Feel free to do one of the following tasks as your final project. I personally prefer the engineering task although it needs to have access to a function generator. In my view, you become familiar with many practical points while doing the engineering task.

ENGINEERING TASK

Task 1

Make a simple oscilloscope by feeding a signal to your laptop via its microphone jack and displaying the captured signal on its monitor. Note that I personally do not take any responsibility for any possible damage to your laptop!

(a) Write a MATLAB/Python code to display the captured signal online. Feed your oscilloscope with sine, triangle, and square waves and observe the results.

```
function simple_oscilloscope()
    % Recording settings
    fs = 44100; % Sampling frequency
    nBits = 16; % Number of bits per sample
    nChannels = 1; % Number of channels (1 for mono, 2 for stereo)
    duration = 1; % Recording duration in seconds

    % Create an audiorecorder object
    recObj = audiorecorder(fs, nBits, nChannels);

    % Create a graphical user interface (GUI)
    hFig = figure('Name', 'Simple Oscilloscope', 'NumberTitle', 'off', ...
                  'CloseRequestFcn', @closeFigure, 'Position', [100, 100, 800,
                      600]);
    hAxes = axes('Parent', hFig, 'Position', [0.1, 0.3, 0.8, 0.6]);
    hLine = plot(hAxes, NaN, NaN);
    xlabel(hAxes, 'Time (seconds)');
    ylabel(hAxes, 'Amplitude');
    grid(hAxes, 'on');

    % GUI controls for displaying frequency and amplitude
    uicontrol('Style', 'text', 'String', 'Frequency (Hz):', 'Position', [20 100
        100 20]);
    hFreq = uicontrol('Style', 'text', 'String', '0', 'Position', [120 100 100
        20]);
    uicontrol('Style', 'text', 'String', 'Amplitude:', 'Position', [20 70 100
        20]);
    hAmp = uicontrol('Style', 'text', 'String', '0', 'Position', [120 70 100
        20]);

    % Controls for changing time axis scale
    uicontrol('Style', 'text', 'String', 'Time Scale (seconds):', 'Position',
        [20 40 100 20]);
    hScale = uicontrol('Style', 'edit', 'String', num2str(duration), 'Position',
         [120 40 100 20]);
    uicontrol('Style', 'pushbutton', 'String', 'Apply', 'Position', [230 40 50
        20], 'Callback', @updateScale);
```

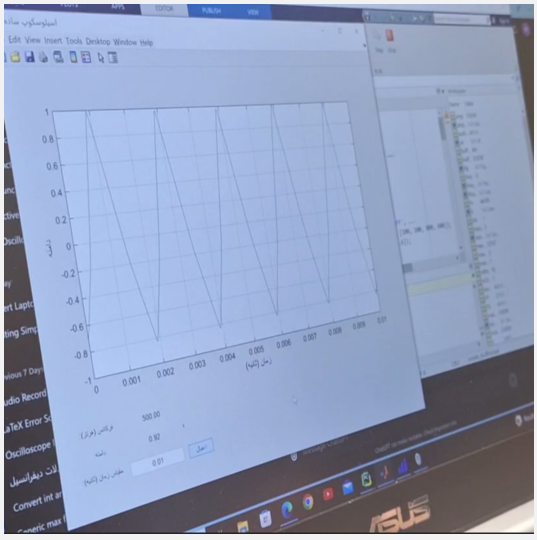
Task 1 continued on next page...

```
30
31      % Continuous recording and display of signal
32      while ishandle(hFig)
33          recordblocking(recObj, duration);
34          y = getaudiodata(recObj);
35
36          % Normalize the signal
37          y = y / max(abs(y));
38
39          t = linspace(0, duration, length(y));
40
41          % Update the plot
42          set(hLine, 'XData', t, 'YData', y);
43          drawnow;
44
45          % Calculate and display frequency and amplitude
46          Y = fft(y);
47          f = (0:length(Y)-1)*fs/length(Y);
48          [~, idx] = max(abs(Y));
49          freq = f(idx);
50
51          % Calculate peak-to-peak amplitude
52          amp = (max(y) - min(y)) / 2;
53
54          set(hFreq, 'String', num2str(freq, '%.2f'));
55          set(hAmp, 'String', num2str(amp, '%.2f'));
56      end
57
58      function closeFigure(~, ~)
59          stop(recObj);
60          delete(hFig);
61      end
62
63      function updateScale(~, ~)
64          newScale = str2double(get(hScale, 'String'));
65          if isnan(newScale) || newScale <= 0
66              errordlg('Time scale must be a positive number.', 'Error');
67          else
68              duration = newScale;
69          end
70      end
71  end
```

The signal generated by the function generator:

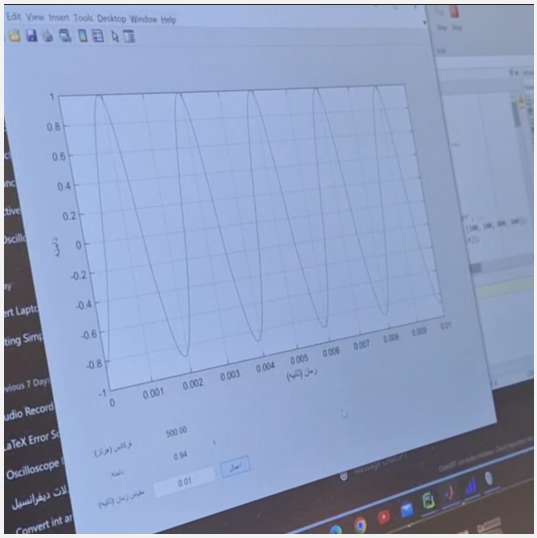
Task 1 continued on next page...

Sinusoidal signal:

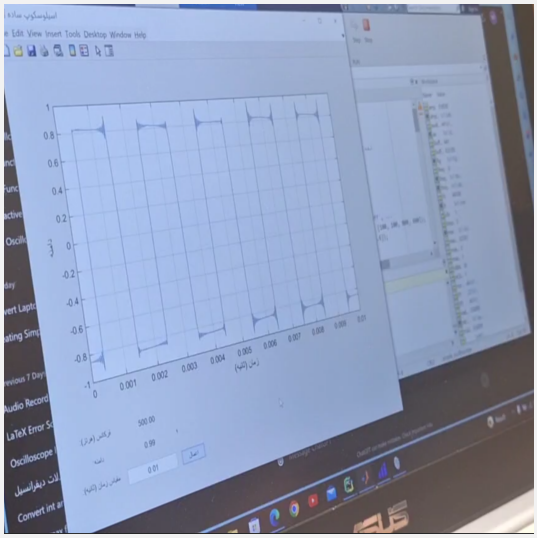


Triangular signal:

Task 1 continued on next page…

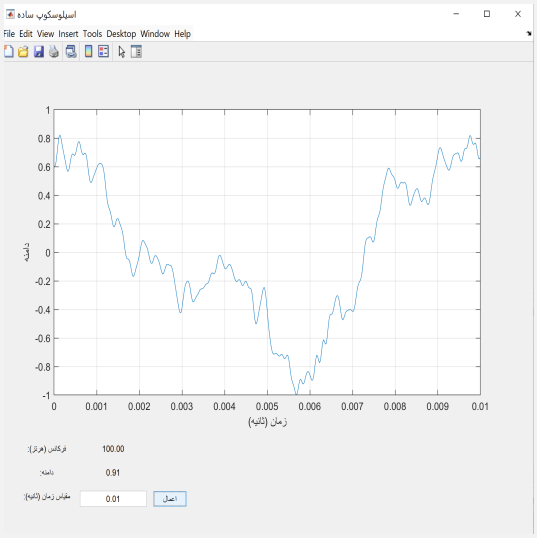


And finally, the square signal:



For the received signal through the laptop microphone.

Task 1 continued on next page...



(b) Is there any limitation on the maximum amplitude and frequency of your oscilloscope?

Yes, the oscilloscope used has limitations on the maximum amplitude and frequency.

Maximum Amplitude Limitations Input Voltage Range: The oscilloscope is simulated using a 3.5mm jack input connected to a sound card. The typical voltage range for a sound card input is between ±1V to ±2V. If the signal amplitude exceeds this range, the sound card and oscilloscope might be damaged or the signal might not be displayed correctly.

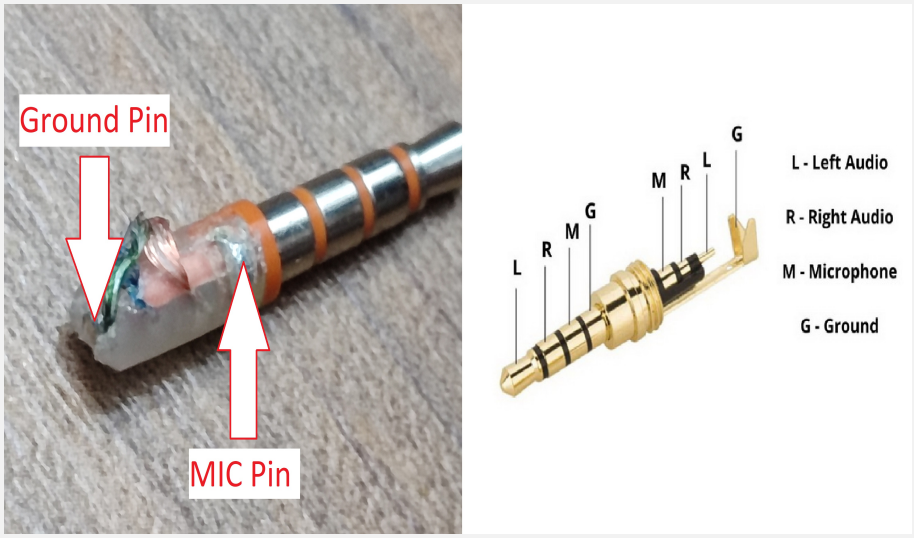
Maximum Frequency Limitations Sampling Rate: The sampling rate of this oscilloscope is 44.1 kHz, which according to the Nyquist theorem, limits the maximum measurable frequency to 22.05 kHz. Signals with frequencies higher than this will not be measured accurately and will experience aliasing effects.

(c) Prepare a short report and describe your work concisely. Use suitable figures or equations to better describe different parts of your code and to make your report more readable and understandable. Take a short video of yourself demonstrating the performance of your oscilloscope.

"The video has been sent along with the LaTeX file."
To display sinusoidal, triangular, and square wave signals (or any signal injected into the laptop's sound card) on a laptop, follow these steps:

First, connect the 3.5mm headphone jack to the laptop's 3.5mm jack as shown in the image below:

Task 1 continued on next page...



As depicted in the image, connect one end of the received signal (positive end) to the microphone pin, and connect the other end of the received signal (negative end) to the headphone ground pin.

Finally, by running the MATLAB code and ensuring specific settings in the sound section of the control panel (Noise Suppression must be disabled for accurate signal display), we can display the received signal from the function generator.
And now, some explanations about the MATLAB code:

This MATLAB code simulates a simple oscilloscope to display audio signals received through the 3.5mm jack input of a laptop. The code sets up recording parameters such as sampling rate (44.1 kHz), bit depth (16 bits), and the number of channels (mono). Then, it creates a graphical user interface (GUI) with a plot to display the signal and controls to show frequency, amplitude, and adjust the time scale. In an infinite loop, the audio signal is recorded, normalized, and displayed on the plot. To calculate the frequency, a Fast Fourier Transform (FFT) is applied to the signal, and the dominant frequency is extracted. The peak-to-peak amplitude is also calculated and displayed.

(d) ***Bonus!*** *Create a GUI for your oscilloscope such that useful information like frequency, amplitude, and so on are reported alongside the captured wave.*

As shown in the image below, a graphical interface has been designed that creates a window named "Simple Oscilloscope." This window includes two axes, vertical and horizontal, which display voltage and time, respectively. Additionally, two labels have been created to show the voltage and frequency values, which are updated in real-time.
For adjusting the horizontal time axis, the time scale value is taken as input from the user, which is set to 0.01 seconds in the image above.
Further details can be seen in the provided MATLAB code.

(e) ***Bonus!*** *Write your report in LaTeX.*