RS-232通訊 Python程式碼

一張含有 文字, 電腦, 電子產品, 電腦硬體 的圖片

AI 產生的內容可能不正確。 一張含有 文字, 人員, 小工具, 行動電話 的圖片

AI 產生的內容可能不正確。

import time  
import csv  
from datetime import datetime  
import serial.tools.list\_ports  
from pymodbus.client import ModbusSerialClient  
import matplotlib.pyplot as plt  
from matplotlib.animation import FuncAnimation  
import tkinter as tk  
from tkinter import ttk  
  
# ────── COM 埠選擇 ──────  
def select\_com\_port():  
 ports = serial.tools.list\_ports.comports()  
 available = [port.device for port in ports]  
 if not available:  
 print("❌ 沒有可用的 COM 埠，請確認設備已插入")  
 exit()  
  
 def on\_select():  
 selected\_port.set(combo.get())  
 window.destroy()  
  
 window = tk.Tk()  
 window.title("選擇 COM 埠")  
 tk.Label(window, text="請選擇 COM 埠：").pack(padx=10, pady=10)  
 combo = ttk.Combobox(window, values=available, state="readonly")  
 combo.current(0)  
 combo.pack(padx=10, pady=5)  
 tk.Button(window, text="開始", command=on\_select).pack(padx=10, pady=10)  
 selected\_port = tk.StringVar()  
 window.mainloop()  
 return selected\_port.get()  
  
# ────── 警告值輸入 ──────  
def get\_warning\_thresholds():  
 def on\_submit():  
 try:  
 thresholds['rpm'] = float(entry\_rpm.get())  
 thresholds['voltage'] = float(entry\_v.get())  
 thresholds['current'] = float(entry\_a.get())  
 input\_window.destroy()  
 except ValueError:  
 tk.Label(input\_window, text="⚠️ 請輸入數值").pack()  
  
 thresholds = {}  
 input\_window = tk.Tk()  
 input\_window.title("輸入警告值")  
  
 tk.Label(input\_window, text="轉速警告值 (RPM)").pack()  
 entry\_rpm = tk.Entry(input\_window)  
 entry\_rpm.pack()  
  
 tk.Label(input\_window, text="電壓警告值 (V)").pack()  
 entry\_v = tk.Entry(input\_window)  
 entry\_v.pack()  
  
 tk.Label(input\_window, text="電流警告值 (A)").pack()  
 entry\_a = tk.Entry(input\_window)  
 entry\_a.pack()  
  
 tk.Button(input\_window, text="確認", command=on\_submit).pack(pady=10)  
 input\_window.mainloop()  
 return thresholds  
  
# 執行設定  
com\_port = select\_com\_port()  
thresholds = get\_warning\_thresholds()  
  
# ────── 初始化 Modbus RTU ──────  
client = ModbusSerialClient(  
 port=com\_port,  
 baudrate=19200,  
 parity='N',  
 stopbits=2,  
 bytesize=8,  
 timeout=1  
)  
  
# ────── 資料初始化 ──────  
times, rpm\_51, rpm\_53, rpm\_55 = [], [], [], []  
v\_52, v\_54, v\_56 = [], [], []  
a\_52, a\_54, a\_56 = [], [], []  
  
# ────── 繪圖設定 ──────  
plt.style.use('ggplot')  
fig, (ax\_rpm, ax\_v, ax\_a) = plt.subplots(3, 1, figsize=(12, 9), sharex=True)  
  
# 折線資料物件  
line\_rpm\_51, = ax\_rpm.plot([], [], label='RPM\_51', color='tab:blue')  
line\_rpm\_53, = ax\_rpm.plot([], [], label='RPM\_53', color='tab:orange')  
line\_rpm\_55, = ax\_rpm.plot([], [], label='RPM\_55', color='tab:green')  
line\_v\_52, = ax\_v.plot([], [], label='V\_52', color='tab:blue')  
line\_v\_54, = ax\_v.plot([], [], label='V\_54', color='tab:orange')  
line\_v\_56, = ax\_v.plot([], [], label='V\_56', color='tab:green')  
line\_a\_52, = ax\_a.plot([], [], label='A\_52', color='tab:blue')  
line\_a\_54, = ax\_a.plot([], [], label='A\_54', color='tab:orange')  
line\_a\_56, = ax\_a.plot([], [], label='A\_56', color='tab:green')  
  
# 警告虛線標示  
ax\_rpm.axhline(  
 y=thresholds['rpm'], color='red', linestyle='--',  
 label=f'RPM Warning ({thresholds["rpm"]})'  
)  
ax\_v.axhline(  
 y=thresholds['voltage'], color='red', linestyle='--',  
 label=f'Voltage Warning ({thresholds["voltage"]})'  
)  
ax\_a.axhline(  
 y=thresholds['current'], color='red', linestyle='--',  
 label=f'Current Warning ({thresholds["current"]})'  
)  
  
# 標籤設定  
ax\_rpm.set\_ylabel('RPM')  
ax\_rpm.set\_title('Real-time RPM')  
ax\_rpm.legend()  
  
ax\_v.set\_ylabel('Voltage (V)')  
ax\_v.set\_title('Real-time Voltage')  
ax\_v.legend()  
  
ax\_a.set\_xlabel('Time')  
ax\_a.set\_ylabel('Current (A)')  
ax\_a.set\_title('Real-time Current')  
ax\_a.legend()  
  
# ────── CSV 建立 ──────  
csvfile = open('rs232\_monitor\_all.csv', mode='w', newline='')  
csvwriter = csv.writer(csvfile)  
csvwriter.writerow(['Timestamp', 'RPM\_51', 'V\_52', 'A\_52',  
 'RPM\_53', 'V\_54', 'A\_54',  
 'RPM\_55', 'V\_56', 'A\_56','set-rpm', 'set-voltage','set-current'])  
  
# ────── 資料讀取 ──────  
def read\_rpm(slave\_id):  
 disp = client.read\_holding\_registers(address=0x0040, count=2, slave=slave\_id)  
 scale = client.read\_holding\_registers(address=0x002C, count=2, slave=slave\_id)  
 if disp.isError() or scale.isError():  
 return None  
 raw\_disp = (disp.registers[0] << 16) + disp.registers[1]  
 raw\_scale = (scale.registers[0] << 16) + scale.registers[1]  
 if raw\_disp & 0x80000000:  
 raw\_disp -= 0x100000000  
 if raw\_scale & 0x80000000:  
 raw\_scale -= 0x100000000  
 return raw\_disp \* (raw\_scale / 10000)  
  
def read\_signed\_16(slave, address):  
 res = client.read\_holding\_registers(address=address, count=1, slave=slave)  
 if res.isError():  
 return None  
 val = res.registers[0]  
 return val - 0x10000 if val & 0x8000 else val  
  
# ────── 更新圖表 ──────  
def update(frame):  
 now = datetime.now().strftime("%H:%M:%S")  
 times.append(now)  
  
 r51 = read\_rpm(51)  
 r53 = read\_rpm(53)  
 r55 = read\_rpm(55)  
 v52 = read\_signed\_16(52, 0x0046)  
 a52 = read\_signed\_16(52, 0x0048)  
 v54 = read\_signed\_16(54, 0x0046)  
 a54 = read\_signed\_16(54, 0x0048)  
 v56 = read\_signed\_16(56, 0x0046)  
 a56 = read\_signed\_16(56, 0x0048)  
  
 if None in [r51, r53, r55, v52, a52, v54, a54, v56, a56]:  
 print("❌ 通訊錯誤")  
 return  
  
 v52, v54, v56 = v52 / 100, v54 / 100, v56 / 100  
 a52, a54, a56 = a52 / 1000, a54 / 1000, a56 / 1000  
  
 rpm\_51.append(r51)  
 rpm\_53.append(r53)  
 rpm\_55.append(r55)  
 v\_52.append(v52)  
 v\_54.append(v54)  
 v\_56.append(v56)  
 a\_52.append(a52)  
 a\_54.append(a54)  
 a\_56.append(a56)  
  
 for lst in [times, rpm\_51, rpm\_53, rpm\_55, v\_52, v\_54, v\_56, a\_52, a\_54, a\_56]:  
 lst[:] = lst[-60:]  
  
 line\_rpm\_51.set\_data(times, rpm\_51)  
 line\_rpm\_53.set\_data(times, rpm\_53)  
 line\_rpm\_55.set\_data(times, rpm\_55)  
 ax\_rpm.set\_xlim(times[0], times[-1])  
 ax\_rpm.set\_ylim(  
 min(rpm\_51 + rpm\_53 + rpm\_55 + [thresholds['rpm']]) \* 0.9,  
 max(rpm\_51 + rpm\_53 + rpm\_55 + [thresholds['rpm']]) \* 1.1  
 )  
  
 line\_v\_52.set\_data(times, v\_52)  
 line\_v\_54.set\_data(times, v\_54)  
 line\_v\_56.set\_data(times, v\_56)  
 ax\_v.set\_xlim(times[0], times[-1])  
 ax\_v.set\_ylim(  
 min(v\_52 + v\_54 + v\_56 + [thresholds['voltage']]) \* 0.9,  
 max(v\_52 + v\_54 + v\_56 + [thresholds['voltage']]) \* 1.1  
 )  
  
 line\_a\_52.set\_data(times, a\_52)  
 line\_a\_54.set\_data(times, a\_54)  
 line\_a\_56.set\_data(times, a\_56)  
 ax\_a.set\_xlim(times[0], times[-1])  
 ax\_a.set\_ylim(  
 min(a\_52 + a\_54 + a\_56 + [thresholds['current']]) \* 0.9,  
 max(a\_52 + a\_54 + a\_56 + [thresholds['current']]) \* 1.1  
 )  
  
 csvwriter.writerow([now, r51, v52, a52, r53, v54, a54, r55, v56, a56,thresholds["rpm"],thresholds["voltage"],thresholds["current"]])  
 print(f"{now} ➤ RPM: {r51:.1f}, {r53:.1f}, {r55:.1f} | V: {v52:.2f}, {v54:.2f}, {v56:.2f} | A: {a52:.3f}, {a54:.3f}, {a56:.3f}")  
  
# ────── 主流程執行 ──────  
try:  
 if not client.connect():  
 print(f"❌ 無法連接 {com\_port}")  
 else:  
 print(f"✅ 成功連線 {com\_port}，開始資料更新...")  
 ani = FuncAnimation(fig, update, interval=1000)  
 plt.tight\_layout()  
 plt.show()  
except KeyboardInterrupt:  
 print("🛑 手動中止")  
finally:  
 client.close()  
 csvfile.close()

一張含有 文字, 字型, 螢幕擷取畫面, 行 的圖片

AI 產生的內容可能不正確。

一張含有 文字, 螢幕擷取畫面, 字型, 數字 的圖片

AI 產生的內容可能不正確。 一張含有 文字, 螢幕擷取畫面, 數字, 字型 的圖片

AI 產生的內容可能不正確。

一張含有 文字, 螢幕擷取畫面, 數字, 軟體 的圖片

AI 產生的內容可能不正確。

一張含有 文字, 圖表, 行, 螢幕擷取畫面 的圖片

AI 產生的內容可能不正確。

關掉程式, 就會產生出數據

一張含有 文字, 數字, 文件, 黑與白 的圖片

AI 產生的內容可能不正確。

程式分析:

這份程式碼是 **RS232 通訊即時監控系統**，透過 Python + tkinter + PyModbus + matplotlib 建構而成，具有：

**✅ 核心功能分析**

| **功能區塊** | **說明** |
| --- | --- |
| 🧭 COM 埠選擇 | 使用 tkinter 的下拉選單介面選擇可用串口 |
| ⚠️ 警告值輸入 | GUI 輸入轉速、電壓、電流的警告門檻 |
| 🧠 Modbus 讀取 | 支援 3 組設備（51/52、53/54、55/56），從特定暫存器取值 |
| 📈 即時動畫圖表 | 每秒更新一次三張圖表（轉速、電壓、電流） |
| 📊 警告線 | 各圖表都有對應的紅色虛線，標示警告門檻 |
| 💾 CSV 紀錄 | 每筆數據皆儲存至 rs232\_monitor\_all.csv，含時間與設定警戒值 |
| 🧪 串接格式 | MODBUS RTU 通訊，19200 baud, 8 data bit, N parity, 2 stop bits |

**📌 感應器與地址分配對應**

| **裝置地址** | **資料型態** | **地址說明** |
| --- | --- | --- |
| 51, 53, 55 | 轉速 | 0x0040 (2 個暫存器) 與 0x002C (scale) |
| 52, 54, 56 | 電壓/電流 | 0x0046 (Voltage), 0x0048 (Current)，16-bit 有號數值 |

**🧠 模組分析亮點**

| **模組** | **重點說明** |
| --- | --- |
| tkinter | 使用兩階段 GUI：選擇串口與設定警告門檻 |
| pymodbus.client.ModbusSerialClient | 負責 RS232 的 slave 資料交換 |
| matplotlib.animation.FuncAnimation | 用來實現每秒自動更新圖表 |
| csv | 紀錄每筆時間與讀值資料，可供離線分析 |

**📦 1. 匯入模組**

python

複製編輯

import time

import csv

from datetime import datetime

import serial.tools.list\_ports

from pymodbus.client import ModbusSerialClient

import matplotlib.pyplot as plt

from matplotlib.animation import FuncAnimation

import tkinter as tk

from tkinter import ttk

| **模組** | **功能說明** |
| --- | --- |
| time, datetime, csv | 控制時間與寫入 CSV 檔案 |
| serial.tools.list\_ports | 偵測目前可用的 COM port |
| pymodbus.client | 操作 MODBUS RTU 串列通訊協定 |
| matplotlib.pyplot | 畫即時圖表 |
| FuncAnimation | 定時更新圖表用 |
| tkinter | 建立圖形化介面（下拉選單與警告值輸入） |

**🎛️ 2. GUI 選擇 COM port**

python

複製編輯

def select\_com\_port():

...

return selected\_port.get()

* 顯示下拉式選單列出可用 COM port。
* 使用者選完按「開始」，程式才正式執行。
* 若無可用埠，則結束程式。

**⚠️ 3. 輸入警告閥值**

python

複製編輯

def get\_warning\_thresholds():

...

return thresholds

* 使用第二個 tkinter 小視窗要求使用者輸入：
  + ⚠️ 轉速警告值（rpm）
  + ⚠️ 電壓警告值（voltage）
  + ⚠️ 電流警告值（current）
* 若輸入非數值會提示錯誤。

**⚙️ 4. 初始化裝置連線設定**

python

複製編輯

com\_port = select\_com\_port()

thresholds = get\_warning\_thresholds()

* 執行上方兩個 GUI 功能
* 接著建立 MODBUS Serial Client：

python

複製編輯

client = ModbusSerialClient(

port=com\_port,

baudrate=19200,

parity='N',

stopbits=2,

bytesize=8,

timeout=1

)

* 通訊參數：RS232 通常設定為 19200, 8N2

**📈 5. 初始化圖表與資料儲存變數**

python

複製編輯

times, rpm\_51, rpm\_53, rpm\_55 = [], [], [], []

v\_52, v\_54, v\_56 = [], [], []

a\_52, a\_54, a\_56 = [], [], []

* 建立資料用 list，用於畫圖與 CSV

python

複製編輯

fig, (ax\_rpm, ax\_v, ax\_a) = plt.subplots(3, 1, figsize=(12, 9), sharex=True)

* 三張圖分別代表：轉速、電壓、電流（共用 x 軸）

python

複製編輯

line\_rpm\_51, = ax\_rpm.plot(...)

line\_v\_52, = ax\_v.plot(...)

* 分別繪製三組設備的三種資料（共 9 條線）

python

複製編輯

ax\_rpm.axhline(...)

* 每張圖畫上紅色虛線警告門檻（來自 GUI 輸入）

**💾 6. 建立 CSV 檔案**

python

複製編輯

csvfile = open('rs232\_monitor\_all.csv', mode='w', newline='')

csvwriter = csv.writer(csvfile)

csvwriter.writerow([...])

* 開啟檔案並寫入標題欄（包含所有設備資料與設定警告值）

**📡 7. MODBUS 通訊資料讀取函式**

python

複製編輯

def read\_rpm(slave\_id):

...

return rpm\_value

* 讀兩組 32bit 整數（表示 RPM 與比例尺），並做乘法轉換為實際轉速。

python

複製編輯

def read\_signed\_16(slave, address):

...

return val

* 讀 16-bit 的 signed 整數（表示電壓或電流）

**🔄 8. update() 圖表更新主邏輯**

python

複製編輯

def update(frame):

...

1. 每秒執行一次（由 FuncAnimation 呼叫）
2. 讀取所有資料（RPM + Voltage + Current × 3 組）
3. 判斷若有任何錯誤 None 則跳過該次更新
4. 將原始值做單位轉換（V 除以 100, A 除以 1000）
5. 更新清單（最多保留 60 筆）
6. 重設圖表資料與 y 軸範圍
7. 寫入 CSV 並列印到 console log

**🚀 9. 主程式執行與關閉**

python

複製編輯

try:

if not client.connect():

print("❌ 無法連接...")

else:

...

plt.show()

except KeyboardInterrupt:

print("🛑 手動中止")

finally:

client.close()

csvfile.close()

* 嘗試連接 COM 埠
* 若成功則開始圖表更新與資料儲存
* 支援 Ctrl+C 中斷與自動關閉裝置

**✅ 總結一句話**

這份程式是**極完整的 RS232 + Modbus 通訊圖表即時監控與資料記錄系統**，具備：

* 多 COM 裝置支援
* 警告值輸入與圖表虛線提示
* 多裝置多欄位繪圖
* 自動儲存資料為 .csv
* GUI 操作便利，適合現場工程師部署