TUGAS PROJEK

MATAKULIAH MIKROKONTROLLER

Dosen Pengampu :

Akhmad Hendriawan

PROGRAM STUDI TEKNIK ELEKTRONIKA

DEPARTEMEN TEKNIK ELEKTRO

POLITEKNIK ELEKTRONIKA NEGERI SURABAYA

2024

1. Pendahuluan

   Dalam era digital saat ini, pemantauan konsumsi daya listrik secara real-time menjadi kebutuhan penting baik di rumah tangga maupun industri. Sistem monitoring tegangan berbasis mikrokontroler dan GUI (Graphical User Interface) memberikan kemudahan pengguna untuk memantau, menganalisis, dan mengendalikan penggunaan listrik melalui tampilan visual yang interaktif dan informatif.
   Pada proyek ini, dikembangkan dua jenis antarmuka GUI menggunakan Python dengan dua pendekatan berbeda:

- GUI berbasis Tkinter standar + ttk
- GUI berbasis CustomTkinter (ctk) dengan tampilan modern

Kedua GUI berfungsi untuk membaca data tegangan dari serial (baik real maupun dummy), menampilkannya dalam grafik, menghitung energi (kWh), estimasi biaya, dan memberikan kontrol beban (on/off lampu) melalui tombol.

2. Tujuan Pembelajaran

   - Mempelajari penerapan Tkinter dan CustomTkinter dalam membuat aplikasi GUI monitoring berbasis Python.

   - Mengintegrasikan data serial ke dalam GUI untuk menampilkan informasi secara real-time.

   - Membandingkan performa, estetika, dan kemudahan penggunaan dari dua pendekatan GUI.

   - Memahami bagaimana struktur program GUI yang baik, modular, dan dapat diperluas untuk fitur tambahan seperti IoT atau database.

3. Perbandingan program GUI

   Program yang pertama dibuat :

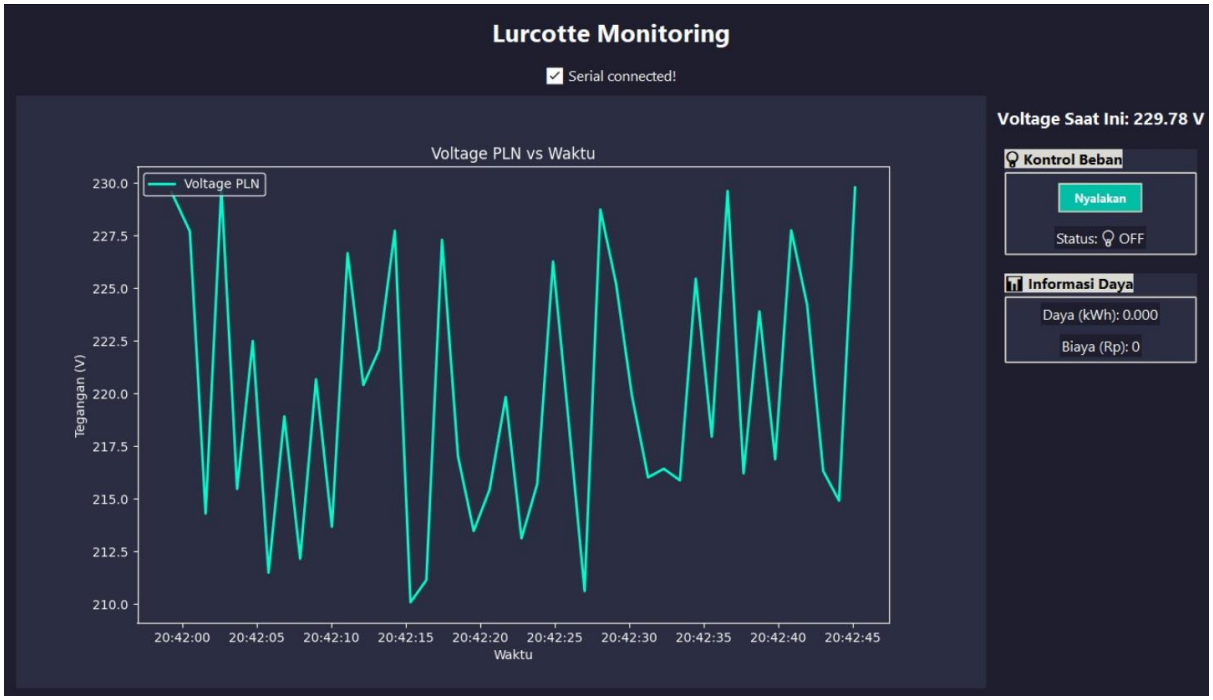
```
import tkinter as tk
from tkinter import ttk
import matplotlib.pyplot as plt
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg
import matplotlib.dates as mdates
import datetime
import serial
import threading
import time
import random
```

```
# --- Inisialisasi koneksi serial ---
try:
    ser = serial.serial_for_url('rfc2217://localhost:4000', baudrate=115200,
timeout=1)
    serial_status = "✅ Serial connected!"
except Exception as e:
    serial_status = f"❌ Gagal terhubung serial: {e}"
    ser = None

# --- GUI Setup ---
root = tk.Tk()
root.title("Smart Voltage Monitor")
root.geometry("1350x800")
root.configure(bg="#0f172a")  # dark navy blue

# --- Global Variables ---
time_data = []
voltage_data = []
load_on = False
power_accumulated_kwh = 0.0
last_update_time = time.time()

# --- Styles ---
style = ttk.Style()
style.theme_use('clam')
style.configure("TLabel", background="#0f172a", foreground="white",
font=("Segoe UI", 12))
style.configure("TButton", background="#1e293b", foreground="white",
font=("Segoe UI", 10, "bold"))
style.configure("Custom.TLabelframe", background="#1e293b",
foreground="white", font=("Segoe UI", 12, "bold"))
style.configure("Custom.TLabelframe.Label", background="#1e293b",
foreground="white")

# --- Header ---
header_frame = tk.Frame(root, bg="#1e293b")
header_frame.pack(fill=tk.X, padx=0, pady=0)

app_title = tk.Label(header_frame, text="⚡ Lurcotte", font=("Segoe UI", 20,
"bold"), fg="#00f5d4", bg="#1e293b")
app_title.pack(pady=10)
status_label = ttk.Label(header_frame, text=serial_status)
status_label.pack(pady=2)

# --- Main Frame ---
main_frame = tk.Frame(root, bg="#0f172a")
main_frame.pack(fill=tk.BOTH, expand=True, padx=20, pady=10)
```

```
# --- Chart Frame ---
frame_chart = tk.Frame(main_frame, bg="#1e293b")
frame_chart.grid(row=0, column=0, columnspan=2, sticky="nsew", padx=10,
pady=10)
main_frame.grid_rowconfigure(0, weight=3)
main_frame.grid_columnconfigure(0, weight=3)
main_frame.grid_columnconfigure(1, weight=1)

fig, ax = plt.subplots(figsize=(10, 4), facecolor='#1e293b')
fig.patch.set_facecolor('#1e293b')
ax.set_facecolor('#1e293b')
ax.tick_params(colors='white')
for spine in ax.spines.values():
    spine.set_color('white')
ax.xaxis.set_major_formatter(mdates.DateFormatter('%H:%M:%S'))
line, = ax.plot([], [], color='#00ffcc', linewidth=2, label='Voltage PLN')
ax.legend(loc='upper left', facecolor='#1e293b', edgecolor='white',
labelcolor='white')
canvas = FigureCanvasTkAgg(fig, master=frame_chart)
canvas.get_tk_widget().pack(fill=tk.BOTH, expand=True)

# --- Info Panel ---
info_panel = tk.Frame(main_frame, bg="#1e293b")
info_panel.grid(row=0, column=2, sticky="nsew", padx=10, pady=10)

current_voltage_label = ttk.Label(info_panel, text="Voltage Saat Ini: -- V",
font=("Segoe UI", 16, "bold"))
current_voltage_label.pack(pady=15)

# --- Load Control Frame ---
load_frame = ttk.Labelframe(info_panel, text="Kontrol Beban",
style="Custom.TLabelframe")
load_frame.pack(fill=tk.X, padx=10, pady=10)

def toggle_load():
    global load_on
    load_on = not load_on
    load_status_label.config(text="Lampu: ❁ ON" if load_on else "Lampu: 💡
OFF")
    btn_load_toggle.config(text="Matikan" if load_on else "Nyalakan")

btn_load_toggle = ttk.Button(load_frame, text="Nyalakan", command=toggle_load)
btn_load_toggle.pack(pady=8)

load_status_label = ttk.Label(load_frame, text="Lampu: 💡 OFF", font=("Segoe
UI", 14))
load_status_label.pack(pady=5)
```

```
# --- Energy Info Frame ---
info_energy_frame = ttk.Labelframe(info_panel, text="Informasi Daya",
style="Custom.TLabelframe")
info_energy_frame.pack(fill=tk.X, padx=10, pady=10)

power_label = ttk.Label(info_energy_frame, text="Daya (kWh): 0.000")
power_label.pack(pady=8)

cost_label = ttk.Label(info_energy_frame, text="Biaya (Rp): 0")
cost_label.pack(pady=8)

# --- Footer Note ---
footer = tk.Label(root, text="Developed by Lurcotte Corporation | Politeknik
Elektronika Negeri Surabaya", font=("Segoe UI", 10), bg="#0f172a",
fg="#94a3b8")
footer.pack(pady=5)

# --- Update Graph ---
def update_graph():
    ax.clear()
    ax.set_facecolor('#1e293b')
    ax.tick_params(colors='white')
    for spine in ax.spines.values():
        spine.set_color('white')
    ax.xaxis.set_major_formatter(mdates.DateFormatter('%H:%M:%S'))
    ax.plot(time_data, voltage_data, color='#00ffcc', linewidth=2,
label='Voltage PLN')
    ax.set_title("Voltage PLN vs Waktu", color='white')
    ax.set_xlabel("Waktu", color='white')
    ax.set_ylabel("Tegangan (V)", color='white')
    ax.legend(loc='upper left', facecolor='#1e293b', edgecolor='white',
labelcolor='white')
    canvas.draw()

# --- Serial Read Thread ---
def read_serial():
    global power_accumulated_kwh, last_update_time

    while True:
        try:
            if ser and ser.in_waiting:
                line = ser.readline().decode('utf-8').strip()
                if line.startswith("V:"):
                    voltage = float(line[2:].strip())
                else:
                    continue
            else:
                voltage = random.uniform(210.0, 230.0)
```

```
            now = datetime.datetime.now()
            time_data.append(now)
            voltage_data.append(voltage)
            if len(time_data) > 60:
                time_data.pop(0)
                voltage_data.pop(0)

            current_voltage_label.config(text=f"Voltage Saat Ini:
{voltage:.2f} V")
            update_graph()

            current_time = time.time()
            elapsed = current_time - last_update_time
            last_update_time = current_time

            if load_on:
                current_power = voltage * 0.2
                energy = current_power * (elapsed / 3600.0)
                power_accumulated_kwh += energy
                biaya = power_accumulated_kwh * 1500

                power_label.config(text=f"Daya (kWh):
{power_accumulated_kwh:.3f}")
                cost_label.config(text=f"Biaya (Rp):
{int(biaya):,}".replace(',', '.'))

            time.sleep(1)

        except Exception as e:
            print("Error membaca data serial:", e)
            time.sleep(2)

# --- Start Thread ---
threading.Thread(target=read_serial, daemon=True).start()

# --- Run App ---
root.mainloop()
```



Program yang digunakan sementara ini :

```
import customtkinter as ctk
import tkinter as tk
import matplotlib.pyplot as plt
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg
import matplotlib.dates as mdates
import datetime
import serial
import threading
import time
import random

# Setup CustomTkinter appearance
ctk.set_appearance_mode("light")
ctk.set_default_color_theme("blue")

# --- Inisialisasi koneksi serial ---
try:
    ser = serial.serial_for_url('rfc2217://localhost:4000', baudrate=115200,
timeout=1)
    serial_status = "✅ Serial connected!"
except Exception as e:
    serial_status = f"❌ Gagal terhubung serial: {e}"
    ser = None

# --- Global Variables ---
```

```
time_data = []
voltage_data = []
load_on = False
power_accumulated_kwh = 0.0
last_update_time = time.time()

# --- Root Window ---
root = ctk.CTk()
root.title("Smart Voltage Monitor")
root.geometry("1350x800")
root.configure(fg_color="#f3f4f6")  # Background soft grey

# --- Header ---
header = ctk.CTkFrame(root, corner_radius=10, fg_color="#cbd5e1")
header.pack(fill="x", padx=20, pady=(20, 10))

ctk.CTkLabel(header, text="⚡ Lurcotte", font=("Segoe UI", 24, "bold"),
text_color="#334155").pack(pady=10)
ctk.CTkLabel(header, text=serial_status, font=("Segoe UI", 12),
text_color="#1e293b").pack(pady=2)

# --- Main Frame ---
main_frame = ctk.CTkFrame(root, fg_color="#f3f4f6")
main_frame.pack(fill="both", expand=True, padx=20, pady=10)
main_frame.grid_columnconfigure((0, 1, 2), weight=1)
main_frame.grid_rowconfigure(0, weight=1)

# --- Grafik ---
graph_card = ctk.CTkFrame(main_frame, corner_radius=20, fg_color="white")
graph_card.grid(row=0, column=0, columnspan=2, sticky="nsew", padx=10,
pady=10)

fig, ax = plt.subplots(figsize=(9, 3.5))
fig.patch.set_facecolor('#ffffff')
ax.set_facecolor('#ffffff')
ax.tick_params(colors='#1e293b')
for spine in ax.spines.values():
    spine.set_color('#94a3b8')
ax.xaxis.set_major_formatter(mdates.DateFormatter('%H:%M:%S'))
line, = ax.plot([], [], color='#0284c7', linewidth=2, marker='o',
label='Voltage PLN')
ax.legend(loc='upper left')

canvas = FigureCanvasTkAgg(fig, master=graph_card)
canvas.get_tk_widget().pack(fill='both', expand=True, padx=10, pady=10)

# --- Panel Kontrol dan Info ---
control_card = ctk.CTkFrame(main_frame, corner_radius=20, fg_color="#e0f2fe")
```

```
control_card.grid(row=0, column=2, sticky="nsew", padx=10, pady=10)

# --- Kontrol Beban ---
ctk.CTkLabel(control_card, text="💡 Kontrol Beban", font=("Segoe UI", 18,
"bold"), text_color="#0f172a").pack(anchor="w", padx=20, pady=(15, 5))
control_inner = ctk.CTkFrame(control_card, corner_radius=15,
fg_color="#ffffff")
control_inner.pack(fill="x", padx=20, pady=5)

emoji_label = ctk.CTkLabel(control_inner, text="🔌", font=("Segoe UI", 42))
emoji_label.grid(row=0, column=0, rowspan=2, padx=(15, 10), pady=10)

load_status_label = ctk.CTkLabel(control_inner, text="Lampu: 💡 OFF",
font=("Segoe UI", 14))
load_status_label.grid(row=0, column=1, sticky="w", padx=5, pady=(15, 5))

btn_load_toggle = ctk.CTkButton(control_inner, text="Nyalakan",
command=lambda: toggle_load(), width=120)
btn_load_toggle.grid(row=1, column=1, sticky="w", padx=5, pady=(0, 15))

# --- Info Energi ---
ctk.CTkLabel(control_card, text="🔋 Informasi Daya", font=("Segoe UI", 18,
"bold"), text_color="#0f172a").pack(anchor="w", padx=20, pady=(20, 5))
info_inner = ctk.CTkFrame(control_card, corner_radius=15, fg_color="#fef9c3")
info_inner.pack(fill="x", padx=20, pady=5)

emoji_energy = ctk.CTkLabel(info_inner, text="⚡", font=("Segoe UI", 42))
emoji_energy.grid(row=0, column=0, rowspan=2, padx=(15, 10), pady=10)

power_label = ctk.CTkLabel(info_inner, text="Daya (kWh): 0.000", font=("Segoe
UI", 14))
power_label.grid(row=0, column=1, sticky="w", padx=5, pady=(15, 5))

cost_label = ctk.CTkLabel(info_inner, text="Biaya (Rp): 0", font=("Segoe UI",
14))
cost_label.grid(row=1, column=1, sticky="w", padx=5, pady=(0, 15))

# --- Footer / Placeholder / Quotes ---
footer_card = ctk.CTkFrame(root, corner_radius=10, fg_color="#cbd5e1")
footer_card.pack(fill="x", padx=20, pady=(0, 10))

quote = "\u2728 Selalu pantau dan kendalikan penggunaan daya listrikmu untuk
masa depan yang lebih hemat dan ramah lingkungan."
ctk.CTkLabel(footer_card, text=quote, font=("Segoe UI", 12, "italic"),
text_color="#334155").pack(pady=8)

# --- Function: Toggle Load ---
def toggle_load():
```

```
    global load_on
    load_on = not load_on
    load_status_label.configure(text="Lampu: ❁ ON" if load_on else "Lampu:
💡 OFF")
    btn_load_toggle.configure(text="Matikan" if load_on else "Nyalakan")

# --- Function: Update Graph ---
def update_graph():
    ax.clear()
    ax.set_facecolor('#ffffff')
    ax.tick_params(colors='#1e293b')
    for spine in ax.spines.values():
        spine.set_color('#94a3b8')
    ax.xaxis.set_major_formatter(mdates.DateFormatter('%H:%M:%S'))
    ax.plot(time_data, voltage_data, color='#0284c7', linewidth=2, marker='o',
label='Voltage PLN')
    ax.set_title("Tegangan PLN Terhadap Waktu", color='#0f172a')
    ax.set_xlabel("Waktu", color='#0f172a')
    ax.set_ylabel("Tegangan (V)", color='#0f172a')
    ax.legend(loc='upper left')
    canvas.draw()

# --- Function: Serial Thread ---
def read_serial():
    global power_accumulated_kwh, last_update_time
    while True:
        try:
            if ser and ser.in_waiting:
                line = ser.readline().decode('utf-8').strip()
                if line.startswith("V:"):
                    voltage = float(line[2:].strip())
                else:
                    continue
            else:
                voltage = random.uniform(210.0, 230.0)

            now = datetime.datetime.now()
            time_data.append(now)
            voltage_data.append(voltage)
            if len(time_data) > 60:
                time_data.pop(0)
                voltage_data.pop(0)

            update_graph()

            current_time = time.time()
            elapsed = current_time - last_update_time
            last_update_time = current_time
```

```
            if load_on:
                current_power = voltage * 0.2
                energy = current_power * (elapsed / 3600.0)
                power_accumulated_kwh += energy
                biaya = power_accumulated_kwh * 1500

                power_label.configure(text=f"Daya (kWh):
{power_accumulated_kwh:.3f}")
                cost_label.configure(text=f"Biaya (Rp):
{int(biaya):,}".replace(',', '.'))

            time.sleep(1)

        except Exception as e:
            print("Error membaca data serial:", e)
            time.sleep(2)

# --- Start Serial Thread ---
threading.Thread(target=read_serial, daemon=True).start()

# --- Run App ---
root.mainloop()
```

Perbandingan kedua program GUI tersebut :

Aspek	GUI 1 (Tkinter + ttk)	GUI 2 (CustomTkinter)
Tampilan Visual	Sederhana, bergaya klasik (gelap), menggunakan ttk styling manual	Lebih modern, bersih, dengan tampilan flat dan tema yang konsisten
Komponen GUI	Menggunakan `tk.Frame`, `ttk.Label`, `ttk.Button`, dan `matplotlib`	Menggunakan `ctk.CTkFrame`, `ctk.CTkLabel`, `ctk.CTkButton`, dan `matplotlib`
Tema & Warna	Dark theme dengan tone biru tua	Light theme dengan tone biru terang dan putih
Kemudahan Customisasi	Perlu styling manual per elemen (`ttk.Style`)	Sudah tersedia default theme dan `corner_radius`, `text_color`, dll
Modularitas & Estetika	Kurang fleksibel untuk komponen modern	Lebih fleksibel dan rapi untuk komponen modern
Fungsi Dasar	Menampilkan grafik tegangan, kontrol beban, perhitungan kWh dan biaya	Fungsi serupa dengan antarmuka yang lebih estetis dan user-friendly
Status Serial	Ditampilkan dengan `ttk.Label` di bagian header	Ditampilkan dengan `ctk.CTkLabel` dengan warna konsisten
Fitur Unik	Lebih ringan dijalankan, dependensi lebih sedikit	UI yang lebih menarik dan profesional, cocok untuk produk final

Kesimpulan :

Kedua GUI memiliki fungsionalitas utama yang sama, namun perbedaan terletak pada tampilan dan fleksibilitas desain. GUI berbasis CustomTkinter lebih unggul dalam hal user experience dan profesionalitas tampilan, meskipun membutuhkan dependensi tambahan.

GUI Tkinter cocok untuk prototipe cepat dan sistem ringan, sementara GUI CustomTkinter lebih ideal untuk aplikasi yang siap dipresentasikan ke pengguna akhir atau untuk integrasi sistem monitoring skala lebih besar.