```
import tkinter as tk
from tkinter import ttk
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg
import matplotlib.pyplot as plt
import numpy as np
from scipy.signal import butter, lfilter

# Fungsi untuk menghasilkan sinyal dasar
def generate_signal(sensor_type):
    t = np.linspace(0, 2, 500)
    if sensor_type == "Seismic":
        signal = 10 * np.sin(2 * np.pi * 1 * t) + np.random.normal(0, 0.5, len(t))
    elif sensor_type == "Vibration":
        signal = 5 * np.sin(2 * np.pi * 10 * t) + np.random.normal(0, 0.5, len(t))
    elif sensor_type == "Camera":
        signal = np.abs(5 * np.sin(2 * np.pi * 0.5 * t))
    elif sensor_type == "DHT22":
        signal = 25 + 2 * np.sin(2 * np.pi * 0.2 * t) + np.random.normal(0, 0.2, len(t))
    elif sensor_type == "Electrochemical":
        signal = 1 + 0.5 * np.sin(2 * np.pi * 5 * t) + np.random.normal(0, 0.1, len(t))
    else:
        signal = np.zeros_like(t)
    return t, signal

# Fungsi filter low-pass
def low_pass_filter(signal, cutoff=2, fs=250, order=2):
    nyq = 0.5 * fs
    normal_cutoff = cutoff / nyq
    b, a = butter(order, normal_cutoff, btype='low', analog=False)
    y = lfilter(b, a, signal)
    return y

# Fungsi untuk memperbarui grafik
def update_plot(sensor_type):
    global ax1, ax2, ax3, ax4, fig, canvas
    t, signal = generate_signal(sensor_type)
    noise_amplitude = scale_noise_amplitude.get()
    noise_frequency = scale_noise_frequency.get()
    noise = noise_amplitude * np.sin(2 * np.pi * noise_frequency * t)
    noisy_signal = signal + noise
    filtered_signal = low_pass_filter(noisy_signal)

    # Clear all axes
    ax1.clear()
    ax2.clear()
    ax3.clear()
```

```
    ax4.clear()

    # Set background color to black
    for ax in [ax1, ax2, ax3, ax4]:
        ax.set_facecolor("black")
        ax.tick_params(colors="white")
        ax.spines[:].set_color("white")

    # Plot sinyal asli
    ax1.plot(t, signal, label=f"{sensor_type} Signal", color="cyan")
    ax1.set_title(f"{sensor_type} Signal", color="white")
    ax1.set_xlabel("Time (s)", color="white")
    ax1.set_ylabel("Amplitude", color="white")
    ax1.legend()
    ax1.grid(color="gray")

    # Plot sinyal dengan noise
    ax2.plot(t, noisy_signal, label=f"{sensor_type} + Noise", color="yellow")
    ax2.set_title(f"{sensor_type} + Noise Signal", color="white")
    ax2.set_xlabel("Time (s)", color="white")
    ax2.set_ylabel("Amplitude", color="white")
    ax2.legend()
    ax2.grid(color="gray")

    # Plot sinyal yang difilter
    ax3.plot(t, filtered_signal, label="Low Pass Filtered Signal", color="lime")
    ax3.set_title("Low Pass Filtered Signal", color="white")
    ax3.set_xlabel("Time (s)", color="white")
    ax3.set_ylabel("Amplitude", color="white")
    ax3.legend()
    ax3.grid(color="gray")

    # Plot DFT dari sinyal
    dft_signal = np.abs(np.fft.fft(signal))
    freq = np.fft.fftfreq(len(t), d=t[1] - t[0])
    ax4.plot(freq[:len(freq)//2], dft_signal[:len(dft_signal)//2], label="DFT of Signal",
color="magenta")
    ax4.set_title("DFT of Signal", color="white")
    ax4.set_xlabel("Frequency (Hz)", color="white")
    ax4.set_ylabel("Magnitude", color="white")
    ax4.legend()
    ax4.grid(color="gray")

    canvas.draw()

# Setup GUI
```

```
root = tk.Tk()
root.title("Signal Processing with Multiple Sensors")
root.configure(bg="black")  # Background GUI to black

# Frame kiri untuk memilih sensor
frame_left = tk.Frame(root, bg="black")
frame_left.pack(side=tk.LEFT, fill=tk.Y, padx=10, pady=10)

sensor_var = tk.StringVar(value="Seismic")
tk.Label(frame_left, text="Select Sensor:", bg="black", fg="white").pack(anchor=tk.W)
sensors = ["Seismic", "Vibration", "Camera", "DHT22", "Electrochemical"]
for sensor in sensors:
    tk.Radiobutton(frame_left, text=sensor, variable=sensor_var, value=sensor,
command=lambda: update_plot(sensor_var.get()),
                   bg="black", fg="white", selectcolor="gray").pack(anchor=tk.W)

# Slider untuk noise
tk.Label(frame_left, text="Noise Amplitude", bg="black", fg="white").pack(anchor=tk.W,
pady=(10, 0))
scale_noise_amplitude = tk.Scale(frame_left, from_=0, to=5, resolution=0.1,
orient=tk.HORIZONTAL, bg="black", fg="white")
scale_noise_amplitude.pack(fill=tk.X)

tk.Label(frame_left, text="Noise Frequency", bg="black", fg="white").pack(anchor=tk.W,
pady=(10, 0))
scale_noise_frequency = tk.Scale(frame_left, from_=0, to=20, resolution=0.1,
orient=tk.HORIZONTAL, bg="black", fg="white")
scale_noise_frequency.pack(fill=tk.X)

# Frame kanan untuk grafik
frame_right = tk.Frame(root, bg="black")
frame_right.pack(side=tk.RIGHT, fill=tk.BOTH, expand=True, padx=10, pady=10)

# Membuat subplots
fig, ((ax1, ax2), (ax3, ax4)) = plt.subplots(2, 2, figsize=(10, 8))
fig.patch.set_facecolor("black")  # Background figure to black
canvas = FigureCanvasTkAgg(fig, master=frame_right)
canvas_widget = canvas.get_tk_widget()
canvas_widget.pack(fill=tk.BOTH, expand=True)

# Inisialisasi grafik awal
update_plot("Seismic")

# Menjalankan GUI
root.mainloop()
```