# PLC仿真器说明文档

1. **软件框架**

软件在OpenPLC Editor的基础上进行改造，去除了wxPython相关的前端代码，修改了部分程序结构，以实现B/S架构。软件整体框架如图1所示，后端和Runtime部分采用Python语言实现。

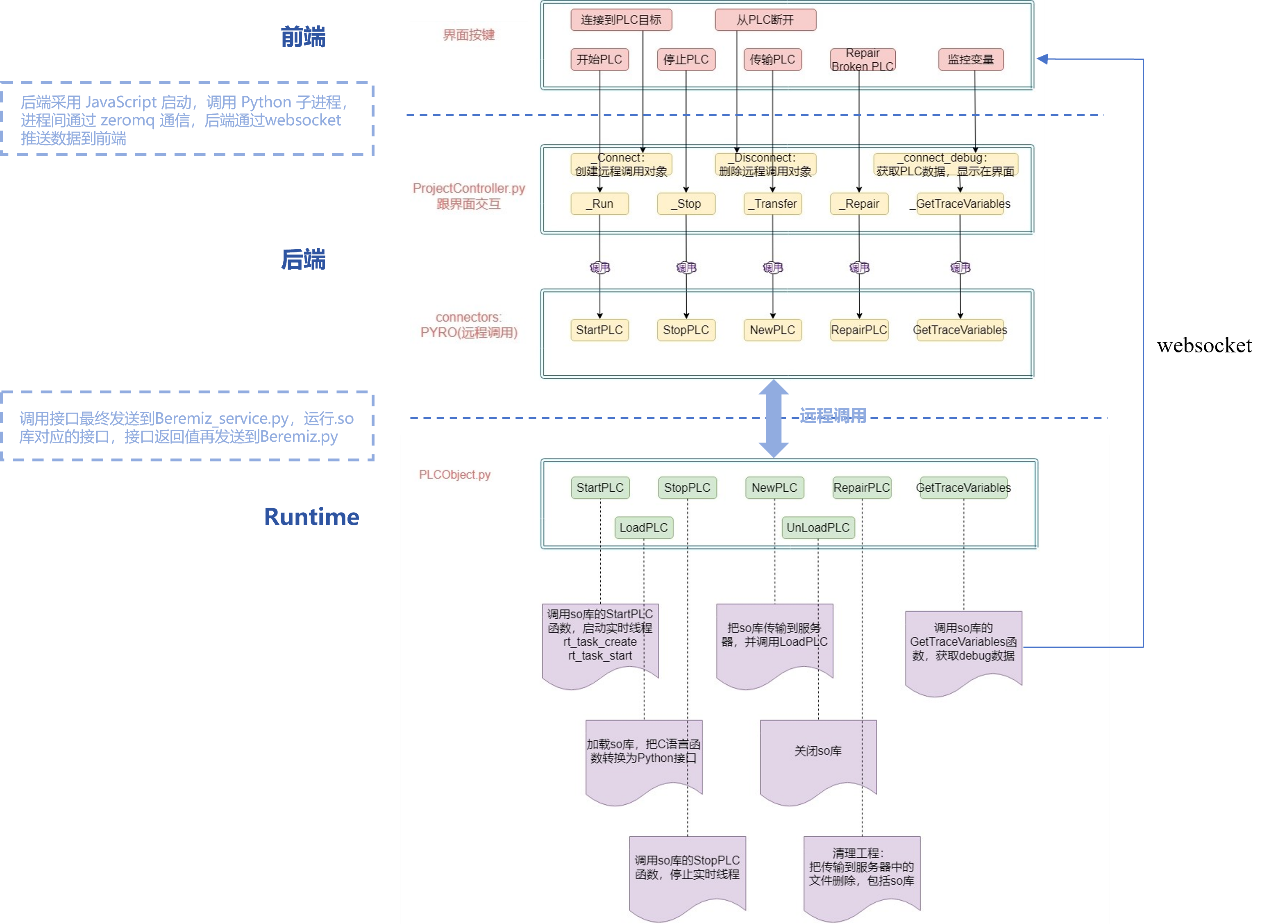
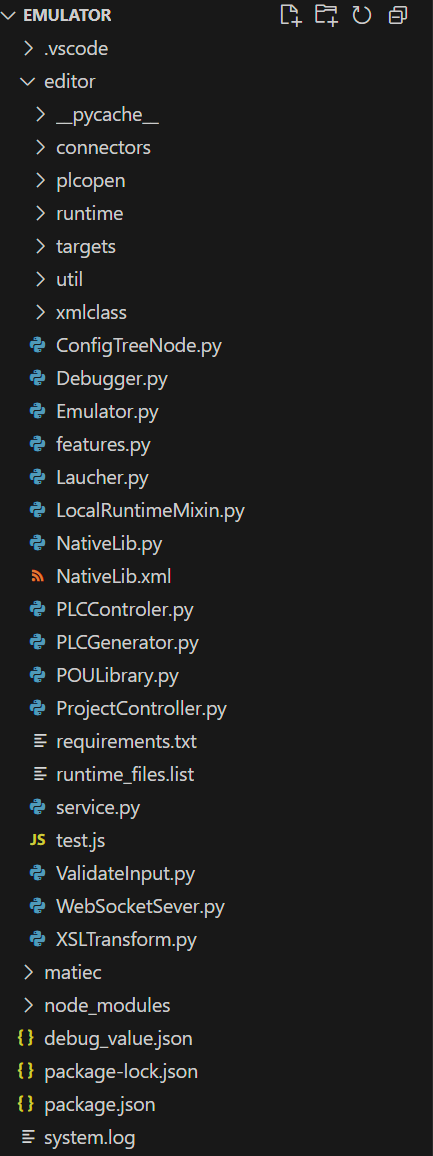


图1 软件整体框架

后端采用 nodejs 运行程序，其监听前端请求，启动python子进程，并向python子进程发送命令。Python子进程持续监听nodejs进程的消息，执行PLC程序的编译、运行、停止以及变量的订阅、取消订阅、强制、释放等操作。nodejs和python进程之间的通信采用zeromq库进行，python进程通过websocket将订阅的变量的值持续推送到前端。

1. **程序目录**

程序目录如图2所示，其中test.js是javascript后端程序示例，展示了如何在js中调用python子进程以及与子进程进行通信。Laucher.py是Python进程的启动文件，可通过该文件对Emulator进行控制。ProjectorController.py是控制文件，通过远程调用执行runtime中的方法，PLC程序的编译、运行、停止以及变量的订阅、取消订阅、强制、释放都在该文件中实现。



**用于变量订阅、取消订阅、强制、释放**

**PLC仿真器，用于操作本地运行时**

**启动文件**

**本地运行时**

**主要修改的文件**

**后端程序示例**

图2 程序目录

1. **前端**

前端需要实现变量窗口、调试窗口、控制台窗口，相关需求在“前端.pptx”文件中介绍。

1. **后端**
2. 程序启动

后端采用js调用python的方式运行仿真，js程序根据前端用户操作，启动python子进程或向python子进程发送相关的操作指令。

在js文件中，需要指定当前PLC项目的路径，启动python子进程，创建ZeroMQ客户端并与python子进程中的ZeroMQ服务器建立连接。 test.js文件展示了该用法，可通过nodejs editor/test.js运行程序，该示例持续监听用户的命令行输入，根据用户输入以json的格式向python子进程发送相应的操作请求以及参数。

在Laucher.py文件中，需要持续监听ZeroMQ客户端发送的操作请求以及参数，根据请求的类型执行相应的操作，包括运行、编译、停止、订阅、取消订阅、强制、释放等。进行调试时，可以直接运行Laucher.py文件，通过命令行输入来进行测试（详见Laucher.py注释部分），运行方式为“python editor/Laucher.py <文件名>”，如“python editor/Laucher.py 'E:\Projects\Blink'”。

1. 变量的读写

OpenPLC Editor在运行PLC程序时，会启动两个定时器，即StatusTimer和DispatchDebugValuesTimer，分别用于定时获取PLC状态并更新方法的使能状态以及定时读取和更新PLC变量的值。这两个定时器基于wx.Timer类创建，可以在主线程中调用绑定的事件方法。本项目中需要去除wxPython前端架构，因此需要用两个线程来代替原有的两个定时器，但这会导致pyro5的所有权问题，这是因为与runtime建立的pyro5连接是在主线程中创建的，但是使用子线程创建的定时器的回调是在子线程中执行的，pyro5连接不是线程安全的，在子线程中使用主线程中的pyro5连接会导致所有权问题。

为了解决所有权问题，在ProjectController类中添加了一个本地定时器，该定时器可以实现添加单次或者重复触发的定时器，以及删除已有的定时器，所有的定时器回调函数在同一个子线程中执行，在该子线程中创建了一个新的pyro5代理，回调函数使用该pyro5代理进行远程调用，从而可以实现定时更新PLC状态以及变量的值，并避免多线程导致的所有权问题。

1. 数据传输

将订阅的数据传输到前端可以通过文件读写的方式，即python中的数据分发事件回调将数据值写入本地文件中，在js中定时读取文件中的数据发送到前端，但这会导致文件读写冲突以及数据一致性问题。考虑到服务器需要主动向前端推送数据，可以采用websocket进行数据传输，为此封装了一个WebSocketServer类（文件WebSocketServer.py），该类可向websocket客户端（前端）主动传输订阅的变量的值。但测试中发现，前端数据更新缓慢，比本地文件读写效率还要低，这是由于服务器推送数据太快（约20ms一次），导致前端来不及渲染，为此更改了WebSocketServer的实现方式，改为一定时间间隔推送一次。

前端中需要实现websocket客户端进行数据接收，此处提供了一个用于测试的简单客户端，可通过“http-server -p 8080”运行该客户端，通过访问<http://127.0.0.1:8080>可以查看订阅的变量，变量以json的格式传输，count字段为聚合传输的数据的数目。为了提高传输的效率，后续还可以采用protobuf等更高效的数据格式。

