OpenPLC User Guide

For VisionFive 2 and VisionFive 2 Lite

Version: 1.2

Date: 2025/06/23

Doc ID: VisionFive2-ANEN-020

Legal Statements

Important legal notice before reading this documentation.

PROPRIETARY NOTICE

Contact Us

Address: Room 506, Building 2, No. 61 Shengxia Rd., China (Shanghai) Pilot Free Trade Zone, Shanghai, 201203, China

Website: http://www.starfivetech.com

Email:

- Sales: sales@starfivetech.com
- Support: support@starfivetech.com

Contents

List of Tables....5

List of Figures....6

Legal Statements....2

Preface....8

1. Introduction....9

2. Preparation....12

2.1. Environment Requirements....12

2.2. Required Hardware....12

2.3. Burn Image....12

3. Install OpenPLC....13

3.1. Preparing Software....13

3.2. Installation....13

4. OpenPLC Editor....15

4.1. Install OpenPLC Editor....15

4.2. Programming and Generating Program....15

4.2.1. Single Program....15

4.2.2. Multiple Programs....16

4.3. PLC Simulation....18

5. OpenPLC HMI....20

5.1. Run OpenPLC....22

5.2. Upload New Program....22

5.3. Connect slave device....24

5.3.1. Modbus-TCP....25

5.3.2. Modbus-RTU....28

5.4. Monitoring....31

5.5. Hardware Layer....32

5.6. Users and Settings....33

6. Basic Functions....34

6.1. Basic digital I/O & Analog Output Test....34

6.2. Bistable Test....36

6.3. Timer Test....37

6.4. Counter Test....38

6.5. Arithmetic Test....39

7. SCADA....41

7.1. FUXA....41

7.1.1. Prerequisite....41

7.1.2. Install FUXA....41

7.1.3. FUXA Operations....41

7.2. ScadaBR....50

7.2.1. Prerequisite....50

7.2.2. Install ScadaBR....50

7.2.3. ScadaBR Operation....51

8. Test and improve method of Real-time performance for OpenPLC....58

8.1. Test Method for Real-Time Performance........ 58
8.1.1. Cyclictest........ 58
8.1.2. Test Method of Real-Time Performance for OpenPLC Runtime........ 59
8.2. Improving Real-Time Performance of OpenPLC Runtime........ 61
8.2.1. Isolate a CPU core and bind the OpenPLC process........ 61
8.2.2. Run OpenPLC on RT-Linux:........ 64
9. Examples........ 65
9.1. Temperature Control........ 65

List of Tables

Table 0-1 Revision History....8
Table 1-1 Overview of OpenPLC.... 9
Table 1-2 Pins Distribution 9
Table 2-1 Required Hardware.... 12
Table 5-1 Hardware Preparation....25
Table 5-2 Hardware Preparation....28
Table 6-1 Hardware Preparation....34
Table 9-1 Hardware Preparation....65

List of Figures

Figure 1-1 Layer File....11
Figure 3-1 Python Version.... 13
Figure 3-2 Successful Installation....13
Figure 3-3 Check Python Version....14
Figure 3-4 whl File List....14
Figure 4-1 Create New Program.... 15
Figure 4-2 Create a new POU.... 15
Figure 4-3 Select Required Elements....15
Figure 4-4 Basic I/O Program....16
Figure 4-5 Example.... 16
Figure 4-6 Save the Program.... 16
Figure 4-7 Part_trig Program.... 16
Figure 4-8 trig Settings....17
Figure 4-9 Create a New POU.... 17
Figure 4-11 Declare an Instance.... 17
Figure 4-12 Create and Bound a New Task to an Instance.... 18
Figure 4-13 PLC Simulation.... 18
Figure 4-14 Config0.Res0.instance0.... 19
Figure 4-15 Debugger Bar....19
Figure 5-1 OpenPLC HMI.... 20
Figure 5-2 HMI Main Page....21
Figure 5-3 Dashboard....22
Figure 5-4 Uploading a New Program.... 23
Figure 5-5 Example Program Information.... 23
Figure 5-6 Compiling Program.... 24
Figure 5-7 Go to Dashboard.... 24
Figure 5-8 Test Program....25
Figure 5-9 Slave Device Settings.... 26
Figure 5-10 Connect to 40-Pin GPIO Header....27
Figure 5-11 Run Blank Program....27
Figure 5-12 Successful Connection.... 28
Figure 5-13 Check USB2Serial Connection.... 29
Figure 5-14 Arduino UNO.... 29
Figure 5-15 Photo of Hardware Setup....30
Figure 5-16 Hardware Setup....30
Figure 5-17 Configure Slave Device.... 31
Figure 5-18 Running io_test....32
Figure 5-19 Hardware Layer....33
Figure 6-1 Ladder Diagram.... 35
Figure 6-2 Address of Each Input/Output Point....36
Figure 6-3 View I/O Point Status 36
Figure 6-4 Bistable Test....37
Figure 6-5 Timer Test.... 38

Figure 6-6 Counter Test......39
Figure 6-7 Arithmetic Test......40
Figure 7-1 Enter Homepage of FUXA...... 42
Figure 7-2 Click Editor......42
Figure 7-3 Verify Modus Installation...... 43
Figure 7-5 Connections...... 44
Figure 7-7 Device Property...... 45
Figure 7-8 Click OK......45
Figure 7-9 Click Edit Device Tags...... 46
Figure 7-10 Example Circular Control......47
Figure 7-11 Example...... 48
Figure 7-12 Launch Current View......49
Figure 7-13 Save Project......49
Figure 7-14 SCADA...... 50
Figure 7-15 Data source......51
Figure 7-16 Select Modbus IP as Data Source......51
Figure 7-17 Modebus IP Settings......52
Figure 7-18 Point Settings......53
Figure 7-19 Data Point Values...... 54
Figure 7-21 Crate New Binary Graphic......54
Figure 7-22 Binary Graphic Settings...... 55
Figure 7-23 Point Settings......56
Figure 7-25 Edit Graphics renderer...... 56
Figure 7-27 Display Real-Time Status...... 57
Figure 8-1 Delay between Input and Output Signals...... 59
Figure 8-3 Example Scan Time Test 60
Figure 8-4 Minimum Scan Time......60
Figure 8-5 Example Output......61
Figure 8-6 Example Output......62
Figure 8-7 Example Output......62
Figure 8-8 Example Output......62
Figure 8-9 Example Setting...... 63
Figure 9-1 Transferred to FUXA via Modbus TCP...... 65
Figure 9-2 PLC Functional Block Program......67
Figure 9-3 Variable Definition......67
Figure 9-4 Temperature Control System......68

Preface

About this guide and technical support information.

About this document

This application note provides the steps for the application of OpenPLC based on VisionFive 2 and VisionFive 2 Lite. The OpenPLC application used is adapted to the platform of VisionFive 2 and VisionFive 2 Lite

Revision History

Table 0-1 Revision History

Version	Released	Revision
1.2	2025/06/23	• Updated Environment Requirements *(on page 12)*. • Updated Preparing Software *(on page 13)*. • Updated Installation *(on page 13)*. • Updated the file path in Install OpenPLC *(on page 13)*. • Updated the download path in Install OpenPLC *(on page 13)*.
1.1	2025/06/10	• Updated the official default version of Python as 3.11 and the installation steps in Preparing Software *(on page 13)*. • Updated step 2 in Installation *(on page 13)*.
1.0	2023/10/27	The First Official Release.

Notes and notices

The following notes and notices might appear in this guide:

- **Tip:**
  Suggests how to apply the information in a topic or step.
- **Note:**
  Explains a special case or expands on an important point.
- **Important:**
  Points out critical information concerning a topic or step.
- **CAUTION:**
  Indicates that an action or step can cause loss of data, security problems, or performance issues.
- **Warning:**
  Indicates that an action or step can result in physical harm or cause damage to hardware.

1. Introduction

This application note provides the steps for the application of OpenPLC based on VisionFive 2 and VisionFive 2 Lite. The OpenPLC application used is adapted to the platform of VisionFive 2 and VisionFive 2 Lite

Official repository: https://github.com/thiagoralves/OpenPLC_v3

The overview of OpenPLC can be seen below:

Table 1-1 Overview of OpenPLC

Overview of OpenPLC		
Functions	**Items**	**Usage**
Input/Output	Digital input(8)	Gets digital input
	Digital output(8)	Produces digital output
	Analog output(1)	Produces analog(pwm) output
Communication protocol	Modbus RTU(Serial)	Modbus-RTU master device
	Modbus TCP	Modbus-TCP master/slave device; Connect Scada
	DNP3	Connect Scada
	Ethernet/IP	Connect Scada
HMI(internal)	Dashboard	View the status of OpenPLC
	Program	Upload PLC program
	Slave device	Connect&config slave devices
	Monitoring	Viewpoints value
Scada	FUXA	Support OPCUA, Modbus-TCP, Modbus-RTU, BACnet. Siemens-S7, Ethernet/IP
	ScadaBR	Support DNP3, Modbus-TCP, Modbus-RTU, BACnet...

There are 8 digital inputs, 8 digital outputs, and 1 analog output defined in VisionFive 2 and VisionFive 2 Lite hardware layer, pins distribution are shown below:

Table 1-2 Pins Distribution

OpenPLC I/O	Pin name	Pin num	Pin num	Pin name	OpenPLC I/O
3v3 Power	3v3 Power	1	2	5V Power	5v Power
N/A	GPIO 58 (I2C SDA)	3	4	5V Power	5v Power
N/A	GPIO 57 (I2C SCL)	5	6	Ground	Ground
%IX0.0	GPIO 55 (GPCLK0)	7	8	GPIO 5 (UART TX)	N/A
Ground	Ground	9	10	GPIO 6 (UART RX)	N/A
%IX0.1	GPIO 42	11	12	GPIO 38 (PCM CLK)	**%QW0**
%IX0.2	GPIO 43	13	14	Ground	Ground
%IX0.3	GPIO 47	15	16	GPIO 54	**%QX0.0**
3v3 Power	3v3 Power	17	18	GPIO 51	**%QX0.1**

Table 1-2 Pins Distribution (continued)

OpenPLC I/O	Pin name	Pin num	Pin num	Pin name	OpenPLC I/O
N/A	GPIO 52 (SPI MOSI)	19	20	Ground	Ground
N/A	GPIO 53 (SPI MISO)	21	22	GPIO 50	**%QX0.2**
N/A	GPIO 48 (SPI SCLK)	23	24	GPIO 49 (SPI0 CE0)	N/A
Ground	Ground	25	26	GPIO 56	**%QX0.3**
N/A	GPIO 45	27	28	GPIO 40	**%QX0.4**
%IX0.4	GPIO 37	29	30	Ground	Ground
%IX0.5	GPIO 39	31	32	GPIO 46 (PWM0)	N/A
N/A	GPIO 59 (PWM1)	33	34	Ground	Ground
%IX0.6	GPIO 63	35	36	GPIO 36	**%QX0.5**
%IX0.7	GPIO 60	37	38	GPIO 61	**%QX0.6**
Ground	Ground	39	40	GPIO 44	**%QX0.7**

OpenPLC Runtime uses the IEC 61131-3 nomenclature to address input (I), output (Q), and memory (M) locations. Moreover, memory bits (%MX) locations were not implemented, create any variable of type bool with no location (leave location blank) for it to serve as memory bits. If those variables need to be accessed by Modbus, locate those variables in the %QX area outside of device bounds, for example '%QX80.0'. More details can be seen at: https://openplcproject.com/docs/2-3-input-output-and-memory-addressing/

In the table above, **%IX0.x**, **%QX0.x** and **%QWx** represent digital input, digital output, and analog output respectively. All these I/Os are controlled by python lib:'**Visionfive.gpio**'. The pin num that is marked green indicates that this pin can be enabled by the Python library. More details of 'VisionFive.gpio' can be seen in Preparing Software.

The physical pins and number (if VisionFive.gpio supports) of OpenPLC I/O can be defined in the hardware layer file (`openplc_v3/webserver/core/hardware_layers/VisionFive2_py.cpp`):

Figure 1-1 Layer File

```
// Defines the number of pins to be used for digital input/output and analog output.
#define MAX_INPUT 8
#define MAX_OUTPUT 8
#define MAX_PWM_OUT 1

// inputPinMask: pin mask for each input, which.
// Means what pin is mapped to that OpenPLC input.
unsigned int inputPinMask[MAX_INPUT] = {
    7,  // pin7,  GPIO55,   %IX0.0
    11, // pin11, GPIO42,   %IX0.1
    13, // pin13, GPIO43,   %IX0.2
    15, // pin15, GPIO47,   %IX0.3
    29, // pin29, GPIO39,   %IX0.4
    31, // pin31, GPIO59,   %IX0.5
    35, // pin35, GPIO63,   %IX0.6
    37  // pin37, GPIO60,   %IX0.7
};

// outputPinMask: pin mask for each output, which.
// Means what pin is mapped to that OpenPLC output.
unsigned int outputPinMask[MAX_OUTPUT] = {
    16, // pin16, GPIO54,   %QX0.0
    18, // pin18, GPIO51,   %QX0.1
    22, // pin22, GPIO50,   %QX0.2
    26, // pin26, GPIO56,   %QX0.3
    28, // pin28, GPIO40,   %QX0.4
    36, // pin36, GPIO36,   %QX0.5
    38, // pin38, GPIO61,   %QX0.6
    40  // pin40, GPIO44,   %QX0.7
};

// pwmOutputPin: channel for the analog PWM output of the VisionFive2.
unsigned int pwmOutputPinMask[MAX_PWM_OUT] = {
    12  // pin12, GPIO38    %QW0
};
```

2. Preparation

Before installing OpenPLC, some preparations need to be completed.

2.1. Environment Requirements

The environment requirements are as follows:

- Linux Kernel: Linux 6.6.20
- OS: Debian 13
- SBC: VisionFive 2 or VisionFive 2 Lite
- SoC: JH-7110

2.2. Required Hardware

Table 2-1 Required Hardware

Items	M/O*	Notes
VisionFive 2 or VisionFive 2 Lite	M	-
• An Micro-SD card with a capacity of at least 32 GB • Micro-SD card reader • Computer (Windows/Mac OS/ Linux) • Network cable • Power adapter (5 V/ 3 A) • USB serial port converter	M	The above items are used to burn Debian OS to Micro-SD card.
• An HDMI display • An HDMI cable	M	Used to operate Debian systems.
USB Key Board	M	
USB Mouse	M	

2.3. Burn Image

Please reference to https://doc.rvspace.org/VisionFive2/Quick_Start_Guide/VisionFive2_QSG/getting_started.html to download, burn and scale a Debian image.

3. Install OpenPLC

Download link: https://debian.starfivetech.com/.

File path: `VisionFive2/Engineering Release/202409/debian-packs/openplc.zip`.

3.1. Preparing Software

Follow the steps below to prepare software environment:

1. Execute the following command to preparing software:

```
sudo apt-get install -y curl git udev libxml2-dev autoconf automake autotools-dev icu-devtools
libicu-dev libsigsegv2 m4 autoconf-archive gnu-standards libtool gettext m4 make cmake build-essential
 pkg-config bison flex libtool nodejs libbz2-dev sqlite3 libgdbm-dev openssl libexpat1-dev
 python${version}-dev python3 libxslt-dev libmodbus5 python3-venv libasio-dev
```

Note:
The version in `python${version}` indicates the current version of python, which can be found at `python3 -V`; the official default is 3.11.

Figure 3-1 Python Version

```
root@starfive:~# python3 -V
Python 3.11.4
```

3.2. Installation

Perform the following steps to install OpenPLC:

1. Execute the following command to install:

```
sudo dpkg -i openplc-vf2.deb
```

Result:

After the installation is completed, there will be:

Figure 3-2 Successful Installation

```
user@starfive:~$ ls
openplc-vf2.deb
user@starfive:~$ sudo dpkg -i openplc-vf2.deb
Selecting previously unselected package openplc-vf2.
(Reading database ... 107686 files and directories currently installed.)
Preparing to unpack openplc-vf2.deb ...
Unpacking openplc-vf2 (0.1) ...
Setting up openplc-vf2 (0.1) ...
Run ldconfig
[OPENPLC SERVICE]
Enabling OpenPLC Service...
```

2. Create python virtual environment and install python packages:

```
cd /home/user/openplc_v3
sudo python3 -m venv .venv
sudo .venv/bin/python3 -m pip install requests wget bs4 flask==2.3.3 werkzeug==2.3.7 flask-login==0.6.2
 pyserial pymodbus==2.5.3
```

3. After installation of OpenPLC runtime, the python library `VisionFive.gpio` should also be installed by referring to the Preparing Software section.

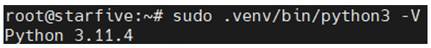
Note:
The OpenPLC runtime is installed under `sudo`, so `VisionFive.gpio` should also be installed by `sudo`. The *Step g* in the above link should be modified as:

```
sudo .venv/bin/python3 -m pip install
 $(path_to_whl)/VisionFive.gpio-1.3.3-cp31x-cp31x-linux_riscv64.whl
```

Where `$(path_to_whl)` is the path pointing to the `VisionFive.gpioxxx.whl` file. The **x** in `VisionFive.gpio-1.3.3-cp31x-cp31x-linux_riscv64.whl` depends on the Python version used by the current venv. You can check it by running the following command:

```
sudo .venv/bin/python3 -V
```

Figure 3-3 Check Python Version



If it is version 3.10, install the cp310 version; if it is version 3.11, install the cp311 version.

Figure 3-4 whl File List

4. After installing `VisionFive.gpio`, a `reboot` is required.

4. OpenPLC Editor

The OpenPLC Editor is an IEC 61131-3 compliant PLC code editor. It allows you to create, compile and upload your IEC 61131-3 programs to the OpenPLC Runtime. OpenPLC Editor implements all the languages described in the IEC-61131-3 standard: Ladder Logic (LD), Function Block Diagram (FBD), Instruction List (IL), Structured Text (ST), and Sequential Function Chart (SFC).

Basic knowledge of PLC programming can be seen at: http://www.plcs.net/

4.1. Install OpenPLC Editor

Download Link: https://openplcproject.com/download/.

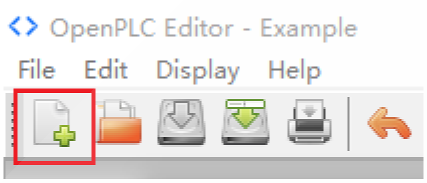
More details of PLC programming language and examples: https://openplcproject.com/docs/3-2-creating-your-first-project-on-openplc-editor/.

4.2. Programming and Generating Program

4.2.1. Single Program

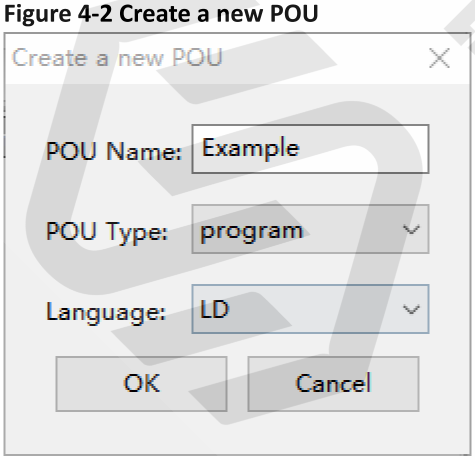
1. Click **NEW** in the upper left corner to create a new project.

   Figure 4-1 Create New Program

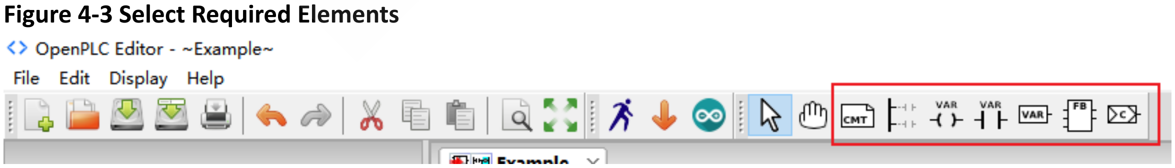
2. Create a new POU (Programming Organisation Unit), fill in the program name, and select the language to use (ladder diagram is selected here):

   Figure 4-2 Create a new POU

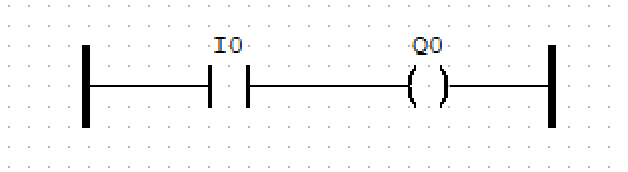
3. Select the required elements to form a ladder diagram program:

   Figure 4-3 Select Required Elements

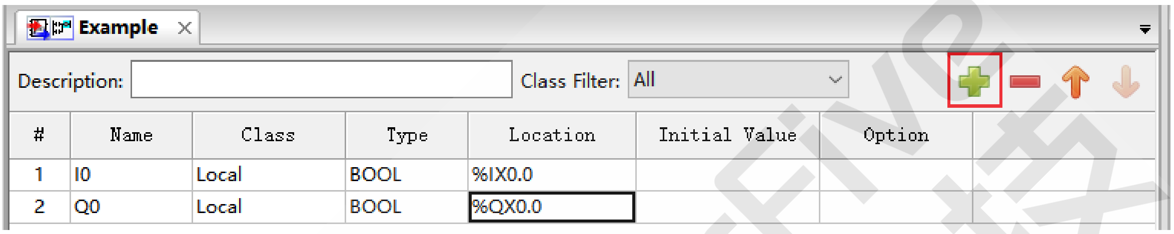
   Here's a basic I/O program:

Figure 4-4 Basic I/O Program



In this program, Q0 will output a high level accordingly when I0 is triggered by a high level. In addition to the basic logical connection, the address of each component needs to be assigned. In this program, I0 and Q0 correspond to the actual digital input and output respectively, and should be assigned usable I/O point addresses, such as:

Figure 4-5 Example

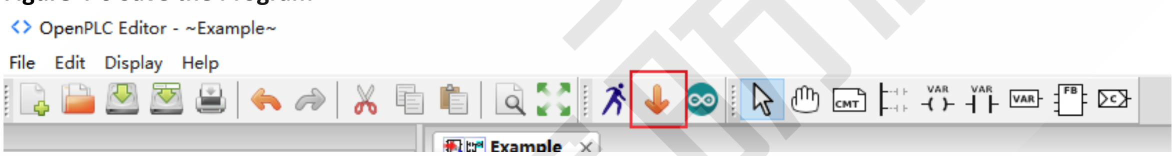


#	Name	Class	Type	Location	Initial Value	Option
1	I0	Local	BOOL	%IX0.0		
2	Q0	Local	BOOL	%QX0.0		

4. Click **Add variable** to add a variable that is used in this program. Generally only need to fill in the corresponding variable name, variable type and assign the appropriate address.

   After programming and assigning the location of variables, click **Generate program for OpenPLC Runtime** and save the program as a `.st` file:

Figure 4-6 Save the Program

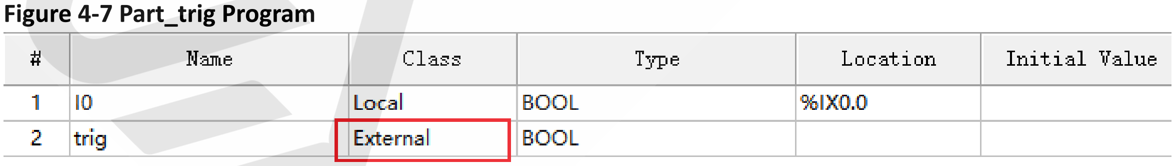


4.2.2. Multiple Programs

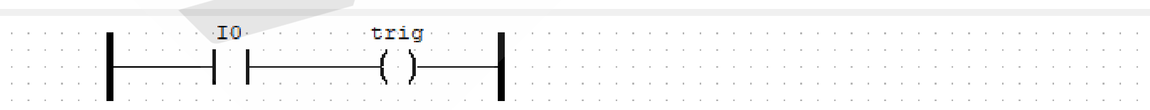
Sometimes, multiple programs are needed to perform different functions at different intervals within a project. Here's a simple demonstration:

1. Similar to the ladder diagram program in section Single Program *(on page 15)*, create a program **Part_trig** to get an input signal and trigger another program:

Figure 4-7 Part_trig Program



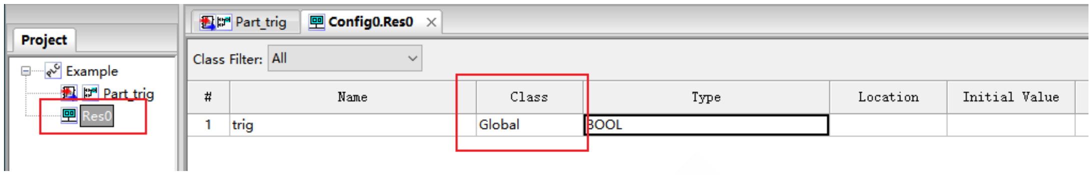
#	Name	Class	Type	Location	Initial Value
1	I0	Local	BOOL	%IX0.0	
2	trig	External	BOOL		



Note:

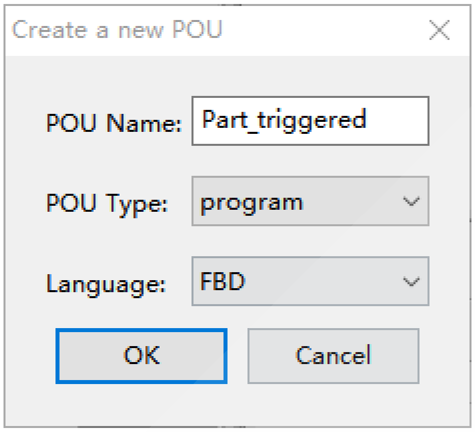
trig is a variable that triggers another program and needs to be defined as **External** in this program and declared as **Global** in **Res0**:

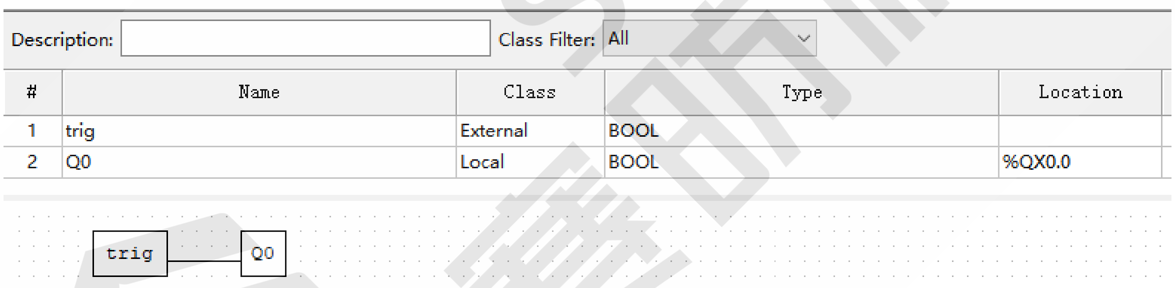
Figure 4-8 trig Settings



2. Then, create another program that is triggered by program ‘Part_trig’. Programs in different languages can be in the same project, so a new POU using FBD(Function block diagram) can be created in this project:

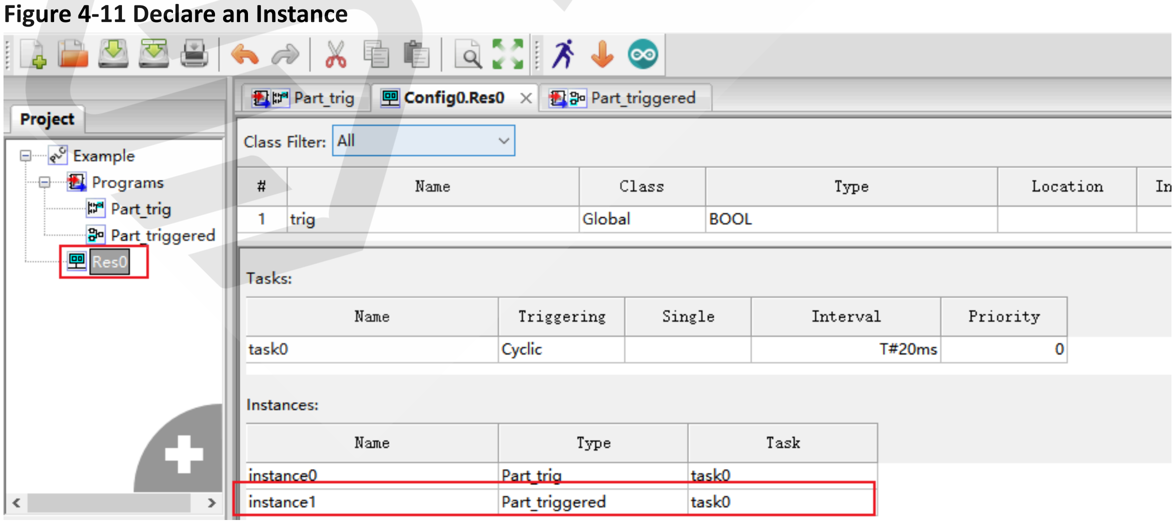
Figure 4-9 Create a New POU





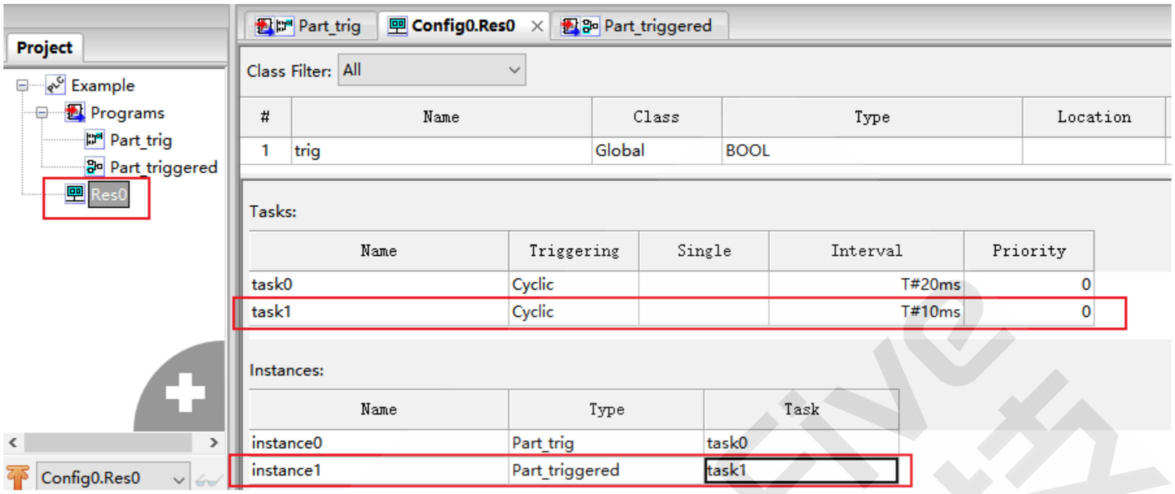
3. To run this program, an instance should be declared for it in the Res.0:

Figure 4-11 Declare an Instance



4. Moreover, if programs need to run at different intervals, a new task should be created and bound to corresponding instance:

Figure 4-12 Create and Bound a New Task to an Instance

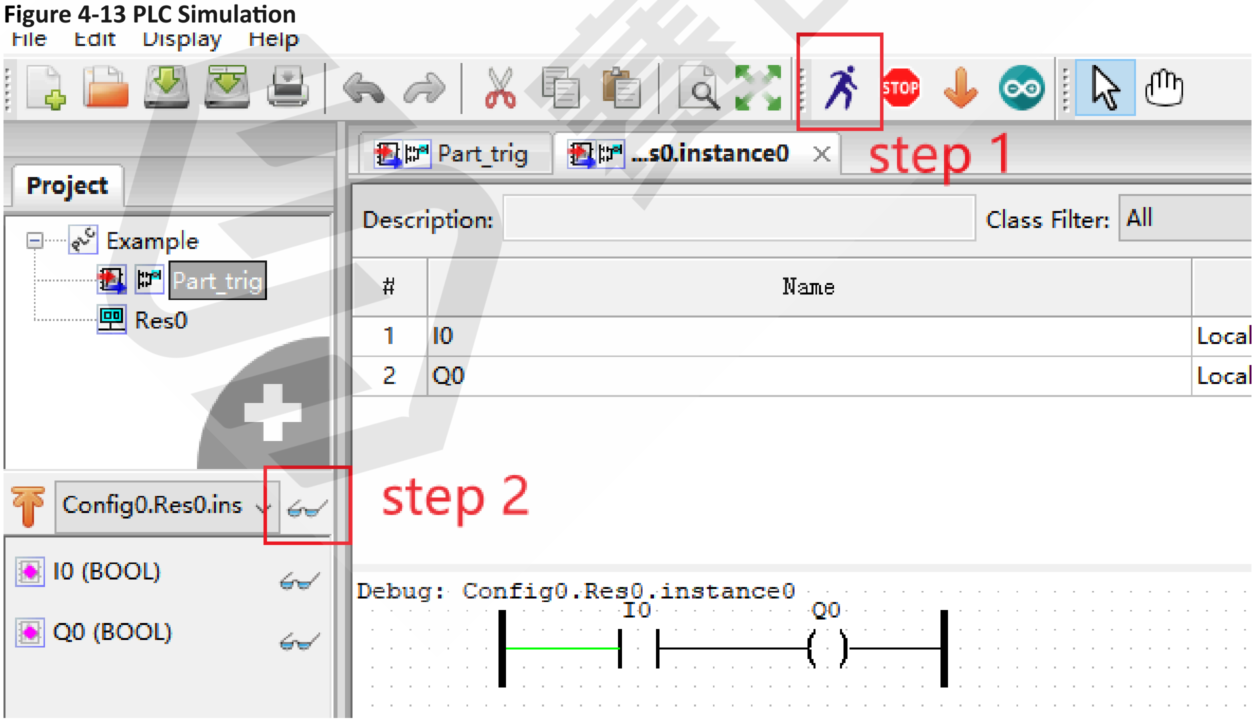


As shown above, program 'part_trig' will run at 20ms intervals while program 'part_triggered' on 10ms. It should be mentioned that the program may not be able to run on this interval. The minimum interval of the program depends on the real-time performance of the system.

4.3. PLC Simulation

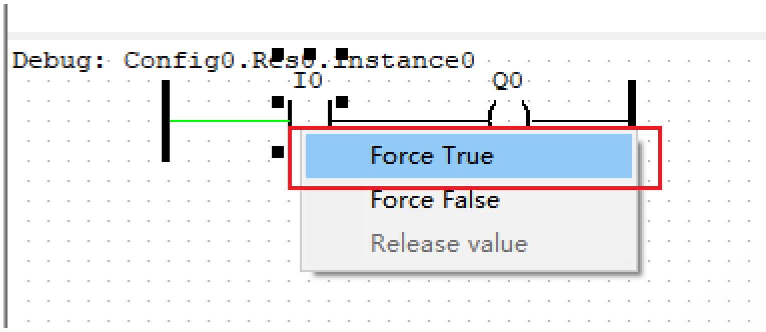
After programming, it is necessary to determine whether the logic is correct and the function is complete, and then upload it to run in PLC, the simulation function of the OpenPLC editor can be used to show how a program is performing at every step. Take the 'single program' mentioned before as an example:

Figure 4-13 PLC Simulation



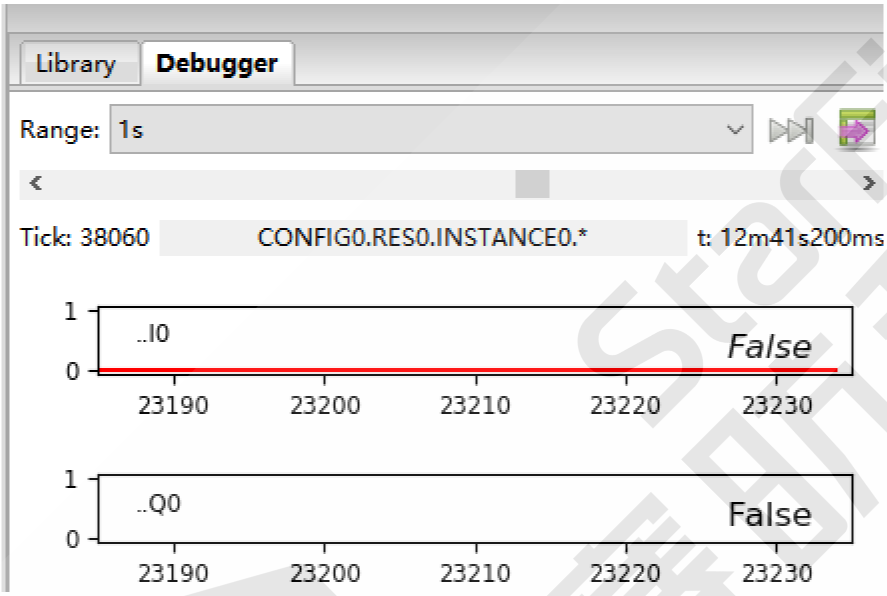
As shown in the figure above, click **Start PLC Simulation** and **Debug instance**, and a new tab **Config0.Res0.instance0** will appear. Then the value of input point **I0** can be changed and the variation of point **Q0** can be observed.

Figure 4-14 Config0.Res0.instance0



The current values of each point can be seen from the Debugger bar on the right side:

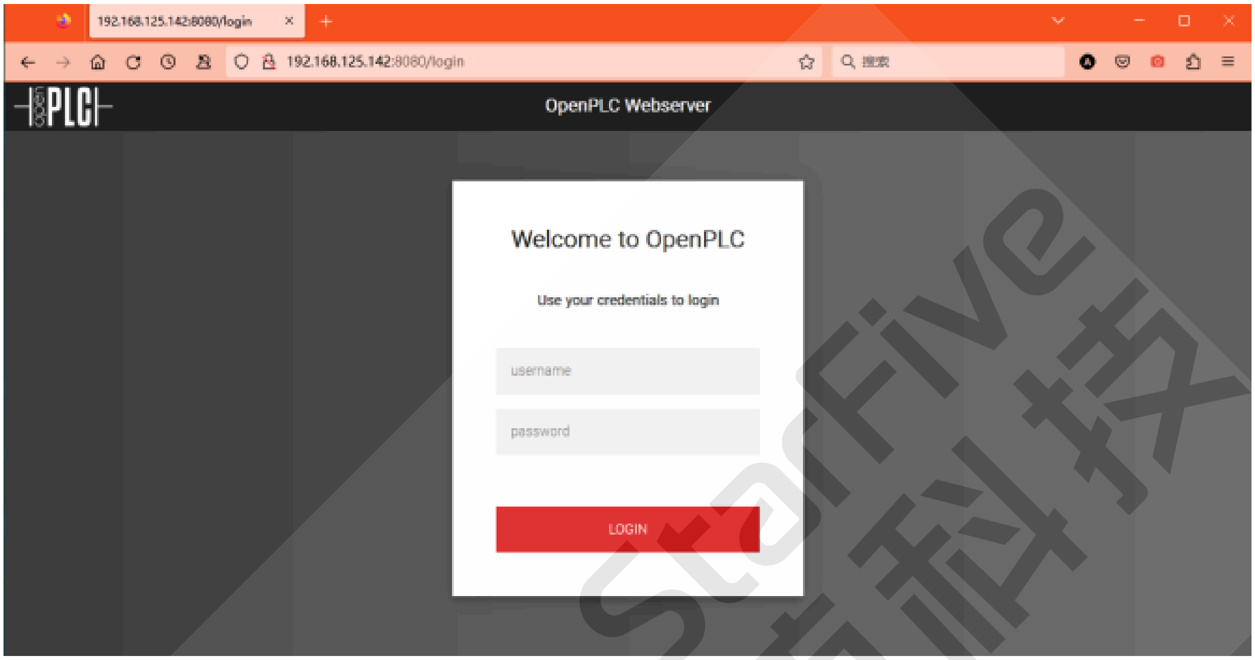
Figure 4-15 Debugger Bar



5. OpenPLC HMI

After installation and restart of the system, the OpenPLC HMI (Human-Machine Interface) can be viewed by accessing port 8080 of VisionFive 2 or VisionFive 2 Lite in your browser:

Figure 5-1 OpenPLC HMI

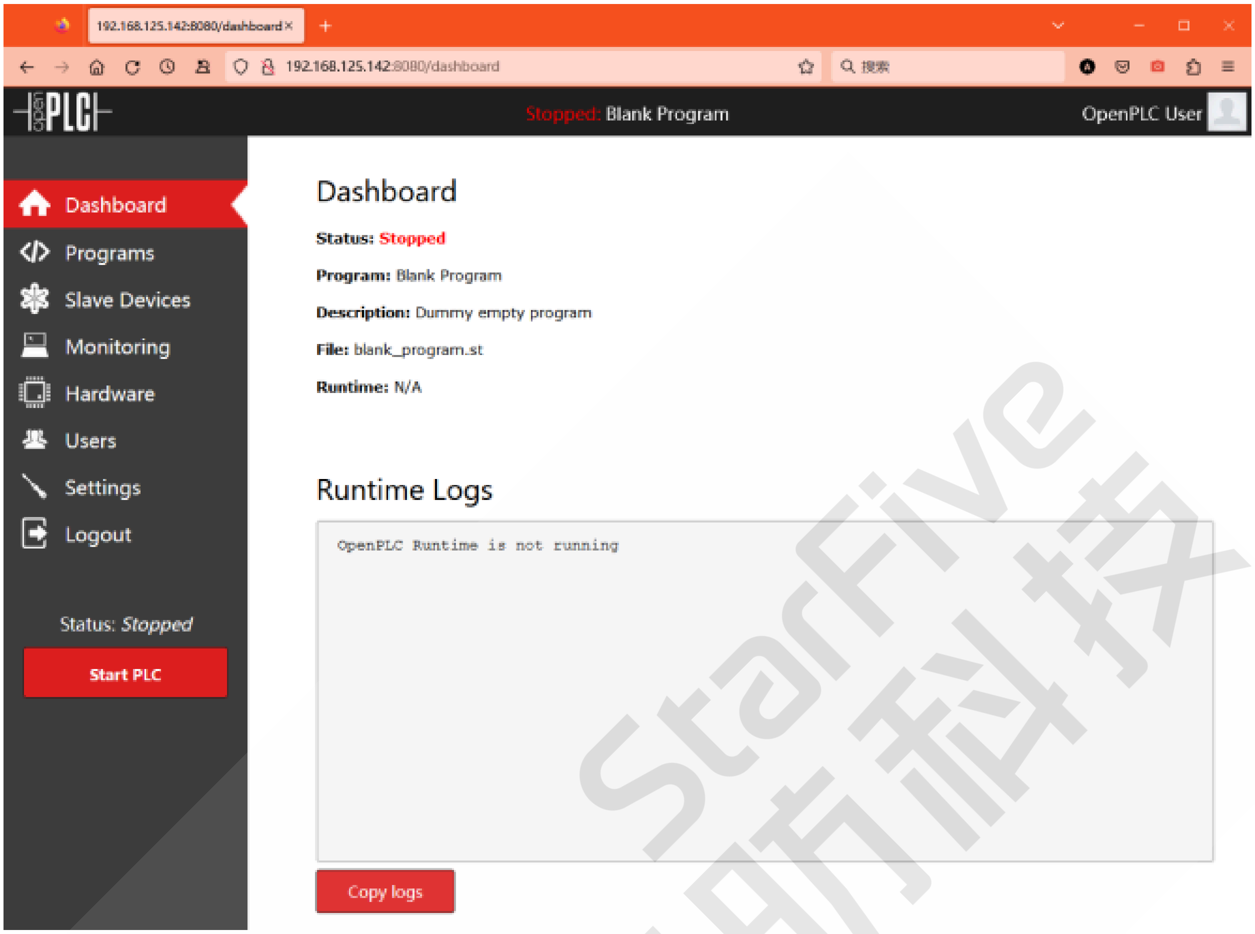


The followings are the default credentials and can be modified in the **Users** menu:

- Username: openplc
- Password: openplc

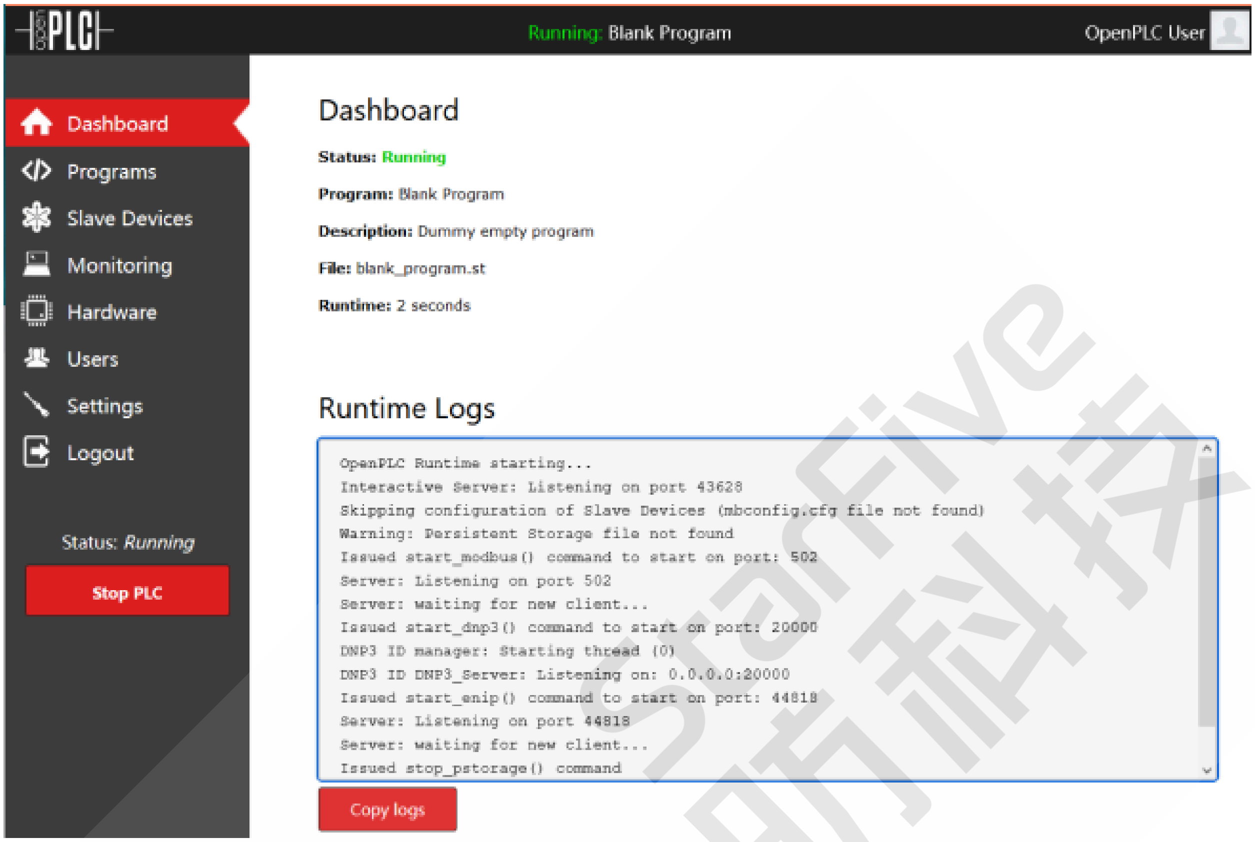
The HMI main page is shown below by default:

Figure 5-2 HMI Main Page



5.1. Run OpenPLC

Figure 5-3 Dashboard



On the **Dashboard** menu, the information on OpenPLC status, active program, and runtime log are shown.

After clicking the **Start PLC** button, OpenPLC runtime will start, and the information of runtime will be printed in the **Runtime Log** field.

It's recommended to run the CPU at the highest frequency by running the following commands:

```
su -
echo performance > /sys/devices/system/cpu/cpu0/cpufreq/scaling_governor
exit
```

to achieve optimal system performance.

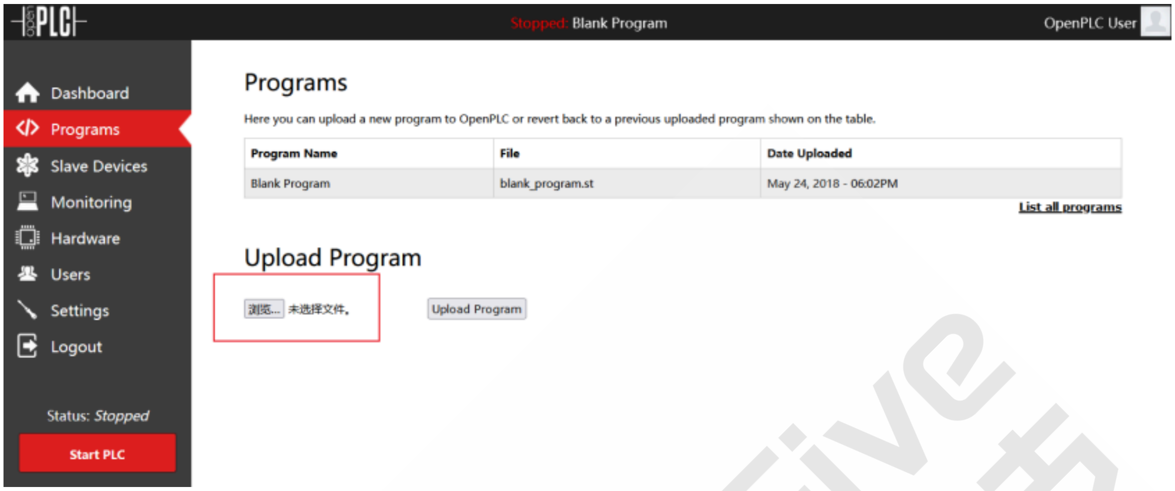
5.2. Upload New Program

On the **Programs** menu, the new program can be updated to OpenPLC or reverted to a previously uploaded program shown on the table.

Take the example of uploading a new program: and click

1. Click the label **view** to choose a new program:

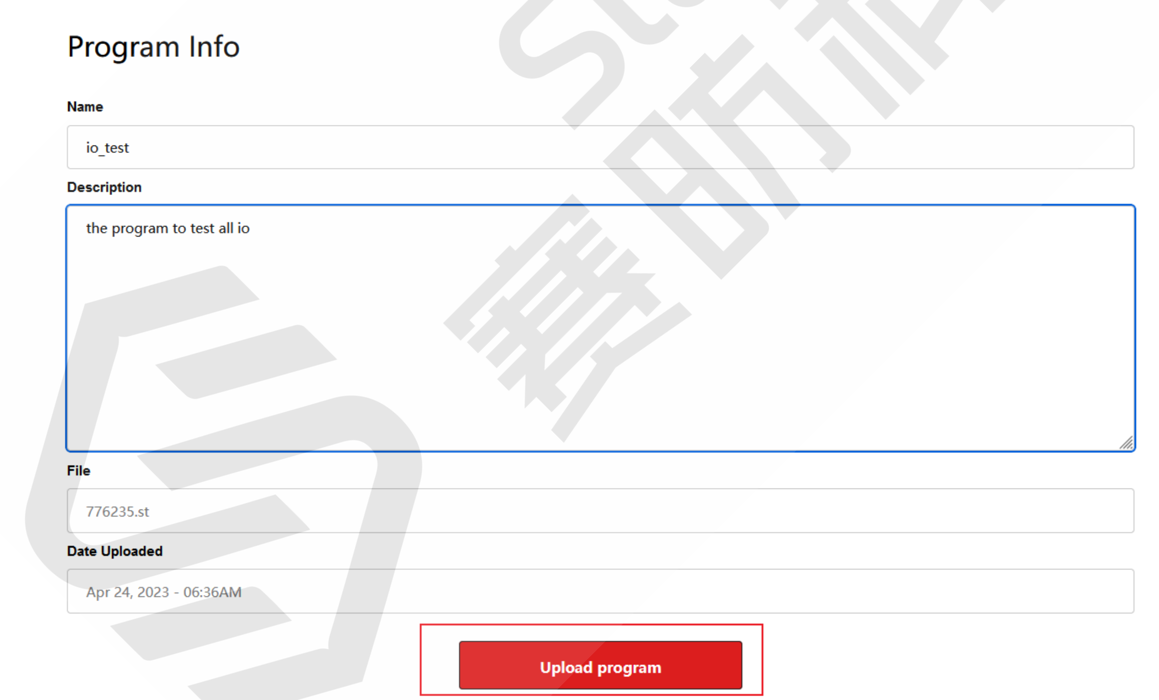
Figure 5-4 Uploading a New Program



2. Select an `.st` file for the new program and click **Open**. The name of the file that will be uploaded is then displayed next to the 'view' label.

3. Click the label **Upload Program**, complete the information, and click **Upload Program**

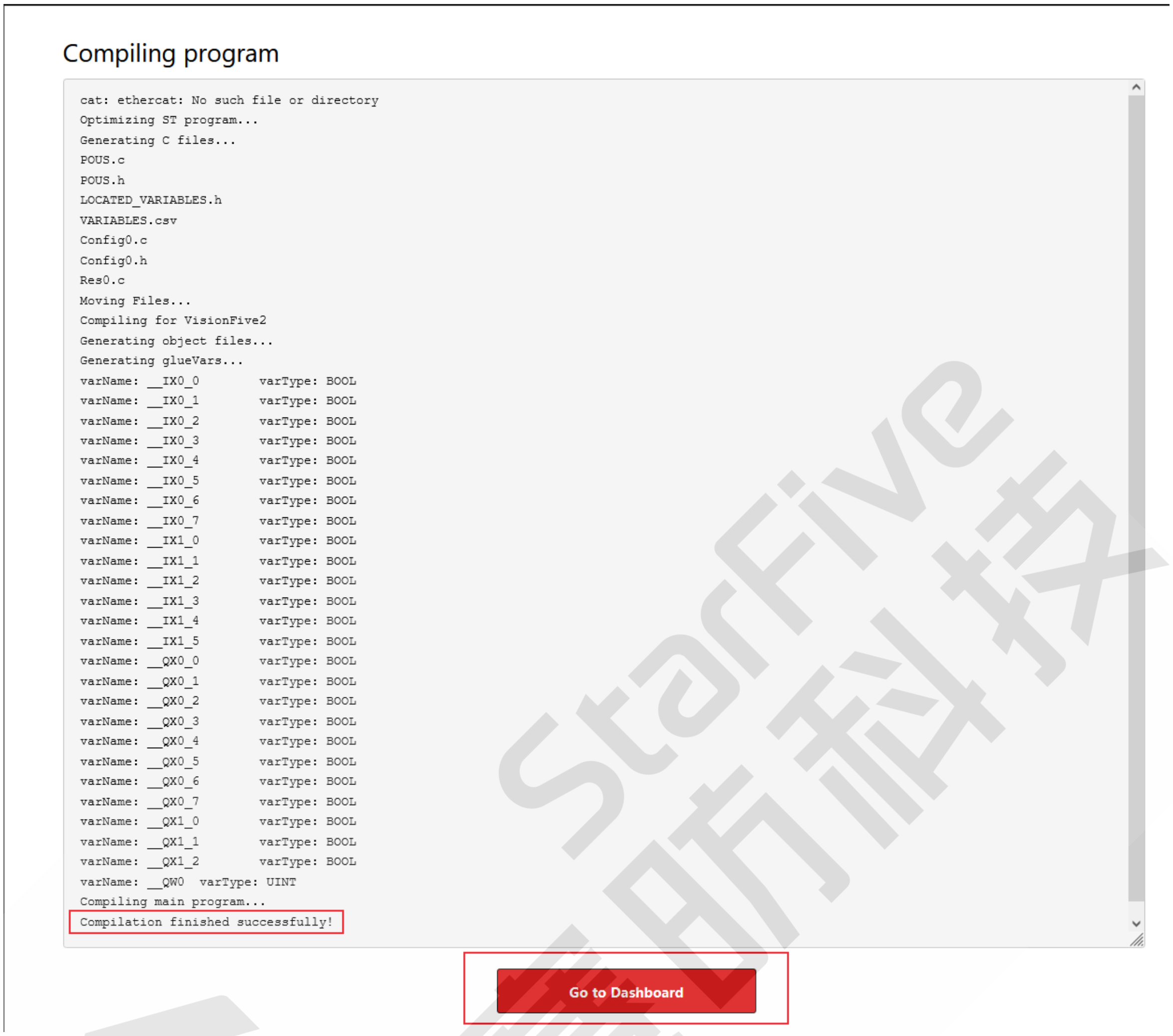
Figure 5-5 Example Program Information



Result:

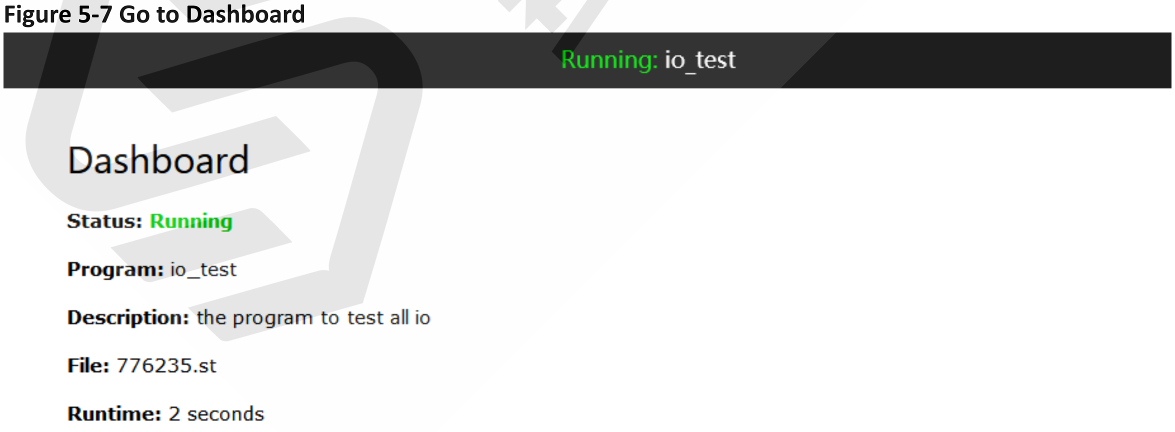
After the successful compilation, the compilation log will be printed, and information that "*Compilation finished successfully!*" will be displayed.

Figure 5-6 Compiling Program



4. Click **Go to Dashboard** and Start the OpenPLC runtime, the information about the running program can be viewed:

Figure 5-7 Go to Dashboard



5.3. Connect slave device

5.3.1. Modbus-TCP

Modbus TCP is an industrial Ethernet protocol based on TCP, which enables the OpenPLC Modbus-TCP master to connect sub-devices and achieve PLC I/O expansion.

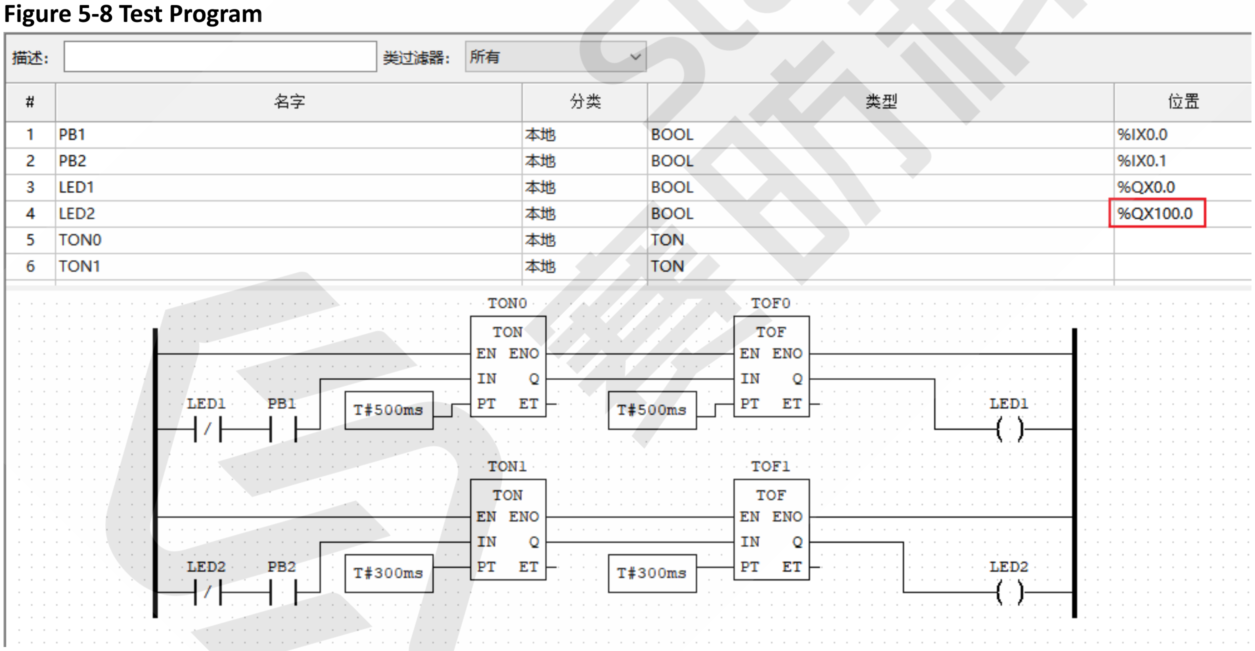
Table 5-1 Hardware Preparation

Items	Number	Note
VisionFive 2 or VisionFive 2 Lite	2	One is the master and the other is the slave.
Breadboard	1	-
LED	2	Preferably two different colors.
Button	2	-
Dupont line	Several	-

5.3.1.1. Test program

Prepare two pieces of VisionFive 2 or VisionFive 2 Lite in the same network segment with OpenPLC runtime, and opload the PLC ladder diagram program into the main device. The function of the test program is to flash the corresponding LED light when the key switch is pressed, and set different flashing frequencies for the two LEDs:

Figure 5-8 Test Program



It should be noted that the I/O point starting address of the slave device is mapped to 100.0 in a master device (eg. The starting address of digital output is %QX0.0 in the slave device, and it's mapped as %QX100.0 in the master device.

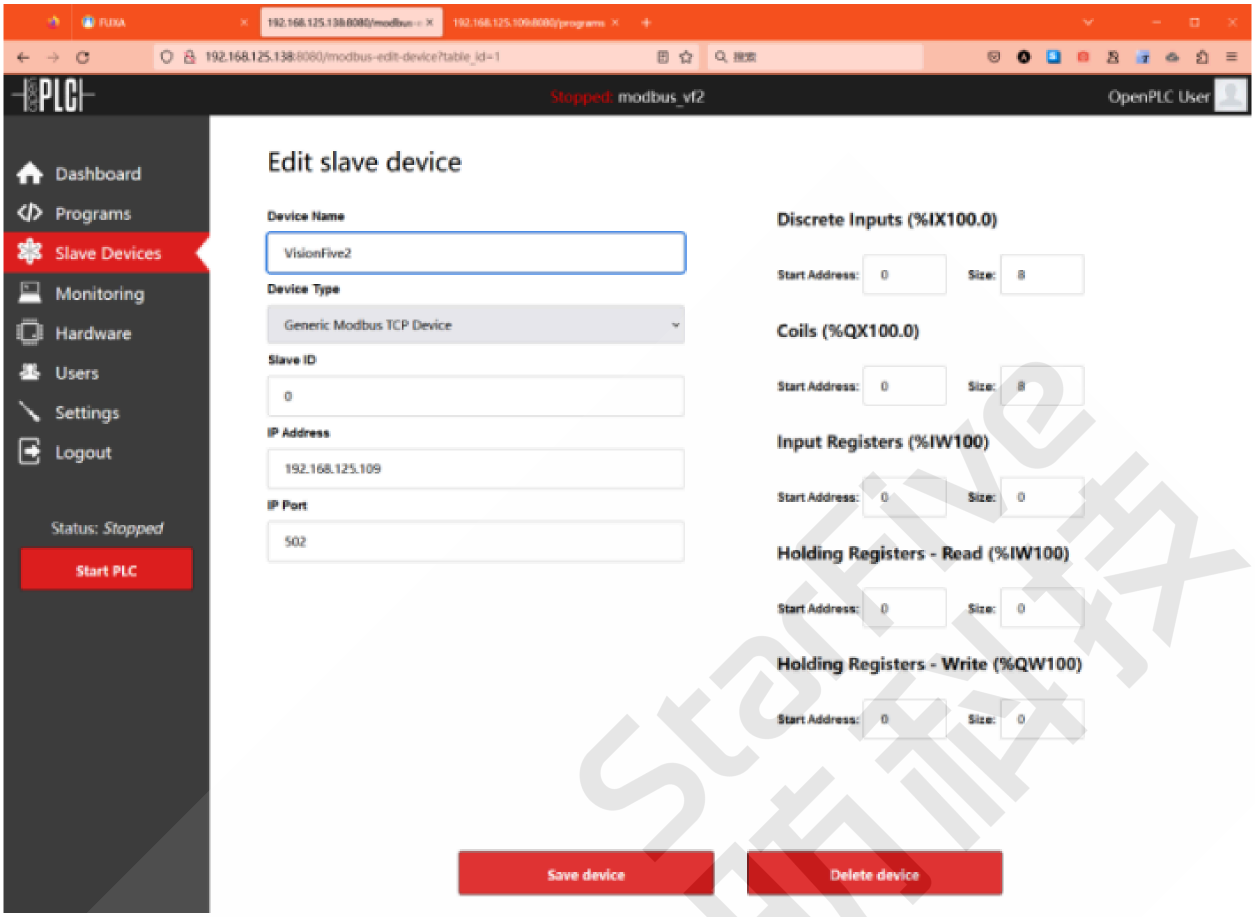
Note:
Details see in Pin mapping part in: https://openplcproject.com/docs/2-6-slave-devices/

Ladder diagram program is located in your computer after you download and unzipped the `openplc.zip`, for example: `C:\Users\chloe.chen\Downloads\openplc\5.3 Connect slave device.zip`.

5.3.1.2. Settings of Slave Device

Entry the Slave device menu in HMI, enter a device name, select the type of the device as Generic Modbus-TCP Device, enter a slave ID and the IP of the slave device, and set the IP port as default 502.

Figure 5-9 Slave Device Settings

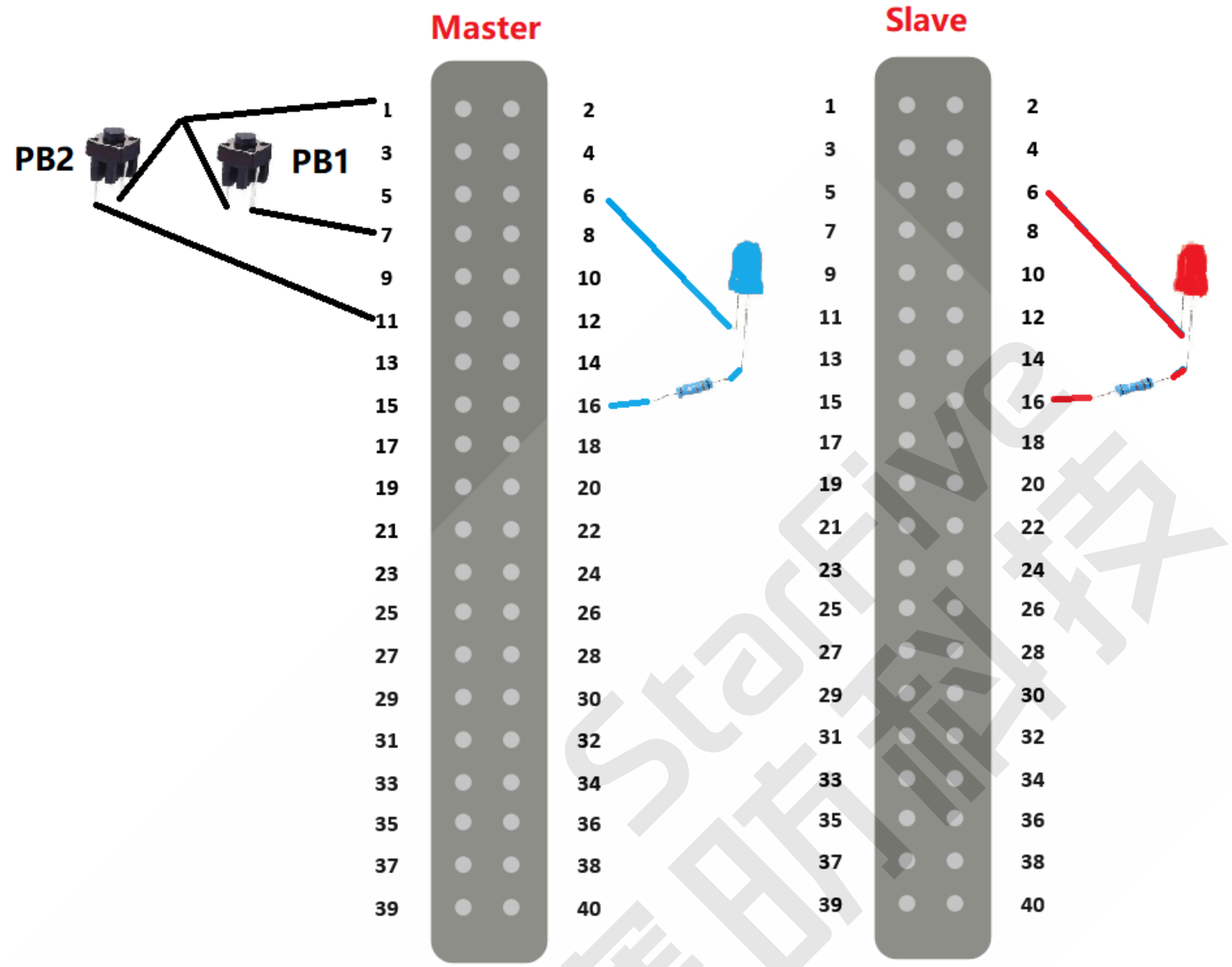


The size on the right is the number of the available I/O point, for example, if there are 8 digital input points, 8 digital output points, and 1 analog output point configured on VisionFive 2 or VisionFive 2 Lite, then the size of Discrete Inputs, Coils, and Holding Registers-Writer can be defined as 8, 8 and 1 accordingly.

5.3.1.3. Hardware circuit:

As shown in the figure below, two pieces of VisionFive 2 or VisionFive 2 Lite are selected as the Modbus-TCP master/slave devices respectively. The button that controls the blue/red led flashing is connected to %IX0.0 and %IX0.1 of the master device, the blue led is connected to %QX0.0 of the master device, and the red led is connected to %QX0.0 of the slave device, which is %QX100.0 of the master device. There is no electrical connection between the master and slave devices.

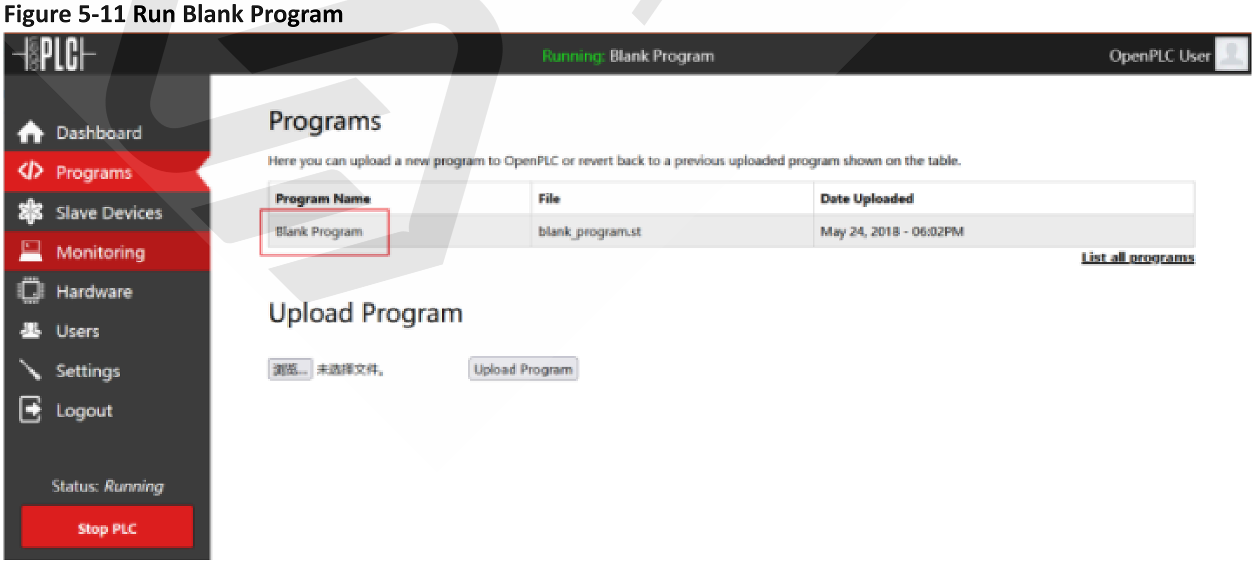
Figure 5-10 Connect to 40-Pin GPIO Header



Verification

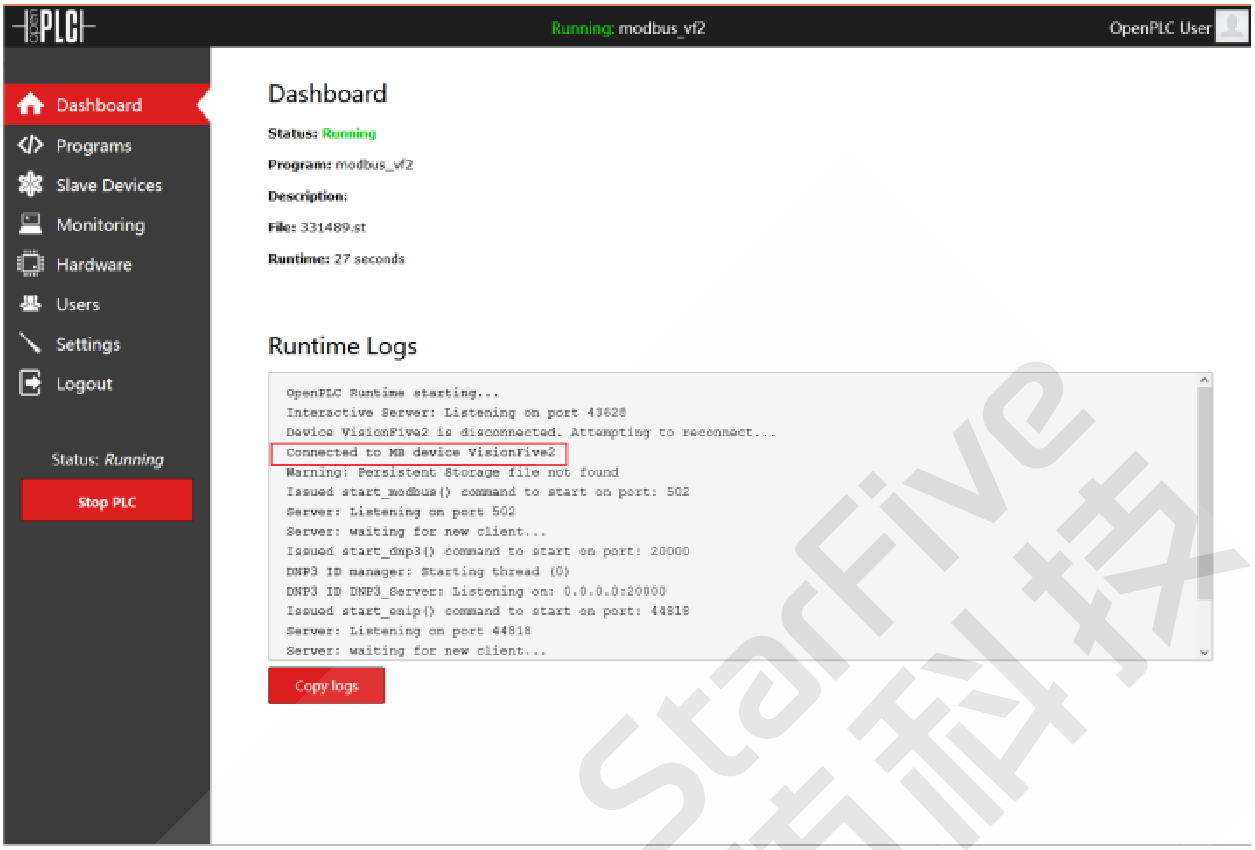
When running as a Modbus-TCP slave, the slave device must run the default Blank Program:

Figure 5-11 Run Blank Program



After confirming the ladder diagram program to be launched by the main device and the configurations of the slave device is correct, click **Start PLC** to start the OpenPLC runtime. It can be seen that the prompt of successful Modbus-TCP device connection appears in the Runtime Logs:

Figure 5-12 Successful Connection



Then, press the button that controls the red led blinking and check whether the red light starts to blink to determine whether the master and slave devices communicate properly.

5.3.2. Modbus-RTU

Modbus-RTU enables the Modbus communication between OpenPLC master and slave through serial. OpenPLC only supports Modbus-RTU master on Linux and Windows. Microrun-time devices (such as Arduino) without an operating system are supported only as Modbus-RTU slaves: https://openplcproject.com/docs/2-5-modbus-addressing/

Table 5-2 Hardware Preparation

Items	Number	Note
VisionFive 2 or VisionFive 2 Lite	2	One is the master and the other is the slave.
Breadboard	1	-
LED	2	Preferably two different colors.
Button	2	-
Dupont line	Several	-
USB2Serial	1	Corresponding to the enabled type in the kernel options.

5.3.2.1. Check USB2Serial Connection

After plugging in the USB2Serial module, run `ls /dev/ttyUSB*` to check whether the USB2Serial device exists.

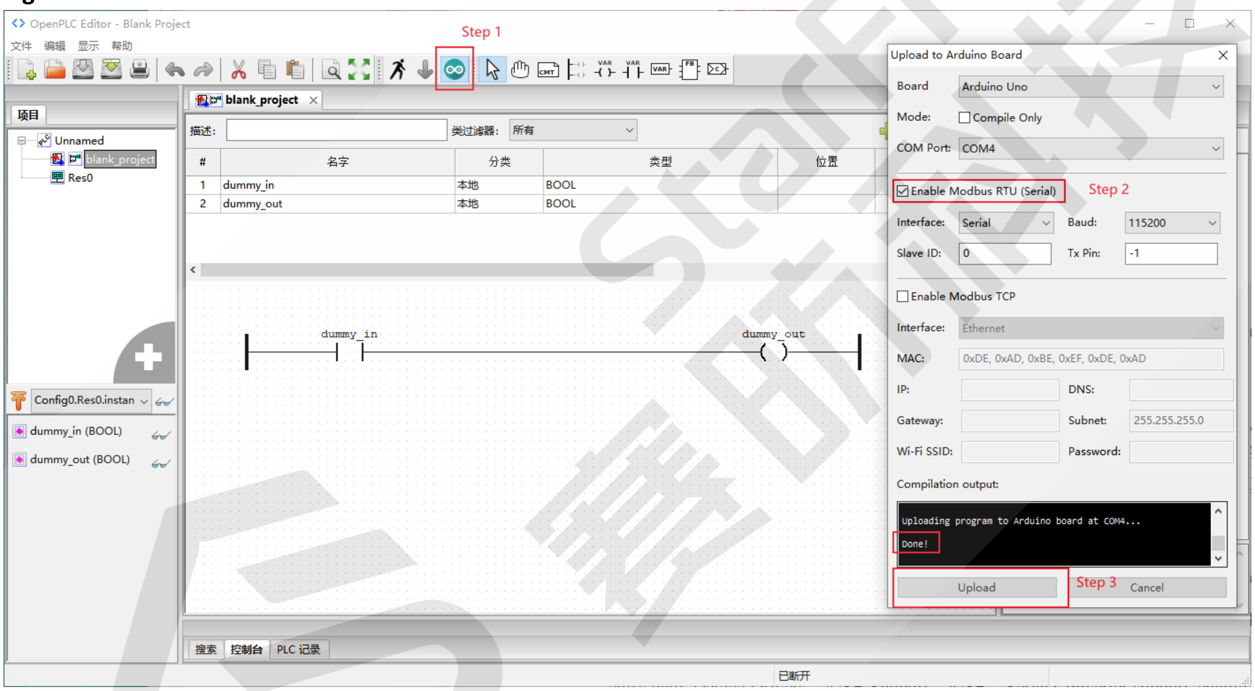
The following is an example figure:

Figure 5-13 Check USB2Serial Connection

5.3.2.2. Burn the OpenPLC Runtime for Arduino

1. Connect Arduino UNO and PC by USB;
2. Open the OpenPLC runtime and prepare the program needed to run Arduino UNO as a Modbus-TCP slave device;
3. Click the icon, as in Step 1, to open the detailed settings. Choose **Board type** as Arduino UNO and select the corresponding COM port. Select **Enable Modbus RTU (Serial)** to enable communication between the Arduino UNO and the master device via Serial;
4. Click the **Upload** button to compile and upload the program and OpenPLC micro-runtime to the Arduino UNO. The message **Done** will be displayed when the upload process is completed.

Figure 5-14 Arduino UNO



The special blank ladder diagram program to the Arduino UNO by reference to: https://openplcproject.com/docs/2-6-slave-devices/

Then upload the ladder diagram program (the same as the program in Modbus-TCP) to the Modbus-RTU master (VisionFive 2 or VisionFive 2 Lite).

5.3.2.3. Hardware Setup

The hardware connection is shown below:

Figure 5-15 Photo of Hardware Setup

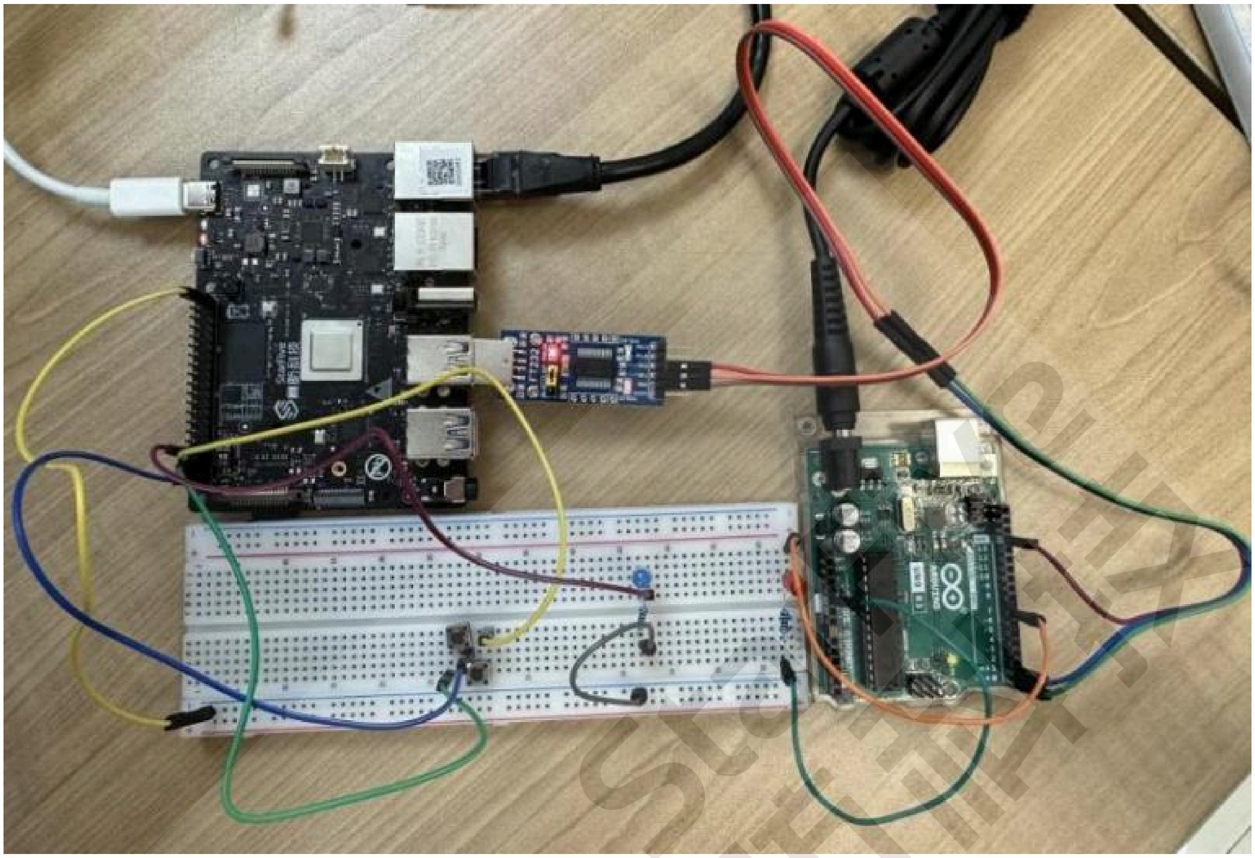
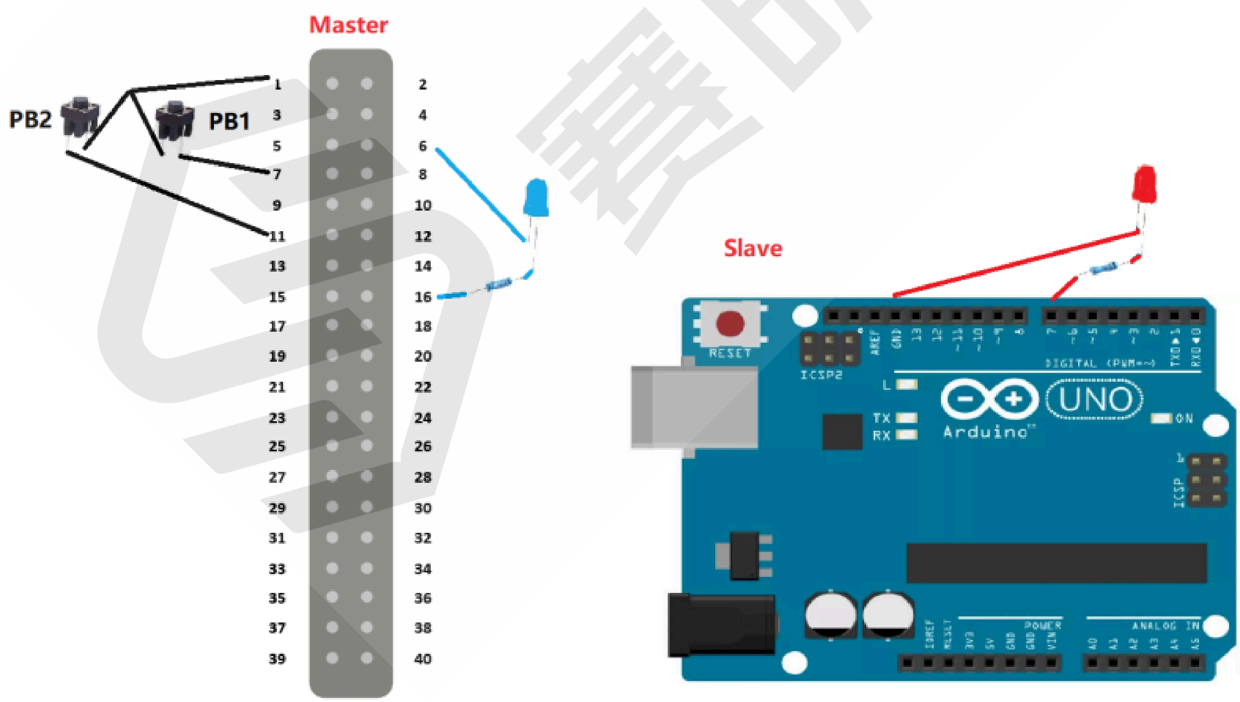


Figure 5-16 Hardware Setup

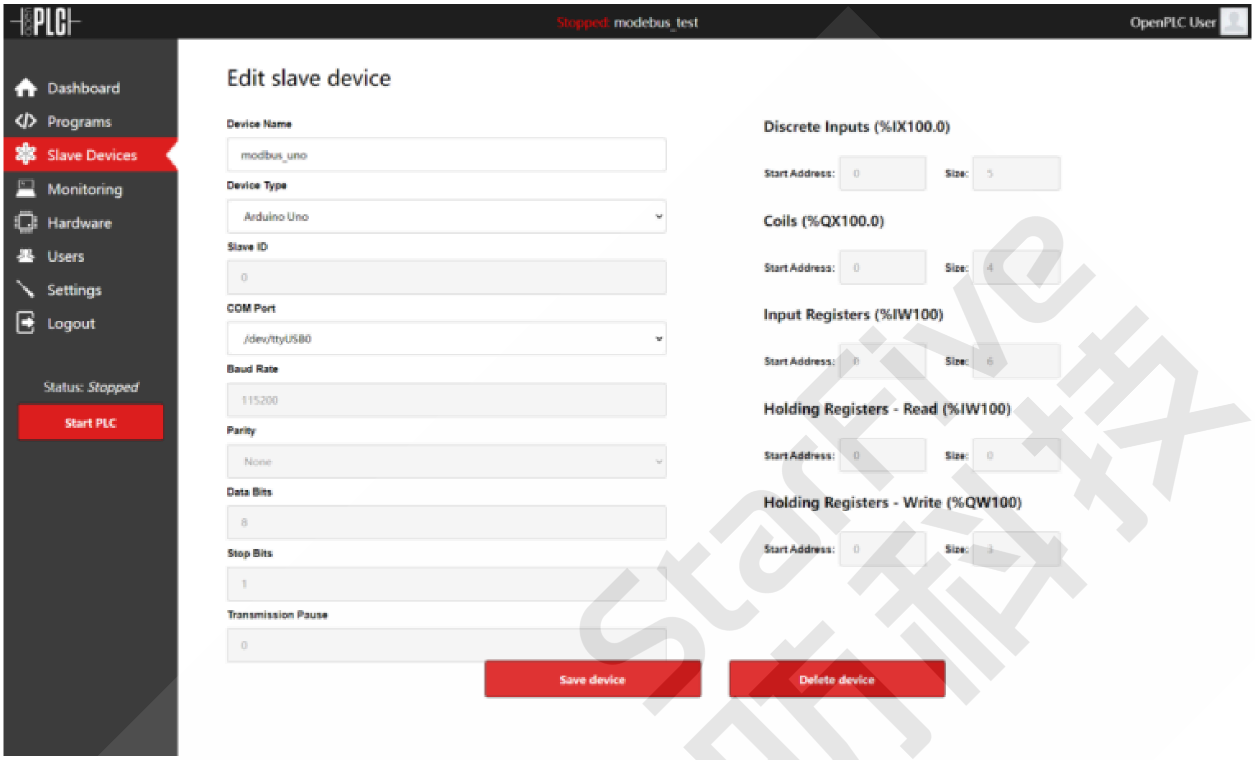


> **Note:**
> The Arduino UNO here is the original Italian version, and the built-in USB2 Serial chip is ATMEGA16U2. So that the VisionFive 2 or VisionFive 2 Lite and Arduino UNO cannot directly communicate through the serial by using USB2USB, the FT232 USB2Serial module of Waveshare is selected.

5.3.2.4. Configure Slave Device

OpenPLC supports Arduino UNO by default, only the device name, device type, and connected port need to be configured, the COM Port should be the same as '/dev/ttyUSB*' above:

Figure 5-17 Configure Slave Device

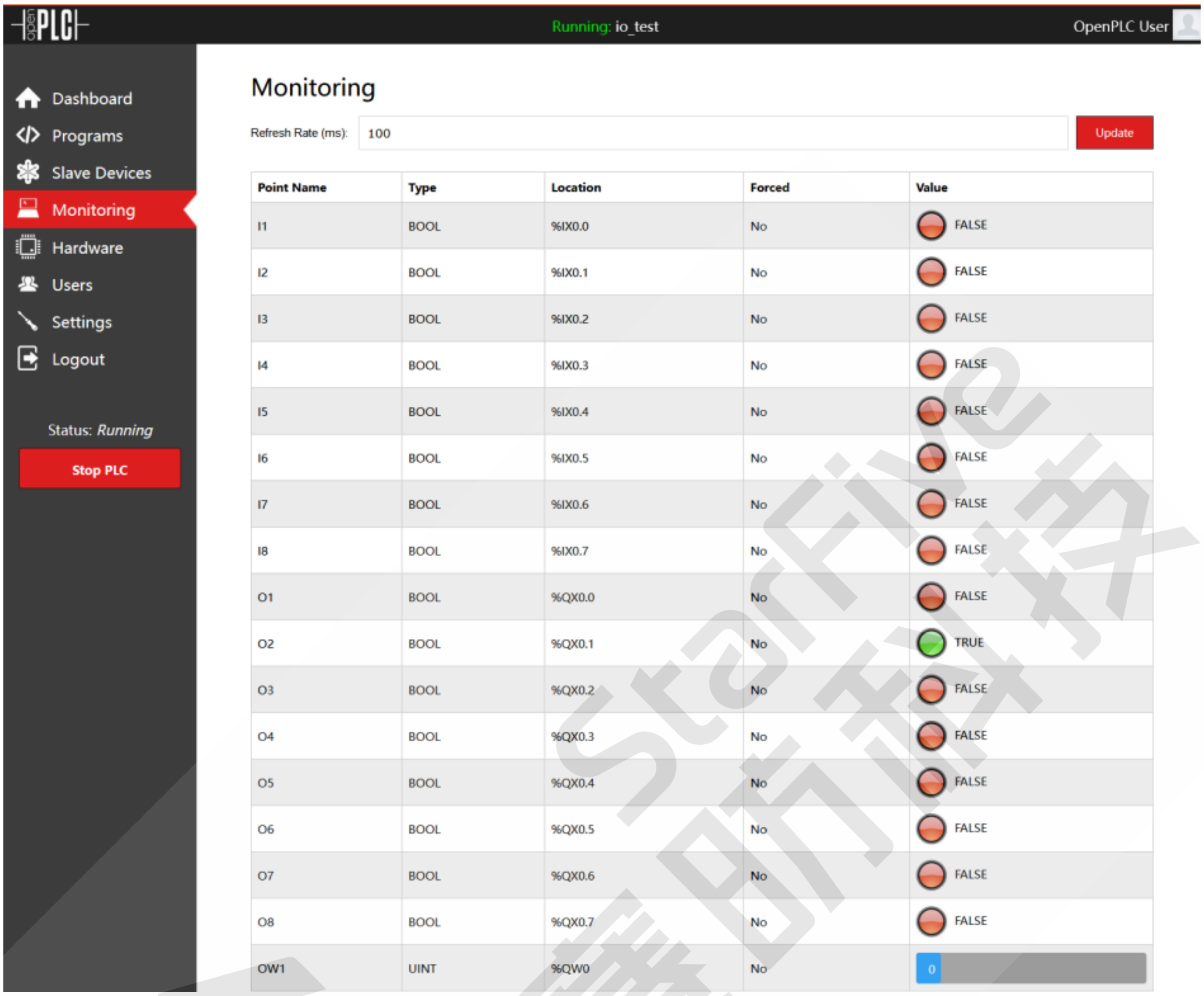


After completing the configuration and saving, start PLC Runtime. VisionFive 2 or VisionFive 2 Lite can connect Arduino through serial and use it as a sub-device of Modbus-RTU to expand its IO. The verification of the connections of Modbus-RTU master and slave is the same as Modbus-TCP.

5.4. Monitoring

In the **Monitoring** menu, the state of each I/O point defined in the ladder diagram can be seen. For example, after uploading the program `io_test.st` mentioned in the Basic digital I/O & Analog Output Test *(on page 34)* section and starting the OpenPLC runtime, the state and details of each I/O points are shown as:

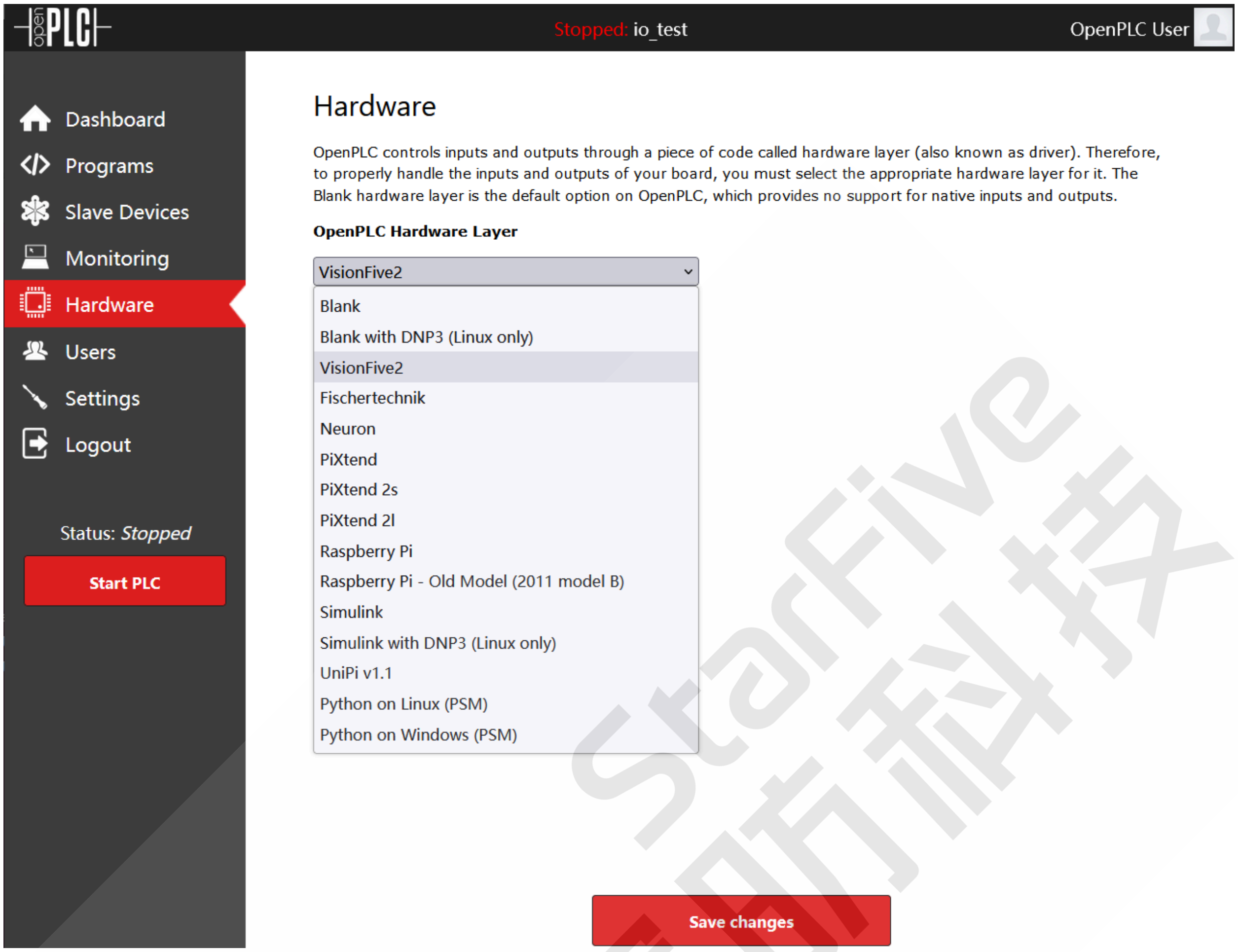
Figure 5-18 Running io_test



5.5. Hardware Layer

OpenPLC controls inputs and outputs through a piece of code called hardware layer, different hardware layers can be selected for different devices.

Figure 5-19 Hardware Layer



For VisionFive 2 or VisionFive 2 Lite, the corresponding hardware layer is selected by default after installation.

5.6. Users and Settings

The user of OpenPLC can be added or modified in the **Users** menu. And some settings, such as ports of server and start mode can be changed in the **Settings** menu.

6. Basic Functions

This section record OpenPLC digital input/output, analog output, and basic program function block test methods and results.

Table 6-1 Hardware Preparation

Items	Number	Note
VisionFive 2 or VisionFive 2 Lite	1	-
Breadboard	1	-
LED	Several	Used for receiving output.
Button	Several	Used for control input.
Resistors	Several	100ohms, connected in series with led to prevent it from burning out.
L298N	1	DC motor driver board, used to test analog output.
DC motor	1	-
Dupont line	Several	-
USB2Serial	1	Corresponding to the enabled type in the kernel options.

Ladder diagram program is located in your computer after you download and unzipped the `openplc.zip`, for example: `C:\Users\chloe.chen\Downloads\openplc\6. Basic functions.zip`.

6.1. Basic digital I/O & Analog Output Test

The OpenPLC hardware layer file for VisionFive 2 or VisionFive 2 Lite defines 14 digital outputs, 11 digital inputs, and 1 analog output, the basic functions need to be verified. The figure below shows the ladder diagram program and the address of each input/output point.

Figure 6-1 Ladder Diagram

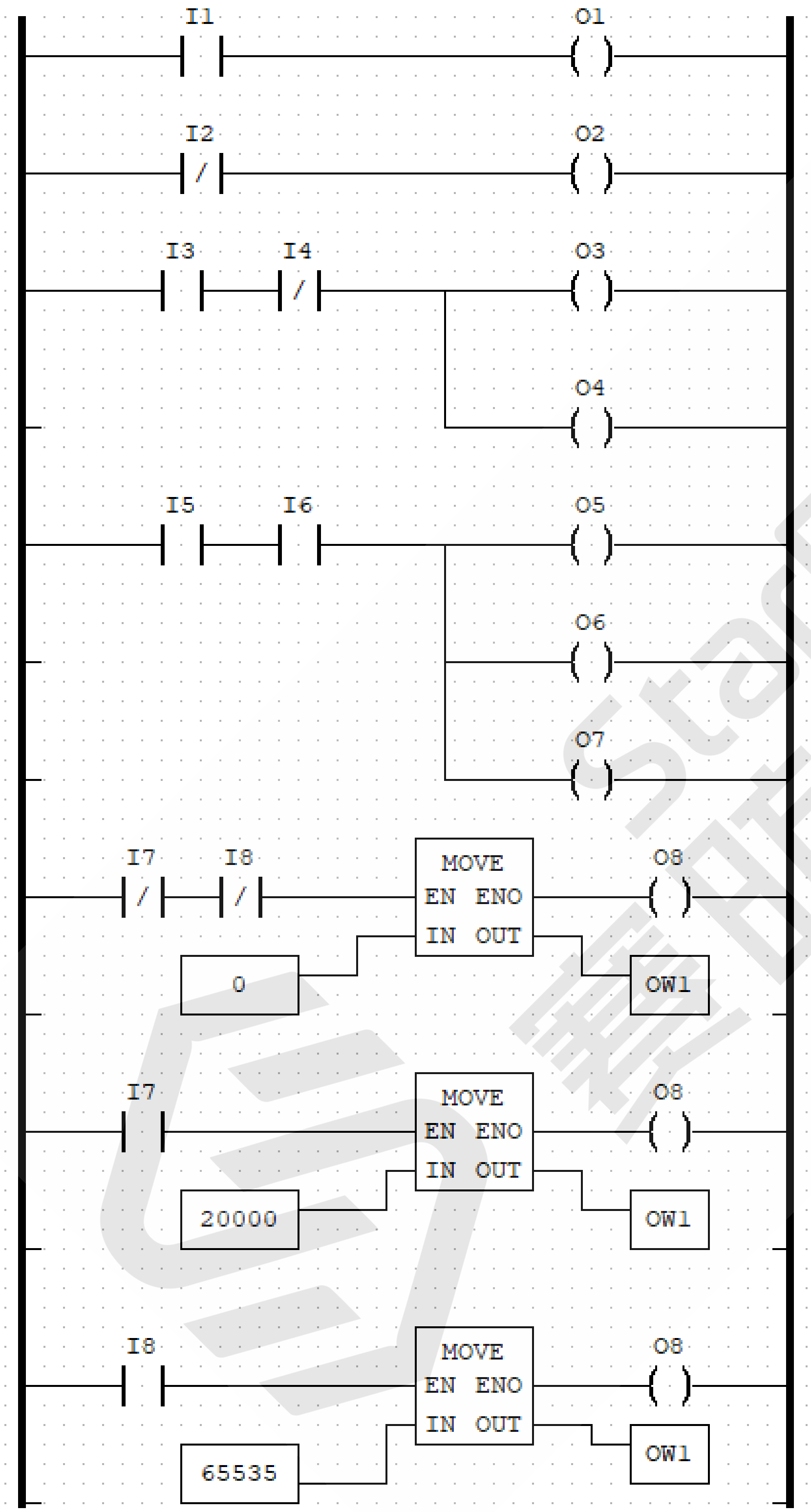
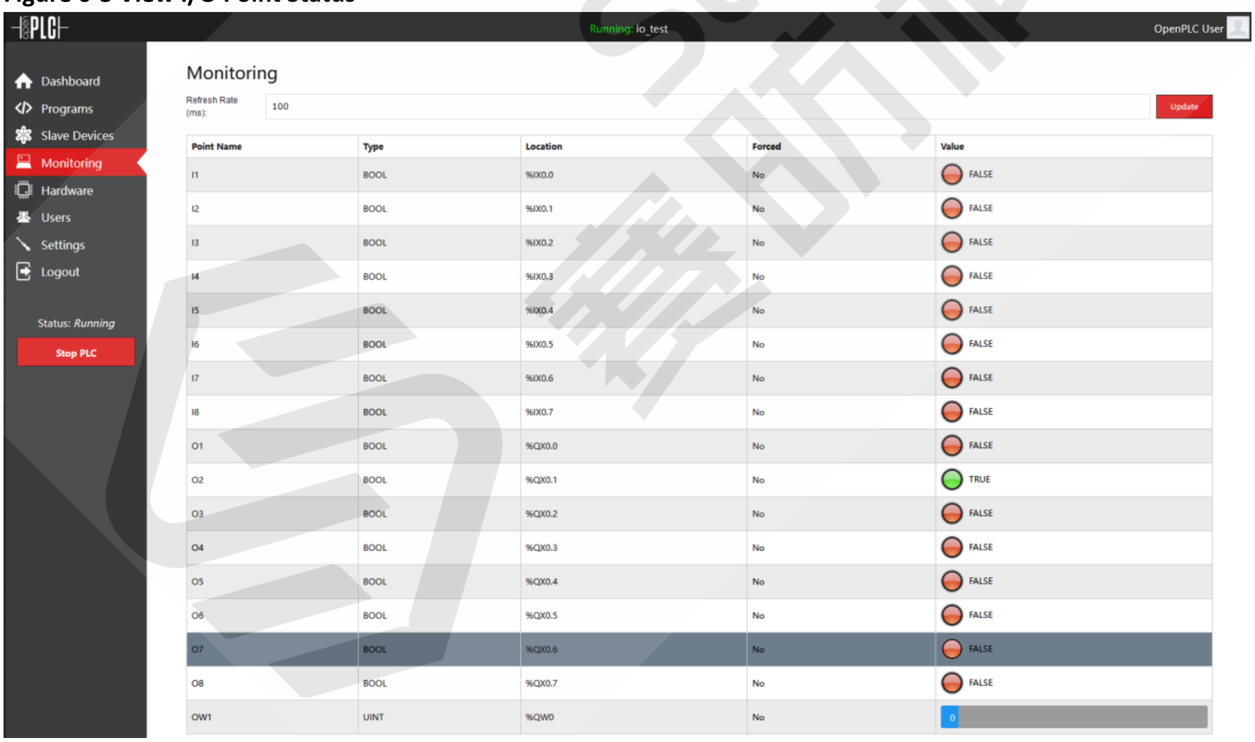


Figure 6-2 Address of Each Input/Output Point

I1	本地	BOOL	%IX0.0
I2	本地	BOOL	%IX0.1
I3	本地	BOOL	%IX0.2
I4	本地	BOOL	%IX0.3
I5	本地	BOOL	%IX0.4
I6	本地	BOOL	%IX0.5
I7	本地	BOOL	%IX0.6
I8	本地	BOOL	%IX0.7
O1	本地	BOOL	%QX0.0
O2	本地	BOOL	%QX0.1
O3	本地	BOOL	%QX0.2
O4	本地	BOOL	%QX0.3
O5	本地	BOOL	%QX0.4
O6	本地	BOOL	%QX0.5
O7	本地	BOOL	%QX0.6
O8	本地	BOOL	%QX0.7
OW1	本地	UINT	%QW0

In the first two lines of this Ladder diagram program, I1, I2, and O1, O2 were tested for positive and negative logic switches. I3, I4, O3, O4, and I5, O5, O6, and O7 are tested for serial and parallel logic respectively. In the last three lines of the ladder diagram, the analog output of OpenPLC is tested based on I7, I8, and O8, and the DC motor drive and speed governing are realized through the DC motor drive board (L298N). The connection of Led, resistance, button, and Dupont is omitted here. The connection of the DC motor and L298N can see in Temperature Control *(on page 65)*. The status of I/O points can be viewed on the HMI as shown in the following figure:

Figure 6-3 View I/O Point Status

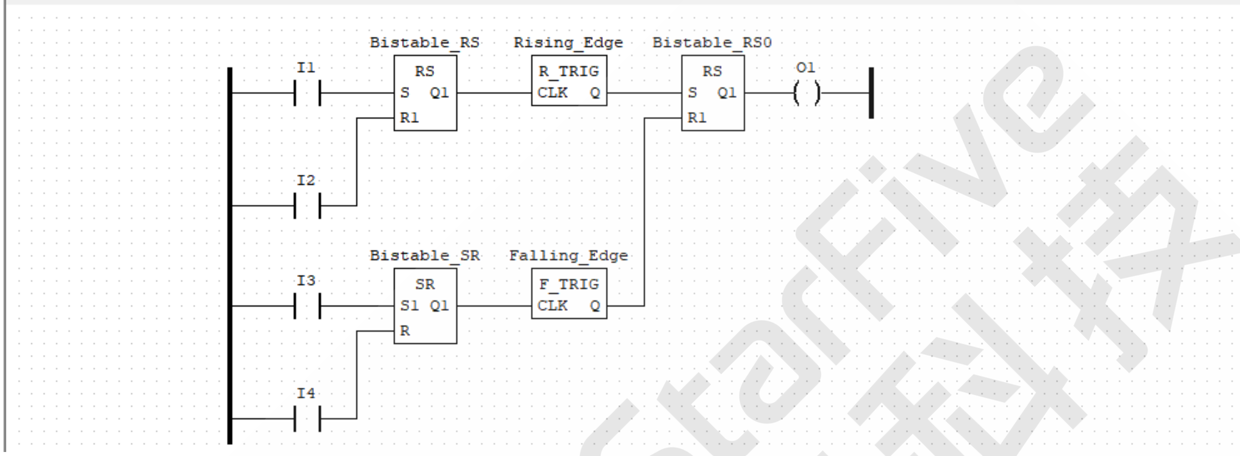


6.2. Bistable Test

The bistable elements and edge detection are tested in this section. IEC 61131-3 defines two types of bistable elements: set dominant (SR) and reset dominant (RS). Inputs I1 and I2 are connected to an RS bistable and inputs I3 and I4 are connected to an SR bistable. The output of the RS and SR bistable block goes to a rising edge detection and a falling edge detection block respectively. Finally, the rising edge detector is connected to the set input of the bistable RS0 and the falling edge detector is connected to the reset input of the bistable RS1.

Figure 6-4 Bistable Test

#	名字	分类	类型	位置
1	Bistable_RS	本地	RS	
2	Bistable_SR	本地	SR	
3	Rising_Edge	本地	R_TRIG	
4	Falling_Edge	本地	F_TRIG	
5	Bistable_RS0	本地	RS	
6	I1	本地	BOOL	%IX0.2
7	I2	本地	BOOL	%IX0.3
8	I3	本地	BOOL	%IX0.4
9	I4	本地	BOOL	%IX0.5
10	O1	本地	BOOL	%QX0.0



In this program, the output O1 is set if input I1 is active before inputs I2 and I3. The output should remain set until I2 is triggered or I3-I4 is triggered in sequence. Detailed changes in the output point can also be observed in HMI.

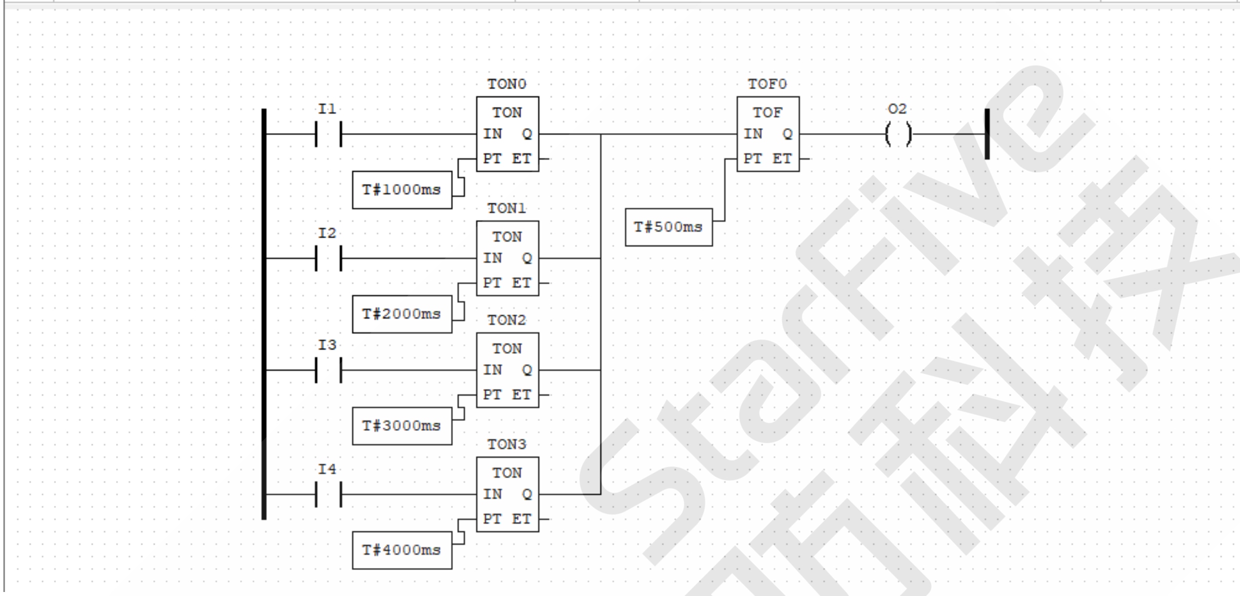
6.3. Timer Test

Timer elements are tested in this section. Both TON and TOF types were used, with different expiration times. As per IEC 61131-3 definition, a TON timer supplies a rising edge of the input IN at the output Q after a time delay PT. If the input pulse is shorter than PT, the timer is not started and the output Q remains false. The TOF timer performs the inverse function to TON, i.e. it supplies a rising edge on the input IN at the output Q immediately but delays a falling edge of the input into the output Q.

For the segment of the ladder test program shown below, the output O2 is set to true if one of the parallel TON timers receives an input pulse larger than the PT value. Timers TON0 to TON3 are activated by the inputs I1, I2, I3, and I4. O2 should remain active until the TOF timer in the series expires. For example, O2 will be activated if I1 has been active for 1000 ms continuously, and it will be automatically inactivated after 500 ms.

Figure 6-5 Timer Test

#	名字	分类	类型	位置
1	TON0	本地	TON	
2	TON1	本地	TON	
3	TON2	本地	TON	
4	TON3	本地	TON	
5	TOF0	本地	TOF	
6	I1	本地	BOOL	%IX0.2
7	I2	本地	BOOL	%IX0.3
8	I3	本地	BOOL	%IX0.4
9	I4	本地	BOOL	%IX0.5
10	O2	本地	BOOL	%QX0.0



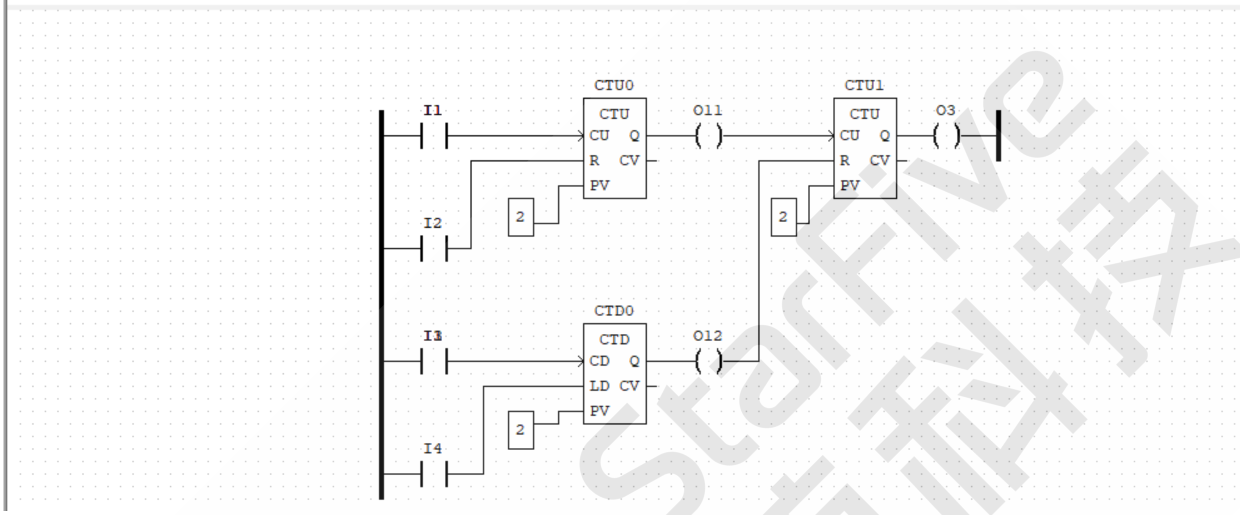
6.4. Counter Test

The counter blocks are verified in this section. Both up (CTU) and down (CTD) counters were used for this test. A CTU counter increments the internal count variable CV on every rising edge of the input CU. If the value in CV matches the user-defined PV variable, the output Q is set to true. In the event of a rising edge in the input R, the internal count variable CV is reset to zero. Similarly, the CTD counter loads the internal CV variable value from PV on the first rising edge of the input CD and decrements CV on every subsequent rising edge of the input CD. Once CV reaches zero, the output Q is set to true. Upon a rising edge on the input LD, the original PV value is restored to CV.

The output O3 is set to true if CTU1 evaluates to true at least two times before a reset pulse is received from CTD0. Inputs I1 and I2 are connected to the CU and R inputs of the CTU0 counter, while inputs I3 and I4 are connected to the CD and LD inputs of the CTD0 counter.

Figure 6-6 Counter Test

#	名字	分类	类型	位置
1	I1	本地	BOOL	%IX0.2
2	I2	本地	BOOL	%IX0.3
3	I3	本地	BOOL	%IX0.4
4	I4	本地	BOOL	%IX0.5
5	O3	本地	BOOL	%QX0.0
6	O11	本地	BOOL	%QX0.1
7	O12	本地	BOOL	%QX0.2
8	CTU0	本地	CTU	
9	CTD0	本地	CTD	
10	CTU1	本地	CTU	



6.5. Arithmetic Test

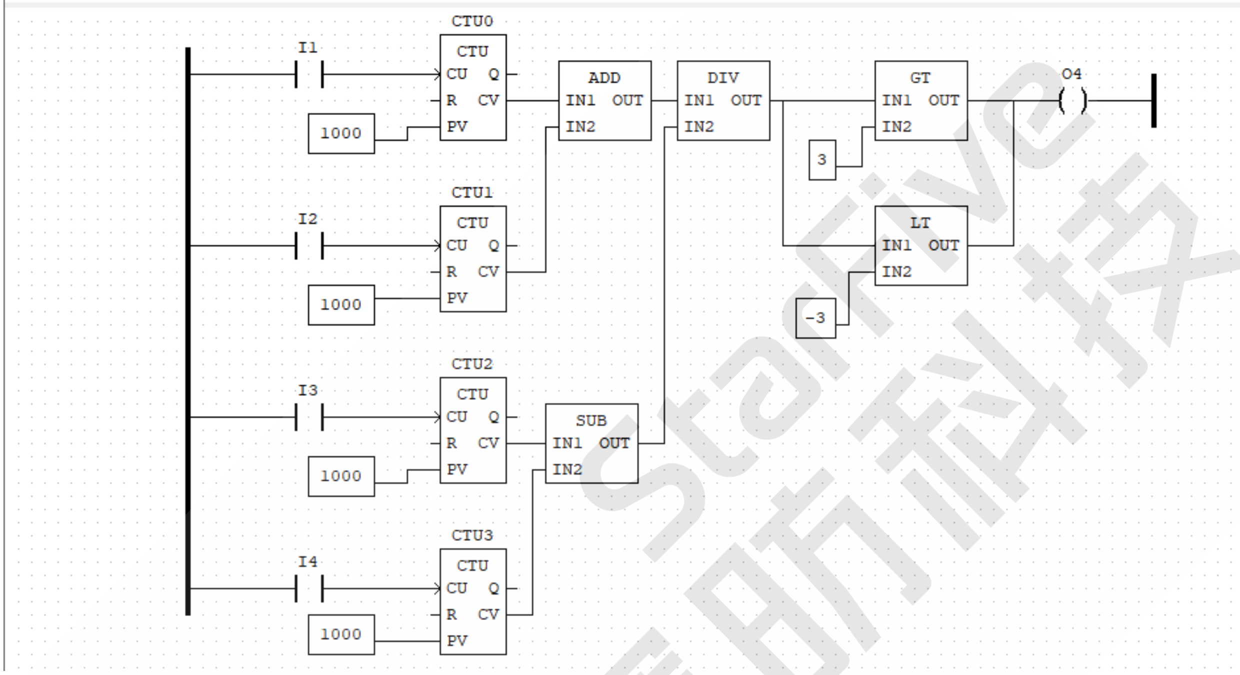
The arithmetic and comparison blocks are tested in this section. For this test, a CTU counter was added to each one of the four inputs. The CV variable for each CTU was wired to arithmetic blocks to perform the following equation:

$$\frac{CTU0.CV + CTU1.CV}{CTU2.CV - CTU3.CV}$$

The result of the arithmetic expression goes to two Comparator blocks. If the result is greater than 3 or less than -3, the output is set to true. The ladder diagram for this test segment is shown below.

Figure 6-7 Arithmetic Test

#	名字	分类	类型	位
1	CTU0	本地	CTU	
2	CTU1	本地	CTU	
3	CTU2	本地	CTU	
4	CTU3	本地	CTU	
5	I1	本地	BOOL	%IX0.2
6	I2	本地	BOOL	%IX0.3
7	I3	本地	BOOL	%IX0.4
8	I4	本地	BOOL	%IX0.5
9	O4	本地	BOOL	%QX0.1



7. SCADA

SCADA means 'supervisory control and data acquisition'. It is used to monitor and control a large area, usually an entire site or factory. SCADA systems are a combination of many systems, including sensors, RTUs (Remote Terminal Units), and PLCS. The data for all these systems is then sent to the central SCADA unit. Some SCADA unit has their own HMI (Human-Machine interface). In OpenPLC, each OpenPLC has its own HMI, that is, Monitor in the webserver, but only displays itself and its slave device. Therefore, SCADA is required to display data information of each unit comprehensively. This document introduces the use of two open-source SCADA software: FUXA and ScadaBR.

Note:

Ladder diagram program is located in your computer after you download and unzipped the `openplc.zip`, for example: `C:\Users\chloe.chen\Downloads\openplc\7. Scada.zip`.

7.1. FUXA

It is recommended to use FUXA as Scada to cooperate with OpenPLC because it can be easily installed onVisionFive 2 or VisionFive 2 Lite.

7.1.1. Prerequisite

Execute the following commands before install FUXA:

1. Install nodejs:

   ```
   sudo apt-get install nodejs -y
   ```

2. Install npm

   ```
   sudo apt-get install npm -y
   ```

Tip:

Network issue may occur in mainland China. Therefore, users in mainland China are recommended to install cnpm by executing

```
npm install cnpm -g --registry=https://registry.npmmirror.com
```

7.1.2. Install FUXA

Execute the following commands to install FUXA:

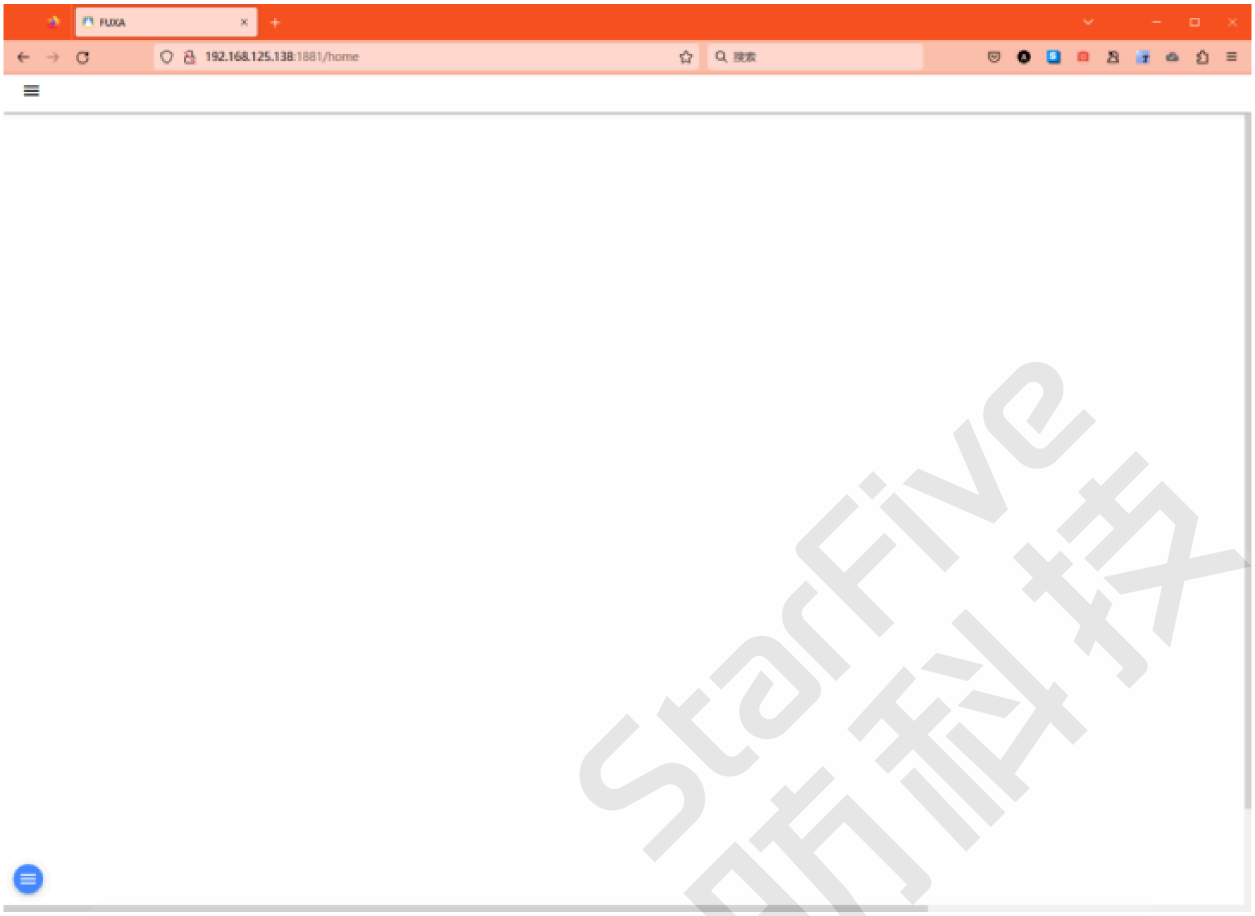
```
git clone https://github.com/frangoteam/FUXA.git
cd FUXA/server/
npm install or cnpm install
Run FUXA: npm start or cnpm start
```

7.1.3. FUXA Operations

7.1.3.1. Login to the Home Page

Login to the web server through port **1881** of the device, for example, `192.168.125.38:1881`, to enter the homepage of FUXA:

Figure 7-1 Enter Homepage of FUXA

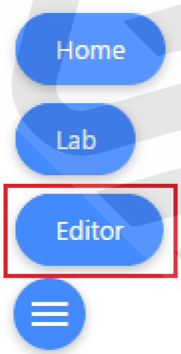


7.1.3.2. Add Modbus Component

Perform the following steps to add Modbus component:

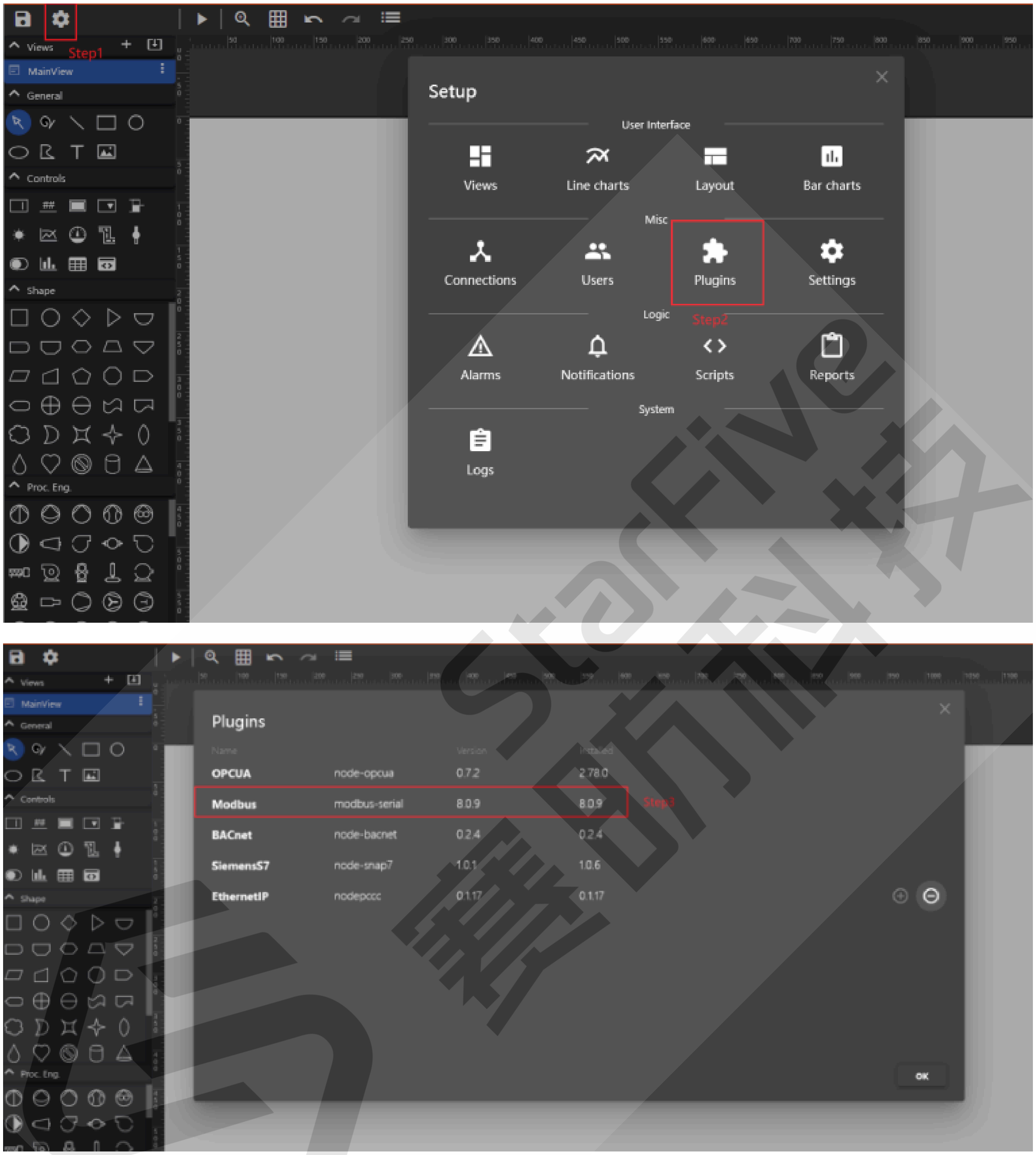
1. Click Editor in the lower left corner of the home screen to enter the editor:

   Figure 7-2 Click Editor

2. Select **Edit Project > Plugins** to see if the Modbus component is installed, or add it if it is not.

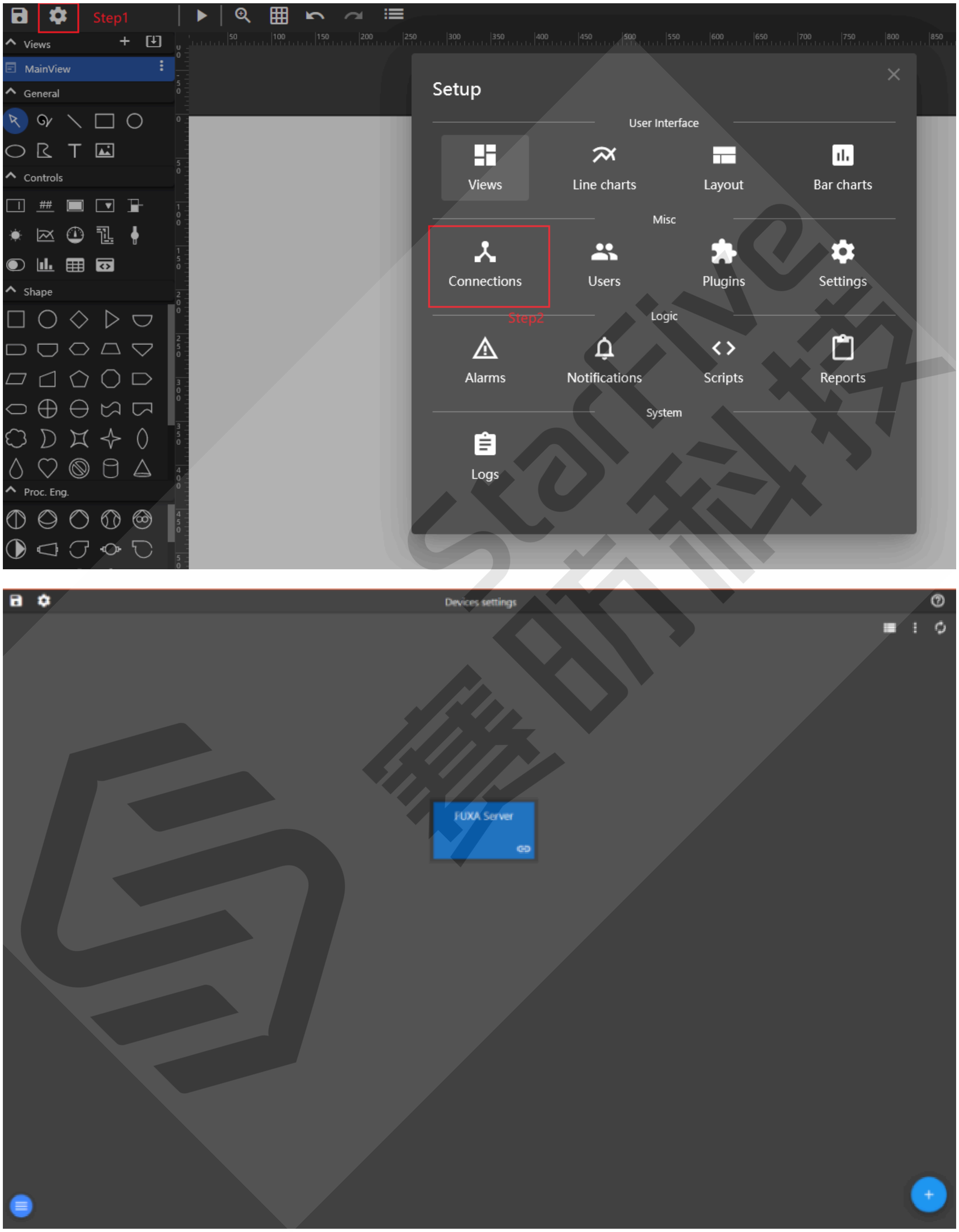
Figure 7-3 Verify Modus Installation



7.1.3.3. Connect the Modbus Devices

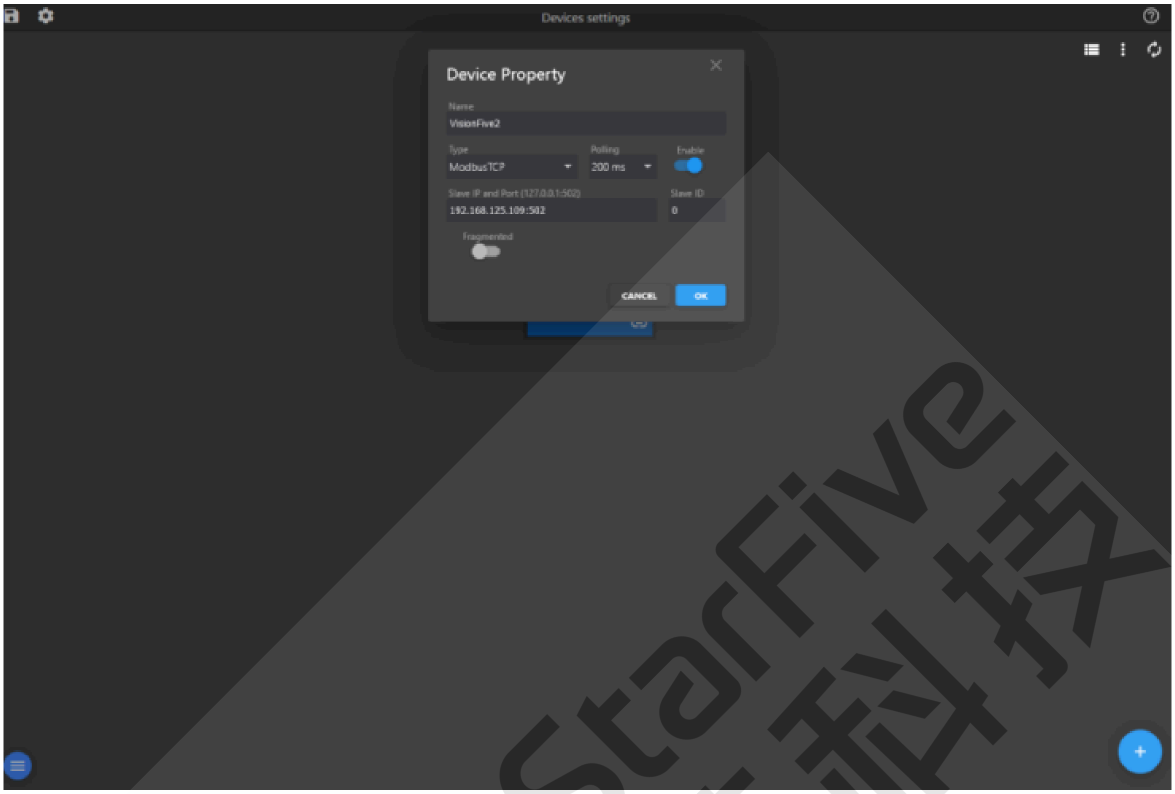
1. Click **Edit Project > Edit Project > Connections** to enter the connection management:

Figure 7-5 Connections



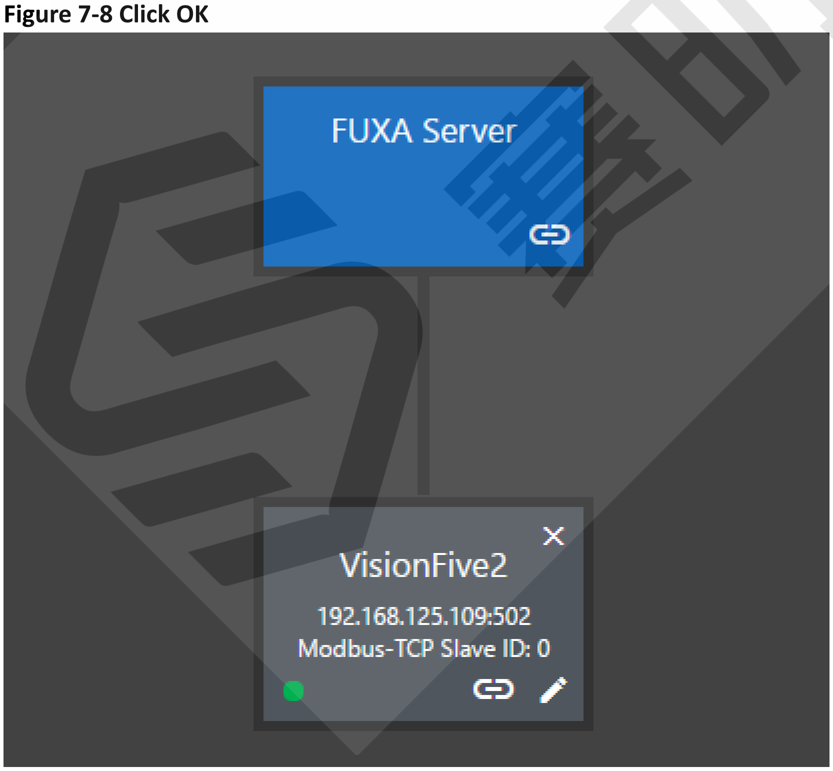
2. Click **Add** in the lower right corner, set device name, device type (ModbusTCP is selected here), refresh time, sub-device IP and port (502 by default), sub-device ID(do not repeat), and select Enable to enable connection:

Figure 7-7 Device Property



3. Click **OK** to save. If the connection is successful, a green mark will appear in the lower-left corner of the device frame:

Figure 7-8 Click OK

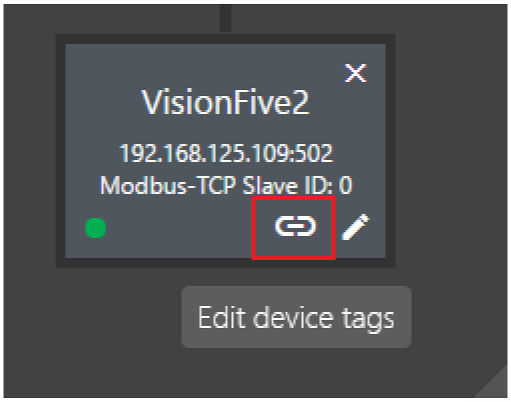


7.1.3.4. Connect I/O Points in OpenPLC

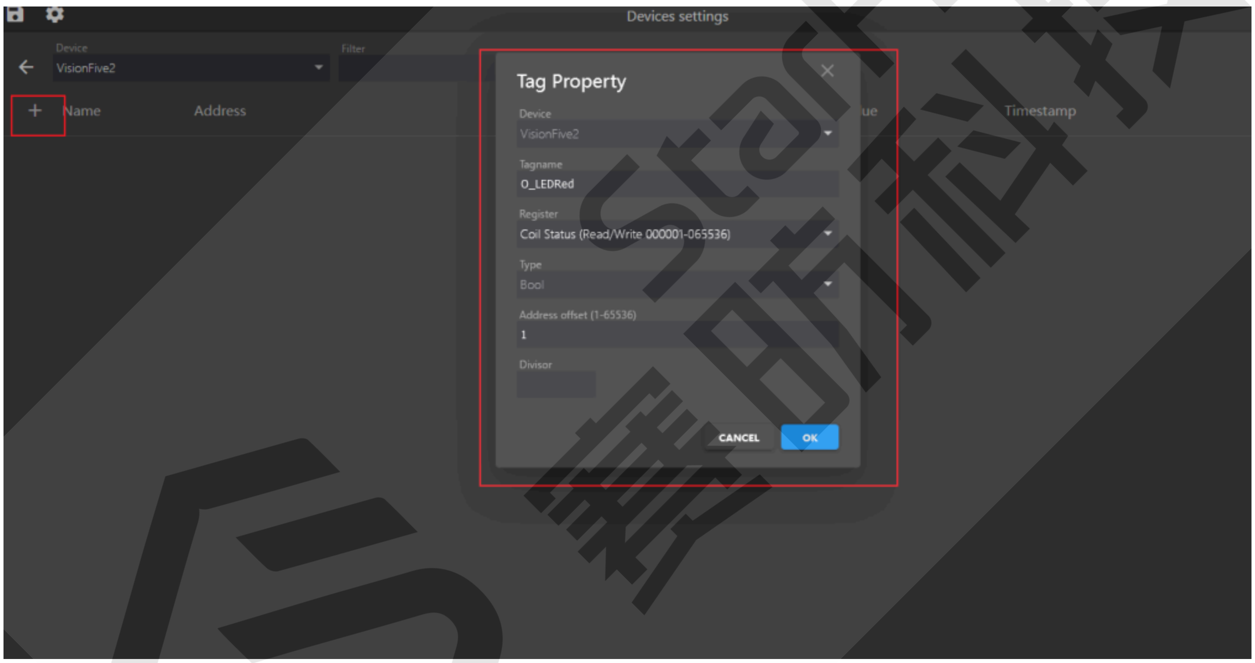
Perform the following steps to connect I/O points in OpenPLC:

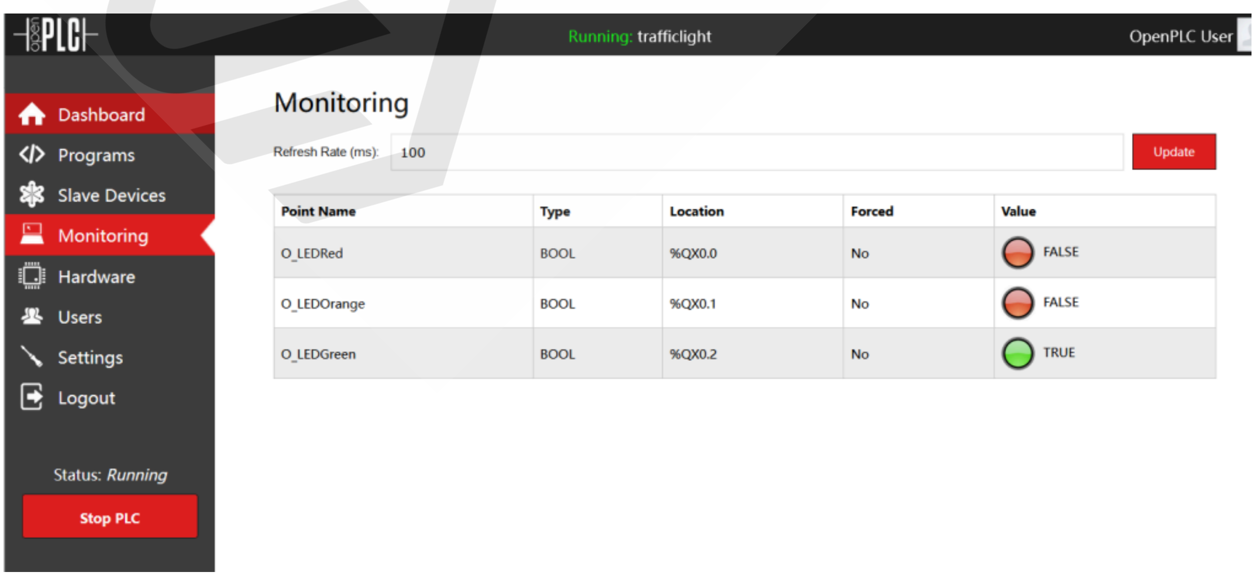
1. Click **Edit device tags** in the Device box:

Figure 7-9 Click Edit Device Tags

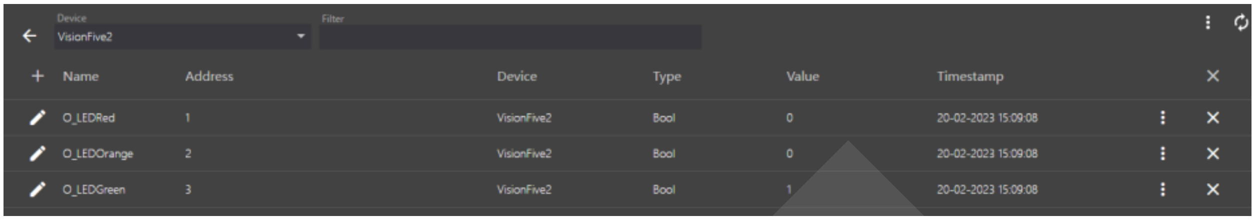


2. Select **add tags** and fill in Tagname (optional, but preferably corresponding to OpenPLC interface -> Monitoring -> Point Name), Register (point data type), Address offset (starting from 1 in FUXA, i.e. 0.0 in PLC):





Add other I/O points in the same way to observe the values of each point in real-time:



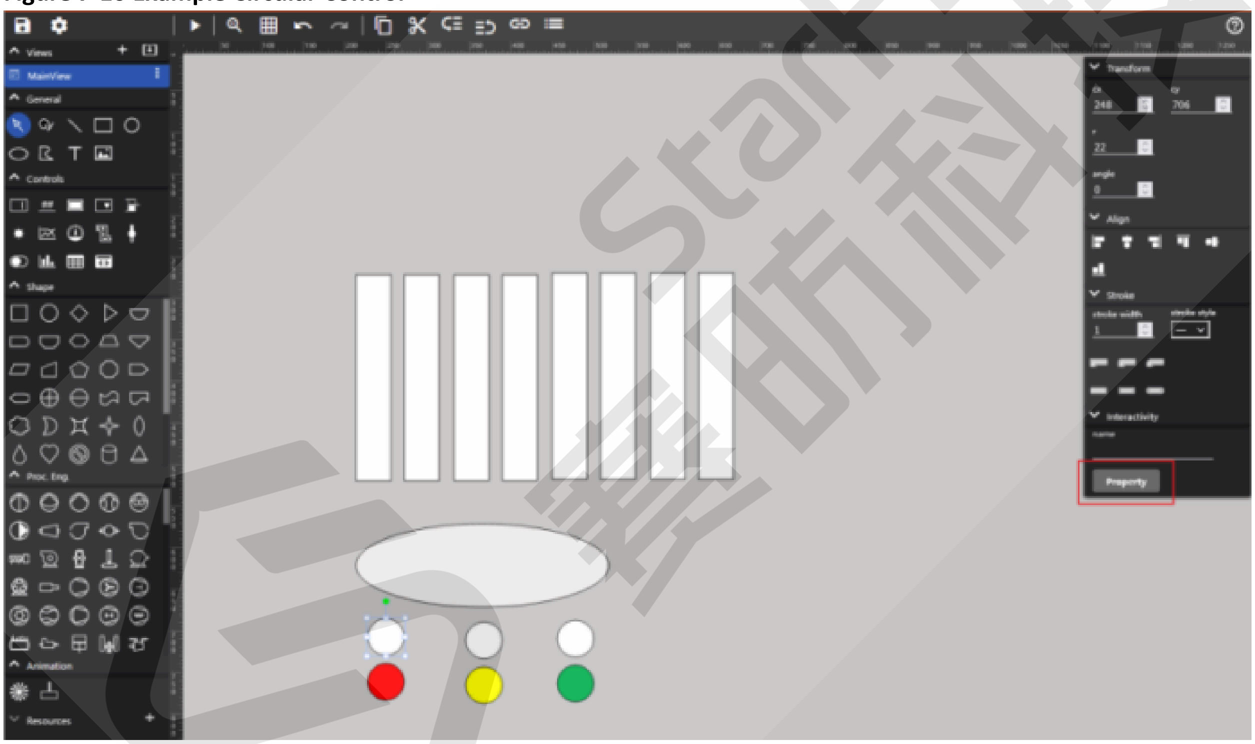
Name	Address	Device	Type	Value	Timestamp
O_LEDRed	1	VisionFive2	Bool	0	20-02-2023 15:09:08
O_LEDOrange	2	VisionFive2	Bool	0	20-02-2023 15:09:08
O_LEDGreen	3	VisionFive2	Bool	1	20-02-2023 15:09:08

7.1.3.5. Build a Real-Time GUI Scada/HMI Dashboard

Click the option in the lower left corner to enter the Editor interface. The PLC program used in this document is traffic light control, so the HMI of the traffic light scene is drawn. The drawing process is omitted here, and the main interface is as follows, describing how to connect the input and output points.

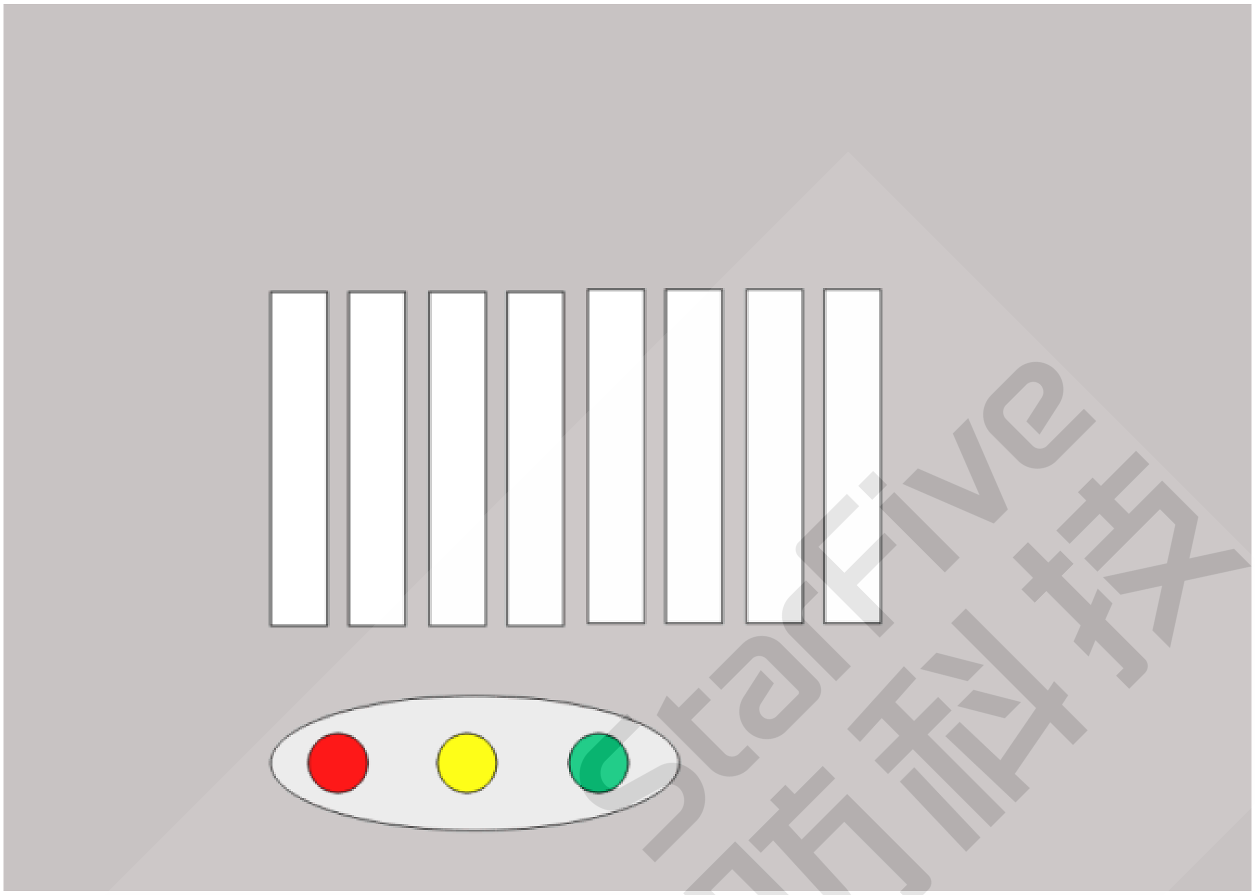
First, edit the circular space Property that serves as the traffic light, taking the white circular control that indicates the red light is off as an example:

Figure 7-10 Example Circular Control



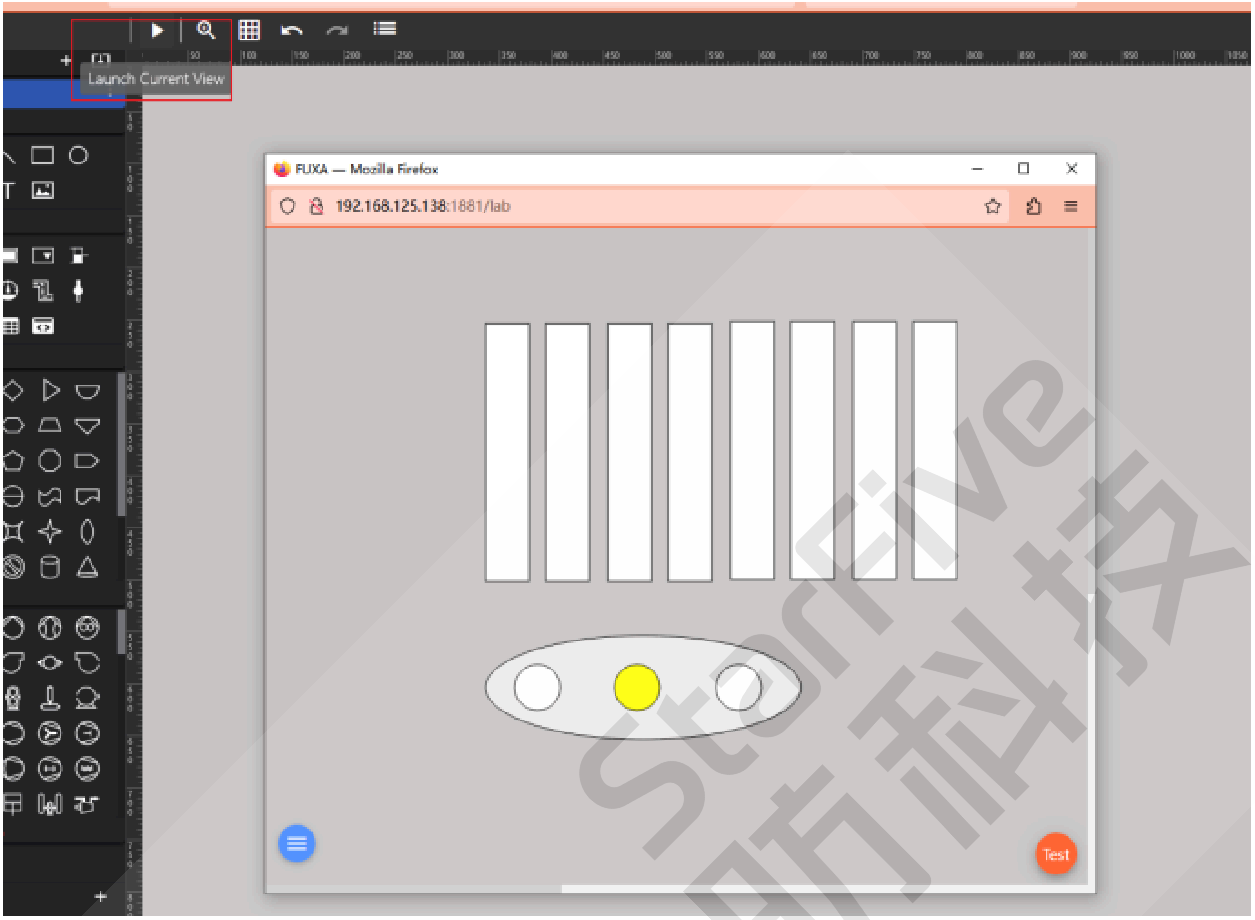
Select **Actions**, this PLC program traffic light has two states: on and off, so need to add another action. Bind both Actions with O_LEDRed point and set Bitmask to 0, which only has two values of 0/1. Therefore, set the Min and Max of the two actions to 0/0 and 1/1, and select Type Hide and Show to indicate that the white component will be displayed when O_LEDRed is 0. When O_LEDRed is 1, this component is hidden, and then the red circle component is set to the opposite logic, and the logic of yellow and green is set. After editing, the following figure is shown:

Figure 7-11 Example



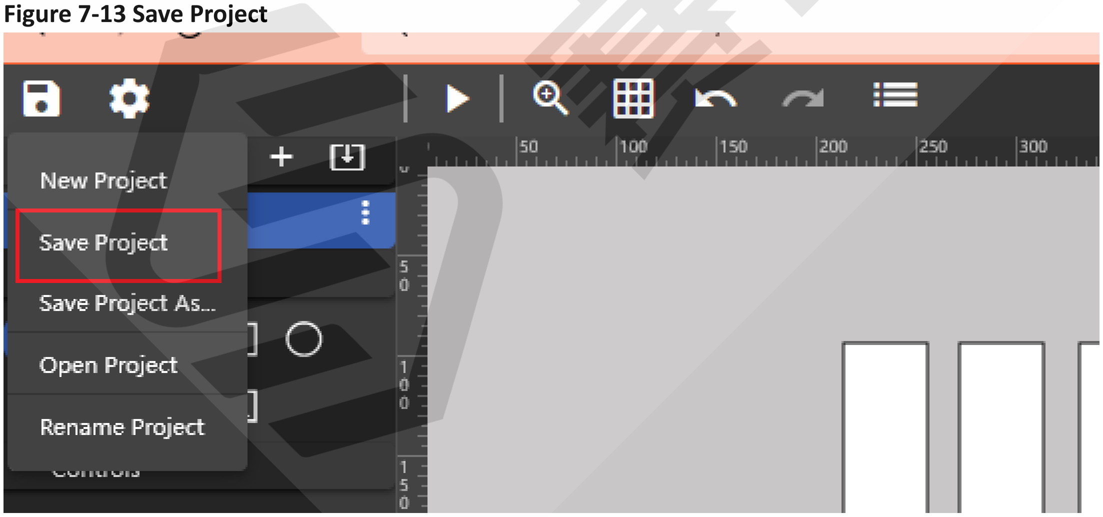
You can also preview this interface from the **Launch Current View** in the upper left corner:

Figure 7-12 Launch Current View



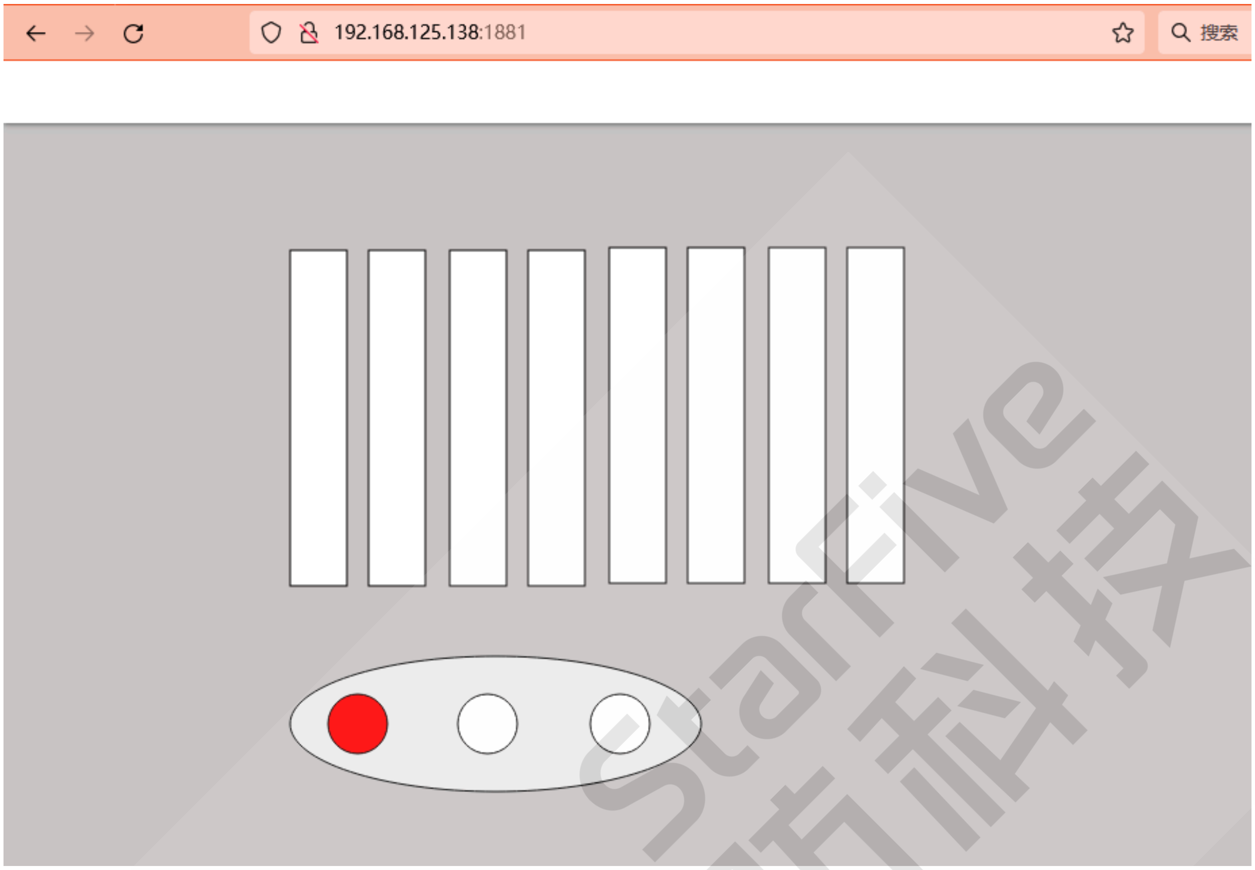
If satisfied, click the icon on the upper left corner to save the project:

Figure 7-13 Save Project



Then login to the `192.168.125.138:1881/` to display the SCADA interface and display the situation of each point:

Figure 7-14 SCADA



7.2. ScadaBR

ScadaBR and Scada-LTS have more requirements (Java/TomCat/MySQL/Gradle/NodeJS version) for the installation environment, and are difficult to install on VisionFive 2 or VisionFive 2 Lite.

Ref: https://github.com/ScadaBR/ScadaBR

Ref: https://github.com/SCADA-LTS/Scada-LTS

The following sections introduce the use of ScadaBR (LTS needs Gradle and MySQL) installed in Windows as an example.

7.2.1. Prerequisite

Before installing ScadaBR, install OpenJDK and TomCat9 by referring to the following links:

- Install OpenJDK
- Install TomCat9

7.2.2. Install ScadaBR

1. Download the ScadaBR `.war` file and put it into the TomCat `webapps/` directory. Double-click on the TomCat `bin/starup.bat`, then visit `localhost:8080/ScadaBR` to enter the ScadaBR interface.

   Note:

   If garbled words occur after running '`startup.bat`', modify the file as: ``conf/logging.properties`` ``java.util.logging.ConsoleHandler.encoding = GBK``.

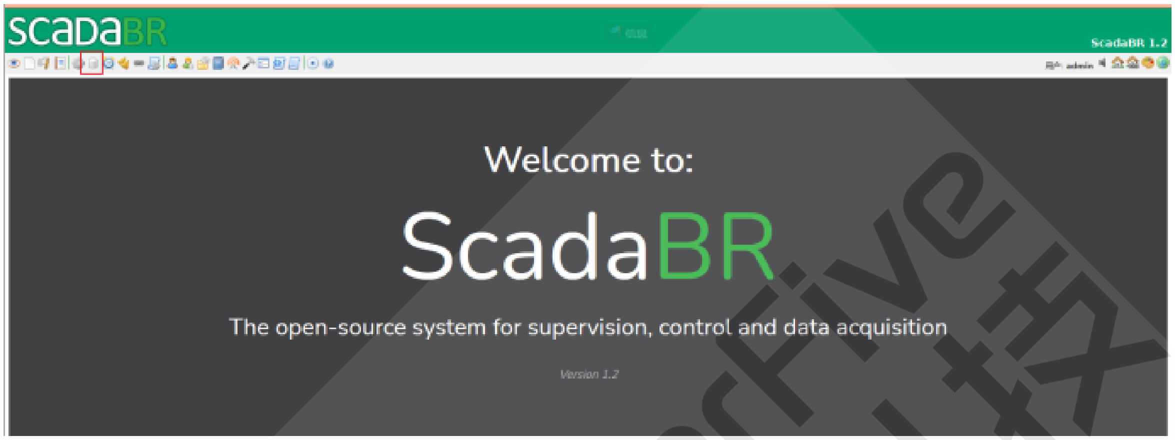
2. Enter the default user name and password (`admin` and `admin`) as prompted.

7.2.3. ScadaBR Operation

7.2.3.1. Connect PLC and I/O point

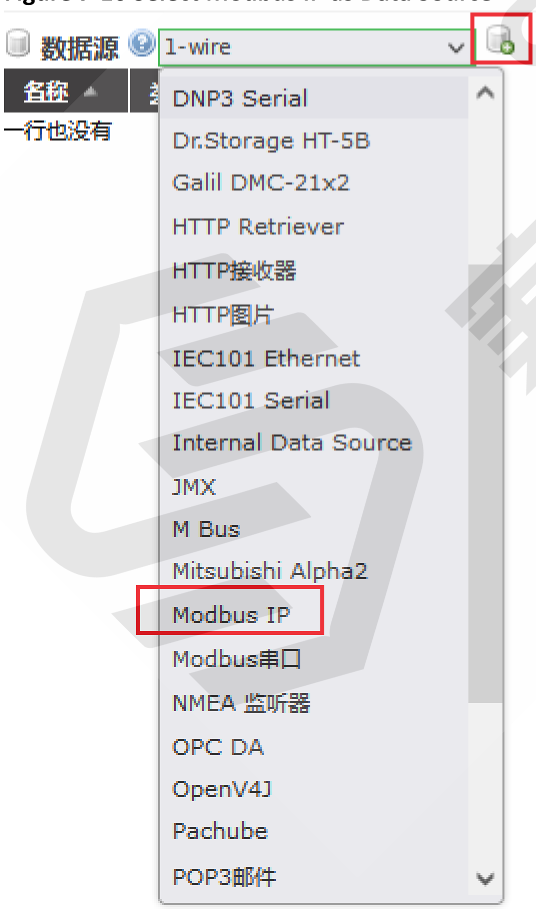
1. Click the following icon for **Data source** in the upper left corner of the main interface:

Figure 7-15 Data source



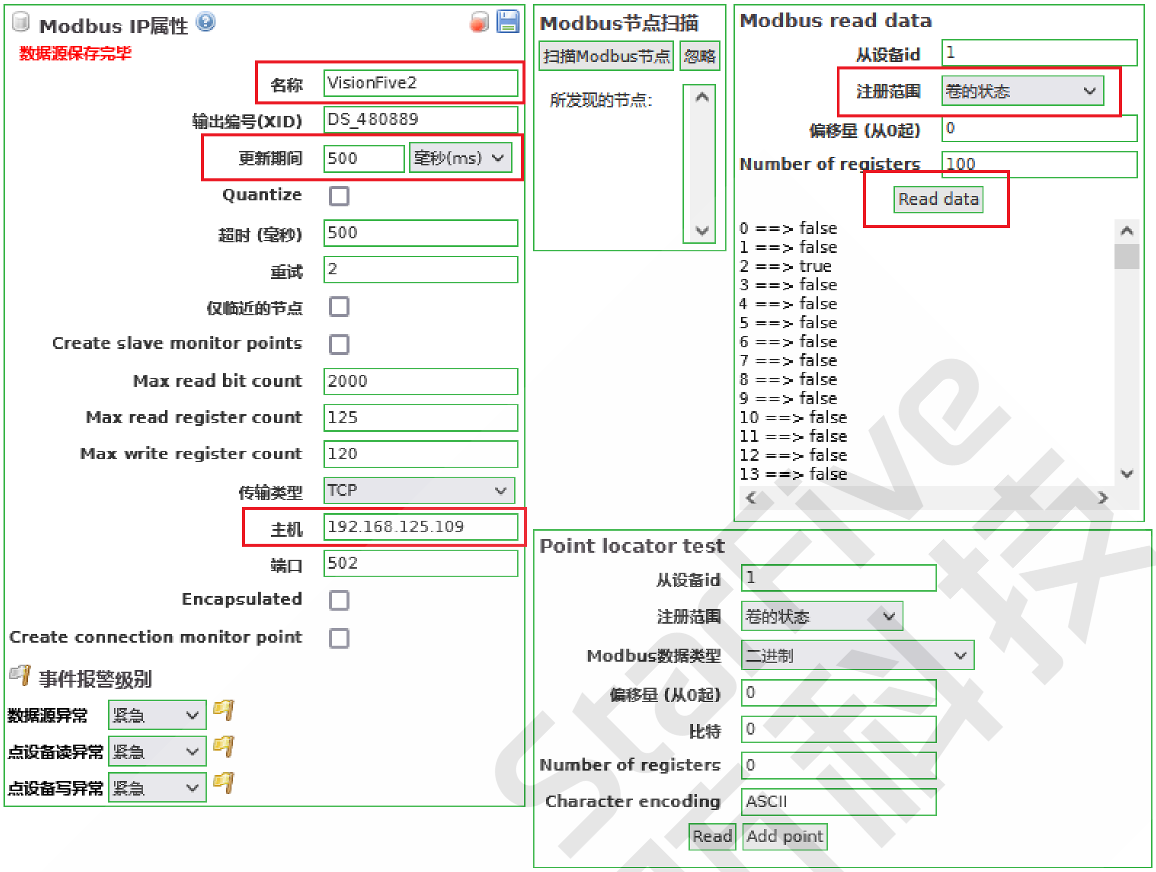
2. Select **Modbus IP** as the data source type and click **Add**:

Figure 7-16 Select Modbus IP as Data Source



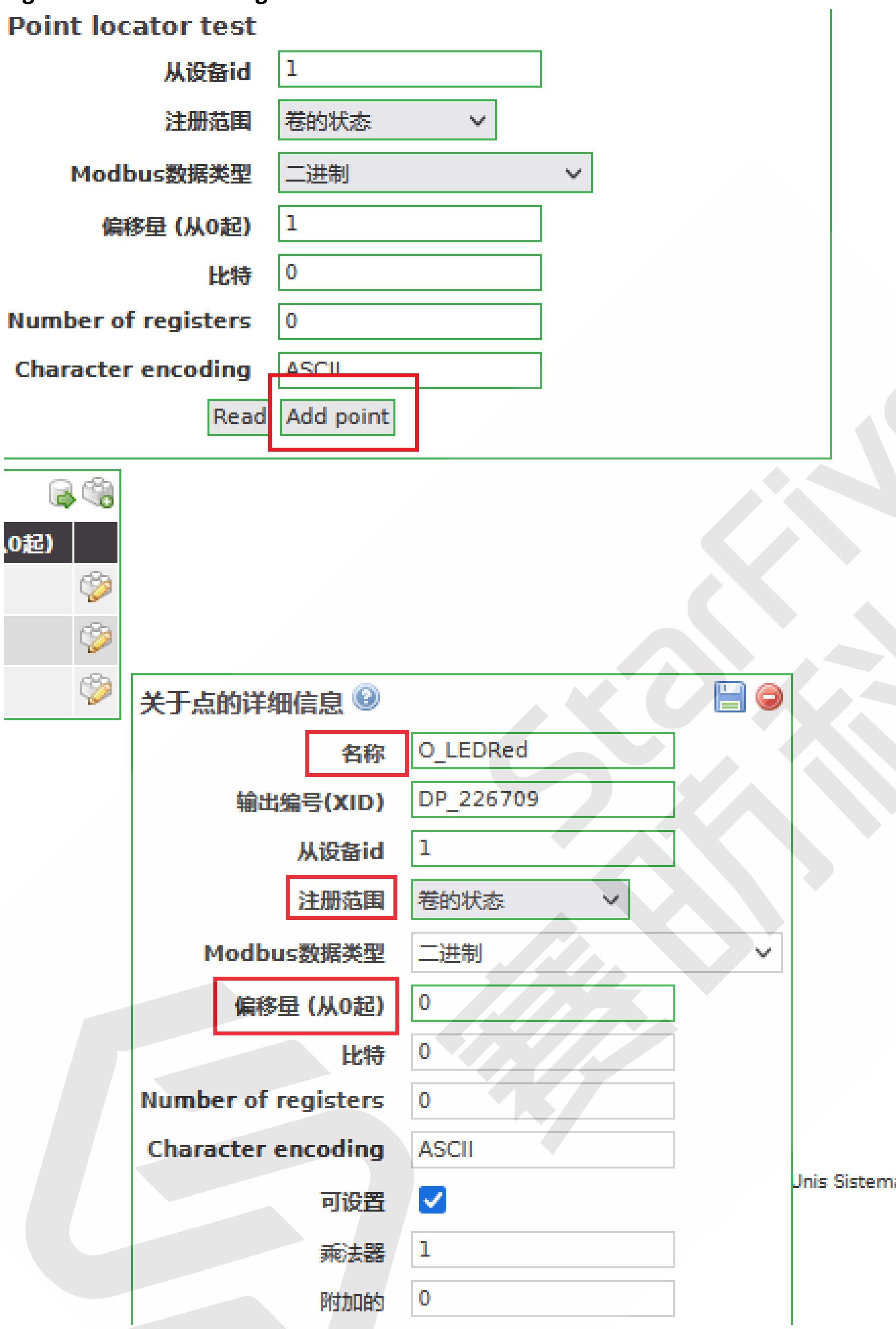
3. In Modbus IP, set the name, select the appropriate update time, fill in the host IP, and save the data source. Then select the value type to be read in the registration range of Modbus read data on the right and click **Read Data** to read the current value of the corresponding type quantity of the corresponding host through Modbus:

Figure 7-17 Modebus IP Settings



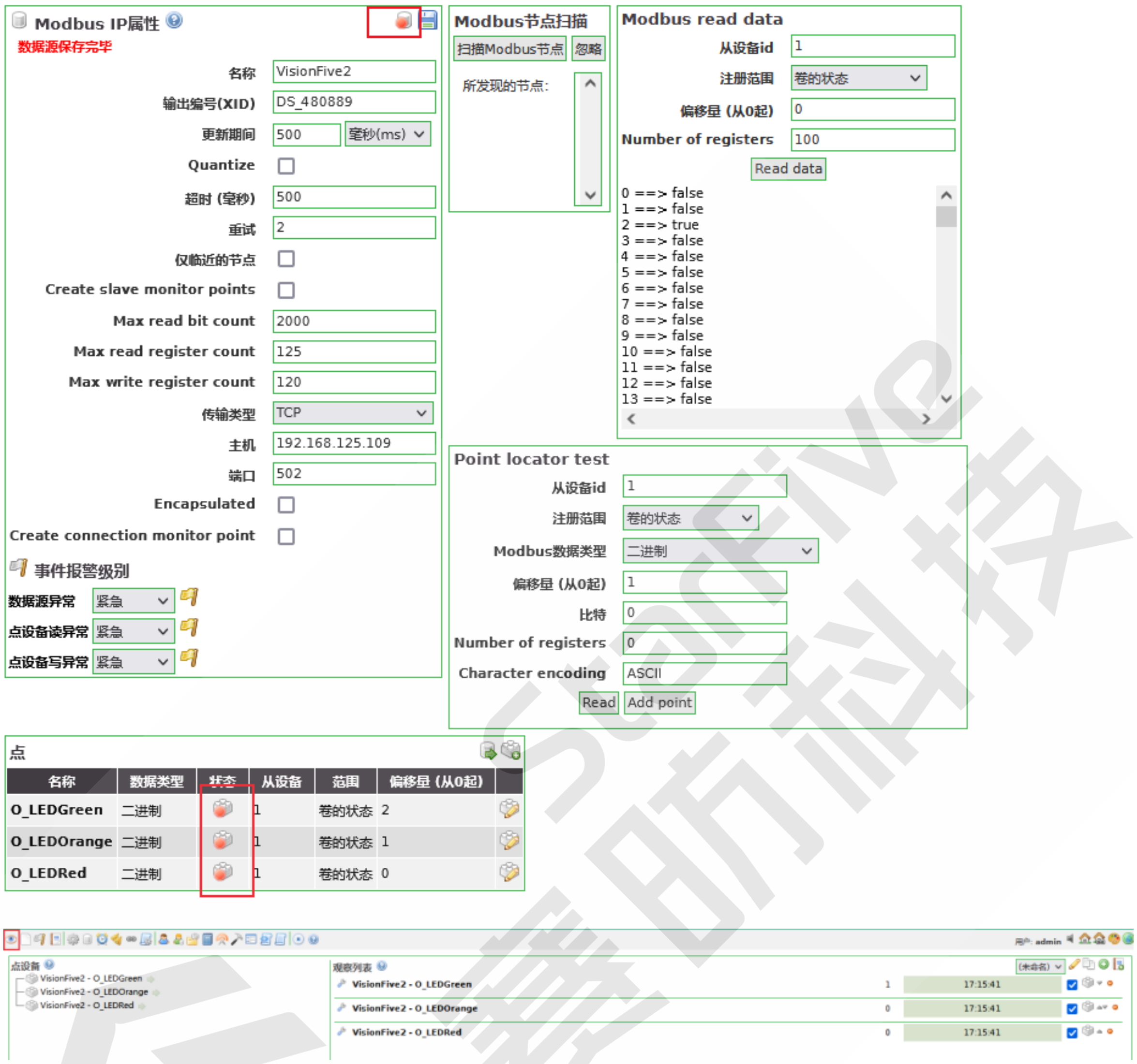
4. Then click '**add point**' on the Point locator test, enter the **Name** of the data Point (preferably the same as **PLC > monitoring > Point Name**), the offset (starting at 0, corresponding to 0.0 in PLC), and save:

Figure 7-18 Point Settings



5. Then activate each point and the device, that is, click the observation list of the first icon in the upper left corner to view the value of each data point of the device in real-time:

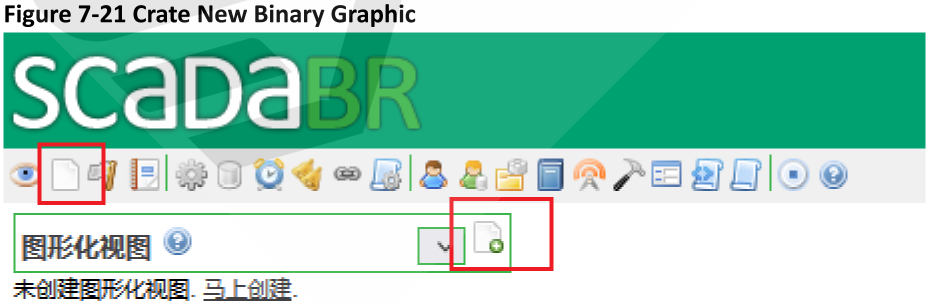
Figure 7-19 Data Point Values



7.2.3.2. Studio Interface

1. Select the second icon on the upper left corner and click 'New':

Figure 7-21 Crate New Binary Graphic

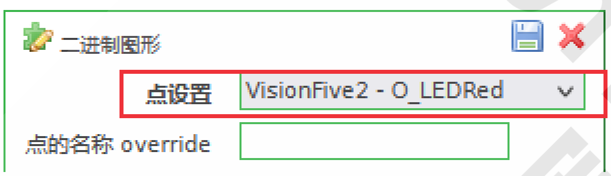


2. Enter a name, select **binary graphic**, and click **New**:

Figure 7-22 Binary Graphic Settings

3. The following screen will display an undefined binary graphic component. Select **Edit Point Build Settings** and link the corresponding data point:

Figure 7-23 Point Settings



4. Then select , select the appropriate graphics component, and select the graphics with a value of 0 or 1:

Figure 7-25 Edit Graphics renderer

5. Add other components in the same steps and save them. Then enter the graphical interface to display real-time status:

Figure 7-27 Display Real-Time Status

8. Test and improve method of Real-time performance for OpenPLC

Real-time performance refers to the ability of the operating system to respond to external events in a timely and deterministic manner. In other words, the system can guarantee that a certain task will be completed within a specific time frame. Real-time performance is important in many applications, such as industrial automation, robotics, and telecommunications. In these applications, the system must respond to external events within a specific time frame to ensure that the system operates correctly.

8.1. Test Method for Real-Time Performance

For OpenPLC, the real-time performance can be measured in the system and its own program response. Among them, the real-time performance of the system can be tested by Cyclictest, and OpenPLC itself can be tested by input/output delay and minimum scan period.

8.1.1. Cyclictest

RT-tests is a collection of programs that test the real-time capabilities of Linux. Cyclictest is one of the programs in the RT-tests suite that most commonly used for benchmarking RT systems. It is one of the most frequently used tools for evaluating the relative performance of real-time systems.

For details, refer to this link.

```
git clone git://git.kernel.org/pub/scm/linux/kernel/git/clrkwllms/rt-tests.git
git checkout -b stable/v1.0
```

Or

```
wget https://mirrors.edge.kernel.org/pub/linux/utils/rt-tests/rt-tests-1.10.tar.gz
make all
sudo make install
```

8.1.1.1. Run cyclictest with load

The following is an example to run cyclictest with load:

```
run ./cyclictest -S -p 95 -d 0 -i 1000 -D 24h -m -n
```

Run the following command to see more details:

```
./cyclictest --help
```

8.1.1.2. Simulated Load

When running cyclictest, there needs to be enough load on the test system for the statistical delay to be meaningful. The load can be simulated using the Hackbench tool provided in the rt-tests source package, the official load simulation script, and the stress-ng tool.

Execute the following command for details:

```
Hackbench: run ./hackbench --help under the path rt-test/
```

Refer to following links for more information:

- Simulation script: Link
- Stress-ng tool: Link

Cyclictest mainly tests the real-time performance at the system level. For OpenPLC runtime, some more detailed testing methods are needed.

8.1.2. Test Method of Real-Time Performance for OpenPLC Runtime

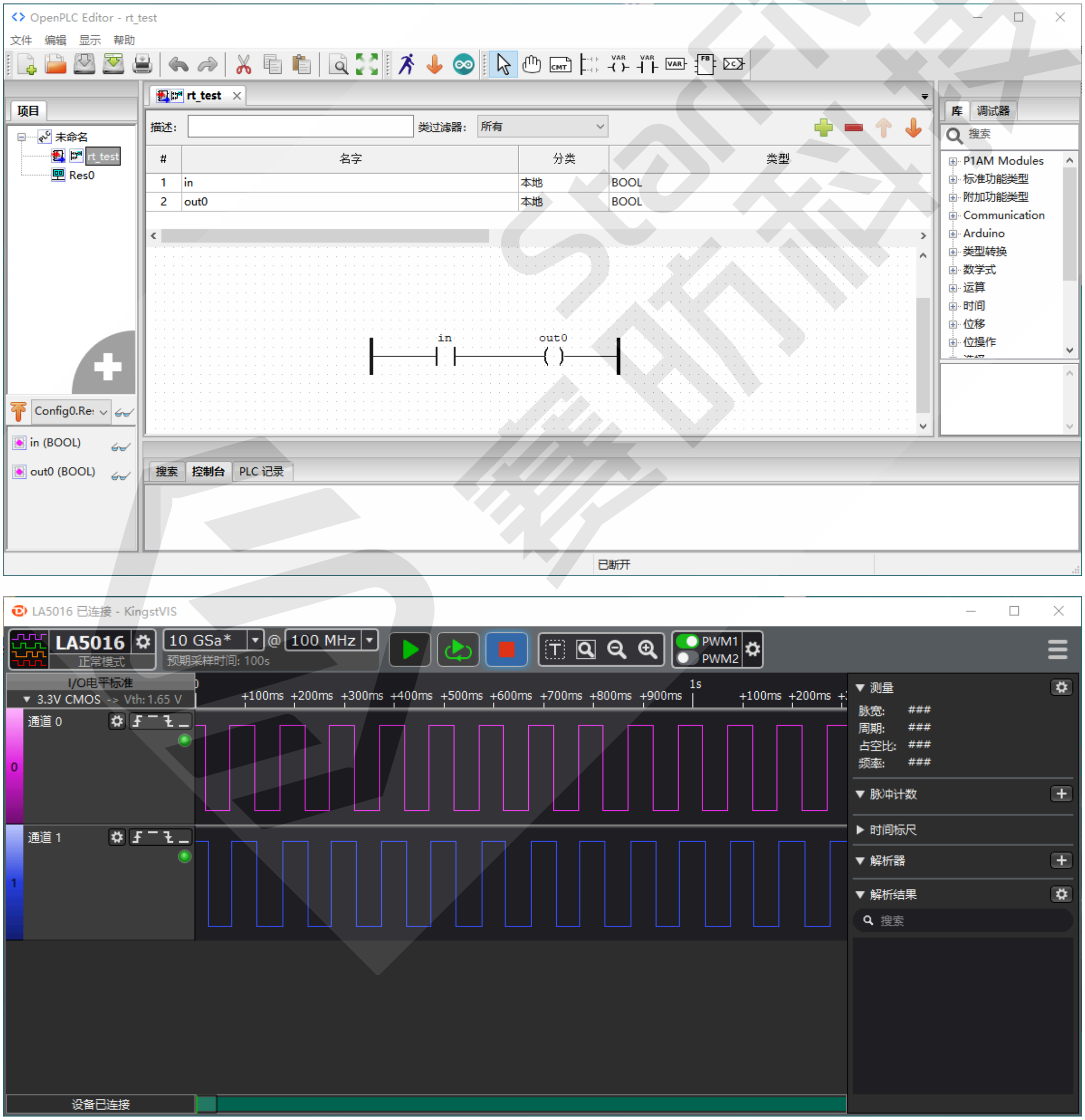
A PLC is an example of a real-time system since output results must be produced in response to input conditions within a limited time, otherwise, unintended operation will occur.

Cyclictest is used to test the real-time performance at the system level, and here the real-time performance of OpenPLC running needs to be analyzed. Therefore, the real-time performance of OpenPLC during operation is measured by the **delay between input and output signals** and the **minimum scanning time**.

8.1.2.1. Delay between Input and Output Signals

Run the ladder diagram program as shown below on OpenPLC, and input PWM signals with frequency of 10Hz and duty ratio of 50% to the input port %IX0.2 through the logic analyzer. Meanwhile, detect the input signals %IX0.2 and output signals %QX0.0. Run it for a while, record the time delay between the rising/falling edges of the input and output signals and calculate the mean and standard errors:

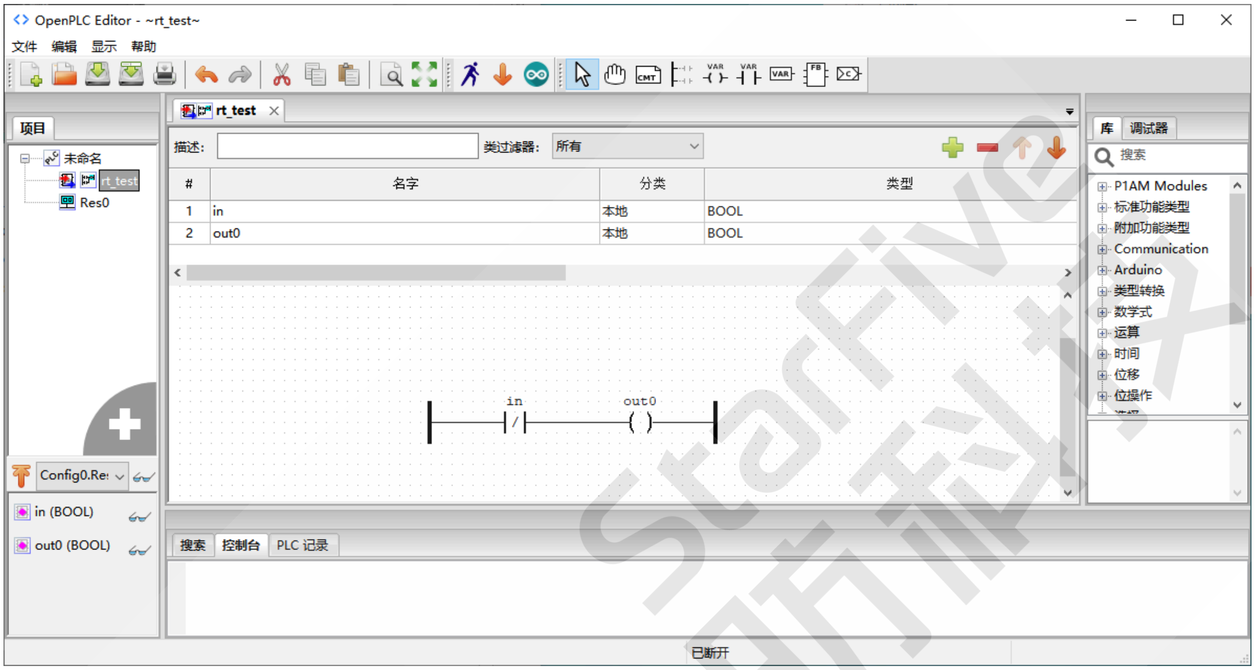
Figure 8-1 Delay between Input and Output Signals



8.1.2.2. Minimum Scanning Time

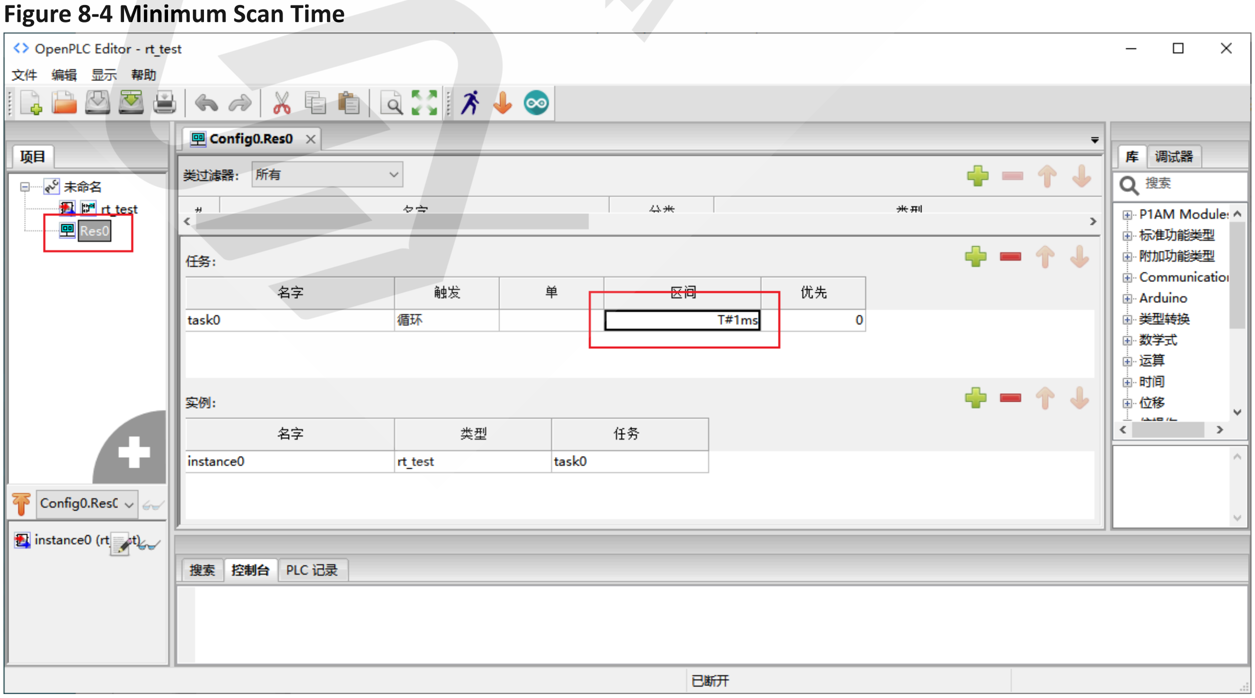
The real-time behavior of a PLC is usually associated with its scan time, which is the time the PLC takes to read all inputs, execute the logic program and write all outputs back. The execution of the logic program can normally be reduced to the evaluation of conditional statements and some arithmetic operations, most of the scan time is spent reading inputs and writing outputs. Therefore, the proposed scan time test is comprised of a single ladder logic line with one input and one output, as shown below:

Figure 8-3 Example Scan Time Test



By running this program and physically wiring the output %QX0.0 and the input %IX0.2 on the PLC together, it is possible to create an oscillator. This oscillator outputs a square wave with the highest frequency the PLC can generate in software. Since the output toggles at every scan, the scan time is defined by equation $t = \frac{1}{2f}$, where f is the square wave frequency measured by an oscilloscope. In addition to the difference in the ladder diagram, to measure the minimum scan time, the program of the minimum scan time in the scan interval of the Res0 file is set to **T#1ms**, while in the delay test, it is **T#5ms**:

Figure 8-4 Minimum Scan Time



This method is referred to: Thiago Alves, Thomas Morris. OpenPLC: An IEC 61131-3 Compliant Open Source Industrial Controller for Cyber Security Research[J]. Computer & Security, 2018, 78:364-379.

8.2. Improving Real-Time Performance of OpenPLC Runtime

To improve the real-time performance of OpenPLC runtime, the method of isolating a CPU core and bind the OpenPLC process and running the OpenPLC runtime in RT-linux is tried.

8.2.1. Isolate a CPU core and bind the OpenPLC process

8.2.1.1. Isolate a CPU Core

1. Run `cat /proc/cpuinfo` to check the information of CPU:

   Figure 8-5 Example Output

```
root@starfive:~# cat /proc/cpuinfo
processor       : 0
hart            : 1
isa             : rv64imafdc
mmu             : sv39
uarch           : sifive,u74-mc

processor       : 1
hart            : 2
isa             : rv64imafdc
mmu             : sv39
uarch           : sifive,u74-mc

processor       : 2
hart            : 3
isa             : rv64imafdc
mmu             : sv39
uarch           : sifive,u74-mc

processor       : 3
hart            : 4
isa             : rv64imafdc
mmu             : sv39
uarch           : sifive,u74-mc
```

2. Add parameters '**isolcpus**' in the startup command line file '`extlinux.conf`', and isolate specific CPUs from the general SMP balancing and scheduler algorithms.

   Tip:
   Click this link for more details.

Figure 8-6 Example Output

```
root@starfive:~# cat /boot/extlinux/extlinux.conf
## /extlinux/extlinux.conf
##
## IMPORTANT WARNING
##
## The configuration of this file is generated automatically.
## Do not edit this file manually, use: u-boot-update

default l0
menu title U-Boot menu
prompt 0
timeout 50


label l0
        menu label Debian GNU/Linux bookworm/sid 5.15.0-starfive
        linux /vmlinuz-5.15.0-starfive
        initrd /initrd.img-5.15.0-starfive


        fdtdir /dtbs
        append  root=/dev/mmcblk1p4 rw console=tty0 console=ttyS0,115200 earlycon rootwait stmmaceth=chain_mode:1 selinux=0 isolcpus=3
```

3. Modify the parameters as shown in the figure above. After restarting, the core/processor 3 will not participate in system scheduling.

8.2.1.2. Run OpenPLC Runtime on the Isolated Core

After CPU core3 is isolated, the OpenPLC process needs to be bound to the core. This can be done in two ways: one is by using the command task set, and the other is to modify main.cpp of OpenPLC to set kernel affinity.

8.2.1.2.1. Taskset

The taskset command is used to query or specify the CPU core on which the process is running by the corresponding PID.

1. After OpenPLC runtime is started, run '`systemctl status openplc`' to view the processes related to the OpenPLC runtime:

Figure 8-7 Example Output

```
root@starfive:~# systemctl status openplc
● openplc.service - OpenPLC Service
     Loaded: loaded (/lib/systemd/system/openplc.service; enabled; preset: enabled)
     Active: active (running) since Thu 2023-03-16 09:42:34 UTC; 23h ago
   Main PID: 437 (start_openplc.s)
        CPU: 33.031s
     CGroup: /system.slice/openplc.service
             ├─ 437 /bin/bash /root/OpenPLC_v3/start_openplc.sh
             ├─ 440 python2.7 webserver.py
             └─3140 ./core/openplc
```

2. Then, run '`sudo taskset -pc x PID`' to bind the 'PID' process to the core 'x', for example: run 'sudo taskset -pc 3 3140' to bind the main process './core/openplc' of OpenPLC runtime to core 3 and check it by running: `sudo taskset -p 3140`:

Figure 8-8 Example Output

```
root@starfive:~# taskset -pc 3 3140
pid 3140's current affinity list: 0-3
pid 3140's new affinity list: 3
root@starfive:~# taskset -p 3140
pid 3140's current affinity mask: 8
```

Tip:
Click this link to see more details.

8.2.1.2.2. Kernel Affinity

In additions, the OpenPLC runtime process can be assigned to the specified CPU core by modify the main.cpp file. It is implemented through the '**cpuset**', '**sched_set/getaffinity**' and '**pthread_setaffinity_np**' functions.

Refer to following links for more details:

- **cpuset**: cpuset.7.html
- **sched_set/getaffinity**: sched_setaffinity.2.html
- **pthread_settaffinity_np**: pthread_setaffinity_np.3.html

For example, modify `main.cpp` to run the main OpenPLC task on the isolated core. Something like this on the REAL-TIME INITIALIZATION section in main.cpp should do the job:

```
//Set process affinity to core 3
cpu_set_t mask;
CPU_ZERO(&mask);
CPU_SET(3, &mask);
if (sched_setaffinity(0, sizeof(mask), &mask) < 0)
{
printf("WARNING: Error setting affinity of main thread\n");
}
if (pthread_setaffinity_np(pthread_self(), sizeof(mask), &mask) < 0)
{
printf("WARNING: Error setting thread affinity of main thread\n");
}
```

Figure 8-9 Example Setting

```
#ifdef __linux__
    //===============================================
    //              REAL-TIME INITIALIZATION
    //===============================================
    // Set our thread to real time priority
    struct sched_param sp;
    sp.sched_priority = 30;
    printf("Setting main thread priority to RT\n");
    if(pthread_setschedparam(pthread_self(), SCHED_FIFO, &sp))
    {
        printf("WARNING: Failed to set main thread to real-time priority\n");
    }

    // Lock memory to ensure no swapping is done.
    printf("Locking main thread memory\n");
    if(mlockall(MCL_FUTURE|MCL_CURRENT))
    {
        printf("WARNING: Failed to lock memory\n");
    }
#endif

        //gets the starting point for the clock
        printf("Getting current time\n");
        struct timespec timer_start;
        clock_gettime(CLOCK_MONOTONIC, &timer_start);

        //Set process affinity to core 3
        cpu_set_t mask;
        CPU_ZERO(&mask);
        CPU_SET(3, &mask);
        if (sched_setaffinity(0, sizeof(mask), &mask) < 0)
        {
        printf("WARNING: Error setting affinity of main thread\n");
        }
        if (pthread_setaffinity_np(pthread_self(), sizeof(mask), &mask) < 0)
        {
        printf("WARNING: Error setting thread affinity of main thread\n");
        }
```

```
root@starfive:~# systemctl status openplc
● openplc.service - OpenPLC Service
     Loaded: loaded (/lib/systemd/system/openplc.service; enabled; preset: enabled)
     Active: active (running) since Thu 2023-03-16 09:42:34 UTC; 23h ago
   Main PID: 437 (start_openplc.s)
        CPU: 33.031s
     CGroup: /system.slice/openplc.service
             ├─ 437 /bin/bash /root/OpenPLC_v3/start_openplc.sh
             ├─ 440 python2.7 webserver.py
             └─3140 ./core/openplc
```

After modifying `main.cpp` as shown above, re-compile the modified `main.cpp` file by re-selecting the PLC program or re-saving the hardware layer file and setting kernel isolation, the OpenPLC main process `./core/openplc` will automatically runs on the isolated core after restarting.

It's recommend to run CPU in highest frequency by run: '`echo performance > /sys/devices/system/cpu/cpu0/cpufreq/scaling_governor`' to achieve optimal system performance.

8.2.2. Run OpenPLC on RT-Linux:

There has long been a Linux real-time side-project which develops and maintains realtime extensions to the Linux kernel. Hard real-time kernel options available as a patchset from the Linux RT project.

For details: https://wiki.linuxfoundation.org/realtime/start

To run OpenPLC on RT-Linux, perform the following:

1. Follow the instructions in this link to run OpenPLC on RT-Linux.
2. After replace the RT-kernel into Debian image by reference to: https://doc.rvspace.org/VisionFive2/SW_TRM/VisionFive2_SW_TRM/adding_new_file%20-%20VF2.html.
3. Install the OpenPLC runtime. It's recommend to run CPU in highest frequency by run: '`echo performance > /sys/devices/system/cpu/cpu0/cpufreq/scaling_governor`' to achieve optimal system performance.

9. Examples

9.1. Temperature Control

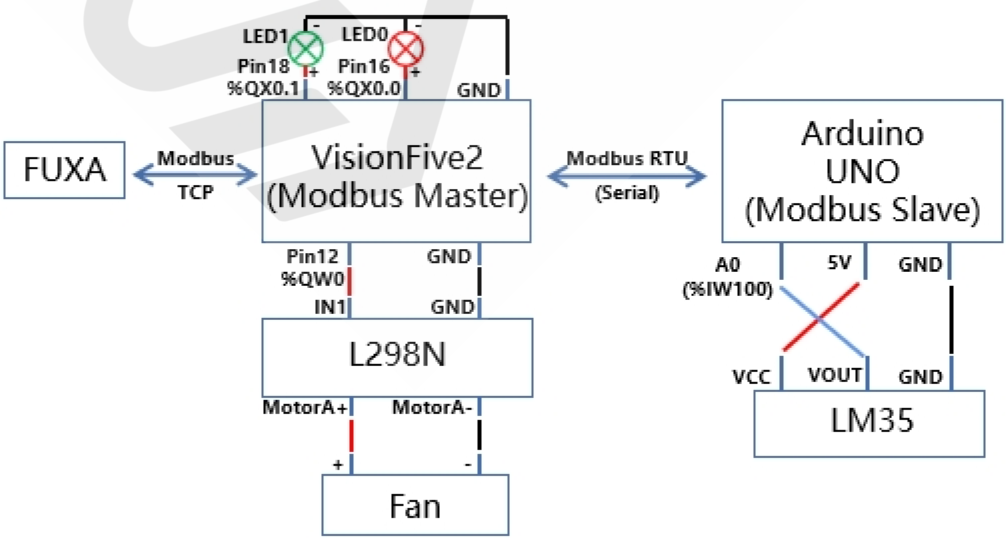
A demo of temperature control through OpenPLC and the information of each component can be viewed comprehensively through FUXA is given in this section.

Table 9-1 Hardware Preparation

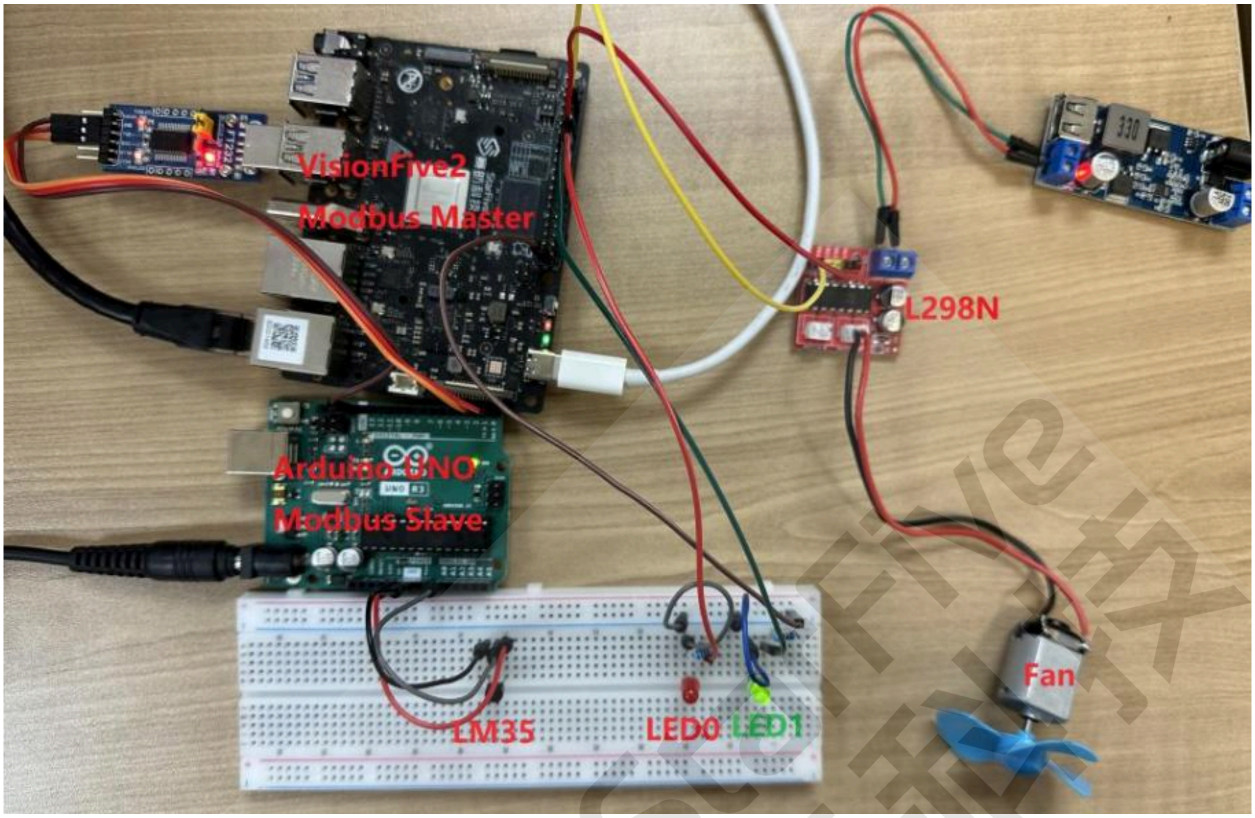
Items	Number	Note
VisionFive 2 or VisionFive 2 Lite	1	Modbus-RTU master device.
Arduino UNO	1	Modbus-RTU slave device.
LM35	1	Temperature sensor.
L298N	1	Motor drive board.
Voltage converter	1	A 5V input voltage is provided to the motor.
DC motor	1	Turn the fan.
Breadboard	1	
LED	2	Preferably two different colors.
Button	2	
Dupont line	Several	
USB2Serial	1	Corresponding to the enabled type in the kernel options.

The overall system structure diagram is shown in the figure below. Since VisionFive2 lacks analog input, Arduino UNO is used as a slave device controlled by VisionFive 2 or VisionFive 2 Lite through Modbus RTU (serial) to read the input of temperature sensor LM35 and send it back to VisionFive2. Based on the obtained temperature information, VisionFive 2 or VisionFive 2 Lite reflects the temperature is in the normal/high range through the LED, outputs PWM signals with different duty ratios according to different temperature ranges, and controls fan start-stop and speed through the DC motor drive module L298N. All information on the system is transferred to FUXA via Modbus TCP.

Figure 9-1 Transferred to FUXA via Modbus TCP



The physical connection is as follows:



PLC functional block program and variable definition as shown below:

Figure 9-2 PLC Functional Block Program

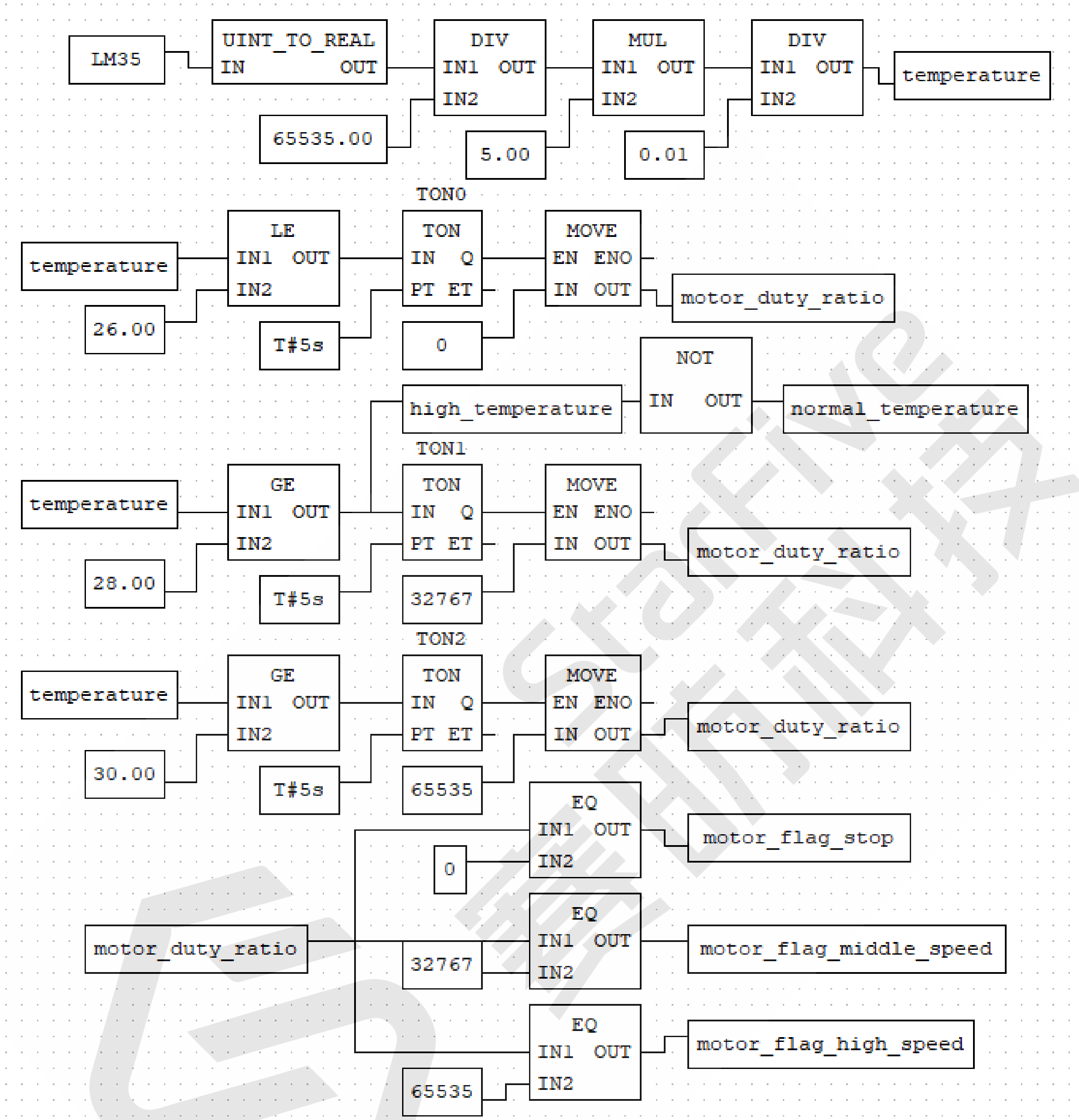


Figure 9-3 Variable Definition

#	名字	分类	类型	位置
1	LM35	本地	UINT	%IW100
2	temperature	本地	REAL	%MD0
3	high_temperature	本地	BOOL	%QX0.0
4	normal_temperature	本地	BOOL	%QX0.1
5	motor_flag_stop	本地	BOOL	%QX0.2
6	motor_flag_middle_speed	本地	BOOL	%QX0.3
7	motor_flag_high_speed	本地	BOOL	%QX0.4
8	motor_duty_ratio	本地	UINT	%QW0
9	TON0	本地	TON	
10	TON1	本地	TON	
11	TON2	本地	TON	

Tip:

Ladder diagram program is located The functional block diagram program is located in your computer after you download and unzipped the `openplc.zip`, for example: `C:\Users\chloe.chen\Downloads\openplc\9. Example temperature_fan.zip`.

In OpenPLC, the analog input resolution is 16 bits, that is, the analog input in OpenPLC ranges from 0 to 65535. In the function diagram program, the value of the analog input LM35 (%IW100) needs to be converted into the temperature value. For every 1 degree Celsius rise in temperature, the output voltage of LM35 rises by 10mV. Therefore, 'temperature = LM35 / 65535.00 * 5.00 / 0.01'.

In this function diagram program, if the temperature remains below 28 degrees Celsius, the fan will not work. When the temperature exceeds 28 degrees Celsius for 5s or more, VisionFive2's analog output pin(%QW0) outputs a 50% duty ratio PWM signal to drive the fan. When the temperature continues to rise above 30 degrees, the drive fan works at full load. The fan will not stop working until the temperature is below 26 degrees Celsius and lasts for more than 5 seconds.

After completing program writing and uploading, slave-device connection, and FUXA setting, OpenPLC can be started to check the status of the whole temperature control system in FUXA

Figure 9-4 Temperature Control System