University of Bahrain
College of Information Technology
Department of Computer Engineering

# Heart Analysis

**Prepared By**

**Name:** Ali Redha Ali/ Sayed Mohammed Baqer

**ID:** 20195330/202008216

**Sec:** 01

**Course Number:** ITCE340/272

# Contents

# List of figures

## Objectives

- ☐ Implement Heart Analysis Algorithm

- ☐ Make an GUI for the Heart Analysis

- ☐ Utilize what you learned in class to solve problems

## 1.Introduction :

We have chosen to do this project since it relates to our daily life and it's important thing that a human being should know what is heart beat at since it will inform you a lot about your body and this project could help us in the future if we want to design a gadget for to capture the heart sound then turn it into real data that you can benefit from and data could be sent to the cloud to be processed and give the information back to the user and tell what they should be looking for

# 2. Tools

## 2.1Vscode

We use it as code editor since have very useful extension that help you code and many shortcuts that let you work much faster and efficiently and some it’s have some AI that suggest code biased on your style of coding also you can use MATLAB terminal inside

## 2.2Matlab App designer

We use MATLAB App designer since it’s the fastest way we can design an app for a MATLAB and since it’s object-oriented programming we didn’t have a problem understanding the code

## 3. Code discussion

```
% Callbacks that handle component events
methods (Access = private)
    % This method is called when the Button is released
    % Button pushed function: Button
    function ButtonPushed(app, event)
        % This this will prompt the user to select a file to open and then will choose type of file to open either a .wav or .mp3 fileor a .mp4 file
        [fileName, pathName] = uigetfile({'*.mp4;*.mp3;*.wav;'},'Select a video');
        if ~isequal(fileName,0)
            % This will  openth the file and then will play the file using Video Reader funcction
            processed_video =  VideoReader(fullfile(pathName,fileName));

            % This will take the Height and width of the video and then will set the size of the axes to the size of the video
            vidHeight= processed_video.Height;
            vidWidth = processed_video.Width;
            % The number of channels in the video
            nChannels = 3;

            framerate = processed_video.FrameRate; % The framerate of the video

            len = processed_video.NumFrames; % The number of frames in the video
```

```
    UIAxes                  matlab.ui.control.UIAxes  % UIAxes
end
```

*Figure 1 class of the app*

This the class of app and it' have all of it's properties and each explain in as comments .


```
% Callbacks that handle component events
methods (Access = private)
    % This method is called when the Button is released
    % Button pushed function: Button
    function ButtonPushed(app, event)
        % This this will prompt the user to select a file to open and then will choose type of file to open either a .wav or .mp3 fileor a .mp4 file
        [fileName, pathName] = uigetfile({'*.mp4;*.mp3;*.wav;'},'Select a video');
        if ~isequal(fileName,0)
            % This will  openth the file and then will play the file using Video Reader funcction
            processed_video =  VideoReader(fullfile(pathName,fileName));

            % This will take the Height and width of the video and then will set the size of the axes to the size of the video
            vidHeight= processed_video.Height;
            vidWidth = processed_video.Width;
            % The number of channels in the video
            nChannels = 3;

            framerate = processed_video.FrameRate; % The framerate of the video

            len = processed_video.NumFrames; % The number of frames in the video
```


*Figure 2 Callback main function*

This the main fucntion where the user is required to choose the video format that is supported by matlab then fileName and pathname will be storted to be but in the fucntion of VideoReader that is natively by matlab then it calculates the information needed for caluclation, such as the height, width, and number of channels of the video, as well as the frame rate.

```
% This will create a matrix of the size of the video and then will
          % read the video into the matrix
          Heart_data_type= struct('cdata', zeros(vidHeight, ...
              vidWidth, nChannels, 'uint8'), 'colormap', []);



          T = 1/framerate;  % The time between each frame
          tlen = len/framerate; % The length of the video in seconds
          t1 = linspace(0, tlen, len); % The time vector for the video
f
```

*Figure 3 fucntion video size*

In this function, a matrix of the size of the video is generated based on the height, width, and channel of the video. After that, the function calculates the time period and time of the signal.

```
f = -framerate/2:framerate/len:framerate/2-framerate/len;
            mov(1:len) = struct('cdata', zeros(vidHeight, vidWidth, ...
                nChannels, 'uint8'), 'colormap', []); % Preallocate movie structure.

            app.Button.Text = 'Processing...'; % Change the text of the button to Processing...
            % Put text on top of icon
            app.Button.IconAlignment = 'bottom'; % Align the text to the bottom of the icon
```

*Figure 4 frame rate*

This function calculates f, and then moves the struct preallocated by its prototype based on the video's height, width, and channel. If we insert a video, the button will change to processing and be repositioned at the bottom.

```
% Find the center 5th to average

            centerVerMin = ceil((vidHeight/5) ...
                + (vidHeight/5));  % The minimum vertical center point


            centerVerMax = ceil((vidHeight/5) ...
                + (vidHeight/5) + (vidHeight/5)); % The maximum vertical center point

            centerHorMin = ceil((vidWidth/5) ...
                + (vidWidth/5)); % The minimum horizontal center point


            centerHorMax = ceil((vidWidth/5)  ...
                + (vidWidth/5) +  + (vidWidth/5)); % The maximum horizontal center point

            sampleHor = centerHorMax - centerHorMin; % The horizontal sample size
            sampleVer = centerVerMax - centerVerMin; % The vertical sample size

            sample = zeros(sampleVer, sampleHor, len); % Preallocate sample matrix


            t = zeros(len); % Preallocate time vector
            avg = zeros(len);  % Preallocate average vector
```

*Figure 5 Signal sample size*

The function prepares a matrix of sample horizontal size and sample vertical size according to maximum horizontal and vertical lines and minimum horizontal and vertical lines based on the number of frequencies of the horizontal and vertical lines.

```
wbar = permute(repmat(app.Button.BackgroundColor,15,1,200),[1,3,2]);
        % Black frame around waitbar
        wbar([1,end],:,:) = 0;
        wbar(:,[1,end],:) = 0;
        % Load the empty waitbar to the button
        app.Button.Icon = wbar;

        % Grab all the frames and put into our struct
        for k = 1 : len
            mov(k).cdata = read(processed_video, k);
        end

```

*Figure 6 Button installation*

The function creates a loading button and gives it a black frame, then it loads the frame to the button, then a for loop is used to save all readable frames and add them to the struct.

```
for k = 1 : len

              currentProg = min(round((size(wbar,2)-2)*(k/len)), ...
                  size(wbar,2)-2); % Current progress of the waitbar


               RGB = app.Button.Icon; % Get the current icon
               RGB(2:end-1, 2:currentProg+1, 1) = 0.6350  ; % Change the red channel
               RGB(2:end-1, 2:currentProg+1, 2) = 0.0780 ;  % Change the green channel
               RGB(2:end-1, 2:currentProg+1, 3) = 0.1840;   % Change the blue channel
               app.Button.Icon = RGB; % Update the icon

               % Pause to slow down animation
               pause(.3)
               for j = centerHorMin : centerHorMax
                   for i = centerVerMin : centerVerMax
                       sampleHorIndex = j - centerHorMin + 1;
                       sampleVerIndex = i - centerVerMin + 1;
                       sample(sampleVerIndex, ...
                           sampleHorIndex, k) = ...
                           mov(k).cdata(i, j, 1);
                   end
               end
               t(k) = k;   % value of k (frames) for graphing
               avg(k) = mean(mean(sample(:,:,k)));


           end
```

*Figure 7 loading animation*

In this function, the progress loading button is set and its color is changed. Next, a sample is created based on the vertical index, the horizontal index, and the frequency number. The average of the sample is calculated based on the sample frequency number.

```
% remove waitbar
        app.Button.Icon = '';
        % Change button name
        app.Button.Text = 'Browse';

        Y = fftshift(fft(avg)); % FFT of the average
        plot(app.UIAxes,t,avg ,'LineWidth',2 ,'Color', '#eb4939'); % Plot the average


        title(app.UIAxes,['{\bf Mangitude of Frequency' ...
            ' to find Heart Rate }'], 'color', ...
            '#92b9e4','FontSize', ...
            14,'FontName' ...
            ,'TimeNewRoman'); grid on % Title the graph

        xlabel(app.UIAxes,'Frame', 'color', '#ff5d8e' ...
            ,'FontSize', ...
            14,'FontName' ...
            ,'TimeNewRoman'); % Label the x axis


        ylabel(app.UIAxes,'Pixel intensity', ...
            'color', '#ff5d8e','FontSize', ...
            14,'FontName' ...
            ,'TimeNewRoman'); % Label the y axis

        axis(app.UIAxes, 'auto') % Set the axis to auto scale


          yo=1/len*abs(Y) ; % Magnitude of the FFT

        stem(app.UIAxes2,f,yo,'filled', ...
            'LineWidth',2 ,'Color', '#eb4939'); % Plot the FFT



        title(app.UIAxes2,['{\bf Mangitude of ' ...
            'Frequency to find Heart Rate }'], 'color', ...
            '#92b9e4','FontSize', ...
            14,'FontName' ...
            ,'TimeNewRoman'); grid on  % Title the graph



        xlabel(app.UIAxes2,'Frequency (Hz)', ...
            'color', '#ff5d8e','FontSize', ...
            14,'FontName','TimeNewRoman'); % Label the x axis


        ylabel(app.UIAxes2,'Magnitude', 'color', ...
            '#ff5d8e','FontSize', ...
            14,'FontName' ...
            ,'TimeNewRoman'); % Label the y axis
```

*Figure 8 plot FFT signal*

This function sets the button to null, removing the label and then changing it to Browse. Next, use Fast Fourier transform to calculate the average signal. Give the signal a title and label on the graph and set the axis to autoscale. Next, plot the signal in continuous and discrete form

```
axis(app.UIAxes2, 'tight') % Set the axis to tight scale


    r1=max(real(yo), [], 'all'); % Find the maximum of the real part of the FFT
    app.YourBPMEditField.Value=r1; % Display the BPM

```

*Figure 9 display the bpm*

This function sets the axis of the discrete signal to tight scale, then finds the maximum real part of magnitude signal FFT and uses it to display the BPM value

```
 Component initialization
methods (Access = private)

    % Create UIFigure and components
    function createComponents(app)

        % Get the file path for locating images
        pathToMLAPP = fileparts(mfilename('fullpath'));

        % Create UIFigure and hide until all components are created
        app.UIFigure = uifigure('Visible', 'off'); % Create the figure
        app.UIFigure.Color = [0.1255 0.1412 0.1569];  % Set the background color
        colormap(app.UIFigure, 'cool');  % Set the colormap to cool
        app.UIFigure.Position = [100 100 1120 808]; % Set the figure size
        app.UIFigure.Name = 'MATLAB App'; % Set the figure name

        % Create UIAxes
        app.UIAxes = uiaxes(app.UIFigure); % Create the axes
        xlabel(app.UIAxes, 'Frame') % Label the x axis
        ylabel(app.UIAxes, 'Pixel insitisy ') % Label the y axis
        zlabel(app.UIAxes, 'Z') % Label the z axis
        app.UIAxes.FontName = 'Times New Roman'; % Set the font name
        app.UIAxes.XColor = [1 1 1]; % Set the x axis color
        app.UIAxes.YColor = [1 1 1]; % Set the y axis color
        app.UIAxes.ZColor = [1 1 1]; % Set the z axis color
        app.UIAxes.Color = 'none'; % Set the axes color to none
        app.UIAxes.FontSize = 14; % Set the font sizek
        app.UIAxes.GridColor = [0.15 0.15 0.15];  % Set the grid color
        colormap(app.UIAxes, 'cool') % Set the colormap to cool
        app.UIAxes.Position = [12 219 516 385]; % Set the axes position

        % Create UIAxes2
        app.UIAxes2 = uiaxes(app.UIFigure); % Create the axes
        xlabel(app.UIAxes2, 'Frequency(Hz)') % Label the x axis
        ylabel(app.UIAxes2, 'Magnitude') % Label the y axis
        zlabel(app.UIAxes2, 'Z') % Label the z axis
        app.UIAxes2.FontName = 'Times New Roman'; % Set the font name
        app.UIAxes2.FontWeight = 'bold'; % Set the font weight
        app.UIAxes2.XColor = [1 1 1]; % Set the x axis color
        app.UIAxes2.YColor = [1 1 1]; % Set the y axis color
        app.UIAxes2.ZColor = [1 1 1]; % Set the z axis color
        app.UIAxes2.Color = 'none'; % Set the axes color to none
        app.UIAxes2.FontSize = 14; % Set the font sizek
        app.UIAxes2.MinorGridColor = [0.1 0.1 0.1];
        colormap(app.UIAxes2, 'cool')
        app.UIAxes2.Position = [592 226 484 378];
```

*Figure 10 The creation of UI figures*

In this function, the access will be changed to private call then will create UI-figure for user and load the picture from specific path then will create Ui-figure and display it until all other components are set to visible. The UI-axis will be created by aligning the x, y, and z axes and changing their color and position as shown in the UI-axis 2 by setting the font to Times Roman.

```
% Create Button
        app.Button = uibutton(app.UIFigure, 'push');
        app.Button.ButtonPushedFcn = createCallbackFcn(app, @ButtonPushed, true);
        app.Button.HandleVisibility = 'callback';
        app.Button.Icon = fullfile(pathToMLAPP, 'Folder_opener.png');
        app.Button.IconAlignment = 'center';
        app.Button.BackgroundColor = [0.1255 0.1412 0.1569];
        app.Button.FontColor = [1 1 1];
        app.Button.Position = [446 75 230 82];
        app.Button.Text = '';

        % Create Image
        app.Image = uiimage(app.UIFigure);
        app.Image.Position = [197 630 311 168];
        app.Image.ImageSource = fullfile(pathToMLAPP, 'heartbeat_1_Regular.gif');

        % Create Image2
        app.Image2 = uiimage(app.UIFigure);
        app.Image2.Position = [480 645 114 139];
        app.Image2.ImageSource = fullfile(pathToMLAPP, 'main_logo.gif');

        % Create YourBPMEditFieldLabel
        app.YourBPMEditFieldLabel = uilabel(app.UIFigure);
        app.YourBPMEditFieldLabel.HorizontalAlignment = 'center';
        app.YourBPMEditFieldLabel.VerticalAlignment = 'top';
        app.YourBPMEditFieldLabel.FontName = 'Times New Roman';
        app.YourBPMEditFieldLabel.FontSize = 40;
        app.YourBPMEditFieldLabel.FontWeight = 'bold';
        app.YourBPMEditFieldLabel.FontColor = [1 1 1];
        app.YourBPMEditFieldLabel.Position = [135 174 188 53];
        app.YourBPMEditFieldLabel.Text = 'Your BPM';

        % Create YourBPMEditField
        app.YourBPMEditField = uieditfield(app.UIFigure, 'numeric');
        app.YourBPMEditField.HorizontalAlignment = 'center';
        app.YourBPMEditField.FontName = 'Times New Roman';
        app.YourBPMEditField.FontSize = 40;
        app.YourBPMEditField.FontColor = [1 1 1];
        app.YourBPMEditField.BackgroundColor = [0.1255 0.1412 0.1569];
        app.YourBPMEditField.Position = [111 51 234 106];

        % Create Image_2
        app.Image_2 = uiimage(app.UIFigure);
        app.Image_2.Position = [566 630 311 168];
        app.Image_2.ImageSource = fullfile(pathToMLAPP, 'heartpulse_inverese.gif');

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

*Figure 11 BPM label and creation button*

In this function, first we will create a push button and modify its function by changing its icon, alignment and background color. Then we will set the text to null, create a GIF image by aligning its position and selecting its path, create the label for the BPM field by aligning its position and name and selecting its font and color, and finally make the figure visible upon completion of the creation of all other components.

```
% App creation and deletion
methods (Access = public)

    % Construct app
    function app = Heart

        % Create UIFigure and components
        createComponents(app)

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

        if nargout == 0
            clear app
        end
    end

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

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

*Figure 12 Creation of the app's GUI*

This the creation of app it's self so we can call the function as an objects

And this defualt of configuration done by matlab

# 4. GUI

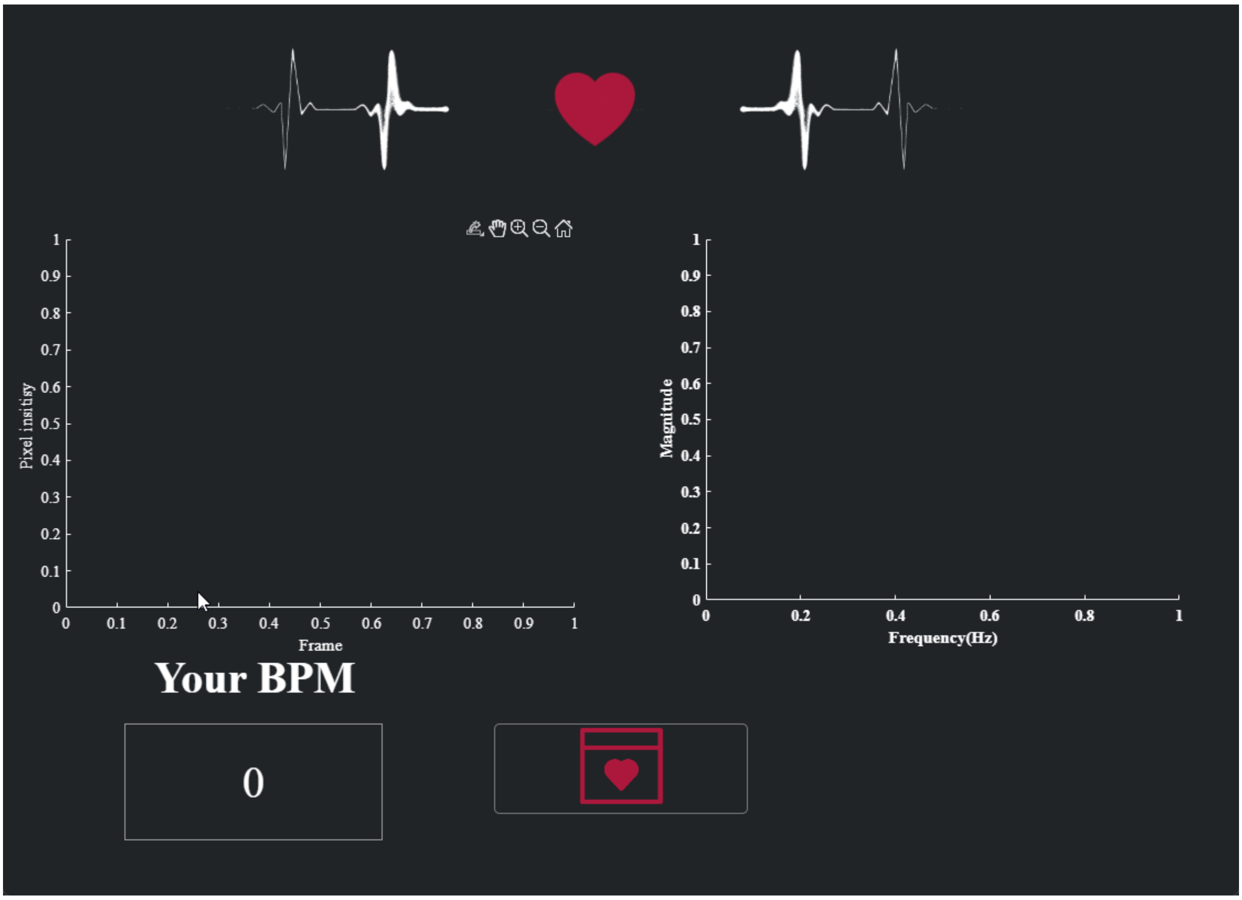


*Figure 13 GUI without data loading*

This and overlook of the GUI without loading the data

# 5. Implementation

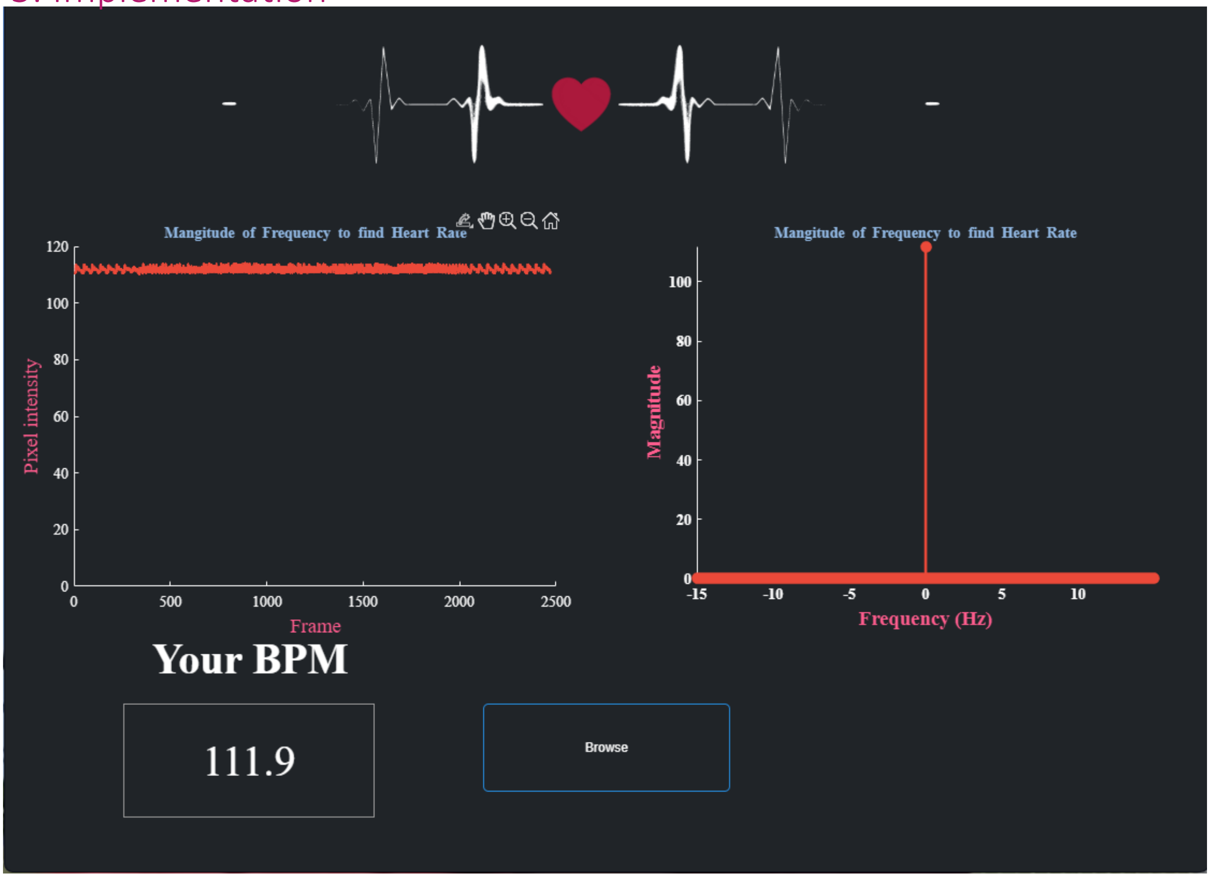


*Figure 14 GUI with data loading*

Test the loading of the data with real data and the GUI while loading it

## 5.Conclusion

As a result of the implementation of the algorithm and the development of the GUI, we have achieved our objectives and have learned how we can apply our knowledge to solve problems. The next step in implementing this project will be to use hardware to collect data directly from users and analyze it.

# 6. Reference

[1]"heart_rate_from_video." *GitHub*, Nov-2017. Available: https://github.com/aaronpenne/dsp/blob/master/heart_rate/heart_rate_from_video.m. [Accessed: 15-Aug-2022].

[2] "This project generates salt and pepper and gaussian noises, and applies low pass filter." *GitHub*, Jun-2018. Available: https://github.com/Mohammed-Raafat/Image-Processing-Project. [Accessed: 15-Aug-2022].

[3] "Is it possible to change label names in Matlab." *mathworks*, Jun-2015. Available: https://www.mathworks.com/matlabcentral/answers/225165-is-it-possible-to-change-label-names-in-matlab. [Accessed: 15-Aug-2022].

[4] "Convert complex double to double type," *mathworks*, May-2016. Available: https://www.mathworks.com/matlabcentral/answers/285594-convert-complex-double-to-double-type. [Accessed: 15-Aug-2022].

[5] "How can I get the Max of the values of an array with numeric type double," *mathworks*, Aug-2014. Available: https://www.mathworks.com/matlabcentral/answers/151559-how-can-i-get-the-max-of-the-values-of-an-array-with-numeric-type-double. [Accessed: 15-Aug-2022].

[6] "Displaying numbers in Gui," *mathworks*, Oct-2013. Available: https://www.mathworks.com/matlabcentral/answers/103988-displaying-numbers-in-gui. [Accessed: 15-Aug-2022].

[7] "How to get the value of radio buttons from app designer," *mathworks*, Jun-2019. Available: https://www.mathworks.com/matlabcentral/answers/468609-how-to-get-the-value-of-radio-buttons-from-app-designer. [Accessed: 15-Aug-2022].

[8] "Define Custom Event Data," *mathworks*. Available: https://www.mathworks.com/help/matlab/matlab_oop/class-with-custom-event-data.html. [Accessed: 15-Aug-2022].

[9] “App button animation & truecolor images,” *mathworks*, Apr-2020. Available: https://www.mathworks.com/matlabcentral/discussions/highlights/132277-new-in-r2020a-app-button-animation-truecolor-images. [Accessed: 15-Aug-2022].

[10] “Install MATLAB Engine API for Python,” *mathworks*. Available: https://www.mathworks.com/help/matlab/matlab_external/install-the-matlab-engine-for-python.html. [Accessed: 15-Aug-2022].

[11] “Heart-Rate-Monitoring-System,” *GitHub*, Apr-2018.. Available: https://github.com/samyakag/Heart-Rate-Monitoring-System. [Accessed: 15-Aug-2022].

[12] “MATLAB GUI function - progress bar-waitbar” *YouTube*, 20-Apr-2013.Available: https://www.youtube.com/watch?v=u4DQ5-KNASw&ab_channel=MYFunNyBosS. [Accessed: 15-Aug-2022].

[13] “simple calculator (Newton's 2nd law),” *YouTube*, 20-Aug-2019. Available: https://www.youtube.com/watch?v=5nccNUE09o4&t=358s&ab_channel=BenitoSebastian. [Accessed: 15-Aug-2022].

[14] “FFT in Matlab,” *YouTube*, 28-Apr-2022. Available: https://www.youtube.com/watch?v=XEbV7WfoOSE&ab_channel=MATLAB. [Accessed: 15-Aug-2022].

[15] “GUI text output - advanced techniques,” *YouTube*, 18-Feb-2016.Available: https://www.youtube.com/watch?v=HBBAlHdZoBI&ab_channel=DoctorBear. [Accessed: 15-Aug-2022].