```python
import tkinter as tk
import threading
import queue
import time
import math
import random

import matplotlib.pyplot as plt
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg

# ==================================================
# CONFIG
# ==================================================
TEST_MODE = True   # False when real hardware is connected

# ==================================================
# GLOBALS
# ==================================================
data_queue = queue.Queue()
running = True
paused = False

# ==================================================
# SEND DATA (SIMULATED)
# ==================================================
def send_data():
    data = entry.get().strip()
    if data:
        text_box.insert(tk.END, f"Sent: {data}\n")
        text_box.see(tk.END)
        entry.delete(0, tk.END)

# ==================================================
# PAUSE / RESUME
# ==================================================
def toggle_pause():
    global paused
    paused = not paused
    pause_btn.config(text="Resume" if paused else "Pause")

# ==================================================
# FAKE MULTI-SENSOR DATA
# Format: value1,value2
# ==================================================
def fake_serial():
    t = 0
    while running:
        if not paused:
            s1 = 50 + 10 * math.sin(t) + random.uniform(-1, 1)
            s2 = 30 + 8 * math.cos(t) + random.uniform(-1, 1)
            data_queue.put(f"{s1:.2f},{s2:.2f}")
            t += 0.1
        time.sleep(0.1)

# ==================================================
# GUI UPDATE LOOP
# ==================================================
x_data = []
y1_data = []
y2_data = []
start_time = time.time()

def update_gui():
    if paused:
        root.after(100, update_gui)
        return

    while not data_queue.empty():
        line = data_queue.get()

        text_box.insert(tk.END, f"Received: {line}\n")
        text_box.see(tk.END)
```

```python

            # Limit text size
            if int(text_box.index("end-1c").split(".")[0]) > 200:
                text_box.delete("1.0", "2.0")

            try:
                v1, v2 = map(float, line.split(","))
                x_data.append(time.time() - start_time)
                y1_data.append(v1)
                y2_data.append(v2)

                if len(x_data) > 100:
                    x_data.pop(0)
                    y1_data.pop(0)
                    y2_data.pop(0)

                line1.set_data(x_data, y1_data)
                line2.set_data(x_data, y2_data)

                ax.relim()
                ax.autoscale_view()
                canvas.draw_idle()

            except:
                pass

        root.after(100, update_gui)

# ==================================================
# SAFE CLOSE
# ==================================================
def on_close():
    global running
    running = False
    time.sleep(0.2)
    root.destroy()

# ==================================================
# TKINTER UI
# ==================================================
root = tk.Tk()
root.title("Serial Plot - Pause/Resume + Multi-Sensor (Simulation)")
root.geometry("950x650")
root.protocol("WM_DELETE_WINDOW", on_close)

top_frame = tk.Frame(root)
top_frame.pack(pady=10)

entry = tk.Entry(top_frame, width=25)
entry.pack(side=tk.LEFT)

tk.Button(top_frame, text="Send", command=send_data).pack(side=tk.LEFT, padx=5)

pause_btn = tk.Button(top_frame, text="Pause", command=toggle_pause)
pause_btn.pack(side=tk.LEFT, padx=5)

text_box = tk.Text(root, height=10)
text_box.pack(padx=10, pady=10, fill=tk.X)

# ==================================================
# MATPLOTLIB
# ==================================================
fig, ax = plt.subplots(figsize=(8, 4))
line1, = ax.plot([], [], label="Sensor 1", linewidth=2)
line2, = ax.plot([], [], label="Sensor 2", linewidth=2)

ax.set_title("Live Multi-Sensor Data")
ax.set_xlabel("Time (s)")
ax.set_ylabel("Value")
ax.legend()
ax.grid(True)

```

```python
canvas = FigureCanvasTkAgg(fig, master=root)
canvas.get_tk_widget().pack(fill=tk.BOTH, expand=True)

# ==================================================
# START
# ==================================================
threading.Thread(target=fake_serial, daemon=True).start()
root.after(100, update_gui)
root.mainloop()

```