**Source Code (Asha N)**

Highlighted portion in the below script(Client code) is the implemented by Asha N

import socket

import numpy as np

import matplotlib.pyplot as plt

from matplotlib.lines import Line2D

import matplotlib.animation as animation

from matplotlib.backends.backend\_tkagg import FigureCanvasTkAgg

import math

from math import pi

from scipy.ndimage import gaussian\_filter1d

import pickle

import tkinter as tk

from tkinter import Frame, Button, Canvas

from scipy.signal import butter, lfilter

from scipy.signal import freqz

from numpy.compat.py3k import long

HOST = '127.0.0.1'

PORT = 5032

HEADERSIZE = 10

lis= []

lis2=[]

Vaverage=0

fs=1000

f=50

T = 1/f

Ts = 1/fs

vrms\_list=[]

with socket.socket(socket.AF\_INET, socket.SOCK\_STREAM) as s:

    s.connect((HOST, PORT))

    data = s.recv(640016)

    data = data.decode('utf-8')

    data = eval(data)

    type\_of\_measurement = data[0]

    Range = int(data[1])

    input\_volt=float(data[2])

    attenuation\_factor=float(data[3])

    print(f"type\_of\_measurement:{type\_of\_measurement}")

    print(f"Range:{Range}")

    print(f"attenuation\_factor:{attenuation\_factor}")

    print(f"input\_volt:{input\_volt}")

    del data[0:4]

    for i in data :

        if not i:

            break

        j=(i[0]<<8)+i[1]

        lis.append(j)

    opc = []

    arr = np.array(lis)

    iterDC = 0

    iterAC = 0

    maxval = len(arr)

    x = np.arange(maxval, step = 1)

def plot():

    iterDC = 0

    iterAC = 0

    maxval = len(arr)

    x = np.arange(maxval, step = 1)

    if (type\_of\_measurement == 'Voltage - DC'):

        while(iterDC < maxval ):

            opc.append((lis[iterDC]/65536)\*Range)

            iterDC = iterDC+1

            Vaverage=round(sum(opc)/len(opc),2)

    elif (type\_of\_measurement == 'Voltage - AC'):

        while(iterAC < maxval ):

            opc.append(((lis[iterAC]-32768)/65536)\*(Range)\*2.5\*1.2)

            iterAC = iterAC+1

    InVolt\_label\_value1.set(input\_volt)

    range\_label\_value1.set(Range)

    mt1['text']=type\_of\_measurement

    u1['text']="Volts"

    attenuation\_value.set(attenuation\_factor)

    comp = 50               #number of samples considered for RMS calculation. Shall ATLEAST be the input frequency of wave(f) value for proper working

    w = comp                #it is the maximum value "Upto" for said iteration

    v = 0

    if (type\_of\_measurement == 'Voltage - AC'):            #it is the minimum value "From" for said iteration

        def rmsValue(arr, v,w): #the function which runs the iterations. takes input from the "opc" list which is the output variable for outputting the graph

            square = 0          #variable to store the sum of all squares

            mean = 0.0          #mean value of all the squares

            root = 0.0          #return value of the RMS for said iteration

            for i in range(v,w):    #loop to do the iteration for RMS value

                square += (arr[i]\*\*2)       #squaring and summation of all values

            mean = (square / (float)(w-v))  #mean(average) of all the summed squares

            root = math.sqrt(mean)          #square root of the mean of the summed squares

            return root                     #function returning RMS value for said iteration

        while(w < fs):                      #while loop to make the iterator run through the entire size of the input dataset

            print(rmsValue(opc,v, w))       #calling the above function for the said iterating values...TAKE THIS VALUE AND UPDATE REALTIME ON THE OUTPUT DISPLAY GUI AS VOLTAGE RMS OUTPUT

            vrms\_list.append(rmsValue(opc,v, w))

            v = w                           #giving the final value of previous iteration to initial value of next iteration

            w = w + comp                    #incrementing the final value of the iteration to the next maximum value.

        vrms= vrms\_list[18]

        vrms\_value1.set(round(vrms))

        vmax= 1.414\*vrms

        vmax\_value1.set(round(vmax))

        print("Vrms valuse is:",round(vrms))

        print("Vmax valuse is:",round(vmax))

        continous  = True

        fig1 = plt.Figure()

        x = np.arange(1000)

        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(opc[i])

        fig1 = plt.Figure(figsize=(6,4))

        canvas = FigureCanvasTkAgg(fig1, master=root)

        canvas.get\_tk\_widget().place(x=0, y=280)

        ax1 = fig1.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(fig1, scope.aupdate, sineEmitter, interval=10,blit=True)

    elif (type\_of\_measurement == 'Voltage - DC'):

        vavg\_value1.set(Vaverage)

        print("Vavg value is:",Vaverage)

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

        canvas = FigureCanvasTkAgg(fig, master=root)

        canvas.get\_tk\_widget().place(x=0, y=260)

        ax1 = fig.add\_subplot(111)

        ax1.set\_xlabel("Time")

        ax1.set\_ylabel("Amplitude")

        ax1.plot(opc)

        plt.show()

        canvas.draw()

def reset():

    InVolt\_label\_value1.set(0)

    u1['text']=" "

    mt1['text']=" "

    range\_label\_value1.set(0)

    attenuation\_value.set(0)

def exitWindow():

    root.destroy()

#Execution starts here

root = tk.Tk()

root.title("Display")

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

title\_bar.pack()

root.geometry("600x800")

root.configure(bg='#FFFFFF')

#Variable data types

InVolt\_label\_value1 = tk.DoubleVar()

unit\_label\_value1 = tk.StringVar()

mtype\_label\_value1 = tk.StringVar()

range\_label\_value1 = tk.DoubleVar()

attenuation\_value = tk.DoubleVar()

vrms\_value1=tk.DoubleVar()

vmax\_value1=tk.DoubleVar()

vavg\_value1=tk.DoubleVar()

#Measurement Calculation

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

iv.place(x=55, y=20)

InVolt\_label\_value = tk.Label(text="",bg='DarkSeaGreen3',fg='black',font='Helvetica 8 bold',width=15, textvariable = InVolt\_label\_value1)

InVolt\_label\_value.place(x=50, y=40)

#Unit

u = tk.Label(text="Unit",bg='White',fg='black',font='Helvetica 8 bold')

u.place(x=390,y=20)

u1 = tk.Label(text="",bg='DarkSeaGreen3',fg='black',font='Helvetica 8 bold',width=15)

u1.place(x=360, y=40)

#Vrms

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

vrms\_value = tk.Label(text="",bg='DarkSeaGreen3',fg='black',font='Helvetica 8 bold',width=15, textvariable = vrms\_value1)

#Vmax

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

vmax\_value = tk.Label(text="",bg='DarkSeaGreen3',fg='black',font='Helvetica 8 bold',width=15, textvariable = vmax\_value1)

#Vavg

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

vavg\_value = tk.Label(text="",bg='DarkSeaGreen3',fg='black',font='Helvetica 8 bold',width=15, textvariable = vavg\_value1)

selected = type\_of\_measurement

if selected ==  "Voltage - AC":

    #Vrms

    vrms\_label.place(x=50,y=160)

    vrms\_value.place(x=50, y=180)

    #Vmax

    vmax\_label.place(x=220,y=160)

    vmax\_value.place(x=200, y=180)

    #vavg forget

    vavg\_label.place\_forget()

    vavg\_value.place\_forget()

if selected == "Voltage - DC":

    #Vavg

    vavg\_label.place(x=50,y=160)

    vavg\_value.place(x=50, y=180)

    #Vrms forget

    vrms\_label.place\_forget()

    vrms\_value.place\_forget()

    #Vmax forget

    vmax\_label.place\_forget()

    vmax\_value.place\_forget()

#Measurement Type

mt = tk.Label(text="Measurement Type",bg='White',fg='black',font='Helvetica 8 bold',width=15)

mt.place(x=50, y=90)

mt1 = tk.Label(text="",bg='DarkSeaGreen3',fg='black',font='Helvetica 8 bold',width=15)

mt1.place(x=50, y=120)

#Range

r = tk.Label(text="Range",bg='White',fg='black',font='Helvetica 8 bold',width=15)

r.place(x=365, y=90)

r1 = tk.Label(text="",bg='DarkSeaGreen3',fg='black',font='Helvetica 8 bold',width=15, textvariable = range\_label\_value1)

r1.place(x=360, y=120)

#Attenuation factor

a = tk.Label(text="Attenuation factor",bg='White',fg='black',font='Helvetica 8 bold',width=15)

a.place(x=360, y=160)

a1 = tk.Label(text="",bg='DarkSeaGreen3',fg='black',font='Helvetica 8 bold',width=15, textvariable = attenuation\_value)

a1.place(x=355, y=180)

#Draw button

graph\_button = tk.Button(root, text="Execute",bg='brown',fg='white',width=10, command=plot)

graph\_button.place(x=50, y=230)

#Reset button

reset\_button = tk.Button(root,text="Reset", bg='brown',fg='white',width=10, command=reset)

reset\_button.place(x=400, y=230)

'''

#Exit button

exit\_button = tk.Button(root,text="Exit", bg='brown',fg='white',width=10, command=exitWindow)

exit\_button.place(x=400, y=170)

'''

root.mainloop()