import tkinter as tk

from tkinter import Frame, Button, Canvas

import matplotlib

matplotlib.use('TkAgg')

import numpy as np

import matplotlib.pyplot as plt

from math import pi

import math

from matplotlib.backends.backend\_tkagg import FigureCanvasTkAgg

#from matplotlib.backends.backend\_tkagg import ( FigureCanvasTkAgg, NavigationToolbar2Tk)

from matplotlib.figure import Figure

from matplotlib.lines import Line2D

import matplotlib.animation as animation

import pylab

from scipy.signal import butter, lfilter

from scipy.signal import freqz

from scipy.ndimage import gaussian\_filter1d

import datetime as dt

import random

#from pylab import \*

#lowcut = 40.0

#highcut = 70.0

#o = 3

list1=[]

def butter\_bandpass(lowcut, highcut, fs, order=5):

    nyq = 0.5 \* fs

    low = lowcut / nyq

    high = highcut / nyq

    b, a = butter(order, [low, high], btype='band')

    return b, a

def butter\_bandpass\_filter(data, lowcut, highcut, fs, order=5):

    b, a = butter\_bandpass(lowcut, highcut, fs, order=order)

    y = lfilter(b, a, data)

    return y

def butter\_lowpass(lowcut, fs, order=5):

    nyq = 0.5 \* fs

    low = lowcut / nyq

    b, a = butter(order, [low], btype='low')

    return b, a

def butter\_lowpass\_filter(data, lowcut, fs, order=5):

    b, a = butter\_lowpass(lowcut, fs, order = order)

    y = lfilter(b, a, data)

    return y

#fig = plt.Figure()

def plot ():

    input\_volt=int(input\_volt\_amplitude.get())#----------input from amplitude

    fs=int(sampling.get())

    #range2=440

    #input\_volt=230#-------------input from amplitude

    type\_of\_measurement = measurement\_choices.get()

    range2 = choices.get()

    print("Type of measurement : ",type\_of\_measurement)

    print("Range is : ",range2)

    if type\_of\_measurement == "Voltage - AC":

        f=int(signal.get())

        lowcut = 40.0

        highcut = 70.0

        o = 3

        if input\_volt<range2:

            print("Signal Frequency is ",f)

            print("Input Voltage(Amplitude) is ",input\_volt)

            print("Sampling Frequency is ",fs)

            #print("Range is ",range2)

            #Parameters

            #input\_volt=230#-------------input from amplitude

            amp = 1.414\*input\_volt       #          (Amplitude)

            #f = 50        #      (Frequency)---------------signal frequency

            #fs = 5000    #     (Sample Rate)-----------------sampling freq

            T = 1/f

            Ts = 1/fs

            harmonic\_amp1=0.05\*amp

            harmonic\_amp2=0.05\*amp

            #attenuation\_factor=(amp+harmonic\_amp1+harmonic\_amp2)/2.5

            attenuation\_factor= (range2 + 0.5\*range2) /2.5

            #attenuation\_factor\_value\_label['text'] = attenuation\_factor

            attenuation.set(attenuation\_factor)

            continous  = True

            #f = Figure(figsize=(6,4), dpi=100)

            fig = plt.Figure()

            x = np.arange(fs)

            #print(x)

            yout= [ ((amp\*np.sin(2\*np.pi\*f \* (i/fs)))+(0.05\*amp\*np.sin(6\*pi\*f \* (i/fs)))+(0.05\*amp\*np.sin(12\*pi\*f \* (i/fs)))) for i in x ]

            yo= [ ((amp\*np.sin(2\*np.pi\*f \* (i/fs)))+(0.05\*amp\*np.sin(6\*pi\*f \* (i/fs)))+(0.05\*amp\*np.sin(12\*pi\*f \* (i/fs))))\*(1/attenuation\_factor) for i in x ]

                    #Add If condition for AC selection of configurator

            adc=[((yo[i]/5)\*65536)+(32768) for i in x]

            for i in x:

                #print((y[i]))

                op1 = 32768 + butter\_bandpass\_filter(adc, lowcut, highcut, fs, order=o)

                list1 = op1.tolist()

                #print(type(list1))

                #op2 = gaussian\_filter1d(op1, 4)

                #print(math.ceil(op2[i]))

                #adc=int((y[i]/5)\*65536)+(32768)

                #print(adc)

                    #End of IF condition

                    #Add ELSE IF condition for DC selection of configurator

                    #End ELSE IF condition

            #adc=((y/10)\*65536)+(32768)

            print(list1)

            print(type(op1))

            class aScope(object):

                def \_\_init\_\_(self, ax, maxt=2\*T, dt=Ts):

                    self.ax = ax

                    self.dt = dt

                    self.maxt = maxt

                    self.tdata = [0]

                    self.ydata = [0]

                    self.line = Line2D(self.tdata, self.ydata)

                    self.ax.add\_line(self.line)

                    self.ax.set\_ylim(-350,350)

                    self.ax.set\_xlim(0, self.maxt)

                def aupdate(self, y):

                    lastt = self.tdata[-1]

                    if continous :

                        if lastt > self.tdata[0] + self.maxt:

                            self.ax.set\_xlim(lastt-self.maxt, lastt)

                    t = self.tdata[-1] + self.dt

                    self.tdata.append(t)

                    self.ydata.append(y)

                    self.line.set\_data(self.tdata, self.ydata)

                    return self.line,

            def sineEmitter():

                for i in x:

                    yield math.ceil(yout[i])

            #fig, ax = plt.subplots()

            #f = Figure(figsize=(6,4), dpi=100)

            fig = plt.Figure(figsize=(6,4))

            canvas = FigureCanvasTkAgg(fig, master=root)

            canvas.get\_tk\_widget().place(x=0, y=190)

            ax1 = fig.add\_subplot(111)

            ax1.set\_xlabel("Time")

            ax1.set\_ylabel("Amplitude")

            line, = ax1.plot(x, np.sin(x))

            scope = aScope(ax1)

            # pass a generator in "sineEmitter" to produce data for the update func

            plot.ani = animation.FuncAnimation(fig, scope.aupdate, sineEmitter, interval=10,blit=True)

        else:

            #error messege

            tk.messagebox.showerror("Error", "Exceeds the range")

            tk.messagebox.showinfo("Hint","The Input Voltage(Amplitude) is always lesser than range value.")

    elif type\_of\_measurement == "Voltage - DC":

        #fs = 5000

        lowcut = 25

        o = 3

        N = 40000

        N2 = 400

        Ts=1/fs

        if input\_volt<range2:

            print("Input voltage(Amplitude) is ",input\_volt)

            print("Sampling Frequency is ",fs)

            #print("Range is ",range2)

            continous  = True

            #time = np.arange(0\*np.pi, 5\*np.pi, 0.01)

            #amplitude1 = 100\*np.sin(time)

            #deactivate the signal frequency

            input\_volt=int(input\_volt\_amplitude.get())#----------input from amplitude

            #ip = 100 + ((3\*np.random.randn(N)))

            attenuation\_factor = range2/5

            attenuation.set(attenuation\_factor)

            #fig = plt.figure()

            #ax = fig.add\_subplot(1,1,1)

            #fs = 500

            x = []

            #input\_volt = 50

            N = fs

            y = []

            def animate(i,x,y):

                x.append(i)

                y.append(input\_volt + ((0.01\*input\_volt\*random.randint(0,10))))

                i+=1

                #x=x[-50:500]

                #y=y[-50:500]

                x=x[:fs]

                y=y[:fs]

                ax1.clear()

                ax1.plot(x,y)

                plt.xticks(rotation=45, ha='right')

            fig = plt.Figure(figsize=(6,4))

            canvas = FigureCanvasTkAgg(fig, master=root)

            canvas.get\_tk\_widget().place(x=0, y=190)

            plot.ani = animation.FuncAnimation(fig, animate, fargs=(x,y),interval=1)

            ax1 = fig.add\_subplot(111)

            ax1.set\_xlabel("Time")

            ax1.set\_ylabel("Amplitude")

            line, = ax1.plot(x, np.sin(x))

        else:

            #error messege

            tk.messagebox.showerror("Error", "Exceeds the range")

            tk.messagebox.showinfo("Hint","The Amplitude value is always lesser than range value.")

def reset():

    signal.set(0)

    input\_volt\_amplitude.set(0)

    sampling.set(0)

    measurement\_choices.set("Voltage - AC")

    choices.set(0)

    attenuation.set(0)

    #canvas.delete('all')

    #animation.destroy()

def on\_option\_change(event):

    selected = measurement\_choices.get()

    attenuation.set(0)

    if selected == "Voltage - AC":

        input\_range = {110,230,440,500}

    elif selected == "Voltage - DC":

        input\_range = {10,24,48}

    input\_range=sorted(input\_range)

    range\_label= tk.Label(root,text="",bg='White',fg='black',font='Helvetica 8 bold')

    range\_option = tk.OptionMenu(root,choices, \*input\_range)

    choices.set(0)

    range\_option.config(bg = "LightYellow2")

    range\_option.configure(width=15)

    if selected == "Voltage - AC":

        signal.set(0)

        input\_volt\_amplitude.set(0)

        sampling.set(0)

        range\_label['text'] = "Range"

        range\_label.place(x=300, y=10)

        #choices.set(0)

        range\_option.configure(width=15)

        range\_option.place(x=260, y=35)

        #signal\_freq\_label.place(x=30, y=100)

        #signal\_freq\_entry.place(x=30, y=130)

        signal\_freq\_label.place(x=430, y=100)

        signal\_freq\_entry.place(x=430, y=130)

    elif selected == "Voltage - DC":

        signal.set(0)

        input\_volt\_amplitude.set(0)

        sampling.set(0)

        range\_label['text'] = "Range"

        range\_label.place(x=300, y=10)

        #choices.set(0)

        range\_option.configure(width=15)

        range\_option.place(x=260, y=35)

        signal\_freq\_label.place\_forget()

        signal\_freq\_entry.place\_forget()

#Execution starts here

root = tk.Tk()

root.title("Configurator")

title\_bar = Frame(root, bg='brown', relief='raised', bd=2)

title\_bar.pack()

root.geometry("600x700")

root.configure(bg='#FFFFFF')

#Variable data types

measurement\_choices = tk.StringVar()

ac\_choices = tk.IntVar()

dc\_choices = tk.IntVar()

choices=tk.IntVar()

#Assigning values

measurement\_type = {"Voltage - AC", "Voltage - DC"}

#Label - Type of Measurement

type\_label=tk.Label(root,text="Type of Measurement",bg='White',fg='black',font='Helvetica 8 bold')

type\_label.place(x=50, y=10)

measurement\_choices.set("Voltage - AC")

#Option Menu to select Measurement type

measur\_type = tk.OptionMenu(root,measurement\_choices, \*measurement\_type, command=on\_option\_change)

measur\_type.config(bg = "LightYellow2")

measur\_type.configure(width=15)

measur\_type.place(x=50, y=35)

#Signal Frequency

signal\_freq\_label = tk.Label(text="Signal Frequency",bg='White',fg='black',font='Helvetica 8 bold')

#signal\_freq\_label.place(x=30, y=100)

signal\_freq\_label.place(x=430, y=100)

signal = tk.DoubleVar()

signal\_freq\_entry = tk.Entry(root, width = 15, textvariable = signal)

#signal\_freq\_entry.place(x=30, y=130)

signal\_freq\_entry.place(x=430, y=130)

#Amplitude

amplitude\_label = tk.Label(text="Input Voltage",bg='White',fg='black',font='Helvetica 8 bold')

amplitude\_label.place(x=160, y=100)

input\_volt\_amplitude = tk.DoubleVar()

amplitude\_entry = tk.Entry(root, width = 15, textvariable = input\_volt\_amplitude)

amplitude\_entry.place(x=160, y=130)

#Sampling Frequency

sampling\_freq\_label = tk.Label(text="Sampling Frequency",bg='White',fg='black',font='Helvetica 8 bold')

sampling\_freq\_label.place(x=290, y=100)

sampling = tk.DoubleVar()

sampling\_freq\_entry = tk.Entry(root, width = 15, textvariable = sampling)

sampling\_freq\_entry.place(x=290, y=130)

#Attenuation Factor

attenuation\_factor\_label = tk.Label(text="Attenuation Factor",bg='White',fg='black',font='Helvetica 8 bold',width=15)

#attenuation\_factor\_label.place(x=430, y=100)

attenuation\_factor\_label.place(x=30, y=100)

attenuation = tk.DoubleVar()

attenuation\_factor\_value\_label = tk.Label(text="Attenuation Value",bg='DarkSeaGreen3',fg='black',font='Helvetica 8 bold',width=15, textvariable = attenuation)

#attenuation\_factor\_value\_label.place(x=430, y=130)

attenuation\_factor\_value\_label.place(x=30, y=130)

#Draw button

graph\_button = tk.Button(root, text="Execute",bg='brown',fg='white', command=plot)

graph\_button.place(x=45, y=160)

#Reset button

#reset\_button = tk.Button(root,text="Reset", bg='brown',fg='white', command=reset)

#reset\_button.place(x=250, y=160)

root.mainloop()