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 an oscilloscope. In my view, you become familiar with many practical points while doing the engineering task.

ENGINEERING TASK

Task 1

Stereo audio has two audio signals designed for two separate audio channels, which creates a perception of space. If you have a stereo headphone, each audio signal is played separately on a speaker. The two audio signals can be injected to an oscilloscope in xy mode to create a Lissajous curve.

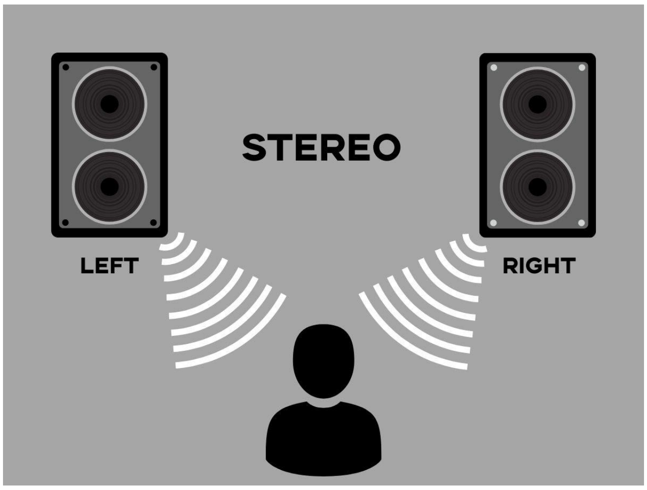

Figure 1: Stereo audio on speakers.

(a) Write a MATLAB/Python code to create suitable stereo audio such that the corresponding Lissajous curve on an oscilloscope looks like a heart shape. Connect the audio output of your laptop using a stereo audio jack to the channels of an oscilloscope and see the heart-shaped Lissajous curve.

If we want to use python, we must use "sounddevice", "numpy" and "matplotlib.pyplot" library. In case of making heart once the program is executed, it will generate a sine wave for X axis and a combination of four cosine waves for Y axis with a frequency of 44100 Hz and duration of 10 seconds. The signal will be outputted through the specified audio output device, which should be connected to your oscilloscope via the AUX cable.

Heart:

```python
import numpy as np
import sounddevice as sd
```

Task 1 continued on next page. . .

```
import matplotlib.pyplot as plt

duration = 10
sampling_freq = 44100
t = np.linspace(0, duration, int(duration * sampling_freq), endpoint=False)

"""
        --------Heart--------
"""

signalX = 16*(np.sin(2 * np.pi * 1000 * t)**3)
signal1 = 13*np.cos(2 * np.pi * 1000 * t)
signal2 = -5*np.cos(2 * np.pi * 2000 * t)
signal3 = -2*np.cos(2 * np.pi * 3000 * t)
signal4 = -1*np.cos(2 * np.pi * 4000 * t)
signalY = signal1 + signal2 + signal3 + signal4
signalY /= np.max(np.abs(signalY))
signalX /= np.max(np.abs(signalX))

stereo_signal = np.column_stack(( signalX , signalY))

sd.play(stereo_signal, samplerate=sampling_freq, blocking=True)
```

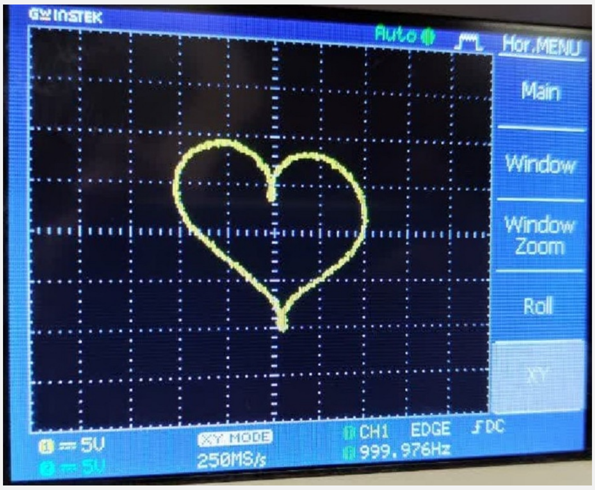

Figure 2: Heart in xy mode of oscilloscope.

(b) Extend your code such that common geometric shapes like circle, ellipse, square, and so on appear as a Lissajous curve on the oscilloscope.

In case of making circle once the program is executed, it will generate a cosine wave for X axis and sine waves for Y axis with a frequency of 44100 Hz and duration of 10 seconds.

Circle:

```
import numpy as np
import sounddevice as sd
import matplotlib.pyplot as plt

duration = 10
sampling_freq = 44100
t = np.linspace(0, duration, int(duration * sampling_freq), endpoint=False)

"""
        --------Circle--------
```

Task 1 continued on next page. . .

```
"""
signalX = np.cos(2 * np.pi * 20000 * t);
signalY = np.sin(2 * np.pi * 20000 * t);


stereo_signal = np.column_stack(( signalX , signalY))

sd.play(stereo_signal, samplerate=sampling_freq, blocking=True)
```

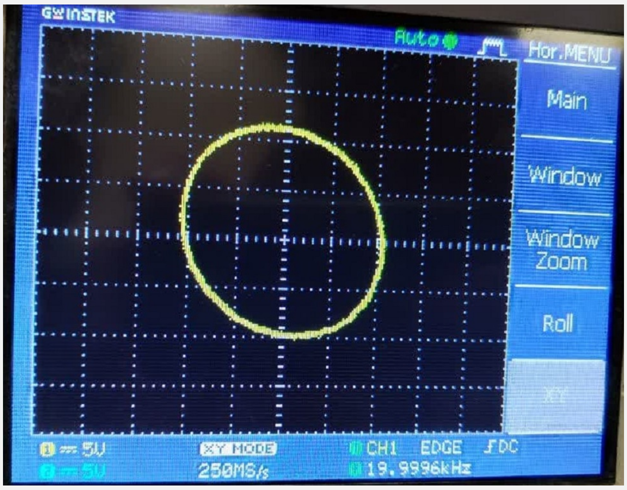

Figure 3: Circle in xy mode of oscilloscope.

In case of making Ellipse once the program is executed, it will generate a cosine wave for X axis and sine waves for Y axis with a frequency of 44100 Hz and duration of 10 seconds.
Ellipse:

```
import numpy as np
import sounddevice as sd
import matplotlib.pyplot as plt

duration = 10
sampling_freq = 44100
t = np.linspace(0, duration, int(duration * sampling_freq), endpoint=False)

"""
        --------Elipse--------
"""
signalX = np.cos(2 * np.pi * 20000 * t);
signalY = .3*np.sin(2 * np.pi * 20000 * t);



stereo_signal = np.column_stack(( signalX , signalY))

sd.play(stereo_signal, samplerate=sampling_freq, blocking=True)
```

Task 1 continued on next page...

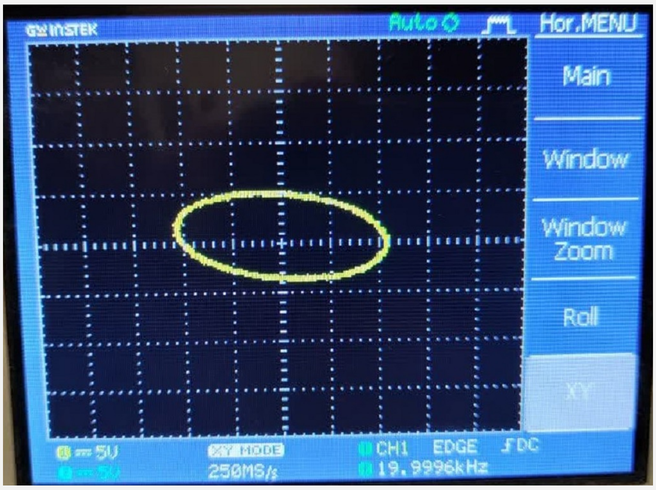

Figure 4: Ellipse in xy mode of oscilloscope.

Square:

```
cycle_duration = .005
duration = 10
sampling_freq = 44100

elif(self.arg1=="square"):
            signalY = np.sign(np.sin(2 * np.pi * 500 * t))
            signalX = np.sign(np.sin(2 * np.pi * 500 * t + np.pi/2))
```

Diamond:

```
import numpy as np
import sounddevice as sd
import matplotlib.pyplot as plt

duration = 10
sampling_freq = 44100
t = np.linspace(0, duration, int(duration * sampling_freq), endpoint=False)


"""
        --------Diamond--------
"""
cycle_duration = .001
amplitude = 1.0
signalY = np.zeros_like(t)
signalX = np.zeros_like(t)
num_samples = int(duration * sampling_freq)
for i in range(num_samples):
    time_in_cycle = t[i] % cycle_duration
    if time_in_cycle < cycle_duration / 4:
        signalY[i] = 1
    elif time_in_cycle < 2*cycle_duration / 4:
        signalY[i] = 0
    elif time_in_cycle < 3*cycle_duration / 4:
        signalY[i] = -1
    else :
        signalY[i] = 0
for i in range(num_samples):
    time_in_cycle = t[i] % cycle_duration
    if time_in_cycle < cycle_duration / 4:
        signalX[i] = 0
```

Task 1 continued on next page. . .

```
    elif time_in_cycle < 2*cycle_duration / 4:
        signalX[i] = -1
    elif time_in_cycle < 3*cycle_duration / 4:
        signalX[i] = 0
    else :
        signalX[i] = 1


stereo_signal = np.column_stack(( signalX , signalY))

sd.play(stereo_signal, samplerate=sampling_freq, blocking=True)
```

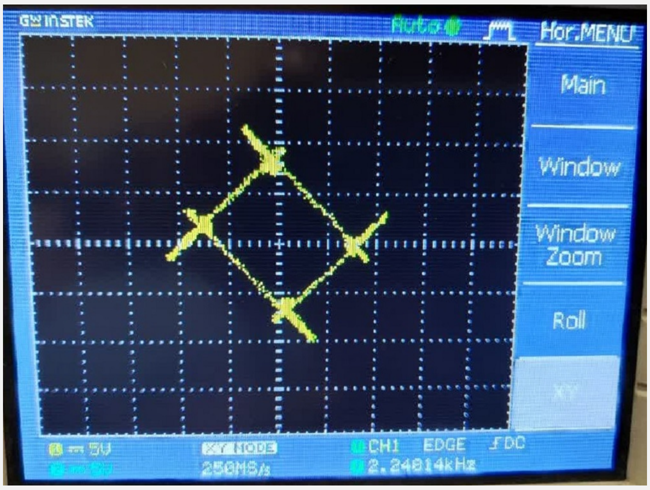

Figure 5: Diamond in xy mode of oscilloscope.

Animated Circle:

```

cycle_duration = .005
duration = 10
sampling_freq = 44100

elif(self.arg1=="animatedCircle"):
            signalX = np.exp(-t)* np.cos(2 * np.pi * 20000 * t);
            signalY = np.exp(-t)* np.sin(2 * np.pi * 20000 * t);
```

(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 creation of the desired Lissajous curves.

*(d) **Bonus!** Create a GUI for your code such that the desired curve is taken as a two-dimensional function $f(x, y) = 0$ and its corresponding Lissajous curve appears on the oscilloscope.*

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