????????????

> 撰寫人 : 范紹捷 /動力組 /5~9代

??

- 簡介
- 安裝需求
- 基本使用
- 自定義開發

??

這是一個基於 ttkbootstrap 的 uart監控基礎框架，提供了基本的串口通信界面和日誌記錄功能。

????

```
pip install ttkbootstrap
pip install pyserial
```

????

?????

```
from base_monitor import BaseMonitor

app = BaseMonitor(title="監控程序")
app.run()
```

????

初始化時可設置的參數:

- title：窗口標題（默認: "監視器 "）
- size：窗口大小（默認: "500x600"）
- theme：界面主題（默認: "darkly"）

?????

```
darkly,cosmo,flatly,litera,minty,lumen,sandstone,yeti,pulse,united,morph
```

?????

?????

```
class CustomMonitor(BaseMonitor):
    def __init__(self):
        super().__init__(title="自定義監控")
```

??????

1. create_data_panel：自定義數據顯示面板

```
def create_data_panel(self):
    data_frame = ttk.LabelFrame(self.main_container, text="自定義數據", padding=10)
    data_frame.pack(fill=X, pady=10)
    # 添加自己的顯示元件
```

2. start_monitor：實現數據接收邏輯

```
def start_monitor(self):
    try:
        self.serial = serial.Serial(
            port=self.port_var.get(),
            baudrate=int(self.baud_var.get()),
            timeout=1
        )
        # 添加數據處理邏輯
```

```
    except Exception as e:
        self.log_message(f"啟動失敗: {str(e)}")
```

3. stop_monitor : 實現停止邏輯

```
def stop_monitor(self):
    if self.serial:
        self.serial.close()
        self.serial = None
```

????

使用 log_message 方法記錄日誌:

```
self.log_message("自定義消息")
```

????

1. 串口連接前請確保選擇了正確的端口和波特率
2. 程序退出時會自動關閉串口連接
3. 建議在子類中實現異常處理機制

????

```
import ttkbootstrap as ttk
from ttkbootstrap.constants import *
import threading
import serial
import serial.tools.list_ports

class BaseMonitor:
    def __init__(self, title="監視器", size="500x600", theme="darkly"):
        self.root = ttk.Window(title=title, themename=theme, resizable=(False, False))
        self.root.geometry(size)

        # 狀態變量
        self.is_running = False
```

```
        # 創建界面
        self.create_widgets()

    def create_widgets(self):
        # 主容器
        self.main_container = ttk.Frame(self.root, padding=10)
        self.main_container.pack(fill=BOTH, expand=YES)

        # 控制區域
        self.create_control_panel()

        # 數據顯示區域
        self.create_data_panel()

        # 日誌區域
        self.create_log_panel()

    def create_control_panel(self):
        """控制面板 - 可在子類中重寫"""
        control_frame = ttk.LabelFrame(self.main_container, text="控制面板", padding=10)
        control_frame.pack(fill=X, pady=5)

        # 端口設置
        port_frame = ttk.Frame(control_frame)
        port_frame.pack(fill=X, pady=5)
        ttk.Label(port_frame, text="端口:").pack(side=LEFT, padx=5)
        self.port_var = ttk.StringVar()
        self.port_combo = ttk.Combobox(port_frame, textvariable=self.port_var)
        self.port_combo.pack(side=LEFT, fill=X, expand=YES)

        # 刷新端口按鈕
        ttk.Button(
            port_frame,
            text="刷新",
            command=self.refresh_ports,
            style="info.TButton",
            width=8
        ).pack(side=LEFT, padx=5)

        # 波特率設置
        baud_frame = ttk.Frame(control_frame)
```

```
        baud_frame.pack(fill=X, pady=5)
        ttk.Label(baud_frame, text="波特率:").pack(side=LEFT, padx=5)
        self.baud_var = ttk.StringVar(value="9600")
        baud_choices = ['9600', '19200', '38400', '57600', '115200']
        ttk.Combobox(baud_frame, textvariable=self.baud_var,
values=baud_choices).pack(side=LEFT, fill=X, expand=YES)

        # 控制按鈕
        self.control_btn = ttk.Button(
            control_frame,
            text="連接",
            command=self.toggle_running,
            style="primary.TButton"
        )
        self.control_btn.pack(pady=10)

        # 初始化端口列表
        self.refresh_ports()

    def refresh_ports(self):
        """更新可用的串口列表"""
        ports = [port.device for port in serial.tools.list_ports.comports()]
        self.port_combo['values'] = ports
        if ports:
            self.port_var.set(ports[0])
        else:
            self.port_var.set('')
            self.log_message("未檢測到可用的串口")

    def toggle_running(self):
        """切換運行狀態"""
        self.is_running = not self.is_running
        if self.is_running:
            self.control_btn.configure(text="斷開", style="danger.TButton")
            self.status_label.configure(text="已連接", style="success.TLabel")
            self.log_message("串口連接成功")
            self.start_monitor()
        else:
            self.control_btn.configure(text="連接", style="primary.TButton")
            self.status_label.configure(text="未連接", style="danger.TLabel")
            self.log_message("串口已斷開")
```

```
            self.stop_monitor()

    def create_data_panel(self):
        """數據顯示面板 - 可在子類中重寫"""
        data_frame = ttk.LabelFrame(self.main_container, text="數據顯示", padding=10)
        data_frame.pack(fill=X, pady=10)

        # 狀態顯示
        self.status_label = ttk.Label(
            data_frame,
            text="未運行",
            style="danger.TLabel"
        )
        self.status_label.pack(pady=5)

    def create_log_panel(self):
        """日誌面板"""
        log_frame = ttk.LabelFrame(self.main_container, text="系統日誌", padding=10)
        log_frame.pack(fill=BOTH, expand=YES, pady=5)

        self.log_text = ttk.Text(log_frame, height=10, width=40)
        self.log_text.pack(fill=BOTH, expand=YES)

        scrollbar = ttk.Scrollbar(log_frame, orient="vertical", command=self.log_text.yview)
        scrollbar.pack(side=RIGHT, fill=Y)
        self.log_text.configure(yscrollcommand=scrollbar.set)

    def toggle_running(self):
        """切換運行狀態"""
        self.is_running = not self.is_running
        if self.is_running:
            self.control_btn.configure(text="停止", style="danger.TButton")
            self.status_label.configure(text="運行中", style="success.TLabel")
            self.log_message("系統啟動")
            self.start_monitor()
        else:
            self.control_btn.configure(text="開始", style="primary.TButton")
            self.status_label.configure(text="未運行", style="danger.TLabel")
            self.log_message("系統停止")
            self.stop_monitor()
```

```
    def start_monitor(self):
        """啟動監控 - 在子類中實現"""
        pass

    def stop_monitor(self):
        """停止監控 - 在子類中實現"""
        pass

    def log_message(self, message):
        """記錄日誌消息"""
        import time
        self.log_text.insert(END, f"{time.strftime('%H:%M:%S')} - {message}\n")
        self.log_text.see(END)

    def run(self):
        """運行程序"""
        self.root.mainloop()

if __name__ == "__main__":
    # 基礎使用示例
    app = BaseMonitor(title="基礎監視器")
    app.run()
```

Revision #3
Created 2026-04-01 02:06:17 UTC by TaipeiTechRacing
Updated 2026-04-11 14:38:31 UTC by AI Agent