Комп’ютерний практикум 4. Застосунок із GUI на Python із динамічно змінним графіком

План роботи

1. Приклади застосунків
2. Індивідуальні завдання

1. Приклади застосунків

1.1. Створимо застосунок із динамічно змінним графіком

Лістинг 1

```
1   import tkinter as tk
2   from tkinter import ttk
3   import matplotlib
4   import numpy as np
5   from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg
6   from matplotlib.figure import Figure
7
8   matplotlib.use('TkAgg')
9
10
11  class GUI(ttk.Frame):
12
13      def __init__(self, master=None, **kwargs):
14          super().__init__(master, **kwargs)
15          self.pack(side=tk.TOP, fill=tk.BOTH, expand=tk.YES)
16
17          self._create_widgets()
18
19          self._update_canvas()
20
21          self._create_timer()
22
23      def _create_widgets(self):
24          self._create_plot_widget()
25
26          ttk.Button(self, text='Quit',
27                     command=self.master.destroy).pack(side=tk.RIGHT)
28
29      def _create_plot_widget(self):
30          self._canvas = FigureCanvasTkAgg(Figure(dpi=100), master=self)
31          self._canvas.get_tk_widget().pack(side=tk.TOP,
32                                            fill=tk.BOTH,
33                                            expand=tk.YES)
34          self._ax = self._canvas.figure.add_subplot()
35
```

```
36            self._update_plot()
37
38        def _update_plot(self):
39            self._ax.set_xlabel('t, s')
40            self._ax.set_ylabel('y(t)')
41            self._ax.grid()
42
43        def _create_timer(self):
44            self._timer = self._canvas.new_timer(500,
45                                                 [(self._update_canvas, (), {})])
46            self._timer.start()
47
48        def _update_canvas(self):
49            self._ax.clear()
50
51            t, y = self._get_measurements()
52
53            self._ax.plot(t, y)
54            self._update_plot()
55            self._ax.figure.canvas.draw()
56
57        def _get_measurements(self):
58            # measurement duration time
59            t_meas = 2  # s
60            t_start = 0  # s
61            dt = 0.001
62            n = (t_meas - t_start) / dt
63            # time samples
64            t = np.arange(t_start, t_meas, dt)
65
66            # signal settings
67            amp = 1.
68            f = 1.  # Hz
69            phi = 0.
70            y = amp * np.sin(2 * np.pi * f * t + phi) + np.random.randn(
71                t.shape[0]) / 3
72
73            return t, y
74
75
76    if __name__ == '__main__':
77        root = tk.Tk()
78        root.title('Oscilloscope')
79        gui = GUI(root)
80        gui.master.mainloop()
```

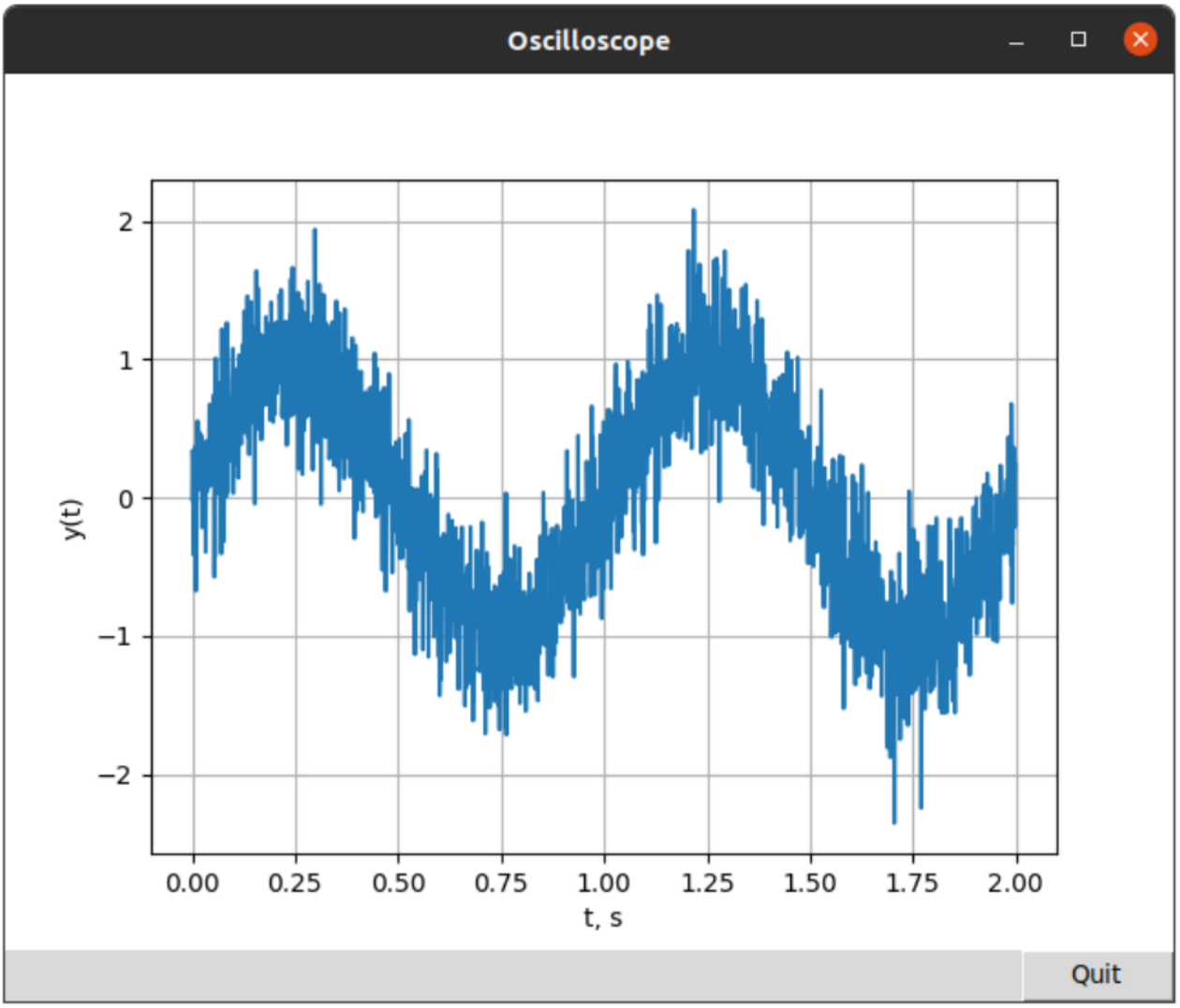

Рис. 1.

1.2. Додамо до графіку поля для вводу тривалості сигналу (sample time) та частоти оновлення графіку (FPS)

Лістинг 2

```
1   import tkinter as tk
2   from tkinter import ttk
3   import matplotlib
4   import numpy as np
5   from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg
6   from matplotlib.figure import Figure
7
8   matplotlib.use('TkAgg')
9
10
11  class GUI(ttk.Frame):
12
13      def __init__(self, master=None, **kwargs):
```

```
14          super().__init__(master, **kwargs)
15          self.pack(side=tk.TOP, fill=tk.BOTH, expand=tk.YES)
16
17          self._create_timer()
18          self._create_widgets()
19
20      def _create_widgets(self):
21          self._create_plot_widget()
22
23          tk.Label(self, text='Sample time, s').pack(side=tk.LEFT)
24          self._time_meas_entry = tk.Entry(self, width=5)
25          self._time_meas_entry.pack(side=tk.LEFT)
26          self._time_meas_entry.insert(0, '1')
27
28          tk.Label(self, text='FPS, ms').pack(side=tk.LEFT)
29          self._fps_entry = tk.Entry(self, width=5)
30          self._fps_entry.pack(side=tk.LEFT)
31          self._fps_entry.insert(0, '500')
32
33          ttk.Button(master=self, text='Run',
34                     command=self._update_timer).pack(side=tk.LEFT)
35          ttk.Button(self, text='Quit',
36                     command=self.master.destroy).pack(side=tk.RIGHT)
37
38      def _create_plot_widget(self):
39          self._canvas = FigureCanvasTkAgg(Figure(dpi=100), master=self)
40          self._canvas.get_tk_widget().pack(side=tk.TOP,
41                                            fill=tk.BOTH,
42                                            expand=tk.YES)
43          self._ax = self._canvas.figure.add_subplot()
44
45          self._update_plot()
46
47      def _update_plot(self):
48          self._ax.set_xlabel('t, s')
49          self._ax.set_ylabel('y(t)')
50          self._ax.grid()
51
52      def _create_timer(self):
53          self._timer = None
54          self._time_meas = 0.
55          self._is_started = False
56
57      def _update_timer(self):
58          if self._is_started:
```

```
59              self._timer.stop()
60              self._is_started = False
61          else:
62              self._time_meas = eval(self._time_meas_entry.get())
63              self._timer = self._canvas.new_timer(eval(self._fps_entry.get()),
64                                                    [(self._update_canvas,
65                                                      (), {})])
66
67              self._timer.start()
68              self._is_started = True
69
70      def _update_canvas(self):
71          self._ax.clear()
72
73          t, y = self._get_measurements(self._time_meas)
74
75          self._ax.plot(t, y)
76          self._update_plot()
77          self._ax.figure.canvas.draw()
78
79      def _get_measurements(self, t_meas):
80          # measurement duration time
81          t_start = 0  # s
82          dt = 0.001
83
84          # time samples
85          t = np.arange(t_start, t_meas, dt)
86
87          # signal settings
88          amp = 1.
89          f = 1.  # Hz
90          phi = 0.
91          y = amp * np.sin(2 * np.pi * f * t + phi) + np.random.randn(
92              t.shape[0]) / 3
93
94          return t, y
95
96
97  if __name__ == '__main__':
98      root = tk.Tk()
99      root.title('Oscilloscope')
100     gui = GUI(root)
101     gui.master.mainloop()
```

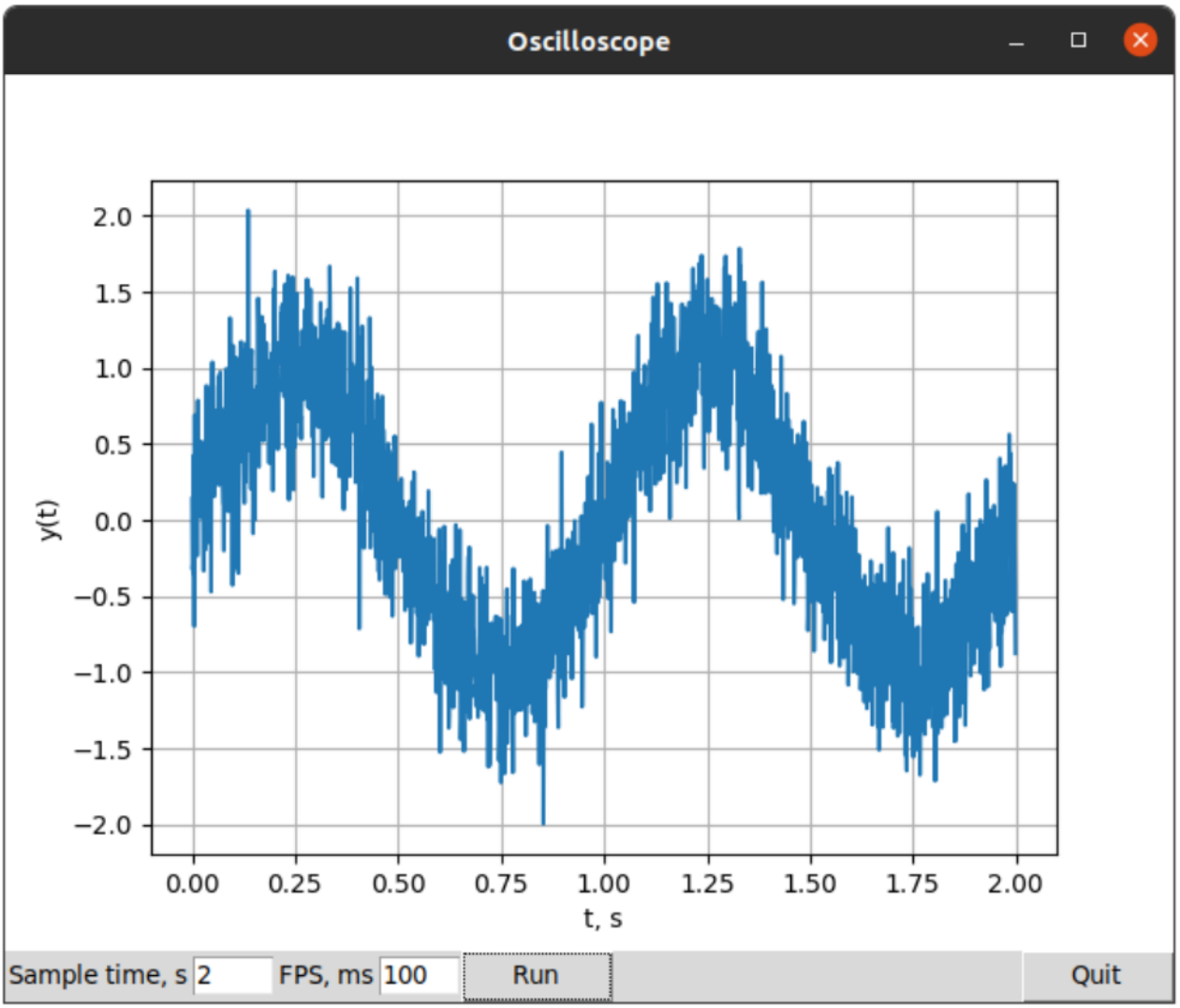

Рис. 2.

2. Індивідуальні завдання

2.1. Створити застосунок за прикладом лістингу 2.

2.2. Використати застосунок із пункту 2.1 та відобразити на графіку сигнал із шумом. Тип сигналу вибрати із таблиці 1.

Таблиця 1

№. п/п	Індивідуальне завдання
1.	Пилоподібний сигнал.
2.	Трикутний сигнал.
3.	Міандр.
4.	Прямокутний сигнал.
5.	Суму двох синусоїд із різними амплітудами та частотами.
6.	Добуток двох косинусних сигналів.

7.	Суму двох косинусних сигналів.