

# Labdcs 分布式控制软件说明书

## V1.1

殷 亮

2021-6-18 南京

qq:290443202, mobile:13770850156

## 目录

<b>1 LABDCS 软件 .....</b>	<b>3</b>
1.1 简介 .....	3
1.2 框架 .....	3
<b>2 LABDCS 软件的安装 .....</b>	<b>3</b>
2.1 LABDCS 安装 .....	3
2.2 目录结构 .....	5
2.3 LABDCS 成功安装验证测试 .....	5
2.3.1 启动 labdcs 配置服务客户端 .....	5
2.3.2 配置服务客户端界面 .....	6
2.3.3 测试 test_sinfunc 案例 .....	6
2.3.4 labdcs 软件的 licence 注册 .....	12
2.3.5 window 下启动 labdcs 服务管理界面 .....	22
<b>3 LABDCS 的界面 .....</b>	<b>25</b>
3.1 服务器配置管理界面 .....	25
3.2 客户端人机界面 .....	25
<b>4 LABDCS 的案例学习 .....</b>	<b>25</b>
4.1 案例 TEST_SINFUNC .....	25
4.1.1 案例设计和接口 .....	25
4.1.2 案例的配置 .....	25
4.1.3 案例运行 .....	49
4.1.4 案例的人机界面 .....	50
4.2 案例 TEST_PIDHEATER .....	56
4.3 案例 TEST_STATEGRAPH .....	66
4.4 实际案例展示 .....	69

## 1 labdcs 软件

### 1.1 简介

labdcs 是基于服务器、客户端体系的分布式控制系统平台软件。配置和人机界面可以运行在客户端电脑，服务器与客户端通过网络进行通信。本软件平台建立在配置数据库、实时数据库、历史存储数据库的基础上，实现了数据采集、控制算法、历史数据存储等方面的信息处理功能。本软件平台将每个功能模块作为单独的进程处理，进程间通过网络进行数据的交换，充分实现了功能的模块化。运行时以实时数据库为中心，实现了低延迟和高性能，可以进行大规模点位的部署，同时可方便的进行控制算法的调试和运行。目前软件还处于继续开发阶段，由于软件模块化的构架，因此可以方便的进行功能的横向扩展。比如，目前支持的设备协议只有 Modbus TCP，未来可以方便支持其它设备协议，并且由于平台的接口的开放性，用户也可以自己编写相应的协议。目前平台直接支持的算法语言是 modelica 和 python，用户也可以根据协议用任意语言实现算法。在本软件框架下，控制和数据采集是统一的，没有必要进行严格的区分，数据采集多数对应的是实际的系统，控制算法多数对应的是算法程序，它们在本软件框架下都被抽象为具有确定输入和输出接口的功能单元。

### 1.2 框架

labdcs 最小的数据存储成为 tag，每个 tag 称为就是一个数值，最基本的数据是点位 pnt，每个点位可以由若干个 tag 组成，而最常用的点位结构称为 vqt，即由三个 tag 组成，分别代表点位的值(value)、质量(quality)、时间戳(timestamp)。

若干个点位可以形成组(group)，组与点位构成形式上的逻辑关系，一个点位可以属于多个组。单元(ele)是由点位构成的，在形式上把这些点进行分类，可以分为单元的输入点位和单元的输出点位，因此也就直接可以用输入组和输出组这两个组来描述单元所包含的点位，单元除了这两个基本组之外，还可以包含其它的组，由此就可以获得不属于它的点位的信息，labdcs 特别定义了引用组(reference)，可以让单元实现对其它单元的点位的访问。

单元与单元之间通过连接关系来描述它们之间的信息的交换，某个单元的输出点位可以与另一个单元的输入点位进行连接，这样当输出点有数据输出时，输入点就会自动获得该数据，从而实现信息流从一个单元流向另一个单元。

单元的接口和单元间的连接关系定义了单元的外延，而单元自身从输入到输出的关系则构成了单元的内涵。若单元对应于实际的系统，内涵关系就是对象本身，输出对应于对系统的数据采集，输入对应于对系统的操控变量的改变。若单元对应于虚拟的模型或者控制算法，这个输入输出关系就是该模型或该算法本身。

## 2 labdcs 软件的安装

labdcs 软件以数据库作为基础，因此主机的要求偏高，尽量采用 SSD 的固态硬盘和采用具有尽可能多核数的 CPU，这样可以大大提高整体的性能。另外，建议将配置程序和人机界面程序与服务器的运行进行分离，也就是说用一台主机来运行 labdcs 服务器，另外再配一台普通电脑作为客户端，来运行配置和人机界面程序，客户端主机的性能不需要太高，而且客户端的操作系统可以使用 linux，也可以使用 windows。

labdcs 软件的服务器端需要安装在 linux 操作系统下，而 labdcs 的操作界面和组态界面可以运行在 linux 下，也可以运行在 windows 下。

### 2.1 labdcs 安装

在百度网盘中有以下文件：

<input type="checkbox"/>	 install.txt	20
<input type="checkbox"/>	 labdcs_install.zip	20
<input type="checkbox"/>	 ladbs_client_Windows.zip	20
<input type="checkbox"/>	 rufus-3.11.exe	20
<input type="checkbox"/>	 ubuntu-20.04.3-desktop-amd64.iso	20

### 1、安装文件

- labdcs\_install.zip 为 linux 下的安装文件，labdcs 的服务器必须安装在 linux 下
- ladbs\_client\_windows.zip 为 windows 下的客户端的程序，包括了配置客户端和人机界面的客户端
- rufus-3.11.exe 为 ubuntu 操作系统的安装 U 盘的制作程序
- ubuntu-20.04.3-desktop-amd64.iso 为 ubuntu 操作系统的镜像文件，通过 rufus 工具将他镜像到 U 盘上，由于依赖库的原因，必须使用该版本的镜像。

### 2、安装步骤

- 安装 ubuntu20.04.3 操作系统
  - 在 window 下，准备一个空白的 U 盘，  
运行 rufus-3.11.exe，使用 ubuntu-20.04.3-desktop-amd64.iso 镜像文件制作启动盘
  - 将此 U 盘插入要安装 ubuntu 的操作系统的电脑  
重启电脑，设置从 U 盘启动，安装 ubuntu
- 安装 labdcs
  - 进入 ubuntu，将 labdcs\_install.zip 拷贝到操作系统的任意目录，  
进入此目录，解压此文件  
"unzip ./labdcs\_install.zip"
  - 运行  
"./labdcs\_install"，  
进行安装，安装过程中会提示输入管理员密码
  - 安装完成后，重启电脑，labdcs 服务器自动运行，  
桌面会有 labdcs 配置客户端程序 dbmg\_bin 和人机界面客户端 hmi\_bin。  
在\$HOME/labdcs/doc/目录下有 labdcs\_manual.pdf 的使用说明文件。

### 3、windows 安装 labdcs 客户端

- 直接解压 ladbs\_client\_windows.zip 任意目录，就可运行 windows 客户端。  
特别提醒 windows 客户端的 IP 需要输入 labdcs 服务器的地址(在服务器上运行 "ip address | grep inet")。
  - 若是在 linux 的 labdcs 服务器下运行客户端，直接用本地地址 127.0.0.1 即可。  
另外人机界面要能连接，对应配置程序中进程地址要用 0.0.0.0 。

## 2.2 目录结构

安装后，系统会自动启动。在桌面会有 dbmg\_bin 和 hmi\_bin 的链接文件。

- bin 目录包含了可执行程序，dbmg\_bin 和 hmi\_bin，它们分别是 labdcs 的配置管理主界面程序和用户组态程序。

- data 目录包含了各类服务的挂载目录。

- doc 目录包含了使用教程，以及相关的文件

- script 目录存放脚本 uninstall.sh

若要卸载 labdcs 平台，则运行 uninstall.sh

- store 存放与 docker 进程相关的目录，用于存放 docker 的目录

平台的运行情况用 systemctl 进行查看、开启、停止

systemctl status labdcs

systemctl start labdcs

systemctl stop labdcs

若要让平台开机时取消或开启自动启动，使用

systemctl disable labdcs

systemctl enable labdcs

## 2.3 labdcs 成功安装验证测试

### 2.3.1 启动 labdcs 配置服务客户端

- 运行 dbmg\_bin

- 进入主界面程序，首先出现的是服务器连接配置窗口



若是在宿主机器上运行，则服务器主机地址就是本地地址，即 127.0.0.1，若是管理程序运行在远程机器上，则填写服务器在网络中的主机地址，比如在局域网中，运行./dbmg\_bin 的主机的地址是 192.168.1.150，而 labdcs 服务的宿主机器在该局域网中的地址是 192.168.1.30，则这里主机地址应该填写 192.168.1.30。

labdcs 服务默认的使用端口号是 4243，这个不用改。

连接限时，是指服务器连接时间若超过这个时间，若连接服务器不成功或连接超时，则会报错。要注意的是，只允许一个 dbmg\_bin 的管理界面允许，不允许两个管理界面程序同时允许，这是为了保证数据的一致性。

- 点击刷新按钮

若服务器没有启动，或网络不通，或连接超时，则会提示连接失败

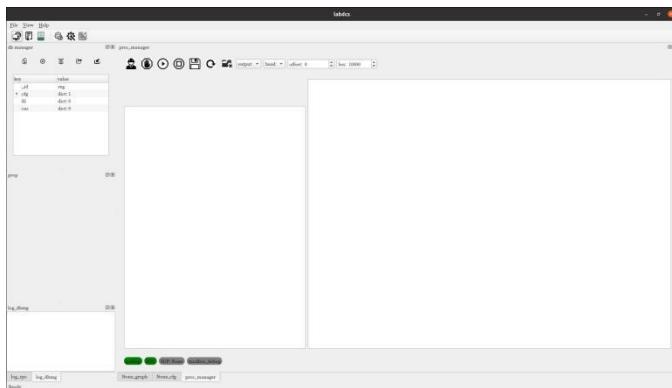
若连接成功，则出现



这时可点击确定，进入服务主管理界面

- 点击确定按钮

### 2.3.2 配置服务客户端界面



### 2.3.3 测试 test\_sinfunc 案例

- 引入 test\_sinfunc 案例

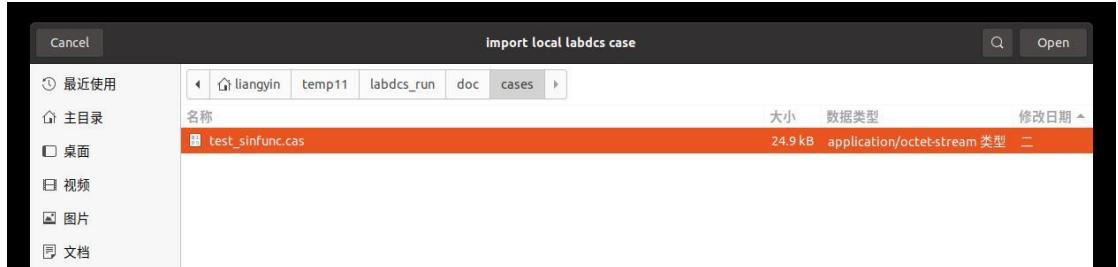
点击 db\_manager 窗口中的 cas 条目，让其高亮

db manager

key	value
_id	reg
cfg	dict: 1
fil	dict: 0
cas	dict: 0

/cas

点击 ，出现案例目录的选择窗口，默认的是在程序运行目录 bin，退到上级目录下 doc/cases，选择 test\_sinfunc



点 open, 出现



若不改名，则点击 ok

db manager

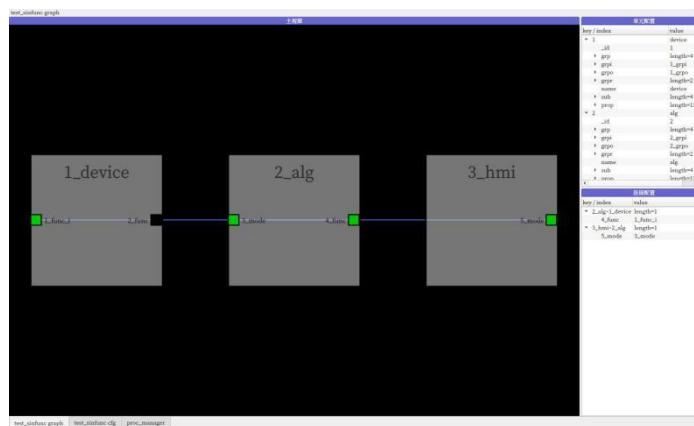
key	value
_id	reg
cfg	dict: 1
fil	dict: 0
cas	dict: 1
test_sinfunc	dict: 1

稍等一下，则 cas 下出现 test\_sinfunc 的下级目录。

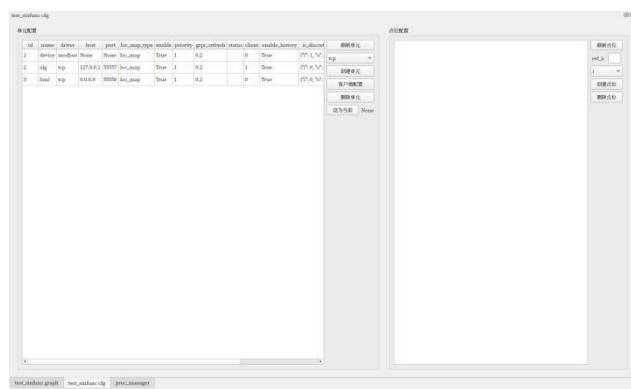
若没有出现，可能是主机处理速度慢，有显示的延缓，此时双击 cas，刷新一下。

### ● 查看 test\_sinfunc 案例

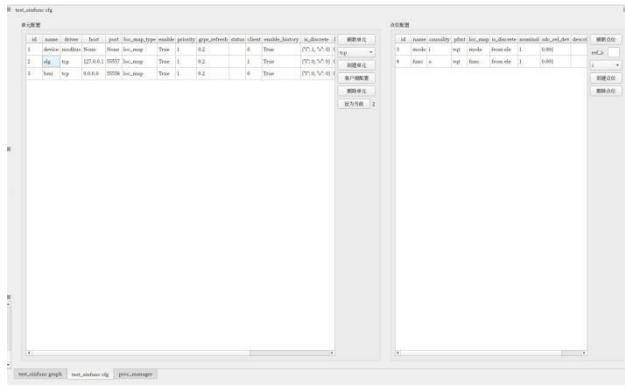
在 test\_sinfunc 被选中的条件下，点击 ，则标签后缀为 graph 和 cfg 的窗口被更新，



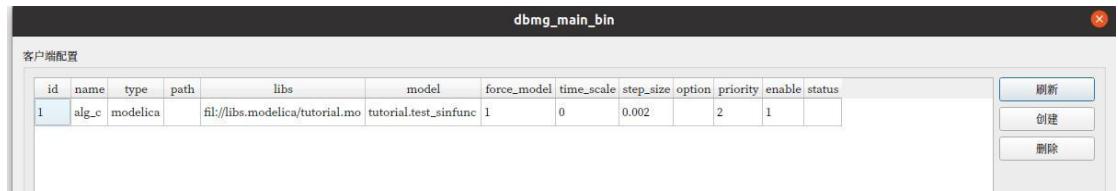
对于标签 cfg 的窗口，要点击刷新单元，方可显示单元的配置信息



选中某个单元，并点击设为当前，则显示它的点位信息



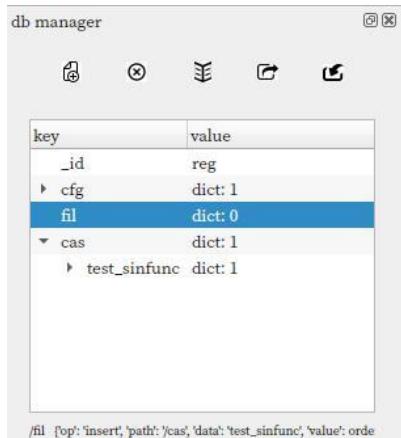
若单元的 clinet 值大于 1，还可有客户端配置信息，在该单元为当前的条件下，点击客户端配置



从此信息可以看出，单元 2 的客户端是 modelica 类型的模型，该模型来自数据库，数据库存放的位置是 fil://libs.modelica/tutorial.mo，该模型的全名为 tutorial.test\_sinfunc，因此需要将该文件引入数据库。

### ● 引入模型文件 tutorial.mo

选中 db manager 中的 fil，



点击 ，在出来的窗口中输入 libs，因为案例算法模型的引入地址为 fil://libs.modelica/tutorial.mo，



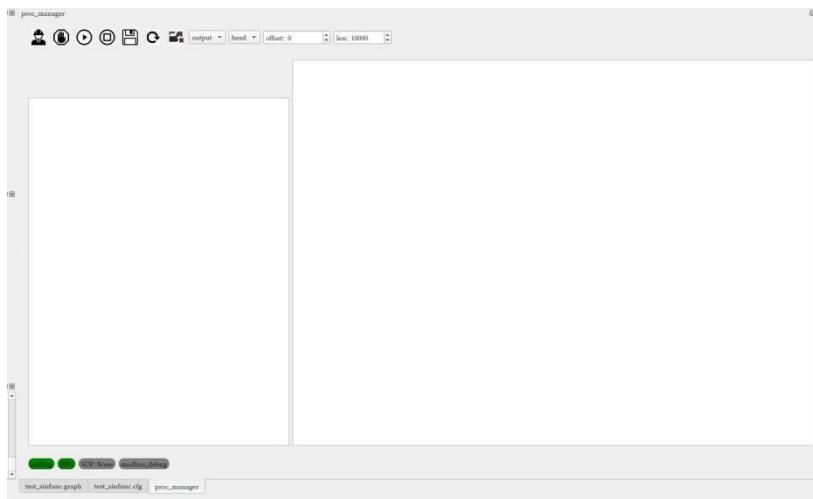
点击 ok 后，在 libs 被选中的条件下，再点击 ，创建子目录 modelica



点击 ok 后，在新建的 modelica 被选中的条件下，再点击 ，在弹出的文件选择窗口中，选择父目录下 doc 中的 tutorial.mo 文件  
这时该文件被导入了数据，如下图

db manager	
key	value
_id	reg
cfg	dict: 1
fil	dict: 1
libs	dict: 1
modelica	dict: 1
tutorial.mo	dict: 0
cas	dict: 1
test_sinfunc	dict: 1

- 运行 test\_sinfunc 案例  
点击 proc\_manager 窗口



点击该窗口中的 ，选择 test\_sinfunc 项目，



点击 ok，就可以启动该项目

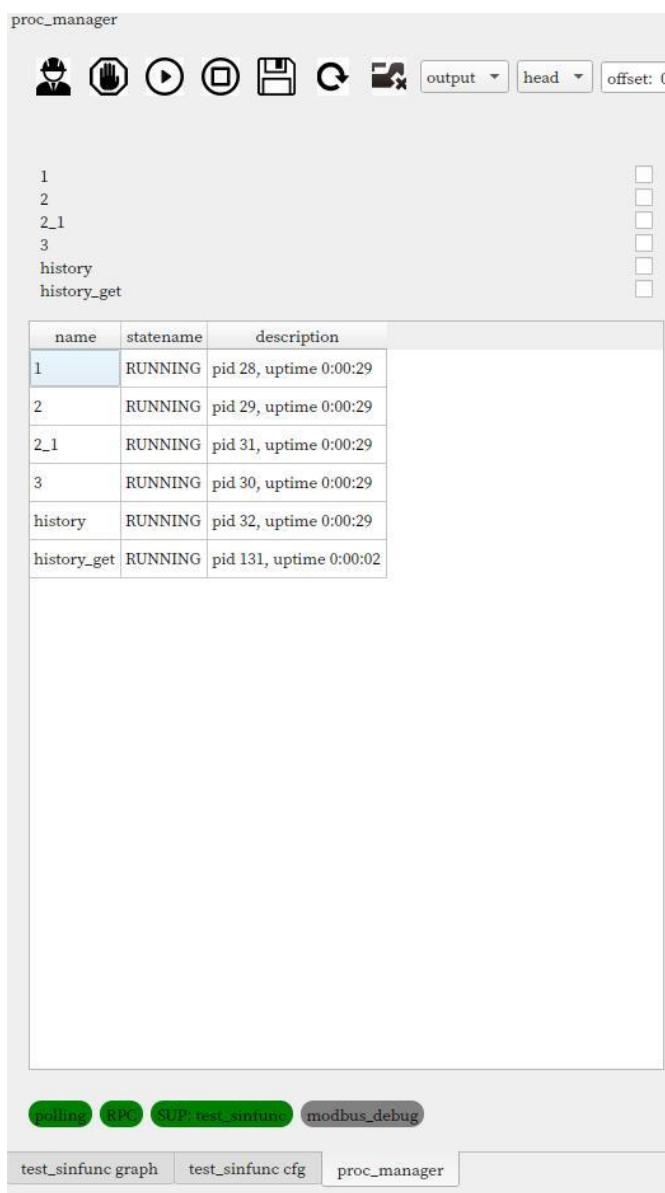


若弹出 ，说明没有注册，需要点击主窗口工具条中



的 ，进行注册。

若是注册过的软件，则出现该案例的单元进程信息



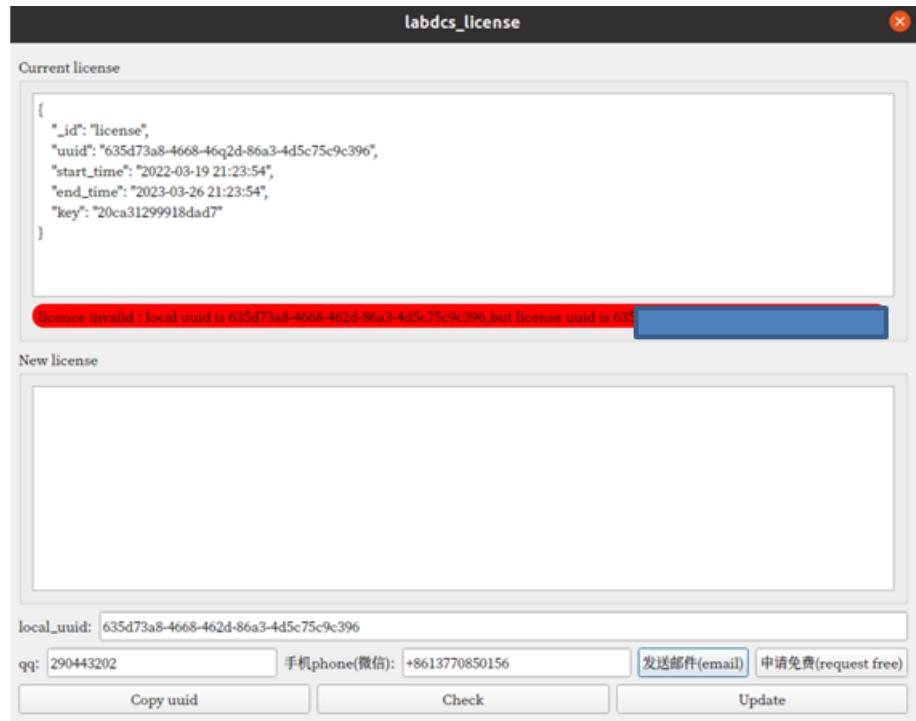
由于该案例使用了一个调试用的 modbus 设备，该设备实际上是一个主机的进程，用以模拟一个虚拟的 modbus 设备(此处仅仅起数据存储的作用)，作为我们调试用。

因此要点击主工具条上的 ，来开启该虚拟设备，这时 proc\_manager 窗口中 modbus\_debug 标签显示是绿色的



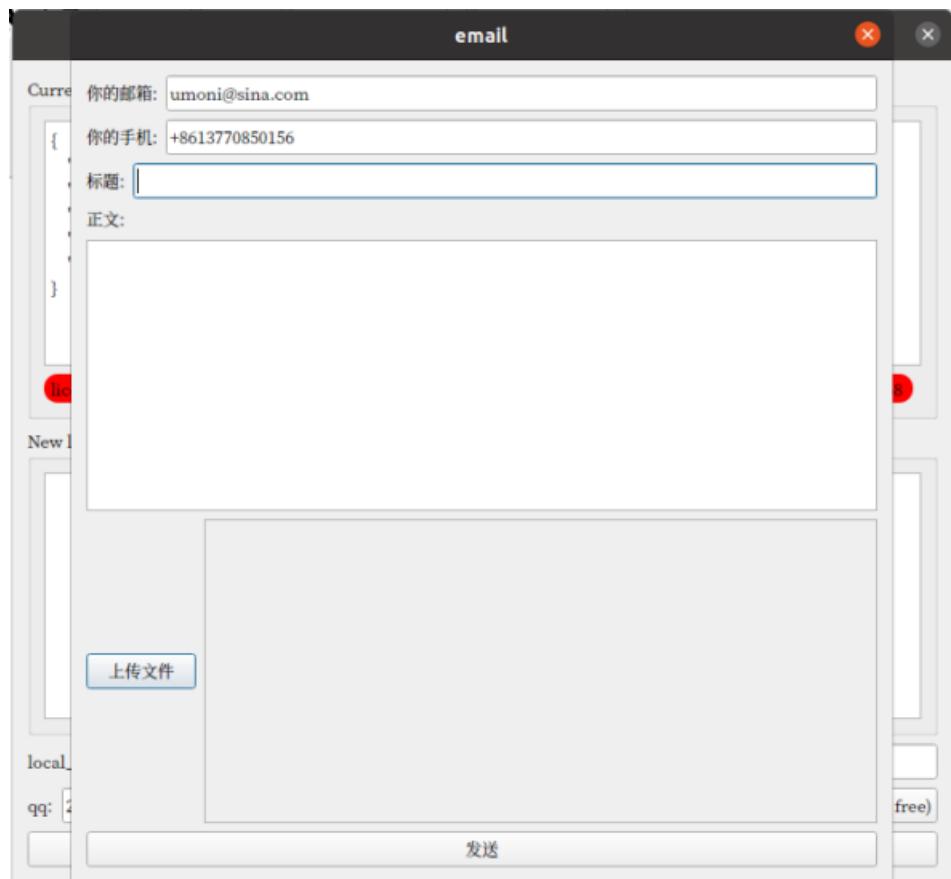
#### 2.3.4 labdcs 软件的 licence 注册

点击工具栏的 ，弹出



提示 license 无效，需要注册，目前软件为教育用户（学生和老师）提供了免费注册，提供 1 年期的软件免费使用。

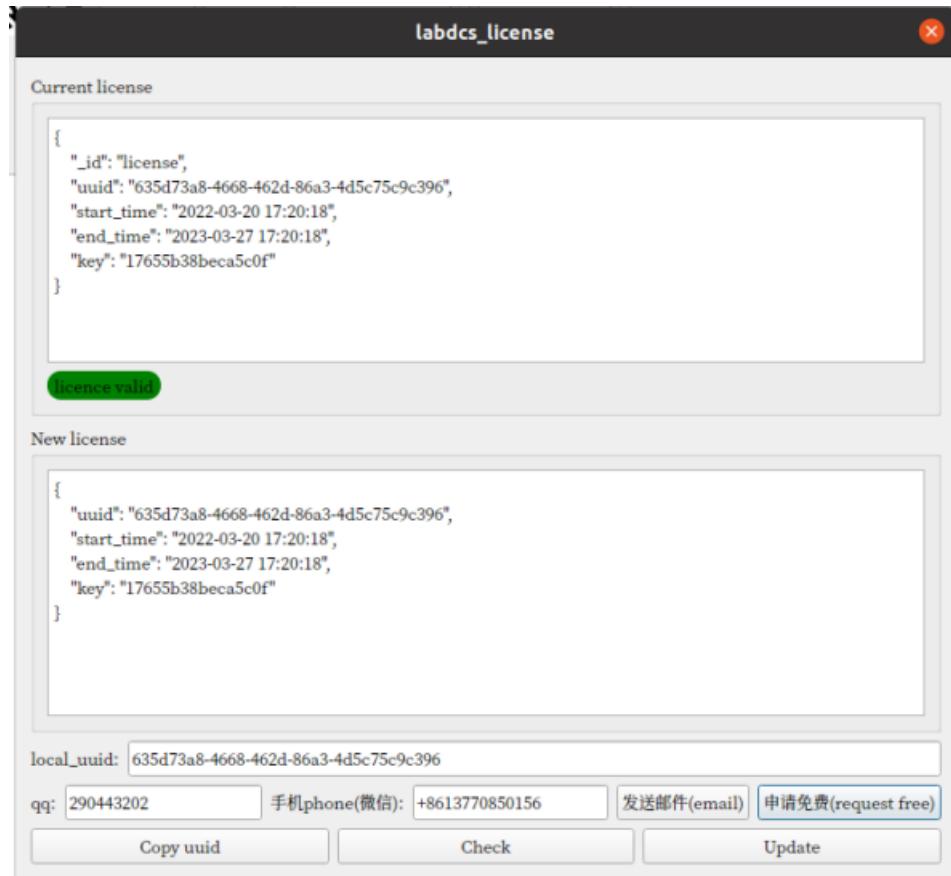
点击“申请免费”



填写有效的邮箱和手机号，即能获得免费使用。

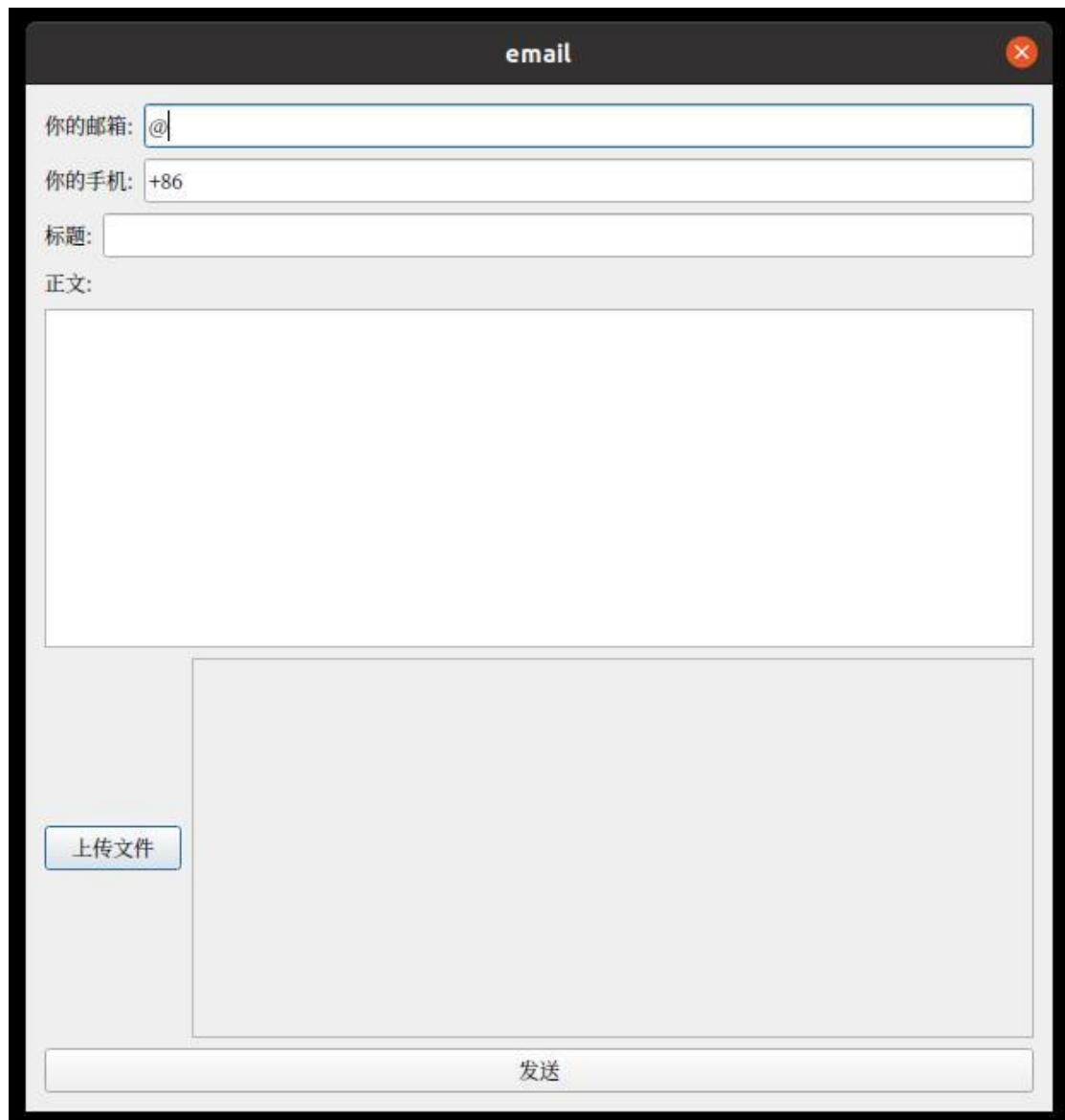


点击发送，若出现邮件发送成功的信息，说明通过了申请，点击 OK，



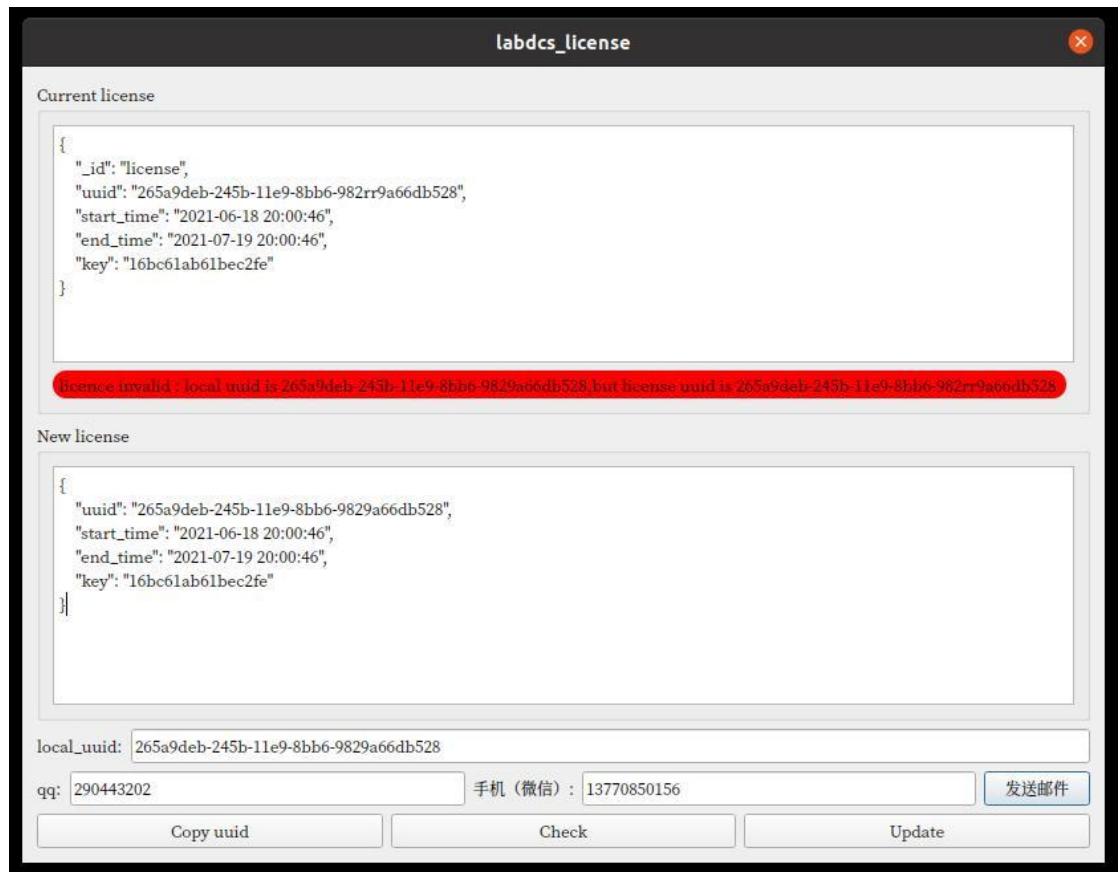
则显示软件注册成功。

若有任何问题，请联系作者的 qq 或手机微信，可以发邮件。点击发送邮件按钮。  
若是商业用户，请联系作者获得 licence，操作如下：



填写你自己的邮箱，特别注意邮箱要填写正确，作者收到邮件后，会将有效的 licence 信息发到此邮箱，注意信息要完善。然后点发送。当 qq 或微信联系时，点击 Copy uuid，将此信息发给作者。

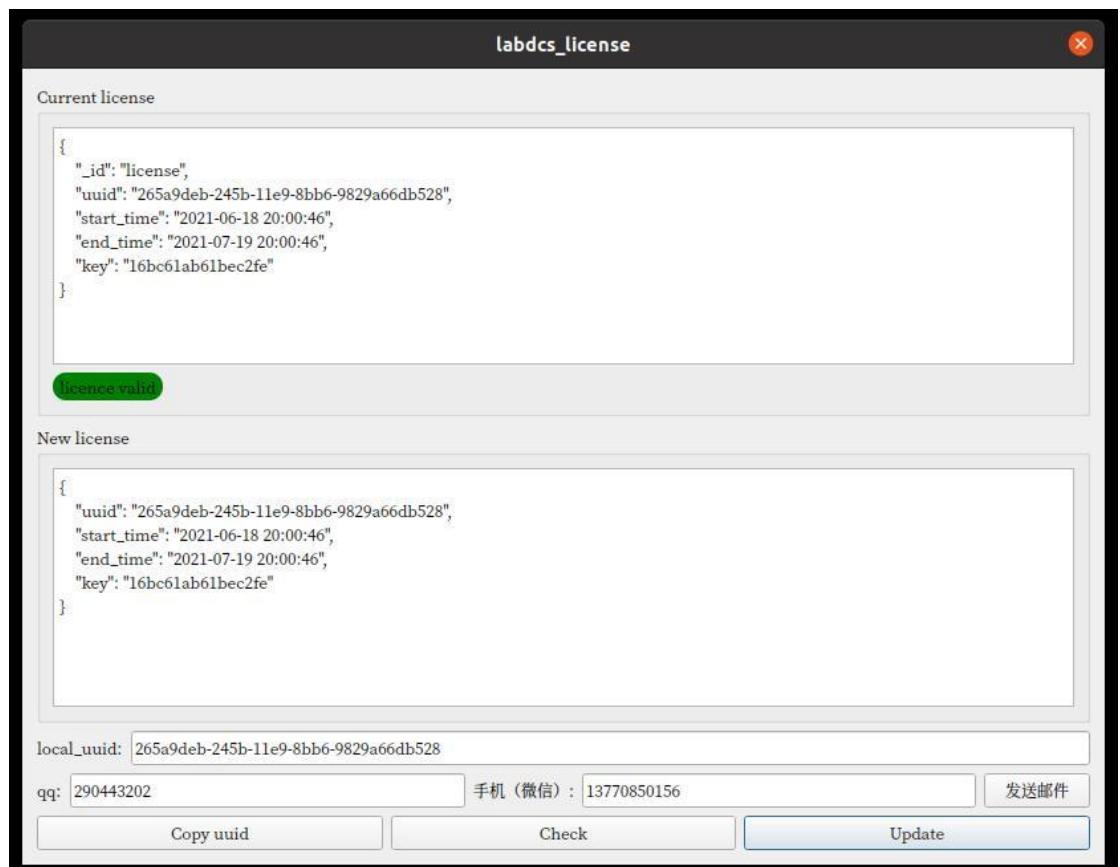
作者不管使用哪种反馈方式，新的 licence 都是 json 数据的格式，你将此数据复制到 New licnece 中



点击 Update 按钮，若 licence 有效，则弹出



，并且窗口的 licence 状态显示为绿色



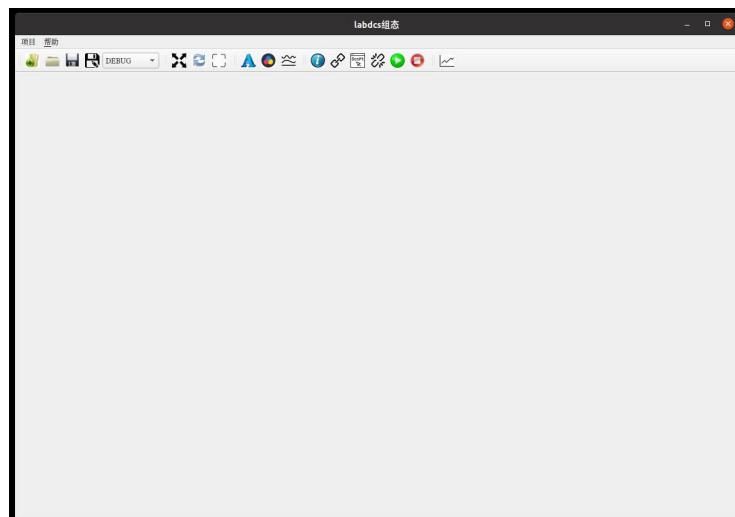
由此，注册完毕。

### ● 运行 labdcs 人机组态界面

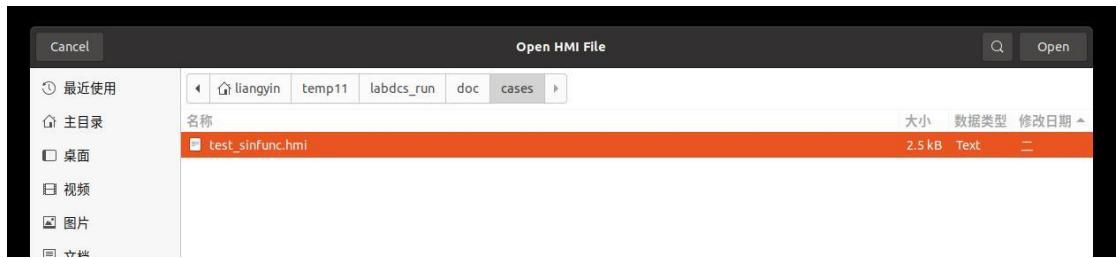
运行组态界面，可以对 test\_sinfunc 案例的点位进行监控

进入~/labdcs\_run/bin 目录，运行

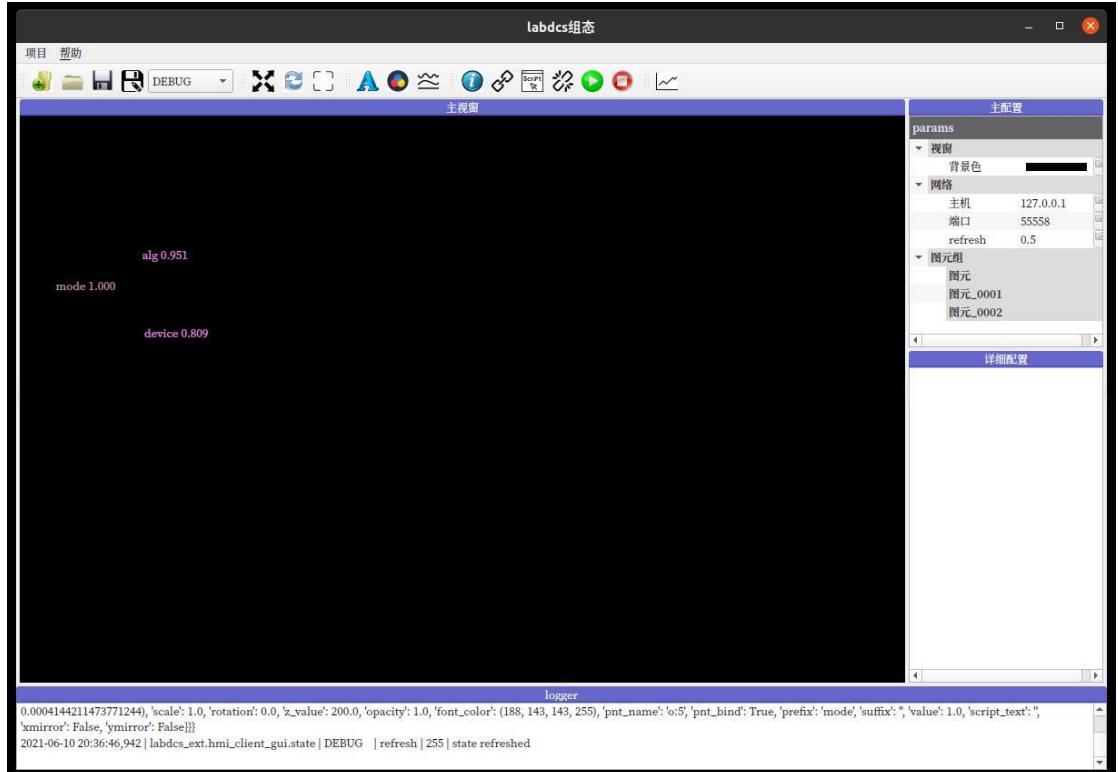
./hmi\_client\_bin



点击 ，选择 test\_sinfunc.hmi 文件



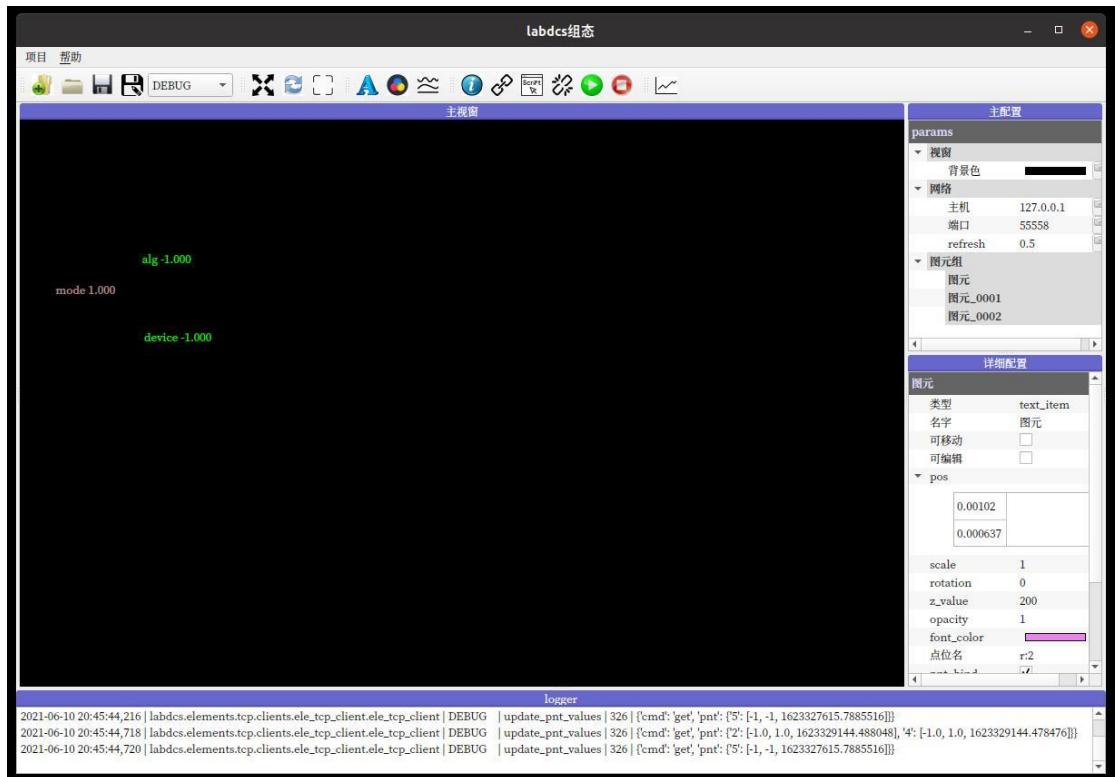
则出现



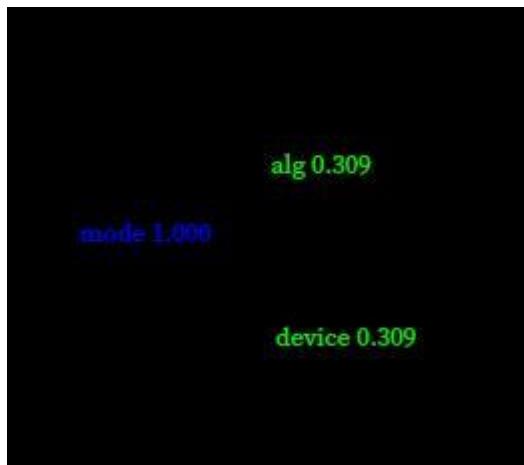
该人机界面显示了该案例的三个点位，分别是 mode, alg, device， mode 为 人机界面的输出，它设置着控制算法的信号输出的模式，算法单元把信号输出给 modbus 设备，图中的 alg 和 device 分别是算法单元和设备单元的输出信号。

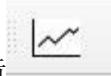


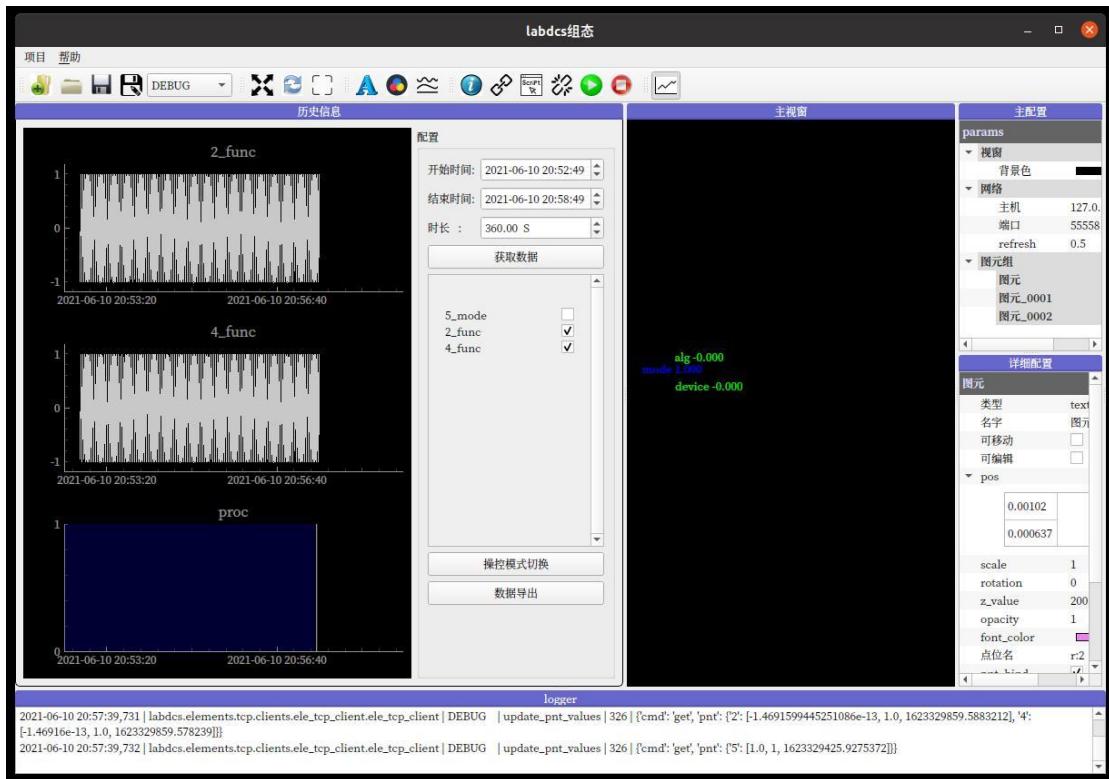
点击 ，则启动监控，



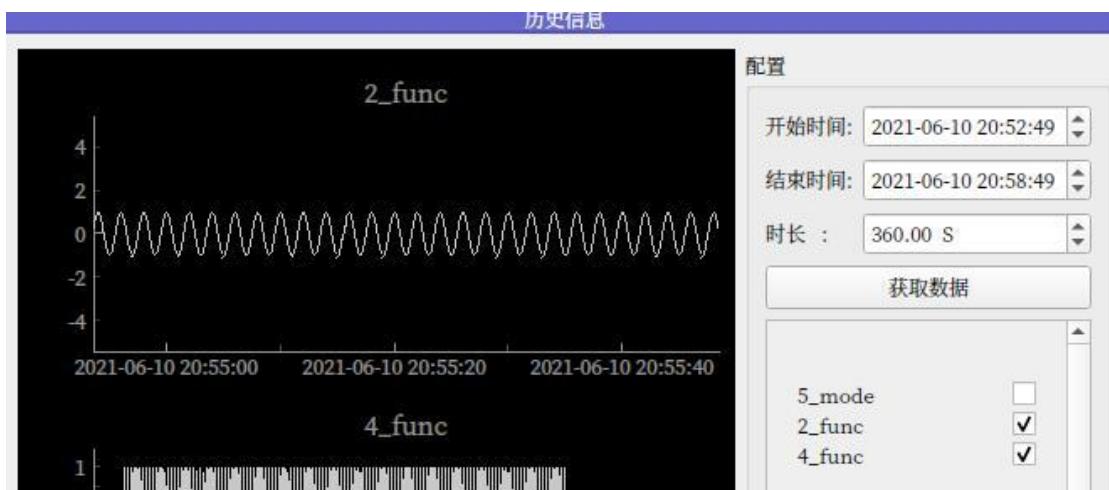
如图中，alg,device 都显示绿色，若是其它颜色，说明案例运行有问题。  
若 alg 显示黄色，并且后缀为 x，则再稍等，可能是控制算法单元还在启动。  
在图中的 mode 标签上，点击右键，在弹出的窗口中，输入 1，并点确定，则 mode 会显示蓝色，并且 alg,device 的数值不断变化



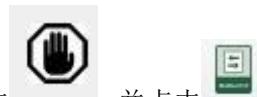
点击 ，可以查看历史信息，在左边会显示历史信息窗口，可以调整它的大小，让其窗口变的大些。可修改开始时间和结束时间，获得这段时长的数据。  
修改结束时间，比开始时间大 6 分钟，并选择要显示的点位 2 和 4，点击获取数据，则



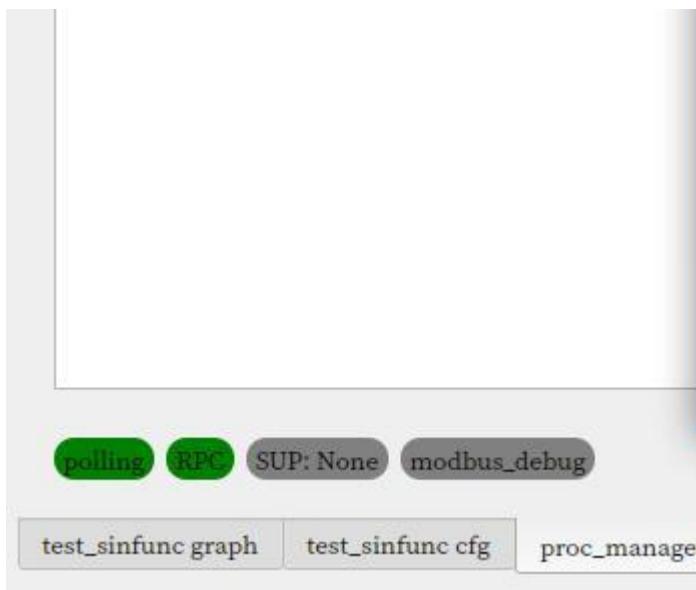
可以通过鼠标缩放，获得细节数据



至此，可以说明 labdcs 服务已经成功安装了。



若要停止案例的运行，则进入管理程序的 proc\_manager 窗口，点击 。并点击 。关闭 modbus\_debug，则

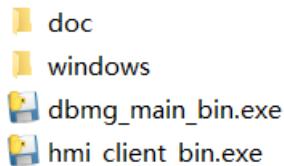


sup 标签为灰色，说明案例运行已经停止，并且 modbus\_debug 也停止了。

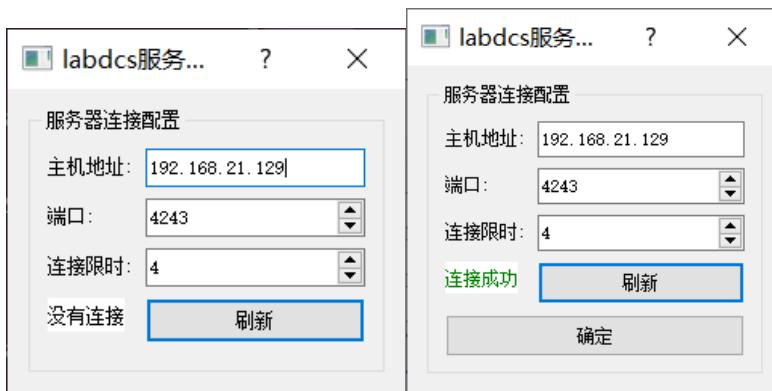
### 2.3.5 window 下启动 labdcs 服务管理界面

在 Windows 下启动 labdcs 服务管理，需要安装 labdcs 管理服务程序的主机与 labdcs 服务器在同一个网络下。

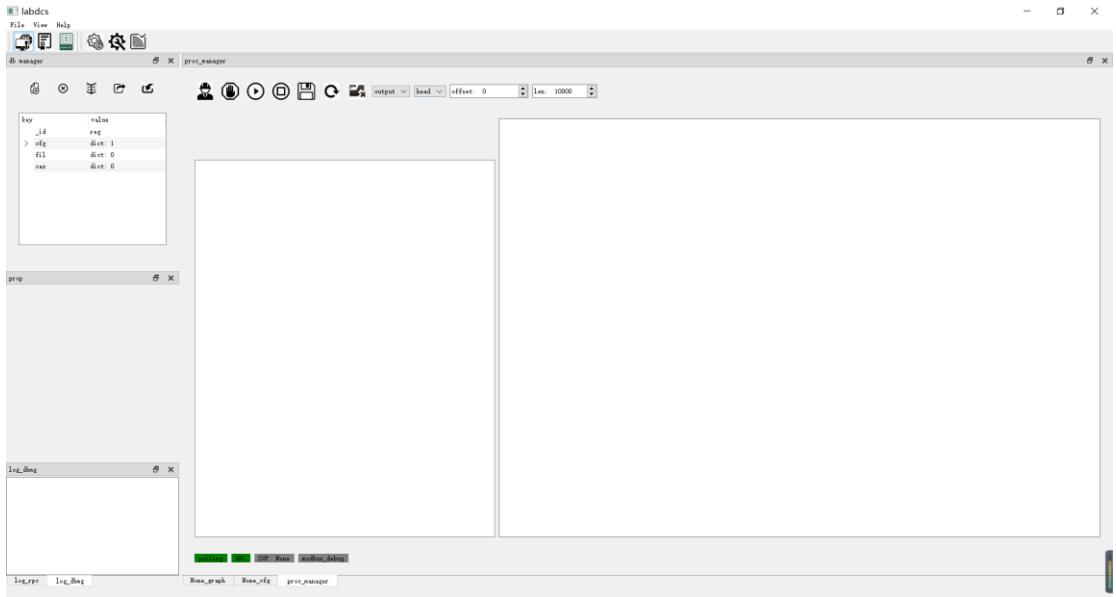
将 labdcs\_windows.zip 文件解压缩任意目录下，它包含



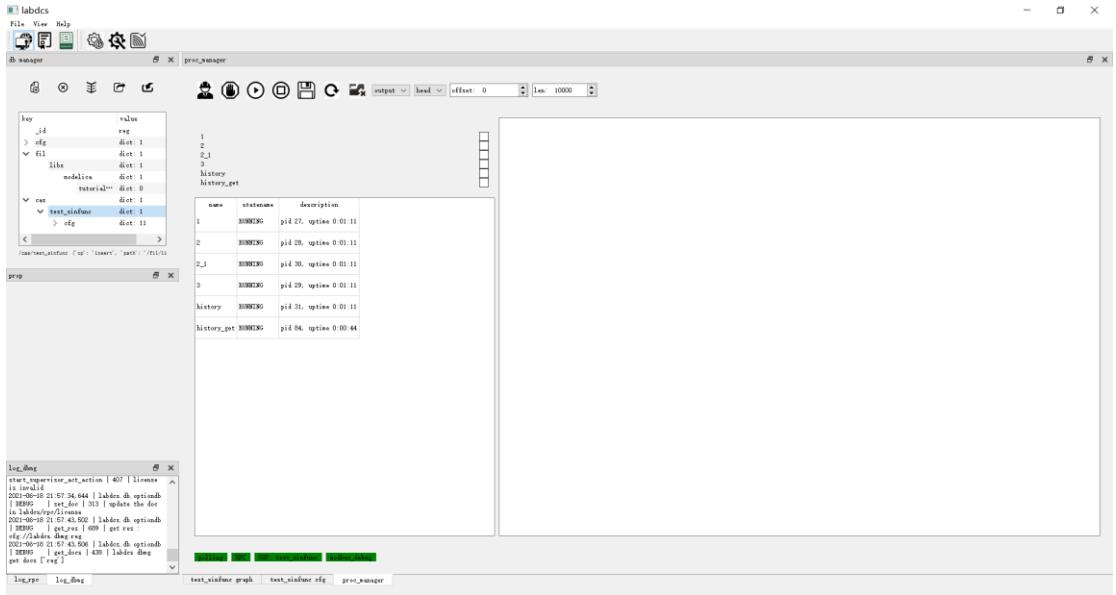
双击 dbmg\_bin.exe，



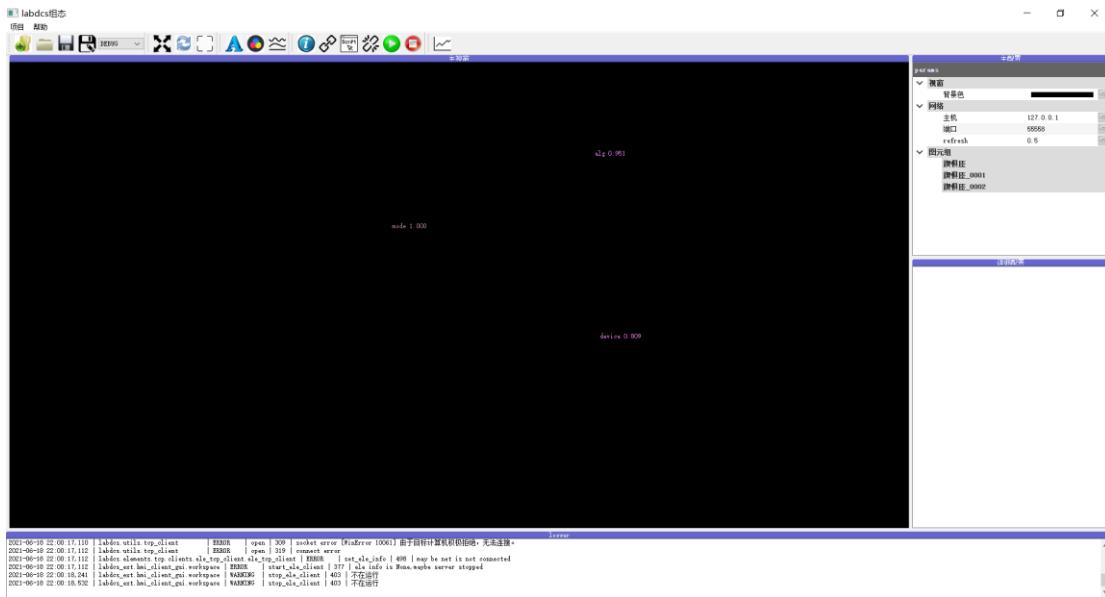
主机地址 192.168.21.129 是 labdcs 服务器在网络中的地址，连接成功后点击确定



这是 Windows 下的界面，同样导入 cas，并且上传 tutorial.mo，具体操作如前所述。



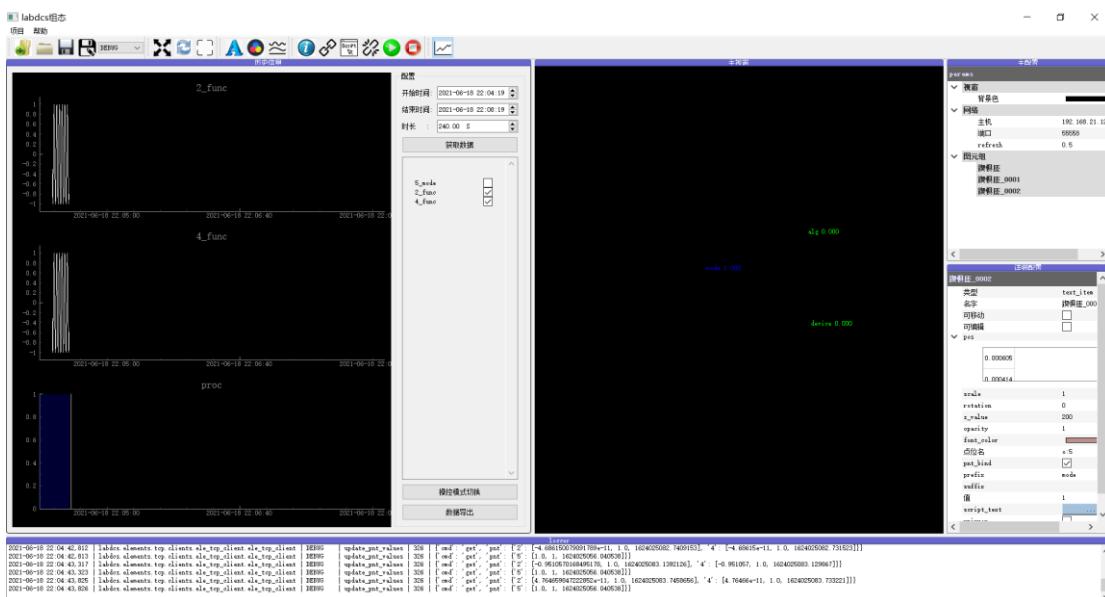
同样，双击 hmi\_client\_bin，



这时要注意，由于是远程主机，因此这里的主机地址要改为 192.168.21.129，即



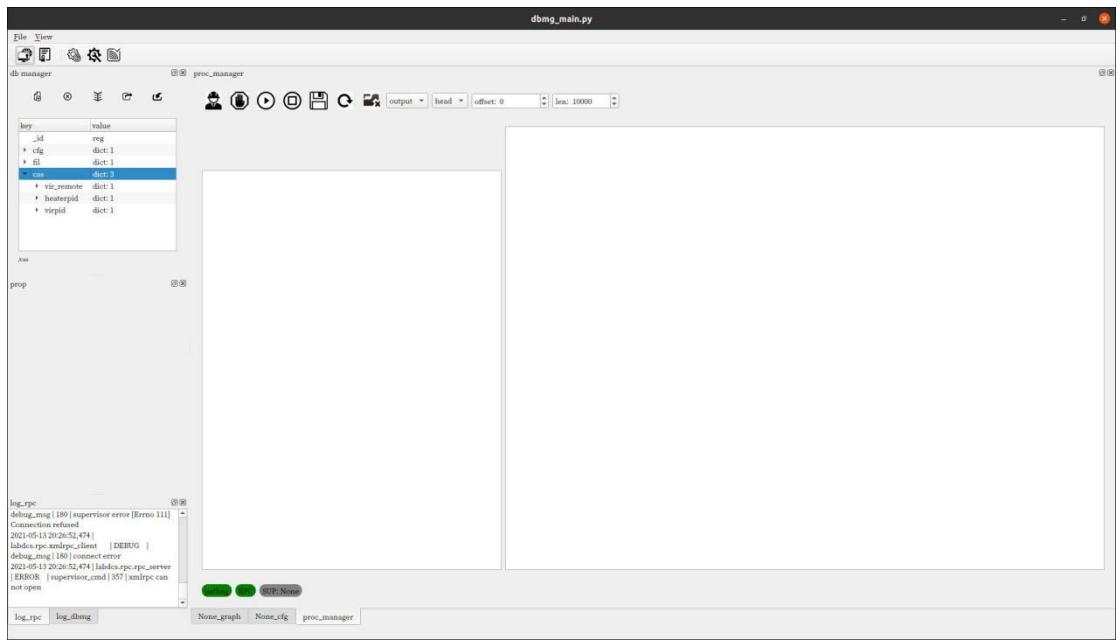
其它操作如前所述



### 3 labdcs 的界面

#### 3.1 服务器配置管理界面

##### 3.1.1 基本界面



配置管理界面由菜单栏、工具栏、功能窗口组成

功能窗口为可悬浮窗口，左边从上到下，分别是数据库管理窗口、属性窗口、日志输出窗口，日志输出窗口又分为远程服务端日志输出 rpc 和配置管理程序日志输出 dbmg;右边的窗口包含了三个窗口，分别是案例的图形显示窗口、案例的配置窗口和案例进程的管理窗口。

#### 3.2 客户端人机界面

##### 3.2.1

### 4 labdcs 的案例学习

#### 4.1 案例 test\_sinfuc

此案例是系统自带的案例，这里从零开始讲解如何建立此案例

##### 4.1.1 案例设计和接口

案例设计如下：

此案例包括了三个单元 hmi、alg、device，单元 alg 一个输入信号 mode 和一个输出信号 func，当有输入事件，并且输入的值为 1 时，输出标准的 sin 信号，否则输出为 0 的信号。

device 为 modbus 的设备，alg 的输出信号输出到 device 的一个 32 位浮点数寄存器 func。hmi 为一个输出信号，即为 mode 信号，控制 alg 输出的类型，并两个参考信号，一个参考 alg 的输出 func，一个参考 device 的寄存器 func。

##### 4.1.2 案例的配置

###### ● 创建案例

点击 db manager 中的 cas

db manager

The screenshot shows the db manager interface with a tree view of database keys and their corresponding values. The tree structure is as follows:

- key: \_id, value: reg
- key: cfg, value: dict: 1
- key: fil, value: dict: 1
  - key: libs, value: dict: 1
    - key: modelica, value: dict: 1
      - key: tutoria..., value: dict: 0
- key: cas, value: dict: 1
  - key: test\_sinfunc, value: dict: 1

The path /cas is displayed at the bottom left.

key	value
_id	reg
cfg	dict: 1
fil	dict: 1
libs	dict: 1
modelica	dict: 1
tutoria...	dict: 0
cас	dict: 1
test_sinfunc	dict: 1

点击 ，弹出



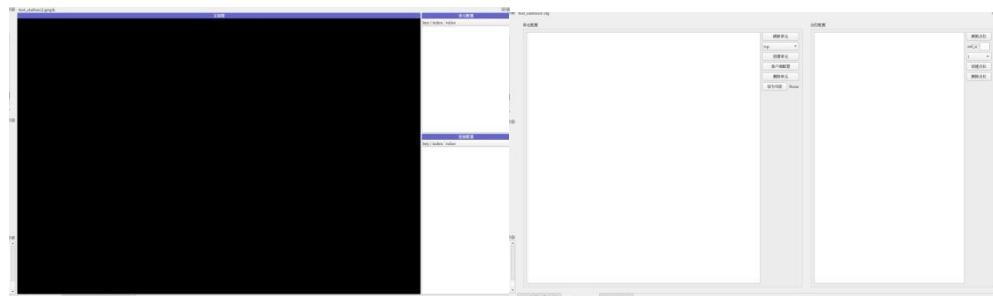
输入案例的名字 test\_sinfunc2，点击 ok，则 db manager 的 cas 中将新增案例 test\_sinfunc2，如下图

The screenshot shows the 'db manager' application window. At the top, there are standard window control buttons (minimize, maximize, close). Below the title bar is a toolbar with icons for file operations: add, delete, search, refresh, and others. The main area is a tree view table with columns 'key' and 'value'. The tree structure is as follows:

key	value
_id	reg
‣ cfg	dict: 1
‣ fil	dict: 1
‣ libs	dict: 1
‣ modelica	dict: 1
‣ tutorial	dict: 0
‣ cas	dict: 2
‣ test_sinfunc	dict: 1
‣ test_sinfunc2	dict: 1

At the bottom of the tree view, there is a status message: '/cas/test\_sinfunc2 {op: 'insert', 'path': '/cas', 'data': 'test\_sinfunc2'}

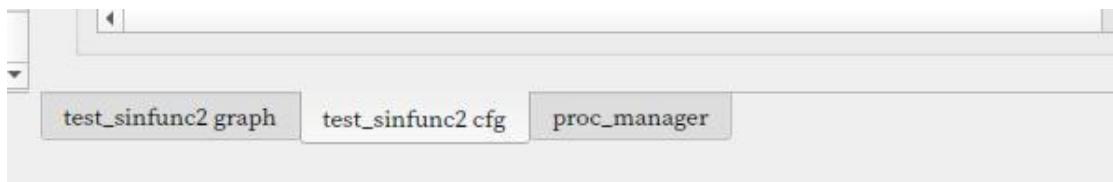
在 test\_sinfunc2 选中的条件下，点击 ，打开此案例，这时右侧会出现空白的图 graph 和配置 cfg



### ● 创建 device 单元

在本案例中，要创建 3 个单元，分别是设备 device，算法 alg 和人机交互 hmi, 其中 device 的类型是 modbus，alg 和 hmi 的类型是 tcp

点击主窗口的 test\_sinfunc2 cfg 的标签，



点击 ，cfg 面板中依然是空白，这说明我们初始是没有单元的。

选择单元类型为 modbus，，点击 ，单

元配置表中会出现新的单元，若机器比较慢，可以稍等，再刷新单元，

单元配置														
id	name	driver	host	port	loc_map_type	enable	priority	grpr_refresh	status	client	enable_history	is_discrete	1	mc
1	new	modbus	None	None	loc_map	True	1	0.4	0	True	{"i": 1, "o": 0}	0	mc	mc

双击 name 列中的 new,可以进行编辑改名，改为 device，可以再刷新单元，以验证改名成功

id	name	driver	host	port	loc_map_type	enable	priority	grpr_refresh	status	client	enable_history	is_discrete	1	mc
1	device	modbus	None	None	loc_map	True	1	0.4	0	True	{"i": 1, "o": 0}	0	mc	mc

其它参数的修改也是双击此单元格，其含义如列名所述。



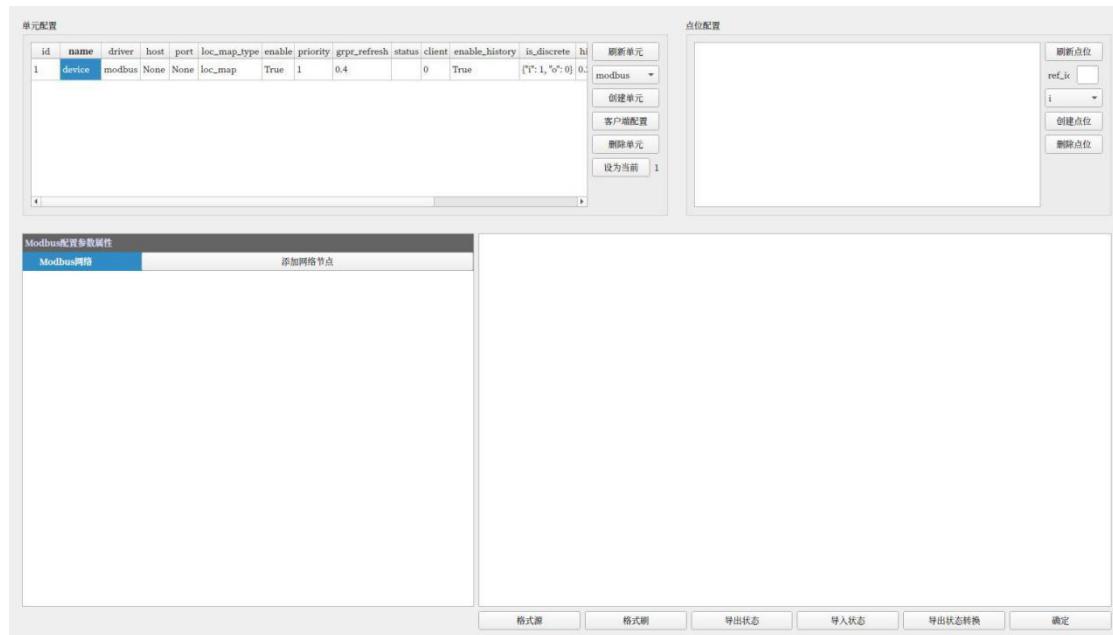
再回到 test\_sinfunc2 cfg 标签页，

点击刚创建的 device 单元，让该单元被选中

id	name	driver	host	port	loc_map_type	enable	priority	grpr_refresh	status	client	enable_history	is_discrete	1	mc
1	device	modbus	None	None	loc_map	True	1	0.4	0	True	{"i": 1, "o": 0}	0	mc	mc

点击 ，则设为当前按钮的右边会显示单元的 id，

此时显示 1，并且由于单元是 modbus，界面会在配置下半面显示 modbus 的配置窗口，



该配置窗口是针对 modbus TCP 设备开发的，通过此窗口可以方便的进行 modbus TCP 设备的配置和参数修改。

点击添加网络节点按钮，



则会出现



这里的网络地址和端口不需要修改，即使是远程客户端进行配置，网络地址也不需要修改，因为单元是运行在服务器上的，调试用 modbus 设备也是运行在服务器上的，它们可相互视为本地，因此可以使用 127.0.0.1。端口号 2701 是 modbus 调试设备默认的端口号。注意许多真实的 modbus TCP 设备的端口号是 502。

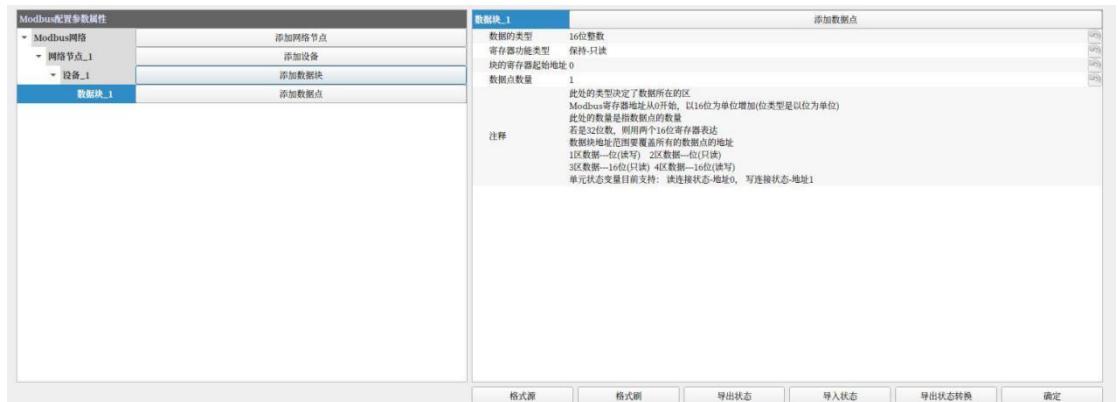
点击点击添加设备，则会出现设备配置窗口，这里就用设备地址 0。



在真实的设备中，可能是多个 modubs rtu 设备构成的 RS485 的网络，这是可以使用 modubs rtu 转 modbus tcp 的设备，由此通过此 modbus TCP 转换器可以访问多个 modbus rtu 的设备，这里要注意，rtu 设备的地址在同一 485 总线下必须唯一，此处的现场总线地址就是 rtu 设备地址，若本身就是 modbus TCP 设备，就用默认的 0，

字节顺序涉及该 modbus TCP 设备的字节的存储顺序，默认是大端存储，因为网络传输和文件存储多采用大端，也符合人的阅读习惯（高字节在前）。注意，字节顺序是针对 32 位的，因为 modubs 标准规定 16 位的都是大端存储。因此 labdcs 不支持 16 位的小端存储，因为这是非标准的 modbus，可能会有，但正规厂商不会采用非标 modbus。

点击添加数据块，

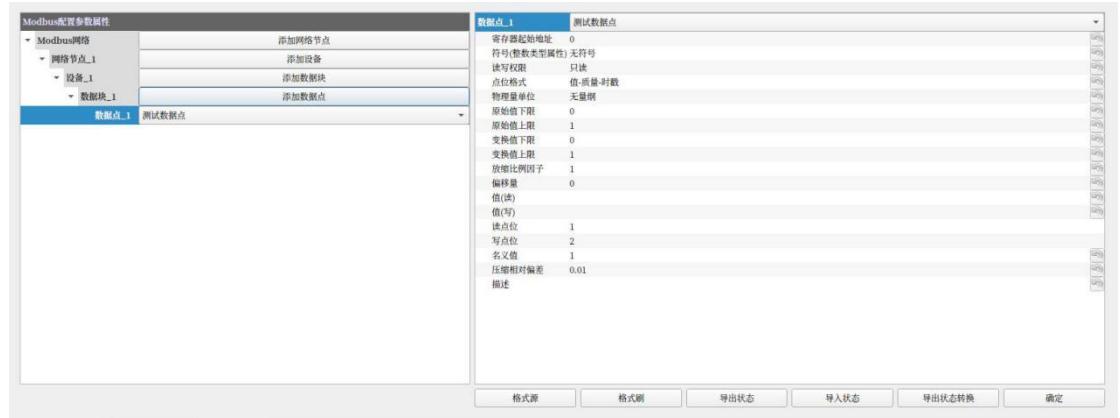


因为块的读写可以加快读写速度，因此首先要设置寄存器所处的块，即使是一个寄存器地址，也要配置块。

考虑到本处要输出的是正弦函数，因此选择数据的类型为 32 位浮点数



起始地址就用 0 地址的寄存器，数量就是一个正弦信号，因此为 1  
点击添加数据点

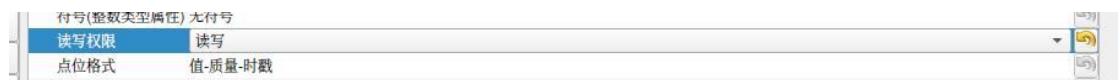


右键点击“数据点\_1”，可以进行改名



改为 func，

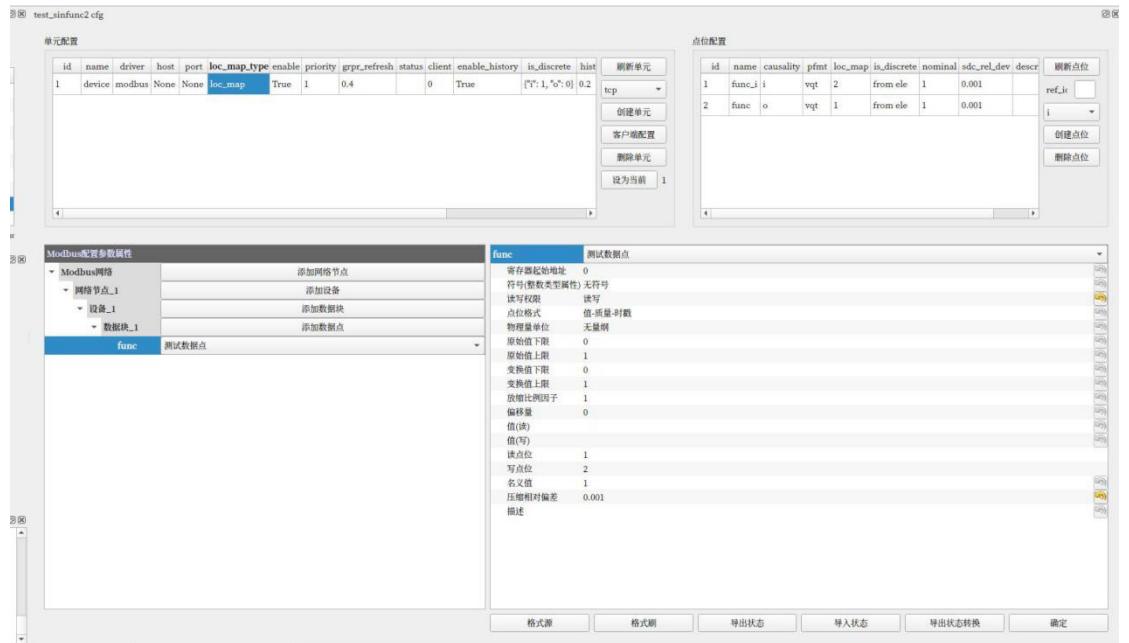
由于它的值是可以被算法修改的，因此要将读写权限设为”读写“，



由于 labdcs 会进行历史数据的压缩存储，压缩算法采用了分布式系统最通用的旋转门算法，因此要设置这里的压缩相对偏差，减小偏差则，则降低压缩率，这里可以改为 0.001，以提高数据的表达密度。

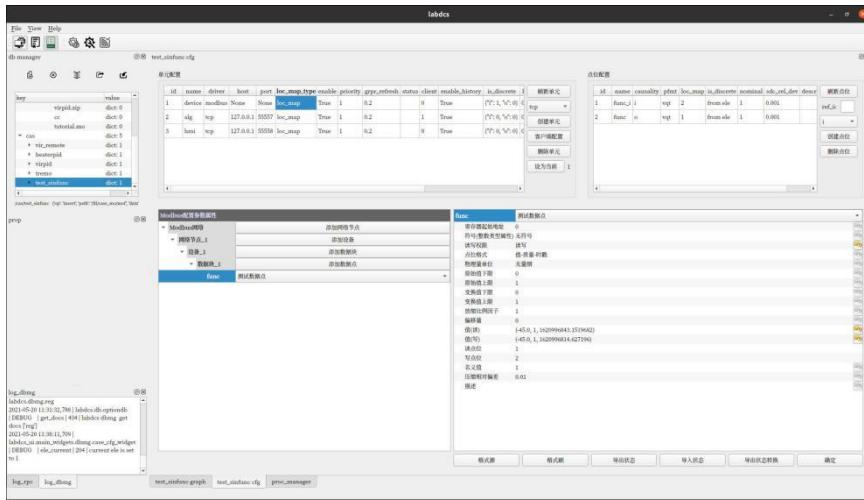


modbus 配置完成后，要点击确定，labdcs 将实现 modbus 的点与 labdcs 的点位的一致性匹配。



这时点位配置表中将出现两个点位，func\_i 对应于 modbus 的 func 的输入（写），func 对应于 modbus 的 func 的输出（读）

案例的配置如下：



### ● 创建 alg 单元

alg 是算法单元，它的功能是作为信号源输出信号，选择单元类型 tcp，点击创建单元，并改名为 alg

单元配置

	id	name	driver	host	port	loc_map_type	enable	priority	grpr_refresh	status	client	enable_history	is_discrete	刷新单元
1	device	modbus	None	None	loc_map	True	1	0.4		0	True	{"i": 1, "o": 0}	tcp	<input type="button" value="创建单元"/>
2	alg	tcp	127.0.0.1	55557	loc_map	True	1	0.4		0	True	{"i": 0, "o": 0}	None	<input type="button" value="客户端配置"/> <input type="button" value="删除单元"/> <input type="button" value="设为当前"/>

tcp 类型的单元的运行分两部分，单元本身只提供接口的外延服务，输入输出关系由内涵部分提供，内涵部分即为单元客户端的配置。先进行外延服务的配置。这里的端口号 55557 是 labdcs 自动生成的，它会被单元客户端使用，可以自由的修改，只要不出现端口的冲突。

算法单元的功能是根据输入的模式 mode，来决定输出的信号。因此它的输入是 mode，它的输出是 func，

点击选中 alg 单元，点击设为当前

test\_sintfunc2.ctg

单元配置

	id	name	driver	host	port	loc_map_type	enable	priority	grpr_refresh	status	client	enable_history	is_discrete	刷新单元
1	device	modbus	None	None	loc_map	True	1	0.4		0	True	{"i": 1, "o": 0}	tcp	<input type="button" value="创建单元"/> <input type="button" value="客户端配置"/> <input type="button" value="删除单元"/> <input type="button" value="设为当前"/>
2	alg	tcp	127.0.0.1	55557	loc_map	True	1	0.4		0	True	{"i": 0, "o": 0}	None	<input type="button" value="点位配置"/>

目前该单元中还不包含点位，选点位的属性为 i，代表要创建的点位是输入



点位配置

	id	name	causality	pfmt	loc_map	is_discrete	nominal	sdc_rel_dev	descri	刷新点位
3		new	i	vqt	None	from ele	1	0.01		<input type="button" value="ref_ic"/> <input type="button" value="i"/> <input type="button" value="Create Point"/> <input type="button" value="Delete Point"/>

双击新建点位的 name 的 new，进行改名，改为 mode

点位配置

id	name	causality	pfmt	loc_map	is_discrete	nominal	sdc_rel_dev	descri
3	mode	i	vqt	None	from ele	1	0.01	

由于存在单元客户端的配置，对同一点位要配置客户端的名字，因此在 loc\_map 列，修改客户端对应的名字，本处还是使用 mode，当然可以采用不一样的名字，这在算法编写中会用到。

点位配置

id	name	causality	pfmt	loc_map	is_discrete	nominal	sdc_rel_dev	descri
3	mode	i	vqt	mode	from ele	1	0.01	

再次选择点位的类型为输出 o，点击创建点位，并改名为 func，并修改 loc\_map 的名也为 func

点位配置

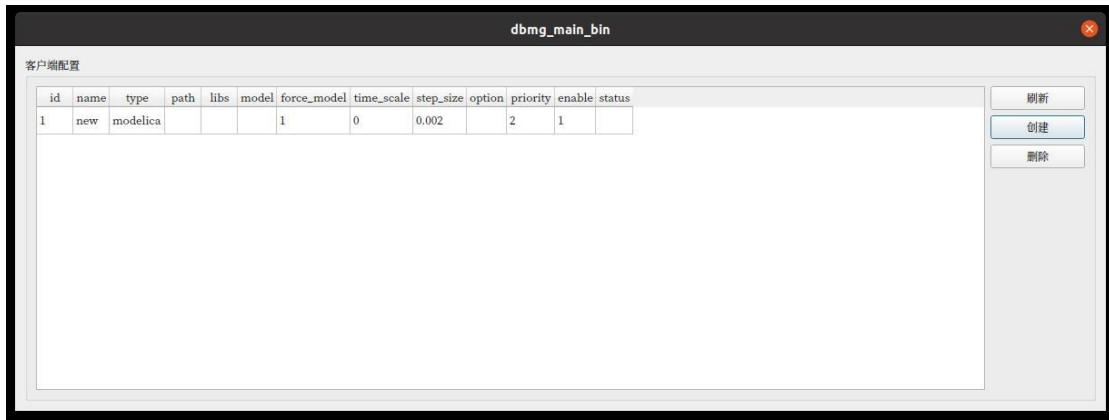
id	name	causality	pfmt	loc_map	is_discrete	nominal	sdc_rel_dev	descri
3	mode	i	vqt	mode	from ele	1	0.01	
4	func	o	vqt	func	from ele	1	0.01	

### ● 创建 alg 单元的客户端

算法单元的点位定义了它的外延，算法单元的内涵需要在它的客户端进行定义，在算法单元 2 为当前单元的情况下，点击客户端配置，

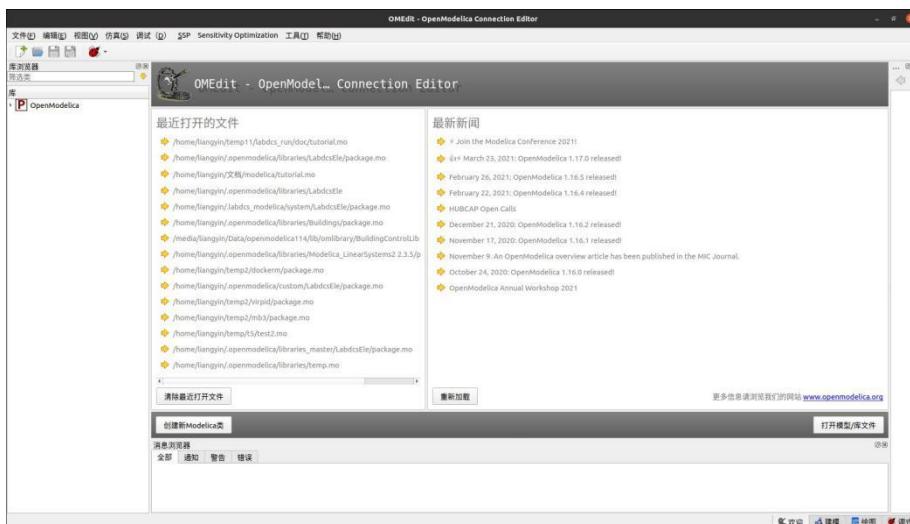


点击创建，则会创建一个客户端

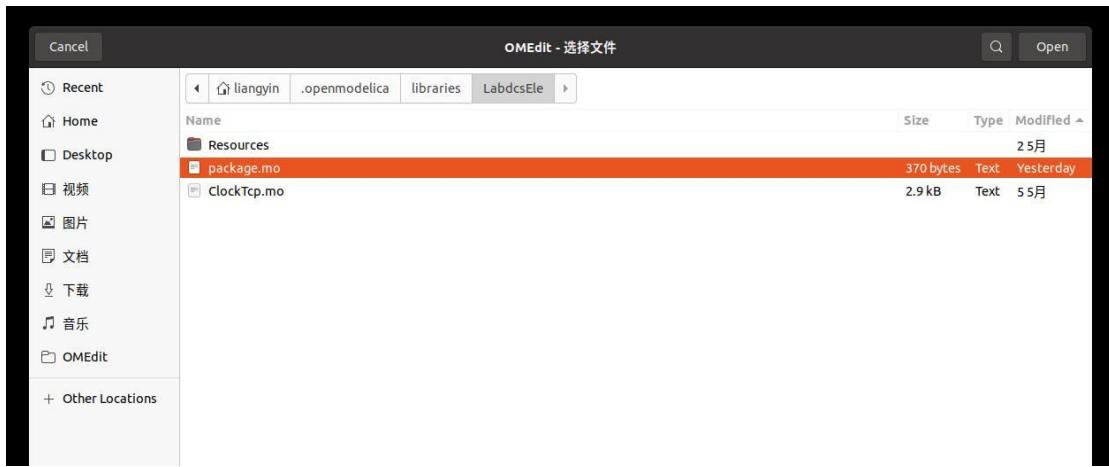


我们把名字改为 alg\_c，代表是算法的客户端，类型选择为 modelica，因此这里要提供 modelica 的源程序，并且这个源程序的文件需要在配置数据库中能找到，也就是说需要上传到 fil 的目录的某个数据库的数据集中。

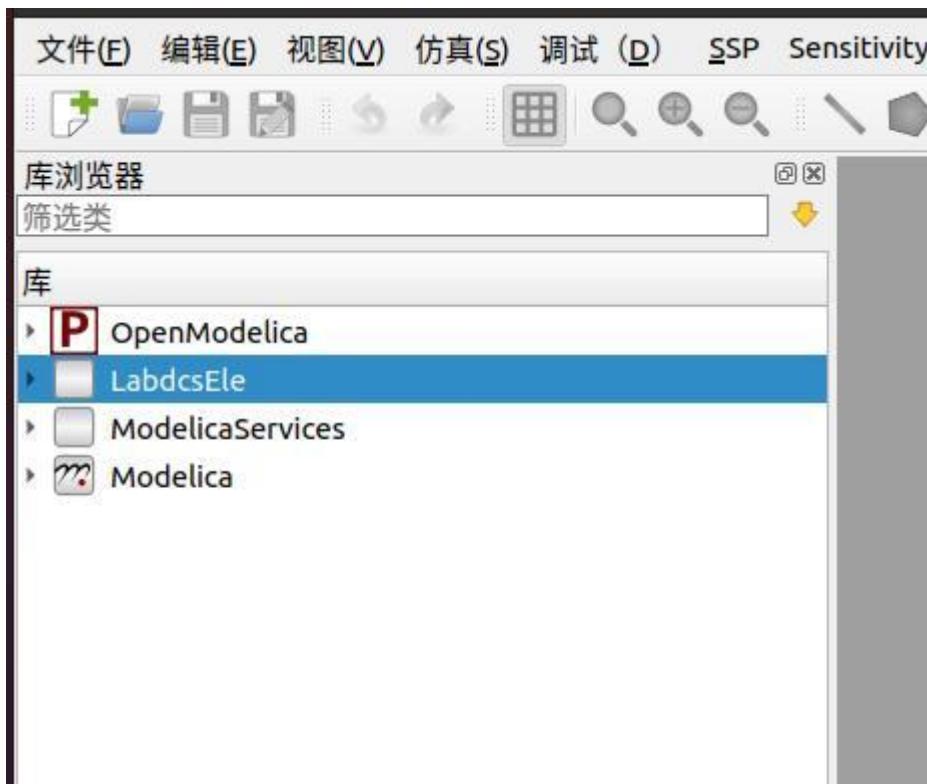
我们选择使用 openmodelica 来编辑 modelica 的源程序，在终端输入 OMEdit，打开 openmodelica 的编辑界面



点击文件—打开模型/库文件，选中 LabdcsEle 的 package.mo 的文件



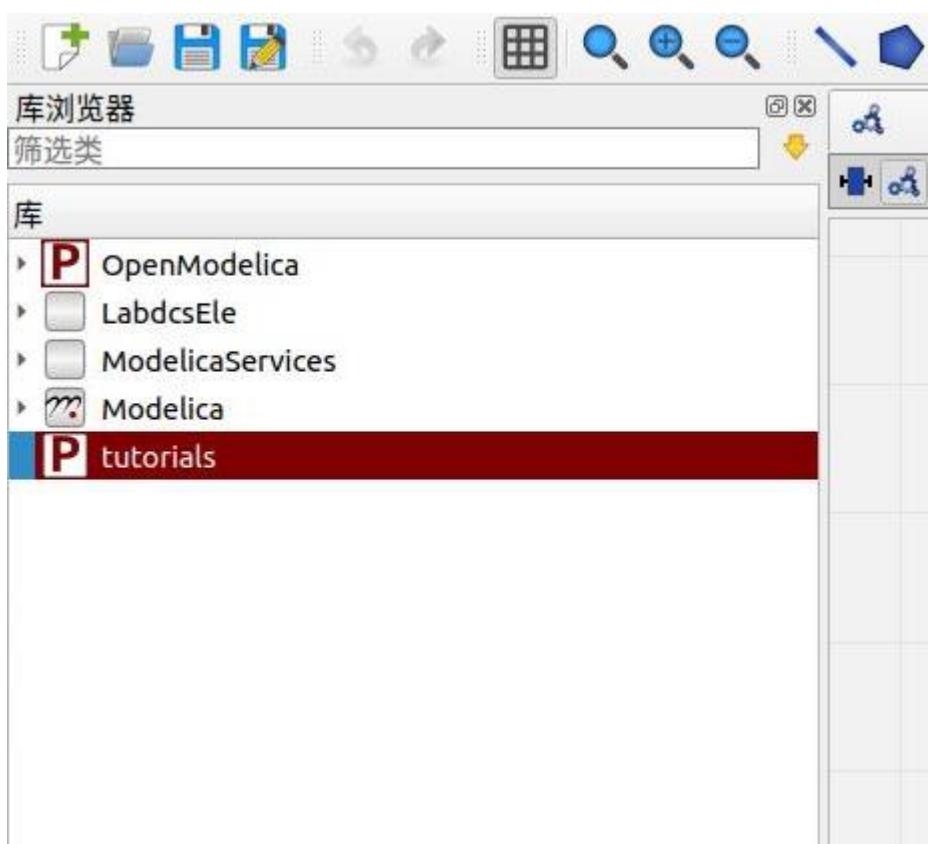
点击 open，则库目录中出现两个库 LabdcsEle 和 modelica



由于 LabdcsEle 使用的是 Modelica3.2.3 的库，因此要使用该版本的 Modelica 标准库。  
文件—新建 Modelica 类，名称用 tutorial2, 以表示与自带案例的区别，特殊化选择 Pacakge，



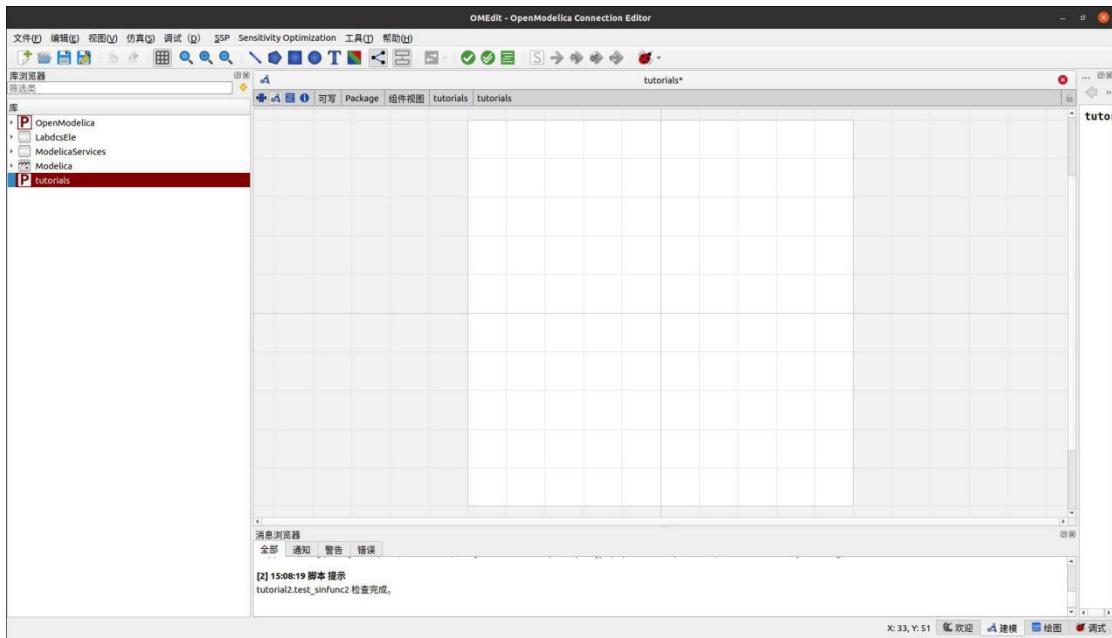
点击确定，若保存内容为一个文件被选中，则该库的内容都放到一个文件中，若没有被选中，则 openmodleica 会建立文件夹，此时可以把文件夹压缩成 zip 文件进行上传。



在 tutorial2 上点击右键，选择新建 modelica 类，使用 test\_sinfunc2 的名字，以示与自带案例区别。



点击确定后，双击 tutorial2 下的 test\_sinfunc2，则

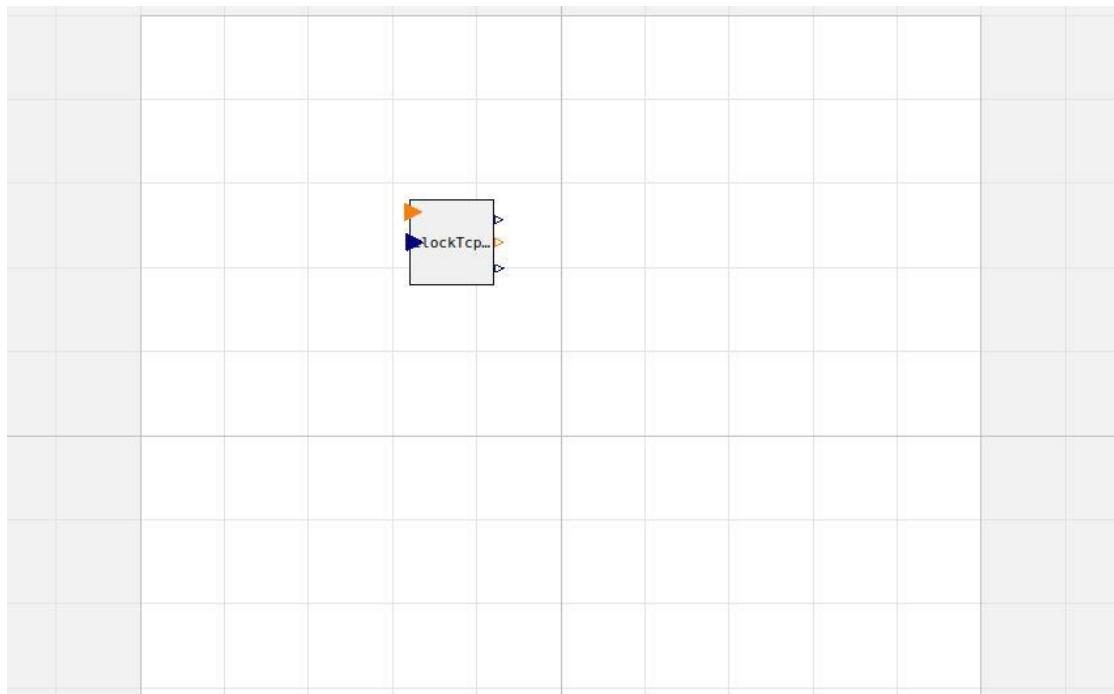


目前这是一个空的模型。

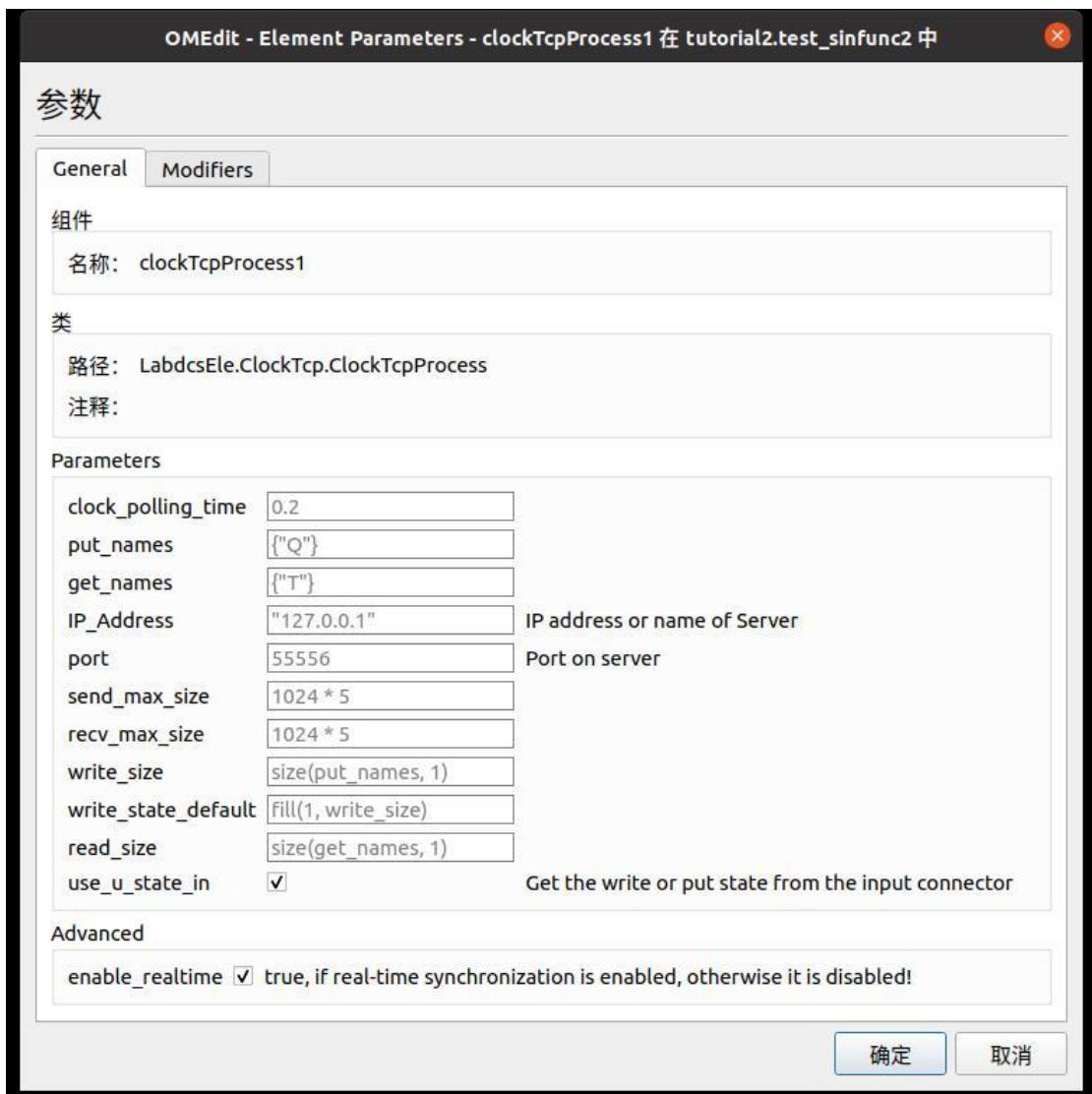
将 LabdcsEle.ClockTcp.ClockTcpProcess 拖到 test\_sinfunc2 的模型组件视图中，并命名为 clockTcpProcess1，最好不要与类名同名，



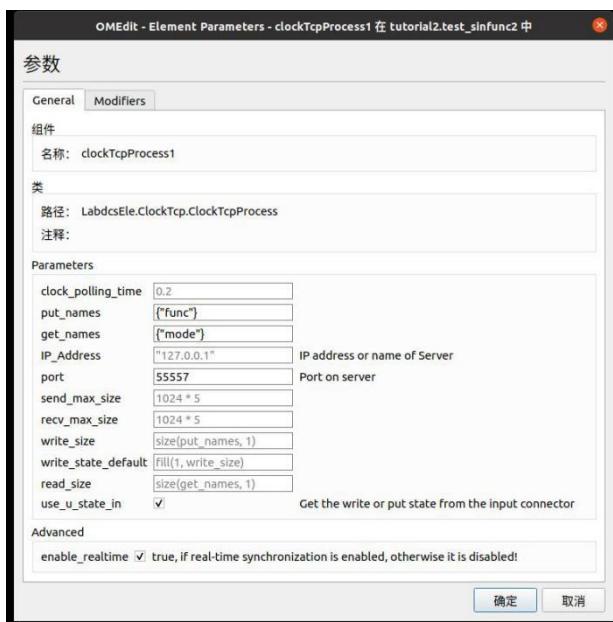
点击确定，则组件视图中出现了与 labcs 单元的接口对象



现在对该对象进行配置，右键该对象，并选择参数选项，则出现



修改 put\_names 为输出，即 {"func"}，修改 get\_names 为输入，即 {"mode"}, 这里的 port 改为为算法单元配置的端口号 55557，

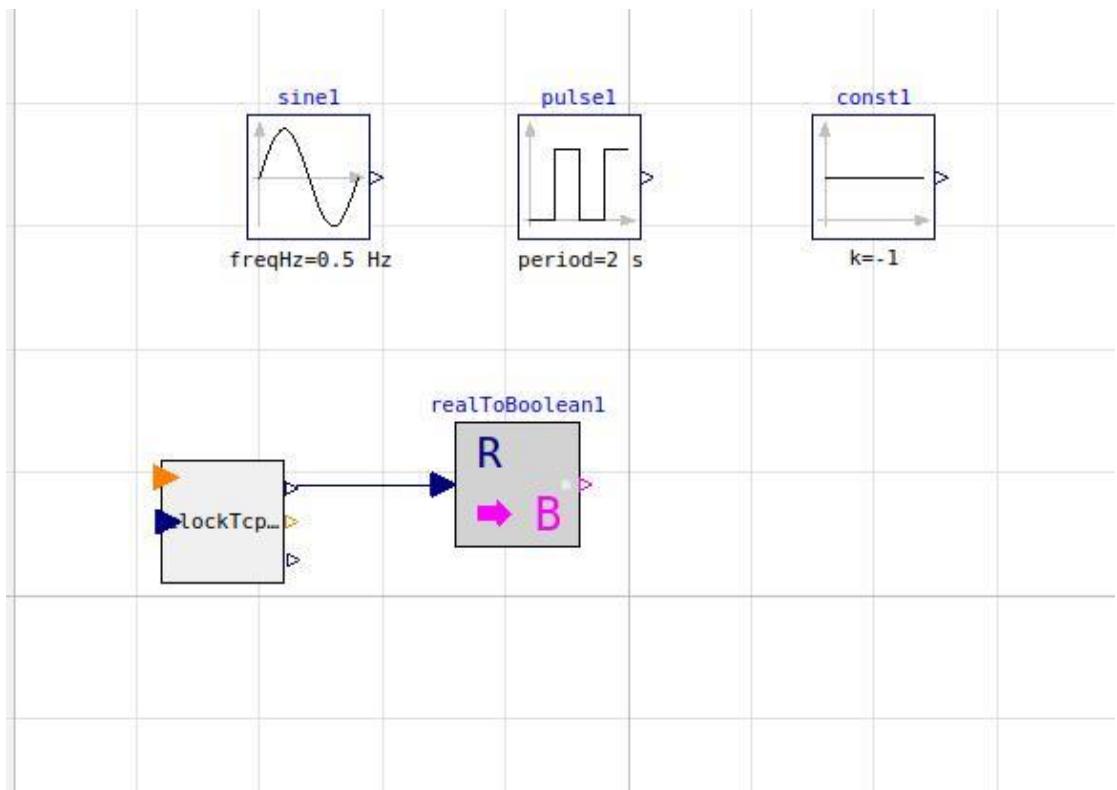


输出的 3 个端口分别是值(value),质量(quality),时间戳(timestamp),由于 labdcs 中所有的值都是浮点数存储的,在使用时要转化为所需的类型。

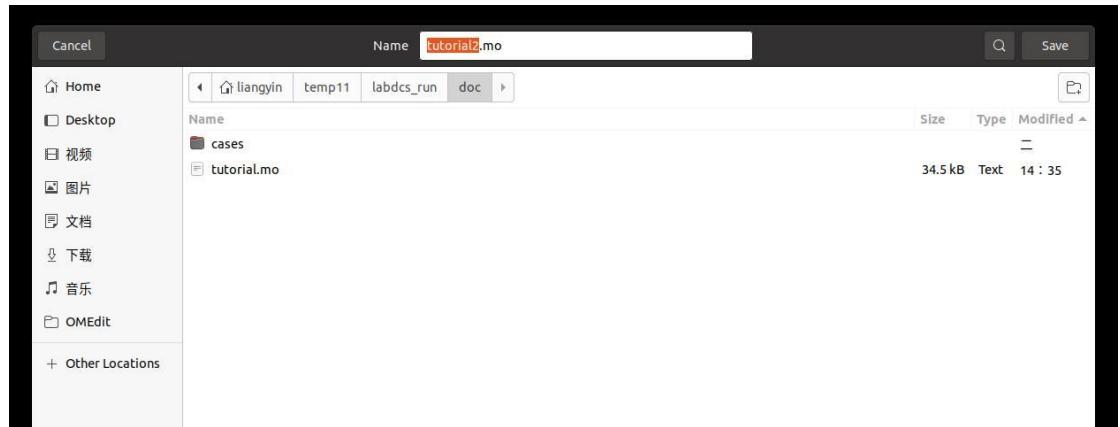
把 Modelica.blocks.math.RealToBoolean 拖进组件视图,并且将 clockTcpProcess1 对象的值输出 y\_value 与它相连接,由于只有一个 get\_names, 变量 mode 的数组位置为 1, 注意 modelica 数组从 1 开始计数。



从 Modelica.Blocks.Sources 中选择 Sine, Pulse, Constant 这三种数据源, 频率设为 0.5, 即周期为 2 秒, Constant 常数 k 设为-1, 如

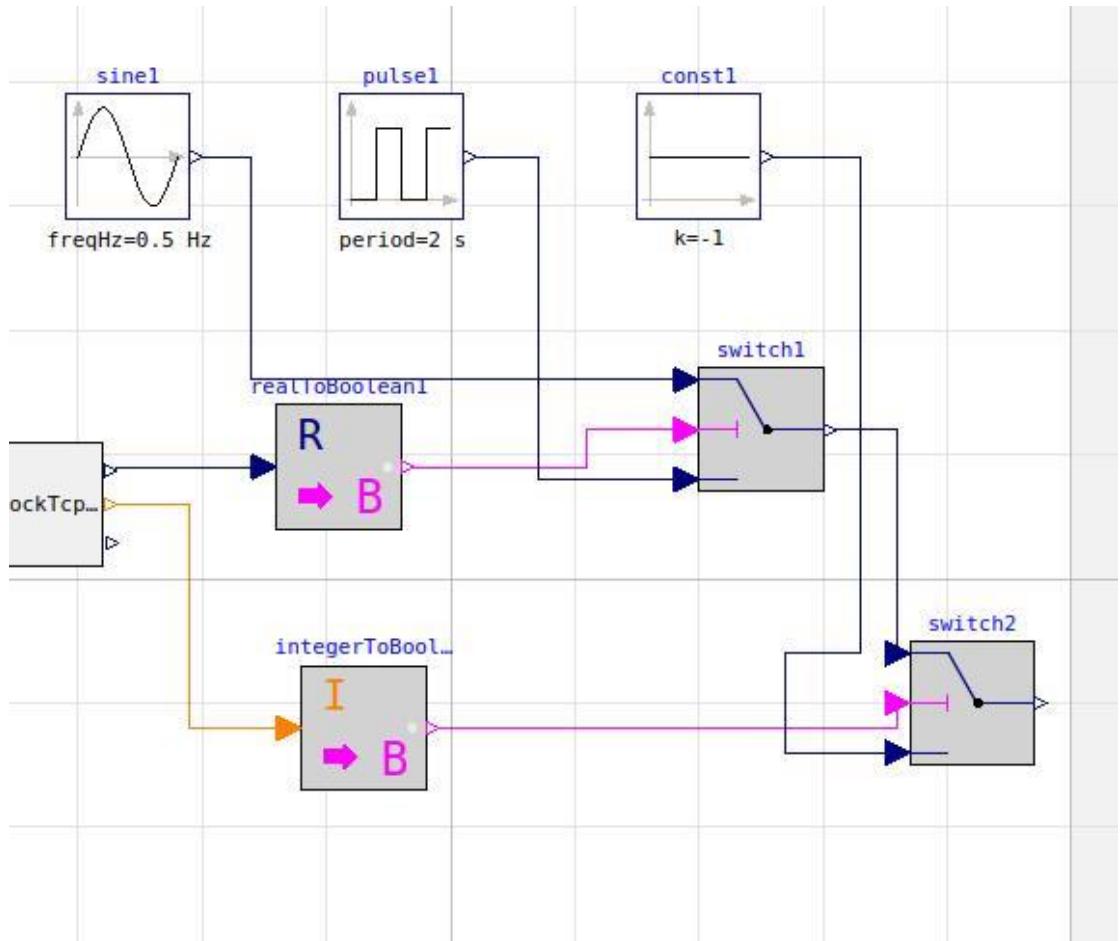


现在可以保存一下这个 tutorial2 库,  
名字不需要改, 找个目录存下



按照功能设定, mode 为 1, 输出正弦, mode 为 0, 输出脉冲, mode 信号错误或没有得到该信号, 则输出-1。这可以使用 Modelica.Blocks.Logical.Switch 来完成, 拖进组件视图两个 Switch,

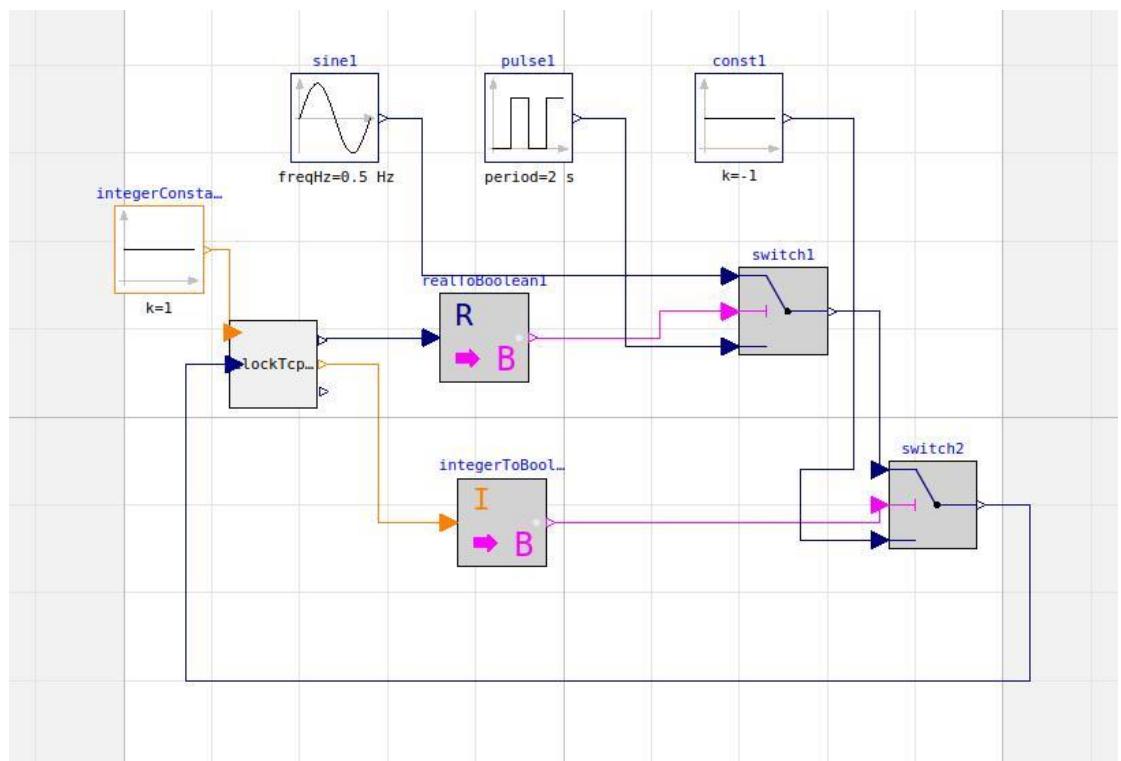
由于要使用 mode 信号的质量信息, 该量位于第二个输出口, 是一个整数量, 1 为正确, 0 为错误, -1 为未知, 因此先要转化为 boolean 量, 即完成如下的连接



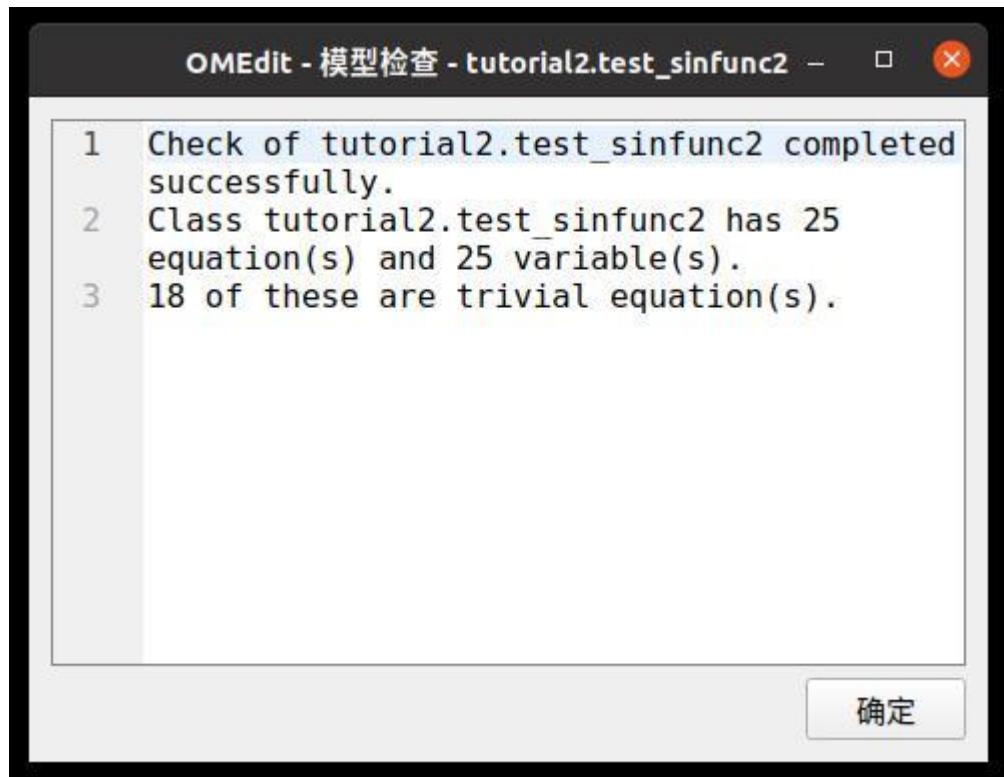
**switch1** 实现根据 mode 的切换, **switch2** 根据 mode 的质量进行切换, 然后将 **switch2** 的输出连接到 **clockTcpProcess1** 的输入, 本案例中对应的就是算法单元的输出 func。在理解上要特别注意, **clockTcpProcess1** 对象以为这外界的环境, 因此它的输出对应了单元的输入, 它的输入对应于单元的输出。



同时，对于该点位的输出质量，可以用常整数 1 来表达。则总的模型的组件图为



点击 ，进行模型检查，



若显示检查成功，方程数等于变量数，则可就可以保存上传了。

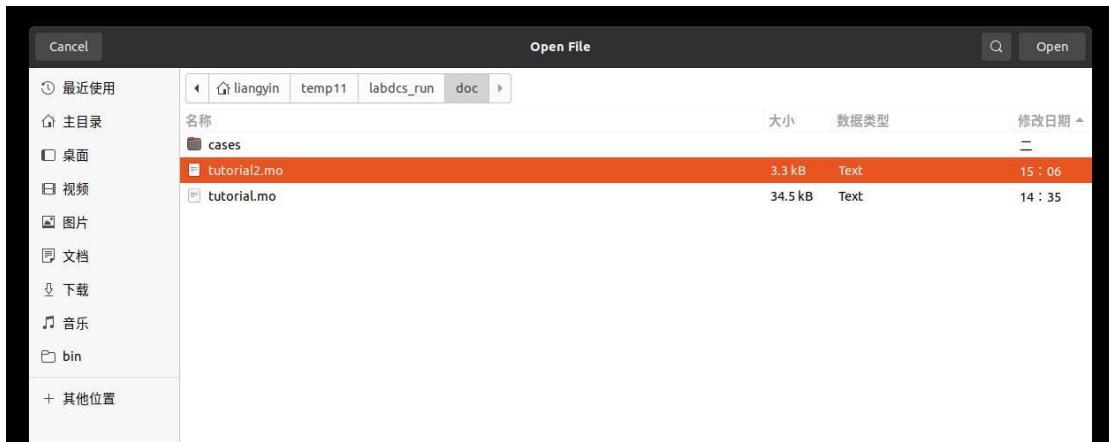
上传 tutorial2.mo 的模型库文件，在 labdcs 管理界面软件中点击 fil 下 libs 中的 modelica

The screenshot shows the "db manager" interface with a tree view of a database structure. The structure is as follows:

- key
- value
- \_id reg
- cfg dict: 1
- fil dict: 1
  - libs dict: 1
    - modelica dict: 1
      - tutoria... dict: 0
    - cas dict: 2
      - test\_sinfunc dict: 1
      - test\_sinfunc2 dict: 1

At the bottom of the interface, there is a command line: /fil/libs/modelica {op: 'insert', 'path': '/cas', 'data': 'test\_sinfun'}

点击 ，选中刚建立的 tutorial2.mo 的文件



点击 Open,进行模型文件的上传。

The db manager interface displays a configuration table with the following data:

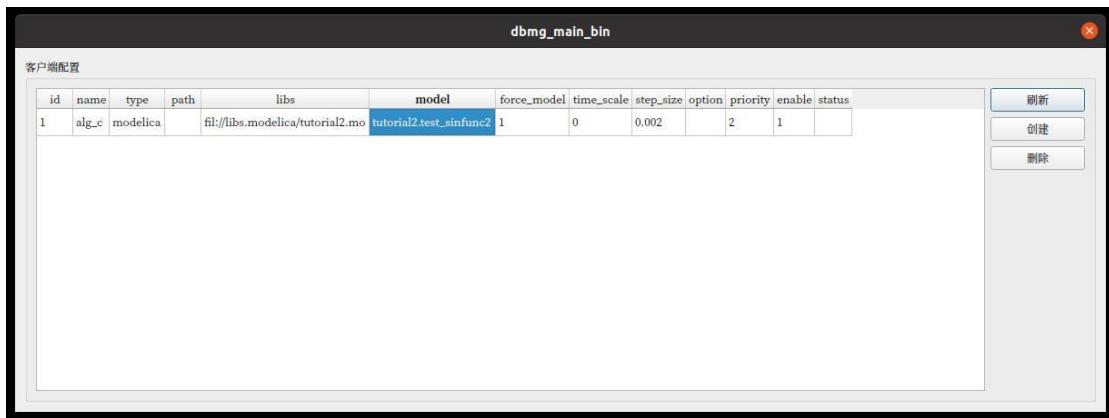
key	value
_id	reg
cfg	dict: 1
fil	dict: 1
libs	dict: 1
modelica	dict: 2
tutorial.mo	dict: 0
tutorial2.mo	dict: 0
cas	dict: 2
test_sinfunc	dict: 1

Below the table, a command line window shows the path: /fil/libs/modelica/tutorial2.mo {op: 'insert', 'path': '/fil/libs/mo'}

这时可以进行单元 2 的客户端的完整的配置：

将 libs 列设置为新上传的文件地址 fil://libs.modelica/tutorial2.mo

将 model 列设置为算法模型的 modelica 的路径， tutorial2.test\_sinfunc2

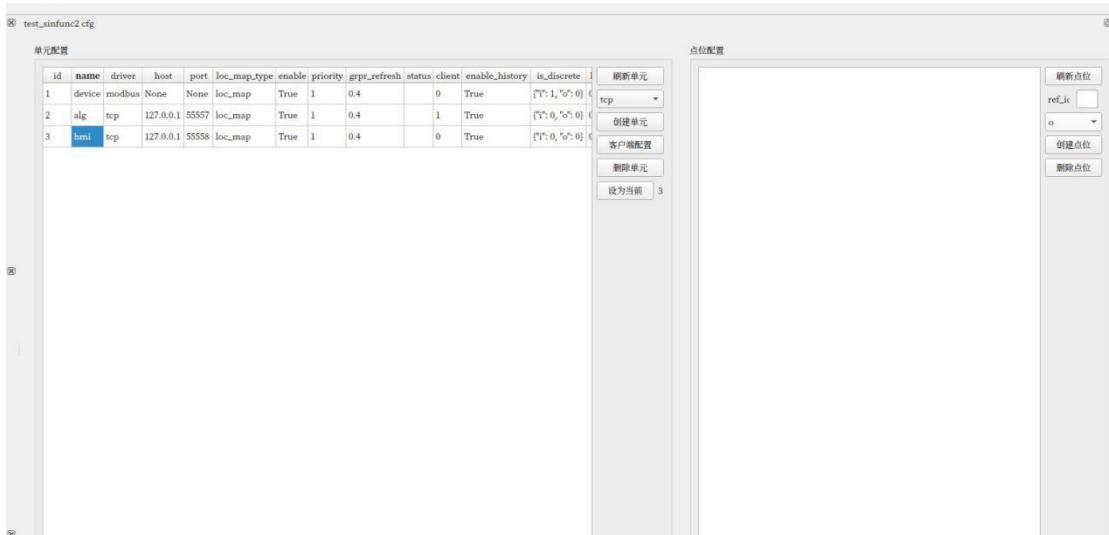


由此完成了算法单元的创建。

### ● 创建人机交互 hmi 单元

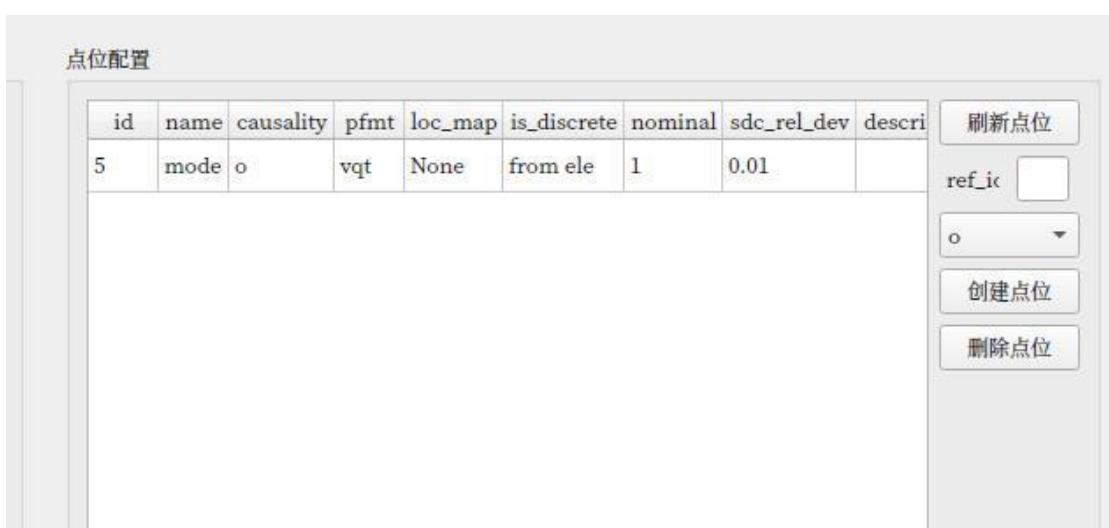
hmi 的作用是进行人机交互，它的输出变量为 mode，它控制这算法的运行模式。

将 hmi 单元设为当前，



labdcs 自动分配了端口为 55558，这个端口会在人机界面中用到。若是远程配置，则要将地址改为 0.0.0.0

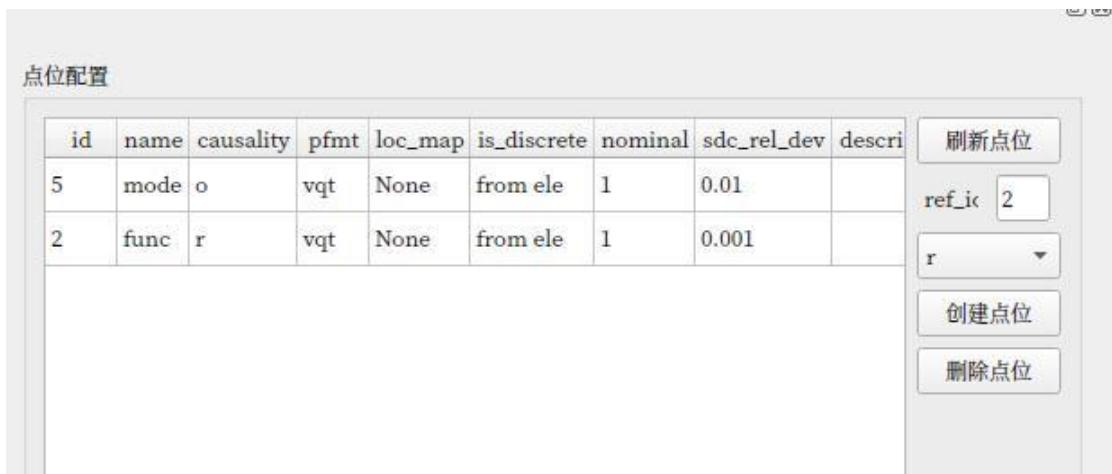
选中点位输出 o，点击创建点位，并改名为 mode。



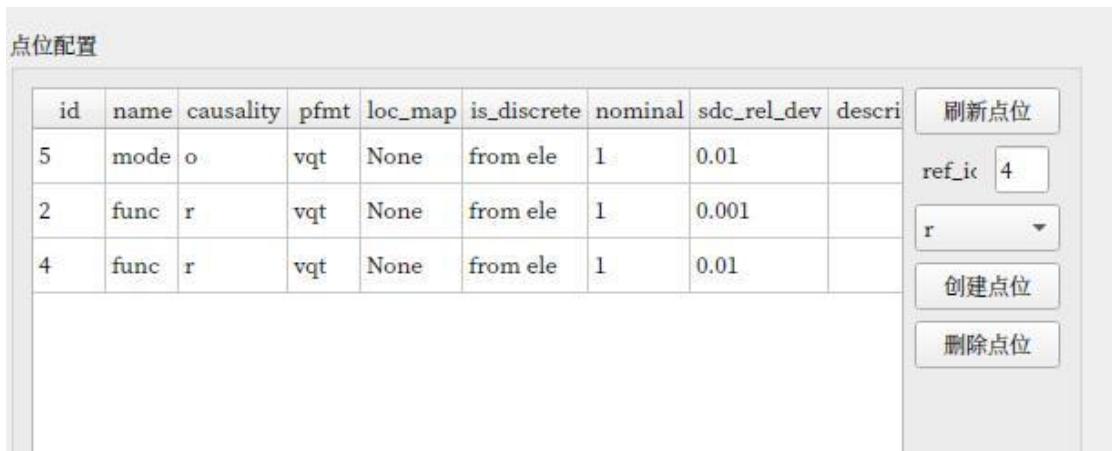
同时人机界面还需要获取设备 device 输出电位 2 和算法 alg 的输出点位 4，这可以通过

引用点位实现。

选择点位类型为 r, 代表是引用, 在 ref\_id 中输入, 2, 点击创建点位,



同理引用点位 4



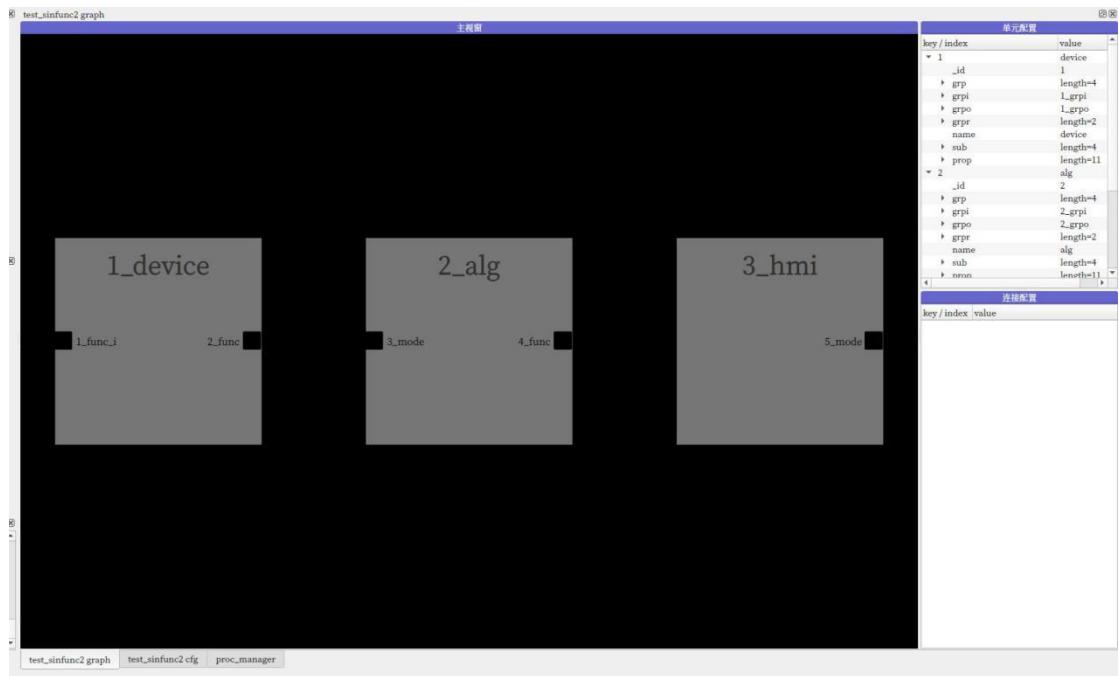
这样就完成了人机界面的单元的外延创建。

至此就完成了单元的创建。下面就是要配置单元点位之间的连接性。

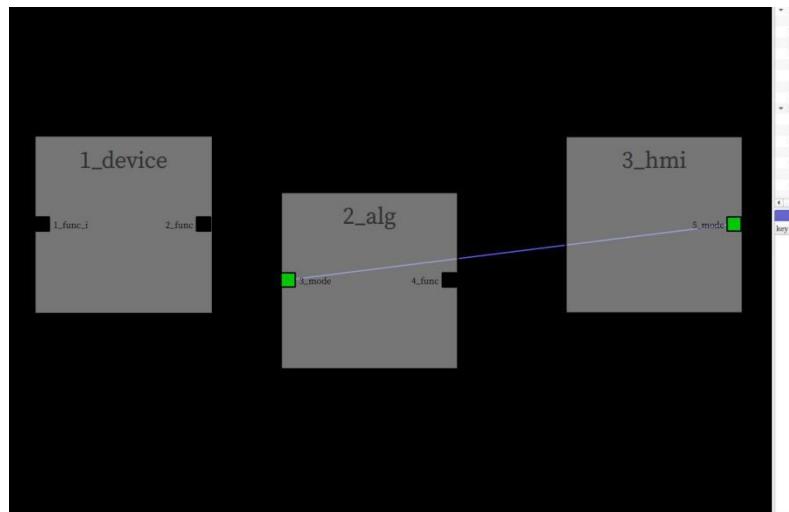
### ● 连接性的配置



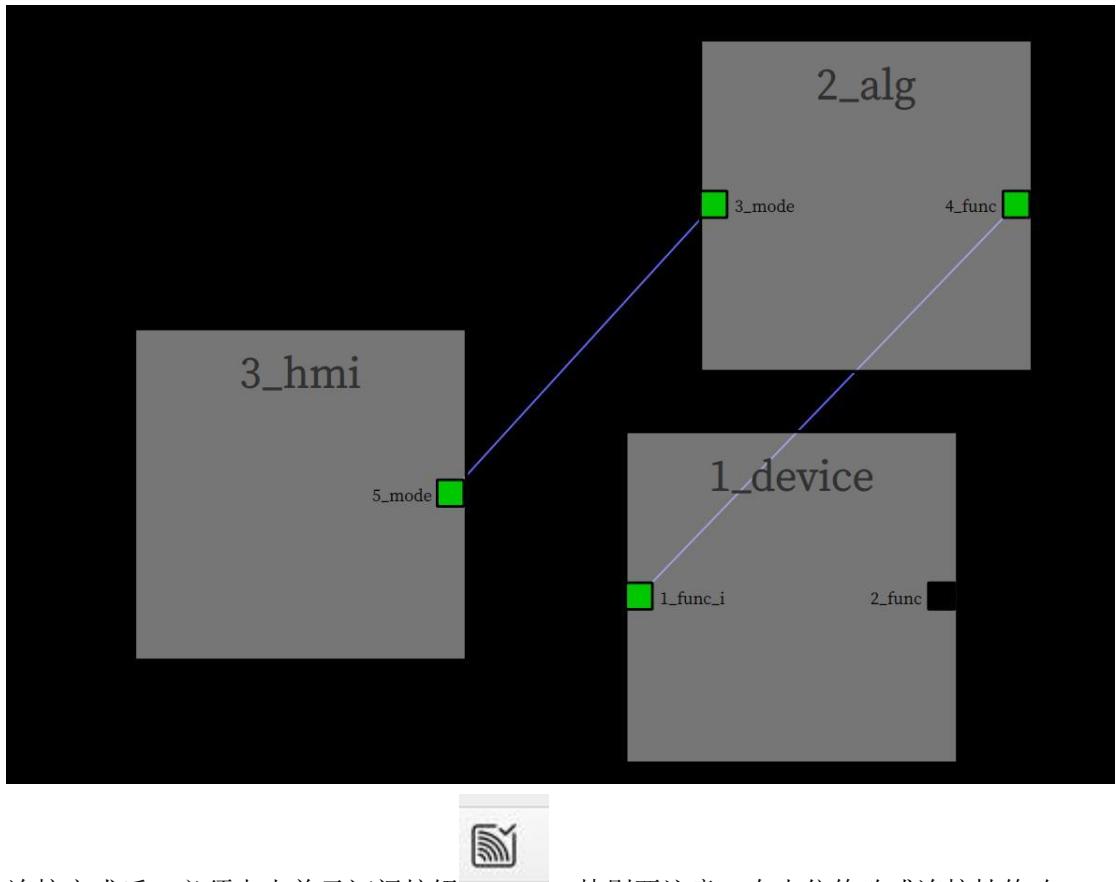
点击工具栏的 ，刷新配置图 graph，并点击 test\_sinfunc2 graph 标签



将 hmi 的 5\_mode 点位与 alg 的 3\_mode 点位连接(在 5\_mode 点位上拖住鼠标到 3\_mode 上),



同样的连接, alg 的 4\_fun 对应于 device 的 1\_func\_i,



连接完成后，必须点击单元订阅按钮 。特别要注意，在点位修改或连接性修改后，都要重新建立单元间的信息订阅系统，因此都要点击此按钮。

### ● 案例配置补充说明

本处的算法部分是根据 mode 来确定输出的行为，若 mode 为 1，则输出为 sin 函数，若 mode 为 0，则输出 pulse 函数，若没有进行 mode 设置，即 mode 的数据 quality 为 False，则输出 -1 的信号。

没有安装 openmodelica 的，需要安装 openmodelica 的编译器。

用 OMEedit 进行上述 modelica 算法的编写。打开 OMEedit，并将安装文件中的 labdcs\_modelica 解压缩到 MODELICAPATH 的目录下，OpenModelica 的 MODELICAPATH 变量设置的是 /home/liangyin/.openmodelica/libraries。对于 OpenModelica 软件，设置环境变量 OPENMODELICALIBRARY，就意味着设置 MODELICAPATH 变量。因此也可以把库解压缩某个目录，在启动 OMEedit 之前，export OPENMODELICALIBRARY 变量。

使用 OMEedit 打开解压缩中的库 LabdcsEle，该库提供了 modelica 的模型或算法与 labdcs 平台的接口。通过它，modelica 模型可以获得单元的输入，并给出单元的输出。要注意的是，LabdcsEle 库只与 modelica3.23 库相兼容，因此你使用的 modelica 的标准库必须是 3.2.3 的版本。

#### 4.1.3 案例运行



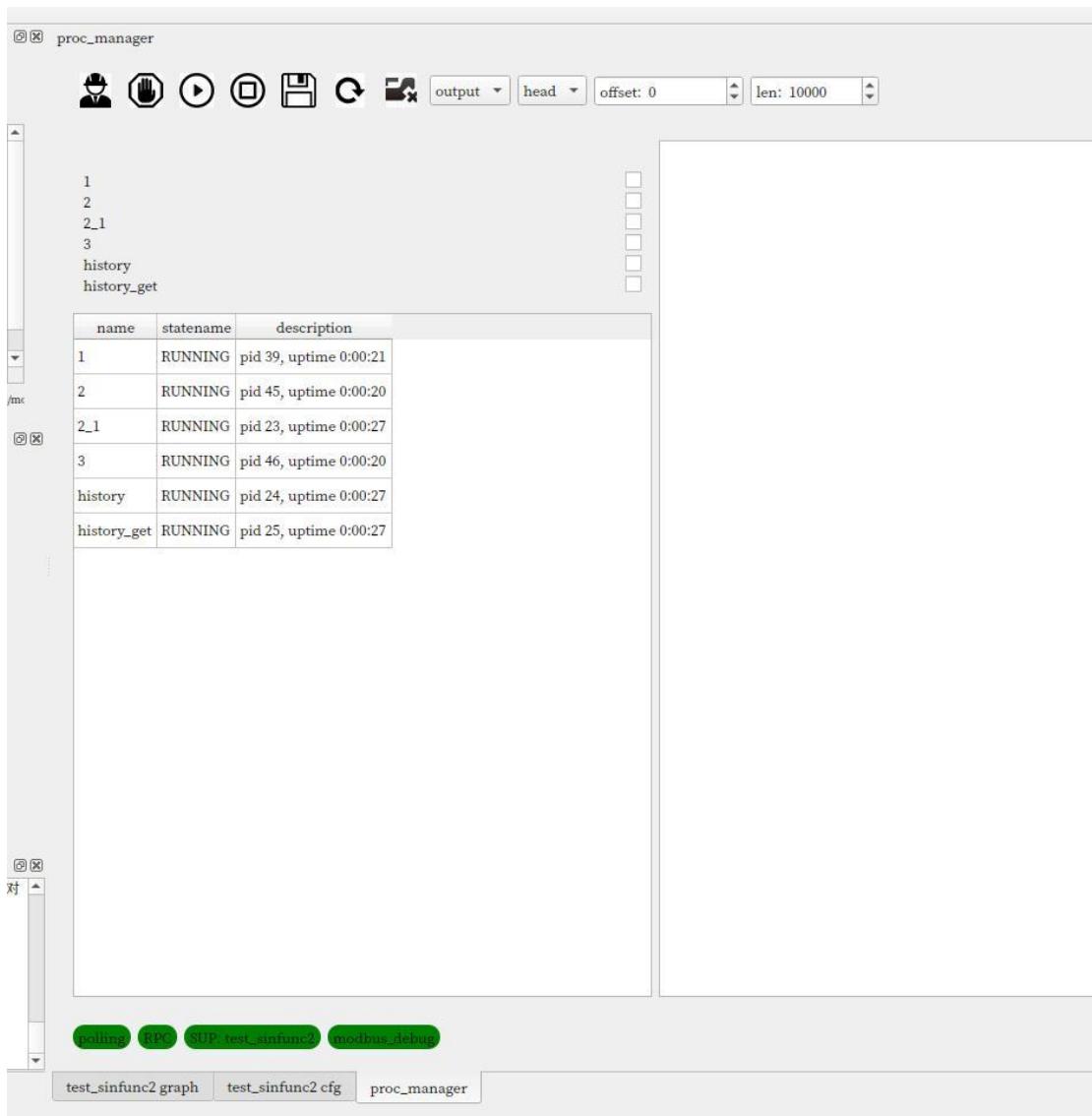
案例运行时，要打开工具栏中的 modbus\_debug。



然后，在 proc\_manager 界面，点击



来启动项目。当一切都正确时，会有如下界面

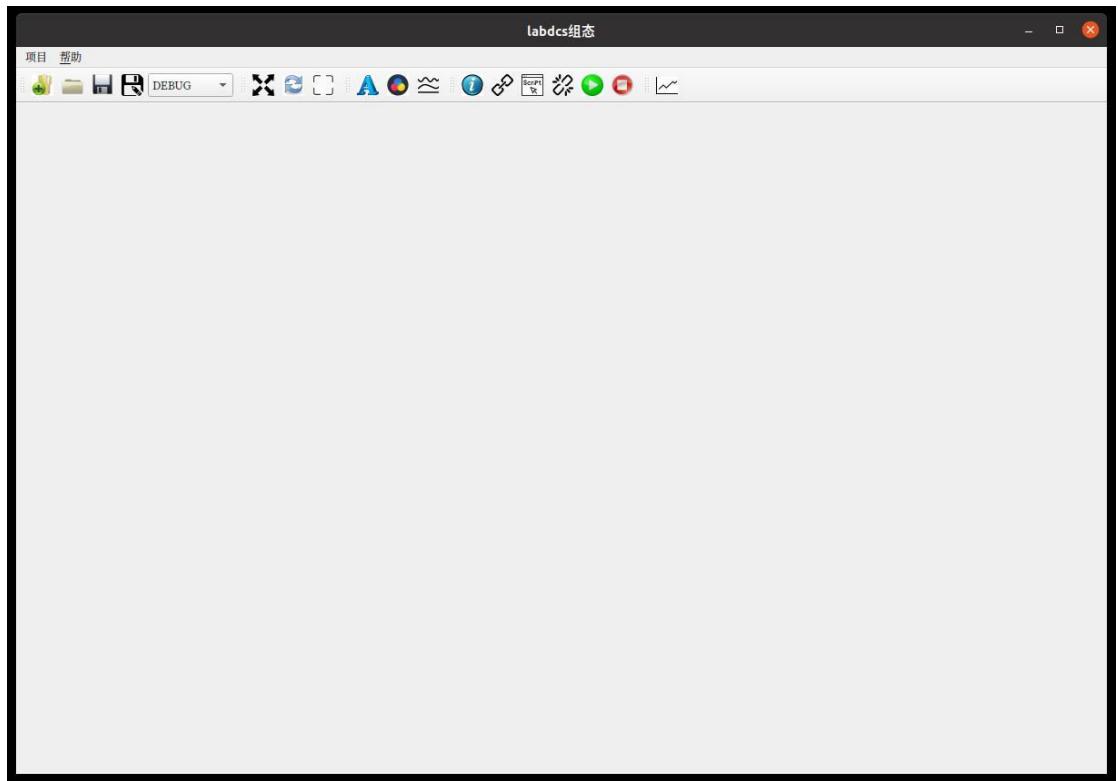


#### 4.1.4 案例的人机界面

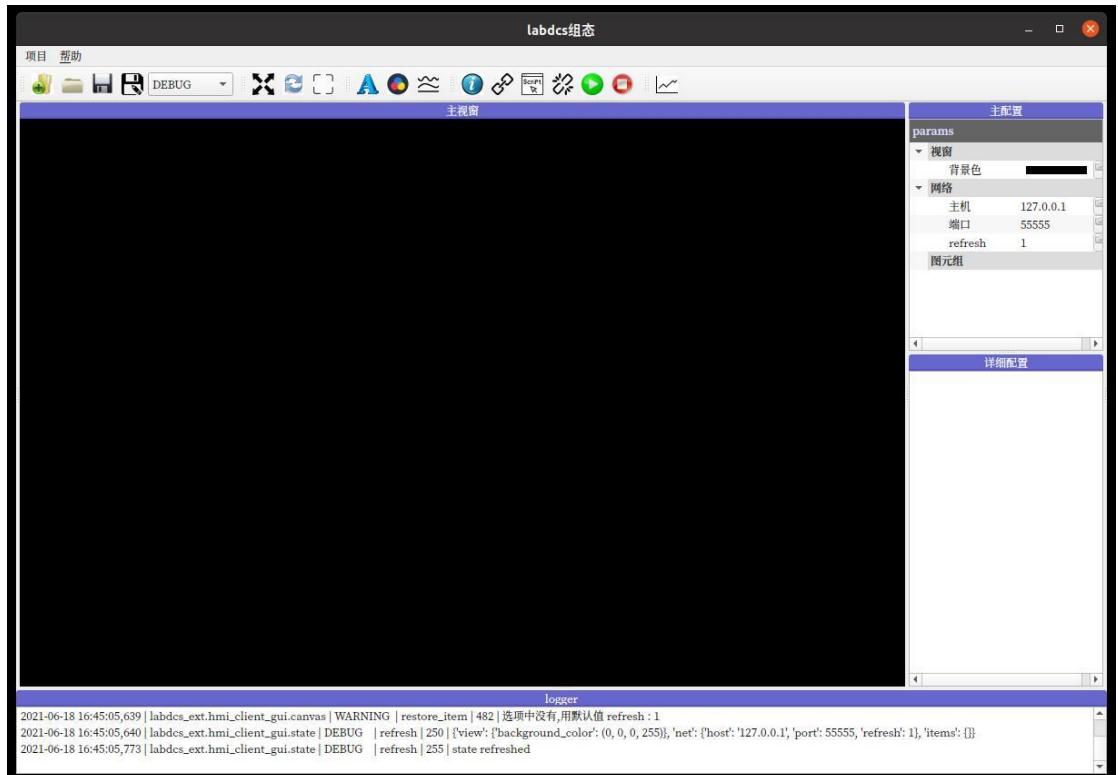
##### ● 人机界面的配置

人机界面本质上是 labdcs 单元的内涵表达，是单元的客户端。

进入终端运行，hmi\_client\_bin，



点击 ，新建一个案例的人机界面



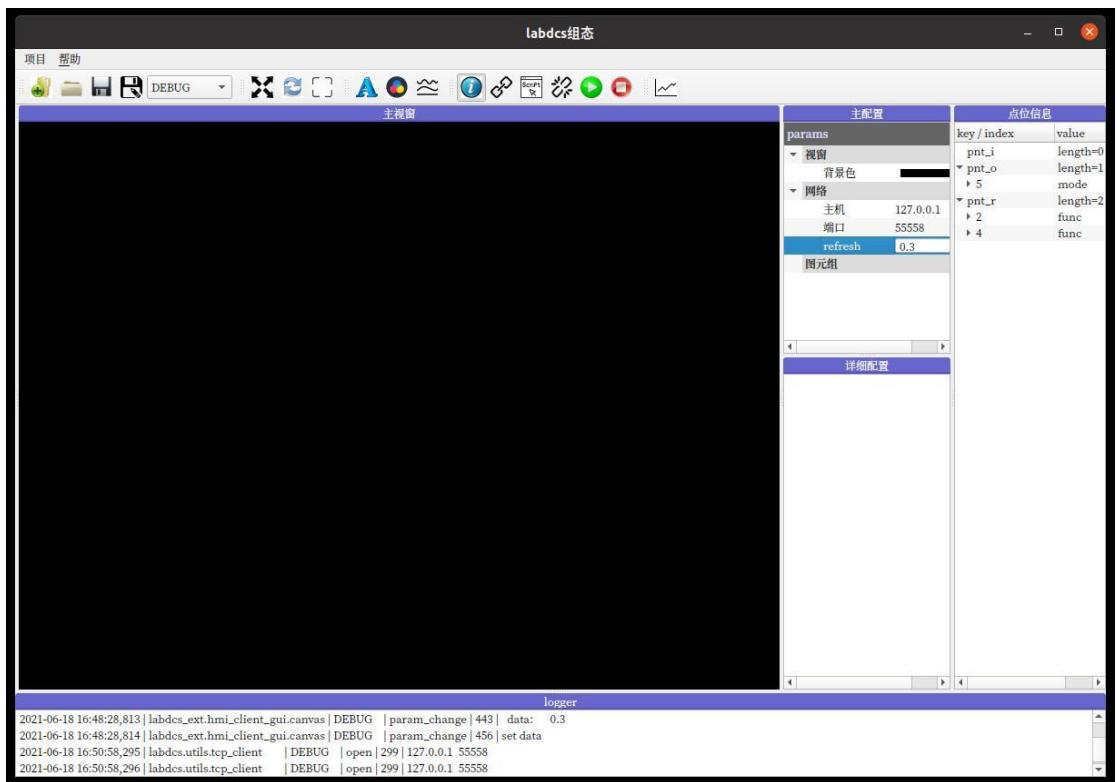
由于之前人机界面的端口为 55558

3	hmi	tcp	127.0.0.1	55558	loc_map	True	1
---	-----	-----	-----------	-------	---------	------	---

这里的端口也是 55558，主机是填写服务器的地址，若人机界面运行的程序与 labdcs 服务器程序不在一台机器上，则要填写 labdcs 服务器在网络中的地址。刷新时间是指客户端读取数据的周期。



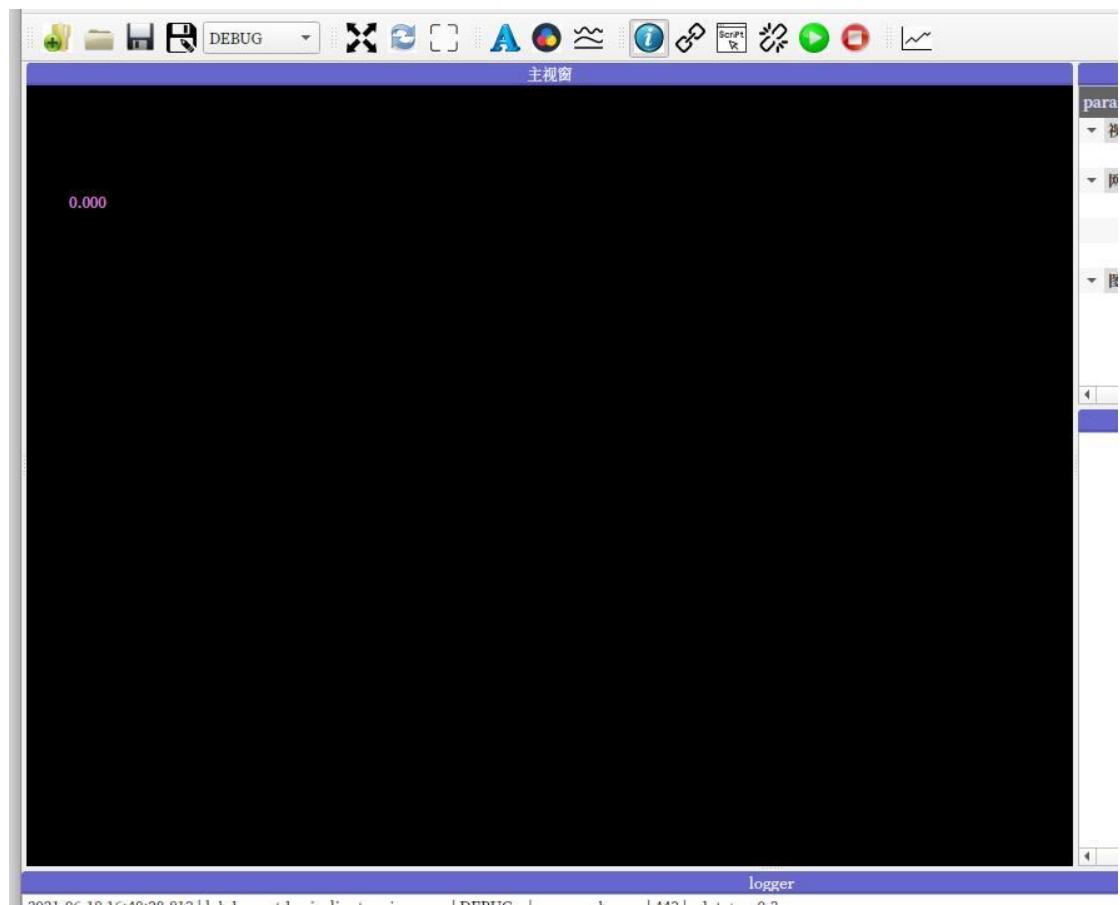
这时，点击 ，若服务器运行正常，则会在右侧显示点位信息



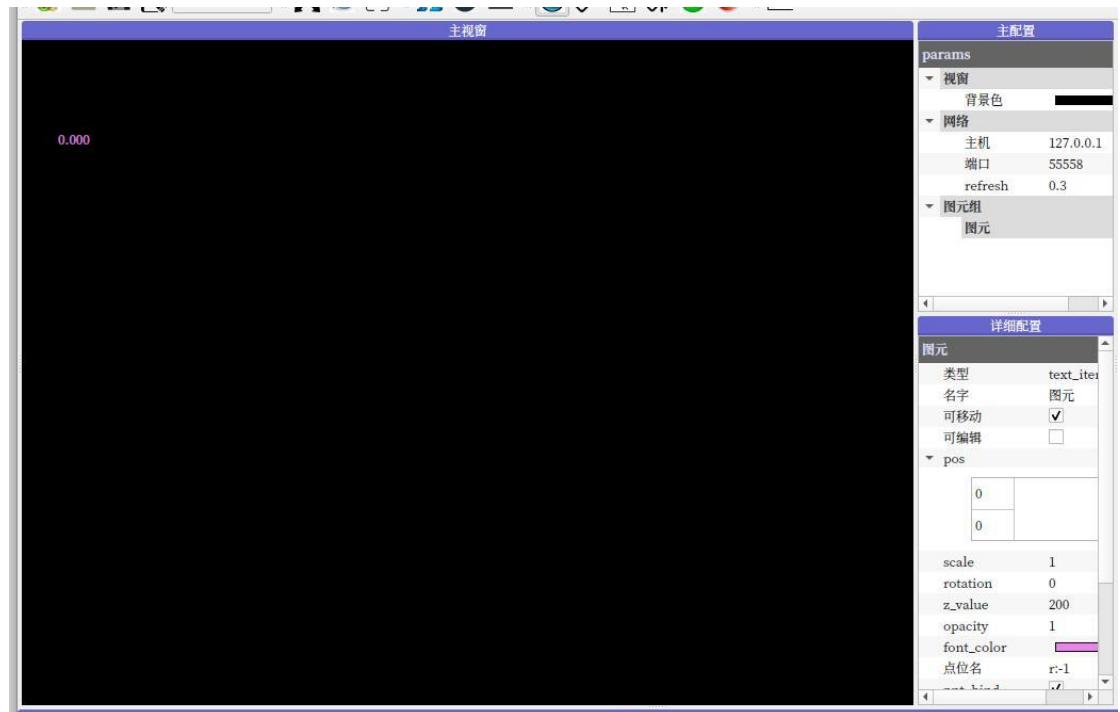
点位信息被分为输入点位，输出点位和参考点位。



点击 ，则在主窗口中生成一个文字图元



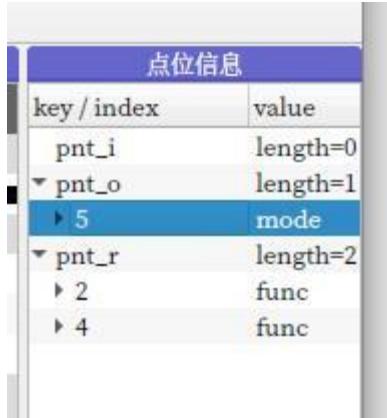
点击此文字图元，图元属性会显示在右侧，



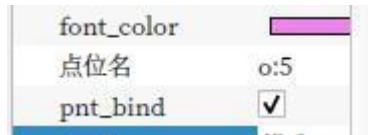
修改属性中的 prefix,为模式



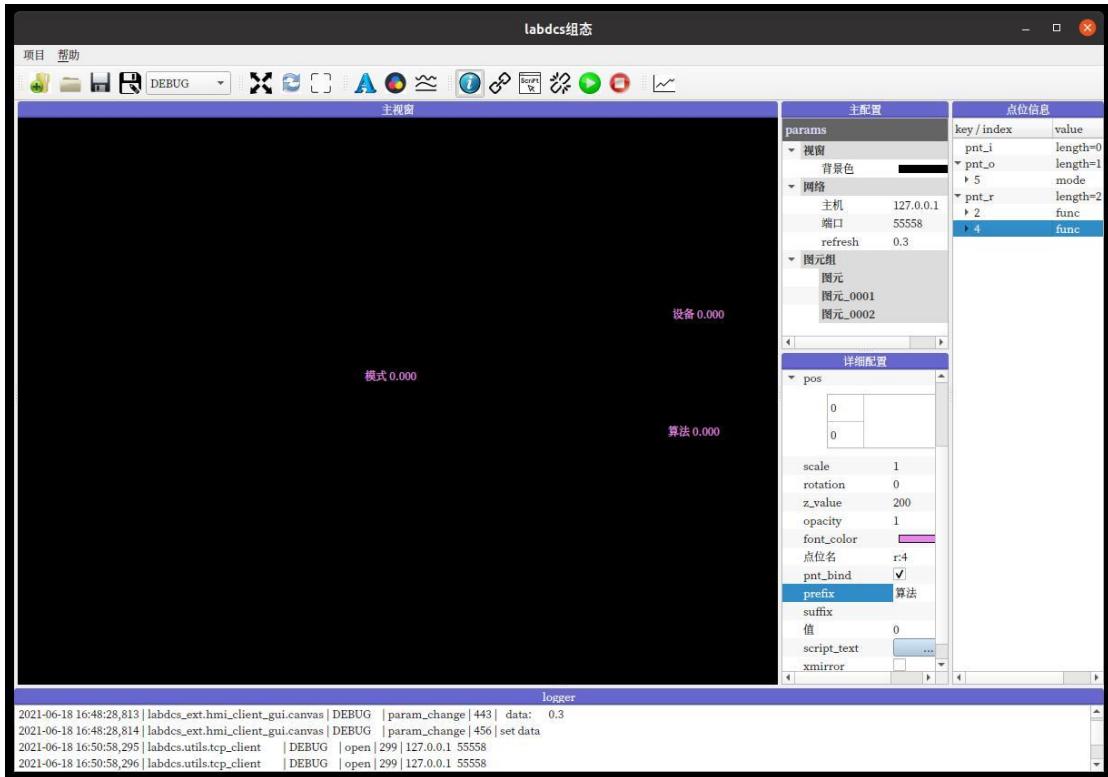
点击点位信息中的点位 5，



在点击工具栏中的 ，就可以实现该图元与点位 5 的捆绑，这时图元的点位名属性中将显示 o:5，表示该图元与输出点位 5 进行了捆绑。



同理可以建立图元与参考点位 2 和 4 进行捆绑，则



这时可以点存盘，保存该配置文件，配置文件的文件后缀名为 hmi

### ● 人机界面的运行

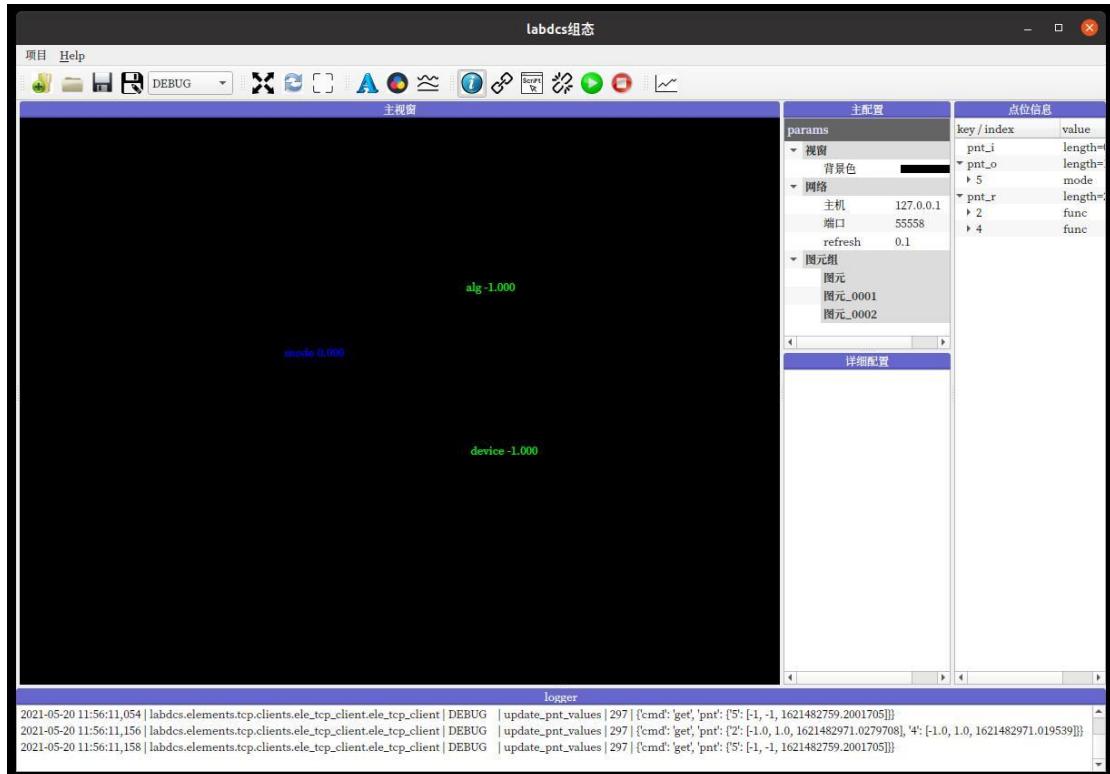
人机界面的运行，意味着周期性的读取输入和引用点位，并且实时的接受用户输入，进行单元的输出，或者运行脚本，进行单元的输出。



点击 ，让人机界面客户端进入运行模式。



最后，打开人机界面，并对输出和参考进行如下的配置，并点击

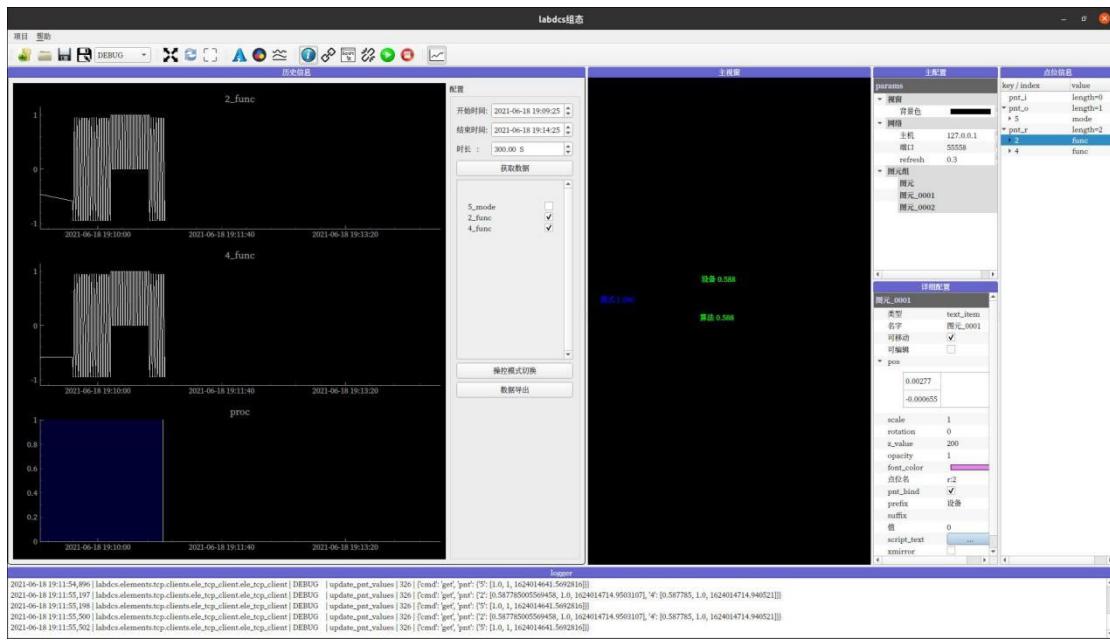


在人机界面操作之前，算法和设备的输出都是-1，这正如算法所要求的，因为此时还没有 mode 的设置，则 mode 的默认的 mode\_qaulity 为 0。

现在右键人机界面的 mode 标签，设置 mode 的值为 1，这时 alg 和的 device 的 func 值都会是正弦变化。可在设 mode 为 0，则会是脉冲变化。



打开历史显示，，并配置要显示的时间和点位，可得



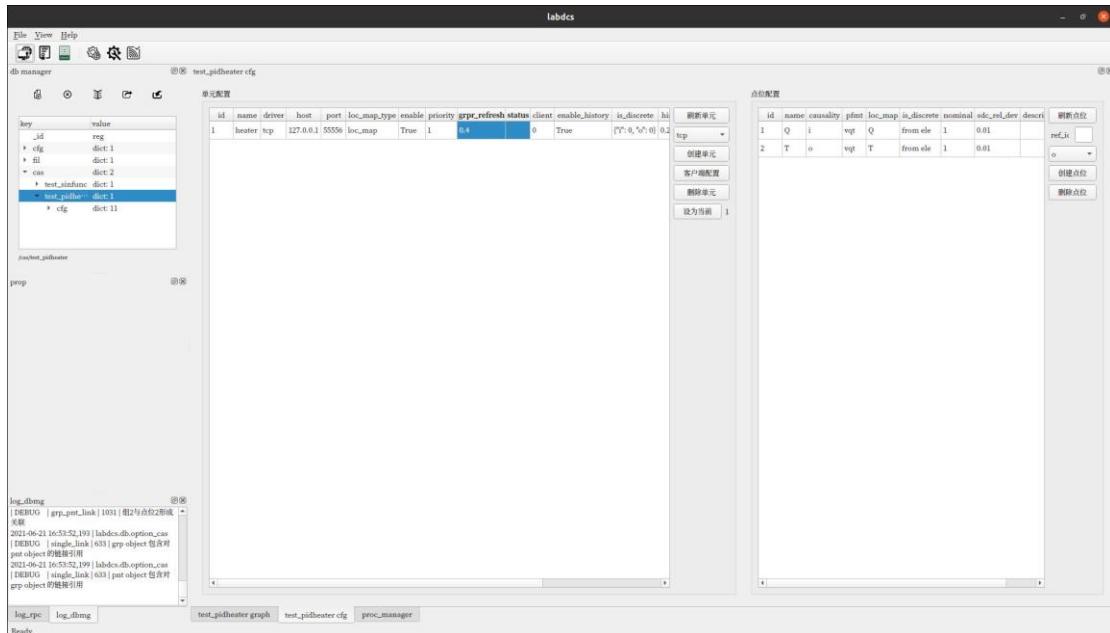
## 4.2 案例 test\_pidheater

### 4.2.1 案例设计和接口

本案例是进行加热过程的控制，此处的加热过程是一个用 modelica 建立的虚拟的系统，为了更具有真实性，该虚拟系统的进出口都与一个 modbus 系统想对接，相当于 modbus 系统承担了对虚拟系统的数据采集，以及控制信号的输入。

#### ● 创建加热系统单元

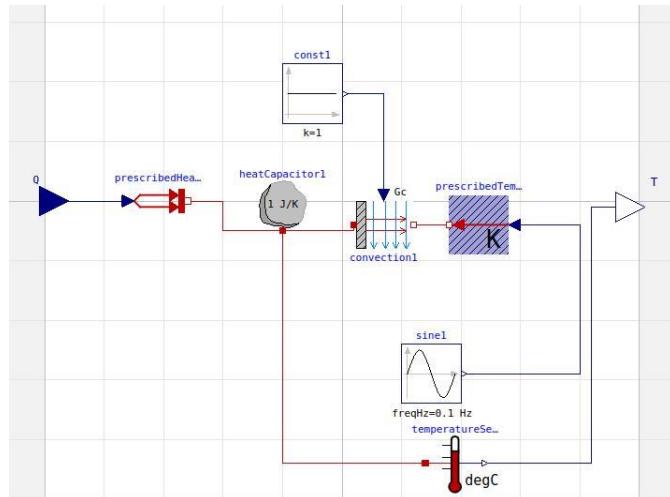
该单元相当于一个虚拟加热系统的进程，接口是输出温度 T，输入热流 Q，创建该单元



创建 heater 单元，它是 tcp 类型，拥有两个点位，因为要与 modelica 客户端交换信息，因此 loc\_map 要赋与合适的名字。

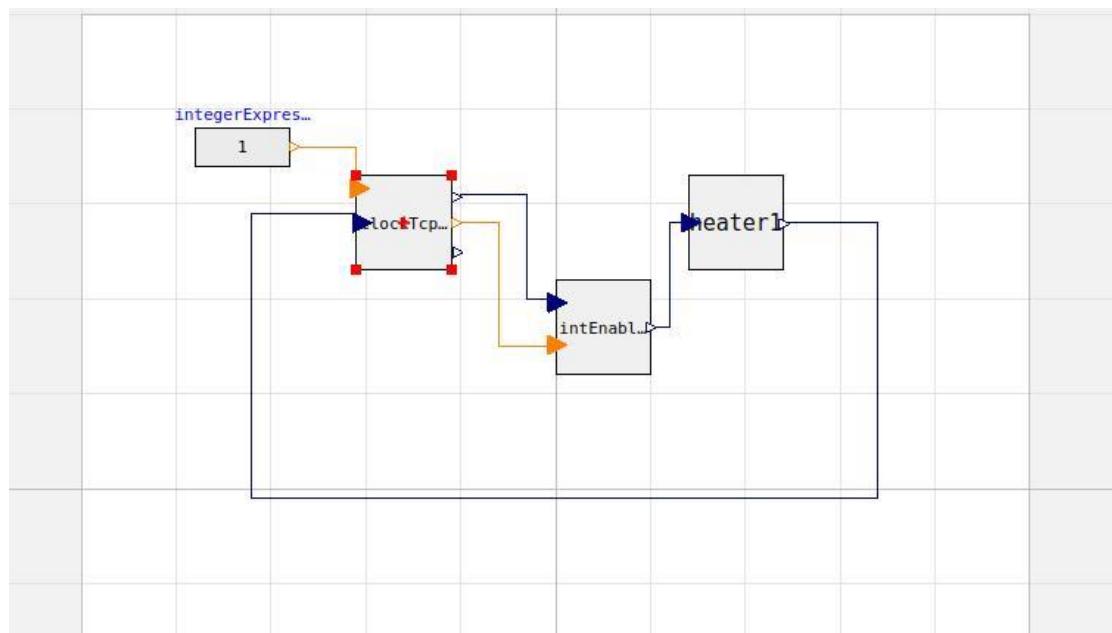
#### ● 创建加热系统单元的客户端

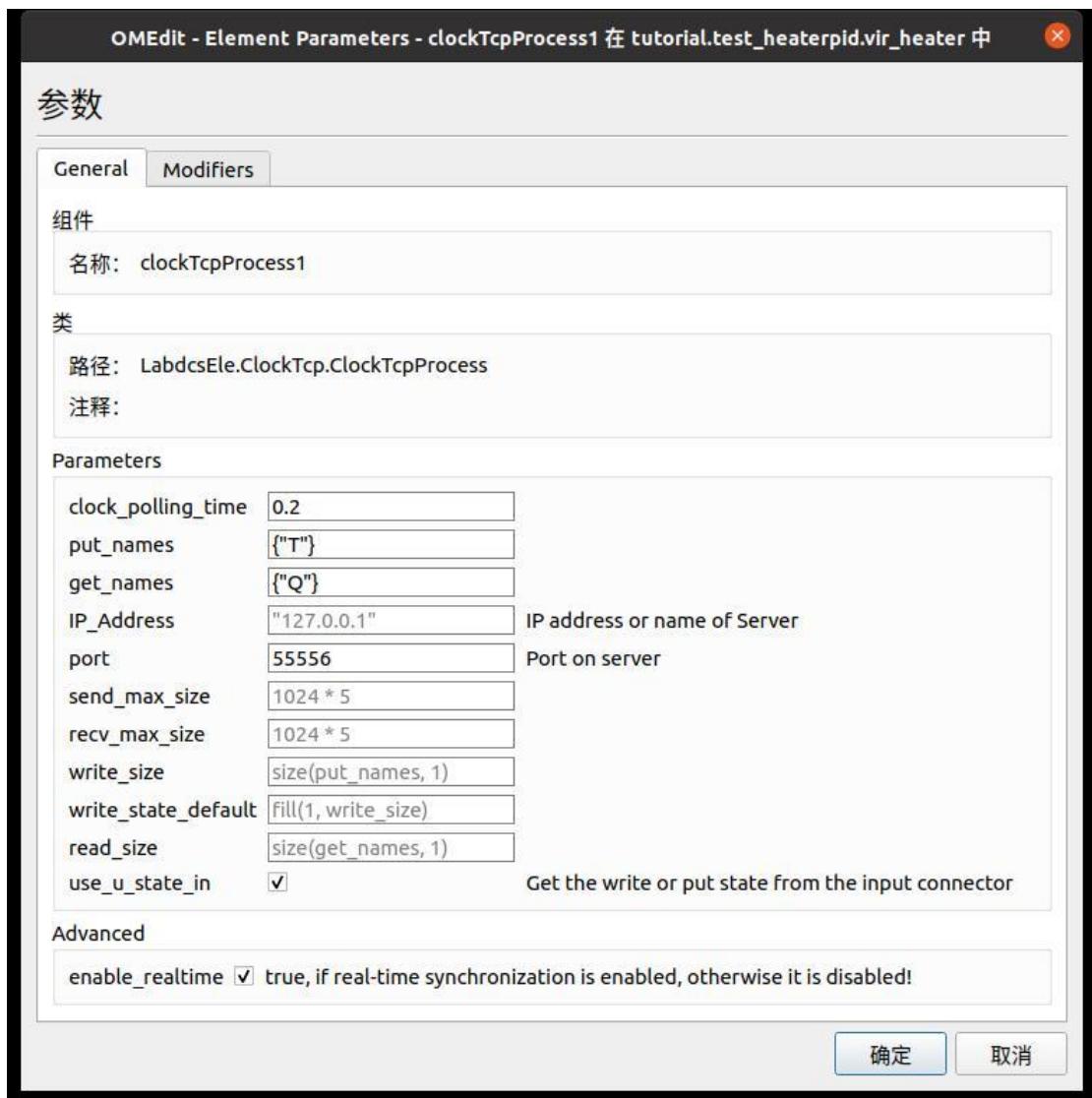
用 modelica 虚拟加热系统如下图：



其控制对象为热容的温度，它受外界温度正弦波的影响，外界温度正弦波，变化范围从-1~1 摄氏度。系统的输入为  $Q$ ，输出为  $T$ 。注意物理建模用的是开尔文的绝对温度，模型的传感器输出使用了摄氏温度。

该模型要与单元对接，则复合的 modelica 模型为





其中的 heater 组件就是虚拟加热系统，IntEnablePass 组件的功能是对输入 Q 进行过滤，若 Q 的数据质量为 False，则会保持之前的 Q，其初始值为 0。

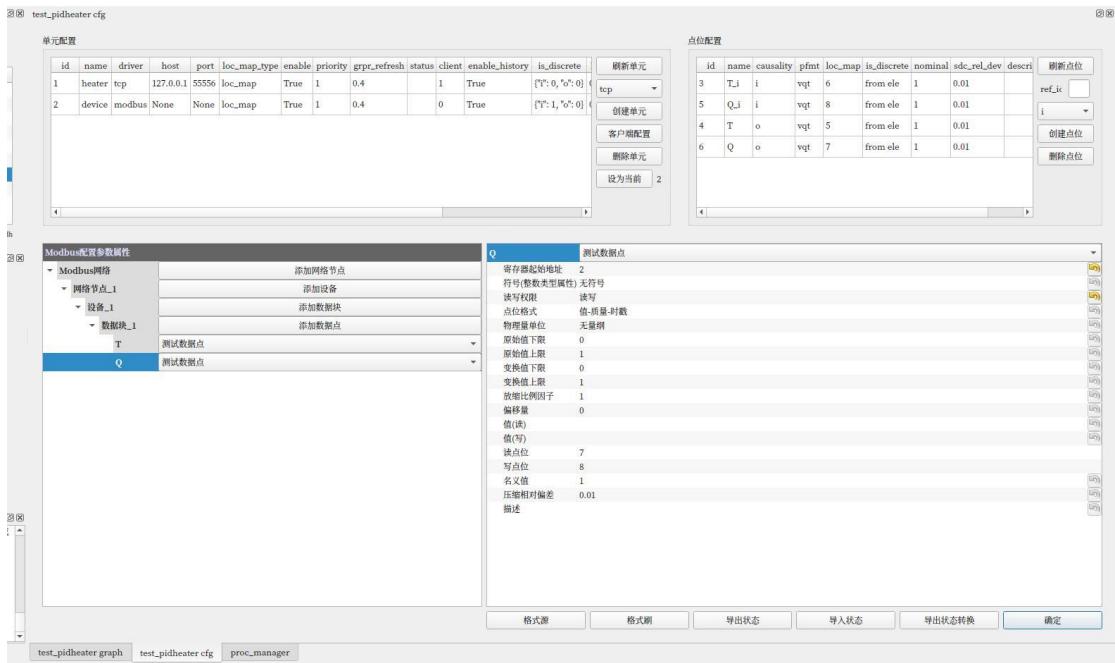
点击客户端配置，创建加热系统单元的客户端。libs 设为 fil://libs.modelica/tutorial.mo，model 设为 tutorial.test\_heaterpid.vir\_heater。如图



## ● 创建 device 设备单元

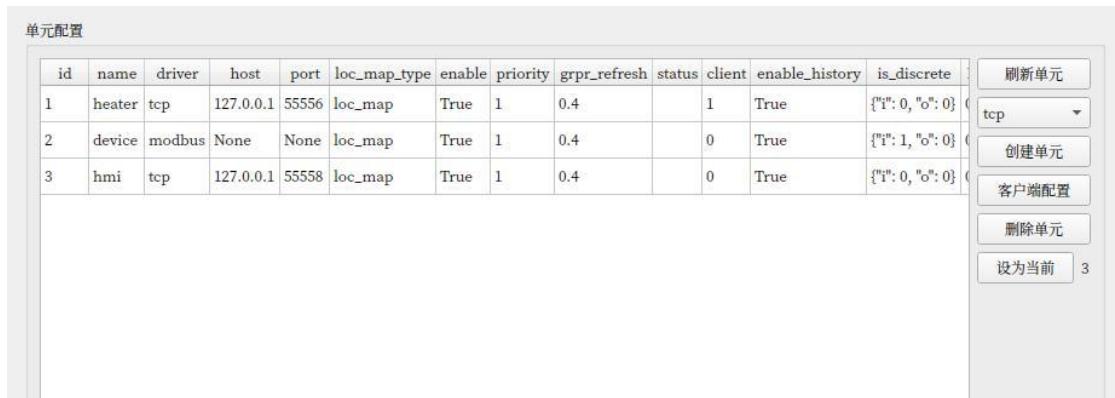
该单元为 modbus 类型，它是 labdcs 提供的调试 modbus 设备。虚拟加热系统的数据要通过这个 modbus 设备与算法单元进行相互作用。对于算法单元，它起着从虚拟加热器读取温度，输出热流量的作用。其实这里没有它也可以将算法与虚拟加热器的数据直接连接，但是这不符合实际的流程，实际中总是会存在这样的采集和控制输出设备的。

该单元的 modbus 设置有两个数据 T、Q，都是可以读写的，且都是 32 为浮点数，因此单元拥有 4 个点位。注意，由于 32 为浮点数会占有两个寄存器，因此第二个数据的地址是 2。



## ● 创建 hmi 设备单元

人机界面的输出点位就是设定点的温度，它还会有多个参考点，这个可以等到算法单元完成后，来弄。



## ● 创建 alg 算法设备单元

算法单元就是读温度，比较参考温度，输出热流。因此它由两个输入 T, Ts，有一个输出 Q。

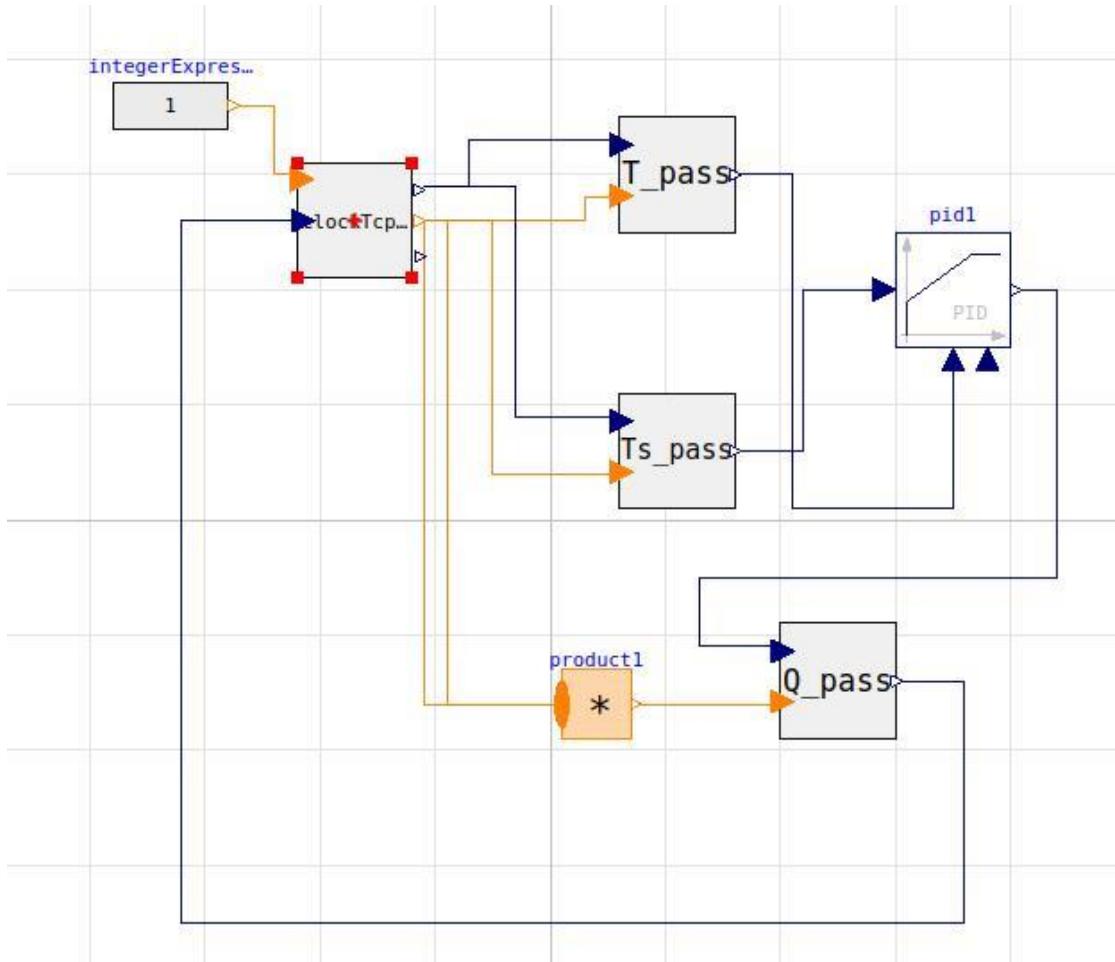
test_pidheater.cfg										
点位配置										
刷新单元										
1	heater	tcp	127.0.0.1	55556	loc_map	True	1	0.4	1	True
2	device	modbus	None	None	loc_map	True	1	0.4	0	True
3	hmi	tcp	127.0.0.1	55558	loc_map	True	1	0.4	0	True
4	alg	tcp	127.0.0.1	55559	loc_map	True	1	0.4	0	True

客户端配置										
删除单元										
8	T	i	vqt	T	from ele	1	0.01	ref_ic	o	创建点位
9	Ts	i	vqt	Ts	from ele	1	0.01	o	o	创建点位
10	Q	o	vqt	Q	from ele	1	0.01	删除点位	删除点位	

### ● 创建 alg 算法设备单元的客户端

算法的客户端读取 T, Ts, 使用 Modelica 提供的 PID 算法，输出热流 Q



该系统的输入要经过 IntTablePass，即值正确时同通过，错误时保持，并且若发生错误，输出则保持前一次的值，该处理方法可以最大限度的抑制可能的传感器错误对系统的冲击。  
当然也可以进一步添加更复杂的处理方法，比如使用状态机。

#### 始件. MODELICA.BLOCKS.CONTINUOUS.LIMITPID

注释： P, PI, PD, and PID controller with limited output, anti-windup compensation, setup

#### Parameters

controllerType .Modelica.Blocks.Types.SimpleController.PID

k 1

Ti 1

Td 0

yMax 30

yMin 0

wp 1

wd 0

Ni 0.9

Nd 10

withFeedForward

kFF 1

#### Initialization

initType .Modelica.Blocks.Types.InitPID.DoNotUse\_InitialIntegratorState

xi\_start 0

xd\_start 0

y\_start 0

homotopyType Modelica.Blocks.Types.LimiterHomotopy.Linear

这时 PID 内部的设定，目前参数的选择都是任意的，需要看响应。

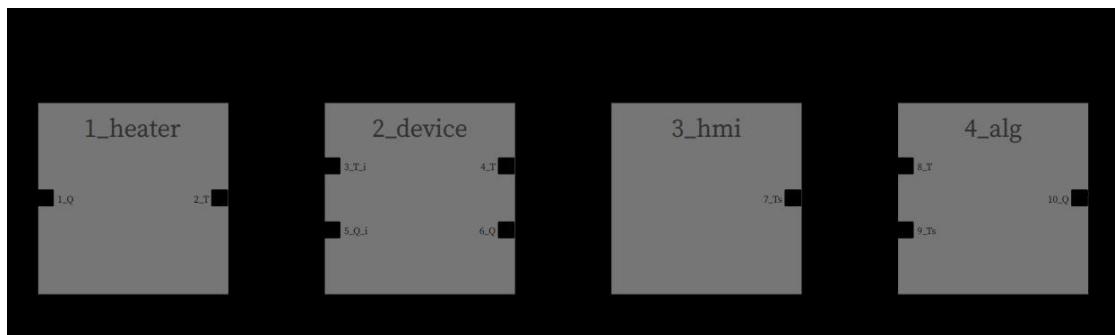
在 Labdcs 平台上创建算法单元的客户端，



把 model 设为 tutorial.test\_heaterpid.vir\_alg

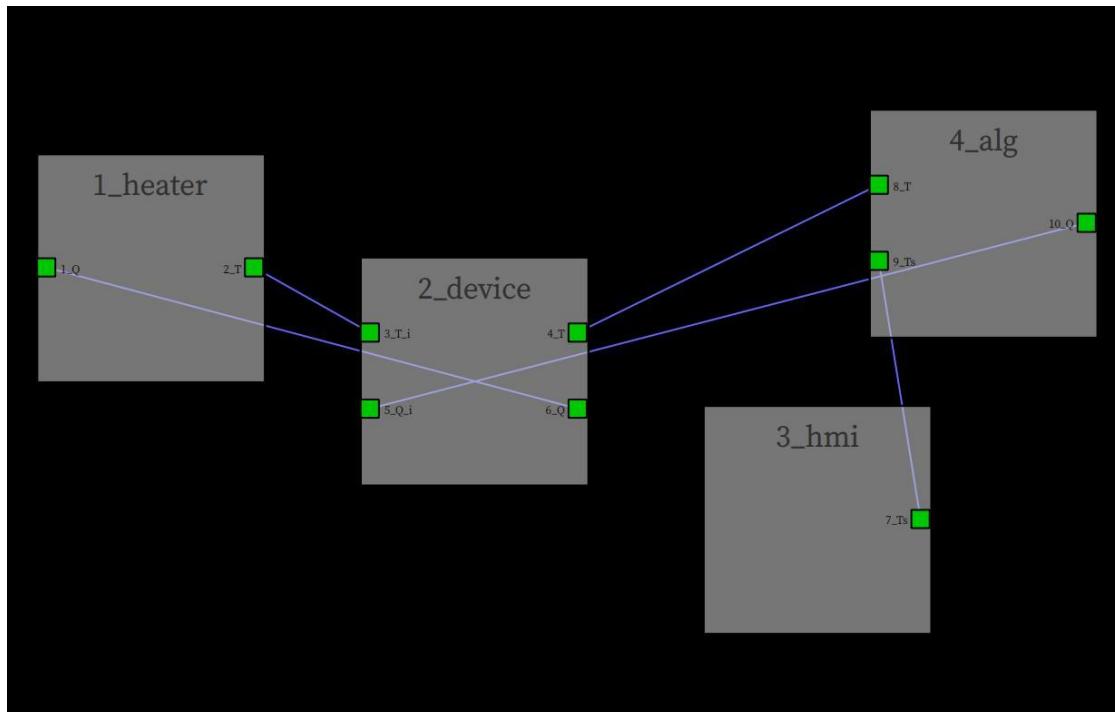
#### 4.2.2 案例单元的连接

单元创建完成后，单元的接口如下



连接之前，还要回到 hmi 单元，创建对点位的参考，这里参考的应该时 device 单元的 T 和 Q，同时我们为了比较，也参考了 heater 单元的 T，实际工程中这个 T 是不能获得的，它要通过 device 的测量获得。

连接后：



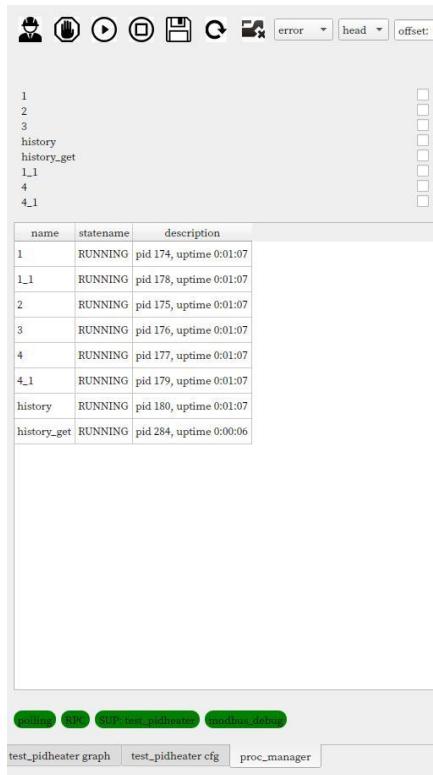
人机界面单元 hmi 输出 Ts 到算法 alg 单元，算法单元还接受 T 的输入，算法单元输出 Q 给 modbus 的 device 单元。device 单元的输出 Q 给 heater 单元的 Q。

，完成配置。

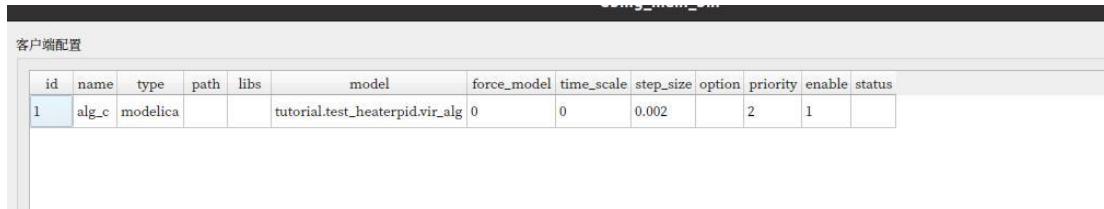
#### 4.2.3 案例的运行和人机界面

- 运行案例



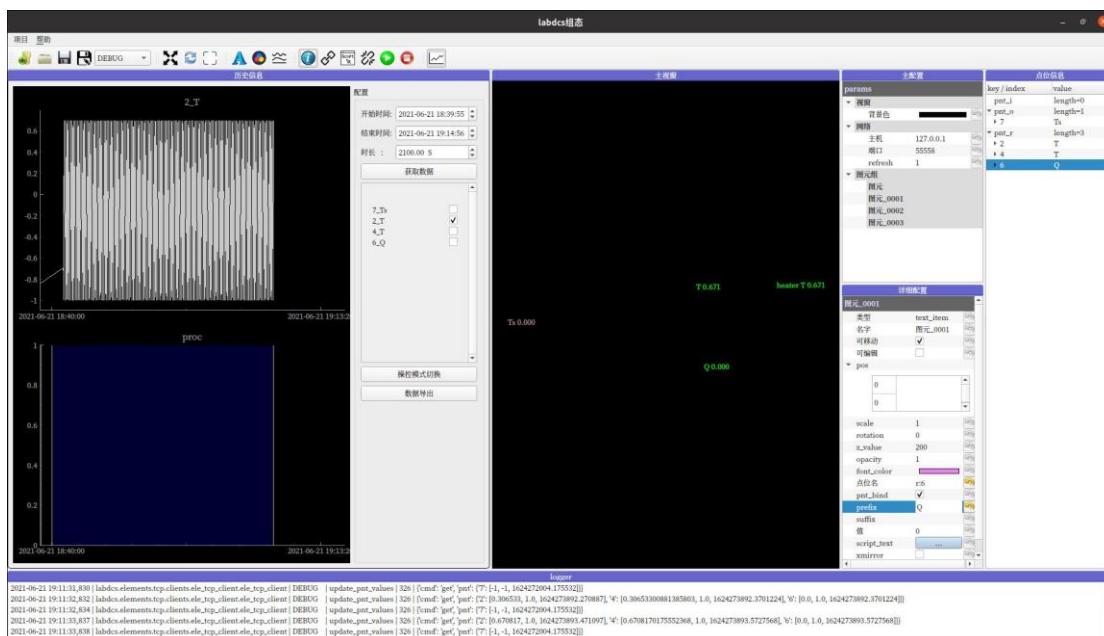


在一次运行后，目录中会保存 modelica 编译后的可执行代码，因此可以



将 force\_model 设为 0，这样系统会在已经存在可行代码的条件下，跳过重新编译。

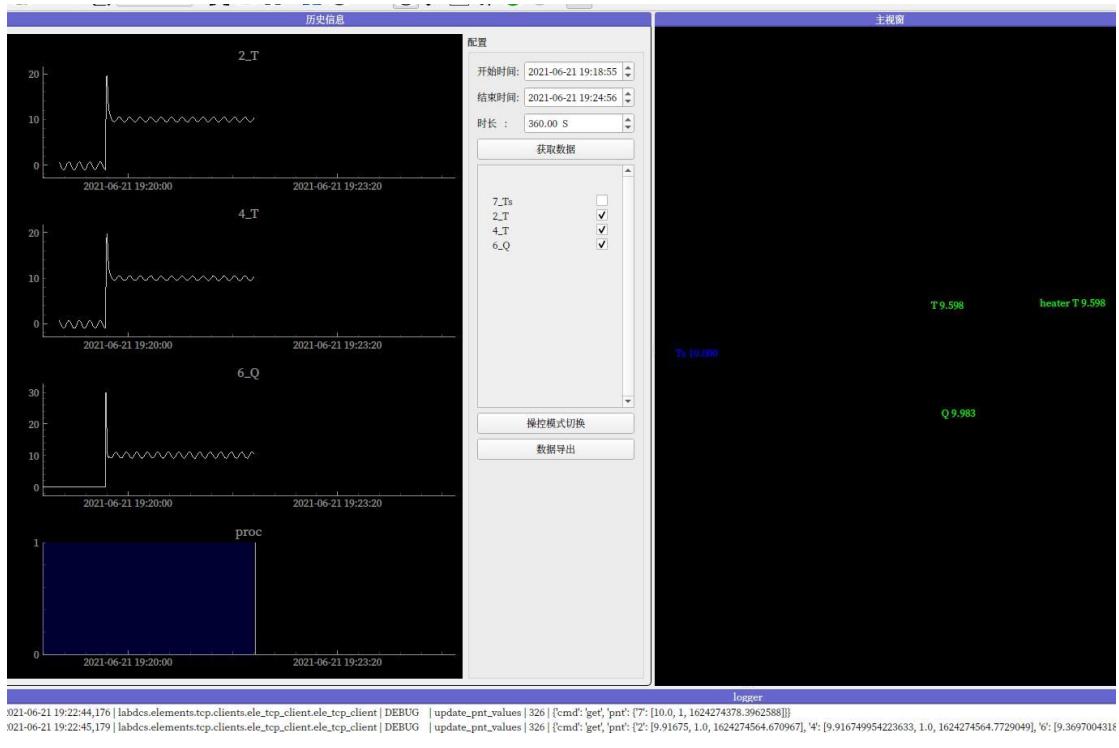
## ● 人机界面



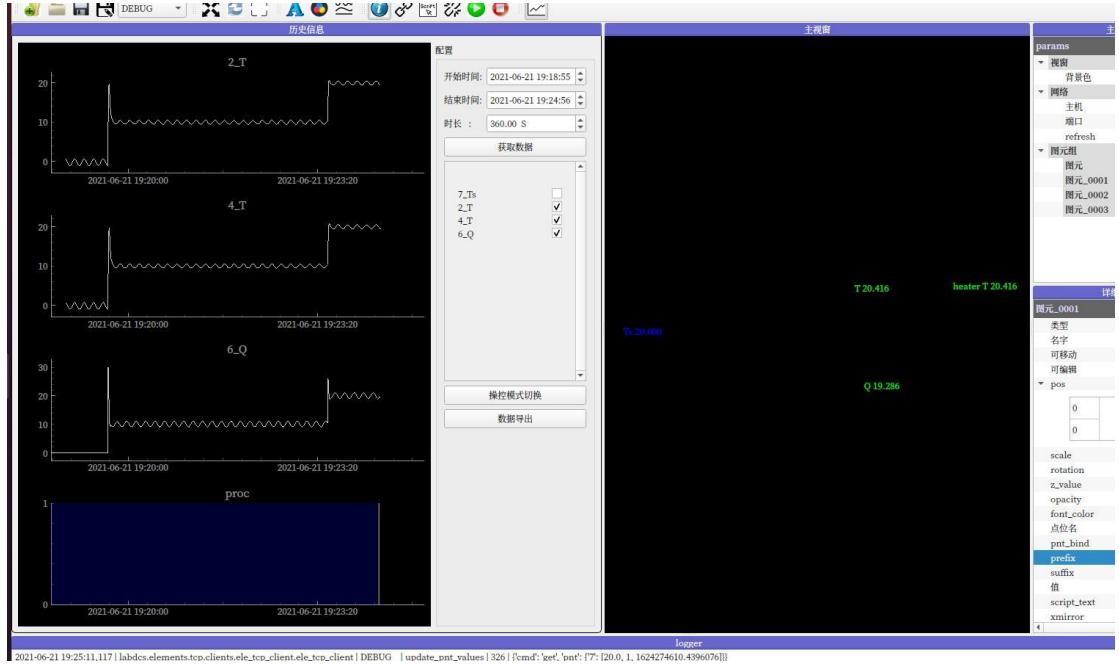
进入界面，进行配置输出和参考的读取。运行后，发现数据已经在变动，这是因为采集和控

制即使在没有控制输出也在进行着，并且由于模型中环境温度是正弦变化，因此控制对象的温度也随这个外界温度正弦变化。

在人机界面上，将  $T_s$  设定为 10，则可以获得



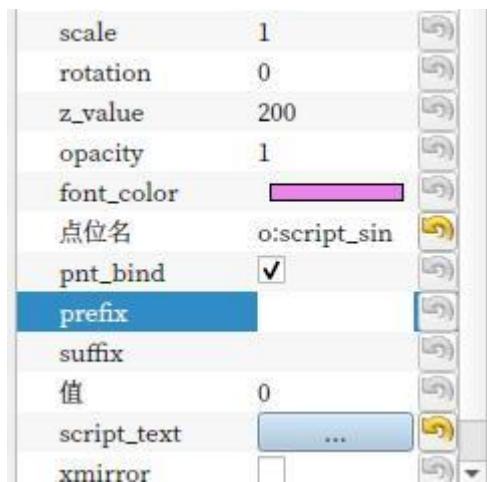
若在将  $T_s$  设为 20，则如图



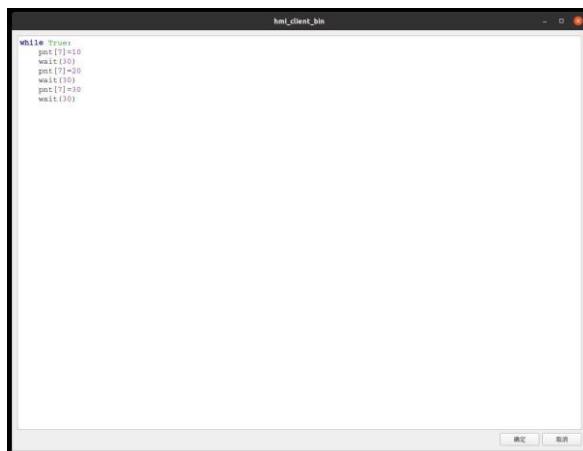
在人机界面程序也支持脚本的运行，比如设定点  $T_s$  从 10 开始每隔 30 秒增加 10，达到 30 后，再循环。

先停止 ，再点击 ，创建图元，选中这个图元，然后点击 ，进行脚本的捆

绑，在弹出的对话框中给出一个脚本名，则在该图元的属性中会显示



若图元捆绑的是脚本，则点位名会以 o:script\_ 开头，后面的 sin 是脚本名  
点击属性的 script\_text 的按钮，可进行脚本的编辑

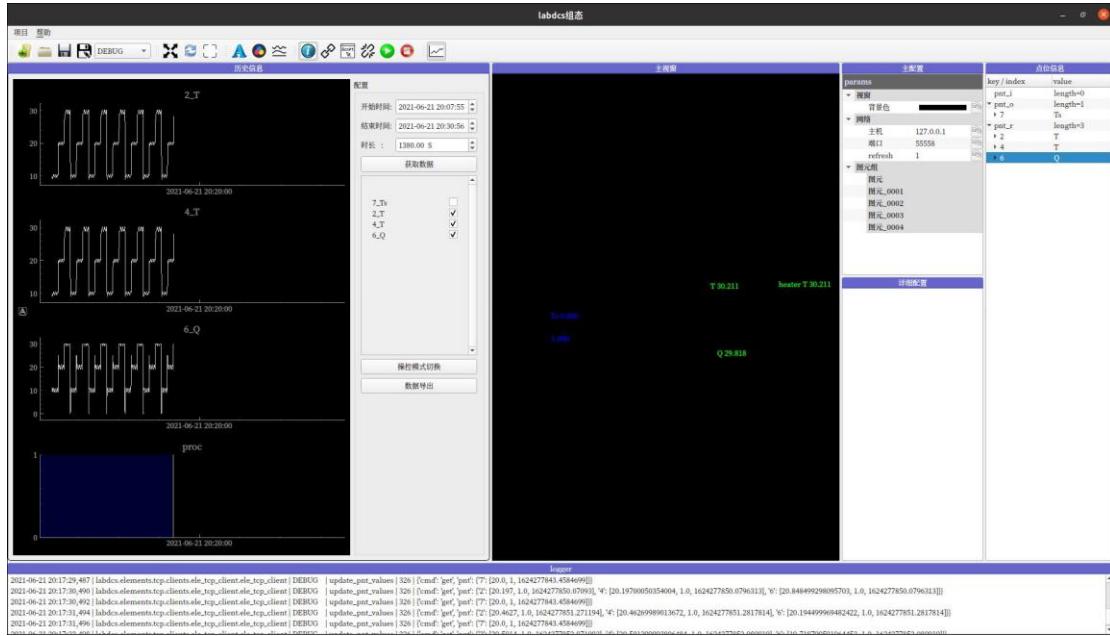


在这个对话框中编辑脚本，注意脚本语言是 python，因此要符合 python 的语法和格式，特别是缩进的大小。

点击确定即可完成编辑。



点击 ，则客户端进入运行模式，右击该脚本图元，输入 1，则运行脚本，则如图



从图中可以看到，其正如我们希望的温度变化曲线。

### 4.3 案例 test\_stategraph

#### 4.3.1 案例设计和接口

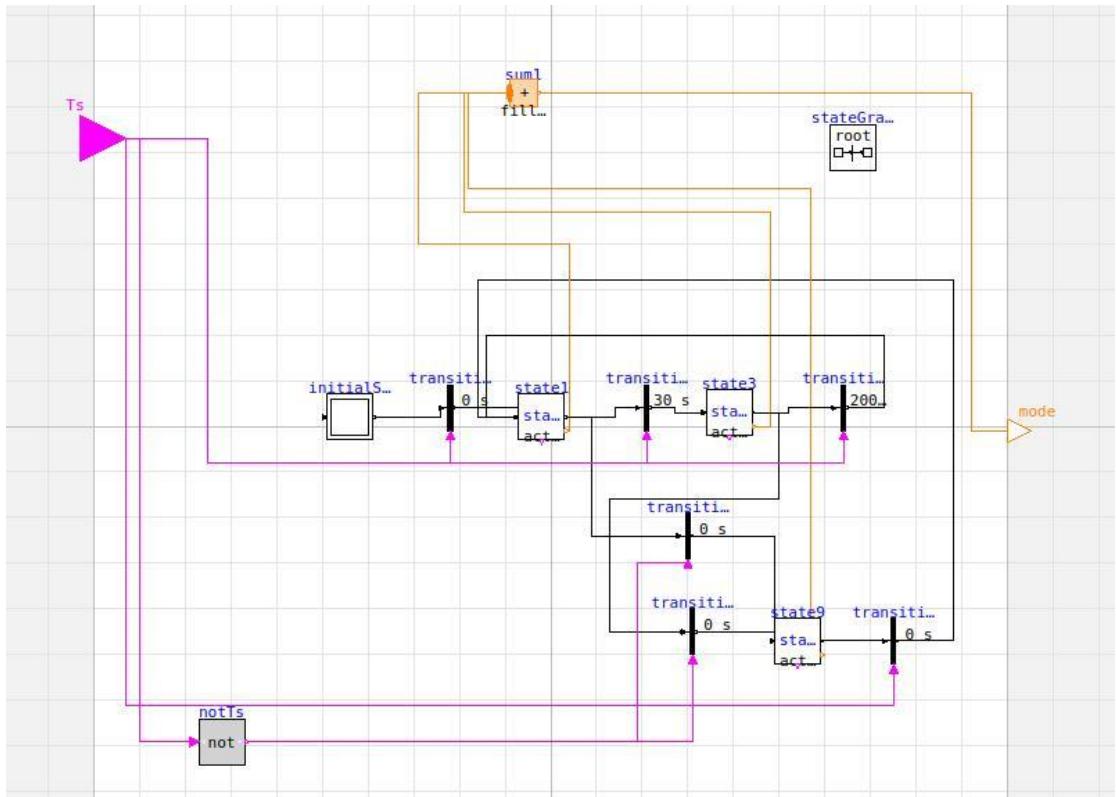
本案例的接口与上一个案例相同，不同的是算法和 Ts 的含义，此案例中 Ts 是开关变量。此案例的算法与上个案例的区别主要是设定点本身有一套变化规则，当 Ts 为 True 后，进入设定温度模式，增加设定温度 10 度，这个设定模式的时长设为 30 秒，然后进入稳态等待模式，这个等待模式的时长为 200 秒，若 Ts 还为 True，则进入循环，继续设定模式和等待模式，由此设定温度从 10 度开始一直增加到 100 度。此案例中 Ts 为 0，则停止上述算法，若为 1，则开始此算法。

该算法我们完全放到 modelica 的算法客户端来完成，此案例与上个案例的唯一区别就是设定点本身有一套变化规则，我们用状态机来实现。

模式为三种，设定、等待、停止，我们用枚举量来定义

```
|type OperationModes = enumeration(off, on_set, on_wait) "Enumeration for modes of operation";
```

状态机的定义如下



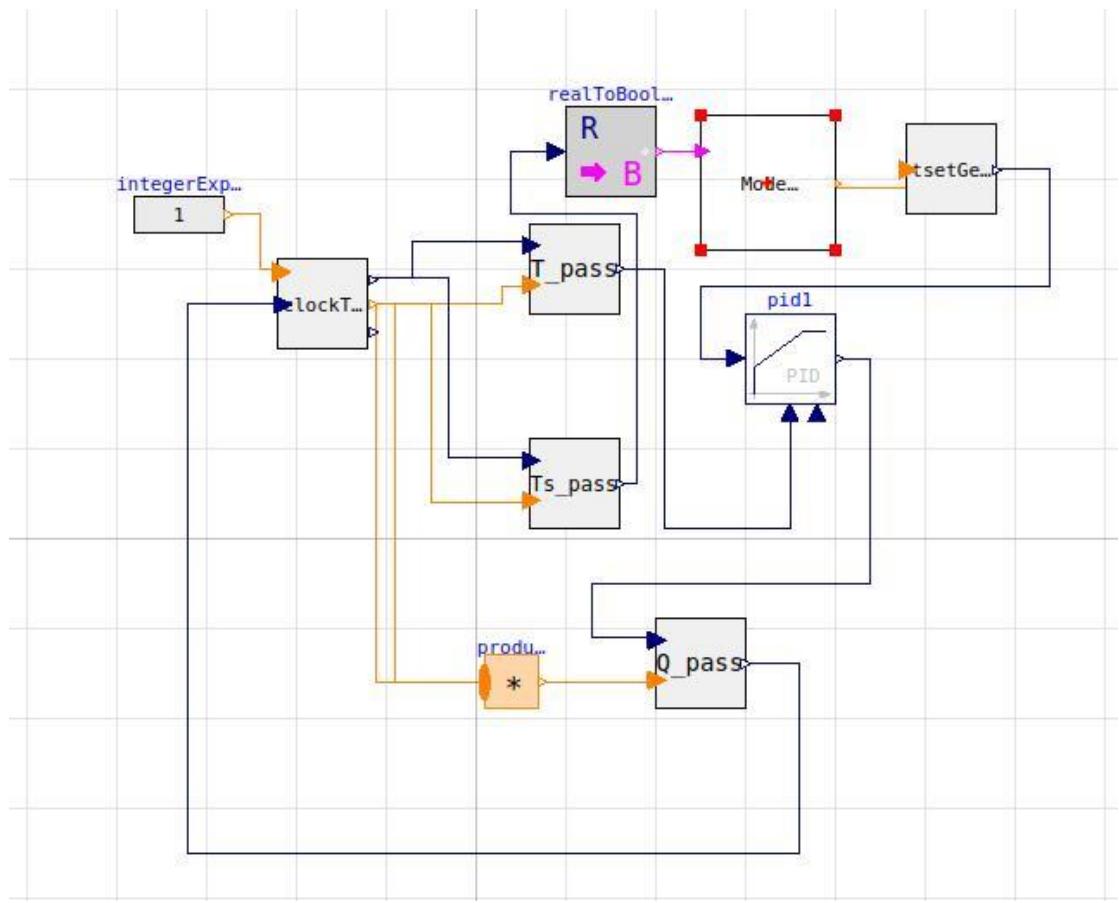
最左边是初始状态，当  $Ts$  为 1，则进入设定模式，设定模式的时长 30 秒，然后是等待模式，等待模式的时长是 200 秒，对于上述任何模式，若任何时刻  $Ts$  为 0，则会进入停止模式。该组件时状态图，它输出当前的模式，模式转化为设定值则由

```

block TsetGen
  "when on_set mode becomes True, change Tset
  when off mode becomes True , turn Tset to initial
  "
  parameter Real Tset_init = 10;
  parameter Real Tset_step = 10;
  parameter Real Tset_max = 100;
  > Modelica.Blocks.Interfaces.IntegerInput mode annotation( ... );
  > Modelica.Blocks.Interfaces.RealOutput Tset(start = Tset_init,fixed=true) annotation( ... );
  equation
    when {mode == Integer(OperationModes.on_set)} then
      if Tset < Tset_max then
        Tset = pre(Tset)+Tset_step;
      else
        Tset = Tset_init;
      end if;
    elsewhen {mode == Integer(OperationModes.off)} then
      Tset = Tset_init;
    end when;

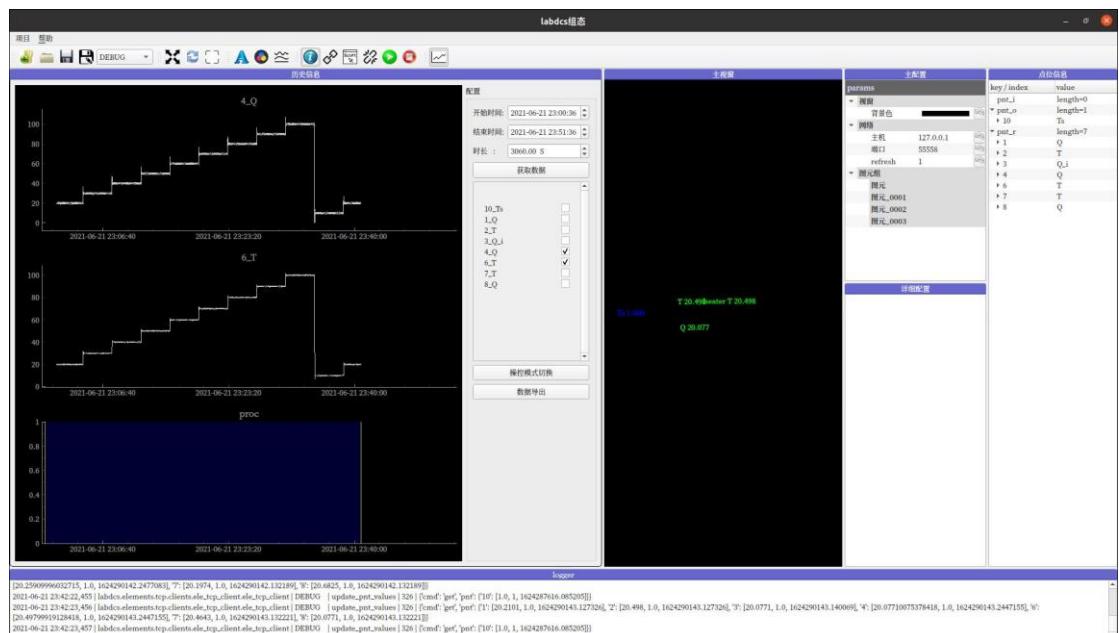
  end TsetGen;
  
```

完成，此组件根据模式的改变，来重新设定  $Ts$ ，模式时离散变化，则  $Ts$  也是离散变化。最终的算法如下图



状态图输出模式，模式由组件转化为设定点，输入的 Ts 是开关量，决定了循环继续，还是停止

#### 4.3.2 案例运行及人机界面



从温度变化的趋势看，正如我们算法所设定的那样温度逐步增加，当达到 100，则从新循环。

## 4.4 实际案例展示

