# A Project Report on

# "Heart Rate Detection from ECG Signal using MATLAB"

**Submitted in partial fulfillment of the requirements of the degree of**

**Bachelor of Technology**

**Submitted by**


B.Sarath -(O180081)

B.Renuka Devi -(O180873)

T.Krishna Veni -(O180366)

P.Naveen -(O180892)


**Under the Supervision of**

Mr.G.Bala Nagireddy

**Department of ECE**

ELECTRONICS AND COMMUNICATION ENGINEERING

RAJIV GANDHI UNIVERSITY OF KNOWLEDGE TECHNOLOGIES

ONGOLE CAMPUS

2022-2023

# Approval Sheet

This report entitled “Heart Rate Detection from ECG Signal using MATLAB” by B. Sarath -(O180081), B.Renuka Devi -(O180873), T.Krishna Veni -(O180366), P. Naveen -(O180892) is approved for the degree of Bachelor of Technology Electronics and Communication Engineering.

**Examiner**(s)

**Supervisor**(s)

**Chairman**

**Date:** _____

**Place:** _____

## Candidate's Declaration

I declare that this written submission represents my ideas in my own words and where others' ideas or words have been included. I have adequately cited and referenced the original sources. I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in my submission. I understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

**Signature**

| | | |
|---|---|---|
| **B. Sarath** | **-(O180081)** | **…….** |
| **B. Renuka Devi** | **-(O180873)** | **…….** |
| **T. Krishna Veni** | **-(O180366)** | **…….** |
| **P. Naveen** | **-(O180892)** | **…….** |

**Date :** _____

## DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

## RAJIV GANDHI UNIVERSITY OF KNOWLEDGE TECHNOLOGIES ONGOLE CAMPUS

# Certificate

This is to certify that the report entitled "Heart Rate Detection from ECG Signal using MATLAB" submitted by B.Sarath - (O180081), B.Renuka Devi- (O180873) T.Krishna Veni- (O180366) and P.Naveen- (O180892) in partial fulfilment of the requirements for the award of Bachelor of Technology in Electronics and Communication Engineering is a bonafide work carried by them under my supervision and guidance.

**Project Internal Guide**
Mr.G.Bala Nagireddy
Assistant Professor
Depatment of ECE

**Head of the Department**
Mr.G.Bala Nagireddy
Assistant Professor
Department of ECE

## Acknowledgement

I would like to express my sincere gratitude to **Mr.G.Bala Nagireddy**, my project guide for valuable suggestions and keen interest throughout the progress of my course and research.

I am grateful to **Mr.G.Bala Nagireddy**, HOD of **Electronics & Communication Engineering**, for providing excellent computing facilities and a congenial atmosphere for progressing with my project.

I would like to thank **Prof.B.Jayarami Reddy Garu**, director of RAJIV GANDHI UNIVERSITY OF KNOWLEDGE TECHNOLOGIES, ONGOLE for providing all the necessary resources for the successful completion of my course work. At last, but not the least I thank my teammates and other students for their physical and moral support.

**With Sincere Regards,**

B.Sarath -(O180081)

B.Renuka Devi -(O180873)

T.Krishna Veni -(O180366)

P.Naveen -(O180892**)**

**Date:** _____

# Abstract

Heart rate detection is a fundamental task in analysing electrocardiogram (ECG) signals, as it provides important insights into cardiac health. This project focuses on developing an algorithm to detect the heart rate from an ECG signal and applies signal processing techniques to extract relevant information. The algorithm incorporates peak detection and autocorrelation methods to identify R-peaks, which represent individual heartbeats. MATLAB is used as the programming platform for implementing the algorithm.

The project begins by loading the ECG signal data and applying a bandpass filter to remove noise and unwanted frequency components. Next, the algorithm detects R-peaks using peak detection techniques such as finding local maxima or analysing the autocorrelation function. The RR intervals between successive R-peaks are calculated to determine the heart rate.

To assess the severity of a heart condition, additional analysis can be performed on the ECG signal. This includes evaluating the regularity of rhythm, analysing P-wave and PR interval, evaluating QRS complex morphology, studying ST segment and T-wave abnormalities, and evaluating the QT interval. These features provide insights into potential arrhythmias, conduction abnormalities, ischemia, and other cardiac conditions.

The severity assessment and interpretation of the ECG signal are crucial steps in determining the overall health of the heart. The project code provides a framework for processing and analysing ECG signals, but it is important to consult with medical professionals, such as cardiologists, for accurate diagnosis and severity assessment. This project serves as a starting point for heart rate detection and severity assessment, highlighting the importance of signal processing techniques in cardiac health analysis.

# TABLE OF CONTENTS

# CHAPTER-1
# INTRODUCTION

# 1.INTRODUCTION

## 1.1 Introduction

An electrocardiogram (ECG) is a diagnostic tool used to measure and record the electrical activity of the heart. It provides valuable information about the heart's rhythm, rate, and overall cardiac health. One of the key components of an ECG waveform is the R-peak, which represents the peak of the QRS complex and corresponds to the depolarization of the ventricles.

In this introduction, we'll explore how to detect R-peaks in an ECG signal using MATLAB, a popular software tool for numerical computation and data analysis.

To begin, you'll need an ECG signal in the form of a vector or a time series. MATLAB provides various functions for loading and manipulating data, so ensure that your ECG signal is available as a variable.

Next, you can use MATLAB's built-in signal processing functions and algorithms to detect R-peaks. One commonly used method is the Pan-Tompkins algorithm, which involves several steps:

- **Preprocessing:** Remove baseline wandering and noise from the ECG signal. You can apply filters, such as a high-pass filter to remove low-frequency noise and a low-pass filter to eliminate high-frequency noise.

- **Differentiation:** Take the derivative of the preprocessed signal to emphasize the QRS complex and make it easier to detect the R-peaks.

- **Squaring:** Square the differentiated signal to further enhance the QRS complex.

- **Moving Window Integration:** Apply a moving window integration to the squared signal, summing the signal values within a certain window size. This step helps identify the QRS complex peaks, including the R-peaks.

- **Thresholding:** Set an appropriate threshold to identify significant QRS complexes. You can experiment with different threshold values to achieve the desired sensitivity and specificity.

- **R-peak detection:** Determine the locations of R-peaks based on the thresholded QRS complex values. You can find local maxima or peaks in the QRS complex that surpass the threshold.

# CHAPTER-2
# INTRODUCTION TO SIGNAL PROCESSING

# 2. INTRODUCTION TO SIGNAL PROCESSING

## 2.1 Signal

In the context of signal processing, a signal refers to a representation of information that varies over time, space, or some other independent variable. A signal can be a physical phenomenon, such as sound or light, or an abstract representation of data, such as numerical values or binary codes.

Signals are typically represented as a function of one or more independent variables, such as time (in the case of time-domain signals), frequency (in the case of frequency-domain signals), or space (in the case of spatial signals). The values of a signal at different points in time or space convey information about the underlying phenomenon or data being represented.

Signals can be classified based on various attributes, such as their nature, domain, or representation. Some common types of signals include:

1. **Analog Signals:** Analog signals are continuous-time signals that vary smoothly over time or space. Examples include continuous waveforms, such as audio signals or electrical signals from sensors.

2. **Digital Signals:** Digital signals are discrete-time signals that are represented using a series of discrete samples. Each sample represents the value of the signal at a specific point in time or space. Digital signals are commonly used in modern communication systems, computing, and digital media.

3. **Continuous-time Signals:** Continuous-time signals are defined and exist for all points in time or space within a given interval. Analog signals are continuous-time signals.

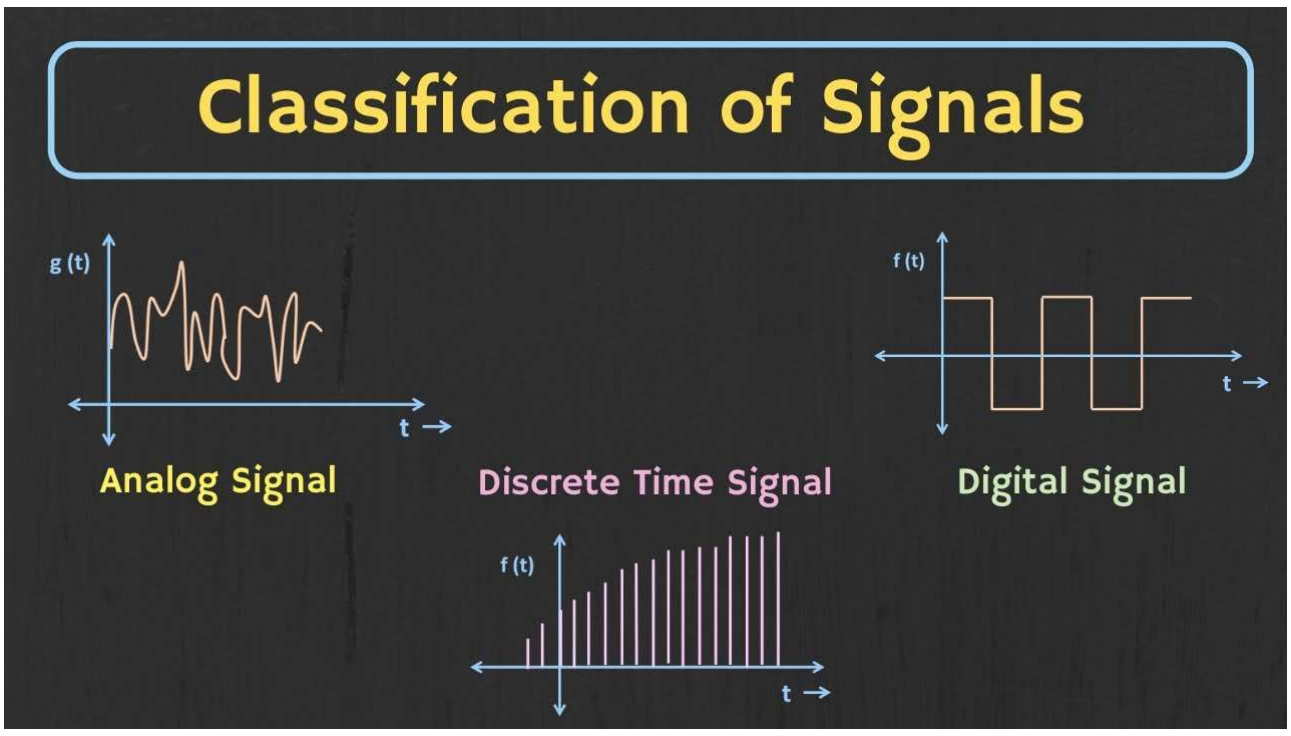
4. **Discrete-time Signals:** Discrete-time signals are defined only at specific, discrete points in time or space. Digital signals are discrete-time signals.

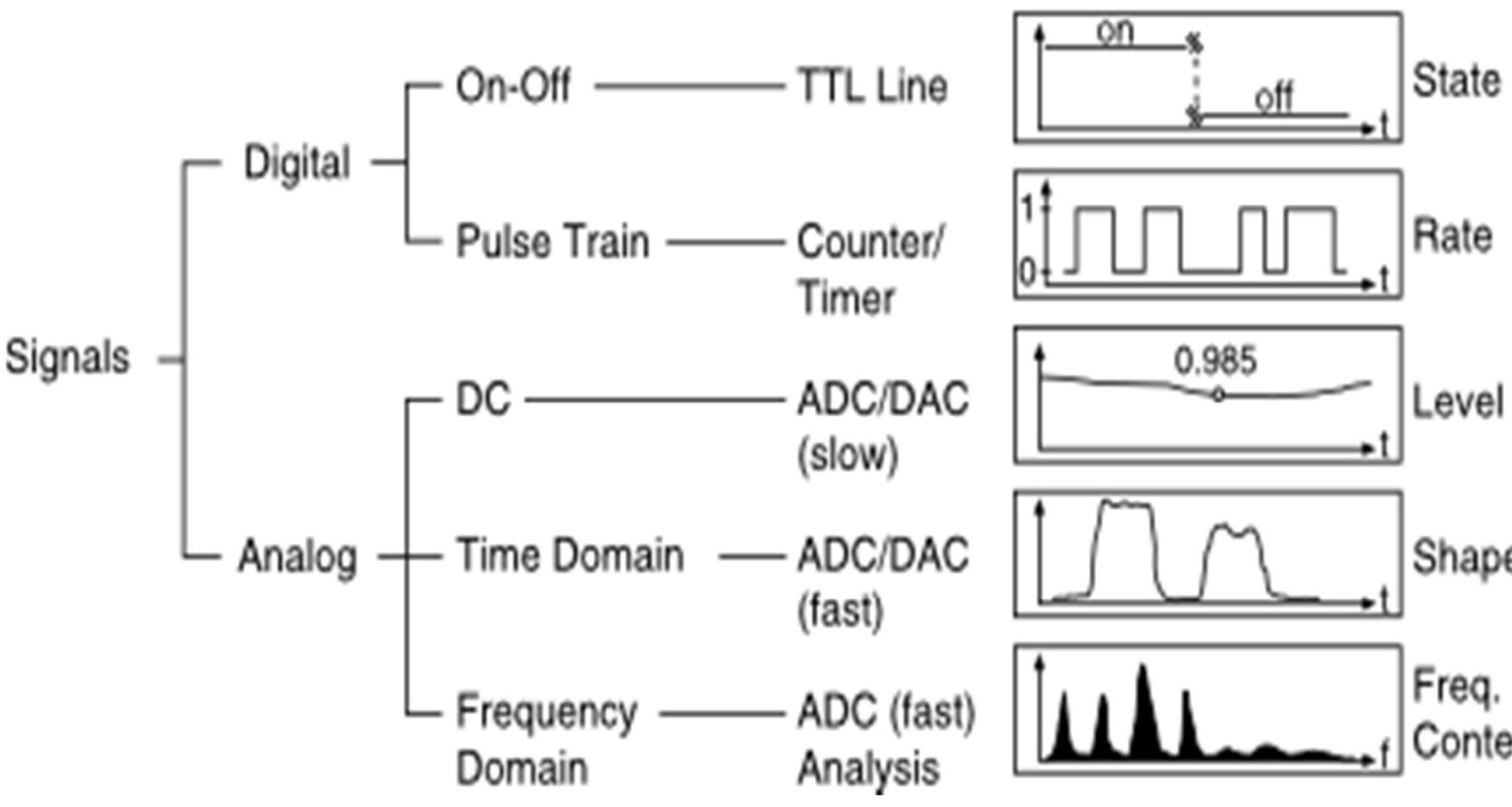
5. **Time-Domain Signals:** Time-domain signals represent the variation of a signal over time. They are typically plotted as a waveform, with time on the x-axis and signal amplitude on the y-axis.

6. **Frequency-Domain Signals:** Frequency-domain signals represent the variation of a signal in terms of its frequency content. The frequency domain

provides information about the different frequency components present in a signal and their respective amplitudes and phases. Frequency-domain representations are often obtained through techniques such as the Fourier transform or wavelet transform.

Signals are fundamental to many areas of science, engineering, and technology. Signal processing techniques aim to extract, analyze, manipulate, and interpret information from signals to enable tasks such as noise reduction, feature extraction, compression, pattern recognition, and communication.



## 2.2 Signal Processing

. Signal processing is a fundamental field in engineering and applied mathematics that deals with the analysis, modification, and manipulation of signals. A signal can be any form of information that varies over time or space, such as audio, images, video, or sensor data. The goal of signal processing is to extract meaningful information from signals, enhance their quality, or transform them into a more suitable representation for further analysis or transmission.

Signal processing techniques can be broadly categorized into two main areas: analog signal processing and digital signal processing.

**Analog Signal Processing:**

Analog signal processing involves the manipulation of continuous-time signals. It often relies on electrical circuits and analog devices to perform operations such as amplification, filtering, modulation, demodulation, and analog-to-digital conversion. Applications of analog signal processing include audio systems, radio frequency (RF) communications, and analog image processing.

**Digital Signal Processing (DSP):**

Digital signal processing focuses on the processing of discrete-time signals. These signals are typically represented as sequences of numbers or samples, obtained by periodically sampling the continuous-time signal. DSP algorithms operate on these discrete samples using computational techniques. Digital signal processing offers advantages such as flexibility, accuracy, and the ability to implement complex algorithms. It is widely used in various applications, including audio and video processing, telecommunications, biomedical signal analysis, image and video compression, and control systems.

Key concepts and techniques in signal processing include:

- **Filtering:** Filtering is the process of modifying a signal to remove or emphasize certain frequency components. It can be used to remove

noise, extract specific information, or enhance signal quality. Common types of filters include low-pass, high-pass, band-pass, and notch filters.

- **Fourier Analysis:** Fourier analysis is a mathematical technique used to decompose a signal into its frequency components. The Fourier transform and its variants, such as the Fast Fourier Transform (FFT), are widely used for spectral analysis and signal representation in the frequency domain.

- **Time-Frequency Analysis:** Time-frequency analysis methods provide information about the varying frequency content of a signal over time. Techniques such as the Short-Time Fourier Transform (STFT) and wavelet transforms are used to analyze signals that exhibit non-stationary behavior.

- **Sampling and Reconstruction:** Sampling refers to the process of converting a continuous-time signal into a discrete-time signal by capturing its values at regular intervals. Reconstruction is the reverse process of converting a discrete-time signal back into a continuous-time signal. The Nyquist-Shannon sampling theorem provides guidelines for avoiding aliasing and accurately representing continuous-time signals in the digital domain.

- **Compression:** Signal compression techniques aim to reduce the storage requirements or transmission bandwidth of signals without significant loss of information. Techniques such as lossless and lossy compression are widely used in audio, image, and video applications.

- **Statistical Signal Processing:** Statistical methods play a crucial role in signal processing. They enable the modeling and analysis of signals under uncertain or noisy conditions. Techniques such as estimation, detection, and pattern recognition are used to extract information from signals in the presence of noise or other disturbances.

Signal processing techniques are implemented using various software tools, programming languages (such as MATLAB, Python, or C++), and dedicated hardware platforms. These tools provide libraries, functions, and algorithms to perform signal processing operations efficiently.

Signal processing is a multidisciplinary field that finds applications in diverse areas such as telecommunications, audio and video processing, medical diagnostics, radar and sonar systems, seismology, speech recognition, and many others. It continues to advance and evolve as new algorithms and technologies are developed, enabling us to extract valuable information from signals and improve our understanding of the world around us.

## 2.3 Purpose of Signal Processing

The purpose of signal processing is to analyze, manipulate, and extract meaningful information from signals to achieve specific objectives. Signal processing techniques are employed in various fields and applications to enhance, interpret, or transform signals to facilitate decision-making, data analysis, and communication. Some of the key purposes of signal processing include:

- **Signal Enhancement:** Signal processing techniques can improve the quality of signals by reducing noise, removing artifacts, or enhancing desired features. By selectively modifying signal characteristics, signal processing enhances the signal's usability and interpretability.

- **Feature Extraction:** Signal processing enables the extraction of relevant features from signals. These features capture specific characteristics or patterns of interest and provide concise representations for subsequent analysis, classification, or decision-making. Feature extraction is crucial in applications such as speech recognition, image processing, and biomedical signal analysis.

- **Data Compression:** Signal processing techniques are used to reduce the size of data by eliminating redundancies or compressing information. Data compression is essential for efficient storage, transmission, and streaming of signals in various applications, including multimedia, telecommunications, and data storage systems.

- **Pattern Recognition and Classification:** Signal processing enables the recognition and classification of patterns within signals. By extracting informative features and employing classification algorithms, signals can be categorized into different classes or categories. Pattern recognition and classification find applications in speech recognition, image recognition, object detection, and anomaly detection.

- **Signal Analysis and Interpretation:** Signal processing techniques provide tools for analyzing and interpreting signals. By applying techniques such as spectral analysis, time-frequency analysis, or statistical methods, signal properties, trends, and relationships can be explored and understood. Signal

analysis aids in identifying patterns, trends, anomalies, or meaningful information embedded in the data.

- **Real-Time Processing:** Signal processing algorithms can be implemented in real-time, enabling the processing and analysis of signals in live or streaming scenarios. Real-time processing is essential in applications such as audio and video streaming, real-time monitoring, control systems, and rapid decision-making.

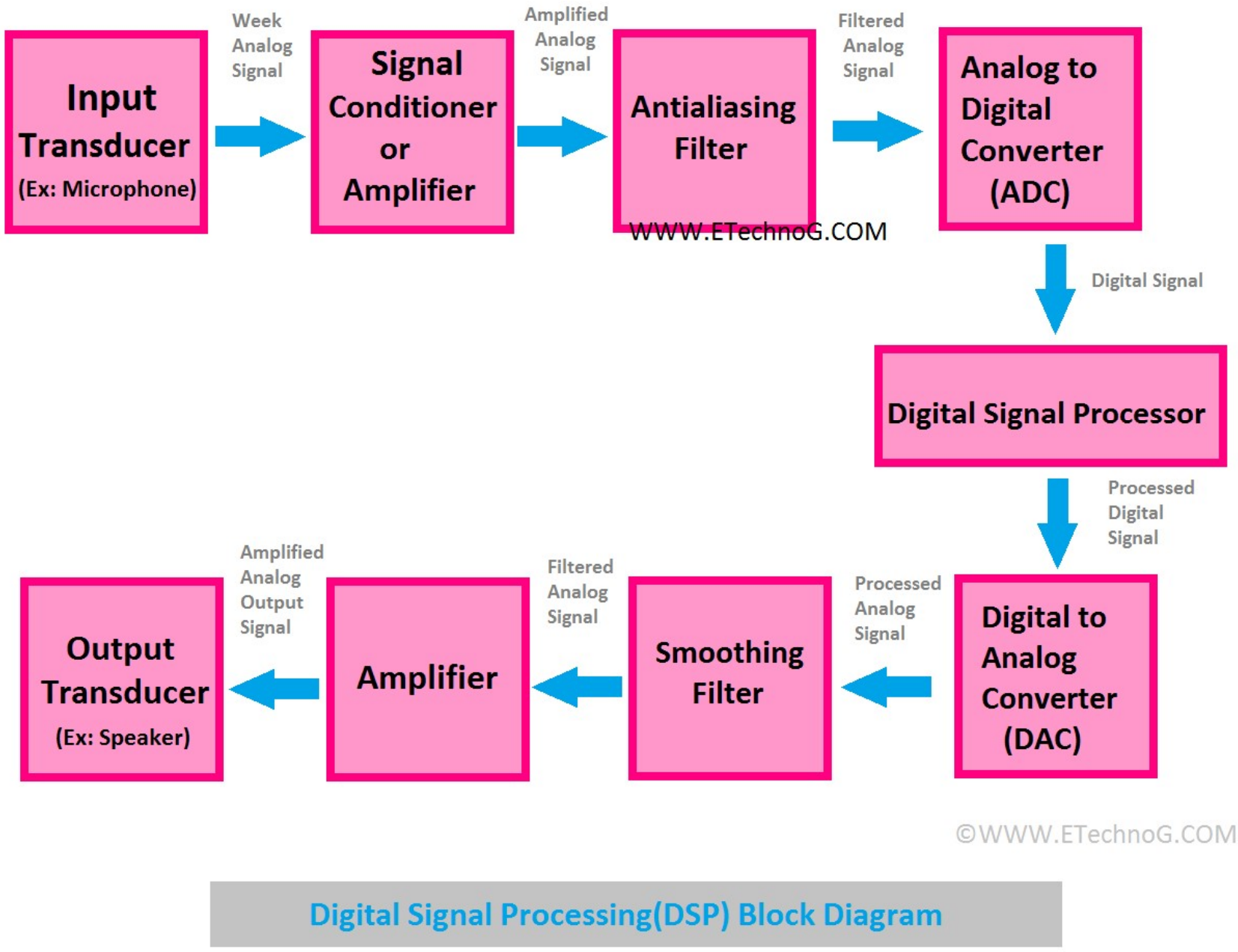
- **Communication and Information Transmission:** Signal processing plays a crucial role in various communication systems. It involves encoding, modulation, demodulation, equalization, error correction, and synchronization techniques to ensure reliable and efficient transmission of information signals over different channels and media.

- **Adaptability and Flexibility:** Signal processing techniques can be customized and adapted to suit specific signal characteristics, noise environments, or application requirements. Algorithms, filters, or feature extraction methods can be tailored to address specific challenges or exploit domain knowledge, enabling optimal performance and adaptability.

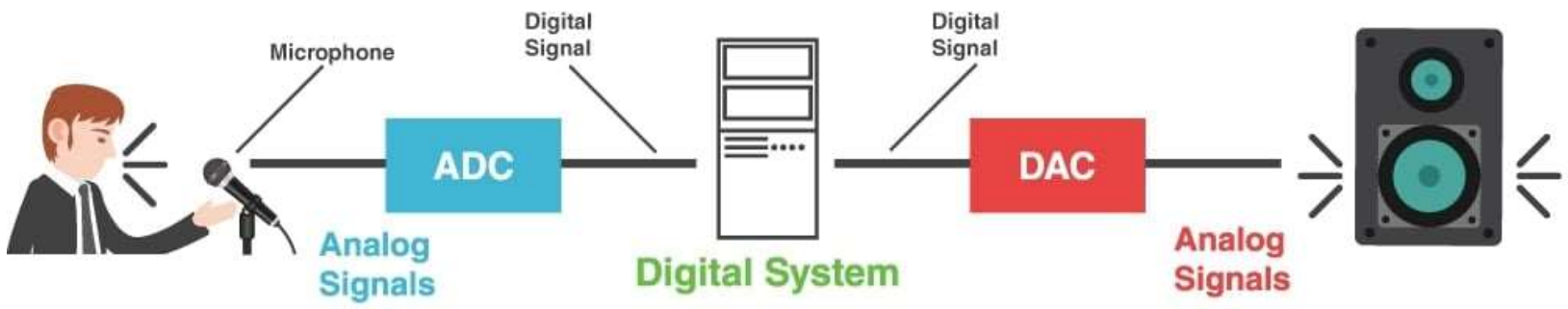
Signal processing is a fundamental tool in numerous domains, including telecommunications, audio and video processing, medical diagnostics, sensor networks, robotics, finance, and more. The purpose of signal processing is to extract meaningful information, improve signal quality, facilitate data analysis, and enable effective decision-making based on the characteristics and content of signals.

# 2.4. BLOCK DIAGRAM OF SIGNAL PROCESSING

**Figure: 01**




Digital Signal Processing(DSP) Block Diagram




## 2.5 Approach

When approaching signal processing, there are several key steps and considerations to keep in mind:

- **Define the Problem:** Clearly define the specific signal processing problem you want to solve. Identify the type of signal you are working with (e.g., audio, image, sensor data) and the specific objective you want to achieve, such as noise removal, feature extraction, or pattern recognition.

- **Signal Representation:** Determine how to represent the signal in a suitable format for processing. This may involve converting analog signals to digital form through sampling and quantization. Choose an appropriate representation, such as time-domain or frequency-domain representation, depending on the nature of the signal and the specific processing tasks.

- **Preprocessing:** Preprocess the signal to improve its quality and remove unwanted noise or artifacts. This may involve filtering (e.g., high-pass, low-pass, or band-pass filters) to eliminate noise or resampling the signal to a different frequency. Other preprocessing techniques may include baseline correction, normalization, or signal conditioning.

- **Feature Extraction:** Extract meaningful features from the signal that capture relevant information for further analysis or classification. Feature extraction techniques depend on the specific application and can range from simple statistical measures to more sophisticated methods such as Fourier analysis, wavelet transforms, or time-frequency analysis.

- **Algorithm Selection:** Choose appropriate signal processing algorithms or techniques that align with your problem and data characteristics. Consider factors such as computational complexity, accuracy, robustness to noise, and interpretability of the results. Common signal processing algorithms include filtering, spectral analysis, correlation, regression, and machine learning algorithms.

- **Implementation:** Implement the chosen algorithms using programming languages or dedicated software tools such as MATLAB, Python, or C++.

Leverage existing libraries, frameworks, or toolboxes to streamline development and utilize optimized functions for signal processing tasks.

- **Validation and Testing:** Evaluate the performance of your signal processing methods through validation and testing. Use appropriate metrics and performance measures to assess the effectiveness of the algorithms in achieving the desired objectives. Validate against ground truth or known results if available, or use cross-validation techniques to assess generalization performance.

- **Iterative Refinement:** Iterate and refine your signal processing approach based on the results and feedback obtained during the validation process. Adjust algorithm parameters, try alternative techniques, or consider different feature sets to improve the performance or address any limitations identified.

- **Documentation and Interpretation:** Document your signal processing pipeline, including the preprocessing steps, feature extraction methods, and algorithmic implementations. Document any assumptions or constraints made during the process. Interpret and communicate the results effectively to stakeholders or domain experts, providing meaningful insights derived from the processed signal.

Throughout the entire signal processing workflow, it is crucial to have a solid understanding of the underlying principles and theory behind the techniques employed. Keep in mind the specific requirements and constraints of your application, as well as any domain-specific knowledge that can inform the signal processing approach.

**Benefits of Signal Processing**

Signal processing offers several benefits and advantages across various fields and applications:

❖ **Signal Enhancement:** Signal processing techniques can improve the quality of signals by removing noise, interference, or artifacts. Filtering algorithms

can selectively remove unwanted components while preserving important signal features, resulting in cleaner and more reliable data.

- **Feature Extraction:** Signal processing allows for the extraction of relevant features from signals. These features can capture important information and patterns, enabling subsequent analysis, classification, or decision-making. Feature extraction is particularly valuable in applications such as speech recognition, image processing, and biomedical signal analysis.

- **Data Compression:** Signal processing techniques facilitate efficient data compression, reducing storage requirements or transmission bandwidth. Compression algorithms exploit redundancies or statistical properties of signals to represent them in a more compact form while maintaining an acceptable level of fidelity. Compression is widely used in multimedia applications, telecommunications, and data storage systems.

- **Pattern Recognition and Classification:** Signal processing enables the recognition and classification of patterns within signals. By extracting discriminative features and employing classification algorithms, signals can be categorized into different classes or categories. This capability is utilized in applications such as speech recognition, image recognition, object detection, and anomaly detection.

- **Signal Analysis and Visualization:** Signal processing techniques provide tools for analyzing and visualizing signals, facilitating the understanding of complex data. Spectral analysis, time-frequency analysis, and statistical techniques allow for the exploration of signal characteristics and properties. Visualization methods, such as waveform plots, spectrograms, and heatmaps, aid in the interpretation and presentation of signal information.

- **Real-Time Processing:** Signal processing algorithms can be implemented in real-time, enabling the processing and analysis of signals in live or streaming scenarios. Real-time signal processing is essential in applications such as audio and video streaming, radar and sonar systems, and real-time monitoring in healthcare or industrial settings.

- **Adaptability and Flexibility:** Signal processing techniques can be adapted and customized to fit specific signal characteristics, noise environments, or application requirements. Different algorithms, filters, or feature extraction

methods can be chosen or tailored to address specific challenges or exploit domain knowledge. This adaptability allows for the optimization and improvement of signal processing systems.

- **Automation and Efficiency:** Signal processing techniques automate the analysis and processing of signals, reducing manual effort and enabling efficient data processing. Automated signal processing systems are capable of handling large volumes of data, making them valuable in fields such as telecommunications, sensor networks, and industrial automation.

- **Interdisciplinary Applications:** Signal processing finds applications across a wide range of domains, including telecommunications, audio and video processing, medical diagnostics, robotics, environmental monitoring, finance, and more. The versatility of signal processing techniques makes them applicable in diverse fields, enabling advancements in technology and improving decision-making processes.

In summary, signal processing provides numerous benefits, including signal enhancement, feature extraction, data compression, pattern recognition, real-time processing, adaptability, automation, and interdisciplinary applications. These advantages contribute to improved data analysis, decision-making, and understanding in various domains, ultimately leading to advancements in technology and improved efficiency in various applications.

# CHAPTER-3
# SOFTWARE IMPLEMENTATION

# 3.Software Implementation

## 3.1MATLAB

MATLAB (Matrix Laboratory) is a high-level programming language and numerical computing environment widely used in academia and industry. It offers a wide range of tools for data analysis, visualization, and algorithm development. Some of its key features include: Built-in mathematical functions: MATLAB provides a comprehensive set of mathematical functions, including linear algebra, optimization, and statistics. Interactive environment: The MATLAB environment allows users to interact with their data and algorithms in an intuitive way, making it easy to test and debug code. Visualization tools: MATLAB provides powerful tools for data visualization, including 2D and 3D plotting, and animations. Toolboxes and add-ons: MATLAB offers a variety of toolboxes and add-ons for specific domains such as signal processing, control systems, and image processing. Integrations: MATLAB can integrate with other programming languages, such as C, C++, and Java, allowing users to combine the best of both worlds. In conclusion, MATLAB is a versatile tool for numerical computing and data analysis that can be applied to a wide range of applications. Whether you are working in academia or industry MATLAB offers a comprehensive and user-friendly environment for developing and testing your algorithms.

### Uses of MATLAB

- Performing numerical linear algebra.
- Numerical computation of Matrices.
- Data analysis and visualization.
- Plotting larger data sets.
- Developing algorithms.
- Creating interfaces for the user that is the GUI- Graphical User Interface and other applications that is the API – Application Programming Interface.

## 3.2 ALGORITHM

## Pan-Tompkins Algorithm:

The Pan-Tompkins algorithm is a widely used algorithm for the detection of R-peaks in an electrocardiogram (ECG) signal. It was developed by Jiapu Pan and Willis Tompkins in 1985 and has become a standard method for R-peak detection due to its simplicity and effectiveness.

The primary purpose of the Pan-Tompkins algorithm is to accurately identify the R-peaks, which correspond to the depolarization of the ventricles in the heart. Detecting R-peaks is essential in analyzing ECG signals and extracting important information about the heart's electrical activity.

The Pan-Tompkins algorithm consists of several steps:

**Preprocessing:** The ECG signal is preprocessed to remove baseline wandering and noise. This step often involves applying filters, such as a high-pass filter to eliminate low-frequency noise and a low-pass filter to attenuate high-frequency noise.

**Differentiation:** The preprocessed signal is differentiated to emphasize the QRS complex, which contains the R-peaks. Differentiation amplifies the slope of the QRS complex and helps in peak detection.

**Squaring:** The differentiated signal is squared to further enhance the QRS complex. Squaring the signal accentuates the positive values, making it easier to detect the R-peaks.

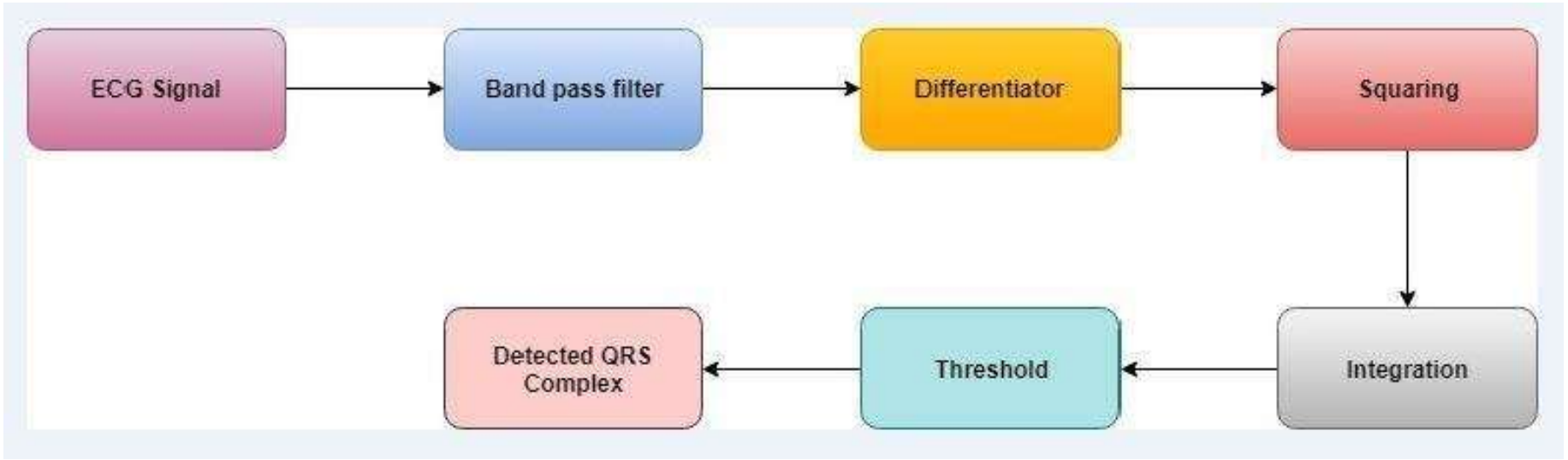
**Integration:** A moving window integration is applied to the squared signal. This involves summing the squared signal values within a specific window size. Integration smoothes the signal and helps identify the QRS complex peaks.

**Thresholding:** A threshold is set to determine significant QRS complexes. The threshold can be adaptive or fixed, depending on the algorithm variant. This step aims to differentiate between the QRS complexes and noise or other non-QRS components.

**R-peak detection:** The locations of the R-peaks are determined based on the thresholded QRS complex values. The algorithm identifies local maxima or peaks in the QRS complex that exceed the threshold.

The Pan-Tompkins algorithm can be implemented using various programming languages, including MATLAB, Python, or C++. It is important to note that while the Pan-Tompkins algorithm provides a robust foundation for R-peak detection, some modifications or adaptations may be required to suit specific ECG signal characteristics or noise environments.

The algorithm's effectiveness lies in its ability to accurately detect R-peaks in real-time ECG signals, making it widely utilized in clinical practice, research, and biomedical engineering applications.



# CHAPTER-4
# CODE IMPLEMENTATION

# 4. CODE IMPLEMENTATTION

## 4.1 MATLAB CODE

```
for demo = 1:2:3
    %   Clear our variables
    clear ecg samplingrate corrected filtered1 peaks1 filtered2 peaks2 fresult

    %   Load data sample
    switch(demo)
        case 1,
            plotname = 'Sample 1';
            load ecgdemodata1;
        case 3,
            plotname = 'Sample 2';
            load ecgdemodata2;
end

%   Remove lower frequencies
fresult=fft(ecg);%
fresult(1 : round(length(fresult)*5/samplingrate))=0;
fresult(end - round(length(fresult)*5/samplingrate) : end)=0;
corrected=real(ifft(fresult));

%   Filter - first pass
WinSize = floor(samplingrate * 571 / 1000);
disp(WinSize)
if rem(WinSize,2)==0
    WinSize = WinSize+1;
end
filtered1=ecgdemowinmax(corrected, WinSize);

%   Scale ecg
peaks1=filtered1/(max(filtered1)/7);
%   Filter by threshold filter
for data = 1:1:length(peaks1)
    if peaks1(data) < 4
        peaks1(data) = 0;
    else
        peaks1(data)=1;
    end
```

```
end
positions=find(peaks1);
distance=positions(2)-positions(1);
%   Returns minimum distance between two peaks
for data=1:1:length(positions)-1
    if positions(data+1)-positions(data)<distance
        distance=positions(data+1)-positions(data);
    end
end

% Optimize filter window size
QRdistance=floor(0.04*samplingrate);
if rem(QRdistance,2)==0
    QRdistance=QRdistance+1;
    end
    WinSize=2*distance-QRdistance;
    disp(WinSize)

    % Filter - second pass
    filtered2=ecgdemowinmax(corrected, WinSize);
    peaks2=filtered2;
    for data=1:1:length(peaks2)
        if peaks2(data)<4
            peaks2(data)=0;
        else
            peaks2(data)=1;
        end
    end

positions2=find(peaks2);
distanceBetweenFirstAndLastPeaks=positions2(length(positions2))-
positions2(1);
averageDistanceBetweenPeaks=distanceBetweenFirstAndLastPeaks/length(
positions2);
averageHeartRate=60*samplingrate/averageDistanceBetweenPeaks;
disp('Average Heart Rate = ');
disp(averageHeartRate);


    %   Create figure - stages of processing
    figure(demo); set(demo, 'Name', strcat(plotname, ' - Processing Stages'));
```

```
%   Original input ECG data
subplot(3, 2, 1);
plot((ecg-min(ecg))/(max(ecg)-min(ecg)));
title('\bf1. Original ECG');
ylim([-0.2 1.2]);

%   ECG with removed low-frequency component
subplot(3,2,2);
plot((corrected-min(corrected))/(max(corrected)-min(corrected)));
title('\bf2. FFT Filtered ECG');
ylim([-0.2 1.2]);

%   Filtered ECG (1-st pass) - filter has default window size
subplot(3,2,3);
stem((filtered1-min(filtered1))/(max(filtered1)-min(filtered1)));
title('\bf3. Filtered ECG - 1^{st} Pass');
ylim([0 1.4]);

%   Detected peaks in filtered ECG
subplot(3, 2, 4);
stem(peaks1);
title('\bf4. Detected Peaks');
ylim([0 1.4]);

%   Filtered ECG (2-d pass) - now filter has optimized window size
subplot(3,2,5);
stem((filtered2-min(filtered2))/(max(filtered2)-min(filtered2)));
title('\bf5. Filtered ECG - 2^d Pass');
ylim([0 1.4]);

%   Detected peaks - final result
subplot(3, 2, 6); stem(peaks2);
title('\bf6. Detected Peaks - Finally');
ylim([0 1.4]);

%   Create figure - result
figure(demo+1);
set(demo+1, 'Name', strcat(plotname, ' - Result'));

%   Plotting ECG in green
plot((ecg-min(ecg))/(max(ecg)-min(ecg)), '-g');
title('\bf Comparative ECG R-Peak Detection Plot');
```

```
    %   Show peaks in the same picture
    hold on

  %   Stemming peaks in dashed black
    stem(peaks2'.*((ecg-min(ecg))/(max(ecg)-min(ecg)))', ':k');

    %   Hold off the figure
    hold off
end
```

## 4.2 CODE DESCRIPTION

**fft( )**

fft Discrete Fourier transform.

fft(X) is the discrete Fourier transform (DFT) of vector X. For matrices, the fft operation is applied to each column. For N-D arrays, the fft operation operates on the first non-singleton dimension.

**ifft( )**

ifft Inverse discrete Fourier transform.

ifft(X) is the inverse discrete Fourier transform of X.

ifft(X,N) is the N-point inverse transform.

ifft(X,[],DIM) or ifft(X,N,DIM) is the inverse discrete Fourier transform of X across the dimension DIM.

ifft(..., 'symmetric') causes ifft to treat X as conjugate symmetric along the active dimension. This option is useful when X is not exactly conjugate symmetric merely because of round-off error. See the reference page for the specific mathematical definition of this symmetry.

**switch( )**

switch Switch among several cases based on expression.

The general form of the switch statement is:

```
switch switch_expr
  CASE case_expr,
    statement, ..., statement
  CASE {case_expr1, case_expr2, case_expr3,...}
    statement, ..., statement
 ...
  OTHERWISE,
    statement, ..., statement
END
```

**figure( )**

figure Create figure window.

figure, by itself, creates a new figure window, and returns its handle.

figure(H) makes H the current figure, forces it to become visible, and raises it above all other figures on the screen. If Figure H does not exist, and H is an integer, a new figure is created with handle H.

**subplot( )**

subplot Create axes in tiled positions.

H = subplot(m,n,p), or subplot(mnp), breaks the Figure window into an m-by-n matrix of small axes, selects the p-th axes for

the current plot, and returns the axes handle. The axes are counted along the top row of the Figure window, then the second row, etc.

**stem( )**

stem Discrete sequence or "stem" plot.

stem(Y) plots the data sequence Y as stems from the x axis terminated with circles for the data value. If Y is a matrix then each column is plotted as a separate series.

stem(X,Y) plots the data sequence Y at the values specified in X.

stem(...,'filled') produces a stem plot with filled markers.

stem(...,'LINESPEC') uses the linetype specified for the stems and markers. See PLOT for possibilities.

stem(AX,...) plots into axes with handle AX. Use GCA to get the handle to the current axes or to create one if none exist.

H = stem(...) returns a vector of stemseries handles in H, one handle per column of data in Y.

**plot( )**

plot Linear plot.

plot(X,Y) plots vector Y versus vector X. If X or Y is a matrix, then the vector is plotted versus the rows or columns of the matrix, whichever line up. If X is a scalar and Y is a vector, disconnected line objects are created and plotted as discrete points vertically at X.

**ylim( )**

ylim Y limits.

YL = ylim gets the y limits of the current axes.

ylim([YMIN YMAX]) sets the y limits.

**Set( )**

set Set object properties.

set(H,'PropertyName',PropertyValue) sets the value of the specified property for the graphics object with handle H. H can be a vector of handles, in which case set sets the properties' values for all objects of H.

set(H,a) where a is a structure whose field names are object property names, sets the properties named in each field name with the values contained in the structure.

**title( )**

title Graph title.

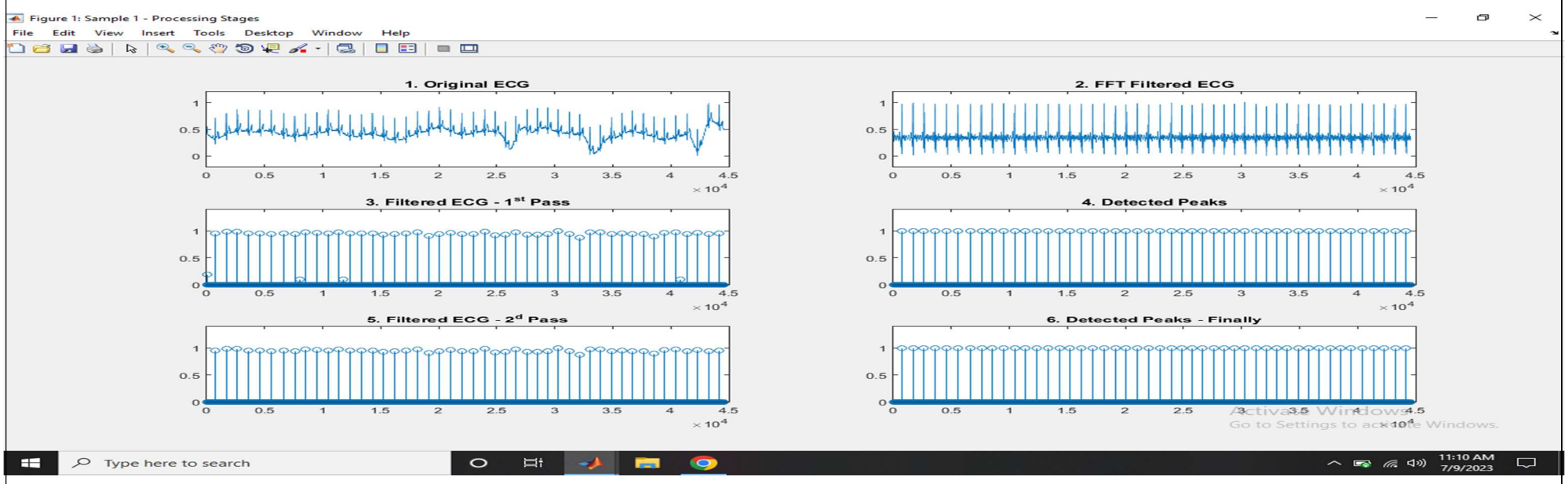
title('text') adds text at the top of the current axis.

title('text','Property1',PropertyValue1,'Property2',PropertyValue2,...) sets the values of the specified properties of the title.

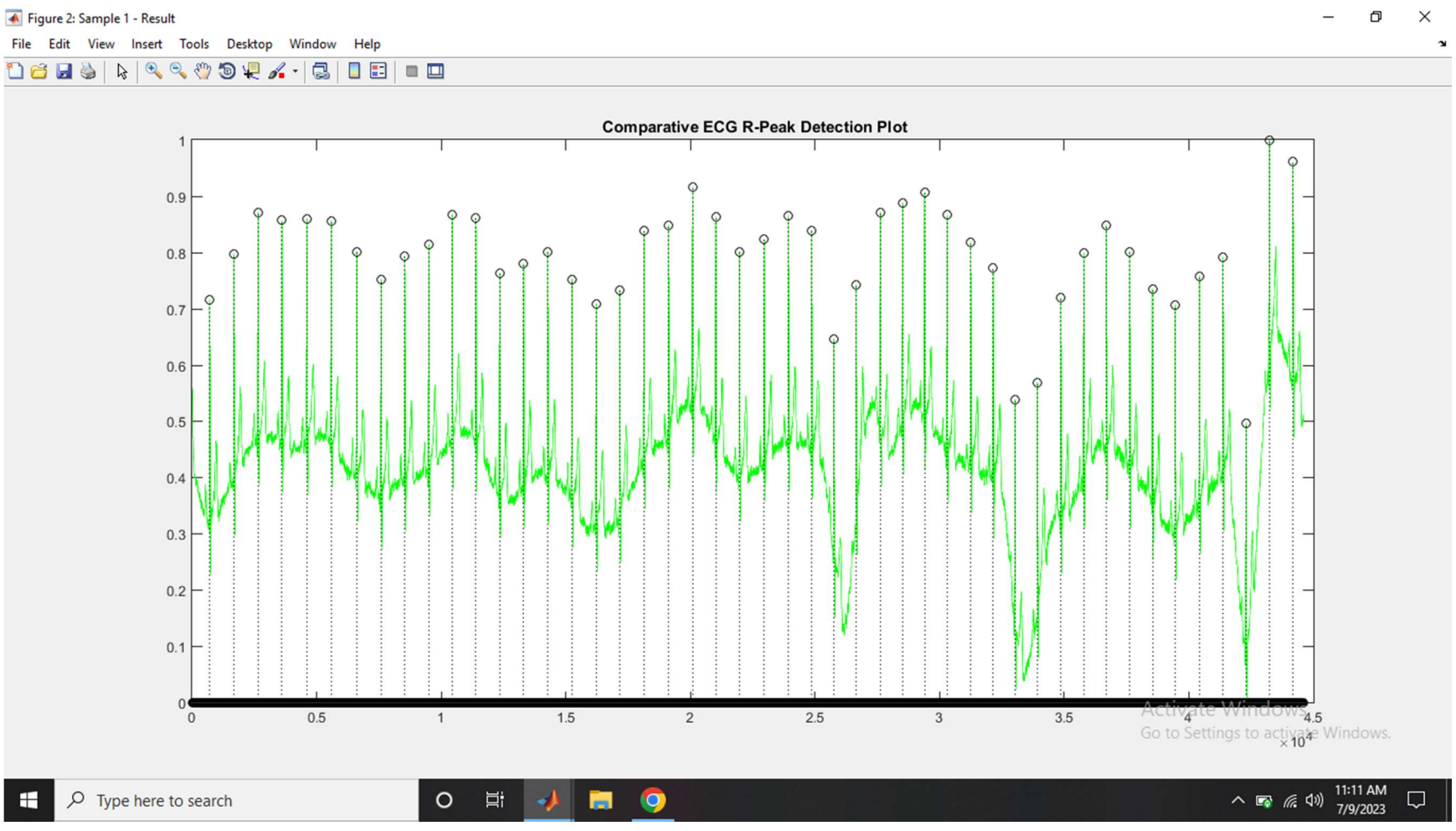
# CHAPTER-5
# OUTPUT

## 5.1. RESULT

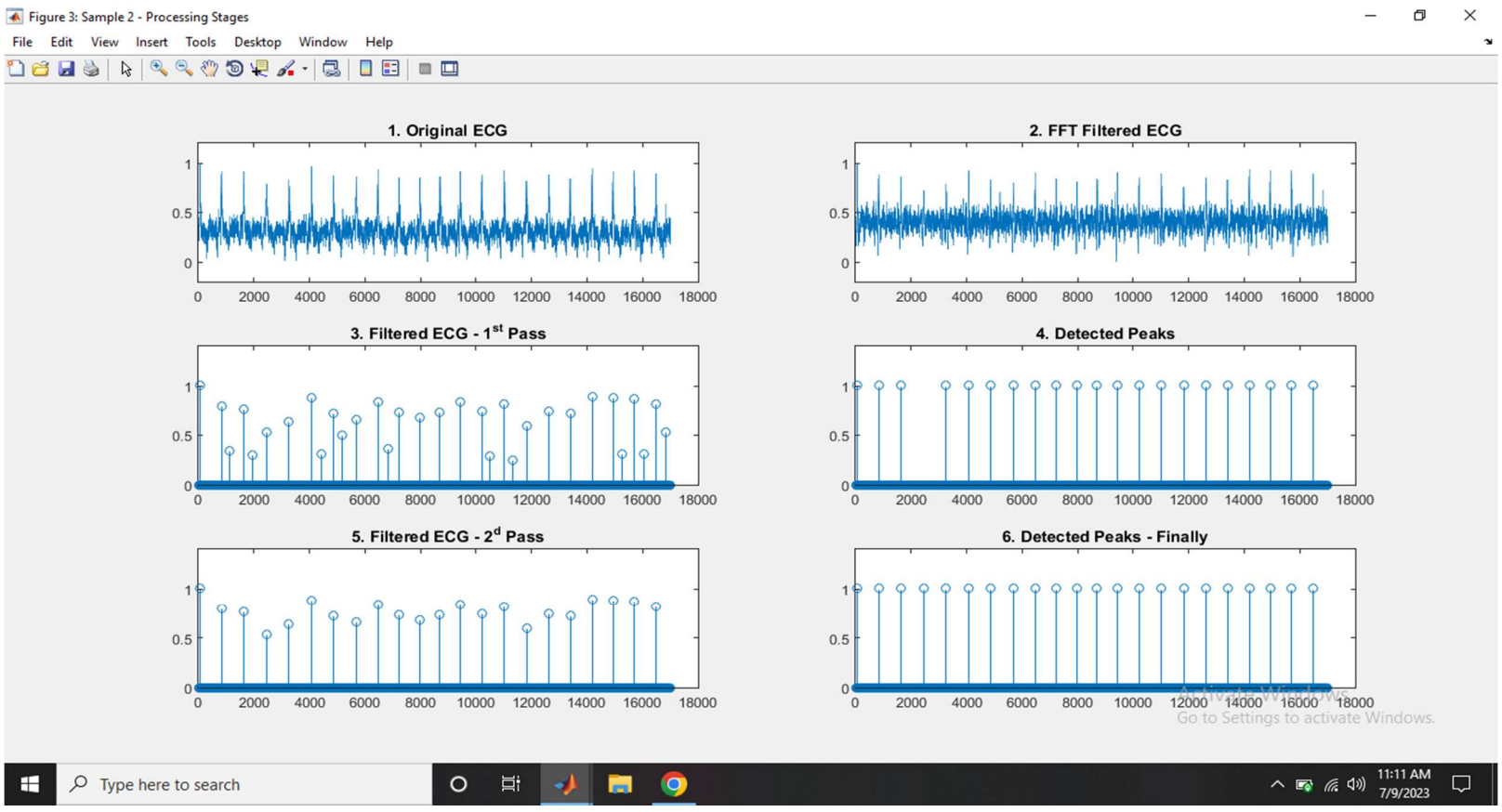
**Sample-1:-** Processing stages



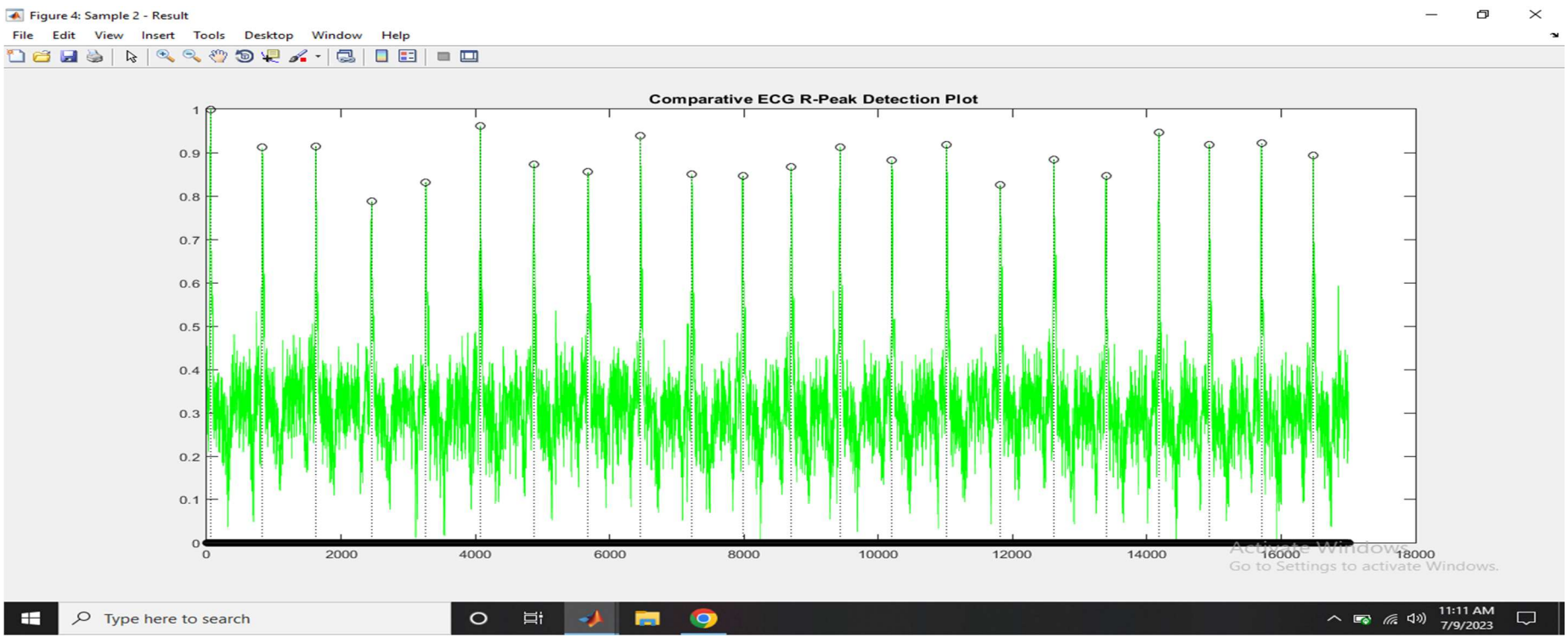

**Sample-2:-** Result



# **Sample-2:-** Processing stages




# **SAMPLE-2:-** Result



# CHAPTER-6

# ADVANTAGES AND DISADVANTAGES

# 6. ADVANTAGES AND DISADVANTAGES

## 6.1 Advantages

Accurate R-peak detection: MATLAB provides powerful signal processing functions and algorithms, enabling accurate detection of R-peaks in ECG signals. The Pan-Tompkins algorithm, commonly used in ECG R-peak detection, can be implemented efficiently in MATLAB, leading to reliable results.

- **Comprehensive toolset:** MATLAB offers a comprehensive set of functions, toolboxes, and visualization capabilities specifically designed for signal processing and ECG analysis. These tools simplify the implementation process and provide a wide range of options for preprocessing, feature extraction, and result visualization.

- **Fast prototyping and development:** MATLAB's interactive environment and intuitive syntax facilitate rapid prototyping and iterative development. You can quickly implement, test, and refine different signal processing algorithms and parameters to optimize the R-peak detection performance.

- **Extensive community support:** MATLAB has a large and active user community, providing access to resources, forums, and documentation to address any challenges you may encounter during the project. You can benefit from the experiences and insights of other users to improve your implementation.

- **Integration with other domains:** MATLAB seamlessly integrates with other domains such as data analysis, machine learning, and visualization. This integration allows you to leverage advanced analytics, statistical methods, and visualization techniques to gain deeper insights from the ECG data and enhance the project's outcomes.

## 6.2 Disadvantages

Cost: MATLAB is a commercial software, and licensing fees may be required to use it. However, there are open-source alternatives such as Octave that provide similar functionality at no cost.

- **Learning curve:** MATLAB has a wide range of features and functions, which may require some learning and familiarity to effectively utilize them for the project. Understanding MATLAB syntax, functions, and toolboxes may take some time, especially for users with limited programming experience.

- **Processing time:** Depending on the size of the ECG dataset and the complexity of the implemented algorithms, processing time can be a concern. MATLAB provides optimization techniques and parallel processing capabilities, but for real-time or high-throughput applications, additional optimization may be necessary.

- **Customization and adaptation:** While MATLAB offers a wide range of built-in functions and algorithms, customization or adaptation may be required to address specific challenges or optimize the R-peak detection performance for your ECG dataset. Implementing custom algorithms or modifying existing ones might involve additional coding and testing.

- **Algorithm validation:** It is crucial to validate the R-peak detection algorithm against ground truth or reference data to ensure its accuracy and reliability. Proper validation requires access to high-quality annotated ECG datasets and expertise in ECG interpretation.

# CHAPTER-7
# CONCLUSION

# 7. CONCLUSION

In conclusion, utilizing MATLAB for the detection of R-peaks in an electrocardiogram (ECG) signal offers several advantages. MATLAB provides powerful signal processing capabilities, including filtering, differentiation, integration, and thresholding functions, which are essential for implementing the Pan-Tompkins algorithm or other R-peak detection methods. Additionally, MATLAB's extensive library of functions and toolboxes, specifically designed for signal processing, simplifies the development process and accelerates implementation.

The active MATLAB community and available documentation ensure that users can find support and resources to overcome challenges during the project. MATLAB's interactive environment facilitates rapid prototyping, iterative refinement, and visualization of the ECG signals and detected R-peaks. Furthermore, MATLAB's integration with other domains, such as data analysis and machine learning, allows for enhanced analysis and interpretation of the ECG data.

While there may be considerations such as the cost of MATLAB licenses, the learning curve associated with mastering MATLAB's features, and potential processing time issues, the benefits of using MATLAB for ECG R-peak detection outweigh these concerns. MATLAB provides a robust platform for accurate R-peak detection, enabling researchers, clinicians, and biomedical engineers to gain valuable insights into the heart's electrical activity and improve cardiac health assessment.

Overall, leveraging MATLAB's signal processing capabilities for ECG R-peak detection project allows for efficient implementation, effective analysis, and visualization of ECG signals, ultimately contributing to improved diagnosis, monitoring, and understanding of cardiac function.

# REFERENCES

- Most of the content in this presentation has been picked up from
  - Medo Yasar YouTube Video - https://www.youtube.com/watch?v=KWrzdJY4G9Q
  - Sergey Chernenko's code - http://www.librow.com/cases/case-2