Додаток А
Листінг програми Python

```
import tkinter as tk
from tkinter import ttk, messagebox
from datetime import datetime
import matplotlib.pyplot as plt
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg
import matplotlib.dates as mdates
from menu_bar3 import MenuBar

class MonitorApp:
    def __init__(self, root):
        self.root = root
        self.root.title("Інтелектуальний модуль моніторингу")
        self.root.state('zoomed')

        self.count = 0
        self.start_time = datetime.now()
        self.last_time = "-"
        self.plan = 100
        self.operator = "Іваненко С.В."
        self.status = "□ Активний"
        self.event_log = []
        self.is_paused = False
        self.data_points = []

        self.temperature = 25
        self.temp_alert_shown = False

        self.gas_level = 500
        self.gas_alert_shown = False
```

```
        self.sound_level = 0
        self.sound_alert_shown = False

        self.last_event_time = datetime.now()
        self.alert_shown = False

        self.create_widgets()
        self.menu = MenuBar(self.root, self)
        self.update_data()

    def create_widgets(self):
        self.notebook = ttk.Notebook(self.root)
        self.notebook.pack(fill="both", expand=True)

        self.main_frame = ttk.Frame(self.notebook, padding=10)
        self.notebook.add(self.main_frame, text="Основна")

        self.info_frame = ttk.Frame(self.notebook, padding=10)
        self.notebook.add(self.info_frame, text="Інфо")

        self.setup_main_tab()
        self.setup_info_tab()

    def setup_main_tab(self):
        self.main_frame.rowconfigure(0, weight=3)
        self.main_frame.rowconfigure(1, weight=1)
        self.main_frame.columnconfigure(0, weight=1)

        top_frame = ttk.Frame(self.main_frame)
        top_frame.grid(row=0, column=0, sticky="nsew")
```

```
        top_frame.columnconfigure(0, weight=1)

        info_frame = ttk.LabelFrame(top_frame, text="Інформація про процес",
padding=10)
        info_frame.grid(row=0, column=0, sticky="ew", pady=5)
        info_frame.columnconfigure(0, weight=1)

        self.label_operator = ttk.Label(info_frame, text=f"Оператор:
{self.operator}", font=("Arial", 14))
        self.label_operator.grid(row=0, column=0, sticky="w", pady=2)

        self.label_status = ttk.Label(info_frame, text=f"Статус: {self.status}",
font=("Arial", 14))
        self.label_status.grid(row=1, column=0, sticky="w", pady=2)
        self.label_count = ttk.Label(info_frame, text="Кількість деталей: 0",
font=("Arial", 18, "bold"))
        self.label_count.grid(row=2, column=0, sticky="w", pady=8)
        self.label_last = ttk.Label(info_frame, text="Час останньої деталі: -",
font=("Arial", 14))
        self.label_last.grid(row=3, column=0, sticky="w", pady=2)
        self.label_elapsed = ttk.Label(info_frame, text="Час з початку зміни:
00:00:00", font=("Arial", 14))
        self.label_elapsed.grid(row=4, column=0, sticky="w", pady=2)

        progress_frame = ttk.LabelFrame(top_frame, text="Виконання плану",
padding=10)
        progress_frame.grid(row=1, column=0, sticky="ew", pady=10)
        self.progress = ttk.Progressbar(progress_frame, orient="horizontal",
mode="determinate")
        self.progress.grid(row=0, column=0, padx=5, pady=5, sticky="ew")
```

```
        self.label_plan = ttk.Label(progress_frame, text=f"План: {self.plan} |
Виконання: 0%", font=("Arial", 14))
        self.label_plan.grid(row=1, column=0, sticky="w")

        button_frame = ttk.Frame(top_frame)
        button_frame.grid(row=2, column=0, sticky="ew", pady=10)
        button_frame.columnconfigure((0, 1, 2, 3), weight=1)

        self.btn_simulate = ttk.Button(button_frame, text="Імітувати деталь",
command=self.simulate_detail)
        self.btn_simulate.grid(row=0, column=0, padx=5)
        self.btn_pause = ttk.Button(button_frame, text="Пауза",
command=self.toggle_pause)
        self.btn_pause.grid(row=0, column=1, padx=5)
        self.btn_clear_log = ttk.Button(button_frame, text="Очистити журнал",
command=self.clear_log)
        self.btn_clear_log.grid(row=0, column=2, padx=5)
        self.btn_show_graph = ttk.Button(button_frame, text="Показати графік",
command=self.show_graph_window)
        self.btn_show_graph.grid(row=0, column=3, padx=5)

        log_frame = ttk.LabelFrame(self.main_frame, text="Журнал подій",
padding=10)
        log_frame.grid(row=1, column=0, sticky="nsew", pady=(10, 0))
        log_frame.rowconfigure(0, weight=1)
        log_frame.columnconfigure(0, weight=1)

        self.text_log = tk.Text(log_frame, height=6, state=tk.DISABLED,
bg="#f0f0f0", font=("Consolas", 10))
        self.text_log.grid(row=0, column=0, sticky="nsew")
```

```
        scroll_log = ttk.Scrollbar(log_frame, orient="vertical",
command=self.text_log.yview)
        scroll_log.grid(row=0, column=1, sticky="ns")
        self.text_log['yscrollcommand'] = scroll_log.set

    def setup_info_tab(self):

        temp_frame = ttk.LabelFrame(self.info_frame, text="Температура",
padding=10)
        temp_frame.pack(fill="x", pady=10)

        self.label_temp = ttk.Label(temp_frame, text=f"Температура:
{self.temperature:.1f}°C", font=("Arial", 14))
        self.label_temp.pack(anchor="w", pady=2, padx=5)

        self.slider_temp = ttk.Scale(temp_frame, from_=0, to=100,
orient="horizontal", command=self.on_temp_change)
        self.slider_temp.set(self.temperature)
        self.slider_temp.pack(fill="x", padx=5, pady=5)

        gas_frame = ttk.LabelFrame(self.info_frame, text="Рівень газу",
padding=10)
        gas_frame.pack(fill="x", pady=10)

        self.label_gas = ttk.Label(gas_frame, text=f"Рівень газу: {int(self.gas_level)}
ppm", font=("Arial", 14))
        self.label_gas.pack(anchor="w", pady=2, padx=5)
```

```
        self.slider_gas = ttk.Scale(gas_frame, from_=0, to=2000, orient="horizontal",
command=self.on_gas_change)
        self.slider_gas.set(self.gas_level)
        self.slider_gas.pack(fill="x", padx=5, pady=5)

        sound_frame = ttk.LabelFrame(self.info_frame, text="Рівень звуку",
padding=10)
        sound_frame.pack(fill="x", pady=10)

        self.label_sound = ttk.Label(sound_frame, text=f"Рівень звуку:
{int(self.sound_level)} dB", font=("Arial", 14))
        self.label_sound.pack(anchor="w", pady=2, padx=5)

        self.slider_sound = ttk.Scale(sound_frame, from_=0, to=400,
orient="horizontal", command=self.on_sound_change)
        self.slider_sound.set(self.sound_level)
        self.slider_sound.pack(fill="x", padx=5, pady=5)

    def simulate_detail(self):
        if self.is_paused:
            return
        self.count += 1
        now = datetime.now()
        self.last_time = now.strftime("%H:%M:%S")
        self.last_event_time = now
        self.event_log.append((self.last_time, f"Зібрана деталь №{self.count}"))
        self.data_points.append((now, self.count))
        self.update_display()
        self.update_log()
```

```
    def update_display(self):
        elapsed = datetime.now() - self.start_time
        percent = int((self.count / self.plan) * 100) if self.plan else 0
        self.label_count.config(text=f"Кількість деталей: {self.count}")
        self.label_last.config(text=f"Час останньої деталі: {self.last_time}")
        self.label_elapsed.config(text=f"Час з початку зміни:
{str(elapsed).split('.')[0]}")
        self.label_plan.config(text=f"План: {self.plan} | Виконання: {percent}%")
        self.label_status.config(text=f"Статус: {self.status}")
        self.progress['value'] = percent
        self.label_operator.config(text=f"Оператор: {self.operator}")

    def update_log(self):
        self.text_log.config(state=tk.NORMAL)
        self.text_log.delete(1.0, tk.END)
        for time, msg in reversed(self.event_log[-20:]):
            self.text_log.insert(tk.END, f"[{time}] {msg}\n")
        self.text_log.config(state=tk.DISABLED)
        self.text_log.see(tk.END)

    def clear_log(self):
        self.event_log.clear()
        self.data_points.clear()
        self.text_log.config(state=tk.NORMAL)
        self.text_log.delete(1.0, tk.END)
        self.text_log.config(state=tk.DISABLED)

    def show_graph_window(self):
        if hasattr(self, 'graph_window') and self.graph_window.winfo_exists():
            self.graph_window.lift()
```

```
            return

        self.graph_window = tk.Toplevel(self.root)
        self.graph_window.title("Графік зібраних деталей")
        self.graph_window.geometry("700x400")

        fig, ax = plt.subplots(figsize=(7, 4))
        canvas = FigureCanvasTkAgg(fig, master=self.graph_window)
        canvas.get_tk_widget().pack(fill="both", expand=True)

        if self.data_points:
            times, counts = zip(*self.data_points)
            ax.plot(times, counts, marker='o', linestyle='-')
        ax.set_title("Кількість деталей по часу")
        ax.set_xlabel("Час")
        ax.set_ylabel("Кількість")
        ax.xaxis.set_major_formatter(mdates.DateFormatter('%H:%M:%S'))
        fig.autofmt_xdate()
        ax.grid(True)
        canvas.draw_idle()

    def toggle_pause(self):
        self.is_paused = not self.is_paused
        self.btn_pause.config(text="Відновити" if self.is_paused else "Пауза")
        self.status = "□ На паузі" if self.is_paused else "□ Активний"
        self.label_status.config(text=f"Статус: {self.status}")

    def update_data(self):
        if not self.is_paused:
            self.simulate_detail()
```

```
        now = datetime.now()
        diff = (now - self.last_event_time).total_seconds()
        if diff > 15:
            if not self.alert_shown:
                self.status = "● УВАГА! Зупинка / уповільнення"
                self.label_status.config(text=f"Статус: {self.status}")
                time_str = now.strftime("%H:%M:%S")
                self.event_log.append((time_str, "Простій: понад 15 секунд"))
                self.update_log()
                messagebox.showwarning("Увага!", "Виконання плану уповільнилося
або зупинилося!")
                self.alert_shown = True
        else:
            self.alert_shown = False

        self.check_temperature()
        self.check_gas()
        self.check_sound()

        self.root.after(5000, self.update_data)

    def on_temp_change(self, value):
        self.temperature = float(value)
        self.label_temp.config(text=f"Температура: {self.temperature:.1f}°C")
        self.check_temperature()

    def check_temperature(self):
        if self.temperature < 10 or self.temperature > 50:
            if not self.temp_alert_shown:
```

```
                self.status = "● Температура поза межами!"
                self.label_status.config(text=f"Статус: {self.status}")
                time_str = datetime.now().strftime("%H:%M:%S")
                self.event_log.append((time_str, "Аварія: Температура поза межами"))
                self.update_log()
                messagebox.showerror("Температура!", "Температура критична!
Перевірте умови!")
                self.temp_alert_shown = True
        else:
            if self.temp_alert_shown:
                self.status = "□ Активний"
                self.label_status.config(text=f"Статус: {self.status}")
            self.temp_alert_shown = False

    def on_gas_change(self, value):
        self.gas_level = float(value)
        self.label_gas.config(text=f"Рівень газу: {int(self.gas_level)} ppm")
        self.check_gas()

    def check_gas(self):
        if self.gas_level < 300 or self.gas_level > 1000:
            if not self.gas_alert_shown:
                self.status = "● Газ поза межами!"
                self.label_status.config(text=f"Статус: {self.status}")
                time_str = datetime.now().strftime("%H:%M:%S")
                self.event_log.append((time_str, "Аварія: Газ поза межами"))
                self.update_log()
                messagebox.showerror("Газ!", "Рівень газу критично поза межами!
Перевірте середовище!")
                self.gas_alert_shown = True
```

```
        else:
            if self.gas_alert_shown:
                self.status = "□ Активний"
                self.label_status.config(text=f"Статус: {self.status}")
            self.gas_alert_shown = False

    def on_sound_change(self, value):
        self.sound_level = float(value)
        self.label_sound.config(text=f"Рівень звуку: {int(self.sound_level)} dB")
        self.check_sound()

    def check_sound(self):
        if self.sound_level > 300:
            if not self.sound_alert_shown:
                self.status = "● Звук поза межами!"
                self.label_status.config(text=f"Статус: {self.status}")
                time_str = datetime.now().strftime("%H:%M:%S")
                self.event_log.append((time_str, "Аварія: Звук поза межами"))
                self.update_log()
                messagebox.showerror("Звук!", "Рівень звуку критично високий!
Перевірте середовище!")
                self.sound_alert_shown = True
        else:
            if self.sound_alert_shown:
                self.status = "□ Активний"
                self.label_status.config(text=f"Статус: {self.status}")
            self.sound_alert_shown = False

if __name__ == "__main__":
    root = tk.Tk()
```

```
app = MonitorApp(root)
root.mainloop()
```

Додаток Б

Демонстраційний матеріал

№ докумен-та	Позначення	Найменування	Додаткові відомості
		Текстові документи	
1	ГЮИК. 503320.007 ПЗ	Пояснювальна записка	А4, 66 с.
		Додаткові матеріали	
2		Лістинг програми Python	А4, 11 с.
3		Демонстраційний матеріал у	А4, 16 с.
		вигляді презентації	

Змін.	Арк.	Номер докум.	Підпис	Дата	ГЮИК. 503320.007 ВД			
Розроб.		Михайленко К.А.			Розробка інтелектуального модулю для моніторингу технологічного процесу складання в рамках концепції Індустрія 4.0	Літера	Аркуш	Аркушів
Перевір.		Максимова С.С.				Н	1	1
						Кафедра КІТАР ХНУРЕ		
Н.контр.		Демська Н.П.						
Затв.		Невлюдов І.Ш.						