基于Python的实验控制软件框架开发

报告人： 夏守腾[1]

导　师： 张银鸿[1]　钱　森[1]

1. 中国科学院高能物理研究所

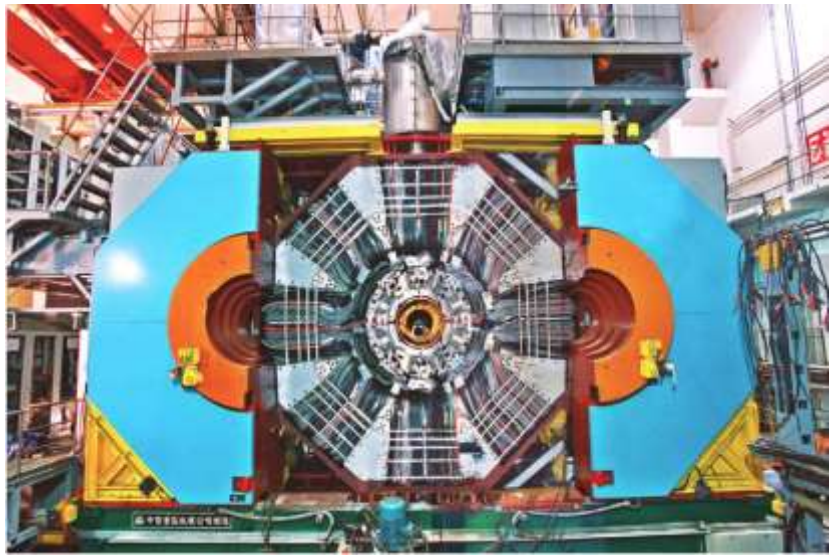
2023年7月11日

CONTENT

研究背景及意义

1. 研究背景及意义

不同的物理实验对控制软件的开发有着不同的需求，通常，控制软件的开发是针对特定的实验，在我国的大型物理实验中，有专门的慢控制软件，如BESIII慢控制系统，HEPS的Pyapas软件架构，一些小的实验甚至在没有控制软件的条件下。因此，一个可用于所有物理实验的控制软件框架开发是非常有必要的。



BES外观图

HEPS外观图

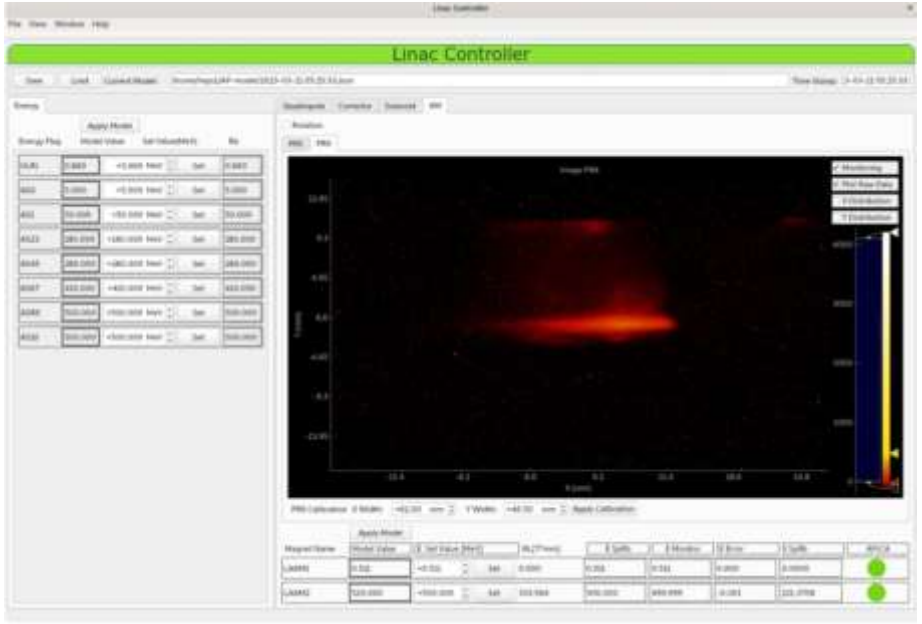
设计目标

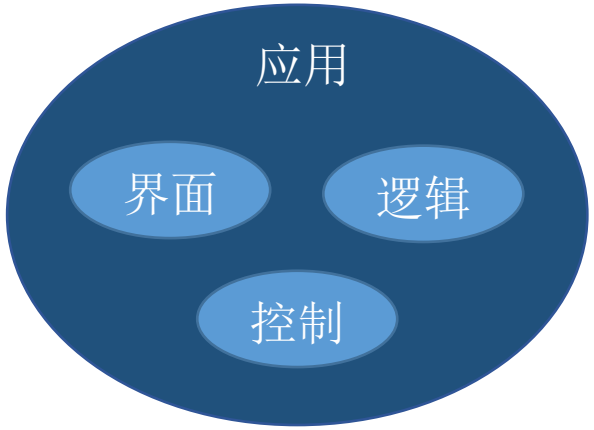
- 开源
- 适用于大数据量物理实验或实验室实验
- 简单、可靠

1. 研究背景及意义

Epics

目前的大型物理实验控制系统大多基于Epics，比如HEPS，合肥光源、上海光源。但其驱动开发难度较高，且在大型物理实验下多个同种设备的变量配置非常复杂。



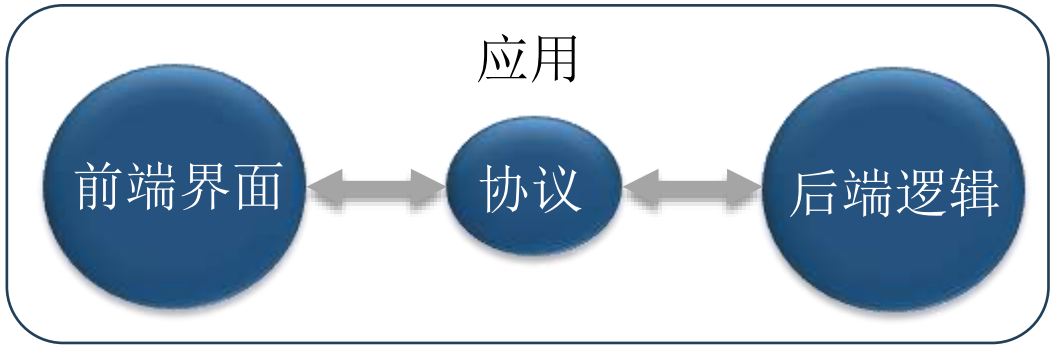


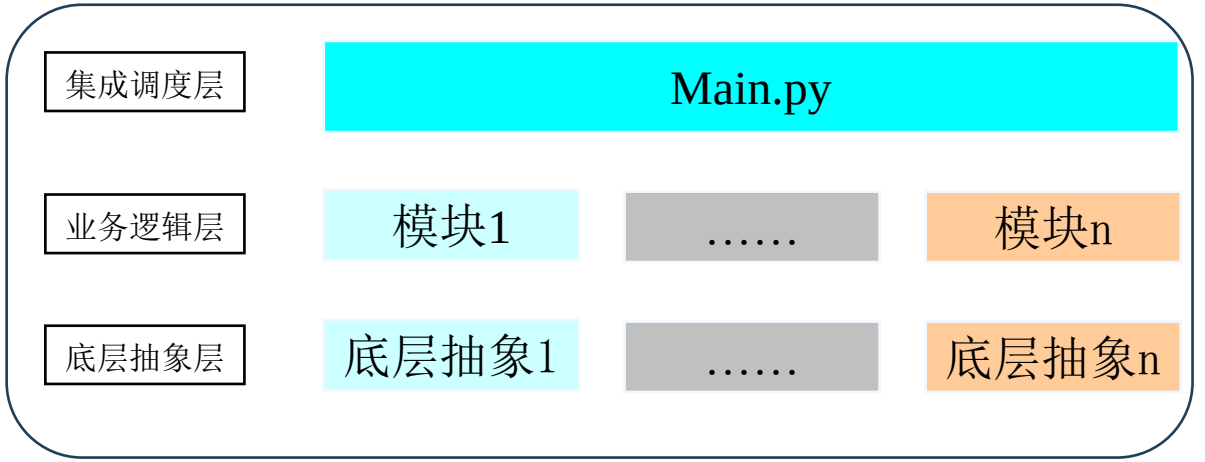
LabView

- 容器化部署困难
- 商业化软件
- 跨平台复杂

1. 研究背景及意义

◘ 基于Python的实验控制软件框架





- 基于**Python**的软件设计可以实现前后端分离，并且通过协议可以实现控制与显示的分离。
- 代码高度模块化，快速开发、构建应用。
- 利用Python的跨平台性，与**容器**技术可以快速、方便的维护应用。

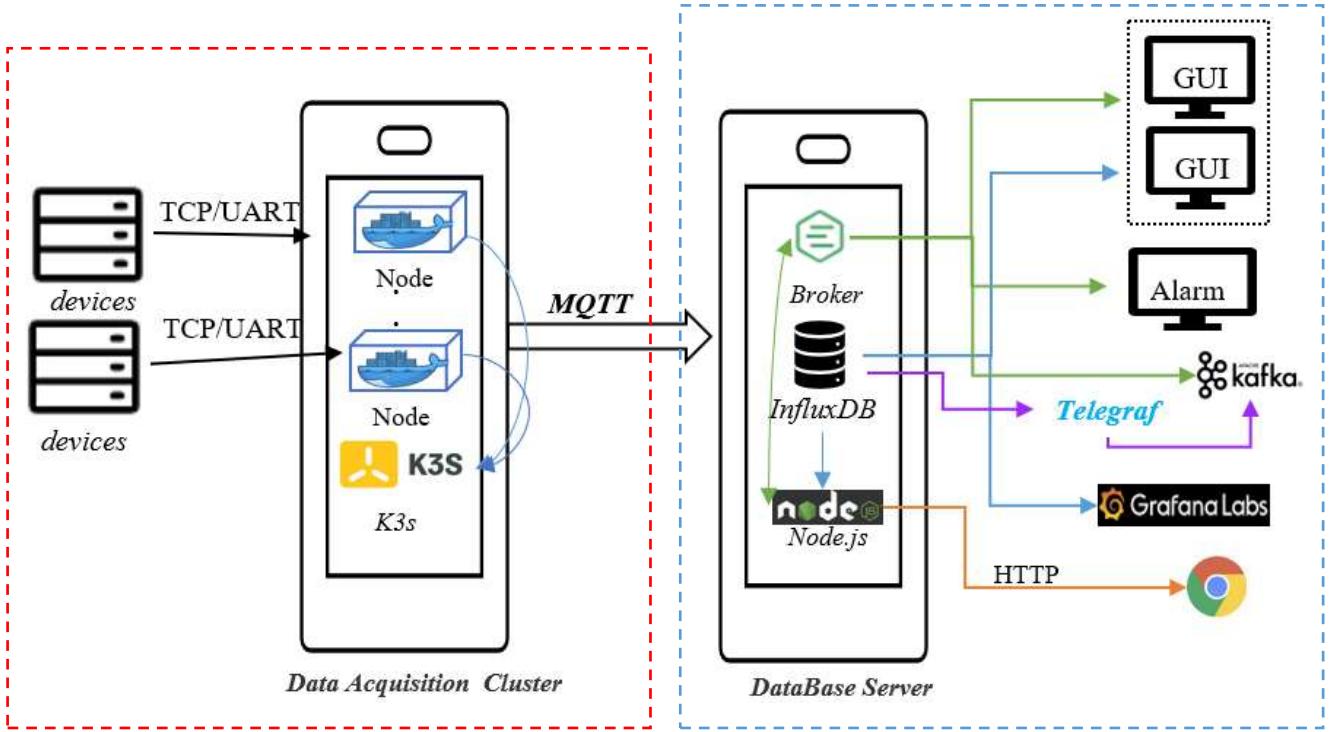
2 技术路线

2. 技术路线——总体架构

◆总体架构图

基于Arm平台的部署

- IoT
- MQTT
- Kubernetes



基于瘦客户端的数据可视化

- 低带宽
- 大规模数据
- 集中管理

◘ 分布式系统

- 将任务和数据分布在多个节点上，可以提高系统的可靠性。
- 并行处理和负载均衡来提高系统的性能。
- 良好的可扩展性，可以根据需求方便地增加或减少节点。

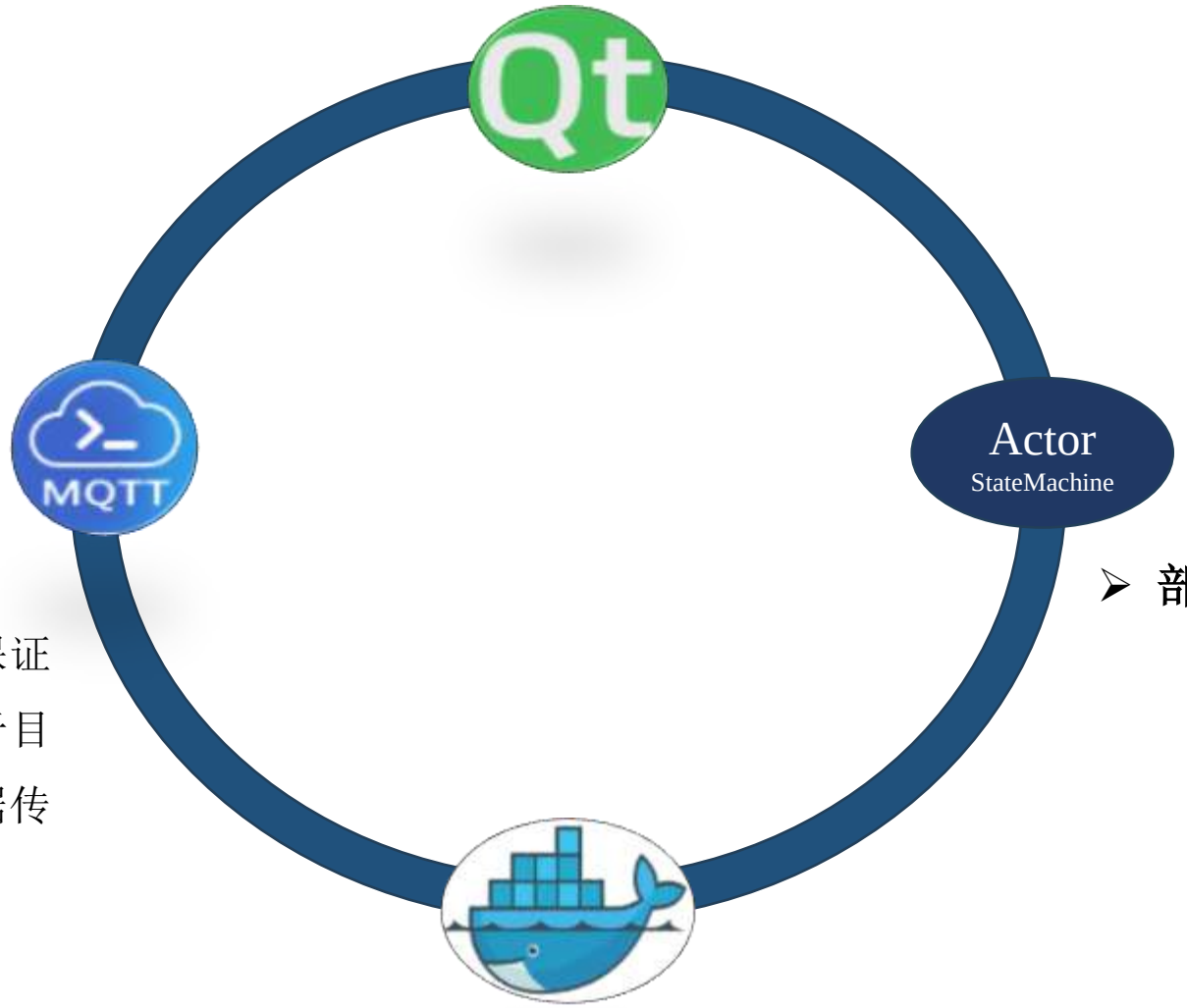
2. 技术路线——实现方法

- GUI--PyQt5

  PyQt5是基于QT、功能强大的第三方库。其信号与槽机制以及丰富控件丰富，可以满足软件设计的需要。

- **采集**

  对于数据的采集使用Actor模型的高并发性与可扩展性，避免出现死锁、竞态条件等问题，提高系统的容错性与数据采集效率。使用StateMachine管理Actor的生命周期。



- **MQTT**

  MQTT协议是以数据为中心，保证数据的可靠性与安全性，适用于目前实验室的大型物理设备的数据传输。

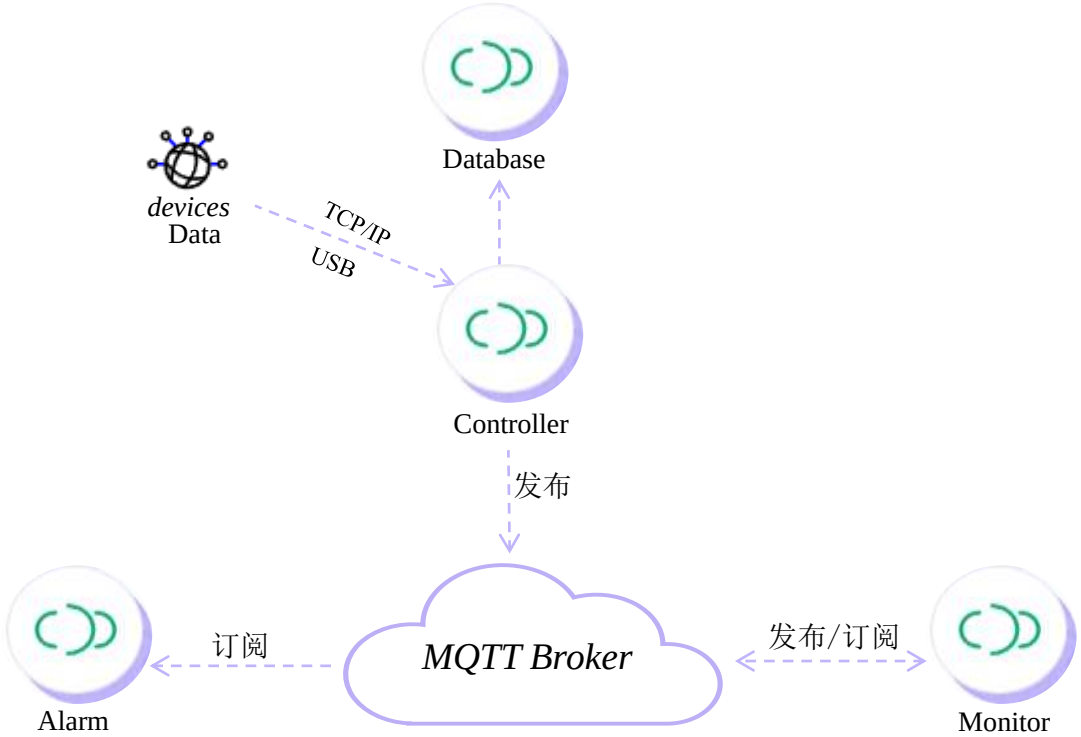
- **部署—Docker**

  Docker容器化技术能够将应用程序与基础架构分开，可以快速交付软件，方便的对软件运行与维护。

2. 技术路线——软件设计原理

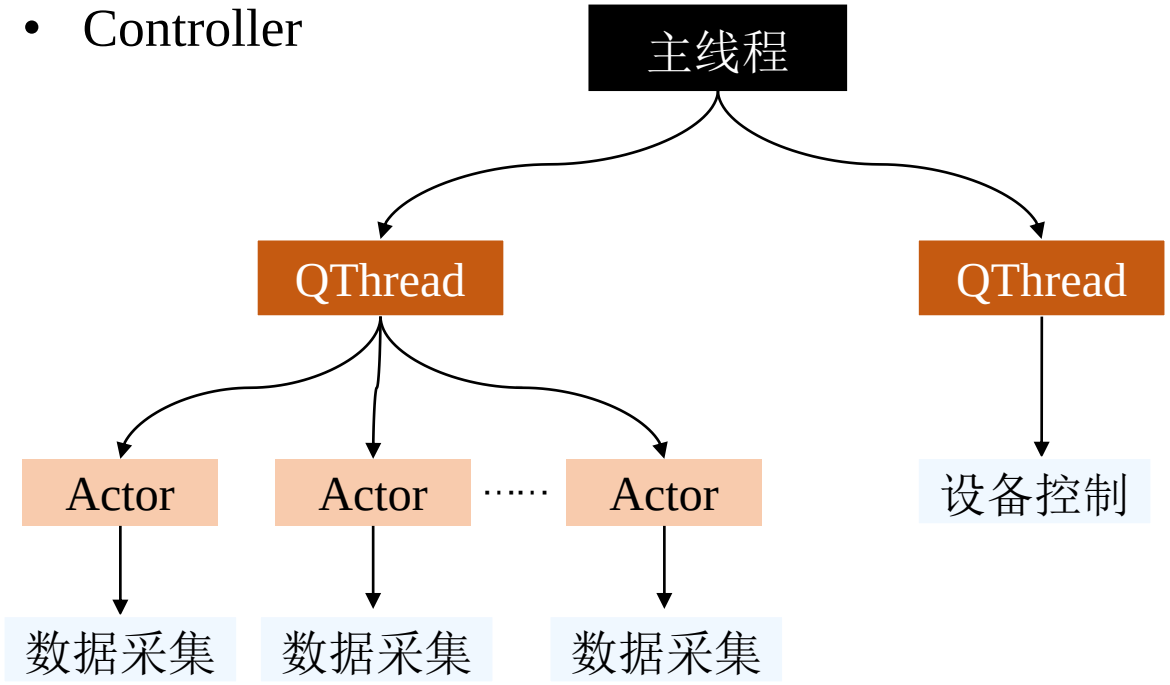
软件数据流设计

- 整体设计采用MVC(M: Model, V: View, C: Controller)架构，即前端界面与后端逻辑分离。
- 各模块之间通过MQTT实现通讯，保证各模块的工作关系。
- 数据从Controller发出，为保证数据的实时性，数据会直接存入数据库；Alarm与Monitor通过MQTT进行接收数据，而Monitor会通过MQTT发出对于设备的操作命令由Controller进行接收并做出相应操作。
- EMQX5.0每秒处理信息可达100万条，可以满足目前任何大型物理实验上的数据量。



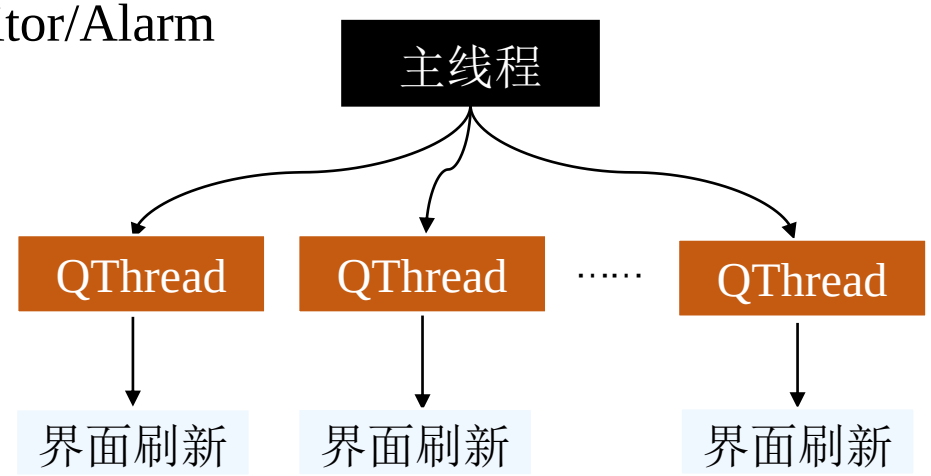
2. 技术路线——软件整体设计

- Controller



- QThread负责界面的刷新
- Actor负责数据采集，充分利用Actor的高并发处理大量任务。
- StateMachine管理状态转换

- Monitor/Alarm



- 在多个设备的界面生成过程间，为每个界面开启一个线程，保证界面的刷新效率。
- 在多个设备的管理中，在Monitor进行设备状态的转换。

3 实验控制软件框架的实现方法

3. 实验控制软件框架的实现方法——配置文件

◆配置文件

➢ 各模块之间的联系由统一的配置文件管理。Toml不仅支持多种数据结构，还支持嵌套结构，可满足多设备信息记录。

□ 基本信息：

- Broker：MQTT服务地址
- Database：数据库服务地址

□ 设备信息：

- devicename：设备名称，统一的命名格式(system/subsystem/device)，同时作为MQTT数据发布topic。
- link:设备连接类型
- address:设备地址
- parameterName:参数名称

etc.

CAEN SY5527配置文件

基本信息

```
[broker]
address = "127.0.0.1"
username=""
password=""
[database]
address = "127.0.0.1"
[hv.1]
devicename = "MDC/HV/SY5527"
board = "SY5527"
slot = 1
link = "TCPIP"
address = "192.168.0.11"
user = "admin"
password = "admin"
parameterName = [ "B0", "B1", "B2", "B3", "B4", "B5", "B6", "B7", "B8", "B9", "B10", "B11",]
[hv.1.channels.default]
V0Set = 200
I0Set = 400
[hv.1.channels.channels.4]
V0Set = 300.0
I0Set = 500.0
[hv.1.channels.channels.5]
V0Set = 300.0
I0Set = 500.0
```

设备信息

3. 实验控制软件框架的实现方法——Topic与数据格式

◆Topic与数据格式

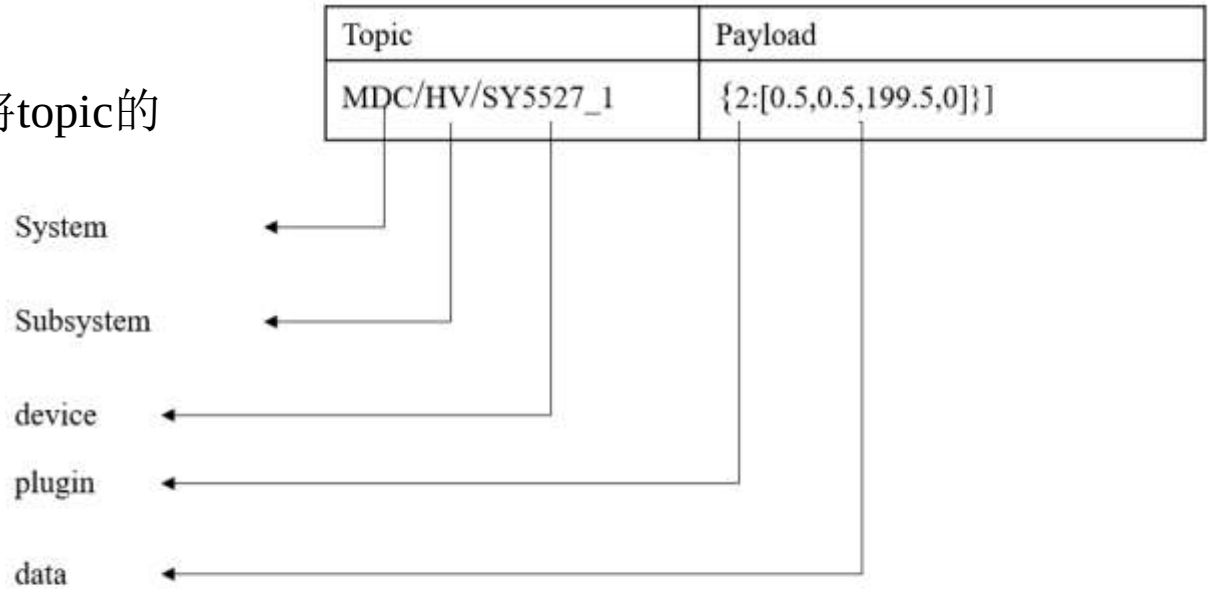
- 为保持数据一致性，定义统一的数据结构，方便数据的解析和存储。
- 为表明MQTT中topic层级以及设备之间的关系，将topic的定义分为三级。

◘ Topic：

- 数据topic：配置文件中的devicename
- 命令topic： devicename /command
- controller与monitor之间topic的发布与订阅关系相反。

◘ 数据结构：

- MQTT中的payload以字典形式发布，其中键是代表插件的整数类型，值由浮点数类型的数组。



(a)

(b)

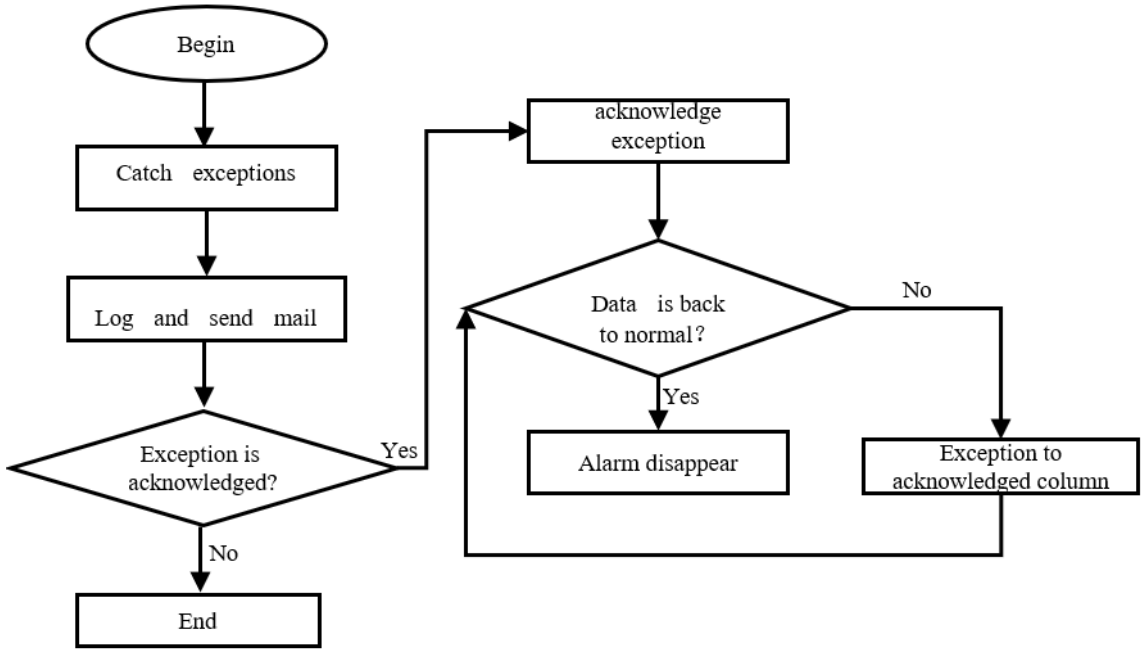
- Topic层次结构在MQTT中表现清晰的结构

3. 实验控制软件框架的实现方法——Alarm

◆Alarm

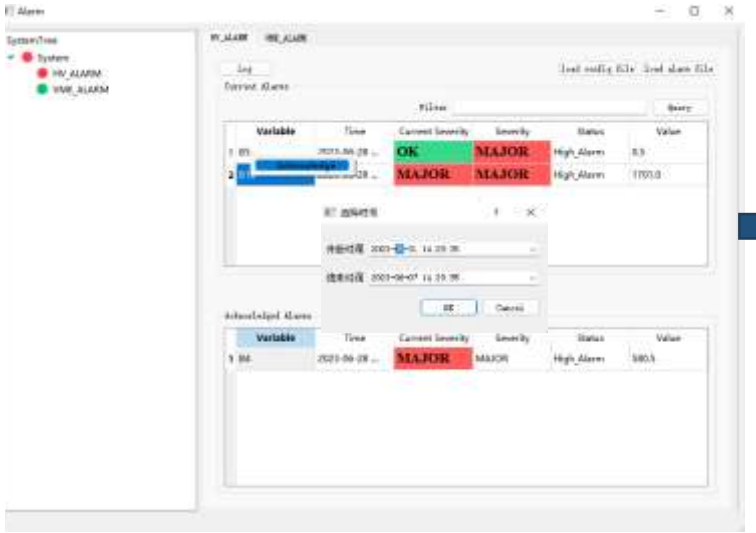
- 在设计一个控制系统时，系统的可靠性是非常重要的，而对异常情况的正确处理是实现这一目标的关键。

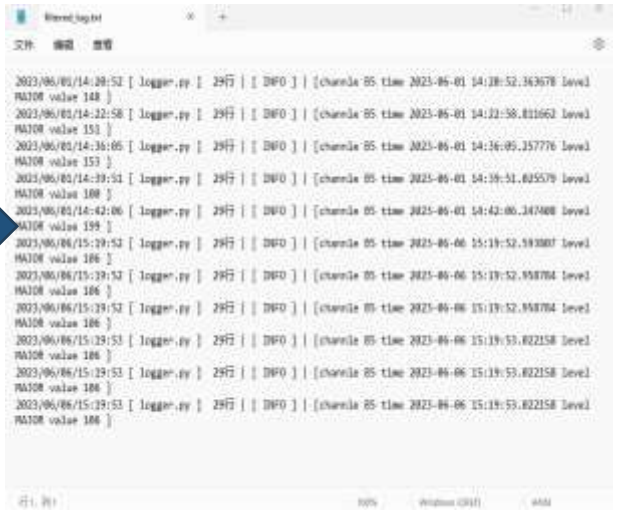
□ 报警处理逻辑



□ 报警实现:

- 每个系统配置一个报警配置文件(toml)，为每个参数设置报警阈值。每个报警值存入日志文件以便回溯。





3. 实验控制软件框架的实现方法——数据存储

- InfluxDB（时序数据库），常用的一种使用场景：监控数据统计。在大型物理实验中，需要监控各种各样的数据，这些数据通常都是时序性数据。
- 与传统的关系型数据库不同，InfluxDB 字段可以在原有表基础上会自动增加。当记录新的变量时不用创建新的表。
- 数据查询与存储方面优于关系型数据库。

注：批量插入，每批1万条

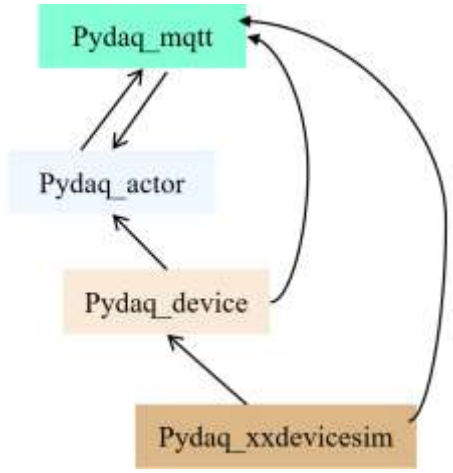
分类 / 指标		100w		1000w		10000w	
		mysql	influxdb	mysql	influxdb	mysql	influxdb
插入	用时	70s	47s	787s	470s	9704s	4673s
	CPU占用	40%	400%	40%	450%	40%	500%
	内存占用	3%	35%	3%	40%	3%	58%
查询	用时	712ms	66ms	5338ms	201ms	25820ms	571ms
	CPU占用	9%	10%	9%	10%	9%	10%
	内存占用	3%	30%	3%	35%	3%	35%

4 框架应用

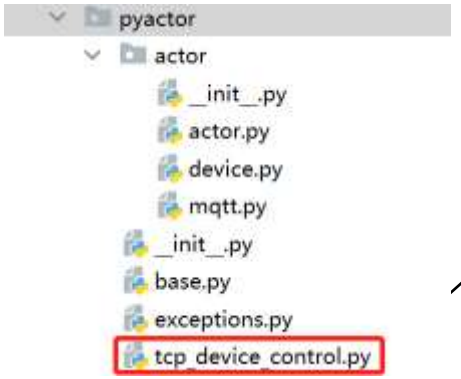
4. 框架应用

开发者使用方法.

采集



- 一套完整的用于采集的Actor框架，方便开发者快速写出采集应用。

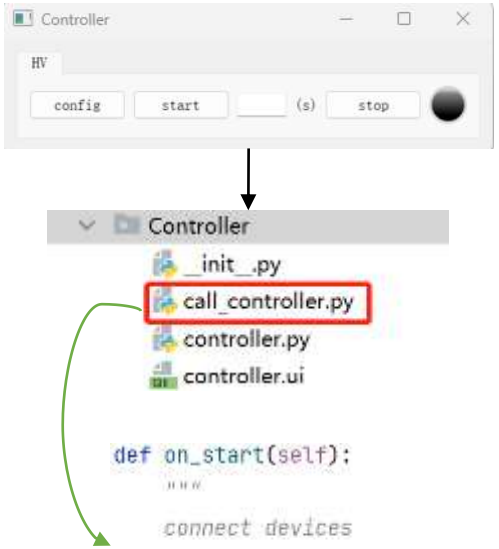


采集方法

```
def _periodic(self, reason):
    """

    Read data from instrument.

    Returns
    -------
    None.

    """
```

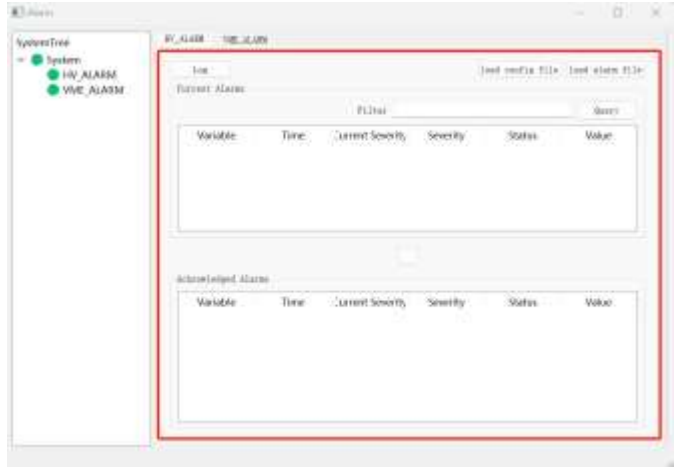
界面

Controller




```
def on_start(self):
    """
    connect devices
    :return:
    None
    """
```

Alarm

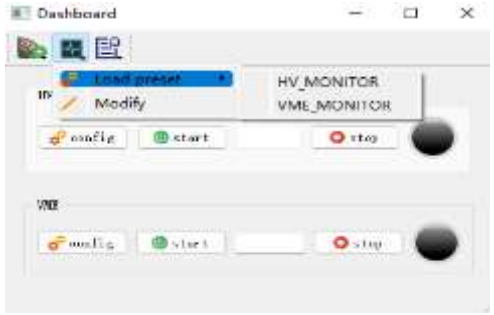


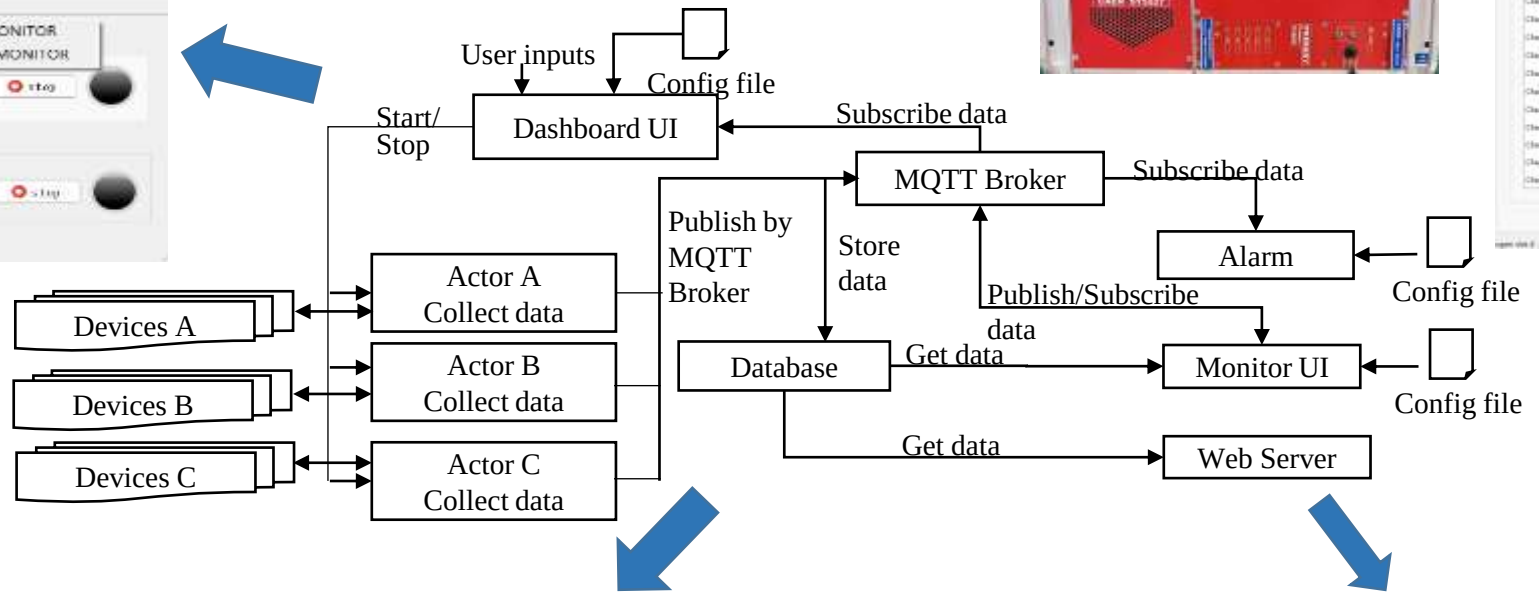
- 启动Actor之前的设备连接方法

- 由于设备的多样性，根据需求开发Monitor应用。
- 组件化与模块化开发，统一的界面形式开发者快速构建自己的应用。

4. 框架应用——实例开发

框架应用实例开发

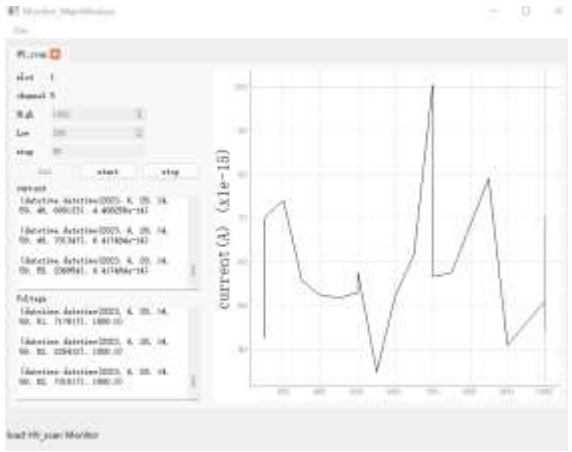




- SY5527

- VME机箱

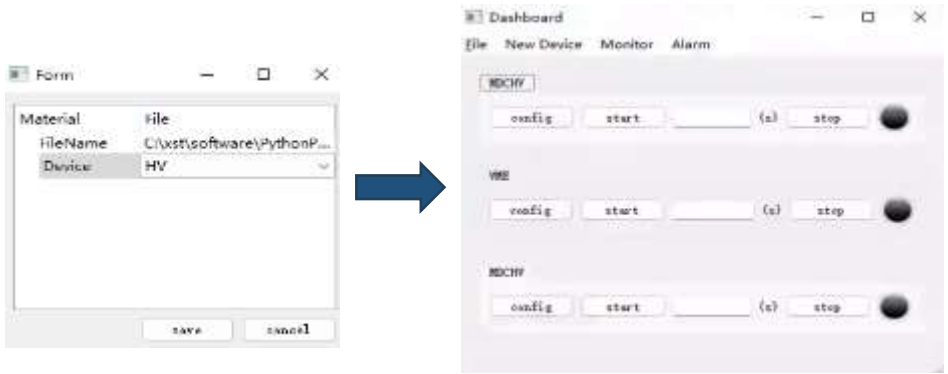
- 高压扫描应用



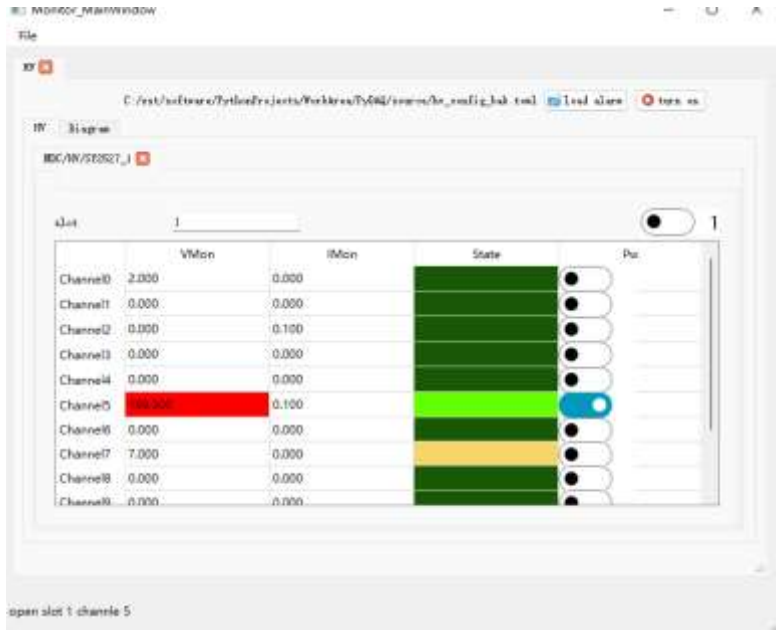
4. 框架应用——Controller

- 统一管理多个系统或设备

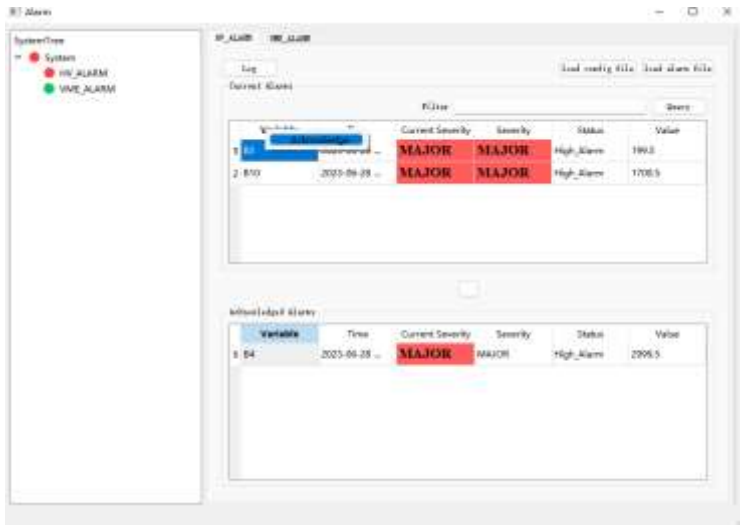
Dashboard：



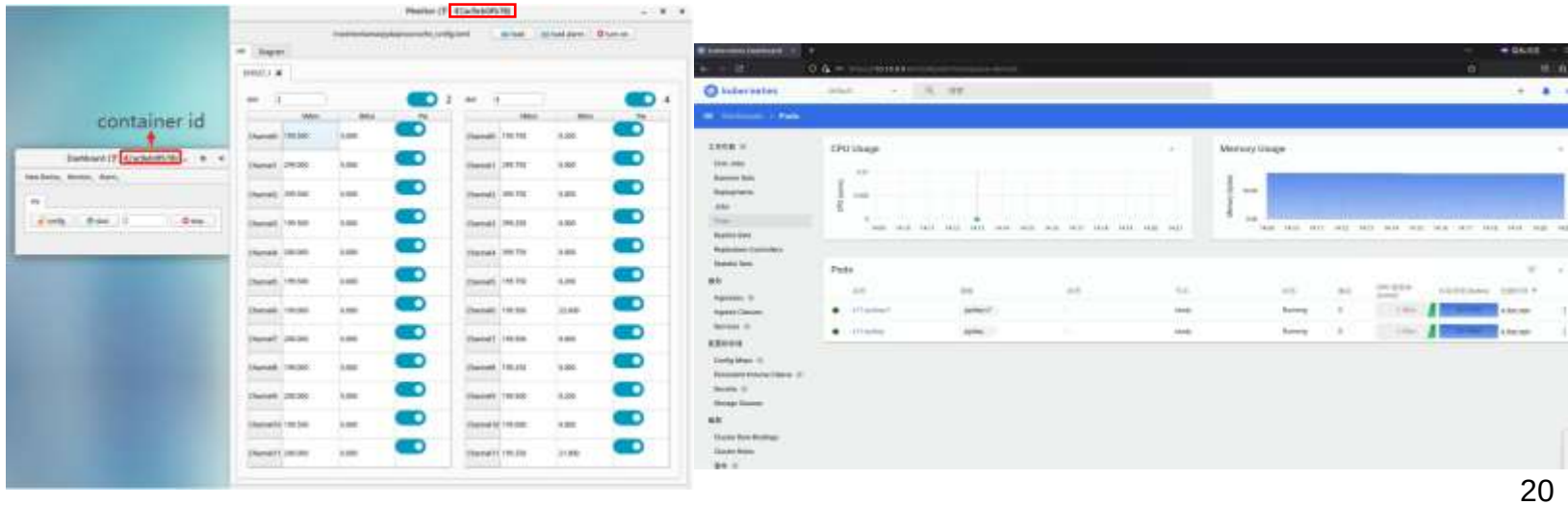
Monitor：



Alarm：



部署：



结论

5. 结论

① 慢控制系统开发中相较于商业软件LabView，使用Python作为开发环境，使得软件开发更加灵活，不仅降低了开发成本，也使得系统更好地维护。

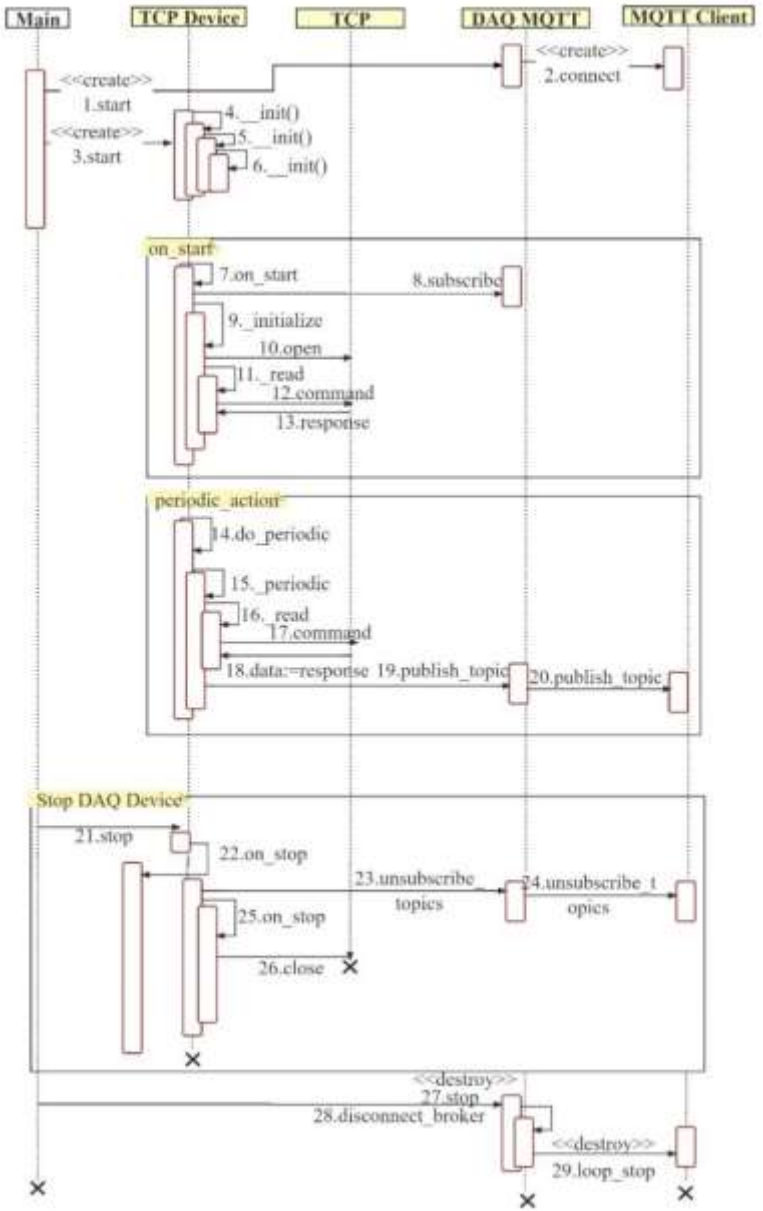
② Docker容器化技术，解决了应用移植性的问题，省去了复杂的环境配置。使用K3S对容器进行编排与管理，更好的利用计算机的资源。

③ 本框架已成功应用于电源慢控制系统，并初步建立了基本框架和开发模式。代码的模块化设计使得开发过程中只需要修改必要的部分。但仍存在改进的空间，特别是在处理大数据量时的压力测试方面。接下来，我们将继续努力进行改进和完善，以进一步提升系统的性能和稳定性。

谢谢观看！

backup

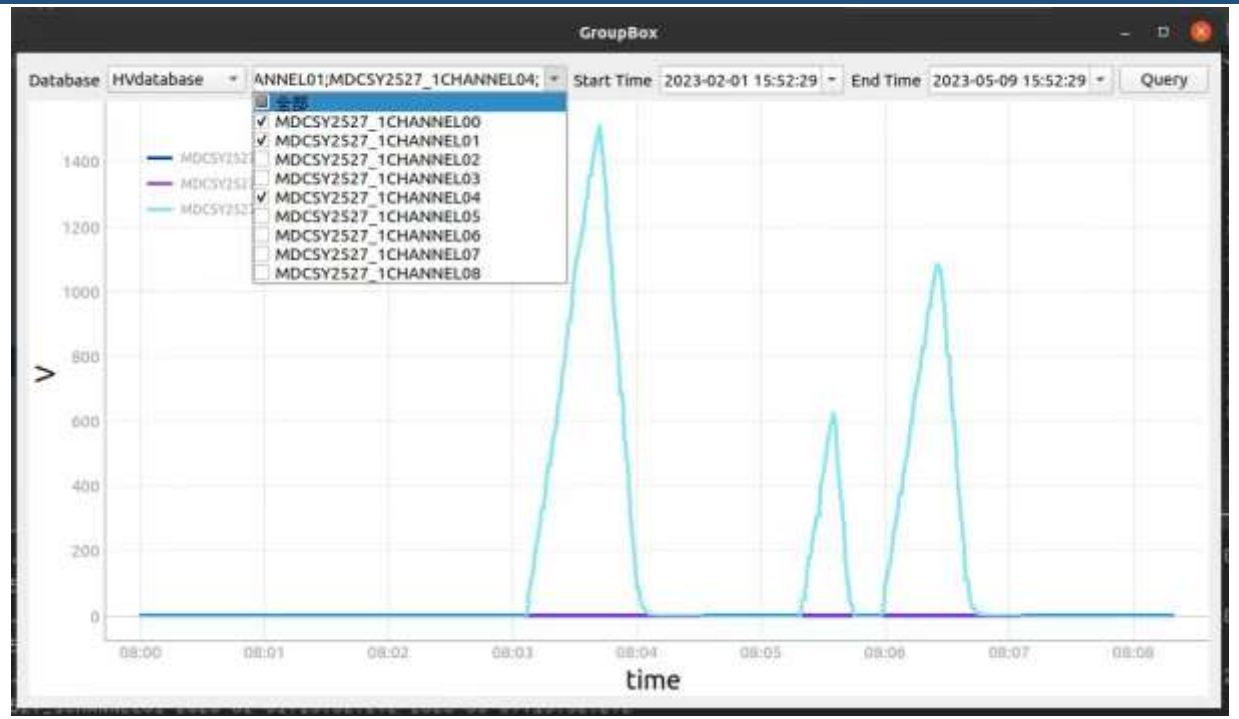
Actor时序图



backup

- 数据查询

应用端



网页端