Комп’ютерний практикум 5. Багатоканальний застосунок із GUI на Python для зображення вимірювальної інформації

Розширити функціонал програми із попередньої практичної додавши декілька вимірювальних каналів, які генерують сигнали (відповідно до вашого варіанту). Зобразити вибірки сигналів на графіку, а також зберігати вибірки в файл в реальному часі. Також додати можливість виводити збережені дані на графік.
Для збереження даних в файл потрібно визначитись з форматом файлу. Тип файлу — текстовий. Для прикладу був вибраний наступний формат:

```
Sun, 22 Apr 2018 12:48:43 +0300 # Дата і час
dt=0.01 # Крок дискретизації
ch0 ch1 ch2 # Назви каналів
0.1244 1.0033 0.1253
-0.0713 1.0070 0.0610
0.0949 0.9382 0.1402
0.1963 1.0498 0.2058
0.1857 1.0113 0.1924
0.3573 0.9694 0.3350
0.3568 0.9375 0.3881
...
```

Значення часу не зберігались. При завантаженні файлу, час розраховується наступним чином: перша часова мітка = 0, остання = довжина вибірки * dt - dt, крок = dt.

План роботи

1. Приклади застосунків
2. Індивідуальні завдання

1. Приклади застосунків

1.1. Моделювання вимірювальної інформації

Лістинг 1

```
import matplotlib
matplotlib.use('TkAgg')

# Бібліотека для математичних розрахунків
import numpy as np

from time import localtime, strftime

# Канва для роботи з графіком
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg

from matplotlib.figure import Figure
```

```
13
14  import tkinter as tk
15
16  tby = dict(side=tk.TOP, \
17             fill=tk.BOTH, \
18             expand=tk.YES)
19
20  txn = dict(side=tk.TOP, \
21             fill=tk.X, \
22             expand=tk.NO)
23
24  class CanvasFrame(tk.Frame):
25      def __init__(self, master):
26          super().__init__(master)
27          self.pack(tby)
28
29          self._canvas = FigureCanvasTkAgg(Figure(figsize=(5, 4), \
30                                                  dpi=100), \
31                                           master=self)
32          self._canvas.get_tk_widget().pack(tby)
33          self._ax = self._canvas.figure.add_subplot(111)
34          self._ax.set_xlabel('t, c')
35          self._ax.grid()
36
37  class RunFrame(tk.Frame):
38      def __init__(self, master):
39          super().__init__(master)
40          self.pack(txn)
41          tk.Label(self, text='dt, c').pack(side=tk.LEFT)
42          self.dt = tk.Entry(self, width=5)
43          self.dt.pack(side=tk.LEFT)
44          self.dt.insert(0, '0.01')
45          tk.Label(self, text='Вибірка, с').pack(side=tk.LEFT)
46          self.sample = tk.Entry(self, width=5)
47          self.sample.pack(side=tk.LEFT)
48          self.sample.insert(0, '1')
49          tk.Label(self, text='Оновлення, мс').pack(side=tk.LEFT)
50          self.fps = tk.Entry(self, width=5)
51          self.fps.pack(side=tk.LEFT)
52          self.fps.insert(0, '500')
53          tk.Button(master=self,
54                    text='Старт',
55                    command=master.start).pack(side=tk.LEFT)
56          tk.Button(master=self,
57                    text='Вихід',
```

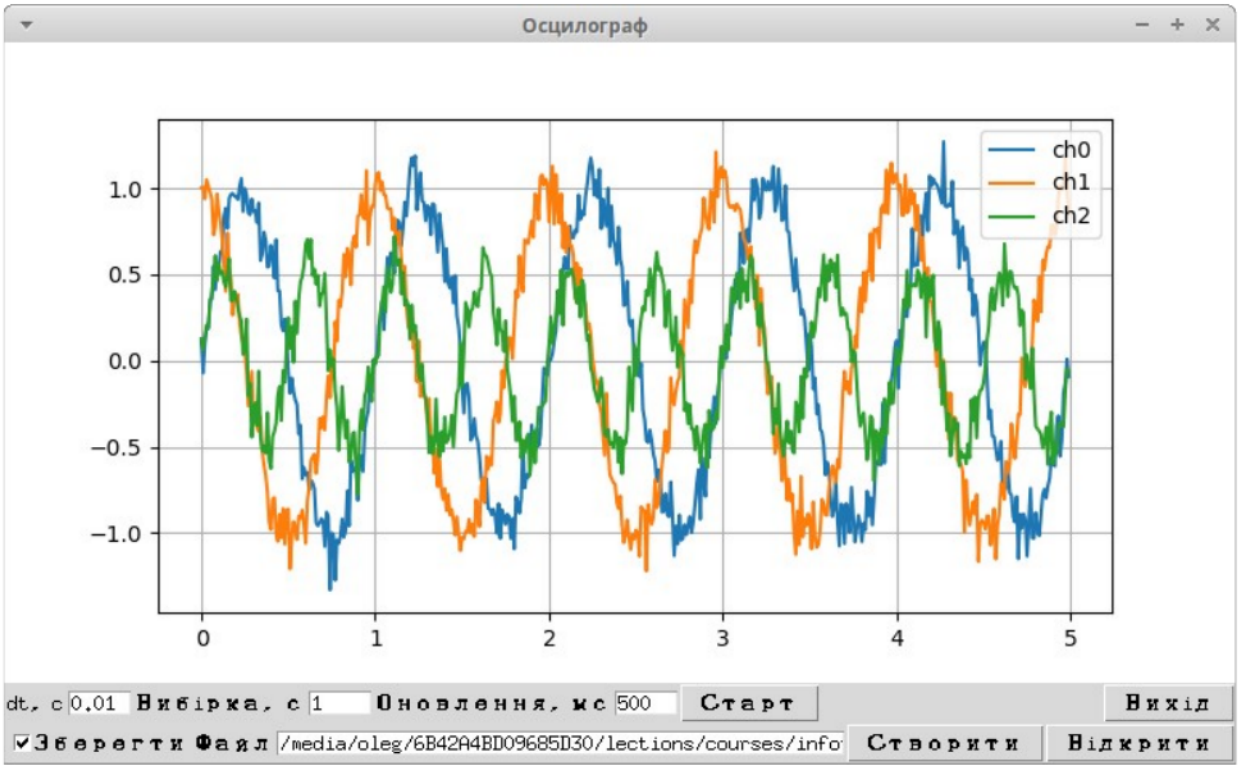
```
58                  command=self._quit).pack(side=tk.RIGHT)
59
60          def _quit(self):
61              self.quit()
62              self.destroy()
63
64          def get_sample_size(self):
65              return eval(self.sample.get())
66
67          def get_fps(self):
68              return eval(self.fps.get())
69
70          def get_dt(self):
71              return eval(self.dt.get())
72
73      class SaveOpenFrame(tk.Frame):
74          def __init__(self, master):
75              super().__init__(master)
76              self.pack(txn)
77              self.fileHandle = None
78              self.checkVar = tk.IntVar()
79                      self.saveCheckbutton  =  tk.Checkbutton(self,  text="Зберегти",
variable=self.checkVar)
80              self.saveCheckbutton.pack(side=tk.LEFT)
81              self.pathVar = tk.StringVar()
82              self.filedialog = tk.filedialog
83              tk.Label(self, text='Файл').pack(side=tk.LEFT)
84              self.file_path = tk.Entry(self, textvariable=self.pathVar, width=50)
85              self.file_path.pack(side=tk.LEFT)
86              tk.Button(master=self,
87                      text='Створити',
88                      command=self._file_save).pack(side=tk.LEFT)
89              tk.Button(master=self,
90                      text='Відкрити',
91                      command=self._file_open).pack(side=tk.RIGHT)
92
93      # Вибір файлів для збереження або зображення даних виконується за
допомогою діалогових вікон:
94      # - збереження -  filedialog.asksaveasfilename;
95      # - відкривання -  filedialog.askopenfilename.
96
97      # Дані дії можуть бути реалізовані у вигляді функцій:
98
99          def _file_save(self):
100             self.pathVar.set(self.filedialog.asksaveasfilename(initialdir = "", \
```

```
                                    title = "Створити файл", \
                                    filetypes = (("Txt files","*.txt"),("all files","*.*")))

Функція _file_save тільки записує шлях до файлу в текстове поле.

    def _file_open(self):
        path = self.filedialog.askopenfilename(initialdir = "", \
                                    title = "Відкрити файл", \
                                    filetypes = (("Txt files","*.txt"),("all files","*.*")))

        chanNames = []
        dt = 0.0
        with open(path, 'r') as f: # Відкриття файлу
            f.readline().split()
            dt = eval(f.readline()[3:])
            chanNames = f.readline().split()
        data = np.genfromtxt(path, skip_header=3)
          self.master.update_canvas([np.arange(0, data.shape[0]*dt, dt), data[:, 0],
data[:, 1], data[:, 2]], chanNames)

Функція _file_open виводить дані на графік.

class Application(tk.Frame):
    def __init__(self, master):
        super().__init__(master)
        self.pack(tby)
        self._timer = None
        self.sampleLen = 0
        self.dt = 0
        self.isStarted = False
        self.channels = ['ch0', 'ch1', 'ch2']
        self._create_widgets()

    def _create_widgets(self):
        self.canvasFrame = CanvasFrame(self)
        self.canvasFrame.pack(tby)
        self.runFrame = RunFrame(self)
        self.runFrame.pack(txn)
        self.saveFrame = SaveOpenFrame(self)
        self.saveFrame.pack(txn)
# Моделювання роботи каналів виконує наступна функція:
    def _genarator(self):
        t = np.arange(0, self.sampleLen, self.dt)
        ch0 = np.sin(2*np.pi*t) + np.random.randn(t.shape[0])/10
        ch1 = np.cos(2*np.pi*t) + np.random.randn(t.shape[0])/10
```

```
145         ch2 = np.sin(2*np.pi*t) * np.cos(2*np.pi*t) + \
146                         np.random.randn(t.shape[0])/10
147         #return np.transpose(np.vstack((t, ch0, ch1, ch2)))
148         return [t, ch0, ch1, ch2]
149
150     def update_canvas(self, data, chanNames):
151         '''
152           Update figure plots
153         '''
154         self.canvasFrame._ax.clear()
155         self.canvasFrame._ax.plot(data[0], data[1], label=chanNames[0])
156         self.canvasFrame._ax.plot(data[0], data[2], label=chanNames[1])
157         self.canvasFrame._ax.plot(data[0], data[3], label=chanNames[2])
158         self.canvasFrame._ax.legend(loc=1)
159         self.canvasFrame._ax.grid()
160         self.canvasFrame._ax.figure.canvas.draw()
161
162     def _save2file(self, data):
163         '''
164           Save data to file
165         '''
166         for x0, x1, x2 in zip(data[1], data[2], data[3]):
167             self.saveFrame.fileHandle.write('{:.4f} {:.4f} {:.4f}{}'.format(x0, x1,
x2, '\n'))
168          self.saveFrame.fileHandle.flush()
169
170     def _plot(self):
171         data = self._genarator()
172         self.update_canvas(data, self.channels)
173
174     def _plot_save(self):
175         data = self._genarator()
176         self.update_canvas(data, self.channels)
177         self._save2file(data)
178
179     def start(self):
180         if self.isStarted:
181             self._timer.stop()
182             if self.saveFrame.fileHandle:
183                 self.saveFrame.fileHandle.close()
184                 self.saveFrame.fileHandle = None
185             self.isStarted = False
186         else:
187             self.sampleLen = self.runFrame.get_sample_size()
188             self.dt = self.runFrame.get_dt()
```

```
189             if self.saveFrame.checkVar.get() and self.saveFrame.pathVar.get():
190                 # Відображення та збереження даних
191 
192                 # Створення і відкриття файлу для запису.
193                 self.saveFrame.fileHandle = open(self.saveFrame.pathVar.get(), 'w')
194                 # Запис дати і часу
195                 self.saveFrame.fileHandle.write('{}{}'.format(\
196                     strftime("%a, %d %b %Y %H:%M:%S +0300", localtime()), '\n'))
197                 # Кроку дискретизації
198                 self.saveFrame.fileHandle.write('dt={}{}'.format(self.dt, '\n'))
199                     # Назви каналів
200                  self.saveFrame.fileHandle.write('{} {} {}{}'.format('ch0', 'ch1', 'ch2',
'\n'))
201                 self.saveFrame.fileHandle.flush()
202                 self._timer = self.canvasFrame._canvas.new_timer(
203                             self.runFrame.get_fps(), \
204                             [(self._plot_save, (), {})])
205             else: # Відображення даних
206                 self._timer = self.canvasFrame._canvas.new_timer(
207                             self.runFrame.get_fps(), \
208                             [(self._plot, (), {})])
209             self._timer.start()
210             self.isStarted = True
211 
212     if __name__ == "__main__":
213         root = tk.Tk()
214         root.wm_title("Осцилограф")
215         app = Application(master=root)
216         app.mainloop()
```

Вікно програми, побудоване з використанням бібліотеки tkinter.



2. Індивідуальні завдання

2.1. Створити застосунок для відображення змодельованих сигналів за прикладом лістингу 1.

2.2. Використати застосунок із пункту 2.1 та відобразити на графіку сигнали із шумом. Типи сигналів вибрати із таблиці 1.

Таблиця 1

№. п/п	Індивідуальне завдання
1.	Синус, Косинус, Меандр
2.	Трикутний, Косинус, Меандр
3.	Синус, Косинус, Меандр
4.	Пилкоподібний, Синус, Косинус,
5.	Синус, Пилкоподібний,

	Меандр
6.	Трикутний, Косинус, Меандр
7.	Пилкоподібний, Косинус, Меандр