# PortMonitor

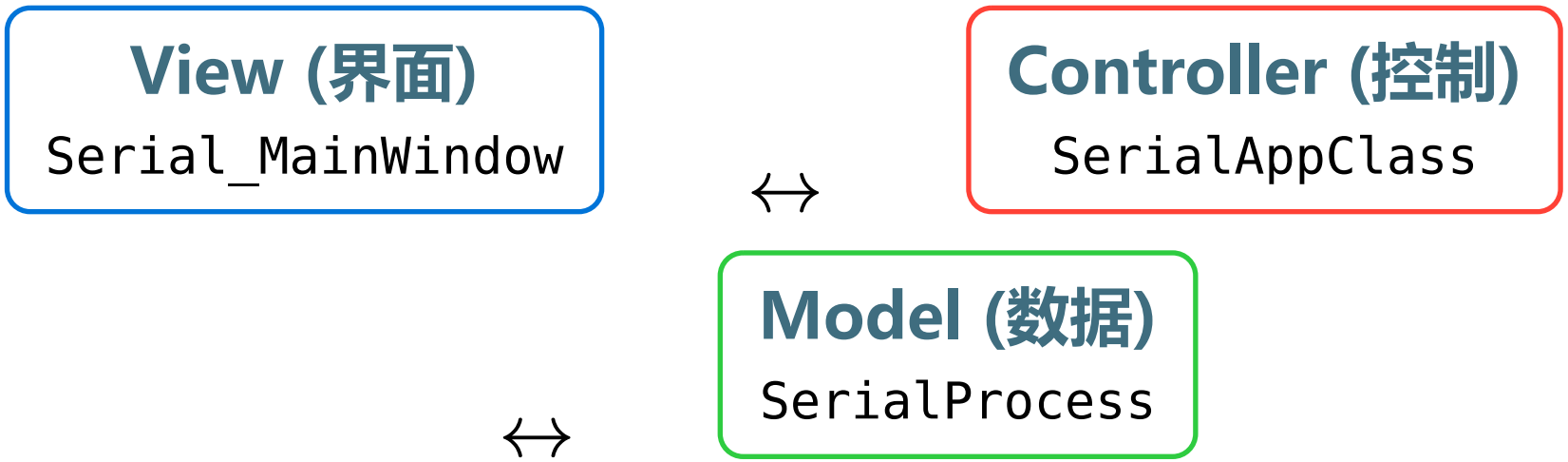
## 深度架构剖析与优势说明

MVC 架构 | PyQt5 | 工业级设计

2026-01-17

# 1. 架构概览：精密的 MVC 模式

本项目采用 **Model-View-Controller** 架构，结合 PyQt5 **信号与槽**机制。




**设计理念**: 高内聚、低耦合，专业软件工程素养。

# Model (模型层)

## 核心逻辑与数据

- **主要组件**:
  - `SerialProcess.py`: 串口引擎
  - `config_manager.py`: 配置管家
- **核心职责**:
  - 封装底层 `QSerialPort` 操作
  - 负责应用状态的持久化 (Load/Save)

- **架构亮点**:
  - **完全独立于 UI**
  - 纯逻辑封装，可轻松移植到自动化脚本或无头环境

# View (视图层)

## 用户交互界面 - “Passive View”

- **主要组件**: `Serial_MainWindow.ui / .py`
- **核心职责**:
  - 定义布局、样式、控件属性
  - **不**包含复杂的业务逻辑代码
- **架构亮点**:
  - 代码由 UI 设计器自动生成

- 与逻辑手写代码物理分离
- 界面调整与业务逻辑互不干扰

# Controller (控制器层)

## 业务逻辑的中枢神经

- **主要组件**: `app_SerialWindows.py`
- **核心职责**:
  - 程序的“胶水”：初始化 View 和 Model
  - **Signal & Slot** 的调度中心
- **工作流**:
  1. Model 发出 `data_received` 信号

2. Controller 捕获信号
3. Controller 调用 View 的 `append_to_receive` 刷新界面

# 2. 核心优势 - A. 极致解耦 (Decoupling)

**传统初学者写法**

- `serial.read()` 阻塞在 GUI 线程
- 逻辑散落在按钮点击事件中
- 难以维护，无法复用

**PortMonitor 架构**

- `SerialProcess` 作为独立对象
- 仅通过信号通讯
- **极大利于单元测试 (Unit Testing)**

# 核心优势 - B. 响应式与异步 I/O

## 不再有 “界面假死”

- **机制**: 基于 PyQt5 强大的事件循环 (Event Loop)
- **表现**:
  - 无论串口波特率多高 (115200+ bps)
  - 无论数据量多大
  - 界面始终保持**丝滑流畅**

- **原理**: 硬件操作与界面绘制分离，数据到达 → 信号触发 → 异步更新

# 核心优势 - C. 像素级配置还原

## JSONConfigManager 的细节体验

- **“修改即保存” 策略**:
  - 改变波特率 → Auto Save
  - 调整窗口大小 → Auto Save
  - 勾选 Hex 显示 → Auto Save
- **用户价值**:
  - 程序重启后**完全还原**上次工作状态

- 这是区分 **Demo** 与 **专业工具** 的关键细节

# 核心优势 - D. 架构扩展性

## 为未来而设计

- **类型安全**: 广泛使用的 Python Type Hints
  - `window_manager: 'WindowManagerClass'`
- **窗口管理**:
  - `QStackedWidget` 架构
  - 轻松扩展多页面应用
  - 预留了 “波形分析”、“网络调试” 等模块接口

# 3. 总结

**一份优秀的 PyQt5 架构教科书**

- Model-View-Controller 完美实践
- 兼顾 高稳定性 与 高性能

- 清晰的代码结构赋予项目无限生命力