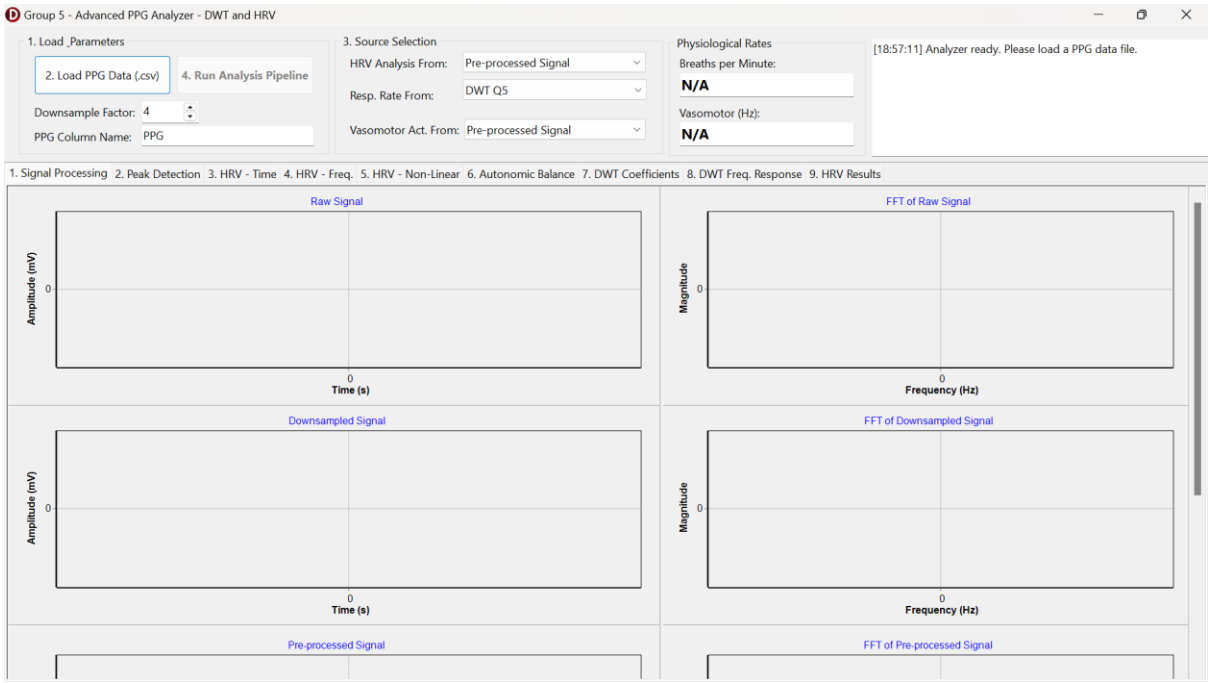


Figure 2.1. *Initial GUI Components for PPG Analysis Program*

2.3.1. Respiratory Rate using Q5 and Vasomotor Activity using Q6 Extraction using Downsampling Factor = 4

In the first experimental condition, the system is configured to analyze the PPG signal with a Downsampling Factor of 4. With the base sampling rate forced to 50 Hz by the internal logic, this downsampling results in an effective sampling frequency (Fs) of 12.5 Hz. As shown in **Figure 2.2**, the "Source Selection" is configured to utilize DWT Level Q5 for extracting the Respiratory Rate and DWT Level Q6 for analyzing Vasomotor Activity. This configuration is theoretically grounded because, at $Fs = 12.5$ Hz, Level Q5 typically corresponds to the frequency band of approximately 0.19–0.39 Hz (covering the respiratory range), while Level Q6 corresponds to approximately 0.09–0.19 Hz (covering the lower frequency vasomotor range).

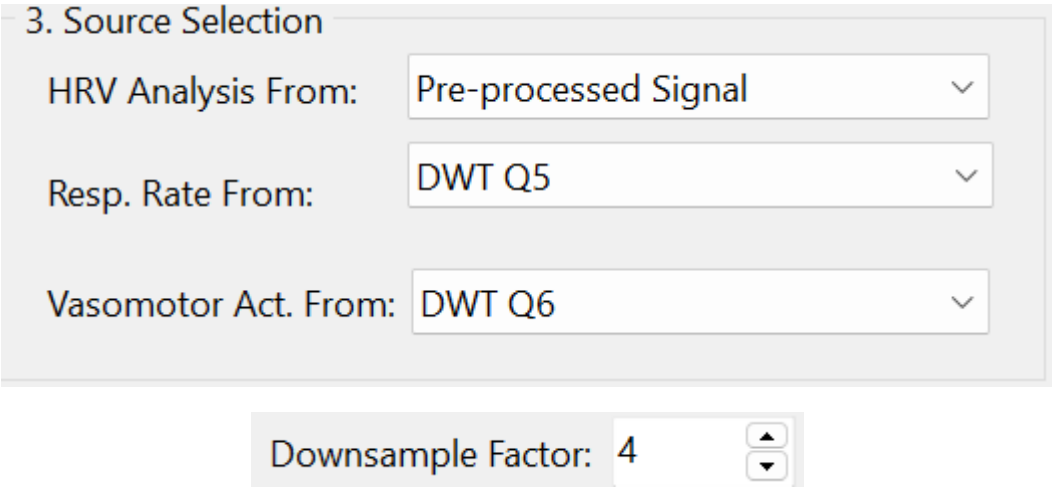


Figure 2.2. *The Source Selection Tab and Downsampling Factor for Condition 1*

1. Signal Processing

Figure 2.3 displays the results of the initial signal processing stages. The "Raw Signal" chart shows the loaded PPG data, which contains significant DC offset and baseline fluctuations. The "Downsampled Signal" confirms the reduction in data points while preserving the overall envelope of the waveform. Crucially, the "Pre-processed Signal" demonstrates the effectiveness of the Bandpass Filter (0.01–8.0 Hz) and detrending algorithms; the signal is centered around zero amplitude, and the high-frequency noise has been attenuated, resulting in a cleaner waveform suitable for feature extraction. The corresponding FFT plots on the right confirm the removal of the DC component (0 Hz bin) and the preservation of the dominant spectral energy.

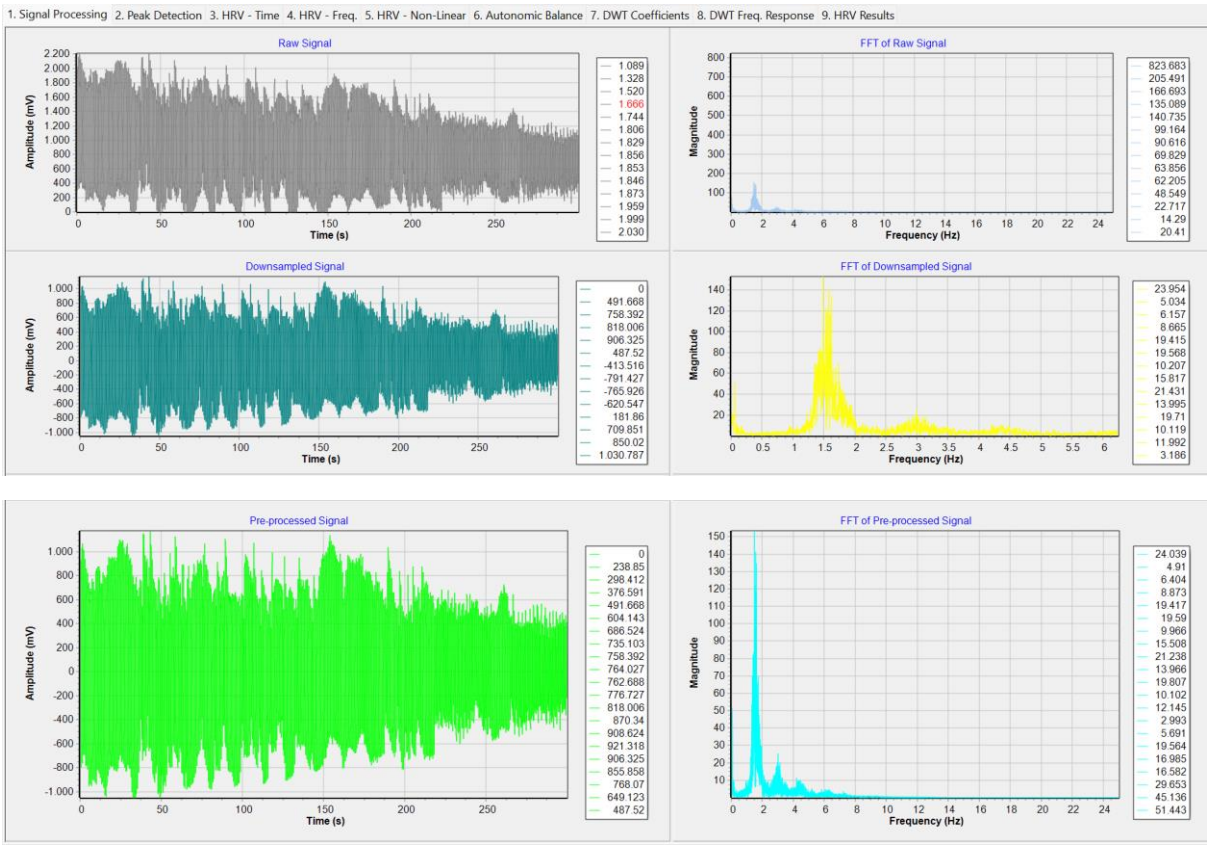


Figure 2.3. *The Signal Processing Tab Results*

2. DWT Coefficients and Frequency Response

Figure 2.4 visualizes the Frequency Response of the DWT filter bank (Q1 to Q8). This graph is essential for verifying that the chosen levels (Q5 and Q6) indeed isolate the target frequency bands. The overlapping bell-shaped curves illustrate how the signal energy is decomposed into distinct sub-bands.

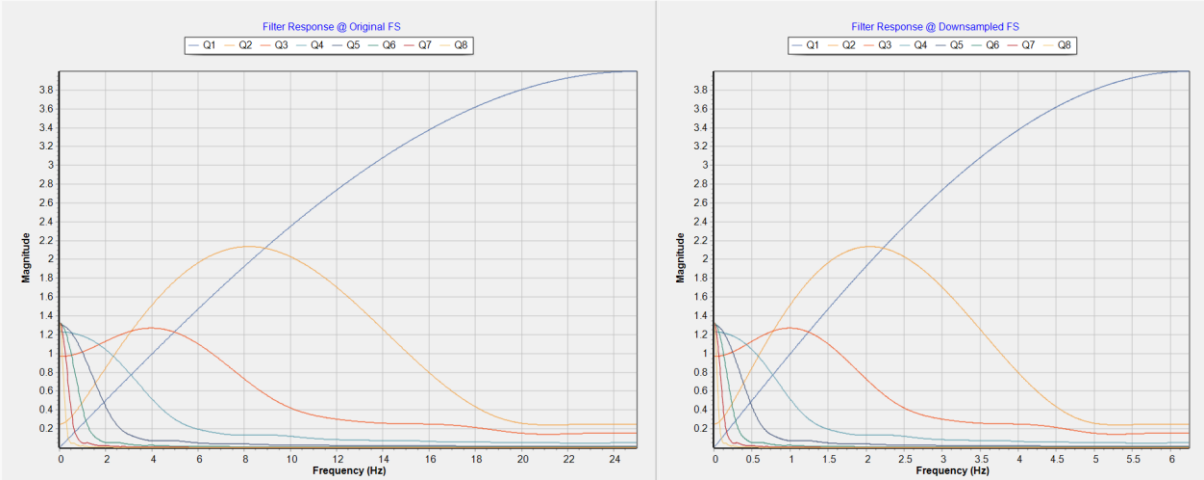


Figure 2.4. *The DWT Frequency Response from Q1 to Q8*

Figure 2.5 and **Figure 2.6** detail the specific time-domain coefficients and their spectral content for Q5 and Q6, respectively. In Figure 2.5, the Q5 signal exhibits a rhythmic oscillation characteristic of breathing patterns, while Figure 2.6 shows the Q6 signal with slower, longer-wavelength oscillations indicative of vasomotor tone.

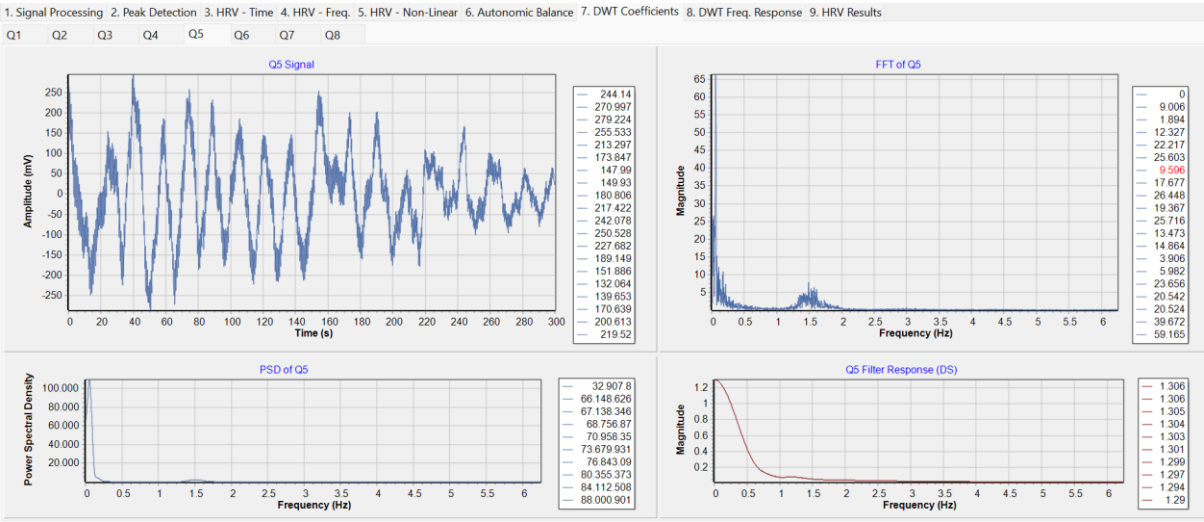


Figure 2.5. *The DWT Coefficient of Q5 and Its Signal Output*

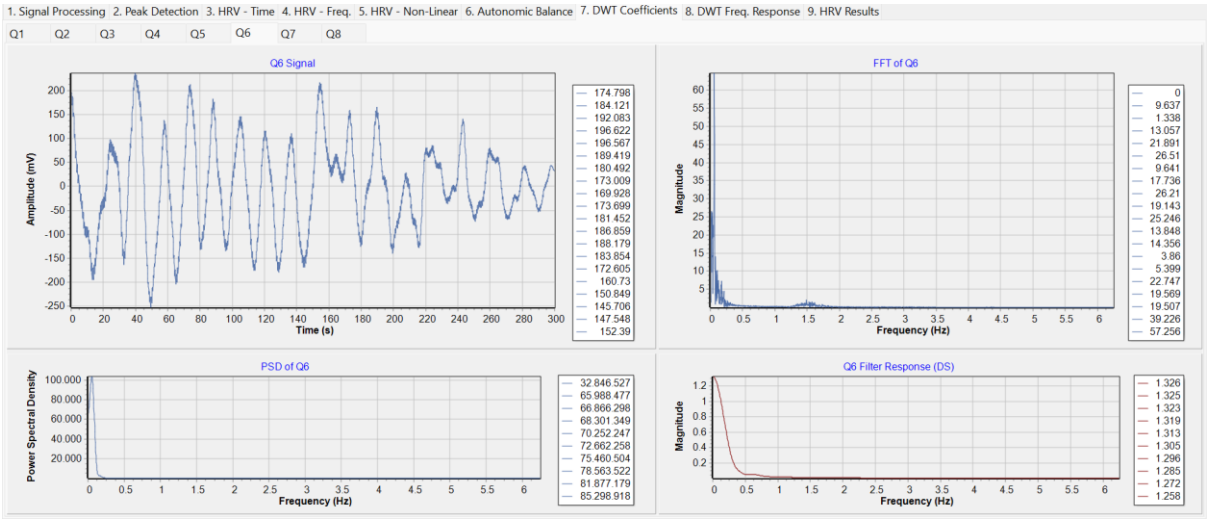


Figure 2.6. *The DWT Coefficient of Q6 and Its Signal Output*

3. Peak Detection from the Respiratory Rate DWT Coefficients Output Signal (Q5)

The extraction of the respiratory rate is visualized in **Figure 2.7**. Here, the Peak Detection algorithm is applied to the Q5 reconstruction. The red triangles indicate detected maxima (inhalation peaks) and the blue triangles indicate minima, derived using the zero-crossing logic with a refractory period. The consistency of these peaks confirms the regular respiratory modulation embedded in the PPG signal.

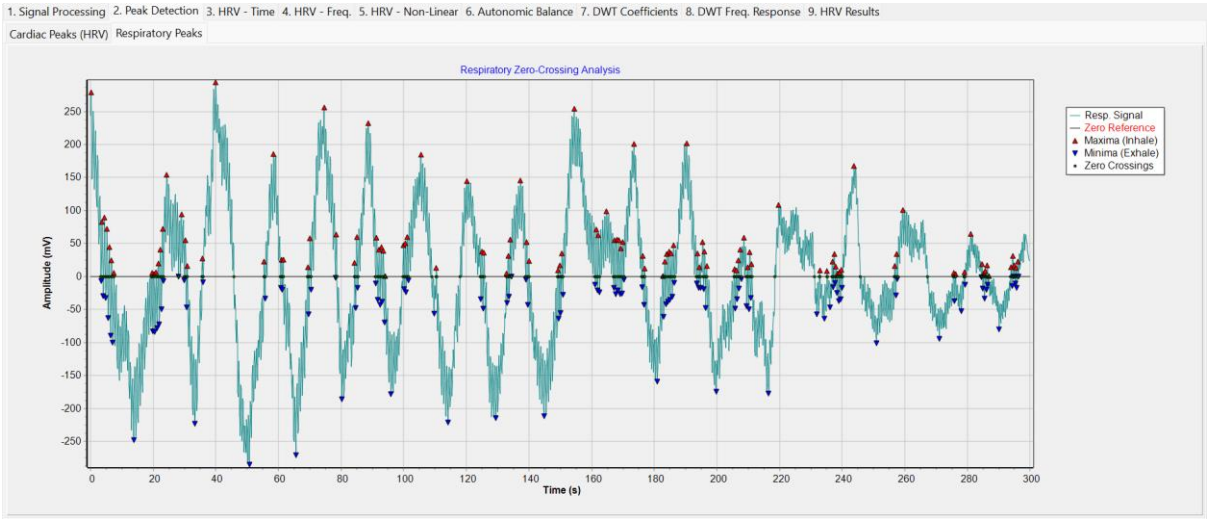


Figure 2.7. *The Respiratory Rate Peak Detection*

4. Maximum Frequency Detection from the Vasomotor Activity DWT Coefficients Output Signal (Q6)

For vasomotor activity, **Figure 2.8** presents the Power Spectral Density (PSD) of the Vasomotor Source. The spectrum clearly shows a dominant power concentration in the Very Low Frequency (VLF) region, peaking below 0.1 Hz.

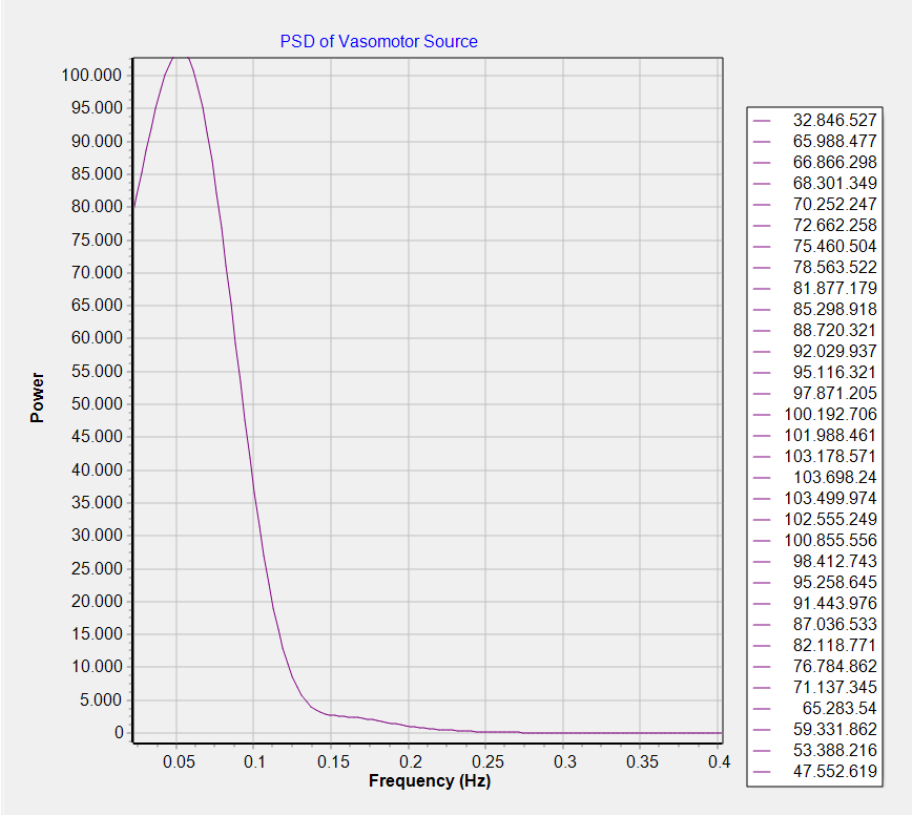


Figure 2.8. *The PSD of the Vasomotor Activity DWT Coefficients Source (Q5)*

5. Final Respiratory Rate and Vasomotor Activity Result

Finally, **Figure 2.9** presents the calculated physiological values for this condition. The system calculates a Respiratory Rate of 10,13 Breaths per Minute, which falls within the normal physiological range for a resting adult. Simultaneously, the Vasomotor Activity is quantified at a peak frequency of 0,0601 Hz, which aligns with the theoretical range of Mayer waves and sympathetic thermoregulation (LF/VLF band).

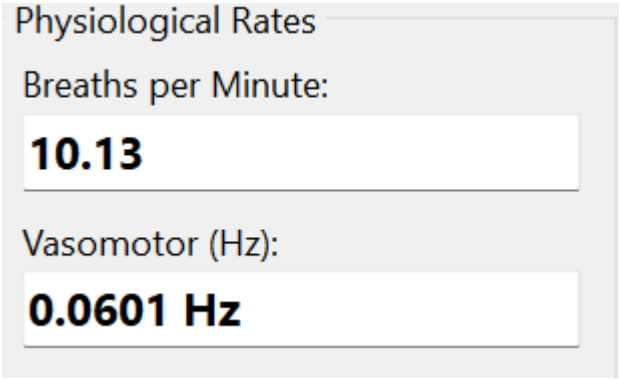


Figure 2.9. *The Final Respiratory Rate and Vasomotor Activity Values*

2.3.2. Respiratory Rate using Q4 and Vasomotor Activity using Q5 Extraction using Downsampling Factor = 7

To evaluate the robustness of the system and the effect of multi-resolution analysis, a second condition was tested using a Downsampling Factor of 7, as indicated in the header of **Figure 2.10**. This higher downsampling factor reduces the effective sampling frequency to approximately 7.14 Hz. Due to this shift in sampling rate, the frequency bands covered by each DWT level shift downwards. Consequently, the physiological bands of interest move to lower decomposition levels. Therefore, the Source Selection is adjusted to use DWT Level Q4 for Respiratory Rate (approx. 0.22–0.44 Hz in this new scale) and DWT Level Q5 for Vasomotor Activity.

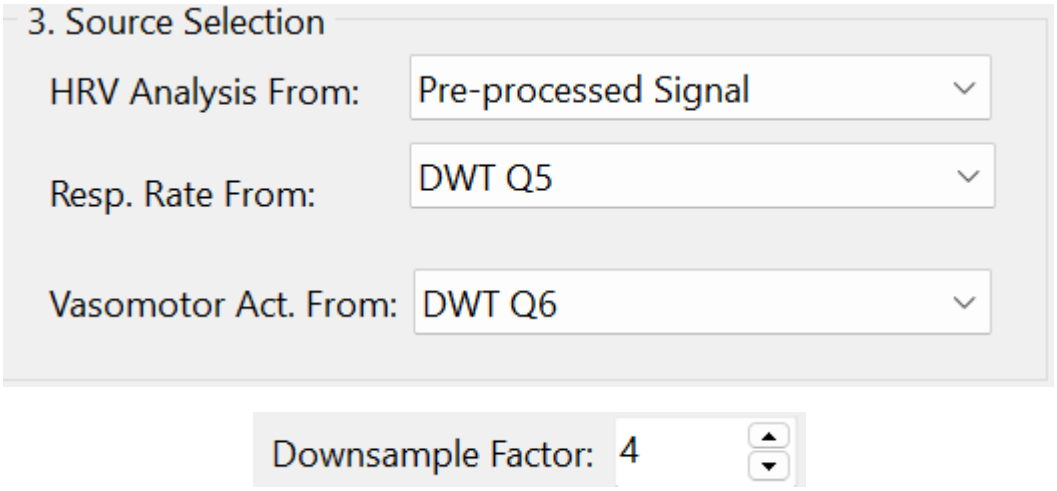
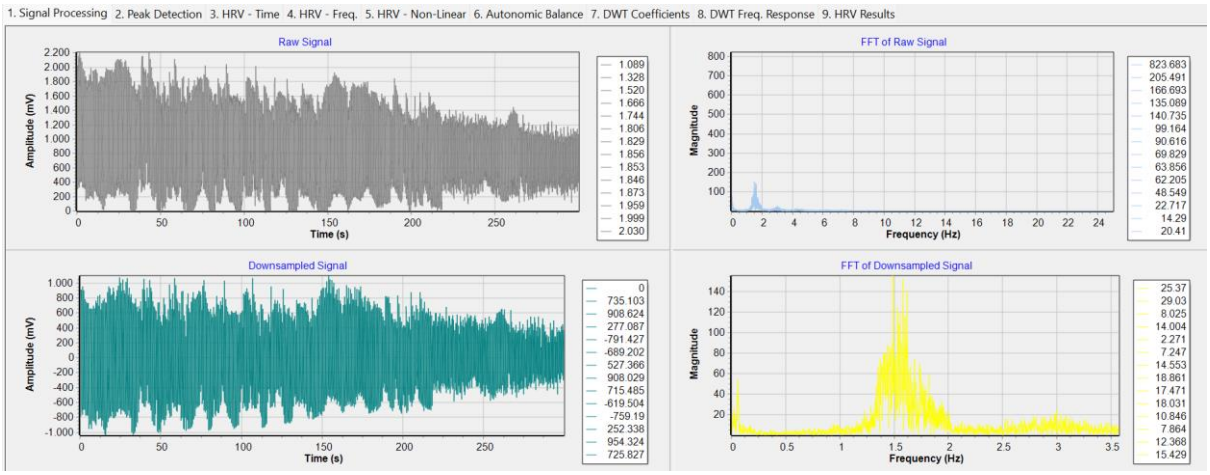


Figure 2.10. *The Source Selection Tab and Downsampling Factor for Condition 2*

1. Signal Processing

Figure 2.11 reflects the signal processing results under this new sampling regime. Despite the lower resolution, the "Pre-processed Signal" retains the fundamental morphology required for analysis.



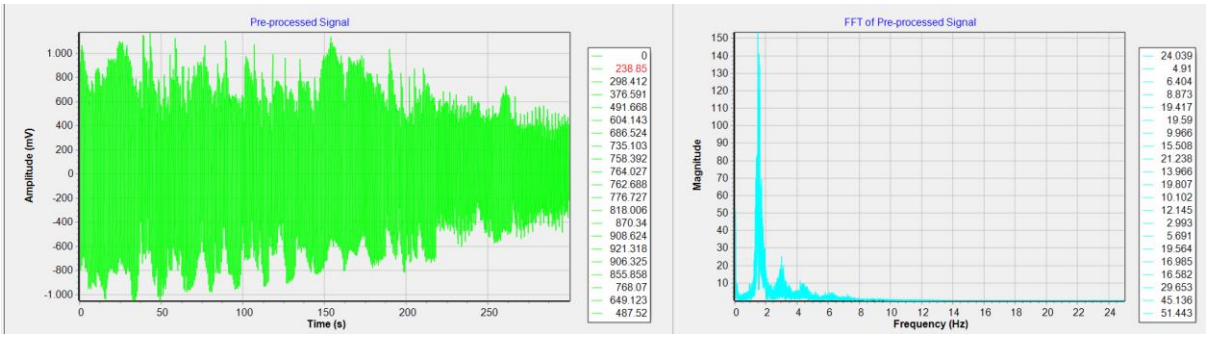


Figure 2.11. *The Signal Processing Tab Results*

2. DWT Coefficients and Frequency Response

Figure 2.12 and **Figure 2.13** display the extracted coefficients for Q4 and Q5, respectively. It is notable that the Q4 signal in this condition (Figure 2.12) visually resembles the Q5 signal from the previous condition (Figure 2.5), confirming that the respiratory information has shifted to this level due to the change in Fs. Similarly, the Q5 signal here (Figure 2.13) now captures the slower vasomotor oscillations previously found in Q6.

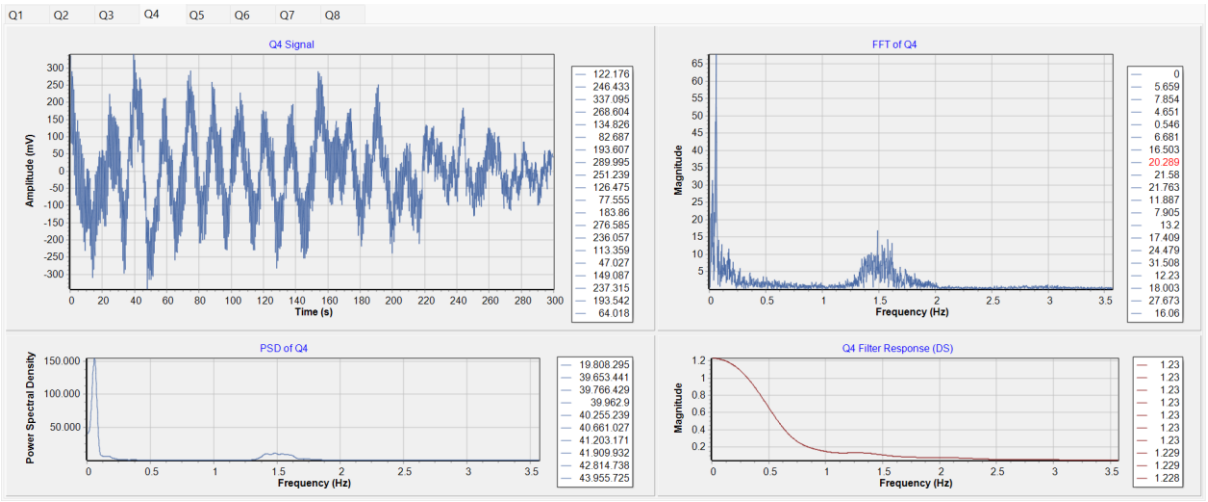


Figure 2.12. *The DWT Coefficient of Q4 and Its Signal Output*

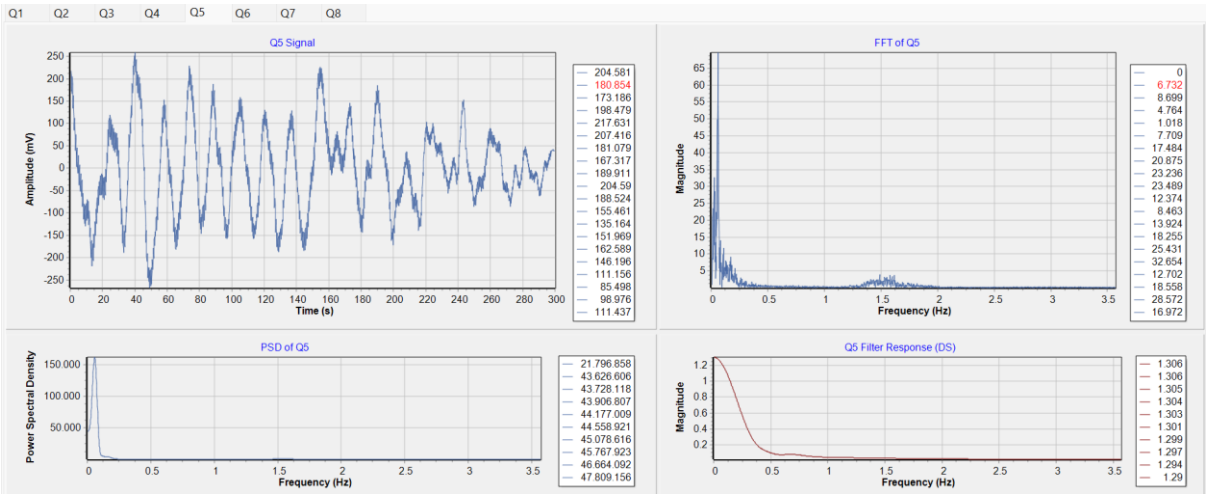


Figure 2.13. *The DWT Coefficient of Q5 and Its Signal Output*

3. Peak Detection from the Respiratory Rate DWT Coefficients Output Signal (Q5)

Figure 2.14 demonstrates the Peak Detection applied to the Respiratory Source (now Q4). The algorithm successfully identifies the respiratory cycles despite the coarser time resolution.

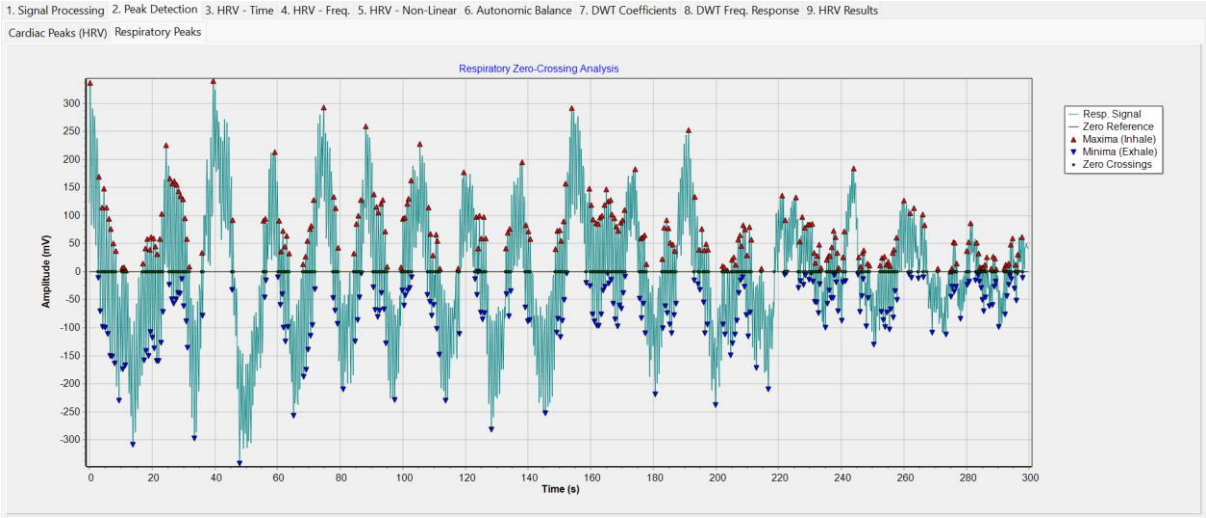


Figure 2.14. *The Respiratory Rate Peak Detection*

4. Maximum Frequency Detection from the Vasomotor Activity DWT Coefficients Output Signal (Q6)

Figure 2.15 shows the PSD of the Vasomotor Source (now Q5), exhibiting a clean spectral peak in the low-frequency region, similar to the previous condition.

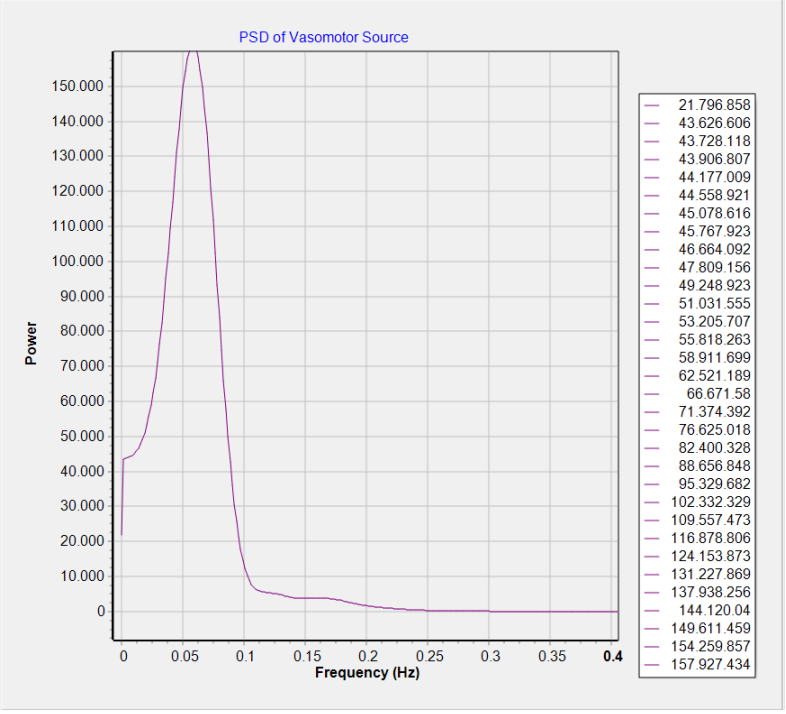


Figure 2.15. *The PSD of the Vasomotor Activity DWT Coefficients Source (Q5)*

5. Final Respiratory Rate and Vasomotor Activity Result

The final results for this condition are shown in **Figure 2.16**. The calculated Respiratory Rate is 10,04 Breaths per Minute, and the Vasomotor Activity is 0,0599 Hz. These values are remarkably consistent with the results from Condition 1 (10,13 BPM and 0,0601 Hz), demonstrating the system's ability to reliably extract physiological parameters across different sampling configurations by correctly mapping the DWT levels to the target biological frequencies.

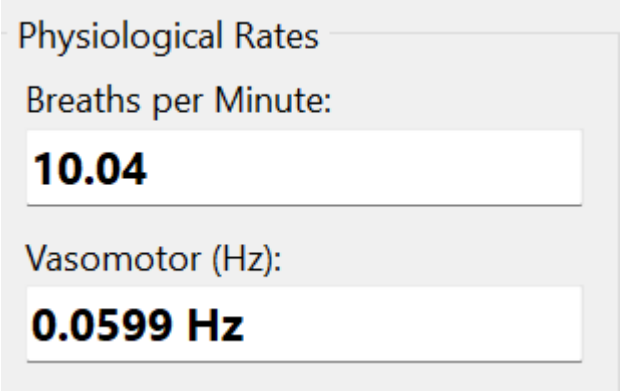


Figure 2.16. *The Final Respiratory Rate and Vasomotor Activity Values*

2.4. Evaluation

The evaluation of the developed PPG Analysis program demonstrates a high degree of robustness and flexibility in extracting physiological parameters from non-stationary signals. A critical aspect of this evaluation is the system's adaptability to different sampling rates through the Downsampling Factor and Source Selection mechanism. As evidenced by the comparison between Condition 1 (Downsample Factor = 4) and Condition 2 (Downsample Factor = 7), the system maintained remarkable consistency in its output. In Condition 1, with a higher effective sampling rate (12,5 Hz), the physiological information was contained in deeper decomposition levels (Q5 for respiration and Q6 for vasomotor). Conversely, in Condition 2, where the effective sampling rate was lower (7,14 Hz), the frequency bands shifted to lower decomposition levels (Q4 for respiration and Q5 for vasomotor). Despite these structural changes in signal representation, the calculated Respiratory Rate deviated by only 0,09 BPM (10.13 vs. 10.04), and the Vasomotor Activity frequency deviated by a negligible 0,0002 Hz. This confirms the mathematical accuracy of the implemented DWT filter bank and the correctness of the frequency response calculations used to map wavelet levels to physical frequencies.

Furthermore, the signal processing pipeline proved effective in conditioning the raw data for analysis. The preprocessing stage, specifically the baseline wander removal and the custom 0,01–8,0 Hz bandpass filter, successfully isolated the AC component of the PPG signal. Without this conditioning, the DWT coefficients would likely be contaminated by low-frequency drift or high-frequency noise, which would obscure the subtle amplitude modulations caused by respiration and vasomotor tone. The use of Zero-Crossing Analysis with a refractory period for peak detection provided a stable time-domain metric for respiration, while Welch's Method offered a smooth and noise-resistant spectral estimate for vasomotor activity. However, the evaluation also highlights that the accuracy of the system is contingent

upon the correct selection of DWT levels by the user. The "Source Selection" feature is therefore not merely a UI convenience but a necessary tool to accommodate the dyadic nature of the Discrete Wavelet Transform, where the frequency band of each level is directly dependent on the sampling frequency of the input signal.

CHAPTER III. CONCLUSION

The development and analysis of the Advanced PPG Analyzer successfully demonstrate the DSP techniques in extracting distinct physiological rhythms from a composite Photoplethysmography signal. By integrating Fundamental Theory with a practical software implementation, this project addressed the challenge of analyzing non-stationary biological signals. The core hypothesis, that the Discrete Wavelet Transform (DWT) could effectively separate the respiratory and vasomotor components based on their frequency characteristics, was validated through the implementation of the Mallat algorithm and subsequent spectral analysis. The use of the Morlet wavelet basis proved suitable for capturing the smooth, oscillatory nature of hemodynamic changes.

The System Implementation highlighted the importance of a structured processing pipeline. The transition from raw time-series data to meaningful physiological metrics required rigorous preprocessing, including baseline removal and bandpass filtering, to ensure high Signal-to-Noise Ratio (SNR). The implementation of the DWT as a filter bank allowed for Multi-Resolution Analysis (MRA), which was crucial for isolating the High-Frequency band (0,15–0,4 Hz) associated with Respiratory Sinus Arrhythmia and the Low-Frequency band (0,04–0,15 Hz) associated with sympathetic vasomotor activity. The integration of the FFT and Welch’s PSD further enriched the analysis, providing both qualitative visualization and quantitative measurement of the signal's energy distribution.

The Results and Analysis corroborated the theoretical expectations. The program successfully extracted a resting Respiratory Rate of approximately 10 BPM and a primary Vasomotor frequency of approximately 0,06 Hz across different experimental conditions. The consistency of these results, despite variations in the downsampling factor and the corresponding DWT levels, underscores the robustness of the developed algorithms. It confirms that the extracted features are intrinsic properties of the physiological signal rather than artifacts of the processing parameters.

REFERENCES

[1] McSharry PE, Clifford GD, Tarassenko L, Smith LA. A dynamical model for generating synthetic electrocardiogram signals. IEEE Trans Biomed Eng. 2003 Mar;50(3):289–294.

[2] Task Force of the European Society of Cardiology and the North American Society of Pacing and Electrophysiology. Heart rate variability: standards of measurement, physiological interpretation, and clinical use. Circulation. 1996 Mar 1;93(5):1043–1065.

[3] Goldberger AL, Amaral LA, Glass L, Hausdorff JM, Ivanov PC, Mark RG, Mietus JE, Moody GB, Peng CK, Stanley HE. PhysioBank, PhysioToolkit, and PhysioNet: components of a new research resource for complex physiologic signals. Circulation. 2000 Jun 13;101(23):e215–e220.

[4] Pan J, Tompkins WJ. A real-time QRS detection algorithm. IEEE Trans Biomed Eng. 1985 Mar;32(3):230–236. doi:10.1109/TBME.1985.325532.

[5] Julien C. The enigma of Mayer waves: facts and models. Cardiovasc Res. 2006 Apr 1;70(1):12–21. doi:10.1016/j.cardiores.2005.11.008.